Spatial information and end-to-end learning for visual recognition

Mingyuan Jiu

Numéro d'ordre: Année 2014

Institut National Des Sciences Appliquées de Lyon

Laboratoire d'InfoRmatique en Image et Systèmes d'information

École Doctorale Informatique et Mathématique de Lyon

Thèse de l'Université de Lyon

Présentée en vue d'obtenir le grade de Docteur,
spécialité Informatique

par

Mingyuan Jiu

Spatial information and end-to-end learning for visual recognition

Thèse soutenue le 3 Avril 2014 devant le jury composé de:

Mme.	Michèle Rombaut	Professeur, Université Joseph Fourier Grenoble	Rapporteur
M.	Matthieu Cord	Professeur, UPMC-Sorbonne Universities	Rapporteur
Mme.	Cordelia Schmid	INRIA Research Director	Examinateur
M.	Alain Trémeau	Professeur, Université Jean Monnet Saint Etienne	Examinateur
M.	Graham Taylor	Maître de Conférences, University of Guelph, Canada	Examinateur
M.	Atilla Baskurt	Professeur, INSA Lyon	Directeur
M.	Christian Wolf	Maître de Conférences, HDR, INSA Lyon	Co-encadrant

Laboratoire d'InfoRmatique en Image Systèmes d'information
UMR *5205* CNRS - INSA de Lyon - Bât. Jules Verne
69621 Villeurbanne cedex - France
Tel: *+33(0)4 72 43 60 97* - Fax: *+33(0)4 72 43 71 17*

Acknowledgments

In this moment, I would like to express my sincere appreciations to numerous people who have given me helps and supports throughout my study during the past three and half years.

First of all, my most sincere and deep gratitude is for my supervisors, Prof. Atilla Baskurt and Dr. Christian Wolf, without whose constant guidance and assistances this thesis would not be possible.

Secondly, I would like to sincerely acknowledge our collaborators Prof. Christopher Garcia and Dr. Graham Taylor of University of Guelph, Canada. I learned too much from discussion with them.

Besides, I would like to sincerely acknowledge Prof. Michèle Rombaut, Prof. Matthieu Cord, Prof. Cordelia Schmid and Prof. Alain Trémeau for their kindness to become my thesis reviewers and examiners and for their time on constructive and meticulous assessment of this thesis work.

I also want to thank all the members I have met/worked with in the Bâtment Jules Verne, INSA de Lyon, including Guillaume Lavoué, Eric Lombardi, Julien Mille, Stefan Duffner, Yuyao Zhang, Peng Wang, Oya Celiktutan, Louisa Kessi, Thomas Konidaris, Lilei Zheng, Jinjiang Guo, Vincent Vidal et al. for their kind help, constructive discussions and also the enjoyable time they have brought to me.

In addition, I would like to thank all my friends, especially Zhenzhong Guo, Huagui Zhang et al., for sharing their enjoyable time with me during my stay in Lyon! Certainly, all my family members as well as Wanying Chen deserve the most important appreciations for their endless loves, their understandings and supports over all the past years of my study till completing my Ph.D.

Finally, China Scholarship Council (CSC) for providing the scholarship is gratefully acknowledged.

Abstract

In this thesis, we present our research on visual recognition and machine learning. Two types of visual recognition problems are investigated: action recognition and human body part segmentation problem. Our objective is to combine spatial information such as label configuration in feature space, or spatial layout of labels into an end-to-end framework to improve recognition performance.

For human action recognition, we apply the bag-of-words model and reformulate it as a neural network for end-to-end learning. We propose two algorithms to make use of label configuration in feature space to optimize the codebook. One is based on classical error backpropagation. The codewords are adjusted by using gradient descent algorithm. The other is based on cluster reassignments, where the cluster labels are reassigned for all the feature vectors in a Voronoi diagram. As a result, the codebook is learned in a supervised way. We demonstrate the effectiveness of the proposed algorithms on the standard KTH human action dataset.

For human body part segmentation, we treat the segmentation problem as classification problem, where a classifier acts on each pixel. Two machine learning frameworks are adopted: randomized decision forests and convolutional neural networks. We integrate *a priori* information on the spatial part layout in terms of pairs of labels or pairs of pixels into both frameworks in the training procedure to make the classifier more discriminative, but pixel-wise classification is still performed in the testing stage. Three algorithms are proposed: (i) Spatial part layout is integrated into randomized decision forest training procedure; (ii) Spatial pre-training is proposed for the feature learning in the ConvNets; (iii) Spatial learning is proposed in the logistical regression (LR) or multilayer perceptron (MLP) for classification.

Keywords: Spatial learning, end-to-end learning, randomized decision forest, convolutional neural network, action recognition, human part segmentation.

Résumé

Dans cette thèse nous étudions les algorithmes d'apprentissage automatique pour la reconnaissance visuelle. Un accent particulier est mis sur l'apprentissage automatique de représentations, c.à.d. l'apprentissage automatique d'extracteurs de caractéristiques; nous insistons également sur l'apprentissage conjoint de ces dernières avec le modèle de prédiction des problèmes traités, tels que la reconnaissance d'objets, la reconnaissance d'activités humaines, ou la segmentation d'objets.

Dans ce contexte, nous proposons plusieurs contributions :

Une première contribution concerne les modèles de type bags of words (BoW), où le dictionnaire est classiquement appris de manière non supervisée et de manière autonome. Nous proposons d'apprendre le dictionnaire de manière supervisée, c.à.d. en intégrant les étiquettes de classes issues de la base d'apprentissage. Pour cela, l'extraction de caractéristiques et la prédiction de la classe sont formulées en un seul modèle global de type réseau de neurones (end-to-end training). Deux algorithmes d'apprentissage différents sont proposés pour ce modèle : le premier est basé sur la retro-propagation du gradient de l'erreur, et le second procède par des mises à jour dans le diagramme de Voronoi calculé dans l'espace des caractéristiques.

Une deuxième contribution concerne l'intégration d'informations géométriques dans l'apprentissage supervisé et non-supervisé. Elle se place dans le cadre d'applications nécessitant une segmentation d'un objet en un ensemble de régions avec des relations de voisinage définies a priori. Un exemple est la segmentation du corps humain en parties ou la segmentation d'objets spécifiques.

Nous proposons une nouvelle approche intégrant les relations spatiales dans l'algorithme d'apprentissage du modèle de prédication. Contrairement aux méthodes existantes, les relations spatiales sont uniquement utilisées lors de la phase d'apprentissage. Les algorithmes de classification restent inchangés, ce qui permet d'obtenir une amélioration du taux de classification sans augmentation de la complexité de calcul lors de la phase de test.

Nous proposons trois algorithmes différents intégrant ce principe dans trois modèles :

- l'apprentissage du modèle de prédiction des forêts aléatoires;
- l'apprentissage du modèle de prédiction des réseaux de neurones (et de la régression logistique);

- l'apprentissage faiblement supervisé de caractéristiques visuelles à l'aide de réseaux de neurones convolutionnels.

Mots-clés: L'apprentissage spatiale, forêts aléatoires, réseaux de neurones convolutionnels, la reconnaissance d'activités humaines, la segmentation du corps humain en parties.

Contents

List of Figures

List of Tables

Chapter 1

General Introduction

Contents

1.1 Visual recognition problem

The ultimate goal of artificial intelligence (AI), to enable machines to behave like human being, has driven an enormous amount of research from different communities to advance the frontier since the mid-20th century. Machine learning, as a core branch of artificial intelligence, is a highly active research direction. A notable definition of machine learning from Mitchell [Mitchell, 1997] is: "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E." This definition basically outlines the general methodology for machine learning: The regularities or models are learned from the experience in any way whatsoever, and then are employed to predict current data or future data. In the recent decades, remarkable progresses on artificial intelligence have been achieved and it seems that the pace toward the ultimate goal is accelerating.

Humans are able to perform many different tasks, from recognizing objects, understanding languages to complex logical reasoning and decision making etc. The common strategy is to learn an efficient model which is oriented to one particular task, for example, visual recognition, natural language understanding, scene understanding, and migrate them together in the final step.

Among the versatile abilities of human beings, the human visual system is one of the most outstanding ones after millions of years of evolution. The whole system contains billions of neurons, connecting from the retina to ventral cortical areas, and further to more abstract areas in the brain, which are

responsible for decision making. It not only enables us to perceive the colorful world, but also can rapidly direct our attention to interesting objects in the environment, segment the scene, recognize and track objects of interest. This remarkable function is of significance in an evolutionary perspective because it helps humans survive in the brutal biological law. Therefore, it attracts much attention of researchers from different disciplines, such as the neurobiology community to understand the scheme of our visual system, the computer vision community to design and implement computational models for vision tasks, the machine learning community to provide statistical and structural models for learning which can be applied to different problems.

This thesis mainly focuses on visual recognition, including object segmentation as a preliminary step for recognition, and video understanding, in particular, pose estimation and action recognition. It investigates several popular models in the computer vision and machine learning communities, and then proposes several models.

Visual recognition is one of the fundamental problems in computer vision. It is more often quoted as "object" recognition, where "object" varies according to the media containing it. There are several different recognition problems as follows:

1. "Object" in a traditional image actually is the reflection of a real 3D object on the imaging plane, since an image is the projection of a 3D scene on the imaging plane. This recognition problem covers various specific tasks like face recognition, person recognition and specific recognition from learned examples. Figure 1.1 shows several objects to be detected in images and their groundtruth bounding boxes [Everingham and Zisserman, 2010]. It has been well studied for decades, and some applications have been successfully employed in real life.

2. Another source are depth images, in which each pixel delivers the distance to the camera. Combined with traditional RGB images, they provide more information about the scene. Figure 1.2 displays the output shot by a moving Kinect in the LIRIS HARL dataset [Wolf et al., 2012]. At the same time, real-time 3D representations of objects become possible, which allows new possibilities for object recognition.

3. Video is another important media source. Objects in videos differ from the ones in images, because videos in essence are sequences of continuous frames (i.e. images); as a consequence, objects are composed of 2D dimensions plus another time dimension, which indicates the time evolution of objects and is considered to be time sequences. A typical problem in this context is action/activity recognition with a wide range of po-

Horse Motorbike Person

Figure 1.1: The images in PASCAL VOC dataset. [Everingham and Zisserman, 2010]

(a) Grayscale image (b) depth image (c) colored depth image

Figure 1.2: The handshake example in LIRIS HARL dataset. [Wolf et al., 2012]

tential applications such as security and surveillance, human-computer interaction, content-based video analysis, as shown in figure 1.3.

Visual recognition, as a crucial problem of visual systems, has continued to attract remarkable interest since its beginning, not only due to the basic research of seeking a pure interpretation of human vision cognition system, but also because of promising applications in different domains. Of course, the development of artificial visual recognition system goes along with the demand of other related industries. The earliest applications of visual recognition were limited character recognition for office automation tasks, for example, automated teller machines (ATMs) for banks. Compared to the early very limited applications to promote production efficiency, visual recognition is

Figure 1.3: The video of action "Long jump" in the Olympic dataset [Niebles et al., 2010].

widely used in numerous applications nowadays, such as military, security, health care, movies and entertainment, gaming etc. We will describe some applications in the following:

Human-computer interaction (HCI): It is a basic application that aims to improve the interaction between users and computers by making computers more usable for users and understanding users' behaviors [Sinhal et al., 2008]. One example for entertainment are game console systems based on gesture recognition. The seminal technology is the Xbox 360 gaming system equipped with Kinect, in which the key intelligence part is accurate gesture and pose recognition. Figure 1.4 demonstrates how the players control the actions in the game through physical movement in front of Kinect. The success of Kinect further promotes a series of consumer technologies related to human-computer interaction. In chapters 4 and 5 we propose human body segmentation algorithms, which improve upon the existing algorithms for the Kinect system.

Health care: This application has a large promising market because of the problem of the aging global population. Installed in homes or hospitals, visual recognition systems can help elderly, disabled or chronically sick people to live independently. Typical tasks are the recognitions of events like falling, or the verification of drug intake. Assistance with machine interaction is another possibility in this context. Figure 1.5 shows a real system for disabled people.

Surveillance: Visual recognition has been widely used in various surveillance applications. Efficient recognition and query algorithms in large scale databases enable surveillance systems to be installed in many public environments, such as highways, airports, train stations etc. Current research focuses on the prediction of crime by detecting abnormal events.

Biometrics: It refers to the identification of humans by their biological properties such as fingerprints, faces etc. In normal cases, this biometric information is unique. Some countries employ systems where users are asked to provide fingerprints on entry, which are compared to information in the passport. The Federal Bureau of Investigation (FBI) of United States is planning to build up its next generation identification biometrics database, containing multimodal biometric identifiers such as face-recognition-ready photos, and voice data.

Robotics: Visual recognition systems are usually considered as sub-systems in robotics, which allow robots to perform tasks in dangerous situations or in unreachable places to replace humans.

Figure 1.4: Schematic diagram of Microsoft's Kinect game controller system, (taken from www.edition.cnn.com).

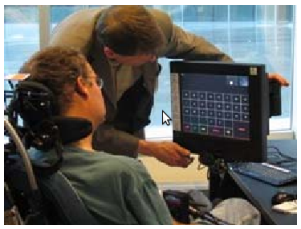


Figure 1.5: Gazing detection system for the disabled person [Sinhal et al., 2008].

1.2 Scientific challenges

As mentioned earlier, we introduce contributions to the visual recognition problem in different domains. Three different types of media (i.e. images, depth images, videos) associated with visual recognition will be dealt with. Although each has its specific characteristics and also induces particular challenges for each sub-problem, there are some common challenges for visual recognition.

Figure 1.6 shows a frequent framework for visual recognition. A test image is first processed by a feature extraction module and represented by features, which are then classified to give a decision in a classification module. The parameters are learned in the training stage which is not shown in figure 1.6. The use of features instead of raw pixel values is necessary. Compared to the high-dimensional raw data, low-dimensional features are preferable. Besides, raw pixel values are sensitive to objects' variation.

Generally, we seek a (learned or handcrafted) discriminative representation ("features") which is able to cope with different variations in the data, depending on intrinsic and extrinsic factors:

Intrinsic factors: They are associated with the inherent properties of the objects in term of appearance, pose, even gait. For object recognition,

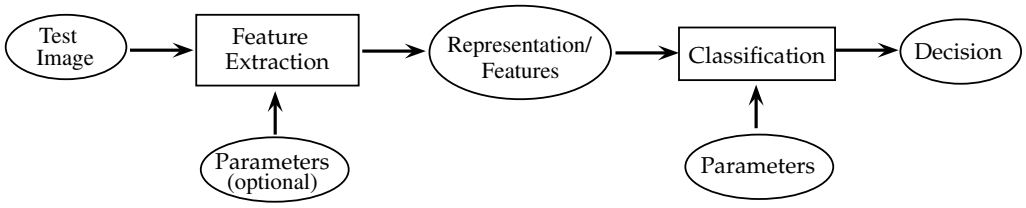


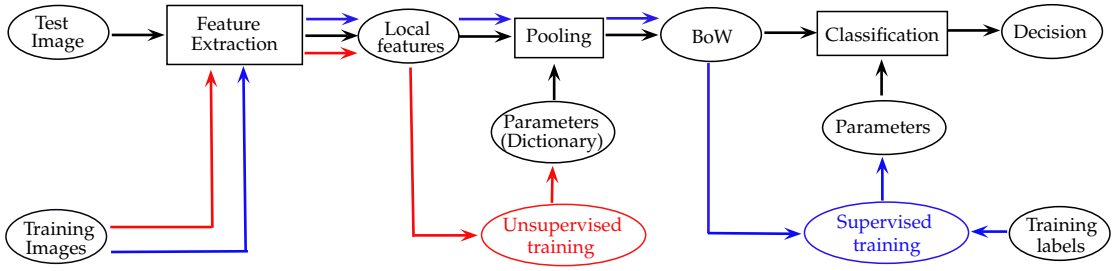
Figure 1.6: A frequent framework.

the subject can appear differently due to different clothes and poses; similarly, for action recognition, the same actions performed by different subjects look differently due to different pace rate, for instance, the length of hand swing when walking. All of these easily induce large intra-class variations. At the same time, low inter-class variations can occur between different categories. So, robust feature extractor algorithms should minimize intra-class variations and simultaneously maximize inter-class variations. In addition, the objects we are interested in are often articulated, rather than rigid. In this case, it is impossible to learn a global rigid model to represent all the possibilities of distributions of the articulated parts.

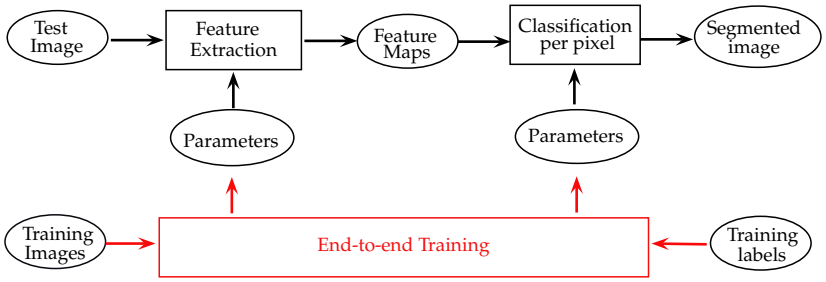
Extrinsic factors: They mainly concern the variations due to imaging conditions: viewpoint changes, and changes in scale, rotation, lighting, noise, occlusions, complex background etc. It is easily seen that even the same object displays differently on the frame if the imaging device changes its viewpoint. A good representation should be invariant to most of these transformations.

Besides the above mentioned challenges, real-time processing is often required in the practical applications of visual recognition. Moreover, some applications are installed on mobile platforms. Both requirements further limit the complexity of the algorithm. However, the model with less complexity usually yields more errors, therefore, in a real algorithm design, the cost between complexities and errors should be seriously considered and the best trade-off strategy is adopted according to the needs.

From the above discussion, we can conclude that visual recognition is a difficult task, but a large amount of applications encourage the researchers in computer vision and machine learning communities to seek more efficient and effective representations and learning algorithms to achieve the level of the human visual system. In this thesis, we address visual recognition mainly by using machine learning algorithms, or employing the machine learning framework to improve the models populated in computer vision community.



(a): The bag-of-words framework.



(b): The segmentation framework.

Figure 1.7: Two frameworks for specific recognition problems. (best viewed in color).

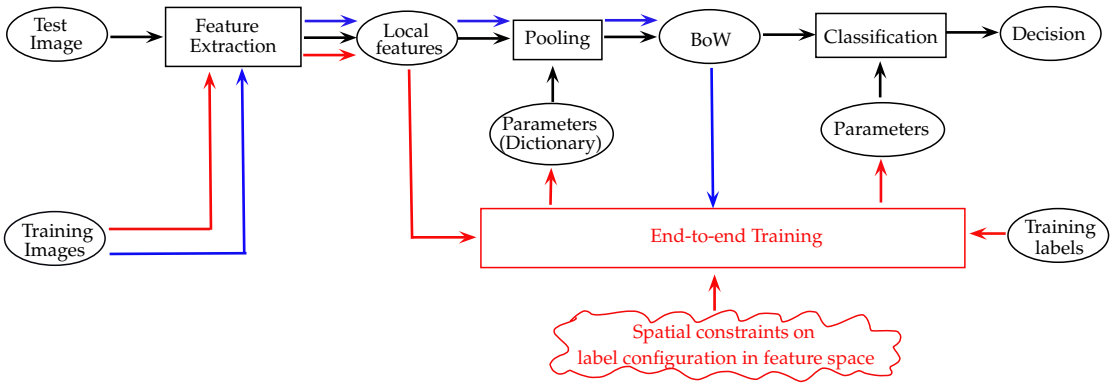
1.3 Our contributions

In sections 1.1 and 1.2, we have briefly discussed visual recognition and its challenges. In this thesis, we focus on the role of learning for visual recognition, for example, learning a more discriminative representation for objects through end-to-end learning.

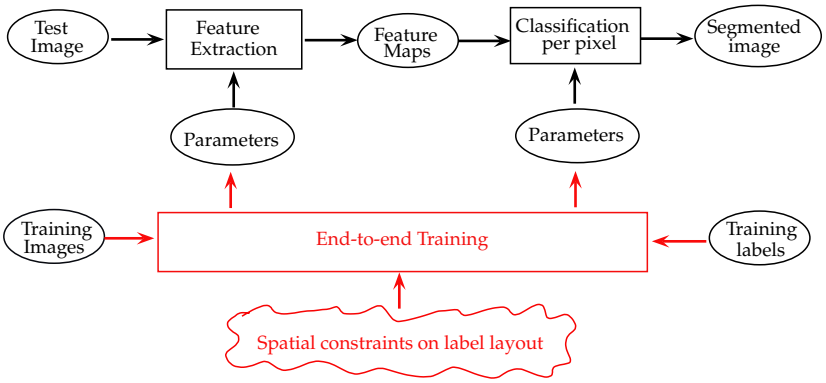
Figure 1.6 shows a common framework for visual recognition. It can be refined to several cases according to specific recognition problems:

- In the case of bag-of-words models, which will be described in more detail in section 2.3.2.2 of chapter 2, the representation corresponds to a histogram, which is calculated from a set of descriptors pooled from sparse primitives. The descriptors are usually handcrafted. The pooling process is calculated through a codebook (dictionary), which is classically learned in an unsupervised way through clustering. This is shown in figure 1.7a. The black path shows the data flow during the testing procedure. The red path shows the data flow during the unsupervised codebook training used for pooling and the blue path shows the data flow during the supervised classifier learning.

- The case of object segmentation by classification is shown in figure 1.7b. In this case, the generic framework shown in figure 1.6 is applied inde-



(a): The bag-of-words framework.



(b): The segmentation framework.

Figure 1.8: Our improved frameworks. (best viewed in color).

pendently to each pixel of an image. Here we do not deal with regularization like it is performed in discrete models (MRF, CRF etc.) or continuous-discrete formulations (Mumford-Shah functional etc.). The feature in this case can either be handcrafted (HOG, SIFT etc., see section 2.2 chapter 2) or learned in an end-to-end training setup (see section 2.4 of chapter 2). In this case, the features are learned together with the prediction model (see section 2.3 of chapter 2).

The goal of our work is the integration of as much as information as possible into the learning process. We proceed by different ways: (i) integrating additional constraints (related to prior information) into the learning algorithm; (ii) coupling stages which are classically performed separately. A large emphasis is put on geometric and spatial information. Here our contributions are given as follows:

- We propose an end-to-end training approach for BoW models. In the framework shown in figure 1.8a, the dictionary is learned together with

the prediction model in a supervised way, i.e. including training labels. We impose spatial constraints on label configurations in feature space. In this way, the features are directly related to the classification problem, and they are updated as learning becomes more discriminative. We experimented this approach on the bag-of-words (BoW) model for action recognition.

- We integrate prior information in terms of spatial label layout of parts into different machines, i.e. Randomized Decision Forest (RDF), Convolutonal Neural Networks (CNN), and classifiers such as logistical regression (LR) or multilayer perceptron (MLP). respectively, as shown in figure 1.8(b). This concerns applications, where a classifier takes decisions on labels, which can be spatially interpreted, for instance where objects are segmented into meaningful parts. Labels here refer to parts, which are located in the human body. Spatial relationships can significantly improve performance in this case when the size of available training instances is very limited.

  Traditionally, these constraints are injected using models which require to solve a combinational problem, e.g. graphical models like MRFs and CRFs. The additional computational burden makes real-time operation impossible. We propose to integrate these constraints directly into the learning algorithm of a traditional classifier (RDF, CNN, LR or MLP). Classification still proceeds independently pixel per pixel, which means that the computational complexity is equivalent to the classical approaches.

- All the frameworks we propose do end-to-end training as shown in figures 1.8(a) and 1.8(b). Representations are automatically learned, rather than hand-engineered [1].

- We employ our spatial learning algorithms on specific problems, namely human body part estimation and segmentation. Visual recognition of human parts is an important intermediate step, which can be used for person recognition or human pose estimation. It is necessary to mention that our algorithms can be used for the segmentation of any other objects, articulated or not.

[1] In the case of BoW model, the dictionary of the representation is learned, while the low-level features stay handcrafted.

1.4 Overview

Here we overview the contents in the following chapters:

Chapter 2 discusses the state-of-the-art on visual recognition. We present popular feature extraction techniques, and widely used models in computer vision. We also review common machine learning algorithms for visual recognition with a focus on the ones related to the thesis.

Chapter 3 proposes two algorithms for end-to-end learning of bag-of-words (BoW) models on action recognition, where the dictionaries are learned in a supervised way. The codebook for BoW models are updated through error backpropagation. One algorithm described in section 3.3.2 is based on classical error backpropagation, the other described in section 3.3.3 is based on cluster reassignments. We apply both algorithms to the human action recognition problem, and show that our end-to-end learning technique makes the BoW model more discriminative.

Chapter 4 proposes spatial learning algorithms for Randomized Decision Forests in section 4.3 and for Convolutional Neural Networks and LR/MLP classifier in section 4.4, respectively. We integrate the spatial layout of object configurations in terms of pairwise relationship of neighbors and non-neighbors into the learning machines with different architectures. In other words, we attempt to learn structural machines for recognition. The learned machines demonstrate better performance on object segmentation problems.

Chapter 5 presents one application for the algorithms proposed in chapter 4, namely segmentation of humans into body parts. Part segmentation is an important research area in computer vision, which can be regarded as the bridge between low level image processing to high level scene understanding. In this work we consider it as a problem of classification, which is addressed by our algorithms.

Chapter 6 gives general conclusions and describes ongoing and future work.

Chapter 2

State of the art

Contents

2.1 State of the art

The visual recognition problem is sometimes quoted as the object recognition problem (the word "object" being interpreted in a very large sense such as object, activity, gesture etc.), i.e. how to make a machine recognize objects in an image or in a video without any other assistance. The ultimate goal is to produce an artificial visual system having a performance equivalent to the human visual system, which is capable of recognizing as many as millions of objects. However, this powerful and universal visual system has not been reported until now.

The type of the input depends on the sensing device. It can be a gray or real colored projection to a 2 dimensional image plane of a common object by

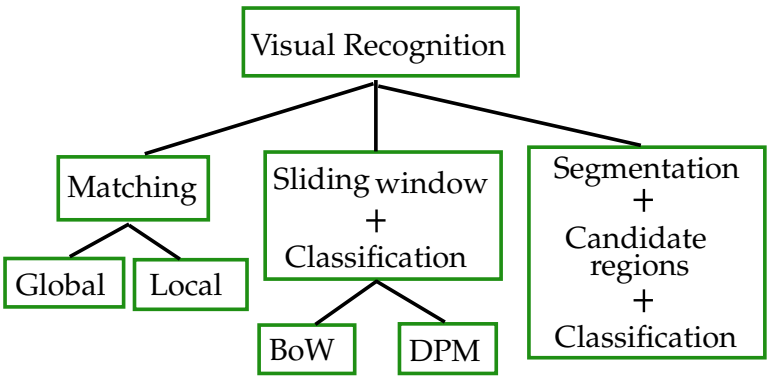


Figure 2.1: The general visual recognition framework. BoW is the abbrev. for bag-of-words [Csurka et al., 2004] models, and DPM for deformable part based models [Felzenszwalb et al., 2010].

a traditional camera, a classical problem in computer vision. It can also be a depth image of an object, where each pixel value corresponds to the distance to the depth-sensing camera. Moreover, it can be a spatio-temporal "object" in videos such as an action and an activity associated with one or several persons.

There is a large amount of literature on visual recognition, and various methods have been introduced for the "object" recognition problem. A frequently used pipeline can be shown in figure 1.6 in chapter 1. There are two modules: given an input (image or video), features are firstly extracted from objects, eventually dimension reduction techniques such as Principle Component Analysis (PCA), clustering etc. are further applied on the raw features to get amenable models. Finally, machine learning approaches, for instance, support vector machines (SVM), randomized decision forests (RDF), etc. are employed to create a decision for specific tasks (e.g. recognition, detection, localization etc.). Hence, research on visual recognition focuses on two aspects: on one hand, much attention is taken to construct and optimize features and representations, discriminatively representing the "object". More discussions on this can be found in section 2.2. On the other hand, research in machine learning attempts to seek powerful classification techniques, which better learn the boundaries between the categories. Beyond simple classification of vectorial data, other techniques are able to deal with structural data and spatial, temporal or spatio-temporal relationships. Examples are Structural SVM, Hidden Markov Models (HMM), Markov Random Fields (MRF), graph matching, etc.

Visual recognition can be performed through different families of approaches. Figure 2.1 gives a taxonomy of some frequent ones. They all resolve

the same main problem, which is to identify (detect) the region containing the object of interest and then to recognize the object itself. This is a chicken-and-egg problem, since detecting an object theoretically requires its recognition, whereas recognition requires extracting features from the detected region. We would not talk more about this, several techniques thus are presented to avoid this dilemma.

Matching is a prototype based technique, which is performed by searching for instances of a model (image, point etc.) in the scene through similarity, and by thresholding the similarity, taking into account structural information. According to the used features, there are global matching and local matching techniques. Another simple solution is to slide a window over the image or video, and to recognize the presence of the object by classification. It is also further divided into local sparse features based (e.g. bag-of-words models [Csurka et al., 2004]) and local dense features based (e.g. deformable part based models [Felzenszwalb et al., 2010]). Another possibility is to first over-segment an image into a set of candidate object segments, fuse small regions together and followed by a classification step.

Existing methods can also be organized by the differences in feature types, where most types of methods require a specific type of features, for instance, local matching requires local features. Figure 2.2 shows a taxonomy of feature types.

The next three sections are dedicated to these aspects:

- Section 2.2 will describe different feature types, as shown in figure 2.2.
- Section 2.3 will describe different recognition techniques, as shown in figure 2.1.
- Section 2.4 will describe feature learning techniques, where features are not handcrafted, but learned.

Features and techniques will be presented for **both**, spatial data and spatio-temporal data, i.e. for **object recognition** and for **activity recognition**.

2.2 Type of features

As mentioned in Chapter 1, features and representations should possibly be as invariant and as discriminative as possible, to make the subsequent recognition stage less difficult.

Figure 2.2 shows a taxonomy of different feature types. They can be global or local; they can be handcrafted, i.e. designed in all details; or they can be automatically learned in an unsupervised or a supervised manner.

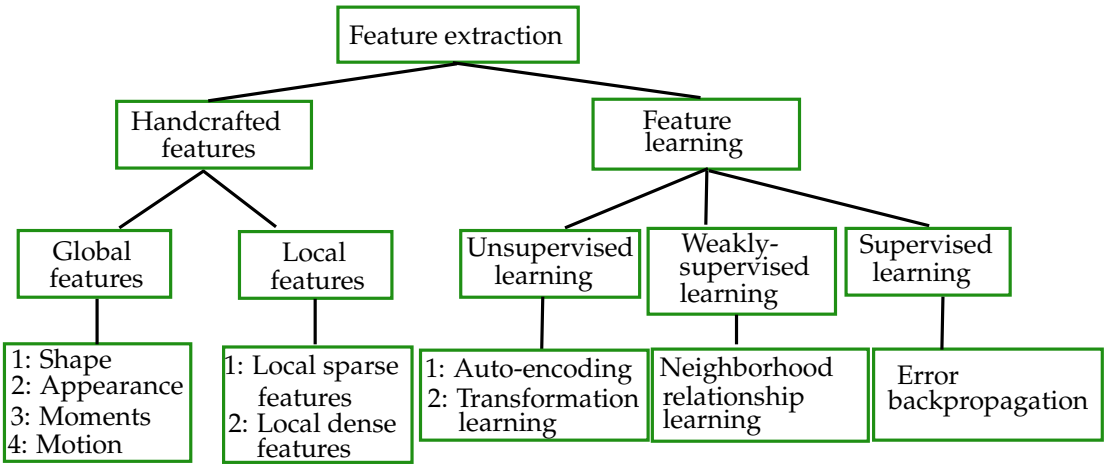


Figure 2.2: Feature extraction taxonomy.

One advantage of this framework is that it can be composed of an end-to-end framework where features are learned together with the prediction model, as a result, the features are learned to best suit the classification stage.

In this section, we will present several types of global and local features. Automatic feature learning will be discussed in its own dedicated section 2.4.

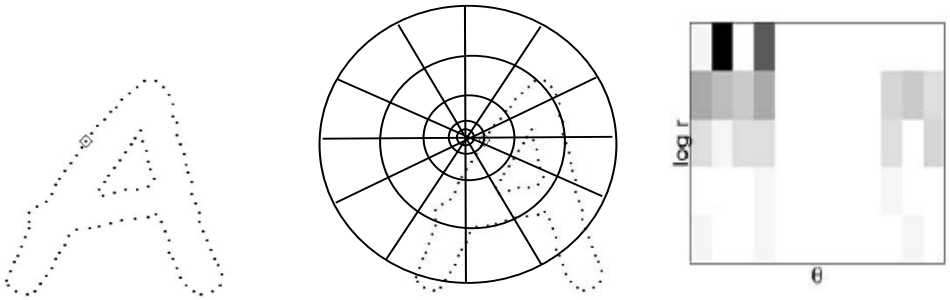
2.2.1 Global features

Global features are extracted on the whole area covered by the object of interest, which is therefore described by a single representation (mostly vectorial). The research on global representations dominated the beginning of object recognition.

2.2.1.1 Shape based features

Shape is an important property of an object, it is related to contours for 2 dimensional objects and to surfaces for 3 dimensional objects. The earlier shape features focus on low level information such as edge or texture. Snakes introduced in [Kass et al., 1988] apply energy minimization to contour construction and the edges are the features. Active Shape Models [Cootes et al., 1995] integrate global shape constraints into snake constructions to capture the deformable shape.

Shape context is a contour based descriptor, which describes a coarse distribution of all other points with respect to a given point on the shape [Belongie et al., 2002]. It works well on poorly textural objects, such as letters, sketches, and is widely adopted by the community. The main computation steps are as follows: an edge image of an object is firstly obtained and its



(a) (b) (c)

Figure 2.3: Shape context illustration: (a) the shape of letter "A", (b) the log-polar coordinate on the shape, (c) the shape context of the given point. This figure is reproduced from [Belongie et al., 2002].

shape is discretized into a number of points. A log-polar coordinate system is put on the shape with its origin at a given point. The shape context for the given point is a histogram, where each bin in the log-polar coordinate counts the number of points falling in that bin. Each point's shape context features are composed of the relative position information of all other points w.r.t itself; therefore, shape context is very rich and discriminative. It is very suitable for point-to-point matching problems.

Shape based features can be also used to describe spatio-temporal objects. Gorelick et al. [Gorelick et al., 2007] propose a global descriptor for space-time data. They stack 2D object shapes to form a volumetric space-time shape. Space-time saliency and space-time orientation is extracted from the space-time shape volume, serving as features. Space-time saliency indicates which portion of human action has the highest likelihood of action occurrence, and space-time orientation estimates the local orientation and aspect ratio of different space-time parts. Figure 2.4 shows the space-time saliency of waving hands and walking and the space-time orientations of plates and sticks for walking. Their method does not require prior video alignment and has proper robustness to partial occlusions, imperfections in the extracted silhouettes. Another similar space-time volume is extracted from 2D contours for actions in [Yilmaz and Shah, 2005]. Several geometry properties are computed to describe the space-time volume, such as peaks, pits, valleys and ridges. These features are quite invariant because the convex and concave parts of the object are view invariant.

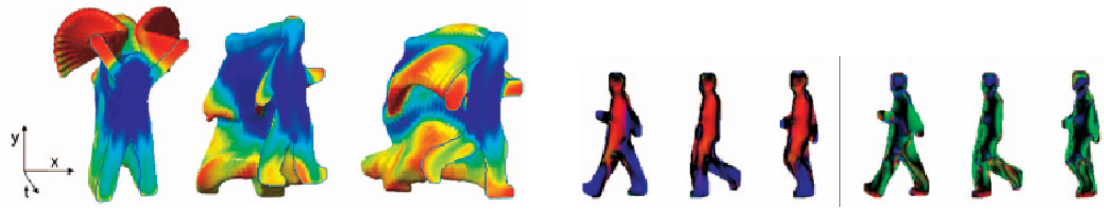


Figure 2.4: Space-time shape features. left: space-time saliency, the color indicates the saliency strength: from red (high) to blue (low); right: space-time orientations of plates and sticks: the red regions indicate horizontal plates, the blue regions indicate plates in temporal direction, and the green regions indicate vertical sticks. This figure is reproduced from [Gorelick et al., 2007] (best viewed in color).

Figure 2.5: The examples of eigenfaces. (This figure is reproduced from [Heseltine et al., 2003].)

2.2.1.2 Appearance based features

The appearance of an object is expressed in terms of pixel intensities, which is also a clue for global features. The earlier work investigated low image features based on intensity and texture for visual recognition. However, they are high dimensional vectors and sensitive to the noise and lighting conditions. Dimension reduction techniques are common to use. The appealing concept for face recognition is "eigenfaces" [Turk and Pentland, 1991], which means that eigenvectors of face images are used for recognition. A set of eigenfaces can be derived by performing principle component analysis (PCA) on a large collection of face images. These eigenfaces build a basis and other face images can be represented by a set of reconstruction coefficients. "Eigenfaces" provide a compact representation of face appearance, while simultaneously reducing the dimensions. The follow-up work in [Murase and Nayar, 1995, Leonardis and Bischof, 2000] extend it to general visual recognition system instead of specific to face recognition.

An alternative to "eigenfaces" are "fisherfaces" [Belhumeur et al., 1997], which use linear discriminant analysis (LDA) for reduction. Another fur-

ther alternative is Active Appearance Models proposed by Cootes et al. [Cootes et al., 2001], which is a statistical learning method for shapes and textures of faces. PCA is performed on a set of landmarks on the face to encapsulate the variation of different faces.

More recently, these kinds of techniques are used in conjunction with deep learning. The combination is also sometimes called "intelligent dimensionality reduction" and briefly described in section 2.4.2.

2.2.1.3 Moment based features

Moments are an extensively used concept in different disciplines. In mathematics, a moment is used to measure the characteristics of the shape of a set of points, such as: mean, variance, and skewness. Moment invariants can also be applied to two-dimensional images, where they measure the content of images with respect to different axes.

Hu moments were first introduced for visual pattern recognition [Hu, 1962]. They are derived from the theory of algebraic invariants. For a gray image of size $N*M$, the image intensities in the image coordinate (x, y) are represented by a function $f(x, y)$, the moment of order $(p+q)$ is defined by:

$$m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q f(x, y), \tag{2.1}$$

where $p, q = 0, 1, 2, 3, \cdots$.

In order to achieve translation invariance, the central moment μ_{pq} is given by:

$$\mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \bar{x})^p (y - \bar{y})^q f(x, y), \tag{2.2}$$

where $\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{10}}{m_{00}}$. To be further scale invariant, the central moment is once again normalized by:

$$\mu_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \quad \gamma = 1 + (p+q)/2. \tag{2.3}$$

The seven Hu moments are given as:

$$\begin{cases} M_1 &= \mu_{20} - \mu_{02} \\ M_2 &= (\mu_{20} - \mu_{02})^2 + 4\mu_{11}^2 \\ M_3 &= (\mu_{30} - \mu_{12})^2 + (3\mu_{21} - \mu_{03})^2 \\ M_4 &= (\mu_{30} + \mu_{12})^2 + (3\mu_{21} - \mu_{03})^2 \\ M_5 &= (\mu_{30} - 3\mu_{12})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] \\ & \quad +(3\mu_{21} - \mu_{03})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] \\ M_6 &= (\mu_{20} - \mu_{02})[(\mu_{30} + \mu_{12})^2 - (\mu_{12} + \mu_{03})^2] + 4\mu_{11}(\mu_{12} + \mu_{30})(\mu_{21} + \mu_{03})] \\ M_7 &= (3\mu_{21} - \mu_{03})(\mu_{30} + \mu_{12})[(\mu_{30} + \mu_{12})^2 - 3(\mu_{21} + \mu_{03})^2] \\ & \quad +(3\mu_{21} - \mu_{30})(\mu_{21} + \mu_{03})[3(\mu_{30} + \mu_{12})^2 - (\mu_{21} + \mu_{03})^2] \end{cases}$$

Although Hu moments are invariant to scale, translation, and rotation, one problem is that they are derived from low order moments, whose capacity of describing a shape is very limited. Another disadvantage is their difficulty to reconstruct the image. An example of their usage is their application to action recognition in [Bobick and Davis, 2001].

Zernike moments [Teague, 1980] have been proposed to overcome these disadvantages. They are computed based on the basis set of orthogonal Zernike polynomials. Interesting readers can find more details in [Teh and Chin, 1988]. They have been successfully applied to object recognition [Revaud et al., 2009, Ta et al., 2008].

2.2.1.4 Motion features

Motion is a special property of objects in video, which is an important clue for action recognition. Several attempts have been made to construct motion features. Motion Energy Images (MEI) and Motion History Images (MHI) introduced by Bobick et al. [Bobick and Davis, 2001] are representative ones. Background is firstly subtracted, and then a sequence of silhouettes is aggregated into a single static image, which indicates where the motion occurred and when w.r.t. the current time instant, respectively. Let $I(x, y, t)$ be the intensity value at time t and location (x, y) in a video; we can obtain the region of motion by the difference of consecutive frames. A binary map $D(x, y, t)$ is obtained by thresholding the differences where there is high likelihood of motion:

$$D(x, y, t) = \begin{cases} 1, & |I(x, y, t) - I(x, y, t-1)| > T \\ 0, & \text{otherwise} \end{cases} \tag{2.4}$$

Then the MHI function is defined by:

$$h_\tau = \begin{cases} \tau, & \text{if} \quad D(x, y, t) = 1 \\ \max(0, h_\tau(x, y, t-1)), & \text{otherwise} \end{cases} \tag{2.5}$$

Figure 2.6: MEI (left) and MHI (right) of a person raising both hands. This figure is reproduced from [Bobick and Davis, 2001].

where τ is the total number of frames. MHI are constructed by giving the sequence frames decaying weights with higher weight given to new frames and low weight given to older frames. MEI can be obtained through setting the same value for the pixels above zero in MHI. Figure 2.6 illustrates MEI and MHI of a person raising both hands. MEI and MHI are not suitable to complex actions due to over-writing over time. Several extensions on these features exist in [Weinland et al., 2006, Hu and Boulgouris, 2011]. They also have been extended to depth videos in [Roh et al., 2010], where they are called "Volume Motion Template".

Global features are not restricted to the above mentioned ones. More types of global features for visual recognition can be found in [Poppe, 2010]. However, global features share a number of common disadvantages: (i) many of them need pre-segmentation to get rid of the background scenes; (ii) global features are not robust to noise, a little noise on the boundary may totally change the orientation of the global features; (iii) they can not cope with occlusion. Local features described in section 2.2.2 can technically avoid these disadvantages.

2.2.2 Local sparse features

Local features differ from their global ones in that they describe the local content of an object, i.e. the content on a small patch. The object can be described by the combination of all the local features from the object. Of course, it is not always practical to use all the possible local primitives, because most of them fall in non-interesting areas containing little information about the object, such as the background and visually identical areas.

There is another way to look at local features-based recognition: it is interpreted as a low-level part-based model. A local primitive can be regarded as a small part of the object, all the parts therefore are arranged to an object in a particular spatial configuration. Much work on building a high level part

as group of local primitives has been reported, rather than based on local primitives. The readers can find more details about part based models in Section 2.3.

Local features generally describe the properties of the neighborhood of some local primitive, e.g. a point, an edgel, or a small segmented region. Interest points are the most widely used local primitives containing much information, and they are the ones we are interested in. Two questions arise in this context:

- How can we detect interesting local primitives, e.g. point locations? Several measure criteria are proposed to constrain the interest points in image or video.

- How can the area around each point be described locally?

Here we would like to point out that interest points and descriptors are independent: once interest points are detected, any descriptor techniques can been applied. In the following, several well-known interest point detectors are presented as well as descriptors.

2.2.2.1 Interest point detectors

The work on interesting points can be traced back to [Moravec, 1981] on stereo matching using corner detection. His corners are detected by shifting a local window in the image to determine the average intensity changes. It was improved by Harris and Stephens to discriminate corners and edges [Harris and Stephens, 1988]. The basic idea to find the point whose value is above the threshold with Harris measure, which is defined on the first order derivatives as follows:

$$\mu = G(x, y) * \begin{bmatrix} L_x^2 & L_x L_y \\ L_x L_y & L_y^2 \end{bmatrix} \tag{2.6}$$

$$\det(\mu) - \alpha \ \mathrm{trace}^2(\mu) > \mathrm{threshold} \tag{2.7}$$

where $G(x, y)$ is two-dimensional Gaussian function, L_x and L_y are the Gaussian derivatives on the image along with x and y direction, α is a parameter to control the interesting point types, e.g. isotropic points, or edge points. Harris's detector was first employed to image matching problems. The work in [Schmid and Mohr, 1997] extended Harris's detector to the general recognition problem for image retrieval. It had been widely used in the community due to the fact that it can produce repeatable corners and for some time hardware implementations were available.

Arguably, the most widely used interest point detector is the SIFT detector introduced by Lowe [Lowe, 2004]. He proposes a scale-invariant keypoint detector by searching over all scales and image locations, implemented as a difference-of-gaussian operation:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{2.8}$$

where $G(x, y, k\sigma)$ is a two-dimensional Gaussian function with scale σ, k is a constant to define its nearby scale, $*$ is the convolution operation. The detected local extrema are further rejected if they have low contrast by equation (2.9) or are located along an edge by equation (2.10).

$$D(\hat{\mathbf{x}}) = D + \frac{1}{2}\frac{\partial D^T}{\partial \mathbf{x}}\hat{\mathbf{x}}, \tag{2.9}$$

$$\frac{\mathrm{Tr}(\mathbf{H})}{\mathrm{Det}(\mathbf{H})} < \frac{(r+1)^2}{r}. \tag{2.10}$$

Here, $\hat{\mathbf{x}}$ is the offset of the local extrema, and $\mathbf{H}$ is the 2×2 Hessian matrix of the local extrema.

Mikolajczyk and Schmid [Mikolajczyk and Schmid, 2002] propose an algorithm to address affine invariant interest points. Interest point candidates are first detected by using multi-scale Harris detector, and then for each point an iterative procedure is performed to simultaneously modify the position, scale and shape of the point neighborhood, and an affine transformation is estimated to obtain affine invariant interest points.

Additional work on 2D interest point detectors includes SURF (Speeded up robust features) [Bay et al., 2006], FAST (Features from Accelerated Segment Test) [Rosten and Drummond, 2006] etc.

We temporally turn our attention to present some work on 3D interest point detector for action/activity recognition. Laptev [Laptev, 2005] extends the 2D Harris detector to 3D space-time detector. Space-time interest points (STIP) are those points where the local neighborhood has a significant variation in both the spatial and the temporal domain. Similarly, the spatio-temporal second-moment matrix is a 3-by-3 matrix composed of first order spatial and temporal derivatives as following:

$$\mu = G(x, y) * \begin{bmatrix} L_x^2 & L_x L_y & L_x L_t \\ I_x I_y & I_y^2 & L_y L_t \\ L_x L_t & L_y L_t & L_t^2 \end{bmatrix} \tag{2.11}$$

One drawback of this method is that it provides a relatively small number of stable interest points (see figure 2.7); it also consumes expensive computational resources due to a iterative searching procedure in a 5 dimensional space over (x, y, t, σ, τ).

Figure 2.7: Illustration of Harris3D interest points of a walking action. This figure is reproduced from [Laptev, 2005].

Figure 2.8: Illustration of interest points of a walking action by Dollar's detector [Dollar et al., 2005]. This figure is reproduced from [Niebles et al., 2008]

Dollar et al. [Dollar et al., 2005] apply two linear separable Gaussian filters in the space dimensions and a Gabor filter on the temporal dimension individually to obtain dense interest points. The form of the response function is given as:

$$R = (I * g * h_{ev})^2 + (I * g * h_{od})^2 \tag{2.12}$$

Where g is the 2D spatial Gaussian smoothing kernel and, $h_{ev}(t;\tau,\omega) = -cos(2\pi t\omega)\exp(-t^2/\tau^2)$ and $h_{od}(t;\tau,\omega) = -sin(2\pi t\omega)\exp(-t^2/\tau^2)$ are a pair of 1D Gabor filters applied temporally. The local maxima of the response function are selected as spatio-temporal interest points. Figure 2.8 shows the interest point of a walking action.

As mentioned above, Laptev's detector is scale-invariant, yet it detects very sparse interest points. Dollar's detector extracts a more dense set of interest points, yet they are not scale-invariant. Willems et al. [Willems et al., 2008] therefore propose an efficient dense and scale-invariant spatio-temporal interest point detector, which uses the determinant of the Hessian as saliency measure. There are other attempts to detect the corners in the video: Yu et al. [Yu et al., 2010] propose V-FAST which actually is the extension FAST corner detection. The FAST detector is applied to the three planes: XY, YT and XT. Saliency is detected if n contiguous pixels on the circle are all brighter or darker than a center pixel. Le et al. [Le et al., 2011] obtain interest points by thresholding the responses from an Independent Subspace Analysis (ISA) network. Everts et al. [Everts et al., 2013] investigate multi-color

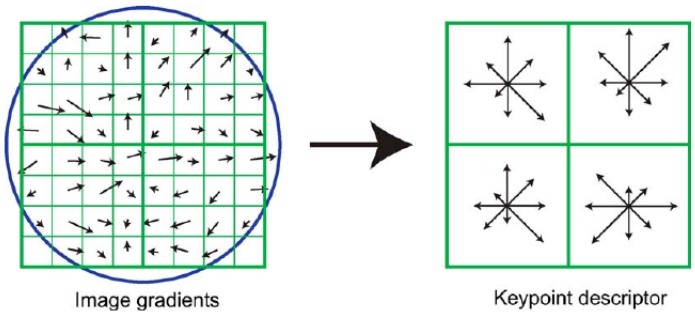


Figure 2.9: The SIFT descriptors. This figure shows a 2*2 subregions from a 8*8 neighborhood. This figure is reproduced from [Lowe, 2004].

channel spatio-temporal interest points, compared to intensity-based STIP in [Laptev, 2005]. They are invariant to highlights and shadows and perform better on action recognition than their intensity-based counterparts.

2.2.2.2 Descriptors

Descriptors are local in an image or video, i.e. in this case, they describe a local neighborhood around each primitive (e.g. point). We will review some popular local descriptors in this section.

SIFT and its extensions — SIFT (Scale Invariant Feature Transform) in [Lowe, 2004] is a very successful local descriptor due to its robustness and discriminative power. It first computes the gradient magnitude and orientation at each pixel in the 16*16 neighborhood centered around an interest point. They are weighted by a Gaussian window overlaid on this neighborhood, giving more emphasis to the pixels near the center and less emphasis to the ones far from the center. For each 4*4 subregion, an orientation histogram of 8 bins is accumulated, resulting in a descriptor of size 4*4*8 for each interest point. The contribution to each bin for each pixel is also weighted by the gradient magnitude. Figure 2.9 illustrate the scheme of SIFT features.

Ke and Sukthankar [Ke and Sukthankar, 2004] developed a descriptor similar to the SIFT descriptor. It applies Principal Components Analysis (PCA) to the normalized image gradient patches and gives a more distinctive and more compact descriptor. It performs better than the SIFT descriptor in accuracy or efficiency. Mikolajczyk and Schmid [Mikolajczyk and Schmid, 2005] propose Gradient location-orientation histograms (GLOH). They first compute SIFT descriptors for a log-polar location grid with 3 bins in radial direction and 8 bins in orientation, and then apply PCA to get a compact descriptor, which shows better performance than the SIFT descriptor.

Although SIFT and its extensions are very discriminant, it is com-

putationally demanding, and also requires a large amount of memory to store the descriptors due to its 128-vector. Much work has been done to improve its efficiency while preserving its power. The SURF descriptor [Bay et al., 2006] makes use of integral images to speed up histogram computation, yielding good approximation to SIFT. However, its descriptor length is an issue. Binary Robust Independent Elementary Features (also BRIEF) [Calonder et al., 2010] addresses both issues by building a string of binary values based on intensity difference comparison, but the experiments show that BRIEF is very sensitive to rotation. ORB (Oriented FAST and Rotated BRIEF) [Rublee and Rabaud, 2011] is proposed to make it rotation invariance. SIFT have also been extended to spatio-temporal space for action recognition [Scovanner et al., 2007].

HOG and HOF — Histograms of oriented gradients (HOG) [Dalal and Triggs, 2005] are very similar to SIFT. They are implemented by dividing the neighborhood of an interesting point into small spatial regions ("cells"), accumulating a local 1-D histogram of gradient orientation for each cell. All the histogram entries from different cells are aggregated into one histogram. Contrast normalization is performed on a larger spatial region ("block") to make the representation invariance to illumination, etc. Minetto et al. [Minetto et al., 2013] propose a HOG-based texture descriptor(T-HOG) for text detection. In [Dalal et al., 2006], Histograms of Flow (HOF) are proposed for videos and are computed from optical flow instead of gradient direction, keeping the rule of the technique similar to HOG. HOG and HOF features are often combined to a single descriptor for activity recognition from videos [Laptev et al., 2008]. HOG3D descriptor [Klaser et al., 2008] is another extension of HOG, which computes histograms of 3D gradient orientations for video.

LBP and its extensions — Local binary patterns (LBP) are a highly successful descriptor, which has originally been proposed by Ojala et al. [Ojala et al., 1996] for texture classification, and extended to various applications, for example, face recognition [Ahonen et al., 2006] and human detection [Huang, 2008]. Advantages are robustness to illumination changes and computation efficiency. LBP is also employed as descriptor for interest points.

LBP describes each pixel by a vector of binary values from its neighborhood. A set of binary values form a binary number, serving as features. N pixels are sampled on the circle with radius R centered at a given pixel. Each sampled pixel is compared to the gray value of the center pixel, if the intensity is higher or equal, the output is set to 1, otherwise to 0. A series of bits from

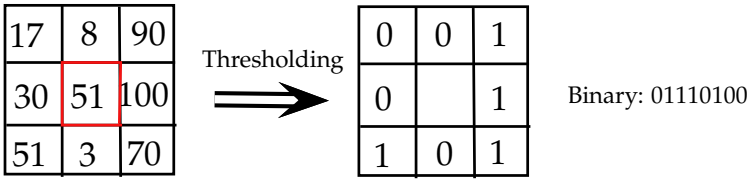


Figure 2.10: Illustration of the LBP descriptor. The values in the left grid are gray-scale values, and the values in the right grid is the binary mask after thresholding by the center gray values.

the circle form a binary number:

$$LBP_{R,N} = \sum_{i=0}^{N-1} s(n_i - n_c)2^i, \quad s(x) = \begin{cases} 1, & x \geq 0, \\ 0, & \text{otherwise} \end{cases} \tag{2.13}$$

where n_c is the gray-scale value of the center pixel and n_i is the gray-scale value for N pixels equally sampled on the circle with a radius R. Figure 2.10 shows an illustration of LBP features with $R = 1$ and $N = 8$. It can be seen that multi-scale LBPs are easy to extract by changing the radius R.

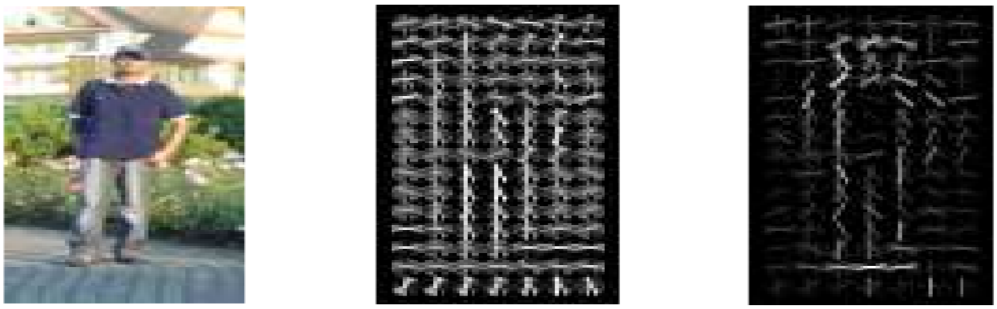
One extension is uniform LBP, which is a special case of LBP. A local binary pattern is called uniform LBP if and only if at most two times bitwise transition between 0 and 1 occurs in the binary pattern, for instance, the LBP 00011100 has two transitions, whereas 00010110 has four transitions. The uniform LBPs allow to build a macro representation for texture, because most of the local binary patterns in natural image are uniform, as observed by Ojala et al. Each uniform LBP has a separate label and the rest non-uniform LBP share one label.

Rotation invariant LBP can be achieved by two different ways: one is by rotation invariant mapping, based on the fact that patch rotation will induce a circular rotation by the same angles of the LBP codes. In detail, each LBP binary code is mapped to its minimum value, which is obtained by circular LBP rotation:

$$LBP^{ri}_{R,N} = \min_i \mathrm{ROR}(LBP_{R,N}, i), \tag{2.14}$$

where $\mathrm{ROR}(x, i)$ is the bit values after bitwise shift version by i steps for code x. Another way is to perform the Discrete Fourier Transform on the uniform LBP histogram, where each bin is the number of occurrence of the uniform LBP in image. More illustration of these features can be found in [Ahonen et al., 2009].

A spatio-temporal extension of LBPs is introduced [Zhao and Pietikäinen, 2007]. The LBP codes from three orthogonal planes: XY, XT and YT are concatenated into one vector. Dominant LBPs



(a) (b) (c)

Figure 2.11: The visualization of the HOG descriptor: (a) an image window with a person, (b) the HOG feature for each cell; (c) the learned weights for each feature in HOG. This figure is reproduced from [Dalal and Triggs, 2005].

have been introduced by Liao [Liao et al., 2009], making use of the dominant local binary patterns, which frequently occur in the image to capture the texture information.

Raw data is not suitable to describe the local content. However, recent work by Song et al. [Song et al., 2013] proposes to learn affine transformations between image space and color space, which are used to construct descriptors combining color and spatial information. As a result, the descriptors are invariant to affine transformations and photometric changes.

2.2.3 Local dense features

The above descriptors are computed locally on interest point locations, so their discriminative abilities also rely on the interest point detection. At the same time, there are some descriptors which are computed densely in the image without interest point detection.

Local dense features are obtained by computing local features for each pixel in the image. All descriptors described in section 2.2.2.2 can be adopted. The common practice is that an image is divided into a batch of grids; for each grid, a vector of local descriptor is extracted, and all the descriptors are eventually aggregated into one vector. Figure 2.11 shows an example of dense HOG features.

Compared to global features and local sparse features, local dense features are capable of capturing the global character of the object, and also integrating local properties. At the same time, local dense features also combine information from the context, which seems to be an important supplementary cue for visual recognition.

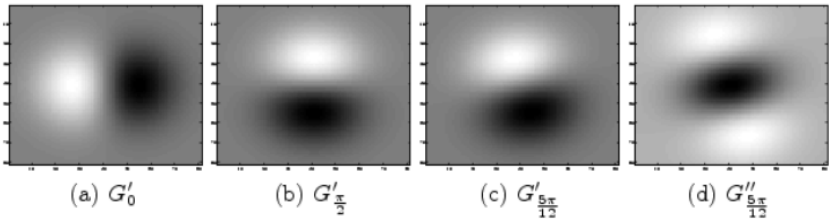


Figure 2.12: First and second order steerable filters. (a-b) Gaussian basis, (c-d) Gaussian oriented filters. This figure is reproduced from [Villamizar et al., 2006].

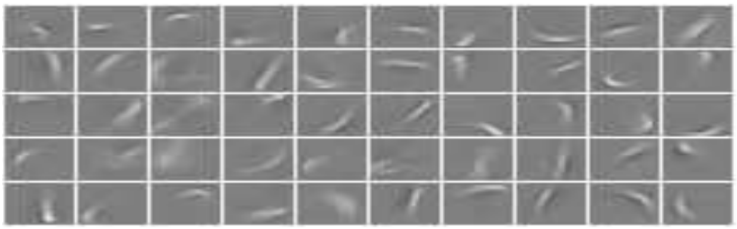


Figure 2.13: The Gabor-like filters learned by the CNN. This figure is reproduced from [Ranzato et al., 2007].

There is another different kind of local dense features based on convolution/correlation between the input image and a set of filters. In these approaches, a stack of feature maps, after convolving the image with the filters, is obtained, serving as local dense features. The filter size usually is much smaller compared to the image size, allowing to combine the local neighborhood into feature construction. The resulting feature maps are dense because a descriptor is computed for each pixel, and they are also local due to the receptive size of filter, taking the local neighborhood into feature construction. The used filters usually describe the content, for example edge orientation and strength, in the local neighborhood.

The filters can be Gabor filters [Kamarainen et al., 2007], steerable filters [Yokono and Poggio, 2004], or complex exponential functions, etc. Gabor filters are discovered to model the behavior of simple cell in the visual cortex of mammalian brains, therefore they are biologically inspired. Steerable filters are Gaussian derivative filters, which are specific to particular orientation and frequency. Figure 2.12 shows one example of steerable filters.

The filters also can be learned in a supervised or an unsupervised way, e.g. with Convolutional Neural Networks (CNN) [LeCun et al., 1998], where the filters are considered as the connections between the layers. The learned filters in the CNN can, however, often similar to Gabor-like filters, as shown in figure 2.13. Section 2.4 is devoted to this direction of automatic feature

learning.

2.3 Visual recognition: methods

In section 2.2, we discussed the popular global and local features for visual recognition in the literature. This section focuses on models and methods. A large number of approaches have been proposed for the recognition stage. In this section, we restrict our scope to three families of techniques, which covers a large part of the state of the art: matching, sliding windows, and segmentation followed by classification, which are shown in figure 2.1.

The research on visual recognition is strongly inspired by research in machine learning, especially learning machines for classification such as support vector machines (SVMs) and randomized decision forests (RDF) etc. Object models can be learned efficiently and effectively, if the representation is based on vectorial global features. However, they are not suited to local representations based on local features, since local features are inherently structural and not capable of describing the global characters of the object in a simple numerical(vectorial) way, unless some grouping techniques are employed on the local features such as Bag-of-Words (BoW) models. On the other hand, there are other techniques, for instance, recognition based on matching, which are capable of directly dealing with structural data.

2.3.1 Matching

Matching is a flexible and powerful technique, which has proved to be effective in many computer vision tasks. The philosophy of matching is that a decision is made for a new instance according to its similarity to a pattern. The pattern can be a training instance, or a learned model/prototype. Classification is frequently done through the KNN classifier working on distances defined on the matching results. It is able to obtain comparable results on a general dataset, where machine learning techniques also work well, but it can also handle with a special case — one-shot learning problem, i.e. only one example is available in the training set. An important advantage is the possibility of defining distances on non-vectorial/structural data.

2.3.1.1 Global matching

Global matching is one type of matching schemes, where the similarity between two images is computed globally without local neighborhood correspondence problem. The distance between two images indicates the similarity: the less distance and the more similarity.

Intensity matching — The initial idea is to sum all the gray-scale differences of pairs of pixels from the same locations in two images:

$$D(I_x, I_y) = \sum_{i=1}^{N} |x_i - y_i|^2 = ||\mathbf{X} - \mathbf{Y}||_p \tag{2.15}$$

where I_x, I_y are two images, x_i, y_i are gray-scale values for pixel i, $\mathbf{X}, \mathbf{Y}$ are vectorized pixel values, p is the norm (e.g. $p = 2$ is Euclidean distance).

However, matching based on the pixel values rarely works well due to that the pixel values contain little information about the object. Three distance metrics are also proposed for object matching in the following.

Feature matching — In principle, most global features described in section 2.2.1 can also be used for global matching in a straightforward way, if they are of vectorial nature. In this case a distance can be defined, e.g. through any normed difference (e.g. $L2$-norm).

Chamfer matching — It is widely used for edge based object recognition due to its tolerance to misalignment in scale and position. The main computation is the chamfer distance, which was first introduced by [Barrow et al., 1977] to compare the shapes of two collections of shape fragments. Given an image I, an edge map E is first computed for I. The basic chamfer distance of a template T at location x in the edge map is calculated by:

$$D_{cham}^{T,E}(x) = \frac{1}{|T|} \sum_{x_t \in T} \min_{x_e \in E} ||(x_t + x) - x_e||_2, \tag{2.16}$$

where $||.||_2$ is $L2$ norm and $|T|$ is the number of edge points in the template, x_e is an edge point in the edge map E. Equation 2.16 computes the mean of the distances between each point on the template to its closest point in the edge map. Equation 2.16 can be efficiently computed through a distance transformation (DT). For the edge map E, the distance transform is first performed to get a distance map, each value on each pixel gives the distance to its nearest edge:

$$DT_E(x) = \min_{x_e \in E} ||x - x_e||_2 \tag{2.17}$$

The distance between the template T and the image is the mean distance value over the template point locations in the distance map, showing as follows:

$$D_{cham}^{T,E}(x) = \frac{1}{|T|} \sum_{x_t \in T} DT_E(x_t + x). \tag{2.18}$$

Finally, the matching values usually are truncated to a threshold.

The original chamfer distance needs a good initialization for a template. Another drawback is that it suffers from cluttered background. One

modification is hierarchical chamfer matching [Borgefors, 1988], which performs coarse-to-fine chamfer matching. Shotton et al. [Shotton et al., 2008a] proposed oriented chamfer matching (OCM) to compute edge orientation mismatching between the template and the image. Integrated with original chamfer distance, a higher discriminative power is achieved. Ma et al. [Ma et al., 2010] significantly boost the performance of oriented chamfer matching to discriminate the false positive from the background clutter by comparing the matching score to normalizers, which are learned.

Hausdorff Distance — It is another distance metric widely used for image matching. For two sets of points $A = \{a_1, a_2, \cdots, a_p\}$ and $B = \{b_1, b_2, \cdots, b_q\}$, the Hausdorff distance [Huttenlocher et al., 1993] is defined as:

$$H(A, B) = \max(h(A, B), h(B, A)), \tag{2.19}$$

where

$$h(A, B) = \max_{a \in A} \min_{b \in B} ||a - b||_2 \tag{2.20}$$

$||.||_2$ is the $L2$ norm. The Hausdorff distance in essence is a max-min distance. The directed distance $h(A, B)$ gives the maximum distance from all the points in A to its nearest points in B. To make the distance symmetric, the maximum between two permutations is taken. It measures the degree of mismatch between two point sets. The authors in [Huttenlocher et al., 1993] also define several partial Hausdorff distances, which are based on the percentile rankings:

$$h_K(A, B) = K^{th}_{a \in A} h(A, B), \tag{2.21}$$

where $K^{th}_{a \in A}$ denotes the K^{th} ranked values of $h(A, B)$. Several extensions to the Hausdorff distance have been proposed in [Sim et al., 1999, Dubuisson and Jain, 1994]. A general Hausdorff distance from two sets of points to two sets of curves, so-called as Curve segment Hausdorff Distance (CsHD) is proposed to recognize shapes in [Yu and Leung, 2006].

Global matching techniques, such as the ones discussed above, define a measure criteria between an object and a template without solving a feature correspondence problem. They work well for rigid object, but are not suitable to non-rigid objects. Local matching is capable of calculating non-rigid object matchings, which is presented in the next subsection. Another possible way is to learn a deformable template for the object. This will be discussed in section 2.3.2.3 on deformable models.

2.3.1.2 Local matching

Local matching matches two collections of local features, which requires to solve a correspondence problem. The intuition is that an object is matched

to a template, if enough points (or other local primitives) extracted from the object in the scene are matched well to points on the model. The matching score of object matching can be related to the matching scores of all the local points. Another advantage is that local point matching inherently indicates the object's structure and position, which can also be integrated into the score.

One possible technique for solving the point correspondence problem is RANSAC (RANdom SAmpling AConsensus). It is an iterative method which estimates parameters of a global and rigid model from observed data, which may contain outliers (e.g. affine, homography etc.). The parameterized model can be fitted to the observation through inlier correspondence. However, this method can not handle non-rigid transformations of an object. Graph matching thus is an alternative method, which is very suitable to non-rigid transformations.

The object can be represented as a graph, which consists of a set of vertices, and a set of edges between the vertices. Each vertex is often an interest point (please refer section 2.2.2.1), region of interest, etc. The vertices in the graph are connected in several ways:

- Full connection: Every pair of nodes in the graph is connected through an edge. This often is not the optimal choice for large graphs, as the edge number grows exponentially, and edges do not code any information.
- Proximity: The neighbor nodes are connected according to their distance in the image or video.
- Adjacency: The edges are estimated from the adjacency relations of the vertices.

Features can be extracted from each vertex and used to measure the similarity of two nodes. Given two feature vectors F_i and F_j, different metrics are adopted:

- $L-$norm: $D(F_i, F_j) = ||F_i - F_j||_p$, amongst Euclidean distance is widely used.
- χ^2 test statistic distance: This distance is a natural choice for two histogram-type features such as HOG, SIFT:

$$D(F_i, F_j) = \frac{1}{2} \sum_{k=1}^{K} \frac{[F_i(k) - F_j(k)]^2}{F_i(k) + F_j(k)}. \tag{2.22}$$

Here we briefly review the philosophy of graph matching. Let $G^m = (\mathcal{V}^m, \mathcal{E}^m, F^m)$ and $G^s = (\mathcal{V}^s, \mathcal{E}^s, F^s)$ be two graphs extracted from the model

and scene, respectively, where $\mathcal{V}$ is a set of vertices, $\mathcal{E}$ is the edges between the vertices, and F is the unary measurement, i.e. descriptor associated with each vertex (please refer to section 2.2.2.2). The number of the nodes are respectively denoted as $N^m = |\mathcal{V}^m|$ and $N^s = |\mathcal{V}^s|$. The aim of graph matching is to determine a mapping x_i from the vertex i in the model to a vertex in the scene.

The two most frequent formulations for graph matching are the following:

1. Exact matching searches for a structure preserving correspondence. Examples are graph isomorphism and sub-graph isomorphism.

2. Inexact matching minimizes an energy function which models the quality of the matching, often taking into account geometric and appearance information associated with the graphs.

Here we focus on the second type only, as it is arguably more frequent in computer vision. The energy functions can be expressed in two ways:

Summation form — The mapping cost is expressed as the summation of each correspondence cost, defined as:

$$E(x) = \lambda_1 \sum_{i \in \mathcal{V}} \psi_1(i, x_i) + \lambda_2 \sum_{ij \in \mathcal{E}} \psi_2(i, j, x_i, x_j) \tag{2.23}$$

which computes the energy for the potential mapping of two graphs. A node $i \in \{1, \cdots, N^m\}$ in the model is assigned to a node $x_i \in \{1, \cdots, N^s\}$ in the scene. Noted that in equation (2.23) two functions are defined: ψ_1 is the unary potential function, which measures the distance between the features of node i in the model and its corresponding node x_i. ψ_2 is a pairwise potential function, defining the compatibility of pairs of nodes (i.e. edges) in the model and its corresponding edges.

The graph matching problem can be reformulated to minimize the energy function expressed in equation (2.23):

$$\hat{x} = \arg\min_x E(x) \tag{2.24}$$

In above equation, x is the only variable to minimize, and also x should be a discrete value in $\{1, 2, \cdots, N^s\}$. The minimization in equation (2.24) in general is a NP-hard problem [Torresani et al., 2008].

Matrix form — Equation (2.24) can also be formulated in a matrix form: The mapping variable x is reshaped to a $N^m \times N^s$ assignment matrix X, in which X_{ij} is set to 1 if node v_i^m in the model is mapped to v_j^s in the scene. $X \in \{0, 1\}^{N^m \times N^s}$ is further reshaped to vector $x = \{0, 1\}^{N^s N^m}$, which is the row-wise vectorized replica of X. An affinity matrix A encodes cost

$\psi_1(i;j)$ from equation (2.23) in its diagonal elements and cost $\psi_2(i,j,x_i,x_j)$ from equation (2.23) in its off-diagonal elements. The optimal solution for assignment is achieved by minimizing:

$$E(x) = x^T A x, \tag{2.25}$$

One constraint can be imposed on X to ensure one-to-one correspondences such that each row contain at most a single 1.

The solutions to solve equation (2.24) and (2.25) are beyond the scope of this chapter. They are mostly based on discrete optimization (relaxation, Lagrangian algorithms, graph cuts, local/tabu search etc.) or a combination of these, or on continuous optimization techniques applied to a relaxed version of the problem [Torresani et al., 2008, Duchenne et al., 2009, Marius et al., 2011, Lee et al., 2011, Lin et al., 2009].

2.3.2 Sliding windows

In section 2.3.1, matching techniques have been discussed for visual recognition. One advantage of these methods is that matching can tell us more information about an object, not only the presence of the object in the image, but also its location i.e. where it occurs in the image. However, it suffers from heavy computation burden, since it requires to solve a discrete optimization problem.

In this section, an alternative framework — the sliding window technique, is discussed. This system slides a window over an image. For each window, a classifier determines whether an object is present at this location. The window's position is assigned to be the object's location. Machine learning tools can be applied to classification, which can be rapidly executed in the test stage. Sliding window technique is a trade-off method, which performs object detection task in image/video. In the following, bag-of-words models (BoWs) and deformable part based models (DPMs) are respectively reviewed, since they can be easily employed in each window.

2.3.2.1 Based on global features

In principle, any global features (described in section 2.2.1) can be combined with the sliding window technique and any learning machines can be adopted for object detection and recognition. In particular, appearance and shape features are widely used. The feature extractor can also be automatically learned, which is discussed in a dedicated section 2.4.

2.3.2.2 Bag-of-Words models

The *bag-of-words* (BoW) model has been widely used in computer vision [Csurka et al., 2004, Dollar et al., 2005] in last decade. This popular model originates in natural language processing, where it is used to analyze the content of documents. The model is global in nature, since a single vector describes the content. However, this content is obtained by pooling descriptors from local primitives. Each document is expressed as a histogram of frequencies of orderless words, which are taken from a dictionary. Similarly, an image or video can also be regarded as a document. However, BoW models can not be directly employed in this case, since dictionaries about objects are not available. In order to employ the BoW model in computer vision applications, a dictionary must be built, and each word in the dictionary, named as visual concept, describes local characteristics of the object.

BoW models have been designed to model whole documents. In principle, they can also be applied to sliding windows, provided the window contains a sufficiently large amount of local primitives (e.g. key points) to allow calculating reliable statistics (histograms). In the video case (activity recognition), the 3D window can often cover the whole image and several frames, and is shifted temporally over the video.

Visual concepts traditionally are constructed by extracting features from the local primitives like interest points, patches, regions, etc (refer to section 2.2.2). Each visual primitive can not serve as a codeword due to a large amount of visual primitives in image/video. Therefore, further special procedure should be performed. The codebook is constructed offline. A collection of feature vectors are extracted from the training set, and a clustering technique such as k-means is applied on the feature vectors. The obtained cluster centers are viewed as the codewords. Thus, codewords (visual concepts) are defined as frequently occurring visual patterns. A set of codewords produces a codebook. The codebook is stored and can be retrieved in the training/test stage. The procedure of BoW models (refer to figure 1.7(a) in chapter 1) is given as follows:

- The training stage comprises two parts:
    1. The unsupervised clustering part, which calculates the dictionary. Clustering techniques are applied on the extracted feature vectors, and the cluster centers are regarded as codewords.
    2. The supervised training part calculates the prediction model. For each training image, each extracted feature vector is projected to the codeword with nearest distance in the codebook. A histogram, where each bin counts frequencies of each codeword occurring in

the image, serves as an indicator of the content. Each training image is represented as a histogram. A set of histograms and their associated class labels are used to learn the prediction model of a learning machine such as SVM, NN, etc.

- In the test stage, for a new test image, a similar procedure is performed, without codebook construction. A histogram is constructed from all the feature vectors to represent the test image, which is eventually fed into a learning machine for classification.

BoW models have several advantages: (i) they are very simple and easy to implement; (ii) they are robust to background clutter and occlusion in the image; (iii) they are also invariant to viewpoint changes.

1. **BoW without structural information**

In classical BoW models, an image is represented as a histogram of occurrences of codewords, which are independently extracted. The histogram contains no information of the location of each point of interest, and no information about the relationships between the points. Therefore, the classical BoW model is a model without any structural information. They mainly differ from each other in the way how features and codebooks are obtained.

Sivic and Zisserman [Sivic and Zisserman, 2003] and Csurka et al. [Csurka et al., 2004] first introduce "bag-of-features" models into computer vision community, which have the same meaning as "bag-of-words" models. They use SIFT features to construct codebooks. They have been successfully used for object retrieval problems and visual categorization problems. A texture vocabulary (i.e. texton) [Leung and Malik, 2001] is learned to recognize the texture objects. Matikainen et al. [Matikainen et al., 2009] learn a trajectons vocabulary based on trajectory snippets of tracked features. Each action is represented as a histogram of trajectons. A mid-level sparse visual vocabulary is learned in [Boureau et al., 2010].

The BoW models have also been extended to the video case for activity recognition. Visual concepts are detected around space-time interest points [Laptev, 2005, Dollar et al., 2005]. Semantic visual codebook by using diffusion distance has been proposed for action recognition [Liu et al., 2009].

2. **BoW without structural information – optimized models**

Traditional codebooks are constructed by classical k-means clustering. There have been several attempts to optimize the codebook. Jurie and Triggs [Jurie and Triggs, 2005] show that cluster centers by k-means are exclusively

around dense regions in feature space. They propose scalable acceptance-radius based clusters to capture non-uniform feature distributions. Ballan et al. [Ballan et al., 2009] extend this work to human action recognition. Moosmann et al. [Moosmann et al., 2007] learn a discriminative visual codebook by means of randomized clustering forests.

Liu and Shah [Liu and Shah, 2008] learn an optimal codebook by information maximization. Considering two random variables $X = \{x_1, x_2, \ldots, x_n\}$ and $Y = \{y_1, y_2, \ldots, y_m\}$, X indicates codewords and Y indicates videos. Their goal is to find an optimal codebook $\hat{X}$ with maximum mutual information given the initial codebook X :

$$I(\hat{X}; Y) = \sum_{y \in Y, x \in \hat{X}} p(x, y) \log \frac{p(x, y)}{p(x)p(y)}. \tag{2.26}$$

where $p(x, y)$ is the joint distribution of $\hat{X}$ and Y, $p(x)$ and $p(y)$ are probability distributions of $\hat{X}$ and Y respectively. The function is defined as follows:

$$\max(I(\hat{X}; Y) - \lambda^{-1} I(\hat{X}; X)) \tag{2.27}$$

Here, $I(\hat{X}, X)$ measures the compactness of the new codebook $\hat{X}$ compared to X, which can be computed by a similar equation to (2.26) given the mapping $p(\hat{x}|x)$. λ^{-1} is a Lagrange multiplier.

Equation (2.27) is solved by a greedy bottom-up pairwise merging procedure. The cost of merging two codewords $\hat{x}_1$ and $\hat{x}_2$ is expressed as:

$$\triangle I(\hat{x}_1, \hat{x}_2) = I(\hat{X}_{bef}; Y) - I(\hat{X}_{aft}; Y) \tag{2.28}$$

where $I(\hat{X}_{bef}; Y)$ and $I(\hat{X}_{aft}; Y)$ denote the mutual information before and after merging. Two codewords with minimum loss of equation (2.28) are merged at each step. The merging operation is continued until $\triangle I(\hat{x}_1, \hat{x}_2)$ is larger than a predefined threshold or numer of clusters is obtained.

The aforementioned approaches of codebook optimization proceed by unsupervised learning on the feature space, they are not oriented to classification stage. On the contrast, our codebook optimization by an end-to-end framework introduced in chapter 3 is based on supervised learning, which makes it possible to use the backpropagated errors from the classifier to update the codewords. Goh et al. [Goh et al., 2012] propose to learn a visual codebook by the framework of Restricted Boltzmann Machines (RBMs) (see section 2.4.2.2) in a supervised way for image category classification.

3. **BoW with structural information**

The BoW model based on interest points usually ignores the spatial (or geometric) information, i.e. the spatial or spatial-temporal relationships between the interest points. These relationships contain many important complementary clues for recognition. The codewords in BoW models are orderless. Other work attempts to introduce structural information into BoW models.

Grauman and Darrell [Grauman and Darrell, 2005] propose a pyramid match kernel, which maps unordered features to multi-resolution histograms and implicitly finds correspondences on the finest resolution. Lazebnik and Schmid [Lazebnik and Schmid, 2006] partition the image into increasingly fine sub-regions and compute a BoW model for each sub-region, and use the pyramid kernel in [Grauman and Darrell, 2005] to train a SVM for recognition. Cao et al. [Cao et al., 2010] encode geometric information of objects into ordered BoW models. A histogram transformation is applied to get a spatial bag-of-features, which is tolerant to variations in translation, rotation, and scale.

The main distinction between video and image is that the video also contains temporal information, which describes continuous information of the action. Some work exploits encoding the spatio-temporal relationships into codewords to improve the BoW models for action recognition.

Oikonomopoulos et al. [Oikonomopoulos et al., 2009] take spatial co-occurrences of pairs of codewords into account, exploring the relative positions of codeword pairs with respect to the object's center to create a spatio-temporal model. Finally they propose a probabilistic spatio-temporal voting framework for localizing and recognizing human action. Matikainen et al. [Matikainen et al., 2010] express pairwise relationships with a sequence code map and estimate the relative location and temporal relationship probabilities from a training dataset. Ryoo and Aggarwal [Ryoo and Aggarwal, 2009] define the spatio-temporal relationship of pairs of neighboring local features such as near, far etc, creating a 3D histogram (two codeword dimensions and one relationship dimension). Ta et al. [Ta et al., 2010b] also propose pairwise features, which encode both appearance descriptors and spatio-temporal relationships between spatio-temporal interest points. In contrast to [Ryoo and Aggarwal, 2009], the structural information is discovered by clustering in a space-time geometric space.

Gaidon et al. [Gaidon et al., 2011] propose a new kernel which addresses the dynamics and temporal structure of actions on bags of words. They compute the Hilbert-Schmidt distance between two auto-correlations of actions and provide a kernelized formulation. In contrast to the previous works, their method does not align two videos to compute their similarity. Bettadapura et al. [Bettadapura et al., 2013] augment BoW models integrating temporal structure by quantizing time and defining temporal events for activity recogni-

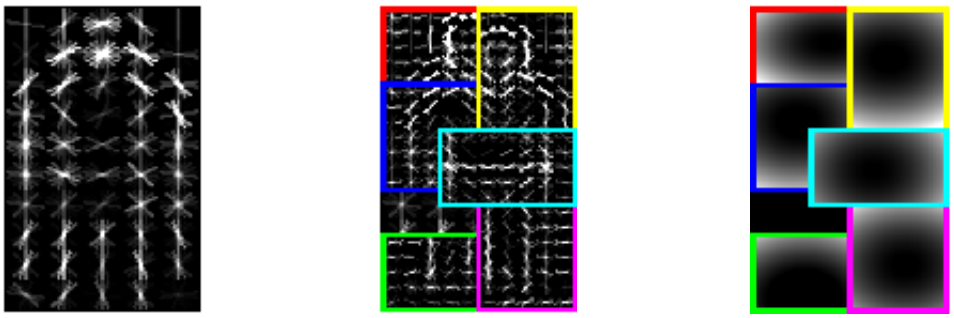
tion. Three n-grams encoding schemes are used to represent BoW models. To capture global structure, randomly regular expressions are created and treated as new words, adding into the original BoW models. Li et al. [Li et al., 2013] represent an activity as short-term segments, which are characterized by the dynamics of attributes (i.e. the score evolution of attribute detectors along the activity). They learn a dictionary of attribute dynamics. The activity is represented as bag of attribute dynamic words.

2.3.2.3 Deformable part based models (DPMs)

DPMs have first been proposed in [Fischler and Elschlager, 1973]. A recent version by Felzenszwalb et al. [Felzenszwalb et al., 2010] has achieved state-of-the-art results for object localization on many categories in the recently PASCAL VOC challenges. DPM is a kind of part based models. An object is represented as a set of meaningful parts and their associated spatial structure. Compared to low-level feature based models such as bag-of-words models, part based models, as a middle-level representation, are able to better express the structural properties of objects.

A large body of work has addressed these models. A part can be regarded as a constellation of low level features, such as interest points, edges etc. Fergus et al. [Fergus et al., 2003] propose a model of constellations of parts, which is expressed as a joint probability from all aspects of the object: appearance, shape, scale and occlusion. They first compute the local saliency for each location and scale in an image and the regions whose saliency is above a threshold are considered as parts for recognition. In the process of learning, the parameters for the probability are estimated by expectation-maximization. In the recognition step, a new image is recognized in a Bayesian manner. Fergus et al. [Fergus et al., 2004] extend their method by integrating curve segments into the framework: Each part can be either defined by a region or curve. Curve segments are obtained by first employing the canny detector and decomposing the curve into independent segments at bi-tangent points.

Parts also can be learned in a hierarchical structure, where low level features in child nodes are merged into a father node, as a part. Mikolajczyk et al. [Mikolajczyk et al., 2006] propose a hierarchical representation through codebooks learned from shared appearance features around edge points. A joint appearance-geometry probability distribution is learned for each cluster in the hierarchical tree. Recognition is done in a Bayesian way. Zhu et al. [Zhu et al., 2010] propose Recursive Compositional Models (RCMs) for multi-view multi-object detection, where each RCM is a probability distribution over a hierarchical graph corresponding to a specific object and viewpoint, which encodes the root appearance cues, the boundary potential corresponding for



(a) (b) (c)

Figure 2.14: A DPM for category "person": (a) the root filter, (b) the part filters, and (c) the gray scaled deformation cost.

the boundary segments at the leaf nodes, and also pairwise spatial relations between the parent node and their children nodes.

Pictorial structures [Fischler and Elschlager, 1973, Felzenszwalb and Huttenlocher, 2005] are another kind of part based models, where an object is modeled as a set of parts and their structure is captured by a set of "springs" connections in terms of Gaussian distribution between pairs of parts. The original pictorial structure [Fischler and Elschlager, 1973] is an energy based function, which is the sum of the mismatch cost of each part, and also deformation cost between pairs of parts. The best configuration of an object is the one with minimal energy. Felzenszwalb and Huttenlocher [Felzenszwalb and Huttenlocher, 2005] reformulate the pictorial structure in a statistical framework. The detection problem transfers to maximum a posterior (MAP) estimation. The parameters can be learned by using maximum likelihood estimation. In their star shaped model, only root/part relationships are included, pairwise part relationships are not modeled. This allows efficient inference through dynamic programming.

The aforementioned part based models are learned in a generative way. Felzenszwalb et al. [Felzenszwalb et al., 2010] attempt to learn deformable part based models in a discriminative way. Learning and testing are relatively efficient. The advantage of DPMs is that the parts are considered as latent variables in the learning stage and can be inferred efficiently without part annotations for each training image. We briefly review the latter formulation of DPMs as follows:

A DPM is composed of a root filter P_0 and a set of part filters $P_i, i \in (1, \ldots, n)$. Each part filter also defines a two-dimensional vector $a_i = \langle a_{ix}, a_{iy} \rangle$ specifying the anchor position relative to the root, and four quadratic deformation coefficients d_i taking into account the cost of the part relative to its anchor position. Figure 2.14 shows a learned DPM for category "person".

The score of a hypothesis in a window is the sum of the scores from the root filter and part filters, minus the deformation cost of each part real (i.e. inferred) location w.r.t the anchor position, and plus a bias b, shown as follows:

$$f(P_0, \ldots, P_n) = \sum_{i=0}^{n} P_i \cdot \phi(x, z_i) - \sum_{i=1}^{n} d_i \cdot \psi(z_i) + b. \tag{2.29}$$

where z_i is the inferred part location, and $\phi(x, z_i)$ are local dense features extracted from z_i, $\psi(z_i)$ is the displacement of the inferred position z_i w.r.t the anchor position a_i by:

$$\psi(z_i) = [(z_{ix} - a_{ix})^2, (z_{ix} - a_{ix}), (z_{iy} - a_{iy})^2, (z_{iy} - a_{iy})] \tag{2.30}$$

The real part locations, treated as latent variables, are inferred in the learning and test stage to maximize the score. Equation (2.29) can be rewritten as:

$$f(P_0, \ldots, P_n) = \max_{z \in Z(x)} (\sum_{i=0}^{n} P_i \cdot \phi(x, z_i) - \sum_{i=1}^{n} d_i \cdot \psi(z_i) + b). \tag{2.31}$$

where $Z(x)$ is a collection of all the possible positions for the parts. The objective function to minimize is the hinge loss, which is expressed by:

$$L(\beta) = \frac{1}{2}||\beta||^2 + C \sum_{i=1}^{N} \max(0, 1 - y_i f(x_i)) \tag{2.32}$$

where β is the set of learning parameters containing $(P_0, P_1, \cdots, P_n, d, b)$, x_i and y_i is the training example and its groundtruth, $f(x_i)$ is its score, C controls the regularization term, N is the training instance size. Latent SVM is applied to minimize equation (2.32). The training stage learns part filters and deformation costs. In the test stage, part locations are inferred. Note that the anchor positions are NOT learned in the supervised framework. Instead, they are set to locations where the root filter gives high responses. This might not be optimal, since parts do not necessarily look like the object.

Other follow-up work has be done to further improve DPM models by sharing parts between object classes [Ott and Everingham, 2011], through visual mixture models [Divvala et al., 2012] where the components for each class are defined by visual similarity instead of height-width ratio of the groundtruth bounding box, etc. DPMs have also been extended to spatio-temporal data for activity recognition [Tian et al., 2013]. DPMs are accelerated by replacing the convolution operations by locality-sensitive hashing, which enables to efficiently perform the detection of 100000 objects on a single machine

[Dean et al., 2013]. Sharma et al. [Sharma et al., 2013] discriminatively learn a collection of part templates in the images, and each part explains each image individually. The best set of part templates is chosen for each image. This is in contrast with other part based models which use all the parts for each image. At the same time, a large set of part templates also are able to encode the variance of humans in the dataset, avoiding highly structured model learning.

2.3.3 Segmentation for visual recognition

Segmentation is a fundamental problem in the community. It aims to segment an object into several meaningful parts, which is used to obtain more information about object, such as the spatial layout, or object pose. Segmentation plays several roles in visual recognition, which are addressed in the following.

2.3.3.1 Segmentation for selective candidate regions

The sliding window technique presented in section 2.3.2 is often applied to object detection in images/videos. However, it has several drawbacks: on one hand, it works best on objects with a fixed aspect ratio. For general objects, exhaustive search has to be performed over location, scale, aspect ratio, which is apparently computationally expensive. DPMs can be efficiently performed through generalized distance transformation [Felzenszwalb et al., 2010], but are still quite expensive.

Recently, an alternate way in favor of a segmentation based pre-precessing step has been proposed to overcome the drawbacks of sliding window techniques. The core is to first over-segment an image into thousands of object segments (i.e. regions) candidates, which cover most of the objects in the image. The regions are converted to bounding boxes and the features are extracted from the bounding box and then fed to a classification machine. Segmentation as Selective Search [van de Sande et al., 2011] is one of recent algorithms to generate regions/bounding boxes. They start with oversegmentation, a set of small regions are obtained. A hierarchical segmentation tree is constructed through grouping of similar regions in a greedy algorithm. Bounding boxes containing regions across the tree are considered for the sequent classification. Selective search enables to use more expensive features for each region and improves the state-of-the-art for 8 out of 20 classes on Pascal VOC 2007 dataset.

Wang et al. [Wang et al., 2013] apply the selective search approach [van de Sande et al., 2011] to get a set of small regions (named as "regionlets") in the bounding box. Several groups containing regionlets are organized

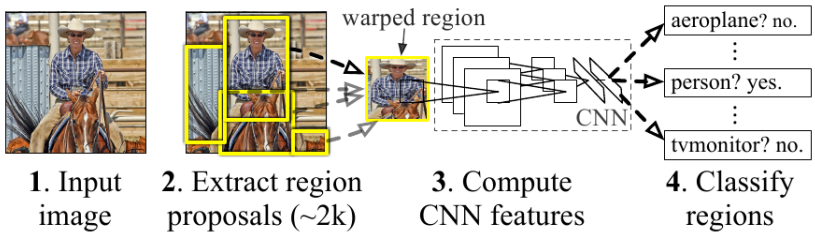


Figure 2.15: R-CNN: Regions with convolutional neural network features. This figure is reproduced from [Girshick et al., 2013].

(a) (b)

Figure 2.16: Illustration of segmentation in visual recognition. (a) part segmentation for pose estimation; (b) scene full labeling, this figure is reproduced from [Gould et al., 2009].

to capture the spatial layout of objects. The features are aggregated into a vector to tolerate deformations. A cascaded boosting classifier is trained to classify the bounding box candidates. Recent work by Girshick et al. [Girshick et al., 2013] obtains a large improvement in mAP (mean average precision) on PASCAL VOC 2011/12 datasets. They also apply the selective search [van de Sande et al., 2011] approach to generate a set of candidate bounding boxes. The content in each bounding box is cropped, fed into a convolutional neural network (see section 2.4) to extract features and eventually classified. Figure 2.15 shows their framework. They successfully combined feature learning (see section 2.4) into object detection and obtained very impressive results.

2.3.3.2 Segmentation as goal in itself

Segmentation sometimes can be cast as a classification problem, in which case, each pixel in an image is classified and assigned to a label. It is frequently used for some applications, for instance, human pose estimation. Part segmentation is a intuitive approach to obtain human parts, allowing to estimate joint positions from the segmentation.

The seminal work by Shotton et al. [Shotton et al., 2011] for human pose estimation in depth images utilizes this pipeline. They treat the segmentation

problem as a classification problem. For each image, background is subtracted and each pixel is annotated to one of N labels (N indicates the part number of human body). A randomized forest with N labels is learned. Each pixel in the test image is classified to one of N labels. Mean-shift with a weighted Gaussian kernel is applied finally to find the joint positions between the parts.

There is another visual recognition task — scene full labeling, where segmentation is needed. It attempts to label each pixel in the image with a category. This is a very challenging task due to its combination of segmentation and recognition. A common strategy is to pre-segment an image into super-pixels or other region candidates, and extract features from the candidates and finally make a classification. The difference here from selective search approach [van de Sande et al., 2011] is that we assign a label for each region, rather than some particular regions with high confidence to determine the bounding box of object. Socher et al. [Socher et al., 2011] propose a max-margin structure prediction based on recursive neural networks to label the image. The image is first over-segmented into small regions. Visual features are extracted from the regions. A neural network is trained to output a higher value when the neighboring region should be merged and also a label for the region. Farabet et al. [Farabet et al., 2012] adopt the same idea, but they differ from [Socher et al., 2011] in that: 1): the features are learned in the framework of convolutional neural networks, rather than hand-engineered features; 2) For a hierarchical segmentation tree, a classifier is learned to find a subset of nodes, which optimally covers the image and also maximizes the "purity" of the nodes, where purity is defined as a quantity that is inversely proportional to the entropy of the class distribution for each node.

It is also worth mentioning several recent work on scene full labeling. Pinheiro and Collobert [Pinheiro and Collobert, 2013] propose to apply recurrent convolutional neural networks to this task. The output of a ConvNet is fed to another ConvNet with tied parameters, which allows to model context and long range dependencies. Kontschieder et al. [Kontschieder et al., 2013] propose a new geodesic forest, achieving spatially consistent semantic image segmentation by encoding long range, soft connectivity features via generalized geodesic distance transforms. Shapovalov et al. [Shapovalov et al., 2013] propose spatial inference machines, taking into account mid-range and long-range dependencies for semantic segmentation of 3D point clouds. The framework proposed in [Farabet et al., 2012] has been extended to indoor scene segmentation with RGB-D images [Couprie et al., 2013].

Our work in chapters 4 and 5 focuses on part segmentation as a classification problem. We learn a classification machinery targeted to pixels, rather than regions, and employ it on each pixel in image to estimate parts.

2.4 Feature/Representation learning

In section 2.3, we briefly reviewed three kinds of recognition techniques. Different features can be adopted in these techniques. Generally speaking, feature extraction is an independent module compared to the following recognition module in the dominant pipeline shown in figure 1.6 in chapter 1. However, the presented extractors produce hand-crafted features (refer to section 2.2), which have the following drawbacks:

- Hand-crafted features usually are designed for specific properties such as HOG/HOF for appearance and shape context for shape.
- A certain amount of pre-processing needs to be performed to prepare for hand-crafted feature calculation. For examples, the gradient is computed first for each pixel and aggregated into a histogram of gradient orientation for a block; others require optical flow etc. However, pre-processing is mostly expensive and depends on numerous parameters which need to be tuned.
- In this classical framework, feature computation is independent from the following classification stage. Performance can be improved by learning the features as well as the classifier together, resulting in an end-to-end framework.

Given the aforementioned arguments on the hand-crafted features, feature learning techniques have received much consideration lately. Feature learning can be operated in different ways as shown in figure 2.17:

- Supervised learning: The labels of the training examples or the variables induced from the labels (e.g. sensitivity functions in a neural network) are directly used in the feature learning stage.
- Unsupervised learning: The labels of the training examples are not used for feature learning. Other criteria such as reconstruction error, or image transformations are adopted.
- Weakly-supervised learning: Compared to unsupervised learning, the labels of the training examples are known in the learning, but they are not associated with feature learning and no classification error is calculated. The availability of the labels is often necessary to generate new training examples (e.g. pairs of examples from the same labels or different labels).

In the following section, we briefly present common strategies used for feature learning.

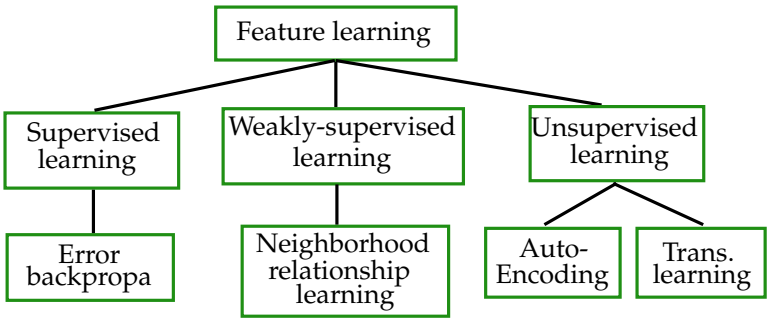


Figure 2.17: The feature learning criteria, where "backpropa" means back-propagation, "Trans" means transformation.

2.4.1 Supervised feature learning

We aim to learn a more discriminative feature extractor using annotated training images. An intuitive idea is to calculate classification errors for the training set and employ them to update the features through minimization, which is similar to train neural networks(i.e. error backpropagation).

2.4.1.1 Error backpropagation

Earlier work on supervised feature learning is LeCun's work on handwritten document recognition [LeCun et al., 1998]. They adopted a framework called Convolutional Neural Networks (CNN or ConvNets) shown in figure 2.18. The features are directly computed from the raw data by convolving an image with a set of filters, resulting in a set of feature maps. Non-linear transformation (e.g. hyperbolic tangent function) and spatial reduction/pooling are applied to the feature maps to make the output robust to noise and slight translations. Similar operations can be repeated several times by treating the output of previous layers as the input of next layer. The resulting feature maps are vectorized into a vector, and then fed to a fully connected logistic regression layer (or to a MLP) to predict the category of the input. This framework is an end-to-end framework because the feature extraction module is connected with the classification module. This means that the error is back propagated all the way back to the input, updating the weights in the logistic regression layer as well as the filters.

The learning procedure is described as follows: The filters are first randomly initialized. Given an input and its associated groundtruth, the network is stimulated and the output layer delivers an response for each unit. A loss function (for example sum of errors or cross-entropy loss) is calculated; then a gradient of the loss function w.r.t each output unit is computed, which is

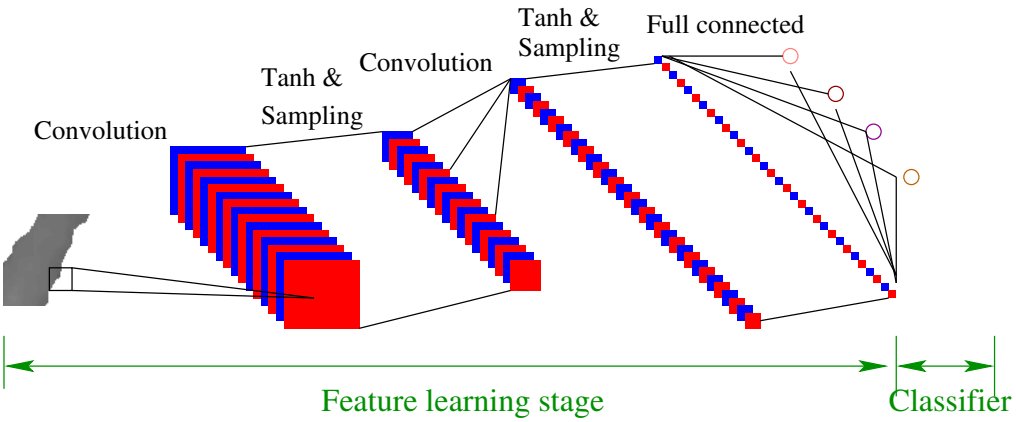
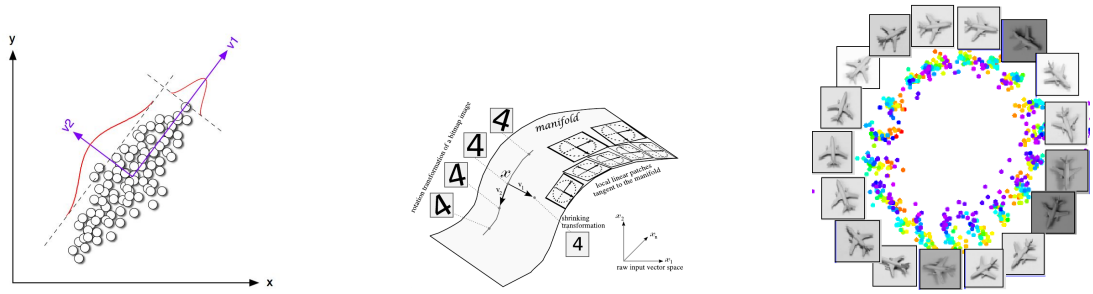


Figure 2.18: The convolutional neural network.

.



(a) (b) (c)

Figure 2.19: Dimensionality reduction. (a): PCA linear embedding; (b)non-linear embedding, this figure is reproduced from [Bengio, 2009]; (c) DrLIM results, this figure is reproduced from [Hadsell et al., 2006].

used to update the weights in the classification layer. This gradient is again backpropagated to the previous layer, and the sensitivity is used to update the filters. This is the classical error backpropagation, which can be efficiently implemented as only inner products are involved in the learning procedure. After two decades of development, CNN has been used in many applications [LeCun et al., 2010] and has won important competitions such as ImageNet 2012 [Krizhevsky et al., 2012].

2.4.2 Unsupervised learning

Unsupervised feature learning casts the task as an intelligent dimensionality reduction problem. The data are passed from a high dimensional input space (pixel) to a lower dimensional feature space. Classical methods like principle component analysis (PCA) perform this task. PCA method embeds the input

to the eigenspace defined by the eigenvector of the input feature matrix. But it suffers from its main limitation –linearity, i.e. only linear embedding (see figure 2.19(a)). Intelligent dimensionality reduction attempts to find non-linear structure in the data and to encode it in a suitable non-linear transformation (see figure 2.19(b)).

In this section, we will present two unsupervised feature learning paradigms: auto-encoding and transformation learning. Auto-encoding tries to preserve as much as information about the input from the representation, while the latter tries to learn a transformation between neighboring (or consecutive) images, which can be very useful to capture the temporal relationship between frames in video.

2.4.2.1 Auto-encoding

Auto-encoding is based on the following principle: an encoder encodes the input data into a compact code, which is smaller than the input data. A decoder tries to reconstruct the data with minimum error. As the code is small, the encoder and decoder need to find patterns and structures in the data in order to be efficient. This structure is postulated to be useful for recognition.

In more detail, let x be the raw data, an encoder first maps x to a representation code z through a non-linear activation function f, typically a sigmoid function:

$$z = Enc(x, W_e) = f(W_e * x + b_e) \tag{2.33}$$

where W_e are the parameters of the encoder. A decoder reconstructs the input to be y from z through another activation function g, typically an identity function (i.e. linear case) or sigmoid function:

$$y = Dec(z, W_d) = g(W_d * z + b_d) \tag{2.34}$$

where W_d are the parameters of the decoder. In general, for a set of training examples, the loss function is expressed by sum of error or cross-entropy:

$$L = \begin{cases} \sum_{i=1}^{N} ||x_i - y_i||_2 & \text{if linear case} \\ \sum_{i=1}^{N} x_i \log(y_i) + (1 - x_i) \log(1 - y_i) & \text{if sigmoid case} \end{cases} \tag{2.35}$$

The learning procedure attempts to minimize the loss function over the parameters W_e and W_d on the training set. Gradient descent can be applied to minimize the error. Different training algorithms have been developed according to different regularization items added into equation (2.35): regularized auto-encoder with weight decay [Lee et al., 2007], denoising auto-encoder [Vincent et al., 2010], contractive auto-encoder [Rifai et al., 2011].

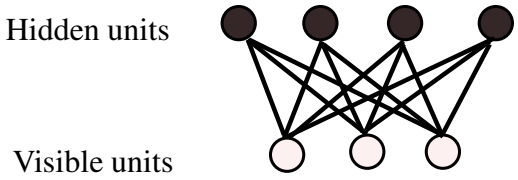


Figure 2.20: The Restricted Boltzmann Machine architecture [Hinton et al., 2006].

Ranzato et al. [Ranzato et al., 2006] propose another auto-encoder training algorithm based on an energy function, including sparsifying term. It consists of two items:

$$E = \sum_{i=1}^{N}(E_C + E_D) = \sum_{i=1}^{N} \frac{1}{2}||z - Enc(x, W_e)||^2 + \frac{1}{2}||x - Dec(\hat{z}, W_d)||^2 \quad (2.36)$$

where E_C is the prediction error between the encoder output code and the optimal code. E_D is the reconstruction error using the sparsified code $\hat{z}$. The training procedure not only learns to minimize the reconstruction error, but also makes the predicted code as much as close to the optimal code. This algorithm simplifies the code optimization procedure and reconstruction problem.

2.4.2.2 Models for unsupervised feature/representation learning

The framework of CovNets can also be used for unsupervised learning [Ranzato et al., 2007]. In the auto-encoder, W_e can be specified as a stack of filters, composed of a non-linear embedding function. The feature maps are the output codes. Similarly, W_d is also a set of filters to reconstruct the input.

Besides ConvNets, Restricted Boltzmann Machines (RBMs) can be regarded as a special "encoder-decoder" model. They have been initially used to learn the model distribution of binary images (i.e. only two states for each pixel), and several extensions are made to make it applicable for real valued data. RBM is a type of graphical model, which has two set of nodes: visible nodes and hidden nodes. Each visible node connects to each hidden node and vice-versa. There are no intra-visible or intra-hidden connections. Figure 2.20 shows a simple RBM architecture. The energy of a RBM is expressed by:

$$E(\mathbf{v}, \mathbf{h}) = -\sum_{i=1}^{V}\sum_{j=1}^{H} v_i h_j w_{ij} - \sum_{i=1}^{V} v_i b_i^v - \sum_{j=1}^{H} h_j b_j^h \quad (2.37)$$

where $\mathbf{v}$ and $\mathbf{h}$ are binary state vectors for visible nodes and hidden nodes. v_i is the state of visible node i, and h_j is the state of hidden node j, b_i^v and b_j^h are the biases for each visible node and hidden node, w_{ij} are the weights between the visible nodes and hidden nodes. Intuitively speaking, the weights store information producing invisible state estimations from observed data, which can be reconstructed from the binary hidden states. The learning criteria maximizes the log probability of the training set under the model's distribution:

$$\sum_{i=1}^{N} \log p(\mathbf{v}^c) = \sum_{i=1}^{N} \log \sum_{\mathbf{h}} p(\mathbf{v}, \mathbf{h}) = \sum_{i=1}^{N} \log \frac{\sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}}{\sum_{\mathbf{v}} \sum_{\mathbf{h}} e^{-E(\mathbf{v},\mathbf{h})}} \tag{2.38}$$

The learning algorithm to maximize equation (2.38) can be formated as:

$$w_{ij} = \varepsilon_w(E_{\text{data}}[v_i h_j] - E_{\text{model}}[v_i h_j]). \tag{2.39}$$

where E_{data} is the expectation when the visible nodes are clamped to the data, which is easy to computed, while E_{model} is the expectation when the visible nodes are unclamped. The approximation can be obtained through Markov Chain Monte Carlo (MCMC) to get the reconstruction given the hidden units. Contrastive divergence, which performs one step of MCMC sampling in inference [Hinton, 1995], is an efficient algorithm to train a RBM.

A stack of auto-encoders [Vincent et al., 2010] and RBMs [Hinton et al., 2006] (called "deep belief nets") can be constructed to learn high level features, where the output of the current level can be fed as the input of next level. It is trained layer by layer, i.e. the weights of the current layer are fixed when training the next layer. When all the layers are learned, another round of supervised learning is performed to fine-tune all the weights. Here unsupervised learning helps the parameters locate at a good start-point and further supervised learning can be continued to improve the performance. RBMs have been extended to temporal data by Taylor et al. [Taylor et al., 2011a]. In these conditional RBMs, the current hidden and visible units are conditioned by past observations.

2.4.2.3 Transformation learning

Another aspect of unsupervised learning is to learn an image transformation given pairs of images. This can be very useful in various applications involving dynamic context, for instance to capture the motion information between subsequent frames in the video.

Memisevic and Hinton [Memisevic and Hinton, 2007] propose gated Boltzmann machines. The input is the current image (or patch), and the output is

the next image (or patch). In order to capture the relations between the input and the output, a hidden binary variable is introduced to connect the input and output. The energy function as shown in equation (2.37) is changed to associate with the input, output and hidden variable. The weights are also changed to three-way tensor variables.

Taylor et al. [Taylor et al., 2010] extend this work to convolutional gated Restricted Boltzmann Machines (convGRBMs) and use them to learn spatio-temporal features in video. Their method is able to learn the "flow fields" between pairs of input frames. They apply their model for action recognition, and obtain competitive performance on benchmark datasets.

2.4.3 Weakly-supervised learning

Weakly-supervised learning is a type of learning, which does not use label information in the feature learning stage in a direct way, i.e. classification error is not directly minimized. Instead this label information can provide some other information such as information on neighborhood relationships. We refer this kind of information as neighborhood relationship, which is defined either by manual labels (i.e. the instances with same label are neighbors) or the distances between the input data. The neighboring instances from the same category in the input space should generate the neighboring outputs in the output space. These methods try to learn a mapping function from high-dimensional input data to low-dimensional output, at the same time, similar inputs are mapped to close points and dissimilar inputs to distant points in the low-dimensional space.

Neighborhood Components Analysis (NCA) [Goldberger et al., 2004] and its variants [Salakhutdinov and Hinton, 2007] implicitly learn a mapping function from high dimensional space to low dimensional space while preserving the neighbourhood relationship. However, NCA is optimized for nearest neighbor classification. Dimensionality Reduction by Learning an Invariant Mapping (DrLIM) proposed by Hadsell et al. [Hadsell et al., 2006] is an online, non-probabilistic energy based method which explicitly learns a non-linear invariant embedding function. The framework is a tying *siamese* architecture. The training thus works on pairs, embedding the pair of instances with same label to close outputs and pair of instances with different labels to distant outputs. But testing still works on single input and gives a feature mapping. The energy on N pairs of instances is defined by:

$$\mathcal{L}(W) = \sum_{i=1}^{N} L(W, (Y, X_1, X_2)^i), \tag{2.40}$$

where W is the learned coefficients, Y indicates the status of the pair (X_1, X_2),

i.e. if they are from the same label, $Y = 0$, otherwise 1. The energy from a pair therefore is:

$$L(W, (Y, X_1, X_2)^i) = (1 - Y)L_S(D_W^i) + Y L_D(D_W^i) \tag{2.41}$$

$D_W^i = ||G_W(X_1) - G_W(X_2)||_2$ is the distance after mapping. L_S and L_D are the loss for similar pairs and dissimilar pairs, having the following forms:

$$L_S(D_W^i) = \frac{1}{2}(D_W)^2, \tag{2.42}$$

$$L_D(D_W^i) = \frac{1}{2}(\max(0, m - D_W))^2 \tag{2.43}$$

where m is a margin. L_S would push together similar instances, and L_D would pull apart the dissimilar instances if they were close enough. The mapping function G_W is formulated as a convolutional neural network (refer to figure 2.18) in their framework. It is trained by error backpropagation. Figure 2.19c shows the results learned by DrLIM. Taylor et al. [Taylor et al., 2011b] extend their work to soft similarity between pairs of input.

Our work on spatial pre-training presented in chapter 4.4.1 is very similar to DrLIM, in that both employ the *siamese* architecture. But ours differs from DrLIM in that: (i) we aim to perform pixelwise feature embedding, while DrLIM is operated on the whole image. (ii) There is no structural information on labels in DrLIM, i.e. similar pairs and dissimilar pairs, while we discriminate the pairs of pixels according to their spatial label layout: pairs of pixels with same label, pairs of pixels with neighboring labels, pairs of pixels with non-neighboring labels. The energy function 2.41 is also adapted.

2.5 Conclusion

In this chapter, we presented the state-of-the-art of visual recognition. We discussed the common used feature types, three families of recognition techniques, i.e. matching, sliding window technique, and segmentation followed by classification. We also reviewed feature learning techniques for recognition.

In next chapter, we will present our first contribution – supervised end-to-end learning for bag-of-words models. The codebook used for BoW models is optimized with the classifier training in an end-to-end framework. *A priori* information of label spatial configuration in feature space is integrated into optimize the codebook. We employ this approach on human action recognition.

CHAPTER 3

Supervised end-to-end learning of bag-of-words models

Contents

3.1 Introduction

The *bag-of-words* (BoW) model has been widely used in computer vision applications such as object class [Csurka et al., 2004, Lowe, 2004] or human action recognition [Dollar et al., 2005]. In these applications the objective is to recognize high level information (objects, actions) from a large quantity of low level data, e.g. image or video pixels. BoW has proved to be an efficient representation in this context. This popular model was first introduced in natural language processing, in which each document is expressed as a histogram of frequencies of orderless words. In order to employ the BoW model in computer vision applications, an image or a video is treated as a document, which can be considered as a collection of interesting local events often called

visual concepts [Csurka et al., 2004]. Information on the presence of these visual concepts, i.e. whether each one of them is present or not, and with which frequency, serves as an indicator of the contents.

Visual concepts can be generated in different ways, usually through the extraction of discriminant and invariant descriptors (features) around local primitives like interest points, patches, regions, edges etc., followed by clustering in order to identify clusters in feature space of descriptors. The thus obtained clusters are considered as visual concepts, or visual codewords. A set of such visual codewords produces a visual codebook.

Traditionally, a visual codebook is learned by unsupervised clustering or vector quantization of feature vectors extracted from the local primitives in the image or video, often with algorithms such as k-means [Csurka et al., 2004] or random forests [Moosmann et al., 2007]. The BoW for a new image or video is calculated in a similar way: extraction of descriptors on local primitives, projection of the descriptors on the codebook precalculated on a training set, calculation of a histogram of the occurrences of each codeword of the codebook. An image or video is classified passing the BoW model to any learning machine, for instance a support vector machine (SVM) or a neural network (NN).

The traditional method of codebook creation through unsupervised clustering ignores the class labels of the feature vectors in the training set. Of course the labels are used for classification of the BoW models, but not for the creation of the codebook. As a consequence, the visual codebook is less discriminative. In section 2.3.2.2 of chapter 2 we gave a short state of the art on BoW models and their extensions. We here refine this state of the art and briefly put into the context the work relevant to this chapter.

In order to create a more discriminative codebook, several attempts have been made to reveal the semantic relations between the codewords of k-means. In the work of Liu and Shah [Liu and Shah, 2008], they iteratively obtain an optimal and compact codebook via maximal mutual information. At each iteration they merge two clusters which have minimum loss in mutual information. The iterative procedure continues until a threshold of maximal mutual information or minimum cluster number is achieved. The optimal codebook size is found by unsupervised learning. In [Liu et al., 2009], Liu uses a diffusion map to embed a mid-level codebook into a semantic codebook. However, it is not appropriate to measure semantic distances using diffusion distances. In recent work, Saghafi [Saghafi et al., 2010] proposes a concept space to illustrate the semantic relations between the visual codewords. They apply generative models such as latent semantic analysis (LSA) and probabilistic latent semantic analysis (pLSA) to discover the latent semantic relations between the initial codewords. In contrast to the unsupervised pLSA learning

framework in which the number of latent topics is equal to the number of classes [Niebles et al., 2008], the number of topics is variable in this method. In all methods mentioned above, the codebooks are created by unsupervised methods, and the label information of feature vectors is ignored in the codebook creation. Intuitively, the discriminative power of the codewords could be increased by learning using the label information.

In contrast to unsupervised codebook learning, we propose a supervised learning and codebook optimization framework in this chapter. The whole sequence, codebook creation and class learning, is formulated as an artificial neural network, and the error gradient information is used to update the codebook cluster centers as well as the classical multilayer perceptron (MLP) weights for class recognition.

Our work is very close to the recent work by Goh et al. [Goh et al., 2012]. Both adopt an end-to-end framework to learn a discriminative visual codebook for BoW models in a supervised way. However, unlike our framework which integrates BoW model construction and codebook learning together, they instead learn a visual dictionary in an end-to-end framework, and then train another classifier (i.e SVM) for classification. In their codebook learning stage, they resort to RBMs (see 2.4.2.2) to first pre-train the codebook in an unsupervised way, and to fine-tune the codebook for each input feature vector associated with the image category label by error backpropagation in a supervised way. In contrast, we build a BoW model for each input entity in our framework.

The learning and optimization framework we present in this work is well suited for any application for which bag of words models can be successfully used. We restrict ourselves here to human action recognition in videos, for which BoW models and extensions are widely used [Csurka et al., 2004, Niebles et al., 2008, Gilbert et al., 2011, Laptev, 2005, Schuldt et al., 2004]. The improvements we propose make the codebook more discriminative, and therefore are likely to improve many of the existing extensions of the basic BoW model, such as, for instance, correlograms [Liu and Shah, 2008], topic models [Niebles et al., 2008], local grouping and compound features [Gilbert et al., 2011], spatial co-occurrences of pairs of features [Oikonomopoulos et al., 2009, Ryoo and Aggarwal, 2009, Ta et al., 2010b] and parts based models [Mikolajczyk and Uemura, 2011]. To the best of our knowledge, this is the first attempt to combine codebook learning with action classification in a unified framework.

The chapter is organized as follows. Our formulation of the BoW model and the subsequent classification phase as a unique global neural model is presented in section 3.2. Section 3.3 describes the integrated and joint learning algorithm which updates the cluster centers as well as the MLP weights dis-

criminating between the targeted classes. Two different learning algorithms are presented, a classical backpropagation algorithm as well as cluster reassignment algorithm specific to the BoW model. Section 3.4 briefly reviews the literature for human action recognition. In section 3.5 we evaluate the proposed approach on the public KTH human action dataset [Schuldt et al., 2004]. Section 3.6 gives our conclusion.

3.2 The neural model

In our application, visual concepts are defined on the local primitives (i.e. space-time interest points). A large amount of work exits on space-time interest point detectors using different criteria. We adopted Laptev's method to obtain space-time interest points and describe them by histogram of gradient (HOG) and histogram of optical flow (HOF) features (refer to section 2.2.2).

Space-time interest points are calculated on each video and discriminant and invariant features are calculated on a space-time cuboid around each interest point location. Initially, a video is therefore described as a collection of feature vectors. In traditional ways to translate this description into a BoW model, codebook creation and learning of the BoW models of the training set are treated as two different phases addressed with two different methods. Here we present a novel formulation as a single artificial neural network.

In classical neural networks, each entity is classified separately by the learned NN after a stimulation phase. In our proposed model, for each video multiple feature vectors (one per interest point) are presented sequentially while the NN integrates this information internally. Classification is done after all feature vectors have been presented. The scheme in Figure 3.1 illustrates this concept. The box indicated as "$\sum$" corresponds to the module which integrates responses over individual interest points for each video. We first give an overview of its purpose before explaining each layer in detail.

The goal of our proposed model is to express several parts into a single model, which are classically separated: (i) clustering; (ii) feature vector projection on a codebook; (iii) class prediction. For the bag-of-words model, a codebook is traditionally constructed by k-means algorithm on the feature space, where each cluster center is represented by a vector in the feature space and considered as a codeword. Feature vector projection on the codebook is completed by computing the distances between the feature vector and different codewords and selecting the one with minimal distance. Since the procedure only involves vector inner product, it thus can be reformulated as a neural model.

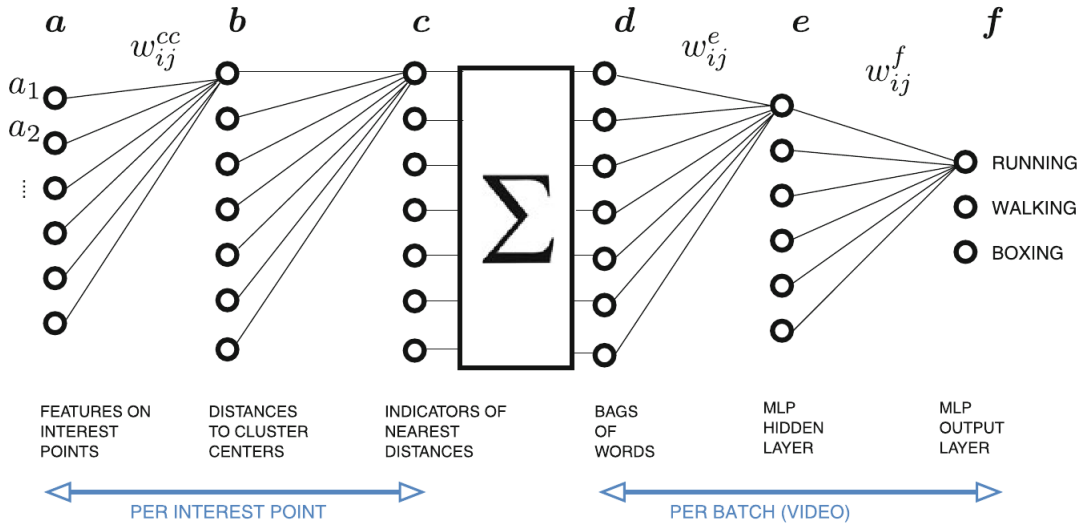


Figure 3.1: A scheme of the different layers of the neural network. The left part is stimulated per interest point. The right part is a classical MLP taking decisions for each video.

The NN consists of two parts: an initial part at the left which processes feature vectors and projects them to a codebook. The cluster centers of the codebook are stored as "weights" of this part of the NN. While passing through the left part of the network, the feature vector is translated into a binary vector indicating which cluster it activates, i.e. which cluster center it is closest to. When several feature vectors are presented, the left part of the NN is stimulated multiple times, once for each feature vector. The right part of the NN is a classical MLP, which takes a decision on the action class for each BoW model.

In this formulation, the cluster centers are coded into the weights of the network, in particular into the weights between the first layer $\boldsymbol{a}$ and the second layer $\boldsymbol{b}$: $\boldsymbol{w}^{cc}$ (where "cc" stands for cluster centers). The connections of the second layer are such that the units of this layer calculate distances between the input and the different cluster centers. Stimulating the network therefore is equivalent of projecting a new input to cluster centers (calculating distances to the different centers). At the same time, changing weights $\boldsymbol{w}^{cc}$ in the neural model is equivalent to changing cluster center positions in feature space.

In Figure 3.1, there are three kinds of different weights: w_{ij}^{cc}, w_{ij}^{e} and w_{ij}^{f}. In the following sections of this chapter, the same subscript index notation is used unless otherwise specified, i.e. the weight w_{ij} denotes the connection between the unit i in the current layer to the unit j in the previous layer. Note that subscripts ij are indices which can take integer values. On the

other hand, superscripts are not indices, i.e. they can not take any values. $\boldsymbol{w}^e$ merely means the symbol denoting the weights for layer $\boldsymbol{e}$.

As a summary, the global model can be linked to classical BoW models in the following way:

- Projecting feature vectors on a learned codebook through the nearest neighbor criterion corresponds to stimulating our network up to layer $\boldsymbol{c}$;
- Calculating a single bag-of-words for a video corresponds to integrating several instances of layer $\boldsymbol{c}$ into a single instance of layer $\boldsymbol{d}$;
- Learning a prediction model of a classifier corresponds to updating weights w_{ij}^e and w_{ij}^f for different ij;
- Learning cluster centers (a.k.a. "clustering") corresponds to updating weights w_{ij}^{cc} for different ij.

3.2.1 The layers of the proposed NN model

The input layer consists of a set $\boldsymbol{a}$ of M input nodes $\boldsymbol{a} = [a_1, \dots a_M]^T$ corresponding to the feature values of length M assigned to a single local primitive, i.e. an interest point.

The N nodes of the second layer $\boldsymbol{b} = [b_1, \dots b_N]^T$ correspond to the distances of the input feature vectors to each of N cluster centers. To each node i and each distance b_i is thus assigned a cluster center $\boldsymbol{w}_{i\cdot}^{cc}$, i.e. a vector of dimension M, denoted by $\boldsymbol{w}_{i\cdot}^{cc} = [w_{i1}^{cc}, w_{i2}^{cc}, \cdots, w_{iM}^{cc}]^T$, which is involved in the distance computation:

$$b_i = ||\boldsymbol{a} - \boldsymbol{w}_{i\cdot}^{cc}|| \tag{3.1}$$

The N nodes of the third layer compute an indicator of the nearest cluster center. The nearest corresponding node will be assigned 1, the other nodes 0. The minimum distance will result in the largest value. This is approximated through a softmin function $g^b(x)$, similar to the classical softmax:

$$c_i = g^b(b_i) = \frac{\exp\left(-b_i/T\right)}{\sum_{j=1}^{N} \exp\left(-b_j/T\right)} \tag{3.2}$$

where T is a parameter controlling the stability of the softmin function.

The network layers described above propagate the stimulation of a single feature vector corresponding to a single local primitive. As mentioned before, in our stimulation strategy, multiple feature vectors of the same entity (a video in our case) are presented iteratively, resulting in different values for different

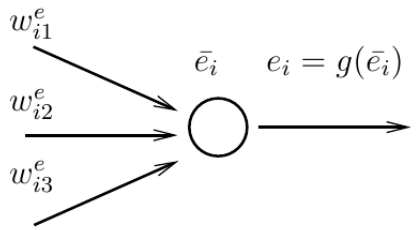


Figure 3.2: Illustration of the notation used for units in this chapter: symbols with bars (e.g. $\bar{e}_i$) indicate the result of the linear combinations performed by each unit. Symbols without bars indicate the output of the unit after the activation function, e.g. $e_i = g(\bar{e}_i)$. The chosen activation function depends on the layer.

feature vectors, which we will denote as $c_i(p)$. The nodes of the next layer integrate the responses for a single video over all P points:

$$d_i = \sum_{p=1}^{P} c_i(p) \tag{3.3}$$

Whenever it is clear from the context, $c_i(p)$ will be abbreviated as c_i. The next two layers, $\boldsymbol{e}$ and $\boldsymbol{f}$, are the hidden and output layer of a classical MLP with weights $\boldsymbol{w}^e$, $\boldsymbol{w}^f$ and activation functions for different layers. They respectively have N^e and C nodes, where N_e is empirically chosen and C is the class label size:

$$\bar{e}_i = \sum_{j=1}^{M} w^e_{ij} d_j, \quad e_i = g(\bar{e}_i) \tag{3.4}$$

$$\bar{f}_i = \sum_{j=1}^{N^e} w^f_{ij} e_j, \quad f_i = g(\bar{f}_i) \tag{3.5}$$

where $\bar{e}_i$ and $\bar{f}_i$ is the linear combination of the output in the previous layer, and e_i and f_i is the values after applying a nonlinear activation function. Figure 3.2 illustrates the operations in equation 3.4 when $M = 3$. Here we use the same notation g to denote the activation functions for different layers. However the choice of activation function depends on the specific situation, for instance, the sigmoid function is commonly adopted for the hidden layer and the softmax function for the output layer. The last layer is thus the output layer with the set of nodes $\boldsymbol{f} = [f_1, \dots f_C]^T$, giving the confidence value for each of the C class labels.

Algorithm 1: The iterative codebook learning framework

Input: F_{tr} (training features), F_{val} (validation features)
Output: $\boldsymbol{w}^{cc}$ (optimal codebook)

$\boldsymbol{w}^{cc} \leftarrow$ k-means ;
repeat
  $\boldsymbol{B}_{tr} \leftarrow$ Bags-of-words computation $(\boldsymbol{w}^{cc}, F_{tr})$;
  $\boldsymbol{B}_{val} \leftarrow$ Bags-of-words computation $(\boldsymbol{w}^{cc}, F_{val})$;
  $\boldsymbol{w}^{e,f} \leftarrow$ random ;
  $\boldsymbol{w}^{e,f} \leftarrow$ MLP learning $(\boldsymbol{w}^{e,f}, \boldsymbol{w}^{cc}, \boldsymbol{B}_{tr}, L_{tr})$;
  $\boldsymbol{w}^{cc} \leftarrow$ Cluster center learning $(\boldsymbol{w}^{e,f}, F_{tr}, L_{tr})$ { *section 3.3.2 or 3.3.3* } ;
  $E \leftarrow$ Validation error $(\boldsymbol{w}^{e,f}, \boldsymbol{w}^{cc}, \boldsymbol{B}_{val}, L_{val})$;
until *convergence(E)* ;

3.3 Joint codebook and class label learning

Assuming a set of video files with interest points and their corresponding feature vectors, as well as a groundtruth action label per video, backpropagation propagates the error between the stimulated response and the groundtruth back to the input layers, adjusting weights during the process. Here the groundtruth of course indicates the action class of a video, and not a cluster center. In our case, this means adjusting two different types of parameters, namely the weights of the MLP as well as the cluster centers $\boldsymbol{w}^{cc}$. The weights of the MLP are updated with a classical error backpropagation scheme, which is recalled in section 3.3.1. The cluster centers can be updated by two algorithms, which are respectively addressed in section 3.3.2 and 3.3.3. These two different types of weights are sequentially and iteratively learned. At any time the proposed system only learns one type of weights. The former is used to learn an optimal MLP model and the latter makes use of the backward errors of the optimal MLP. The pseudo code of the proposed framework is shown in Algorithm 1.

3.3.1 MLP learning

In the proposed framework, we adopt a MLP network (i.e. layers $\boldsymbol{d}$, $\boldsymbol{e}$ and $\boldsymbol{f}$) with one output unit for each class. No activation functions are used in the layer $\boldsymbol{d}$ of the MLP. Sigmoid and softmax activation functions are respectively employed in the hidden layer $\boldsymbol{e}$ and output layer $\boldsymbol{f}$. An 1-of-c coding scheme is used to describe the target outputs, which is coded as a binary indicator

vector of dimension (the number of action classes). The weights $\boldsymbol{w}^e$ and $\boldsymbol{w}^f$ are adjusted by the classical backpropagation algorithm:

The different input vectors $\boldsymbol{a}$ forward propagate through the network, until all the input vectors over the whole video are summed in the integration layer, producing a bag-of-words representation $\boldsymbol{d}$ of the video, and then the activations of all hidden and output layer units are computed. Given the desired output (the target) t_j from the groundtruth and the response f_j for each unit, the goal is to minimize a loss. Here the classical cross-entropy loss has been chosen [Bishop, 1994]:

$$E = -\sum_{j=1}^{C} t_j \ln f_j \tag{3.6}$$

We remind the reader, that the target value t_j is related to the label of the video activity class. For instance, if a video is known to be of class 3, then, according to the classical 1-of-c coding scheme, the ground truth vector t is set as

$$t = [0, 0, 1, 0, 0, \ldots, 0]^T \tag{3.7}$$

where the size of t is equal to C, the number of activity classes.

The goal is to calculate the derivatives of loss with respect to all different weights of the model (w^e_{ij}, w^f_{ij} and w^{cc}_{ij}). For this reason, we start by calculating the derivatives of the loss w.r.t. different intermediate values, starting with the network outputs f_j, which can be given as follows:

$$\frac{\partial E}{\partial f_j} = \frac{-t_j}{f_j} \tag{3.8}$$

Let us recall from the beginning of this subsection that we use a softmax activation function in the last layer. Therefore, the network output f_j is given as follows from the linear input $\bar{f}_j$ (we recall the notation f_j and $\bar{f}_j$ which are illustrated in Figure 3.2):

$$f_j = \frac{\exp(\bar{f}_j)}{\sum_{k=1}^{C} \exp(\bar{f}_k)} \tag{3.9}$$

The derivative of the loss E w.r.t $\bar{f}_j$ requires the derivative of the softmax function, which is hard to calculate, and whose derivation is beyond the scope of this paper. Fortunately it is well known, we refer the reader to [Bishop, 1994]. We can give the derivative of E w.r.t. the linear network outputs $\bar{f}_j$ as follows (a known result):

$$\frac{\partial E}{\partial \bar{f}_j} = (f_j - t_j) \tag{3.10}$$

In the following, we will abbreviate the derivative of the loss as follows:

$$\delta_j^f = \frac{\partial E}{\partial \bar{f}_j} \quad \forall j = \{1, \cdots, C\} \tag{3.11}$$

We can now backpropagate the errors in the output layer $\boldsymbol{f}$ into the hidden layer $\boldsymbol{e}$, which sequentially pass through linear combinations and activation function of layer $\boldsymbol{e}$. The error δ_j^e for each unit j in the hidden layer $\boldsymbol{e}$ is calculated according to the chain rule for partial derivatives:

$$\delta_j^e = g'(\bar{e_j}) \sum_{i=1}^{C} w_{ij}^f \delta_i^f, \quad \forall j = \{1, \cdots, N^e\} \tag{3.12}$$

where the sum runs over all the C units in the output layer $\boldsymbol{f}$ connecting to the unit j in the hidden layer $\boldsymbol{e}$. $g'(\bar{e_j})$ is the derivative value of activation function at $\bar{e_j}$ employed in the hidden layer (i.e. sigmoid function), which is computed as follows:

$$g'(\bar{e_j}) = g(\bar{e_j})(1 - g(\bar{e_j})). \tag{3.13}$$

We next backpropagate the errors in the hidden layer $\boldsymbol{e}$ to the input layer $\boldsymbol{d}$ of the MLP. The error δ_j^d for each unit j in the layer $\boldsymbol{d}$ is calculated by:

$$\delta_j^d = \sum_{i=1}^{N_e} w_{ij}^e \delta_i^e, \quad \forall j = \{1, \cdots, M\} \tag{3.14}$$

where the chain rule is once again employed. The derivative of the activation function is not involved, because no activation function is adopted in the layer $\boldsymbol{d}$.

Lastly we compute the increments for all the weights of the MLP (i.e. $\boldsymbol{w}^e$ and $\boldsymbol{w}^f$). It can be obtained by multiplying the error at the output end of the weight (i.e. the unit indicated by the first index of the weight) by the value at the input end of the weight (i.e. the unit indicated by the second index of the weight). For instance, for the weight w_{ij}^f connecting the unit i in the layer $\boldsymbol{f}$ to the unit j in the layer $\boldsymbol{e}$, the error of its output end is δ_i^f, and the value of its input end is e_j. The increments are calculated by:

$$\Delta w_{ij}^f = \eta \delta_i^f e_j \quad \Delta w_{ij}^e = \eta \delta_i^e d_j, \tag{3.15}$$

where η is a learning parameter.

3.3.2 Supervised codebook learning with error backpropagation

The classical error backpropagation algorithm can be adapted to our novel formulation. In particular, the errors in layer $\boldsymbol{d}$ are continued backpropagating

into layer $\boldsymbol{c}$ and then into layer $\boldsymbol{b}$, which can directly act on the cluster centers w_{ij}^{cc}. The goal of the derivations in this section is to relate the loss coming from the right side of the network to the weights w_{ij}^{cc}. The particularity of our model allows a geometric interpretation of this update, which corresponds to a movement of a cluster center, i.e. an M-dimensional vector in M dimensional feature space.

We therefore first recall the goal of the following derivations, namely to calculate the derivative of the loss E with respect to the parameter w_{ij}^{cc} :

$$\frac{\partial E}{\partial w_{ij}^{cc}} \tag{3.16}$$

A particularity of our model is the integrator between the *per-feature-vector* part and the *per-video* part. As mentioned before, the network output is calculated over multiple stimulations of the left side of the network, one for each feature vector. The loss E has been calculated on the right side of the network for each video. We will denote by p the index of the feature vectors and for clarity we temporarily add the dependency on the feature vector to the notation. The feature vector indexed by p is denoted as $a(p)$ with values $a(p) = [a_1(p), a_2(p), \dots a_M(p)]$; the value of unit i of layer $\boldsymbol{b}$ obtained when the network was stimulated with feature vector $a(p)$ is therefore $b_i(p)$ etc.

The error $\delta_i^c(p) = \frac{\partial E}{c_i(p)}$ of layer $\boldsymbol{c}$ (nearest distance indicator for each feature point) is supposed to be backpropagated from the succeeding layer $\boldsymbol{d}$ (bag of words, i.e. the histogram). The hickup here is the unconventionnal structure of our model, in particular the integrator between the right *per-video* part and the left *per-feature-vector* part. We circumvented this problem by taking the error on the right layer $\boldsymbol{d}$ and equally distributing it over the corresponding feature points p:

$$\delta_i^c(p) = \frac{\delta_i^d}{d_i} \quad \forall p = \{1, \dots, P\} \tag{3.17}$$

The special case of $d_i = 0$ for unit i is a singularity. However it has a meaningful explanation from a model perspective, i.e. $d_i = 0$ means that there is no feature vector belonging to this cluster center for the input, and the unit i in the layer $\boldsymbol{d}$ can thus be ignored in the classification. We therefore do not update the weights in this case.

Let us recall that the goal is to update the weights $\boldsymbol{w}^{cc}$, which correspond to the encoded cluster centers. We therefore need to backpropagate the error further to the previous layer $\boldsymbol{b}$. Let us recall the definition of layer $\boldsymbol{b}$ by reproducing equation (3.1):

$$b_i = ||\boldsymbol{a} - \boldsymbol{w}_{i\cdot}^{cc}||^2 = \sum_{j=1}^{M} (a_j - w_{ij}^{cc})^2. \tag{3.18}$$

The equation above can be interpreted as a common neural network layer with an uncommon activation function. Indeed, in traditional MLPs, the activation function consists of a non-linearity acting on the linear sum of the input variables, and the coefficients are the parameters of the model. In our case, the activation function acts on a sum of a two-order polynomial of input variables. Since b_i is differentiable with respect to w_{ij}^{cc}, and the softmin component is also differentiable, we thus can resort to gradient descent to adjust the cluster centers. Calculating the gradient requires the following derivates :

$$\frac{\partial E}{\partial w_{ij}^{cc}} = \sum_{p=1}^{P} \frac{\partial E(p)}{\partial w_{ij}^{cc}} = \sum_{p=1}^{P} \frac{\partial E(p)}{\partial b_i(p)} \frac{\partial b_i(p)}{\partial w_{ij}^{cc}} \tag{3.19}$$

In the following, only the values for a single feature vector p will be considered. For clarity we therefore remove (again) the index p from the notation; in particular $E(p)$ will be noted as E.

According to the chain rule, the derivative of E with respect to b_i can be decomposed of two parts: the derivative of E with respect to the units in the layer $\boldsymbol{c}$, and the derivative of the softmin function between layer $\boldsymbol{b}$ and layer $\boldsymbol{c}$. We should consider the input node i in the layer $\boldsymbol{b}$ to all the output units in the layer $\boldsymbol{c}$. So we have

$$\frac{\partial E}{\partial b_i} = \sum_{k=1}^{N} \frac{\partial E}{\partial c_k} \frac{\partial c_k}{\partial b_i}. \tag{3.20}$$

Let us recall the softmin function in layer $\boldsymbol{c}$:

$$c_k = \frac{\exp\left(-b_k/T\right)}{\sum_{i=1}^{N} \exp\left(-b_i/T\right)} \tag{3.21}$$

The derivative involves a rather long series of algebra, which we will not reproduce here. It can be given as:

$$\frac{\partial c_k}{\partial b_i} = -\frac{1}{T}(c_k \Delta_{ik} - c_k c_i) \tag{3.22}$$

where Δ_{ik} is the Kronecker delta function with $\Delta_{ik} = 1$ if $i = k$ and 0 else. It can be seen that the computations of the partial derivatives of c_k with respect to b_i are not complex, which only need the current values of layer $\boldsymbol{c}$.

Substituting equations (3.17) and (3.22) into equation (3.20), we obtain

$$\frac{\partial E}{\partial b_i} = -\sum_{k=1}^{N} \frac{\delta_k^c}{T}(c_k \Delta_{ik} - c_k c_i). \tag{3.23}$$

From equation (3.18), we can get the derivative of b_i with respect to w_{ij}^{cc}:

$$\frac{\partial b_i}{\partial w_{ij}^{cc}} = -2(a_j - w_{ij}^{cc}). \tag{3.24}$$

Now equations (3.23) and (3.24) are substituted into equation (3.19), which gives us the final goal, namely the derivatives of the loss E with respect to the cluster centers w_{ij}^{cc}:

$$\frac{\partial E}{\partial w_{ij}^{cc}} = \frac{2}{T}(a_j - w_{ij}^{cc}) \sum_{k=1}^{N} \frac{\delta_k^c}{T}(c_k \Delta_{ik} - c_k c_i). \tag{3.25}$$

In order to update the weights, one step of gradient descent algorithm is performed as follows:

$$\Delta w_{ij}^{cc} \longleftarrow \alpha \Delta w_{ij}^{cc} - \eta_b \sum_{p=1}^{P} \frac{\partial E(p)}{\partial w_{ij}^{cc}} \tag{3.26}$$

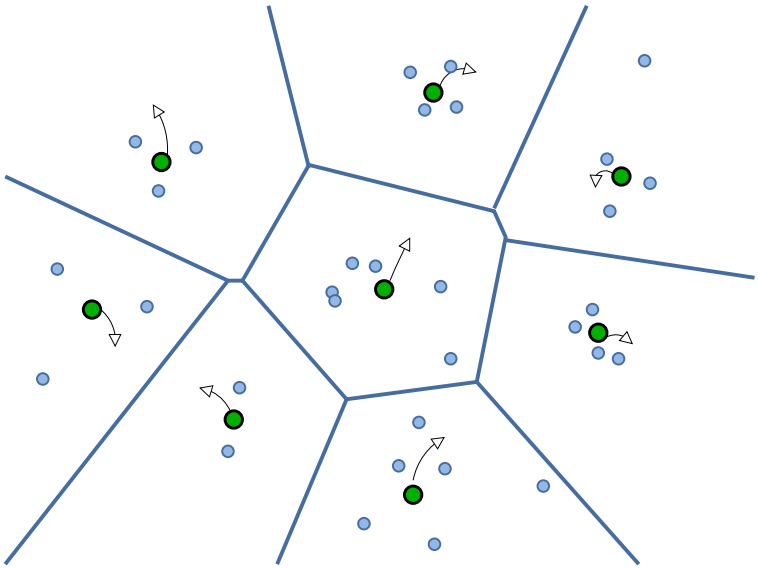
where α and η_b are learning parameters and the feature vector index p has been used again to distinguish the batch entries. The cluster centers are adjusted after stimulation for each input feature vector.

3.3.3 Supervised codebook learning through cluster reassignment

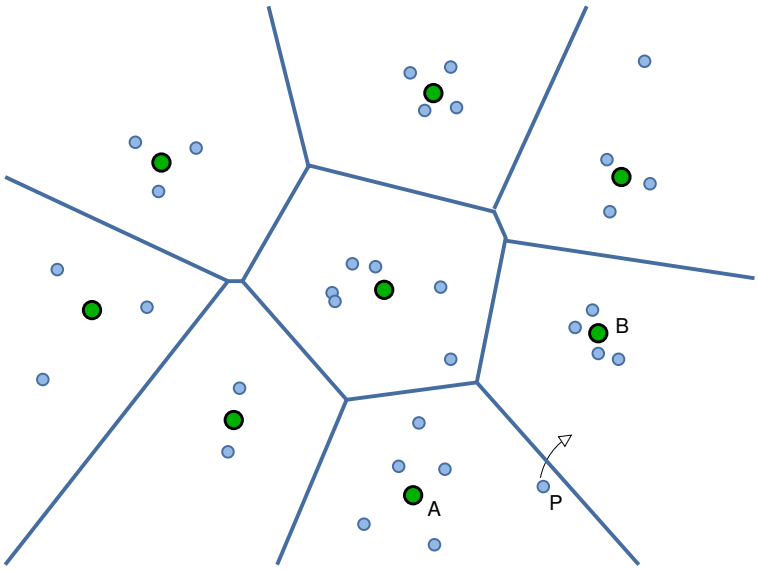
In the previous subsection we have presented a learning algorithm which adopted gradient descent after classical error backpropagation to the particular functional form of our NN architecture. The BoW in layer $\boldsymbol{d}$ was interpreted as a general numerical vector without any special structure.

In this subsection we propose another algorithm which uses our prior knowledge that the information stored in layer $\boldsymbol{d}$ is a BoW, i.e. a histogram. Instead of simply backprojecting an error of this layer through the softmin function which, after all, is an approximation of the required minimum function, we change it by moving input feature vectors from one histogram bin to another one.

This strategy is illustrated in Figure 3.3(b), in comparison to Figure 3.3(a) corresponding to the method described in 3.3.2. The feature space is divided into several Voronoi cells, each of which denotes one cluster. The blue and small points are training points, the green and large points are the cluster centers. The training points are projected into cluster center and are assigned the cluster index as their labels. d_A and d_B give the training point number in the cluster A and B. The error δ_A^d and δ_B^d can be respectively interpreted



(a)



(b)

Figure 3.3: Two different ways to learn the cluster centers illustrated through a Voronoi diagram of the feature space, for simplicity in 2D. Cluster centers are green and large, training points are blue and small. Recall that Voronoi cells do not correspond to decision areas of prediction model for actions. A video is represented by a bag of multiple points, i.e. a histogram of cluster indicators/Voronoi cells. (a) The method described in section 3.3.2 directly updates the cluster centers w_{ij}^{cc} by gradient descent on the loss of the video. The error is equally distributed over the clusters/Voronoi cells; (b) The method described in section 3.3.3 indirectly updates the cluster centers by passes individual feature vectors from one Voronoi to another one according to the error in the BoW layer.

as the difference between the real training point number and the optimal training point number in the corresponding cell. Supposing the error δ_A^d is positive (too many feature vectors) and the error δ_B^d is negative (not enough feature vectors), at each weight update a single feature vector is moved from the Voronoi cell of A into the Voronoi cell of B, minimizing the sum of the errors, and followed by an update of the cluster centers as a calculation of the mean of the assigned training points.

It is generally required to move several feature vectors in order to significantly change the errors of the BoW layer $\boldsymbol{d}$, denoted as $\delta^d = \{\delta_0^d, \delta_1^d, \cdots, \delta_N^d\}$. A positive error of a value (histogram bin) of the BoW, e.g. $\delta_i^d > 0$, indicates that at least one feature vector has been assigned to the cluster center corresponding to this bin which should been assigned to a different cluster according to the ground truth. In the same sense, a negative error $\delta_j^d < 0$ indicates that at least one feature vector should be added to this histogram bin of the BoW. In the following we suppose that the solution to this problem is specified as a multi set $\boldsymbol{x} = \{x_1, x_2, \ldots, x_D\}$ of indices indicating from where a vector is moved, and a multi set $\boldsymbol{y} = \{y_1, y_2, \ldots, y_D\}$ of indices indicating to which cluster a vector is moved, where D shows the movement number. For instance, $x_1 = 5$ and $y_1 = 7$ indicate that the first move will go from cluster 5 to cluster 7.

A good solution should minimize two criteria. First, the error should be low, i.e. we should minimize

$$\min_{\boldsymbol{x},\boldsymbol{y}} \left[\sum_k \delta_k^d - |\{x_s : x_s = k\}| + |\{y_s : y_s = k\}| \right] \tag{3.27}$$

where $|\{x_s : x_s = k\}|$ is the number of source indices equal to k, and the second expression can be understood in a similar way. Secondly, the feature vector movements performed by the solution should be minimal, i.e. we should minimize

$$\min_{\boldsymbol{x},\boldsymbol{y}} \left[\sum_{s=1}^{D} |b_{x_s} - b_{y_s}| \right] \tag{3.28}$$

One possibility would be to minimize an energy function consisting of a weighted linear combination of (3.27) and (3.28). Instead, we opted for an iterative greedy solution, where cluster pairs (i, j) are chosen decreasing (3.27) and then for each pair of clusters a feature vector is chosen such that its move from cluster i to cluster j minimizes (3.28). We added an additional constraint requiring that the chosen feature vector to move — which is (naturally) closest to cluster center A — also be second closest to cluster center B. The details of the update algorithm are given as follows:

1. Randomly choose a pair (i, j) of histogram bins (thus of cluster centers), where the error of one bin is positive and the other is negative, i.e. $\delta_i^d > 0 \ \wedge \ \delta_j^d < 0$.

2. Calculate the Voronoi diagram of the cluster centers in feature space and determine all the feature vectors of the training set falling into the sets of $\boldsymbol{w}_i^{cc}$ and $\boldsymbol{w}_j^{cc}$, respectively.

3. Pick a single feature vector f such that:

   - it falls into the Voronoi cell i (distance between f and cluster center $\boldsymbol{w}_i^{cc}$) is minimum, i.e. nearest).
   - its distance to cluster center $\boldsymbol{w}_j^{cc}$ is second nearest.
   - if several vectors satisfy the above two criteria, choose the one minimizing the distance to the border of the two Voronoi cells, i.e. the one minimizing $|b_i - b_j|$.

4. The chosen feature vector f is reassigned from histogram bin i to histogram bin j with the following consequences:

   - the two centers $\boldsymbol{w}_i^{cc}$ and $\boldsymbol{w}_j^{cc}$ are recalculated as the means of the feature vectors of their respective Voronoi cells.
   - the errors of the BoW layer of the corresponding bins are updated:

$$\begin{aligned} \delta_i^{d\ [t+1]} &= \delta_i^{d\ [t]} - 1 \\ \delta_j^{d\ [t+1]} &= \delta_j^{d\ [t]} + 1 \end{aligned} \tag{3.29}$$

5. The reassignments are continued (back to step 1) until the error of layer $\boldsymbol{d}$ is zero or none of the feature vectors satisfy the above conditions.

3.4 BoW models for human action recognition

Along with the wide use of portable imaging devices, a great mass of videos have been created everyday, thus analyzing so many videos manually frame-by-frame seems impossible. In order to alleviate human labors, video analysis automatically by the machine is necessary. Recognizing human actions from videos is one of such directions, which has become an active research topic in computer vision community over the past three decades due to a big amount of potential promising applications, for instance, video surveillance, content-based video retrieval, human-computer interaction and so on.

An action is typically considered as a simple motion pattern performed by a single person, usually having a short duration of time, such as walking, bending, jumping etc. An activity is a more complex motion pattern, composing of a number of sequential actions performed by several persons. It is often associated with interactions between several persons or between humans and objects, of course having a relative longer duration. Examples of activities include two lovers kissing, a team playing football etc. Activity recognition to some extent is more challenging than action recognition. Some researchers also stated the hierarchy relationship between the terms "Action" and "Activity". However these two terms are frequently used interchangeably in most literature. Unless otherwise specified, we ignore their difference and consider the two terms as the same thing in the following.

Action recognition is a challenging problem. Actions in videos present larger variations than objects in images because of several reasons: (i) videos also vary in the time dimension; (ii) relevant motion information is mixed with irrelevant appearance information in the signal; (iii) activities are inherently and highly articulated.

Due to its huge promising applications in various domains, the amount of literature on action recognition sky-rocketed in the last years and it is not possible anymore to give an exhaustive account in this thesis. We refer the interested reader to some very recently published surveys [Aggarwal and Ryoo, 2011, Turaga et al., 2008, Weinland et al., 2011]. While early work on modeling human activities focused on articulated motion (e.g. [Goncalves and Perona, 2003]), most recent work on activity and event recognition does not explicitly model the human body. Instead, the current state of the art focuses on sparse local features like interest points and space-time interest points [Csurka et al., 2004, Lowe, 2004], or on motion segmentation through background subtraction [Wang and Geng, 2008, Weinland et al., 2007], dense optical flow [Sukthankar and Hebert, 2005] or other holistic features [Mikolajczyk and Uemura, 2008, Zhang et al., 2008], with possible hybrid methods [Bregonzio et al., 2009, Liu et al., 2008, Sun et al., 2009] and classification through dense matching [Seo and Milanfar, 2010, Shechtman and Irani, 2005]. Fully taking into account spatial relationships through graph matching has recently been proposed [Ta et al., 2010a], but this requires matching against several graph models per action class.

Pure statistical and unstructured machine learning without feature extraction is difficult in this context due to several reasons: (i) the non-rigid nature of the relevant information and (ii) the mixture of relevant motion information and irrelevant texture information in the signal (iii) the high dimension of the feature space in which the data is embedded. For these reasons, machine learn-

ing of human actions has been dominated by methods learning the temporal evolution of features like HMMs, Semi-Markov models and dynamic belief networks [Abdelkader et al., 2010, Boiman and Irani, 2007, Cuntoor et al., 2008, Shi et al., 2010, Xiang and Gong, 2008a, Xiang and Gong, 2008b, Zhang and Gatica-Perez, 2005, Zhou and Kimber, 2006]. Typically, a vectorial description is created frame per frame and its temporal evolution is modeled and learned, such as Transferable Belief model [Ramasso et al., 2007, Ramasso et al., 2009]. Other learning-based methods include biologically-inspired ones [Jhuang et al., 2007], convolutional deep learning [Taylor et al., 2010, Baccouche et al., 2011], methods based on topic models [Niebles et al., 2008], boosting low-level features [Fathi and Mori, 2008], trajectory matching [Dyana and Das, 2009], statistics calculated on the results of tracking [Stauffer and Grimson, 2000], learning of spatio-temporal predicates and grammars [Ryoo and Aggarwal, 2009, Ryoo and Aggarwal, 2011, Wang et al., 2010] and other probabilistic graphical models [Niebles and Fei-Fei, 2007]. Among them, the bag-of-words (BoW) model is one of the most popular models due to its simplicity. The next section will present experiments applying the proposed model to human action recognition.

3.5 Experimental Results

The proposed model and learning algorithms have been evaluated on the publicly available KTH action dataset [Schuldt et al., 2004]. It is one of the largest available published datasets and contains 6 actions – boxing, hand clapping, hand waving, jogging, running and walking, performed by 25 subjects in four different scenarios – indoors, outdoors, outdoors with scale variation and clothes changing. It contains 2391 video sequences and each sequence lasts four seconds in average. Representative frames of the dataset are shown in Figure 3.4. For video representation, space-time interest points were detected by the 3D Harris-corner detector proposed by Laptev [Laptev et al., 2008]. A space-time cuboid was extracted around each interest point and described by features of type histogram of gradient (HOG) and histogram of oriented flow (HOF) descriptors, which were obtained from the software supplied by Laptev [Laptev et al., 2008].

As usual, a cross-validation scheme and early stopping strategy are employed to control the learning phase. The dataset is divided into three independent sets: training (12 people), validation (4 people) and test (9 people), as in [Schuldt et al., 2004]. The MLP is trained on the training set and evaluated on the validation set for stopping to avoid over-fitting. Unless said

Figure 3.4: The KTH dataset [Schuldt et al., 2004].

otherwise, all errors are reported on the 863 test instances.

Different MLP architectures and learning hyper-parameters were tested, and the best ones were chosen from the performances on the test set. Values are given below. Executions times are given for a C++ implementation running on an Intel Core i7 640 PC under Linux.

Classical unsupervised codebook learning — To demonstrate the discriminative power of classical codebook and compare with our methods, we created a baseline with classical unsupervised k-means clustering and MLP learning of the BoW descriptors. With respect to Figure 3.1, this corresponds to a scheme where the weights $\boldsymbol{w}^{cc}$ are set without taking into account cluster labels of the groundtruth, and with supervised MLP learning of the action class decisions.

In our baseline experiments, the cluster centers were learned by k-means and then the MLP part was trained given the BoW models. To cope for random initialization of the MLP weights, we repeated our baseline experiments in order to obtain statistically sound results: first a codebook is created using k-means clustering. Then, for each run, the cluster centers were kept fixed and the MLP weights were randomly initialized between -0.5 and 0.5 and learned. We ran 100 runs for each codebook in our experiments. Different MLP architectures were explored with numbers of hidden units of 25, 75, 100 for 50, 150, 300 codewords. Figure 3.5 shows an example using 150 codewords, where learning stops at the 54^{th} iteration.

Table 3.1 shows error rates (on the test set) of the learned MLP with different codebooks. A local running mean filter was applied to the results. From the table, we can see that the MLP learning is robust. On the other hand, a larger codebook is more discriminative, resulting in lower error.

Supervised codebook learning with error backpropagation — Re-

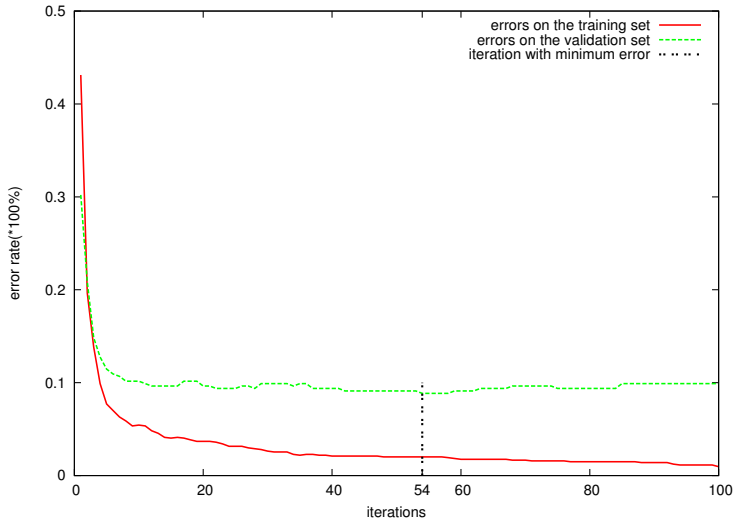


Figure 3.5: A schematic illustration of the early stopping strategy during MLP learning with 150 codewords

Table 3.1: Errors in(%) on the test by the MLP with classical unsupervised learned codebook : mean and standard deviation over 100 independent runs.

Codebook size	50	150	300
Error rate	20.86($\pm$0.06)	18.34($\pm$0.02)	16.86($\pm$0.05)

sults with supervised learning through the backpropagation method (section 3.3.2) are shown in Figure 3.6. We repeated the above experiments with the same architecture, except that the cluster centers were adjusted by using a gradient descent algorithm according to the back-propagated errors of the optimal MLP in each iteration and the BoW entities of the videos were recomputed. We tried several values and selected the best parameters for gradient descent. α was set to 0.00001 for all the codebooks and η_b varied for different codebooks. The MLP was retrained and the error rates are depicted in Figure 3.6, again after applying a local running mean to smooth the data. It can be seen that the error decreases at the beginning and converges for 50 codewords and for 150 codewords. However, for 300 codewords the error oscillates after 120 iterations due to the non-adaptive characteristics of gradient descent. Comparative results are presented in Table 3.2. We can see that supervised codebook learning through error backpropagation approximately gains 2.1% for 50 codewords and 0.8% for 300 codewords with respect to the baseline codebooks obtained with k-means clustering.

Supervised codebook learning with cluster reassignment — Re-

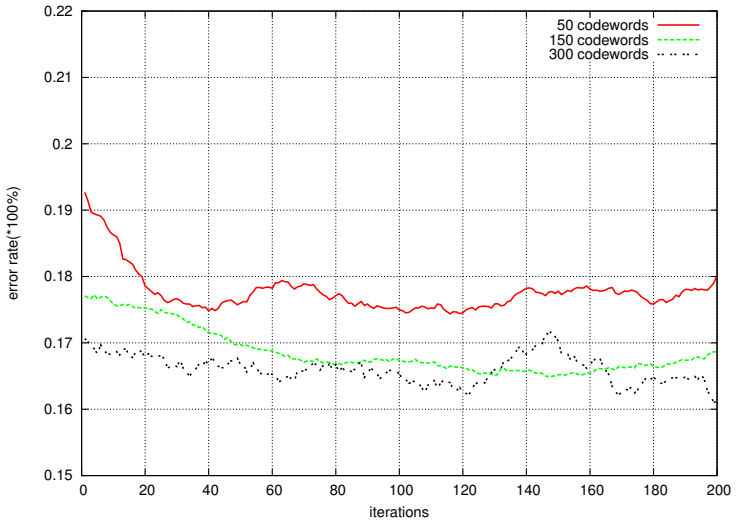


Figure 3.6: Supervised learning with error backpropagation (section 3.3.2): errors on the test set over different iterations.

sults with supervised learning through the cluster reassignment method (section 3.3.3) are shown in Figure 3.7. We again repeated the above experiments with the same neural architecture. At each iteration, the cluster centers were adjusted using the Voronoi cell updates. and then the MLP is retrained. Figure 3.7 shows the results, which are obtained by applying a local running mean. As we can see the classification accuracy on the test set increases as the cluster centers are adjusted. The improvement is higher with smaller codebook, which is also observed from Figure 3.6. The learned codebooks through cluster reassignment therefore improve by 1.7% for 50 codewords and 0.8% for 300 codewords with respect to the baseline codebooks obtained with k-means clustering.

From Table 3.2, we can see that both methods clearly improve the discriminative quality of the codebook when the codebook size is small. This is an important advantage since larger codebooks significantly increase the computational complexity of the classification algorithm due to the nearest neighbor search necessary when the feature vectors are projected on the codebook. Indexed data-structures like KD-trees are not always helpful in these situations since visual data is generally embedded in feature spaces of very high dimensions — 162 dimensions for the HoG/HoF features we employed in our experiments.

The performance improvement can be explained by the nature of the features it creates. The k-means algorithm clusters features based on the appearance of the cuboids only. When the codebook is small, the intra-cluster variance is large, which lowers discriminative power. Our methods regroup the

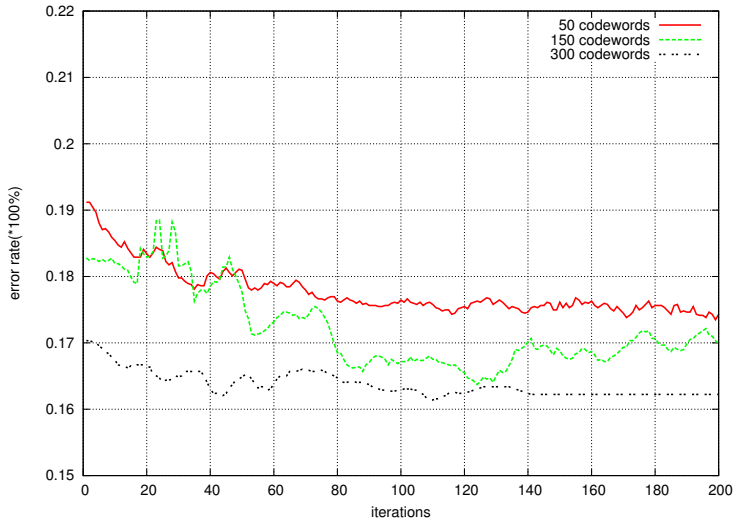


Figure 3.7: Supervised learning with cluster reassignment (section 3.3.3): errors on the test set over different iterations.

feature vectors into different clusters based on class labels of the groundtruth, thus choosing optimal codewords.

Retraining with SVM classifiers — The experiments above show that the combined codebook and MLP learning outperforms the classical sequential steps k-means clustering followed by MLP learning. However, the question arises whether the improvement is due to a better codebook or to the integration of the two methods. We therefore performed an additional experiment with a two-step learning process:

1. Codebook learning according to one of the three methods (k-means; supervised codebook learning with error backpropagation; supervised codebook learning with cluster reassignment).

2. Class label retraining with Support Vector Machines (SVM) on the learned codebook.

We trained an SVM with a radial basis function kernel on the training set and validation set, which were the same with the ones used in the above experiments [Chang and Lin, 2011]. The errors are shown in the lower block of Table 3.2. Classification time includes the projection of feature vectors on the codebook and the classification with MLP and SVM classifiers. The SVM outperforms the MLP by up to 2% with different codebooks. However, the recognition of the MLP is faster due to its simple calculations which do not need inner-products of high dimensional vectors. As shown in Table 3.3, the MLP gains more benefits from computational complexity with respect to the

Table 3.2: Error in (%) on the test with different methods, different classifiers and different codebook sizes: mean and standard deviation over 3 independent runs.

Classifier	Codebook size	50	150	300
MLP	k-means	19.31(±0.26)	17.44(±0.31)	16.98(±0.64)
	error backpropagation	**17.19**(±0.51)	16.56(±**0.07**)	**16.13**(±0.12)
	cluster reassignment	17.57(±**0.11**)	**16.29**(±0.53)	16.18(±**0.07**)
	recognition time per video(ms)	1.679	4.696	9.294
SVM	k-means	17.16(±0.66)	15.47(±1.06)	15.7(±0.24)
	error backpropagation	**15.94**(±0.25)	15.30(±0.33)	**14.54**(±0.41)
	cluster reassignment	16.80(±**0.16**)	**14.89**(±**0.24**)	14.89(±**0.24**)
	recognition time per video(ms)	1.797	4.905	10.488

Table 3.3: Cost-benefit table (in %) of the MLP compared to the SVM with the results of cluster reassignment method.

Codebook size	50		150		300	
Classifier	SVM	MLP	SVM	MLP	SVM	MLP
Error	100	107	100	109	100	105
Time	100	88	100	96	100	93

costs from error with two codebooks when compared to the SVM. We can also see that the classification performance of our two joint supervised methods is maintained after retraining with a different classifier, indicating a real gain in discriminative power of the codebooks.

Figure 3.8 shows the recognition results for different learning methods on 150 codewords and after retraining with the SVM classifier. Not surprisingly, the largest error is between the classes *running* and *jogging*, which are indeed very similar in behavior. The supervised codebook learning methods can achieve significant gains on some of these classes, as the recognition rate jumps from 65 to 78 for *jogging* with error backpropagation — confirmed by a z-test as described in [Dietterich, 1998].

In this chapter we proposed two different joint supervised learning algorithms. The reformulated backpropagation algorithm adjusts the cluster centers directly through gradient descent algorithm. In contrast to cluster reassignment, two more parameters besides the learning rate η in MLP learning need to be set: the momentum coefficient α and the learning rate η_b. The drawback of a gradient descent algorithm applied to a non-linear system are well-known: it is difficult to learn a set of optimal parameters, the algorithms mostly converge to local minima and sometimes even diverge. As shown in

	Box	Clap	Wave	Jog	Run	Walk
Box	**95**	0	0	0	0	5
Clap	12	**88**	0	0	0	0
Wave	2	7	**91**	0	0	0
Jog	0	0	0	**65**	26	9
Run	0	0	0	23	**73**	4
Walk	0	0	0	7	2	**91**

(a)

	Box	Clap	Wave	Jog	Run	Walk
Box	**94**	0	0	1	0	5
Clap	10	**90**	0	0	0	0
Wave	3	7	**90**	0	0	0
Jog	0	0	0	**78**	14	8
Run	0	0	0	30	**66**	4
Walk	1	0	0	5	1	**93**

(b)

	Box	Clap	Wave	Jog	Run	Walk
Box	**98**	0	0	0	0	2
Clap	13	**87**	0	0	0	0
Wave	2	6	**92**	0	0	0
Jog	0	0	1	**72**	19	8
Run	0	0	0	24	**72**	4
Walk	0	0	1	6	2	**91**

(c)

Figure 3.8: Confusion matrix for a codebook with 150 codewords according to different learning algorithms and after retraining with SVMS. The codebook has been learned with: (a) k-means (b) error backpropagation (section 3.3.2) (c) cluster reassignment (section 3.3.3).

Figure 3.6, the error with 300 codewords began to converge after 60 iterations, but it begin to diverge from 120 iterations.

In comparison, the cluster reassignment algorithm adjusts the cluster centers indirectly by rearranging the cluster labels for all the feature vectors. It does not need any more learning parameters except η in MLP learning, and is easier to control, but needs more iterations to converge, fortunately often to a better solution. From figure 3.7 we can see that the errors converge after 100 iterations. This can also be observed for the errors on 300 codewords — it becomes constant after 140 iterations compared with the errors on 300 codewords in Figure 3.6.

3.6 Conclusion

In this chapter we proposed a joint supervised codebook learning and optimization framework, which can be applied to any method based on bag of words models such as object class recognition or human action recognition. The codebook learning process and the recognition phase are integrated into a uniform framework. The codebook therefore is created in a goal-directed way and is more discriminative than classicial ones. We have presented two algorithms to update the cluster centers (codewords) through the back-propagated errors: one is based on classical error backpropagation, in which the codewords are adjusted using a gradient descent algorithm. The other is based on cluster reassignments, in which we reassign the cluster labels for all the feature vectors based on the errors. Our framework has been tested on the public KTH action dataset, and we have obtained very promising and close results for both methods. At the same time, they demonstrated that error backpropagation learned the optimal codebook faster than cluster reassignment. However it may suffer more from over-fitting, while cluster reassignment is easier to control. The experiments on the KTH human action dataset have confirmed that our framework is able to optimize the codebooks and that it makes them more discriminative. It is also able to increase the speed of a method by decreasing the codebook size while keeping its discriminative power.

Chapter 4

Spatial learning

Contents

4.1 Introduction

Object recognition is one of the fundamental problems in computer vision, as well as related problems like face detection and recognition, person detection, and associated pose estimation. Local representations as collections of descriptors extracted from local image patches [Csurka et al., 2004] are very popular. This representation is robust to occlusions and permits non-rigid matching of articulated objects, like humans and animals. However, the representation is inherently structural and is therefore difficult to use in a statistical learning framework.

In the literature, many vision recognition problems can be solved in part by an initial step which segments an image, a video, or their constituent objects into regions, which are called parts in this context. This general class of problems corresponds to various applications in computer vision. For example, pose estimation methods are also often naturally solved through a decomposition into body parts. A preliminary pixel classification step segments the

object into body parts, from which joint positions can be estimated in a second step. In this case, the subject is each pixel, the recognition techniques presented in section 2.3 are applied to the pixel and give a label it belongs to.

In this chapter, we will propose new ways of integrating spatial relationships into segmentation algorithms, or more precisely, into training algorithms of various learning machines. We will first briefly review important work modeling spatial relationships in computer vision.

For object recognition tasks, the known methods in the literature vary in their degree of usage of spatial relationships, between methods not using them at all, for instance the bag-of-words model [Sivic and Zisserman, 2003], and rigid matching methods using all available information, e.g. based on RANSAC [Fischler and Bolles, 1981]. The former suffers from low discriminative power, whereas the latter only works for rigid transformations and cannot be used for articulated objects.

Methods for non-rigid matching exist. Graph-matching and hyper-graph matching, for instance, restrict the verification of spatial constraints to neighbors in the graph (see section 2.3.1.2 in chapter 2). However, non trivial formulations require minimizing a complex energy functions and are NP-hard [Torresani et al., 2008, Duchenne et al., 2009]. Pictorial structures, have been introduced as early as in 1973 [Fischler and Elschlager, 1973]. The more recent seminal work creates a Bayesian parts based model of the object and its parts, where the possible relative part locations are modeled as a tree structured Markov random field [Felzenszwalb and Huttenlocher, 2005]. The absence of cycles makes minimization of the underlying energy function relatively fast — of course much slower than a model without pairwise terms. In [Felzenszwalb and Zabih, 2011] the Bayesian model is replaced with a more powerful discriminative model, where scale and relative position of each part are treated as latent variables and searched by Latent SVM.

The most existing segmentation algorithms typically consider local appearance information, and frequently also model the spatial relationships between different pixels or different parts. Unfortunately, considering these relationships within the segmentation process mostly amounts to solving constraint satisfaction problems or performing inference in a graphical model with cycles and a non sub-modular energy function, both of which are intractable in the general case. We address the problem of efficiently modeling spatial relationships without the need for solving complex combinatorial problems.

A similar problem occurs in tasks where joint object recognition and segmentation is required. Layout CRFs and extensions model the object as a collection of local parts (patches or even individual pixels), which are related through an energy function [Winn and Shotton, 2006]. However, unlike pictorial structures, the energy function here contains cycles which

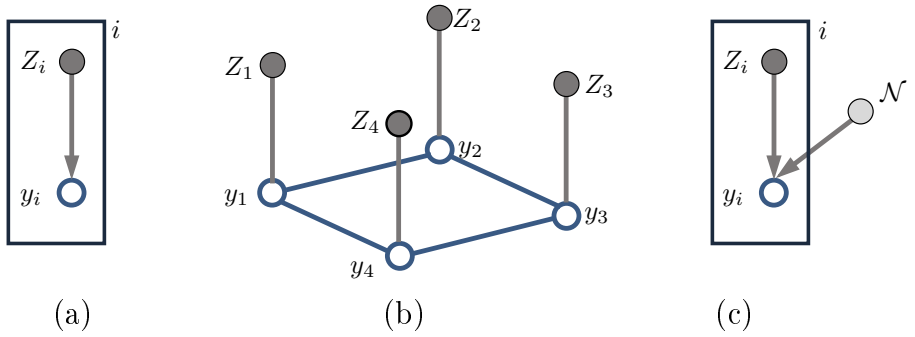


Figure 4.1: Different ways to include spatial layout, or not, into learning parts labels y_i from features Z_i for pixels i: (a) pixelwise independent classification, where spatial layout information is not taken into account; (b) A Markov random field with pairwise terms coding spatial constraints; (c) our method: pixelwise independent classification including spatial constraints $\mathcal{N}$.

makes minimization more complex, for instance through graph cuts techniques. Furthermore, the large number of labels makes the expansion move algorithms inefficient [Kolmogorov and Zabih, 2004]. In the original paper [Winn and Shotton, 2006], and as in one of several contributions in this chapter, the unary terms are based on randomized decision forests. Another related application which could benefit from this contribution is full scene labeling [Farabet et al., 2012].

In all cases, the underlying discrete optimization problem is very similar: an energy function encoding the spatial relationships in pairwise terms needs to be minimized. A typical dependency graph for this kind of problem is shown in Figure 4.1(b): unary terms relate each label y_i to a feature vector Z_i, and pairwise terms encode prior knowledge on the possible configurations of neighboring labels y_i and y_j.

Classical learning machines working on data embedded in a vector space, like neural networks, SVM, randomized decision trees, Adaboost etc., are in principal capable of learning arbitrary complex decision functions, if the underyling prediction model (architecture) is complex enough. The well-known system described in [Shotton et al., 2011], installed on millions of gaming consoles and taking as input images from consumer depth cameras, completely ignores spatial relationships between the object parts and puts all of the classification burden on the pixel-wise random forest classifier. To achieve its state-of-the-art level of performance, it required training on an extremely large training set of $2 \cdot 10^9$ training vectors. These simple systems consist of unary terms only and do not model spatial relationships, i.e. interactions between neighboring labels. In graphical model notation, this situation is shown in

(a) human body (b) motorcycle

Figure 4.2: Part layouts from human body and motorcycle.

figure 4.1(a).

In reality the available amount of training data and computational complexity limit the complexity which can be learned. In most cases only few data are available with respect to the complexity of the problem. It is therefore often useful to impose some structure on the model, or process which is refined to as structural risk minimization in machine learning. In this chapter we propose to use prior knowledge in the form of the spatial layout of the labels to add structure to the decision function learned by the learning machinery, which is evalutated on an application of human part estimation in the depth image.

In this chapter, we propose two methods which segment an image or an object into parts through pixelwise classification, integrating the spatial layout of the part labels on two different learning machineries (i.e. Randomized decision forests and Convolutional neural networks), as shown in figure 4.1(c). Like methods which ignore the spatial layout, they are extremely fast as no additional step needs to be added to pixelwise classification and no energy minimization is necessary during testing. The (slight) additional computational load only concerns learning at an offline stage. The goal is not to compete with methods based on energy minimization, which is impossible through pixelwise classification only. Instead, we aim to improve the performance of pixelwise classification by using all of the available information during learning.

In each of the problems that we consider, the labels we aim to predict have spatial structure. Figure 4.2 illustrates the kind of spatial information we would like to integrate into the classification and learning process. We suppose that class labels correspond to parts of an object or of a human, and that these class labels/parts have neighborhood/adjacency relationships between them. In figure 4.2a, for instance, we can see that the part "head" is a neighbor of part "neck", but not of part "shoulder".

In figure 1.8b in chapter 1, we showed the end-to-end framework for the segmentation problems considered here. Figure 4.3 shows the two different learning frameworks into which we integrate these spatial constraints. Ran-

dom decision forests learn the whole set of parameters (parameters related to the feature extractor and parameters of the prediction model) jointly through maximization of entropy gain [Shotton et al., 2011]. In section 4.3 we propose a way how the spatial information can be introduced into these criteria. Another popular framework is based on the combination of Convolutional Neural Networks (CNN) for feature learning and logistical regression (LR) or MLPs for classification. In section 4.4 we present contributions for the integration of spatial information into two different training algorithms: (i) Weakly supervised dimensionality reduction for feature pre-training [Hadsell et al., 2006]; (ii) Error backpropagation for joint LR/MLP weight learning and fine-tuning of CNN parameters (end-to-end training).

Our proposed methods use different energy functions to enforce a spatial consistency in learned features which reflects the spatial layout of labels. Unlike combinatorial methods, our energy function is minimized during training (i.e. while learning features) but is unused at test time. It is based on two main assumptions. First, different high-dimensional features with the same label are embedded into a lower-dimensional manifold which preserves the original semantic meaning. Second is our belief that greater loss should be incurred when misclassification occurs between features coming from non-neighbor labels than features coming from the same or neighboring labels. In other words, the geometry of learned features, to some extent, reflects the spatial layout of labels. We will show that this new loss function increases the classification performance of the learned prediction model.

Another way of looking at our contributions is to interpret it as a way of structuring the prediction model of a learning machine. Although classical techniques working on data represented in a vector space are capable of learning arbitrary complex decision functions if enough training data and computational resources are available, these strict conditions are not easily achieved. It is therefore often useful to impose some structure on the model. We already mentioned structured models based on energy minimization and their computational disadvantages. Manifold learning is another technique which assumes that the data, although represented in a high dimensional space, is distributed according to a lower dimensional manifold in that space. Semi-supervised learning uses a large amount of additional training data, which is unlabeled, to help the learning machine better infer the structure of the decision function. In this work, we propose to use prior knowledge in the form of the spatial layout of the labels to add structure to the task of learning the decision function.

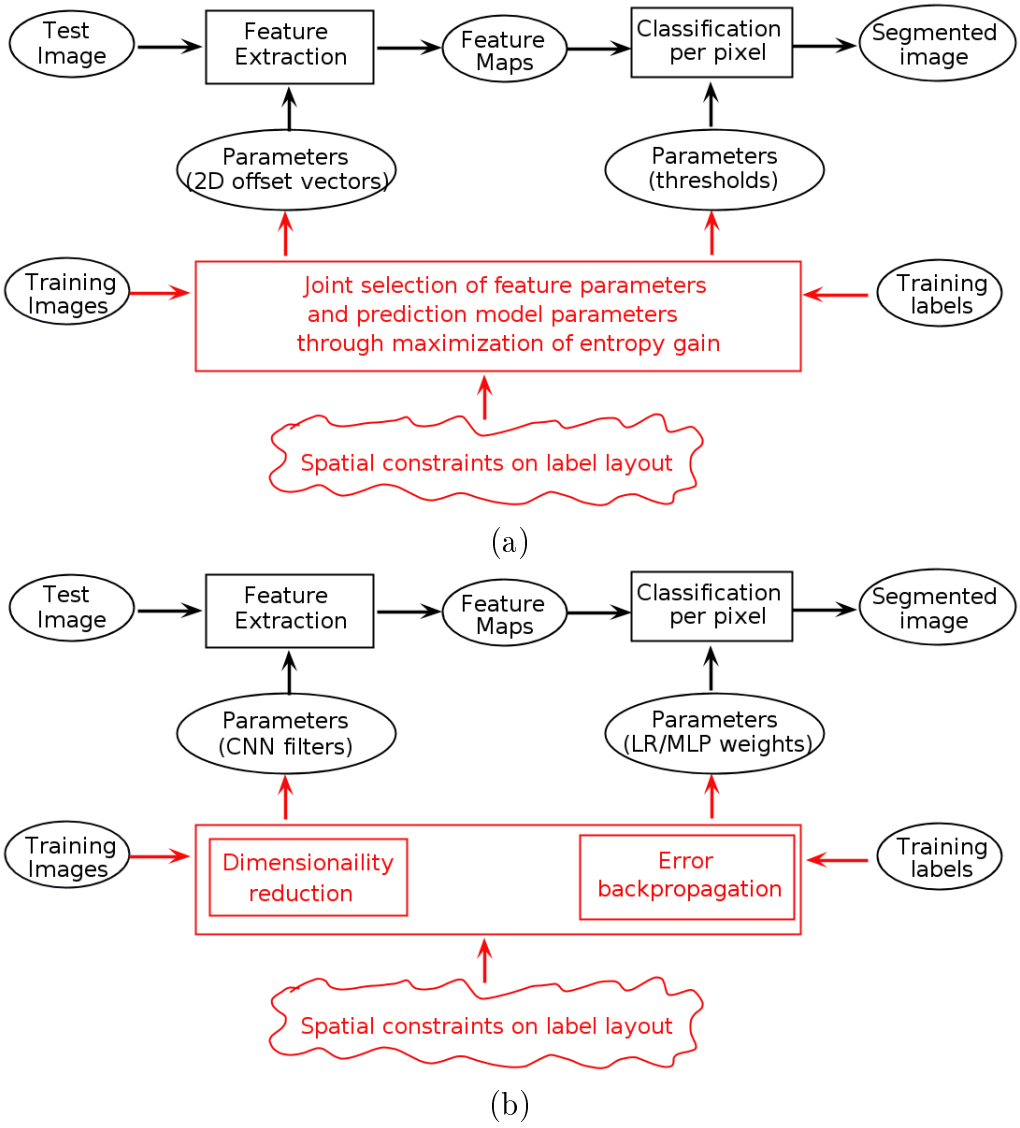


Figure 4.3: Spatial relationships are integrated into different models and training algorithms. (a) RDFs learn all parameters jointly; (b) CNN + LR/MLP, the feature extractor parameters are pre-trained by dimensionality reduction and then learned (fine-tuned) together with the weights of the prediction model.

4.1.1 Related work

Our work takes into account the prior information of object spatial layouts into classification learning. It is partially related to these segmentation problems involving parts, like semantic full scene labeling, and it is related to context based models. There is a general consensus that context plays an important role in the recognition procedure. It provides additional information capable of resolving ambiguity. Several recent work study the role of context in different recognition tasks. Shotton et al. [Shotton et al., 2008b] use bag of semantic textons and region priors in the segmentation, which allows to exploit both texture and semantic context. Heitz and Koller [Heitz and Koller, 2008] propose to combine the objects in the spatial context (e.g. tree, sky) to improve the detection. Divvala and Hoiem [Divvala and Hoiem, 2009] investigate different sources of context and the ways to utilize them to leverage the recognition. However, our methods differ from them in the ways how we treat context in the learning procedure, most existing work extracts features from the context and the object respectively and learns them together, whereas we explicitly model the spatial relationships between labels, not features.

One of our contributions based on ConvNets (see figure 4.3b) also learns a feature extractor from raw data combining the spatial layout of labels, in order to produce a better decision function for segmentation. It is equivalent to learning a mapping function from high dimensional space to a low-dimensional manifold space, which is related to dimensionality reduction.

Unsupervised approaches for learning a mapping capturing global structure are well-known; most notably Principal Component Analysis (PCA) [Jolliffe, 1986] and Multi-Dimensional Scaling [Cox and Cox, 1994]. However, our aim is to embed based on the spatial layout of part labels, so we restrict our discussion to supervised methods. Neighborhood Components Analysis (NCA) [Goldberger et al., 2004] implicitly learn a mapping function from high dimensional space to low dimensional space while preserving the neighborhood relationship defined by class labels. However, NCA is optimized for nearest neighbor classification and does not take into account structure within the labels, only their identity. DrLIM [Hadsell et al., 2006] is an online, non-probabilistic method which explicitly learns a non-linear invariant embedding function. In particular, pairwise distance in feature space are, respectively, minimized or maximized according to label information. More details can be found in section 2.4.3 of chapter 2 . Similarly, we parameterize our embedding with a convolutional neural network (ConvNet) [LeCun et al., 1998], however, like NCA, DrLIM does not consider structure in the labels. Our method differs from NCA and DrLIM by incorporating the spatial layout of labels into an energy function, rather than a binary notion of neighbors defined by data

points with the same label.

Farabet et al. [Farabet et al., 2012] have recently introduced multi-scale ConvNets and applied them successfuly to Full Scene Labeling. Our work is similar in its motivation and the fact that we adopt their multi-scale approach to learn scale-invariant features. However, the way in which we approach the problem is very different. They apply sophisticated machinery based on optimal purity cover to search the best spatial grouping of feature vectors from which to predict a label. Our model has no such notion of adaptive pooling. Instead, we use weakly-supervised learning to introduce spatial context into the features. We believe the two approaches are complimentary, and although beyond the scope of this thesis, could be applied together.

This chapter is organized as following: Section 4.2 introduces the problem formulation. We propose in section 4.3 an algorithm which enforces spatial configuration information of object to one classical learning machine — Randomized Decision Forest. Section 4.4 further introduces another two algorithms having similar spirit into a classical deep network — Convolutional Neural Networks. Section 4.5 gives our partial conclusion on this aspect of work.

4.2 Problem formulation

We consider problems where the pixels i of an image are classified as belonging to one of L target labels by a learning machine whose alphabet is $\mathcal{L} = \{1 \dots L\}$. To this end, descriptors Z_i are extracted on each pixel i and a local patch around it, and the learning machine takes a decision $y_i \in \mathcal{L}$ for each pixel. A powerful prior can be defined over the set of possible labellings for a spatial object. Beyond the classical Potts model known from image restoration [Geman and Geman, 1984], which favors equal labels for neighboring pixels over unequal labels, additional (soft) constraints can be imposed. Labels of neighboring pixels can be supposed to be equal, or at least compatible, i.e. belonging to parts which are neighbors in the spatial layout of the object. If the constraints are supposed to be hard, the resulting problem is a constraint satisfaction problem. In computer vision this kind of constraints is often modeled soft through the energy potentials of a global energy function:

$$E(l_1, \dots, l_N) = \sum_i U(y_i, Z_i) + \mu \sum_{i \sim j} D(y_i, y_j) \tag{4.1}$$

where the unary terms $U(\cdot)$ integrate confidence of a pixelwise employed learning machine and the pairwise terms $D(\cdot, \cdot)$ are over couples of neighbors $i \sim j$.

In the case of certain simple models like the Potts model, the energy function is submodular and the exact solution can be calculated in polynomial time using graph cuts [Kolmogorov and Zabih, 2004]. Taking the spatial layout of the object parts into account results in non-submodular energy functions which are difficult to solve. Let's note that even the complexity of the submodular problem (quadratic on the number of pixels in the worst case) is far beyond the complexity of pixelwise classification with unary terms only.

The goal of our work is to improve the learning machine in the case where it is the only source of information, i.e. no pairwise terms are used for classification. Traditional learning algorithm in this context are supervised and use as only input the training feature vectors Z_i as well as the training labels y_i, where i is over the pixels of the training set. We propose to provide the learning machine with additional information, namely the spatial layout of the labels of the alphabet $\mathcal{L}$. Some pairs of labels are closer than other pairs in that they correspond to neighboring parts. The risk associated for misclassifying a label with a neighboring label should therefore be lower than the risk of misclassifying a label with a not neighboring label.

4.3 Spatial randomized decision forests

In this section we focus on randomized decision forests (RDF) as learning machines, because they have shown to outperform other learning machines in this kind of problem and they have become very popular in computer vision lately [Shotton et al., 2011]. Decision trees, as simple tree structured classifiers with decision and terminal nodes, suffer from over-fitting. Randomized decision forests, on the other hand, overcome this drawback by integrating distributions over several trees.

One of the classical learning algorithm for RDFs [Lepetit et al., 2004] trains each tree separately, layer by layer, which allows the training of deep trees with a complex prediction model. The drawback of this approach is the absence of any gradient on the error during training. Instead, training maximizes the gain in information based on Shannon entropy. In the following we give a short description of the classical training procedure.

We describe the version of the learning algorithm from [Shotton et al., 2011] which jointly learns features and the parameters of the tree, i.e. the thresholds for each decision node. We denote by θ the set of all learned parameters (features and thresholds) for each decision node. We will not explain the specific nature of the features here, the reader is referred to section 5.2.2 in chapter 5 conceiving these details. We only suppose that the features are vectorial, and that one value is used per node of each tree

– a classical assumption in this context. For each tree, a subset of training instances is randomly sampled with replacement.

1. Randomly sample a set of candidates θ for each node. This includes candidates for the threshold τ, and candidates for parameters related to the feature values used for this node.

2. Partition the set of input vectors into two sets, one for the left child and one for the right child according to the threshold $\tau \in \theta$. Denote by Q the label distribution of the parent and by $Q_l(\theta)$ and $Q_r(\theta)$ the label distributions of the left and the right child node, respectively.

3. Choose θ with the largest gain in information:

$$\begin{aligned} \theta^* &= \arg\max_{\theta} \quad G(\theta) \\ &= \arg\max_{\theta} \quad H(Q) - \sum_{s\in\{l,r\}} \frac{|Q_s(\theta)|}{|Q|} H(Q_s(\theta)) \end{aligned} \tag{4.2}$$

   where $H(Q)$ is the Shannon entropy from class distribution of set Q.

4. Recurse the left and right child until the predefined level or largest gain is arrived.

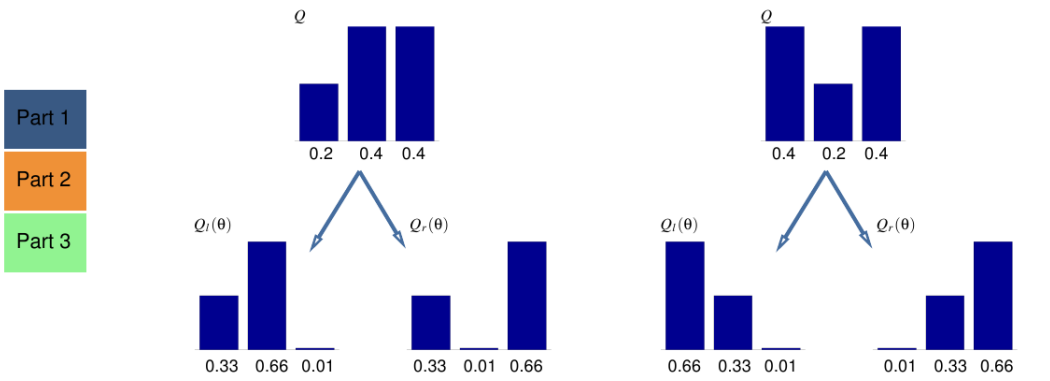
4.3.1 Spatial learning for randomized decision forests

In what follows we will integrate the additional information on the spatial layout of the object parts into the training algorithm, which will be done without using any information on the training error. Let us first recall that the target alphabet of the learning machine is $\mathcal{L} = \{1 \dots L\}$, and then imagine that we create groups of pairs of two labels, giving rise to a new alphabet $\mathcal{L}' = \{11, 12, \dots 1L, 21, 22, \dots 2L, \dots, LL\}$. Each of the new labels is a combination of two original labels. Assuming independent and identically distributed (i.i.d.) original labels, the probability of a new label ij consisting of the pair of original labels i and j is the product of the original probabilities, i.e. $p(ij) = p(i)p(j)$. The Shannon entropy of a distribution Q' over the new alphabet is therefore

$$H(Q') = \sum_{k} -p(k) \log p(k) \tag{4.3}$$

where k is over the new alphabet. This can be expressed in terms of the original distribution over the original alphabet:

$$H(Q') = \sum_{i,j} -p(i)p(j) \log[p(i)p(j)] \tag{4.4}$$



	Classical:	$G(\theta)$=0.37	Classical:	$G(\theta)$=0.37
	Spatial:	$G'(\theta)$=0.22	Spatial:	$G'(\theta)$=0.31
(a)	(b)		(c)	

Figure 4.4: An example of three parts: (a) part layout; (b) a parent distribution and its two child distributions for a given θ; (c) a second more favorable case. The entropy gain for the spatial learning cases are given with $\lambda = 0.3$.

We can now separate the new pairwise labels into two different subsets, the set of neighboring labels $\mathcal{L}'^1$, and the set of not neighboring labels $\mathcal{L}'^2$, with $\mathcal{L}' = \mathcal{L}'^1 \cup \mathcal{L}'^2$ and $\mathcal{L}'^1 \cap \mathcal{L}'^2 = \varnothing$. We suppose that each original label is neighbor of itself. In the same way, a distribution Q' over the new alphabet can be split into two different distributions Q'^1 and Q'^2 from these two subsets.

Then a learning criterion can be defined using the gain in information obtained by parameters θ as a sum over two parts of the histogram Q', each part being calculated over one subset of the labels:

$$G'(\theta) = \lambda\, G'^1(\theta) \quad + \quad (1-\lambda)\, G'^2(\theta) \tag{4.5}$$

where

$$G'^i(\theta) = H(Q'^i) - \sum_{s\in\{l,r\}} \frac{|Q'^i_s(\theta)|}{|Q'^i|} H(Q'^i_s(\theta)) \tag{4.6}$$

Here, λ is a weight, and $\lambda < 0.5$ in order to give separation of non neighboring labels a higher priority.

To be better explain the motivation of this choice, let's consider a simple parts based model with three parts numbered from 1 to 3 shown in figure 4.4a. We suppose that part 1 is a neighbor of 2, that 2 is a neighbor of 3, but that 1 is not a neighbor of 3. Let's also consider two cases where a set of tree parameters θ splits a label distribution Q into two distributions, the left distribution $Q_l(\theta)$ and the right one $Q_r(\theta)$.

The distributions for the two different cases are given in figure 4.4b and 4.4c, respectively. We can see that the child distributions for the second case are permuted versions of the child distributions of the first case. Here, we did not compare the the individual values for Shannon entropy gain between classical measure and spatial measure, as the former is calculated from the unary distribution and the latter on a pairwise distribution. However, the difference between two cases can be observed using the same measure. It can been seen that the classical measure is equal for both cases: the entropy gains are both 0.37. If we take into account the spatial layout of the different parts, we can see that the entropy gain is actually higher in the second case ($G'(\theta)$=0.31) than the first case ($G'(\theta)$=0.22) when setting λ=0.3. In the first case, the highest gain in information is obtained for parts 2 and 3, which are equally probable in the parent distribution Q, whereas a high difference in probability is obtained for the child distributions $Q_l(\theta)$ and $Q_r(\theta)$. However, the highest gain comes from parts 1 and 3 which are not neighbors in the second case. This is consistent with what we advocate above that a higher priority is set to non-neighbor labels.

4.4 Spatial learning in ConvNets

We have proposed an algorithm for integrating spatial information into Randomized Decision Forest in section 4.3. In the recent years, deep neural learning has gained momentum, especially due to some major successes [Krizhevsky et al., 2012]. ConvNets are a successful deep architecture widely studied in various applications, such as object recognition [LeCun et al., 1998, Jarrett and Kavukcuoglu, 2009, Krizhevsky et al., 2012], scene parsing [Grangier et al., 2009, Farabet et al., 2012] and connectomics [Turaga et al., 2010]. By parameter tying and feature pooling, ConvNets can automatically learn shift-invariant, discriminative low- and mid-level features from the raw pixels, avoiding the problem of generalization with hand-engineered features.

Apart the structure classifier learning of the ConvNets, another key aspect of our technique is end-to-end *feature learning*. The dominant methodology in computer vision, though changing in light of recent successes [Krizhevsky et al., 2012], is to extract engineered features such as SIFT [Lowe, 2004] or HOG [Dalal and Triggs, 2005], pool responses, and learn a classifier from this fixed representation. Our objective is to apply learning at all stages of the pipeline, from pixels to labels. However, compared to contemporary Deep Learning approaches, we learn representations that are informed by the spatial structure of the part labels instead of simply their identity.

We are given a set of M images $\{X^1, \ldots, X^M\}$ and their associated labeled groundtruths. In our notation, the pixels of an image are indexed by a linear index: $X^m = \{X^m(i)\}$. We seek to learn a segmentation model consisting of two parts:

- A parametric mapping function $Z = f(X|\theta_f)$ which embeds each image X to a feature representation. This representation consists of Q *maps*, each of which have the same dimensions as X, and therefore can be indexed linearly: $Z(i) \in \mathbb{R}^Q$. The parameters θ_f are learned from training data, taking into account Euclidean distances of pairs of features in the embedding space: $d(i,j) = ||Z(i) - Z(j)||_2$ (see Section 4.4.1).
- A classifier $\hat{y}(i) = g(Z(i)|\theta_g)$ which classifies the features $Z(i)$ given trained parameters θ_g giving an estimate $\hat{y}(i)$ of the part label (see Section 4.4.2).

These two parts are illustrated in figure 4.3b. As is common in the Deep Learning literature, the embedding can be pre-trained in an unsupervised [Hinton and Salakhutdinov, 2006] or supervised [Salakhutdinov and Hinton, 2007] manner based on auto-encoding or some other kind of inductive principle. Then, the classifier and the embedding are jointly learned in a supervised way by minimizing some classification-based loss[1]. Our method proceeds in a similar way. We assume that the spatial part layout remains consistent across the different images of a corpus. In particular, adjacency information of parts is assumed not to vary across images. In body part estimation, for example, we suppose that the upper arm and the forearm are always adjacent. The next two subsections will describe how spatial constraints can be integrated into, respectively, the training procedure for the embedding $f(\cdot)$, as well as the training procedure for the classifier $g(\cdot)$. The two contributions can be applied independently, or combined. The result is a method proceeding as illustrated in Figure 4.1c: the information on the neighborhood layout is injected into a classifier working independently for each pixel i.

4.4.1 Spatial pre-training

Common ways of learning the embedding function $f(\cdot)$ are to minimize reconstruction error in an unsupervised auto-encoding setting [Hinton and Salakhutdinov, 2006], or in a supervised setting, map pixels with the same labels to close-by points in feature space and pixels with different

[1] If pre-training is supervised, the common setting is to perform classification by k-nearest neighbor search in embedded space, requiring no additional learning step.

(a) (b) (c)

Figure 4.5: Three spatial relations: (a) shows two pixels from the same label: i.e. $\delta_{a,b} = 1$; (b) shows two pixels from neighboring labels: i.e. $\nu_{a,b} = 0$; (c) shows two pixels from non-neighboring labels: i.e. $\nu_{a,b} = 1$.

labels to distant points [Salakhutdinov and Hinton, 2007, Hadsell et al., 2006, Taylor et al., 2011b].

Given the availability of ground-truth data (i.e. the part labels), we advocate its use in the pre-training step as supervised or semi-supervised training has been shown to produce a more discriminative embedding [Salakhutdinov and Hinton, 2007]. However, instead of using classification loss as a training criteria, we aim to benefit from the knowledge of the neighborhood layout of the parts in the corpus. We introduce a new energy function which exploits the spatial layout of different parts:

$$E = \sum_{i,j} \delta_{y(i),y(j)} L_S(i,j) \; + \; \sum_{i,j} \nu_{y(i),y(j)} L_D(i,j) \tag{4.7}$$

where $y(i)$ and $y(j)$ are the ground truth labels for pixels indexed by i and j, respectively; $\delta_{a,b}$ is the Kronecker delta defined as $\delta_{a,b}$=1 if a=b and 0 otherwise; $\nu_{a,b}$ is defined as $\nu_{a,b}$=1 if parts a and b are *not* neighbors in the corpus, and 0 otherwise. Figure 4.5 shows three spatial relations for pairwise pixels. L_S and L_D are measured between pairs of features indexed by i and j in the feature space, which is embedded from raw data X through the embedding function. Note that we simply ignore contributions from pairs which have different labels but whose respective parts are neighbors in the corpus. As the exact boundaries between neighbors (given a human annotator) are somewhat arbitrary and often noisy, we choose to remain agnostic toward the relationship between features across a single boundary. We have chosen this scheme for simplicity, though we admit there are many other ways to design an energy function based on preserving spatial relationships. For example, if we could quantify the distance between labels (e.g. mean distance between body parts), such a "soft" neighbour criterion could be incorporated into the energy function. This is reserved for future work.

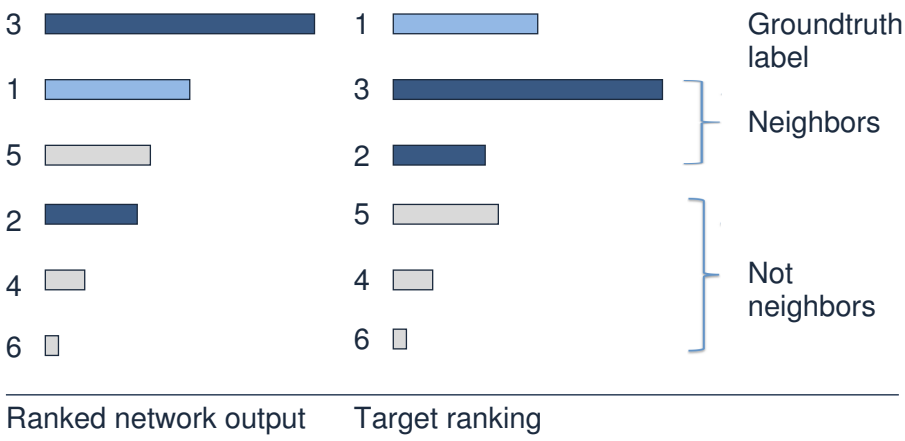


Figure 4.6: The proposed loss function based on differences in ranking.

L_S is the loss component for a pair of pixels having the same label, which has the effect of pushing their associated embeddings together. L_D is a contrastive loss component for a pair having different labels which pulls their embeddings apart. Similar to [Hadsell et al., 2006], L_S and L_D are set to

$$L_S(i,j) = \frac{1}{2}(d(i,j))^2, \tag{4.8}$$

$$L_D(i,j) = \frac{1}{2}[\max(0, \alpha - d(i,j)]^2, \tag{4.9}$$

where $\alpha > 0$ is a margin. It controls the contrastive scope such that only dissimilar inputs which lie close-by in the embedded space contribute to the energy.

The gradient of Equation (4.7) with respect to the parameters can be computed via the chain rule (backpropagation):

$$\frac{\partial E}{\partial \theta_f} = \frac{\partial E}{\partial Z(i)}\frac{\partial Z(i)}{\partial \theta_f} + \frac{\partial E}{\partial Z(j)}\frac{\partial Z(j)}{\partial \theta_f} \tag{4.10}$$

where $\frac{\partial E}{\partial Z(i)}$ and $\frac{\partial E}{\partial Z(j)}$ can be obtained from (4.8) and (4.9). $\frac{\partial Z(i)}{\partial \theta_f}$ and $\frac{\partial Z(j)}{\partial \theta_f}$ depend on the particular architecture of the embedding function $f(\cdot)$ which is described in Section 4.4.3.

We finally want to point out that, although the energy function is defined on pairwise pixel terms, testing is done pixelwise, so testing does not require solving a combinatorial problem.

4.4.2 Supervised spatial LR learning

In Section 4.4.1, we introduced spatial constraints into unsupervised pre-training of deep learning architectures, which are ConvNets in our setting

(see Section 4.4.3). In this section we show that a similar principle can also be applied to supervised learning of classifiers. In our work, we choose a ConvNet with a single fully-connected classification layer (i.e. the top layer is equivalent to logistic regression) which are trained end-to-end. The same principle, introducing spatial relationships into pixelwise classification, can also be applied to other classifiers, although the method may differ if the classifier is not learned to minimize classification error (or something akin to it, such as cross-entropy). In particular, our method includes spatial constraints into learning randomized decision forests by adapting the training algorithm based on maximizing gain in entropy (see section 4.3).

Classification-based neural nets are typically trained to minimize cross-entropy. When the normalized outputs of the net are viewed as probabilities, this is equivalent to maximizing the log probability the net assigns to the true class. In the multi-class setting, this involves normalizing the outputs of the network via a softmax function and comparing them to the groundtruth label. However, minimizing cross-entropy does not take into account the layout of the part labels.

We propose the following new loss function, which is based on the ranking of class labels according to network output. For each input vector, a forward pass gives a network response for each class label, which can be used to rank the class labels in decreasing order. A loss can be defined based on the difference between this ranking and a desired target ranking, which is defined by the following properties:

- The highest ranked class label should be the target groundtruth label. This constraint is related to the entropy loss in traditional neural network learning;
- The next highest-ranked class labels should be neighbors of the groundtruth label in the class neighborhood definition of the corpus. We advocate that better generalization to unseen data can be achieved by forcing the net to learn these constraints.

An example for this is given in Figure 4.6, where the groundtruth label for the pixel is 1. The actual output ranks the groundtruth label at second place. The target ranking ranks groundtruth label 1 at first place, followed by labels 3 and 2 which, in this example, are neighbors of label 1.

Learning to rank is a classical problem in machine learning which has been addressed in the literature [Burges et al., 2005, Freund et al., 2003, Dekel et al., 2004]. We adopt a loss function similar in spirit to *RankNet* [Burges et al., 2005], defined on pairwise constraints. Given a pair of labels (u, v), we denote by $g_u(Z(i))$ and $g_v(Z(i))$ the respective topmost (i.e. classification) layer outputs for location i and by $o_{uv}(i) = g_u(Z(i)) - g_v(Z(i))$ their

difference. The probability of label u being ranked higher than v is mapped through a logistic function:

$$P_{uv}(i) = \frac{e^{o_{uv}(i)}}{1 + e^{o_{uv}(i)}} \tag{4.11}$$

Given a target probability $\bar{P}_{uv}(i)$, the loss function $C_{uv}(i)$ is the cross entropy loss [Burges et al., 2005]:

$$C_{uv}(i) = -\bar{P}_{uv}(i) \log P_{uv}(i) - \left(1 - \bar{P}_{uv}(i)\right) \log \left(1 - P_{uv}(i)\right) \tag{4.12}$$

The target probability $\bar{P}_{uv}(i)$ is set to $\lambda{>}0.5$ if class u is ranked higher in the desired ranking, and to $1{-}\lambda$ otherwise.

Given the properties of the desired ranking described above, the following two sets of pairwise constraints have been derived:

1. A set of $L - 1$ constraints, where each constraint specifies that the groundtruth label is to be ranked higher than one of the other labels;

2. A set of constraints each one specifying that a label u, which is a neighbor of the groundtruth label, should be ranked higher than another label v, which is not a neighbor of the groundtruth label.

The loss function $\mathcal{E}$ for a single pixel is the sum over the pairwise constraints of the pairwise loss C_{uv}:

$$\mathcal{E}(i; l) = \sum_{u \neq l} C_{lu}(i) + \sum_{(u,v):\nu_{u,l}=0,\nu_{v,l}=1} C_{uv}(i) \tag{4.13}$$

where i is the index of the pixel, and we use the shortform $l = y(i)$ as its ground truth label.

This loss function based on rankings provides a principled way of combining classification loss and spatial layout. Two types of constraints can be set through the λ parameter controlling the target probability $\bar{P}_{uv}$. Varying this parameter, different priorities could be given to different constraints.

It is important to note that this formulation allows one to include unlabeled data into the learning procedure in a semi-supervised setting. In this case, labeled images will be presented to the network with the loss function described above, whereas the loss function for unlabeled data does not contain the pairwise constraints including the groundtruth label - we consider *neighbours of the network's strongest output* instead of the ground truth neighbours.

As in Section 4.4.1, we would like to avoid confusion created by the term "pairwise", which here involves pairs of labels selected during training, which typically are of low cardinality compared to pixels. Testing proceeds independently for each pixel, conforming to the objectives of our work.

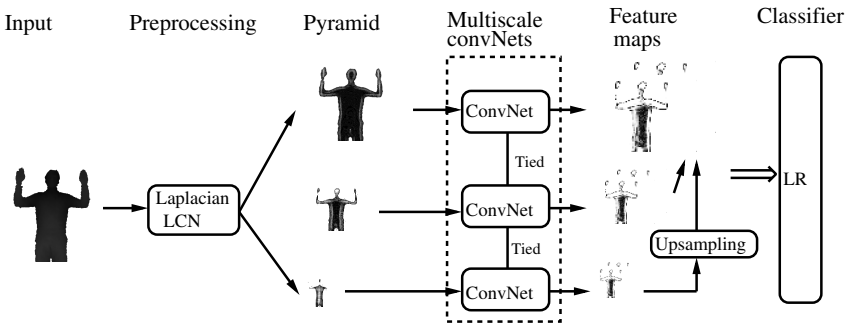


Figure 4.7: Multiscale ConvNets framework [Farabet et al., 2012]. LCN means local contrast normalization. Each ConvNet contains several layers of convolutions and pooling described in 4.4.3.

4.4.3 The deep learning architecture

While our mapping $f(\cdot)$ can be any differentiable architecture, we adopt a two-stage ConvNet [LeCun et al., 1998]. Each stage consists of a convolutional layer with a bank of filters, an element-wise non-linear squashing mapping (i.e. tanh function), followed by a spatial pooling and subsampling operator. A bank of feature maps are produced after convolving with the filters. The output of the first stage is fed as input to the convolutional layer at the second stage, where each feature map is connected to several maps in the previous stage. Connectivity is chosen uniform randomly before learning begins. The gradient of the energy function is computed with respect to the network output, and backpropagation is applied to update the parameters θ_f. The parameters θ_f consist of the filters and element-wise additive and multiplicative biases at the feature maps.

Many visual recognition problems require a large context due to complex interactions of different parts. However, the contextual size of ConvNets is limited by the filter size and sampling rate. Simply selecting larger filters does not address the fact that important cues may be observed at different scales. To overcome this dilemma, multiscale ConvNets [Farabet et al., 2012] are employed in our architecture, as shown in Figure 4.7. We use the notation $f^s(\theta_f^s), \forall s \in \{1, \ldots, N\}$ to denote the output produced by each scale ConvNet where s indexes the scales and N is the total number of scales. By employing the weight sharing across scales, ConvNets can learn scale-invariant features.

A multiscale Laplacian pyramid $X^s, \forall s \in \{1, \ldots, N\}$ is constructed for each image X in the pre-processing stage, where X^1 has the same size as the original image. Local contrast normalization is applied to the Laplacian images to ensure that the local neighborhood has zero mean and standard

deviation. A batch consisting of pairs of patches[2] is randomly extracted from each scale image X^s. The patches are processed by the corresponding ConvNet f^s, where the learned parameters θ_f^s are shared across the scales.

The training procedure of the above architecture contains two steps. The first step is the spatial deep learning described in section 4.4.1, where the labels are only used to define spatial layout. We therefore call this stage weakly supervised pre-training. At this stage, the ConvNet parameters are initialized such that the features are consistent with the spatial arrangement of the labels. The second step is supervised spatial learning in section 4.4.2. A logistic regression layer parameterized with θ_g is connected to the topmost feature maps of the ConvNet to predict the labels. We also apply a fine-tuning scheme in which the gradients from the LR are back-propagated to update the ConvNet parameters θ_f.

4.5 Conclusion

In this chapter, we discussed the integration of spatial information on part layout into machine learning. We proposed three different algorithms, one for randomized decision forest and two for a deep architecture. Here we surmise that a learning machine can be improved when it is given *a priori* information on the object structure.

In next chapter, we will demonstrate one application of human body part estimation by employing the proposed algorithms. It is worth noting that our algorithms can not only be used for human body part estimation, moreover, they could be applied to any object segmentation problem with an apparent spatial configuration.

[2]Since our loss function operates on pairs of features, i.e. particular locations i and j in the feature maps, we do not perform a full convolution while training. Instead, we extract patches corresponding to the receptive fields at locations i and j and map these patches to their respective feature vectors.

Chapter 5

Human body segmentation

Contents

5.1 Introduction

This chapter discusses one application of the algorithms introduced in chapter 4 to human body estimation for depth images. This kind of technique is frequently employed as a preliminary step of human pose estimation. Many applications associated with human-computer interaction depend on human pose estimation.

The proposed algorithm has been evaluated on the *CDC4CV Poselets* dataset [Holt et al., 2011]. Our goal is not to beat the state of the art in pose estimation, but to show that spatial learning is able to improve pixel-wise classification of parts based models. The dataset contains upper body poses taken with Kinect and consists of 345 training and 347 test depth images. The authors also supplied corresponding annotation files which contain the locations of 10 articulated parts: head(H), neck(N), left shoulder(LS), right shoulder(RS), left elbow(LE), left hand(LHA), right elbow(RE), right hand(RHA), left hip(LH), right hip(RH). We created groundtruth segmentations through nearest neighbor labeling. In our experiments, the left/right

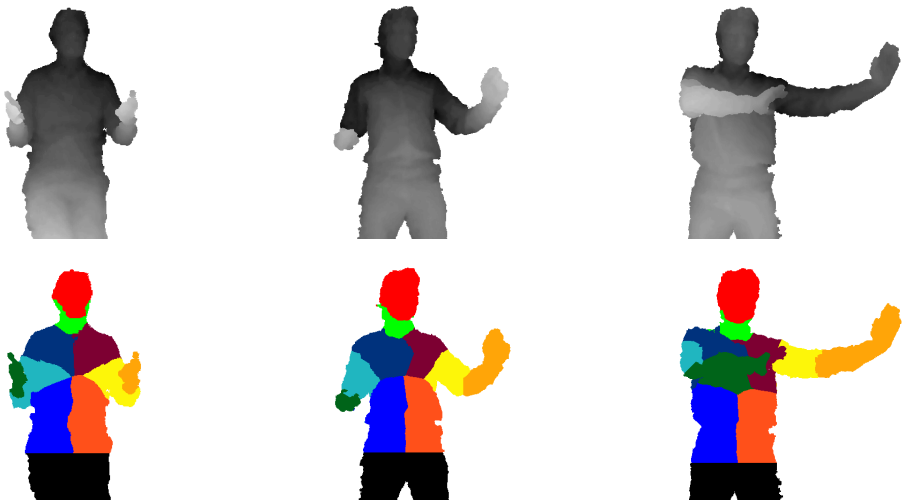


Figure 5.1: The depth images (the first row) and their groundtruth (the second row) in CDC4CV Poselets dataset. [Holt et al., 2011].

elbow (LE,RE) and hand (LHA,RHA) parts were extended to left/right upper arm (LUA,RUA) and forearm (LFA, RFA) parts, we also defined the part below the waist as other. Figure 5.1 shows several examples and their groundtruth.

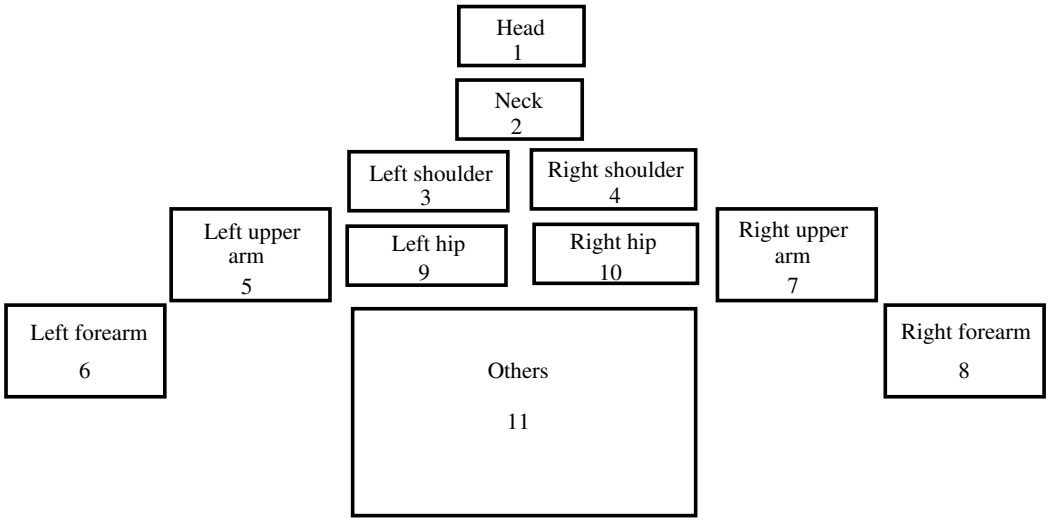
We define a spatial layout of 28 neighborhood relations among the 11 parts calculated from a canonical pose where a subject stretches his or her arms. The spatial layout of human body and part adjacency matrix are shown in figure 5.2. It is worth noting that the spatial relationship matrix is symmetrical. Our spatial relationships are very simple, they are expressed as binary values. Soft relationships may be one possible extension, which could take into account part occlusion in real images. Further research can go to this direction. We adopt the same spatial relationships in the following experiments, unless otherwise specified.

This chapter will discuss two sets of experiments:

- Section 5.2 presents experiments using spatial learning for randomized decision forest, introduced in section 4.3 of chapter 4. This part is completed by an additional contribution, namely edge features which complement existing depth features, and whose parameters can be learned easily as the latter ones.

- Section 5.3 describes experiments using spatial learning with ConvNets, as introduced in section 4.4 of chapter 4.

- Section 5.4 gives our discussion and conclusion.



(a)

Label \ Label	1	2	3	4	5	6	7	8	9	10	11
1	1	1	0	0	0	0	0	0	0	0	0
2	1	1	1	1	0	0	0	0	0	0	0
3	0	1	1	1	1	0	0	0	1	1	0
4	0	1	1	1	0	0	1	0	1	1	0
5	0	0	1	0	1	1	0	0	1	0	0
6	0	0	0	0	1	1	0	0	0	0	0
7	0	0	0	1	0	0	1	1	0	1	0
8	0	0	0	0	0	0	1	1	0	0	0
9	0	0	1	1	1	0	0	0	1	1	1
10	0	0	1	1	0	0	1	0	1	1	1
11	0	0	0	0	0	0	0	0	1	1	1

(b)

Figure 5.2: A spatial layout for the human body: (a) An illustration of human upper body model, (b) An adjacency matrix showing the spatial relationships. 1 indicates the relation of neighbors, otherwise non-neighbors.

5.2 Experiments for spatial randomized decision forest

Depth images are different from RGB images with different properties. There is no texture, which makes it easier to extract shape information. On the other hand, classical appearance features such as HOG/HOF, SIFT etc. are not optimal for this kind of images. Shotton et al. [Shotton et al., 2011]

propose a depth comparison feature, which uses a set of depth differences around the given pixel as features. At the same time, they take advantage of randomized decision forest to learn the best offsets for nodes, which are involved in feature selection. We first present depth comparison features in section 5.2.1 and then introduce our complementary edge features in section 5.2.2.

5.2.1 Depth features

In [Shotton et al., 2011], depth features have been proposed for pose estimation from Kinect depth images. One of their main advantages is their simplicity and their computational efficiency. Briefly, at a given pixel x, the depth difference between two offsets centered at x is computed:

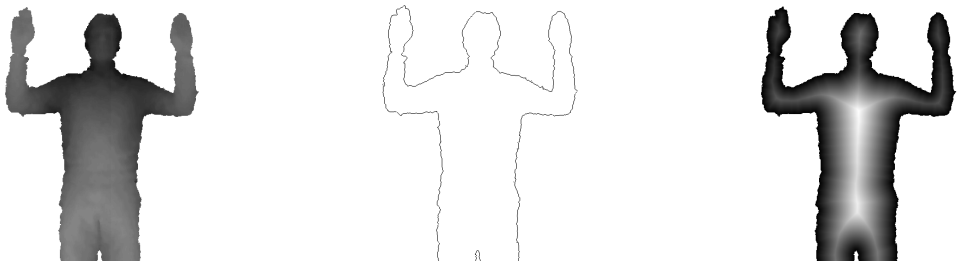
$$f_\theta(I, \mathbf{x}) = d_I(\mathbf{x} + \frac{\mathbf{u}}{d_I(\mathbf{x})}) - d_I(\mathbf{x} + \frac{\mathbf{v}}{d_I(\mathbf{x})}) \tag{5.1}$$

where $d_I(\mathbf{x})$ is the depth at pixel $\mathbf{x}$ in image I, parameters $\theta = (\mathbf{u}, \mathbf{v})$ are two offsets which are normalized by the current depth for depth-invariance. A single feature vector contains several differences, each comparison value being calculated from a different pair of offsets $\mathbf{u}$ and $\mathbf{v}$. These offsets are learned during training together with the prediction model, as described in section 4.3 of chapter 4.

5.2.2 Edge features

The depth comparison features described above capture simple differences between pairs of neighboring pixels. One weakness is their inability to capture richer shape information. Here we extend this concept further by introducing edge comparison features extracted from edges. We propose two different types of features based on edges, the first using edge magnitude, and the second using edge orientation. Our features capture slightly richer information, such as differences in distances to part borders, and differences in part orientation.

In principle, these features can be extracted from grayscale, RGB or depth images, where an extraction from grayscale or RGB requires to solve a correspondence problem between RGB and depth coordinate systems, if the edge features are to be combined with the existing depth features. This can be done easily with calibrated Kinect sensors. In our experiments we calculated edges from the depth images, mainly because the used dataset did not provide RGB data. As will can be seen, even edge information from depth images significantly improves performance.



(a) (b) (c)

Figure 5.3: Illustration of edge features: (a) an input depth image; (b) an edge map E; (c) the distance transform DT_E.

In our settings, we need features whose positions can be sampled by the training algorithm. However, contours are usually sparsely distributed, which means that comparison features can not directly be applied to edge images. Our solution to this problem is inspired by chamfer distance matching, which is a classical method to measure the similarity between contours [Barrow et al., 1977]. We compute a distance transform on the edge image, where the value of each pixel is the distance to its nearest edge. Given a grayscale image I and its binary edge image E, the distance transform DT_E is computed as:

$$DT_E(\mathbf{x}) = \min_{\mathbf{x}':E(\mathbf{x}')=1} ||\mathbf{x} - \mathbf{x}'|| \tag{5.2}$$

The distance transform can be calculated in linear time using a two-pass algorithm. Figure 5.3 shows a depth image and its edge image and distance image.

The edge magnitude feature is defined as:

$$f_\theta^{EM}(\mathbf{x}) = DT_E(\mathbf{x} + \mathbf{u}) + DT_E(\mathbf{x} + \mathbf{v}) \tag{5.3}$$

where $\mathbf{u}$ and $\mathbf{v}$ are offsets similar to the ones in equation 5.1. This feature indicates the existence of edges near two offsets.

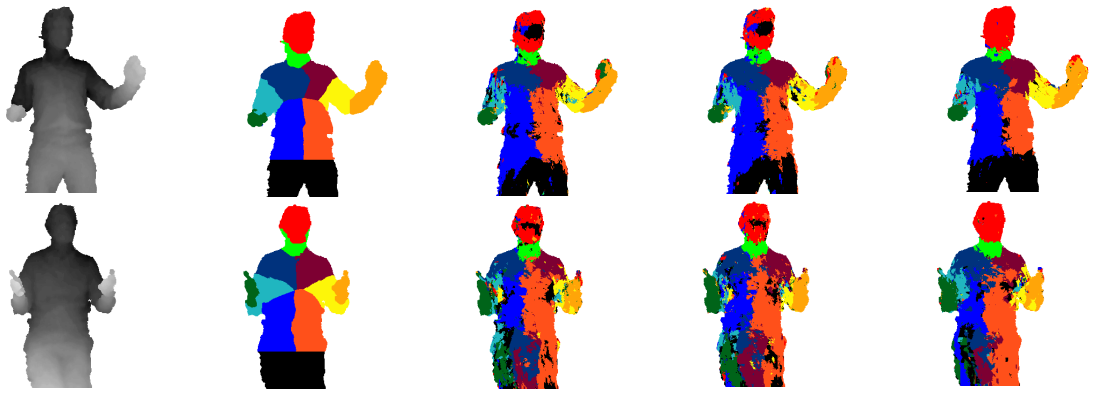
Edge orientation features can be computed in a similar way. In the procedure of distance image, we can get another orientation image O_E, in which the value of each pixel is the orientation of its nearest edge:

$$O_E(\mathbf{x}) = Orientation\left(\arg \min_{\mathbf{x}':E(\mathbf{x}')=1} ||\mathbf{x} - \mathbf{x}'||\right) \tag{5.4}$$

The feature is computed as the difference in orientation for two offsets:

$$f_\theta^{EO}(\mathbf{x}) = O_E(\mathbf{x} + \mathbf{u}) - O_E(\mathbf{x} + \mathbf{v}) \tag{5.5}$$

where the minus operator takes into account the circular nature of angles. We discretize the orientation to alleviate the effect of noise.



(a) (b) (c) (d) (e)

Figure 5.4: Examples of the pixelwise classification: each row is an example, each column is a kind of classification results, (a) test depth image; (b) part segmentation; (c) classical classification; (d) spatial learning with depth features; (e) spatial learning with depth features and edge features.

The objective of both features is to capture the edge distribution at specific locations in the image, which will be learned by the RDF.

5.2.3 Results

Unless otherwise specified, the following parameters have been used for RDF learning: 3 trees each with a depth of 9; 2000 randomly selected pixels per image, randomly distributed over the body; 4000 candidate pairs of offsets; 22 candidate thresholds; offsets and thresholds have been learned separately for each node in the forest. For spatial learning, 28 pairs of neighbors have been identified between the 11 parts based on a pose where the subject stretches his arms. The parameter λ was set to 0.4.

During learning, depth features are mixed with edge features. More precisely, for each tree node, the learning algorithm selects one of the two features. This choice is integrated into the learning algorithm described in section 4.3 of chapter 4 in the same way as training of all other parameters (offsets $\mathbf{u}$ and $\mathbf{v}$ and threshold τ): it is learned to maximize the gain in entropy, i.e. maximizing equation 4.6.

We evaluate our method at two levels: pixelwise classification error and part recognition and localization. Pixelwise decisions are directly provided by the random forest. Part localizations are obtained from the pixelwise results through pixel pooling. We create a posterior probability map for each part from the classification results: After non-maximum suppression and low pass filtering, the location with largest response is used as an estimate of

Table 5.1: Results on body part classification in pixelwise level.

		Accuracy
Classical RDF with Depth features [Shotton et al., 2011]		60.30%
Spatial RDF with Depth features	$\lambda = 0.4$	61.05%
Spatial RDF with Depth + Edge features	$\lambda = 0.4$	67.66%

the part. Then the positions of pairs of detected parts are calculated, which approximately correspond to joints and serve as an intermediate pose indicator [Ferrari et al., 2008]. In the following, we denote them by the pair of neighboring parts.

Table 5.1 shows classification accuracies of three different settings. A baseline has been created with classical RDF learning and depth features. Spatial learning with depth features only and with depth and edge features together are also shown in table 5.1. We can see that that spatial learning can obtain a performance gain, although no pairwise terms have been used during testing. Figure 5.4 shows some classification examples, which demonstrate that spatial learning makes the randomized forest more discriminative. The segmentation output is cleaner, especially at the borders.

At part level, we report our results of pairs of parts according to the estimation metric by [Ferrari et al., 2008]: a pair of groundtruth parts is matched to a detected pair if and only if the endpoints of the detected pairs lie within a circle of radius r=50% of the length of the groundtruth pair and centered on it. Table 5.2 shows our results on part level using several settings. It demonstrates that spatial learning improves recognition performance for most of the parts.

The experiments at both pixelwise and part level demonstrate that spatial learning makes randomized decision forests more discriminative by integrating the spatial layout into its prediction model. This proposition is very simple and fast to implement, as the standard pipeline can be still used. Only the learning method has been changed, the testing code is unchanged.

5.3 Experiments for spatial ConvNets

In this section, we present experiments on the same dataset using a combination of ConvNets for feature learning and logistical regression (LR) for classification.

Unless otherwise specified, we use a 2-stage ConvNet in all of the experiments below. The first convolutional layer consists of 8 filters of size 9×9, followed by an element-wise tanh function and a 2×2 non-overlapping average

Table 5.2: Correct rate(%) on pairs of parts for different feature settings in part level: D=deph features; E=edge features.

		H-N	LS-RS	LS-LUA	LUA-LFA	RS-RUA	RUA-RFA	LS-LH	RS-RH	LH-RH	Ave.
Classical E		46.69	0.29	20.46	2.02	0	21.90	77.81	1.15	19.88	21.13
Spatial E	$\lambda = 0.4$	52.45	0	25.65	1.44	0	14.41	82.13	0.86	22.48	22.16
Classical D		85.30	0.29	39.77	1.15	0.29	17.58	49.86	0.29	16.14	23.41
Spatial D	$\lambda = 0.4$	88.47	1.15	34.58	0.28	0.28	10.66	67.72	5.18	28.24	26.29
Spatial D+E	$\lambda = 0.4$	89.05	0	53.31	0.86	0	25.65	72.91	0	13.54	28.37

pooling operator. The second convolutional layer consists of 32 filters of size 9×9, each of which combines 4 feature maps of the previous layer, with the same activation and pooling operator. The output is a set of 32 feature maps. The receptive field for a single ConvNet on a single scale is 28×28. We resort to a multiscale setting with a pyramid of 3 scales, with each scale consisting of 2 convolutional layers [Farabet et al., 2012]. The total receptive field is of size 112×112, thus a large spatial context is used to learn better features, comparable to a best probe distance of 120×120 of the randomized decision forest described earlier.

Local contrast normalization is applied to the inputs of the ConvNet, and the pixels from the background or the context are set to an arbitrary high value (in our case, 4) to distinguish from the zero-distributed pixels of the object. The margin α for pre-training was set to 1.25 [Hadsell et al., 2006], and λ was set constant to 1 for all constraints in all experiments, i.e. $\bar{P}_{uv} = 1$ if u is ranked higher than v, otherwise 0. Weakly-supervised feature learning and supervised learning each used 30 epochs through the entire dataset, where mini-batch gradient descent is adopted. End-to-end fine-tuning, if applied, used another 30 epochs. Different learning hyper-parameters (such as learning rates for different layers) were chosen empirically, e.g. ε_1 is set to 10^{-6} for the first convolutional layer and ε_2 to 10^{-5} for the second convolutional layer. When performing classification, we consider each pixel in isolation, applying its corresponding 32-dimensional feature vector as input to an LR. The parameters of the LR are shared at each pixel location. In the following experiments, we report mean pixel-wise accuracy.

5.3.1 Results

For the application of part estimation from depth images, our baselines are the algorithms based on randomized decision forests described in the last section, as well as single- and multi-scale ConvNets with supervised end-to-end training without any spatial context. From table 5.3, which gives performance results on 4 different baseline algorithms, it is clear that the large receptive field of the multiscale architecture is necessary to reliably estimate body part labels, and also that the Deep Learning-based architecture outperforms random forests on the CDC4CV dataset.

We investigate the performance of different combinations of our framework based on spatial pre-training of the purely convolutional layers (which we call ConvNet) and the classification layer (which we call LR). As a non-spatial baseline, we also implemented a "pixel-wise" version of DrLIM [Hadsell et al., 2006], a similar pre-training strategy in which spatial layout is not taken into consideration, i.e. $\delta_{a,b}$=1 if a=b and 0 otherwise; $\nu_{a,b}$=1 if

Table 5.3: Evaluation of different baselines on the CDC4CV dataset. In our implementation of [Farabet et al., 2012], only the multiscale ConvNets have been used. Purity trees and CRFs have not been implemented, to make the work comparable to the rest of this chapter.

Methods	Accuracy
Randomized forest [Shotton et al., 2011]	60.30%
Spatial RDF (Section 4.3 of chapter 4)	61.05%
Single-scale (vanilla) ConvNet [LeCun et al., 1998]	47.17%
Multi-scale ConvNet [Farabet et al., 2012]	62.54%

$a \neq b$ and 0 otherwise in equation 4.7. The results are shown in Table 5.4. For each setting (fine-tuning or no fine-tuning), spatial training outperforms non spatial training, and in many cases, the gains are high. Fine-tuning means end-to-end training of the LR (top two layers) and ConvNet (remaining layers) with the same objective used to train the LR.

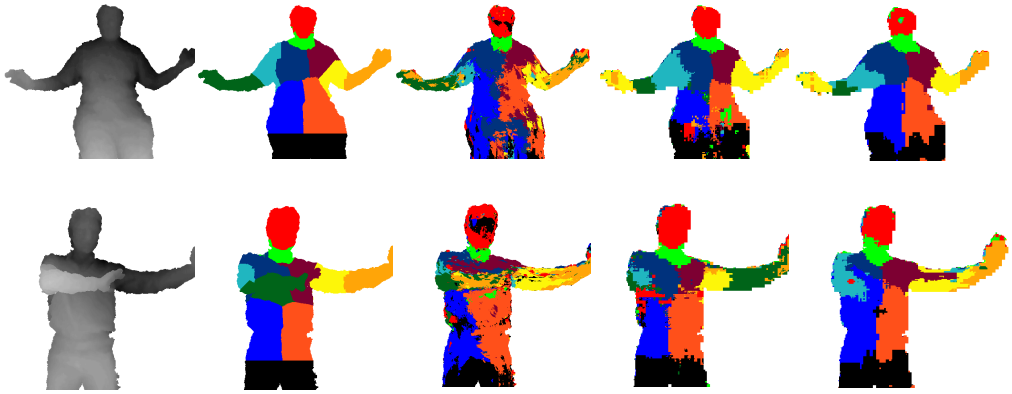
Table 5.4: Results of different combinations of classical and spatial learning on the CDC4CV dataset.

Convolutional layers	LR	Fine-tuning	Accuracy
DrLIM [Hadsell et al., 2006]	classical	no	35.10%
DrLIM [Hadsell et al., 2006]	spatial	no	41.05%
spatial	classical	no	38.60%
spatial	spatial	no	**41.65%**
DrLIM [Hadsell et al., 2006]	classical	yes	64.39%
DrLIM [Hadsell et al., 2006]	spatial	yes	65.12%
spatial	classical	yes	65.18%
spatial	spatial	yes	**66.92%**

Examples of segmentation results are shown in Figure 5.5. From visual inspection we can see that the segmentation results produced by spatial learning are better than the ones by the non spatial methods. In particular, the segmentation produced by spatial learning is more consistent (less noisy), especially for the arms.

5.3.2 Spatial distribution of the error

The proposed method injects neighborhood relationships into the training algorithm. The question arises, whether the benefit of the proposed method



(a) (b) (c) (d) (e)

Figure 5.5: Classification examples from the CDC4CV dataset. (a) input depth image; (b) groundtruth segmentation; (c) appropriate baseline: randomized forest for CDC4CV; (d) DrLIM+LR without spatial learning; (e) our method (spatial pre-training and spatial LR learning).

only applies to pixels at the boundaries, which would imply a rather trivial improvement. To rule this out, an in-depth analysis has been performed to verify that the proposed methods generally improve classification error independent of the pixels' positions in the part.

For each pixel in the test images the distance to the nearest part boundary was calculated using a distance transform, which allowed us to calculate histograms over these distances, where each bin corresponds to a range of distances. Distribution H_g gives the number of pixels of the test images over distances, shown in Figure 5.6(a). This histogram largely depends on the sizes of the different body parts, and we can see that the pixel count decreases with distance. Distributions H_s and H_b are over pixels which are better classified by the baseline or by the proposed method, respectively:

- Distribution H_s, shown in Figure 5.6(b), is the distribution of pixels which have been correctly classified by the proposed method but wrongly classified by the baseline method (DrLIM and classical LR with fine-tuning);
- Distribution H_b, shown in Figure 5.6(c), is the distribution of pixels which have been correctly classified by the baseline method (DrLIM and classical LR with fine-tuning) but wrongly classified by the proposed method.

The normalized histogram H_d, shown in Figure 5.6(d), illustrates the contribution of the proposed spatial learning method as a function of the the pixels'

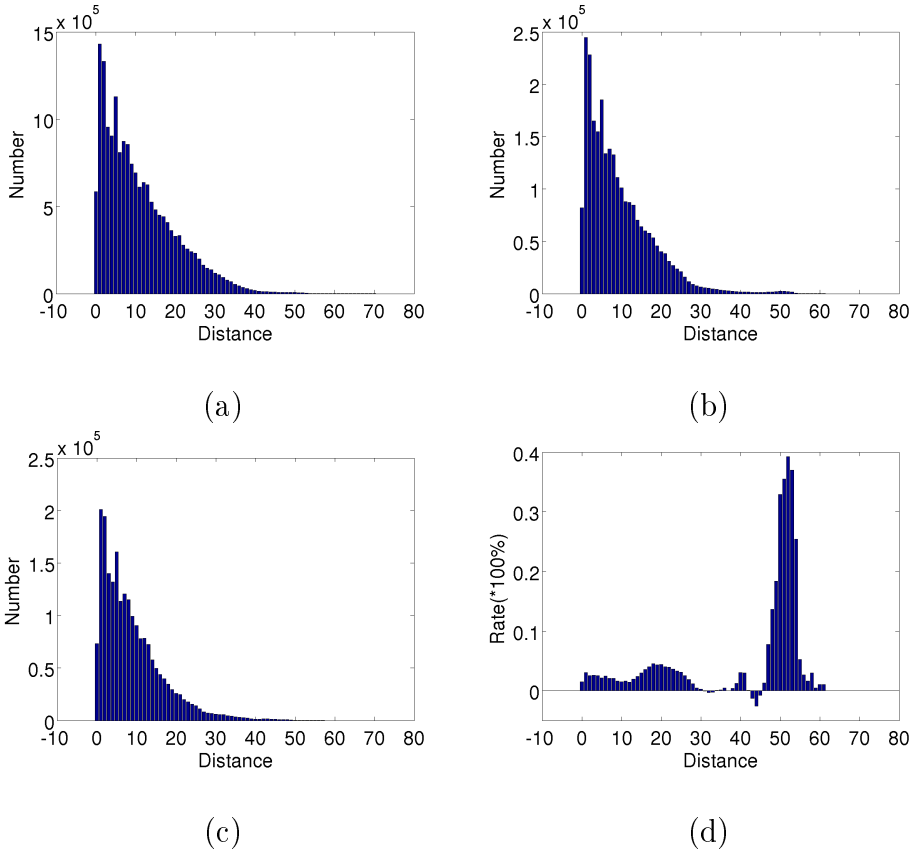


Figure 5.6: Histograms over different distances to part boundaries: (a) H_g — the histogram of all pixels; (b) H_s — the histogram of pixels for which the proposed method outperforms the baseline; (c) H_b — the histogram of pixels for which the baseline outperforms the proposed method; (d) H_d — is the normalized difference histogram.

distances from the part boundaries:

$$H_d(i) = (H_s(i) - H_b(i))/H_g(i). \tag{5.6}$$

Our method gains over a very large majority of distances except for a few outliers. We see a strong bimodality, i.e. our method wins close-to and away from the borders. Moreover, the mode corresponding to pixels near the center of parts shows an even higher proportion of improvement than the mode closer to the border. In particular, the classification rates are significantly improved between distance 50 and 60, which is more or less half of the receptive size. At the same time, this analysis verifies that our system indeed does benefit from the multi-scale framework.

Table 5.5: Running times on the CDC4CV dataset of our proposed methods. For the ConvNets method, training includes pre-training, LR learning and fine-tuning, and testing time on one image is given. The resolution of testing depth image is 320*240.

Task	Method	Machine	Architecture	Time
Testing	spatial ConvNet	Xeon E56620, 8 cores, 2.4GHz	MATLAB+IPP	00.41 sec
Testing	spatial ConvNet	Xeon E56620, 8 cores, 2.4GHz	MATLAB+GPU	00.31 sec
Testing	spatial RDF	Xeon E56620, 8 cores, 2.4GHz	C++ / CPU only	00.18 sec
Testing	classical RDF	Xeon E56620, 8 cores, 2.4GHz	C++ / CPU only	00.18 sec
Training	spatial ConvNet	Xeon E56620, 8 cores, 2.4GHz	MATLAB+GPU	36.00 h
Training	spatial RDF	Xeon E56620, 8 cores, 2.4GHz	C++ / CPU only	15.8 h

5.4 Discussion and conclusion

The experiments above demonstrate that even pixelwise independent classification indeed benefits from the *a priori* additional information contained in the spatial part layout.

The quantitative evaluation measures presented in the different tables report pixelwise classification accuracy. This proves that the proposed learning algorithms improve pixelwise classification, although the injected additional information does not specifically state this. An additional advantage of our methods are *not* captured by this evaluation: if a pixel is wrongly classified, then spatial learning will increase the probability that the wrong part label is a neighbor of the correct part label. In some applications this can be an advantage, for instance if the segmentation is post-processed to estimate joint positions, as in [Shotton et al., 2011].

Currently, we demonstrated a single application of the proposed methods to human part estimation in depth images. The approach is, however, not specific to pose estimation, and can be applied to other problems, as for instance object segmentation from RGB images, or semantic full scene labeling. In these cases, the context may be dynamic and the spatial layout might be more difficult to extract from the corpus, requiring learning techniques. This is reserved for future work.

Two different methods have seen introduced: one based on randomized decision forests, and a second one based on a combination of convolutional neural networks and logistical regression (or MLPs). The latter deep learning method achieves better results than the RDF based method, which is mostly explained by the more powerful features learned by the ConvNets. However, this classification performance comes at a cost, as the method is computationally more complex.

Figure 5.7: The VOIR platform.

Table 5.5 gives running times for testing and for training. Note that testing times seem to be comparable for the RDF and ConvNets methods, RDFs being roughly twice as fast as ConvNets. However, the ConvNets method was implemented with GPU support for convolutions. While GPU implementation is certainly possible for the RDF method to make it real-time, which has not been done in our experiments.

Indeed, the main computational complexity of the deep learning method comes from the convolution operations, which can be significantly accelerated by parallel computation, e.g. on multi-core CPUs or GPUs. We implemented the system in MATLAB with multi-threaded convolutions provided by the GPU (CUDA) library (for training and testing), and the testing time by the IPP framework is also given as a comparison.

The RDF method has been implemented on our mobile robotics platform VOIR ("Vision and Observation In Robotics"), as illustrated in figure 5.7. A video is available online at `http://liris.cnrs.fr/voir`.

The ConvNets method can be easily extended to semi-supervised learning, taking advantage of additional unlabeled training data. On this unlabeled data, the loss function would not include any terms based on classification loss (which require the ground truth part label), but only terms based on the spatial layout of the parts, as mentioned in Section 4.4.2 of chapter 4.

It is worth noting that the two proposed contributions of the ConvNets based method can be applied independently. In particular, the supervised spatial learning algorithm can be combined with any supervised or unsupervised training algorithms for ConvNets, for example auto-encoding or other algorithms reviewed in Section 2.4 of chapter 2.

Chapter 6

General conclusion and discussion

Contents

In this chapter, we first conclude our work and review the different contributions. We then discuss some limitations, and finally introduce perspectives and future work.

6.1 Summary of our contributions

In this thesis, we have presented several contributions on visual recognition. Our objective was to combine the available information, such as spatial configurations of labels in feature space, or spatial layouts of objects, to improve the performance of recognition algorithms. Two types of visual recognition problems have been investigated: action recognition and human body part segmentation. These two problems differ from each other in nature. However, they are actually related to each other: human body part segmentation can be treated as a preliminary step for action recognition, due to its relation to human pose.

We resolve the two visual recognition problems by machine learning approaches, which is a common choice in computer vision. Features are first extracted from the input image/video and then machine learning techniques are employed to learn a classifier for classification. Our contributions provide more available information in the learning stage, resulting in a better representation of the object, or a better classifier. We briefly summarize them in the following:

Supervised end-to-end learning for BoW models: We applied BoW models to action recognition. Each video is represented as a histogram of occurrence of codewords in a pre-learned codebook. We reformulated

the BoW model as a neural network in an end-to-end training framework with two modules (see figure 3.1). The first module creates and learns BoW models, and the second module is a two-layer Multi-Layer Perceptron(MLP), which assigns a label to each input video. The gradient information computed from the MLP module is back-propagated to the BoW module and used to learn the codebook. However, this kind of backpropagation is not straight forward in the network architecture due to the pooling process over interest points. We proposed two algorithms to make use of the gradient information to optimize the codebook. One is based on classical error backpropagation, and the other is based on cluster reassignments, in which we reassign the cluster labels for all the feature vectors based on the errors in a Voronoi diagram. In this way, the codebook is learned in a supervised way according to the gradient information from the MLP. We demonstrated the effectiveness of the proposed algorithms on the standard KTH human action dataset.

Spatial learning for randomized decision forest: The randomized decision forest (RDF) is a simple method for classification. The feature vector of each entity (object/pixel etc.) is passed down the tree through a comparison between each feature value with a threshold. The threshold for each node (as well as feature parameters) are chosen maximizing the entropy gain from the parent to children nodes. RDFs work well when a large amount of training data is available. However, when the training data is very limited, the performance drops. We proposed to integrate the spatial layout of objects into RDF learning. The original label set is split into two sets of label pairs: neighboring labels and non-neighboring labels. The gain in entropy is adopted to each set. We applied our algorithm to the human part estimation problem and demonstrated that our algorithm is more discriminative than classical RDF learning. A by-product are edge features, similar to the existing depth comparison features, which are capable of improving classification performance.

Spatial learning for convolutional neural networks: Compared to feature threshold learning in RDFs, convolutional neural networks can learn features through end-to-end training even in the existing literature. We proposed spatial learning for the feature learning module and for the classification module, respectively. For the feature learning module, our algorithm operates on pairs of pixels. The pairs are divided into three sets: the same label, neighboring labels, non-neighboring labels. A loss function is minimized in order to make the mapped feature vectors close for pairs of pixels with the same label and distant for pairs of pixels

with non-neighboring labels. For the classification stage, we structured the outputs by a ranking technique. We aim to rank the response from the groundtruth label highest, to rank the responses from the labels which are neighbors of the groundtruth label next, and finally to rank which are not neighbors last. Weakly-supervised learning and supervised learning have been combined to further improve the recognition performance. Experiments on human part estimation in depth images demonstrated the effectiveness of our algorithms.

General conclusion

From the results we have obtained, we can conclude that visual recognition can be improved by adding information and constraints such as labels, and spatial layout into different learning stages i.e. into the feature learning stage, into the classification stage or into both. An end-to-end framework is very interesting for feature learning because it connects the path from the raw input data to recognition results. It is goal-oriented, since classification error is directly used for learning all of different parameters of the system. We draw the following conclusions:

- Our algorithms work better in the case of small and medium amount of training data. A large amount of training data allows the classifier to better generalize and produce optimal performance. On the other hand, *a priori* information is additionally provided to the small/medium training data to help to generalize the learned machine, playing a similar role as regularization terms.

- In the training procedure, pairs of entities are fed into the learning. However, testing is still performed on the single entity (pixel), therefore there is no additional computation cost.

- End-to-end learning can be useful, but sometimes should be completed by pre-training (unsupervised learning or weakly supervised learning).

- It is not always clear when end-to-end is better, which thus depends on the complexity of the problems (i.e. the function to minimize), the amount of available data, the amount of available training time etc.

6.2 Limitations and Future work

Limitations of supervised end-to-end learning for bag-of-words models


Figure 6.1: The motorbike on the right is severely self-occluded, compared to the one on the left. Most of the spatial part relations are useless in the energy function.

In the current work, the input is comprised of extracted features, and the BoW model construction is formulated as a neural network. The dictionary is coded into the weights between the layers. Low level features like HOG/HOF are chosen, rather than learned. A feature learning module could be added at the beginning, e.g. a convolutional layer as described in section 4.4.1. The error backpropagation scheme can be applied to learn the filters in the convolutional layer. The advantage would be to learn the features and the dictionary together in a longer end-to-end learning scheme.

Limitations of spatial learning

In this thesis, we also proposed three spatial learning algorithms for two different frameworks: spatial learning for randomized decision forests, spatial learning for pre-training ConvNets, and spatial learning for supervised learning of LR and MLP classifiers. They differ from each other, but they share a minor common limitation: constant spatial part layout structure, as shown in figure 5.2 for human upper body part segmentation. The reason is that the spatial layout depends on inherent object properties, even for a non-rigid object. The spatial layout is correct for all the objects in the same category. However, in reality, due to occlusions induced by different object poses, some parts may be not visible in the image, their spatial relations to other parts still occur in the energy function. At the same time, some new spatial relations emerge due to dynamic poses, on the contrary, which are ignored in the energy function. Figure 6.1 and 6.2 show two limitation cases, respectively.

A possible solution to deal with occlusion and dynamic pose is to adopt a dynamic spatial layout for each image. In principle, the spatial layout could been detected for each image by neighborhood search algorithms. But energy divergence might be a problem in this context.

Limitations of spatial ConvNets learning

We proposed a ranking technique in the spatial supervised learning of LR

Figure 6.2: In comparison to the left person, new part spatial layout relations emerge due to pose changes in the right person, e.g. the left lower arm and the right lower arm is not taken into account in the model.

and MLP classifiers. The response of the groundtruth label is ranked first, followed by the neighboring labels of the groundtruth, and last by the non-neighboring labels of the groundtruth. The desired probabilities we used in the current version are rigid, i.e. the desired probability is set to 1 for all the correct rankings, otherwise 0. However, it is intuitive that the desired probabilities should be "soft", i.e. values in $[0, 1]$, keeping certain constraints.

The exact probabilities are hyperparameters, which could be learned over the validation data. Additional constraints will be required to ensure that the learned ranking measure is valid, as suggested in [Burges et al., 2005] in a different context.

6.3 Other types of spatial information

The current pipeline presented in chapter 4 is used to classify pixels for human part estimation. It works for simple parts and for rigid objects (faces etc.). The variance in pose and occlusion is not large. The parts and their *a priori* information of spatial layout are given as groundtruth.

If we aim to recognize non-rigid objects (and not just their parts), in this context, a possible solution is first to detect parts, and then to treat constellation of the features from different parts as the object features, which are fed into a classifier. The features also can be learned in an end-to-end way. This framework is a combination of DPMs (see section 2.3.2.3 of chapter 2) and deep feature learning (see section 2.4 of chapter 2). However, it requires to solve a combinational problem over part positions, which makes it difficult to calculate gradients used for feature learning. Figure 6.3 shows this framework. This is our ongoing work.

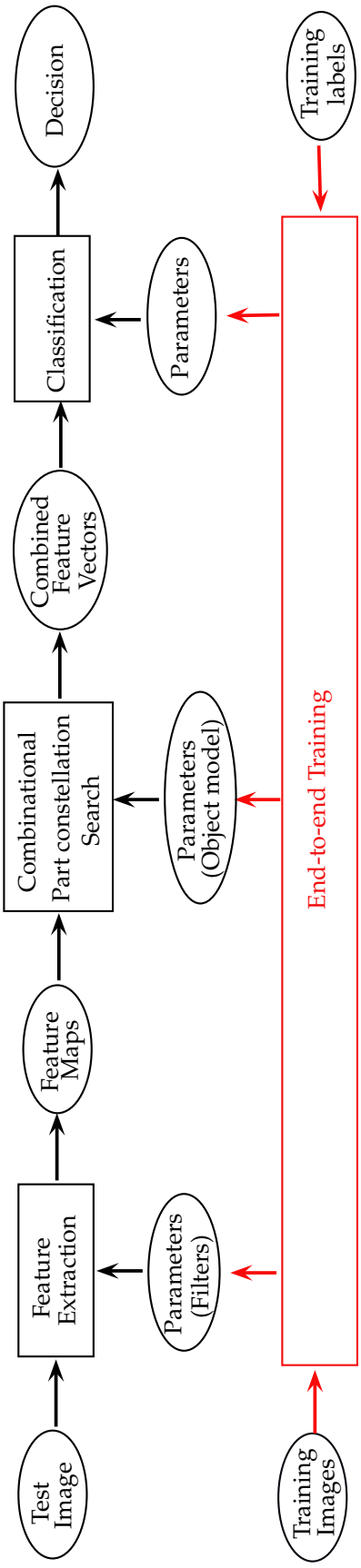


Figure 6.3: The framework of combination of DPMs and deep feature learning.

Bibliography

[Abdelkader et al., 2010] Abdelkader, M. F., Abd-Almageed, W., Srivastava, A., and Chellappa, R. (2010). Silhouette-based Gesture and Action Recognition via Modeling Trajectories on Riemannian shape manifolds. *Computer Vision and Image Understanding*, 115(3):439–455. (Cited on page 70.)

[Aggarwal and Ryoo, 2011] Aggarwal, J. and Ryoo, M. (2011). Human activity analysis: A review. *ACM Computing Surveys (CSUR)*, 43(3). (Cited on page 69.)

[Ahonen et al., 2006] Ahonen, T., Hadid, A., and Pietikäinen, M. (2006). Face description with local binary patterns: application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–41. (Cited on page 24.)

[Ahonen et al., 2009] Ahonen, T., Matas, J., He, C., and Pietikäinen, M. (2009). Rotation invariant image description with local binary pattern histogram fourier features. *Scandinavian Conference on Image Analysis, Lecture Notes in Computer Science.*, pages 61–70. (Cited on page 25.)

[Baccouche et al., 2011] Baccouche, M., Mamalet, F., Wolf, C., Garcia, C., and Baskurt, A. (2011). Sequential Deep Learning for Human Action Recognition. *2nd International Workshop on Human Behavior Understanding*, pages 1–11. (Cited on page 70.)

[Ballan et al., 2009] Ballan, L., Bertini, M., and Bimbo, A. D. (2009). Effective codebooks for human action categorization. *in Proc. of ICCV Workshop on Video-oriented Object and Event Classification (VOEC)*. (Cited on page 36.)

[Barrow et al., 1977] Barrow, H. G., Tenenbaum, J. M., Bolles, R. C., and Wolf, H. C. (1977). Parametric correspondence and chamfer matching: two new techniques for image matching. *In Proc. 5th International joint conference Artificial Intelligence*. (Cited on pages 29 and 103.)

[Bay et al., 2006] Bay, H., Tuytelaars, T., and Van Gool, L. (2006). Surf: Speeded up robust features. *ECCV*, pages 404–417. (Cited on pages 21 and 24.)

[Belhumeur et al., 1997] Belhumeur, P., Hespanha, João, P., and Kriegman, D. J. (1997). Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(7):711–720. (Cited on page 16.)

[Belongie et al., 2002] Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522. (Cited on pages ix, 14 and 15.)

[Bengio, 2009] Bengio, Y. (2009). Learning Deep Architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127. (Cited on pages x and 46.)

[Bettadapura et al., 2013] Bettadapura, V., Schindler, G., Plötz, T., and Essa, I. (2013). Augmenting bag-of-words: Data-driven discovery of temporal and structural information for activity recognition. *CVPR*. (Cited on page 37.)

[Bishop, 1994] Bishop, C. (1994). Neural networks for pattern recognition. pages 140–145. (Cited on page 61.)

[Bobick and Davis, 2001] Bobick, A. F. and Davis, J. W. (2001). The recognition of human movement using temporal templates. *IEEE Transactions on PAMI*, 23(3):257267. (Cited on pages ix, 18 and 19.)

[Boiman and Irani, 2007] Boiman, O. and Irani, M. (2007). Detecting Irregularities in Images and in Video. *International Journal of Computer Vision*, 74(1):17–31. (Cited on page 70.)

[Borgefors, 1988] Borgefors, G. (1988). Hierarchical chamfer matching: A parametric edge matching algorithm. *IEEE Transcations on PAMI*, 10(6):849–865. (Cited on page 30.)

[Boureau et al., 2010] Boureau, Y.-l., Bach, F., LeCun, Y., and Ponce, J. (2010). Learning mid-level features for recognition. *CVPR*. (Cited on page 35.)

[Bregonzio et al., 2009] Bregonzio, M., Gong, S., and Xiang, T. (2009). Recognising Action as Clouds of Space-Time Interest Points. In *CVPR*. (Cited on page 69.)

[Burges et al., 2005] Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., and Hullender, G. (2005). Learning to rank using gradient descent. In *ICML*. (Cited on pages 94, 95 and 117.)

[Calonder et al., 2010] Calonder, M., Lepetit, V., Strecha, C., and Fua, P. (2010). Brief: Binary robust independent elementary features. *ECCV*. (Cited on page 24.)

[Cao et al., 2010] Cao, Y., Wang, C., Li, Z., Zhang, L., and Zhang, L. (2010). Spatial-bag-of-features. *CVPR*. (Cited on page 37.)

[Chang and Lin, 2011] Chang, C.-C. and Lin, C.-J. (2011). LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2(3):1–27. (Cited on page 74.)

[Cootes et al., 2001] Cootes, T., Edwards, G. J., and Taylor, C. J. (2001). Active appearance models. *PAMI*, 23(6):681685. (Cited on page 17.)

[Cootes et al., 1995] Cootes, T., Taylor, C., Cooper, D., and Graham, J. (1995). Active shape models-their training and application. *Computer Vision and Image Understanding*. (Cited on page 14.)

[Couprie et al., 2013] Couprie, C., Farabet, C., Najman, L., and LeCun, Y. (2013). Indoor semantic segmentation using depth information. *arXiv*, pages 1–8. (Cited on page 43.)

[Cox and Cox, 1994] Cox, T. and Cox, M. (1994). *Multidimensional scaling*. Chapman and Hill. (Cited on page 85.)

[Csurka et al., 2004] Csurka, G., Dance, C., and Fan, L. (2004). Visual categorization with bags of keypoints. *ECCV International Workshop on Statistical Learning in Computer Vision*. (Cited on pages ix, 12, 13, 34, 35, 53, 54, 55, 69 and 79.)

[Cuntoor et al., 2008] Cuntoor, N. P., Yegnanarayana, B., and Chellappa, R. (2008). Activity modeling using event probability sequences. *IEEE transactions on image processing*, 17(4):594–607. (Cited on page 70.)

[Dalal and Triggs, 2005] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *CVPR*. (Cited on pages x, 24, 26 and 90.)

[Dalal et al., 2006] Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. *ECCV*, pages 428–441. (Cited on page 24.)

[Dean et al., 2013] Dean, T., Ruzon, M. A., and View, M. (2013). Fast , accurate detection of 100 , 000 object classes on a single machine. *CVPR*. (Cited on page 41.)

[Dekel et al., 2004] Dekel, O., Manning, C., and Singer, Y. (2004). Log-linear models for label-ranking. In *NIPS*. (Cited on page 94.)

[Dietterich, 1998] Dietterich, T. G. (1998). Approximate Statistical Tests for Comparing Supervised Classi cation Learning Algorithms 1 Introduction. *Neural Computation*, 10(7):1895–1923. (Cited on page 75.)

[Divvala et al., 2012] Divvala, S., Efros, A., and Hebert, M. (2012). How important are "deformable parts" in the deformable parts model? *ECCV Workshops*. (Cited on page 40.)

[Divvala and Hoiem, 2009] Divvala, S. and Hoiem, D. (2009). An empirical study of context in object detection. *CVPR*, pages 1271–1278. (Cited on page 85.)

[Dollar et al., 2005] Dollar, P., Rabaud, V., Cottrell, G., and Belongie, S. (2005). Behavior Recognition via Sparse Spatio-Temporal Features. *IEEE International Workshop on Visual Surveillance and Performance Evaluation of Tracking and Surveillance*, pages 65–72. (Cited on pages x, 22, 34, 35 and 53.)

[Dubuisson and Jain, 1994] Dubuisson, M. and Jain, A. (1994). A modified hausdorff distance for object matching. *International Conference on Pattern Recognition*, pages 566–568. (Cited on page 30.)

[Duchenne et al., 2009] Duchenne, O., Bach, F., Kweon, I., and Ponce, J. (2009). A tensor-based algorithm for high-order graph matching. (Cited on pages 33 and 80.)

[Dyana and Das, 2009] Dyana, a. and Das, S. (2009). Trajectory representation using Gabor features for motion-based video retrieval. *Pattern Recognition Letters*, 30(10):877–892. (Cited on page 70.)

[Everingham and Zisserman, 2010] Everingham, M., V. G. L. W. C. K. I. W. J. and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338. (Cited on pages ix, 2 and 3.)

[Everts et al., 2013] Everts, I., van Gemert, J., and Gevers, T. (2013). Evaluation of color stips for human action recognition. pages 2850–2857. (Cited on page 22.)

[Farabet et al., 2012] Farabet, C., Couprie, C., Najman, L., and LeCun, Y. (2012). Scene parsing with multiscale feature learning, purity trees, and

optimal covers. *NIPS*. (Cited on pages xii, xv, 43, 81, 86, 90, 96, 107 and 108.)

[Fathi and Mori, 2008] Fathi, A. and Mori, G. (2008). Action recognition by learning mid-level motion features. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, pages 1–8. IEEE. (Cited on page 70.)

[Felzenszwalb et al., 2010] Felzenszwalb, P., Girshick, R., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part based models. *IEEE Tr on PAMI*, 32(9):1627–1645. (Cited on pages ix, 12, 13, 38, 39 and 41.)

[Felzenszwalb and Huttenlocher, 2005] Felzenszwalb, P. F. and Huttenlocher, D. P. (2005). Pictorial Structures for Object Recognition. *IJCV*, 61(1):55–79. (Cited on pages 39 and 80.)

[Felzenszwalb and Zabih, 2011] Felzenszwalb, P. F. and Zabih, R. (2011). Dynamic programming and graph algorithms in computer vision. *IEEE transactions on pattern analysis and machine intelligence*, 33(4):721–40. (Cited on page 80.)

[Fergus et al., 2003] Fergus, R., Perona, P., and Zisserman, A. (2003). Object class recognition by unsupervised scale-invariant learning. In IEEE, editor, *CVPR*, volume 2, pages 264–271. (Cited on page 38.)

[Fergus et al., 2004] Fergus, R., Perona, P., and Zisserman, A. (2004). A visual category filter for google images. *ECCV*, pages 1–14. (Cited on page 38.)

[Ferrari et al., 2008] Ferrari, V., Marin-Jimenez, M., and Zisserman, A. (2008). Progressive search space reduction for human pose estimation. In *CVPR*, pages 1–8. (Cited on page 105.)

[Fischler and Bolles, 1981] Fischler, M. and Bolles, R. (1981). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communcations of the ACM*, 24:381â395. (Cited on page 80.)

[Fischler and Elschlager, 1973] Fischler, M. and Elschlager, R. (1973). The representation and matching of pictorial structures. *Computers, IEEE Transactions on*, c(1):67–92. (Cited on pages 38, 39 and 80.)

[Freund et al., 2003] Freund, Y., Iyer, R., Schapire, R., and Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969. (Cited on page 94.)

[Gaidon et al., 2011] Gaidon, A., Harchoui, Z., and Schmid, C. (2011). A time series kernel for action recognition. *Procedings of the British Machine Vision Conference 2011*, pages 63.1–63.11. (Cited on page 37.)

[Geman and Geman, 1984] Geman, S. and Geman, D. (1984). Stochastic Relaxation, Gibbs Distributions, and the Bayesian Restoration of Images. *IEEE Transactions on PAMI*, 6(6):721–741. (Cited on page 86.)

[Gilbert et al., 2011] Gilbert, A., Illingworth, J., and Bowden, R. (2011). Mined Hierarchical Compound Features. *International Journal of Computer Vision*, 33(5):883–897. (Cited on page 55.)

[Girshick et al., 2013] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. (Cited on pages x and 42.)

[Goh et al., 2012] Goh, H., Thome, N., Cord, M., and Lim, J. (2012). Unsupervised and supervised visual codes with restricted boltzmann machines. *ECCV*. (Cited on pages 36 and 55.)

[Goldberger et al., 2004] Goldberger, J., Roweis, S., Hinton, G., and Salakhutdinov, R. (2004). Neighbourhood component analysis. In *NIPS*. (Cited on pages 50 and 85.)

[Goncalves and Perona, 2003] Goncalves, L. and Perona, P. (2003). Unsupervised learning of human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(7):814–827. (Cited on page 69.)

[Gorelick et al., 2007] Gorelick, L., Blank, M., Shechtman, E., Irani, M., and Basri, R. (2007). Actions as space-time shapes. *IEEE transactions on PAMI*, 29(12):22472253. (Cited on pages ix, 15 and 16.)

[Gould et al., 2009] Gould, S., Fulton, R., and Koller, D. (2009). Decomposing a scene into geometric and semantically consistent regions. *ICCV*, pages 1–8. (Cited on pages x and 42.)

[Grangier et al., 2009] Grangier, G., Bottou, L., and R.Collobert (2009). Deep convolutional networks for scene parsing. In *ICML Deep Learning Workshop*. (Cited on page 90.)

[Grauman and Darrell, 2005] Grauman, K. and Darrell, T. (2005). The pyramid match kernel: discriminative classification with sets of image features. *ICCV*, pages 1458–1465. (Cited on page 37.)

[Hadsell et al., 2006] Hadsell, R., Chopra, S., and LeCun, Y. (2006). Dimensionality reduction by learning an invariant mapping. In *CVPR*, pages 1735–1742. (Cited on pages x, 46, 50, 83, 85, 92, 93, 107 and 108.)

[Harris and Stephens, 1988] Harris, C. and Stephens, M. (1988). A combined corner and edge detector. In *In Proc. of Fourth Alvey Vision Conference*, pages 147–151. (Cited on page 20.)

[Heitz and Koller, 2008] Heitz, G. and Koller, D. (2008). Learning spatial context: Using stuff to find things. *ECCV*, pages 1–14. (Cited on page 85.)

[Heseltine et al., 2003] Heseltine, T., Pears, N., Austin, J., and Chen, Z. (2003). Face Recognition : A Comparison of Appearance-Based Approaches 2 The Direct Correlation Method. *Proc. VIIth Digital Image Computing: Techniques and Applications*, pages 59–68. (Cited on pages ix and 16.)

[Hinton et al., 2006] Hinton, G., Osindero, S., and Teh, Y.-W. (2006). A fast learning algorithm for deep belief nets. *Neural Computation*. (Cited on pages x, 48 and 49.)

[Hinton and Salakhutdinov, 2006] Hinton, G. and Salakhutdinov, R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507. (Cited on page 91.)

[Hinton, 1995] Hinton, G. E. (1995). The wake-sleep algorithm for unsupervised neural networks. *Science*. (Cited on page 49.)

[Holt et al., 2011] Holt, B., Ong, E.-J., Cooper, H., and Bowden, R. (2011). Putting the pieces together: Connected poselets for human pose estimation. In *ICCV Workshop on Consumer Depth Cameras for Computer Vision*. (Cited on pages xii, 99 and 100.)

[Hu and Boulgouris, 2011] Hu, J. and Boulgouris, N. V. (2011). Fast human activity recognition based on structure and motion. *Pattern Recognition Letters*, 32(14):1814–1821. (Cited on page 19.)

[Hu, 1962] Hu, M.-K. (1962). Visual Pattern Recognition by Moment. *IRE Transactions on Information Theory*, pages 66–70. (Cited on page 17.)

[Huang, 2008] Huang, T. (2008). Discriminative local binary patterns for human detection in personal album. *CVPR*, pages 1–8. (Cited on page 24.)

[Huttenlocher et al., 1993] Huttenlocher, D., Klanderman, G. A., and Rucklidge, W. J. (1993). Comparing images using the hausdorff distance. *IEEE Transcations on PAMI*, 15(9):850–863. (Cited on page 30.)

[Jarrett and Kavukcuoglu, 2009] Jarrett, K. and Kavukcuoglu, K. (2009). What is the best multi-stage architecture for object recognition? In *ICCV*. (Cited on page 90.)

[Jhuang et al., 2007] Jhuang, H., Serre, T., and Wolf, L. (2007). A Biologically Inspired System for Action Recognition. *ICCV*. (Cited on page 70.)

[Jolliffe, 1986] Jolliffe, T. I. (1986). Principal component analysis. In *New York: Springer-Verlag*. (Cited on page 85.)

[Jurie and Triggs, 2005] Jurie, F. and Triggs, B. (2005). Creating efficient codebooks for visual recognition. *ICCV*. (Cited on page 35.)

[Kamarainen et al., 2007] Kamarainen, J. K., Kyrki, V., and Kälviäinen, H. (2007). Local and global Gabor features for object recognition. *Pattern Recognition and Image Analysis*, 17(1):93–105. (Cited on page 27.)

[Kass et al., 1988] Kass, M., Witkin, A., and Terzopoulos, D. (1988). Snakes: Active contour models. *International journal of computer vision*. (Cited on page 14.)

[Ke and Sukthankar, 2004] Ke, Y. and Sukthankar, R. (2004). PCA-SIFT: A more distinctive representation for local image descriptors. *CVPR*, 2:506–513. (Cited on page 23.)

[Klaser et al., 2008] Klaser, A., Marszalek, M., and Schmid, C. (2008). A spatio-temporal descriptor based on 3d-gradients. *BMVC*. (Cited on page 24.)

[Kolmogorov and Zabih, 2004] Kolmogorov, V. and Zabih, R. (2004). What energy functions can be minimized via graph cuts? *IEEE Transactions on PAMI*, 26(2):147–159. (Cited on pages 81 and 87.)

[Kontschieder et al., 2013] Kontschieder, P., Kohli, P., Shotton, J., and Criminisi, A. (2013). Geof: Geodesic forests for learning coupled predictors. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 65–72. (Cited on page 43.)

[Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *NIPS*. (Cited on pages 46 and 90.)

[Laptev, 2005] Laptev, I. (2005). On Space-Time Interest Points. *International Journal of Computer Vision*, 64:107–123. (Cited on pages ix, 21, 22, 23, 35 and 55.)

[Laptev et al., 2008] Laptev, I., Marszalek, M., Schmid, C., and Rozenfeld, B. (2008). Learning realistic human actions from movies. In *CVPR*, pages 1–8. IEEE. (Cited on pages 24 and 70.)

[Lazebnik and Schmid, 2006] Lazebnik, S. and Schmid, C. (2006). Beyond bags of features : Spatial pyramid matching for recognizing natural scene categories. In *CVPR*. (Cited on page 37.)

[Le et al., 2011] Le, Q., Zou, W., Yeung, S., and Ng, A. (2011). Learning hierarchical invariant spatio-temporal features for action recognition with independent subspace analysis. *CVPR*, pages 3361–3368. (Cited on page 22.)

[LeCun et al., 1998] LeCun, Y., Botto, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of IEEE*, 86(11):2278–2324. (Cited on pages 27, 45, 85, 90, 96 and 108.)

[LeCun et al., 2010] LeCun, Y., Kavukcuoglu, K., and Farabet, C. (2010). Convolutional networks and applications in vision. *Proceedings of 2010 IEEE International Symposium on Circuits and Systems*, pages 253–256. (Cited on page 46.)

[Lee et al., 2007] Lee, H., Ekanadham, C., and Ng, A. (2007). Sparse deep belief net model for visual area v2. *NIPS*, pages 1–8. (Cited on page 47.)

[Lee et al., 2011] Lee, J., Cho, M., and Lee, K. M. (2011). Hyper-graph matching via reweighted random walks. In *CVPR*. (Cited on page 33.)

[Leonardis and Bischof, 2000] Leonardis, A. and Bischof, H. (2000). Robust recognition using eigenimages. *Computer Vision and Image Understanding*, 78(1):99118. (Cited on page 16.)

[Lepetit et al., 2004] Lepetit, V., Pilet, J., and Fua, P. (2004). Point matching as a classification problem for fast and robust object pose estimation. In *CVPR*, pages 244–250. (Cited on page 87.)

[Leung and Malik, 2001] Leung, T. and Malik, J. (2001). Representing and recognizing the visual appearance of materials using three-dimensional textons. *International Journal of Computer Vision*, 43(1):29–44. (Cited on page 35.)

[Li et al., 2013] Li, W., Yu, Q., Sawhney, H., and Vasconcelos, N. (2013). Recognizing activities via bag of words for attribute dynamics. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2587–2594. (Cited on page 38.)

[Liao et al., 2009] Liao, S., Law, M., and Chung, A. (2009). Dominant local binary patterns for texture classification. *IEEE Transactions on Image Processing*, 18(5):1107–1118. (Cited on page 26.)

[Lin et al., 2009] Lin, L., Zeng, K., Liu, X., and Zhu, S.-C. (2009). Layered graph matching by composite cluster sampling with collaborative and competitive interactions. *CVPR*, pages 1351–1358. (Cited on page 33.)

[Liu et al., 2008] Liu, J., Ali, S., and Shah, M. (2008). Recognizing human actions using multiple features. In *CVPR*. (Cited on page 69.)

[Liu and Shah, 2008] Liu, J. and Shah, M. (2008). Learning human actions via information maximization. *CVPR*, pages 1–8. (Cited on pages 36, 54 and 55.)

[Liu et al., 2009] Liu, J., Yang, Y., and Shah, M. (2009). Learning semantic visual vocabularies using diffusion distance. *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 461–468. (Cited on pages 35 and 54.)

[Lowe, 2004] Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110. (Cited on pages x, 21, 23, 53, 69 and 90.)

[Ma et al., 2010] Ma, T., Yang, X., and Latecki, L. (2010). Boosting chamfer matching by learning chamfer distance normalization. *ECCV*, (1). (Cited on page 30.)

[Marius et al., 2011] Marius, L., Andrei, Z., and Cristian, S. (2011). Semi-supervised learning and optimization for hypergraph matching. In *ICCV*, pages 2274–2281. (Cited on page 33.)

[Matikainen et al., 2009] Matikainen, P., Hebert, M., and Sukthankar, R. (2009). Trajectons: Action recognition through the motion analysis of tracked features. *2009 IEEE 12th International Conference on Computer Vision Workshops*, pages 514–521. (Cited on page 35.)

[Matikainen et al., 2010] Matikainen, P., Hebert, M., and Sukthankar, R. (2010). Representing pairwise spatial and temporal relations for action recognition. *ECCV*, pages 508–521. (Cited on page 37.)

[Memisevic and Hinton, 2007] Memisevic, R. and Hinton, G. (2007). Unsupervised learning of image transformations. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. (Cited on page 49.)

[Mikolajczyk et al., 2006] Mikolajczyk, K., Leibe, B., and Schiele, B. (2006). Multiple object class detection with a generative model. *CVPR*, 1:26–36. (Cited on page 38.)

[Mikolajczyk and Schmid, 2002] Mikolajczyk, K. and Schmid, C. (2002). An affine invariant interest point detector. *ECCV*. (Cited on page 21.)

[Mikolajczyk and Schmid, 2005] Mikolajczyk, K. and Schmid, C. (2005). Performance evaluation of local descriptors. *IEEE transactions on PAMI*, 27(10):1615–30. (Cited on page 23.)

[Mikolajczyk and Uemura, 2008] Mikolajczyk, K. and Uemura, H. (2008). Action recognition with motion-appearance vocabulary forest. *CVPR*. (Cited on page 69.)

[Mikolajczyk and Uemura, 2011] Mikolajczyk, K. and Uemura, H. (2011). Action recognition with appearancemotion features and fast search trees. *Computer Vision and Image Understanding*, 115(3):426–438. (Cited on page 55.)

[Minetto et al., 2013] Minetto, R., Thome, N., Cord, M., Leite, N. J., and Stolfi, J. (2013). T-HOG: An effective gradient-based descriptor for single line text regions. *Pattern Recognition*, 46(3):1078–1090. (Cited on page 24.)

[Mitchell, 1997] Mitchell, T. M. (1997). Machine Learning. page 2. (Cited on page 1.)

[Moosmann et al., 2007] Moosmann, F., Triggs, B., and Jurie, F. (2007). Fast discriminative visual codebooks using randomized clustering forests. In *Advances in neural information processing systems*, volume 19, page 985. Citeseer. (Cited on pages 36 and 54.)

[Moravec, 1981] Moravec, H. (1981). Rover Visual Obstacle Avoidance. *IJCAI*. (Cited on page 20.)

[Murase and Nayar, 1995] Murase, H. and Nayar, S. K. (1995). Visual learning and recognition of 3-d objects from appearance. *International Journal of Computer Vision*, 14(1):524. (Cited on page 16.)

[Niebles and Fei-Fei, 2007] Niebles, J. C. and Fei-Fei, L. (2007). A hierarchical model of shape and appearance for human action classification. *CVPR*. (Cited on page 70.)

[Niebles et al., 2010] Niebles, J. C., W, C. C., and Fei-Fei, L. (2010). Modeling temporal structure of decomposable motion segments for activity classification. page 392405. (Cited on pages ix and 3.)

[Niebles et al., 2008] Niebles, J. C., Wang, H., and Fei-Fei, L. (2008). Unsupervised Learning of Human Action Categories Using Spatial-Temporal Words. *International Journal of Computer Vision*, 79(3):299–318. (Cited on pages x, 22, 55 and 70.)

[Oikonomopoulos et al., 2009] Oikonomopoulos, A., Patras, I., and Pantic, M. (2009). An implicit spatiotemporal shape model for human activity localization and recognition. In *CVPR*, pages 27–33. (Cited on pages 37 and 55.)

[Ojala et al., 1996] Ojala, T., Pietikäinen, M., and Harwood, D. (1996). A comparative study of texture measures with classification based on featured distributions. *Pattern recognition*, 29. (Cited on page 24.)

[Ott and Everingham, 2011] Ott, P. and Everingham, M. (2011). Shared parts for deformable part-based models. *CVPR*. (Cited on page 40.)

[Pinheiro and Collobert, 2013] Pinheiro, P. and Collobert, R. (2013). Recurrent convolutional neural networks for scene parsing. *arXiv*. (Cited on page 43.)

[Poppe, 2010] Poppe, R. (2010). A survey on vision-based human action recognition. *Image and Vision Computing*, 28(6):976990. (Cited on page 19.)

[Ramasso et al., 2007] Ramasso, E., Panagiotakis, C., Pellerin, D., and Rombaut, M. (2007). Human action recognition in videos based on the transferable belief model. *Pattern Analysis and Applications*, 11(1):1–19. (Cited on page 70.)

[Ramasso et al., 2009] Ramasso, E., Panagiotakis, C., Rombaut, M., Pellerin, D., and Tziritas, G. (2009). Human shape-motion analysis in athletics videos for coarse to fine action/activity recognition using transferable belief model. *Electronic Letters on Computer Vision and Image Analysis*, 7(4):32–50. (Cited on page 70.)

[Ranzato et al., 2007] Ranzato, M., Huang, F. J., Boureau, Y.-L., and LeCun, Y. (2007). Unsupervised learning of invariant feature hierarchies with applications to object recognition. *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. (Cited on pages x, 27 and 48.)

[Ranzato et al., 2006] Ranzato, M., Poultney, C., Chopra, S., and LeCun, Y. (2006). Efficient learning of sparse representations with an energy-based model. *NIPS*. (Cited on page 48.)

[Revaud et al., 2009] Revaud, J., Lavoué, G., and Baskurt, A. (2009). Improving zernike moments comparison for optimal similarity and rotation angle retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):627–36. (Cited on page 18.)

[Rifai et al., 2011] Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. (2011). Contractive auto-encoders: Explicit invariance during feature extraction. *ICML*. (Cited on page 47.)

[Roh et al., 2010] Roh, M.-C., Shin, H.-K., and Lee, S.-W. (2010). View-independent human action recognition with volume motion template on single stereo camera. *Pattern Recognition Letters*, 31(7):639–647. (Cited on page 19.)

[Rosten and Drummond, 2006] Rosten, E. and Drummond, T. (2006). Machine learning for high-speed corner detection. *ECCV*, pages 1–14. (Cited on page 21.)

[Rublee and Rabaud, 2011] Rublee, E. and Rabaud, V. (2011). ORB: an efficient alternative to SIFT or SURF. *ICCV*, pages 2564–2571. (Cited on page 24.)

[Ryoo and Aggarwal, 2011] Ryoo, M. and Aggarwal, J. (2011). Stochastic representation and recognition of high-level group activities. *International journal of computer vision*. (Cited on page 70.)

[Ryoo and Aggarwal, 2009] Ryoo, M. S. and Aggarwal, J. K. (2009). Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities. *2009 IEEE 12th International Conference on Computer Vision*, pages 1593–1600. (Cited on pages 37, 55 and 70.)

[Saghafi et al., 2010] Saghafi, B., Farahzadeh, E., Rajan, D., and Sluzek, A. (2010). Embedding visual words into concept space for action and scene recognition. *Procedings of the British Machine Vision Conference 2010*. (Cited on page 54.)

[Salakhutdinov and Hinton, 2007] Salakhutdinov, R. and Hinton, G. (2007). Learning a nonlinear embedding by preserving class neighbourhood structure. In *AISTATS*, volume 11. (Cited on pages 50, 91 and 92.)

[Schmid and Mohr, 1997] Schmid, C. and Mohr, R. (1997). Local Greyvalue Invariants for Image Retrieval. *IEEE Transcations on PAMI*, 5:530–534. (Cited on page 20.)

[Schuldt et al., 2004] Schuldt, C., Laptev, I., and Caputo, B. (2004). Recognizing human actions: a local SVM approach. *Proceedings of the 17th International Conference on Pattern Recognition*, pages 32–36 Vol.3. (Cited on pages xi, 55, 56, 70 and 71.)

[Scovanner et al., 2007] Scovanner, P., Ali, S., and Shah, M. (2007). A 3-dimensional sift descriptor and its application to action recognition. *Proceedings of the 15th international conference on Multimedia - MULTIMEDIA '07*, page 357. (Cited on page 24.)

[Seo and Milanfar, 2010] Seo, H. J. and Milanfar, P. (2010). Action Recognition from One Example. *IEEE transactions on pattern analysis and machine intelligence*, pages 1–16. (Cited on page 69.)

[Shapovalov et al., 2013] Shapovalov, R., Vetrov, D., and Kohli, P. (2013). Spatial inference machines. *CVPR*, pages 2985–2992. (Cited on page 43.)

[Sharma et al., 2013] Sharma, G., Jurie, F., and Schmid, C. (2013). Expanded parts model for human attribute and action recognition in still images. *CVPR*. (Cited on page 41.)

[Shechtman and Irani, 2005] Shechtman, E. and Irani, M. (2005). Space-time behavior based correlation. *CVPR*. (Cited on page 69.)

[Shi et al., 2010] Shi, Q., Cheng, L., Wang, L., and Smola, A. (2010). Human Action Segmentation and Recognition Using Discriminative Semi-Markov Models. *International Journal of Computer Vision*, 93(1):22–32. (Cited on page 70.)

[Shotton et al., 2008a] Shotton, J., Blake, A., and Cipolla, R. (2008a). Multiscale categorical object recognition using contour fragments. *IEEE transactions on PAMI*, 30(7):1270–81. (Cited on page 30.)

[Shotton et al., 2011] Shotton, J., Fitzgibbon, A., Cook, M., Sharp, T., Finocchio, M., Moore, R., Kipman, A., and Blake, A. (2011). Real-time human pose recognition in parts from single depth images. In *CVPR*. (Cited on pages 42, 81, 83, 87, 101, 102, 105, 108 and 111.)

[Shotton et al., 2008b] Shotton, J., Johnson, M., and Cipolla, R. (2008b). Semantic texton forests for image categorization and segmentation. *2008*

IEEE Conference on Computer Vision and Pattern Recognition, pages 1–8. (Cited on page 85.)

[Sim et al., 1999] Sim, D., Kwon, O., and Park, R. (1999). Object matching algorithms using robust hausdorff distance measures. *IEEE Transactions on Image Processing*, 8(3):425–429. (Cited on page 30.)

[Sinhal et al., 2008] Sinhal, A., Chaturvedi, A., and Khan, V. (2008). Human Computer Interaction: Overview on State of the Art. *Internatinal Journal on Smart Sensing and Intelligent System*, 1(1):137–159. (Cited on pages ix, 4 and 5.)

[Sivic and Zisserman, 2003] Sivic, J. and Zisserman, A. (2003). Video Google: A text retrieval approach to object matching in videos. In *Proceedings of the International conference on computer vision (ICCV)*, volume 2, pages 1470–1477. (Cited on pages 35 and 80.)

[Socher et al., 2011] Socher, R., Lin, C., Andrew, N., and DM, C. (2011). Parsing natural scenes and natural language with recursive neural networks. *NIPS*. (Cited on page 43.)

[Song et al., 2013] Song, X., Muselet, D., and Trémeau, A. (2013). Affine transforms between image space and color space for invariant local descriptors. *Pattern Recognition*, 46(8):23762389. (Cited on page 26.)

[Stauffer and Grimson, 2000] Stauffer, C. and Grimson, W. (2000). Learning patterns of activity using real-time tracking. *Pattern Analysis and Machine ...*. (Cited on page 70.)

[Sukthankar and Hebert, 2005] Sukthankar, R. and Hebert, M. (2005). Efficient Visual Event Detection Using Volumetric Features. *ICCV*, pages 166–173. (Cited on page 69.)

[Sun et al., 2009] Sun, X., Chen, M., and Hauptmann, A. (2009). Action recognition via local descriptors and holistic features. *CVPR Workshops on human communicative behavior analysis*, pages 58–65. (Cited on page 69.)

[Ta et al., 2010a] Ta, A., Wolf, C., Lavoué, G., and Baskurt, A. (2010a). Recognizing and localizing individual activities through graph matching. In *2010 Seventh IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 196–203. IEEE. (Cited on page 69.)

[Ta et al., 2010b] Ta, A. P., Wolf, C., Lavoué, G., Baskurt, A., and Jolion, J. M. (2010b). Pairwise features for human action recognition. In *2010*

International Conference on Pattern Recognition, pages 3224–3227. IEEE. (Cited on pages 37 and 55.)

[Ta et al., 2008] Ta, A. P., Wolf, C., Lavoué, G., and Baskurt, A. (2008). 3D object detection and viewpoint selection in sketch images using local patch-based zernike moments. In *International workshop on content-based multimedia indexing (CBMI)*. (Cited on page 18.)

[Taylor et al., 2010] Taylor, G., Fergus, R., Lecun, Y., and Bregler, C. (2010). Convolutional learning of spatio-temporal features. In *European conference on Computer vision*. (Cited on pages 50 and 70.)

[Taylor et al., 2011a] Taylor, G., Hinton, G., and Roweis, S. (2011a). Two distributed-state models for generating high-dimensional time series. *Journal of Machine Learning Research*, 12:1025–1068. (Cited on page 49.)

[Taylor et al., 2011b] Taylor, G., Spiro, I., Bregler, C., and Fergus, R. (2011b). Learning invariance through imitation. In *CVPR*, pages 2729–2736. (Cited on pages 51 and 92.)

[Teague, 1980] Teague, M. R. (1980). Image analysis via the general theory of moments∗. *J. Opt. Soc. Am.*, 70(8):920–930. (Cited on page 18.)

[Teh and Chin, 1988] Teh, C. H. and Chin, R. T. (1988). On Image Analysis by the Methods of Moments. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 10(4):496–512. (Cited on page 18.)

[Tian et al., 2013] Tian, Y., Sukthankar, R., and Shah, M. (2013). Spatiotemporal deformable part models for action detection. *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2642–2649. (Cited on page 40.)

[Torresani et al., 2008] Torresani, L., Kolmogorov, V., and Rother, C. (2008). Feature correspondence via graph matching: Models and global optimization. *ECCV*. (Cited on pages 32, 33 and 80.)

[Turaga et al., 2008] Turaga, P., Chellappa, R., Subrahmanian, V. S., and Udrea, O. (2008). Machine Recognition of Human Activities: A Survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488. (Cited on page 69.)

[Turaga et al., 2010] Turaga, S. C., Murray, J. F., Jain, V., Roth, F., Helmstaedter, M., Briggman, K., Denk, W., and Seung, H. S. (2010). Convolutional networks can learn to generate affinity graphs for image segmentation. *Neural Computation*, 22:511–538. (Cited on page 90.)

[Turk and Pentland, 1991] Turk, M. and Pentland, A. (1991). Face recognition using eigenfaces. *CVPR*. (Cited on page 16.)

[van de Sande et al., 2011] van de Sande, K. E. a., Uijlings, J. R. R., Gevers, T., and Smeulders, A. W. M. (2011). Segmentation as selective search for object recognition. *2011 International Conference on Computer Vision*, pages 1879–1886. (Cited on pages 41, 42 and 43.)

[Villamizar et al., 2006] Villamizar, M., Sanfeliu, A., and Andrade-Cetto, J. (2006). Orientation invariant features for multiclass object recognition. *11th Iberoamerican Congress on Pattern Recognition*. (Cited on pages x and 27.)

[Vincent et al., 2010] Vincent, P., Larochelle, H., and Lajoie, I. (2010). Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research*, 11:3371–3408. (Cited on pages 47 and 49.)

[Wang and Geng, 2008] Wang, L. and Geng, X. (2008). Moving shape dynamics: A signal processing perspective. *Computer Vision and ...*, pages 0–7. (Cited on page 69.)

[Wang et al., 2010] Wang, L., Wang, Y., and Gao, W. (2010). Mining Layered Grammar Rules for Action Recognition. *International Journal of Computer Vision*, 93(2):162–182. (Cited on page 70.)

[Wang et al., 2013] Wang, X., Yang, M., Zhu, S., and Lin, Y. (2013). Regionlets for generic object detection. (Cited on page 41.)

[Weinland et al., 2007] Weinland, D., Boyer, E., and Ronfard, R. (2007). Action Recognition from Arbitrary Views using 3D Exemplars. *2007 IEEE 11th International Conference on Computer Vision*, pages 1–7. (Cited on page 69.)

[Weinland et al., 2006] Weinland, D., Ronfard, R., and Boyer, E. (2006). Free viewpoint action recognition using motion history volumes. *Comput. Vis. Image Underst.*, 104(2):249–257. (Cited on page 19.)

[Weinland et al., 2011] Weinland, D., Ronfard, R., and Boyer, E. (2011). A survey of vision-based methods for action representation, segmentation and recognition. *Computer Vision and Image Understanding*, 115(2):224–241. (Cited on page 69.)

[Willems et al., 2008] Willems, G., Tuytelaars, T., and Gool, L. V. (2008). An efficient dense and scale-invariant spatio-temporal interest point detector. *ECCV*. (Cited on page 22.)

[Winn and Shotton, 2006] Winn, J. and Shotton, J. (2006). The layout consistent random field for recognizing and segmenting partially occluded objects. In *CVPR*, volume 1, pages 37–44. (Cited on pages 80 and 81.)

[Wolf et al., 2012] Wolf, C., Mille, J., Lombardi, E., Celiktutan, O., Jiu, M., Baccouche, M., Dellandrea, E., Bichot, C.-E., Garcia, C., and Sankur, B. (2012). The liris human activities dataset and the icpr 2012 human activities recognition and localization competition. *Technical Report RR-LIRIS-2012-004, LIRIS Laboratory*. (Cited on pages ix, 2 and 3.)

[Xiang and Gong, 2008a] Xiang, T. and Gong, S. (2008a). Activity based surveillance video content modelling. *Pattern Recognition*, 41:2309 – 2326. (Cited on page 70.)

[Xiang and Gong, 2008b] Xiang, T. and Gong, S. (2008b). Incremental and adaptive abnormal behaviour detection. *Computer Vision and Image Understanding*, 111(1):59–73. (Cited on page 70.)

[Yilmaz and Shah, 2005] Yilmaz, A. and Shah, M. (2005). Actions as objects: A novel action representation. *CVPR*. (Cited on page 15.)

[Yokono and Poggio, 2004] Yokono, J. and Poggio, T. (2004). Oriented filters for object recognition: an empirical study. *Proceedings of 6th IEEE interational conference on Automatic Face and Gesture Recognition*, pages 755–760. (Cited on page 27.)

[Yu et al., 2010] Yu, T.-H., Kim, T.-K., and Cipolla, R. (2010). Real-time Action Recognition by Spatiotemporal Semantic and Structural Forests. *Procedings of the British Machine Vision Conference 2010*, pages 52.1–52.12. (Cited on page 22.)

[Yu and Leung, 2006] Yu, X. and Leung, M. (2006). Shape recognition using curve segment hausdorff distance. *International Conference on Pattern Recognition*, pages 441–444. (Cited on page 30.)

[Zhang and Gatica-Perez, 2005] Zhang, D. and Gatica-Perez, D. (2005). Semi-supervised adapted hmms for unusual event detection. *CVPR*, 00(c):0–7. (Cited on page 70.)

[Zhang et al., 2008] Zhang, Z., Hu, Y., Chan, S., and Chia, L. T. (2008). Motion context: A new representation for human action recognition. *ECCV*, (Mc):817–829. (Cited on page 69.)

[Zhao and Pietikäinen, 2007] Zhao, G. and Pietikäinen, M. (2007). Dynamic texture recognition using local binary patterns with an application to facial expressions. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 29(6):915 – 928. (Cited on page 25.)

[Zhou and Kimber, 2006] Zhou, H. and Kimber, D. (2006). Unusual event detection via multi-camera video mining. *ICPR*, 00(c):0–5. (Cited on page 70.)

[Zhu et al., 2010] Zhu, L., Chen, Y., Torralba, A., Freeman, W., and Yuille, A. (2010). Part and appearance sharing: Recursive compositional models for multi-view multi-object detection. *CVPR*. (Cited on page 38.)