DOCTORAL THESIS

Learning to recognize human actions: from hand-crafted to deep-learning based visual representations

Author:
Albert CLAPÉS SINTES

A thesis submitted in fulfillment of the requirements for the degree of Doctor of Engineering and Applied Sciences

in the

Departament de Matemàtiques i Informàtica
Universitat de Barcelona

September 2018

UNIVERSITAT DE BARCELONA

Supervisor	**Dr. Sergio Escalera Guerrero** Dept. Matemàtiques i Informàtica, Universitat de Barcelona, Barcelona, Spain Centre de Visió per Computador, Universitat Autònoma de Barcelona, Bellaterra, Spain
Thesis committee	**Dr. Jordi González Sabaté** Dept. Ciències de la Computació & Centre de Visió per Computador, Universitat Autònoma de Barcelona, Bellaterra, Spain **Dr. Simone Balocco** Dept. Matemàtiques i Informàtica, Universitat de Barcelona, Barcelona, Spain Centre de Visió per Computador, Universitat Autònoma de Barcelona, Bellaterra, Spain **Dr. Cagri Ozcinar** School of Computer Science and Statistics, Trinity College Dublin, Dublin, Ireland

This document was typeset by the author using LaTeX 2ε.

The research described in this book was carried out at Universitat de Barcelona and the Computer Vision Center in Universitat Autònoma de Barcelona.

ISBN: 978-84-948531-2-8

Printed by Ediciones Gráficas Rey, S.L.

Acknowledgements

First of all, I would like to thank my advisor Sergio Escalera. This Thesis would have not been possible without his guidance, encouragement, and patience during all these years. Sergio has not only been my advisor, but also a mentor and an exemple of a hard working, highly motivaded, and very kind person. Muchas gracias, de verdad.

I am also very grateful to Thomas B. Moeslund for giving me the chance to spend some time in his research group at Aalborg University in Denmark. And especial thanks to Malte Petersen for introducing me to the department and to the new city. Mange tak.

I want to acknowledge also the many people who I had the chance to collaborate with. Especial thanks to Oriol Pujol, Àlex Pardo, and Mohamed.

To my brothers in arms, the past and current members of the Human Pose Recovery and Behavior Analysis research group (HuPBA) at University of Barcelona, especially Toni, Miguel Ángel, Miguel Reyes, Xavi, Victor, Dani, Marc, Chip, Cristina, Julio, and Meysam. You have not only been extremely supportive, but also wonderful people to be around with. Gràcies companys.

Vull agraïr enormement als meus pares, en Jaume i na Joana, l'impagable suport que he rebut per part seva durant aquest llarg camí acadèmic. Així com també al meu germà Lluís per estar al meu costat en els moments més difícils d'aquest darrer any. Vos estim.

Gràcies als meus amics per la seva presència i incondicional caliu. Grácies Dani A., Paula, Toño, Lorna, Núria, Manuel, Magí, Paulie i Rubén.

I per acabar, gràcies Gemma. Grácies per ser a la meva vida.

Abstract

Action recognition is a very challenging and important problem in computer vision. Researchers working on this field aspire to provide computers with the ability to visually perceive human actions – that is, to observe, interpret, and understand human-related events that occur in the physical environment merely from visual data. The applications of this technology are numerous: human-machine interaction, *e*-health, monitoring/surveillance, and content-based video retrieval, among others. Hand-crafted methods dominated the field until the apparition of the first successful deep learning-based action recognition works. Although earlier deep-based methods underperformed with respect to hand-crafted approaches, these slowly but steadily improved to become state-of-the-art, eventually achieving better results than hand-crafted ones. Still, hand-crafted approaches can be advantageous in certain scenarios, specially when not enough data is available to train very large deep models or simply to be combined with deep-based methods to further boost the performance. Hence, showing how hand-crafted features can provide extra knowledge the deep networks are not able to easily learn about human actions.

This Thesis concurs in time with this change of paradigm and, hence, reflects it into two distinguished parts. In the first part, we focus on improving current successful hand-crafted approaches for action recognition and we do so from three different perspectives. Using the dense trajectories framework as a backbone: first, we explore the use of multi-modal and multi-view input data to enrich the trajectory descriptors. Second, we focus on the classification part of action recognition pipelines and propose an ensemble learning approach, where each classifier learns from a different set of local spatiotemporal features to then combine their outputs following an strategy based on the Dempster-Shaffer Theory. And third, we propose a novel hand-crafted feature extraction method that constructs a mid-level feature description to better model long-term spatiotemporal dynamics within action videos.

Moving to the second part of the Thesis, we start with a comprehensive study of the current deep-learning based action recognition methods. We review both fundamental and cutting edge methodologies reported during the last few years and introduce a taxonomy of deep-learning methods dedicated to action recognition. In particular, we analyze and discuss how these handle the temporal dimension of data. Last but not least, we propose a residual recurrent network for action recognition that naturally integrates all our previous findings in a powerful and promising framework.

Resum

El reconeixement d'accions és un repte de gran rellevància pel que fa a la visió per computador. Els investigadors que treballen en el camp aspiren a proveir als ordinadors l'habilitat de percebre visualment les accions humanes – és a dir, d'observar, interpretar i comprendre a partir de dades visuals els events que involucren humans i que transcorren en l'entorn físic. Les aplicacions d'aquesta tecnologia són nombroses: interacció home-màquina, *e*-salut, monitoració/vigilància, indexació de videocontingut, etc. Els mètodes de disseny manual han dominat el camp fins l'aparició dels primers treballs exitosos d'aprenentatge profund, els quals han acabat esdevenint estat de l'art. No obstant, els mètodes de disseny manual resulten útils en certs escenaris, com ara quan no es tenen prou dades per a l'entrenament dels mètodes profunds, així com també aportant coneixement addicional que aquests últims no són capaços d'aprendre fàcilment. És per això que sovint els trobem ambdós combinats, aconseguint una millora general del reconeixement.

Aquesta Tesi ha concorregut en el temps amb aquest canvi de paradigma i, per tant, ho reflecteix en dues parts ben distingides. En la primera part, estudiem les possibles millores sobre els mètodes existents de característiques manualment dissenyades per al reconeixement d'accions, i ho fem des de diversos punts de vista. Fent ús de les trajectòries denses com a fonament del nostre treball: primer, explorem l'ús de dades d'entrada de múltiples modalitats i des de múltiples vistes per enriquir els descriptors de les trajectòries. Segon, ens centrem en la part de la classificació del reconeixement d'accions, proposant un assemblat de classificadors d'accions que actuen sobre diversos conjunts de característiques i fusionant-ne les sortides amb una estratégia basada en la Teoria de Dempster-Shaffer. I tercer, proposem un nou mètode de disseny manual d'extracció de característiques que construeix una descripció intermèdia dels videos per tal d'aconseguir un millor modelat de les dinàmiques espai-temporals de llarg termini presents en els vídeos d'accions.

Pel que fa a la segona part de la Tesi, comencem amb un estudi exhaustiu els mètodes actuals d'aprenentatge profund pel reconeixement d'accions. En revisem les metodologies més fonamentals i les més avançades darrerament aparegudes i establim una taxonomia que en resumeix els aspectes més importants. Més concretament, analitzem com cadascun dels mètodes tracta la dimensió temporal de les dades de vídeo. Per últim però no menys important, proposem una nova xarxa de neurones recurrent amb connexions residuals que integra de manera implícita les nostres contribucions prèvies en un nou marc d'acoblament potent i que mostra resultats prometedors.

Contents

List of Figures

List of Tables

Als meus pares, Jaume i Joana, i al meu germà Lluís...

Chapter 1

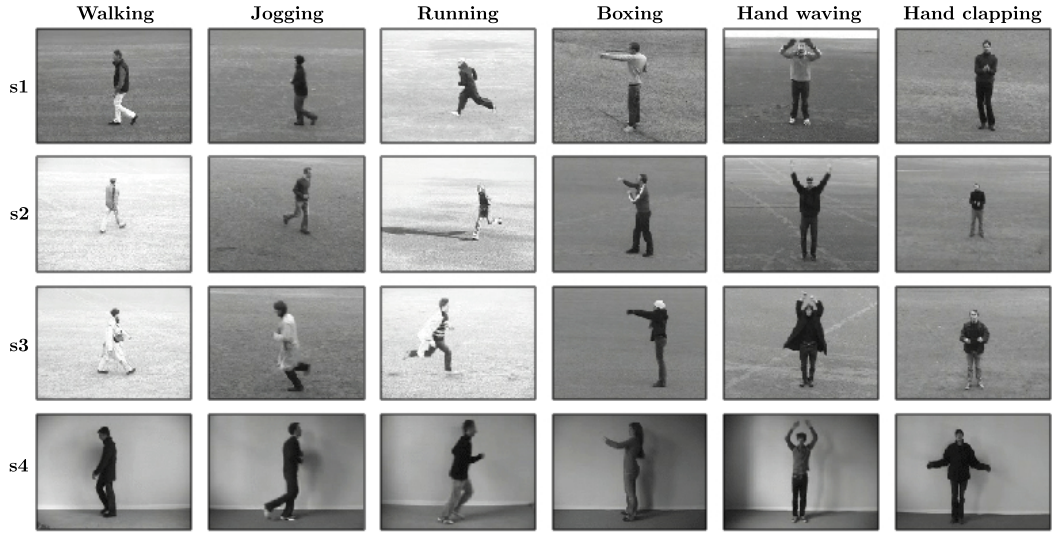
Introduction

In the era of Artificial Intelligence, there are machines that are able to "see". These vision-enabled machines are in practice capable of dealing with a large variety of visual problems: recognition and description of visual content [159, 181, 16, 37], autonomous driving [20, 226], inference of apparent personality traits [4, 219], or detection of cancerous tissue [63, 162], just to name a few. This kind of visual perception has been researched by the computer vision field, part and parcel of AI, which has recently shown major advances thanks to deep learning [150], powerful machinery, and abundance of data. So much so, deep-based image classification networks surpassed human accuracy [147]. In some problems, however, computer vision is still far from the desirable human-level performance.

1.1 Motivation

The recognition of human actions in videos has been studied in computer vision for nearly 30 years. Earliest works [87, 209, 30] approached the problem by making numerous oversimplifications about the nature of human actions, e.g. only a few action categories were being considered, actions were performed in front of a plain non-cluttered background, no object manipulation, etc. Nowadays, we are able to deal with much more realistic video datasets collected from movies or Internet multimedia platforms. Current action classification methods, e.g. [16, 35, 45], obtain more than 90% of accuracy on datasets such as UCF-101 [165] with 101 classes and over 10,000 examples, and more than acceptable results on harder datasets such as HMDB-51 [85] or Kinetics [16]. In Figure 1.1, we compare examples of video frames from UCF-101 categories to the ones from an easier but not much older dataset, KTH [89]

Action recognition can be posed as a fully-supervised pattern recognition problem, consisting of a set of subsequent stages: input, preprocessing, feature extraction, classification, postprocessing, and output. For feature extraction, the so-called hand-crafted approaches rely on manually and carefully designed feature representations [100, 29, 88, 10]. Ensuring a reliable hand-crafted representation often requires going through a painful loop of re-design, re-adjustment of parameters, and re-evaluation. More conveniently, deep learning approaches seamlessly integrate feature extraction and classification – namely end-to-end training – so an optimal feature representation is automatically learned to optimize the recognition performance. The superior results demonstrated by the latter approach [16, 35, 177, 210, 169] caused a drastic shift in action recognition approaches towards deep-based methods. However, hand-crafted features are still valuable given their complementarity with feature learning [66]. Knowing so, most of the deep-based works combine their outputs with the ones from Improved Dense Trajectories (IDTs), obtaining quite often a substantial boost in performance [94, 45, 44, 177, 177].



(A) KTH

(B) UCF-101

FIGURE 1.1: Two examples of action classification datasets. The older KTH [151] with 9 actions and plain background. In contrast, the newer UCF-101 [165] is a much more complex dataset consisting of 101 actions and with different backgrounds and objects present

Optical flow (OF) has been shown to be a particularly useful motion cue in both hand-crafted and deep-based methods for action recognition. A popular trend is to combine spatial and motion cues [185, 159, 177, 47]. Besides the spatial and motion cues obtained from the RGB modality, there exist other kinds of input that are suitable for action recognition scenarios. The Depth modality has been vastly exploited in close-range in-door action classification [38, 137, 193, 99]. Cost-effective RGB-Depth sensors, e.g. Microsoft Kinect or Intel RealSense, capture continuous streams of depth maps in real-time where each values indicates the distance of the captured points to the sensor. Using then its intrinsic parameters, it is possible to reconstruct the 3D view of the scene and register the 3D points with their corresponding RGB pixel values. Besides such geometric cue, Depth also provides a fast and robust way to detect people and infer their skeletal joints positions [157]. The skeletal cue is widely used for action, activity, and gesture recognition [221, 106, 38].

Prior to classification, the feature representations derived from multiple cues (or modalities) need to be properly combined to successfully tackle the recognition task. In hand-crafted approaches, one possibility is to combine the descriptors from different information sources in an early fusion fashion, i.e. the different video descriptors are concatenated in a larger vector and then fed to a classifier [185, 183, 211]. Any missleading source of information might however harm the overall performance of the method. In contrast, deep-based approaches usually adopt the more convenient late-fusion scheme, averaging the softmax classification scores from the network streams corresponding to different cues [78, 159, 177]. In this context of late-fusion, Ensemble Learning brings more sophisticated late-fusion strategies that can be applied in both hand-crafted and deep-learning based approaches, especially benefitial as we increase the number of streams/models/classifiers [86].

A crucial aspect in action recognition is to effectively model spatial and temporal information. While spatial relations and shorter range spatiotemporal dynamics are easier to capture, modeling longer term temporal information requires more complex models accounting for the temporal dimension. Hand-crafted approaches typically relied in aggregation processes. such as 3D convolutional networks [16, 177], deep sequential models [210, 169], or cumbersome mid-level feature representations in hand-crafted approaches [48]. In particular, 3D-CNNs achieve significantly better results than RNNs on shorter videos from current action classification datasets, 3D convolution and 3D poolings operations for very long temporal extents are computationally unfeasible. RNN-like models are, in contrast, able to model very long temporal sequences, but only modeling temporal information. To get the best of both worlds, the rich spatiotemporal representation provided by 3D-CNN and the long range temporal modeling from RNN-like networks, both can be combined [195, 196]. The 3D-CNN can act as a volumetric spatiotemporal feature extractor and the RNN model the temporal information from the 3D-CNN spatiotemporal descriptors from video volumes. Also RNN layers can be stacked on top of each other to create a deep recurrent networks that is able to learn high-order temporal features [94, 35, 221, 37].

1.2 Contributions of the Thesis

In this Thesis, we tackle the challenges of action recognition from both hand-crafted and deep-learning based approaches. In the hand-crafted scenario, we target different levels of the pattern recognition pipeline, mainly input, feature extraction, and classification, following different but complementary lines of research. We then

move towards deep-learning to seamlessly integrate those three lines together. The contributions of the Thesis can be summarized as follows:

1. **Multi-modal and multi-view Dense Trajectories for action detection** (input and feature extraction). In a monitoring scenario, we combine two RGB-Depth visual sensors with an inertial sensor placed on the dominant hand's wrist of the subject to detect performed actions. In the vision module, we compute Multi-modal multi-view Dense Trajectories (MmDT) enriching the set of trajectory-aligned descriptors from [185] with a geometric descriptor, i.e. Histogram of Oriented Normals, computed from the depth modality. Then, following a Bag-of-Visual-Words aggregation, we generate a codebook for each kind of trajectory-aligned descriptor. The codebooks are mined with trajectories from the two views. For the temporal detection, we slide a temporal window whose descriptor is efficiently computed using integral sum over the pre-computed per-frame BoVW descriptors. The detection outputs of the vision module are late-fused with the outputs of the inertial module following different fusion strategies. The proposed framework is validated in a real-case scenario with elderly from an elder home, showing the effectiveness of multi-modal approaches.

2. **Ensembling action classifiers** (classification and postprocessing). We study how the late-fusion via an ensemble of classifiers trained on different action features performs compared to early fusion in an holistic classifier. For the action features we base on Dense Trajectories (DT) [185] and space-time interest point (STIP) [88] features. DT features consist of the four original trajectory-aligned descriptors, each of very different nature, i.e. on trajectory shape, appearance, motion, and first derivative of motion. We test both training each individual classifier on a feature type and training a forest of learners each on a random subset of the features. During ensemble classification, the late-fusion strategy bases on the Dempster-Shafer rule of combination instead of simply averaging the class scores from the classifiers. The method outperforms competing methods in state-of-the-art benchmarking dataset, UCF-101 [165].

3. **A novel hand-crafted feature representation to model long-term temporal dynamics for action classification** (feature extraction). We model the evolution of IDT features[183] not only throughout the entire video sequence, but also on subparts of the video. Subparts are obtained using a spectral divisive clustering [52] that yields an unordered binary tree decomposing the entire cloud of trajectories of a sequence. We then compute videodarwin [47] on video subparts, exploiting more finegrained temporal information and reducing the smoothing degradation from videodarwin. After decomposition, we model the evolution of features through both frames in subparts and descending/ascending paths in tree branches, namely node-darwintree and branch-darwintree. For the final classification, we define our darwintree kernel representation and combine it with holistic videodarwin. Our approach achieves better performance than standard videodarwin and defines the current state-of-the-art on UCF-Sports and Highfive action recognition datasets.

4. **A comprehensive study of current deep-learning based methods for action recognition**. We review both fundamental and cutting edge methodologies reported in the last few years. We introduce a taxonomy that summarizes important aspects of deep learning when approaching the task: architectures, fusion

strategies, and benchmarking results on public datasets. More in detail we discuss the more relevant works with emphasis on how they treat the temporal dimension of the videos, other highlighting features, and their challenges.

5. **Residual recurrent networks for action recognition** (feature extraction, classification, and postprocessing). After extensively studying current works on deep-based action recognition, we contribute to the deep-based sequential modeling with our multi-layer Residual Recurrent Neural Network (Res-RNN). While most RNN-based works base 2D-CNNs to extract per-frame descriptors, we build on top of the last convolutional layer of a pre-trained 3D-CNN that provides rich spatiotemporal descriptors of video volumes. We then model the temporal relationships of such through the stack of multiple recurrent layers with residual connections between layers. Our novel Res-RNN exploits appearance and motion cues into two separate streams and the outputs are late-fused using element-wise product instead of the more common summation operation. Our method outperforms other deep sequential-based methods in HMDB-51 and obtains competitive results in UCF-101. We also demonstrate how the combination with Improved Dense Trajectories (IDTs) [183] further increases our performance, especially in the case of HMDB-51.

1.3 Publications

This Thesis resulted in the following list of publications (directly or indirectly related to its central theme):

1.3.1 Journal papers

- Clapés, A., Reyes, M., & Escalera, S. (2013). Multi-modal user identification and object recognition surveillance system. Pattern Recognition Letters, 34(7), 799-808.
- Reyes, M., Clapés, A., Ramírez, J., Revilla, J. R., & Escalera, S. (2013). Automatic digital biometry analysis based on depth maps. Computers in Industry, 64(9), 1316-1325.
- Cepero, A., Clapés, A., & Escalera, S. (2015). Automatic non-verbal communication skills analysis: A quantitative evaluation. AI Communications, 28(1), 87-101.
- Palmero, C., Clapés, A., Bahnsen, C., Møgelmose, A., Moeslund, T. B., & Escalera, S. (2016). Multi-modal rgb–depth–thermal human body segmentation. International Journal of Computer Vision, 118(2), 217-239.
- Clapés, A., Pardo, À., Vila, O. P., & Escalera, S. (2018). Action detection fusing multiple Kinects and a WIMU: an application to in-home assistive technology for the elderly. Machine Vision and Applications, 1-24.

1.3.2 International conferences and workshops

- Clapés, A., Reyes, M., & Escalera, S. (2012, July). User identification and object recognition in clutter scenes based on rgb-depth analysis. In International Conference on Articulated Motion and Deformable Objects (pp. 1-11). Springer, Berlin, Heidelberg.

- Pardo, À., Clapés, A., Escalera, S., & Pujol, O. (2014). Actions in context: system for people with dementia. In Citizen in Sensor Networks (pp. 3-14). Springer, Cham.
- Konovalov, V., Clapés, A., & Escalera, S. (2013, October). Automatic Hand Detection in RGB-Depth Data Sequences. In CCIA (pp. 91-100).
- Cepero, A., Clapés, A., & Escalera, S. (2013). Quantitative Analysis of Non-Verbal Communication for Competence Analysis. In CCIA (pp. 105-114).
- Mogelmose, A., Bahnsen, C., Moeslund, T., Clapés, A., & Escalera, S. (2013). Tri-modal person re-identification with rgb, depth and thermal features. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 301-307).
- Bagheri, M., Gao, Q., Escalera, S., Clapes, A., Nasrollahi, K., Holte, M. B., & Moeslund, T. B. (2015). Keep it accurate and diverse: Enhancing action recognition performance by ensemble learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 22-29).
- Escalante, H. J., Ponce-López, V., Wan, J., Riegler, M. A., Chen, B., Clapés, A., ... & Müller, H. (2016, December). ChaLearn Joint Contest on Multimedia Challenges Beyond Visual Analysis: An overview. In ICPR (pp. 67-73).
- Ponce-López, V., Chen, B., Oliu, M., Corneanu, C., Clapés, A., Guyon, I., ... & Escalera, S. (2016, October). Chalearn lap 2016: First round challenge on first impressions-dataset and results. In European Conference on Computer Vision (pp. 400-418). Springer, Cham.
- Asadi-Aghbolaghi, M., Clapes, A., Bellantonio, M., Escalante, H. J., Ponce-López, V., Baró, X., ... & Escalera, S. (2017, May). A survey on deep learning based approaches for action and gesture recognition in image sequences. In Automatic Face & Gesture Recognition (FG 2017), 2017 12th IEEE International Conference on (pp. 476-483). IEEE.
- Clapés, A., Tuytelaars, T., & Escalera, S. (2017, October). Darwintrees for action recognition. In The IEEE Int. Conf. on Computer Vision (ICCV) (pp. 3169-3178).
- Asadi-Aghbolaghi, M., Clapés, A., Bellantonio, M., Escalante, H. J., Ponce-López, V., Baró, X., ... & Escalera, S. (2017). Deep learning for action and gesture recognition in image sequences: A survey. In Gesture Recognition (pp. 539-578). Springer, Cham.

1.4 Thesis outline

The Thesis is divided into two main parts. The chapters within each part are similarly structured, most of them including an introduction, specific related work, method, and experimental results.

Part I consists of four chapters dedicated to hand-crafted approaches for the task of action recognition. Chapter 2 is a preamble reviewing the state-of-the-art of local spatiotemporal representations and a brief introduction to the Dense Trajectories framework (backbone of the three following chapters in Part I). Chapter 3 discusses

the multi-modal and multi-view framework for action detection. Chapter 4 introduces the ensemble learning on action features. Chapter 5 studies the hierarchical evolution of features as a way to model spatiotemporal information in action classification.

Part II is divided into two chapters. Chapter 6 motivates the shift from handcrafted to deep learning and discusses the state-of-the-art of deep learning in the field of action recognition. Chapter 7 presents the stacked residual recurrent model for the task of action recognition.

Finally, Chapter 8 concludes the Thesis with final discussion and conclusions.

Please note the symbol definitions are not shared among the two parts. A unified set of symbols would have been impractical, so each of them is mathematically self-contained.

Part I

Hand-crafted methods

Chapter 2

Preamble: Local spatiotemporal feature representations

The latest hand-crafted methods experimentally demonstrated local context representations to be highly effective for human action recognition in realistic videos [183, 185, 129, 47]. Compared to global representations, feature descriptors from local regions are more invariant to body deformations, heavy occlusions, changes in the point of view, camera motion, and illumination changes [23, 198, 135, 110]. Moreover, local features avoid relying on error-prone preprocessing steps, e.g. background subtraction [164, 208] or tracking [91, 114]. The high dimensionality of videos however leads to the obtention of a large and arbitrary number of local spatiotemporal regions per video/clip. How to sample and describe the spatiotemporal regions, as well as how to aggregate the resulting descriptors into a compact representation for the video/clip, are key issues that need to be addressed.

In this chapter, Section 2.1 reviews different approaches to the location, description, and aggregation of local spatiotemporal features, with special emphasis on the Dense Trajectories (DT) framework [185, 183]. Then, Section 2.2 introduces Dense Trajectories more in detail, used in the following chapters as the base of our feature extraction processes.

2.1 Related work

Local representations typically relied on the detection of sparse spatiotemporal keypoints. The original idea of [62] of corner detection in images was extended to the spatiotemporal domain by [88]. It defined a spatiotemporal second-moment matrix composed of first order spatial and temporal derivatives and determined points that are local maxima to the harris-like function combining the determinant and trace of that matrix. In practice, these corresponded to local spatiotemporal neighborhoods with non-constant motion. [36] argued spatiotemporal corners were not well suited for subtle and gradually changing motion, so they proposed the Cuboid detector with a response function based on a 2D gaussian smoothing kernel along the spatial dimension and a quadrature pair of 1D Gabor filters in time. It provided the larger responses on periodic motions, while still being able to detect spatiotemporal corners and avoiding activations on translational motions. [200] localized interest points both in the spatiotemporal domain at multiple scales simultaneously by using the determinant of the Hessian matrix as a saliency measure; and doing so efficiently by removing the iterative scheme from [88].

Differently from those, [186] proposed dense sampling of 3D patches at regular positions and different scales. The approach outperformed the keypoint-based localization of local features in realistic datasets, at the expense of producing 15-20

times more candidate regions. Motivated by these results, [185] defined the Dense Trajectories (DT) framework, which uses dense optical flow maps to densely subsample regions containing potentially meaningful motion patterns. While still being a dense form of sampling, it considerabily reduces the computational cost of regular space-time sampling from [186]. The trajectories are constructed by tracking every pixel motion following their corresponding optical flow vectors. Then, the surrounding image patches along 15-frame trajectories are described using appearance and motion descriptors and, finally, concatenated into the trajectory descriptors. The aggregation of trajectory descriptors is done following a Bag-of-Visual-Words (BoVW) approach, a general pipeline consisting of a codebook generation (k-Means clustering of trajectories) followed by vector quantization (counting assignation of video/clip trajectories to cluster prototypes). [183] introduced Improved Dense Trajectories (IDT), which use warped flow [73] to alleviate camera motion during the optical flow estimation and Gaussian Mixture Models and Fisher Vectors to replace, respectively, k-Means and vector quantization.

After the success of DT, the more recent efforts have been put into building mid-level representations to further improve action classification performance. [120] clusters trajectories into spatiotemporal groups and learn to assign a weight to each spatiotemporal group in such a way that more discriminative (weighted) fisher vectors are obtained, attenuating the effect of irrelevant groups of trajectories for each particular action. One criticism on BoVW- and FV-like approaches these do not take into account spatial and temporal relations among visual words. In order to capture complex and long-term motions of spatiotemporal parts, as well as their relations, [52] proposes the hierarchical decomposition of the cloud of trajectories from a video into a tree. The nodes of a tree define different groupings of trajectories that are aggregated using BoVW. For classification, they define a kernel based on the similarity between a pair of trees – each representing an action instance – that is on the accumulated similarities between tree edges (parent-child node pairs). In a similar fashion, [129] stacks two levels of fisher vectors to model semantic information from videos in a hierarchical way. In the first layer, fisher vectors are computed over the IDTs on densely sampled multi-scale cuboids. Then, a second (stacked) fisher vector representation is computed on FVs from cuboids to obtain the final FV video representations. Focused on better modeling temporal information in videos, [47, 48] compute IDT-based FVs in a per-frame basis. Then a linear/ranking model learns to regress/rank the orderings of the video frames based on the per-frame representations.

DTs and IDTs have been successfully applied to different scenarios than action classification. [145] temporally detect cooking activities using DT+BoVW on a multi-scale sliding window and detections are filtered using a non-maximum supression filter. [84] combines a per-frame IDT+FV representation with a composite Hidden Markov Model to recognize and temporally segment breakfast preparation activities, where each action unit (or subactivity) is modeled by a different HMM and the transitions between actions are modeled by a language grammar. In this same context, [142] propose IDT+FV but instead modeling action units via recurrent neural networks that are combined with a coarse probabilistic. [54] and [215] both locate actions in space and time generating localizaton (tubelet) proposals from DTs. Following this same idea, the trajectiories DT/IDT have been used to pool deep features from CNN filter responses [189]. Later in Chapter 6 we discuss the combination of trajectories with deep-based approaches more in depth.

2.2 Dense trajectories in a nutshell

The Dense Trajectories framework (DT) [185] is based on the idea of dense sampling of local spatiotemporal features, but only considering moving parts – as these are assumed to contain the dynamic information that can be relevant for the action recognition task. dense sampling has shown to provide smoother trajectories (illustrated in Figure 2.1) and hence better results.

Next, we first review the trajectory construction and the *trajectory descriptor* calculation, and then the computation of the trajectory-aligned descriptors. The different steps are depicted in Figure 2.2.

2.2.1 Trajectory construction and shape descriptor

The trajectories are sampled at different spatial scales. For each scale, points are sampled at regular intervals of W pixels across the two spatial dimensions, and then tracked during L frames using the directions of dense optical flow vectors [43]. The length L of the trajectory is limited to a few frames to avoid drifting during the tracking of the point. Points of subsequent frames are concatenated to form a trajectory $(P_t, P_{t+1}, P_{t+2}, \ldots)$, where $P_t = (x_t, y_t)$. The trajectories that reach length L are removed from the tracking. Also, the framework enforces the sampling of points when some $W \times W$ neighborhood does not contain any point being tracked. However, points in homogeneous image regions or those presenting large displacements are discarded during the trajectory construction.

FIGURE 2.1: KLT feature tracker [172]) versus Dense Trajectories. Reprinted from [185]

Let describe a trajectory shape $\mathcal{T} = (\Delta P_t, \ldots, \Delta P_{t+L-1})$ by the sequence of displacement vectors $\Delta P_t = (P_{t+1} - P_t)$. We can now compute the *trajectory descriptor* (TS) as follows:

$$\mathcal{T}' = \frac{(\Delta P_t, \ldots, \Delta P_{t+L-1})}{\sum_{j=t}^{t+L-1} ||\Delta P_j||}$$

.

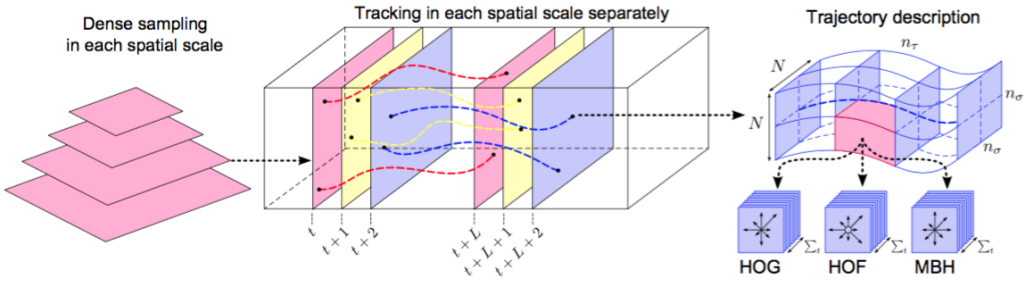


FIGURE 2.2: Trajectory construction and computation of trajectory-aligned descriptors. Reprinted from [185]

2.2.2 Trajectory-aligned descriptors

Other very robust and reliable hand-crafted descriptors are included to complement the TS descriptor. More precisely, *Histogram of Oriented Gradients* (HOG) [29],*Histogram of Oriented Flows* (HOF) [88], and *Motion Boundary Histogram* (MBH) [185] are computed in the $N \times N$ image patches centered at the trajectory points along its path. This $N \times N \times L$ spatiotemporal tube is then sub-divided into $n_x \times n_y \times n_t$ cells, so the trajectory-aligned descriptors are computed and averaged within their corresponding cells. The final trajectory representation is the concatenation of the four descriptors (TS, HOG, HOF, and MBH) from the $N \times N \times L$ spatiotemporal tube is then sub-divided into $n_x \times n_y \times n_t$ cells. HOG and HOF orientations are quantized into 8 and 9 bins respectively. Since MBH computes the derivative of the optical flow separately for horizontal and vertical motion components, this leads to two histograms, each of 8 bins.

In the following chapters, we use the default values of the previously described parameters ($W = 5$, $L = 15$, $N = 32$, $n_x = n_y = 2$, and $n_z = 3$). Doing so, the final trajectory descriptors are of length 436, the 10 first elements being information associated to the trajectories. The length of TS is $L * 2 = 15 * 2 = 30$. HOG and HOF orientations are quantized into 8 and 9 bins respectively. Since MBH computes the derivative of the optical flow separately for horizontal and vertical motion components, this leads to two histograms, each of 8 bins. Therefore, the length of these descriptors in the final trajectory description are $2 * 2 * 3 * 8 = 96$ for HOG, $2 * 2 * 3 * 9 = 108$ for HOF, and $2 * 2 * 3 * 8 = 96$ for MBH along each spatial axis.

Since trajectories are local representations, the next step is to aggregate them into a compact representation for the video. However, we omit these parts of the framework in here since these processes differ with the problems presented in the following chapters of this dissertation.

Chapter 3

Multi-modal and multi-view Dense Trajectories for action detection

We propose a two-module system combining two RGB-Depth (or RGB-D) visual sensors together with a wireless inertial movement unit (WIMU) in order to detect actions of the daily living in a real-world scenario with elderly people. The two RGB-D sensors face to each other, so to have a complete occlusion-free view of the scenario. Their streams are processed to compute multi-modal multi-view Dense Trajectories (MmDT). In particular, the trajectories are enriched with a Histogram of Oriented Normals (HON) descriptor computed from the depth maps, complementing the trajectory shape (TS), Histogram of Oriented Gradients (HOG), Histogram of Oriented Flows (HOF), and Motion Boundary Histogram (MBH) descriptors. Trajectories are then bagged into multi-view codebooks. Following the approach of [185], a codebook is generated for each kind of description. Then, in order to perform the classification, a multi-class Support Vector Machine (SVM) with Chi-square kernel combines the descriptions at kernel level. For the detection itself, a sliding window approach is followed, so that Bag-of-Visual-Words (BoVW) for the windows are built from the extracted trajectories in an efficient ' integral" way.

In parallel, an egocentric module is in charge of performing gesture detection. In particular, the WIMU used is a Shimmer sensor placed in the elderly's dominant wrist. In order to recognize the gesture we first preprocess the data in order to extract relevant information such as accelerometer, rotation angles, and jerk. Then, we select a set of models from the sequences, which are used to obtain alignment distances (costs) by means of a Dynamic Time Warping (DTW) algorithm. During the process of detecting the gestures, DTW performs the alignments respect to the models and determines if new gestures being performed by comparing alignment-to-model costs to a set of learnt thresholds. Finally, the outputs of the two modules are combined, to provide more accurate detection results.

The system is validated in a real-case dataset with elderly people from the SARQuavitae Claret elder home. We guided the elders on different scripted scenarios involving gestures while interacting with objects: "taking a pill", "drinking from a glass", "eating from a plate", and "reading a book". We recorded a total of 31 different sequences of 1-3 minutes of duration each, with 14 elderly people. The dataset is manually annotated with the begin-end of the actions and gestures. Gestures are defined at a different level of annotation than actions, so as to be more atomic and palliate the inherent noisiness from these kind of devices. The obtained results show the effectiveness of the system. Moreover, the learning-based fusion demonstrates the success of this kind of multi-modal approaches.

This chapter is organized as follows: Section 3.1 briefly reviews the related work that is specific to this chapter, more precisely on in-home monitoring systems and

also works utilizing RGB-D technology and WIMUs for action recognition; Section 3.2 describes the dataset, hardware, and acquisition settings; Section 3.3 introduces the system. Both vision and wearable modules are explained more in depth in Subsection 3.3.1-3.3.2; and, finally, Section 3.4 presents the results got by the different modules of the system and their integration for final detection output.

3.1 Related work

There exist many vision systems for action/activity monitoring applied to in-home care. [113] uses location cues to determine actions. Models of spatial context are learned employing a tracker that uses a coarse ellipse shape and a particle filter to cope with cluttered scenes seen from a fisheye camera. Despite being tested in a realistic environment, the learned models are not transferable to other scenarios. Moreover, the location of the human in the scene is not enough to discern among certain activities. [39] defines 8 different activities and model the transitions from one to another by means of a Hidden Markov Model (HMM). They segment people silhouettes by means of simple but adaptive background subtraction and characterize the silhouette poses in frames with a set of three handcrafted features: height of center of mass, vertical speed of the center, and sparsity of points. [28] proposes to recognize events in a knowledge-driven approach by using an event modeling framework. In knowledge-based approaches, events need to be defined by an human expert.

The apparition of RGB-D devices, such as Kinect – supposed the emergence of new systems and techniques that could be applied to in-home assistive technology. In [148], the system monitors human activities while seeking for signs of limb or joint pain. The work defined a set of 7 pain gestures performed by people on an average age of 40 and above. They report good results using MLP for classification on the skeletal features extracted from Kinect. However, the pain gestures are static and in a highly controlled environment. [11] present a system to monitor and control elderly people in a smart house environment. It recognizes gestures and communicates them through the network to a caregiver. They match candidate gestures to template simply by computing a distance. In this case, quantitative results are not presented. [32] propose a system capable of recognizing full body actions, such as walk, jump, grab something from the floor, stand, sit, and lie (on the floor). The action class is determined in a heuristic rule-based fashion based on skeletal joints' positions. [13] detect pointing gestures in a smart bedroom to facilitate the elderly people interaction with the environment. In [8], the authors are detecting very simple activities such as standing, sitting, and the standing-sitting transition. In their work, silhouettes are subtracted first using a background subtraction algorithm. From these silhouettes, image moments are extracted, which are then clustered using fuzzy clustering techniques to produce fuzzy labels in the activity categories. During the functioning of the system, the classification is done by a fuzzy clustering prototype classifier. They test several imaging technologies with night vision capacities, including Kinect. The system is tested at a senior house facility with older adults with physical health issues. [220] define a set of 13 activities of the daily living and the skeletal features are used to generate a codebook of poses. Then, in a BoVW approach, poses from individual frames are binned in a histogram representing the pre-segmented activity videos and classified using a SVM classifier. The system is tested in a controlled environment and is not performing detection in continuous streams. [132] utilize a Gaussian mixture models-based HMM for human daily activity recognition using 3D skeleton features. [58] introduce a system prototype for telerehabilitation for

post-stroke patients. They monitor the range of motion of different limbs during the realization of daily living activities using a Kinect and accelerometers attached to objects. However, they only monitor the min and max and compare to the ones from the previous day and quantify the progress. Their proposal does not tackle the action recognition task. Nonetheless, being able to recognize particular activities will make the evaluation of ranges of motions much more meaningful for therapists.

Skeleton features are widely used for action/gesture recognition in the literature of in-home assistive systems. The systems report very good results using these kind of features, but most of them are applied in very controlled scenarios in which the body is fully visible. However, the skeleton is not reliable in the presence of body occlusions or non-frontal camera angles – as it often occurs when dealing with real world situations. Fortunately, there exist many approaches to action recognition that do not require the use of such features in the literature of computer vision. Next, we review some of the state-of-the-art of action detection and recognition that could deal with some of the aforementioned problems.

There are also works combining RGB-D vision with the inertial information provided by WIMUs. [31] combine the Kinect sensor with 5 WIMUs in order to recognize activities. They fuse the multiple modalities in an early fusion fashion. They use a sliding window approach together with a set of binary trained MLP classifiers to perform the detection in one-vs-all setup. [98] perform hand gesture recognition fusing the inertial and depth data within a HMM framework, demonstrating an overall improvement. Nonetheless, the method is tested on a relatively simple dataset of gestures with 5 gestures, i.e., "wave", "hammer", "punch", "draw an X", "draw a circle", thus with considerable inter-class variability and at a relatively small distance to the Kinect camera. Our dataset was recorded in a much more uncontrolled scenario, actions are observed from a farther position, we deal with occlusions, and have much smaller inter-class variability; in fact, some classes can be only distinguished mostly by considering the interacted object, e.g., drinking and taking the pill. [70] present some preliminary analyses for fusing Kinect and inertial sensors' data in order to monitor intake gestures. However, they do not present any performance on the task, but only some qualitative results.

3.1.1 RGB-D action detection in video

Different approaches exist to action detection in video. Typically, the detection consists in localizing the action within the video either in the temporal domain [53] or both in space and time [199, 57, 71]. In fact, the spatial localization of the action makes the problem more demanding. In our case, and since we are not intended to do that, we focus on reviewing only temporal localization methods. These often use a variable-size sliding window in which actual action classification is performed. There exist more cumbersome approaches, like the one of [53], that break down actions into sub-actions (or *actoms*) and model explicitly the length of the window for each class. Unfortunately, it requires expensive manual annotations of *actoms*. In this work, we simply convolve a sliding window of different sizes and perform action classification in it. Next, we review the state-of-the-art methods for action classification using depth information.

Many classification works take advantage of the skeleton data provided by Kinect

[141, 9]. Skeletal data demonstrated their reliability, even outperforming low/mid-level features when there are not heavy occlusions [75][1]. Moreover, the skeleton representation is low-dimensional data, which makes potentially easier the learning of transitions in sequential models, e.g. HMM, and also reduces the cost of aligning action candidates to class exemplars in the DTW. Not only sequential models, but also template-based methods are also applicable in depth data cue [95, 92]. [95] extend the energy-based method from [30] to depth data, extracting the motion-energy features in the three cartesian planes got from depth maps separately to later build a spatio-temporal pyramid cuboid representation of the action videos.

Regarding local features, [67] propose a Bag-of-Visual-and-Depth-Words framework for gesture recognition. The authors use the STIP detector separately in RGB and depth modalities, and then describe the color interest points using HOG and HOF, and the depth interest points with VFH+CRH[2]. Then, in order to compute the global representation of gestures they use spatio-temporal pyramids and a Bag-of-Visual-Words approach. Finally, the global gesture representation is matched to the training samples using lazy learning.

Despite locally extracted depth features have been used for action/gesture recognition, there is not (to our extent) any work using Dense Trajectories in RGB-D videos. In this work, we propose to enrich the description of the trajectories consisting of the track's relative position, HOG, HOF, and Motion Boundary Histogram (MBH) with a Histogram of Oriented Normals (HON).

Multi-modal fusion techniques

Focusing on the fusion part, there are two different strategies: combine the values for each modality at the beginning of the pipeline (feature-level), named early fusion or after computing the prediction values for each of the inputs (decision-level), late fusion.

State-of-the-art late-fusion strategies use the scores given as outputs from early stages. [60] use product, sum and weight as fusion strategies for probabilities. [115] present a bayesian model based on the scores given by the classifiers. [163] using a SVM trained with the concatenation of the outputs. [224] follow a top-down approach: in a first stage, a coarse label is generated and, then, it is fed to the fusion module which gives a fine-grained category. Finally, in [18] two different strategies are compared, a uniform average and a weighted average, the latter giving better results.

3.2 Data, hardware, and acquisition setup

This section describes the dataset, the hardware used to record the data, and the system's physical settings and software infrastructure.

3.2.1 Data

The SARQuavitae Claret dataset consists of a total of 31 sequences of 1-3 minutes of duration each, with 14 elderly people performing different scripted scenarios that

[1] This is not stated in the published manuscript, but in an errata document. Check [75] and the errata document for more detail: `http://jhmdb.is.tue.mpg.de/show_file?filename=Errata_JHMDB_ICCV_2013.pdf`

[2] The concatenation of the Viewpoint Feature Histogram (VFH) and Camera Roll Histogram (CRH).

	Modules	
	Vision	Wearable
Task	Action detection (4 actions)	Gesture spotting & recognition (4 gestures)
Hardware	2x RGB-D sensors (Kinect)	1x WIMU (Shimmer)
Type of data	RGB Depth	Accelerometer Gyroscope Magnetometer
No. sequences	31	
No. subjects	14	
No. frames	3,747 + 3,701	36,858
No. actions (gestures)	86 (162)	
General challenges	Elderly subjects, uncontrolled behavior	
Specific challenges	Ambient light reflections and shadows Small objects Low framerate Depth noise	Gesture intra-inter variability Device noise

TABLE 3.1: Summary of the SARQuavitae Claret dataset characteristics

involve the realisation of activities of the daily living: "taking pill", "drinking", "eating", and "reading". These activities emerge from the interaction with four different objects: a plastic eating plate that may appear with a plastic fork, a plastic cup, a photography book, and a small two-lid pillbox. Yet other irrelevant objects appear. For instance, a juice tetra-brick or the objects the elderly bring to the scene, e.g., purses, wallets, or a walking stick. Table 3.1 summarizes the most important aspects of the dataset.

We defined and manually annotated two levels of annotations: activities and gestures. Whereas the vision part directly performs recognition on activities, the wearable module requires more atomic annotations - we named *gestures*. The number of gesture classes coincide with the number of activities, those being "spoonful", "drink", "turn page", and "take-pill". These more atomic annotations reduce intra-class variability of the original activities, making the task of the wearable model easier. Figure 3.1 introduces both the visual data (frames), whereas Figure 3.2 and Figure 3.3 show respectively the interaction objects and gesture data (accelerometer's gestures).

For the manual annotation task, we first synchronized the streams of the two cameras along with the one from the sensor. Then, we annotated both activities and gestures at frame-level. In the case of the activities, beginning and end coincide with the interaction with each particular object – an interaction being considered the intentional manipulation of an object. For instance, for "eating" the activity starts when the subject reaches the dish and/or fork, continues during a variable number of spoonfuls, and finishes when the subject drops the fork after the last spoonful. On the other hand, we defined the beginning and the end of gestures in different ways depending on the class. For "take-pill", "drink", or "spoonful", the gesture begins when the hand – already touching either the pillbox, cup, or fork – starts accelerating towards the mouth of the subject and ends in the very same instant when the hand starts accelerating again moving away from the mouth. In the case of "turn page", the gesture begins when the page starts to be turned and ends when it has been completely turned.

The dataset presents several challenges that have to be addressed:

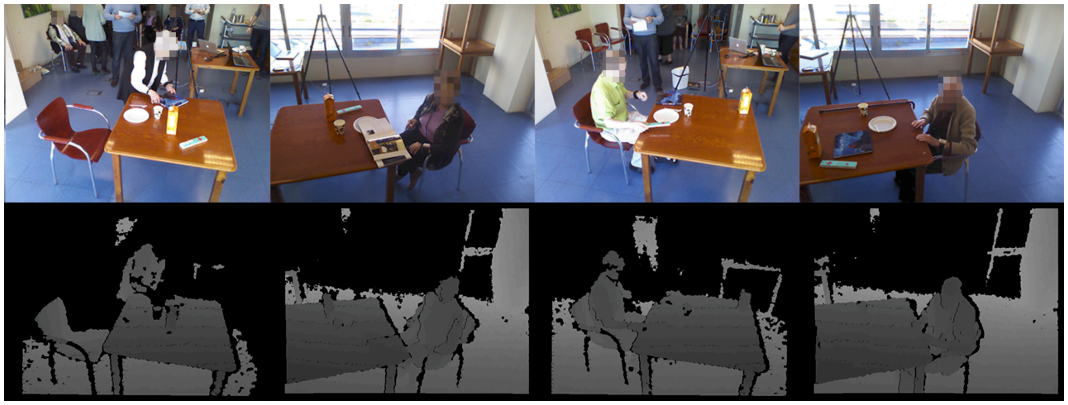


FIGURE 3.1: The visual data acquired, with color frames in the top row and depth maps in the bottom row. First and third column correspond to view from RGB-D sensor #1, whereas second and fourth to sensor #2

FIGURE 3.2: Views of the objects the elder is asked to interact with

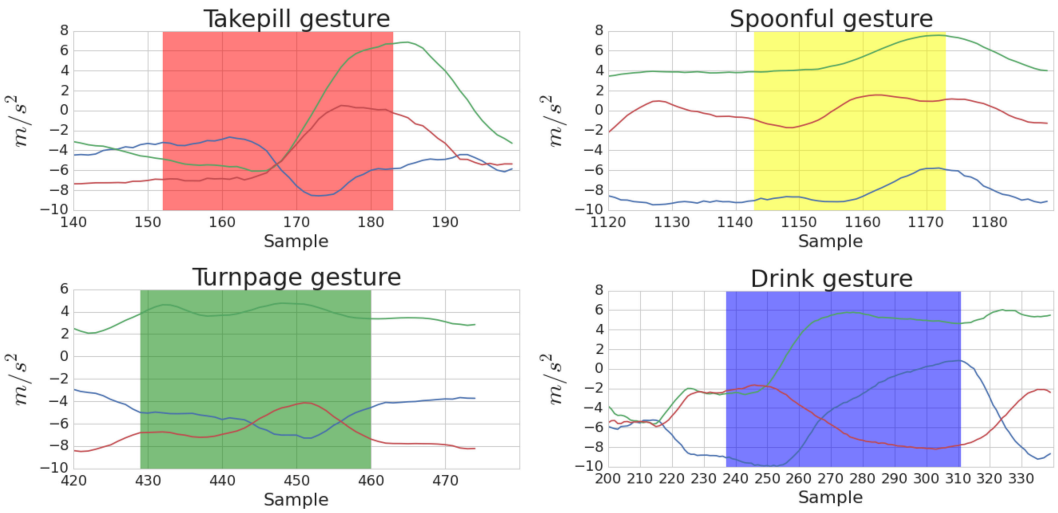


FIGURE 3.3: Examples of the four gestures' accelerometer readings in the three axis, x (red), y (green), and z (blue)

Objects' viewpoint variability and size. The objects can present heavy changes in terms of appearance in the color cue, due to either partial occlusions or the viewpoint. The viewpoint might in fact cause also huge variations in the objects' shape observed from depth maps. Note the variability of the blue book, opened versus closed in Figure 3.1. What is more, the relatively small size of some of the objects causes them to be completely shapeless when observed at 2 meters distance because of the inherent noise introduced by the Kinect depth sensor.

Uncontrolled behavior and introduction of external objects. Despite the scenarios were scripted to ensure a balance of activities' examples in the dataset, the participants were not always following the given instructions, thus introducing a certain degree of improvisation in the scenarios; such as, for example, the entrance of external objects – like the walking stick in the righmost frame of Figure 3.1.

Direct sunlight and specular surfaces. As seen in Figure 3.1, because of the light coming from the window, one of the views are darker (second and fourth frame) than the other (first and third). Moreover, sunlight reflections into shiny objects can also increase the noisiness of the depth sensor reading. In the view corresponding to second and fourth frames in Figure 3.1, the table surface reflects ambient infrared light coming straight from the window.

Inter- and intra-variability of activity/gesture examples. Some of the activities/gestures we are intended to recognize are quite similar one to another: the arm movement is very similar in "drinking" and "taking pill" from the perspective of the vision module. In addition, in the inertial cue, there is also a certain degree of similarity between gestures of different classes compared to the "no-gesture" – that is, when the participants are almost steady. On the other hand, the dataset presents a significant variability within each category regardless of the data cue utilized. This becomes particularly evident when observing the recorded instances of "reading a book"/"turn page". In addition, dealing with the inertial data becomes even harder when the sensor is worn by elderly people with shaky hands.

3.2.2 Hardware

We used two RGB-D cameras, each connected to a different laptop computer. Among the existing RGB-D cameras, we chose the popular Kinect device for its price and reliability. The device uses a structured IR light pattern which is projected to the scene by the emitter and read back by the IR sensor.

Among existing WIMU sensors, we chose Shimmer, which consists of accelerometer, gyroscope, and magnetometer. An IMU is a high-frequency sampling sensor; and particularly Shimmer is able to sample in a wide range of frequencies, up to 200Hz. In contrast to other IMU providing inertial data in millivolts (mV), the Shimmer device converts the mV to standard units, that are m/s^2 for accelerometer, rad/s for the gyroscope, and *Gauss* for the magnetometer. The device can also stream the signals to other devices via Bluetooth.

3.2.3 Acquisition setup

The Kinect devices were both elevated at 2 meters height using tripods and pointing to the table in which the activities took place. In both views, the closest table point was at 1 meter and the furthest at 1.8 meters. The devices recorded the scene from

complementary viewpoints with intersecting frustums, so as to obtain the most complete picture of the scenario. However, using such setup, IR patterns interfere with each other causing the devices to provide unreliable depth measures. The solution adopted was to use two Kinect for Windows edition, which offered the possibility to turn on and off the IR-emitter. Since the turn-on/turn-off have to be synchronous, we implemented a ping-pong recording setup alternating in time the devices' acquisition. While this solved the problem, it also reduced the sampling rate to 4 FPS (2 frames per second and per view); and the exact time was determined empirically from an experiment explained in more detail in Section 3.4.1.

The Shimmer sensor was attached with a velcro strap to the right wrist of each of the participants. We set the sampling rate to 25Hz with the aim to minimize delays on the communication due to data processing bottlenecks. Since the sensor communicates over Bluetooth, and this technology has a short range, we used an Android phone as a bridge for transmitting data from this Personal Area Network (PAN) to the Local Area Network (LAN) by emitting over WiFi to a laptop computer. The phone is also responsible of time-labeling the samples using a timestamp. This configuration increases the freedom of the user in the environment and provides a more realistic setting. Since the magnetometer measurements depend on the sensor's orientation w.r.t. the magnetic north, we discarded these features so as to make the gesture recognition magnetic-orientation invariant.

3.3 System

The proposed system consists of two main components, the vision component and the wearable one. After performing separately, a third component integrates their outputs in a late fusion fashion, as shown in Figure 3.4.

In the *vision module*, multi-modal Dense Trajectories (MmDT) are extracted from the multiple RGB-D streams acquired from the Kinect devices. We refer to the DT as "multi-modal" since a HON descriptor computed from the depth modality in addition to the ones from RGB. The additional depth-based descriptor adds extra geometric/shape information to the appearance or motion information throughout the trajectory.

Following a Bag-of-Visual-Words approach, we generate next a set of codebooks for the different kinds of MmDT features: shape features of the trajectories ("TS" from now on), HOG, HOF, MBH, and HON features; making a total of 5 codebooks. Codebooks are multi-view, i.e., they are trained using MmDT from different views.

For the detection, we slide a temporal window over the videos. A word representation is built for each window and descriptor. We then determine its category using a SVM classifier. The classifier is trained with examples of each of the activities (positives) altogether with negative examples.

The *wearable module* preprocesses the acceleration and angular velocity data as a first step. Since raw inertial data is inherently noisy, we normalize and filter out outliers. Then, we extract a set of features that are: raw data from accelerometer, sorted data from accelerometer, jerk, and complementary filter.

Secondly, the module clusters the data in order to find a set of representative gesture models. For each model, we learn a distribution of alignment costs (from a separate data sample), in such a way during the prediction phase we can simply threshold the alignment cost of any potentially observed gesture instance and hence determine if it is one of the gesture classes being performed. For the computation of thresholds, we use a random-selection Montecarlo method.

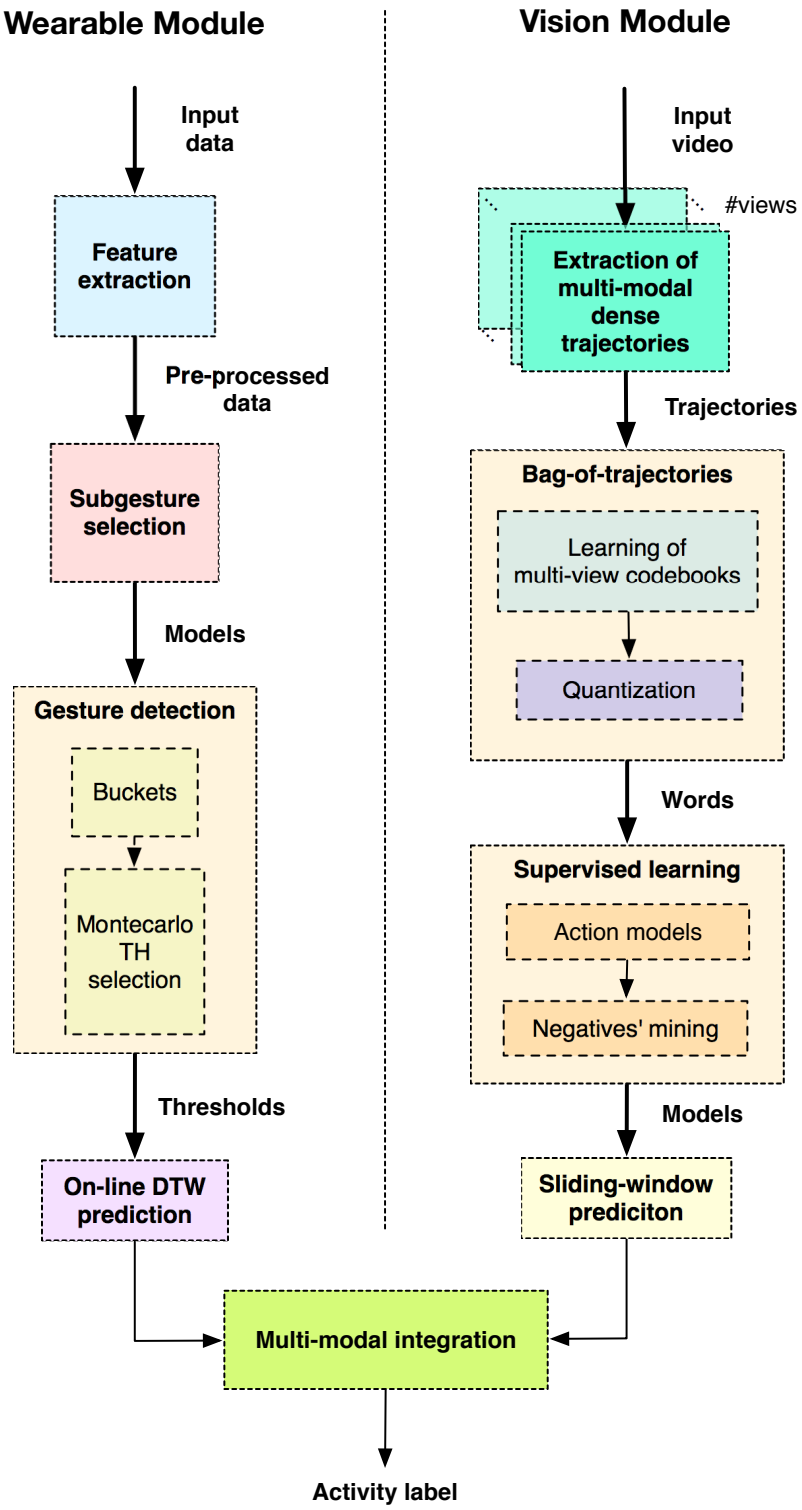


FIGURE 3.4: General pipeline of the system consisting of two modules: a vision module and a wearable module

Finally, the *fusion module* takes the binary outputs of the vision and wearable components and fuses them. This is done by applying the intersection. This is a fast method which improves the performance over the single modalities.

3.3.1 Vision module

The vision module implements a pipeline with three main stages: the feature extraction, the construction of a mid-level representation, and the sliding window-based action detection itself. Next, we explain in more detail each of the stages of the vision pipeline (illustrated in Figure 3.4).

Multi-modal Dense Trajectories (MmDT)

While Dense Trajectories descriptors (TS, HOG, HOF, and MBH) are computed solely from the color cue, we compute the surface normals on the RGB-registered depth maps and summarize the orientations of the normals within trajectory-aligned regions in a Histogram of Oriented Normals (HON) that we add to the usual descriptors from the color modality. Figure 3.6 illustrates the construction of MmDT. Next, we briefly explain how we compute the HON representation of a depth map.

Histogram of oriented (depth) normals (HON) Based the work of [170], we compute a histogram counting the orientations of normal vectors computed from depth map. In order to do so, we first transform the map to a point cloud $\mathcal{R}$ in which we have 3D points in "real-world" coordinates (values representing actual distances in $\mathbb{R}^3$). Then, finding the surface normal 3D vector at a given point $\mathbf{r} = (r_x, r_y, r_z) \in \mathcal{R}$ can be seen as a problem of determining the perpendicular vector to a 3D plane tangent to the surface at $\mathbf{r}$. Let denote this plane by the origin point $\mathbf{q}$ and its normal vector $\mathbf{u} = (u_x, u_y, u_z) \in \mathbb{R}^3$. From the neighboring points $\mathcal{K}$ of $\mathbf{r} \in \mathcal{R}$, we first set $\mathbf{q}$ to be the average of those points:

$$\mathbf{q} \equiv \bar{\mathbf{r}} = \frac{1}{|\mathcal{K}|} \sum_{\mathbf{r} \in \mathcal{K}} \mathbf{r}. \tag{3.1}$$

The solution of $\mathbf{u}$ can be then approximated as the smallest eigenvector of the covariance matrix $\mathbf{A} \in \mathbb{R}^{3\times3}$ of the points in $\mathcal{R}^{\mathcal{K}}_{\mathbf{r}}$. The sign of $\mathbf{u}$ can be either positive or negative, so we adopt the convention of consistently re-orienting all normal vectors towards the depth sensor $\mathbf{z}$ viewpoint. In Figure 3.5, we illustrate the normals extracted.

The normal vectors already computed are represented in cartesian coordinates using 3 parameters. However, when expressing them in spherical coordinates (radius r, inclination θ, and azimuth φ), one of the parameters (r) turns out to be constant in our case. This more compact representation is calculated as follows: $\theta := \arctan(u_z/u_y)$ and $\varphi := \arccos\sqrt{(u_y^2 + u_z^2)/u_x}$. Hence, the final HON representation consists of a two-dimensional $\delta_\theta \times \delta_\varphi$ histogram, with each bin counting occurrences of pairs of (θ,φ). This structure is vectorized an added as the fifth feature of our MmDT framework.

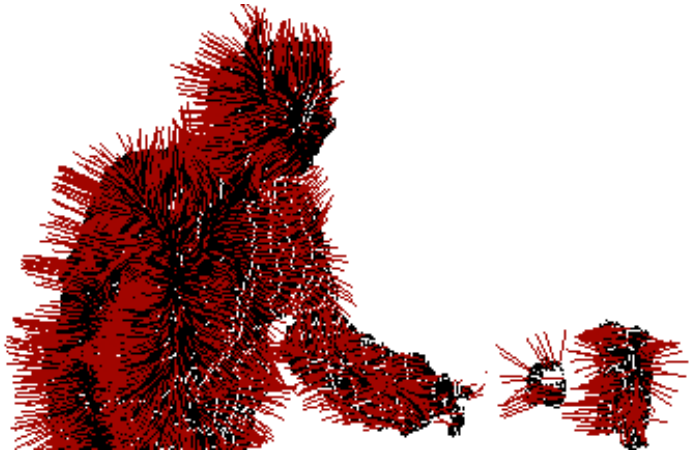


FIGURE 3.5: Surface normals computed from a depth map in which a person tries to reach some objects. Black dots are 3D points and red lines are vectors representing surface normals (arrow heads are not drawed for the sake of the visualization)

Mid-level Bag-of-Visual-Words (BoVW) representation

Since MmDT are locally extracted along videos, we need to compute a mid-level representation for each temporal segment we are intended to classify later during the detection phase. As in [185], we use a BoVW-like approach. For this purpose, we generate multi-view codebooks of k_{vis} visual words as K-Means (with the euclidean distance metric) from a sample of M examples each, one for each of the five trajectory description types: $\mathcal{D} = \{\text{TS}, \text{HOG}, \text{HOF}, \text{MBH}, \text{HON}\}$. From them, we generate the mid-level representations or words. The words simply count the frequency of each of the k_{vis} codes in a particular temporal video segment.

Action detection

In order to perform detections in a video sequence, we follow a sliding window and detection-by-classification approach. We choose BoVW to be particularly convenient in the sliding window scenario. Having the BoVW representation computed at frame-level, it is possible to compute the representation of a window centered at certain frame in an "integral" efficient way.

Let us denote $\mathbf{B} \in \mathbb{N}^{k_{\text{vis}} \times F}$ a matrix-like structure representing the set of BoVW descriptors for a sequence of F frames as columns. Then, the representation BoVW representation of a window $\mathbf{w}_t$ centered at frame t can be computed as follows:

$$\mathbf{w}_t := \sum_{i=t+\lfloor \frac{s}{2} \rfloor}^{t-\lfloor \frac{s}{2} \rfloor - 1} \mathbf{B}_{:,i}, \tag{3.2}$$

where $\mathbf{w}_t \in \mathbb{N}^{k_{\text{vis}}}$, s the width of the window, and $\mathbf{B}_{:,i}$ the i-th column of $\mathbf{B}$. Note however this equation can be efficiently computed using the column-wise accumulated version of $\mathbf{B}$, namely $\widetilde{\mathbf{B}}$. If we define the t-th column of $\widetilde{\mathbf{B}}_{:,t} = \sum_{i=0}^{t} \mathbf{B}_{:,i}$. Then, $\mathbf{w}_t = \widetilde{\mathbf{B}}_{:,t+\lfloor \frac{s}{2} \rfloor} - \widetilde{\mathbf{B}}_{:,t-\lfloor \frac{s}{2} \rfloor - 1}$. Finally, and prior to the classification, each resulting window descriptors is L1-normalized, $\hat{\mathbf{w}} = ||\mathbf{w}||_1$.

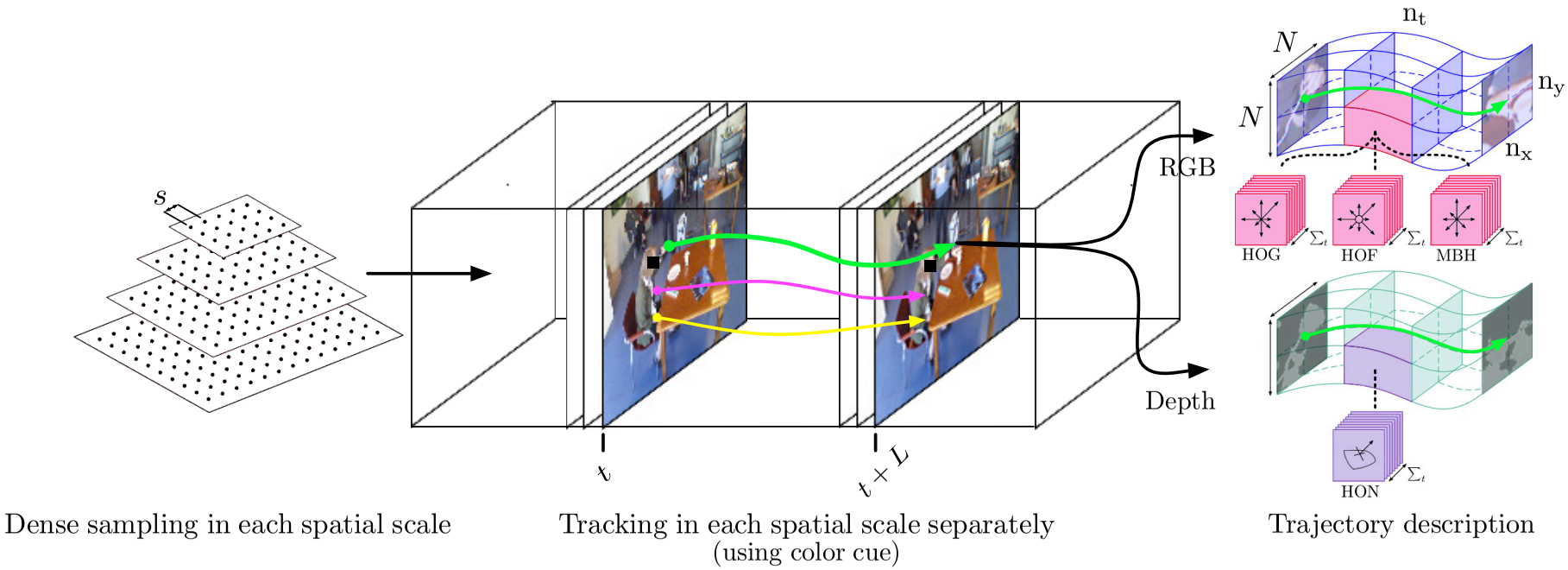


FIGURE 3.6: The multi-modal trajectories are extracted at different spatial scales. In each spatial scale, dense optical flow fields extracted from the color cue are used to track the pixels during L frames at most. A trajectory descriptor is represented then by its shape (TS) together with a set of descriptors (HOG, HOF, MBH, and HON) computed within the $n_x \times n_y \times n_t$ sub-volumes surrounding the trajectory path

In order to classify a window, we use a non-linear SVM with a RBF with chi-square kernel. As in [185], the different trajectory descriptors are combined at kernel level [176]:

$$K(\hat{\mathbf{w}}_i, \hat{\mathbf{w}}_j) := \exp\left(-\sum_{d\in\mathcal{D}} \varepsilon^{(d)} \frac{1}{\bar{\zeta}^{(d)}}\, \zeta(\hat{\mathbf{w}}_i^{(d)}, \hat{\mathbf{w}}_j^{(d)})\right), \tag{3.3}$$

where $\zeta(\hat{\mathbf{w}}_i, \hat{\mathbf{w}}_j)$ is the chi-square distance between the pair of window descriptors $(\hat{\mathbf{w}}_i, \hat{\mathbf{w}}_j)$, $\bar{\zeta}^{(d)}$ is the mean chi-square distance among training d descriptors, and $\varepsilon^{(d)}$ is an experimentally chosen weight assigned to d. The weights sum up to 1. Words are l1-normalized as suggested for the computation of non-linear kernel maps [178].

Finally, we generate the detection output. Since we slide temporal windows of different sizes, at a certain position t windows from different sizes can give different responses. Let us define $\mathbf{Y} \in \mathbb{N}^{|\Omega|\times F}$ as the response matrix for a particular video, where Ω is the set of action categories and F the duration of the video. Then, given a category $\omega \in \Omega$ and temporal instant t, $\mathbf{Y}$ is assigned as follows:

$$\mathbf{Y}_{\omega, t-\lfloor \frac{s}{2} \rfloor : t+\lfloor \frac{s}{2} \rfloor} := \mathbb{1}\{g(\hat{\mathbf{w}}_t) = \omega\}, \tag{3.4}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function and $g(\cdot)$ is the functional representation of the multi-class classifier. In other words, if the classifier's response for the window descriptor $\hat{\mathbf{w}}_t$ is class ω, the set of positions $\{(\omega, i) \mid t - \lfloor s/2 \rfloor \leq i \leq t + \lfloor s/2 \rfloor\}$ in $\mathbf{Y}$ are marked as positive detection, i.e. 1.

3.3.2 Wearable module

The wearable module takes accelerometer and gyroscope data recorded using the WIMU and learns the models with the goal of detecting the gestures. First of all, features are extracted from the raw data, namely, raw acceleration, "sorte" acceleration, complementary filter, and "jerk". The large variability in the execution of gestures motivates to look for different patterns under the same named class. In this sense, we apply a clustering strategy over segmenting gestures in order to obtain the most representative models for each class. The detection of gestures in a sequence requires the elastic comparison of each possible sub-sequence with all the model gesture patterns. For this task, we use Dynamic Time Warping (DTW). The final detection of a gesture is obtained when the accumulated aligned similarity between a sub-sequence and the tested pattern is below an acceptance threshold. The selection of the threshold is a critical step for obtaining accurate detection results. In the training step one has to find an acceptance threshold for each candidate gesture. However, if we use multiple models for each gesture the process becomes considerably more complex. For this reason a Montecarlo optimization technique is used for establishing the acceptance threshold for each sub-pattern in each gesture.

Next, we describe the details of this module, corresponding to the left part of Figure 3.4.

Feature extraction

Prior to the feature extraction, we smooth the sequences using a mean filter with kernel size of 10 samples. Errors in the measurements in the form of outliers that

largely deviate from the mean are removed by applying a thresholding operation on values above or below three standard deviations.

With the input signal properly preprocessed, we compute a set of features. These features are the following:

Raw accelerometer data Data recorded on the scenario regarding only to accelerometer. Used in some works as in [105].

Sorted accelerometer A set of discrete features which account for a relative rank among the three axis of the accelerometer is defined. For each sample we assign a value (-1, 0 or 1) according to the ranking of its value compared to the other axis, i.e. the axis with the lowest value is set to -1, the axis with the largest value is set to 1, and the remaining one to 0.

Complementary filter The complementary filter mixes gyroscope and accelerometer values in order to get a smoother signal with less noise and transforming acceleration into rotations. In essence, we transform the acceleration vector $\mathbf{a} = (a_x, a_y, a_z) \in \mathbb{R}^3$ of each sample into the rotation vector, then we apply a low-pass filter to the accelerometer in order to remove noise, and a high-pass filter to the gyroscope for removing the drift (an almost constant component). Then we merge both measures in order to get the orientation of the sensor. The rotation vector from the accelerometer is computed as follows: $\alpha := cos^{-1}(a_x/||\mathbf{a}||_2)$, $\beta := cos^{-1}(a_y/||\mathbf{a}||_2)$, and $\gamma := cos^{-1}(a_z/||\mathbf{a}||_2)$. Then, for each sample i we are able to apply the complementary filter defined on the next equation:

$$C_x^{(i)} := \sigma\psi_x^{(i)} + (1-\sigma)\alpha^{(i)}, \tag{3.5}$$

$$C_y^{(i)} := \sigma\psi_y^{(i)} + (1-\sigma)\beta^{(i)}, \tag{3.6}$$

$$C_z^{(i)} := \sigma\psi_z^{(i)} + (1-\sigma)\gamma^{(i)}, \tag{3.7}$$

where (ψ_x, ψ_y, ψ_z) represent the gyroscope values, while the value of σ controls the response of the filter. As demonstrated in [81], the complementary filter reduces drift and noise presented by accelerometer and gyroscope while maintaining the computational complexity (compared to Kalman Filter [101]).

Jerk We use Jerk, which is the derivative of the acceleration $\mathbf{j}(t) = \frac{\mathrm{d}\mathbf{a}(t)}{\mathrm{d}t}$. It shows the transitions of the acceleration and is numerically computed using centered differences [17].

Sequence similarity using Dynamic Time Warping

The DTW algorithm aligns two time-series of different length and returns an alignment cost [180]. This particular property makes possible to compare sequences with different duration without losing information. This procedure is used in two different steps in our system. First, it will be used as a similarity metric in the definition of the sub-patterns in each gesture category. And second, it will be used when checking the sequence against each sub-pattern, both in the training step for the threshold selection and then in the test step for the detection.

Sub-gesture selection

As previously commented, we observe a large variability in the way a gesture is performed. This motivates the search of a set of models under each gesture category. In order to perform this task we use K-Medoids algorithm using DTW as a distance function. The training sequences are segmented in order to obtain the individual gestures. Then, we compute the DTW alignment costs for all of them. Because DTW is not a proper metric (it is not symmetric) we define a pseudo-metric by adding to the DTW matrix its transpose. The result of the K-Medoids is a set of k_{wear} training examples. These examples will be considered as different model prototypes, M_i, for the gesture.

Gesture detection

The dynamic programming matrix from DTW enables the reconstruction of the matching path. This provides the temporal extent of the matched pattern within a sequence of observations. Because we want to detect full patterns inside the sequence, we need to set the first column of the dynamic programming matrix to infinity. In this way we ensure the best alignment to always start at the first row. Also, we want the algorithm to detect the pattern as a sub-sequence inside our full sequence. For doing so, we set the first row to 0, then we allow a gesture to begin at any position.

A cell $\{i, j\}$ of the DTW matrix is computed by taking the minimum of the three upper-left neighbours $min(\{i-1, j\}, \{i, j-1\}, \{i-1, j-1\})$ and adding the euclidean distance between the two corresponding frames. Since the last row of DTW matrix represents the alignment cost of a certain pattern against a sub-sequence, we should expect that a gesture will have lowest value than other parts of the sequence. Then, we use a threshold per model for detecting the gestures. Next, we explain Montecarlo optimization for the selection of those thresholds.

Montecarlo threshold optimization As commented, the main difficulty for learning the acceptance thresholds is the multiplicity of models for each gesture. The problem lies in the fact that the groundtruth data only defines the gesture category. However, because we have several models per gesture we do not know which model best represents that gesture in a given sequence. The naive strategy for solving this problem would be to select a single acceptance threshold for all models in a gesture. However this severely hinders the expression power of each of the models. We opt for learning a different threshold per gesture model. Thus the learning problem needs to find the best acceptance threshold for the best model among the models for each gesture. In order to solve this problem we use Montecarlo optimization.

We assume that given a sequence, there will be a unique model for each gesture, since that sequence is performed by a single user. Then, for each gesture, we compute all the DTW matrices associated to all the models of that gesture. We will have as many matrices as number of models for each sequence.

Montecarlo optimization is based on randomly generating solutions of the problem at hand and choosing the solution that optimizes the objective function. This kind of optimization technique is specially suitable when the objective function is easily computed and the solutions are complex, for example, structured solutions. The convergence speed of this method is very slow, of the order of the square root of the number of samples generated.

In our problem the objective function is the F1 score. Given a set of S sequences and N_{wear} models per gesture, a solution is composed by S pairs model-threshold

$\{\mathbf{M}_i, \tau_i\}$ out of the N_{wear} models, one for each of the sequences. If we order all the sequences, this is easily illustrated as a graph in which, for each sequence we have a node for every model. Then, a solution is the combination of the path that goes through all sequences combined with the corresponding thresholds selection for each model. An example of the graph is shown in Figure 3.7. Observe that each path defines the correspondence of a single model for each sequence. Thus, a path may involve the same model applied on different sequences. For example, in Figure 3.7, the blue path considers $\mathbf{M}_1$ in sequence 1 and 2. Given a path, the threshold selected for each model is the one that maximizes the average F1 score over all sequences that consider that model. Each path is then scored according to the average F1 score achieved applying the selected thresholds. The final solution corresponds to the path that achieves the highest average F1 score.

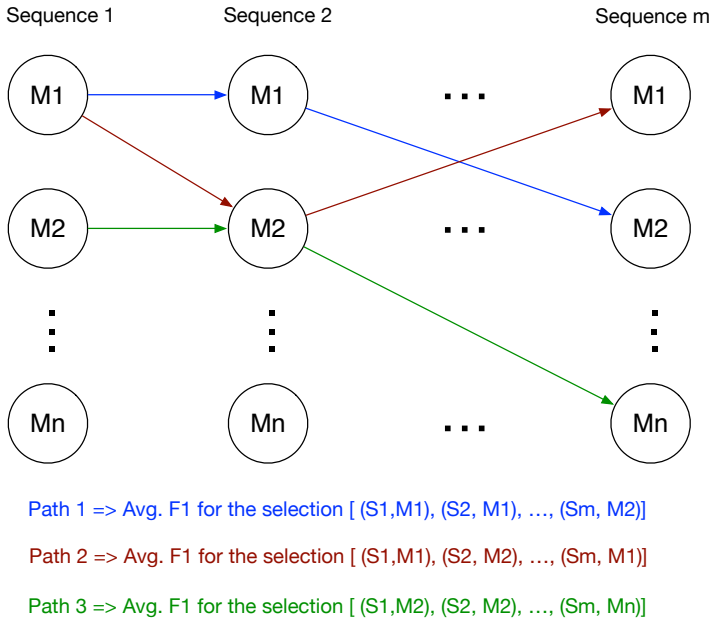


FIGURE 3.7: The montecarlo threshold-selection method

As a practical note, in order to define the range of thresholds we evaluate for each model, we compute the DTW matrices corresponding to each model over each training sequence. Using the annotations of the groundtruth, we can get the alignment costs of each of the gestures. We take not only those values but also, the ones corresponding to the frames close to the end point defined by the groundtruth. All these values will be used as positive samples. By doing so, we allow a certain distortion on the gestures. The parameter that controls the number of points we take is called tolerance and is a percentage over the length of the gesture. We also consider all values corresponding to negative points. The range of thresholds to be evaluated goes from the minimum value of the positive samples up to the first quartile value of the negative ones.

3.3.3 Integration module: learning-based fusion

Given a particular time instant, the vision and wearable modules provide a detection decision for each of the activity/gesture classes. We designed a learning-based

fusion strategy consisting on stacking a centered window of size around each predictions of size ωN on a $2\omega N$-valued feature vector representation that can be input to a discriminative classifier:

$$\mathbf{x}_{\mathrm{v+w}} = [\mathbf{y}_1^{\mathrm{v}}, \mathbf{y}_2^{\mathrm{v}}, \ldots, \mathbf{y}_{\omega N}^{\mathrm{v}}, \mathbf{y}_1^{\mathrm{w}}, \mathbf{y}_2^{\mathrm{w}}, \ldots, \mathbf{y}_{\omega N}^{\mathrm{w}}] \,,$$

where $[\cdot]$ is the concatenation operation and $\mathbf{y}^{\mathrm{v}} \in \{0, 1\}$ and $\mathbf{y}^{\mathrm{w}} \in [0, 1]$ are prediction values, respectively, from the vision and wearable module. Note the prediction values from the wearable module are binary, whereas the ones from the vision one are real-valued confidences. The latter are calculated as the ratio of positive binary predictions for different sliding window sizes divided by the total number of window sizes.

In order to perform the classification, we train a neural network per activity class, consisting of a $2\omega N$-neuron input layer, two fully connected layers and 2-neuron output. The net is trained using *adam* optimizer and *cross entropy loss* function. For the output layer, we use a soft-max function.

During the training of each epoch, we feed the net with 80% positive examples and 20% of negatives. Hence, the loss function is weighted in order to compensate the bias introduced by this difference.

3.4 Results

First, we illustrate the experiments carried out to establish some settings in the different components of the system and detail which are the system's parameters. We also explain different strategies to fuse the outputs of the visual and inertial modules. Finally, we illustrate the results of the two main modules separately and eventually the visual-inertial fusion results.

3.4.1 System settings and parameters

In this section, we first present a preliminary experiment for the IR-emitter's delay calculation and then we introduce the parametrization of each system's module.

IR-emitter turn-on time delay

The blocking turn-on instruction of Kinect for Windows does not ensure the IR light bulb having reached full power before the acquisition of a new depth frame – thing that causes erroneous depth map readings. In order to determine the "real" reactivation time, we performed the following experiment: we fixed a sensor in an indoor still scene and captured 2,500 depth maps, selecting one out of those as a reference frame. Next, we subtracted the reference frame to each of the rest and calculated for each of those frame differences the accumulation of absolute differences, i.e. the magnitude of the difference of that frame respect to the reference. Figure 3.8 depicts these magnitudes in function of the turn-on time the blocking instruction took, as well as the effect of forcing a turn-on delay or none (0 ms) instead of relying on the instruction's own optimistic delay. Given the results, and seeking to ensure a balance in the frames' quality/quantity trade-off, we decided to force a minimum time delay of 275 ms.

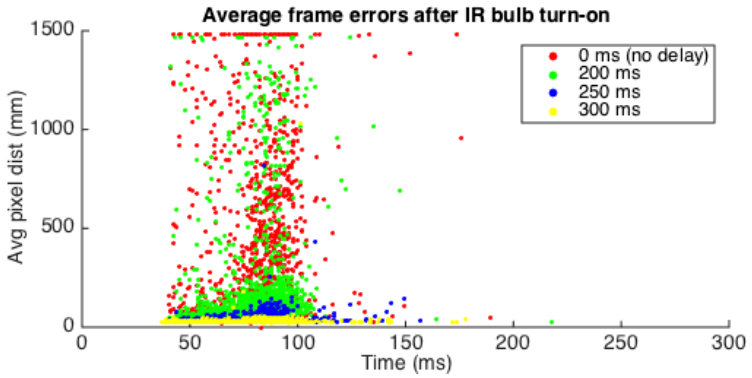


FIGURE 3.8: Error introduced by the IR bulb forcing different time delays in milliseconds (ms) or none (0 ms)

Vision module parameters

In the vision part, we decided to stick to the default parameters for the computation of DT[185] when possible. The number of words in the multi-view codebooks are $k_{\text{vis}} = 4,000$, the sample of trajectories used to compute each codebook is of size 200,000 (100,000 per view), and SVM regularization parameter set to 1. These parameters have been largely validated in many action recognition datasets and thus guaranteed to provide state-of-the-art results. Notwithstanding, we set the trajectory length to a lower value of $L = 4$, more suited to our framerate than the original value of 15. This was done after experimentally testing different values for L in the set {3,4,5}.

Some parameters were fixed also for the computation of HON descriptors. The radius distance when computing the normal vectors was set to 2 cm, this being a standard value used in object recognition scenarios [1]. For the construction of the HON histogram, δ_θ and δ_φ were both set to 5.

For the negative mining of examples in the classification, we randomly sampled temporal segments having less than 0.2 of temporal overlap with activities' annotations. In particular, we sampled 10 negatives for each positive example. Moreover, we also included 1 million negative trajectories during the generation of codebooks, apart from the 200,000 from the positive examples.

Finally, in order to determine the optimal set of weights **$** to assign to the different modalities when computing the SVM kernel during the action detection, we performed an exhaustive search in the training set. For this purpose, we generated all the possible 5-sized vectors of weights that sum up to 1 with incremental steps of 0.1 and evaluated average classification accuracy on the set of pre-segmented action gestures. Moreover, the weights were optimized independently for each action.

Wearable module parameters

Regarding the wearable module, there is also some parameters that needed to be validated. One of them is the parameter that controls the number of clusters (or prototypes) computed by K-Medoids. This depends on the complexity of each gesture class. We tested 1, 2, and 3 prototypes. No more classes are considered due to the reduced number of instances per gesture class.

When we are reconstructing our gesture predictions using DTW, it is common to have more than one reconstruction. This means that there are several matching

paths that reached a cost below the threshold. This phenomenon is caused by having low values along the DTW matrix spread over their neighbour cells. A parameter regulates the activation of these reconstructions by thresholding the reconstruction cost. The tested values were 5, 10, 15, 20, 25, and 30 activations. Moreover, and in order to avoid short activations, we set a minimum length for considering a gesture. The values tested are 0, 10, 15, 20, 25, and 30 frames.

A tolerance parameter was introduced in Section 3.3.2 that controlled how many frames around a gesture end are considered so their values are put inside the positive bucket. Having a large value here will make our threshold be too large and then we will let pass a bigger number of false positives. This value has been experimentally set to 0.2.

Regarding the Montecarlo method, a number of samples has to be defined. We have set this value to 10K. As it can be seen in the following section, increasing the number of paths does not involve a large computational effort. The expected error rate is $E = \frac{1}{\sqrt{n}}$. By setting n to 10K, we expect an error rate of 0.01 which we consider is enough for the system.

3.4.2 Efficiency and computational cost

The vision module is based on the DT framework, which originally runs at 10-12 fps in VGA video. However, we extended the set of descriptors with HON, thus involving the computation of surface normals. The computation of normals is an expensive process when done naively, but can be greatly optimized by parallelizing computations or using approximated methods. If optimized, this module could run much faster than the 2-fps acquisition rate of the two Kinects.

In the case of the wearable module, there are two expensive processes: DTW matrix computation and Montecarlo threshold optimization. DTW matrix computation is $O(\text{modellength} \cdot \text{sequencelength})$, but it is not easily parallelizable. The Montecarlo threshold optimization can also expensive if no optimization is applied. In our case, we have designed the algorithm in order to first precompute all the needed values, that is, when computing a path, the algorithm only has to select values, but do not compute them.

3.4.3 Experimental results on SARQuavitae dataset

We validated the proposed system in the SARQuavitae dataset. We first explain the validation procedure and then we illustrate the results got by both the individual modules and the fused results from the fusion module.

In the experiments, we used a leave-one-subject-out cross-validation (LOSOCV) procedure in order to ensure a proper generalization of the methods. Besides, in order to validate any of the parameters described in 3.4.1, we used an internal cross-validation within the training partition of the LOSOCV.

We used two different metrics in order to quantify the performances: F1-score and intersection-over-union (IoU) overlap. During the computation of the F1-score, we use a 20% of minimum overlap in order to consider a true positive detection. These results were calculated at sequence level and averaged within the corresponding LOSOCV's fold. Then, the final performance was calculated averaging again the performances of all the folds.

We separately computed the performance for all the classes, so we can better illustrate performance issues and difficulties of the system. Regarding the F1-score,

	Modalities					
Action name	TS	HOG	HOF	MBH	HON	Accuracy
Drinking	0.5	0	0	0.3	0.2	91.54%
Eating	0.1	0.1	0	0.4	0.4	86.42%
Reading	0.2	0.1	0.1	0.1	0.5	91.57%
Takingpill	0	0	0.4	0.1	0.5	83.80%

TABLE 3.2: Best performing weight combinations for each of the classes (in terms of accuracy)

if neither the groundtruth nor the prediction presented any activation, we counted the F1-score to be 1.

In the case of IoU overlap, an additional parameter is also taken into account, which is the number of *do-not-care* frames. This value regulates the amount of discrepancy in the borders of the detections when evaluating respect to the groundtruth. Since, a system of this nature does not need a perfect matching but only a rough detection, we could afford using relatively large do-not-care values. The maximum do-not-care value used in the later validation is approximately half of the shorter action's duration, i.e., 50 frames.

Vision experiments

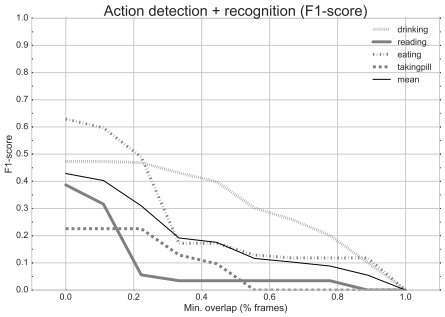
As a preliminary analysis was to determine somehow the contribution of the different modalities in the action detection task. Since we need to find the best set of ω weights for each subject, we can average the performance of all the weight combinations across the LOSOCV's training partitions. The set of weights selected per action is illustrated in Table 3.2. These weights provided us some intuition about the contribution of HON. It demonstrated to be very relevant for the task of action recognition and complementary to the other modalities. TS and MBH also demonstrated to be quite important, in contrast to HOF and HOG, with the latter being the less relevant in our dataset.

Once we had selected the weights, they were used in the action detection task. In Figure 3.9a and Figure 3.9b, we illustrate the performance of the detection, respectively, in terms of F1-score and IoU overlap. These results show the performance of the vision component in detecting quite differs from one action to another. It is able to more successfully detect "eating" and "drinking" actions, while not doing so well at "eating" or "takingpill". Our hypothesis is that the vision part is better at detecting large actions than smaller ones. This causes overlap values to be larger, whereas in terms of F1 the detector is highly penalized.

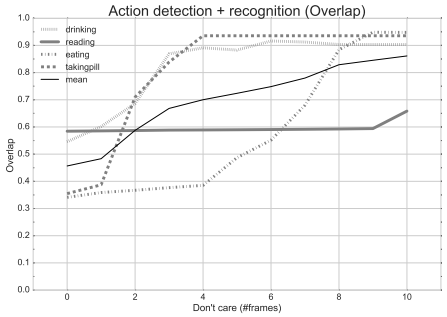
Wearable experiments

Regarding the use of K-medoids and Montecarlo methods in order to have more than one model per class, we performed the same experiments with both modalities. In Figure 3.9c and Figure 3.9d, we can see the improvement, respectively, over to Figure 3.9e and 3.9f. This is due to the better representation obtained by establishing different models per class. Given the nature of this dataset (recorded in the wild, without constraints), the intra-class variability is expected to be large. From the results presented in the supplemental material, it is observed that this happens even with the better representing features we have computed. From these results, we demonstrate the convenience of computing sub-classes in order to better model the gestures.

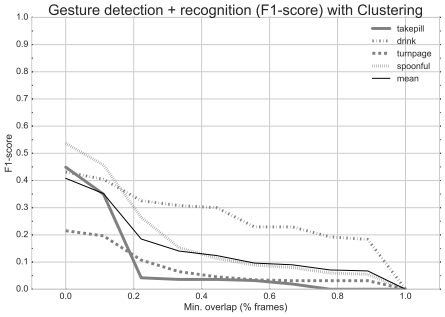
From the final results, one can see the classifier is outperformed in most off the cases by the wearable module, something we expected given the difficulties presented by the module in terms of intra-class variability. The most difficult class in terms of F1-score (concerned to the number of detections) is taking-pill. As it has been shown in the supplemental material, it is the one that is more confused against the others. Nonetheless, eating and drinking, that are the ones obtaining greater F1-scores, were the ones with less confusion as observed in the distance matrices.



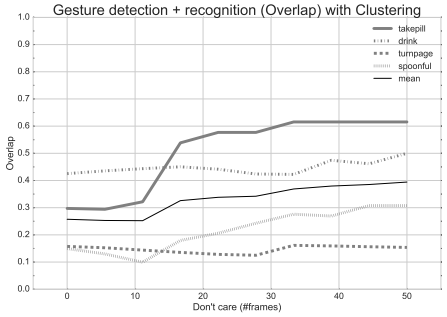
(A) F1-score for each class (and mean) and different minimum overlap values



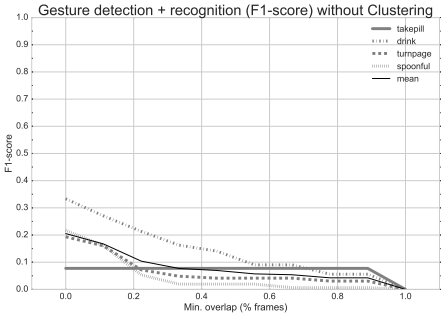
(B) Overlap for each class (and mean) and different *do-not-care* values



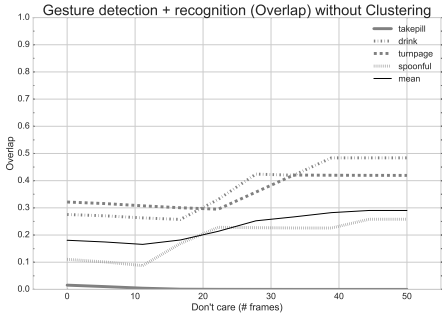
(C) F1-score for each class (and mean) and different minimum overlap values



(D) Overlap for each class (and mean) and different *do-not-care* values

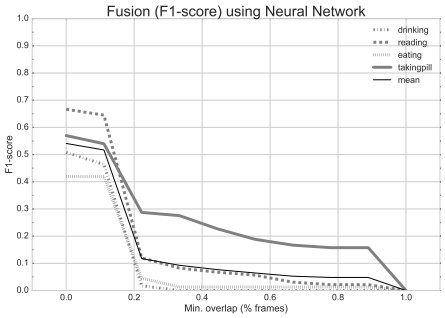


(E) F1-score for each class (and mean) and different minimum overlap values having one model per class

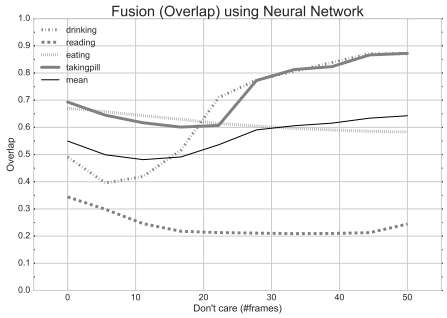


(F) Overlap for each class (and mean) and different *do-not-care* values having one model per class

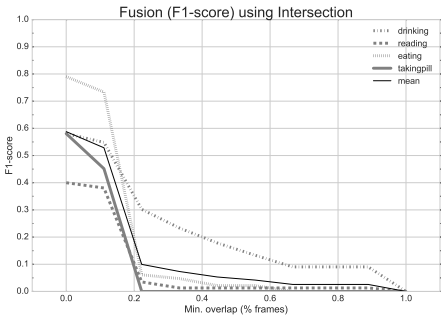
FIGURE 3.9: Detection performances of single modules



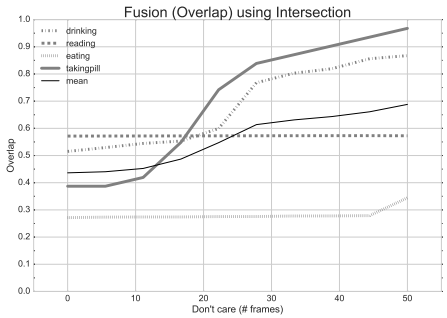
(A) F1-score for each class (and mean) and different minimum overlap using a Neural Network as the integration strategy



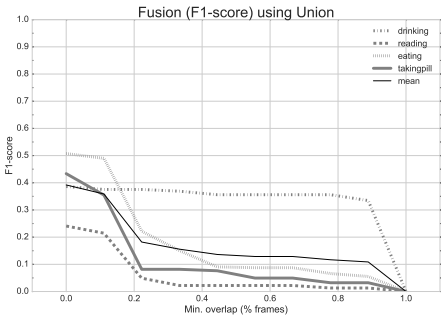
(B) Overlap for each class (and mean) and different *do-not-care* values using a Neural Network as the integration strategy



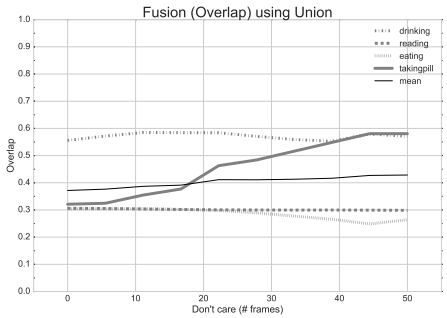
(C) F1-score for each class (and mean) and different minimum overlap using Intersection as the integration strategy



(D) Overlap for each class (and mean) and different *do-not-care* values using Intersection as the integration strategy



(E) F1-score for each class (and mean) and different minimum overlap values using Union as the integration strategy



(F) Overlap for each class (and mean) and different *do-not-care* values using Union as the integration strategy

FIGURE 3.10: Detection performances of different fusion strategies combining the two single modules

Integration experiments

The two modules, vision and inertial, are able to provide binary detection outputs for each of the classes in a particular time instant. Given that, intersection and union are possible alternatives to our learning-based fusion strategy to come up with the final integrated detection. We hence report these as baseline results to compare the learning-based integration.

Recall our goal is to detect actions, not precise temporal localization. Nonetheless, we analyze first overlap results. In Table 3.3 and Table 3.4, we show the results of the three different integration strategies in terms of both F1-score and IoU overlap respectively. We found the vision module performing individually was the most successful in 3 out of 4 classes. Nonetheless, the neural network is able to improve by 2% respect to the vision module.

More importantly, learning-based approach improves F1-score results respect single modalities or baseline integrations for all the classes except for "Drinking" (-4%) and obtained a particularly large improvement for "Taking-pill" (+12%) and "Reading" (+10%). In average, the learning-based fusion improves the vision module by 6%.

For the sake of completeness, we also illustrate the effect of varying the minimum overlap for TP in detection and the don't care size (varying the number of frames) have on learning-based (Figure 3.10a-3.10b), intersection (Figure 3.10c-3.10d), and union (Figure 3.10e-3.10f) integration strategies.

	Single modalities		Integration		
	Vision	Wearable	Intersection	Union	Learning-based
Taking-pill	**0.93**	0.61	0.87	0.51	0.54
Drinking	**0.89**	0.42	0.80	0.56	0.67
Eating	**0.38**	0.27	0.27	0.27	0.33
Reading	0.58	0.16	0.57	0.30	**0.60**
TOTAL (mean)	**0.69**	0.36	0.63	0.41	0.53

TABLE 3.3: Results in terms of overlap, for each of the classes and for all the integration strategies

	Single modalities		Integration		
	Vision	Wearable	Intersection	Union	Learning-based
Taking-pill	0.22	0.04	0.00	0.08	**0.34**
Drinking	**0.46**	0.32	0.30	0.37	0.42
Eating	0.48	0.26	0.06	0.22	**0.49**
Reading	0.05	0.10	0.04	0.05	**0.20**
TOTAL (mean)	0.30	0.18	0.10	0.18	**0.36**

TABLE 3.4: Results in terms of F1-score, for each of the classes and for all the integration strategies

The learning-based integration proved to be the most successful strategy for activity detection, achieving better F1-score than single modalities or baseline integrations (union/intersection). In fact, the results in terms of F1-score and IoU overlap indicate that our system is more effective for action detection than temporally predicting their temporal extent. For the latter task, the vision module performs better. The vision module uses action groundtruth annotations, whereas the wearable one uses gesture annotations. Therefore, when both modules are fused, the temporal extent of actions predicted by the integration no longer coincide with the action groundtruth. Nonetheless, determining if the elder took the medication is far more

important in a system of this kind than knowing the exact time frames in which the action occurred.

Chapter 4

Enhanced action classification via Ensemble Learning

Action recognition is considered a multi-class classification task where each action type is a separate target class. A classification system involves two main stages: selecting and/or extracting informative features and applying a classification algorithm. In such a system, a desirable feature set can reduce the burden of the classification algorithm, and a powerful classification algorithm can work well even with a low discriminative feature set. In here, we aim to enhance the effectivity of the classification module when recognizing human actions. In particular, we argue that the discriminative power of encoded information cannot be fully utilized by a single-classifier recognition technique. And the weakness becomes more evident when the complexity of the recognition problem increases because of many action types and/or similarity of actions, i.e. having small inter-class and large intra-class variance.

We evaluate the performance of an ensemble of action leaners, each performing the recognition task from a different perspective. The underlying idea is that instead of aiming a very sophisticated and powerful representation/learning technique, we can learn action categories using a set of relatively simple and diverse classifiers, each trained on a different feature set. Combining the outputs of several learners can reduce the risk of an unfortunate selection of a learner on an unseen action recognition scenario [86]. Thus having a more robust and general-applicable framework. In order to improve the recognition performance, the Dempster-Shafer Fusion combination strategy is utilized based on the Dempster-Shafer Theory, which can effectively make use of a diversity of base learners trained on different sources of information. The experimental results show that the strategic combination of these learners can significantly improve the recognition accuracy.

The chapter is organized as follows: Section 4.1 briefly surveys the related work on multiple classifiers systems and then introduces Dempster-Shafer Fusion; Section 4.2 presents the framework of our multi-classifier fusion for action recognition; and, finally, Section 4.3 evaluates the proposed method and discusses results.

4.1 Related work

A multiple classifier system [86, 134] is made up of an ensemble of individual classifiers whose outputs are combined in some way to ideally obtain a more accurate classification decision. In an ensemble of classifiers, it is expected each base classifier will focus on different aspects of the data [134]. However, the success of the ensemble approach depends on the assumption that single classifiers' errors are uncorrelated, which is known as classifier diversity in the background literature [201].

If each classifier makes different errors, then the total errors can be reduced by an appropriate combination of these.

Once a set of classifiers is generated, the next step is to construct a combination function to appropriately merge their outputs, also refered to as decision optimization. The most straightforward strategy is simple majority voting, in which each classifier votes on the class it predicts, and the class receiving the largest number of votes is the ensemble decision. Other strategies for combination function include weighted majority voting, sum, product, maximum, minimum, fuzzy integral, decision templates, and the Dempster-Shafer (DS) based combiner [82, 86]. Inspired by the DS Theory of Evidence [33], a combination method is proposed in [144], which is commonly known as the Dempster-Shafer Fusion (DSF) method. By interpreting the output of a classifier as a measure of evidence provided by the source that generated the training data, the DS method fuses an ensemble of classifiers.

In this work, after extracting a set of visual feature sets, we train different action learning models whose outputs are fused based on the DS fusion algorithm. As a result, we show that we can merge predictions made from different learners, trained in different feature spaces, with different dimensionality in both feature space and action sample length. Following the multiple classifiers philosophy, we show that the proposed ensemble approach outperforms standard non-ensemble strategies for action recognition.

Here we introduce the Dempster-Shafer Fusion used to combine the classifiers in Section 4.2.

4.1.1 Dempster-Shafer Fusion

Let $H = \{\mathcal{H}_1, \ldots, \mathcal{H}_{N_H}\}$ denote an ensemble of already trained classifier models on a classification problem on $\Omega = \{\omega_1, \ldots, \omega_{N_\Omega}\}$, i.e. the set of possible category labels. Given then an unseen instance's feature descriptor $\mathbf{x}$, each classifier outputs a class score distribution $\boldsymbol{\pi}_i = \mathcal{H}_i(\mathbf{x})$, $\boldsymbol{\pi}_i \in \mathbb{R}^{N_\Omega}$. Without loss of generality, assume the elements of $\boldsymbol{\pi}_i$ range in [0,1] and their sum is equal to 1. We might see $\pi_{i,j}$ as a degree of support for class ω_j by classifier $\mathcal{H}_i$. The larger the degree of support, the more likely the class label ω_j.

Following [86], we derive the fusion in terms of *decision profile* and *decision templates*. More concretely, the *decision profile* of $\mathbf{x}$ is defined as

$$\boldsymbol{\Pi}(\mathbf{x}) = \begin{pmatrix} \pi_{1,1} & \cdots & \pi_{1,j} & \cdots & \pi_{1,N_\Omega} \\ \vdots & & \vdots & & \vdots \\ \pi_{i,1} & \cdots & \pi_{i,j} & \cdots & \pi_{i,N_\Omega} \\ \vdots & & \vdots & & \vdots \\ \pi_{N_H,1} & \cdots & \pi_{N_H,j} & \cdots & \pi_{N_H,N_\Omega} \end{pmatrix}$$

.

Having set $\boldsymbol{\Pi}(\mathbf{x})$, we can apply a wide range of techniques to find the overall support for each ω_j so we finally choose the one with the largest support. In particular, Dempster-Shafer Fusion falls into the *class-indifferent* category of fusion methods [86], in which the *overall degree of support* to ω_j – hereinafter denoted as $\mu_j(\mathbf{x})$ – is derived using the $N_H \times N_\Omega$ values in $\boldsymbol{\Pi}(\mathbf{x})$[1].

[1] In contrast to these, *class-conscious* only use the N_H degrees of support corresponding to class ω_j in order to compute $\mu_j(\mathbf{x})$ from $\boldsymbol{\Pi}(\mathbf{x})$. One example of *class-conscious* are simple statistics, such as summation, averaging, or weighted averaging a particular class scores from the different classifiers in the ensemble.

Next step is to compute the *decision templates* for the different classes. In particular, the template $\mathbf{Y}(\omega_j; \mathbf{X})$ is:

$$\mathbf{Y}(\omega_j; \mathbf{X}) = \frac{1}{\eta(\omega_j)} \sum_{\{\mathbf{x}_k \mid y_k = \omega_j,\, \mathbf{x}_k \in \mathbf{X}\}} \mathbf{\Pi}_{:,j}(\mathbf{x}_k) \tag{4.1}$$

where $\mathbf{X}$ is the set of training instances, $\eta(\omega_j)$ is the number of instances from $\mathbf{X}$ with groundtruth class ω_j, and $\mathbf{\Pi}_{:,j}(\mathbf{x}_k)$ is the j-th column of $\mathbf{\Pi}(\mathbf{x}_k)$.

Having posed the problem in terms of decision profile and templates, we apply the DS fusion algorithm:

1. Compute the proximity between the decision profile of an instance $\mathbf{x}$ and the different class templates. Let $\mathbf{Y}_{i,:}(\omega_j; \mathbf{X})$ denote the i-th row of the decision template for class ω_j and $\mathbf{\Pi}_{i,:}$ the i-th row of the decision profile of $\mathbf{x}$, then we define the proximity measure as in [144]:

$$\Phi_{j,i}(\mathbf{x}) = \frac{(1 + ||\mathbf{Y}_{i,:}(\omega_j; \mathbf{X}) - \mathbf{\Pi}_{i,:}(\mathbf{x})||^2)^{-1}}{\sum_{j'=1}^{N_\Omega} (1 + ||\mathbf{Y}_{i,:}(\omega_{j'}; \mathbf{X}) - \mathbf{\Pi}_{i,:}(\mathbf{x})||^2)^{-1}}, \tag{4.2}$$

   where $||\cdot||^2$ is the euclidean norm.

2. Find the *degree of belief*, or the amount of evidence on the i-th classifier prediction of $\mathbf{x}$ is class ω_j:

$$b_j(\mathbf{\Pi}_{i,:}(\mathbf{x})) = \frac{\Phi_{j,i}(\mathbf{x}) \prod_{k \neq j} (1 - \Phi_{k,i}(\mathbf{x}))}{1 - \Phi_{j,i}(\mathbf{x}) \left[1 - \prod_{k \neq j} (1 - \Phi_{k,i}(\mathbf{x}))\right]}. \tag{4.3}$$

3. The overall degree of support corresponding to label ω_j is obtained by using Dempster's rule, which states that evidences (degree of belief) from each source (classifier) should be multiplied to obtain the overall support:

$$\mu_j(\mathbf{x}) = K^{-1} \prod_{i=1}^{N_H} b_j(D_i(\mathbf{x})) \tag{4.4}$$

   where $K = \sum_{j'=1}^{N_H} \prod_{i=1}^{N_\Omega} b_{j'}(D_i(\mathbf{x}))$ is a normalizing constant.

4. The predicted label $\hat{y}$ for $\mathbf{x}$ is the ω_j with the largest overall degree of support $\mu_j(\mathbf{x})$. Hence:

$$j^* = \underset{j}{\operatorname{argmax}}\ \mu_j(\mathbf{x}) \tag{4.5}$$

$$\hat{y} = \omega_{j^*} \tag{4.6}$$

4.2 Method

We define three classification approaches to action recognition. A first (holistic) model, considers all the features in one and only classifier. This serves as a benchmarking baseline to the other two ensemble-based strategies we propose. For all the approaches, we apply the five action descriptors: space-time interest points (STIPs)

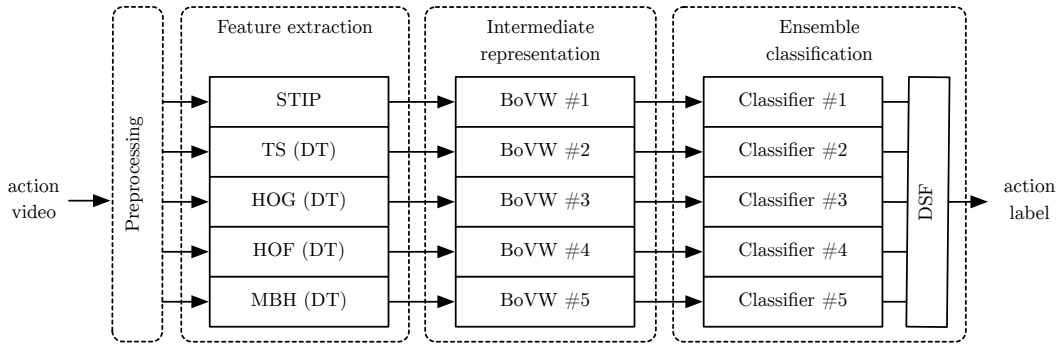


FIGURE 4.1: Ensemble of classifiers (Ensemble II in Section 4.2) combining their outputs using Dempster-Shafer Fusion (DSF).

and the five descriptors from Dense Trajectories (TS, HOG, HOF, and MBH). The extracted local features are quantized separately using Bag-of-Visual-Words in a 4,000-bin histogram and L1-normalized.

Holistic classifier (baseline). The straightforward approach to utilize the different extracted feature sets is to concatenate their 4,000-dimensional descriptors to form a higher dimensional feature vector. The merged 20,000-dimensional descriptor is fed into a single classifier to recognize action classes.

Ensemble method I. The 4,000-dimensional vectors from different representation are fed to their corresponding classifier in the ensemble. Each classifier is hence focusing on a particular kind of descriptor among the five (STIP, TS, HOG, HOF, and MBH). Their outputs are fused by means of DS fusion – as detailed in Section 4.1.1.

Ensemble method II. The third approach follows the underlying idea of the Random Subspace Method (RSM) [68], in which each classifier in the ensemble is trained using a random subset of features. We first concatenated the five above mentioned feature sets and the feature subsets are randomly chosen from 20,000 features. Each subset is used to train an individual classifier in the ensemble. The outputs of ensemble of classifiers are then combined using DS fusion. The number of classifiers as well as the size of the feature subsets need to be experimentally chosen.

Next, we introduce the experimental results on each of the approaches.

4.3 Results

4.3.1 Data and experimental settings

We chose to evaluate our method in the UCF-101 action classification dataset [165]. UCF-101 is a large dataset of manually annotated real action videos collected from YouTube. More precisely, it consists of 13,320 videos belonging to 101 categories, divided into 5 coarse action categories: human-object interaction, body-motion, human-human interaction, playing instruments, and sports. Looking at the videos, we can see there is a large intra-class variation in terms of viewpoint, scale, background clutter, illumination. Also, as expected, real videos contain camera motion, shaking, and bad video quality, which contribute to make the problem even harder.

For the experiments, we divided the 13,320 videos into 3 different train/test splits, following the procedure proposed by [165]. For classification, we adopted a histogram-intersection kernel SVM [107] as our base classifier.

4.3.2 Classification results

We report the results obtained by three different approaches (from Section 4.2) in Table 4.1. As can be seen, the ensemble-based approaches have remarkably improved the results. Specially, our third approach outperforms other state-of-the-art methods with an overall accuracy of 75.05% by averaging over the three training/testing splits. This is slightly better than [130, 112] who reported an average accuracies of 73.39% and 73.10%, and remarkably better than work of Karpathy et al., which is based on Convolutional Neural Networks (CNNs), presented at CVPR 2014 [78]. In addition, the confusion matrix of the third approach for the UCF-101 dataset is shown in Figure 4.3. In the figure, image examples of action classes of low and high confusion are given. In general, the actions which result in the highest amount of confusions, and thereby the lowest recognition accuracies, are actions videos affected by a high amount of camera and/or background motion.

Figure 4.2 shows the classification accuracy of the third ensemble-based approach as a function of the ensemble size for UCF-101 datasets. These observation is consistent with the results of many studies, see [123, 68] as few examples, that is, the ensemble classification performance first improves as the ensemble size increases and then plateaus after a demarcation point, e.g. a value around 40-50% accuracy.

Method	Acc	#1	#2	#3
Karpathy et al. [78]	65.4	-	-	-
Phan, Le, and Satoh [130]	73.39	71.10	73.67	**75.39**
Murthy and Goecke [112]	73.10	-	-	-
Rostamzadeh, Uijlings, and Sebe [146]	70.50	70.45	69.80	71.27
Nga, Kawano, and Yanai [117]	66.26	65.16	66.73	66.90
Cho, Lee, and Jiang [25]	65.95	65.22	65.39	67.24
Páez, Vanegas, and Gonzalez [124]	65.68	65.31	65.48	66.23
Chen, Xu, and Corso [22]	64.30	63.41	65.37	64.12
Burghouts et al. [15]	63.46	62.01	63.46	64.90
Nga and Yanai [118]	60.10	-	-	-
Wang, Li, and Shu [182]	54.74	54.76	55.16	54.29
Soomro, Zamir, and Shah [165]	43.90	-	-	-
Single feature classifiers				
STIP + BoVW	42.56	42.12	41.89	43.67
DT (TS) + BoVW	49.88	49.76	50.05	49.83
DT (HOG) + BoVW	51.10	50.19	51.76	51.35
DT (HOF) + BoVW	46.59	46.47	46.69	46.60
DT (MBH) + BoVW	62.93	62.54	62.78	63.46
Ensembling feature classifiers				
Baseline: early feature fusion	60.73	61.13	60.11	60.95
Ensemble I: DS fusion	69.10	69.43	68.09	69.79
Ensemble II: RMS	75.05	75.11	74.80	75.23

TABLE 4.1: Comparison of our proposal to accuracies (%) from state-of-the-art recognition methods on UCF-101 (global overall accuracy and on the 3 different train/test splits)

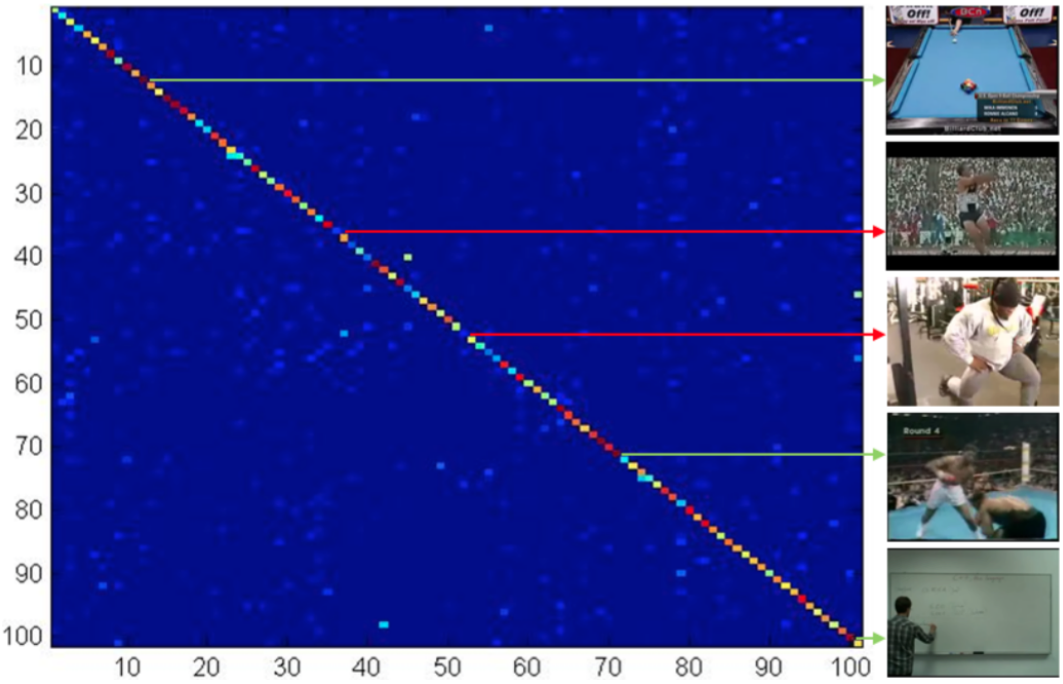


FIGURE 4.2: Confusion matrix of the ensemble classification system (third approach) for the UCF-101 dataset. The green and red arrows point towards image examples of action classes of low (billiards shot, punch and writing on board) and high (hammer throw and lunges) confusion, respectively

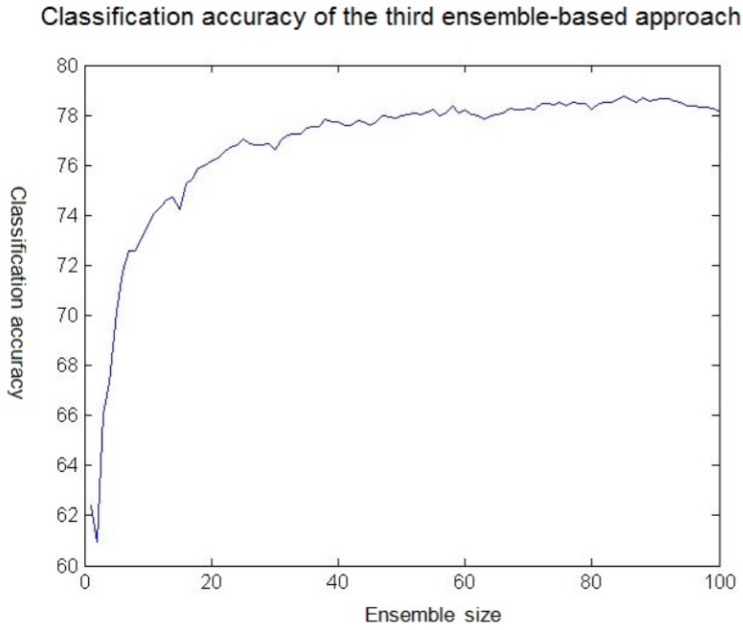


FIGURE 4.3: Accuracy of ensemble classification method versus the ensemble size

Chapter 5

Darwintrees: modeling the hierarchical evolution of spatiotemporal dynamics in action videos

The extra temporal dimension added by videos with respect to images makes it harder for all kinds of recognition approaches. Among the most successful hand-crafted methods, we find the Improved Dense Trajectories (IDT) [183] framework. This improved version presents several advantages over the original work [185]: (1) estimation global motion between frames to account for camera motion and (2) aggregation into a more discriminative Fisher Vector (FV) descriptor (replacing BoVW). Nonetheless, similarly to BoVW, FVs encode local descriptors into a compact representation lacking information about spatial or temporal relations among trajectories. Fernando et al. [47] proposes modeling the temporal evolution of features via a mid-level representation that is able to capture the temporal dynamics. After per-frame IDT+FV feature computation, a linear regressor/ranking model learns the ordering of per-frame features in the video. The parameters of the model are then used as a video's functional representation that can be classified using a discriminative classifier. The approach demonstrated its effectiveness over IDTs and other hand-crafted methods in several action benchmarking datasets, especially when combined with deep-learning [48].

We claim that one of the flaws of this method is a preprocessing step consisting in smoothing the per-frame FV features along the temporal dimension prior to regression/ranking. This makes the learner/ranker parameters more stable to intra-class variations. For smoothing, a time-varying mean operation is used: at each time instant, the per-frame features are averaged with features of previous frames. However, this same operation has the side effect that for later frames the mean operation vanishes any new information if the representation has already accumulated so much information. Given that, [47] proposes a fix that performs videodarwin in the reverse time direction (from the end to the beginning of the video) in order to capture the information in the end-part of the video that is missed when performing forward videodarwin. While this makes the method more robust for relatively short videos, this solution is ill-conditioned for longer ones.

Our hypothesis is that a meaningful spatio-temporal segmentation of the video would be a simple yet effective way to obtain shorter video parts that would potentially suffer less from the smoothing problem caused by the time varying mean. Gaidon et al. [52] already proposed an unsupervised algorithm, based on spectral embedding, to build an unordered binary tree of dense trajectory clusters. They

compute a bag-of-words representation in each cluster and use a tree-distance kernel for classification. They demonstrate the use of clusters yields a better result in terms of action recognition.

In this work, action patterns take the form of unordered binary trees of richly represented nodes, namely *darwintrees*. First, we cluster trajectories as in [52]. Then, we use videodarwin to model the evolution of the features not only in the tree nodes, but also along the tree branches (see Figure 5.1). Using the latter, we model information of how the clusters and their parents relate as we keep dividing/grouping them. This is not only a change of paradigm in videodarwin – initially thought to model the evolution of features throughout time –, but also a way of describing variable-sized tree branches that in our experiments demonstrated to be reliable for classification providing complementary information w.r.t. the node representations. Figure 5.2 illustrates the different stages of the proposed framework for action recognition. We achieve state-of-the-art results in UCF Sports Actions and Highfive datasets. Our darwintree proposal reduces the high computation and memory requirements of deep-learning methods at the same time that achieves state-of-the-art results on two benchmark datasets. Our method does not require large amounts of data, neither clipping segments nor temporal resizing of the video. It naturally handles long-term temporal dependencies and exploits motion information by using specifically designed motion features (in particular, MBH) for the task of action recognition.

The remainder of the chapter is organized as follows: Section 5.1 introduces our approach and Section 5.2 brings details about benchmarking datasets, implementation, parametrization, results, and final discussion.

5.1 Method

The proposed system for action recognition is shown in Figure 5.2. For a particular video, we first extract improved dense trajectories (see Section 5.1.1). Then, we run a divisive hierarchical clustering algorithm based on the spectral embedding using Nyström method on a matrix of tracklet similarities (see Section 5.1.2). In a third stage, we use the derived binary tree structure to construct two main mid-level representations: one modeling the evolution of tracklet features on nodes and another modeling the tracklet features at tree branch level (see Sections 5.1.5 and 5.1.6). As base features, we use the well known Fisher Vectors (FV). In case of node-videodarwin, FV are computed per frame, whereas in the case of branch-videodarwin one FV is used as a global node descriptor. We use a kernel function able to compute the similarity of two trees based on pairwise similarities of mid-levels (Section 5.1.7). Finally, we apply binary SVM classifiers for the actual action recognition. Prediction scores from different kernels are fused in an early fusion approach by using a linear SVM classifier. The method is unsupervised until the classification part.

Next, we describe in detail each stage of darwintrees.

5.1.1 Extraction of improved trajectories and trajectory-pooled descriptors

For the computation of trajectories, we rely in improved dense trajectories (IDTs) from [183]. Differently from DTs, improved ones account for camera motion between consecutive frames. More precisely, feature points from two frames are matched

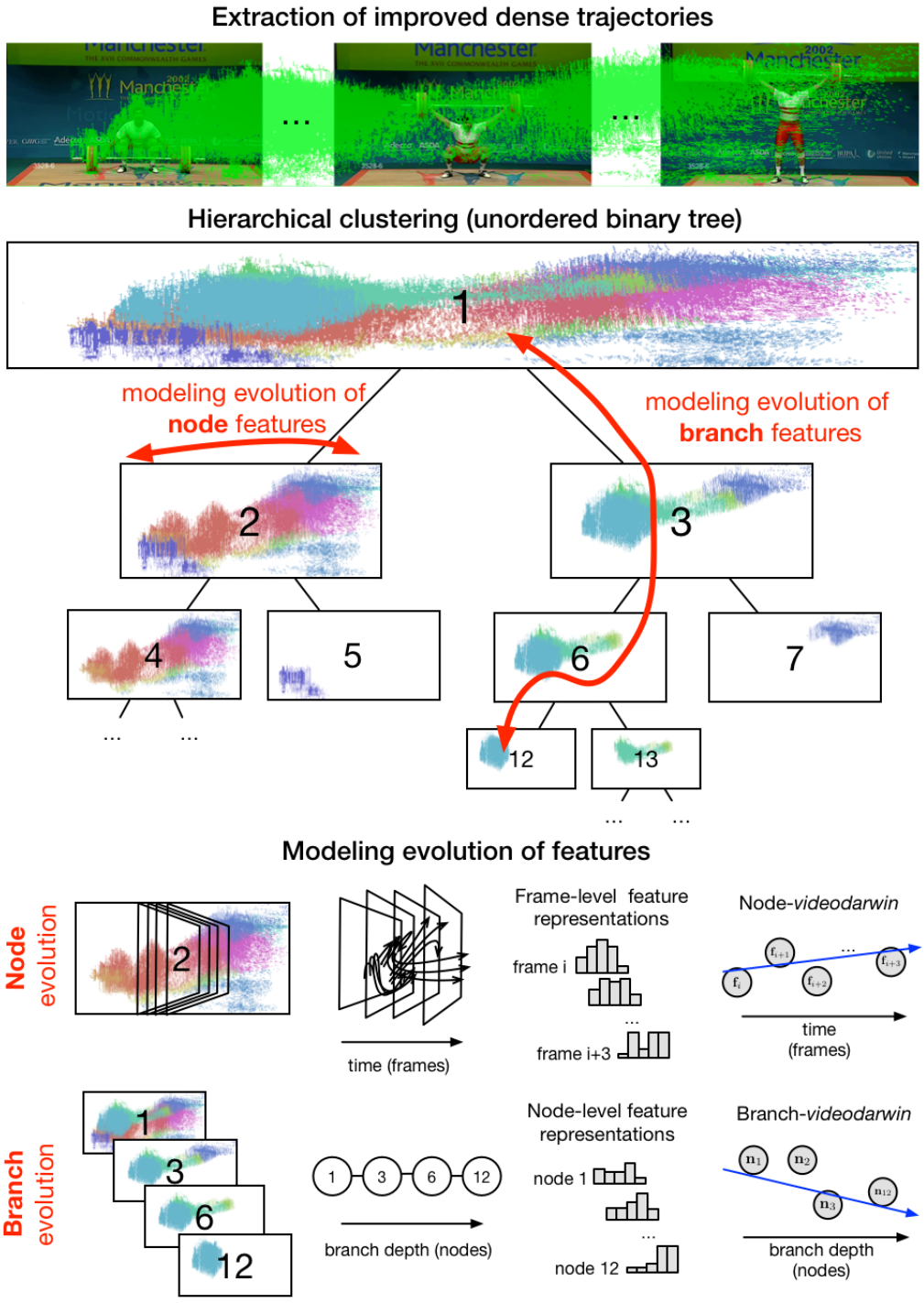


FIGURE 5.1: After the extraction of improved dense trajectories (green), we run a divisive clustering algorithm in order to obtain meaningful groupings of trajectories. Then, we perform *videodarwin* both on nodes (modeling the evolution of node frame features) and on tree branches (modeling the evolution of node global representations)

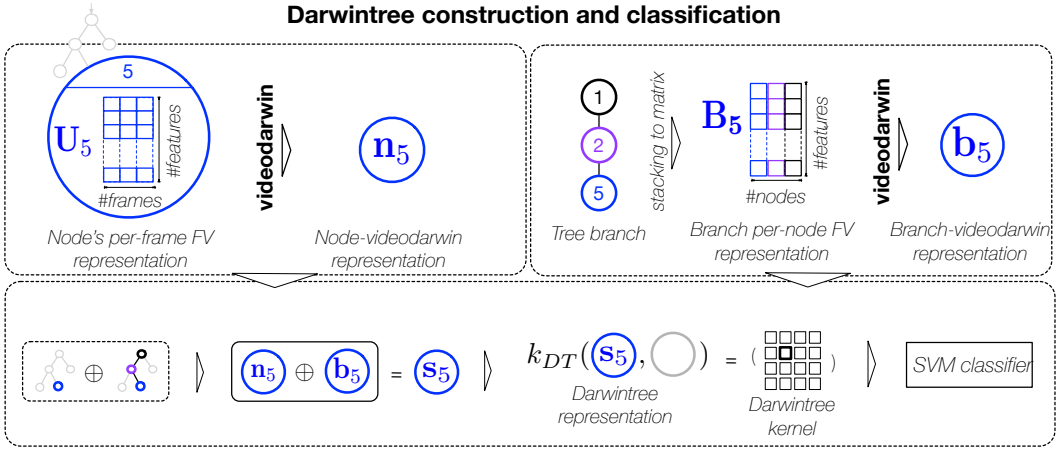


FIGURE 5.2: Proposed framework for action recognition. First, node and branch representations are created. The darwintree representation is constructed from the concatenation of the **n** and **b**. Finally, the darwintree kernel is computed and input to a support vector machine (SVM) classifier

using a combination SURF descriptors [10] and dense optical flow vectors. Then the homography explaining the global between those is computed using RANSAC [49]. The global motion between frames is assumed to be caused by the camera and all the trajectories that are consistent with the homography model are thus discarded.

Among the trajectory-aligned descriptors, we only use the Motion Boundary Histogram (MBH) descriptor since it proved to be enough to obtain state-of-the-art results in most action recognition datasets [122] and will save us computation time. More precisely, the average-pooled MBH descriptor for each of the $n_x \times n_y \times n_t$ cells around the trajectory and concatenated them in a feature vector to represent the trajectory. For the sake of simplicity, the rest parameters and details related to the trajectory extraction are kept as in [183].

5.1.2 Clustering of trajectory paths into binary tree structures

We cluster the extracted trajectories on each particular video, each trajectory instance being represented by its spatio-temporal positions and velocities, i.e. $T = \{\mathbf{t}_x, \mathbf{t}_y, \mathbf{t}_z, \mathbf{t}_{v_x}, \mathbf{t}_{v_y}\}$. Note positions are vectore of size L, i.e. the length of the trajectory's path, and velocities of $L-1$. We also define the mean spatial position of a trajectory as $\bar{\mathbf{p}}_T = (\bar{p}_x, \bar{p}_y, \bar{p}_z) = (\frac{1}{L}\sum_{i=1}^{L} t_{x,i}, \frac{1}{L}\sum_{i=1}^{L} t_{y,i}, \frac{1}{L}\sum_{i=1}^{L} t_{z,i})$.

Following the approach of [52], we first filter sparse trajectories within each video based on a sparsity criterion. A trajectory is filtered out if the average euclidean distance between its position $\bar{p}_T$ and k-nearest neighbor trajectories' positions, i.e. $\frac{1}{k}\sum_{\bar{\mathbf{p}}'_T \in \mathcal{N}(\bar{\mathbf{p}}_T)} ||\bar{\mathbf{p}}_T - \bar{\mathbf{p}}'_T||_2$, is greater than the mean and deviation of spatial distances among all trajectories and their neighbors. The neighbors search $\mathcal{N}$ is limited to a temporal window $[\bar{p}_z - r, \bar{p}_z + r]$ and efficiently computed using KD-trees.

Next is to construct a similary matrix between pairs of filtered trajectories, to be input to the spectral divisive clustering. More precisely, a pairwise similarity matrix is constructed for each feature using a RBF Gaussian kernel: $K_{\text{RFB}-\text{gauss}}(\mathbf{t}, \mathbf{t}') = exp\left(-\frac{||\mathbf{t}-\mathbf{t}'||_2}{2\hat{d}}\right)$, where $\mathbf{t} \in T$ and $\hat{d}$ is the median of the distances between the corresponding tracklet features. Then, similarity matrices of different features are aggregated via element-wise product: $\mathbf{A} = \mathbf{A}_x \odot \mathbf{A}_y \odot \mathbf{A}_t \odot \mathbf{A}_{v_x} \odot \mathbf{A}_{v_y}$, which ensures

the positive-definiteness property of $\mathbf{A}$. This matrix can conveniently be interpreted as the adjacency matrix of a graph weighting pairwise affinities of trajectories from which we want to perform optimal recursive bipartitioning cuts in order to eventually construct our binary tree-form structures.

Given a pairwise affinity matrix such as $\mathbf{A}$, we can use spectral grouping/clustering, that is, to embed the trajectories into a projection in the eigenvector space from which we can compute the actual clusters. However, having on the order of $T = 10^6$ trajectories makes the computation of $\mathbf{A}$ hard for any eigensolver. Nyström approximation method [51] instead, allows to use a small portion of the trajectories to extrapolate the results and obtain the approximate leading eigenvectors we need. After that, we use the divisive hierarchical clustering algorithm proposed by [52] to recursively threshold on the leading eigenvectors' values and build the corresponding unordered binary tree. As in [52], we use set the minimum and maximum number of trajectories per cluster to be 200 and $2,000$ respectively, which helps keeping the trees balanced.

5.1.3 Tree-based mid-level representations

At this point, we have obtained a tree-form representation grouping extracted trajectories in a video. We aim next to build two intermediate representations for the nodes to be further used in the node and branch videodarwin computation.

[52] proposed a bag-of-words (BoW) on the node trajectories as the final representation for the nodes. Instead, we exploit the higher discriminant power of FVs [129], but only as an intermediate representation. More precisely, a FV will be computed for each node and stacked as a prior step to the later branch videodarwin computation. On the other hand, the node videodarwin requires an intermediate representation for a node consisting of stacked per-frame FVs.

5.1.4 Holistic Videodarwin

The key idea behind VideoDarwin is to model how features evolve throughout the video sequence. This temporal modeling demonstrated superior performance over other representations such as plain fisher vectors [47, 48]. However, the sparsity of per-frame FVs requires a smoothing in order to make the final VD representation more robust to noise and hence more invariant across same class sequences. Next, we briefly explain the videodarwin computation for the entire video sequence, followed by node and branch videodarwin variations.

Let denote $\mathbf{V} = [\mathbf{v}_1, \ldots, \mathbf{v}_F] \in \mathbb{R}^{2DK \times F}$ the stack of per-frame feature vectors, where F is the number of sequence frames. The columns of $\mathbf{V}$ are smoothed in both directions, forward (+) and reverse (-), as follows:

$$\tilde{\mathbf{v}}_i^+ = \varrho_1 \left(\frac{1}{i} \sum_{j=1}^{i} \mathbf{v}_j \right) \tag{5.1}$$

$$\tilde{\mathbf{v}}_i^- = \varrho_1 \left(\frac{1}{i} \sum_{j=0}^{i-1} \mathbf{v}_{i-j} \right) \tag{5.2}$$

where $\tilde{\mathbf{v}}$ is the smoothed version of $\mathbf{v}$ and $\varrho_1(\cdot) = \frac{\cdot}{||\cdot||_1}$ is the L1-normalization function.

After that, we simply train two Support Vector regressor (SVR) models – one for each smoothing direction – on $\{(\tilde{\mathbf{v}}_i^+, i)|1 \leq i \leq F\}$ and $\{(\tilde{\mathbf{v}}_i^-, i)|1 \leq i \leq F\}$ respectively. In particular, we use a linear SVR model[1] with $C = 1$. Finally, the parameters of the trained regressors, namely $\mathbf{\Theta}^+, \mathbf{\Theta}^- \in \mathbb{R}^{2DK}$ are concatenated to obtain the final videodarwin signature: $\mathbf{w} = [\mathbf{\Theta}^+, \mathbf{\Theta}^-] \in \mathbb{R}^{4DK}$.

The video meta-descriptor $\mathbf{w}$ can be input to a discriminative classifier, e.g. SVM, for video classification. Despite good results have been obtained, we argue that the time varying mean from Eq. 5.2 can be problematic for longer sequences, by causing $\tilde{\mathbf{v}}_i$ for i values close to F to be too smoothed by averaging over all larger number of frames and therefore highly invariant independently from the features at time i. Next, we see how to compute darwintree over deeper nodes in the tree which have shorter temporal extents and, thus, alleviate this degradation caused by the smoothing.

5.1.5 Node videodarwin

Our approach provides an additional solution to the whole-video smoothing degradation provided by the tree decomposition from Section 5.1.2. Nodes with smaller groups of trajectories are likely to span shorter time intervals within a F-frame video, The node videodarwin representation of a node is hence computed as explained in Section 5.1.4 but with inputs $[\mathbf{v}_a, \ldots, \mathbf{v}_b]$, where $[a, b]$ s.t. $1 \leq a \leq b \leq F$ is the time interval spanned by the node trajectories. Therefore, the two regressors are trained on the smoothed node representations: $\{(\tilde{\mathbf{v}}_i^+, i) \mid a \leq i \leq b\}$ and $\{(\tilde{\mathbf{v}}_i^-, i) \mid a \leq i \leq b\}$. Finally, the node videodarwin is obtained by concatenating the two regressor parameters: $\mathbf{n} = [\mathbf{\Theta}^+_{\text{node}}, \mathbf{\Theta}^-_{\text{node}}]$.

5.1.6 Branch videodarwin

Videodarwin was originally intended to model changes in the temporal dimension. This variation introduces its use in tree branches, i.e. modeling the evolution of node features. We define a branch as the path from a node up to the tree root. In other words, the j-th node has its associated branch whose length is exactly $log_2(j)$.

In particular, let $\mathbf{u}_j \in \mathbb{R}^{2DK}$ be the j-th node global FV representation. Then the stack of per-node representations from the node itself (j) to the tree root node (1):

$$\mathbf{B}_j = [\mathbf{b}_j, \mathbf{b}_{j-1}, \ldots, \mathbf{b}_1] = [\mathbf{u}_{\lfloor j/2^0 \rfloor}, \mathbf{u}_{\lfloor j/2^1 \rfloor}, \ldots, \mathbf{u}_{\lfloor j/2^{\log_2(j)} \rfloor}] \tag{5.3}$$

where $\lfloor \cdot \rfloor$ refers to the floor operation.

Given $\mathbf{B}_j$, the procedure from Section 5.1.4 is also applied. We train two regressors on the smoothed branch representations in both directions $\{(\tilde{\mathbf{b}}_i^+, i) \mid 1 \leq i \leq j\}$ and $\{(\tilde{\mathbf{b}}_i^-, i) \mid 1 \leq i \leq j\}$ so as to model $\mathbf{\Theta}^+_{\text{branch}}$ and $\mathbf{\Theta}^-_{\text{branch}}$ respectively. Those somehow explain the evolution of $\mathbf{u}$ features along the j-th node branch, either ascending (forward videodarwin) or descending (reverse videodarwin) the branch. Finally, branch videodarwin representation of the j-th is computed: $\mathbf{b}_j = [\mathbf{\Theta}^+_{\text{branch}}, \mathbf{\Theta}^-_{\text{branch}}] \in \mathbb{R}^{4DK}$.

Note it does not make sense to compute paths of length of 1, hence branch videodarwin for the root note $\mathbf{b}_1$ is not defined. In other words, the final number of branches is equal to the number of nodes in the tree minus 1.

[1] The SVR code used is from LIBLINEAR (`https://www.csie.ntu.edu.tw/~cjlin/liblinear`). In particular, L2-regularized L2-loss support vector regression (primal) version specified with the program argument "-s 11".

5.1.7 Darwintree classification

First of all, note that binary tree structures may have a variable number of nodes (and branches). In order to perform classification, we need to be able to compute some measure of similarity between any pair of trees $(\mathcal{E}, \mathcal{E}')$. However, each tree may have a different number of nodes – or branches. In this context, the authors of [52] prove the accumulation of pair-wise node similarities to be effective for tree discrimination on their *All Tree Edge Pairs* (ATEP) kernel. They also found better results are obtained by using edge representations, i.e. the concatenation of child and parent node representations, than by only the child. In our work, we define our own representation by combining node-branch representations for the computation of the *darwintree kernel*.

Given a tree $\mathcal{E}$, let first cast $(\mathcal{N}, \mathcal{B})$ to the set of joint node-branch representations $\mathcal{S}$. The tree new structure becomes $\mathcal{E} = \{\mathbf{w}, \mathcal{S}\}$, where $\mathcal{S} = \{\mathbf{s_i} = [\mathbf{n_i}, \mathbf{b_i}] \mid 1 < i < |\mathcal{B}|,\ \mathbf{s}_i \in \mathbb{R}^{8DK}\}$. Given the node-branch representations, we compute the darwintree kernel based on the pairwise similarity of those merged node representations:

$$K_{\mathrm{DT}}(\mathcal{S}, \mathcal{S}') = \frac{1}{|\mathcal{S}||\mathcal{S}'|} \sum_{\mathbf{s}_i \in \mathcal{S}} \sum_{\mathbf{s}_j \in \mathcal{S}'} \phi(\mathbf{s}_i, \mathbf{s}_j)\ ,\ \forall i, j > 1 \tag{5.4}$$

where $\phi(\cdot, \cdot)$ can be any valid mapping function, e.g. dot product for linear mapping. The normalization factor $\frac{1}{|\mathcal{S}||\mathcal{S}'|}$ ensures the amount $K_{\mathrm{DT}}(\mathcal{S}, \mathcal{S}')$ remains scaled for any pair of trees $(\mathcal{S}, \mathcal{S}')$, independently from the number of nodes in each tree.

5.2 Results

5.2.1 Datasets

We evaluate our approach on UCF sports actions dataset [143], Highfive [126], and Olympic Sports [121]. UCF sports actions contains 150 examples and 10 classes of actions from different sports, presenting different backgrounds and camera movement. In our experiments, we used the standard 103/47 train/test split. Highfive consists of 300 examples of human interactions from TV shows, from which 200 are handshake, high five, hug, kiss, whereas the 100 remaining ones are negative examples. For validation, we followed the 2-fold cross validation provided along with the dataset. We report both accuracy and mean average precision (mAP). Finally, Olympic Sports contains 783 instances from 16 classes partitioned in a 640/143 train/test holdout split. The results obtained are expressed in mAP. In Figure 5.3, we introduce the three datasets and some general details.

5.2.2 Code implementation

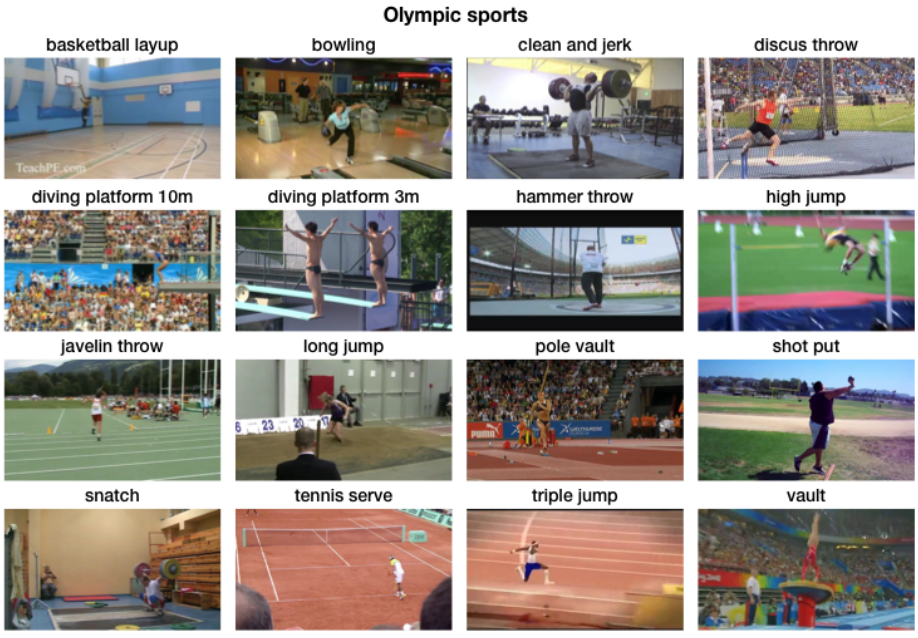
We constructed the unordered binary trees of trajectories using public code[2] (with default parameters) provided by the authors of [52]. On the other hand, we used the videodarwin implementation[3] from [47] as a base for the construction of our darwintrees. For classification and evaluation metrics, we used Python's *sklearn* machine learning library. In particular, for multi-class classification we choose sklearn's one-vs-rest classification over the one-vs-one LibSVM's implementation. Moreover, Eq. 5.4 is optimized for GPU computation with Python's *pycuda* library.

[2]Tree structure and hierarchical divisive algorithm for spectral clustering: `https://gist.github.com/daien`.

[3]Videodarwin code: `https://bitbucket.org/bfernando/videodarwin`.

(A) No. instances: 150, classes: 10, data partitioning: 103/47 train/test hold-out, evaluation metric: accuracy

(B) No. instances: 300, classes: 4 positive + 1 negative, data partitioning: 2-fold cross validation, evaluation metric: mAP



(C) No. instances: 783, classes: 16, data partitioning: 640/143 train/test hold-out, evaluation metric: mAP

FIGURE 5.3: Illustration and details of the datasets used in our experiments

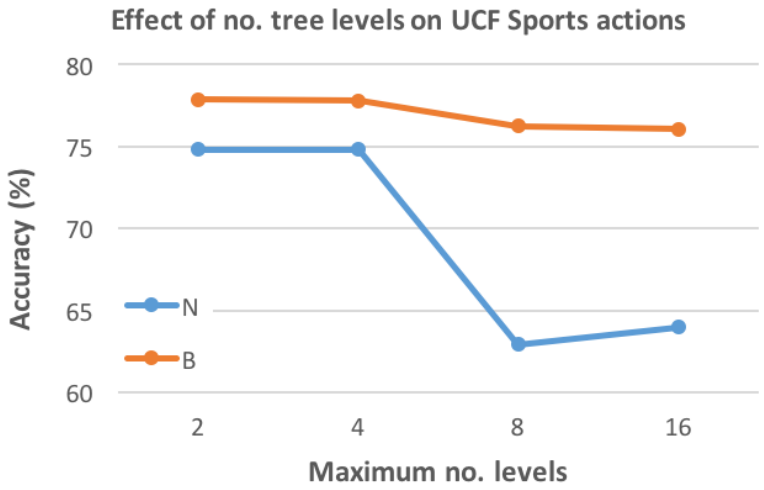


FIGURE 5.4: Performance varying the number of maximum tree levels on UCF Sports actions in terms of accuracy (%). Experiments in the validation dataset showed videodarwin on noisy deeper tree nodes causes our node representation (N) to underperform in comparison to the branch representation (B) that remains much more stable

Method	UCF [143] (ACC)	Highfive [126] (mAP)		
		Fold 1	Fold 2	TOTAL
Node-VD	85.11	76.55	70.41	73.48
Branch-VD	80.85	76.25	**72.53**	74.39
Darwintree (DT)	**91.49**	76.04	70.37	73.21
VD+DT	**91.49**	**79.24**	72.32	**75.78**

TABLE 5.1: Results of the different methods in the two benchmarking datasets for node-videodarwin, branch-videodarwin, darwintree (DT), and the combination of DT with holistic videodarwin (VD)

5.2.3 Trajectory features, GMMs, fisher vectors, and spectral clustering

We extracted MBH features along the trajectories, applied the "square-root trick" as in [183], and reduced their 192-dimensional descriptors by a factor of 0.5 using PCA. We observed the other trajectory features (trj, HOG, and HOF) provided a marginal improvement. For the GMMs, We used 256 mixtures trained on 1 million randomly sampled MBH descriptors. This yielded fisher vectors of dimensionality $2 \cdot 96 \cdot 256 = 49,152$. As suggested by [48], prior to the videodarwin computation we applied posneg mapping first to the fisher vectors; this is $\mathbf{v} = \sqrt{\mathbf{v}_{\text{pos}} \oplus \mathbf{v}_{\text{neg}}}$, where $\mathbf{v}_{\text{pos}}$ is the $\mathbf{v}$ with all zeros except for the positive coefficients and $\mathbf{v}_{\text{neg}}$ all zeros with all the negatives turned into positive. After posneg mapping, we also applied l2-normalization. For the spectral clustering, we stick to the parameters given by [52], except for the maximum number of tree levels (default value is 62). We experimentally found that in very deep trees, deeper nodes tend to be noisy and cause great impact in the performance of node-videodarwin. We experimentally found a value of 4 levels to be a conservative value. Figure 5.4 shows the ablation experiment on number of levels in UCF Sports Actions.

Method	mAP
Videodarwin (VD)	88.34
Node-vd (N-VD)	83.17
Branch-vd (B-VD)	87.70
Darwintree (DT)	84.38
VD + DT	**88.84**

TABLE 5.2: Olympic Sports dataset [121]

5.2.4 VideoDarwin, kernel maps, and classification

Since videodarwin representations consist of both forward and reverse videodarwin (depending on the direction of the mean time varying operation) parts, we come up with a final representation that doubles the size of the fisher vectors, i.e., 98,304 dimensions. This gives a descriptor of $98,304 \times F$ for a video of F frames. For classification, we kernel mapped the VideoDarwin representation using "RootSIFT" [3] and l2-normalized them. As a last step, different mid-level representations were fused at kernel level and the weights assigned were cross-validated. Also kernel normalization factor is applied to the kernels before the aggregation, consisting of dividing each kernel by the maximum value of the diagonal. This is because otherwise when comparing a tree to itself the similarity is not 1. For all our experiments, we fixed the C parameter of the SVM classifiers to 100.

5.2.5 Quantitative results on action classification

We illustrate our results in the benchmarking datasets on Table 5.1, in which we compare our different approaches among them: node-videodarwin (N), branch-videodarwin (B), the combination of both (Darwintree or DT), and the combination of the latter with the holistic representation (VD+DT). Despite DT and VD+DT got the same results, we found VD+DT to be potentially better from training data: +2.81% (79.80 against 76.99) on average. We also compared our approach (VD+DT) to the holistic videodarwin representation in UCF Sports Actions in which DT obtained better performance: 91.49% vs 87.23%.

To provide more insight about the classification accuracy, we show the results for the different action classes on UCF sports dataset in Figure 5.5.

Since UCF and H5 are fairly small datasets, we also performed some analysses on Olympic Sports dataset. In our experiment, we compared H, N, B, NB, and H+NB representations in Table 5.2. Despite N, B, and the combination NB obtained poorer results than H, combining NB with H yields slight improvement of +0.5% mAP points with respect to H.

5.2.6 Confusion matrices

In Figure 5.7, we show confusion matrices for the multi-classification experiment performed in the test set of UCF Sports Actions dataset, which consists of 47 examples. Note the errors of VD, nVD, and bVD are often uncorrelated. For instance, bVD corrects the confusion among true class "Golf Swing" and predicted "Kicking" in VD and nVD. On the other hand, VD correctly predicts an actual "Kicking" example confused in both N and B. When we combine VD+DT, we avoid any confusion between these two classes. VD and nVD also confuse a "Running" example with

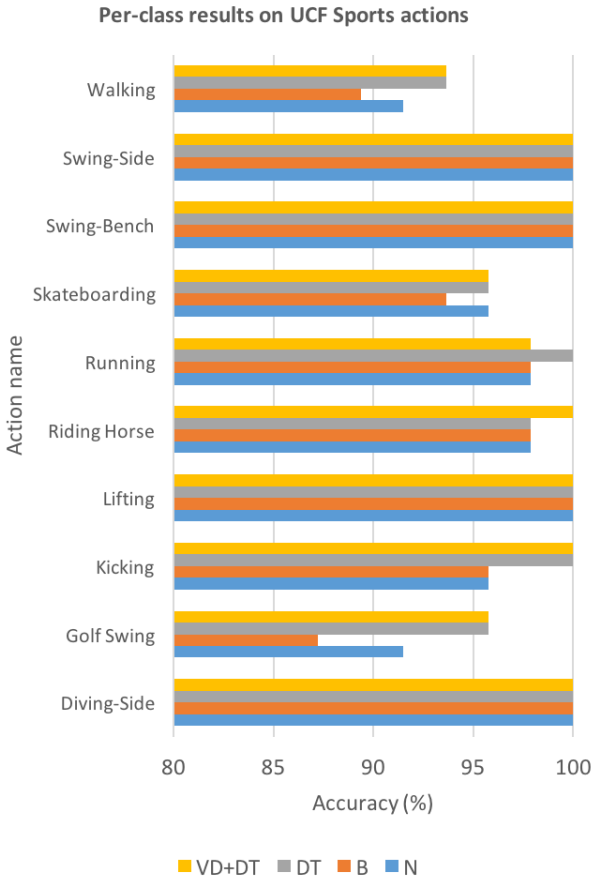


FIGURE 5.5: Results on the different action classes of UCF Sports actions in terms of accuracy (%)

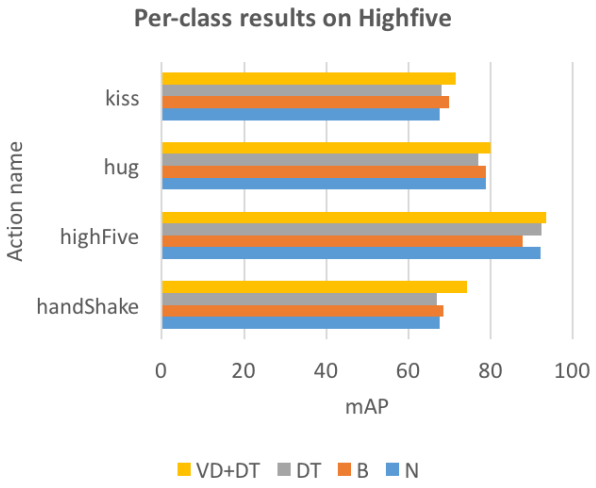


FIGURE 5.6: Results for the different action classes in terms of mean average precision (mAP)

"Walking", whereas bVD does not. DT also predicts all "Running" examples correctly, but then VD reintroduces the error in VD+DT. Nonetheless, VD benefits the DT representation by correcting the confusion among an example of "Walking" and "Riding Horse".

We also show confusion matrices for a multi-classification experiment on Highfive in Figure 5.8. This experiment is independent from the one used to report mAP results in the main text [4]. For this dataset, there is a fifth class containing negative examples ("negative"), which is introduced only used during training. This class introduces most of the confusion in the multi-classification: for VD, only 6 out of 100 confusions are among positive classes, the rest being between a positive class and the negative ones. In particular, nVD and bVD are able to greatly correct the confusion among "hug" and "negative" from 7 to 12 errors (-41.6% error reduction). nVD also demonstrated also to be effective by reducing the confusion among "kiss" and "negative" (-33.3%) and reducing the overall confusion in "highfive" class from 8 to 3 errors (-62.5%).

UCF Sports Actions and Olympic sports are very action-centered, i.e. the camera is focused on the action being performed, without occlusions or challenging illumination conditions, which makes the action recognition task easier – as seen from the results. On the other hand, Highfive contains frequent cutaways and bad illumination conditions (check "handShake" in Figure 5.3b), making the overall recognition task much more challenging.

5.2.7 Qualitative results

We visualized sequence frames with overlaid trajectory clusters, along with the predicted categories by our proposed method. For every sequence shown, 5 frames are evenly spaced frames were sampled. In Figure 5.9 and Figure 5.10, we illustrate results on UCF Sports Actions [143] and Highfive [126] datasets. Notice the compactness of the clusters in both space and time and coherence. Also note, similar actions and viewpoints have similar cluster decompositions (see the two "Golf-Swing-Back" action examples in Figure 5.9a and Figure 5.9b). In simpler actions, as Figure 5.9d and Figure 5.9e, the decompositions are fairly simpler. For instance, Figure 5.9d, it is throughout the temporal dimension (first frames being yellow cluster and latter ones the blue cluster), whereas in in Figure 5.9e the cluster division is along the spatial dimension (upper body trajectories being yellow versus blue ones in lower body parts).

Except for the example depicted in Figure 5.9d, VD+DT is able to predict correctly the actual groundtruth class (GT), despite VD being wrong (only 2/5 hits). Also note that in all the examples except for 5.9d, bVD predicted the wrong class. However, for all of those, nVD was able to predict it right despite nVD also being wrong, as it was the case for 5.9b and 5.9d. This proves DT learns to model the complementary information provided by nVD and bVD.

DT tends to correct bad results got by VD, as seen in 5.9b or 5.9e. Nonetheless, it is also possible that the VD representation causes a bad final prediction, as in 5.9d. In general, however, VD+DT proved to be more effective for the classification task than only DT.

Here, VD+DT correctly predicts Figure 5.10a-5.10d. In Figure 5.10a, while nVD and bVD predict the actual class ("kiss"), VD wrongly predicts "negative" (5). VD faced problems categorizing "hug" – shown in Figure 5.9. We qualitatively illustrate

[4] Recall that when measuring mAP, a binary classifier is trained per class and the mean of their Average Precisions (AP) is reported.

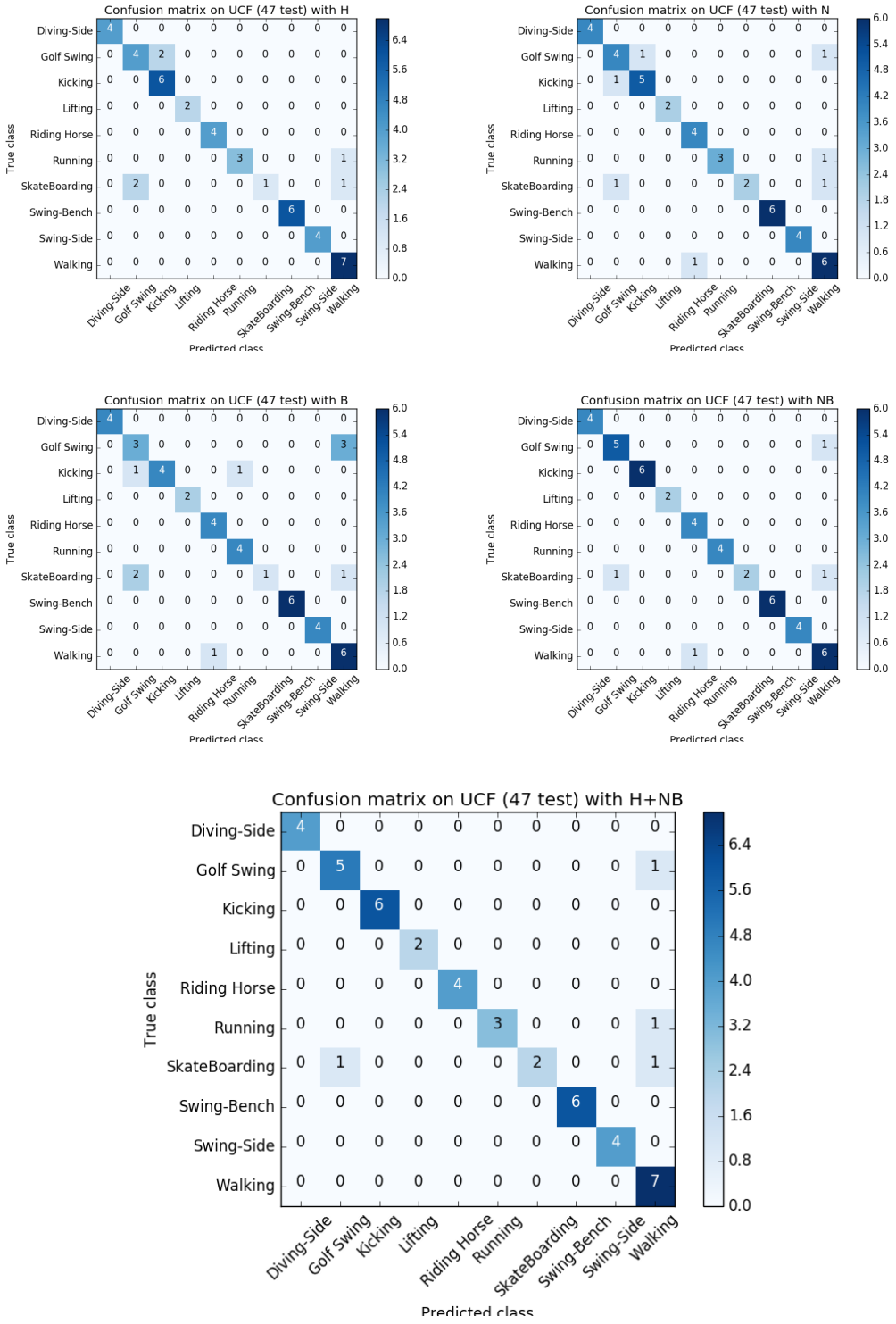


FIGURE 5.7: Confusion matrices from multi-classification experiments on UCF Sports Actions dataset [143]. Numbers in matrix cells refer to absolute quantities (number of examples predicted)

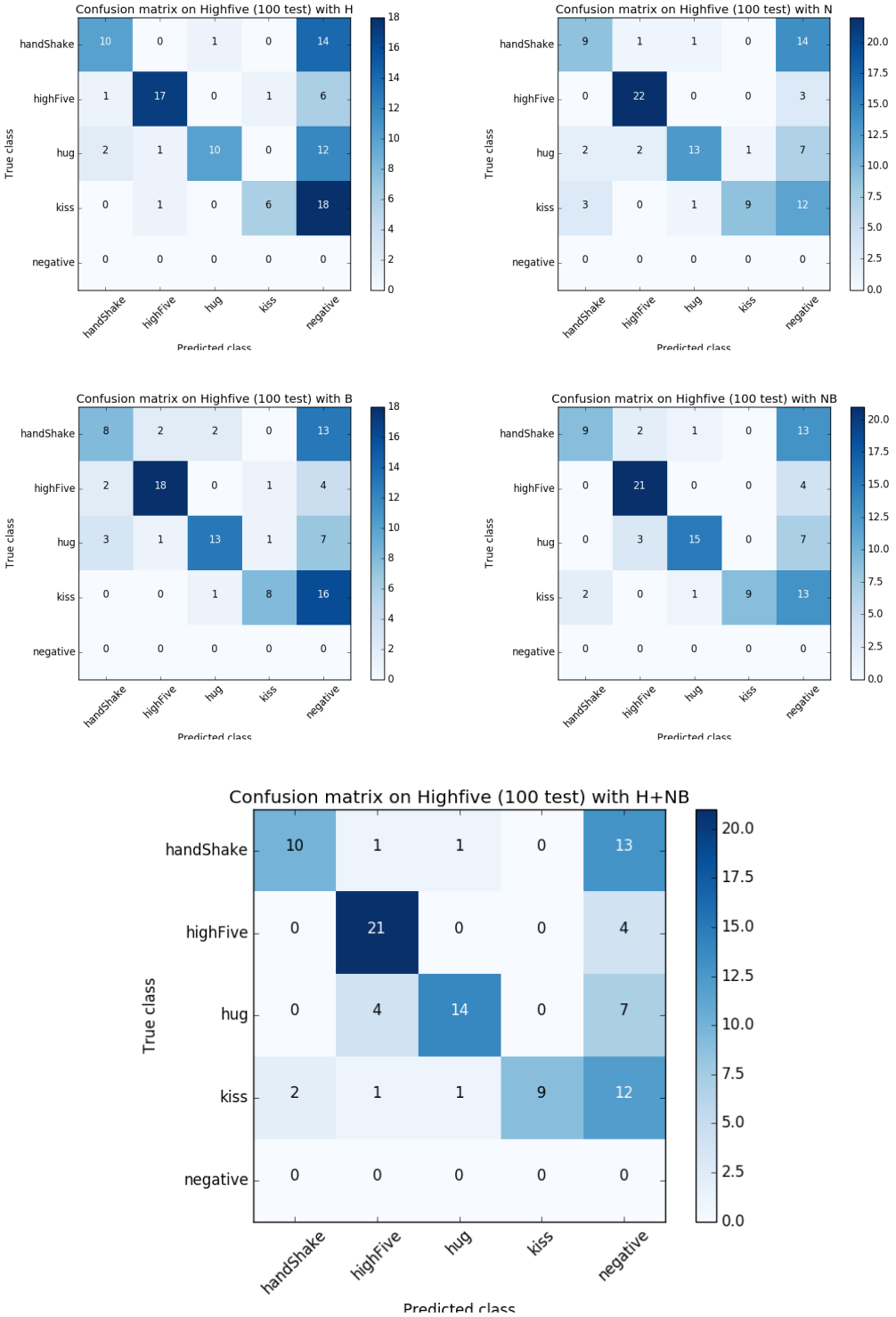


FIGURE 5.8: Confusion matrices from multi-classification experiments on Highfive dataset [126]. Numbers in matrix cells refer to absolute quantities (number of examples predicted). The "negative" was class only used during training phase, but not predicted during testing

(A) **Golf-Swing-Back 002**. GT=1, H=1, N=1, B=9, NB=1, H+NB=1

(B) **Golf-Swing-Back 005**. GT=1, H=2, N=2, B=9, NB=1, H+NB=1

(C) **Kicking-Front 010**. GT=2, H=2, N=2, B=5, NB=2, H+NB=2

(D) **Run-Side 001**. GT=5, H=9, N=9, B=5, NB=5, H+NB=9

(E) **Skate-Boarding-Front 004**. GT=6, H=1, N=6, B=1, NB=6, H+NB=6

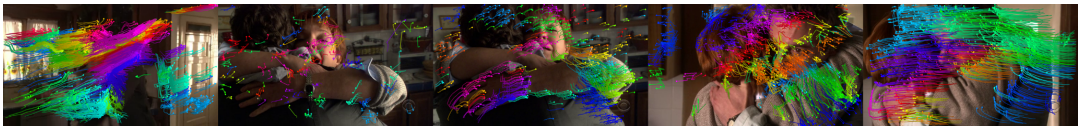
FIGURE 5.9: Visual data and trajectory clusters on 5 frames evenly spaced in time on 5 different UCF Sports Actions' examples [143]. See in the captions of (a)-(e) of subfigures the groundtruth label (GT) and the output of our different methods. Classes are (1) "Diving-Side", (2) "Golf Swing", (3) "Kicking", (4) "Lifting", (5) "Riding Horse", (6) "Running", (7) "Skateboarding", (8) "Swing-Bench", (9) "Swing-Side", and (10) "Walking"

this in Figure 5.10b-5.10d. In Figure 5.10b, bVD agrees with VD; however, nVD provides useful information for correcting this error. In Figure 5.10c and Figure 5.10d, one can see the effectiveness of the DT fusion, which is able to correct both nVD and bVD from a wrong prediction even when they agreed in the wrong label. Finally, in Figure 5.10e, despite the correct prediction of VD, VD+DT missclassifies the example due to the influence of DT in the fused kernel.

5.2.8 Comparison to state-of-the-art methods

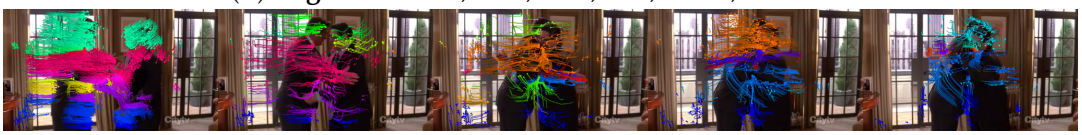
In Table 5.3 and Table 5.4, we compare our approach to state-of-the-art results. As shown, our method achieves better results in both UCF Sports Actions and High-five benchmarking datasets using the standard metrics and evaluation protocols, improving the results in +0.7% and +6.4 points respectively.

(A) **kiss 0038**. GT=4, H=5, N=4, B=4, NB=4, H+NB=4



(B) **hug 0017**. GT=3, H=1, N=3, B=1, NB=3, H+NB=3

(C) **hug 0029**. GT=3, H=1, N=1, B=1, NB=3, H+NB=3



(D) **hug 0002**. GT=3, H=5, N=4, B=4, NB=3, H+NB=3

(E) **kiss 0040**. GT=4, H=4, N=5, B=5, NB=5, H+NB=5

FIGURE 5.10: Visual data and trajectory clusters on 5 frames evenly spaced in time for 5 different examples on the Highfive dataset [126]. See in the captions of (a)-(e) of subfigures the groundtruth label (GT) and the output of our different methods. Classes are: (1) "handShake", (2) "highFive", (3) "hug", (4) "kiss", and (5) negative class

Method	Accuracy (%)
Karaman et al. [77](2014)	90.8
Ma et al. [103](2015)	89.4
Wang et al. [192](2013)	85.2
Ma et al. [104](2013)	81.7
Raptis et al. [139](2012)	79.3
Ours (VD+DT)	**91.5**

TABLE 5.3: UCF-sports dataset [143]

Method	mAP
Wang et al. [184](2015)	69.4
Karaman et al. [77](2014)	65.4
Ma et al. [103](2015)	64.4
Gaidon et al. [52](2014)	62.4
Ma et al. [104](2013)	36.9
Patron-Pérez et al. [126](2012)	42.4
Ours (VD+DT)	**75.8**

TABLE 5.4: Highfive dataset [126]

5.2.9 Discussion

Our results demonstrate the effectiveness of combining node and branch videodarwin in the darwintree representation. In Highfive, the combination of darwintree with holistic videodarwin pushed further the mAP performance. Our method does not need lots of training data in order to generalize. Moreover, the divisive clustering technique based on spectral embedding is an unsupervised technique that does not require annotated training data.

Interestingly, the proposed pipeline is general enough to be applied to any kind of video sequence classification problem. Once a representation per time instant is build, the method can be directly applied.

Part II

Deep-learning methods

Chapter 6

Towards deep action recognition

Despite the application of deep learning to action recognition is relatively new, the amount of research generated is astounding. We think it is critical to compile recent deep-learning based advances on action recoginition. The present chapter aims at capturing a snapshot of current trends in this direction, including an in-depth analysis of the most successful deep models to date, with a particular focus on how these methods treat the temporal dimension of the video data.

The remainder of this chapter is organized as follows: Section 6.1 presents a taxonomy classifying the different deep architectures based on how these deal with the temporal information and Section 6.2 reviews related work for action recognition.

6.1 Taxonomy

We first present a taxonomy that encomprises the main types of deep-learning architectures for action recognition. Then, we explore the fusion strategies used in deep learning-based models for action. After that, we list all the most popular action recognition datasets with benchmarking results.

6.1.1 Architectures

The most crucial challenge in deep-based human action recognition is to properly deal with the temporal dimension. Based on the way this is done, we categorize approaches in a taxonomy that consists of four different groups. The taxonomy is shown in Figure 6.1. The first group are 2D CNNs, those CNNs that purely exploit spatial information. For instance, [168, 194] sample several single frames from the whole video and feed them to a 2D CNN. The predicted class score distributions for the different frames are then averaged to obtain the final video category. The main

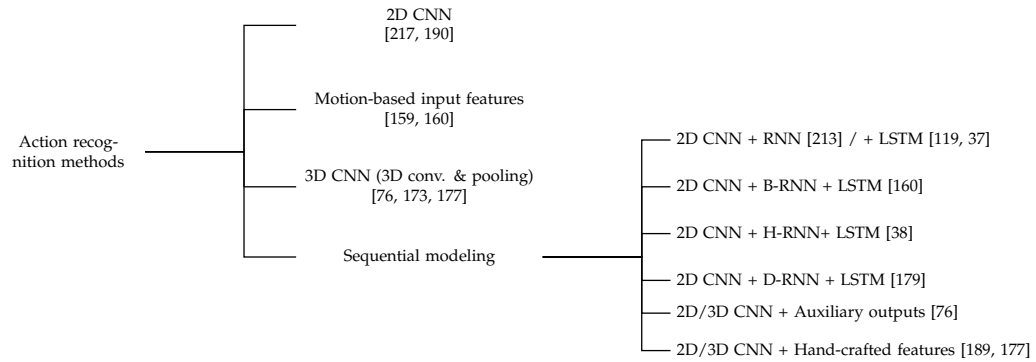


FIGURE 6.1: Taxonomy of deep learning approaches for action recognition

advantage of this kind of models is possibility to use pre-trained 2D CNN models on large image classification datasets, such as ImageNet [83].

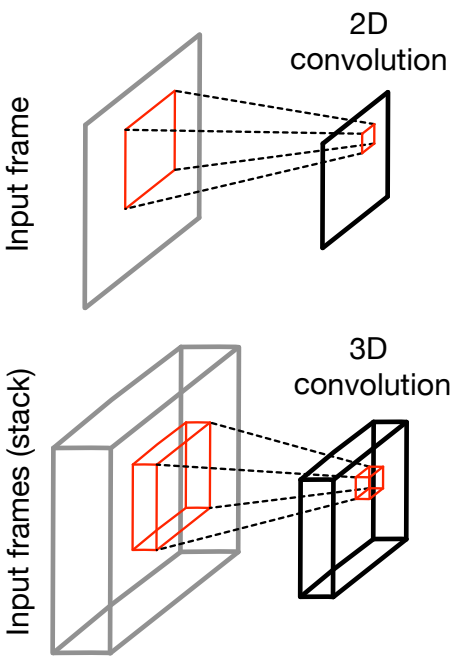
The methods in the second group pre-compute 2D motion features, i.e. optical flow maps, and utilize them as an additional cue apart from RGB frames [159, 189, 57, 168, 199]. Doing so, these kind of methods take into account the very local spatiotemporal information from the pre-computed motion features can provide. A third group extends the 2D filters to 3D, so the convolutional layers produce 3D convolutional outputs responses in the convolutional layers [7, 76, 99, 177]. The 3D convolution and 3D pooling allow to capture discriminative features along both spatial and temporal dimensions while maintaining the temporal structure in contrast to 2D convolutional layers. The spatiotemoral features extracted by this kind of models proven to surpass 2D models trained on the same video data. Figure 6.2a-6.2c illustrate these first three groups.

Finally, the fourth group combines 2D (or 3D) convolutional nets, which are applied at individual (or stacks of) frames, with a temporal sequence modeling. Recurrent Neural Networks (RNN) [41] are one of the most used kind of networks for sequential modeling, being able to take into account the temporal information using recurrent connections in the hidden layers. The drawback of this network is its short memory which is insufficient for real world actions. To solve this problem Long Short-Term Memory (LSTM) networks [55] were proposed. Bidirectional RNN (B-RRN) [131], Hierarchical RNN (H-RNN) [38], and Differential RNN (D-RNN) [179] are some successful extensions of RNN in recognizing human actions. Hidden Markov Models (HMM) have also been used in combination with deep-based methods [202], in this case for gesture recognition. An example of this fourth approach is depicted in Figure 6.2c.

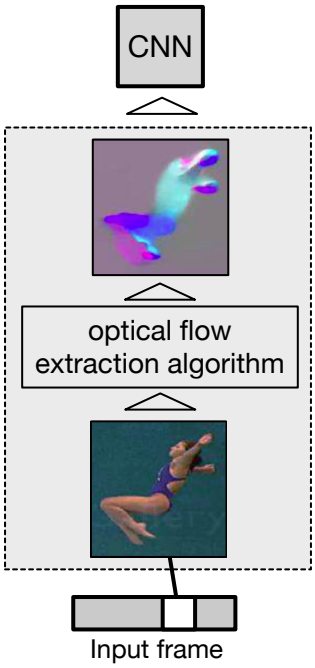
6.1.2 Datasets and benchmarking competitions

At the same time new deep-based methods are being proposed, new and larger datasets appear to fit the requirements of deep learning. That is, basically, having large annotated datasets. Table 6.1 lists the most relevant action recognition datasets. For each entry we specify year of creation; the problems for which the dataset was defined ("action classification" (AC), "temporal localization" (TL), or "spatio-temporal localization" (STL)); involved body parts ("U" for upper body, "L" for lower body, "F" for full body); data modalities available; and, number of classes and the state-of-the-art result. The last column provides best results to date, providing a hint of how difficult the dataset is. Figure 6.3 show some frames of some of the listed datasets.

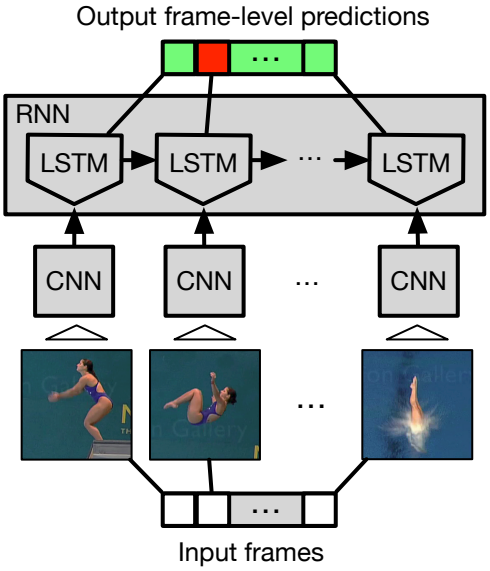
Table 6.2 and Table 6.3 summarize the most recent approaches that obtained remarkable results against two of the most well-known and challenging datasets in action recognition, UCF-101 and THUMOS-14. Reviewing top ranked methods at UCF-101 dataset, we find that the most significant difference among them is the strategy for splitting video data and combine sub-sequence results. [194] encodes the changes in the environment by dividing the input sequence into two parts, precondition and effect states, and then look for a matrix transformation between these two states. [93] processes the input video as a hierarchical structure over the time in 3 levels, i.e. short-term, medium-range and long-range. [177] achieves the best performance by using different temporal 3D convolution temporal extents in both the RGB and optical flow streams.



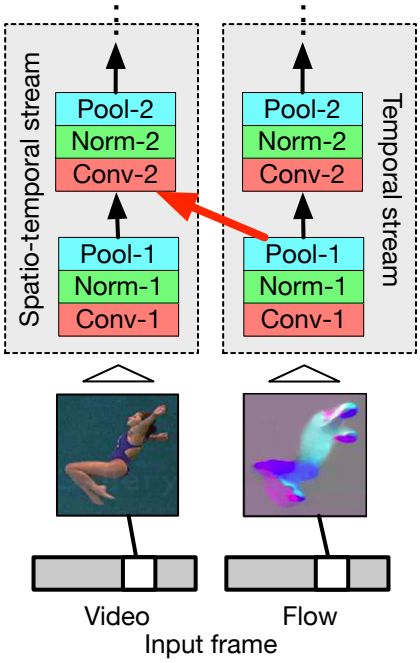
(A) 2D vs. 3D convolutions



(B) Motion-based features (pre-computed optical flow maps as inputs)



(C) Temporal sequence modeling (via LSTM)



(D) Fusion strategies (temporal fused into the spatial stream

FIGURE 6.2: Illustrative examples of the different architectures and fusion strategies

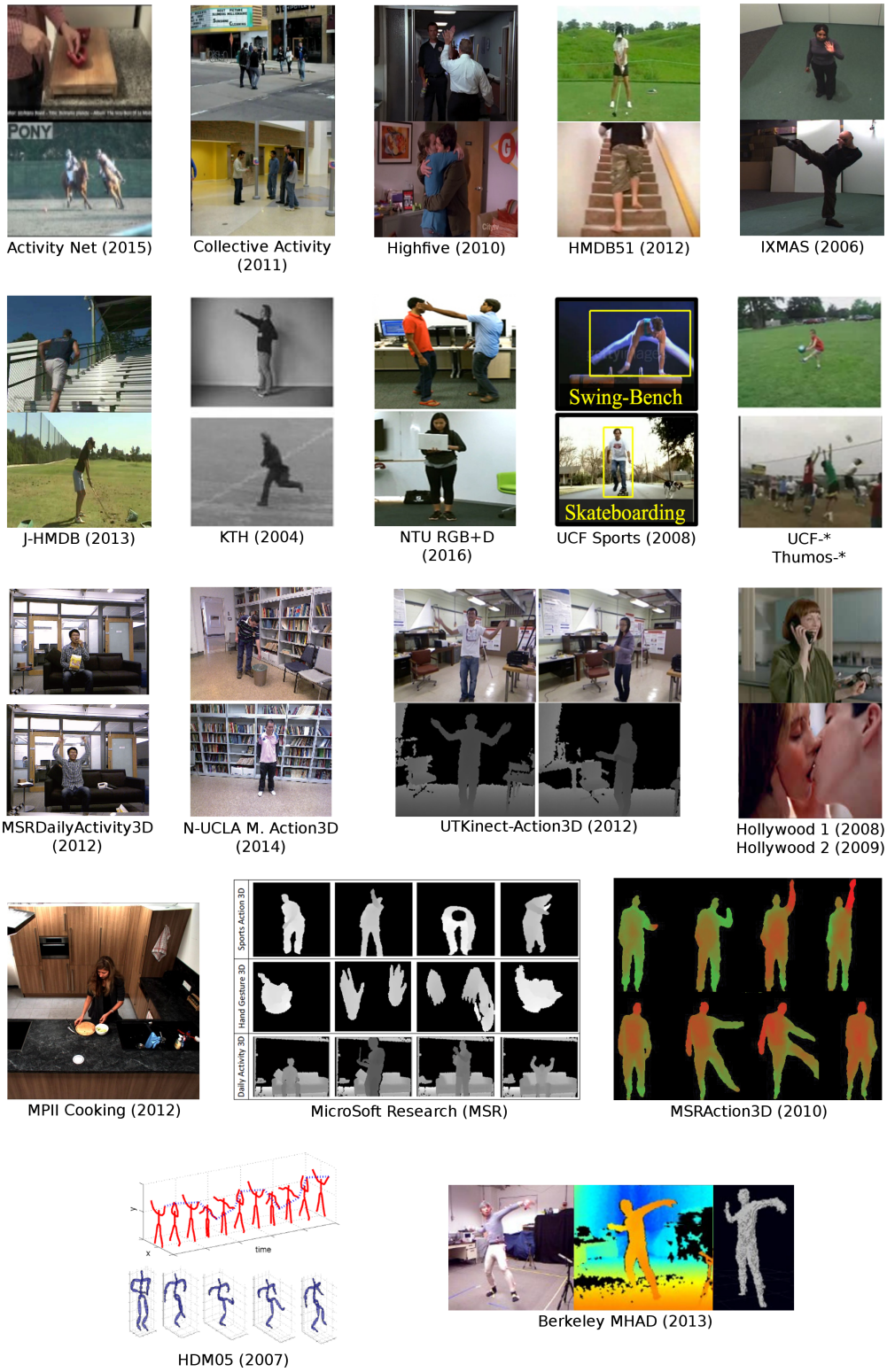


FIGURE 6.3: Sample frames from action datasets

TABLE 6.1: Action datasets

Year	Database	Problem	Body Parts	Modality	No.Classes	Performance
2004	KTH	AC	F	I	6	98.67% Acc [222]
2006	IXMAS	AC	F	RGB, A	13	98.79% Acc [174]
2007	HDM05	AC	F	S	100	98.17% Acc [19]
2008	HOHA (Hollywood 1)	AC, TL	F, U, L	RGB	8	71.90% Acc [149], 0.787@0.5 mAP [109]
2008	UCF Sports	AC, STL	F	RGB	10	95.80% Acc [154], 0.789@0.5 mAP [109]
2009	Hollywood 2	AC	F, U, L	RGB	12	78.50 mAP [96]
2009	UCF11 (YouTube Action)	AC, STL	F	RGB	11	93.77% Acc [129], -
2010	Highfive	AC, STL	F,U	RGB	4	69.40 mAP [184], 0.466 IoU [5]
2010	MSRAction3D	AC	F	D, S	20	97.30% Acc [102]
2010	MSRAction II	STL	F	RGB	3	85.00@0.125% mAP [21]
2010	Olympic Sports	AC	F	RGB	16	96.60% Acc [93]
2011	Collective Activity (Extended)	AC	F	RGB	6	90.23% Acc [2]
2011	HMDB51	AC	F, U, L	RGB	51	73.60% Acc [187]
2012	MPII Cooking	AC, TL	F, U	RGB	65	72.40 mAP [223], -
2012	MSRDailyActivity3D	AC	F,U	RGB, D, S	16	97.50% Acc [152]
2012	UCF101	AC,TL	F, U, L	RGB	101	94.20% Acc [190], 46.77@0.2 mAP (split 1) [199]
2012	UCF50	AC	F, U, L	RGB	50	97.90% Acc [40]
2012	UTKinect-Action3D	AC	F	RGB, D, S	10	98.80% Acc [79]
2013	J-HMDB	AC, STL	F, U, L	RGB, S	21	71.08 Acc [128], 73.1@0.5 mAP [149]
2013	Berkeley MHAD	AC	F	RGB, D, S, A	11	100.00% Acc [19]
2014	N-UCLA Multiview Action3D	AC	F	RGB, D, S	10	90.80% Acc [79]
2014	Sports 1-Million	AC	F, U, L	RGB	487	73.10% Acc [216]
2014	THUMOS-14	AC, TL	F, U, L	RGB	101, 20 *	71.60 mAP [72], 0.190@0.5 mAP [158]
2015	THUMOS-15	AC, TL	F, U, L	RGB	101, 20 *	80.80 mAP [93], 0.183@0.5 mAP (a)
2015	ActivityNet	AC, TL	F, U, L	RGB	200	93.23 mAP (b), 0.594@0.5 mAP [111]
2016	NTU RGB+D	AC	F	RGB, D, S, IR	60	{69.20, 77.70}[1] Acc [97]

In the Modality column: Depth, Skeleton, Audio, grayscale Intensity, InfraRed.
In Performance column: Accuracy, mean Average, Precision, Intersection over Union.
"*" indicates different number of classes is used for different problems. For TL/STL, "@" indicates amount overlap with groundtruth considered for positive localization. For instance, @0.5 indicates a 50% of overlap.
(a) Winner method on `http://activity-net.org/challenges/2016/program.html#leaderboard`.
(b) Winner method on `http://www.thumos.info/results.html`.
[1] {cross-subject accuracy, cross-view accuracy}.

TABLE 6.2: Benchmarking on UCF-101 dataset

Ref.	Year	Features	Architecture	Score
[44]	2016	ST-ResNet + iDT	2-stream ConvNet and ResNet	94.6%
[90]	2016	RNN Fisher Vector	C3D + VGG-CCA + iDT	94.1%
[177]	2016	Opt. Flow, RGB, iDT	LTC-CNN	92.7%
[214]	2016	conv5	2-Stream SR-CNN	92.6%
[46]	2016	conv5, 3D pool	VGG-16, VGG-M, 3D CNN	92.5%
[194]	2016	CNN	Siamese VGG-16	92.4%
[93]	2016	CNN fc7	2 CNNs (spatial + temporal)	92.2%
[188]	2016	3D CNN + RNN hierarchical local	Volumetric R-CNN (DANN)	91.6%
[189]	2015	CNN, Hog/Hof/Mbh	2-stream CNN	91.5%
[108]	2015	CNN feat	3D CNN	89.7%
[12]	2016	Dynamic feat maps	BVLC CaffeNet	89.1%
[72]	2015	H/H/M, iDT, FV+PCA+GMM	8-layer CNN	88.5%
[168]	2015	CNN	F_{ST}CN: 2 CNNs (spat + temp)	88.1%
[159]	2014	CNN	Two-stream CNN (CNN-M-2048)	88.0%
[106]	2016	eLSTM, DCNN fc7	eLSTM, DCNN+LSTM	86.9%
[218]	2016	CNN	2 CNNs (spatial + temporal)	86.4%
[212]	2016	dense trajectory, C3D	RNN, LSTM, 3DCNN	85.4%
[127]	2015	CNN fc6, HOG/HOF/MBH	VGG19 Conv5	79.52%±1.1% (tr2) 66.64% (tr1)
[78]	2014	CNN features	2 CNN converge to 2 fc layers	65.4%, 68% mAP
[74]	2015	ImageNet CNN, word2vec GMM	CNN	63.9%
[199]	2015	CNN	Spatial + motion CNN	54.28% mAP

TABLE 6.3: Benchmarking on THUMOS-14 dataset

Ref.	Year	Features	Architecture	Score
[72]	2015	H/H/M, IDT, FV+PCA+GMM.	8-layer CNN	71.6%
[218]	2016	CNN	2 CNNs (spatial + temporal)	61.5%
[74]	2015	ImageNet CNN, word2vec GMM	CNN	56.3%
[158]	2016	CNN fc6, fc7, fc8	3D CNN, Segment-CNN	19% mAP
[213]	2015	CNN fc7	VGG-16, 3-layer LSTM	17.1% mAP
[42]	2016	fc7 3D CNN	C3D CNN net	.084% mAP@50 .121% mAP@100 .139% mAP@200 .125% mAP@500

Looking at the top ranked deep models on the THUMOS 2014 dataset, almost all the winners in the 2015 challenge edition use different combinations of appearance and motion features. For the appearance ones, most of the methods extract frame-level CNN descriptors, and video representation is generated using a pooling method over the sequence. The motion-based features used by the top ranked methods can be divided into three groups, FlowNet, 3D CNN, and IDTs. [136] provides a comparison of methods showing 3D-CNN outperform other alternatives.

6.2 Deep-based action recognition

This section reviews deep methods for action (or activity) recognition according to the way they treat the temporal dimension.

6.2.1 2D Convolutional Neural Networks

In these kind of approaches, action recognition is often performed at frame-level and then somehow aggregated (averaging the class score predictions on individual frames). Some works further explore the possibility of using several frames as input. In particular, [78] studies the different alternatives for considering multiple frames

in a 2D model; however they concluded there was not a gain in performance using multiple video frames over averaging single frame predictions. Instead, [190] randomly samples video frames from K equal width temporal segments, obtain K class score predictions, compute the consensus scores, and use these in the loss function to learn from video representations directly, instead from one frame or one stack of frames. [217] convolves each frame of the video sequence to obtain frame-level CNN features. They then perform spatio-temporal pooling on pre-defined spatial regions over the set of randomly sampled frames (50-120 depending on the sequence) in order to construct a video-level representation, which is later l2-normalized and classified using SVM. [205] models scene, object, and more generic feature representations using separate convolutional streams. For each frame, the three obtained representations are averaged and input to a three-layer fully connected network which provides the final output. [12] collapses the videos into dynamic images, that can be fed into CNNs for image classification, by using rank pooling [48]. Dynamic images represent are simply the parameters of a ranking function that learned to order the video frames. [137] proposes a CNN, not to classify actions in depth data directly, but to model poses in a view-invariant high-dimensional space. For this purpose, they generate a synthetic dataset of 3D poses from motion capture data that are later fit with a puppet model and projected to depth maps. The network is first trained to differentiate among hundreds of poses to, then, use the features of the penultimate fully-connected layer for action classification in a non-deep action recognition approach. [119] exploits the combination of CNNs and LSTM for interactional object parsing on individual frames. Note LSTMs are not used for temporal sequence modeling but for refining object detections. For the action detection task, they then use object detections for pooling improved dense trajectories extracted on temporal segments.

Note that, independently from the discussed method, 2D convolutional filters in 2D CNNs only consider spatial inter-relations of pixels, ignoring their temporal neighborhood. Next we explore the more effective ways of exploiting spatiotemporal information in image sequences, which consist in either using pre-computed motion-based to include implicit temporal information in 2D CNNs or explicitly modeling temporal information with 3D CNNs or temporal sequence modeling methods.

6.2.2 Motion-based features

Researchers found that motion based features, such as optical flow, were a rich cue that could be fed directly as a network input. There are accurate and efficient methods to compute these kind of features, some of them by exploiting GPU capabilities [50]. The use of optical flow demonstrated to boost the performance of CNNs on action recognition-related tasks [159, 125, 218, 57].

[159] presents a two-stream CNN which incorporated both spatial (video frames) and temporal networks (pre-computed optical flow), and showed that the temporal networks trained on dense optical flow are able to obtain very good performance in spite of having limited training data. Along the same lines,[197] proposes a two-stream (spatial and temporal) net for non-action classification in temporal action localization. Similarly, [225] uses the same architecture for key-volume mining and classification in this case for spatio-temporal localization of actions. [24] extracts both appearance and motion deep features from body part detections instead of

whole video frames. They then compute for each body part the min/max aggregation their descriptors over time. The final representation consists of the concatenation of pooled body part descriptors on both appearance and motion cues, which is comparable to the size of a Fisher vector. [125] uses the magnitude of optical flow vectors as a multiplicative factor for the features from the last convolutional layer. This reinforces the attention of the network on the moving objects when fine-tuning the fully connected layers. [218] explores motion vectors (obtained from video compression) to replace dense optical flow. They adopted a knowledge transfer strategy from optical flow CNN to the motion vector CNN to compensate the lack of detail and noisiness of motion vectors.

[160] uses a multi-stream network to obtain frame-level features. To the full-frame spatial and motion streams from [159], they add two other actor-centered (spatial and motion) streams that compute the features in the actor's surrounding bounding box obtained by a human detector algorithm. Moreover, motion features are not stacks of optical flow maps between pairs of consecutive frames, but among a central frame and neighboring ones (avoiding object's displacement along the stacked flow maps). [57] and [199] propose a similar an approach for action localization. They first generate action region proposals from RGB frames using, respectively, selective search [175] on and EdgeBoxes [227]. Regions are then linked and described with static and motion CNN features. However, high quality proposals can be obtained from motion. [128] shows a region proposals generated by a region proposal network (RPN) [140] from motion (optical flow) were complementary to the ones generated by an appearance RPN.

Note some of the works in Section 6.2.3 use pre-computed motion features, which is not mutually exclusive with using motion features approaches. [177] uses stacks of 60 pre-computed optical flow maps as inputs for the 3D convolutions, largely improving results obtained using raw video frames. [193] computes motion-like image representations from depth data by accumulating absolute depth differences of contiguous frames, namely hierarchical depth motion maps (HDMM).

In the literature there exist several methods which extend the deep-based methods with the popular dense trajectory features. [189] introduces a video representation called Trajectory-pooled Deep-convolutional Descriptor (TDD), which consists on extending the state-of-the-art descriptors along the trajectories with deep descriptors pooled from normalized CNN feature maps. [127] proposes a method based on a concatenation of IDT feature (HOG, HOF, MBHx, MBHy descriptors) and Fisher vector encoding and CNN features (VGG19). For CNN features they use VGG19 CNN to capture appearance features and VLAD encoding to encore/pool convolutional feature maps. [138] utilizes dense trajectories, and hence motion-based features, in order to learn view-invariant representations of actions. In order to model this variance, they generate a synthetic dataset of actions with 3D puppets from MoCap data that are projected to multiple 2D viewpoints from which fisher vectors of dense trajectories are used for learning a CNN model. During its training, an output layer is placed with as many neurons as training sequences so fisher vectors from different 2D viewpoints give same response. Afterwards, the concatenation of responses in intermediate layers (except for last one) provide the view-invariant representation for actions.

Differently from other works, [116] jointly estimates optical flow and recognize actions in a multi-task learning setup. Their models consists in a residual network based on FlowNet [64] with extra additional classification layers, which learns to do both estimate optical flow and perform the classification task.

6.2.3 3D Convolutional Neural Networks

The early work of [76] introduces the novelty of inferring temporal information from raw RGB data directly by performing 3D convolutions on stacks of multiple adjacent video frames, namely 3D-CNN. Since then, many authors tried to either further improve this kind of models [173, 108, 168, 158, 133, 99] or used them in combination with other hybrid deep-oriented models [42, 7, 212, 46, 203, 93].

In particular, [173] proposes 3D convolutions with more modern deep architectures and fixed 3x3x3 convolution kernel size for all layers, that made 3D-CNN more suitable for large-scale video classification. In general, 3D-CNN can be expensive to train because of the large number of parameters, especially when training with bigger datasets such as 1-M sports dataset [78] (which can take up to one month). [168] factorizes the 3D convolutional kernel learning into a sequential process of learning 2D spatial convolutions in lower convolutional layers followed by learning 1D temporal convolutions in upper layers. [108] proposes initializing 3D convolutional weights using 2D convolutional weights from spatial CNN trained on ImageNET. This not only speeds up the training but also alleviates the overfitting problem on small datasets. [177] extends the length of input clips from 16 to 60 frames in order model more long-term temporal information during 3D convolutions, but reduced the input's spatial resolution to maintain the model complexity. [133] introduces a more compact 3D ConvNet for egocentric action recognition by applying 3D convolutions and 3D pooling only at the first layer. However, they do not use raw RGB frames, but stacked optical flow. In the context of depth data, [99] proposes re-scaling depth image sequences to a 3D cuboid and the use of 3D convolutions to extract spatio-temporal features. The network consists of two pairs of convolutional and 3D max-pooling followed by a two-layer fully-connected layer net.

3D convolutions are often used in more cumbersome hybrid deep-based approaches. [158] proposes a multi-stage CNN, in this case for temporal action localization, consisting of three 3D-CNN [173]: a proposal generation network that learns to differentiate background from action segments, a classification network that aims at discriminating among actions and serves as initialization for a third network, the localization network with a loss function that considers temporal overlap with the ground truth annotations. [193] applies 3D-CNN to action recognition from depth data. The authors train a separate 3D ConvNet for each Cartesian plane each of which fed with a stack of depth images constructed from different 3D rotations and temporal scales. [161] proves the combination of both 2D and 3D ConvNet can leverage the performance when performing egocentric action recognition. [93] uses 3D convolutions from [173] to model short-term action features on a hierarchical framework in which linear dynamic systems (LDS) and VLAD descriptors are used to, respectively, model/represent medium- and long-range dynamics.

6.2.4 Deep sequential models

The application of temporal sequence modeling techniques, such as LSTM, to action recognition showed promising results in the past [6, 59]. Earlier works did not try to explicitly model the temporal information, but aggregated the class predictions got from individual frame predictions. For instance [159] samples 25 equally spaced frames (and their crops and flips) from each video and then averages their predicted scores.

Today, we find the combination of recurrent networks, mostly LSTM, with CNN models for the task of action recognition.

[179] proposes a new gating scheme for LSTM that takes into account abrupt changes in the internal cell states, namely differential RNN. They use different order derivatives to model the potential saliency of observed motion patterns in actions sequences. [160] presents a bi-directional LSTM, which demonstrated to improve the simpler uni-directional LSTMs. [213] introduces a fully end-to-end approach on a RNN agent which interacts with a video over time. The agent observe a frame and provides a detection decision (confidence and begin-end), to whether or not emit a prediction, and where to look next. While back-propagation is used to train the detection decision outputs, REINFORCE is required to train the other two (non-differentiable) agent policies. [106] proposes a deep architecture which uses 3D skeleton sequences to regularize an LSTM network (LSTM+CNN) on the video. The regularization process is done by using the output of the encoder LSTM (grounded on 3D human-skeleton training data) and by modifying the standard BPTT algorithm in order to address the constraint optimization in the joint learning of LSTM+CNN. In their most recent work, [188] explores contexts as early as possible and leverage evolution of hierarchical local features. For this, they introduce a novel architecture called deep alternative neural network (DANN) stacking alternative layers, where each alternative layer consists of a volumetric convolutional layer followed by a recurrent layer. [90] introduces a novel Fisher Vector representation for sequences derived from RNNs. Features are extracted from input data via VGG/C3D CNN. Then a PCA/CCA dimension reduction and L_2 normalization are applied and sequential feature are extracted via RNN. Finally, another PCA+L_2-norm step is applied before the final classification. [97] extends the traditional LSTM into two concurrent domains, i.e, spatio-temporal long short-term memory (ST-LSTM). In this tree structure each joint of the network receive contextual information from both neighboring joints and previous frame. [153] proposes a part aware extension of LSTM for action recognition by splitting the memory cell of the LSTM into part-based sub-cells. These sub-cells can yield the models learn the long-term patterns specifically for each part. Finally, the output of each unit is the combination of all sub-cells.

6.2.5 Deep learning with fusion strategies

The goal of the fusion is to exploit information complementariness and redundancy to improve the recognition performance. The fusion may refer to the aggregation of information from different parts in a segmented video sequence, multiple cues or modalities, or the combinination with different deep-learning models trained with different data samples and/or learning parameters.

Some methods have used diverse fusion schemes to improve recognition performance of action recognition.

[159] in order to combine the class-level predictions of the two streams (spatial and temporal), trains a multi-class linear SVM on stacked L_2-normalized softmax scores, which showed to improve the class score averaging. [191], which improves the former work by making the networks deeper and improving the data augmentation, simply performs a linear combination of the prediction scores (2 for temporal net and 1 for the spatial net). [190] combines RGB, RGB difference, flow, and warped flow assigning equal weight to each channel. [46] fuses a spatial and temporal convnets at the last convolutional layer (after ReLU) to turn it into a spatio-temporal stream by using 3D Conv fusion followed by 3D pooling. The temporal stream is kept and the two loss functions are used for training and testing. [34] presents a deep neural-network-based hierarchical graphical model that recognizes individual

and group activity in surveillance scenes. Different CNNs produce action, pose, and scene scores. Then, the model refines the predicted labels for each activity via multi-step Message Passing Neural Network which captures the dependencies between action, poses, and scene predicted labels. [38] proposes an end-to-end hierarchical RNN for skeleton based action recognition. The skeleton is divided into five parts, each of which is feed into a different RNN network, the output of which are fused into higher-layer RNNs. The highest level representations are feed into a single-layer perceptron for the final decision. [161] faces the problem of first person action recognition using a multi-stream CNN (ego-CNN, temporal, and spatial), which are fused by combining weighted classifier scores. The proposed ego-CNN captures hand-crafted cues such as hand poses, head motion, and saliency map. [214] incorporates a region-of-interest pooling layer after the standard convolutional and pooling layers that separates CNN features for three semantic cues (scene, person, and objects) into parallel fully connected layers. They propose four different cue fusion schemes at class prediction level (max, sum, and two weighted fusions). [65] attempts to investigate human action recognition without the human presence in input video frames. They consider whether a background sequence alone can classify human actions. [128] performs action localization in space and time by linking via dynamic time warping the action bounding box detections on single frames. For bounding box classification, they concatenate the representations of multiple regions derived from the original detection bounding box. [44] proposes a two stream architecture (appearance and motion) based on residual networks. In order to model spatiotemporal information, they inject 4 residual connections (namely "skip-streams") from motion to the appearance stream (i.e. middle fusion) and also transform the dimensionality reduction layers from ResNet's original model to temporal convolution layers. [194] trains two Siamese networks modeling, respectively, action's precondition and effect on the observed environment. Each net learns a high-dimensional representation of either precondition or effect frames along with the linear transformation per class that transforms precondition to effect. The nets are connected via their outputs and not sharing weights; i.e. late fusion.

Chapter 7

Sequential residual learning for action classification

The success of deep architectures on image recognition tasks inspired many authors to tackle the problem from a similar perspective. A naive approach might use convolutional neural networks to classify individual video frames based on appearance and then average predictions [78]. However, this does not take into account the temporal structure of the video and could easily confuse "answering phone" and "hanging-up phone" actions. Introducing low level motion information can help, either by inputing clips (instead of single frames) to the network [78] or by exploiting pre-computed optical flow [159]. To more effectively handle the temporality in clips, 3D-CNN extend the convolutions in time [76, 177, 16]. Also, flow information demonstrated to complement appearance, especially if incorporated at the precise network stage [46]. Still, longer range dynamics are ignored.

Effectively modeling temporal dynamics and long-term frame dependencies is key, especially in longer and more complex videos. Frames can be processed by powerful deep convolutional networks and the evolution of their outputs be modeled by sequential learning methods, e.g. Long-Short Term Memory (LSTM) [69]. In its turn, LSTM layers can also be stacked to make the sequential models deeper [216]. However, deeper architectures might suffer from the "degradation problem" [64]. The skip connections from residual learning provide a solution to that while being parameter free. Not increasing the number of optimizable parameters is quite important in action recognition scenarios, in which the amount of training data tends to be limited to a few thousands of videos (10K in the most popular benchmarking datasets, UCF-101 and HMDB-51).

In this chapter, we extend multi-layer LSTMs with residual connections and explicitly learn *appearance* and *motion* into two separate streams. As input we use features extracted from a 3D-CNN, which provide a much richer representation than 2D and reduce redundancy observed by the LSTMs on top. The predictions of the two-stream residual LSTMs are late fused using element-wise dot aggregation as opposed of adding the final score predictions. Experimental results show that the proposed pipeline obtains state of-the-art results for RNN-like architectures on the challenging HMDB-51 dataset and achieves competitive results on UCF-101. To our extent, residual LSTM have not been applied in action recognition scenarios, but only on speech processing [80].

The chapter is organized as follows: Section 7.2 introduces the proposed two-stage pipeline and data generation; Section 7.1 briefly reviews deep sequential learning approaches that are closely related to our approach; and, finally, Section 7.3 covers the experimental protocol and discusses the results.

7.1 Related work

Averaging predictions from a few randomly sampled frames (or clips) may arise several problems. The sampled portions could be insufficient or irrelevant to the action class. [216] explores various temporal pooling layers in order to obtain a per-video global description – instead of per-clip. Their best strategy ("Conv Pooling") performed max-pooling over the last convolutional layer map responses of a 2DCNN across all the frames of the video. Another difficulty is that frames from different classes could present none or very subtle differences. In other words, some actions classes could share some subaction patterns; when classification is performed on frames/clips these are uniquely assigned to a particular class, but should be assigned to multiple action classes. Inspired by this idea, ActionVLAD [56] aggregates local spatiotemporal descriptors over the whole spatial and temporal extent of the video by softly assigning them to subaction anchors. The feature extractor, the anchors, and the classifier are jointly trained.

A more natural way to handle temporal dynamics is sequence modelling, such as recurrent neural networks. A preliminary work [7] processed each frame in a rather shallow 3D-CNN and modelled the output responses using a LSTM. The models were trained separately. More recently, [216] proposed an end-to-end trained architecture combining 2D CNN + 5-layer LSTM.

Given the success of LSTM on sequence modelling, many other out-of-the-box variations started to appear. Gated-recurrent Units (GRU) [26] are a very popular less-complex variation of LSTMs. GRUs consist of two gates (*update* and *reset* gates) instead of three and the internal state (*output state*) is fully exposed. Despite the reduced complexity, empirical studies found there is no significant difference in performance between the two [27]. The more recent Convolutional LSTMs [206] replace the fully connected layers in input-to-state and state-to-state transitions with convolutional layers. This allows to maintain the spatial structure of either input frames or convolutional maps from input to output. Also, substituting the fully connected layer with convolutions, reduces spatial data redundancy. [167] takes advantage of such models in first person interaction recognition. In their case, every pair of frames in the video is passed through two 2D CNN with shared weights, their outputs are then connected in a 3D convolutional layer, and passed to a ConvLSTM. [94] propose ConvALSTM, which combines the benefits of both ConvLSTM and Attention-LSTM [207, 155], but in contrast to ALSTMs, relies on a shallow convolutional network for the soft-attention mechanism. The L^2STM model [169] extends the original LSTM formulation to learn independent hidden state transitions of memory cells for individual spatial locations.

7.2 Method

In this section we present the proposed stacked residual network (depicted in Figure 7.1) and the data generation process. We also present our approach for model selection and evaluation.

7.2.1 The two-stage pipeline

Let $V \in \mathbb{R}^{m_x \times m_y \times l_v}$ be a video of duration l_v that represents an action and whose frames have size $m_x \times m_y$ pixels. Let $K_v = \lfloor \frac{l_v}{r} \rfloor - 1$, be the number of clips in V, where r is the stride between each clip. Let each clip $S_i \in \mathbb{R}^{m_x \times m_y \times s}$ have a duration s empirically defined so that S_i captures a gesture in V. A gesture is a movement

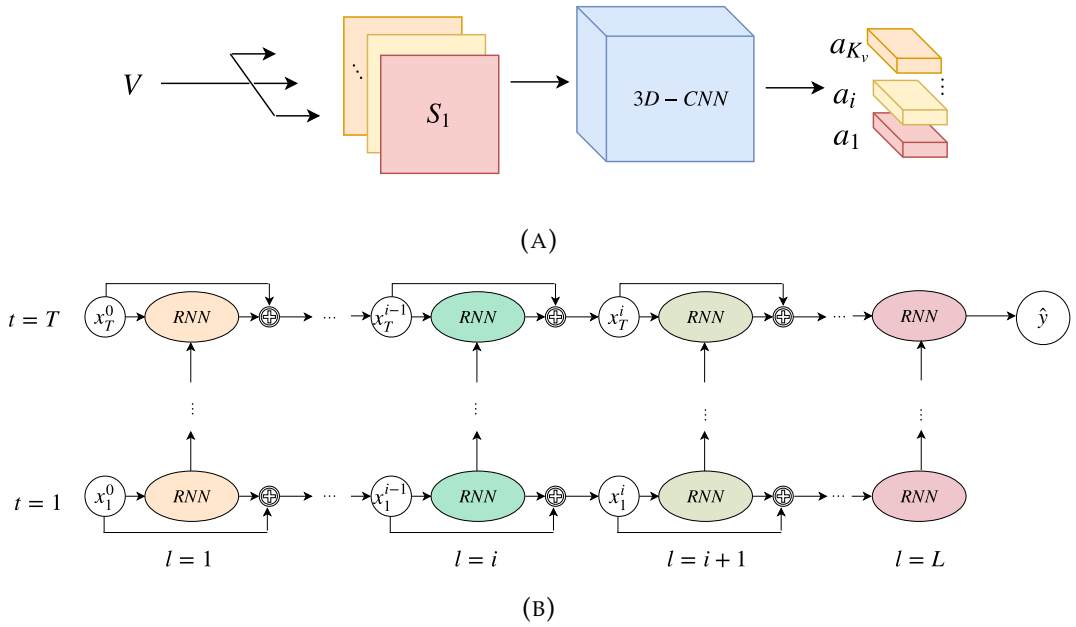


FIGURE 7.1: The proposed residual stacked RNN architecture. The input video, V, is divided into clips from which spatiotemporal features are extracted with a 3D CNN. The residual stacked RNN learns temporal dependencies between an action class and elements of a sampled subset of features, x^0, of duration T

of body parts that can be used to control and to manipulate, or to communicate. A temporal concatenation of gestures composes an action [14]. Unlike frame-level predictions [216], we aim to encapsulate each S_i with a latent representation of much lower dimension, $a_i \in \mathbb{R}^{d_f}$, which is expected to encode a gesture.

Let the spatiotemporal features of S_i be extracted[1] with a 3D CNN [190, 16], as $a_i = E(S_i) \in \mathbb{R}^{d_f}$, thus obtaining $a = \{a_i | i = 1 \dots K_v; a_i \in \mathbb{R}^{d_f}\}$, from which we extract a subset x^0 of size T: $x^0 = \{x_t^0 = a_{\sigma(t)} | t = 1, \dots, T; a_{\sigma(t)} \in a\}$, where $\sigma(t) = 1 + \lfloor (t-1)\frac{K_v - 1}{T-1} \rfloor$.

The final step learns the temporal dependencies between the input sequence x^0 using the residual recurrent neural network. Instead of fitting an underlying mapping $H(x)$, the same stack of layers in residual learning generates $H(x)$ to fit another mapping $F(x) = H(x) + x$. The network learns a residual mapping $F(x)$ to approximate $H(x)$ rather than $F(x)$. Given a stack of recurrent neural units of depth L, at layer l and timestep t the input is x_t^{l-1}. The previous memory and cell states for layer l from timestep $t-1$ are m_{t-1}^l and c_{t-1}^l, respectively. Then, m_t^l and c_t^l are calculated using the recurrent equations e.g. LSTM [69], and the input to layer $l+1$ at timestep t is $x_t^l = m_t^l + x_t^{l-1}$.

Each RNN layer in the residual RNN part has index $l \in \{1, \dots, L\}$. The dimension of the input at time t in layer l must be the same as the memory m_t^l since the addition in the residual equation is performed element-wise. The overall structure is the same as in [204] (see Figure 7.1b).

Let Θ^l be the parameter of the recurrent model at layer l, and L the total number of LSTM layers. If P is total number of action classes, $m_t^l, x_t^l \in \mathbb{R}^{d_f}$, $y \in \mathbb{R}^P$, and $W_y \in \mathbb{R}^{d_f \times P}$ is a fully connected layer, then the recurrent part of our hierarchical

[1] Note that our architecture is independent from the particular CNN structure and other models can be used as extractor, e.g. TSN [190] or I3D [16] (see Figure 7.1a).

residual RNN model is updated using:

$$\begin{aligned} c_t^l, m_t^l &= LSTM_l(c_{t-1}^l, m_{t-1}^l, x_t^{l-1}; \Theta^l) \\ x_t^l &= m_t^l + x_t^{l-1} \end{aligned} \tag{7.1}$$

where T is number of time steps. At the l-th layer we obtain the hidden state from $LSTM_l$ using the input x_t^{l-1}, the input at the $(l+1)$-th layer is the residual equation: $x_t^l = m_t^l + x_t^{l-1}$. We obtain the final score $\hat{y}$ through a softmax layer at the last time step T using

$$\hat{y} = \texttt{softmax}((m_T^L)^\top . W_y). \tag{7.2}$$

Under this formulation, our residual RNN model will learn the temporal dependencies of each clip S_i and perform a video level prediction by adding a softmax layer on top of the last LSTM unit at the last time step T (see Figure 7.1b).

7.2.2 Fusion

We extend the multi-layer residual LSTM to handle two streams, namely the RGB video, V, and the optical flow, which has been shown to be useful for CNN based models [46]. Girdhar [56] explored three fusion methods to combine RGB and Flow CNN models.

Given two feature vectors $u, v \in \mathbb{R}^m$, we consider for fusion the element-wise sum and the element-wise product. For the element-wise sum, $\oplus$, the fusion is $u \oplus v = (u_1 + v_1, \ldots, u_m + v_m)$. This method is expected to help if the two responses have high scores, whereas small perturbations in either vector will be ignored. The element-wise product, $\odot$, is $u \odot v = (u_1.v_1, \ldots, u_m.v_m)$. This scheme encourages the highest scores and tends to diminish small responses.

We feed the input video, V, to a pre-trained 3D-CNN, $x^c = \text{3D-CNN}^c(V)$, and the optical flow, V^f, through a pre-trained flow model, $x^f = \text{3D-CNN}^f(V^f)$. With *mid fusion*, $x^0 = \{x_t^0 = x_t^c \circ x_t^f | t = 1, \ldots, T\}$, where $\circ \in \{\oplus, \odot\}$. We then train a Res-LSTM model using x^0 as input. With *late fusion*, we use the input x^c (x^f) to train a Res-LSTM for the appearance (optical flow) network. Once the models are trained we obtain the softmax predictions, $\hat{y}^c$ and $\hat{y}^f$, for each modality network. The final prediction is $\hat{y} = \hat{y}^c \circ \hat{y}^f$, where $\circ \in \{\oplus, \odot\}$. Results of these fusion methods are shown in Table 7.2.

7.2.3 Data generation

We consider two strategies of data augmentation [169], depending on the CNN architecture used. The first strategy consists of fixing the spatial dimension and varying the temporal axis of the input stream,

$$V_S = \{V_{clip}^{(1)}, V_{clip}^{(2)}, \ldots, V_{clip}^{(v)} | V_{clip}^{(i)} \in \mathbb{R}^{m_x \times m_y \times l_t}\}, \tag{7.3}$$

where $l_t \leq l_v$ and we let r to be a fixed stride between each clip $V_{clip}^{(i)} \subset V$. The second strategy consists of sampling a fixed set of spatial crops with a temporal length of $T < l_v$. We select 10 crops from four corners and the center of each frame, along with their mirrors. We thus sample a new set $V_S \in \mathbb{R}^{10 \times m_x \times m_y \times T}$ for a given input video V. After a forward pass through the CNN, we obtain our spatiotemporal matrix of descriptors $\mathbf{A} \in \mathbb{R}^{10 \times T \times d_f}$, where d_f is the spatiotemporal feature dimension. Finally,

the input to the recurrent network is obtained as follows:

$$x^0 = \{x_t^0 \in \mathbb{R}^{d_f} | t = 1, \ldots, T\}, \tag{7.4}$$

where

$$x_t^0 = \frac{1}{K_c} \sum_{k=1,\ldots,K_c} \mathbf{A}(k, t, :), \tag{7.5}$$

where K_c is the number of crops (in our case $K_c = 10$). We found that the mean over the features of crops is preferable to just considering each crop separately.

7.3 Results

We will first introduce the datasets, evaluation measures, and implementation details. Then we will discuss the choice of key parameters of the proposed method and compare with both static (non-sequential) and sequential state-of-the-art approaches.

We carried out the experiments on HMDB-51 [85] and UCF-101 [165]. HMDB-51 consists of 6,849 videos of 51 categories, each category containing at least 101 instances. Actions can be categorized into 5 groups: facial actions, facial actions involving objects, body movements, body movements involving objects, and human interactions. UCF-101 provides 13,320 videos from 101 action categories, that can be categorized into: human-object interaction, body-motion only, human-human interaction, playing musical instruments, and sports. Results will be reported in terms of accuracy (%) over the split-1 on both datasets, as in [173].

For the parameter discussion in the next section we used as feature extractor a C3D pre-trained on Sports-1M [173]. Input clips were of temporal length $|S_i| = 16$, with a stride of $r = 8$ between them. For our final architecture features were extracted using the TSN [190] model, and the sparse features were of size 25. For training the residual recurrent network we used the RMSProp optimizer [171] with a learning rate of $\epsilon = 10^{-3}$.

7.3.1 Parameters and settings

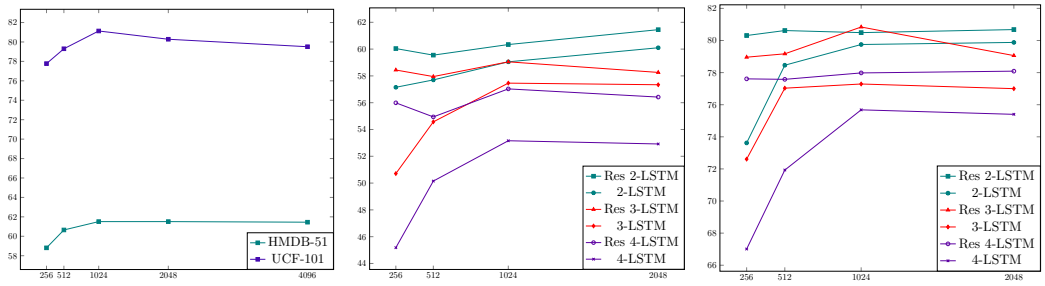
In this section we discuss the choice of the hidden layer size, the depth, the duration, and the fusion strategy for the deep residual network.

We varied the hidden layer size h, given the 4,096-dimensional spatiotemporal features extracted by the pre-trained C3D [173]. The best validation performance of the model increased before $h = 1024$ and stagnated after this value (Figure 7.2 (a)). We therefore used $h = 1024$ as a base parameter to our models.

A number of works [16, 61, 35] have shown that longer temporal window over the input of the 3D-CNN input leads to better performance. In Table 7.1, we can see that even for LSTM, longer time inputs lead to better performance. However, we may face the vanishing gradient problem using this class of model for very long sequence input. As for the temporal duration, $T = 25$ and $T = 35$[2] are the best choices for HMDB-51 and UCF-101, respectively. After these values the model starts overfitting.

To analyze the impact of the *residual connections* in the stacked recurrent neural networks context, we reduce the dimensionality of the 4,096 input features. The

[2]Note however, that the final scores we report are for $T = 25$ to allow for a fair comparison with the TSN model [190].



(A) Hidden layers dimension

(B) Residual depth on HMDB-51

(C) Residual depth on UCF-101

FIGURE 7.2: Influence of size and depth parameters on the performance of the LSTM

dimensionality of input and output need to match in order to perform the residual addition. To do so, we apply PCA over the initial feature of shape $4,096$ (extracted from a pre-trained C3D model) to fit with the dimension of the residual RNN. We select a set of dimensions $\mathcal{D} = \{\mathbb{R}^{d_m} | d_m \in [256, 512, 1024, 2048]\}$, and we train our hierarchical RNN, with and without residual part[3]. Figure 7.2 shows that the residual connections help generalization in both datasets: even when dropping on performance, the residual RNN still performs better.

We tested stacking 2, 3, and 4 recurrent layers (Figure 7.2(b)-(c)): stacking only two layers provides the best depth for the residual model as working with 3D-CNN reduces the number of feature samples per-video and thus a model with more layers is more likely to overfit the data. In contrast, for 2D-CNN feature extraction, the dataset is quite large because each frame is a feature and therefore the residual RNN model will have enough data to tune its parameters for more layers. This is why the authors in [216] were able to train 5-layers RNN.

TABLE 7.1: Impact of the value of the time step T on accuracy

T	5	15	25	35
HMDB-51	59.5	60.2	61.5	61.4
UCF-101	77.9	79.9	79.5	80.9

Finally, late fusion outperforms mid fusion on HMDB-51 using Res-LSTM. For point-wise addition the gain is near 6% and for point-wise product it is 13%, with a clear benefit of the product aggregation "$\odot$". We use the same weights as the original TSN [190], i.e. $(w_1, w_2) \in (1.5, 1)$ (see Table 7.2).

7.3.2 Final model evaluation

We evaluate our model on coarse UCF-101 categories as well as on complex action classes, following [169]. Table 7.3 shows that our model outperforms L^{2}STM in the coarse categories they reported and also in *Mixing Batter* with a gain of 4.41. However, the performance drops for the complex classes *Pizza Tossing* and *Salsa Spins*,

[3]We discarded the $4,096$ feature vector because of computational requirements.

TABLE 7.2: Impact on accuracy of different mid and late fusion strategies on the 2-layer Res-LSTM on the HMDB-51 dataset

Strategy	Mid fusion	Late fusion
$\oplus$ (element-wise sum)	59.3	65.2
$\odot$ (element-wise product)	56.5	68.0
$w_1.Flow + w_2.RGB$	–	63.3

probably because of the speed of the actions that our model was not able to capture well. Figure 7.3 shows the classification of some of the examples with the top confidences for each video example.

TABLE 7.3: Comparison on Split-1 of UCF-101 with complex movements

Data Types	L^{2}STM	Res-LSTM	Gain
Human-Object Interaction	86.7	88.2	$\uparrow$ **1.5**
Human-Human Interaction	95.4	96.9	$\uparrow$ **1.5**
Body-Motion Only	88.6	90.7	$\uparrow$ **2.1**
Playing Instrument	-	97.3	-
Sports	-	93.2	-
Pizza Tossing	72.7	66.7	$\downarrow$ 6.0
Mixing Batter	86.7	91.1	$\uparrow$ **4.4**
Playing Dhol	100	100	$\equiv$
Salsa Spins	100	97.7	$\downarrow$ 2.3
Ice Dancing	100	100	$\equiv$

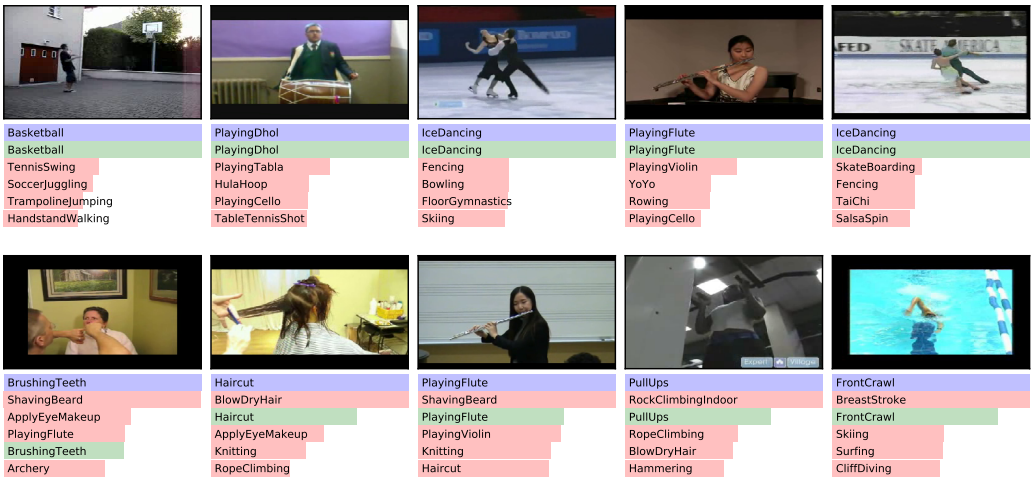
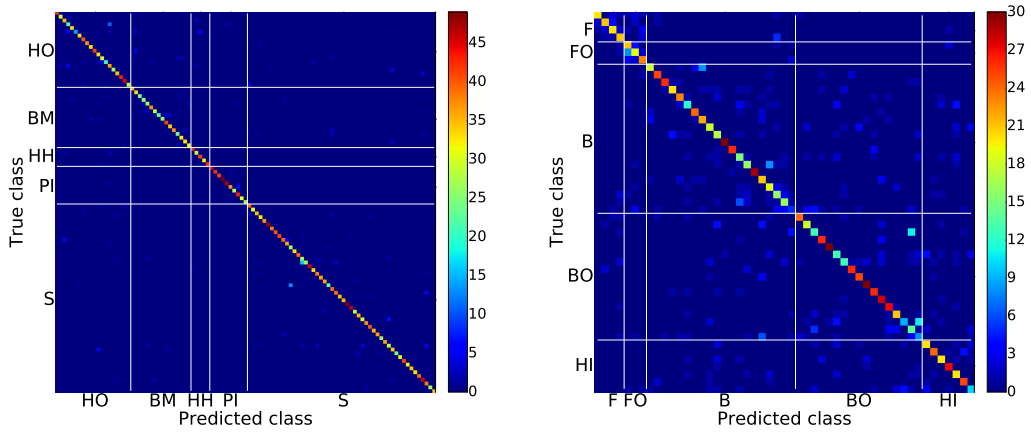


FIGURE 7.3: Sample video classification results showing the top-5 class predictions based on confidence. First row: correctly classified videos; second row: miss-classified videos. Key – blue bar: groundtruth class; green bar: correct class prediction; red bar: incorrect class prediction



(A) (B)

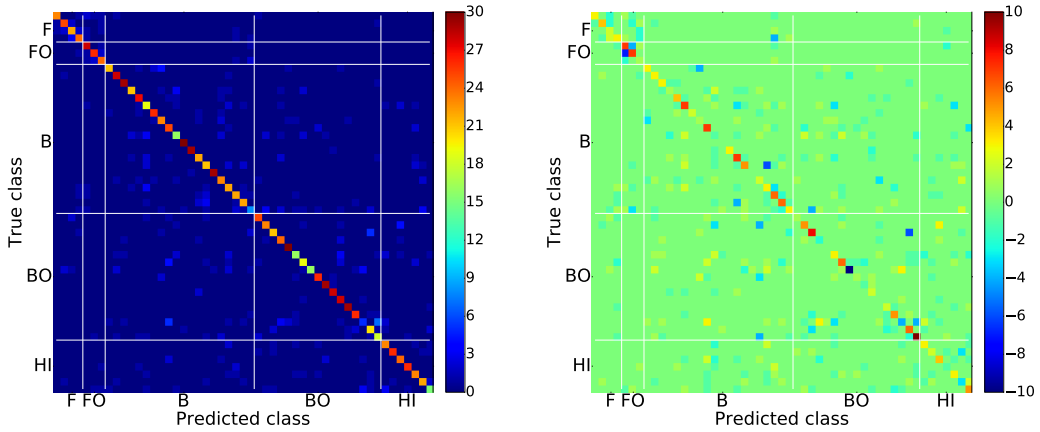
FIGURE 7.4: Confusion matrices with rearranged classes to group coarse categories. For UCF-101: human-object interaction (HO), body-motion only (BM), human-human interaction (HH), playing musical instrument (PI), and sports (S). For HMDB-51: facial actions (F), facial actions w/ object manipulation (FO), body movements (BM), body movements w/ object interaction (BO), and body movements for human interaction (HH). (a) Res-LSTM confusion matrix on UCF-101, (b) Res-LSTM confusion matrix on HMDB-51

Also, we compare our final model to other RNN-like architectures. Table 7.4 lists the performances and pre-training used by each solution. Our Res-LSTM outperforms the LSTM solutions in HMDB-51, while still being close to L^{2}STM [169] and Pre-RNN [210] performances on UCF-101.

In addition, Figure 7.4 reports the confusion matrices of our model on UCF-101 and HMDB-51. The classes in both datasets are rearranged using the coarse category labels provided in each dataset.

We also combined our method with IDT, following other state-of-the-art approaches [177, 56, 94]. From the combination, we obtained an improvement of accuracy of, respectively, +0.5% and +8.9% in UCF-101 and HMDB-51. In order to analyze the larger improvement in HMDB-51, we illustrate the confusion matrix for the combination in Figure 7.5a and the subtraction of the confusion matrices before and after the combination in Figure 7.5b. Finally, Figure 7.6 illustrates the per-class accuracy improvement on the HMDB-51 categories after the combination with IDT which improved (or maintained) the accuracy on 45 of the 51 categories, while only getting worse performance on 7.

Finally, we compare to a broader set of works, either sequential or non-sequential (static) models in Table 7.5. Most of them only report results over the three splits in both UCF-101 and HMDB-51. Those that provide the accuracy in Split-1 are marked with "*". We can see sequential-model based solutions are about 5 to 10% less accurate than the most successful static models pre-trained on very large datasets. However, our approach came quite close to those preformances obtained by static models. In the future, we will investigate how our model can be trained end-to-end in a larger dataset such as the one from [16] and, eventually, see if we are able to further improve the results.



(A) (B)

FIGURE 7.5: Confusion matrices after combining our Res-LSTM with IDT on HMDB-51. The ordering of classes in (a) and (b) is rearranged as in Figure 7.5a. (a) Res-LSTM ⊙ IDT confusion matrix on HMDB-51, (b) Subtraction of Res-LSTM ⊙ IDT and Res-LSTM confusion matrices on HMDB-51.

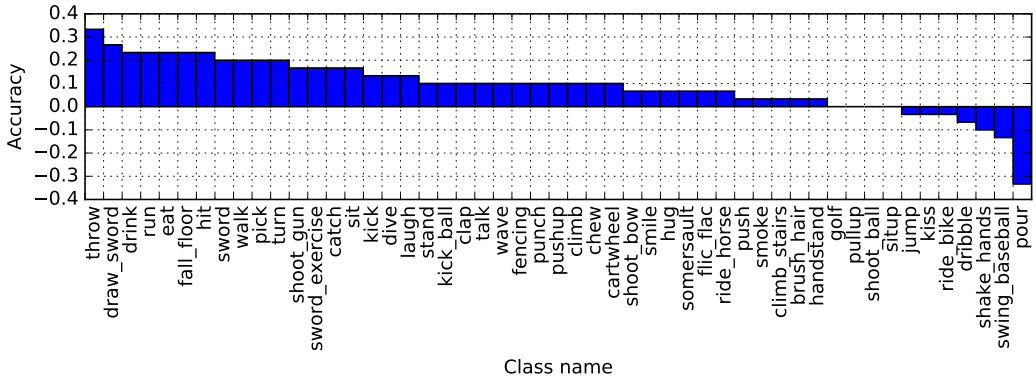


FIGURE 7.6: Per-class accuracy improvement after combining our Res-LSTM with IDT on HMDB-51.

TABLE 7.4: Performance comparison of RNN-like architectures. UCF-101 accuracies are over split-1, except for [169] that only reports accuracy over the three splits. "*" indicates that the method may or not use a pre-trained model, depending on the CNN used

Method	Pre-training		UCF-101	HMDB-51
	ImageNet	1M Sports		
TwoLSTM [216]	3	3	88.3	-
VideoLSTM [94]	3	✗	89.2	56.4
L^2STM [169]	3	✗	**93.6**	66.2
Pre-RNN [210]	3	✗	**93.7**	-
Res-LSTM	*	*	92.5	**68.0**

TABLE 7.5: Comparison on UCF-101 and HMDB-51

Model	Method	UCF-101	HMDB-51
Static models	FST-CNN [168]	88.1	59.1
	TDD [189]	90.3	63.2
	KV-CNN [225]	93.1	63.3
	LTC [177]	91.7	64.8
	TDD + IDT [189]	91.5	65.9
	ST-ResNet [44]	93.4	66.4
	STM-ResNet [45]	94.2	68.2
	LTC + IDT [177]	92.7	67.2
	TSN [190]	94.2	69.4
	ST-ResNet + IDT [44]	94.6	70.3
	STM-ResNet + IDT [45]	94.9	72.2
	STC-ResNext [35]	95.8‡	72.6‡
	I3D [16]	**98.0**	**80.7**
Sequential models	LRCN [37]	82.9	-
	AttLSTM [156]	77.0*	41.3
	UnsuperLSTM [166]	84.3*	44.0‡
	RLSTM-g3 [106]	86.9	55.3
	TwoLSTM [216]	88.3*	-
	VideoLSTM [94]	89.2*	56.4
	VideoLSTM + IDT [94]	91.5*	63.0
	L^2STM [169]	**93.6**	66.2
	PreRNN [210]	**93.7***	–
	Res-LSTM (ours)	92.5*	68.0*
	Res-LSTM (ours) $\odot$ IDT	93.0*	**76.9***

‡ Only RGB modality is used
* Evaluation on split-1

Chapter 8

Final discussion and conclusion

We have seen two very different approaches to action recognition: hand-crafted and deep-learning based methods. Hand-crated approaches dominated the field for a long time, especially the ones based on local spatiotemporal feature representations. These kind of features, often aggregated using a Bag-of-Visual-Words approach, demonstrated their robustness and reliability for the task. A particular instantiation of Bag-of-Visual-Words (BoVW) has been the dense trajectories (DT) framework. The feature extraction step of DTs was inspired by regular dense sampling strategies, but merely focusing on moving parts: pixels are tracked during 15 frames at different spatial scales using dense optical flow maps. The sampled trajectories are then described using a diverse set of feature descriptors: trajectory displacements, HOG, HOF, and MBH, respectively providing trajectory shape, appearance, motion, and motion boundary information about the actions observed. After this feature extraction step, codebook generation and feature encoding from BoVW, provide a global representation for the whole video based on each of those four different kinds of descriptors.

The combination of several sources of information proved to be very effective for action recognition. One way of combining them is simply concatenating the different video descriptors and using them in a holistic classifier. However, the combination not only enlarges the feature space but can also reduce the class separability of the classification examples because of the different nature of the individual representations, eventually confusing the classifier. Ensemble learning provides a solution for that. We showed that having one classifier for each type of feature (or any reduced subset of features) and then combining their outputs can be more effective than holistic action representations. Each classifier specializes in a kind of features and their outputs combined via DS theory-based fusion. In particular, this kind of fusion not only takes into account the scores of the different classifiers, but also uses those scores as evidence to measure how confident they are in the classification so the contribution to the final decision is adapted. As shown, dense trajectories and space-time interest points can also both benefit from such approach.

We explored other modalities than RGB. In particular, depth and inertial information. On the one hand, depth provides geometric shape information that can also be relevant for action recognition. In the monitoring scenario we presented, the main challenge was not large intra-class variability but very low inter-class variability. We showed how geometric information can be integrated in the dense trajectories framework to enhance recognition. On the other hand, inertial information also shown to be a highly complementary cue to vision-based recognition, providing very precise information of hand movements and helping to discriminate very look-a-like actions.

Given the success of dense trajectory framework in all sorts of action recognition problems, we decided to overcome on of their drawbacks. Their inability to model long-term temporal information. For that, we proposed modeling the evolution of iDT features on groupings of trajectories. The groupings were obtained by using a recursive clustering algorithm that performed a hierarchical decomposition of the video's cloud of trajectories and build a tree representation. Then, we modeled the evolution of features throughout both nodes and branches of trees. For the final classification, we constructed a kernel representation combining the two proposed representations. Moreover, our method shows further improvement when used together with holistic videodarwin. We achieved better results than current state-of-the-art on two benchmark datasets (UCF Sports Actions and Highfive) for action recognition. The pipeline is applicable to any pattern recognition problem once a rich representation is obtained at a given time instant. Also, following the trend of combining hand-crafted and deep-learning methods, we could explore the integration of CNN features as an additional cue in darwintrees.

Deep-learning based has become state-of-the-art for action recognition. We introduced a comprehensive state of the art for action recognition and the very related gesture recognition problem. We categorized the approaches in a taxonomy of deep models: 2D-CNN, motion-based CNNs, 3D-CNN, and temporal models. In it, we showed the importance of pre-computed motion cues and the promising results of 3D-CNN over 2D-CNN given their better ability to model spatiotemporal features in videos. Moreover, current trends show how 3D-CNN are able to model the temporal information in action recognition videos and perform better than temporal modeling based ones. However, the former could challenged with longer and semantically richer complex videos, requiring more and more data and computational time to perform better than the latter.

Finally, we defined a two-stream multi-layered residual recurrent neural network that incorporated residual connections between layers. The streams processed RGB and motion independently and then late-fused the class score predictions using element-wise multiplication. Our solution obtained state-of-the-art against other LSTM solutions on the HMDB51 dataset and competitive results in UCF101.

As future work, we are intended to explore better ways of combining 3D-CNNs with sequential modeling based ones, so we can have increased performance and the ability to perform better than purely 3D-CNN models in longer videos while keeping low the computational cost of the approach.

Bibliography

[1] Luís A Alexandre. "3D descriptors for object and category recognition: a comparative evaluation". In: *Workshop on Color-Depth Camera Fusion in Robotics at the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Vilamoura, Portugal*. Vol. 1. 2. IEEE. 2012.

[2] Mohamed R Amer et al. "Montecarlo tree search for scheduling activity recognition". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 1353–1360.

[3] Relja Arandjelović and Andrew Zisserman. "Three things everyone should know to improve object retrieval". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 2911–2918.

[4] Maryam Asadi-Aghbolaghi et al. "Deep learning for action and gesture recognition in image sequences: A survey". In: *Gesture Recognition*. Springer, 2017, pp. 539–578.

[5] Konstantinos Avgerinakis et al. "Moving camera human activity localization and recognition with motionplanes and multiple homographies". In: *ICIP*. IEEE. 2015, pp. 2085–2089.

[6] Moez Baccouche et al. "Action classification in soccer videos with long short-term memory recurrent neural networks". In: *International Conference on Artificial Neural Networks*. Springer. 2010, pp. 154–159.

[7] Moez Baccouche et al. "Sequential deep learning for human action recognition". In: *International Workshop on Human Behavior Understanding*. Springer. 2011, pp. 29–39.

[8] Taposh Banerjee et al. "Day or night activity recognition from video using fuzzy clustering techniques". In: *Fuzzy Systems, IEEE Transactions on* 22.3 (2014), pp. 483–493.

[9] Miguel Angel Bautista et al. "Probability-based dynamic time warping for gesture recognition on RGB-D data". In: *Advances in Depth Image Analysis and Applications*. Springer, 2013, pp. 126–135.

[10] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. "Surf: Speeded up robust features". In: *European conference on computer vision*. Springer. 2006, pp. 404–417.

[11] Asma Ben Hadj Mohamed et al. "Assisting people with disabilities through Kinect sensors into a smart house". In: *Computer Medical Applications (ICCMA), 2013 International Conference on*. IEEE. 2013, pp. 1–5.

[12] Hakan Bilen et al. "Dynamic image networks for action recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3034–3042.

[13] Yoottana Booranrom, Bunthit Watanapa, and Pornchai Mongkolnam. "Smart bedroom for elderly using kinect". In: *Computer Science and Engineering Conference (ICSEC), 2014 International*. IEEE. 2014, pp. 427–432.

[14] Paulo Vinicius Koerich Borges, Nicola Conci, and Andrea Cavallaro. "Video-based human behavior understanding: A survey". In: *IEEE transactions on circuits and systems for video technology* 23.11 (2013), pp. 1993–2008.

[15] GJ Burghouts et al. "Action recognition by layout, selective sampling and soft-assignment". In: (2013).

[16] Joao Carreira and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset". In: *Computer Vision and Pattern Recognition (CVPR), IEEE Conference on*. IEEE. 2017, pp. 4724–4733.

[17] Pierluigi Casale. "Approximate ensemble methods for physical activity recognition applications". In: *ELCVIA: electronic letters on computer vision and image analysis* 13.2 (2014), pp. 22–23.

[18] Shih-Fu Chang et al. "Large-scale multimodal semantic concept detection for consumer video". In: *Proceedings of the international workshop on Workshop on multimedia information retrieval*. ACM. 2007, pp. 255–264.

[19] Rizwan Chaudhry et al. "Bio-inspired dynamic 3d discriminative skeletal features for human action recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2013, pp. 471–478.

[20] Chenyi Chen et al. "Deepdriving: Learning affordance for direct perception in autonomous driving". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2722–2730.

[21] Wei Chen and Jason J Corso. "Action detection by implicit intentional motion clustering". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3298–3306.

[22] Wei Chen, Ran Xu, and J Corso. "Action bank for large-scale action classification". In: *THUMOS: ICCV Workshop on Action Recognition with a Large Number of Classes*. Vol. 7. 2013.

[23] Guangchun Cheng et al. "Advances in Human Action Recognition: A Survey". In: *CoRR* abs/1501.05964 (2015).

[24] Guilhem Chéron, Ivan Laptev, and Cordelia Schmid. "P-cnn: Pose-based cnn features for action recognition". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 3218–3226.

[25] Hyunjong Cho, Hyungtae Lee, and Zhuolin Jiang. "Evaluation of LC-KSVD on UCF101 action dataset". In: *THUMOS: ICCV Workshop on Action Recognition with a Large Number of Classes*. Vol. 7. 2013.

[26] KyungHyun Cho et al. "On the Properties of Neural Machine Translation: Encoder-Decoder Approaches". In: *CoRR* abs/1409.1259 (2014).

[27] Junyoung Chung et al. "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555 (2014).

[28] Carlos Fernando Crispim et al. "Evaluation of a monitoring system for event recognition of older people". In: *Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on*. IEEE. 2013, pp. 165–170.

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

[30] James W Davis and Aaron F Bobick. "The representation and recognition of human movement using temporal templates". In: *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*. IEEE. 1997, pp. 928–934.

[31] Benoît Delachaux et al. "Indoor activity recognition by combining one-vs.-all neural network classifiers exploiting wearable and depth sensors". In: *Advances in Computational Intelligence*. Springer, 2013, pp. 216–223.

[32] Pierangelo Dell'Acqua et al. "An assistive tool for monitoring physical activities in older adults". In: *Serious Games and Applications for Health (SeGAH), 2013 IEEE 2nd International Conference on*. 2013, pp. 1–6.

[33] Arthur P Dempster. "Upper and lower probabilities induced by a multivalued mapping". In: *Classic Works of the Dempster-Shafer Theory of Belief Functions*. Springer, 2008, pp. 57–72.

[34] Zhiwei Deng et al. "Deep Structured Models For Group Activity Recognition". In: *Proceedings of the British Machine Vision Conference (BMVC)*. BMVA Press, 2015, pp. 179.1–179.12.

[35] Ali Diba et al. "Spatio-Temporal Channel Correlation Networks for Action Classification". In: *CoRR* abs/1806.07754 (2018).

[36] Piotr Dollár et al. "Behavior recognition via sparse spatio-temporal features". In: *Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on*. IEEE. 2005, pp. 65–72.

[37] Jeffrey Donahue et al. "Long-term recurrent convolutional networks for visual recognition and description". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 2625–2634.

[38] Yong Du, Wei Wang, and Liang Wang. "Hierarchical recurrent neural network for skeleton based action recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1110–1118.

[39] Amandine Dubois and François Charpillet. "Human activities recognition with RGB-Depth camera using HMM". In: *Engineering in Medicine and Biology Society (EMBC), 2013 35th Annual International Conference of the IEEE*. IEEE. 2013, pp. 4666–4669.

[40] Ionut C Duta et al. "Spatio-Temporal VLAD Encoding for Human Action Recognition in Videos". In: *International Conference on Multimedia Modeling*. Springer. 2017, pp. 365–378.

[41] Jeffrey L Elman. "Finding structure in time". In: *Cognitive science* 14.2 (1990), pp. 179–211.

[42] Victor Escorcia et al. "Daps: Deep action proposals for action understanding". In: *European Conference on Computer Vision*. Springer. 2016, pp. 768–784.

[43] Gunnar Farnebäck. "Two-frame motion estimation based on polynomial expansion". In: *Image analysis*. Springer, 2003, pp. 363–370.

[44] Christoph Feichtenhofer, Axel Pinz, and Richard Wildes. "Spatiotemporal Residual Networks for Video Action Recognition". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3468–3476.

[45] Christoph Feichtenhofer, Axel Pinz, and Richard P Wildes. "Spatiotemporal multiplier networks for video action recognition". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. 2017, pp. 7445–7454.

[46] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. "Convolutional two-stream network fusion for video action recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1933–1941.

[47] Basura Fernando et al. "Modeling video evolution for action recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5378–5387.

[48] Basura Fernando et al. "Rank pooling for action recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 39.4 (2017), pp. 773–787.

[49] Martin A Fischler and Robert C Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395.

[50] Denis Fortun, Patrick Bouthemy, and Charles Kervrann. "Optical flow modeling and computation: a survey". In: *Computer Vision and Image Understanding* 134 (2015), pp. 1–21.

[51] Charless Fowlkes et al. "Spectral grouping using the Nystrom method". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 26.2 (2004), pp. 214–225.

[52] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. "Activity representation with motion hierarchies". In: *International Journal of Computer Vision* 107.3 (2014), pp. 219–238.

[53] Adrien Gaidon, Zaid Harchaoui, and Cordelia Schmid. "Temporal localization of actions with actoms". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.11 (2013), pp. 2782–2795.

[54] JC Gemert et al. "APT: Action localization proposals from dense trajectories". In: *Proceedings of the British Machine Vision Conference*. BMVA Press. 2015, pp. 177.1–177.12.

[55] Felix A Gers, Nicol N Schraudolph, and Jürgen Schmidhuber. "Learning precise timing with LSTM recurrent networks". In: *Journal of machine learning research* 3.Aug (2002), pp. 115–143.

[56] R. Girdhar et al. "ActionVLAD: Learning Spatio-Temporal Aggregation for Action Classification". In: *Computer Vision and Pattern Recognition, IEEE Conference on*. 2017, pp. 3165–3174.

[57] Georgia Gkioxari and Jitendra Malik. "Finding action tubes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 759–768.

[58] Christopher Golby et al. "A low cost 'activities of daily living'assessment system for the continual assessment of poststroke patients, from inpatient/outpatient rehabilitation through to telerehabilitation". In: *Successes and Failures in Telehealth* (2011), pp. 1–8.

[59] Alexander Grushin et al. "Robust human action recognition via long short-term memory". In: *Neural Networks (IJCNN), International Joint Conference on*. IEEE. 2013, pp. 1–8.

[60] Hatice Gunes and Massimo Piccardi. "Affect recognition from face and body: early fusion vs. late fusion". In: *Systems, Man and Cybernetics, 2005 IEEE International Conference on*. Vol. 4. IEEE. 2005, pp. 3437–3443.

[61] Kensho Hara, Hirokatsu Kataoka, and Yutaka Satoh. "Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, USA*. 2018, pp. 18–22.

[62] Chris Harris and Mike Stephens. "A combined corner and edge detector." In: *Alvey vision conference*. Vol. 15. 50. BMVA. 1988, pp. 10–5244.

[63] Mohammad Havaei et al. "Brain tumor segmentation with deep neural networks". In: *Medical image analysis* 35 (2017), pp. 18–31.

[64] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[65] Yun He et al. "Human Action Recognition Without Human". In: *ECCV Workshops (3)*. Vol. 9915. Lecture Notes in Computer Science. 2016, pp. 11–17.

[66] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. "Going deeper into action recognition: A survey". In: *Image and vision computing* 60 (2017), pp. 4–21.

[67] Antonio Hernández-Vela et al. "BoVDW: Bag-of-Visual-and-Depth-Words for gesture recognition". In: *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE. 2012, pp. 449–452.

[68] Tin Kam Ho. "The random subspace method for constructing decision forests". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 20.8 (1998), pp. 832–844.

[69] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[70] Hossein Mousavi Hondori, Maryam Khademi, and Cristina V Lopes. "Monitoring intake gestures using sensor fusion (microsoft kinect and inertial sensors) for smart home tele-rehab setting". In: *2012 1st Annual IEEE Healthcare Innovation Conference*. 2012.

[71] Manan Jain et al. "Action localization with tubelets from motion". In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE. 2014, pp. 740–747.

[72] Mihir Jain, Jan C van Gemert, and Cees GM Snoek. "What do 15,000 object categories tell us about classifying and localizing actions?" In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 46–55.

[73] Mihir Jain, Herve Jegou, and Patrick Bouthemy. "Better exploiting motion for better action recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2013, pp. 2555–2562.

[74] Mihir Jain et al. "Objects2action: Classifying and localizing actions without any video example". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 4588–4596.

[75] Hueihan Jhuang et al. "Towards understanding action recognition". In: *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3192–3199.

[76] Shuiwang Ji et al. "3D convolutional neural networks for human action recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2013), pp. 221–231.

[77] Svebor Karaman et al. "Adaptive Structured Pooling for Action Recognition". In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2014, pp. 112.1–112.12.

[78] Andrej Karpathy et al. "Large-scale video classification with convolutional neural networks". In: *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*. IEEE. 2014, pp. 1725–1732.

[79] Tommi Kerola, Nakamasa Inoue, and Koichi Shinoda. "Cross-view human action recognition from depth maps using spectral graph sequences". In: *Computer Vision and Image Understanding* 154 (2017), pp. 108–126.

[80] Jaeyoung Kim, Mostafa El-Khamy, and Jungwon Lee. "Residual LSTM: Design of a Deep Recurrent Architecture for Distant Speech Recognition". In: *CoRR* abs/1701.03360 (2017).

[81] Jihyoung Kim, Sungwon Yang, and Mario Gerla. "StrokeTrack: wireless inertial motion tracking of human arms for stroke telerehabilitation". In: *Proceedings of the First ACM Workshop on Mobile Systems, Applications, and Services for Healthcare*. ACM. 2011, p. 4.

[82] Josef Kittler et al. "On combining classifiers". In: *IEEE transactions on pattern analysis and machine intelligence* 20.3 (1998), pp. 226–239.

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

[84] Hilde Kuehne, Juergen Gall, and Thomas Serre. "An end-to-end generative framework for video segmentation and recognition". In: *Applications of Computer Vision (WACV), IEEE Winter Conference on*. IEEE. 2016, pp. 1–8.

[85] Hildegard Kuehne et al. "HMDB: a large video database for human motion recognition". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. IEEE. 2011, pp. 2556–2563.

[86] Ludmila I Kuncheva. *Combining pattern classifiers: methods and algorithms*. John Wiley & Sons, 2004.

[87] Yasuo Kuniyoshi, Hirochika Inoue, and Masayuki Inaba. "Design and implementation of a system that generates assembly programs from visual recognition of human action sequences". In: *Intelligent Robots and Systems' 90.'Towards a New Frontier of Applications', Proceedings. IROS'90. IEEE International Workshop on*. IEEE. 1990, pp. 567–574.

[88] Ivan Laptev. "On space-time interest points". In: *International Journal of Computer Vision* 64.2-3 (2005), pp. 107–123.

[89] Ivan Laptev and Tony Lindeberg. "Interest point detection and scale selection in space-time". In: *International Conference on Scale-Space Theories in Computer Vision*. Springer. 2003, pp. 372–387.

[90] Guy Lev et al. "RNN fisher vectors for action recognition and image annotation". In: *European Conference on Computer Vision*. Springer. 2016, pp. 833–850.

[91] Peixia Li et al. "Deep visual tracking: Review and experimental comparison". In: *Pattern Recognition* 76 (2018), pp. 323–338.

[92] Wanqing Li, Zhengyou Zhang, and Zicheng Liu. "Action recognition based on a bag of 3d points". In: *Computer Vision and Pattern Recognition Workshops (CVPRW), 2010 IEEE Computer Society Conference on*. IEEE. 2010, pp. 9–14.

[93] Yingwei Li et al. "VLAD3: Encoding Dynamics of Deep Features for Action Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016.

[94] Zhenyang Li et al. "VideoLSTM convolves, attends and flows for action recognition". In: *Computer Vision and Image Understanding* 166 (2018), pp. 41–50.

[95] Bin Liang and Lihong Zheng. "Spatio-temporal pyramid cuboid matching for action recognition using depth maps". In: *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE. 2015, pp. 2070–2074.

[96] Anan Liu et al. "Hierarchical Clustering Multi-Task Learning for Joint Human Action Grouping and Recognition." In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.1 (2017), pp. 102–114.

[97] Jun Liu et al. "Spatio-temporal lstm with trust gates for 3d human action recognition". In: *European Conference on Computer Vision*. Springer. 2016, pp. 816–833.

[98] Kui Liu et al. "Fusion of inertial and depth sensor data for robust hand gesture recognition". In: *Sensors Journal, IEEE* 14.6 (2014), pp. 1898–1903.

[99] Zhi Liu, Chenyang Zhang, and Yingli Tian. "3d-based deep convolutional neural network for action recognition with depth sequences". In: *Image and Vision Computing* 55 (2016), pp. 93–100.

[100] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International Journal of Computer Vision* 60.2 (2004), pp. 91–110.

[101] Henk J Luinge and Peter H Veltink. "Measuring orientation of human body segments using miniature gyroscopes and accelerometers". In: *Medical and Biological Engineering and computing* 43.2 (2005), pp. 273–282.

[102] Jiajia Luo, Wei Wang, and Hairong Qi. "Group sparsity and geometry constrained dictionary learning for action recognition from depth maps". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 1809–1816.

[103] Shugao Ma, Leonid Sigal, and Stan Sclaroff. "Space-time tree ensemble for action recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 5024–5032.

[104] Shugao Ma et al. "Action recognition and localization by hierarchical space-time segments". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 2744–2751.

[105] David Mace, Wei Gao, and Ayse Coskun. "Accelerometer-based hand gesture recognition using feature weighted naïve bayesian classifiers and dynamic time warping". In: *Proceedings of the companion publication of the 2013 international conference on Intelligent user interfaces companion*. ACM. 2013, pp. 83–84.

[106] Behrooz Mahasseni and Sinisa Todorovic. "Regularizing long short term memory with 3D human-skeleton sequences for action recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3054–3062.

[107] Subhransu Maji, Alexander C Berg, and Jitendra Malik. "Classification using intersection kernel support vector machines is efficient". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[108] Elman Mansimov, Nitish Srivastava, and Ruslan Salakhutdinov. "Initialization Strategies of Spatio-Temporal Convolutional Neural Networks". In: *CoRR* abs/1503.07274 (2015).

[109] Pascal Mettes, Jan C van Gemert, and Cees GM Snoek. "Spot on: Action localization from pointly-supervised proposals". In: *European Conference on Computer Vision*. Springer. 2016, pp. 437–453.

[110] Thomas B Moeslund, Adrian Hilton, and Volker Krüger. "A survey of advances in vision-based human motion capture and analysis". In: *Computer vision and image understanding* 104.2-3 (2006), pp. 90–126.

[111] Alberto Montes, Amaia Salvador, and Xavier Giró i Nieto. "Temporal Activity Detection in Untrimmed Videos with Recurrent Neural Networks". In: *CoRR* abs/1608.08128 (2016).

[112] OV Murthy and Roland Goecke. "Ordered trajectories for large scale human action recognition". In: *Proceedings of the IEEE International Conference on Computer Vision Workshops*. 2013, pp. 412–419.

[113] Hammadi Nait-Charif and Stephen J McKenna. "Activity summarisation and fall detection in a supportive home environment". In: *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. Vol. 4. IEEE. 2004, pp. 323–326.

[114] Hyeonseob Nam and Bohyung Han. "Learning multi-domain convolutional neural networks for visual tracking". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 4293–4302.

[115] Pradeep Natarajan et al. "Multimodal feature fusion for robust event detection in web videos". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 1298–1305.

[116] Joe Yue-Hei Ng et al. "Actionflownet: Learning motion representation for action recognition". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1616–1624.

[117] Do Hang Nga, Y Kawano, and Keiji Yanai. "Fusion of dense SURF triangulation features and dense trajectory based features". In: *THUMOS: ICCV Workshop on Action Recognition with a Large Number of Classes*. Vol. 7. 2013.

[118] Do Hang Nga and Keiji Yanai. "A dense surf and triangulation based spatio-temporal feature for action recognition". In: *Proceedings of the 20th Anniversary International Conference on MultiMedia Modeling*. Vol. 8325. Springer-Verlag New York, Inc. 2014, pp. 375–387.

[119] Bingbing Ni, Xiaokang Yang, and Shenghua Gao. "Progressively parsing interactional objects for fine grained action detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1020–1028.

[120] Bingbing Ni et al. "Motion part regularization: Improving action recognition via trajectory selection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 3698–3706.

[121] Juan Carlos Niebles, Chih-Wei Chen, and Li Fei-Fei. "Modeling temporal structure of decomposable motion segments for activity classification". In: *European conference on computer vision*. Springer. 2010, pp. 392–405.

[122] Dan Oneata, Jakob Verbeek, and Cordelia Schmid. "Action and event recognition with fisher vectors on a compact feature set". In: *Proceedings of the IEEE international conference on computer vision*. 2013, pp. 1817–1824.

[123] David Opitz and Richard Maclin. "Popular ensemble methods: An empirical study". In: *Journal of artificial intelligence research* 11 (1999), pp. 169–198.

[124] Fabián Páez, J Vanegas, and F Gonzalez. "Mindlab at the thumos challenge". In: *THUMOS: ICCV Workshop on Action Recognition with a Large Number of Classes*. Vol. 7. 2013.

[125] Eunbyung Park et al. "Combining multiple sources of knowledge in deep cnns for action recognition". In: *Applications of Computer Vision (WACV), IEEE Winter Conference on*. IEEE. 2016, pp. 1–8.

[126] Alonso Patron-Perez et al. "Structured learning of human interactions in TV shows". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.12 (2012), pp. 2441–2453.

[127] Xiaojiang Peng and Cordelia Schmid. "Encoding Feature Maps of CNNs for Action Recognition". In: *CVPR, THUMOS Challenge 2015 Workshop*. 2015.

[128] Xiaojiang Peng and Cordelia Schmid. "Multi-region two-stream R-CNN for action detection". In: *European Conference on Computer Vision*. Springer. 2016, pp. 744–759.

[129] Xiaojiang Peng et al. "Action recognition with stacked fisher vectors". In: *European Conference on Computer Vision*. Springer. 2014, pp. 581–595.

[130] Sang Phan, Duy-Dinh Le, and Shin'ichi Satoh. "Nii, japan at the first thumos workshop 2013". In: *THUMOS: ICCV Workshop on Action Recognition with a Large Number of Classes*. Vol. 5. 2013, p. 7.

[131] Lionel Pigou et al. "Beyond temporal pooling: Recurrence and temporal convolutions for gesture recognition in video". In: *International Journal of Computer Vision* 126.2-4 (2018), pp. 430–439.

[132] Lasitha Piyathilaka and Sarath Kodagoda. "Gaussian mixture based HMM for human daily activity recognition using 3D skeleton features". In: *Industrial Electronics and Applications (ICIEA), 2013 8th IEEE Conference on*. IEEE. 2013, pp. 567–572.

[133] Yair Poleg et al. "Compact cnn for indexing egocentric videos". In: *Applications of Computer Vision (WACV), IEEE Winter Conference on*. IEEE. 2016, pp. 1–9.

[134] Robi Polikar. "Ensemble based systems in decision making". In: *IEEE Circuits and systems magazine* 6.3 (2006), pp. 21–45.

[135] Ronald Poppe. "A survey on vision-based human action recognition". In: *Image and vision computing* 28.6 (2010), pp. 976–990.

[136] Zhaofan Qiu et al. "Msr asia msm at thumos challenge 2015". In: *CVPR workshop*. Vol. 8. 2015.

[137] Hossein Rahmani and Ajmal Mian. "3d action recognition from novel viewpoints". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1506–1515.

[138] Hossein Rahmani, Ajmal Mian, and Mubarak Shah. "Learning a deep model for human action recognition from novel viewpoints". In: *IEEE transactions on pattern analysis and machine intelligence* 40.3 (2018), pp. 667–681.

[139] Michalis Raptis, Iasonas Kokkinos, and Stefano Soatto. "Discovering discriminative action parts from mid-level video representations". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 1242–1249.

[140] Shaoqing Ren et al. "Faster r-cnn: Towards real-time object detection with region proposal networks". In: *Advances in neural information processing systems*. 2015, pp. 91–99.

[141] Miguel Reyes, Gabriel Domínguez, and Sergio Escalera. "Featureweighting in dynamic timewarping for gesture recognition in depth data". In: *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*. IEEE. 2011, pp. 1182–1188.

[142] Alexander Richard, Hilde Kuehne, and Juergen Gall. "Weakly supervised action learning with RNN based fine-to-coarse modeling". In: *IEEE Conf. on Computer Vision and Pattern Recognition*. Vol. 1. 2. 2017, p. 3.

[143] Mikel D Rodriguez, Javed Ahmed, and Mubarak Shah. "Action mach a spatio-temporal maximum average correlation height filter for action recognition". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[144] Galina Rogova. "Combining the results of several neural network classifiers". In: *Neural networks* 7.5 (1994), pp. 777–781.

[145] Marcus Rohrbach et al. "A database for fine grained activity detection of cooking activities". In: *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE. 2012, pp. 1194–1201.

[146] Negar Rostamzadeh, Jasper Uijlings, and Nicu Sebe. "Action recognition using accelerated local descriptors and temporal variation". In: *THUMOS: ICCV Workshop on Action Recognition with a Large Number of Classes*. Vol. 7. 2013.

[147] Olga Russakovsky et al. "Imagenet large scale visual recognition challenge". In: *International Journal of Computer Vision* 115.3 (2015), pp. 211–252.

[148] Sriparna Saha et al. "Neural network based gesture recognition for elderly health care using kinect sensor". In: *Swarm, Evolutionary, and Memetic Computing*. Springer, 2013, pp. 376–386.

[149] Suman Saha et al. "Deep Learning for Detecting Multiple Space-Time Action Tubes in Videos". In: *Proceedings of the British Machine Vision Conference*. 2016.

[150] Jürgen Schmidhuber. "Deep learning in neural networks: An overview". In: *Neural networks* 61 (2015), pp. 85–117.

[151] Christian Schüldt, Ivan Laptev, and Barbara Caputo. "Recognizing human actions: a local SVM approach". In: *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. Vol. 3. IEEE. 2004, pp. 32–36.

[152] Amir Shahroudy et al. "Deep multimodal feature analysis for action recognition in rgb+ d videos". In: *IEEE transactions on pattern analysis and machine intelligence* 40.5 (2018), pp. 1045–1058.

[153] Amir Shahroudy et al. "NTU RGB+ D: A large scale dataset for 3D human activity analysis". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1010–1019.

[154] Ling Shao, Li Liu, and Mengyang Yu. "Kernelized multiview projection for robust action recognition". In: *International Journal of Computer Vision* 118.2 (2016), pp. 115–129.

[155] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. "Action Recognition using Visual Attention". In: *CoRR* abs/1511.04119 (2015).

[156] Shikhar Sharma, Ryan Kiros, and Ruslan Salakhutdinov. "Action Recognition using Visual Attention". In: *CoRR* abs/1511.04119 (2015).

[157] Jamie Shotton et al. "Real-time human pose recognition in parts from single depth images". In: *Communications of the ACM* 56.1 (2013), pp. 116–124.

[158] Zheng Shou, Dongang Wang, and Shih-Fu Chang. "Temporal action localization in untrimmed videos via multi-stage cnns". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1049–1058.

[159] Karen Simonyan and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos". In: *Advances in Neural Information Processing Systems*. 2014, pp. 568–576.

[160] Bharat Singh et al. "A multi-stream bi-directional recurrent neural network for fine-grained action detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1961–1970.

[161] Suriya Singh, Chetan Arora, and CV Jawahar. "First person action recognition using deep learned descriptors". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2620–2628.

[162] Korsuk Sirinukunwattana et al. "Locality sensitive deep learning for detection and classification of nuclei in routine colon cancer histology images". In: *IEEE transactions on medical imaging* 35.5 (2016), pp. 1196–1206.

[163] Cees GM Snoek, Marcel Worring, and Arnold WM Smeulders. "Early versus late fusion in semantic video analysis". In: *Proceedings of the 13th annual ACM international conference on Multimedia*. ACM. 2005, pp. 399–402.

[164] Andrews Sobral and Antoine Vacavant. "A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos". In: *Computer Vision and Image Understanding* 122 (2014), pp. 4–21.

[165] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild". In: *CoRR* abs/1212.0402 (2012).

[166] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. "Unsupervised learning of video representations using lstms". In: *International conference on machine learning*. 2015, pp. 843–852.

[167] Swathikiran Sudhakaran and Oswald Lanz. "Convolutional Long Short-Term Memory Networks for Recognizing First Person Interactions". In: *Computer Vision Workshop (ICCVW), 2017 IEEE International Conference on*. IEEE. 2017, pp. 2339–2346.

[168] Lin Sun et al. "Human action recognition using factorized spatio-temporal convolutional networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4597–4605.

[169] Lin Sun et al. "Lattice Long Short-Term Memory for Human Action Recognition". In: *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2017, pp. 2166–2175.

[170] Shuai Tang et al. "Histogram of oriented normal vectors for object recognition with a depth sensor". In: *Computer Vision–ACCV 2012*. Springer, 2012, pp. 525–538.

[171] Tieleman Tijmen and Hinton Geoffrey. "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude." In: *COURSERA: Neural Networks for Machine Learning*. 2012.

[172] Carlo Tomasi and Takeo Kanade. *Detection and Tracking of Point Features*. Tech. rep. 1991.

[173] Du Tran et al. "Learning spatiotemporal features with 3d convolutional networks". In: *2015 IEEE International Conference on Computer Vision (ICCV)*. IEEE. 2015, pp. 4489–4497.

[174] Pavan Turaga, Ashok Veeraraghavan, and Rama Chellappa. "Statistical analysis on Stiefel and Grassmann manifolds with applications in computer vision". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE. 2008, pp. 1–8.

[175] Jasper RR Uijlings et al. "Selective search for object recognition". In: *International Journal of Computer Vision* 104.2 (2013), pp. 154–171.

[176] Muhammad Muneeb Ullah, Sobhan Naderi Parizi, and Ivan Laptev. "Improving Bag-of-Features Action Recognition with Non-Local Cues". In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2010, pp. 95.1–95.11.

[177] Gül Varol, Ivan Laptev, and Cordelia Schmid. "Long-term temporal convolutions for action recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 40.6 (2018), pp. 1510–1517.

[178] Andrea Vedaldi and Andrew Zisserman. "Efficient additive kernels via explicit feature maps". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34.3 (2012), pp. 480–492.

[179] Vivek Veeriah, Naifan Zhuang, and Guo-Jun Qi. "Differential recurrent neural networks for action recognition". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 4041–4049.

[180] Taras K Vintsyuk. "Speech discrimination by dynamic programming". In: *Cybernetics and Systems Analysis* 4.1 (1968), pp. 52–57.

[181] Oriol Vinyals et al. "Show and tell: A neural image caption generator". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3156–3164.

[182] Feng Wang, Xiaoyan Li, and Wenmin Shu. "Experimenting motion relativity for action recognition with a large number of classes". In: *THUMOS: ICCV Workshop on Action Recognition with a Large Number of Classes*. Vol. 7. 2013.

[183] Heng Wang and Cordelia Schmid. "Action recognition with improved trajectories". In: *Computer Vision (ICCV), 2013 IEEE International Conference on*. IEEE. 2013, pp. 3551–3558.

[184] Heng Wang et al. "A robust and efficient video representation for action recognition". In: *International Journal of Computer Vision* (2015), pp. 1–20.

[185] Heng Wang et al. "Action recognition by dense trajectories". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE. 2011, pp. 3169–3176.

[186] Heng Wang et al. "Evaluation of local spatio-temporal features for action recognition". In: *Proceedings of the British Machine Vision Conference*. BMVA Press. 2009, pp. 124.1–1.11.

[187] Hongsong Wang, Wei Wang, and Liang Wang. "How scenes imply actions in realistic videos?" In: *Image Processing (ICIP), 2016 IEEE International Conference on*. IEEE. 2016, pp. 1619–1623.

[188] Jinzhuo Wang et al. "Deep alternative neural network: Exploring contexts as early as possible for action recognition". In: *Advances in Neural Information Processing Systems*. 2016, pp. 811–819.

[189] Limin Wang, Yu Qiao, and Xiaoou Tang. "Action recognition with trajectory-pooled deep-convolutional descriptors". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015, pp. 4305–4314.

[190] Limin Wang et al. "Temporal segment networks: towards good practices for deep action recognition". In: *European Conference on Computer Vision*. Springer. 2016, pp. 20–36.

[191] Limin Wang et al. "Towards Good Practices for Very Deep Two-Stream ConvNets". In: *CoRR* abs/1507.02159 (2015).

[192] Ling Wang and Hichem Sahbi. "Directed acyclic graph kernels for action recognition". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013, pp. 3168–3175.

[193] Pichao Wang et al. "Action recognition from depth maps using deep convolutional neural networks". In: *IEEE Transactions on Human-Machine Systems* 46.4 (2016), pp. 498–509.

[194] Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. "Actions˜ transformations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2658–2667.

[195] Xuanhan Wang et al. "Beyond frame-level CNN: saliency-aware 3-D CNN with LSTM for video action recognition". In: *IEEE Signal Processing Letters* 24.4 (2017), pp. 510–514.

[196] Xuanhan Wang et al. "Two-stream 3-D convNet fusion for action recognition in videos with arbitrary size and length". In: *IEEE Transactions on Multimedia* 20.3 (2018), pp. 634–644.

[197] Yang Wang and Minh Hoai. "Improving human action recognition by non-action classification". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2698–2707.

[198] Daniel Weinland, Remi Ronfard, and Edmond Boyer. "A survey of vision-based methods for action representation, segmentation and recognition". In: *Computer vision and image understanding* 115.2 (2011), pp. 224–241.

[199] Philippe Weinzaepfel, Zaid Harchaoui, and Cordelia Schmid. "Learning to track for spatio-temporal action localization". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 3164–3172.

[200] Geert Willems, Tinne Tuytelaars, and Luc Van Gool. "An efficient dense and scale-invariant spatio-temporal interest point detector". In: *European conference on computer vision*. Springer. 2008, pp. 650–663.

[201] Terry Windeatt. "Accuracy/diversity and ensemble MLP classifier design". In: *IEEE Transactions on Neural Networks* 17.5 (2006), pp. 1194–1211.

[202] Di Wu et al. "Deep dynamic neural networks for multimodal gesture segmentation and recognition". In: *IEEE transactions on pattern analysis and machine intelligence* 38.8 (2016), pp. 1583–1597.

[203] Jialin Wu et al. "Action Recognition with Joint Attention on Multi-Level Deep Features". In: *CoRR* abs/1607.02556 (2016).

[204] Yonghui Wu et al. "Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation". In: *CoRR* abs/1609.08144 (2016).

[205] Zuxuan Wu et al. "Harnessing object and scene semantics for large-scale video understanding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 3112–3121.

[206] SHI Xingjian et al. "Convolutional LSTM network: A machine learning approach for precipitation nowcasting". In: *Advances in neural information processing systems*. 2015, pp. 802–810.

[207] Kelvin Xu et al. "Show, attend and tell: Neural image caption generation with visual attention". In: *International Conference on Machine Learning*. 2015, pp. 2048–2057.

[208] Yong Xu et al. "Background modeling methods in video analysis: A review and comparative evaluation". In: *CAAI Transactions on Intelligence Technology* 1.1 (2016), pp. 43–60.

[209] Junji Yamato, Jun Ohya, and Kenichiro Ishii. "Recognizing human action in time-sequential images using hidden markov model". In: *Computer Vision and Pattern Recognition, 1992. Proceedings CVPR'92., 1992 IEEE Computer Society Conference on*. IEEE. 1992, pp. 379–385.

[210] Xiaodong Yang, Pavlo Molchanov, and Jan Kautz. "Making Convolutional Networks Recurrent for Visual Sequence Learning". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 6469–6478.

[211] Xiaodong Yang, Chenyang Zhang, and YingLi Tian. "Recognizing actions using depth motion maps-based histograms of oriented gradients". In: *Proceedings of the 20th ACM international conference on Multimedia*. ACM. 2012, pp. 1057–1060.

[212] Yuancheng Ye and Yingli Tian. "Embedding sequential information into spatiotemporal features for action recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2016, pp. 37–45.

[213] Serena Yeung et al. "End-to-end learning of action detection from frame glimpses in videos". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2678–2687.

[214] Limin Wang Luc Van Gool Yifan Wang Jie Song and Otmar Hilliges. "Two-Stream SR-CNNs for Action Recognition in Videos". In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2016, pp. 108.1–108.12.

[215] Gang Yu and Junsong Yuan. "Fast action proposals for human action detection and search". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1302–1311.

[216] Joe Yue-Hei Ng et al. "Beyond short snippets: Deep networks for video classification". In: *CVPR*. 2015, pp. 4694–4702.

[217] Shengxin Zha et al. "Exploiting Image-trained CNN Architectures for Unconstrained Video Classification". In: *Proceedings of the British Machine Vision Conference*. BMVA Press, 2015, pp. 60.1–60.13.

[218] Bowen Zhang et al. "Real-time action recognition with enhanced motion vector CNNs". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2718–2726.

[219] Chen-Lin Zhang et al. "Deep bimodal regression for apparent personality analysis". In: *European Conference on Computer Vision*. Springer. 2016, pp. 311–324.

[220] Chenyang Zhang and Yingli Tian. "Rgb-d camera-based daily living activity recognition". In: *Journal of Computer Vision and Image Processing* 2.4 (2012), p. 12.

[221] Songyang Zhang, Xiaoming Liu, and Jun Xiao. "On geometric features for skeleton-based action recognition using multilayer lstm networks". In: *Applications of Computer Vision (WACV), IEEE Winter Conference on*. IEEE. 2017, pp. 148–157.

[222] Tongchi Zhou et al. "Learning semantic context feature-tree for action recognition via nearest neighbor fusion". In: *Neurocomputing* 201 (2016), pp. 1–11.

[223] Yang Zhou et al. "Interaction part mining: A mid-level approach for fine-grained action recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3323–3331.

[224] Chun Zhu and Weihua Sheng. "Multi-sensor fusion for human daily activity recognition in robot-assisted living". In: *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*. ACM. 2009, pp. 303–304.

[225] Wangjiang Zhu et al. "A key volume mining deep framework for action recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 1991–1999.

[226] Yuke Zhu et al. "Target-driven visual navigation in indoor scenes using deep reinforcement learning". In: *Robotics and Automation (ICRA), 2017 IEEE International Conference on*. 2017, pp. 3357–3364.

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