Research & Development Report
Master Autonomous Systems
University of Applied Sciences Bonn-Rhein-Sieg

Evaluation of current Approaches for Situation-Awareness in Autonomous Systems from Action Recognition in Video Data

Author:
Maximilian Schöbel
Mat. Nr. 9027059

Advisors:
Prof. Dr. Erwin Prassler
Prof. Dr. Paul G. Plöger

January 12th, 2017

Abstract

Situation awareness is an abstract concept, which includes several independent manifestations and sensory inputs. An important aspect of situation awareness is the knowledge of actions, that are performed by persons in the vicinity of an agent, to derive a suitable policy for the agent's own future actions. This report investigates vision-based human action recognition from video data as a key step towards the understanding of scenes by autonomous (robotic) systems. Recognition of actions furthermore enables applications in surveillance systems, patient monitoring, human-computer interfaces, assisted living environments and AI. So far, computer vision research produced two main approach classes of machine learning based algorithms for classifying actions in video-data: Either using arbitrary classification techniques on top of hand-crafted feature extractors or end-to-end trainable deep learning architectures. This report provides a detailed review of current deep learning approaches for video-based action recognition in addition to a comparison with conventional hand-crafted feature approaches. Furthermore, publicly available video datasets are reviewed, since the availability of labeled large-scale datasets is a big challenge in deep learning. Even very recent video-datasets do not match the size of existing image datasets. Results indicate, that deep learning techniques are competitive but not yet able to significantly outperform conventional hand-crafted feature methods in video-based action recognition.

Contents

1 Introduction **7**
- 1.1 The Action Recognition Problem . . . 7
- 1.2 Action Recognition Surveys (Related Work) . . . 8
- 1.3 Scope and Structure of this Report . . . 9

2 Conventional Methods in Action Recognition **11**
- 2.1 Single-layered Space-Time Approaches . . . 12
  - 2.1.1 Action recognition using Space-Time Volumes . . . 12
  - 2.1.2 Action Recognition using Trajectories . . . 12
  - 2.1.3 Action Recognition using Local Spatio-Temporal Features . . . 14
- 2.2 Single-layered Sequential Approaches . . . 16
  - 2.2.1 Exemplar-based Approaches . . . 16
  - 2.2.2 State Model-based Approaches . . . 17
- 2.3 Hierarchical Approaches . . . 17
  - 2.3.1 Statistical Approaches . . . 17
  - 2.3.2 Syntactic Approaches . . . 18
  - 2.3.3 Description-based Approaches . . . 18
- 2.4 State of the Art Approaches using Local Features . . . 18
  - 2.4.1 Action Recognition by Dense Trajectories (2011) . . . 19
  - 2.4.2 Action recognition with improved trajectories (2013) . . . 22

3 Deep Learning Methods in Action Recognition **23**
- 3.1 Convolutional Networks . . . 24
  - 3.1.1 3D Convolutional Neural Networks for Human Action Recognition (2010/2013) . . . 24
  - 3.1.2 Sequential Deep Learning for Human Action Recognition (2011) . . . 28
  - 3.1.3 Large-scale Video Classification with Convolutional Neural Networks (2014) . . . 31
  - 3.1.4 Learning Spatiotemporal Features with 3D Convolutional Networks (2015) . . . 35
  - 3.1.5 Long-term Temporal Convolutions for Action Recognition (2016) . . . 37
- 3.2 Multiple Stream Networks . . . 42
  - 3.2.1 Two-Stream Convolutional Networks for Action Recognition in Videos (2014) . . . 42
  - 3.2.2 Beyond Short Snippets: Deep Networks for Video Classification (2015) . . . 46
  - 3.2.3 Towards Good Practices for Very Deep Two-Stream ConvNets (2015) . . . 51
  - 3.2.4 Convolutional Two-Stream Network Fusion for Video Action Recognition (2016) . . . 53
- 3.3 Pooling of Deeply Learned Features . . . 57
  - 3.3.1 Action recognition with trajectory-pooled deep-convolutional descriptors (2015) . . . 57
- 3.4 Generative Models . . . 61
  - 3.4.1 Unsupervised Learning of Video Representations using LSTMs (2015) . . . 62
  - 3.4.2 Action Recognition Using Convolutional Restricted Boltzmann Machines (2016) . . . 67
- 3.5 Temporal Coherency Networks . . . 70
  - 3.5.1 Shuffle and Learn: Unsupervised Learning using Temporal Order Verification (2016) 70

4 Datasets and Benchmarks in Action Recognition **73**
- 4.1 Early Benchmarking Datasets – Controlled Conditions . . . 75
  - 4.1.1 KTH – 2004 . . . 75
  - 4.1.2 Weizmann – 2005 . . . 76
  - 4.1.3 IXMAS – 2006 . . . 77
- 4.2 Intermediate Benchmarking Datasets – Television and Movies . . . 78
  - 4.2.1 UCF Sports – 2008 . . . 78
  - 4.2.2 Hollywood – 2008 . . . 79
  - 4.2.3 Hollywood 2 – 2009 . . . 80
- 4.3 Modern Benchmarking Datasets – Videos in the Wild . . . 81
  - 4.3.1 UCF11 Youtube Action – 2009 . . . 81

4.3.2 UCF50 – 2010 . . . 82
4.3.3 HMDB51 – 2011 . . . 84
4.3.4 UCF101 – 2012 . . . 86
4.3.5 ASLAN – 2012 . . . 86
4.3.6 Sports-1M – 2014 . . . 89
4.3.7 YouTube-8M – 2016 . . . 90
4.4 Activities of Daily Living (ADL) Datasets . . . 91
4.4.1 URADL – 2009 . . . 91
4.4.2 Multiple Cameras Fall Dataset – 2010 . . . 91
4.4.3 MPII Cooking Activities Dataset – 2012 . . . 92
4.4.4 ActivityNet – 2015 . . . 93
4.4.5 Charades – 2016 . . . 95

5 Comparison and Evaluation **98**

6 Conclusion and Future Directions **101**

References **102**

Statement of Originality

This is to certify that the content of this report is my own work and that all sources have been acknowledged.

Date:

Signature:

1 Introduction

Humans are perfectly able to act and move in unknown, crowded environments and even react successfully to unexpected situations because they are in general aware of their surroundings. Being able to localize and identify human actions, that are performed in the vicinity of an agent, allows predicting the future state of an agent's environment and therefore induces situation awareness.

In computer vision research, advances in action recognition from video are driven by the vast amount of possible applications, especially since video cameras are a cost-effective and widely utilized technology. Action recognition in robotics, could provide improved navigation by recognizing actions that imply an imminent movement trajectory of pedestrians. Video surveillance in public environments can profit from recognizing potentially dangerous actions. Other possible applications include surveillance of children or the elderly in assisted living environments, patient monitoring in hospitals, and human-computer interaction.

1.1 The Action Recognition Problem

Human action recognition is a classification task and denotes the process of labeling image sequences with action categories, that were introduced by an annotated training dataset. In contrast to object recognition in still images, videos provide an additional temporal dimension, which conveys information in the form of the temporal evolution of motion. On the one hand, this information can be accessed for classification, on the other hand, the amount of possible variations poses additional challenges for robust action recognition algorithms. These challenges, as summarized by Poppe [1], are described below.

Intra-class variations and inter-class similarities:
Intra class variations are differences that occur in the appearance of actions, which belong to the same action category. These difference stem from each person having their very own characteristics in appearance and motion execution. Additionally actions may be superimposed by other actions, a person might gesture, close a jacket or use a mobile phone while *walking*.

Furthermore actions may appear similar, but belong to different action categories. This stems from some action categories being distinguishable only by small details and the large variations in execution styles among different persons.

Background and recording settings:
Recording settings and the environment in which an action is performed are sources for variety in the appearance of an action. There may be other moving objects behind a person, lighting conditions may differ drastically, a person may be occluded by an object and different view-points result in large differences in image observations.

Temporal variations:
Substantial variations can occur from differences in the execution speed of an action, which can stem from different execution styles of persons or different rates of recording the video.

The main task of action recognition research is to overcome these challenges and built systems, that recognize actions robustly, even when they are performed by different persons in different environments at different speeds. Requirements for such an approach are a discriminative architecture that is able to recognize the general characteristics of different action classes while ignoring personal characteristics of different performers and large datasets that provide this information by containing a lot of different examples for each action class.

Action recognition research can be broadly divided in two categories: Conventional hand-crafted feature methods and deep learning approaches. Hand-crafted feature methods (section 2 of this work) build a global video representation by processing extracted features from an input video. Feature extractors for this task are carefully hand-crafted to capture the important motion information in the video.

Motivated by the success of deep learning architectures in image classification [2, 3, 4], deep learning has been widely applied to human action recognition from video (section 3 of this work). Deep Learning approaches require large amounts of input data, to train the parameters of a deep architecture for action recognition.

1.2 Action Recognition Surveys (Related Work)

Given the large number of possible applications, the computer vision community has shown an increased interest in human action recognition from video, which resulted in the publication of several comprehensive survey articles [1, 5, 6, 7, 8, 9].

Poppe [1] uses a hierarchical categorization of human motion in action primitives, actions and activities with increasing complexity. Action primitives denote atomic movements of limbs. Actions are movements of the whole body and consist of multiple action primitives. Activities are formed by the subsequent execution of actions. The survey focuses on the recognition of single-person actions using hand-crafted features without explicitly considering context, such as the environment, object or other persons. The main deficit of [1] is, that deep learning methods are not included in the review and only a brief overview of available action recognition datasets is provided.

Aggarwal and Ryoo [5] provide a very comprehensive and structured discussion of conventional methods in action recognition using an approach-based taxonomy, which has been adopted in other survey publications as well [10]. The authors divide human motion into (atomic) actions and high-level activities. They review approaches for the recognition of actions, activities, human-object interactions and group activities, but focus on the recognition of high-level activities. Although providing an informative and detailed

overview in the field of human action recognition with conventional hand-crafted feature methods, deep learning approaches are not included and only a brief section on available benchmarking datasets is provided.

Chaquet, Carmona, and Fernández-Caballero [6] address the lack of a comprehensive description of publicly available datasets for human action and activity recognition from monocular video data. They provide a detailed overview and description of *heterogeneous* action datasets, which contain actions that can occur in any context and any environment. Additionally examples of *specific* action datasets are given, e.g. for the recognition of abandoned objects, activities of daily living, crowd behaviour, falls, gait and gesture. Datasets containing motion capture data and thermal/infrared imaging are discussed briefly as well. Since the survey was published in 2013, very recent large-scale datasets could not be covered. These are particularly important for deep learning algorithms, which require a large amount of training data.

Längkvist, Karlsson, and Loutfi [7] provide a general overview of deep learning methods for the analysis of time-series data. They provide a comparative description of different deep architectures and how they can be applied to time series data. Action Recognition from video is discussed along other topics like stock-market prediction, speech recognition, music recognition as well as recognition from motion capture and electronic nose data. Given the wide range of the review, action recognition from video is not discussed in detail.

The work of Herath, Harandi, and Porikli [8] is closely related to the scope of this work by considering conventional hand-crafted feature methods as well as deep learning approaches in action recognition. While providing an informative description of conventional approaches, the authors do not describe specifics of deep learning approaches in action recognition. Video action datasets are only discussed very briefly and recent developments are not covered.

The review of Kang and Wildes [9] provides a very detailed description of datasets for action recognition algorithms and evaluation procedures. Additionally common techniques in using hand-crafted features for action recognition are discussed in detail. Main deficit is the lack of deep learning approaches.

1.3 Scope and Structure of this Report

Throughout this work, the term *action* is used to describe atomic actions as well as high-level activities. Whenever a distinction is necessary, we use either *atomic action* or *high-level action* to refer to actions or activities.

Given the amount of comprehensive and detailed publications reviewing conventional hand-crafted feature approaches in action recognition, we build on the approach-based taxonomy of Aggarwal and Ryoo [5] to provide a condensed overview of the main research directions in this area in section 2. This report focuses on deep learning ap-

proaches. Therefore section 3 provides a classification and detailed description of recent deep learning approaches in action recognition. Given the importance of training data for deep learning, section 4 provides characteristics of the most common video action datasets usable for the training and evaluation of action recognition algorithms. Since this report aims at enabling an implementation of an action recognition system in an assisted living environment, we additionally identify and review datasets, that contain activities of daily-living (ADL datasets).

2 Conventional Methods in Action Recognition

This section provides a description of conventional methods in action recognition, i.e. methods that do not utilize deep learning techniques and mostly rely on the extraction of hand-crafted features from input-videos. Due to the availability of several high-quality survey publications in this area, as described in the related work section 1.2, we provide a condensed overview of conventional action recognition methods by describing the main research directions using the taxonomy of Aggarwal and Ryoo [5]. For a more detailed description of specific approaches in this area we refer to the publications described in section 1.2.

The approach-based taxonomy of Aggarwal and Ryoo [5] is depicted in figure 1. Since its publication in 2011, using space-time features has become the standard approach in video classification [11]. Space-time features are also referred to as local spatio-temporal features, because they describe a local spatio-temporal region of a video.

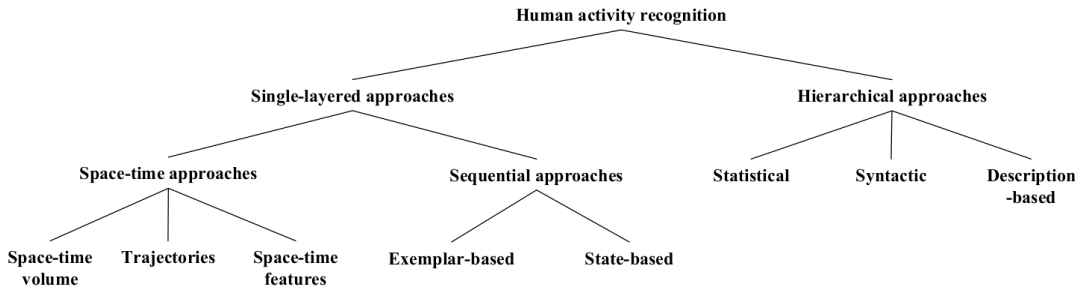


Figure 1: Approach-based taxonomy for conventional methods in human action recognition as given by Aggarwal and Ryoo [5]

Aggarwal and Ryoo [5] divide the field of activity/action recognition into single-layered and hierarchical approaches.

Single-layered Approaches recognize an action by directly processing raw video-data, i.e. based on sequences of video frames.

Hierarchical Approaches model an action as a sequence of explicitly defined and individually recognizable atomic sub-actions.

Single-layered approaches are further categorized into space-time approaches and sequential approaches.

Space-time approaches interpret a video as a 3D space-time volume, that results from stacking the individual video-frames along the temporal dimension.

Sequential approaches interpret a video as a sequence of observations, i.e. feature vectors extracted from individual frames.

2.1 Single-layered Space-Time Approaches

Space-time approaches use the entire video volume for action recognition and can be distinguished by what kind of features they use from the volume [5]. Space-time volume approaches utilize pixel values of the full video volume or parts of it directly for creating a representation of the video, which is then compared with other video volume representations. Trajectory based approaches use motion trajectories of tracked points inside the volume for the recognition of actions contained in it. Space-time feature approaches extract features around interest-points locally and aggregate them into a representation of the video volume.

2.1.1 Action recognition using Space-Time Volumes

A prototypical approach for action recognition with space-time volumes is described in [5] and uses template matching: Given a similarity measure for video volumes, the algorithm constructs or selects video volumes template from the training dataset for each action class that has to be recognized. The template video volumes then act as representations for the action classes. When presented with a test-video, the algorithm constructs the representation for the new video and compares it to the stored training templates by using the similarity measure. The action class, that corresponds to the most similar training template is selected as output class for the test-video.

Approaches in this category mainly differ by how a representation is built and how they are compared. Bobick and Davis [12] create templates from the raw video volumes by stacking the silhouettes of persons that perform an action into 2D images. The resulting binary *motion-energy image* and skalar-valued *motion-history image*, as displayed in figure 2, are then classified by template matching as described above.

2.1.2 Action Recognition using Trajectories

The underlying idea of trajectory-based approaches is, that the motion of a person's joint positions are sufficient for recognizing the performed action [13]. Algorithms that follow this approach use space-time trajectories to represent an action. More specifically the joint positions of a person are tracked in the video volume. The resulting trajectories then represent the performed action and can be used for classification, by either comparing the trajectories directly or by extracting features along the trajectories.

Sheikh, Sheikh, and Shah [14] classify actions by using the trajectories of 13 tracked points directly as displayed in figure 3.

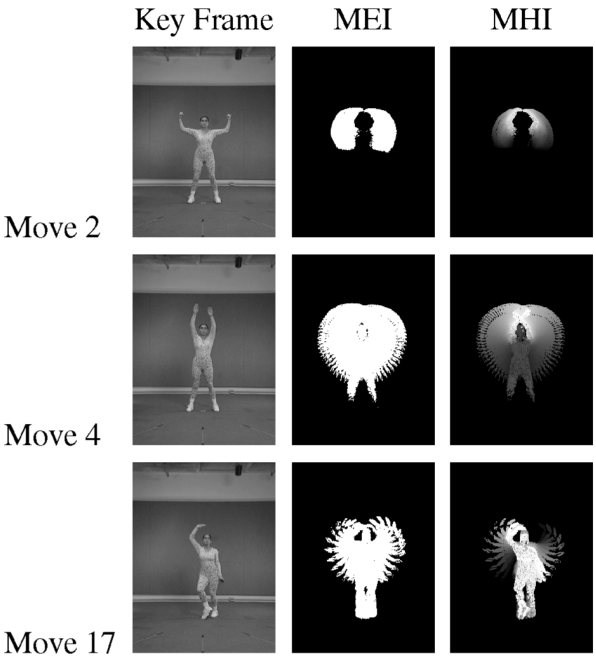


Figure 2: Motion-energy image (MEI), motion-history image (MHI) and example frame of three ballet actions [12]

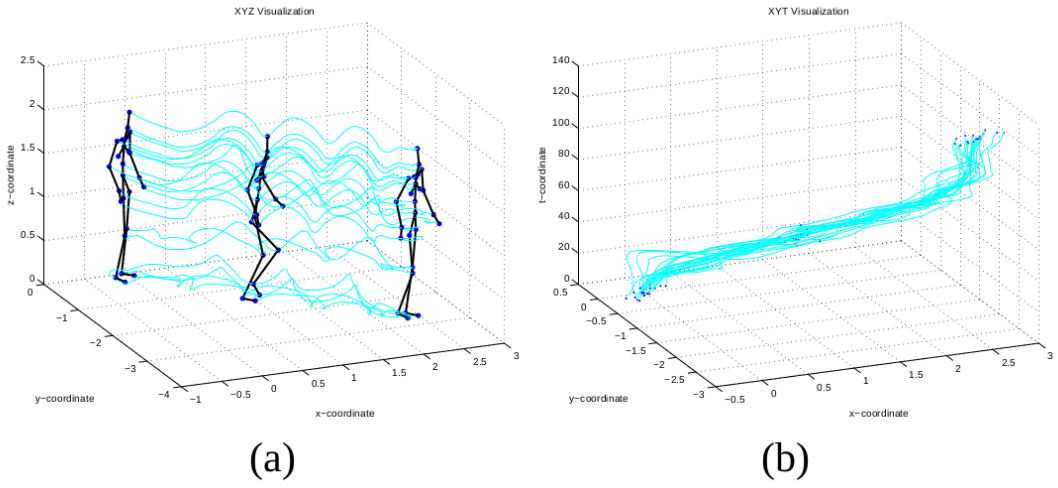


Figure 3: Trajectories of a person's tracked joint positions while performing an action. Trajectories shown in XYZ space (a) and XYT space (b) [14]

2.1.3 Action Recognition using Local Spatio-Temporal Features

The principle idea behind action recognition with local spatio-temporal features is, that each action produces characteristic changes of pixel values in local 3D regions of the video containing the action [1]. Algorithms can therefore recognize actions by learning the correspondence between an action class and the set of the local regions, that are produced by the class. A region, that contains these characteristic changes in pixel values, corresponds to a location in the video and is called a *local spatio-temporal feature* or a *spatio-temporal interest point*. Aggarwal and Ryoo [5] use the terms interchangeably.

Poppe [1] defines spatio-temporal interest points as regions in a video, where sudden changes in motion appear. They note, that these regions are considered highly informative for action recognition and generally, points that perform a simple translation in the spatio-temporal video volume do not produce a local spatio-temporal feature.

More specifically, Schüldt, Laptev, and Caputo [15] define a local space-time feature as "primitive events corresponding to moving two-dimensional image structures at moments of non-constant motion".

In general, a local spatio-temporal feature approach for action recognition contains three main aspects: [11][5]

1. **Feature Extraction**: Determining what kind of features, i.e. changes in pixel values, are considered as characteristic for the ongoing motion and where they are located. Feature extractors are specifically hand-crafted to best capture local motion information.
2. **Feature Encoding**: Creating a fixed-sized representation of a video using the extracted local spatiotemporal features.
3. **Classification**: Learns the correspondence between the video representations and the action class it contains.

Feature extraction can be further divided into interest-point detection and local description [1]. Interest point detectors find and locate the locations of non-constant, informative motion. Local descriptors transform the raw pixel values in neighbourhoods around previously detected interest points, to best capture the inherent motion information.

A comparative evaluation of different interest point detectors and local descriptors was published in 2009 by Wang et al. [16]. They found, that instead of using computationally expensive feature detectors, dense sampling of interest points from the video is a well performing alternative.

Prototypical approaches using local features were proposed by Laptev [17] and Dollár et al. [18]. Laptev [17] extended the Harris corner and edge detector [19] for extracting space-time interest points in videos.

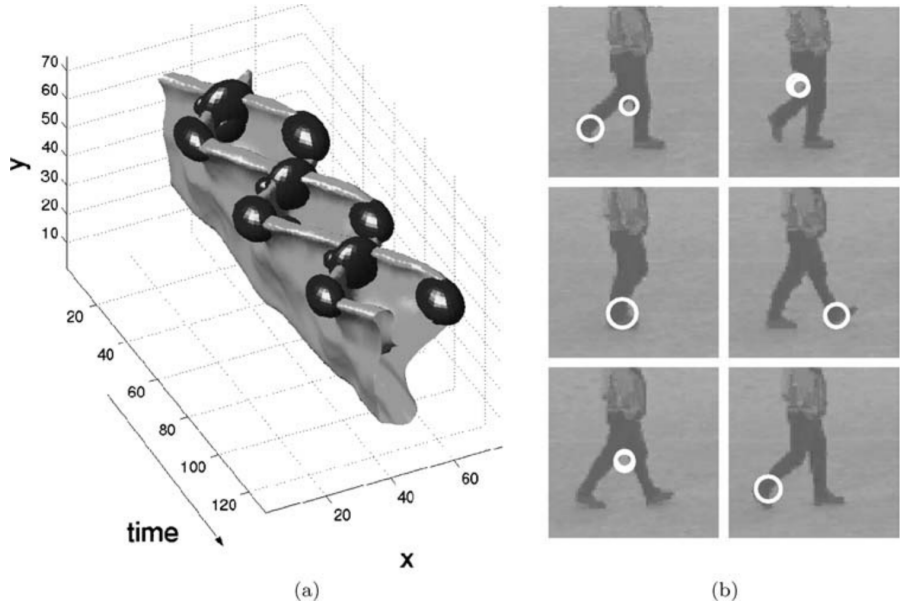


Figure 4: Detection of spatio-temporal interest points from the movement of legs. *a*) Walking pattern of the legs (presented upside down) along with detected interest points (black ellipsoids). *b*) Positions of detected interest points, projected into the original video. [17]

Dollár et al. [18] proposed a feature detector for detecting periodic motions. Once a spatio-temporal interest point is detected, a cube of video pixels called a *Cuboid* as shown in figure 5 is assigned to each detected point. The local appearance information in each cube is then extracted and aggregated into a global video description. They found, that motion information in the cuboids is best captured by transforming each into a flattened vector of brightness gradients to form the final local feature. Given a training dataset, these final local features in the form of several vectors for each video, are extracted and a codebook of feature vectors is formed by using k-means. Applying to the bag-of-words paradigm, a video is represented as a histogram of codeword frequencies given the codebook. Specifically, at first the feature vectors are extracted from a video, each vector is then assigned to its nearest codebook feature vector, which is called a visual word. By counting the number of visual word occurrences, a histogram representation of the video is created. A classifier then learns the correlation between histogram representations and action labels. This approach ignores the spatio-temporal relations between detected interest points.

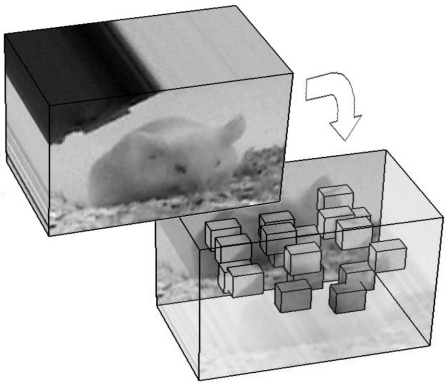


Figure 5: [18]

Further examples of feature detectors are given in [1].

Beyond using the bag-of-words paradigm for encoding local features into a global video representation, other more advanced methods were proposed such as the Fisher Vector[20] and the Vector of Locally Aggregated Descriptors (VLAD) [21]. An empirical evaluation of different encoding methods is given in [22].

2.2 Single-layered Sequential Approaches

In single-layered sequential approaches an action is processed as a sequence of observations, specifically as a sequence of extracted feature vectors [5]. As a first step in sequential approaches, feature vectors need to be extracted from each frame in the video that contains the action. Aggarwal and Ryoo [5] describe an example of using degrees of joint-angles as suitable feature vectors to describe the status of a person while performing an action. Given the sequence of feature vectors from in input video, sequential approaches usually calculate the likelihood of action classes producing this observed sequence of feature vectors. The action class with the highest likelihood is assigned to the input video.

Aggarwal and Ryoo [5] further differentiate sequential approaches into exemplar-based and state model-based approaches.

2.2.1 Exemplar-based Approaches

Exemplar-based approaches store template sequences of feature vectors for each action class [5]: A presented unknown action in an input video is recognized, by comparing its sequence of feature vectors to the stored templates. The action class, whose template is most similar to the feature sequence of the input video is assigned to the input. The approach has to take into account, that the feature sequences may vary because of different execution styles of action among different persons.

The Dynamic Time Warping algorithm (DTW) has been used widely for matching varying sequences in sequential exemplar-based approaches [23][24][25]. Following figure 6 illustrates the concept.

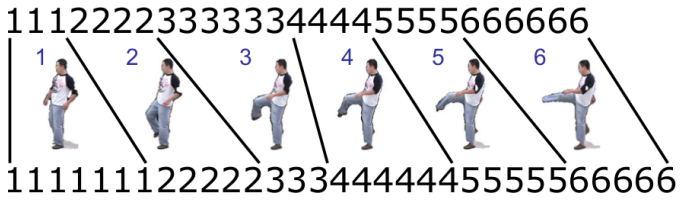


Figure 6: Matching of two action sequences with different execution rates. Each number corresponds to a pose of the person [5]

2.2.2 State Model-based Approaches

In contrast to representing an action as a sequence of observations, state model-based approaches train one statistical model for each action class and an action is represented as a sequence of the model's hidden states [10]. Using the training videos of an action class, a model is trained on the extracted feature vector observations, so that it generates sequences of feature vectors corresponding to the class. Given a test video, the likelihood of each model generating this sequence is calculated. The class, that corresponds to the model with the highest likelihood, is assigned to the test video. Hidden Markov Models and Dynamic Bayesian Networks have been used for these approaches [5].

2.3 Hierarchical Approaches

The main idea of hierarchical approaches is to model complex actions as a hierarchy of simpler sub-actions [5]. Sub-actions themselves can be further decomposed, until the initial complex action is represented as a sequence of non-decomposable atomic sub-actions. A complex action is interpreted as a process that generates sub-actions which can be observed and classified individually. Most hierarchical approaches thereby employ non-hierarchical single-layered action recognition approaches to recognize low-level sub-actions.

Aggarwal and Ryoo [5] further differentiate hierarchical approaches into statistical approaches, syntactic approaches and desciption-based approaches.

2.3.1 Statistical Approaches

Hierarchical statistical approaches use hierarchically stacked state-based models such as Hidden Markov Models (HMMs) [26][27] or Dynamic Bayesian Networks (DBNs) [28][29] for action recognition. Typically two layers of such models are used, where the

bottom layer recognizes simple actions from sequences of feature vectors and the top layer recognizes high-level actions from the resulting sequence of simple actions. The layered Hidden Markov Model approach of Oliver, Horvitz, and Garg [26] is said to be one of the most fundamental forms of hierarchical statistical approaches [5].

2.3.2 Syntactic Approaches

In hierarchical syntactic approaches, a high-level action is represented as a string of symbols [10]. Each symbol therein corresponds to a simpler, possibly atomic, sub-action as described previously. Equivalently to hierarchical statistical approaches, syntactic approaches require the recognition of sub-actions by using any of the previously described methods in order to obtain the string of symbols. An action class is represented as a set of production rules from context-free grammars or stochastic context-free grammars [30][31][32][33], that generate sequences of symbols corresponding to the action class. Syntactic approaches then use parsing techniques from the field of programming languages [34] to recognize high-level actions.

2.3.3 Description-based Approaches

The definition of description-based approaches by Aggarwal and Ryoo [5] states: "A description-based approach is a recognition approach, that explicitly maintains human activities' spatio-temporal structure." High-level actions are represented by occurrences of their underlying sub-actions, while temporal, spatial and logical relationships between the sub-actions are explicitly specified. The temporal relations between sub-actions are usually specified by associating a time interval with an occurring action. Allen [35][36] introduced seven predicates to describe temporal relations between time intervals: *before, meets, overlaps, during, starts, finishes* and *equals*, which have been widely used for hierarchical description-based approaches [37][38][39][40].

2.4 State of the Art Approaches using Local Features

Wang and Schmid [41] introduced the *Improved Dense Trajectories* approach, which is often considered state of the art in action recognition using hand-crafted local features [42][43][44]. Using the categorization proposed by the taxonomy of Aggarwal and Ryoo [5], it can be seen as a hybrid approach of being trajectory-based and using hand-crafted local features.The principle idea behind *Improved Dense Trajectories* is to densely sample spatio-temporal interest points, tracking them using optical flow and extracting local hand-crafted features in regions around the resulting point trajectories.

A basic version of the approach, called *Dense Trajectories*, was first published in 2011 [45] and later improved in 2013 (*Improved Dense Trajectories*) [41].

A comparison of hand-crafted feature approaches to recent deep learning methods in action recognition is given in section 5. Beyond *Improved Dense Trajectories*, the following approaches have been considered state of the art [43][46]and are compared as well:

- Multi-view super vector for action recognition (2014) [47]
- Beyond Gaussian pyramid: Multi-skip feature stacking for action recognition (2015) [46]

2.4.1 Action Recognition by Dense Trajectories (2011)

Wang et al. [45] introduce a tracking technique called *dense trajectories* for action classification from videos. Points are sampled densely from each frame and then tracked using a dense optical flow field. Local spatio-temporal features are extracted in regions around the resulting point trajectories to form trajectory descriptors, which are then aggregated into a global video descriptor using the bag-of-words paradigm.

Since motion in a video, and therefore also the resulting trajectories, can stem from either motion of interest or unwanted camera motion, the authors propose using a descriptor called *Motion Boundary Histogram* (MBH), which aims at focussing on foreground motion. The MBH descriptor is designed to make the classification of actions in a video invariant to camera motion.

The overall approach for obtaining a descriptor along densely extracted trajectories is shown in figure 7

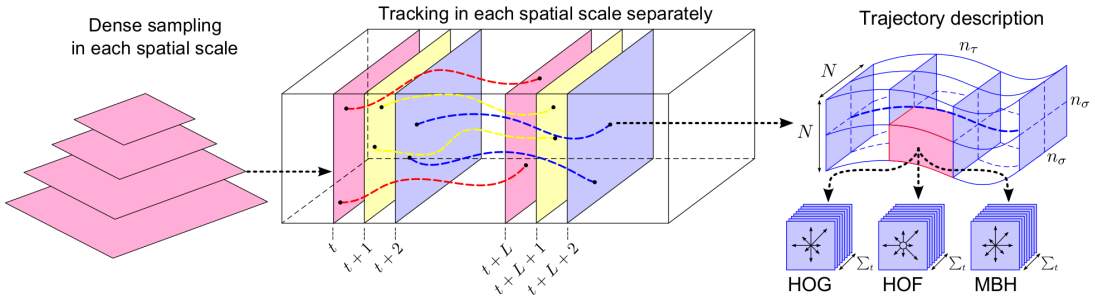


Figure 7: Description of densely extracted trajectories [45]

Dense trajectories are obtained separately from 8 spatial scales, which differ by a factor of $1/\sqrt{2}$. Points are sampled on a grid spaced by W pixels on each scale. Experimentally $W = 5$ has been shown to yield good results. Each point P_t at frame t is tracked to the next frame by using a dense optical flow field, which was extracted by the Farnebäck algorithm [48] as implemented in OpenCV.

Since only dynamic information is important for action recognition, static trajectories

are removed in a pre-processing stage. Erroneous trajectories with sudden large displacements are also removed.

A simple descriptor is obtained from the shape of a trajectory itself. It is formed by normalizing the spatial displacements given by the differences of consecutive points in a trajectory. Formally the *trajectory descriptor* S' is given by:

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

Where $\Delta P_t = (P_{t+1} - P_t) = (x_{t+1} - x_t, y_{t+1} - y_t)$.

Local Feature Descriptors:
Local features are extracted from video volumes of size $N \times N \times L$ around the trajectories as depicted in figure 7, where $N = 32$ has shown to yield good results.

Feature descriptors evaluated in the context of dense trajectories are):

- **HOG** (Histogram of Oriented Gradients) [49]
- **HOF** (Histogram of Optical Flow) [50]
- **MBH** (Motion Boundary Histogram) [51]

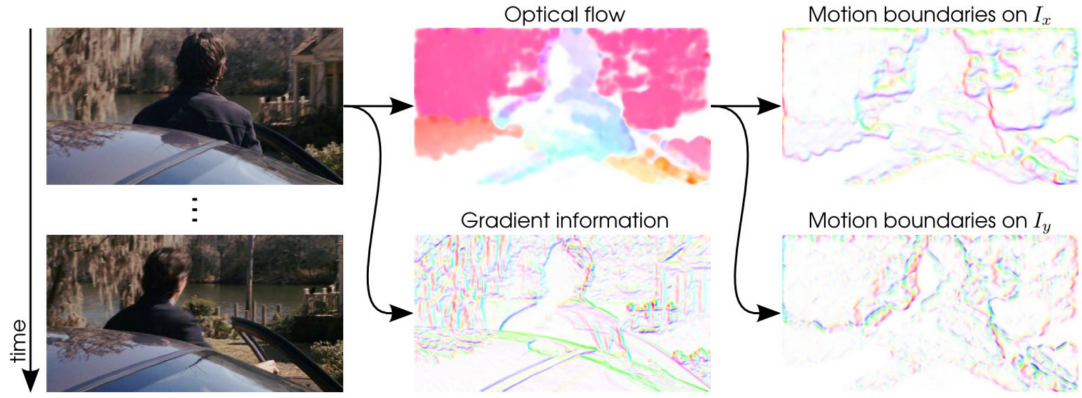


Figure 8: Visualization of the information captured by HOG, HOF and MBH on complete video frames. In each image, the orientation is given by color, the magnitude is given by saturation. [45]

The HOG descriptor encodes static appearance information by computing the orientations of image gradients and aggregating them in a histogram over all subframes of the current video volume along the trajectory. In this approach the histograms contain 8 bins.

The HOF descriptor aggregates the orientations of optical flow vectors in a histogram and therefore captures local motion information. An additional bin is used here.

The MBH descriptor separately calculates the spatial derivatives of the x- and y-component of the optical flow field. The orientations of the derivatives are aggregated into histograms (similarly to the HOG descriptor), which represent the video volume. An advantage of MBH is that is suppresses constant motion, since it takes only the changes in the flow field (i.e. motion boundaries) into account. The authors therefore use MBH as an easy way to filter noise stemming from background camera-motion, as can be seen by comparing the optical flow-image and motion boundaries in figure 8.

The authors evaluate their approach on the KTH, YouTube, Hollywood2 and UCF-Sports dataset using a standard bag-of-features approach as follows:

1. Construction of a codebook for each descriptor-type (trajectory, HOG, HOF and MBH). 100.000 descriptors for each type are randomly chosen from all extracted descriptors over the training split. These descriptors are clustered into a 4000 words long codebook using k-means.
2. Each extracted descriptor from a video is assigned to its nearest codebook-descriptor using the Euclidean distance. The number of occurrences are aggregated in a histogram, which builds the global video descriptor.
3. A classifier (here a non-linear SVM with a χ^2 kernel) is trained to assign the class-labels to the global video descriptors.

Besides densely sampled trajectories, the authors evaluate baseline trajectories obtained from the KLT tracker for comparison. The same descriptors (trajectory, HOG, HOF and MBH) are used around the KLT-trajectories.

	KTH		YouTube		Hollywood2		UCF sports	
	KLT	Dense trajectories	KLT	Dense trajectories	KLT	Dense trajectories	KLT	Dense trajectories
Trajectory	88.4%	90.2%	58.2%	67.2%	46.2%	47.7%	72.8%	75.2%
HOG	84.0%	86.5%	71.0%	74.5%	41.0%	41.5%	80.2%	**83.8%**
HOF	92.4%	93.2%	64.1%	72.8%	48.4%	50.8%	72.7%	77.6%
MBH	93.4%	**95.0%**	72.9%	**83.9%**	48.6%	**54.2%**	78.4%	**84.8%**
Combined	**93.4%**	**94.2%**	**79.9%**	**84.2%**	**54.6%**	**58.3%**	82.1%	**88.2%**

Table 1: Results of dense trajectories compared to KLT-trajectories when using different feature descriptors. [45]

Other approaches used feature trajectories for action recognition by either tracking sparse spatio-temporal interest points using a standard KLT tracker [52] or by matching SIFT features [53] between consecutive frames. The results in table 1 show the superiority of dense trajectories compared to the KLT baseline. The simple trajectory descriptor yields surprisingly good results, which according to the authors confirms the importance of motion information encoded in the trajectory shapes themselves. The MBH descriptor performs significantly better than all the other descriptors. On the YouTube dataset, the advantage of using the MBH descriptor is most prominent, since the videos in this dataset contain a lot of noise from camera-motion (uncontrolled, realistic videos, often recorded by handheld cameras).

2.4.2 Action recognition with improved trajectories (2013)

Wang and Schmid [41] address a problem of their previously released *Dense Trajectories* approach. Sampling of interest points and tracking them to from trajectories uses optical flow fields between frames. Motion that results from camera movement however, also produces optical flow in the background of an image, which results in trajectories that do not convey relevant information about the ongoing action in the scene.

Wang and Schmid [41] address this problem by explicitly estimating a homography between two frames, to compensate camera motion. Results of the approach are illustrated in figure 9.

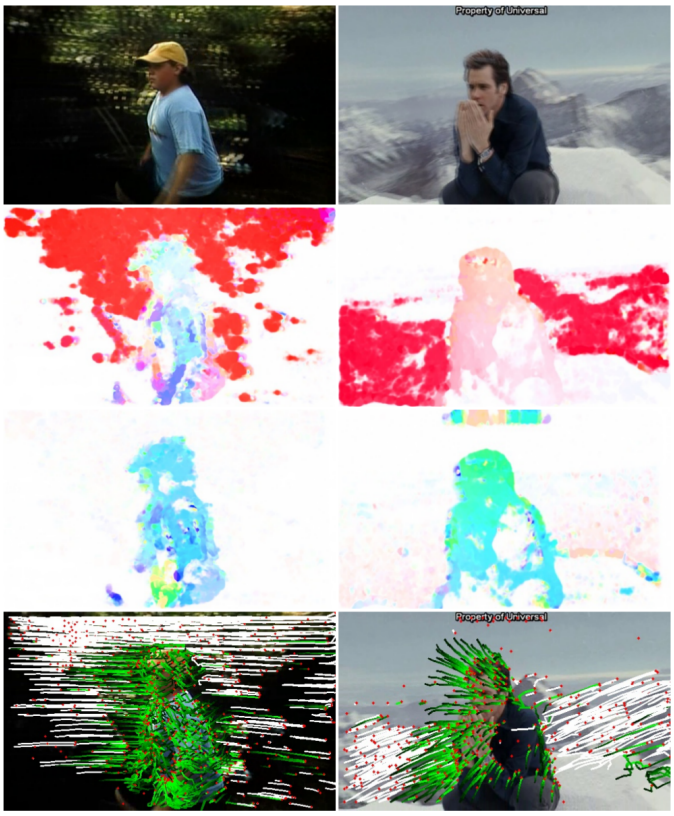


Figure 9: Improved dense trajectories: First row shows two overlaid consecutive frames, second row shows the resulting optical flow (notice the background), third row shows the optical flow after removing camera motion, fourth row shows trajectories that were removed due to camera motion estimation (white). [41]

Camera motion estimation was able to further improve the performance of dense trajectories, resulting in the approach being considered state of the art [42].

3 Deep Learning Methods in Action Recognition

This section provides a detailed review of recent action recognition approaches, that are based on deep learning. In comparison to using local spatio-temporal features, as described in section 2, deep learning algorithms do not require hand-crafted feature engineering. Although local spatio-temporal feature approaches yield competitive performance, the findings of Wang et al. [16] indicate, that there is no universally best hand-engineered set of features for all types of datasets. This suggests, that learning feature representation directly from the available data by using deep learning methods may be advantageous [54].

Similar to [8] we categorize deep learning approaches according to the underlying method of learning feature representations from input videos. In the case of neural network approaches, this corresponds to their architectures. The taxonomy is shown in figure 10. Leaf nodes correspond to specific approaches reviewed in this section.

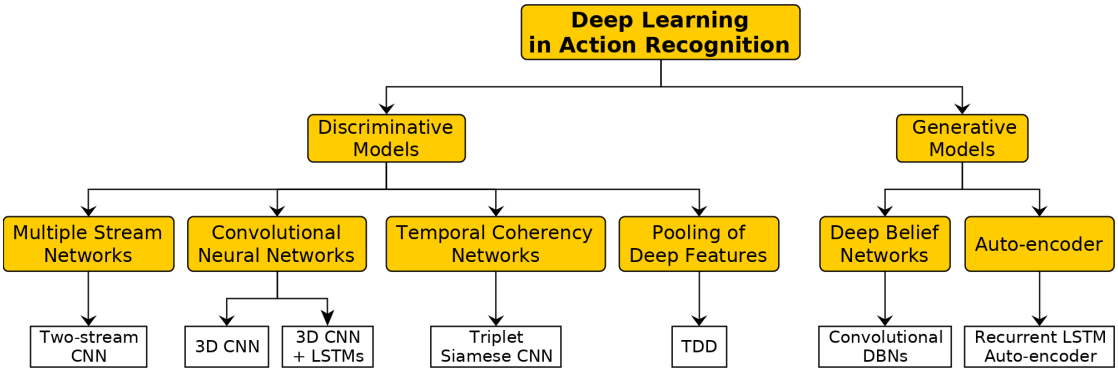


Figure 10: Taxonomy of Deep Learning approaches in Action Recognition.

Convolutional Networks

Using convolutional neural networks for action recognition is motivated by their good performance on image classification tasks. The naive approach for extending CNNs to the video domain involves applying a regular 2D CNN architecture to individual video frames and averaging the framewise class predictions over the video. This approach however ignores motion information and therefore wastes the potential of information encoded in the temporal evolution of a an action. 3D convolutions, an extension of regular 2D convolutions, are proposed to incorporate this information from video volumes. A video volume is constructed by stacking video frames and thereby treating the temporal dimension as an additional spatial dimension.

Multiple Stream Networks

These approaches use the decomposability of videos into a spatial component (analysing different frames) and a temporal component (analysing the change between frames) for action recognition. They represent one of the most successfull architectures in action

recognition [55]. The underlying principle considers, that the temporal dimension has different characteristics as the spatial dimension and should therefore be handled separately, not in a joint 3D space. A prototypical approach of multiple stream networks is represented by the two-stream architecture [44].

Temporal Coherency Networks
Temporal Coherency provides a form of weak supervision and states, that consecutive video frames are correlated semantically and dynamically [8]. To redescribe: adjacent video frames belonging to the same action class is likely and sudden changes in motion are rather unlikely. By learning to classify sequences according to their temporal coherency, i.e. whether they are in correct or incorrect temporal order, results in learning feature representations, that encode motion information of persons and object in the sequence. These representations can be used to perform action recognition.

Pooling of Deeply Learned Features
The prototypical approach of this category constructs *Trajectory-pooled Deep Convolutional Descriptors (TDD)* [55] for action recognition. TDDs are obtained by applying a generic deep feature extractor around densely sampled trajectories and pooling the resulting features for each trajectory. The approach aims at combining the advantages of hand-crafted feature methods (specifically the *Improved Dense Trajectories* approach [41]) with the expressiveness of deep architecture (specifically the *Two-Stream ConvNet* approach [44].

Generative Models are introduced in section 3.4 below.

3.1 Convolutional Networks

3.1.1 3D Convolutional Neural Networks for Human Action Recognition (2010/2013)

Ji et al. [56] propose 3D convolutions for action recognition from video with convolutional neural networks (CNNs). 3D convolutions are able to process spatial as well as temporal information in a convolutional layer.

In regular 2D CNNs the convolutional layers apply 2D convolutional kernels to the previous layers to extract features from them. More formally, in the notation of the authors, the value v_{ij}^{xy} at spatial position (x, y) of feature map j in convolutional layer i of the network is given by:

$$v_{ij}^{xy} = \tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)}\right)$$

The two inner sums over p and q carry out the convolutional operation on feature map $v_{(i-1)m}$ of the previous layer $i-1$ with the 2 dimensional convolutional kernel w_{ijm} (spatial

indices omitted). Thereby w is a tensor-like object, which contains all 2 dimensional convolutional kernels in the network, that produce feature maps through convolution Specifically, w_{ijm}^{pq} denotes the value at spatial positions (p, q) of the 2D convolutional kernel, which is applied to feature map m of the previous layer $i-1$. Feature map j in layer i is finally obtained by summing the results of all the convolutions performed on the feature maps of layer $i-1$ over m, adding a bias and passing it into a non-linear function (here $\tanh(\cdot)$).

P_i and Q_i denote the dimensions of the kernels in x and y-direction respectively. The convolutional operation used here is called *cross-correlation*, which differs from the mathematical discrete convolution in that the convolutional kernel is not flipped. This results in a non-commutative operation as described in chapter 9 of [57].

Ji et al. [56] propose an extension of 2D convolutions by using three dimensional kernels, i.e. two spatial dimensions as above and an additional temporal dimension. More formally the value v_{ij}^{xyz} of feature map j at spatio-temporal position (x, y, z) in convolutional layer i is given by:

$$v_{ij}^{xyz} = \tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}\right)$$

As above, w_{ijm}^{pqr} denotes the value of the now three dimensional kernel at spatio-temporal position (p, q, r), which performs convolution on the mth feature map of the previous layer $i-1$ to obtain feature map j in convolutional layer i. R_i additionally denotes the dimension of the kernel in temporal direction.

Based on 3D convolutions, the authors design a neural network architecture, that takes an input of 7 video-frames of size 60×40 pixels. The details of the architecture as shown in figure 11 are described below:

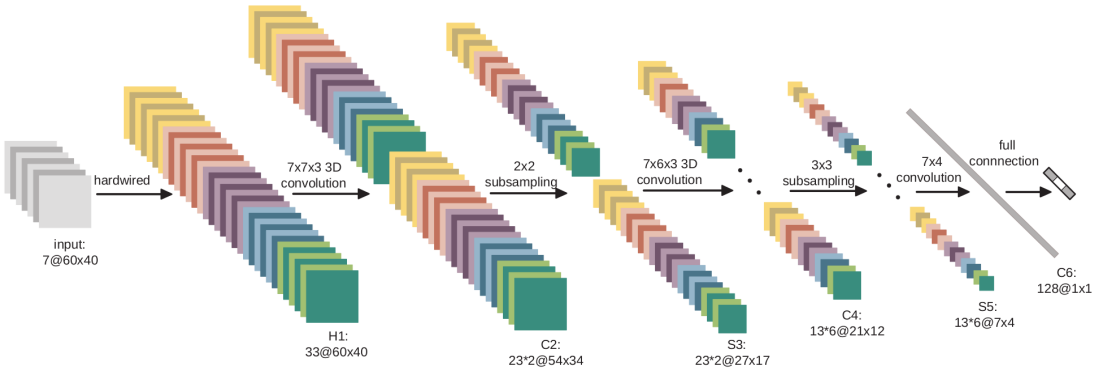


Figure 11: 3D CNN architecture developed for human action recognition [56]

The architecture contains one hard-wired layer, three convolutional layers *C2, C4* and *C6*,

two subsampling layers for dimensionality reduction *S3* and *S5* and one fully connected layer for classification.

At first, hard wired connections are applied to the input frames, in order to extract gray values of the input frames, gradients along the horizontal and vertical direction and the optical flow between two consecutive frames. This results in 33 feature maps in layer $H1$, organized in five different channels: *gray, gradient-x, gradient-y, optflow-x* and *optflow-y*.

The first convolutional layer *C2* applies two 3D kernels with dimensions $7 \times 7 \times 3$ (7×7 pixels in the spatial dimension and 3 frames in the temporal dimension) to each of the five channels in layer $H1$ separately. This results in 2×5 channels with a total of 46 feature maps in layer *C2*. The first convolutional layer therefore requires (kernel-weights) + (biases) $= 2 \times 5 \times 7 \times 7 \times 3 + 2 \times 5 = 1480$ trainable parameters.

After subsampling layer *S3*, three different 3D kernels with size $7 \times 7 \times 3$ are applied to each of the 2×5 channels of the previous layer to get the feature maps in layer *C4*. The last convolutional layer *C6* performs 2D convolutions to obtain 128 feature maps of dimension 1×1.

The resulting vector of these 128 feature maps in layer $C6$ is interpreted as a 128-dimensional feature representation of the input. This feature representation is then classified by a fully connected layer into the required number of output classes.

In total the architecture contains $295,458$ trainable parameters, which are initialized randomly and learned by online error back-propagation as done by LeCun et al. [58]. The authors tried other architectural layouts but conclude that the one described above works best.

The network is evaluated as part of an action detection and recognition system, using the TRECVID 2008 development dataset [59]. Additionally, it's performance is measured as stand-alone approach on the KTH benchmarking datset [15].

Evaluation on TRECVID:
The TRECVID 2008 development dataset [59] contains 49 hours of surveillance videos continuously recorded by five different cameras on five days at London Gatwick Airport. The dataset is annotated for surveillance event detection, i.e. what events occur where. The authors train their model to recognize three action classes from the dataset: *CellToEar*, *ObjectPut* and *Pointing*. The other action classes provided by the dataset were used to generate negative training examples. The distribution of training examples sorted by date is shown in figure 12. Example frames of the dataset are shown in figure 13.

Date\Class	CellToEar	ObjectPut	Pointing	Negative	Total
20071101	2692	1349	7845	20056	31942
20071106	1820	3075	8533	22095	35523
20071107	465	3621	8708	19604	32398
20071108	4162	3582	11561	35898	55203
20071112	4859	5728	18480	51428	80495
Total	13998	17355	55127	149081	235561

Figure 12: Number of action samples per class from the TRECVID 2008 development dataset [56]

Since the dataset provides continuous videos with several persons in a real-world scene, Ji et al. [56] apply a human detector and a detection-driven tracker, to keep track of the heads in the scene. This information is used to extract a bounding box around a person, as soon as an action is performed. Six additional bounding boxes are sampled from three frames before and three frames after the action was detected with a temporal step size of two frames. The bounding boxes have the same size and are sampled at the same spatial location as the initial bounding box.

The contents of these 7 bounding boxes are stacked and used as input of the 3D CNN architecture in order to classify the performed action.

Figure 13: Example scenes of the TRECVID 2008 development dataset with results of human detection and tracking [56]

To evaluate the performance of the 3D CNN model, the authors compare it to three other baseline approaches in the detection and recognition system:

1. A frame-based 2D CNN model, which averages the action class predictions over individual frames.
2. Extraction of dense SIFT features [53] from the seven gray scaled input frames, which are then aggregated using the BoW-Paradigm and classified through a linear SVMs.
3. Extraction of dense SIFT features [53] from motion edge history images (MEHI)[60] of the input frames, which are then aggregated as above.

The 3D CNN model outperforms the other approaches on the TRECVID 2008 development dataset significantly for all classes except *Pointing*, where the 2D frame-based CNN performed best. The authors note, that the number of training examples for *Pointing* are significantly larger than for any other class and conclude, that their architecture performs best, when few positive examples are present.

Evaluation on KTH:
Furthermore, the stand-alone 3D CNN architecture was evaluated on the KTH dataset [15]. It achieves an overall accuracy of 90.2% on that benchmark. In comparison: Schindler and Van Gool [61] achieved 92.7% and Jhuang et al. [62] achieved 91.7% several years earlier.

Ji et al. [56] showed, that 3D convolutions yield competitive performance compared to state-of-the-art approaches at that time as shown on the KTH benchmark [15]. Although the authors reference the work of Schindler and Van Gool [61] which states, that 5-7 video-frames are enough for recognizing simple actions, this short temporal extend is often identified as a deficit of the approach and addressed in following approaches, e.g. by Baccouche et al. [63].

3.1.2 Sequential Deep Learning for Human Action Recognition (2011)

Baccouche et al. [63] identify two deficits of previous approaches for extending CNNs to the video domain, specifically the approach of 3D convolutions such as [56] and [64]:

1. They still rely on hand crafted inputs (hard wired pre-processing of the data to produce image gradients and optical flow in the first processing layer).
2. The models typically process less than 15 input frames, and therefore only classify short sub-clips, not the entire video.

To address these issues, the authors design a two-step deep architecture, which is shown in figure 14 and consists of:

1. A convolutional spatio-temporal feature extractor network, based on 3D convolutions.
2. A recurrent neural network (RNN) classifier, which incorporates LSTM cells [65] to classify the entire sequence of previously extracted spatio-temporal features.

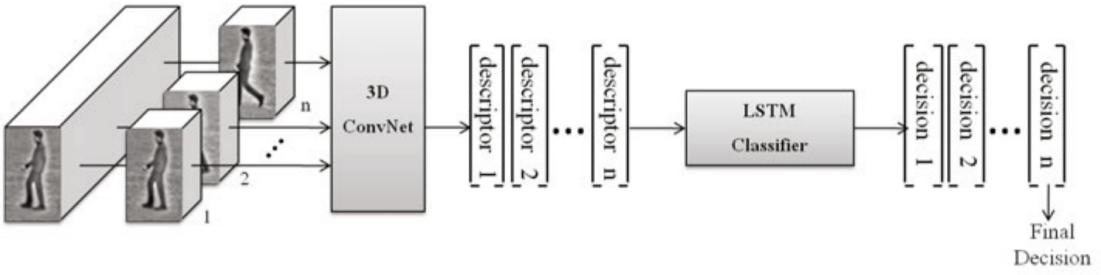


Figure 14: Overview of the two-step architecture consisting of a 3D ConvNet and a recurrent neural network classifier. [63]

Baccouche et al. [63] incorporate the work of Ji et al. [56] by using a 3D ConvNet as feature extraction stage, which processes raw pixel values instead of hand-crafted inputs. To compensate the problem of temporally short inputs and to take the temporal evolution of movements during an action into account, a RNN classifier is added, because they are able to process input sequences of arbitrary length. The RNN classifies a sequence of feature representations, previously extracted by applying the 3D ConvNet to temporally adjacent patches of the input video, in order to recognize an action. Similarly to [56], the overall approach is evaluated on the KTH dataset [15].

Details of the 3D ConvNet architecture are shown in figure 15.

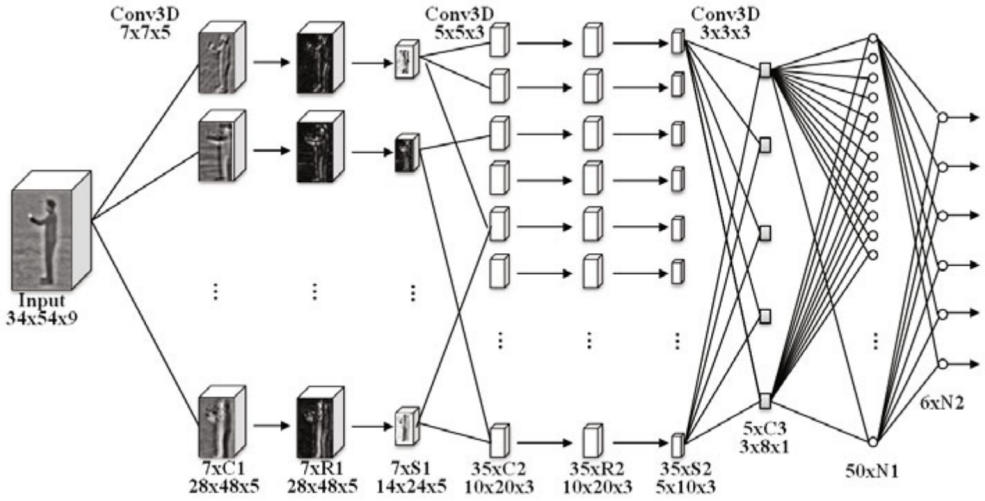


Figure 15: Detailed architecture of the 3D ConvNet for later use with an RNN classifier. [63]

The input to the 3D ConvNet is formed by stacking 9 successive frames from the input video of spatial resolution 34×54 pixels. The network contains three convolutional layers *C1, C2* and *C3*. The first two convolutional layers are followed by rectification and subsampling layers *R1, S1* and *R2, S2*. A rectification layer simply computes the

absolute value of its input [63]. The third convolutional layer is followed by two neuron layers (fully-connected layers) *N1* and *N2*.

The ConvNet model, as shown in figure 15 embeds $17,169$ trainable parameters in total, which is about 15 times less than the $295,458$ parameters used by Ji et al. [56].

Specifically, the layers are configured as follows:

1. Convolutional layer *C1* computes 7 feature maps, by convolving 7 3D $7 \times 7 \times 5$ kernels with the stacked input frames.
2. Layer *R1* and *S1* perform rectification (building of the absolute value) and sub-sampling with a spatial factor of 2 respectively.
3. Convolutional layer *C2* computes 35 feature maps (the 7 feature maps in layer *S1* are connected to two different convolutional kernels, which results in 14 feature maps and pairs of different feature maps in *S1* are connected to one convolutional kernel each, which results in additional 21 feature maps, summing to a total of 35 feature maps).
4. Convolutional layer *C3* computes 5 features maps, which are fully connected to all feature maps in previous layer *S2* by $3 \times 3 \times 3$ convolutional kernels. These five feature maps have dimension $3 \times 8 \times 1$, rendering the raw input encoded as a 120 dimensional feature vector.

The 3D ConvNet is trained individually on the KTH dataset before employing the RNN classifier. For training, the 120 dimensional feature vector is fed into two fully connected layers *N1* and *N2* with 6 output neurons, one for each class of the KTH dataset. The authors use the same training algorithm as Ji et al. [56]: online backpropagation with momentum adapted to weight-sharing.

For training the RNN classifier with online backpropagation through time [66], the fully connected layers *N1* and *N2* of the 3D ConvNet are removed. The 120 output values of the third convolutional layer *C3* are fed into the recurrent neural network as input at each time step. Several configurations were tested by the authors and a single hidden layer with 50 LSTM cells were found to be a good compromise between training time and performance. The LSTM cells are fully connected to the outputs of layer *C3*.

The authors find their 3D ConvNet model alone, without adding the recurrent LSTM classifier, to yield a recognition rate of 91.04% on KTH, when the classification is done by majority voting over several short sub-sequences of the test-video. This result is comparable to other approaches at that time and almost the same as obtained by Ji et al. [56] (90.2%), although the model requires about 15 times less parameters. When using the LSTM classifier network, recognition performance increases to 92.17%.

3.1.3 Large-scale Video Classification with Convolutional Neural Networks (2014)

Karpathy et al. [11] provide a comprehensive examination of techniques to apply convolutional neural networks to the domain of action recognition from video. More specifically, their contribution is four-fold:

1. They gather the Sports-1M dataset containing a collection of 1 million automatically annotated sports videos from YouTube (further described in section 4 of this work).
2. They evaluate several approaches besides three dimensional convolutions for extending CNNs to process spatio-temporal data in a consistent and therefore comparable manner. These methods are called *fusion methods* in the phrasing of the authors, since temporal motion information has to be fused with spatial motion information while propagating through a network.
3. They propose a generic multi-resolution convolutional architecture in oder to speed up training time at no cost in accuracy.
4. They retrain the top layers of a network on the UCF-101 dataset, which has previously been trained on the Sports-1M dataset and thereby achieve a significant increase in performance against training the network on UCF-101 alone (transfer learning).

The authors first implement a baseline CNN architecture, which classifies human action videos by processing a single frame at a time, i.e. without considering temporal relationship between frames. Based on the single frame architecture, several extensions for processing temporal information in the network are being investigated. These types of fusion methods are depicted in figure 16.

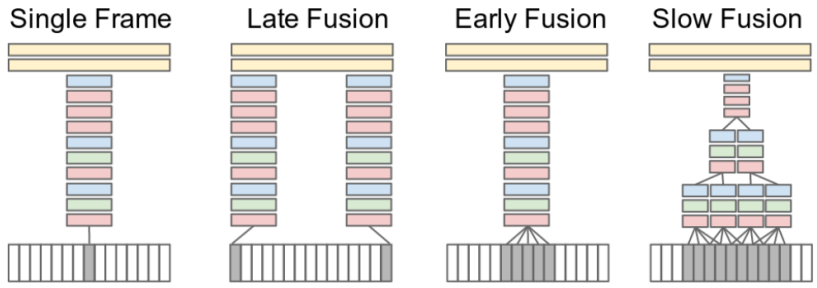


Figure 16: Different methods for fusing temporal with spatial information in convolutional neural network architectures. Red, green and blue denotes convolutional, normalization and pooling-layers. Grey colored inputs are single RGB video frames. Yellow layers are fully-connected for classification. [11]

Single Frame

The CNN model used in the single frame approach is a deep convolutional neural network

with 2D convolutions, that recognizes actions by classifying frames of a given input sequence individually and reporting the averaged prediction. The architecture is similar to AlexNet, which won the 2012 ImageNet Classification Challenge [67], but receives slightly smaller inputs: $170 \times 170 \times 3$ instead of $224 \times 224 \times 3$ in the ImageNet model. The first two dimensions thereby correspond to the spatial resolution of the video, the third dimension to the RGB color channels of a video-frame. This approach is evaluated as a baseline in order to measure the improvement when using temporal information, i.e. several frames, in the recognition process.

Early Fusion
In the early fusion approach temporal information is incorporated in the network on the pixel level by extending the convolutional kernels in the first layer to be of dimension $11 \times 11 \times 3 \times T$, where T is the temporal extend, i.e. the number of input frames to the network. The authors set $T = 10$ which corresponds to a third of a second. Note that the temporal information is completely flattened after the first convolutional layer, since the kernel has the same temporal dimension as number of inputs frames. Therefore only the kernels in the first convolutional layer are three dimensional in nature.

Late Fusion
In the late fusion methods, two separate single-frame networks with shared parameters and without their individual classification layers are used on input frames with a temporal distance of 15 frames. Two shared fully connected layers then merge the individual network's information and classify the input. The fully connected layers are able to compute motion information by comparing the feature representations of the two single-frame networks.

Slow Fusion
Temporal information is processed throughout the network by extending the kernels of each convolutional layer in time, as done in the first layer of the *Early Fusion* approach. Thereby higher layers progressively process more temporal information along the input frames. The *Slow Fusion* approach applies 3D convolutions as done by Ji et al. [56] and Baccouche et al. [63]. The first convolutional layer incorporates a temporal extend of $T = 4$ in its kernels, while the second convolutional layer uses a temporal extend of $T = 2$. This allows the third convolutional layer layer to access the information of all 10 input frames.

Evaluation on the Sports-1M dataset
The recognition performance of the different fusion methods is evaluated on the Sports-1M dataset. The authors use downpour stochastic gradient descent [68] for training the models in a distributed way on a computing cluster. For the evaluation 70% of the dataset were used as training data, 10% as validation set and 20% as test set.

To obtain fixed-sized inputs for the models, the authors interpret an entire video as a set of short, fixed-sized video clips. At test time, 20 short clips are sampled from the current test-video and each clip is presented to the network individually. Each clip is passed through the network 4 times, each time using different crops and flips and the result

is averaged to produce a robust class prediction. The video-level predictions (examples shown in figure 17) are computed from the clip-level predictions simply by averaging.

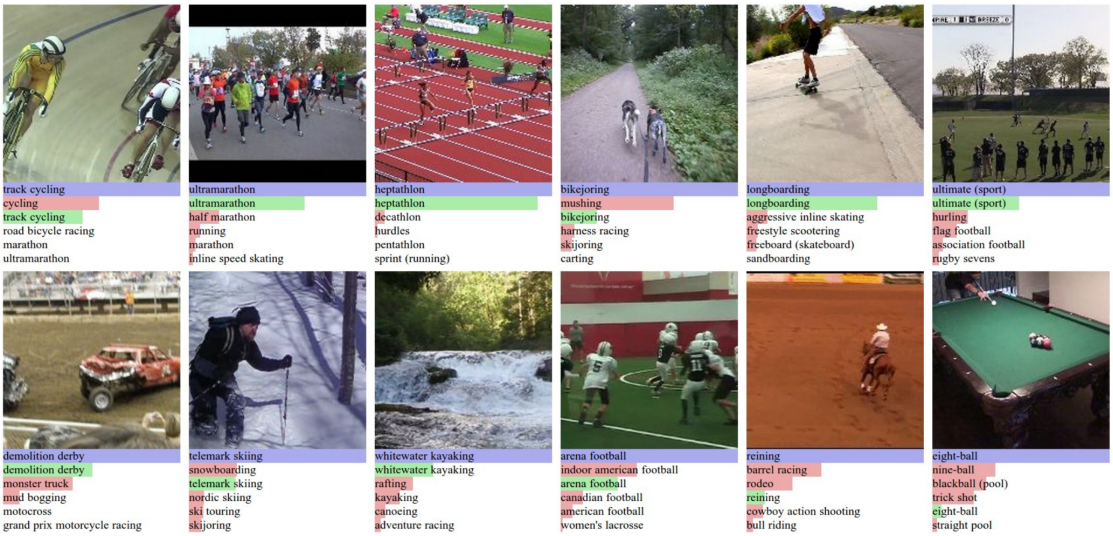


Figure 17: Classification of videos in the Sports-1M dataset. Blue label is the ground truth, below class predictions are shown with decreasing confidence. [11]

In addition to comparing different fusion methods, the authors also implement a hand-crafted feature baseline approach, that extracts multiple kinds of hand-crafted local features from each video and aggregates them according to the bag-of-words paradigm. The resulting feature histogram representations are classified into action classes using a multilayer neural network, with rectified linear activation units.

The performance of the studied approaches is shown in table 2:

Model	**Clip Hit@1**	**Video Hit@1**	**Video Hit@5**
Feature Histograms + Neural Net	-	55.3	-
Single-Frame	41.1	59.3	77.7
Single-Frame + Multires	**42.4**	**60.0**	**78.5**
Single-Frame Fovea Only	30.0	49.9	72.8
Single-Frame Context Only	38.1	56.0	77.2
Early Fusion	38.9	57.7	76.8
Late Fusion	40.7	59.3	78.7
Slow Fusion	**41.9**	**60.9**	**80.2**
CNN Average (Single+Early+Late+Slow)	41.4	63.9	82.4

Table 2: Results of different architectures on the Sports-1M dataset. Hit@k denotes the percentage of test samples, that had at least one of their class labels included in the top k predictions. [11]

The results show, that the deep models (*early*, *late* and *slow-fusion*) consistently outperform the hand-crafted feature baseline. Compared to each other, the deep models perform similarly well, despite their different convolutional architectures. The *slow fusion* model, that uses 3D convolutions in all convolutional layers, outperforms the other fusion approaches by a small margin. Karpathy et al. [11] describe the performance difference of between fusion models as "surprisingly insignificant"[11]. The single-frame model performs noticeably well on it's own. The authors suspect, that the motion aware networks suffer from camera movements such as translations or zoom.

Since the training time of a network heavily influences the amount of evaluations that can be conducted using different hyperparameter settings, it is of great interest to reduce the training time for neural networks, which lies in the in the order of weeks for CNNs[11], while maintaining accuracy.

The authors therefore propose a multi-resolution architecture, that is shown in figure 18.

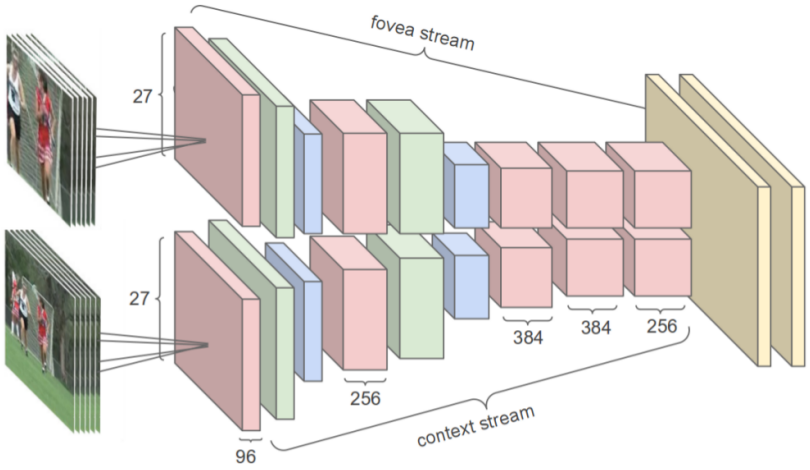


Figure 18: Multi-resolution CNN architecture [11]

The multi-resolution network processes input videos with a resolution of 178×178 pixel. The context stream receives a downsampled version of the input frames, which contain the complete field of view but have a decreased resolution of 89×89 pixel. The fovea stream works on just the 89×89 pixel sized center of the original input frames. Both streams are implemented as the same CNN architecture and the outputs of both streams are fed into two fully connected layers, where their information is merged and a class prediction is calculated.

The input dimensionality of the multi-resolution architecture is decreased by a factor of 2 compared to processing the raw 178×178 pixel input video in one stream. The authors achieved a reduction in training time by a factor of 2-4, while maintaining the accuracy of the system (see table 2).

3.1.4 Learning Spatiotemporal Features with 3D Convolutional Networks (2015)

Tran et al. [42] create a generic spatio-temporal feature extractor, by training a very deep 3D ConvNet on the large-scale action recognition dataset Sports-1M [11]. Given a human action video as input, the activations of the ConvNet's last fully connected layer form a descriptive feature vector, which represents the input. The authors show, that the extracted feature representation, which they call *C3D*, is generic enough to be reused on other vision tasks without requiring task-specific fine tuning of the model. Merely a classifier (the authors use linear SVMs) has to be trained on the extracted *C3D* features, given a video-based vision task. The approach is evaluated for action recognition, action similarity labeling (ASLAN, see section 4 of this work), scene classification and object recognition.

Using a model based on 3D convolutions for *C3D* is motivated by the recent review of fusion methods, published by Karpathy et al. [11], which showed that using 3D convolutions in all convolutional layers of a CNN performs best for action recognition (*Slow Fusion*). The authors conclude that 3D convolutions preserve temporal information along the processing stages of a deep CNN, because they result in a 3 dimensional feature map. In contrast 2D convolutions the temporal relations among frames after each convolutional layer by only processing frames spatially. This is illustrated in following figure 19.

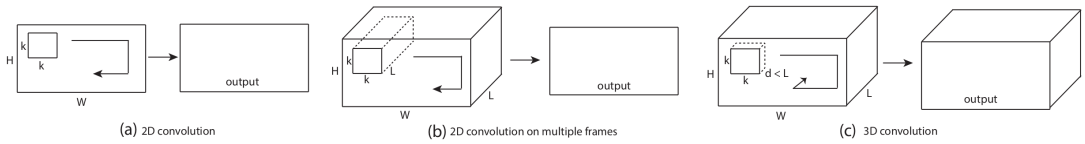


Figure 19: Dimensionality of convolutions: a) 2D convolution on a single image results in a 2D feature map. b) 2D convolution on multiple frames, interpreted as different channels of the input, results in a 2D feature map because results of individual frames are summed. c) 3D convolution on multiple frames results in a 3D feature map, therefore preserving the temporal dimension [11]

All networks take inputs of dimension $128 \times 171 \times 16$ (*frame width* $\times$ *frame height* $\times$ *number of frames*, in three color channels each). To find the best performing architecture, the authors first perform experiments on the medium-scale dataset UCF-101 [69], because using a large-scale dataset would be too time-consuming. More specifically, the network architecture for performing these experiments is designed as follows:

- 5 convolutional layers with 64, 128, 256, 256 and 256 filters respectively.
- Each convolutional layer is followed by a max-pooling layer with filter size $2 \times 2 \times 2$ (except for the first layer, which has a filter size of $1 \times 2 \times 2$ for not collapsing the temporal information too early).
- Two fully connected layers with 2048 neurons each at the end and an additional softmax layer with one output per action class for training.

The work of Simonyan and Zisserman [2], regarding 2D convolutional neural networks, suggests that 3×3 convolutional kernels yield best results in deep architectures . The authors therefore fix the spatial kernel size in their 3D convolutions to 3×3 and vary the temporal depth of the kernel d. For finding the best parameter, the depth d is varied according to the following two methods, results are shown in figure 20:

1. Homogeneous temporal depth: The kernels across all convolutional layers have the same temporal depth. Four networks are evaluated with $d \in \{1, 3, 5, 7\}$.
2. Varying temporal depth: The temporal depth changes across the convolutional layers. Two schemes are being evaluated. Increasing temporal depth $3-3-5-5-7$ and decreasing temporal depth $7-5-5-3-3$.

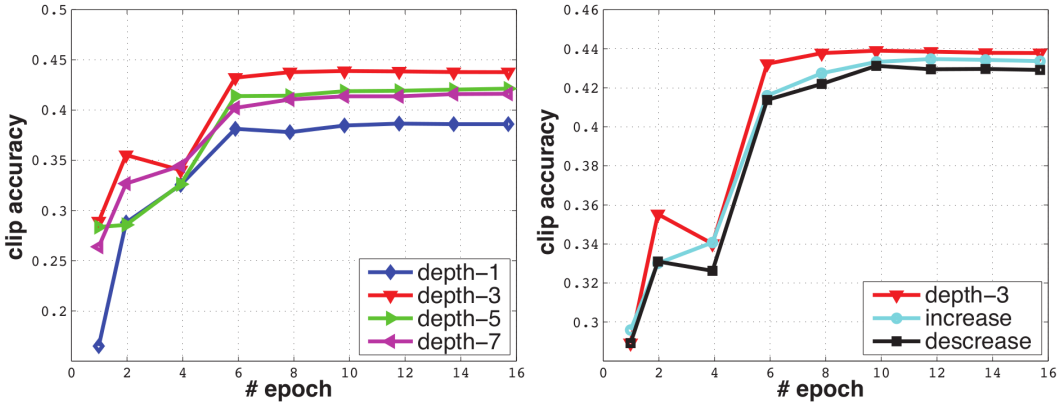


Figure 20: Clip accuracy for different temporal kernel depths over training epochs on UCF101. Left shows homogeneous temporal depth, right shows varying temporal depth. [42]

The results indicate, that a fixed temporal depth of 3 performs best among all tested settings. Testing a spatial kernel resolution of 5×5 with the same variations for d yielded the same result. The authors therefore conclude that $3 \times 3 \times 3$ kernels are the best choice in 3D convolutional networks.

The final architecture for extracting C3D features with $3 \times 3 \times 3$ convolutional kernels is constructed as follows: 8 convolutional layers, 5 pooling layers, two fully connected layers and a softmax output layer. With 8 convolutional layers and 15 layers in total (without counting the softmax output-layer, as illustrated in figure 21), it is the deepest architecture reviewed for the action recognition task in this work.

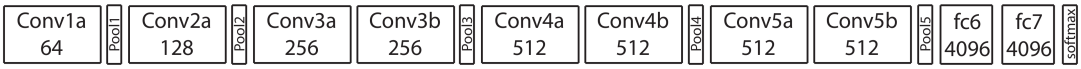


Figure 21: C3D architecture. Number of filters is given in each box. [42]

Training The network is trained on the Sports-1M [11] dataset to learn spatio-temporal features. Five two second long clips are extracted randomly from every training video and resized to a resolution of 128×171 pixel. From these five clips, training inputs for the network of size $112 \times 112 \times 16$ are randomly extracted. Training is conducted with stochastic gradient descent with batch size of 30.

Testing For testing, i.e. classification of unseen videos from the test set, video clips are extracted from a test video as during training. A single crop of the center of a video-clip is passed through the network to obtain a clip-classification. For obtaining video-classifications, the classifications of 10 clips are being averaged.

The *C3D* approach yields video hit@5 accuracy on the Sports-1M dataset of 84.4% and video hit@1 accuracy of 60.0%. The authors also try to boost the performance by pre-training the model on an internal dataset (called I380K) and fine-tuning on Sports-1M, which results in a hit@5 accuracy of 85.5%.

Additionally, the performance of *C3D* features was evaluated on UCF-101, which results in an accuracy of 82.3% when applying a single *C3D* network. Combining the extracted features of three networks before classifying them increases the accuracy to 85.2%.

Note, that these results were obtained by using a simple linear SVM classifier on top of the *C3D* features. Results can be improved by applying more sophisticated aggregation schemes [42].

To promote the usage of *C3D* features, the authors forked from the Caffe framework [70], extended it for handling three dimensional convolutional networks and therein provide a pre-trained *C3D* network model for public use. The usage of *C3D* features is therefore simple and effective, since a trained model is publicly available and the forward pass in neural networks is computationally fast.

3.1.5 Long-term Temporal Convolutions for Action Recognition (2016)

Varol, Laptev, and Schmid [71] address, similar to Baccouche et al. [63], the common deficit in recent CNN extensions to action recognition of class labels being learned from very short video subsequences only. Most often a temporal extend of only 1-16 input frames can be processed by the architectures at a time. [56][11][42]

Instead of using a recurrent neural network for classifying convolutionally extracted spatio-temporal features, as done by Baccouche et al. [63], Varol, Laptev, and Schmid [71] study the effects of increasing the number of input frames to a 3D convolutional architecture in the context of action recognition from video. The authors name their approach long-term temporal convolutions and their contribution is two-fold:

1. A systematical evaluation of the influence of the number of input frames $T = \{20, 40, 60, 80, 100\}$ on the performance of a 3D CNN architectures.

2. A demonstration of the importance of high-quality optical flow inputs, in order to learn accurate video features with a 3D CNN architecture.

Processing an increased temporal extend has to be compensated with a decreased spatial resolution, in order to not exceed computational limitations. The studied architectures therefore process spatial resolutions of 58×58 or 71×71 pixel only. The used 3D CNN architecture is shown in figure 22. T therein denotes the temporal extent, i.e. the number of input frames to the ConvNet, and architectural details are given below.

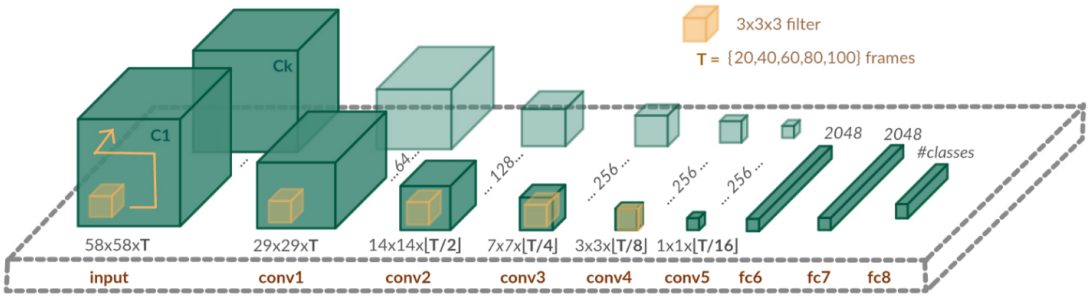


Figure 22: 3D CNN architecture for evaluating the influence of the number of input frames on action recognition accuracy. [71]

- 5 convolutional layers *conv1*, ..., *conv5*, that apply 3D convolutions to their previous layers and contain 64, 128, 256, 256 and 256 feature maps respectively.
- Kernel size of $3 \times 3 \times 3$ in each convolutional layer.
- After each convolutional layer: One layer of rectified linear units (ReLUs) and one layer of max-pooling with filter size $2 \times 2 \times 2$ except in the first layer where it is $2 \times 2 \times 1$ (not shown in the figure).
- Three fully connected layers as classifier with sizes 2048, 2048 and number of classes. The fully connected layers also use rectified linear units as activation function and an additional softmax layer at the end of the network.

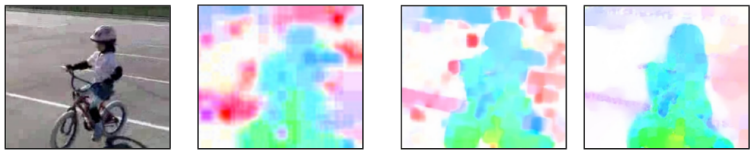
As a baseline approach the authors first evaluate their architecture with inputs of size $112 \times 112 \times 16$, that is a spatial resolution of 112×112 and $T = 16$ input frames, because it can be directly compared to the work of Tran et al. [42]. Additionally, the baseline is implemented with $T = 60$ frames inputs, specifically using an input dimension of $58 \times 58 \times 60$, to provide an initial comparison between 16 and 60 frames (spatial resolution is decreased to remain computationally tractable).

The two baseline models are used to study the influence of different input modalities, network settings and data augmentation techniques such as: optical flow inputs, *dropout*[72], *random clipping* and *multiscale cropping* (explained below).

At first the influence of using optical flow as input is evaluated, specifically stacked optical flow images from the following sources:

1. MPEG flow [73], which directly can be obtained from the video encoding. It is a fast method compared to regular optical flow estimators, but has low spatial resolution and is not available for every video frame.
2. Farneback optical flow estimator [48], which is fast, but calculates noisy flow fields.
3. Brox optical flow estimator [74], which generates high quality optical flow fields but is slower than the other two methods.

Example flow estimation of a given RGB video frame for all three methods is illustrated in figure 23 (left). The table on the right of figure 23 shows the action recognition accuracy of the 60 frame baseline network, when using optical flow or pure RGB frames as input. Results are obtained on UCF-101 (split 1).



Input	Clip	Video
RGB	57.0	59.9
MPEG flow	58.5	63.8
Farneback	66.3	71.3
Brox	74.8	79.6

Figure 23: Optical flow estimation of different algorithms and corresponding results when used as input for the 60 frame baseline network. Color encodes the direction of the optical flow field. [71]

The results show, that high quality optical flow can boost the action recognition performance by nearly 20% against using raw RGB video frames. Even using low quality MPEG flow outperforms regular RGB inputs. The authors therefore conclude, that using high quality optical flow as input is critical for learning competitive human action features from video.

Training inputs can generated by sampling video subsequences with the desired spatial and temporal dimensions from input videos at random spatio-temporal locations. The authors call this form of pre-processing *random clipping*.

The baseline approach is using a sliding-window approach with 75% overlaps to generate fixed length video-clips.

An additional method for pre-processing suitable network inputs during training is called *multiscale cropping*. Input volumes, which are smaller than needed, are cropped from the training videos and then rescaled to fit to the input dimensions of the network. The size of the crop is determined by random factors for frame width and height, sampled from $\{1.0, 0.875, 0.75, 0.66\}$. Additionally, the scaled input is flipped with a probability of 50%.

Dropout [72] can be used during training independently of the selected cropping method.

Accuracy results for using the pre-processing methods *random clipping*, *multiscale cropping* and *dropout* are shown in table 3 below. Best performance is obtained when combining the two methods with a high dropout rate.

Random clipping	**Multiscale cropping**	**Dropout**	**Clip acc. (%)**	**Video acc. (%)**
-	-	0.5	71.6	76.5
✓	-	0.5	74.8	79.6
-	✓	0.5	72.5	78.1
-	-	0.9	74.4	78.5
✓	✓	0.9	76.3	80.5

Table 3: Evaluation of pre-processing methods and dropout on a 60 frame input network, trained on UCF-101 (split1) from scratch using Brox optical flow as input [71]

The two baseline architectures, which differ in the number of input frames, are evaluated on the UCF-101 dataset. In all experiments *random clipping* is used. Results are reported in table 4 for RGB and optical flow inputs as well as optional *multiscale cropping* and different dropout ratios. The results show that long-term temporal convolutions, as present in the 60 frame network, persistently outperform the 16 frame counterpart with a smaller temporal extent.

Input	**Multiscale cropping**	**Dropout**	**Clip acc. (%)**			**Video acc. (%)**		
			16f	**60f**	**gain**	**16f**	**60f**	**gain**
RGB	-	0.5	48.4	**57.0**	+ 8.6	51.9	**59.9**	+ 8.0
Flow	-	0.5	66.8	**74.8**	+ 8.0	77.4	**79.6**	+ 2.2
Flow	✓	0.9	67.1	**76.3**	+ 9.1	78.7	**80.5**	+ 1.8

Table 4: Action recognition accuracy of the two baseline architectures, evaluated on UCF-101 (split1). **16f** corresponds to the architecture with 16 frame inputs, **60f** corresponds to 60 frame inputs. [71]

The authors note, that the complexity of both networks, i.e. the number of trainable parameters, is similar since the 60 frame network processes frames at reduced resolution.

The networks are also evaluated on the HMDB-51 dataset. Both networks are either trained from scratch, i.e. with randomly initialized inputs, or have been pre-trained on UCF-101, which is much bigger than HMDB-51 (see section 4). Results are shown in table 5 below.

Pre-training on UCF101	Clip acc. (%)			Video acc. (%)			
	16f	60f	gain	16f	60f	gain	Two-stream
-	37.0	**52.6**	+ 15.6	43.9	**52.9**	+ 9.0	46.6
✓	40.6	**56.1**	+ 15.5	48.3	**57.1**	+ 8.8	49.0

Table 5: Action recognition accuracy of the two baseline architectures, evaluated on HMDB-51 (split1) with optional pre-training on UCF-101. **16f** corresponds to the architecture with 16 frame inputs, **60f** corresponds to 60 frame inputs. [71]

The results again indicate, that using long-term temporal convolutions leads to significant increase in action recognition accuracy and that pre-training the networks on a larger dataset yields significant improvement over training it from scratch.

In order to systematically investigate the influence of temporal extent on action recognition performance, the authors implement their architecture with different numbers of input frames $T = \{20, 40, 60, 80, 100\}$ and different spatial resolutions $\{58 \times 58, 71 \times 71\}$. Results are shown in the following figure 24.

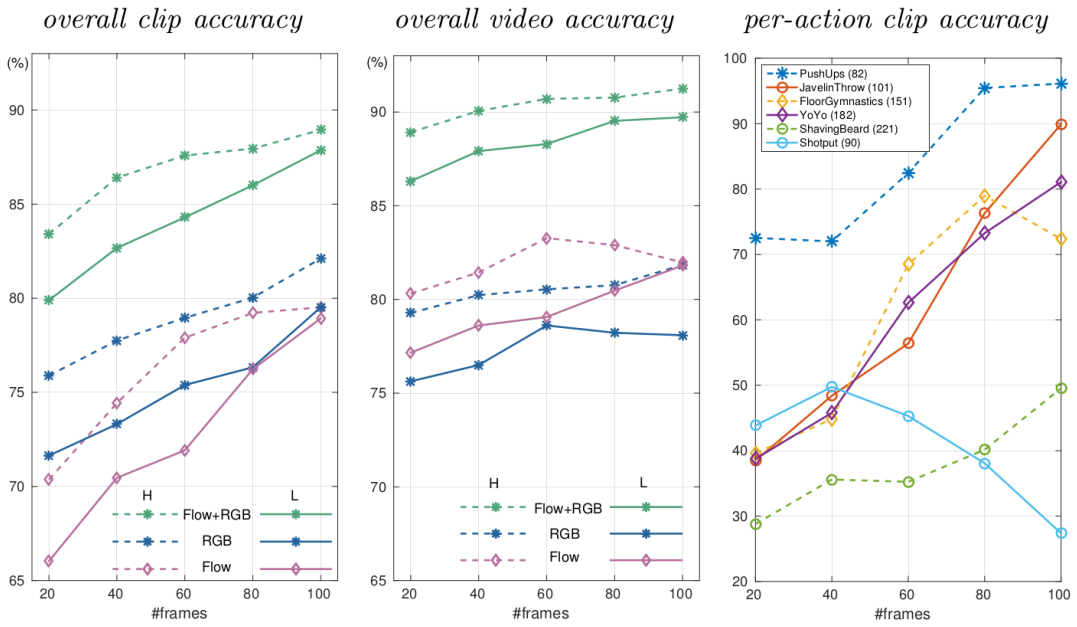


Figure 24: Action recognition performance of long-term convolutional networks for varying number of input frames, evaluated on UCF-101 (split1). H denotes high resolution (71×71 pixel), L denotes low resolution (58×58 pixel). [71]

To conclude, Varol, Laptev, and Schmid [71] show, that longer temporal extent increases the action recognition accuracy of 3D convolutional neural network for action recognition.

3.2 Multiple Stream Networks

3.2.1 Two-Stream Convolutional Networks for Action Recognition in Videos (2014)

Simonyan and Zisserman [44] propose a novel architecture for action recognition with separate spatial and temporal recognition streams, which are fused late (see figure 25. This approach is motivated by the two-streams hypothesis [75], according to which the human visual cortex contains two paths: the ventral stream for object recognition and the dorsal stream for recognising motion.

The authors evaluate two different fusion methods: building the average of both network's outputs and training a linear multi-class SVM on them. Both streams are implemented as deep CNNs, with rectification activation function for all hidden units.

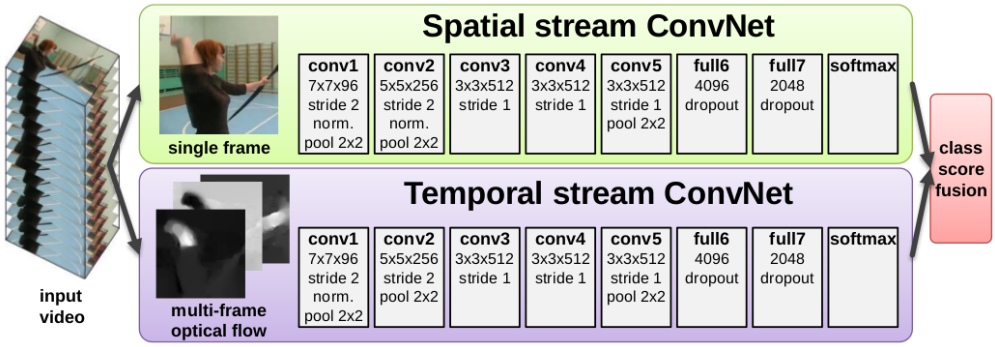


Figure 25: Two-stream architecture for video classification [44]

The spatial stream takes single video frames as input and is fairly competitive on its own. Since it is basically an image recognition architecture, it performs action recognition from still video frames. The spatial part of a video, i.e. the individual static frames, conveys information about the objects and persons in the scene.

The temporal stream is trained to recognize actions from motion given in the form of optical flow. The temporal part of a video, i.e. the difference between adjacent static frames, conveys information about the movement of the observer (camera) and the movement of objects in the scene. The authors propose two methods for constructing the input to the temporal network by stacking optical flow displacement fields along several consecutive frames of the input video.

Optical Flow Stacking:
A dense optical flow field $\mathbf{d}_t(u, v)$ of two consecutive frames at times t and $t+1$ can be thought of as a two dimensional vector-field, which maps the displacement of each pixel along the transition from frame t to $t+1$. In this case $u, v \in \mathbb{N}$ correspond to a position in the frame, $1 \leq u \leq w$ and $1 \leq v \leq h$, where w and h are the width and height of the video frames.

The horizontal and vertical components $d_t^x(u,v)$ and $d_t^y(u,v)$ can be interpreted as image channels.

This method constructs the input volume $I_\tau \in \mathbb{R}^{w \times h \times 2L}$ of the temporal stream network by stacking the horizontal and vertical components of the dense optical flow field along L consecutive frames, beginning at time τ. Formally, with $1 \leq k \leq L$:

$$I_\tau(u,v,2k-1) = d^x_{\tau+k-1}(u,v)$$
$$I_\tau(u,v,2k) = d^y_{\tau+k-1}(u,v)$$

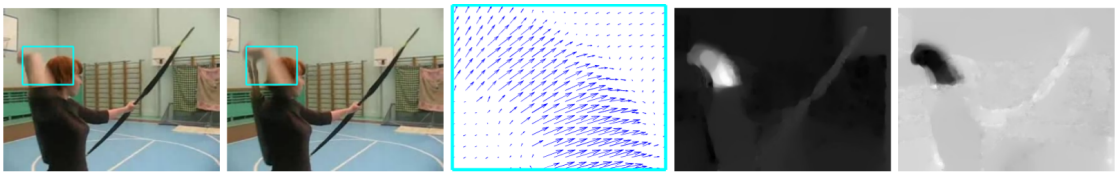


Figure 26: Optical flow between two frames (a) and (b). (c) shows the dense optical flow field, i.e. displacement vectors, of the turquoise region. (d) and (e) illustrate the x and y components of the flow field as images. [44]

Trajectory Stacking:
Instead of sampling at fixed locations in each frame, trajectory stacking samples the dense optical flow field along motion trajectories of the initial points in frame τ. Let $\mathbf{p}_k$ denote the motion trajectory of initial point (u,v). With $1 < k \leq L$ and $\mathbf{p}_1 = (u,v)$ the trajectory is recursively defined by:

$$\mathbf{p}_k = \mathbf{p}_{k-1} + \mathbf{d}_{\tau+k-2}(\mathbf{p}_{k-1})$$

The optical flow input volume I_τ can then be constructed by sampling the horizontal and vertical optical flow components along these trajectories. Specifically:

$$I_\tau(u,v,2k-1) = d^x_{\tau+k-1}(\mathbf{p}_k)$$
$$I_\tau(u,v,2k) = d^y_{\tau+k-1}(\mathbf{p}_k)$$

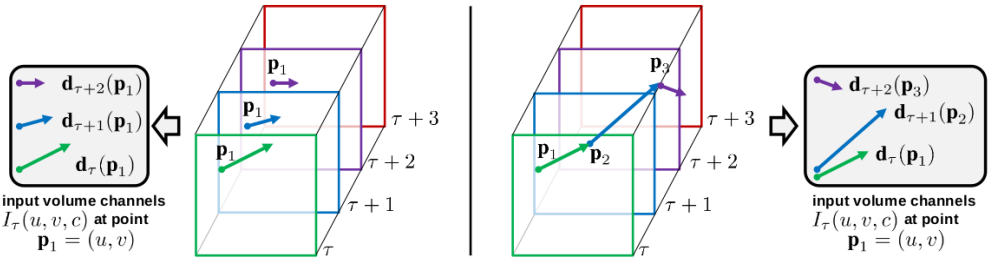


Figure 27: Construction of input volumes from multi-frame optical flow. Left: Optical Flow Stacking. Right: Trajectory Stacking. [44]

The authors further describe two optional techniques that are evaluated for constructing the inputs with either one of optical flow stacking methods.

1. **Bi-directional Optical Flow**: The input volume I_τ is created by using regular forward optical flow from frame τ to $\tau + L/2$ and additionally calculated optical flow from frame τ to $\tau - L/2$ in the backwards direction. Stacking the horizontal and vertical components of these optical flow fields results in an input volume of length L around frame τ.
2. **Mean flow subtraction**: The displacement vectors between two frames can dominantly be caused by global movement of the camera. Compared to regular camera-motion compensation, the authors use a simpler approach and just subtract the mean vector from each displacement field $\mathbf{d}_t$.

Since convolutional networks require fixed sized inputs, a $224 \times 224 \times 2L$ subvolume is randomly sampled from $I_\tau \in \mathbb{R}^{w \times h \times 2L}$ and use it as the temporal networks input. By using optical flow, a representation of motion is explicitly incorporated in the action recognition architecture.

An advantage of separating the spatial and the temporal stream is the possibility of pre-training the spatial stream network with large pre-existing image datasets. The authors use the dataset of the ImageNet Large Scale Visual Recognition Competition ILSVRC 2012 [76] for pre-training.

The temporal stream network is trained, according to the multi-task learning paradigm [77], on the UCF-101 and the HMDB-51 dataset simultaneously. By training a network on several tasks, the network learns more general video representations, since the second task acts as regularizer and more data can be utilized. This is implemented by using two softmax classification layers, one for each dataset. Each softmax layer has its own loss function and the sum of the individual losses is taken as the overall training loss for computing the weight updates by backpropagation.

Training for both networks is conducted with mini-batch gradient descent with 256 randomly selected videos at each iteration. From each of those videos, a single frame is randomly chosen, a 224×224 sub-image is randomly cropped, randomly horizontal flipped,

RGB jittered and then used as training input for the spatial stream network. An optical flow volume is constructed for this selected frame as described above and used as input for the temporal stream network.

Different setups of the spatial and temporal stream network are evaluated individually on the UCF-101 dataset for creating the final design of the two-stream architecture. Besides using two different dropout rates (0.5 and 0.9), the performance of the **spatial**-stream network is evaluated for:

1. Training the complete network from scratch on UCF-101.
2. Pre-training the network on ILSVRC-2012 and fine-tuning it on UCF-101.
3. Pre-training the network on ILSVRC-2012 and fine-tuning of only the last (classification) layer.

For evaluating the **temporal**-stream network, a fixed dropout rate of 0.9 is chosen. The performance is then measured for:

1. Using one or several (stacked) optical flow displacement fields; $L \in 1, 5, 10$.
2. Regular optical flow stacking.
3. Trajectory stacking.
4. Using bi-directional optical flow.
5. Using mean subtraction.

The findings are presented in table 6.

(a) **Spatial ConvNet.**

Training setting	Dropout ratio	
	0.5	0.9
From scratch	42.5%	52.3%
Pre-trained + fine-tuning	70.8%	**72.8%**
Pre-trained + last layer	**72.7%**	59.9%

(b) **Temporal ConvNet.**

Input configuration	Mean subtraction	
	off	on
Single-frame optical flow ($L = 1$)	-	73.9%
Optical flow stacking ($L = 5$)	-	80.4%
Optical flow stacking ($L = 10$)	79.9%	**81.0%**
Trajectory stacking ($L = 10$)	79.6%	80.2%
Optical flow stacking ($L = 10$), bi-dir.	-	**81.2%**

Table 6: Performance of the individual convolutional networks on UCF-101 (split 1) [44]

Regarding these results, the authors decide on pre-training the spatial-stream network on ILSVRC and fine-tuning only the last layer for the design of the final two-stream architecture. In the temporal-stream network mean subtraction and stacking multiple optical flow images is beneficial, so $L = 10$ is used as the default setting. Regular optical flow stacking performs better than trajectory stacking and bi-directional optical flow only yields slight improvement against forward optical flow. Therefore regular forward optical flow stacking is chosen for the temporal-stream network.

The authors highlight that the temporal-stream network significantly outperforms the spatial-stream network, which confirms the importance of motion information for action recognition from video.

After finding the optimal configurations for the individual temporal-stream and spatial-stream networks, different fusion methods (averaging and SVM) are being evaluated. Fusion by SVM performs best, as can be seen in table 7. The fused network's performance significantly improves over the individual network's performance, which implies, that the spatial and temporal recognition stream are complementary.

Method	UCF-101	HMDB-51
Spatial stream ConvNet	73.0%	40.5%
Temporal stream ConvNet	83.7%	54.6%
Two-stream model (fusion by averaging)	86.9%	58.0%
Two-stream model (fusion by SVM)	**88.0%**	**59.4%**

Table 7: Mean accuracy over the three provided splits of UCF-101 and HMDB-51 [44]

The results of Simonyan and Zisserman [44] show, that their two-stream architecture is a competitive approach in human action recognition from video.

3.2.2 Beyond Short Snippets: Deep Networks for Video Classification (2015)

Ng et al. [78] hypothesize that a global video description, that is a representation learned from the full temporal evolution of an input video, is essential for accurate action recognition. Similar to the approaches of Baccouche et al. [63] and Varol, Laptev, and Schmid [71], the goal is to design and evaluate video processing architectures, that are able to incorporate temporal information over long periods in the order tens of seconds, but preferably over the complete video. In doing so, it is challenging to construct video representations with fixed number of parameters from videos with variable length.

The authors discuss the results of Karpathy et al. [11], who discovered that spatio-temporal 3D convolutions applied on frame stacks of short video-subclips only yield marginally better performance than a single-frame baseline. The following approach therefore does not require 3D convolutions and is illustrated in figure 28):

1. CNN image features are obtained from the individual frames of an input video with a 2D ConvNet.
2. The resulting feature vectors, i.e. the activations of the ConvNet's last convolutional layer for each frame, are aggregated into a global video representation by one of the following methods:
    - The ConvNet is applied to a set of input frames in parallel. Feature pooling layers, which are added after the ConvNets, aggregate their activations into a final video representation.

- The ConvNet is applied to a single input frame per time-step. A recurrent neural network (RNN) with layers of LSTM cells [65] is added after the ConvNet to integrate frame information over time.

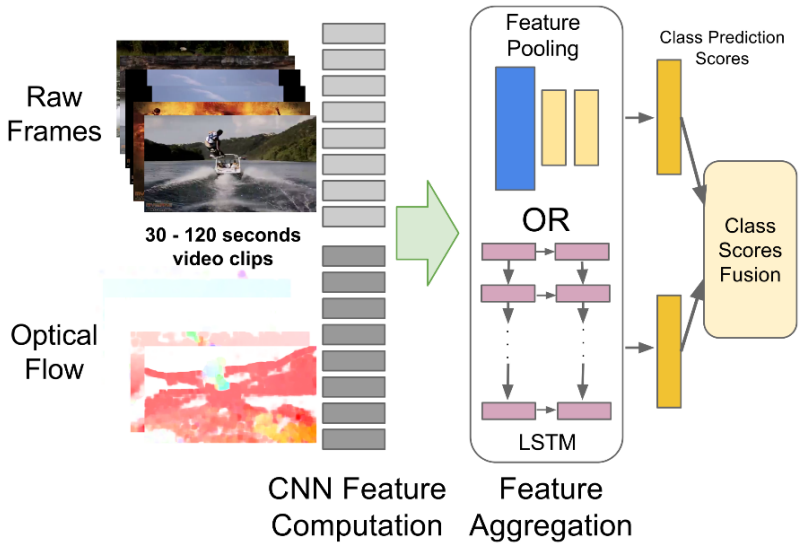


Figure 28: Overview of the approach taken by Ng et al. [78]

An advantage of this approach is that it can build on the independent achievements of CNN-based image processing architectures by directly embedding them as feature extractors for each frame. Hence two CNN architectures are being separately evaluated for this task:

1. AlexNet [79]
2. GoogLeNet [3]

Both networks take 220×220 pixel sized inputs. When evaluating the two different architectures without pre-training on frames of the Sports-1M dataset, GoogLeNet consistently outperforms AlexNet. Nonetheless both architectures are used for the following evaluation.

The temporal extent of this approach can be easily adjusted by changing the number of CNNs that are applied to input frames in parallel before being pooled. The LSTM models are not limited to a fixed input size and could in general process a complete video in a single pass. The CNNs, when applied in parallel, all share weights, which results in a constant number of trainable parameters over the number of input frames.

The authors propose to process input videos at only one frame per second, for being able to cover a long temporal extend while remaining computationally feasible. There-

fore implicit motion information from adjacent frames is lost at this framerate. This is compensated by also using explicit motion information, through optical flow inputs.

Feature Pooling
Different configurations of pooling layers are evaluated for combining feature vector representations of individual input frames, as illustrated in figure 29. The authors consider, max-pooling, average pooling as basic pooling operations. Max-pooling results in faster learning and was therefore chosen as the default pooling method in this approach. The pooling architectures additionally include, fully connected layers, time-domain convolution layers and soft-max layers, as described below.

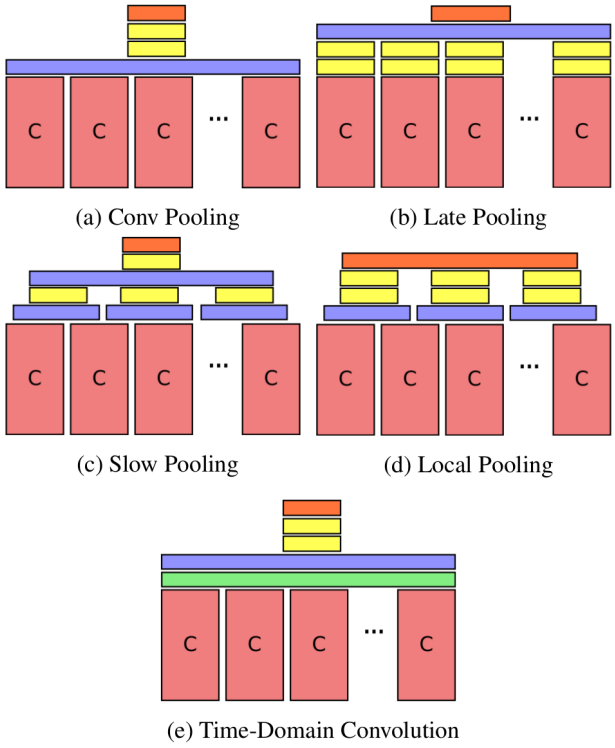


Figure 29: Evaluation of feature pooling architectures with: (Blue) max-pooling layers. (Green) time-domain convolution layers. (Yellow) fully-connected layers. (Orange) soft-max layers. [78]

a) **Conv Pooling:** Max-pooling is performed directly over the activations of the final convolutional layers across all input frames, followed by two fully-connected and a soft-max layer.

b) **Late Pooling:** Convolutional activations are first passed through two fully connected layers individually, before max-pooling is applied over the resulting high-level representations. Weights are shared between all fully-connected layers.

c) **Slow Pooling:** Two stages of pooling are applied: The first stage combines local temporal patches by max-pooling and feeds the activations into fully connected layers with shared weights. A single max-pooling layer then combines the resulting activations of all fully connected layers in the second stage.

d) **Local Pooling:** Similar to slow pooling frame information is combined in local temporal patches but only one stage of max-pooling is implemented. This prevents a potential loss of temporal information.

e) **Time-Domain Convolution:** An additional convolutional layer is implemented before pooling features across frames. It is called *time-domain convolution* since the convolutional layer is applied on features that originate from several frames.

LSTM architecture

As an alternative for for capturing dynamic content, i.e. the information that is encoded in the differences between frames, a LSTM architecture [65] with 5 layers containing 512 memory cells each is used to aggregate sequences of frame-level CNN activations. The following figure 30 shows the approach unfolded in time for 8 timesteps.

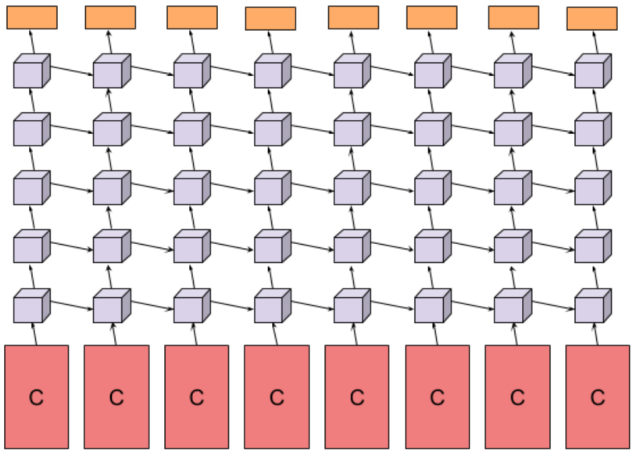


Figure 30: LSTM Architecture. Five layers of LSTM cells process the final CNN activations. The video class is predicted at every timestep by a softmax output layer after the LSTM layers. [78]

In the final architecture, analogously to the two-stream approach of Simonyan and Zisserman [44], each of the above described methods is evaluated in a two-stream setup. The authors train two instances of a model, one on raw video frames, the other on optical flow and perform late fusion over the two resulting streams. The used optical flow inputs are sampled at 15*fps* by using the method proposed in [**zach_duality_2007a**]. Evaluation is performed on the Sports-1M and UCF-101 dataset.

For training and testing on Sports-1M, the first 5 minutes of each input-video are sampled at 1*fps*, which results in 300 potential input frames per video. The frames are first resized

to 256×256 pixel, then a 220×220 pixel region of a randomly selected starting frame is selected and the desired number of following frames is cropped accordingly to create an input. Additionally input frames are flipped horizontally with a probability of 50%.

Models capable of processing up to 120 frames in a single example are trained, which corresponds to a temporal extent of 2 minutes of video. To obtain a video-level classification, 240 random input examples are generated from a video for testing as described above and the predictions are averaged.

At first pooling methods are being evaluated with AlexNet CNN and 120 frames as input. Results show, as given in table 8, that the conv-pooling architecture yields the best accuracy on Sports-1M dataset.

Method	Clip Hit@1	Hit@1	Hit@5
Conv Pooling	**68.7**	**71.1**	**89.3**
Late Pooling	65.1	67.5	87.2
Slow Pooling	67.1	69.7	88.4
Local Pooling	68.1	70.4	88.9
Time-Domain Convolution	64.2	67.2	87.2

Table 8: Evaluation of different feature pooling methods on Sports-1M dataset with a 120-frame AlexNet model. [78]

Fine tuning the models is found to be crucial for high performance. The CNN networks were initialized using pre-trained ImageNet models and then fine-tuned on the Sports-1M dataset in order to decrease training time. Since the CNN models share weights across input frames, the authors note that training time can also be saved by expanding a individually trained single-frame model to 30-frames and then expanding this 30 frame model to 120 frames, by re-training the pooling layers at each step.

Evaluating the influence of the number of input frames confirmed the authors initial hypothesis, that longer sequences of an input video need to be considered for high accuracy in action recognition. The LSTM method using 30 frame inputs only performs marginally worse than convolutional pooling with 120 frame inputs. Results are shown in table 9.

Method	Frames	Clip Hit@1	Hit@1	Hit@5
LSTM	30	N/A	72.1	90.4
Conv pooling	30	66.0	71.7	90.4
	120	70.8	72.3	90.8

Table 9: Comparison of convolutional pooling against LSTM-based RNNs for different numbers of input frames using GoogLeNet CNN models on Sports-1M. [78]

When fusing models with their optical flow counterparts, no significant increase in per-

formance could be observed on the Sports-1M dataset. Also, the models trained on optical flow evaluated individually yield a much lower performance compared to training on still image frames. The authors accredit this result to the noisiness of optical flow in the Sports-1M dataset, since it contains real-world videos with low quality and rough cuts. Overall, the best conv-pooling and lstm models perform significantly better than the state-of-the-art as reported by Karpathy et al. [11].

The UCF-101 dataset contains short videos, lasting 10-15 seconds on average. In comparison to the Sports-1M dataset, optical flow of UCF-101 provides an increase in performance. The authors best models on UCF-101, which combine image frames and optical-flow, outperform the state of the art reported by Simonyan and Zisserman [44] by a small margin.

3.2.3 Towards Good Practices for Very Deep Two-Stream ConvNets (2015)

Wang et al. [43] aim at improving the two-stream ConvNet approach, as introduced by Simonyan and Zisserman [44], by leveraging the advances of very deep ConvNet architectures in image classification. The trends in network design that lead to the success of convolutional architectures in image classification are: smaller convolutional kernels, smaller kernel strides and deeper network architectures.[43]

Examples of such very deep architectures are GoogLeNet [3] and VGGNet [2]. The authors use these two models to create and evaluate enhanced two-stream ConvNet models for video action recognition.

Two main draw-backs of recent ConvNet approaches in video action recognition are identified:

1. The applied models are shallow compared to their image counterparts. The deepest variant VGG-19[2] contains 16 convolutional layers, while the networks in the original two-stream ConvNet approach contain 5 convolutional layers.
2. The available dataset are too small, which results in over-fitting.

Considering these deficits, Wang et al. [43] therefore propose a set of good practices to enable very deep ConvNet architectures for accurate action recognition, specifically in the two-stream ConvNet architecture. These namely are:

1. Pre-training for the spatial- as well as the temporal-stream network.
2. Smaller learning rates.
3. Different and more data augmentation techniques.
4. Higher drop-out ratios.

The authors apply these good practices to a two-stream approach, which they call *Very Deep Two-Stream ConvNets*. The spatial net therein processes a single video frame of

input size ($224 \times 224 \times 3$) at a time. It is therefore an image classification architecture and either GoogLeNet or VGGNet is used.

The temporal net processes 10 stacked optical flow inputs, each splitted into images of the x- and y- coordinate. Its input therefore has a size of $224 \times 224 \times 20$. The convolutional kernels in the first layer need to be different than in the spatial stream to process these higher-dimensional inputs.

Evaluation is done using the UCF-101 dataset, specifically on the three splits into training- and test-set, that are provided with it. For each split, the training set consists of around 10.000 videos while the test-set consists of 3.000 videos (further described in section 4). UCF101 contains 13.320 videos in total and is considered extremely small for training very deep architectures [43].

Pre-training
The spatial net is pre-trained using an ImageNet model as in [44]. This specifically means, that the network's weights are initialized with the values of a trained ImageNet model. Interestingly, the authors find that pre-training is also beneficial for the temporal-stream network. The authors average the ImageNet model's filters across channels and duplicate them 20 times as filters for the temporal net's first layer.

Smaller Learning Rates
Since the networks in both streams have been pre-trained, the authors use smaller learning rates as in the original two-stream approach [44]. The authors note a faster conversion of the training and credit this to the pre-training.

Data Augmentation Techniques
Data augmentation techniques such as random cropping of inputs and horizontal flipping have been widely used to reduce over-fitting. Since random cropping favors cropping regions in the center of an image/frame, a corner cropping strategy is proposed. Specifically this means cropping the four corners and the center of the input images as inputs. This increases the variation of inputs and reduces over-fitting.

Additionally, network inputs are cropped from video frames on multiple scales. The input video is first scaled to 256×340 pixels and the cropping width and height is randomly chosen from $\{256, 224, 192, 168\}$. The cropped region is then scaled to 224×224 to form the network input.

High Dropout Ratio
The authors use dropout ratios of 0.8 and 0.9 for the fully connected layers at the end of the networks.

The proposed good practices are validates, by training very deep two-stream ConvNet models on the UCF-101 dataset. For testing, 25 frames or optical flow inputs are sampled from a test video. For each of these input samples 10 inputs for the very deep ConvNet are created by applying the above mentioned cropping method (four corners, the center and their horizontal flip). The final prediction is obtained by averaging over all inputs.

The temporal and spatial net are fused, using a weighted linear combination of both outputs. The weight for the temporal net is set to 2 and the weight for the spatial net is set to 1. Optical flow fields are extracted using the OpenCV implementation of TVL1 optical flow algorithm [80].

Table 10 shows the accuracy of the very deep Two-Stream ConvNets.

	Spatial nets				Temporal nets				Two-stream ConvNets			
Architecture	Split1	Split2	Split3	Average	Split1	Split2	Split3	Average	Split1	Split2	Split3	Average
ClarifaiNet	72.7%	-	-	73.0%	81.0%	-	-	83.7%	87.0%	-	-	88.0%
GoogLeNet	77.1%	73.2%	75.6%	75.3%	83.9%	86.5%	86.9%	85.8%	89.0%	89.3%	89.5%	89.3%
VGGNet-16	79.8%	77.3%	77.8%	**78.4%**	85.7%	88.2%	87.4%	**87.0%**	90.9%	91.6%	91.6%	**91.4%**

Table 10: Accuracy of the very deep two-stream ConvNet architecture on UCF-101 dataset for different ConvNet models. ClarifaiNet denotes the original two-stream approach and its results are taken from [44]. [43]

These results show, that the proposed good practices and using very deep convolutional network models lead to a significant improvement in performance over the original two-stream approach.

3.2.4 Convolutional Two-Stream Network Fusion for Video Action Recognition (2016)

Feichtenhofer, Pinz, and Zisserman [81] identify two deficits of the two-stream architecture[44] regarding the fusion of both streams and the processed temporal extent of the temporal stream:

1. The architecture is not able to learn pixel-wise relations between the spatial and the temporal stream, i.e. it cannot relate extracted object-features to corresponding motion features, since the two separate streams are fused late (at the final softmax-layer).
2. A temporal extent of only $L = 10$ frames is processed by the temporal stream network in a single pass. In the original approach, this was addressed by averaging network outputs over equally spaced temporal locations in the input video. However this averaging does not consider the temporal evolution of actions.

The authors address these deficits by evaluating fusion methods for better combining the appearance information in the spatial stream network with the motion information in the temporal stream network. Fusion methods are operations, that combine two feature maps from different ConvNets into a single feature map. When combining all feature maps in a specific layer of a first network with all the feature maps in a specific layer of another network, the two network streams are fused into one.

Fusion can be conducted either spatially or temporally. Spatial fusion combines feature maps between the streams of the two-stream architecture, which process information

from the same point in time. Temporal fusion combines feature maps resulting from applying the two-stream architecture to different points in time.

In the original two-stream approach [44] both streams are fused spatially by averaging softmax scores of both streams, which is called *late fusion*. The two-stream network is applied to several randomly sampled temporal locations of an input video and the output scores are temporally fused by averaging.

A fusion function $f : x^a, x^b \rightarrow y$ combines two feature maps $x^a, x^b \in \mathbb{R}^{W \times H \times D}$ into a fused feature map $y \in \mathbb{R}^{W \times H \times D'}$. W and H correspond to the spatial dimensions of the input frames, which get successively reduced by propagation through the layers. D and D' correspond to the number of channels, i.e. the number of filters in the convolutional layer.

The following **spatial fusion** methods are evaluated:

Sum Fusion:
Both feature maps are summed element-wise. The output feature map has the same dimensions as the input feature map. Since the ordering of the channels is arbitrary in each network stream, this fusion technique does not implement a semantic relation between feature maps.

Max Fusion:
This method is similar to Sum Fusion, but the maximum value at each location of the two feature maps is used as the new feature map. The output feature map has again the same dimensions as the input feature maps and does not represent a semantic relation between the feature maps.

Concatenation Fusion:
The two feature maps are stacked into each other along the dimension of channels. The resulting feature map has dimension $W \times H \times 2D$. Since the activations are not combined by a mathematical operation, a relation between the feature maps is not implemented but it can be learned by the following layers.

Conv Fusion:
The feature maps are first concatenated as described above. A set of D trainable filters with dimensions $1 \times 1 \times 2D$ is then convolved with the stacked feature maps and a bias is added to produce a resulting feature map of dimension $H \times W \times D$. The convolution can be seen as a trainable weighted combination of the feature maps across the channels, which is able to learn semantic relations between two feature maps.

Bilinear Fusion:
The channel vectors of the feature maps at each pixel locations are combined by computing their matrix outer product. This results in $H \cdot W$ matrices for each pixel location which are then added to form the new feature map of dimension $D \times D$. This fusion method is adapted from the Bilinear CNN approach in image classification [82]. Usually the fully connected layers are replaced by a linear SVM to make this fusion method usable in practice, i.e. the compensate the high number of parameters.

Implementing a spatial fusion layer between two network streams has significant influence on the number of trainable parameters in the network, especially if the networks are fused before the fully connected layers and if only the merged stream is kept, since most parameters are located in the fully connected layers.

In figure 31 two examples of placing fusion layers into the two-stream architecture are illustrated.

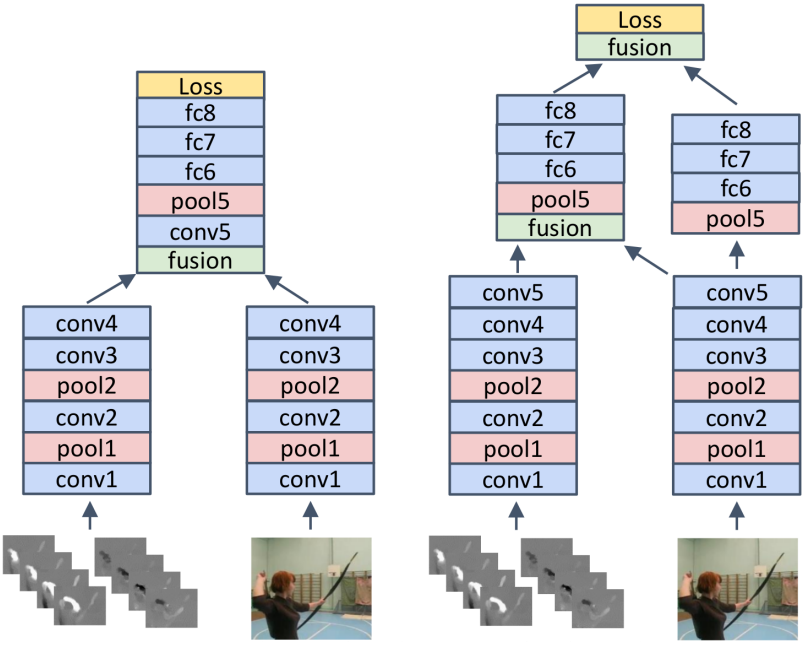


Figure 31: Placement of fusion layers between the spatial and temporal stream in a two-stream architecture. (*left*) Only the merged stream is kept. (*right*) The spatial stream is maintained after first fusion operation and merged before the final layer. [81]

The authors evaluate spatial fusion methods in a two-stream architecture as implemented in the original approach [44]. Each stream consists of 5 convolutional layers, and three fully connected layers. The accuracy on UCF101 (Split1), the number of parameters and overall number of layers in the architecture is given in table 11. The streams are fused after fifth convolutional layer and only the fused stream is kept.

Fusion Method	Fusion Layer	Acc.	#layers	#parameters
Sum	Softmax	85.6%	16	181.42M
Sum (ours)	Softmax	85.94%	16	181.42M
Max	ReLU5	82.70%	13	97.31M
Concatenation	ReLU5	83.53%	13	172.81M
Bilinear	ReLU5	85.05%	10	6.61M+SVM
Sum	ReLU5	85.20%	13	97.31M
Conv	ReLU5	85.96%	14	97.58M

Table 11: Performance and number of parameters for different spatial fusion methods in a two-stream setup, evaluated on UCF101 (split 1) [81]

Results of the original approach are reported in the first row of table 11. The authors' approach yields comparable results, when implementing the same fusion method. Conv Fusion performs best and reduces the number of parameters significantly to roughly half of the parameters in the original approach.

Implementing the fusion layer after an earlier convolutional layer was found to decreases the performance significantly. Best results are obtained by fusion after the rectification layer of convolutional layer 5, keeping both streams and fusing again at the final layer (as shown in figure 31 (right)).

When several inputs from different temporal points in the video are processed in succession, feature maps can be interpreted as being extended over time by stacking them for each input. In order to fuse information over several inputs, temporal pooling layers can be implemented. The input to a temporal pooling layer is a feature map $x \in \mathbb{R}^{H \times W \times T \times D}$ where H and W correspond to the spatial dimension, T corresponds to the temporal dimension and D corresponds to the number of channels in the feature map.

The temporal pooling methods shown in figure 32 are evaluated.

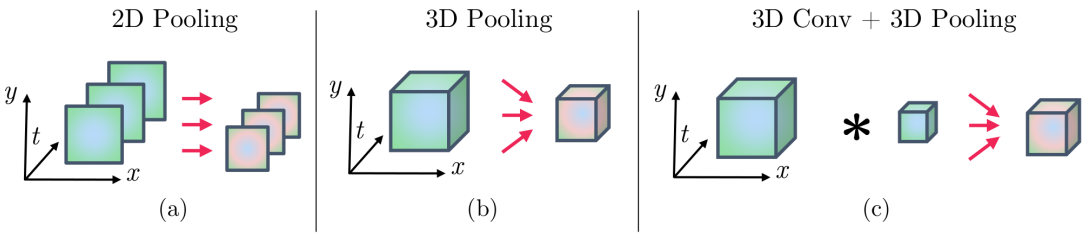


Figure 32: Methods for pooling feature maps along the temporal dimension. [81]

a) 2D pooling. The feature maps in each channel are only pooled spatially. The network predictions are averaged over all outputs for different points in time, where the network has been applied. The pooling operation ignores the temporal dimension.

b) 3D pooling. Max-pooling with a 3D pooling filter of dimensions $W' \times H' \times T'$ is

applied to the stacked feature maps. The max-pooling operation executed for each of the channels D individually. Pooling across different channels is not conducted.

c) 3D Conv + 3D Pooling. A set of D' filters $f \in \mathbb{R}^{W'' \times H'' \times T'' \times D \times D'}$ is convolved with the stacked feature maps and biases are added. The resulting feature map is then pooled with a 3D pooling filter as described above. Since the convolutional filter is trainable, it is able to learn weighted combinations of features in a local spatio-temporal region.

Implementing these temporal pooling methods shows, that 3D Conv + 3D Pooling performs best.

Using these previous results, a two-stream convolutional neural network model for action recognition is proposed by the authors as follows:

- Two VGG-16 models [2] are used as spatial and temporal stream network (13 convolutional and 3 fully connected layers each).
- The temporal stream operates on 10 stacked frames of optical flow and is fused at the last convolutional layer and after the fully connected layers (as illustrated in figure 31 *right*).
- The two-stream model is applied at 5 temporal points in the video with fixed temporal distance.

The approach achieves 92.5% accuracy on UCF-101 and 65.4% accuracy on HMDB-51.

3.3 Pooling of Deeply Learned Features

3.3.1 Action recognition with trajectory-pooled deep-convolutional descriptors (2015)

Wang, Qiao, and Tang [55] propose a novel method for building video representations by combining the advantageous properties of hand-crafted features as in *Dense Trajectories* [41] with deeply learned features as in *Two-Stream Convolutional Networks* [44].

In the *Improved Dense Trajectories* approach [41], local features are extracted along trajectories, which are mostly located near regions of prominent motion in a video. Wang, Qiao, and Tang [55] claim however, that hand-crafted features are not discriminative enough for accurate action recognition.

Deep architectures have proven to learn discriminative features effectively with *Two-Stream Networks* being a successful approach that finally performed comparably to *Improved Dense Trajectories*.

The main outline of an approach for combining these two approaches in action recognition is given by the authors as follows:

1. A two-stream convolutional network architecture is trained on multiple scales of a large dataset.
2. *Dense Trajectories* are being extracted from input videos according to the approach in [41].
3. The learned feature maps of the two-stream architecture are used as local feature extractors by pooling the convolutional activations over areas around the trajectories. The resulting descriptor is called trajectory-pooled deep-convolutional descriptor (TDD).
4. The local TDDs are aggregated over the complete video using the Fisher Vector representation. [83]
5. A linear SVM is trained to assign action labels to the FV representations, i.e. to perform action recognition.

Any convolutional architecture can be embedded in a two-stream setup for TDD extraction. The authors choose the Clarifai network [84] with less filters in the *conv4* layer (512 instead of 1024) and a lower dimensional *full7* layer (2048 instead of 4096). This architecture is used for implementing the spatial stream network as well as the temporal stream network. Details are illustrated in table 12) below.

Layer	conv1	pool1	conv2	pool2	conv3	conv4	conv5	pool5	full6	full7	full8
size	7×7	3×3	5×5	3×3	3×3	3×3	3×3	3×3	-	-	-
stride	2	2	2	2	1	1	1	2	-	-	-
channel	96	96	256	256	512	512	512	512	4096	2048	101
map size ratio	1/2	1/4	1/8	1/16	1/16	1/16	1/16	1/32	-	-	-
receptive field	7×7	11×11	27×27	43×43	75×75	107×107	139×139	171×171	-	-	-

Table 12: Layers of the Clarifai network modified for TDD extraction [55]

For each convolutional or pooling layer with kernel size k, zero padding of the layer's input is applied with size $\lfloor k/2 \rfloor$. This padding prevents an additional reduction in dimensionality between the layer's input and output besides the reduction that stems from applying the kernel with a certain stride. Therefore the position of a trajectory point in the input can be easily related to coordinates in the convolutional feature map at question by incorporating the map size ratio r of the layer, as given in table 12. Specifically, the p-th point in a video trajectory (x_p, y_p, z_p) is represented by point $(\overline{r \cdot x_p}, \overline{r \cdot y_p}, z_p)$. Where $\overline{(\cdot)}$ denotes the rounding operation.

Given a video V, training the two-stream architecture results in a set of feature maps $\mathbb{C}(V)$ from the spatial stream s and temporal stream t.

$$\mathbb{C}(V) = \{C_1^s, C_2^s, \cdots, C_M^s, C_1^t, C_2^t, \cdots, C_M^t\}$$

where:

- $C_m^s, C_m^t \in \mathbb{R}^{H_m \times W_m \times L \times N_m}$ denotes the m-th feature map of the spatial or temporal stream
- H_m is it's height
- W_m is it's width
- L is the number of video frames
- N_m is the number of channels
- M is the number of feature maps in each stream.

There are two steps involved in extracting the local trajectory-aligned descriptors (TDD) from a 3D volume around a trajectory:

1. Feature Map Normalization (two different methods)
    - Spatiotemporal Normalization
    - Channel Normalization
2. Trajectory Pooling

Feature Map Normalization
Normalization has been widely applied to hand-crafted features because it reduces the influence of illumination. The authors use this technique on convolutional feature maps to suppress the activation burstiness of some neurons.

In *Spatiotemporal Normalization*, each feature map is normalized independently across each channel according to it's maximal value. Specifically, for any channel n and a feature map $C \in \mathbb{C}(V)$:

$$\tilde{C}_{st}(x, y, z, n) = C(x, y, z, n) / \max_{x,y,z} C(x, y, z, n)$$

Spatiotemporal Normalization ensures, that the feature maps across all channels range in the interval $(0, 1)$.

In *Channel Normalization* the values of each feature map are normalized according to the values at the same spatial positions but in another channel. Specifically, for an spatial position (x, y, z) in the feature map at question:

$$\tilde{C}_{ch}(x, y, z, n) = C(x, y, z, n) / \max_{n} C(x, y, z, n)$$

Channel Normalization ensures, that the feature map values of each pixel range in the interval $(0, 1)$ and therefore contribute equally to the final representation.

Both normalization methods were evaluated experimentally and the authors found, that combining the resulting video representations from both normalization methods by late fusion yields the best results.

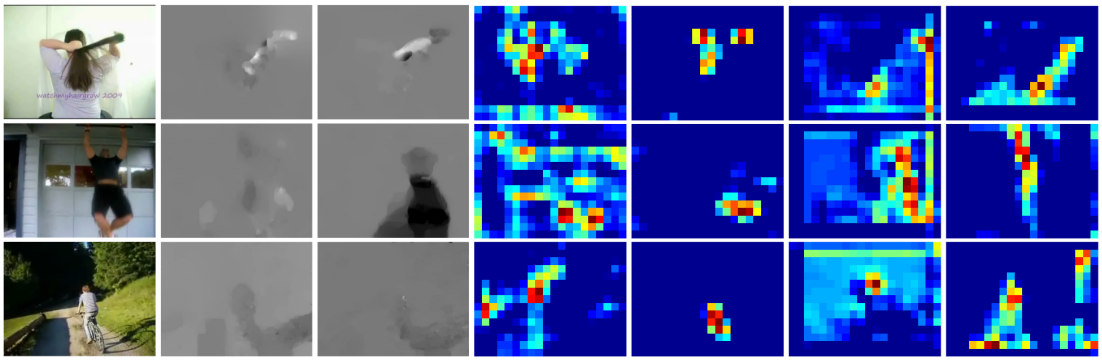
Trajectory Pooling Given the k-th trajectory out of all trajectories $\mathbb{T}(V)$ over a video

V, a trajectory-pooled deep convolutional descriptor in respect to a normalized feature map $\tilde{C}_m$ is constructed by sum-pooling the feature map values along the trajectory-points in the feature map.

$$D(T_k, \tilde{C}_m) = \sum_{p=1}^{P} \tilde{C}(\overline{(r_m \cdot x_p^k)}, \overline{(r_m \cdot y_p^k)}, z_p^k)$$

Where r_m is the map size ratio belonging to feature map $\tilde{C}_m$.

The extracted TDDs over a complete video are then aggregated using the Fisher Vector representation [83] to form a global video representation. A linear SVM is then trained to learn the action classes to these representations.



(a) RGB (b) Flow-x (c) Flow-y (d) S-conv4 (e) S-conv5 (f) T-conv3 (g) T-conv4

Figure 33: a) Snapshots of input videos. *b-c*) Their corresponding optical flow fields in x and y dimension. *d-g*) The corresponding activations of the feature maps used for TDD extraction [55]

Using TDDs for action recognition is evaluated on the UCF-101 and HMDB-51 dataset. Since UCF-101 is bigger than HMDB-51, the authors initially train the two-stream model on the bigger UCF-101 dataset and transfer it for TDD extraction on HMDB-51.

The weights of the **spatial net** are initialized using the publicly available model of [85] and then fine-tuned on the frames of UCF-101 videos. During fine-tuning, the frames of UCF-101 are first resized to have 256 pixels on their smallest side. A randomly cropped region of 224×224 pixels then builds the network training input after random horizontal flipping.

The **temporal net** is trained from scratch on stacked optical flow image volumes. The optical flow between two adjacent frames of a training-video is calculated using the TVL1 algorithm [80] (implemented in OpenCV) and split into images of it's x and y component. This forms the optical flow volume for the complete video. A $224 \times 224 \times 20$ pixel sized sub-volume is randomly cropped from it and randomly flipped to form a training-input for the temporal stream network.

The authors evaluate the two-stream architecture without extracting TDDs and obtain similar results as Simonyan and Zisserman [44]. Further experimental evaluation shows, that using the descriptors from *conv4* and *conv5* in the spatial net as well as from *conv3* and *conv4* in the temporal net yields best performance.

Combining the descriptors of the spatial and temporal net in the TDD approach significantly outperforms the previous state of the art, such as two-stream convolutional network [44]. Detailed results are shown in table 13.

Algorithm	HMDB51	UCF101
iDT	**57.2%**	**84.7%**
Spatial net	40.5%	73.0%
Temporal net	54.6%	83.7%
Two-stream ConvNets	**59.4%**	**88.0%**
Spatial conv4	48.5%	81.9%
Spatial conv5	47.2%	80.9%
Spatial conv4 and conv5	**50.0%**	**82.8%**
Temporal conv3	54.5%	81.7%
Temporal conv4	51.2%	80.1%
Temporal conv3 and conv4	**54.9%**	**82.2%**
TDD	63.2%	90.3%
TDD and iDT	**65.9%**	**91.5%**

Table 13: Action recognition performance of TDDs on HMDB51 and UCF101 compared to improved dense trajectories (iDT) [41] and two-stream ConvNets [44]. [55]

3.4 Generative Models

Palasek and Patras [86] describe the Restricted Boltzmann Machine as one of the most popular deep learning models, that can be trained in an unsupervised way.

Restricted Boltzmann Machine:
Boltzmann Machines [87] are probabilistic networks, specifically undirected graphical models, that can learn the probability distribution inherent in a given dataset. A Boltzmann Machine consists in its simplest form of two binary layers: A visible layer x and a hidden layer h, which are fully connected to each other. The Boltzmann Machine allows connections between units of the same layer, which renders training computationally extremely demanding.

The design of the Restricted Boltzmann Machine [88] addresses this problem by restricting connections to nodes between different layers, which explains its name. Restricted Boltzmann Machines can be trained using a Markov Chain Monte Carlo algorithm, namely Contrastive Divergence. Once a Boltzmann Machine is trained, i.e. it learned the underlying probability distribution of a dataset, further data can be sampled from it. The visible layer functions as both, input-layer during training and output-layer during sampling.

Restricted Boltzmann Machines can be trained with labeled as well as unlabeled data (supervised or unsupervised). When training in an unsupervised manner, the RBM is able to learn feature representations of the inputs by mapping them onto activations of its hidden layer. When labeled data is present, the RBM can perform classification. For training, the labels are being concatenated with the input data and fed into the RBM. A prediction for a given test example can then be obtained by using it as input for the RBM and sampling the model for the missing input, that was used for the labels during training. The RBM then generates the most likely value for the missing label-input given the testing-input. [89]

Restricted Boltzmann Machines correspond to feed-forward neural network models when reusing the learned weights in a topological equivalent neural network model. This is how unsupervised pre-training is conducted. Models that can be trained in an unsupervised manner are especially attractive for the video-domain where labeling data is costly. [89]

Deep Belief Networks are stacked Restricted Boltzmann Machines. In general the learning capacity of a single RBM is limited, but it can be increased by stacking multiple RBMs to form so called Deep Belief Network. [90]

3.4.1 Unsupervised Learning of Video Representations using LSTMs (2015)

Srivastava, Mansimov, and Salakhudinov [91] design recurrent neural network models based on Long-Short-Term-Memory cells (LSTMs) [65] for unsupervised learning of video representations from unlabeled video data. They implement two LSTM networks in an auto-encoder setup: An encoder LSTM network maps input sequences of video frames to a fixed-length representation of hidden states, which is then decoded by another LSTM network to obtain a desired target sequence.

The authors consider three choices for the target sequence:

1. The initial input sequence in reversed order (LSTM Auto-encoder model).
2. A sequence of future inputs from the video, which were not presented to the encoder network before (LSTM Future Predictor model).
3. A hybrid approach with two LSTM networks decoding the representation into both of these choices simultaneously (Composite model).

Inputs to the LSTM encoder-network can be any kind of video representation, the authors evaluate raw image patches as well as high-level percepts, that are extracted by a convolutional neural network from the input.

For a quantitative evaluation in a supervised setting, the models are first pre-trained in an unsupervised way and then fine-tuned for an action recognition task on the UCF-101 and HMDB-51 datasets. Results of the purely unsupervised setting are evaluated qualitatively by printing and analysing decoded sequences.

The **LSTM Auto-encoder model**, as shown in figure 34, consists of an encoder and a decoder LSTM network with parameters W_1 and W_2 respectively. The encoder network is presented with one vector of the input sequence per timestep. After the final input has been processed, the encoder's hidden state is used as a representation of the input. The decoder network is then trained to translate the representation back into the initial input sequence, but in reversed order. If the decoder is able to reconstruct the input sequence accurately from the representation, the essential information about the input must be encoded in the representation, which therefore is a good representation.

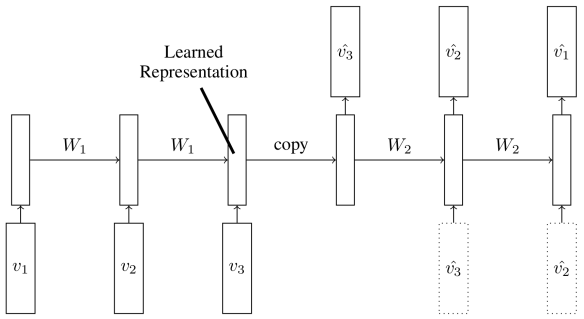


Figure 34: Auto-encoder model [91]

The parameters of the LSTM networks W_1 and W_2 are, according to the network's recurrent nature, applied to its hidden state at timestep i, given an input v_i of the input sequence $\{v_1, \cdots, v_T\}$ of length T. The outputs of the decoder network $\hat{v}_i$, $i \in \{1, \cdots, T\}$ correspond to the decoded input sequence in reversed order. The decoder network can be *conditional*, i.e. it receives its last generated output as input (dotted boxed in figure 34).

The **LSTM Future Predictor model** also consists of two recurrent LSTM networks but predicts frames that continue the previously presented input sequence. The architecture, as shown in figure 35, is the same as for the Auto-encoder model. This decoder LSTM network can also be *conditional* or *unconditioned*.

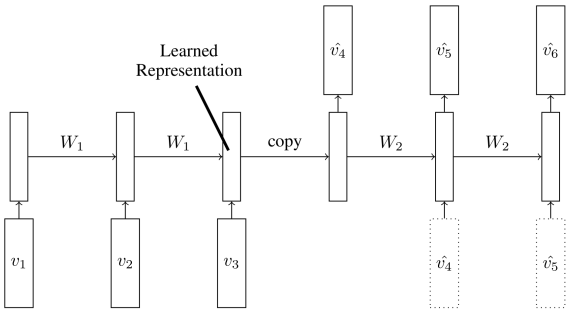


Figure 35: Future-predictor model [91]

The **Composite model** consists of an encoder LSTM network and two decoder LSTM networks, which reconstruct the original input sequence and predict future frames simultaneously. This model tries to overcome the disadvantages that each of the single models have: A high-capacity auto-encoder tends to simply memorize the inputs, while a future predictor tends to store information about just the last few frames of the input, because those are needed most for future prediction. The composite model, as shown in figure 36, therefore puts additional constraints on the encoder networks, i.e. the learned representation must be suitable for both tasks.

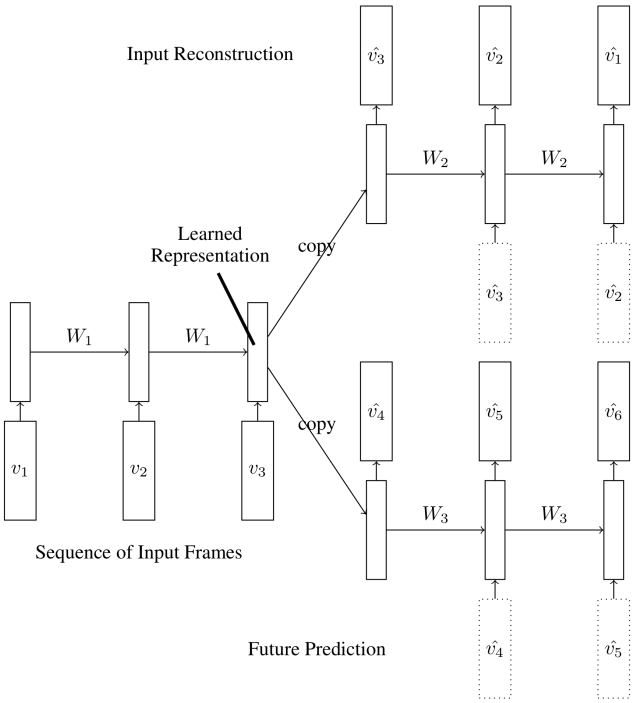


Figure 36: Composite model [91]

Srivastava, Mansimov, and Salakhudinov [91] provide a qualitative analysis of the composite model by visualising its outputs. The model, separately evaluated for one and two layers of 2048 LSTM units each, is trained on 10 frames of synthetically composed video-data consisting of two moving MNIST digits in a 64×64 pixel area. The digits can be superimposed and bounce off the walls. The results for image reconstruction and future prediction are shown in figure 37.

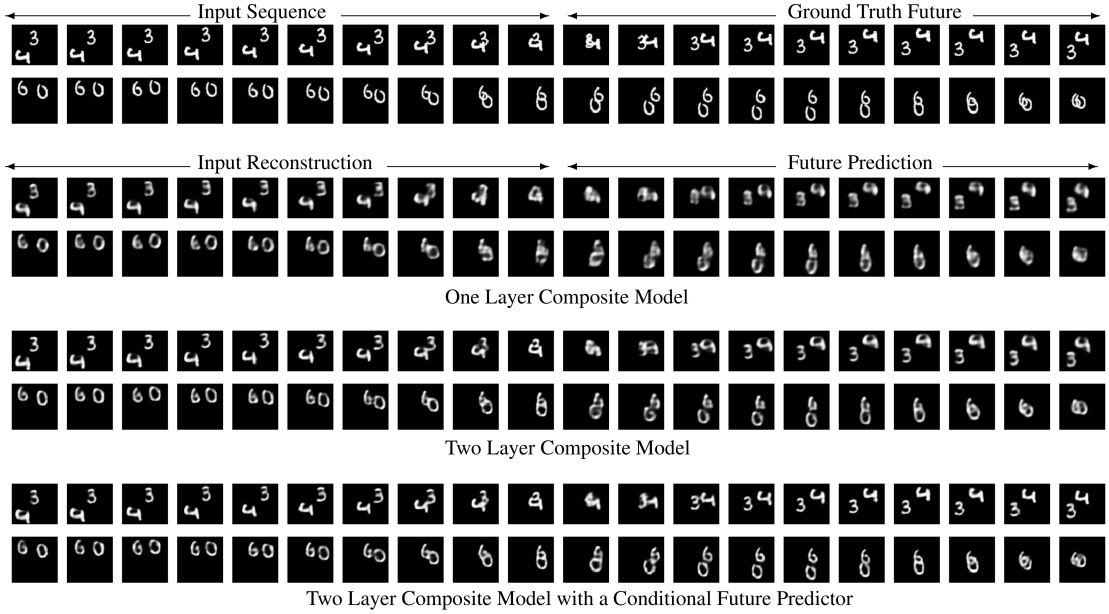


Figure 37: Input reconstruction and future predictions of composite models on a synthetic dataset of moving MNIST digits. [91]

The first row of figure 37 shows the 10 frame input sequence and the following 10 future frames. The model is able to make fairly good predictions on this dataset. Adding a second layer of LSTM units and making the future predictor conditional further improves the predictions. When applying the model on 32×32 natural image patches of UCF-101 dataset, the future prediction quickly blurs out as illustrated in figure 38.

The authors evaluate the effectiveness of features obtained from unsupervised learning for supervised action recognition:

1. A two layer model with 2048 LSTM cells in each layer is trained in an unsupervised way on 300 hours of video, randomly sampled from the Sports-1M dataset. The model is trained to autoencode 16 frames and predict 13 frames.
2. Once training is finished, a LSTM network as shown in figure 39 is initialized with the weights learned by the composite model.
3. Depending on the used benchmark, this LSTM network is then fine-tuned on either UCF-101 or HMDB-51 to function as a supervised classifier.

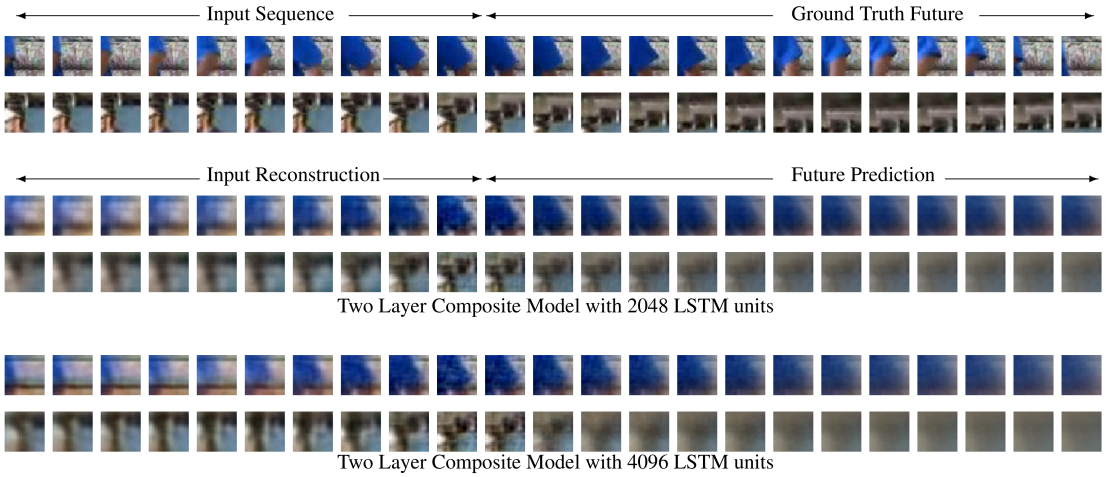


Figure 38: Image reconstruction and future prediction of a two-layer composite model with a different number of LSTM units in each layer. [91]

The model is designed to use center crops of size 224×224 from the datasets or high-level optical-flow percepts, i.e. activations of a temporal stream CNN as in [44], as inputs.

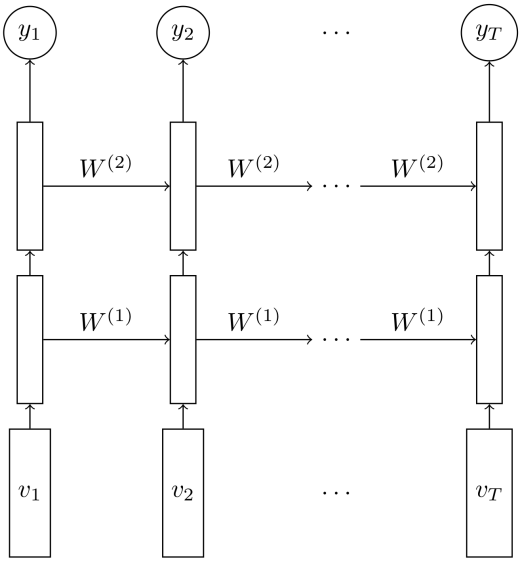


Figure 39: Recurrent LSTM classifier unfolded in time with input sequence $\{v_1, \cdots, v_T\}$ [91]

Baseline method involves initializing the model with random weights. The results as given in table 14 show, that initializing the model with features obtained from unsupervised learning increases the performance significantly, when only very few training examples per class are available (improvement of about 5%). With more available labeled data,

Method	UCF-101 small	UCF-101	HMDB-51 small	HMDB-51
Baseline LSTM	63.7	74.5	25.3	42.8
Autoencoder	66.2	75.1	28.6	44.0
Future Predictor	64.9	74.9	27.3	43.1
Conditional Autoencoder	65.8	74.8	27.9	43.1
Conditional Future Predictor	65.1	74.9	27.4	43.4
Composite Model	67.0	**75.8**	29.1	**44.1**
Composite Model with Conditional Future Predictor	**67.1**	**75.8**	**29.2**	44.0

Table 14: Comparison of unsupervised pre-training methods for an two-layer LSTM classifier. The small versions of benchmarking datasets are subsets containing only 10 (UCF-101) and 4 (HMDB-51) action videos per class. [91]

the improvement becomes smaller (about 1%).

The authors conclude: Pre-training models on just unrelated datasets (here 300 hours of YouTube videos) can increase action recognition performance. Unsupervised learning gives a significant improvement if only few labeled training examples are available.

3.4.2 Action Recognition Using Convolutional Restricted Boltzmann Machines (2016)

Palasek and Patras [86] apply Convolutional Deep Belief Networks (ConvDBNs) for learning video representations in an unsupervised fashion in order to perform human action recognition. ConvDBNs are generated by stacking layers of Convolutional Restricted Boltzmann Machines (ConvRBMs) [92]. The ConvRBM was initially proposed by Lee et al. [92] to address several problems that occur when applying RBMs and DBNs to images:

1. Realistic images are high-dimensional and DBNs do not scale well with the input size. [92]
2. DBNs do not take the special structure of images into account, specifically that objects can occur in any area of an image. Feature extractors therefore have to be learned for each location in the image separately. [92]

Equivalently to Convolutional Neural Networks, the Convolutional Restricted Boltzmann Machine uses weight-sharing and pooling to make the detection of a specific feature in an input image translationally invariant.

The basic **ConvRBM** architecture is shown in figure 40. It has the following properties:[92]

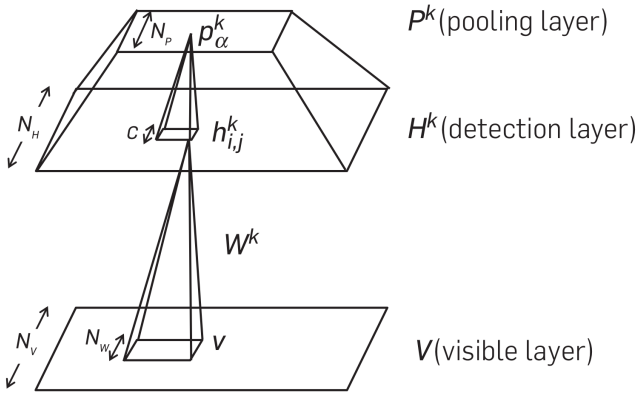


Figure 40: Architecture of the Convolutional Restricted Boltzmann Machine. The figure shows group k in the detection- and pooling-layer. [92]

- The visible layer V consists of binary input units, but can be easily modified to handle real-valued inputs.
- The hidden layer H consists of K groups, with a filter W^k belonging to each group ($k \in \{1, \cdots, K\}$). To illustrate the equivalency to CNNs, which consists of detection and pooling layers, the hidden layer is denoted as *detection layer* in figure 40.
- A group corresponds to a feature map in CNNs, i.e. the activations produced by a convolutional filter.
- The filter weights define the connections between a local patch in the visible layer and a hidden unit in the detection layer of the filter's group.
- All hidden units in the detection layer of a group are connected to their local patches by the same filter weights, i.e the filters in a group share weights.
- The hidden units share a bias b_k per group and all visible units have a single bias c.

To form a more expressive deep architecture, called Convolutional Deep Belief Networks (ConvDBNs), layers of ConvRBMs are stacked. Probabilistic max-pooling is used to reduce the dimensionality of each hidden (detection) layer. Since regular max-pooling was designed for deterministic models such as CNNs, a probabilistic max-pooling method is derived, in order to enable full probabilistic inference in the model. In probabilistic max-pooling, only one unit in the input patch of the pooling operation is allowed to be active. The output unit is active, if and only if one unit in the input patch is active. Pooling operations are needed, to feed progressively more information to higher-level feature detectors and to make high-level representations invariant to translations of features in the lower layers. [92]

The energy function of the ConvDBN is defined by summing the energy functions of the

individual ConvRBMs. ConvDBNs can be trained in a greedy, layer-wise way: After a ConvRBM layer is trained individually, its weights are frozen and its activations in the hidden/detection layer are taken as inputs for the following layer. [92]

Palasek and Patras [86] apply three stacked ConvRBMs (a ConvDBN) to action recognition. They use the Gaussian-Bernoulli version of ConvRBMs, which is the real-valued extension of regular binary ConvRBMs, to extract features from still video frames and aggregate them into a video representation that can be classified using a SVM.

Different configurations for the layers in the ConvDBN are being evaluated:

First layer ConvRBM
: Either 32 or 64 filters sized 5×5 or 3×3 pixels.
Optional probabilistic max-pooling layer with patch size of 2×2 units.

Second layer ConvRBM
: Contains 64 filters of size 3×3 Optional probabilistic max-pooling layer with patch size of 2×2 units.

Third layer ConvRBM
: Contains 128 filters of size 3×3 pixels. Optional probabilistic max-pooling layer with patch size of 2×2 units.

The approach for generating fixed length video representations for variable sized input videos is shown in figure 41.

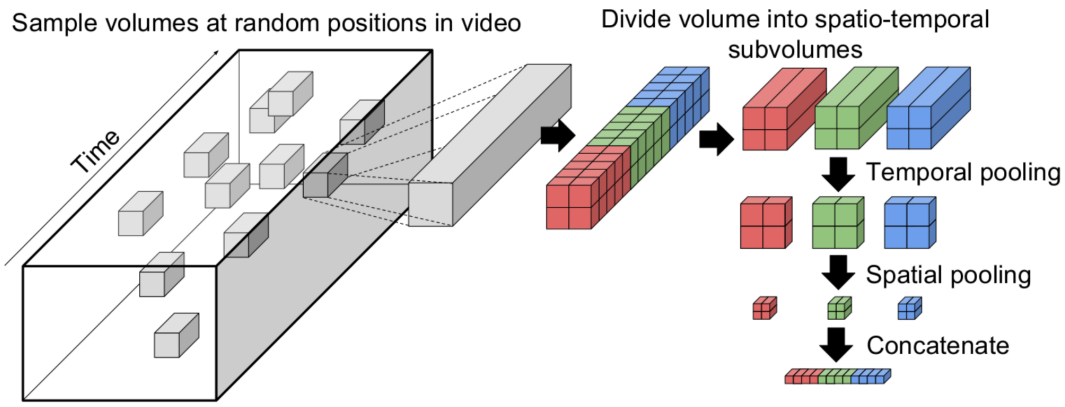


Figure 41: Approach for generating fixed sized video-representations using ConvDBN activations. An input video is handled as video-volume of stacked frames. [92]

Given an input video, interpreted as a volume of stacked video frames, subvolumes of size 32×32 pixels and with a length of 15 frames are extracted at random positions. Each subframe is fed individually into the ConvDBN and the resulting activations of one of its layers are extracted. The activations of all subframes are stacked to form a feature

representation of the subvolume. This feature representation is then divided into $2 \times 2 \times 3$ parts, mean-pooled along the temporal dimension, spatially pooled and concatenated to form the final representation of the given subvolume (see figure 41). [92]

A Gaussian Mixture Model is trained on all the extracted feature representations to encode them into a Fisher Vector representation of the video. These Fisher Vectors can then be classified using a SVM, as in approaches that use hand-crafted local features (see section 2).

Palasek and Patras [86] evaluated their approach using the activations of different ConvDBN layers as features. The ConvDBN is trained in an unsupervised way on grayscaled, static video-frames of the UCF-101 dataset, which contains a total of $1,700,000$ frames. For each of the 9537 training videos UCF-101 dataset, 1000 subvolumes are extracted as training inputs for the ConvDBN.

Activations from the first pooling layer worked best and yield an accuracy of 55.06% in the overall approach. Although, the results are not competitive to other state-of-the-art approaches reported previously in this work, the experiments were conducted to compare the performance of features learned in an unsupervised manner to hand-crafted features.

The authors also evaluated incorporating HOG features in their experimental setup and achieved significantly worse results: 50.75%. They therefore argue, that features learned in an unsupervised way are more descriptive than hand-crafted features for human action recognition from video.

The authors showed, that features learned from video data in an unsupervised way is able to outperform hand-crafted features. The overall performance, that is action recognition accuracy of 55.06% on UCF-101, is rather weak compared to other approaches. This probably stems from the fact, that features were learned from static frames only, instead of incorporating motion information.

3.5 Temporal Coherency Networks

3.5.1 Shuffle and Learn: Unsupervised Learning using Temporal Order Verification – Misra et al. (2016)

Misra, Zitnick, and Hebert [93] propose an efficient method for learning video representations based on temporal order verification. Temporal order verification is a binary classification task. Given a sequence of video frames, a classifier has to determine whether the sequence is in correct temporal order. Datasets to train and test such a classifier can be easily generated by sampling short sequences of frames from a video and exchanging frames in some of them.

In the strictest sense, using temporal order verification for representation learning is not an unsupervised method, since the labels *correct temporal order* and *incorrect temporal*

order are learned for input sequences. The authors argument however, that obtaining the label is free and the method can therefore be attributed as unsupervised.

Forcing an algorithm to reasoning about the temporal order of frames, produces a representation that captures the motion of persons and objects in the scene, since they define the temporal order. A resulting video representation therefore embeds information that is most vital for accurate action recognition.

The authors evaluate this approach using a Convolutional Neural Network, that is applied to a sequence of three frames in parallel. Input sequences of three frames are used, since four or five frames did not yield a significant improvement in performance. Results are obtained on the UCF-101 and HMDB-51 benchmark dataset.

Figure 42 shows the approach for sampling input sequences from an unlabeled video and the triplet CNN architecture for classifying these sequences.

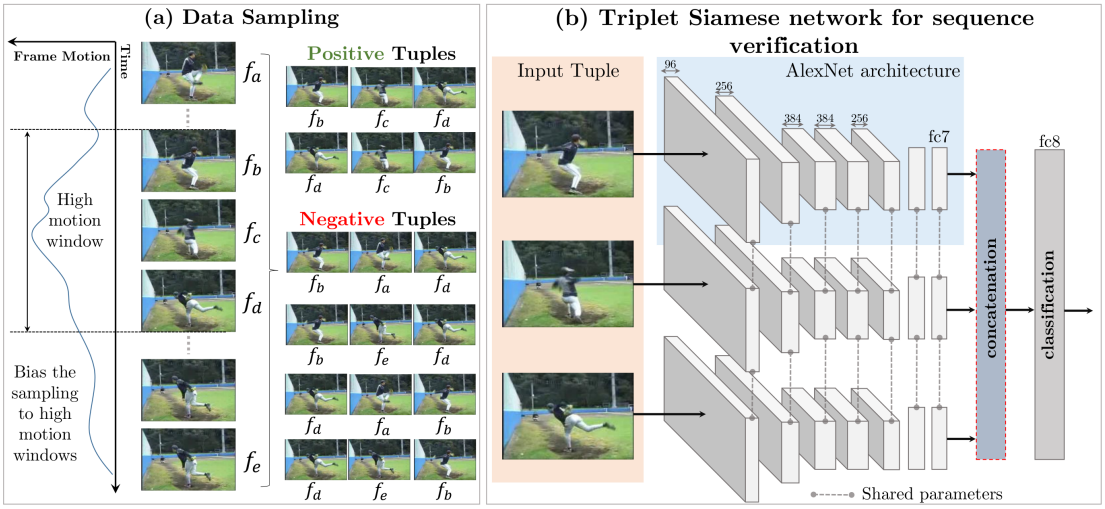


Figure 42: Sampling method of input sequences and triplet Convolutional Neural Network for temporal order verification [93]

The three frames, that are used to construct positive and negative input sequences are sampled from regions in the video, which were previously identified to contain a certain magnitude of motion by using optical flow. This ensures, that the sampled frames differ enough to make positive and negative sequences clearly distinguishable.

The authors use a CNN model called CaffeNet, which is a slightly modified version of AlexNet [67] and is publicly available in Caffe [70]. The CNNs form a siamese triplet, i.e. they all share the same parameters, and each one takes one of the frames of the sequence as input. Each network maps its input frame to a high-level representation of activations in the layer $fc7$. The three representations are concatenated an fed into a classifier for predicting, whether the sequence is in correct or not.

The network is trained on about 900k sequences extracted from the training set of UCF-101 (split 1). The resulting video representations can be reused for supervised action recognition training, by initializing the layers of a new CNN up to $fc7$ with the weights of the unsupervised model, adding an additional layer $fc8$ for the new task and fine-tuning the complete network using labeled training data.

To compare the advantage of their unsupervised pre-training method against no pre-training, the authors report results for the UCF-101 and HMDB-51 dataset (table 15)

Dataset	Initialization	Mean Accuracy
UCF101	Random	38.6
	(Ours) Tuple verification	**50.2**
HMDB51	Random	13.3
	UCF Supervised	15.2
	(Ours) Tuple verification	**18.1**

Table 15: Comparison of mean classification accuracies of a CaffeNet CNN with temporal order pre-training against without pre-training (random initialization of weights) on all three splits of UCF-101 and HMDB-51. [93]

On UCF-101 pre-training with temporal order verification yields a significant improvement of +12.4% against training the network from scratch. On HMDB-51, the authors evaluate no pre-training, supervised pre-training on UCF-101 and pre-training using temporal oder verification. The improvement of the latter is smaller compared to the results of UCF-101 but still significant (increase in mean accuracy of +4.7%).

4 Datasets and Benchmarks in Action Recognition

Using publicly available dataset for action recognition algorithms has two main benefits:

1. Collecting, annotating and storing video datasets is a tedious task, since videos have a higher dimensionality than images and especially deep learning algorithms require large amounts of training data. Additionally, the beginning and end of a performed action has to be included in the action label, possibly along with the spatial position of the action in the scene. Therefore using already existing datasets saves time in the development of action recognition algorithms.

2. More importantly: Using the same datasets in different approaches, along with standardized evaluation schemes, enables the comparison between them. This is the basis of evaluating the quality of action recognition approaches.

The influence of the experimental setup on the accuracy of action recognition algorithms was assessed by [94] on the KTH dataset [15]. Their results show, that using different evaluation protocols can result in accuracy differences of up to 9%. Therefore conclusions about the accuracy of different action recognition algorithms can be distorted and heavily biased by the method of evaluation [63], which emphasizes the importance of standardized evaluation strategies and benchmarks.

Given the importance of datasets in the computer vision community, to our best knowledge only a few reviews on have been published [95][96][97][6]. The review of Ahad et al. [95] and Liu, Feris, and Sun [96] were published back in 2011 and provide a brief overview of benchmarking datasets. Hassner [97] also review available datasets briefly and promote a completely different approach to benchmarking action recognition approaches, namely *Action Similarity Labelling*, which is further describes in this below in this section. Chaquet, Carmona, and Fernández-Caballero [6] provide a very comprehensive and detailed review of human action video datasets in 2013.

Building on [6], this section gives a brief review of the most commonly used datasets and focusses on the newest ones. All datasets in this section are publicly available. Additionally, a review of datasets containing Activities of Daily Living (ADL) is given. These datasets could be used to design and train a real-world action recognition system, to be used in assisted living environments.

Chaquet, Carmona, and Fernández-Caballero [6] provide a fine-grained classification according to the type of actions present in the datasets, as illustrated in figure 43.

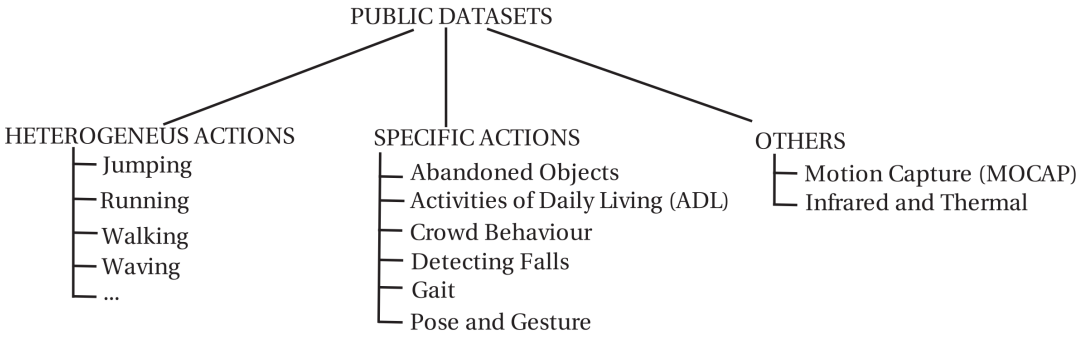


Figure 43: Taxonomy of publicly available datasets provided by [6]

Heterogeneous Actions describes the class of common actions that can appear in a wide variety of scenes and situations. Because of their generality, most benchmarking datasets, as well as the survey of Chaquet, Carmona, and Fernández-Caballero [6] focus on this kind of actions.

Specific Actions denotes the class of action datasets that are used for training an action recognition system for a specific task such as fall detection, gait recognition or crowd behaviour recognition.

Others denotes the category, that is specified by the technique used to capture the action datasets. Examples are Motion Capture, i.e. capturing human motion aided by special instruments such as joint markers and thermal or infrared imaging.

Since the actions of interest in this work are heterogeneous and can appear in any kind of context, we use a different classification scheme. As proposed in [97] action recognition datasets can be roughly categorized into three epochs, which differ in age, source of the videos and complexity.

1. **Early Benchmarking Datasets** provide low resolution videos of a few basic action classes. These datasets are usually recorded under controlled conditions with static cameras, uncluttered backgrounds and actions are performed by actors.
2. **Intermediate Benchmarking Datasets** are larger and more challenging for action recognition algorithms compared to early datasets. Actions in this class are usually sampled from television broadcasts or movies and provide uncontrolled conditions, moving cameras, cluttered backgrounds and occlusion. Given their source, videos in these datasets can be considered high-quality.
3. **Modern Datasets** aim at providing a large number of action classes and examples. Videos in modern datasets are most often sampled from YouTube to provide real-world footage with varying resolutions, different lighting conditions, cuts, cluttered backgrounds and moving cameras. These are the most challenging datasets to date.

4.1 Early Benchmarking Datasets – Controlled Conditions

4.1.1 KTH – 2004

The KTH dataset [15] is an early and widely used benchmarking dataset containing grayscale video-clips of atomic actions performed by 25 different actors in controlled environments. The dataset was created and released by the Swedish KTH Royal Institute of Technology in 2004. It's considered a milestone in computer vision [6] and is the most commonly used publicly available dataset of human actions [63]. It was the largest video dataset of human actions at that time and enabled a systematic and comparable evaluation of action recognition algorithms by using the same input data.

Dataset details: [15]

- 600 videos in controlled conditions.
- 6 action classes recorded in 4 different scenarios (see figure 44).
- 25 different actors.
- Homogeneous, uncluttered backgrounds.
- Static camera.
- 160×120 pixels video-resolution @ $25fps$
- $4s$ average sequence length

Figure 44 shows example frames of the KTH dataset.

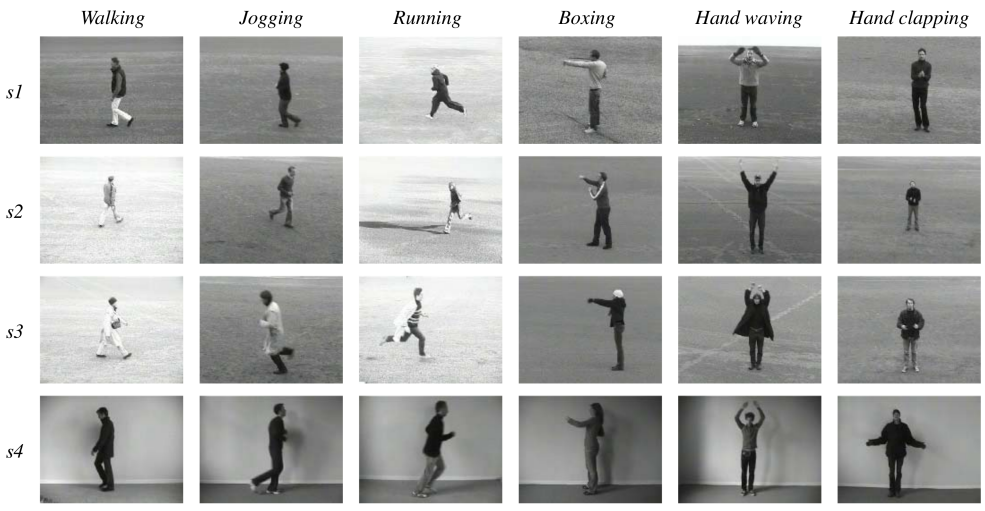


Figure 44: KTH dataset example frames – 6 different actions classes in 4 different recording scenarios: (s1) outdoors, (s2) outdoors with scale change, (s3) outdoors with changing clothes, (s4) indoors [15]

4.1.2 Weizmann – 2005

One year after KTH, the Weizmann dataset [98] (also known as Weizmann actions as space-time shapes dataset) was released in 2005 by the Weizmann Institute of Science in Israel. The dataset provides more action classes than KTH but features less performers per action class. The actions are also recorded in controlled conditions. Additional to the action labels, the silhouettes of the actors as well as the background sequences for background subtraction are provided with the dataset. The actors move horizontally in the scene, so no change of their size due to changing distances to the camera is present. Some actions are repeated in different directions to account for their asymmetrical nature.

Dataset details: [9]

- 90 video clips, each containing multiple instances of a given action class.
- 10 action classes (run, walk, skip, jumping-jack, jump-forward-on-two-legs, jump-in-place-on-two-legs, gallop sideways, wave-two-hands, wave-one-hand and bend)
- 9 different actors.
- Homogeneous, uncluttered backgrounds.
- Static camera.
- 180×144 pixels video-resolution @ $25 fps$.
- $3.66 s$ average clip length.

Figure 45 shows example frames for some of the action classes of Weizmann dataset.

Figure 45: Example frames for the action classes *jumping jacks*, *run*, *walk* and *gallop sideways* along with provided foreground silhouette-masks [98]

4.1.3 IXMAS – 2006

The INRIA Xmas Motion Acquisition Sequences dataset (IXMAS) [99] was initially released in 2006 by the French National Institute for Computer Sciences (INRIA) [99]. It was designed to study robustness of action recognition algorithms against viewpoint changes, actor gender and body size. The dataset therefore contains multi-view shots (5 synchronized cameras) of male and female actors performing atomic actions of the daily-living. The initial release, as described in [99], contained 11 action classes, performed by 10 actors and was later extended.

Dataset details (extended release): [100]

- 13 action classes (checking watch, crossing arms, scratching head, sitting down, getting up, turning around, walking, waving, punching, kicking, pointing, picking up, throwing over head and throwing from bottom up).
- Each action performed 3 times by 11 different actors.

- All actions recorded under controlled conditions (indoors).
- Homogeneous, uncluttered backgrounds.
- Static camera, 5 different perspectives of each performance.

Figure 46 shows the five different camera perspectives of the action *checking watch*.

Figure 46: Action example of *checking watch* captured from five different cameras [99]

4.2 Intermediate Benchmarking Datasets – Television and Movies

4.2.1 UCF Sports – 2008

The UCF Sports Dataset was initially released in 2008 by the Department of Electrical Engineering and Computer Science at the University of Central Florida (UCF), where several human action datasets stem from as well. The dataset contains video clips of sport actions, recorded from broadcast television channels such as the BBC and ESPN. Because of its origin in television, the video data is of professional quality with moving cameras and changing backgrounds.

The initial release of the dataset [101] contained nine action classes: diving, golf swinging, kicking, lifting, horseback riding, running, skating, swinging a baseball bat and pole vaulting Example frames of the first release are shown in figure 47. For a second release [102] pole vaulting as well as swinging a baseball bat have been removed and the additional classes swinging on a bench, swinging on parallel bars and walking have been added. Example frames of the second release are shown in figure 48.

Dataset details (second, final release): [103]

- 150 action clips.
- 10 action classes.
- Moving camera and changing backgrounds.
- 720×480 pixels video-resolution @ 10 *fps*.
- $6.39s$ average clip length.
- 6 to 22 clips per class.

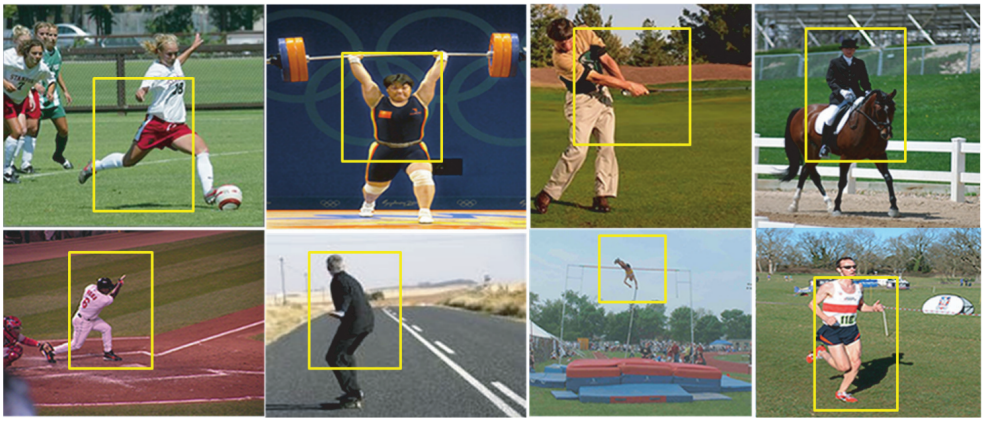


Figure 47: Example frames of UCF Sports Dataset (release 1) [101]

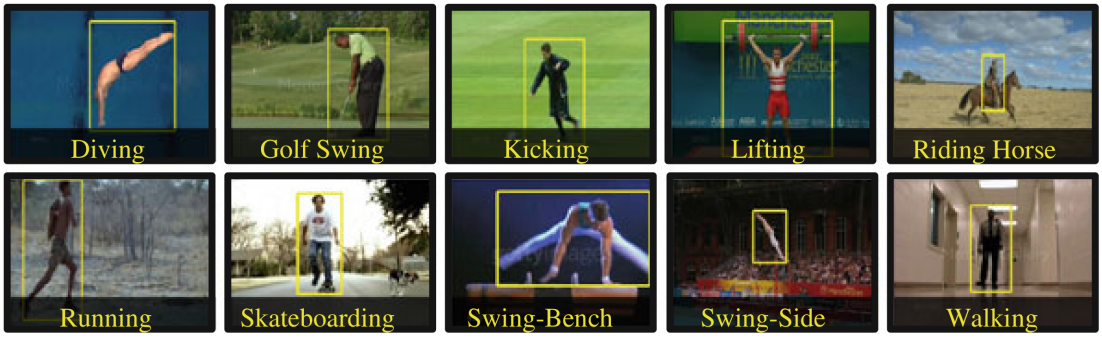


Figure 48: Example frames of UCF Sports Dataset (release 2) [102]

4.2.2 Hollywood – 2008

The Hollywood dataset [50] was released in 2008 at the IRISA Institute in France and contains video samples of human actions from 32 movies, divided into 8 different action classes. The dataset provides a predefined split into training and test set. The test set contains 211 video samples from 20 movies. Two differently annotated training sets with action samples from the remaining 12 movies are provided. The *automatic training set* contains 233 action samples, which are automatically annotated by extracting action labels from the movie scripts. The labels are approximately 60% correct. The *clean training set* contains 219 action samples with manually verified labels.

The Hollywood dataset and its successor Hollywood 2 are considered more challenging than the previously introduced datasets for benchmarking [6]. Since the action clips are sampled from movies, the dataset provides a wide range of variation in persons, gestures, clothing, camera motion, perspective and occlusion. However, since all footage

was filmed under controlled lighting conditions with professional cameras, the datasets are considered not representative for real-world observations [9].

Dataset details: [104]

- 233 + 219 + 211 action clips (automatic training set + clean training set + test set)
- 8 action classes (AnswerPhone, GetOutCar, HandShake, HugPerson, Kiss, SitDown, SitUp and StandUp)
- 240 pixels video-height with varying width @ 24 *fps*

Example frames for all classes of the Hollywood dataset are shown in figure 49.

Figure 49: Example frames for the eight classes of Hollywood I dataset with highest confidence for true/false positives/negatives [50]

4.2.3 Hollywood 2 – 2009

The Hollywood 2 dataset [105] is an extension of the Hollywood dataset and was released in 2009 at the IRISA Institute of France. Besides adding 4 action classes to the Hollywood dataset, Hollywood 2 contains 10 classes of indoor and outdoor scenes to study if action recognition algorithms benefit from correlating scenes and actions. The dataset contains 12 classes of human actions, 10 classes of scenes and features approximately 20.1 hours of video in total from 69 movies. Equivalently to Hollywood 1, automatically and manually verified labels are provided in two training sets.

Dataset details (action clips): [106]

- 810 + 823 + 882 action clips (automatic + clean training set + test set)
- 12 action classes (AnswerPhone, GetOutCar, HandShake, HugPerson, Kiss, SitDown, SitUp, StandUp, Eat, FightPerson, Run, DriveCar)

Figure 50 shows the additional classes in the Hollywood 2 datasets.

(a) DriveCar (b) FightPerson (c) HandShake (d) SitUp

Figure 50: Example frames for the additional classes of Hollywood 2 dataset [105]

4.3 Modern Benchmarking Datasets – Videos in the Wild

4.3.1 UCF11 Youtube Action – 2009

The UCF11 dataset [107], also known as UCF YouTube Action dataset, was released in 2009 by the University of Central Florida (UCF). It contains videos from YouTube in 11 action classes. Since the authors were not involved in the video recognition process, the dataset features a variety of different conditions in the videos: a mix of shaky and steady cameras, different (cluttered) backgrounds, people shown in different scales, varying illumination and low resolution. There are several releases of the dataset, here properties of the last release are reported.

Dataset details: [108]

- 1600 action clips.
- 11 action classes: basketball shooting, biking/cycling, diving, golf swinging, horse back riding, soccer juggling, swinging, tennis swinging, trampoline jumping, volleyball spiking, and walking with a dog.
- Varying spatial resolution (max. 240×320 pixels) @ $29 fps$.
- Clip length between 22 and 900 frames.

Figure 51 shows example frames of the dataset.

Figure 51: Example frames of the 11 classes in the UCF YouTube actions dataset [107]

4.3.2 UCF50 – 2010

The UCF50 dataset [109] is an extension of the UCF11 dataset and was released by the University of Central Florida (UCF) in 2010. It adds 39 action classes to UCF11, which results in a total of 50 action classes.

Dataset details: [9]

- $6,676$ action clips.
- 50 action classes (see figure 52).
- At least 100 clips per class.
- 320×240 pixels video-resolution.

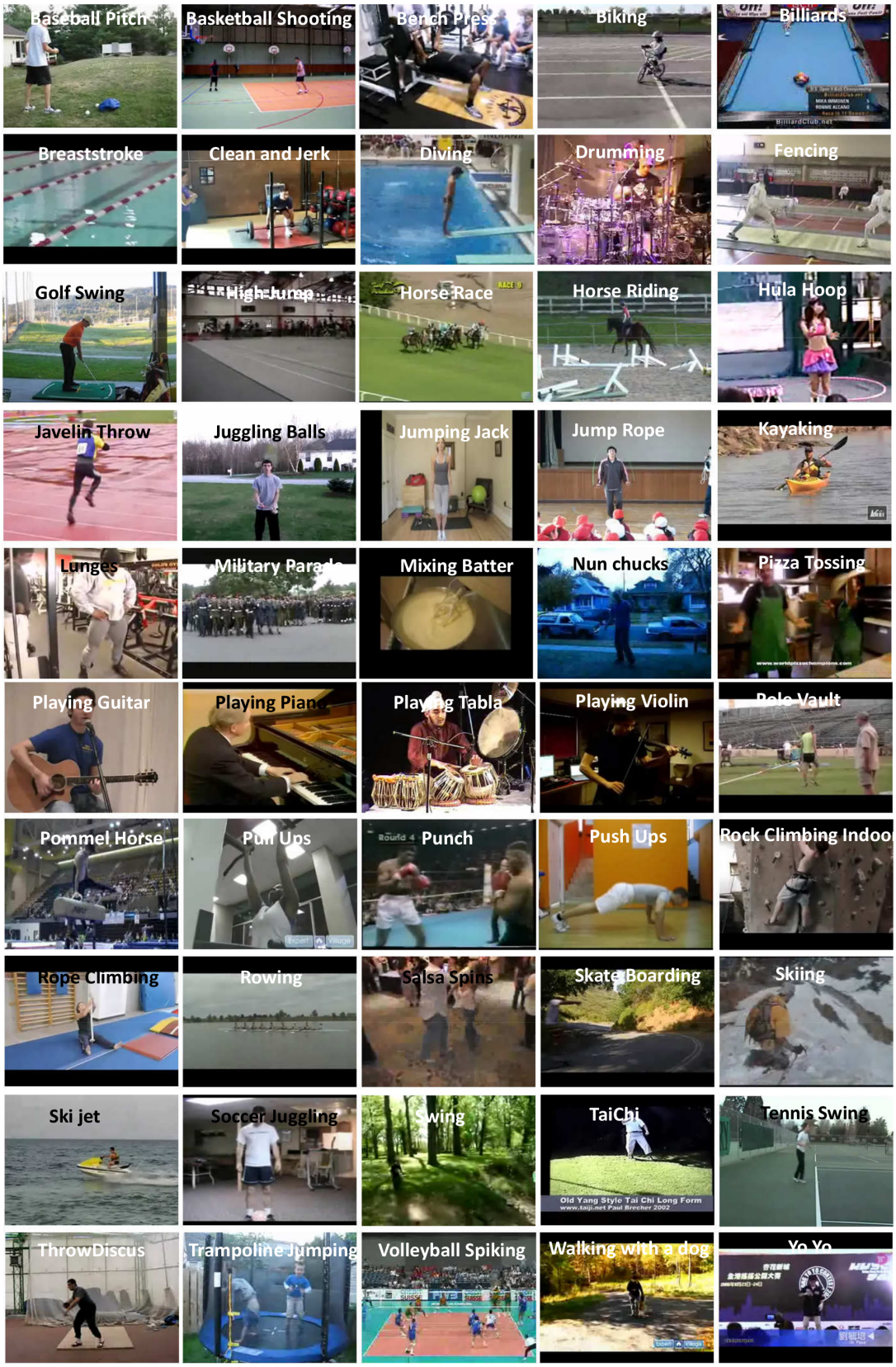


Figure 52: Example frames of the 50 classes in UCF50 [109]

4.3.3 HMDB51 – 2011

The Human Motion Database (HMDB51) [110] was created by the Brown University in the USA and was released in 2011. The HMDB51 dataset consists of $6,849$ action clips in 51 classes sampled mostly from movies, public video databases and YouTube. Each action class contains a minimum of 101 action clips. The source videos presented different resolutions and frame rates. To maintain consistency throughout the dataset, the authors scaled the videos to provide a frame-height of 240 pixels and the width was scaled accordingly. The framerate of all videos was converted to 30 *fps*.

Each video of the dataset contains one action but the exact temporal location of the action is not provided. However, additionally to the action labels, the datasets provides detailed meta information to each action clip: [111]

- Visible body parts: Head, upper body, full body, lower body.
- Camera motion: Motion, static.
- Camera viewpoint: Front, back, left, right.
- Number of people involved in the action: Single, two, three.
- Video quality: good, medium, bad.

Global movement caused by camera/background movement interferes with local motion in the scene, which is essential for action recognition. The authors therefore calculated stabilization mask, which can be used to suppress global motions in the clips up to a certain degree. [110]

The action classes in the dataset can be grouped into the following five categories: [110]

1. General facial actions (e.g. smile, laugh, chew, talk)
2. Facial actions with object manipulation (e.g. smoke, eat, drink)
3. General body movements (e.g. clap hands, climb stairs, jump)
4. Body movements with object interaction (e.g. brush hair, catch, kick ball)
5. Body movements for human interaction (e.g. hug, fencing, kiss)

Figure 53 shows example frames for each of the action classes in HMDB51.

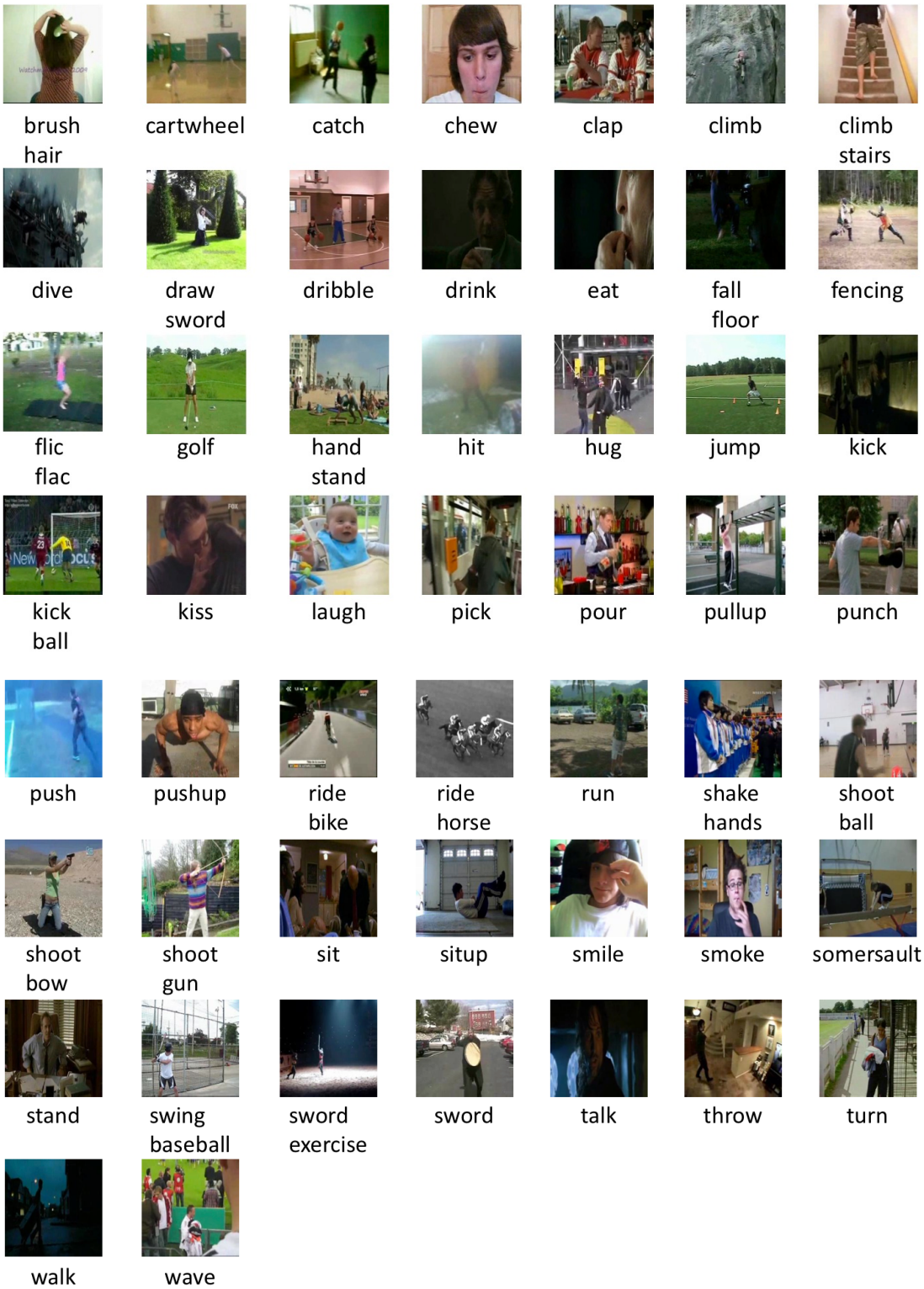


Figure 53: Example frames for the 51 classes in HMDB51 [111]

4.3.4 UCF101 – 2012

The UCF101 dataset [69] is an extension of the UCF50 dataset and was released by the University of Central Florida (UCF) in 2012. It introduces 51 additional classes sampled from YouTube videos, which results in a total of $13,320$ action clips in 101 action classes. The overall length of the dataset is about 27 hours, with an average length of $7.21s$ per action clip. Video clips have a resolution of 320×240 pixels @ 25 *fps*.

Figure 54 shows example frames of UCF101. The action classes can be divided into five categories: [69]

- Blue: Human-object interaction
- Red: Body-motion only
- Purple: Human-human interaction
- Orange: Playing musical instrument
- Green: Sports

4.3.5 ASLAN – 2012

In contrast to the previously presented benchmarking datasets, the *Action Similarity Labeling* collection (ASLAN) [112] emphasizes on a different evaluation protocol for benchmarking action recognition performance. Generally multi-class mean recognition accuracy is used as a performance measure, ASLAN however was designed as binary *same/not same* classification benchmark. Specifically, an algorithm is trained on pairs of action video clips labeled *same/not same* and is then evaluated on determining the similarity of two unknown clips, i.e. determining if they belong to the same class or not. Thereby test-set clips are drawn from different action classes as the training-set clips, which means an algorithm is evaluated on never before seen examples of an unknown class.

The authors Kliper-Gross, Hassner, and Wolf [112] argue, that developing algorithms for measuring the similarity of actions has several advantages:

1. It eases the problem of ambiguous action labels for complex actions, i.e. complex actions can be composed of multiple atomic actions and therefore have multiple actions labels.
2. By focusing on action similarity, instead of on features that define certain actions, the benchmark leads an algorithm towards a generalization ability that is not limited by a given set of actions.
3. The binary nature of the benchmark makes it easier to define tests for a given algorithm.

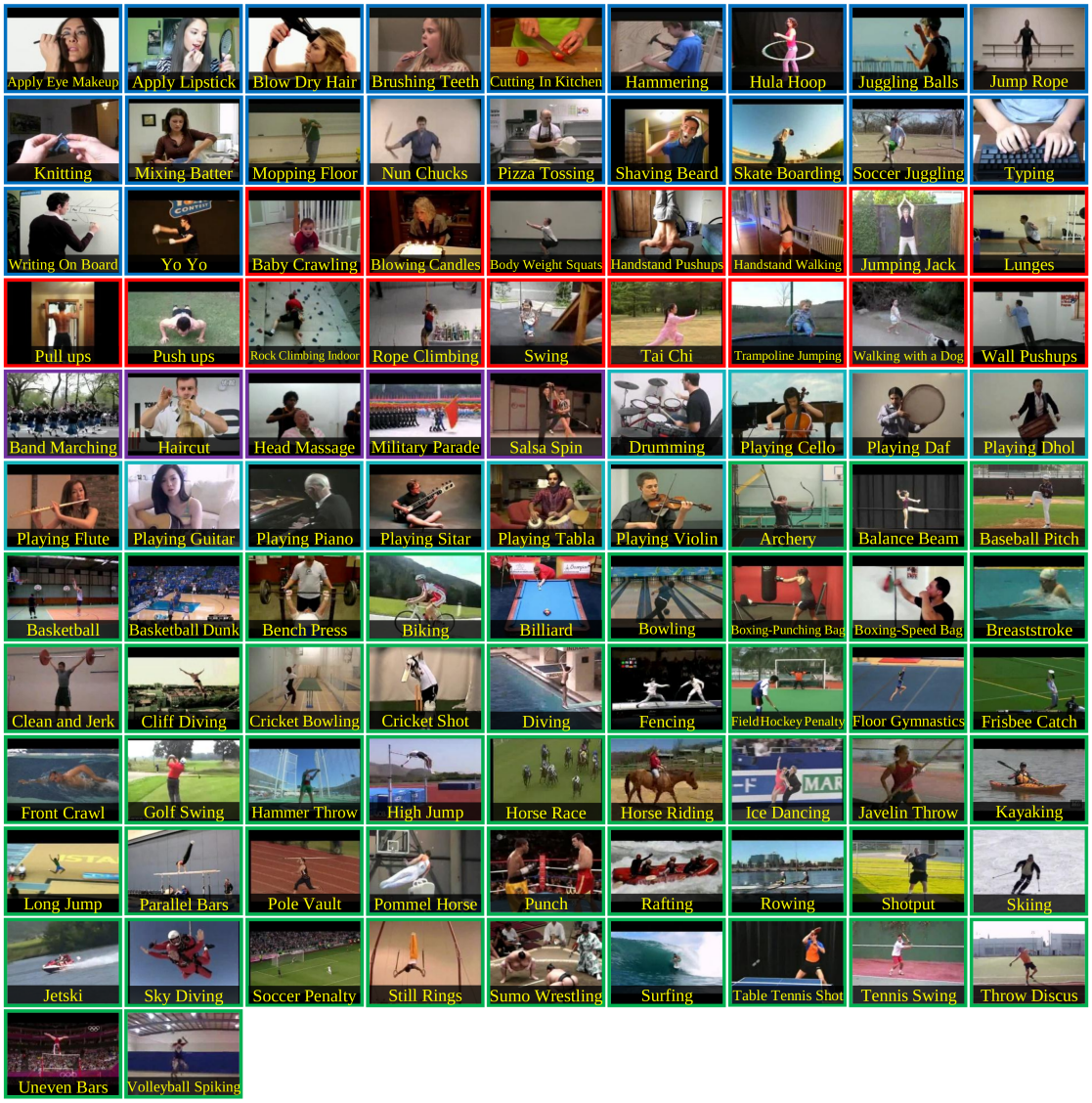


Figure 54: Example frames of the 101 classes in UCF101 [69]

Additionally, determining the similarity of actions can be used in an own class of applications, e.g. when similar clips need to be retrieved autonomously from the internet given a video clip with an unknown action label.

Dataset details: [112]

- $3,631$ unique action instances obtained from YouTube videos.
- 432 action classes, manually labeled.
- 8.5 average samples per class
- 316 classes with more than 1 action sample.
- 71 long samples (duration $> 10s$).
- 187 short sampled (duration $< 1s$).
- Each clip contains one action, no detection needed.

The authors provide two different divisions of their database, which they call *Views*.

1. *View-1* is intended for algorithm development and consists of a training- and testing-set which are mutually exclusive, i.e. do not share action classes. The training-set consists of $1,200$ video pairs, 600 of which labeled *same* and 600 labeled *not same*. The testing-set consists of 600 video pairs, 300 labeled *same* and 300 labeled *not same*.
2. *View-2* is intended for reporting the final accuracy of the classifier and consists of 10 mutually exclusive splits of the database. Each split contains 600 video pairs, 300 labeled *same* and 300 labeled *not same*.

The authors advise reporting the final performance of the algorithm by doing 10-fold cross-validation using the splits of *View-2*.

Figure 55: Example frames of video pairs labeled *same* [112]

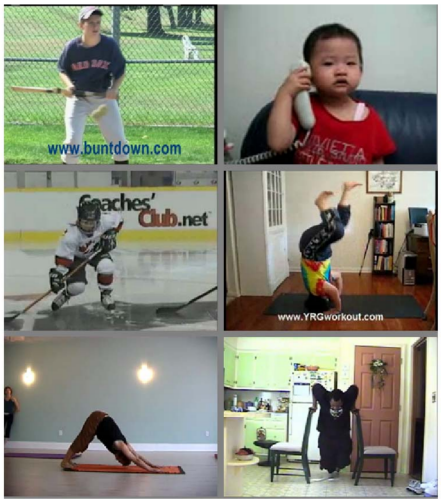


Figure 56: Example frames of video pairs labeled *not same* [112]

4.3.6 Sports-1M – 2014

The Sports-1M dataset [11] was released in 2014 by the Computer Science Department at Stanford University. The dataset contains links to 1,133,158 sports videos from YouTube, which have been labeled automatically into 487 different sports classes by analyzing the text meta-data provided with the videos.

The 487 action classes are arranged according to a manually created taxonomy with more general classes at the top and more specific classes towards the leaves. Each class contains about 1000 to 3000 videos and about 5% of the videos have multiple class labels. The videos are therefore weakly annotated with potentially wrong labels and may provide a lot of variations on the frame level, i.e. the actions are not temporally localized in the videos and may be accompanied by unrelated content (e.g. spectators, interviews or scoreboards).

For evaluation, the authors provide a split by assigning 70% of the videos to a training set, 20% to a test set and 10% to a validation set. Unfortunately around 7% of the dataset is not available anymore, because the videos were removed from YouTube [78].

Because of the above stated properties of the Sports-1M it should be used with caution [9], although it is the biggest action recognition dataset available to date.

4.3.7 YouTube-8M – 2016

The YouTube-8M dataset [113] is, comparable to Sports-1M, a large-scale collection of annotated YouTube video IDs released in 2016 by Google. The dataset is designed as a multi-class benchmark, featuring about 8 million videos with 4800 different video labels in total. Video labels were obtained automatically from video meta-data and Google statistics. The authors therefore note, that the labelling is very precise [113], yet not noiseless.

The dataset is organized in 24 top-level categories, shown in figure 57.

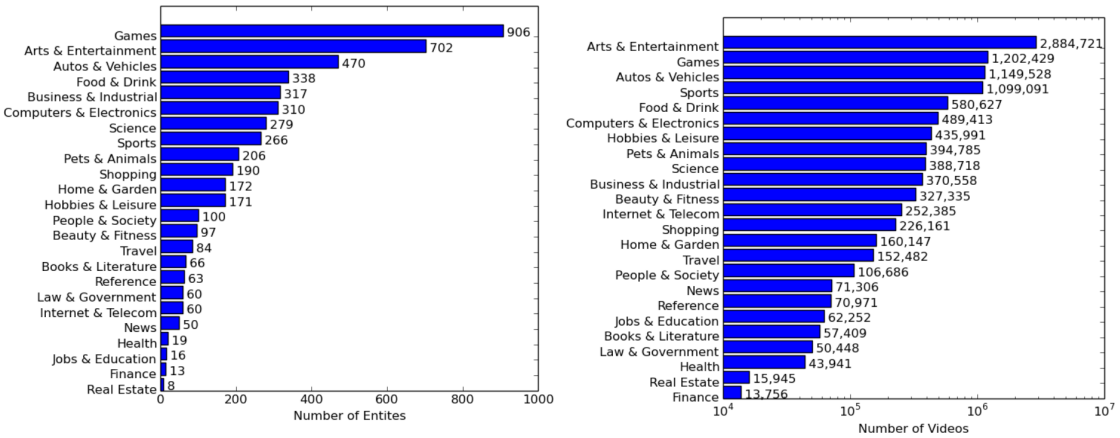


Figure 57: YouTube-8M statistics. *a*) Number of entities, i.e. action labels per top-level category. *b*) Number of training videos per top-level category. [113]

Top 7 action labels for each category are shown in the following table 16.

Top-level Category	1st Entity	2nd Entity	3rd Entity	4th Entity	5th Entity	6th Entity	7th Entity
Arts & Entertainment	Concert	Animation	Music video	Dance	Guitar	Disc jockey	Trailer
Autos & Vehicles	Vehicle	Car	Motorcycle	Bicycle	Aircraft	Truck	Boat
Beauty & Fitness	Fashion	Hair	Cosmetics	Weight training	Hairstyle	Nail	Mascara
Books & Literature	Book	Harry Potter	The Bible	Writing	Magazine	Alice	E-book
Business & Industrial	Train	Model aircraft	Fish	Water	Tractor pulling	Advertising	Landing
Computers & Electronics	Personal computer	Video game console	iPhone	PlayStation 3	Tablet computer	Xbox 360	Microsoft Windows
Finance	Money	Bank	Foreign Exchange	Euro	United States Dollar	Credit card	Cash
Food & Drink	Food	Cooking	Recipe	Cake	Chocolate	Egg	Eating
Games	Video game	Minecraft	Action-adventure game	Strategy video game	Sports game	Call of Duty	Grand Theft Auto V
Health	Medicine	Raw food	Ear	Glasses	Injury	Dietary supplement	Dental braces
Hobbies & Leisure	Fishing	Outdoor recreation	Radio-controlled model	Wedding	Christmas	Hunting	Diving
Home & Garden	Gardening	Home improvement	House	Kitchen	Garden	Door	Swimming pool
Internet & Telecom	Mobile phone	Smartphone	Telephone	Website	Sony Xperia	Google Nexus	World Wide Web
Jobs & Education	School	University	High school	Teacher	Kindergarten	Campus	Classroom
Law & Government	Tank	Firefighter	President of the U.S.A.	Soldier	President	Police officer	Fighter aircraft
News	Weather	Snow	Rain	News broadcasting	Newspaper	Mattel	Hail
People & Society	Prayer	Family	Play-Doh	Human	Dragon	Angel	Tarot
Pets & Animals	Animal	Dog	Horse	Cat	Bird	Aquarium	Puppy
Real Estate	House	Apartment	Condominium	Dormitory	Mansion	Skyscraper	Loft
Reference	Vampire	Bus	River	City	Mermaid	Village	Samurai
Science	Nature	Robot	Eye	Ice	Biology	Skin	Light
Shopping	Toy	LEGO	Sledding	Doll	Shoe	My Little Pony	Nike; Inc.
Sports	Motorsport	Football	Winter sport	Cycling	Basketball	Gymnastics	Wrestling
Travel	Amusement park	Hotel	Airport	Beach	Roller coaster	Lake	Resort
Full vocabulary	**Vehicle**	**Concert**	**Animation**	**Music video**	**Video game**	**Motorsport**	**Football**

Table 16: Top 7 class label for each top-level category [113]

4.4 Activities of Daily Living (ADL) Datasets

4.4.1 URADL – 2009

The University of Rochester Activities of Daily Living Dataset (URADL) [114] was released in 2009. It contains high-resolution videos of daily-living actions, which were recorded in front of a static background (kitchen) by a tripod-mounted, static camera.

The dataset contains 10 action classes: answering a phone, dialing a phone, looking up a phone number in a telephone directory, writing a phone number on a whiteboard, drinking a glass of water, eating snack chips, peeling a banana, eating a banana, chopping a banana, and eating food with silverware. Each action was performed three times by five different actors, resulting in a total of 150 videos in the dataset and 15 actions per class.

Each video provides a resolution of 1280×720 pixels at $30 fps$. A video contains a complete action and lasts between 10 and 60 seconds until the action is finished.

Example frames for the ten action classes are shown in figure 58.

Figure 58: Example frames for the action classes in the URADL dataset. [115]

4.4.2 Multiple Cameras Fall Dataset – 2010

The Multiple Cameras Fall dataset [116] was released in 2010 by the University of Montreal, Canada and contains videos of simulated falls and daily activities recorded with a multi-camera system. It is designed for developing healthcare vision systems to detect falls of elderly people at home.

The dataset is small. It contains recordings of 24 different events in eight different perspectives, performed by a single actor. 22 of the events include a fall, which happen during confounding actions such as walking, housekeeping and activities with similar appearance as falls (e.g. sitting down or crouching). 2 of the recorded events only include confounding actions and no falls. The scenes include occlusions from furniture or other moving objects (see figure 59). The dataset only provides raw recordings without action annotations.

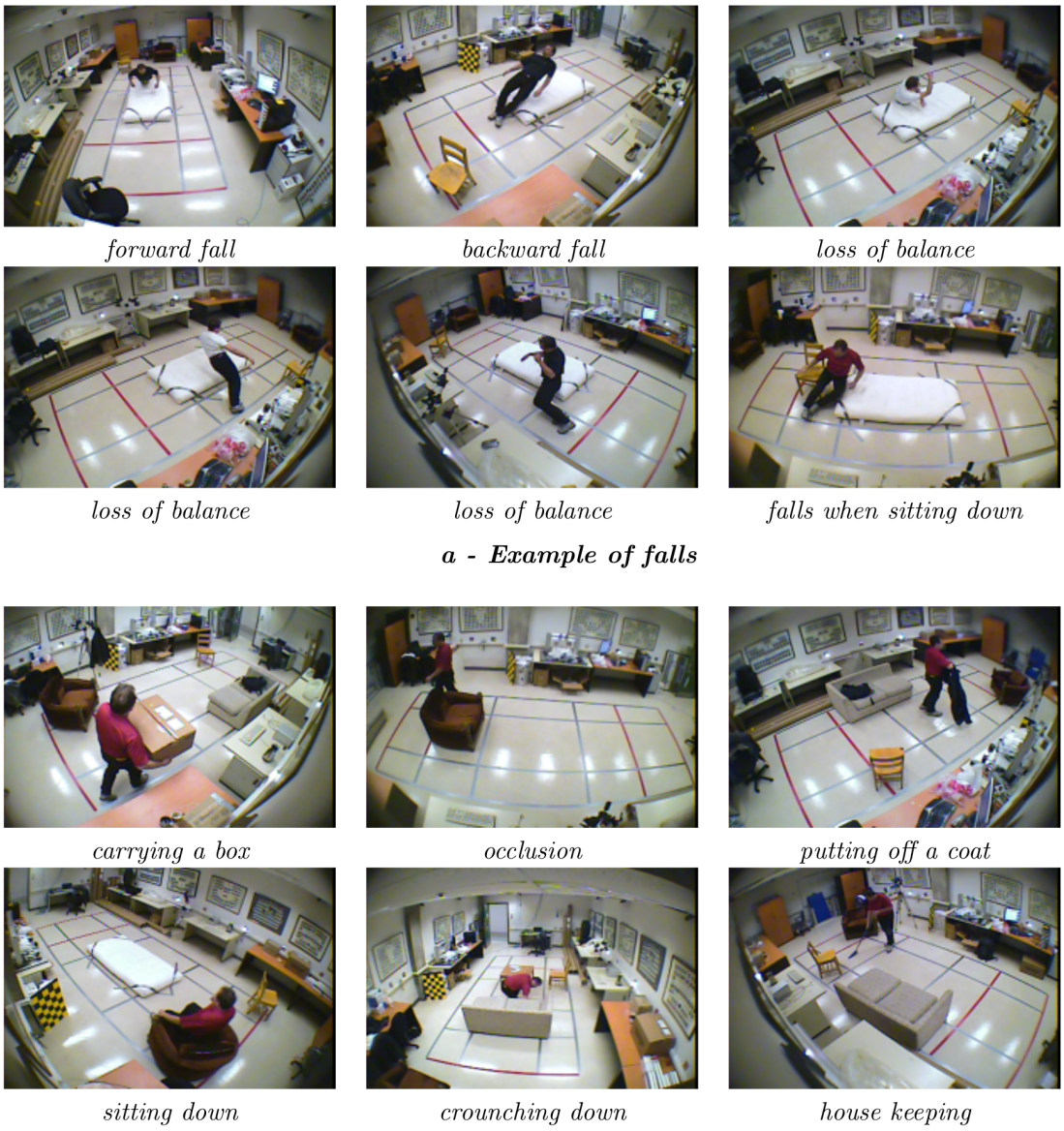


Figure 59: Example frames of falls (a) and confounding activities performed before or after a fall (b) [116]

4.4.3 MPII Cooking Activities Dataset – 2012

The MPII Cooking Activities Dataset [117] was released in 2012 by the Max Planck Institute of Informatics (Germany) to provide a dataset for fine-grained activity recognition The dataset contains videos of 12 participant preparing one to six out of 14 different dishes. The preparation of a dish has been recorded continuously, to apply detection as

well as recognition algorithms to the dataset. In total 44 videos were recorded with a combined length of 8 hours featuring 65 different cooking action classes in 5,609 annotated action clips. The videos have a resolution of 1624×1224 and were recorded in a realistic setting with a stationary camera mounted to the ceiling.

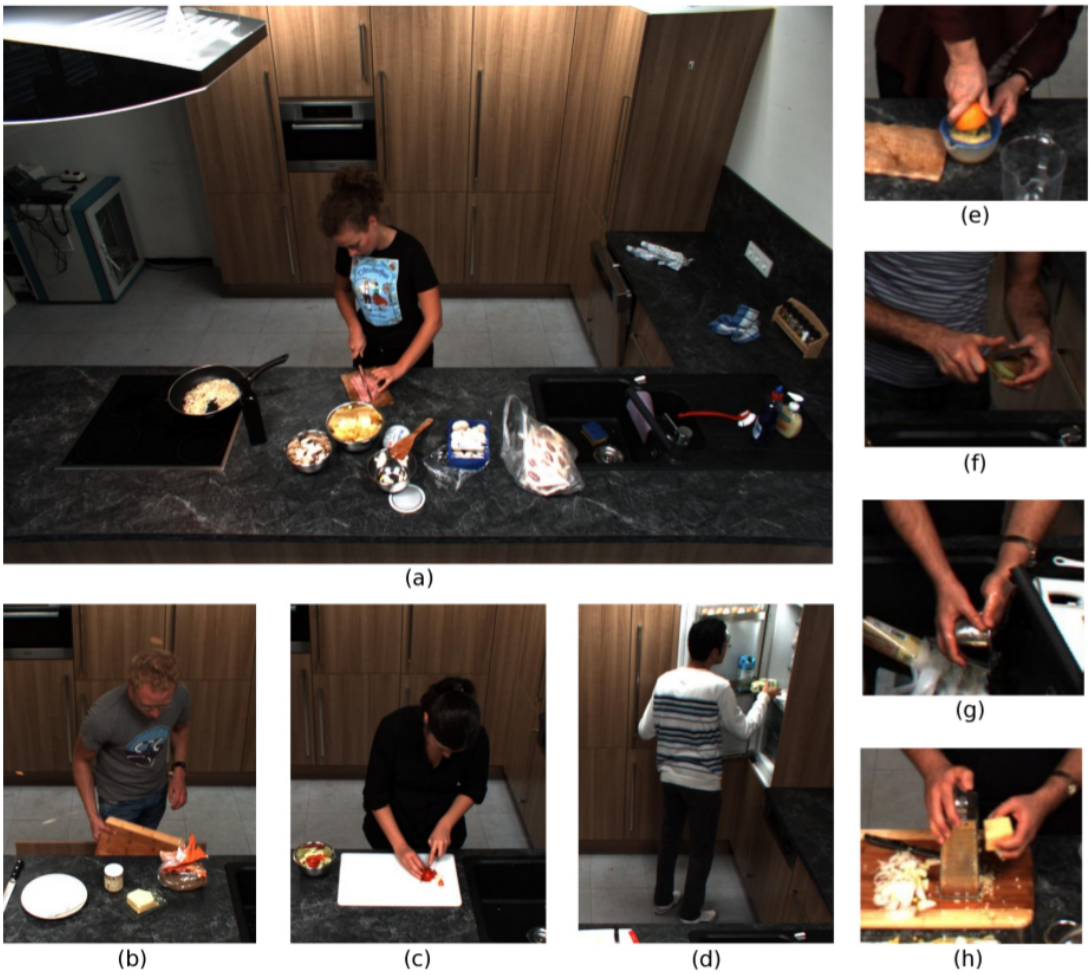


Figure 60: Example frame and crops of cooking activities. (a) cut slices, (b) take out from drawer, (c) cut dice, (d) take out from fridge, (e) squeeze, (f) peel, (g) wash object, (h) grate. [117]

4.4.4 ActivityNet – 2015

ActivityNet [118] is a large-scale activity recognition dataset released in 2015 by researchers of the Universidad del Norte in Colombia and the King Abdullah University of Science and Technology in Saudi Arabia. The dataset contains a wide range of videos with daily living activities, which were obtained from YouTube and labelled manually.

The latest release of the dataset (release 1.3 of march 2016, as available from the official website [119]) consists of annotated links to YouTube videos, organized in 200 activity class. The labels are provided in JSON-format and contain the video-ID, the activity labels along with beginning- and end-time of the action, total video-duration, resolution and url. A single video can contain multiple activity instances of different activity classes. The majority of videos is available in a resolution of 1280×720 pixels and lasts between 5 and 10 minutes.

The database is divided as follows:

- $10,024$ training videos (containing $15,410$ activity instances)
- $4,926$ validation videos (containing $7,654$ activity instances)
- $5,044$ testing videos (labels withheld)

Besides annotated video-links, the dataset provides a taxonomy of daily living activities in the ActivityNet hierarchy with four levels of granularity. The top-level categories being:

- Personal Care
- Eating and Drinking
- Household
- Caring and Helping
- Working
- Socializing and Leisure
- Sports and Exercises

Following figure 61 shows the visualization of the activity category *Household*.

Figure 61: Taxonomy of top-level activity category *Household* [118]

ActivityNet is the biggest manually annotated video-dataset available to date. It is applicable for detection as well as recognition algorithms and was used in the ActivityNet Challenge along CVPR 2016. Although not explicitly mentioned by the authors, it is unclear how many of the videos are currently still available on YouTube (see Sports-1M).

4.4.5 Charades – 2016

The Charades dataset [120] was released in 2016 by the Allen Institute of Artificial Intelligence and contains a large number of real-world activities of the daily living. The authors note, that sampling this kind of actions from YouTube often yields atypical results, since YouTube videos are created for entertaining and do not represent real-world activities well. To address this problem, the Charades dataset was created by crowd-sourcing: 267 people from three continents were provided with scripts of daily-living activities to perform and record in their homes. This results in 9,848 videos featuring a combination of 46 objects and 30 actions in 15 scenes. There are on average 6.8 labeled actions per video and 15% of the videos show more than one person. Video annotation include: multiple free-text descriptions, action class labels, action time-intervals and classes of interacted objects.

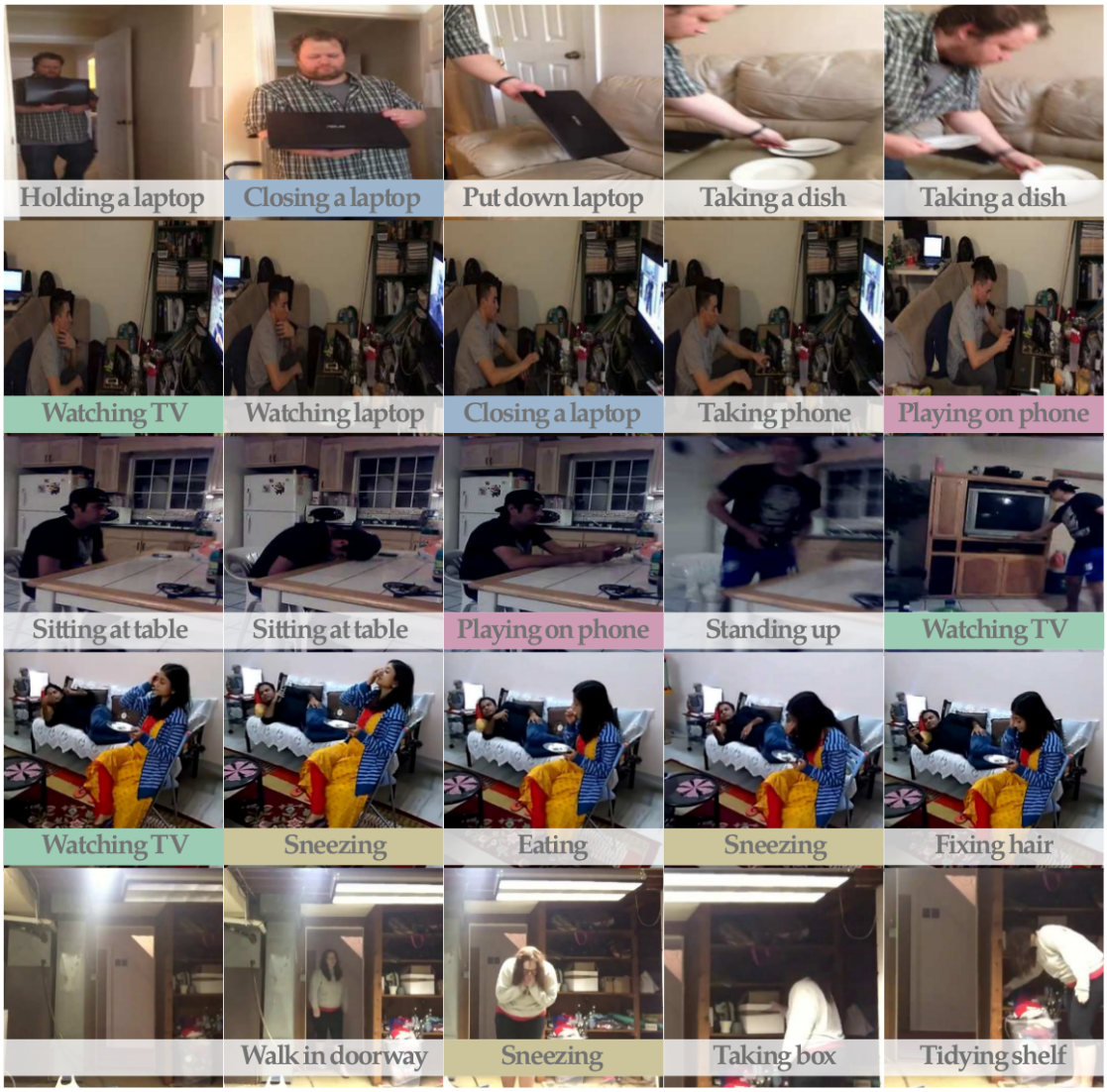


Figure 62: Example frames of five videos in the Charades dataset. A video contains sever different actions, that can occur in many different configurations. [120]

Dataset details:

- $9,848$ videos of $30.1s$ average length.
- Training set: $7,985$ videos; Test set: $1,863$ videos.
- 157 action classes (derived from combinations of 46 object classes, 30 verbs and 15 indoor scenes).
- 66.500 temporally localized actions.

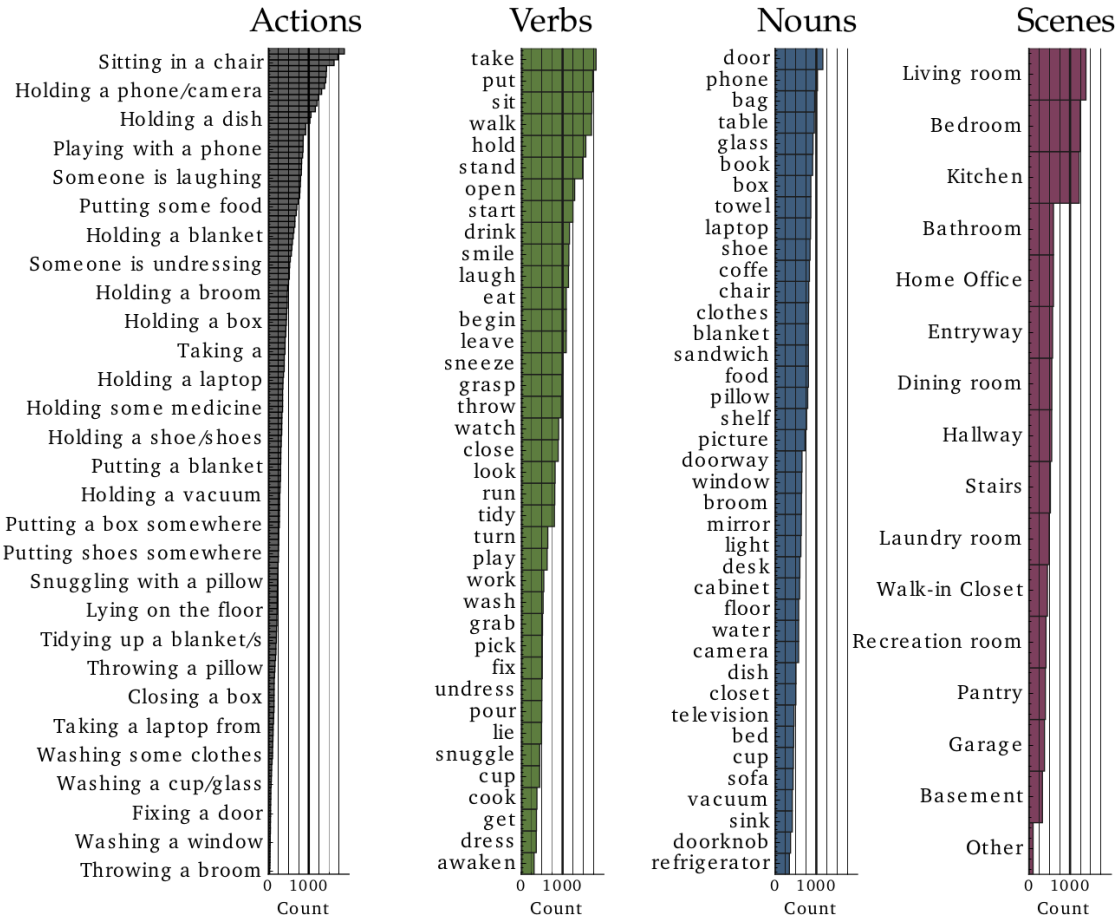


Figure 63: Distribution of actions (every fifth label shown), verbs, nouns and scenes in the Charades dataset [118]

5 Comparison and Evaluation

Using the analysis of conventional and deep learning methods in action recognition, this section provides a comparison and evaluation of the presented methods. Results of deep learning approaches, which were reviewed in this report, are given in the following table 17.

	Method	Year	#Layers (#Conv. Layers)	KTH	UCF-101	HMDB-51	Sports-1M (Hit @ 1)
[56]	3D CNN	2010	7 (3)	90.20			
[63]	3D CNN + LSTMs	2011	8 (3)	92.17			
[11]	Slow Fusion	2014	10 (5)		65.40		60.90
[42]	C3D	2015	15 (8)		85.20		60.00
[71]	LTC	2016	13 (5)		82.20	59.00	
[44]	Two-stream CNN	2014	12 (5)		88.00	59.40	
[78]	ConvPooling + LSTM	2015	22 (10)		88.60		73.10
[43]	Very Deep Two-stream	2015	16 (13)		91.40		
[81]	Temporal Pooling	2016	16 (13)		92.50	65.40	
[55]	TDD	2015	11 (5)		91.50	65.90	
[91]	LSTM Auto-encoder	2015	2		84.30	44.00	
[86]	ConvDBN	2016	3		55.60		
[93]	Siamese Triplet CNN	2016	8 (5)		50.90	19.80	

Table 17: Best reported action recognition accuracies of reviewed deep learning approaches.

We observe from the distribution of used benchmarking datasets, that UCF-101 and HMDB-51 can be seen as the current de facto standard benchmarks in action recognition, although UCF-101 is considered extremely small [43]. Judging by the best performance on UCF-101, the approaches of Wang et al. [43], Feichtenhofer, Pinz, and Zisserman [81] and Wang, Qiao, and Tang [55] perform best. These approaches base on 2D convolutional neural network architectures, that were initially designed for image processing tasks and take advantage of pre-training on large-scale image datasets. [43] and [81] utilise deep ConvNets in an two-stream setup. [55] apply a hybrid conventional-deep approach by using a two-stream ConvNet as generic feature extractor along densely sampled trajectories.

Among the approaches, which explicitly incorporate the spatio-temporal structure of videos in the used architecture by implementing 3D convolutions, the *C3D* approach of Tran et al. [42] performs best. Tran et al. [42] use a 3D ConvNet which is deeper than comparable approaches (three additional convolutional layers) and leverage pre-training on the Sports-1M dataset. Since *C3D* is designed for reusability on different vision tasks, an implementation was made publicly available in the Caffe framework [70].

Given these results it can be derived, that deeper architectures and bigger datasets improve performance in action recognition from video using deep learning approaches. This has also been observed in the area of object recognition from still images [2, 3, 4].

Table 18 shows the performance of state-of-the-art hand-crafted feature methods in action

recognition.

	Method	Year	Hollywood2	HMDB51	Olympic Sports	UCF50	UCF101
[41]	DT + FV	2011	60.10	52.20	84.70	88.60	
[41]	iDT + FV	2013	64.30	57.20	91.10	91.20	85.90
[47]	MVSV	2014		55.90			83.50
[46]	MIFS + FV	2015	68.00	65.10	91.40	94.40	89.10

Table 18: Action recognition accuracies of state-of-the-art hand-crafted feature methods.

The results on UCF-101 show: Although several deep learning based approaches outperform conventional hand-crafted feature methods, the latter nonetheless yield competitive results, especially the approach of Lan et al. [46]. Reasons for this can be found in the sizes of available action recognition video datasets. Table 19 provides a comparison of currently available datasets.

Benchmarks	Year	#Videos	#Labelled Instances	#Action Classes
KTH	2004	600		6
Weizmann	2005	90		10
IXMAS	2006	2,145	429	13
UCF Sports	2008	150	150	10
Hollywood	2008	663	663	8
Hollywood 2	2009	1,700	1,700	12
UCF11	2009	1,600	1,600	11
UCF50	2010	6,676	6,676	50
HMDB51	2011	6,849	6,849	51
UCF101	2012	13,320	13,320	101
ASLAN	2012	3,631	3,631	432
Sports-1M	2014	1,133,158	1,133,158	487
YouTube-8M	2016	8,264,650	8,264,650	4,800

Table 19: Size comparison of reviewed benchmarking datasets.

Although the Sports-1M dataset is roughly 100 times bigger than UCF101, most approaches rely on the smaller UCF101 or HMDB51 for action recognition, because of the label noise in YouTube-1M [81]. There is a noticeable trend towards larger datasets however. The very recently released YouTube-8M dataset is roughly eight times bigger than YouTube-1M, but still does not match the size of common image datasets (ImageNet contains $14,197,122$ images [121]) The difficulty to obtain large-scale video datasets originates in videos being more difficult to store and annotate than images [11].

Given its recency, the YouTube-8M has not been used widely in the action recognition community. Considering its unmatched size in video datasets so far, the creators of the set note, that it will most likely improve the performance of deep learning approaches in action recognition [113].

Table 20 summarizes the characteristics of currently available datasets, that contain

videos of the daily-living.

ADL Datasets	Year	#Videos	#Labelled Instances	#Action Classes
URADL	2009	150	150	10
MCFD	2010	24 x 8		
MPII	2012	44	5,609	65
ActivityNet	2015	14,950	23,064	200
Charades	2016	9,848	66,500	157

Table 20: Size comparison of reviewed daily-living datasets.

The Charades dataset [120] and ActivityNet [118] provide comparably large collections of ADL videos. Especially the Charades dataset has the advantage of containing videos, that were collected in real-world environments by regular people. In order to implement an action recognition system in an assisted living environment, we recommend applying the *C3D* approach [42] to one or possibly both of these datasets. The advantages of *C3D* are:

1. It is the deepest model implementing 3D convolutions.
2. A model pre-trained on Sports-1M is publicly available.
3. It was designed to be repurposed on different vision tasks.

6 Conclusion and Future Directions

This report provides a detailed review of current deep learning approaches in human action recognition from video along with a comparison to conventional hand-crafted feature based approaches. Results show, that deep learning yields state of the art performance, but conventional hand-crafted feature based methods still perform competitive. This has been observed in the action recognition community as well [71].

Two reasons for this phenomenon can be identified:

1. Current deep learning architectures in video action recognition are shallow compared to their image based counterparts [43], because videos are higher dimensional than images and processing them is computationally more expensive.
2. Available action recognition datasets are too small, since videos are more difficult to store and annotate than images [11][44][43]

Considering the challenge of obtaining sufficient training data for deep learning architectures, approaches that leverage data from more than one dataset can be identified as future directions of research:

- **Unsupervised Pre-training** as presented in section 3.4 of this report and Varol, Laptev, and Schmid [71].
- **Transfer Learning** was conducted by Karpathy et al. [11].
- **Multi-task Learning** as implemented by Simonyan and Zisserman [44].

References

[1] Ronald Poppe. "A Survey on Vision-Based Human Action Recognition". In: *Image and vision computing* (2010). URL: http://www.sciencedirect.com/science/article/pii/S0262885609002704.

[2] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *arXiv preprint arXiv:1409.1556* (2014). URL: https://pdfs.semanticscholar.org/45c6/a85a359be655f459516919138a46ae516621.pdf.

[3] Christian Szegedy et al. "Going Deeper with Convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html.

[4] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *arXiv preprint arXiv:1512.03385* (2015). URL: http://arxiv.org/abs/1512.03385.

[5] Jake K. Aggarwal and Michael S. Ryoo. "Human Activity Analysis: A Review". In: *ACM Computing Surveys (CSUR)* (2011). URL: http://dl.acm.org/citation.cfm?id=1922653.

[6] Jose M. Chaquet, Enrique J. Carmona, and Antonio Fernández-Caballero. "A Survey of Video Datasets for Human Action and Activity Recognition". In: *Computer Vision and Image Understanding* (2013). URL: http://romisatriawahono.net/lecture/rm/survey/computer%20vision/Chaquet%20-%20Human%20Activity%20Recognition%20-%202013.pdf.

[7] Martin Längkvist, Lars Karlsson, and Amy Loutfi. "A Review of Unsupervised Feature Learning and Deep Learning for Time-Series Modeling". In: *Pattern Recognition Letters* (June 2014). ISSN: 01678655. URL: http://linkinghub.elsevier.com/retrieve/pii/S0167865514000221.

[8] Samitha Herath, Mehrtash Harandi, and Fatih Porikli. "Going Deeper into Action Recognition: A Survey". In: *arXiv preprint arXiv:1605.04988* (2016). URL: http://arxiv.org/abs/1605.04988.

[9] Soo Min Kang and Richard P. Wildes. "Review of Action Recognition and Detection Methods". In: (Oct. 21, 2016). arXiv: 1610.06906 [cs]. URL: http://arxiv.org/abs/1610.06906.

[10] Guangchun Cheng et al. "Advances in Human Action Recognition: A Survey". In: *arXiv preprint arXiv:1501.05964* (2015). URL: http://arxiv.org/abs/1501.05964.

[11] Andrej Karpathy et al. "Large-Scale Video Classification with Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Karpathy_Large-scale_Video_Classification_2014_CVPR_paper.html.

[12] Aaron F. Bobick and James W. Davis. "The Recognition of Human Movement Using Temporal Templates". In: *IEEE Transactions on pattern analysis and machine intelligence* (2001). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=910878.

[13] Gunnar Johansson. "Visual Motion Perception." In: *Scientific American* (1975). URL: http://psycnet.apa.org/psycinfo/1975-28753-001.

[14] Yaser Sheikh, Mumtaz Sheikh, and Mubarak Shah. "Exploring the Space of a Human Action". In: *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*. IEEE, 2005. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1541250.

[15] 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*. IEEE, 2004. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1334462.

[16] Heng Wang et al. "Evaluation of Local Spatio-Temporal Features for Action Recognition". In: *BMVC 2009-British Machine Vision Conference*. BMVA Press, 2009. URL: https://hal.inria.fr/inria-00439769/.

[17] Ivan Laptev. "On Space-Time Interest Points". In: *International Journal of Computer Vision* (2005). URL: http://link.springer.com/article/10.1007/s11263-005-1838-7.

[18] 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. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1570899.

[19] Chris Harris and Mike Stephens. "A Combined Corner and Edge Detector." In: *Alvey Vision Conference*. Citeseer, 1988. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.434.4816&rep=rep1&type=pdf.

[20] Florent Perronnin and Christopher Dance. "Fisher Kernels on Visual Vocabularies for Image Categorization". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2007. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4270291.

[21] Hervé Jégou et al. "Aggregating Local Descriptors into a Compact Image Representation". In: *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5540039.

[22] Xiaojiang Peng et al. "Bag of Visual Words and Fusion Methods for Action Recognition: Comprehensive Study and Good Practice". In: *arXiv preprint arXiv:1405.4506* (2014). URL: http://arxiv.org/abs/1405.4506.

[23] Trevor Darrell and Alex Pentland. "Space-Time Gestures". In: *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*. IEEE, 1993. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=341109.

[24] D. M. Gavrila, L. S. Davis, et al. "Towards 3-D Model-Based Tracking and Recognition of Human Movement: A Multi-View Approach". In: *International Workshop on Automatic Face-and Gesture-Recognition*. Citeseer, 1995. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.56.5329&rep=rep1&type=pdf.

[25] Ashok Veeraraghavan, Rama Chellappa, and Amit K. Roy-Chowdhury. "The Function Space of an Activity". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. IEEE, 2006. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1640855.

[26] Nuria Oliver, Eric Horvitz, and Ashutosh Garg. "Layered Representations for Human Activity Recognition". In: *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*. IEEE, 2002. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1166960.

[27] Dong Zhang et al. "Modeling Individual and Group Actions in Meetings: A Two-Layer Hmm Framework". In: *Computer Vision and Pattern Recognition Workshop, 2004. CVPRW'04. Conference on*. IEEE, 2004. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1384912.

[28] Peng Dai et al. "Group Interaction Analysis in Dynamic Context". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* (2008). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4407211.

[29] Shaogang Gong and Tao Xiang. "Recognition of Group Activities Using Dynamic Probabilistic Networks". In: *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. IEEE, 2003. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1238423.

[30] Yuri A. Ivanov and Aaron F. Bobick. "Recognition of Visual Activities and Interactions by Stochastic Parsing". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2000). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=868686.

[31] Darnell Moore and Irfan Essa. "Recognizing Multitasked Activities from Video Using Stochastic Context-Free Grammar". In: *AAAI/IAAI*. 2002. URL: http://www.aaai.org/Papers/AAAI/2002/AAAI02-116.pdf.

[32] David Minnen, Irfan Essa, and Thad Starner. "Expectation Grammars: Leveraging High-Level Expectations for Activity Recognition". In: *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*. IEEE, 2003. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1211525.

[33] Seong-Wook Joo and Rama Chellappa. "Attribute Grammar-Based Event Recognition and Anomaly Detection". In: *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*. IEEE, 2006. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1640550.

[34] John E. Hopcroft. *Introduction to Automata Theory, Languages and Computation: For VTU, 3/E*. Pearson Education India, 1979.

[35] James F. Allen. "Maintaining Knowledge about Temporal Intervals". In: *Communications of the ACM* (1983). URL: http://dl.acm.org/citation.cfm?id=358434.

[36] James F. Allen and George Ferguson. "Actions and Events in Interval Temporal Logic". In: *Journal of logic and computation* (1994). URL: http://logcom.oxfordjournals.org/content/4/5/531.short.

[37] Claudio S. Pinhanez and Aaron F. Bobick. "Human Action Detection Using Pnf Propagation of Temporal Constraints". In: *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*. IEEE, 1998. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=698711.

[38] Jeffrey Mark Siskind. "Grounding the Lexical Semantics of Verbs in Visual Perception Using Force Dynamics and Event Logic". In: *Journal of Artificial Intelligence Research* (2001). URL: http://www.academia.edu/download/45992641/Grounding_the_Lexical_Semantics_of_Verbs20160527-28612-cisslg.pdf.

[39] Ram Nevatia, Tao Zhao, and Somboon Hongeng. "Hierarchical Language-Based Representation of Events in Video Streams". In: *Computer Vision and Pattern Recognition Workshop, 2003. CVPRW'03. Conference on*. IEEE, 2003. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4624298.

[40] Michael S. Ryoo and Jake K. Aggarwal. "Recognition of Composite Human Activities through Context-Free Grammar Based Representation". In: *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*. IEEE, 2006. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1640961.

[41] Heng Wang and Cordelia Schmid. "Action Recognition with Improved Trajectories". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2013. URL: http://www.cv-foundation.org/openaccess/content_iccv_2013/html/Wang_Action_Recognition_with_2013_ICCV_paper.html.

[42] Du Tran et al. "Learning Spatiotemporal Features With 3D Convolutional Networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015. URL: http://www.cv-foundation.org/openaccess/content_iccv_2015/html/Tran_Learning_Spatiotemporal_Features_ICCV_2015_paper.html.

[43] Limin Wang et al. "Towards Good Practices for Very Deep Two-Stream ConvNets". In: *arXiv preprint arXiv:1507.02159* (2015). URL: http://arxiv.org/abs/1507.02159.

[44] Karen Simonyan and Andrew Zisserman. "Two-Stream Convolutional Networks for Action Recognition in Videos". In: *Advances in Neural Information Processing Systems*. 2014. URL: http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos.

[45] Heng Wang et al. "Action Recognition by Dense Trajectories". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5995407.

[46] Zhengzhong Lan et al. "Beyond Gaussian Pyramid: Multi-Skip Feature Stacking for Action Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Lan_Beyond_Gaussian_Pyramid_2015_CVPR_paper.html.

[47] Zhuowei Cai et al. "Multi-View Super Vector for Action Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Cai_Multi-View_Super_Vector_2014_CVPR_paper.html.

[48] Gunnar Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Scandinavian Conference on Image Analysis*. Springer, 2003. URL: http://link.springer.com/chapter/10.1007/3-540-45103-X_50.

[49] Navneet Dalal and Bill Triggs. "Histograms of Oriented Gradients for Human Detection". In: *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*. IEEE, 2005. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1467360.

[50] Ivan Laptev et al. "Learning Realistic Human Actions from Movies". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4587756.

[51] Navneet Dalal, Bill Triggs, and Cordelia Schmid. "Human Detection Using Oriented Histograms of Flow and Appearance". In: *European Conference on Computer Vision*. Springer, 2006. URL: http://link.springer.com/chapter/10.1007/11744047_33.

[52] Bruce D. Lucas, Takeo Kanade, et al. "An Iterative Image Registration Technique with an Application to Stereo Vision." In: *IJCAI*. 1981. URL: https://ri.cmu.edu/pub_files/pub3/lucas_bruce_d_1981_2/lucas_bruce_d_1981_2.pdf.

[53] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International journal of computer vision* (2004). URL: http://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94.

[54] Quoc V. Le et al. "Learning Hierarchical Invariant Spatio-Temporal Features for Action Recognition with Independent Subspace Analysis". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5995496.

[55] Limin Wang, Yu Qiao, and Xiaoou Tang. "Action Recognition with Trajectory-Pooled Deep-Convolutional Descriptors". In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7299059.

[56] Shuiwang Ji et al. "3D Convolutional Neural Networks for Human Action Recognition". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* (2013). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6165309.

[57] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. "Deep Learning". In: *2015* (2016). URL: http://www.deeplearningbook.org/front_matter.pdf.

[58] Yann LeCun et al. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* (1998). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=726791.

[59] Travis Rose et al. "The Trecvid 2008 Event Detection Evaluation". In: *Applications of Computer Vision (WACV), 2009 Workshop on*. IEEE, 2009. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5403089.

[60] Ming Yang et al. "Human Action Detection by Boosting Efficient Motion Features". In: *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. IEEE, 2009. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5457656.

[61] Konrad Schindler and Luc Van Gool. "Action Snippets: How Many Frames Does Human Action Recognition Require?" In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4587730.

[62] Hueihan Jhuang et al. "A Biologically Inspired System for Action Recognition". In: *2007 IEEE 11th International Conference on Computer Vision*. Ieee, 2007. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4408988.

[63] Moez Baccouche et al. "Sequential Deep Learning for Human Action Recognition". In: *International Workshop on Human Behavior Understanding*. Springer, 2011. URL: http://link.springer.com/chapter/10.1007/978-3-642-25446-8_4.

[64] Ho-Joon Kim, Joseph S. Lee, and Hyun-Seung Yang. "Human Action Recognition Using a Modified Convolutional Neural Network". In: *International Symposium on Neural Networks*. Springer, 2007. URL: http://link.springer.com/chapter/10.1007/978-3-540-72393-6_85.

[65] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural computation* (1997). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6795963.

[66] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. "Learning Precise Timing with LSTM Recurrent Networks". In: *Journal of machine learning research* (2002). URL: http://www.jmlr.org/papers/v3/gers02a.html.

[67] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. 2012. URL: http://papers.nips.cc/paper/4824-imagenet-classification-w.

[68] Jeffrey Dean et al. "Large Scale Distributed Deep Networks". In: *Advances in Neural Information Processing Systems*. 2012. URL: http://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks.

[69] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild". In: (2012). URL: http://crcv-web.eecs.ucf.edu/papers/UCF101_CRCV-TR-12-01.pdf.

[70] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014. URL: http://dl.acm.org/citation.cfm?id=2654889.

[71] Gül Varol, Ivan Laptev, and Cordelia Schmid. "Long-Term Temporal Convolutions for Action Recognition". In: *arXiv preprint arXiv:1604.04494* (2016). URL: https://hal.inria.fr/hal-01241518/.

[72] Nitish Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." In: *Journal of Machine Learning Research* (2014). URL: http://www.jmlr.org/papers/volume15/srivastava14a.old/source/srivastava14a.pdf.

[73] Vadim Kantorov and Ivan Laptev. "Efficient Feature Extraction, Encoding and Classification for Action Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2014. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Kantorov_Efficient_Feature_Extraction_2014_CVPR_paper.html.

[74] Thomas Brox et al. "High Accuracy Optical Flow Estimation Based on a Theory for Warping". In: *European Conference on Computer Vision*. Springer, 2004. URL: http://link.springer.com/chapter/10.1007/978-3-540-24673-2_3.

[75] Melvyn A. Goodale and A. David Milner. "Separate Visual Pathways for Perception and Action". In: *Trends in neurosciences* (1992). URL: http://www.sciencedirect.com/science/article/pii/0166223692903448.

[76] Olga Russakovsky et al. "Imagenet Large Scale Visual Recognition Challenge". In: *International Journal of Computer Vision* (2015). URL: http://link.springer.com/article/10.1007/s11263-015-0816-y.

[77] Ronan Collobert and Jason Weston. "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning". In: *Proceedings of the 25th International Conference on Machine Learning*. ACM, 2008. URL: http://dl.acm.org/citation.cfm?id=1390177.

[78] Joe Yue-Hei Ng et al. "Beyond Short Snippets: Deep Networks for Video Classification". In: *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. IEEE, 2015. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7299101.

[79] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. 2012. URL: http://papers.nips.cc/paper/4824-imagenet-classification-w.

[80] Christopher Zach, Thomas Pock, and Horst Bischof. "A Duality Based Approach for Realtime TV-L 1 Optical Flow". In: *Joint Pattern Recognition Symposium*. Springer, 2007. URL: http://link.springer.com/chapter/10.1007/978-3-540-74936-3_22.

[81] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. "Convolutional Two-Stream Network Fusion for Video Action Recognition". In: *arXiv preprint arXiv:1604.06573* (2016). URL: http://arxiv.org/abs/1604.06573.

[82] Tsung-Yu Lin, Aruni RoyChowdhury, and Subhransu Maji. "Bilinear CNN Models for Fine-Grained Visual Recognition". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015. URL: http://www.cv-foundation.org/openaccess/content_iccv_2015/html/Lin_Bilinear_CNN_Models_ICCV_2015_paper.html.

[83] Jorge Sánchez et al. "Image Classification with the Fisher Vector: Theory and Practice". In: *International journal of computer vision* (2013). URL: http://link.springer.com/article/10.1007/s11263-013-0636-x.

[84] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *European Conference on Computer Vision*. Springer, 2014. URL: http://link.springer.com/chapter/10.1007/978-3-319-10590-1_53.

[85] Ken Chatfield et al. "Return of the Devil in the Details: Delving Deep into Convolutional Nets". In: *arXiv preprint arXiv:1405.3531* (2014). URL: http://arxiv.org/abs/1405.3531.

[86] Petar Palasek and Ioannis Patras. "Action Recognition Using Convolutional Restricted Boltzmann Machines". In: *Proceedings of the 1st International Workshop on Multimedia Analysis and Retrieval for Multimodal Interaction*. New York, NY, USA: ACM, 2016. URL: http://doi.acm.org/10.1145/2927006.2927012.

[87] David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. "A Learning Algorithm for Boltzmann Machines". In: *Cognitive science* (1985). URL: http://www.sciencedirect.com/science/article/pii/S0364021385800124.

[88] Geoffrey E. Hinton, Simon Osindero, and Yee-Whye Teh. "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural computation* (2006). URL: http://www.mitpressjournals.org/doi/abs/10.1162/neco.2006.18.7.1527.

[89] Asja Fischer and Christian Igel. "An Introduction to Restricted Boltzmann Machines". In: *Iberoamerican Congress on Pattern Recognition*. Springer, 2012. URL: http://link.springer.com/chapter/10.1007/978-3-642-33275-3_2.

[90] Honglak Lee et al. "Unsupervised Learning of Hierarchical Representations with Convolutional Deep Belief Networks". In: *Communications of the ACM* (2011). URL: http://dl.acm.org/citation.cfm?id=2001295.

[91] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. "Unsupervised Learning of Video Representations Using LSTMs". In: *Proceedings of The 32nd International Conference on Machine Learning*. 2015. URL: http://jmlr.org/proceedings/papers/v37/srivastava15.html.

[92] Honglak Lee et al. "Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations". In: *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009. URL: http://dl.acm.org/citation.cfm?id=1553453.

[93] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. "Shuffle and Learn: Unsupervised Learning Using Temporal Order Verification". In: (Mar. 28, 2016). arXiv: 1603.08561 [cs]. URL: http://arxiv.org/abs/1603.08561.

[94] Zan Gao et al. "Comparing Evaluation Protocols on the KTH Dataset". In: *International Workshop on Human Behavior Understanding*. Springer, 2010. URL: http://link.springer.com/chapter/10.1007/978-3-642-14715-9_10.

[95] M. A. R. Ahad et al. "Action Dataset - A Survey". In: *SICE Annual Conference 2011*. SICE Annual Conference 2011. Sept. 2011.

[96] Haowei Liu, Rogerio Feris, and Ming-Ting Sun. "Benchmarking Datasets for Human Activity Recognition". In: *Visual Analysis of Humans*. Ed. by Thomas B. Moeslund et al. Springer London, 2011. URL: http://link.springer.com/chapter/10.1007/978-0-85729-997-0_20.

[97] Tal Hassner. "A Critical Review of Action Recognition Benchmarks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, June 2013. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6595882.

[98] Moshe Blank et al. "Actions as Space-Time Shapes". In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. IEEE, 2005. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1544882.

[99] Daniel Weinland, Remi Ronfard, and Edmond Boyer. "Free Viewpoint Action Recognition Using Motion History Volumes". In: *Computer vision and image understanding* (2006). URL: http://www.sciencedirect.com/science/article/pii/S1077314206001081.

[100] *INRIA 4D Repository - IXMAS Action Dataset*. URL: http://4drepository.inrialpes.fr/public/viewgroup/6.

[101] M. D. Rodriguez, J. Ahmed, and M. Shah. "Action MACH a Spatio-Temporal Maximum Average Correlation Height Filter for Action Recognition". In: *2008 IEEE Conference on Computer Vision and Pattern Recognition*. 2008 IEEE Conference on Computer Vision and Pattern Recognition. June 2008.

[102] Khurram Soomro and Amir R. Zamir. "Action Recognition in Realistic Sports Videos". In: *Computer Vision in Sports*. Springer, 2014. URL: http://link.springer.com/chapter/10.1007/978-3-319-09396-3_9.

[103] *Center for Research in Computer Vision at the University of Central Florida - UCF Sports Action Data Set.* URL: http://crcv.ucf.edu/data/UCF_Sports_Action.php.

[104] *Ivan Laptev | Inria Paris - Datasets Download.* URL: http://www.di.ens.fr/~laptev/download.html.

[105] Michael Marszalek, Ivan Laptev, and Cordelia Schmid. "Actions in Context". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5206557.

[106] *Ivan Laptev | Inria Paris - Hollywood 2 Human Action and Scenes Dataset.* URL: http://www.di.ens.fr/~laptev/actions/hollywood2/.

[107] Jingen Liu, Jiebo Luo, and Mubarak Shah. "Recognizing Realistic Actions from Videos "in the Wild"". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on.* IEEE, 2009. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5206744.

[108] *Center for Research in Computer Vision at the University of Central Florida - UCF YouTube Action Datset.* URL: http://crcv.ucf.edu/data/UCF_YouTube_Action.php.

[109] Kishore K. Reddy and Mubarak Shah. "Recognizing 50 Human Action Categories of Web Videos". In: *Machine Vision and Applications* (2013). URL: http://link.springer.com/article/10.1007/s00138-012-0450-4.

[110] Hildegard Kuehne et al. "HMDB: A Large Video Database for Human Motion Recognition". In: *Computer Vision (ICCV), 2011 IEEE International Conference on.* IEEE, 2011. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6126543.

[111] *Serre Lab - HMDB: A Large Human Motion Database.* URL: http://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/.

[112] Orit Kliper-Gross, Tal Hassner, and Lior Wolf. "The Action Similarity Labeling Challenge". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2012). URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6042884.

[113] Sami Abu-El-Haija et al. "Youtube-8m: A Large-Scale Video Classification Benchmark". In: *arXiv preprint arXiv:1609.08675* (2016). URL: https://arxiv.org/abs/1609.08675.

[114] Ross Messing, Chris Pal, and Henry Kautz. "Activity Recognition Using the Velocity Histories of Tracked Keypoints". In: *2009 IEEE 12th International Conference on Computer Vision.* IEEE, 2009. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5459154.

[115] *University of Rochester Activities of Daily Living Dataset.* URL: http://www.cs.rochester.edu/u/rmessing/uradl/?.

[116] Edouard Auvinet et al. "Multiple Cameras Fall Dataset". In: *DIRO-Université de Montréal, Tech. Rep* (2010). URL: http://www.iro.umontreal.ca/~labimage/Dataset/technicalReport.pdf.

[117] 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. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6247801.

[118] Fabian Caba Heilbron et al. "Activitynet: A Large-Scale Video Benchmark for Human Activity Understanding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2015. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Heilbron_ActivityNet_A_Large-Scale_2015_CVPR_paper.html.

[119] *Activity Net*. URL: http://activity-net.org/.

[120] Gunnar A. Sigurdsson et al. "Hollywood in Homes: Crowdsourcing Data Collection for Activity Understanding". In: *arXiv preprint arXiv:1604.01753* (2016). URL: http://arxiv.org/abs/1604.01753.

[121] *ImageNet*. URL: http://image-net.org/about-stats.