Research & Development
–DRAFT–

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

---

Author:
Maximilian Schöbel

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

January 12th, 2017

Contents

Abstract: We investigate human action recognition from video data as a key step towards the autonomous understanding of scenes in the environment of an autonomous (robotic) system. Describe current approaches. Biggest problems: Availability of data How to overcome: Unsupervised learning

1 Introduction

META: Brief but concise review of the first two "W"s.

The operation of autonomous mobile systems in public, uncontrolled environments is despite active research still a difficult task.

Humans are perfectly able to act and move in unknown, crowded environments and even react successfully to new situations because they are aware of their surroundings.

An important part of Situation Awareness is the knowledge of what actions are currently performed by persons in the vicinity of an agent. This knowledge enables the agent to derive a suitable policy for its own future actions.

Actions of interest are single-person actions, person-person interactions, person-object interactions and group activities.

Enabling situation-awareness in autonomous systems is an important goal, which has an impact on other problems in autonomous systems.

Possible applications:
Pedestrian movement predicition in robotics,
Risk and danger evaluation through video surveillance in public environements,
surveillance of children or the elderly in assisted living environments,
patient monitoring in hospitals,
video retrieval (content-based video indexing),
human-computer interaction.

Requirement: Automated recognition of high-level actions.

Situation awareness is an abstract concept, which includes lots of independent manifestations and involves multiple sensory inputs.

This work focuses on approaches that process time-sequential video data, because video-cameras represent a cost-effective and widely used technology in many existing systems.

Motivation for using videos: Promising results in classfication tasks from images. Video adds another (temporal) dimension, which conveys a lot of information that can be accessed for classification as well. Video provides natural data augmentation cite:simnoyan two-stream paper (??)

For example, YouTube reported that over 300 hours of video are uploaded every minute to their servers [43]. YouTube statistics. http://www.youtube.com/yt/press/statistics.html, 2015.

1.1 Situation Awareness from Video Data

META: General Definition of Situation Awareness in the context of autonomous systems.

Placement of Action Recognition among other vision-based methods, i.e.

Categories: Segmentation, Detection, Tracking, Recognition, Prediction

Segmentation:
Scene Segmentation

Detection:
Person/Pedestrian Detection (Abandoned) Object Detection Fall Detection Action Detection Event Detection Abnormal Event (Anomaly) Detection: O. Boiman and M. Irani. Detecting irregularities in images and in video. 2007 Saliency Detection

Tracking:
Person/Pedestrian Tracking Object Tracking

Recognition:
Human Action Recognition Human Interaction Recognition Crowd Behaviour Recognition Pose/Gesture Recognition Gait Recognition Scene Recognition (YUPENN, UMD)

Prediction:
Trajectory Prediction Action Prediction

Abnormal event detection O. Boiman and M. Irani. Detecting irregularities in images and in video. IJCV, 2007

Activity understanding "activity forecasting"

Definitions of the above methods.

Simple case: Video contains the performance of a single human action which needs to be classified into one of several preknown classses.

General real-world case: System operates on a video stream and needs to perform continuous recognition of human actions, including detection of beginning and endings times of containing acions.

General Processing Pipeline for Action Recognition: Person Detection -> Tracking -> Action Detection -> Segmentation -> Action recognition.

Action Recognition: A part of Computer Vision research, it's goal is to automatically analyze human actions/actitvities from video-data.

Other sensory input than video possible

1.2 The Action Recognition Problem

Action Recognition is a classification-task.

Difference to face/gate recognition: Generalize over person characteristics.

Intra- and inter-class variances.

Background and recording settings.

Temporal variations.

Obtaining and labeling training data.

Main task of action recognition research: Overcome these challenges and built systems, that recognize actions robustly, even when performed by different persons in differently lighted environments at different speeds.

Main components: (i) A discriminative architecture that is able to recognise the general characteristics of different action classes while ignoring personal characteristics of different performers. (ii) Large datasets that provide this information by containing many different examples for each action class.

1.3 Survey Papers in Action Recognition (Related work)

Review of most important/recent review papers in Action Recognition with traditional and Deep Learning approaches.

1.3.1 A survey on vision-based human action recognition, Ronald Poppe (2010)

Definition of action: Uses the hierarchical classification of human motion in action primitives, actions and activities as given in Moeslund et al. (cite ??)

Action primitives are atomic movements at the limb-level.

Actions are possibly cyclic whole body movements and consist of multiple action primitives.

Activities consist of multiple actions whose subsequent execution make the movement interpretable.

Example: Action primitives: Left/right leg forward -> Action: Starting, Running, Jumping -> Activity: Jumping hurdles.

Scope: Gives a very good classification of conventional methods in human action recognition.

The discussion is split according to video representations and classification methods.

Challenges of the domain are described very well.

Deficits: No Deep Learning methods are discussed.

Datasets and benchmarks are only discussed briefly.

1.3.2 Human Activity Analysis: A Review – Aggarwal and Ryoo (2011)

Gives an approach-based taxonomy.

1.3.3 A survey on vision-based methods for action representation, segmentation and recognition – Weinland et al. (2011)

1.3.4 A survey of video datasets for human action and activity recognition – Chaquet et al. (2013)

1.3.5 A review of unsupervised feature learning and deep learning for time-series modeling – Längkvist et al. (2014)

1.3.6 Going Deeper into Action Recognition: A survey – Herath et al. (2016)

Definition of action:

1.4 Important aspects of Action Recognition Approaches

video representations, i.e. how to apply fixed length model to variable length videos.

2 Conventional Methods in Action Recognition

This section provides a description of conventional methods in action recognition, which do not leverage 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 1.3, we provide a condensed overview of these methods by describing the main research directions using the taxonomy of Aggarwal and Ryoo [1] released in 2011. For a more detailed description of specific approaches in this area we also refer to [1]. Additionally we focus on the Bag of Visual Words (BoVW) paradigm, which has become the standard approach in action recognition using hand-crafted features since. We furthermore identify state-of-the art approaches, that employ this standard approach and describe them in more detail.

3 Main components in action recognition using local features: Feature Extraction, Representation Building, Classification.

Methods for feature extraction: Interest point detectors or dense sampling.

Space-time interest point detectors: Harris3D[2], Cuboids[3], Hessian Detector[**willems_efficient_2008**]

Descriptors for 3D volumes around previously detected space-time interest points: Histogram of Gradient HOG[4], Histogram of Optical Flow (HOF)[5], 3D Histogram of Gradient (HOG3D)[6], Extended SURF (ESURF)[**willems_efficient_2008**]

"standard approach to video classification" described in [7]

Laptev and Lindeberg [26] proposed spatio-temporal interest points (STIPs) by extending Harris corner detectors to 3D. SIFT and HOG are also extended into SIFT-3D [34] and HOG3D [19] for action recognition. Dollar et al. proposed Cuboids features for behavior recognition [5]. Sadanand and Corso built Ac- tionBank for action recognition [33]. Recently, Wang et al. proposed improved Dense Trajectories (iDT) [44] which is currently the state-of-the-art hand-crafted feature. Tran 2015

Recently, interest point detectors and local descriptors have been extended from images to videos. Laptev and Linde- berg [13] introduced space-time interest points by extending the Harris detector. Other interest point detectors in- clude detectors based on Gabor filters [1, 5] or on the de- terminant of the spatio-temporal Hessian matrix [33]. Fea- ture descriptors range from higher order derivatives (local jets), gradient information, optical flow, and brightness in- formation [5, 14, 24] to spatio-temporal extensions of image descriptors, such as 3D-SIFT [25], HOG3D [11], extended SURF [33], or Local Trinary Patterns [34]. Action Recognition by dense trajectories – Wang 2011

2.1 Taxonomy-based Overview

Approach-based taxonomy

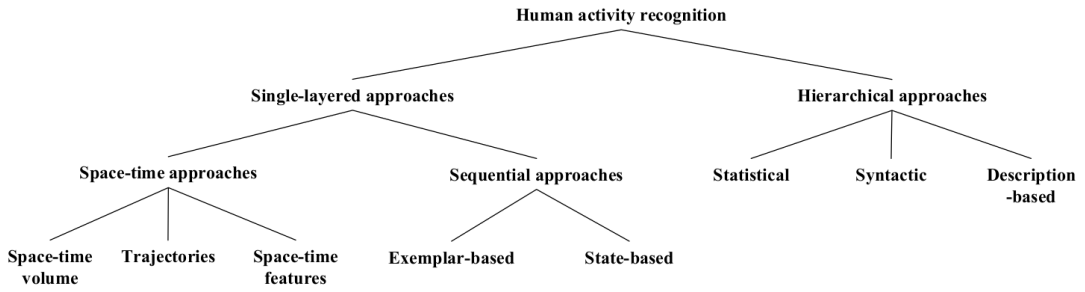


Figure 1: Approach-based taxonomy for conventional methods in human action recognition as given by Aggarwal and Ryoo [1]

Divide the field of activity/action recognition into:

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

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

2nd level of distinction (single layer):

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.

2nd Level distinction hierarchical approaches:

Statistical approaches construct hierarchically stacked statistical state-based models, such as layerd Hidden Markov Models.

Syntactic approaches use a grammar syntax such as stochastic context-free grammar.

Description-based approaches describe atomic sub-actions and their temporal, spatial and logical structure.

2.1.1 Space-time Approaches

Space-time approaches can be further distinguished by what kind of features from the video volume they use. Space-time volume approaches employ the video volume or parts of it directly for creating a representation of the video, which is then compared with other video volume representations. Trajectory approaches use motion trajectories of tracked points inside the volume for the recognition of ongoing actions. Space-time

feature approaches extract features around interest-points locally and aggregate them into a representation of the video volume.

Action recognition using space-time volumes
A typical approach for action recognition with space-time volumes uses template matching: Given a similarity measure for video volumes, the algorithm constructs or selects template video volumes 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 training templates by using the similarity measure. The action class, that corresponds to the most similiar training template is selected as output class.

Approaches in this category mainly differ by how a representation is built and how they are compared. Bobick and Davis [8] 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.

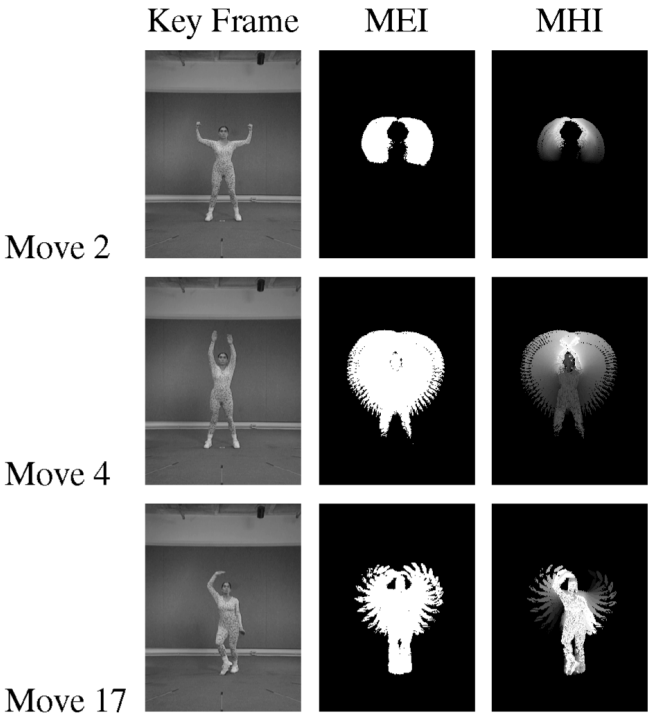


Figure 2: Motion-energy image (MEI) and motion-history image (MHI) and example frame of three ballet actions [8]

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 [9]. 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 while performing an action. 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 [10] classified actions by using the trajectories of 13 tracked points directly as displayed in figure 3.

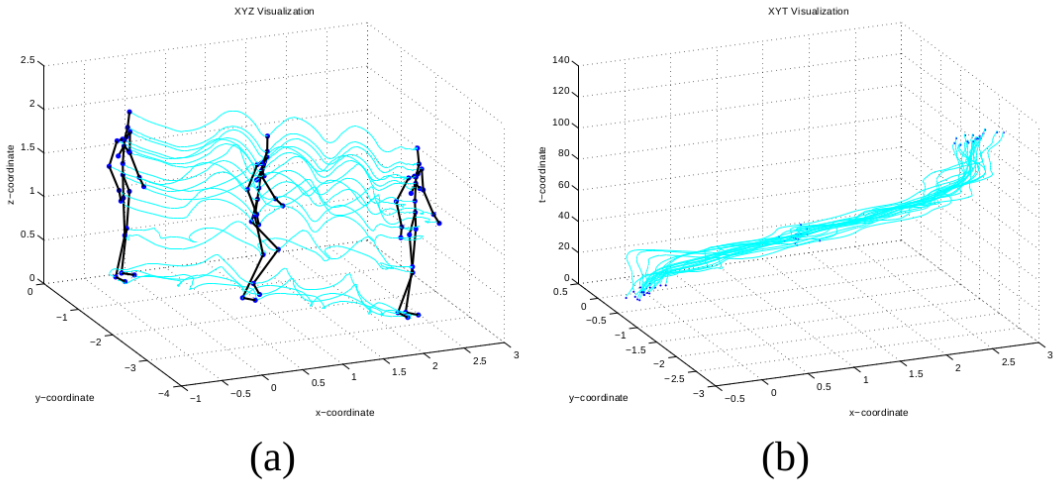


Figure 3: Trajectories of a person's tracked joint positions while performing an action. Trajectories shown in XYZ space (a) and XYT space (b) [10]

Action recognition using space-time features Ohhhh yeeeeeeeeeeeahhhH!!!!

2.1.2 Sequential Approaches

In single-layered sequential approaches an action is processed as a sequence of observations, specifically as a sequence of feature vectors. Therefore, 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 [1] describe using the 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, sequential approaches usually calculate the likelihood of action classes producing the observed sequence of feature vectors. The action class with the highest likelihood is assigned to input video, i.e. to the sequence of feature vector observations.

Aggarwal and Ryoo [1] further differentiate sequential approaches into exemplar-based

and state model-based approaches.

Exemplar-based approaches Exemplar-based approaches store template sequences of feature vectors for each action class (exemplars). A presented unkown action in an input video is then 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[11][12][13].

State model-based approaches Sequential state model-based approaches construct models, which are trained to generate sequences of feature vectors.

2.1.3 Hierarchical Approaches

The main idea of hierarchical approaches is to model complex actions as a hierarchy of simpler sub-actions [1]. 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 the observable low-level sub-actions.

Aggarwal and Ryoo [1] further differentiate hierarchical approaches into statistical approaches, syntactic approaches and desciption-based approaches.

Statistical approaches
Hierarchical statistical approaches use stacked state-based models such as Hidden Markov Models (HMMs) [14][15] or Dynamic Bayesian Networks (DBNs) [16][17] 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 [14] is said to be one of the most fundamental forms of hierarchical statistical approaches [1].

Syntactic approaches
In hierarchical syntactic approaches, a high-level action is represented as a string of symbols. Each symbol therein corresponds to a simpler, possibly atomic, sub-action as described previously. Equivalently to hierarchical statistical approaches, syntactic approaches require the recongition 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 [18][19][20][21], that generate sequences of symbols corresponding to the action class. Syntactic approaches then use parsing techniques from the field of programming

languages [22] to recognize high-level actions.

Description-based approaches
The definition of description-based approaches by Aggarwal and Ryoo [1] states: "A description-based approach is a recognition approach, that explicitly maintains human activities' spatio-temporal structure." High-level actions are represented by occurences 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 occuring action. Allen [23][24] 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 [25][26][27][28].

2.2 Bag of Visual Words

Defintion of local features: "primitive events corresponding to moving two-dimensional image structures at moments of non-constant motion" Schuldt 2004 KTH paper

Interest point detection: Space-time interest points "I. Laptev. On space-time interest points. IJCV, 64(2-3), 2005."

Local feature approaches extract features, i.e different characteristics of pixel values of a video, in locally limited neigbourhoods. The algorithm to extract these features is called a feature extractor.

Three main questions:

1. Where to the features (around what points?)
2. What features to extract?
3. How to aggregate the extracted feature (vectors) into a global fixed-size representation of the video.

"standard approach" karpathy large-scale

2.2.1 Feature Extraction

Cuboids "P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In VS-PETS, 2005."

2.2.2 Feature Encoding

Bag of visual Words paradigm

Can be improved with a multi-channel approach as in "M. M. Ullah, S. N. Parizi, and I. Laptev. Improving bag-of-features action recognition with non-local cues. In BMVC,2010."

Fisher Vector

One limitation of these local features is that they lack semantics and discriminative capacity. Therefore mid-level and high-level features were proposed (cite TDD).

VLAD

2.2.3 Classification

2.3 State of the Art Approaches using local features

2.3.1 Action Recognition by Dense Trajectories – Wang et al. (2011)

Wang et al. [29] 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 features are extracted along the resulting point trajectories to form trajectory descriptors, which are then aggregated into a global video descriptor using the bag-of-words paradigm. cite ??

Before, also other approaches used feature trajectories for action recognition by either tracking sparse spatio-temporal interest points using a standard KLT tracker or by matching SIFT features between consecutive frames.

Dense sampling of interest points has shown to yield improved performance in action recognition over sparse spatio-temporal interest points [30]. However using the KLT tracker to obtain dense trajectories or matching SIFT features on densely sampled points would be computationally too expensive to handle large datasets. The authors approach therefore represents an efficient way to extract dense trajectories.

Since motion in a video, and therefore 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 motion boundary histogram 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 4

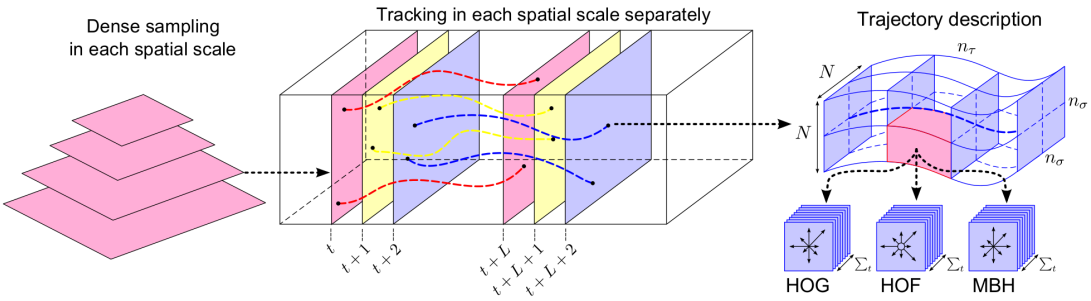


Figure 4: Description of densely extracted trajectories [29]

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 mean-filtering a dense optical flow field, which was extracted by the Farnebäck algorithm [31] as implemented in OpenCV. The tracked points in subsequent frames then form the trajectory $(P_t, P_{t+1}, P_{t+2}, \cdots)$.

The maximum length of a trajectory is limited to $L = 15$ frames to avoid the problem of drifting. Trajectories that exceed this limit are removed from the tracking process. The presence of a trackectory in each $W \times W$ unit of each frame is verified. If no tajectory is present, a new point is sampled and added to the tracking process.

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 information given by the trajectory. 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 4, where $N = 32$ has shown to yield good results.

Feature descriptors evaluated in the context of dense trajectories are (see also ??):

- **HOG** (Histogram of Oriented Gradients) [4]
- **HOF** (Histogram of Optical Flow) [5]
- **MBH** (Motion Boundary Histogram) [32]

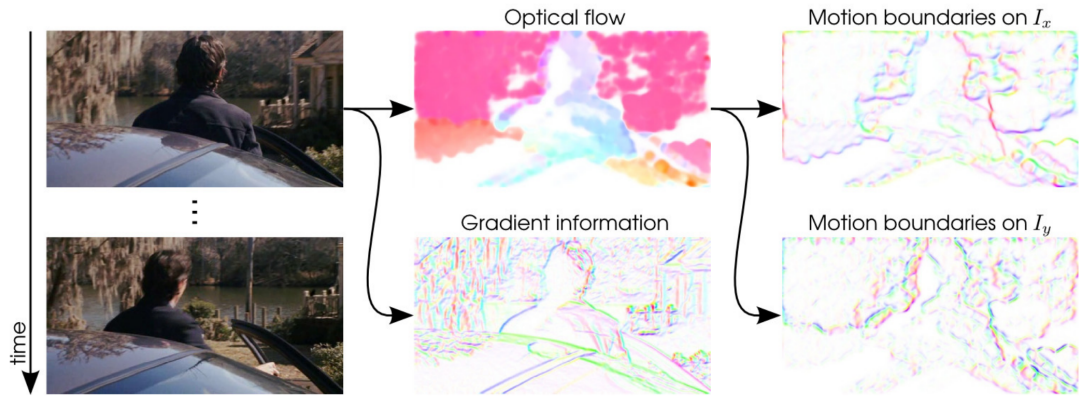


Figure 5: Visualization of the information captured by HOG, HOF and MBH on complete video frames. In each image, the orientation is given by color, the magnitude is given by saturation. [29]

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, i.e. 9 bins.

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 (compare the optical flow-image and motion boundaries in figure 5).

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 dataset's 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 occurences 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.

Different descriptor can be combined ??.

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 aorund 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. [29]

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 confirms the importance of motion information encoded in the trajectory shapes according to the authors.

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).

The dense trajectory approach significantly outperformed the state-of-the art on the YouTube, Hollywood2 and UCF Sports datasets when all descriptors are combined. On the KTH dataset, the approach yields competitive results.

2.3.2 Action recognition with improved trajectories – Wang et al. (2013)

[33] "Recently, Wang et al. proposed improved Dense Trajectories (iDT) [44] which is currently the state-of-the-art hand-crafted feature" tran learning spatio-temporal features with 3D convolutional networks.

2.3.3 Multi-view super vector for action recognition – Cai et al. (2014)

[34]

2.3.4 Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice – Peng et al. (2014)

[35]

2.4 Beyond gaussian pyramid: Multi-skip feature stacking for action recognition

UCF 101 - 89.1%

other iDT-based methods: Beyond gaussian pyramid: Multi-skip feature stacking for action recognition. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice.

3 Deep Learning Methods in Action Recognition

META: Review of approaches that use Deep Learning methods.

Taxonomy of approaches.

3.1 3D-Convolutional Networks

I.e. convolutional methods.

The naive approach: using framewise information, classifying it with a CNN and average the results. Elaborate further. . . ??

"Video analysis provides more information to the recognition task by adding a temporal component through which motion and other information can be additionally used. At the same time, the task is much more computationally de- manding even for processing short video clips since each video might contain hundreds to thousands of frames, not all of which are useful. A naive approach would be to treat video frames as still images and apply CNNs to recognize each frame and average the predictions at the video level. However, since each individual video frame forms only a small part of the video's story, such an approach would be using incomplete information and could therefore eas- ily confuse classes especially if there are fine-grained dis- tinctions or portions of the video irrelevant to the action of interest." Beyond Short Snippets: Deep Networks for Video Classification

explain: feature maps

Treated as an additional spatial dimension.

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

Ji et al. [36] 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 [37].

Ji et al. [36] 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 6 are described below:

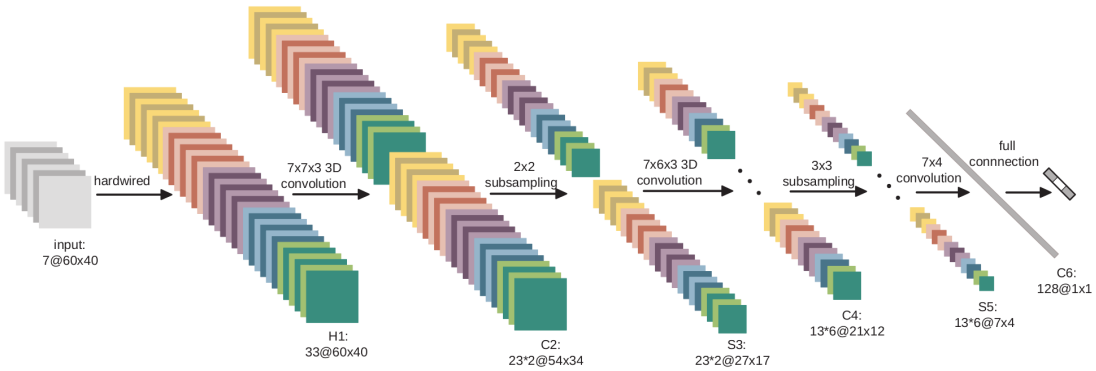


Figure 6: 3D CNN architecture developed for human action recognition [36]

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 dimesions $7 \times 7 \times 3$ (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. [38]. 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 [39]. Additionally, it's performance is measured as stand-alone approach on the KTH benchmarking datset [40].

Evaluation on TRECVID:
The TRECVID 2008 development dataset [39] contains 49 hours of surveillance videos continuously recorded by five different cameras on five days at London Gatwick Airport. The datset 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 7. Example frames of the datset are shown in figure 8.

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 7: Number of action samples per class from the TRECVID 2008 development dataset [36]

Since the dataset provides continuous videos with several persons in a real-world scence, Ji et al. [36] 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 8: Example scenes of the TRECVID 2008 development dataset with results of human detection and tracking [36]

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 [41] from the seven (grayscaled) input frames, which are then aggregated using the BoW-Paradigm and classified through a linear SVMs.
3. Extraction of dense SIFT features [41] from motion edge history images (MEHI)[42] of the input frames, which are then aggregated as above.

The 3D CNN model outperfoms the other approaches on the TRECVID 2008 develop-

ment 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:
Furhermore, the stand-alone 3D CNN architecture was evaluated on the KTH dataset [40]. It achieves an overall accuracy of 90.2% on that benchmark. In comparison: Schindler and Van Gool [43] achieved 92.7% and Jhuang et al. [44] achieved 91.7% several years earlier.

Ji et al. [36] showed, that 3D convolutions yield competitive performace compared to state-of-the-art approaches at that time as shown on the KTH benchmark [40]. Although the authors reference the work of Schindler and Van Gool [43] 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. [45].

3.1.2 Sequential Deep Learning for Human Action Recognition (2011)

Baccouche et al. [45] identify two deficits of previous approaches for extending CNNs to the video domain, specifically the approach of 3D convolutions such as [36] and [46]:

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 9 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 [47] to classify the entire sequence of previously extracted spatio-temporal features.

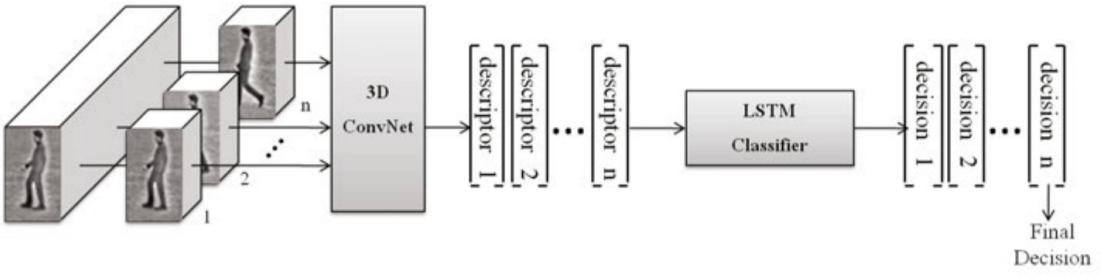


Figure 9: Overview of the two-step architecture consisting of a 3D ConvNet and a recurrent neural network classifier. [45]

Baccouche et al. [45] incorporate the work of Ji et al. [36] 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 [36], the overall approach is evaluated on the KTH dataset [40].

Details of the 3D ConvNet architecture are shown in figure 10.

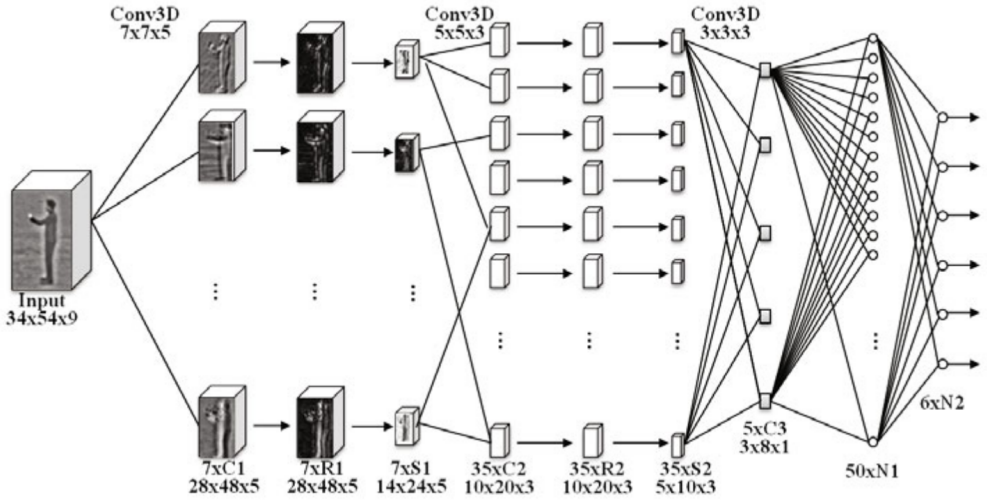


Figure 10: Detailed architecture of the 3D ConvNet for later use with an RNN classifier. [45]

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 [45]. The third convolutional layer is followed by two neuron layers (fully-connected layers) *N1* and *N2*.

The ConvNet model, as shown in figure 10 embeds $17,169$ trainable parameters in total, which is about 15 times less than the $295,458$ parameters used by Ji et al. [36].

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. [36]: online Backpropagation with momentum adapted to weight-sharing.

For training the RNN classifier with online backpropagation through time [48], 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. [36] (90.2%), although the model requires about 15 times less paramerters. 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. [7] 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 multiresolution 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 11.

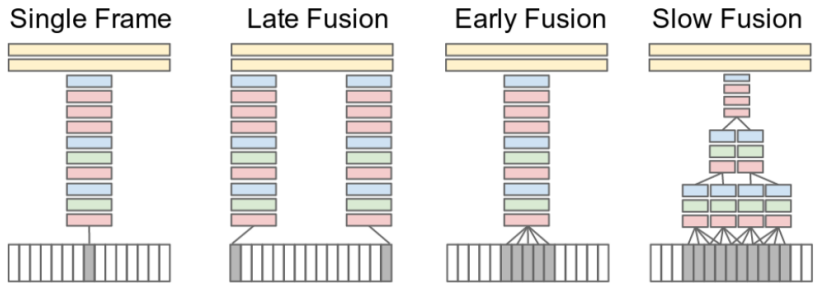


Figure 11: Different methods for fusing temporal with spatial information in convolutional neural network architectures. Red, green and blue denotes convolutional, normalization and pooling-layers. Grey colored inputs are single RGB video frames. Yellow layers are fully-connected for classification. [7]

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 [49], 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 corresprond 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 applys 3D convolutions as done by Ji et al. [36] and Baccouche et al. [45]. 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 [50] 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 12) are computed from the clip-level predictions simply by averaging.

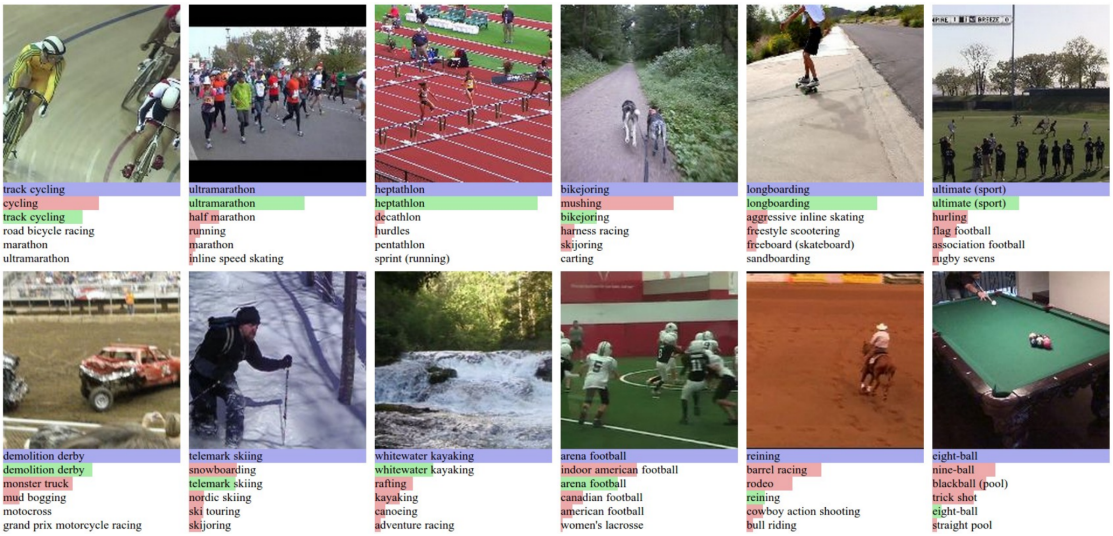


Figure 12: Classification of videos in the Sports-1M dataset. Blue label is the ground truth, below class predictions are shown with decreasing confidence. [7]

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. [7]

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. [7] describe the performance difference of between fusion models as "surprisingly insignificant"[7]. The single-frame model performs noticably 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[7], while maintaining accuracy.

The authors therefore propose a multiresolution architecture, that is shown in figure 13.

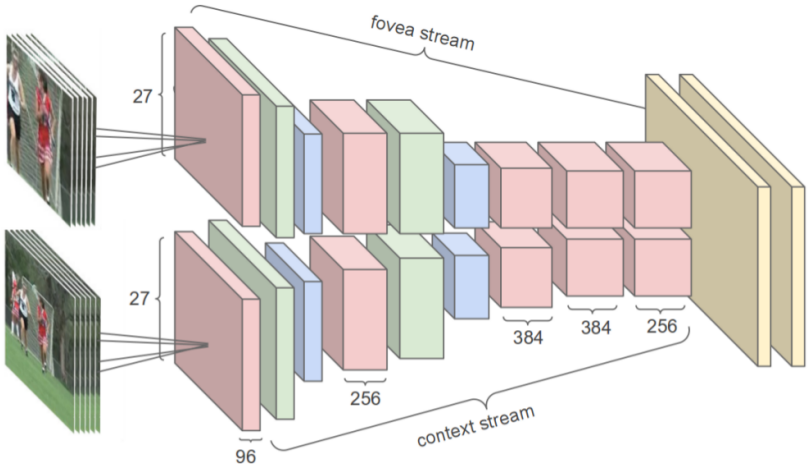


Figure 13: Multiresolution CNN architecture [7]

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 multiresolution 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. [51] create a generic spatio-temporal feature extractor, by training a very deep 3D ConvNet on the large-scale action recognition dataset Sports-1M [7]. 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. [7], which showed that using 3D convolutions in all convolutional layers of a CNN performs best for action recognition (*Slow Fustion*). 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 14.

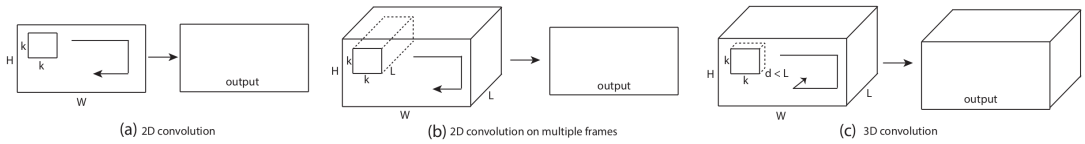


Figure 14: Dimensionality of convolutions: a) 2D convolution on a single image results in a 2D feature map. b) 2D convolution on multiple frames, interpreted as different channels of the input, results in a 2D feature map because results of individual frames are summed. c) 3D convolution on multiple frames results in a 3D feature map, therefore preserving the temporal dimension [7]

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[52], 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 neuraons each at the end and an additional softmax layer with one output per action class for training.

The work of Simonyan and Zisserman [53], 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 15:

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 temproal depth $3-3-5-5-7$ and decreasing temporal depth $7-5-5-3-3$.

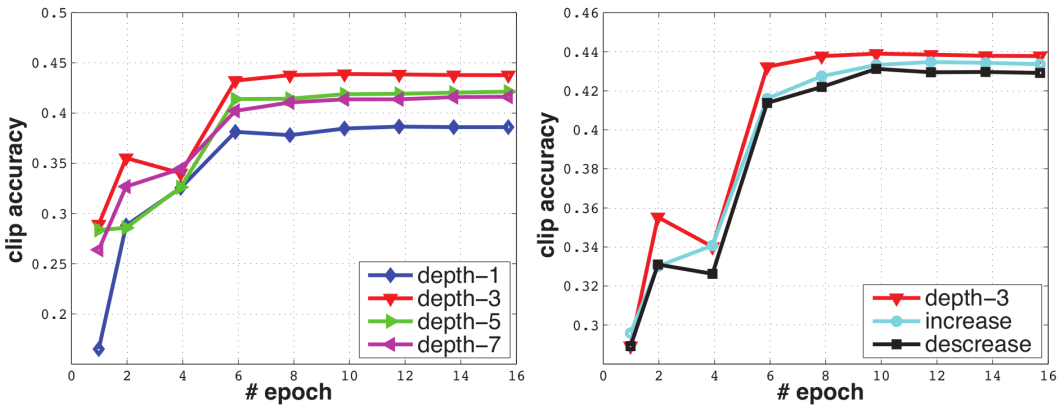


Figure 15: Clip accuracy for different temporal kernel depths over training epochs on UCF101. Left shows homogeneous temporal depth, right shows varying temporal depth. [51]

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 16), it is the deepest architecture reviewed for the action recognition task in this work.

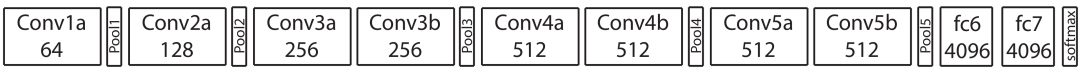


Figure 16: C3D architecture. Number of filters is given in each box. [51]

Training The network is trained on the Sports-1M [7] 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 [51].

To promote the usage of *C3D* features, the authors forked from the Caffe framework [54], extended it for handling three dimensional convolutional networks and threin provide a pretrained *C3D* network model for public use. The usage of *C3D* features is therfore 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 [55] address, similar to Baccouche et al. [45], 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. [36][7][51]

Instead of using a recurrent neural network for classifying convolutionally extracted spatio-temporal features, as done by Baccouche et al. [45], Varol, Laptev, and Schmid [55] 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 demonstratioin 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 17. T therein denotes the temporal extent, i.e. the number of input frames to the ConvNet, and architectural details are given below.

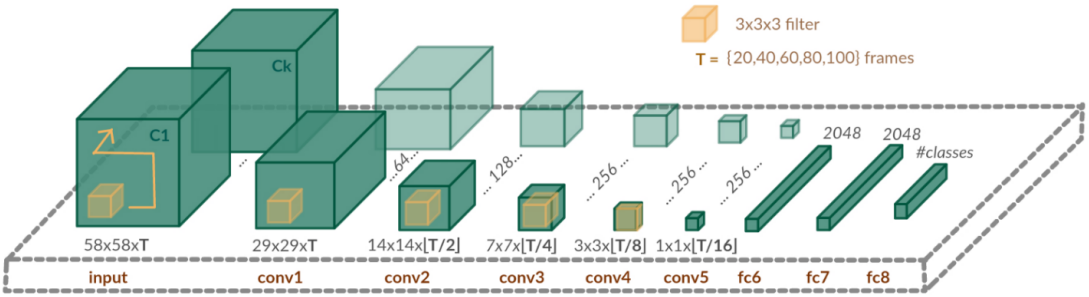


Figure 17: 3D CNN architecture for evaluating the influence of the number of input frames on action recognition accuracy. [55]

- 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.

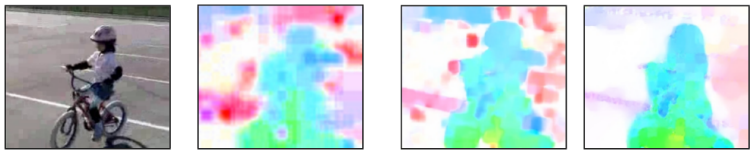
As a baseline approach the authors first evaluate their architecture with inputs of size $112 \times 112 \times 16$, that is a spatial resolution of 112×112 and $T = 16$ input frames, because it can be directly compared to the work of Tran et al. [51]. Additionally, the baseline is implemented with $T = 60$ 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*[56], *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 [57], 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 [31], which is fast, but calculates noisy flow fields.
3. Brox optical flow estimator [58], 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 18 (left). The table on the right of figure 18 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 18: 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. [55]

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% overlapse 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 [56] 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 [55]

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. [55]

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. [55]

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 19.

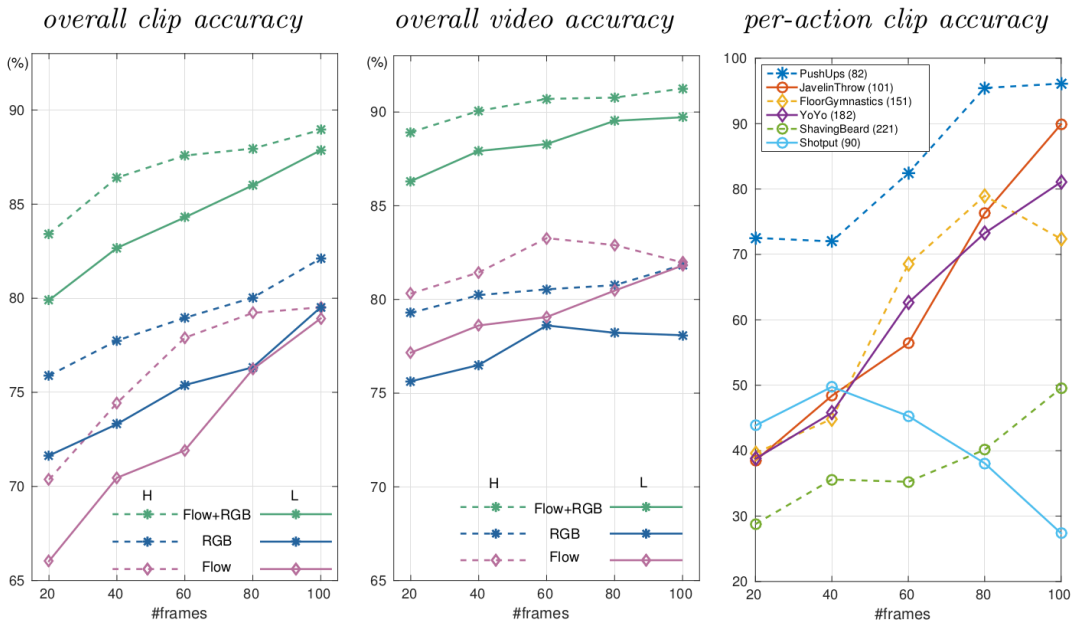


Figure 19: Action recognition performance of long-term convolutional networks for varying number of input frames, evaluated on UCF-101 (split1). H denotes high resolution (71×71 pixel), L denotes low resolution (58×58 pixel). [55]

To conclude, Varol, Laptev, and Schmid [55] show, that longer temporal extent increases the action recognition accuracy of 3D convolutional neural network for action recognition

from video.

3.2 Multiple Stream Networks

The most successfull architecture in action recognition. They are equally powerful as the improved dense trajectories approach. cite TDD ??

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.

Underlying principle: Temporal dimension has different characteristics than the spatial dimension and should therefore be handled separately, not in a joint 3D space.

The first architecture of this kind was proposed in 2014 by Simonyan and Zisserman.

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

Simonyan and Zisserman [59] propose a novel architecture for action recognition with separate spatial and temporal recognition streams, which are fused late (see figure **??**. This approach is motivated by the two-streams hypothesis [60], 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.

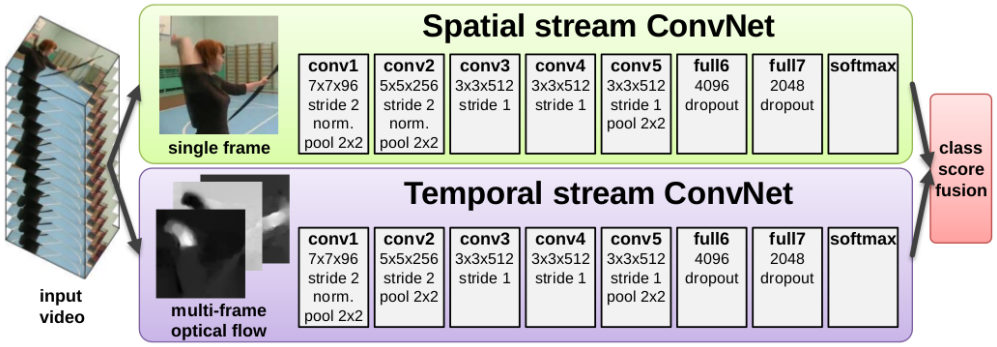


Figure 20: Two-stream architecture for video classification [59]

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 $1 \leq k \leq L$:

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

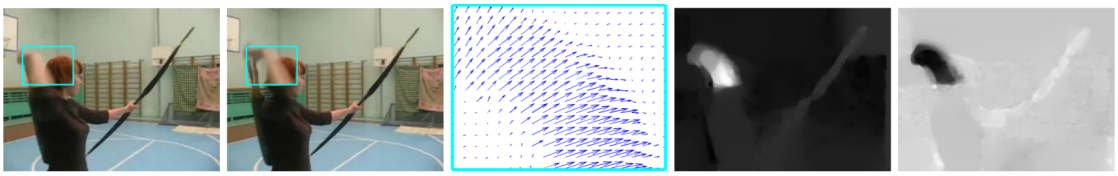


Figure 21: Optical flow between two frames (a) and (b). (c) shows the dense optical flow field, i.e. displacement vectors, of the turquoise region. (d) and (e) illustrate the x and y components of the flow field as images. [59]

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:

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

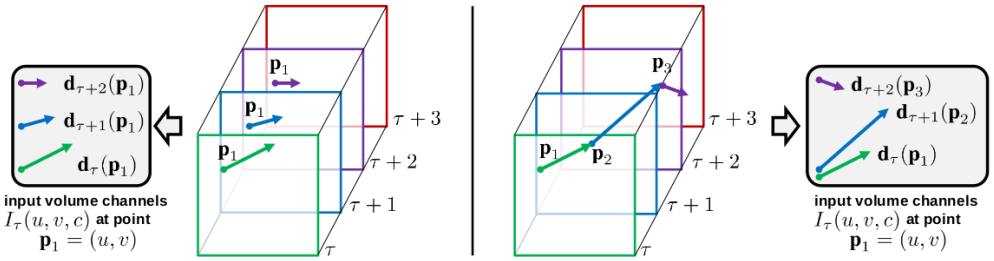


Figure 22: Construction of input volumes from multi-frame optical flow. Left: Optical Flow Stacking. Right: Trajectory Stacking. [59]

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 [61] for pre-training.

The temporal stream network is trained, according to the multi-task learning paradigm [62], 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 regulariser 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) [59]

Regarding these results, the authors decide on pre-trainining 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 having found 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 [59]

?? Conclusion duuuuuuude!

3.2.2 Beyond Short Snippets: Deep Networks for Video Classification – Ng et al. (2015)

Ng et al. [63] hypothesize that a global video description, i.e. a representation learned from the complete temporal evolution of a video, is important for accurate action recognition.

Similar to the approaches made in [45] and [55], the goal of this work is to design and evaluate video processing architectures, that are able to incorporate video information over long time periods (tens of seconds), preferably over the complete video.

The authors discuss the results of Karpathy et al. [7], 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 is therefore evaluated in this work (see also figure 23):

1. 2D image features are obtained from still input frames by feeding them into a convolutional neural network (which incorporates 2D convolutions for processing still images).

2. The resulting feature vectors for each frame, i.e. the activations of the last convolutional layers, are aggregated into a global video representation by using one of the following methods:
    - A feature pooling architecture aggregates the CNN activations through several pooling layers, which are added after the last convolutional layer.
    - A recurrent neural network uses layers of LSTM cells after the last convoultional layer to integrate frame information over time.

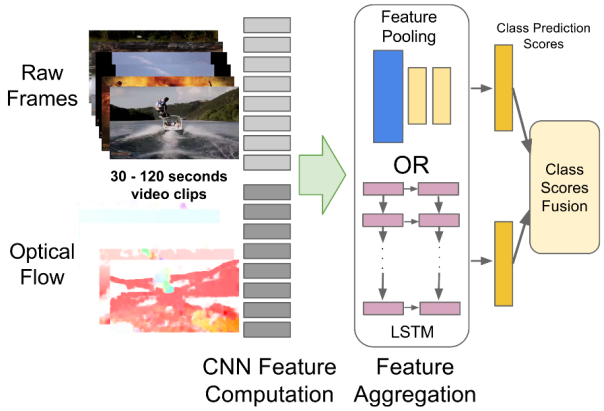


Figure 23: Overview of the appraoch taken by Ng et al. [63]

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

1. AlexNet [64]
2. GoogLeNet [65]

Both networks take 220x220 pixel sized input images/frames.

The temporal extend of this approach can be easily adjusted by changing the number of CNNs that process an input frame each.

The CNNs for processing single frames all share paramters, 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.

Implicit motion information from processing adjacent frames is lost at this frame-rate. This is compensated by using explicit motion information, i.e. optical flow inputs.

The Sports-1M and UCF-101 dataset are used as benchmarks in this work.

Feature Pooling
Pooling layers are used to combine feature vector representations of the input frames.

The authors consider, max-pooling, average pooling and a fully connected layer as pooling layers. Max-pooling results in faster learning architecture and was therefore chosen as the default pooling method in this approach.

Several pooling architectures are evaluated (see figure **??**).

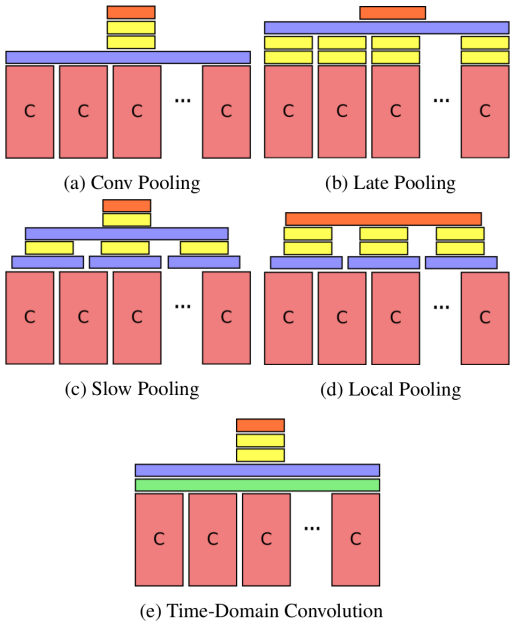


Figure 24: Feature pooling architectures with: Max-pooling layers (blue), time-domain-convolution layer (green), fully-connected layers (yellow), softmax layers (orange). [63]

a) **Conv Pooling:** Max-pooling is performed over the activations of the final convolutional layers across all input frames.

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 convolutional and 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 locally but only one stage of max-pooling is implemented. This prevents a potential loss of

temporal information.

e) **Time-Domain Convolution:** An additional time-domain convolutional layer is implemented before pooling features across frames.

LSTM architecture
To capture the dynamic content, i.e. the information that is encoded in the differences between frames, a deep LSTM [47] architecture with 5 layers containing 512 memory cells each is used to aggregate sequences of frame-level CNN activations.

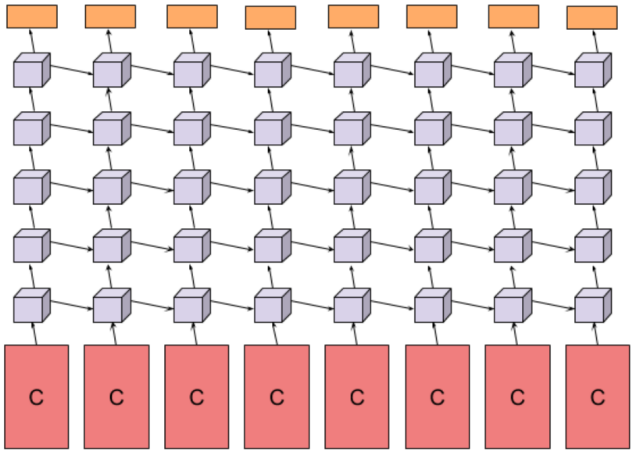


Figure 25: LSTM Architecture. Five layers of LSTM cells take the final CNN activations as inputs. The video class is predicted at every timestep by a softmax output layer. [63]

The max-pooling architectures were trained on a computing cluster using a distributed version of Gradiend Descent Training called Downpour Stochastic Gradient Descent.

The AlexNet and GoogLeNet networks were initialized on pre-trained ImageNet models and then fine-tuned on the Sports-1M dataset in order to safe training time.

Since the CNN models share weights across input frames, the authors note that training time can also be saved by expanding a trained single-frame model to 30-frames and then expanding the 30-frames model to 120-frames.

Analogically to the two-stream approach of [59] the authors additionally train both their models on optical flow and perform late fusion over the two resulting streams. The used optical flow is sampled at 15fps by using the method proposed in [**zach_duality_2007a**].

Evaluation on Sports-1M dataset:
For training and testing on Sports-1M, the first 5 minutes of each input-video are sampled at 1fps as input frames.

The frames are first resized to 256x256 pixels, a 220x220 pixel region of a randomly selected starting frame is selected and the following frames are cropped accordingly to

create a network input of the desired temporal length. Input frames are flipped horizontally with a probability of 50%.

Models capable of processing 120 frames in a single example (which corresponds to 2 minutes of video) are trained.

240 random input examples are generated from a video for testing as described above and the predictions are averaged.

The LSTM models are not limited to a fixed input size and could generally process a complete video in a single pass. The authors report a 3-5% increase in Hit@1 accuracy when using the above mentioned data augmentation methods however.

As first result (see table 8, the authors report the conv-pooling architecture to yield the best accuracy when evaluating different pooling methods.

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 pooling methods on Sports-1M dataset with a 120-frame AlexNet model. [63]

When evaluating the two different CNN architectures, GoogLeNet consistently outperforms AlexNet. Fine tuning the models is noted to be crucial for high performance.

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 accurate action recognition. The conv-pooling architecture with 120 input-frames yields a clip-hit of 70.8%, which is an improvement of 70% against the best clip-hit accuracy of Karpathy et al. [7]. Results are shown in table **??**.

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: Evaluation of the effect of the number of input frames using GoogLeNet CNN models. Karpathy et al. [7]

When fusing 30-frame models with their optical flow counterparts, no significant increase in performance could be observed. Also, the models trained on optical flow alone yield

a much lower performance compared to training on still image frames. The authors constitute this result to the noisyness of optical flow in the Sports-1M dataset (real-world videos with lower quality and rough cuts).

Yet overall, the best conv-pooling and lstm models perform significantly better than the state-of-the-art as reported by Karpathy et al. [7].

Evaluation on UCF-101:
The UCF-101 dataset contains short videos, lasting 10-15 seconds on average. It is therefore possible to extract frames at higher rates such as 6fps and still capture the complete input-video.

The authors compare frame sampling at different frame rates (30 fps and 6fps) for 30-frame models. They find, that decreasing the sampling rate from 30fps to 6fps slightly increases the classification accuracy, since the model captures more context from the input video (1 second of video against 5 seconds).

In comparison to the Sports-1M dataset, optical flow of UCF-101 provides an increase in performance.

The authors best models, which combine image frames and optical-flow, perform better by a small margin than the state of the art on UCF-101 reported by Simonyan and Zisserman [59].

?? read conclusion!!

3.2.3 Towards Good Practices for Very Deep Two-Stream ConvNets – Wang et al. (2015)

[66]

Wang et al. [66] aim at improving the Two-Stream ConvNet approach for action recognition[59] 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.

Examples of such very deep architectures are GoogLeNet[65] and VGGNet[53]. The authors use these two models to train an enhanced Two-Stream ConvNet model for video action recognition.

The authors identify two main draw-backs of recent ConvNet approaches in video action recognition:

1. The used models are shallow compared to their image counterparts.
   The deepest variant VGG-19 contains 16 convolutional layers, while the networks

of the original Two-Stream ConvNet approach contain 5 convolutional layers.

2. The available dataset are too small, which results in over-fitting.

This work therefore proposes and evaluates a series of good practices to enable very deep ConvNet architectures for accurate action recognition, specifically in the two-stream ConvNet architecture. These good practises 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 two-stream ConvNet approach of Simonyan and Zisserman [59] uses pre-training on the ImageNet dataset for the spatial-stream network only and uses two convolutional networks with five convolutional layers each.

The authors name their approach Very Deep Two-Stream ConvNets.

The spatial net processes a single video frame of input size $(224 \times 224 \times 3)$ and its architecture therefore is the same as in image classification, i.e. either GoogLeNet or VGGNet.

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

Evaluations are done using the UCF101 dataset, specifically the three splits into training- and test-sets that are provided with it. For each split, the training set consists of around 10.000 videos while the testing set consists of 3.000 videos. UCF101 contains 13.320 videos in total and is considered extremely small for training very deep architectures.

Pre-training. The spatial net is pre-trained using an ImageNet model as in [59]. Specifically 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 dropout ratios. Since the networks in both streams have been pre-trained, the authors use smaller dropout ratios as in [59].

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 and horizontal flipping have been widely used to reduce over-fitting.

Since random cropping favors cropping regions in the center of an image/frame, the authors design a corner cropping strategy. Specifically, they crop the four corners and the center of the input images. This increases the variation of inputs and reduces overfitting.

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.

For testing 25 frames or optical flow fields are sampled from a test-video. For each of these input instances 10 inputs to the very deep ConvNet are created by the above mentioned cropping method (four corners, the center and their horizontal flip).

The final prediction is obtained by averaging over all inputs.

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.

Table 10 shows the accuracy for the two evaluated 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 Two-Stream ConvNet architecture when building on different ConvNet models. Results for ClarifaiNet are taken from [59], which isthe original two-stream approach. [66]

The results show, that the proposed good practices and the employment of a very deep convolutional network lead to a significant improvement in performance over the original two-stream appraoch.

3.2.4 Convolutional Two-Stream Network Fusion for Video Action Recognition – Feichtenhofer et al. (2016)

[67]

Feichtenhofer, Pinz, and Zisserman [67] aim at improving the two-stream architecture for action recognition by addressing two main disadvantages of the approach:

1. The architecture is not able to learn pixel-wise relations between the spatial stream and the temporal stream, i.e. it cannot relate extracted object-features to motion features, since the two separate streams are fused 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. Originally 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 conduct a systematical evaluation of fusion methods to find a better way of combining the appearance information extracted by the spatial stream network with the motion information extracted by the temporal stream network. More specifically, fusion techniques combine two feature maps of different ConvNets into a single feature map. When combining all feature maps from a given convolutional layer with all the feature maps in the convolutional layer of another network, two network streams can be combined into one.

Fusion can be conducted either spatially or temporally. Spatial fusion combines feature maps between the streams of the two-stream architecture. Temporal fusion combines feature maps resulting from applying the two-stream architecture on frames at different points of time in the input video.

In the original two-stream approach [59] fusion of the two streams is done by *late fusion*, i.e. the two streams are combined by averaging the predictions of the softmax-output layer.

The authors evaluate temporal and spatial fusion methods and design an enhanced version of the two-stream architecture by implementing the best suited methods.

The objective of spatial fusion is to merge two networks after a given convolutional layer by combining the feature maps of the layers.

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 $y \in \mathbb{R}^{W \times H \times D'}$. A feature map of a convolutional layer is given by the outputs of the neurons in the convolutional layer. W and H corresponds 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.

Sum Fusion:
The sum of the activations in the two feature maps is computed at each location and used as the new feature map. The output feature map has the same dimensions as the input feature maps. Since the ordering of the channels is arbitrary in each network stream, this fusion technique does not implement a semantic relation between the 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 represesent a semantic relation between the feature maps.

Concatenation Fusion:
The two feature maps are stacked into each other along the dimesion 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 the matrix outer product of them. 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 [**lin_bilinear_2015a**]. 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 layer that performs spatial fusion between two network streams has significant influence on the number of trainble parameters in the network, especially if the networks are fused before the fully connected layers and if only the merged stream is kept. Most of the parameters are located in the fully connected layers.

In figure 26 two examples of placing fusion layers into the two-stream architecture are shown.

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) [67]

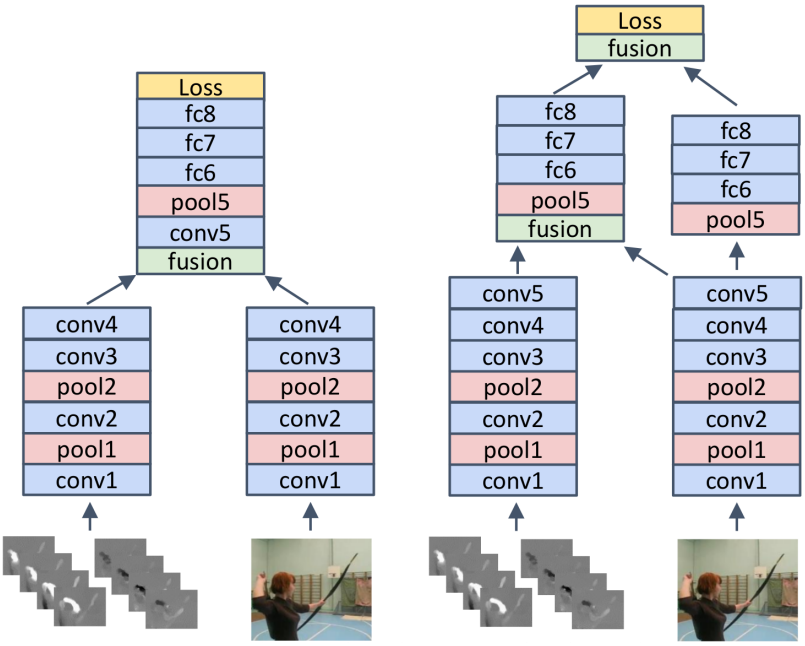


Figure 26: Placement examples of fusion layers for early combination of the spatial and temporal stream in two-stream architectures. (*left*) Only the merged stream is kept. (*right*) The spatial stream is kept after the first fusion layer and fused into it before the final layer [67]

The authors evaluate the spatial fusion methods in a two-stream architecture as implemented in the original approach [59]. Each stream consists of 5 convolutional layers, and three fully connected layers. The accuracy on UCF101 (Split1), the number of paramters and overall number of layers in the architecture is given in table **??**. The streams are fused after the rectification of the fifth convolutional layer and only the fused stream is kept.

Results of the origignal approach are reported in the first row. As can be seen 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 decreses 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 26 (right)).

After evaluating spatial fusion methods, the authors evaluate temporal fusion methods. In order to fuse feature maps temporally, they are considered to be extended over time by applying the two-stream architecture at several temporal points across the video and stacking the resulting feature maps.

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 again correspond to the spatial location, T corresponds to the temporal dimension, i.e. the number of feature maps that get stacked and D again corresponds to the number of channels in the feature map, i.e. the number of filters in the convolutional layer.

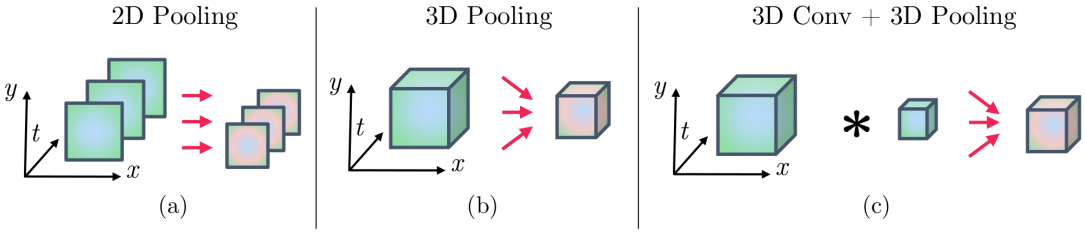


Figure 27: Methods for pooling feature maps along the temporal dimension. [67]

(a) 2D pooling. The feature maps in each channel are only pooled spatially. The network predictions are averaged over all the 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.

The authors evaluate the pooling methods and find that 3D Conv + 3D Pooling performs best.

The final approach for applying the two-stream ConvNet model to action recognition is implemented by the authors as follows:

Take away message in abstract.

3.3 Trajectory Pooling of Deeply Learned Features

3.3.1 Action recognition with trajectory-pooled deep-convolutional descriptors – Wang et al. (2015)

Wang, Qiao, and Tang [68] propose a novel method of building video representations by combining the advantageous porperties of hand-crafted features as in *Dense Trajectories* [33] with deeply learned features as in *Two-Stream Convolutional Networks* [59].

In the *Improved Dense Trajectories* approach, local features are extracted along trajectories, which are mostly located near prominent motion in a video. The authors 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 the approach for action recognition is as follows:

1. A two-stream convolutional network architecture is trained on multiple scales of a large dataset.
2. *Dense Trajectories* are begin extracted from the input video according to the approach in [33].
3. The deeply learned feature maps of the two-stream architecture are used to extract local descriptors, by pooling the convolutional responses 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. cite ??
5. A linear SVM is trained to assign the action label to the FV representations, i.e. to perform action recognition.

The authors evaluate their approach on the HMDB51 and UCF-101 dataset. Results show, that TDD is complementary to HOG, HOF and MBH and therefore fusion of these descriptors can further boos performance.

In contrast to [33], trajectories are only extracted on the original spatial scale, because it is computationally more effective. To compensate, multi-scale TDDs are extracted around the trajectories.

Any convolutional architecture can be embedded in a two-stream setup for TDD extraction. The authors choose the Clarifai network [69] with less filters in the *conv4* layer (512 instead of 1024) and a lower dimensional *full7* layer (2048 instead of 4096). Architectural details are depicted 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 [68]

This architecture is chosen for implementing the spatial stream network as well as the temporal stream network.

After training of the two-stream architecture, the activations of the convolutional layers in each stream (feature maps) are used for extracting the TDD descriptors of an input video.

Before 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 (see table 12). Specifically, the p-th point in a video trajectory (x_p, y_p, z_p) is represented by point $(\overline{r \times x_p}, \overline{r \times y_p}, z_p)$. Where $\overline{(.)}$ denotes the mean-value.

Given a video V, training of 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 descriptor called TDD from a 3D volume around a trajectory:

1. Feature Map Normalization (two different methods)
    - Spatiotemporal Normalization

- Channel Normalization

2. Trajectory pooling

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 $C \in \mathbb{C}(V)$:

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

This normalization method 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:

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

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

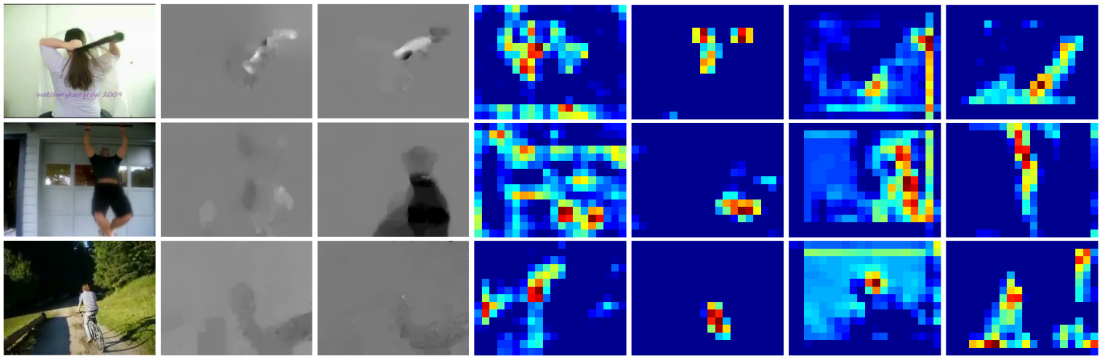
Both normalization methods were evaluated experimentally and the authors found, that combining the resulting video representations from both normalization methods by late fusion yields the best results.

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.

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

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 cite ?? 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 28: Snapshots of input videos (a), their corresponding optical flow fields in x- and y-dimension (b - c), the corresponding activations of the feature maps used for TDD extraction (d - g) [68]

Using TDDs for action recognition is evaluated on the UCF-101 and HMDB-51 datasets. 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 [70] 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 [71] (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 [59].

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 in action recognition.

Combining the descriptors of the spatial and temporal net in the TDD approach significantly outperforms the previous state of the art (two-stream convolutional network [59]). Quantitative 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) [33] and two-stream ConvNets [59]. [68]

3.3.2 ?Pooling the Convolutional Layers in Deep ConvNets for Action Recognition – Zhao et al. (2015)

[72]

3.4 Generative Models

One of the most popular deep learning models that can be trained unsupervised is the restricted Boltzmann machine [5]. A nice property of RBMs is that they can be greedily trained and stacked on top of each other to form deep belief networks (DBNs) [6], allowing us to learn representations of gradually increasing complexity. Training of these models using maximum likelihood learning is intractable, but they can be trained using contrastive divergence [4]. (Palasek Patras)

Auto-Encoders:
The most common architecture used is an auto-encoder which learns representations based on its ability to recon- struct the input images [35, 3, 49, 37]. Wang an Gupta (Unsupervised learning of visual representations using video.

Restricted Boltzmann Machine:
Boltzmann Machines 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 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, 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 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.

Restricted Boltzmann Machines correspond to feed-forward neural network models when reusing the learned weights in a topological equivalent NN model. This is how unsupervised pre-training is conducted.

Models that can be trained in an usupervised manner are especially attractive for the video-domain where labeling data is costly.

The RBM can be extended to process real-valued inputs.

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. (Lee)

3.4.1 Unsupervised Learning of Video Representations using LSTMs – Srivastava et al. (2015)

[73]

Srivastava, Mansimov, and Salakhudinov [73] design models based on Long-Short-Term-Memory (LSTM) recurrent neural networks and evaluate their ability to learn representations of input videos in an unsupervised way from unlabeled video data. Similar to the approach taken in auto-encoders, an encoder LSTM network maps input sequences of vectors into a fixed-length representation, which is then decoded by another LSTM network into a target sequence. The authors consider different choices for the target sequence: the reversed input sequence, a sequence of future frames after the input ended

or a hybrid approach where two LSTM networks decode the representation into both these choices simultaneously. Inputs to the LSTM encoder-network can be any kind of video representation, the authors evaluate raw image patches and high level "percepts" that are extracted by a CNN.

The results of the unsupervised setting are evaluated qualitatively by printing the decoded sequences. For a quantitative evaluation, the unsupervisedly pre-trained models are fine-tuned for an supervised action recognition task on the UCF-101 and HMDB-51 dataset.

Three different types of models are being evaluated, which are described below.

1. LSTM Auto-encoder model.
2. LSTM Future Predictor model.
3. A Composite (hybrid) model.

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

The model is depicted in figure 29.

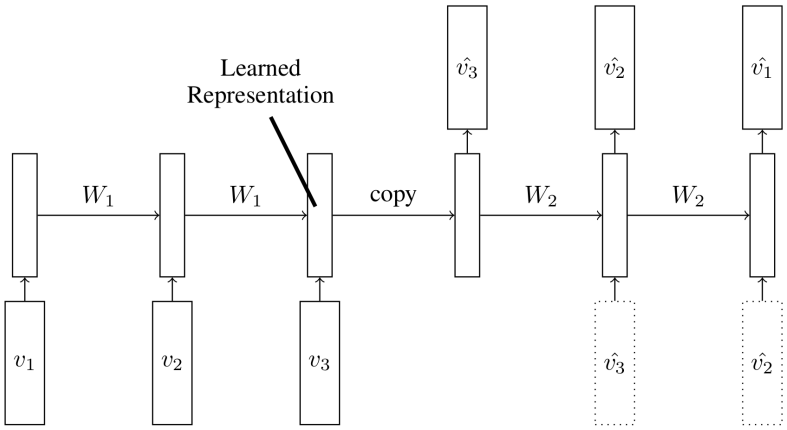


Figure 29: Auto-encoder model [73]

$W_i (i \in \mathbb{N})$ denotes the parameters of the respective LSTM network that are, according to its recurrent nature, applied to its own hidden state given an input v_i of the input

sequence $\{v_1, \cdots, v_T\}$ of length T. $\hat{v}_i (i \in \mathbb{N})$ denotes the outputs of the decoder network. The decoder network can be conditional, i.e. it receives its last generated output as input.

The **LSTM Future Predictor model** also consists of two LSTM networks but predicts frames that continue after the input sequence. The architecture is the same as in the Auto-encoder model, as shown in figure 30. This decoder LSTM network can again be unconditioned or conditional.

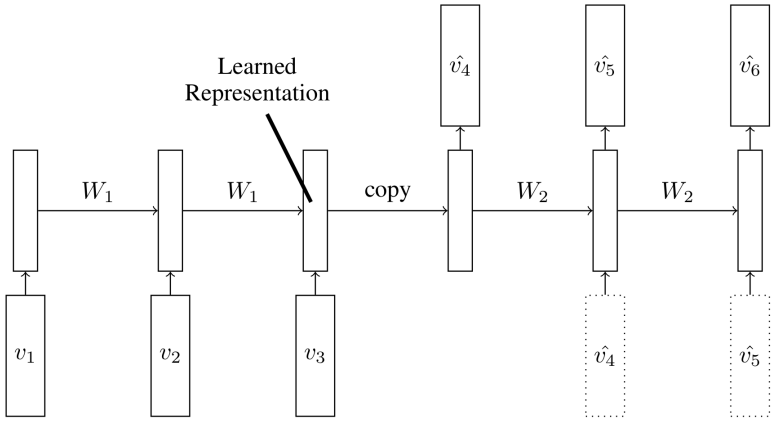


Figure 30: Future-predictor model [73]

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 31, therefore puts the additional constraint on the encoder networ, that its representation must allow both tasks.

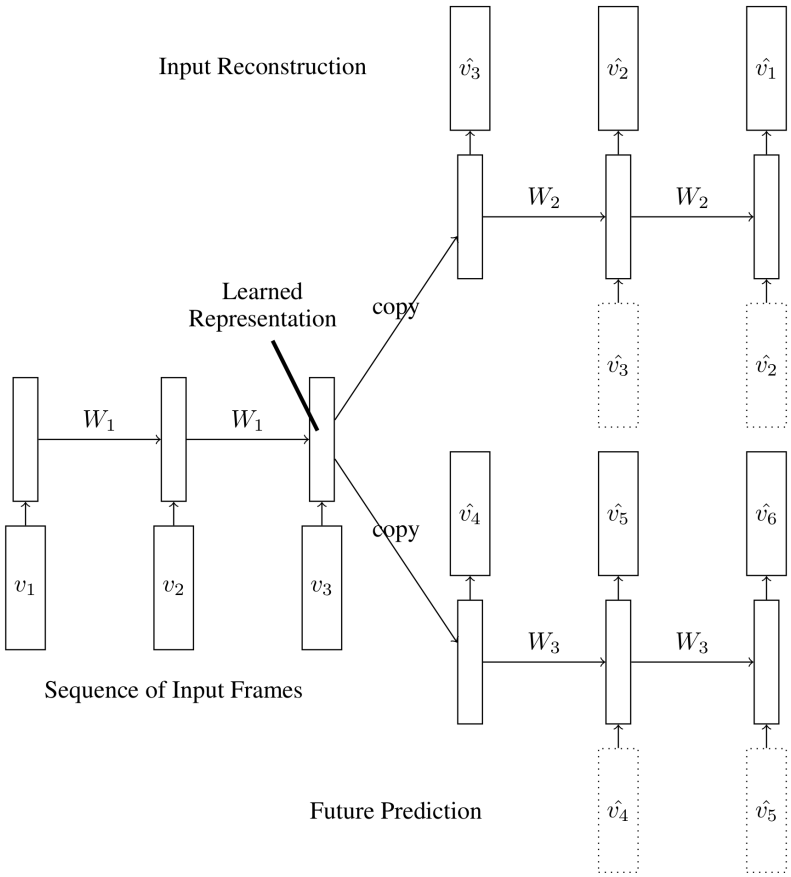


Figure 31: Composite model [73]

The authors provide a qualitative analysis of the properties of their proposed models through visualization. First composite models, with one and two layers consisting of 2048 LSTM units, are trained on a syntethic dataset of two moving MNIST digits in a 64×64 pixel area. The results for image reconstruction and future prediction shown in figure 32.

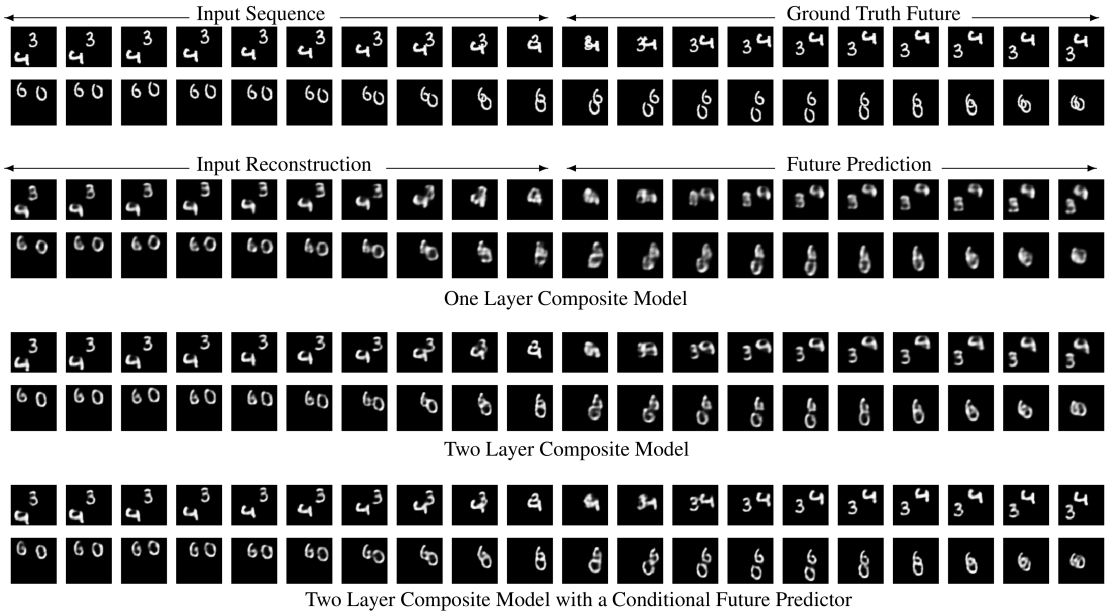


Figure 32: Input reconstruction and future predictions of composite models on a synthetic dataset of moving MNIST digits. [73]

These results show, that the model is able to make fairly good predictions on this dataset. Adding a second layer and making the future predictor conditional further improves the predictions.

When applying the model on 32×32 image patches of real-life videos from the UCF-101 dataset, results show, that the future prediction quickly blurs out.

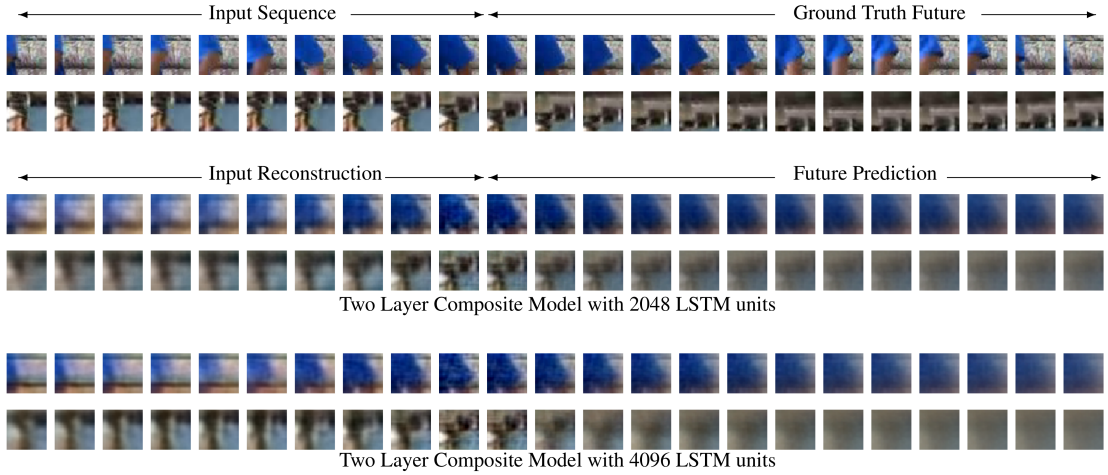


Figure 33: Image reconstruction and future prediction of a two-layer composite model with a different number of LSTM units in each layer. [73]

The authors evaluate if features obtained from unsupervised learning can increase the accuracy in supervised action recognition: A two layer composite model is trained in an unsupervised way on 300 hours of video, composed of 10 seconds clips randomly sampled from the Sports-1M dataset. Once trained, a LSTM classifier is initialized with the weights of the encoder network (figure 34). Depending on the used benchmark, this classifier is then fine-tuned using backpropagation on either UCF-101 or HMDB-51. 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 [59], as inputs.

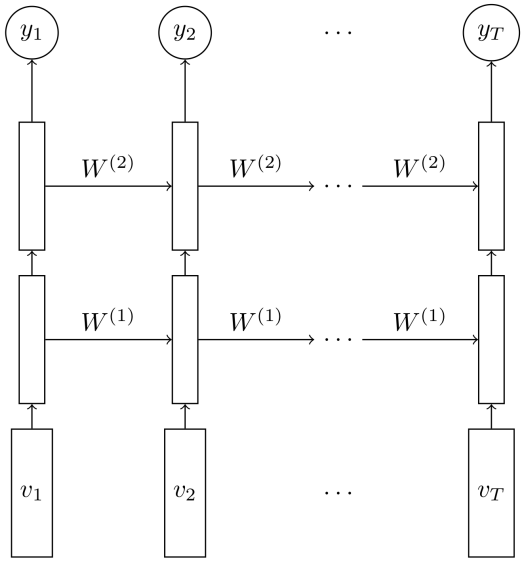


Figure 34: LSTM classifier applied on an input sequence $\{v_1, \cdots, v_T\}$ [73]

Baseline method is initializing the model with random weights. The results 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, the improvement becomes smaller (about 1%).

The authors conclude: Even models pretrained on unrelated datasets (300 hours of YouTube videos) can help action recognition performance. Unsupervised learning gives a significant improvement if only few training examples are available.

3.4.2 Action Recognition Using Convolutional Restricted Boltzmann Machines – Palasek and Patras (2016)

[74]

Palasek and Patras [74] apply Convolutional Deep Belief Networks (ConvDBNs), which are generated by stacking Convolutional Restricted Boltzmann Machines (ConvRBMs), to static video frames to learn video representations for human action recognition in an unsupervised fashion.

The ConvRBM was initially proposed by Lee et al. [75] 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. [75]
2. DBNs do not take the special structure of images into account, specifically that objects can occur in any area of the image. Features therefore have to be learned for each location in the image separately. [75]

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 35. The visible layer V is constituted of binary input units, but can be easily modified to handle real-valued inputs. The hidden layer consists of K groups, with a filter W^k belonging to each group ($k \in \{1, \cdots, K\}$). To illustrate the equivalency to other translational invariant architecture such as CNNs, which consists of detection and pooling layers, the hidden layer is denoted as detection layer in Figure 35. 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. The hidden units share a bias b_k per group and all visible units have a single bias c. [75]

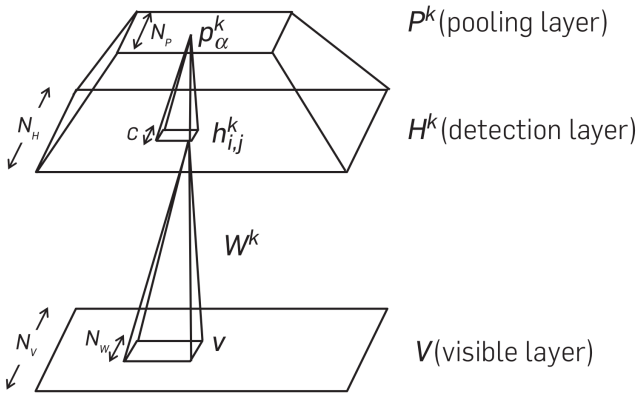


Figure 35: Architecture of the Convolutional Restricted Boltzmann Machine. The figure shows group k in the detection- and pooling-layer. [75]

ConvRBMs are stacked above each other to form a more expressive deep architecture, called Convolutional Deep Belief Networks (ConvDBNs). Equivalently to Convolutional Neural Networks, probabilistic max-pooling is used to reduce the dimensionality of each hidden (detection) layer. Since regular max-pooling was designed for deterministic models, Lee et al. [75] derive a probabilistic max-pooling method, in order to enable full probabilistic inference in their 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. [75]

The energy function of the ConvDBN is defined by summing the energy functions of the individual ConvRBMs. ConvDBNs are trained in a greedy, layer-wise way: After a layer is trained individually, its weights are frozen and its activations in the hidden/detection layer are taken as inputs for the following layer. [75]

Palasek and Patras [74] devise and evaluate an approach for using the activations of three stacked ConvRBMs (a ConvDBN) for 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 patches and aggregate them into a video representation that can be classified using a SVM.

Different configurations for the ConvDBN are being evaluated as feature extractors:

1. 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.

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

3. 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 36.

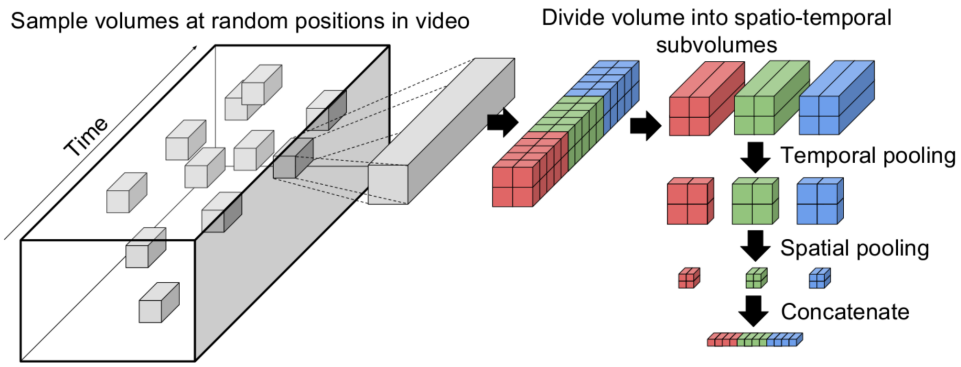


Figure 36: Approach for generating fixed sized video-representations using ConvDBN activations. An input video is handled as video-volume of stacked frames. [75]

Given an input video, video subvolumes of size 32×32 pixels and of length 15 subframes are extracted at random positions in the video-volume. 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 anlong the temporal dimension, spatially pooled and concatenated to form the final representation of the given subvolume (see figure 36). [75]

A Gaussian Mixture Model is trained on all the extracted feature representations to extract the final Fisher Vector representation of the video. These Fisher Vector can then be classified using a SVM.

Their model is unsupervisedly trained on greyscaled, static video-frames of the UCF-101 dataset, which contains a total of 1700000 frames. For each of the 9537 videos in split 1 of the UCF-101 dataset, 1000 subvolumes are extracted to train the ConvDBN in an unsupervised way.

Palasek and Patras [74] evaluated their approach using activations from different ConvDBN layers. 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. [75]

3.5 Temporal Coherency Networks

"Temporal Coherency is a form of weak supervision and states that consecutive frames are correlated both semantically and dynamically." going deeper into action recognition.

Previous unsupervised approaches only yielded moderate increase in accuracy against initializing the weights randomly.

Misra: Example tasks: Ordering of frames, determining the relative temporal proximity of frames.

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

[76]

Misra, Zitnick, and Hebert [76] propose an efficient unsupervised representation learning method 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 drawing short sequences of frames from a video and switching frames in a subset of all sampled sequences.

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.

Learning the validity of a sequence of frames yields a representation that captures the motion of persons and objects in the scene. It therefore embeds information that is important for accurate action recognition.

The authors evaluate their method using a Convolutional Neural Network architecture, that is applied to each frame of the sequence in parallel. They propose using input sequences of three frames, since four or five frames did not yield a significant improvement in performance. Results are obtained using the UCF-101 and HMDB-51 benchmark datasets.

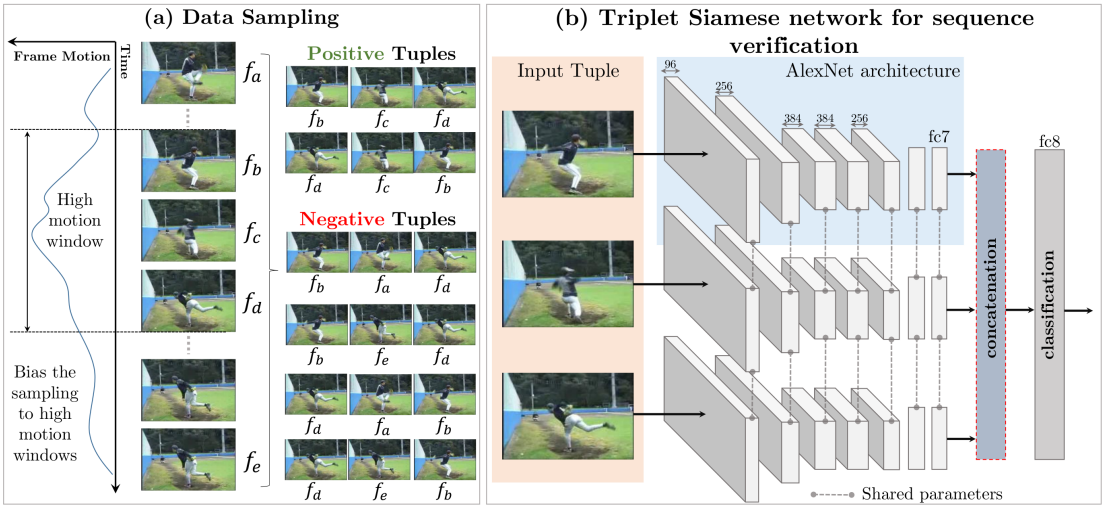


Figure 37: Sampling method of input sequences and triplet Convolutional Neural Network for temporal order verification [76]

Figure 37 shows the approach for sampling input sequences from an unlabeled video and the triplet CNN architecture for classifying these sequences.

The three frames, that are used to construct positive and negative input sequences are sampled from regions in the video, that 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 the standard CaffeNet implementation available in Caffe??, which is a slight modification of AlexNet ??. 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 incorrect order.

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 first layer 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-tune 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 14)

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 14: 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. [76]

On UCF-101 pre-training using 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, pre-training on UCF-101 and pre-training using temporal oder verification. The improvement of the latter is smaller compared to the reulsts of UCF-101 but still significant (increase in mean accuracy of +4.7%).

According to the authors, this results show the informativeness of video representations learned by temporl order verification.

qualitative analysis

3.5.2 Misc

Modeling Video Evolution For Action Recognition? – Fernando

Actions Transformations.

3.6 Comparison

Nice comparison in related work section of "Beyond Temporal Pooling: Recurrence and Temporal Convolutions for Gesture Recognition in Video" Convolutions for Gesture Recognition in Video

One problem of deep learning methods is that they require a large number of labeled videos for training, while most available datasets are relatively small. Meanwhile, most of current deep learning based action recognition methods largely ignore the intrinsic difference between temporal domain and spatial domain, and just treat temporal dimension as feature channels when adapting the architectures of ConvNets to model videos.

Introduction of Feichtenhofer: Current state of the art - Tran and TDD.

4 Datasets and Benchmarks in Action Recognition

The creators of UCF-101 and HMDB-51 provide three splits of their datasets into training- and testing-data. The standard evaluation procedure is to report the average accuracy over those three splits, which the authors follow in this work as well. simonyan zisserman two-stream approach

"From a practical standpoint, there are currently no video classification benchmarks that match the scale and variety of existing image datasets because videos are significantly more difficult to collect, annotate and store." cite large-scale image classification

"In particular, commonly used datasets (KTH, Weizmann, UCF Sports, IXMAS, Hollywood 2, UCF-50) only contain up to few thousand clips and up to few dozen classes. Even the largest available datasets such as CCV (9,317 videos and 20 classes) and the recently introduced UCF-101[22] (13,320 videos and 101 classes) are still dwarfed by available image datasets in the number of instances and their variety [7]." cite large-scale image classification

"In [4], Gao et al. presented a comprehensive study on the influence of the evaluation protocol on the final results. It was shown that the use of different experimental configurations can lead to performance differences up to 9%." "Action recognition methods are usually directly compared although they use different testing protocols or/and datasets (KTH1 or KTH2), which distorts the conclusions." cite sequential deep learning for human action recognition.

"Unfortunately, since the creation of the dataset, about 7% of the videos have been removed by users. We use the remaining 1.1 million videos for the experiments below. Although Sports-1M is the largest publicly available video dataset, the annotations that it provides are at video level. No information is given about the location of the class of interest. Moreover, the videos in this dataset are unconstrained. This means that the camera movements are not guaranteed to be well-behaved, which means that unlike UCF-101, where camera motion is constrained, the optical flow quality varies wildly between videos."

Recent research focuses on realistic datasets collected from movies [20, 22], web videos [21,31], TV shows [28], etc. These datasets impose significant challenges on action recognition, e.g., background clutter, fast irregular motion, occlusion, viewpoint changes. [improved dense trajectories 2013]

"Another large scale dataset is the THUMOS dataset [8] that has over 45M frames. Though, only a small fraction of these actually contain the labelled action and thus are useful for supervised feature learning." Feichtenhofer 2016

Due to the label noise, learning spatiotemporal ConvNets still largely relies on smaller, but temporally consistent datasets such as UCF101 [24] or HMDB51 [13] which contain short videos of actions. This facilitates learning, but comes with the risk of severe overfitting to the training data. Feichtenhofer 2016

UCF101 is considered extremely small. cite Towards good practices for very deep two-stream ConvNets – wang 2015.

The use of publicly available datasets has two main advantages. On the one hand, they save time and resources, that is, there is no need to record new video-sequences or pay for them, so researchers can fo- cus on their particular algorithms and implementations. On the other hand, and this is even more important, the use of the same datasets facilitates the comparison of different approaches and gives insight into the abilities of the different methods. This survey is mainly focused on the video datasets that are composed by heterogeneous action sets, i.e., typical actions that can appear in a variety of situations or scenarios and are recorded by visible spec- trum cameras. Chaquet A survey of video datasets for human action and activity recognition.

4.1 Review of Benchmarking Datasets for Action Recognition Algorithms

Review of the datasets for human action recognition, that contain a single person performing an action.

All datasets are publicly available and the download links can be found in the referenced publications.

We will give a general overview of the most common datasets but focus on newest ones (since 2013).

Detailed and comprehensive review of action recognition dataset ranging from 2001 to 2012 was released in 2013 by Chaquet, Carmona, and Fernández-Caballero [77]. They provide a fine-grained classification by the type of actions present in the videos, i.e. heterogeneos actions, specific actions and others (see figure 38).

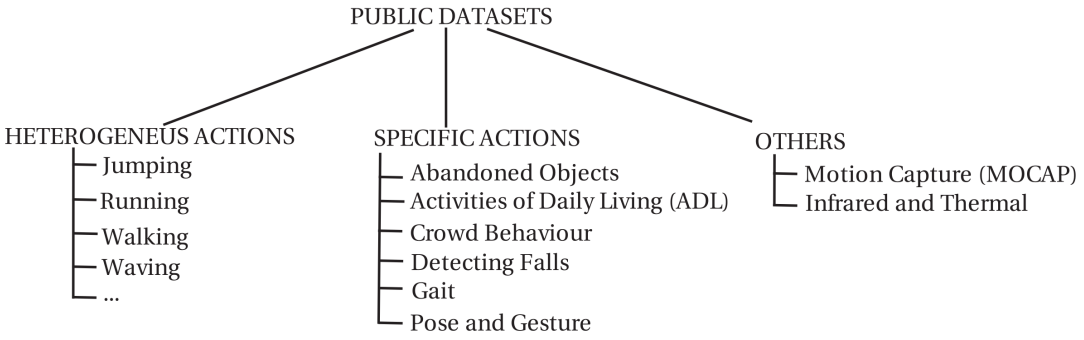


Figure 38: Taxonomy of publicly available datasets provided by [77]

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 [77] 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.

In the following: Classification by age, increasing by complexity, the oldest datasets being the least complex ones, as done in [78]

4.2 Early Benchmarking Datasets – Controlled Conditions

4.2.1 KTH – 2004

[40]
KTH 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 [77] and is the most commonly used publicly available dataset of human actions [45]. 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: [40]

- 2391 action instances in controlled conditions.
- 6 action classes recorded in 4 different scenarios (see figure 39).
- 25 different actors.
- Homogeneous, uncluttered backgrounds.
- Static camera.
- 160×120 pixels video-resolution @ $25 fps$
- $4s$ average sequence length

Example frames of the KTH dataset, ordered by action class and recording scenarios, are depicted in the following figure 39.

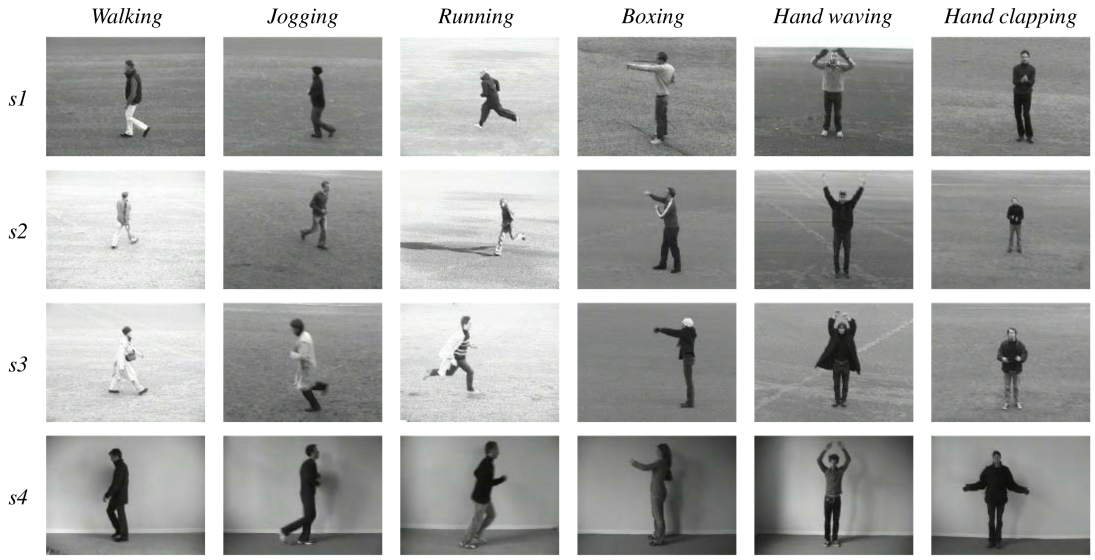


Figure 39: KTH dataset example frames – 6 different actions classes in 4 different recording scenarios: (s1) outdoors, (s2) outdoors with scale change, (s3) outdoors with chaning clothes, (s4) indoors [40]

4.2.2 Weizmann – 2005

[79]
One year after KTH, the Weizmann dataset (also known as Weizmann actions as space-time shapes dataset) was released in 2005 by the Weizmann Institute of Science in Israel. The dataset contains more action classes than KTH but features less performers per action class. The actions are recorded in controlled conditions. Additionally 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 the asymmetrical nature of the action. [80]

Dataset details: [80]

- 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 @ $25fps$.
- $3.66s$ average clip length.

Example frames for some action classes are shown in the following figure 40.

Figure 40: Example frames for the action classes *jumping jacks*, *run*, *walk* and *gallop sideways* along with provided foreground silhouette-masks [79]

4.2.3 IXMAS – 2006

[81]
The INRIA Xmas Motion Acquisition Sequences dataset (IXMAS) was initially released in 2006 by the French National Institute for Computer Sciences (INRIA) [81]. 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 synchro-

nized cameras) of male and female actors performing atomic motions of the daily-living. The initial release, as described in [81], contained 11 action classes, performed by 10 actors and was later extended.

Dataset details (extended release): [82]

- 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.

Example frames of the 11 actions in the initial release are shown in figure 41 below. Figure 42 shows the five different camera perspectives of the action *checking watch*.

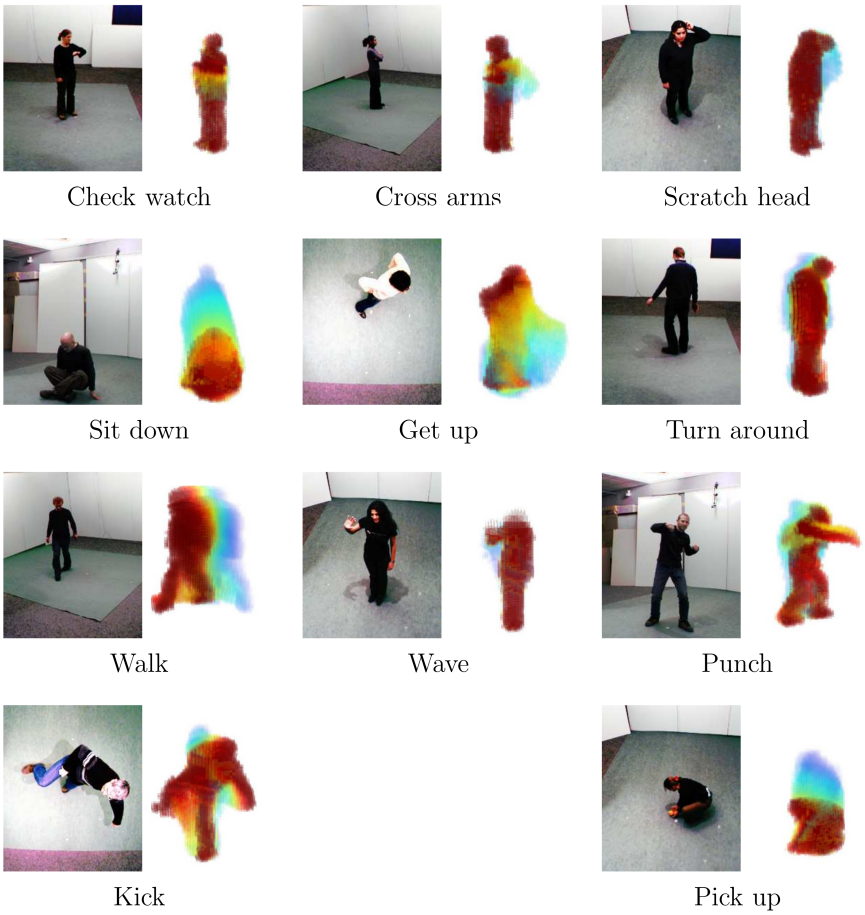


Figure 41: Example frames for the action classes in the initial IXMAS release [81]

Figure 42: Action example of *checking watch* captured from five different cameras [81]

4.3 Interim Benchmarking Datasets – Television and Movies

4.3.1 UCF Sports – 2008

[83][84]
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 were released. 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 high quality and presents moving cameras and changing backgrounds.

The initial release of the dataset [83] contained nine action classes: diving, golf swining, kicking, lifting, horseback riding, running, skating, swinging a baseball bat and pole vaulting Example frames of the first release are shown in figure 43. For a second release [84] 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 44.

Dataset details (second, final release): [85]

- 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.

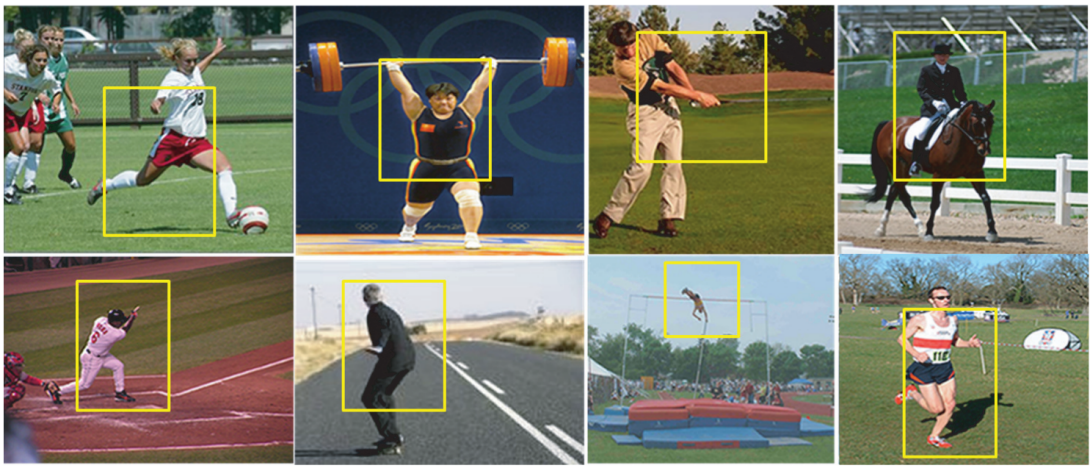


Figure 43: Example frames of UCF Sports Dataset (release 1) [83]

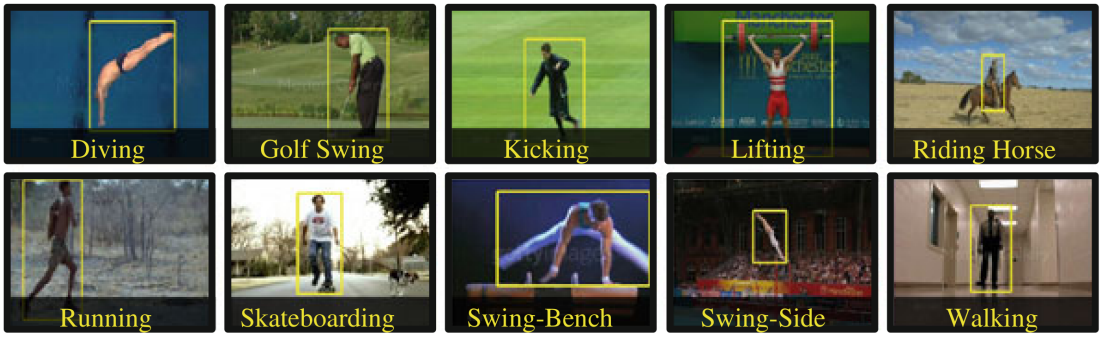


Figure 44: Example frames of UCF Sports Dataset (release 2) [84]

4.3.2 Hollywood – 2008

[5]
The Hollywood dataset was released in 2008 at the IRISA Institue in France and contains video samples of human actions drawn 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 [77]. 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 [80].

Dataset details: [86]

- $233 + 219 + 211$ action clips (automatic + 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 45.

Figure 45: Example frames for the eight classes of Hollywood I dataset with highest confidence for true/false positives/negatives [5]

4.3.3 Hollywood 2 – 2009

[87]
The Hollywood 2 dataset is an extension of the Hollywood dataset and was released a year later 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 a total of 69 movies. Equivalently to the Hollywood dataset, automatically and manually verified labels are provided in two training sets.

Dataset details (action clips): [88]

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

Additional classes of the Hollywood 2 datasets are shown in figure 46.

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

Figure 46: Example frames for the additional classes of Hollywood 2 dataset [87]

4.4 Modern Benchmarking Datasets – Videos in the Wild

4.4.1 UCF11 Youtube Action – 2009

[89]
The UCF11 dataset, 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 proprties of the last release are reported.

Dataset details: [90]

- 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.

Example frames of the dataset are shown in figure 47.

Figure 47: Example frames of the 11 classes in the UCF YouTube actions dataset [89]

4.4.2 UCF50 – 2010

[91]
The UCF50 dataset 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: [80]

- $6,676$ action clips.
- 50 action classes (see figure 48).
- At least 100 clips per class.
- 320×240 pixels video-resolution.

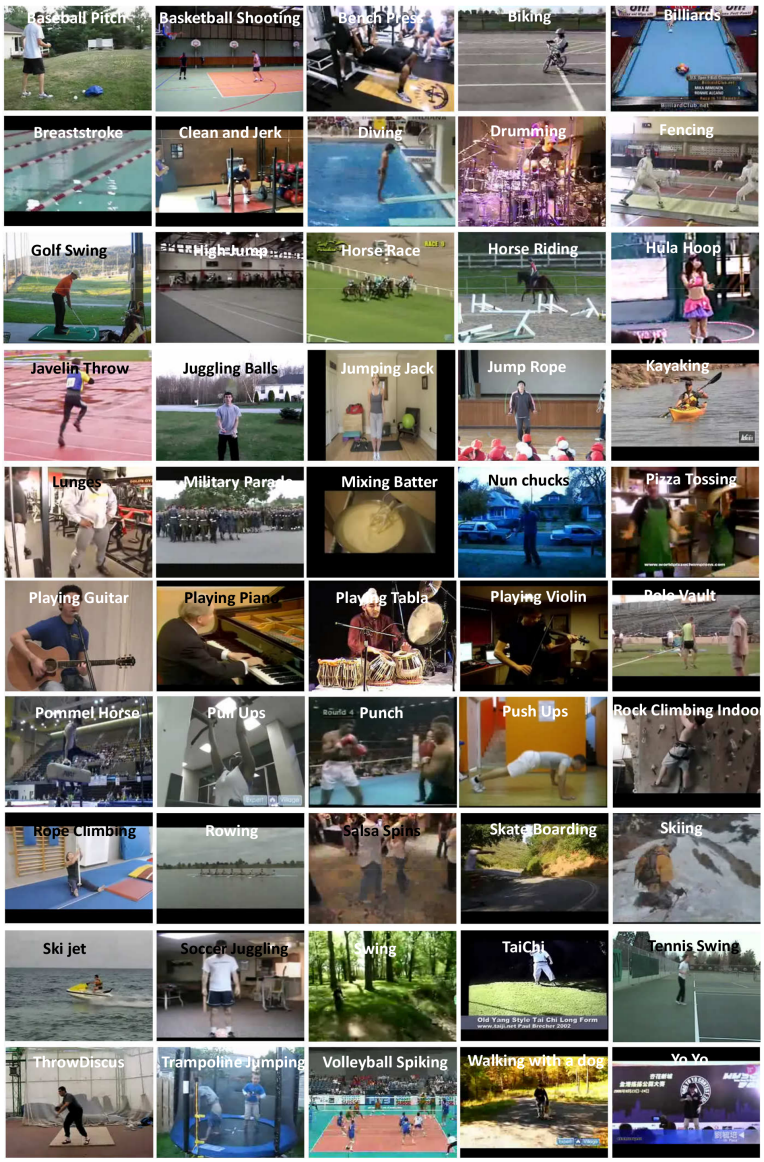


Figure 48: Example frames of the 50 classes in UCF50 [91]

4.4.3 HMDB51 – 2011

[92] The Human Motion Database (HMDB) was created by the Brown University in the USA and was released in 2011 [92]. The authors discuss the big amount of videos uploaded to the internet each day and emphasize on the importance of algorithms to automatically organize and retrieve this data. 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-heigth 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: [93]

- 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 interfers 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. [92]

The action classes in the datset can be grouped into the following five categories: [92]

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)

Example frames for each of the action classes in HMDB51 are shown in figure 49.

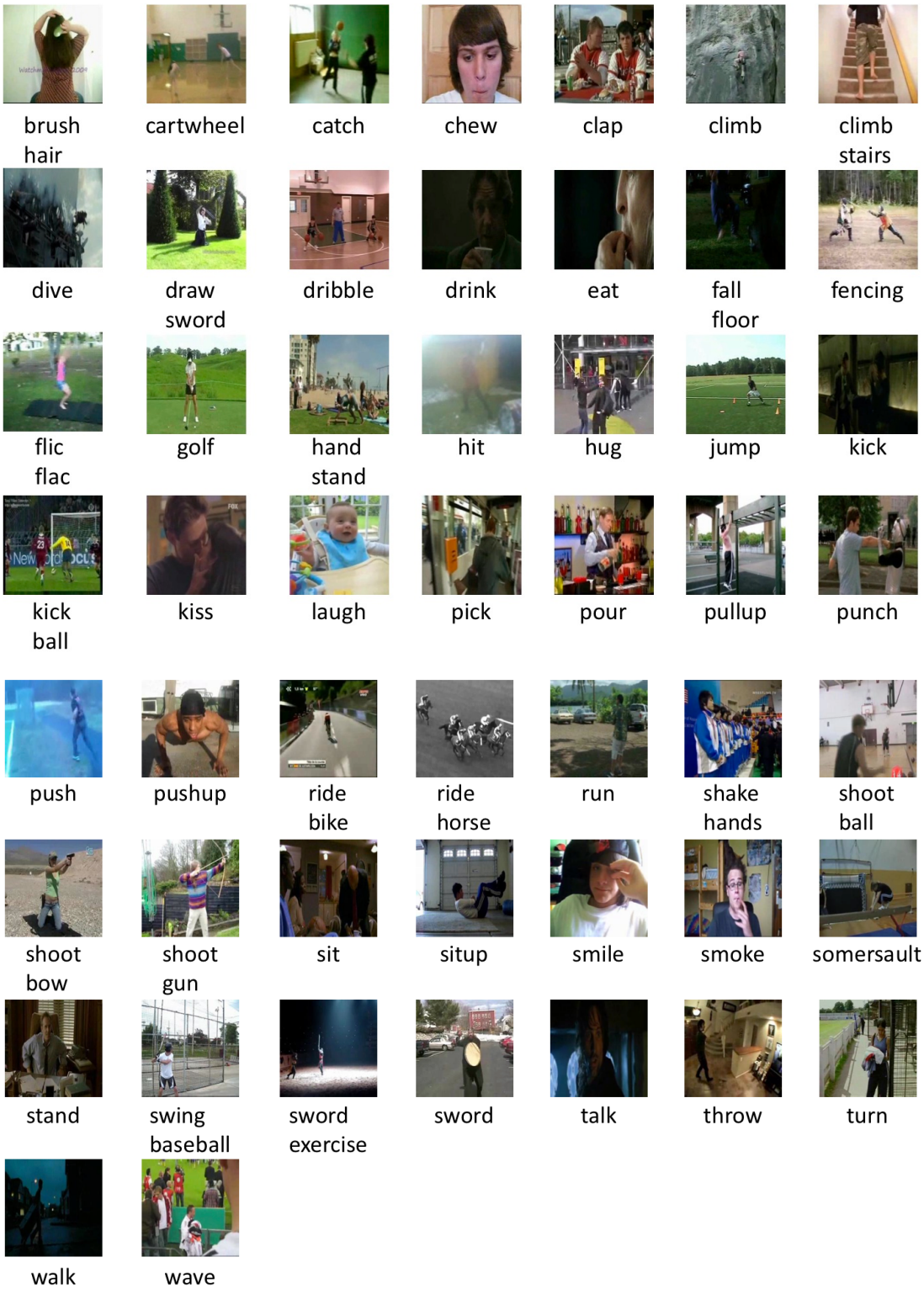


Figure 49: Example frames for the 51 classes in HMDB51 [93]

4.4.4 UCF101 – 2012

The UCF101 dataset 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*. [52]

Example frames of the dataset are shown in the following figure **??**. The action classes can be divided into five categories: [52]

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

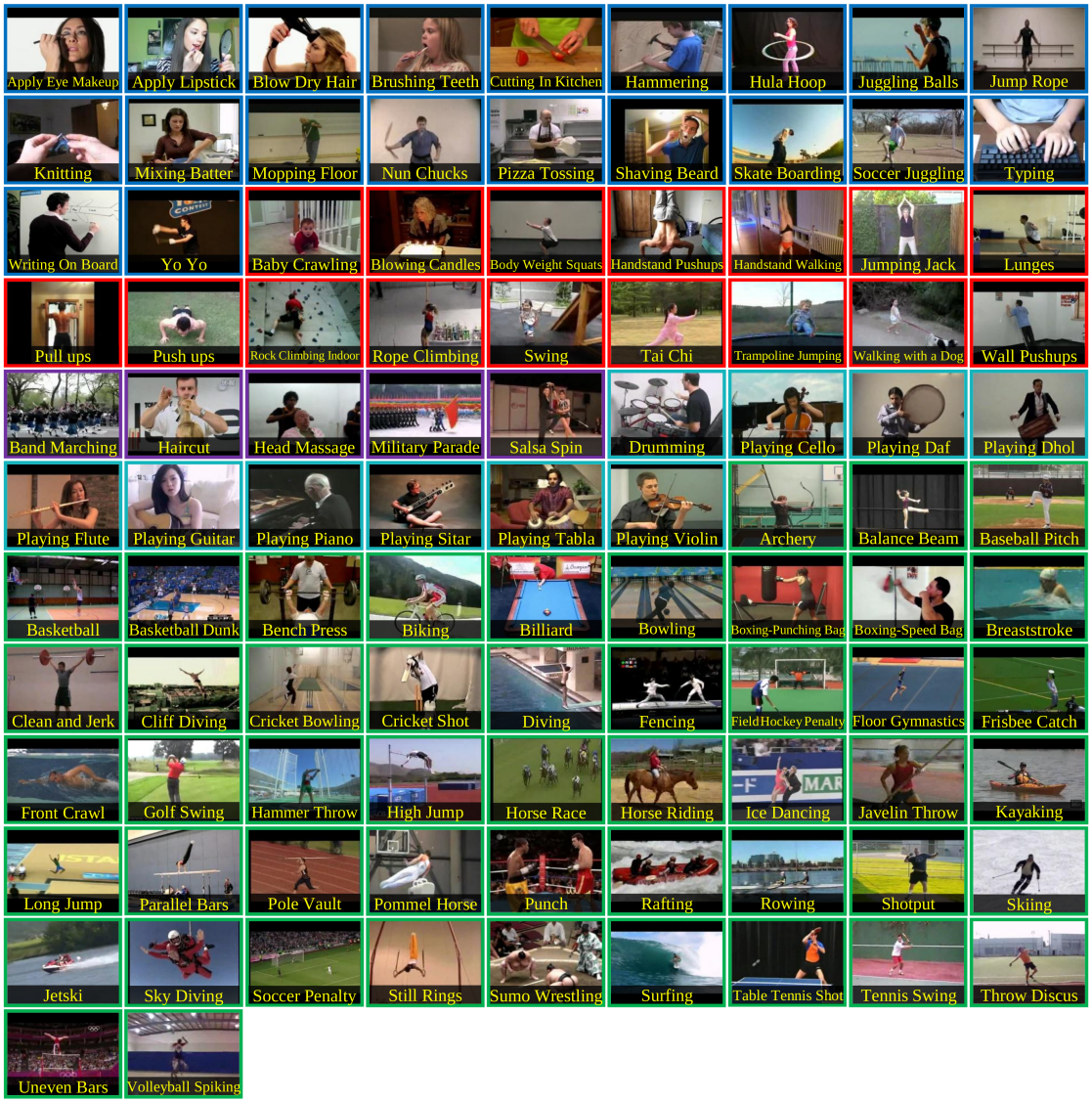


Figure 50: Example frames of the 101 classes in UCF101 [52]

4.4.5 ASLAN – 2012

[94]
In contrast to the previously presented benchmarking datasets, the *Action Similarity Labeling* collection (ASLAN) [94] emphasizes on a different evaluation protocoll 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 [94] 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.

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: [94]

- $3,631$ 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.

Evaluation protocol:
The authors provide two different divisions of their database, wich 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 51: Example frames of video pairs labeled *same* [94]

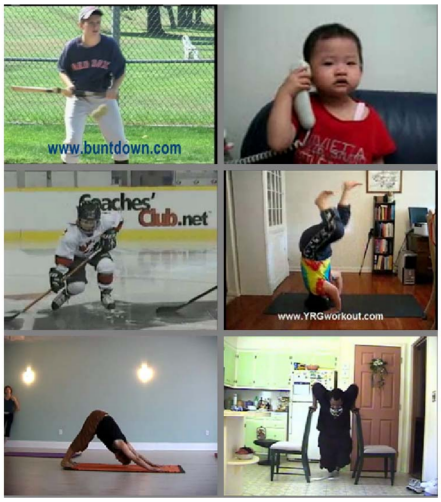


Figure 52: Example frames of video pairs labeled *not same* [94]

4.4.6 Sports-1M – 2014

[7] The Sports-1M dataset was released in 2014 by the Computer Science Department at Stanford University [7]. 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 [63].

Because of the above stated properties of the Sports-1M it is said to be used with caution [80], although it is the biggest action recognition dataset available to date.

4.4.7 THUMOS

THUMOS is a large scale dataset (Feichtenhofer)

4.5 Activities of Daily Living (ADL) Datasets

4.5.1 URADL – 2009

The University of Rochester Activities of Daily Living Dataset (URADL) [95] 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 datset 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 53.

Figure 53: Example frames for the action classes in the URADL dataset [96]

4.5.2 Multiple Cameras Fall Dataset – 2010

[97] The Multiple Cameras Fall Dataset was released in 2010 by the University of Montreal, Canada [97] 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 and contains the recordings of 24 events in eight different perspectives, performed by a single actor. 22 of the events include a falls, which are performed during confounding actions such as walking, housekeeping and activities with similiar 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 54). The dataset only provides the raw recordings without action annotations.

Figure 54: Example frames of falls (a) and confounding activities performed before or after a fall (b) [97]

4.5.3 MPII Cooking Activities Dataset – 2012

[98] The MPII Cooking Activities Dataset 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 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.

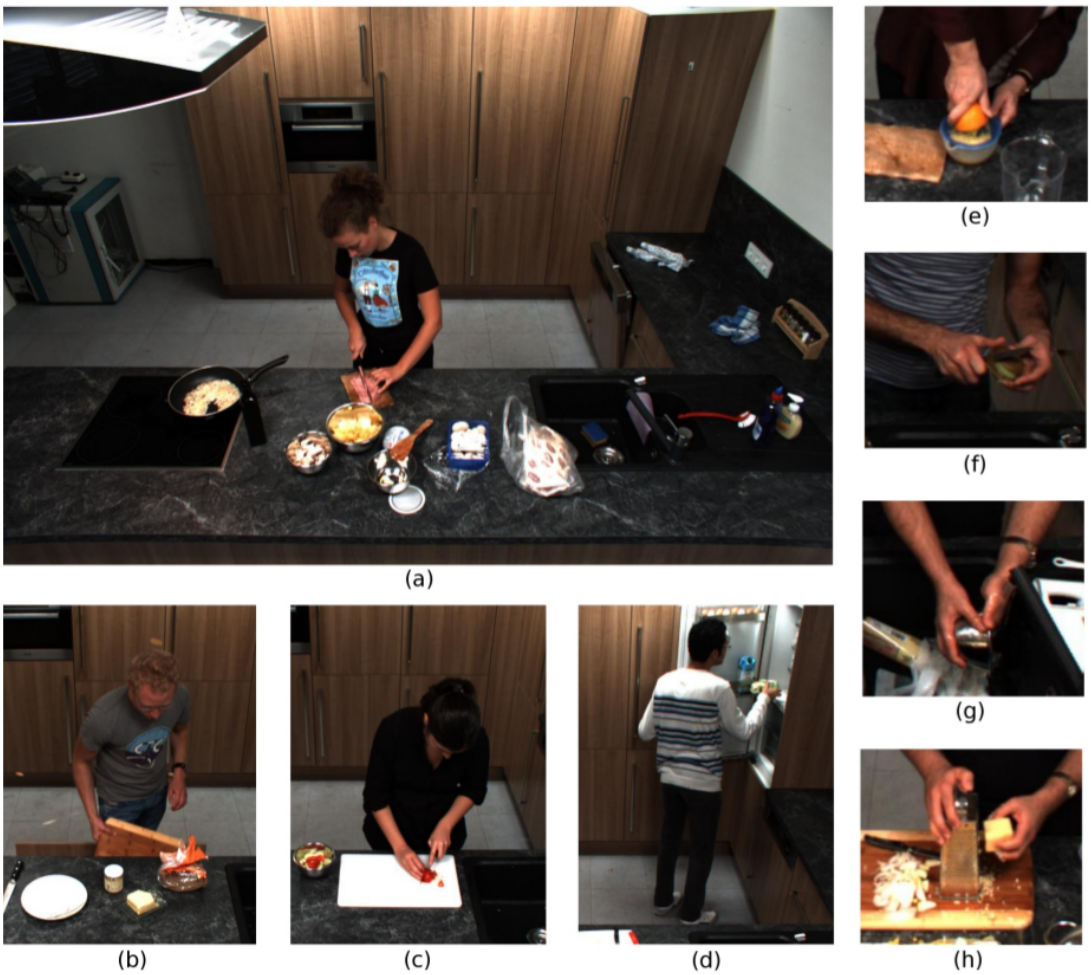


Figure 55: Example frame and crops of cooking activities. (a)cut slices, (b)take out from drawer, (c)cut dice, (d)take out from fridge, (e)squeeze, (f)peel, (g)wash object, (h)grate. [98]

4.5.4 ActivityNet

[99] ActivityNet 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. The latest release of the dataset (release 1.3 of march 2016, as available from the official website [100]) consists of annotated links to YouTube videos, organized in 200 activity classs. The labels are provided in JSON-format and contain the video-ID, the activity labels along with beginning- and end-time, 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 providing annotated video-links, a taxonomy of the 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 56 shows the visualization of the activity category *Household*.

Figure 56: Taxonomy of top-level activity category *Household*

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

4.6 Algorithm Testing Protocol

4.7 Data Augmentation Methods

refer to Imagenet classification with deep convolutional neural networks.

RGB colour Jittering

the improved data augmentation scheme (different aspect-ratio, fixed crops) from [61] for all our methods and baselines. 61: Wang, L., Xiong, Y., Wang, Z., Qiao, Y.: Towards good practices for very deep two-stream convnets. arXiv preprint arXiv:1507.02159 (2015)

4.8 Inter-Dataset Approaches

META: A.k.a methods to relax the need for large dataset sizes

4.8.1 Unsupervised pre-training

4.8.2 Multi-task learning

4.8.3 Transfer learning

See Karparthy 'Large-scale video classification with convolutional neural networks' 2014

A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carls- son. CNN features off-the-shelf: an astounding baseline for recognition

N. Zhang, M. Paluri, M. Ranzato, T. Darrell, and L. Bourdev. Panda: Pose aligned networks for deep attribute modeling. In CVPR, 2014.

B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva. Learning deep features for scene recognition using places database. In NIPS, 2014.

5 Evaluation

What do we need, what do we have, what is best suited so far?

Using unsupervised features for action recognition is still in its beginning but promising. The amount of research that has been put into supervised methods, i.e. 3d convnets two-stream approaches can boost unsupervised methods when done there. Big advantage: no labeling, or less labeling needed when using semi-supervised learning. Amount of video on the internet huge -> potential.

table conventional:

table deep: 3d convolutions [36] KTH : 90.2% conv-layers: 3, total number of layers: , year 2010, max input length: 3d convolutions + LSTM [45] KTH : 92.17% conv-layers: 3, total number of layers: 7+1 (CNN + RNN) , year 2011, max input length: Slow Fusion [7] Sports-1M:60.9% conv-layers: 5, total number of layers: 10 , year 2014, max input length: C3D conv-layers: 8, total number of layers: 15, year: 2015, max input length: 16, UCF101: 85.2%, Sports-1M: 60% LongTermConvolutions on optical flow inputs UCF-101: 82.2, HMDB-51: 59.0

future directions: transfer learning, as done by karpathy, large scale pre-training, as done by varol et. al 2016 multi task learning as done by two stream approach simonyan and zisserman

datasets too small. Citation: karpathy, large-scale classification: "From a practical standpoint, there are currently no video classification benchmarks that match the scale and variety of existing image datasets because videos are significantly more difficult to collect, annotate and store."

datasets too small. citation: simonyan zisserman "Unlike the spatial stream ConvNet, which can be pre-trained on a large still image classification dataset (such as ImageNet), the temporal ConvNet needs to be trained on video data – and the available datasets for video action classification are still rather small".

"Extensions of CNNs to action recognition in video have been proposed in several recent works [6, 12, 13]. Such methods, however, currently show only moderate improvements over earlier methods using hand-crafted video features [5]." Varol Long term temporal convolutions.

Most of the current CNN methods use architectures with 2D convolutions, enabling shift-invariant representations in the image plane. Meanwhile, the invariance to translations in time is also important for action recognition since the beginning and the end of actions is unknown in general. Laptev 2016

References

[1] Jake K. Aggarwal and Michael S. Ryoo. "Human Activity Analysis: A Review". In: *ACM Computing Surveys (CSUR)* 43.3 (2011). 01121, p. 16. URL: http://dl.acm.org/citation.cfm?id=1922653 (visited on 05/23/2016).

[2] Ivan Laptev. "On Space-Time Interest Points". In: *International Journal of Computer Vision* 64 (2-3 2005). 02614, pp. 107–123. URL: http://link.springer.com/article/10.1007/s11263-005-1838-7 (visited on 06/01/2016).

[3] 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*. 02076. IEEE, 2005, pp. 65–72. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1570899 (visited on 05/16/2016).

[4] 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)*. Vol. 1. 15268. IEEE, 2005, pp. 886–893. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1467360 (visited on 07/18/2016).

[5] Ivan Laptev et al. "Learning Realistic Human Actions from Movies". In: *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. 02233. IEEE, 2008, pp. 1–8. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4587756 (visited on 05/25/2016).

[6] Alexander Klaser, Marcin Marsza\lek, and Cordelia Schmid. "A Spatio-Temporal Descriptor Based on 3d-Gradients". In: *BMVC 2008-19th British Machine Vision Conference*. 00929. British Machine Vision Association, 2008, pp. 275–1. URL: https://hal.inria.fr/inria-00514853/ (visited on 10/18/2016).

[7] Andrej Karpathy et al. "Large-Scale Video Classification with Convolutional Neural Networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 00537. 2014, pp. 1725–1732. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Karpathy_Large-scale_Video_Classification_2014_CVPR_paper.html (visited on 05/03/2016).

[8] Aaron F. Bobick and James W. Davis. "The Recognition of Human Movement Using Temporal Templates". In: *IEEE Transactions on pattern analysis and machine intelligence* 23.3 (2001). 02522, pp. 257–267. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=910878 (visited on 01/03/2017).

[9] Gunnar Johansson. "Visual Motion Perception." In: *Scientific American* (1975). 00925. URL: http://psycnet.apa.org/psycinfo/1975-28753-001 (visited on 01/05/2017).

[10] 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*. Vol. 1. 00204. IEEE, 2005, pp. 144–149. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1541250 (visited on 01/03/2017).

[11] Trevor Darrell and Alex Pentland. "Space-Time Gestures". In: *Computer Vision and Pattern Recognition, 1993. Proceedings CVPR'93., 1993 IEEE Computer Society Conference on*. 00473. IEEE, 1993, pp. 335–340. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=341109 (visited on 01/05/2017).

[12] 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*. 00308. Citeseer, 1995, pp. 272–277. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.56.5329&rep=rep1&type=pdf (visited on 01/05/2017).

[13] 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)*. Vol. 1. 00173. IEEE, 2006, pp. 959–968. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1640855 (visited on 01/05/2017).

[14] Nuria Oliver, Eric Horvitz, and Ashutosh Garg. "Layered Representations for Human Activity Recognition". In: *Multimodal Interfaces, 2002. Proceedings. Fourth IEEE International Conference on*. 00335. IEEE, 2002, pp. 3–8. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1166960 (visited on 01/04/2017).

[15] 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*. 00077. IEEE, 2004, pp. 117–117. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1384912 (visited on 01/05/2017).

[16] Peng Dai et al. "Group Interaction Analysis in Dynamic Context". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 38.1 (2008). 00041, pp. 275–282. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4407211 (visited on 01/05/2017).

[17] Shaogang Gong and Tao Xiang. "Recognition of Group Activities Using Dynamic Probabilistic Networks". In: *Computer Vision, 2003. Proceedings. Ninth IEEE International Conference on*. 00268. IEEE, 2003, pp. 742–749. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1238423 (visited on 01/05/2017).

[18] 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* 22.8 (2000). 00750, pp. 852–872. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=868686 (visited on 01/05/2017).

[19] Darnell Moore and Irfan Essa. "Recognizing Multitasked Activities from Video Using Stochastic Context-Free Grammar". In: *AAAI/IAAI*. 00250. 2002, pp. 770–776. URL: http://www.aaai.org/Papers/AAAI/2002/AAAI02-116.pdf (visited on 01/05/2017).

[20] 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*. Vol. 2. 00115. IEEE, 2003, pp. II–626. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1211525 (visited on 01/05/2017).

[21] 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)*. 00073. IEEE, 2006, pp. 107–107. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1640550 (visited on 01/05/2017).

[22] John E. Hopcroft. *Introduction to Automata Theory, Languages and Computation: For VTU, 3/E*. 16985. Pearson Education India, 1979.

[23] James F. Allen. "Maintaining Knowledge about Temporal Intervals". In: *Communications of the ACM* 26.11 (1983). 09208, pp. 832–843. URL: http://dl.acm.org/citation.cfm?id=358434 (visited on 01/05/2017).

[24] James F. Allen and George Ferguson. "Actions and Events in Interval Temporal Logic". In: *Journal of logic and computation* 4.5 (1994). 00793, pp. 531–579. URL: http://logcom.oxfordjournals.org/content/4/5/531.short (visited on 01/05/2017).

[25] 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*. 00135. IEEE, 1998, pp. 898–904. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=698711 (visited on 01/05/2017).

[26] Jeffrey Mark Siskind. "Grounding the Lexical Semantics of Verbs in Visual Perception Using Force Dynamics and Event Logic". In: *Journal of Artificial Intelligence Research* 15 (2001). 00256, pp. 31–90. URL: http://www.academia.edu/download/45992641/Grounding_the_Lexical_Semantics_of_Verbs20160527-28612-cisslg.pdf (visited on 01/05/2017).

[27] 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*. Vol. 4. 00124. IEEE, 2003, pp. 39–39. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4624298 (visited on 01/05/2017).

[28] 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)*. Vol. 2. 00241. IEEE, 2006, pp. 1709–1718. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1640961 (visited on 01/05/2017).

[29] Heng Wang et al. "Action Recognition by Dense Trajectories". In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. 01006. IEEE, 2011, pp. 3169–3176. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5995407 (visited on 05/17/2016).

[30] Heng Wang et al. "Evaluation of Local Spatio-Temporal Features for Action Recognition". In: *BMVC 2009-British Machine Vision Conference*. 00973. BMVA Press, 2009, pp. 124–1. URL: https://hal.inria.fr/inria-00439769/ (visited on 05/17/2016).

[31] Gunnar Farnebäck. "Two-Frame Motion Estimation Based on Polynomial Expansion". In: *Scandinavian Conference on Image Analysis*. 00582. Springer, 2003, pp. 363–370. URL: http://link.springer.com/chapter/10.1007/3-540-45103-X_50 (visited on 11/24/2016).

[32] Navneet Dalal, Bill Triggs, and Cordelia Schmid. "Human Detection Using Oriented Histograms of Flow and Appearance". In: *European Conference on Computer Vision*. 01098. Springer, 2006, pp. 428–441. URL: http://link.springer.com/chapter/10.1007/11744047_33 (visited on 11/25/2016).

[33] Heng Wang and Cordelia Schmid. "Action Recognition with Improved Trajectories". In: *Proceedings of the IEEE International Conference on Computer Vision*. 00636. 2013, pp. 3551–3558. URL: http://www.cv-foundation.org/openaccess/content_iccv_2013/html/Wang_Action_Recognition_with_2013_ICCV_paper.html (visited on 05/17/2016).

[34] Zhuowei Cai et al. "Multi-View Super Vector for Action Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 00059. 2014, pp. 596–603. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Cai_Multi-View_Super_Vector_2014_CVPR_paper.html (visited on 11/30/2016).

[35] 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). 00105. URL: http://arxiv.org/abs/1405.4506 (visited on 05/17/2016).

[36] Shuiwang Ji et al. "3D Convolutional Neural Networks for Human Action Recognition". In: *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 35.1 (2013). 00485, pp. 221–231. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6165309 (visited on 04/27/2016).

[37] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. "Deep Learning". In: *2015* (2016). 00179. URL: http://www.deeplearningbook.org/front_matter.pdf (visited on 01/05/2017).

[38] Yann LeCun et al. "Gradient-Based Learning Applied to Document Recognition". In: *Proceedings of the IEEE* 86.11 (1998). 06106, pp. 2278–2324. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=726791 (visited on 10/31/2016).

[39] Travis Rose et al. "The Trecvid 2008 Event Detection Evaluation". In: *Applications of Computer Vision (WACV), 2009 Workshop on*. 00010. IEEE, 2009, pp. 1–8. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5403089 (visited on 01/06/2017).

[40] Christian Schüldt, Ivan Laptev, and Barbara Caputo. "Recognizing Human Actions: A Local SVM Approach". In: *Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on*. Vol. 3. 00000. IEEE, 2004, pp. 32–36. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1334462 (visited on 06/18/2016).

[41] David G. Lowe. "Distinctive Image Features from Scale-Invariant Keypoints". In: *International journal of computer vision* 60.2 (2004). 37087, pp. 91–110. URL: http://link.springer.com/article/10.1023/B:VISI.0000029664.99615.94 (visited on 10/18/2016).

[42] Ming Yang et al. "Human Action Detection by Boosting Efficient Motion Features". In: *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*. 00030. IEEE, 2009, pp. 522–529. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5457656 (visited on 01/06/2017).

[43] 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*. 00441. IEEE, 2008, pp. 1–8. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4587730 (visited on 05/16/2016).

[44] Hueihan Jhuang et al. "A Biologically Inspired System for Action Recognition". In: *2007 IEEE 11th International Conference on Computer Vision*. 00631. Ieee, 2007, pp. 1–8. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4408988 (visited on 10/30/2016).

[45] Moez Baccouche et al. "Sequential Deep Learning for Human Action Recognition". In: *International Workshop on Human Behavior Understanding*. 00105. Springer, 2011, pp. 29–39. URL: http://link.springer.com/chapter/10.1007/978-3-642-25446-8_4 (visited on 11/02/2016).

[46] 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*. 00018. Springer, 2007, pp. 715–723. URL: http://link.springer.com/chapter/10.1007/978-3-540-72393-6_85 (visited on 11/02/2016).

[47] Sepp Hochreiter and Jürgen Schmidhuber. "Long Short-Term Memory". In: *Neural computation* 9.8 (1997). 02787, pp. 1735–1780. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6795963 (visited on 11/03/2016).

[48] Felix A. Gers, Nicol N. Schraudolph, and Jürgen Schmidhuber. "Learning Precise Timing with LSTM Recurrent Networks". In: *Journal of machine learning research* 3 (Aug 2002). 00264, pp. 115–143. URL: http://www.jmlr.org/papers/v3/gers02a.html (visited on 01/06/2017).

[49] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. "Imagenet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. 08773. 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-w (visited on 01/06/2017).

[50] Jeffrey Dean et al. "Large Scale Distributed Deep Networks". In: *Advances in Neural Information Processing Systems*. 00666. 2012, pp. 1223–1231. URL: http://papers.nips.cc/paper/4687-large-scale-distributed-deep-networks (visited on 11/02/2016).

[51] Du Tran et al. "Learning Spatiotemporal Features With 3D Convolutional Networks". In: *Proceedings of the IEEE International Conference on Computer Vision*. 00081. 2015, pp. 4489–4497. URL: http://www.cv-foundation.org/openaccess/content_iccv_2015/html/Tran_Learning_Spatiotemporal_Features_ICCV_2015_paper.html (visited on 06/21/2016).

[52] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. "UCF101: A Dataset of 101 Human Actions Classes From Videos in The Wild". In: (2012). 00435. URL: http://crcv-web.eecs.ucf.edu/papers/UCF101_CRCV-TR-12-01.pdf (visited on 12/28/2016).

[53] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition." In: *arXiv preprint arXiv:1409.1556* (2014). 02340. URL: https://pdfs.semanticscholar.org/45c6/a85a359be655f459516919138a46ae516621pdf (visited on 11/10/2016).

[54] Yangqing Jia et al. "Caffe: Convolutional Architecture for Fast Feature Embedding". In: *Proceedings of the ACM International Conference on Multimedia*. 01509. ACM, 2014, pp. 675–678. URL: http://dl.acm.org/citation.cfm?id=2654889 (visited on 05/03/2016).

[55] Gül Varol, Ivan Laptev, and Cordelia Schmid. "Long-Term Temporal Convolutions for Action Recognition". In: *arXiv preprint arXiv:1604.04494* (2016). 00001. URL: https://hal.inria.fr/hal-01241518/ (visited on 06/13/2016).

[56] Nitish Srivastava et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” In: *Journal of Machine Learning Research* 15.1 (2014). 01770, pp. 1929–1958. URL: http://www.jmlr.org/papers/volume15/srivastava14a.old/source/srivastava14a.pdf (visited on 01/07/2017).

[57] 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*. 00063. 2014, pp. 2593–2600. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Kantorov_Efficient_Feature_Extraction_2014_CVPR_paper.html (visited on 01/07/2017).

[58] Thomas Brox et al. “High Accuracy Optical Flow Estimation Based on a Theory for Warping”. In: *European Conference on Computer Vision*. 01785. Springer, 2004, pp. 25–36. URL: http://link.springer.com/chapter/10.1007/978-3-540-24673-2_3 (visited on 01/07/2017).

[59] Karen Simonyan and Andrew Zisserman. “Two-Stream Convolutional Networks for Action Recognition in Videos”. In: *Advances in Neural Information Processing Systems*. 00353. 2014, pp. 568–576. URL: http://papers.nips.cc/paper/5353-two-stream-convolutional-networks-for-action-recognition-in-videos (visited on 05/06/2016).

[60] Melvyn A. Goodale and A. David Milner. “Separate Visual Pathways for Perception and Action”. In: *Trends in neurosciences* 15.1 (1992). 04497, pp. 20–25. URL: http://www.sciencedirect.com/science/article/pii/0166223692903448 (visited on 01/07/2017).

[61] Olga Russakovsky et al. “Imagenet Large Scale Visual Recognition Challenge”. In: *International Journal of Computer Vision* 115.3 (2015). 01715, pp. 211–252. URL: http://link.springer.com/article/10.1007/s11263-015-0816-y (visited on 01/08/2017).

[62] 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*. 01219. ACM, 2008, pp. 160–167. URL: http://dl.acm.org/citation.cfm?id=1390177 (visited on 10/21/2016).

[63] 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*. 00135. IEEE, 2015, pp. 4694–4702. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7299101 (visited on 05/16/2016).

[64] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. “Imagenet Classification with Deep Convolutional Neural Networks”. In: *Advances in Neural Information Processing Systems*. 07493. 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-w (visited on 11/02/2016).

[65] Christian Szegedy et al. "Going Deeper with Convolutions". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 01835. 2015, pp. 1–9. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html (visited on 11/14/2016).

[66] Limin Wang et al. "Towards Good Practices for Very Deep Two-Stream Convnets". In: *arXiv preprint arXiv:1507.02159* (2015). 00032. URL: http://arxiv.org/abs/1507.02159 (visited on 06/21/2016).

[67] Christoph Feichtenhofer, Axel Pinz, and Andrew Zisserman. "Convolutional Two-Stream Network Fusion for Video Action Recognition". In: *arXiv preprint arXiv:1604.06573* (2016). 00020. URL: http://arxiv.org/abs/1604.06573 (visited on 11/10/2016).

[68] 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*. 00104. IEEE, 2015, pp. 4305–4314. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=7299059 (visited on 05/16/2016).

[69] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *European Conference on Computer Vision*. 01306. Springer, 2014, pp. 818–833. URL: http://link.springer.com/chapter/10.1007/978-3-319-10590-1_53 (visited on 11/27/2016).

[70] Ken Chatfield et al. "Return of the Devil in the Details: Delving Deep into Convolutional Nets". In: *arXiv preprint arXiv:1405.3531* (2014). 00638. URL: http://arxiv.org/abs/1405.3531 (visited on 11/27/2016).

[71] Christopher Zach, Thomas Pock, and Horst Bischof. "A Duality Based Approach for Realtime TV-L 1 Optical Flow". In: *Joint Pattern Recognition Symposium*. 00685. Springer, 2007, pp. 214–223. URL: http://link.springer.com/chapter/10.1007/978-3-540-74936-3_22 (visited on 11/19/2016).

[72] Shichao Zhao et al. "Pooling the Convolutional Layers in Deep ConvNets for Action Recognition". In: *arXiv preprint arXiv:1511.02126* (2015). 00003. URL: http://arxiv.org/abs/1511.02126 (visited on 05/16/2016).

[73] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. "Unsupervised Learning of Video Representations Using LSTMs". In: *Proceedings of The 32nd International Conference on Machine Learning*. 00116. 2015, pp. 843–852. URL: http://jmlr.org/proceedings/papers/v37/srivastava15.html (visited on 04/25/2016).

[74] 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*. MARMI '16. 00000. New York, NY, USA: ACM, 2016, pp. 3–8. ISBN: 978-1-4503-4362-6. DOI: 10.1145/2927006.2927012. URL: http://doi.acm.org/10.1145/2927006.2927012 (visited on 08/07/2016).

[75] 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*. 01214. ACM, 2009, pp. 609–616. URL: http://dl.acm.org/citation.cfm?id=1553453 (visited on 12/11/2016).

[76] Ishan Misra, C. Lawrence Zitnick, and Martial Hebert. “Shuffle and Learn: Unsupervised Learning Using Temporal Order Verification”. In: (Mar. 28, 2016). 00005. arXiv: 1603.08561 [cs]. URL: http://arxiv.org/abs/1603.08561 (visited on 07/28/2016).

[77] 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* 117 (2013). 00127, pp. 633–659. URL: http://romisatriawahono.net/lecture/rm/survey/computer%20vision/Chaquet%20-%20Human%20Activity%20Recognition%20-%202013.pdf (visited on 04/27/2016).

[78] Tal Hassner. “A Critical Review of Action Recognition Benchmarks”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 00022. IEEE, June 2013, pp. 245–250. ISBN: 978-0-7695-4990-3. DOI: 10.1109/CVPRW.2013.43. URL: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6595882 (visited on 05/29/2016).

[79] Moshe Blank et al. “Actions as Space-Time Shapes”. In: *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*. Vol. 2. 01366. IEEE, 2005, pp. 1395–1402. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1544882 (visited on 06/08/2016).

[80] Soo Min Kang and Richard P. Wildes. “Review of Action Recognition and Detection Methods”. In: (Oct. 21, 2016). 00000. arXiv: 1610.06906 [cs]. URL: http://arxiv.org/abs/1610.06906 (visited on 12/22/2016).

[81] Daniel Weinland, Remi Ronfard, and Edmond Boyer. “Free Viewpoint Action Recognition Using Motion History Volumes”. In: *Computer vision and image understanding* 104.2 (2006). 00710, pp. 249–257. URL: http://www.sciencedirect.com/science/article/pii/S1077314206001081 (visited on 12/22/2016).

[82] *INRIA 4D Repository - IXMAS Action Dataset*. 00000. URL: http://4drepository.inrialpes.fr/public/viewgroup/6 (visited on 12/30/2016).

[83] 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. 00000. June 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587727.

[84] Khurram Soomro and Amir R. Zamir. “Action Recognition in Realistic Sports Videos”. In: *Computer Vision in Sports*. 00015. Springer, 2014, pp. 181–208. URL: http://link.springer.com/chapter/10.1007/978-3-319-09396-3_9 (visited on 12/22/2016).

[85] *Center for Research in Computer Vision at the University of Central Florida - UCF Sports Action Data Set*. 00000. URL: http://crcv.ucf.edu/data/UCF_Sports_Action.php (visited on 12/30/2016).

[86] *Ivan Laptev | Inria Paris - Datasets Download*. 00000. URL: http://www.di.ens.fr/~laptev/download.html (visited on 12/31/2016).

[87] Michael Marszalek, Ivan Laptev, and Cordelia Schmid. "Actions in Context". In: *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. 00762. IEEE, 2009, pp. 2929–2936. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5206557 (visited on 06/06/2016).

[88] *Ivan Laptev | Inria Paris - Hollywood 2 Human Action and Scenes Dataset*. 00000. URL: http://www.di.ens.fr/~laptev/actions/hollywood2/ (visited on 12/31/2016).

[89] 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*. 00742. IEEE, 2009, pp. 1996–2003. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5206744 (visited on 06/01/2016).

[90] *Center for Research in Computer Vision at the University of Central Florida - UCF YouTube Action Datset*. 00000. URL: http://crcv.ucf.edu/data/UCF_YouTube_Action.php (visited on 12/31/2016).

[91] Kishore K. Reddy and Mubarak Shah. "Recognizing 50 Human Action Categories of Web Videos". In: *Machine Vision and Applications* 24.5 (2013). 00243, pp. 971–981. URL: http://link.springer.com/article/10.1007/s00138-012-0450-4 (visited on 10/19/2016).

[92] Hildegard Kuehne et al. "HMDB: A Large Video Database for Human Motion Recognition". In: *Computer Vision (ICCV), 2011 IEEE International Conference on*. 00498. IEEE, 2011, pp. 2556–2563. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6126543 (visited on 04/27/2016).

[93] *Serre Lab - HMDB: A Large Human Motion Database*. 00000. URL: http://serre-lab.clps.brown.edu/resource/hmdb-a-large-human-motion-database/ (visited on 12/28/2016).

[94] Orit Kliper-Gross, Tal Hassner, and Lior Wolf. "The Action Similarity Labeling Challenge". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.3 (2012). 00060, pp. 615–621. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6042884 (visited on 10/19/2016).

[95] 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*. 00359. IEEE, 2009, pp. 104–111. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5459154 (visited on 12/21/2016).

[96] *University of Rochester Activities of Daily Living Dataset*. 00000. URL: http://www.cs.rochester.edu/u/rmessing/uradl/? (visited on 01/01/2017).

[97] Edouard Auvinet et al. "Multiple Cameras Fall Dataset". In: *DIRO-Université de Montréal, Tech. Rep* 1350 (2010). 00045. URL: http://www.iro.umontreal.ca/~labimage/Dataset/technicalReport.pdf (visited on 12/21/2016).

[98] 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*. 00155. IEEE, 2012, pp. 1194–1201. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6247801 (visited on 12/21/2016).

[99] 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*. 00048. 2015, pp. 961–970. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2015/html/Heilbron_ActivityNet_A_Large-Scale_2015_CVPR_paper.html (visited on 01/01/2017).

[100] *Activity Net*. 00125. URL: http://activity-net.org/ (visited on 01/02/2017).