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 (??)

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.

Action Prediction, Anomaly Detection, Event and Action Detection, Person/Pedestrian Detection, Gesture Recognition.o

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 differ-

ent 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

META: Condensed overview and description of conventional Methods in action Recognition using the taxonomy of Aggarwal and Ryoo's fine survey paper. More detailed description of methods using local-features, since these have become the standard approach in action recognition after Aggarwal and Ryoo's overview.

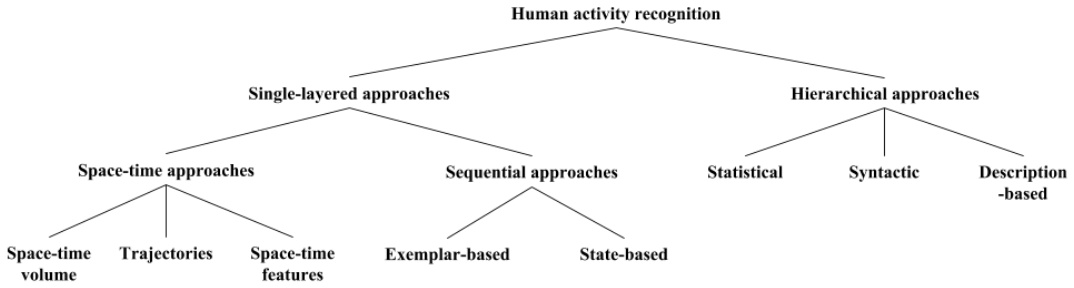


Figure 1: Approach-based taxonomy for conventional methods in human activity recognition as given by Aggarwal and Ryoo[1]

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

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.

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 Overview

2.2 Local Features

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.

2.2.1 Feature extractors

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

2.2.2 Aggregation Methods

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).

2.3 State of the Art Approaches using locaf features

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

Wang et al. [8] 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 [9]. 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 2

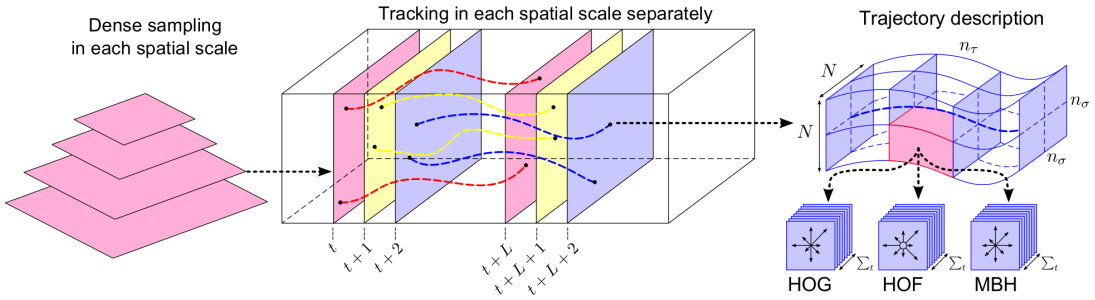


Figure 2: Description of densely extracted trajectories [8]

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 [10] 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 2, 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) [11]

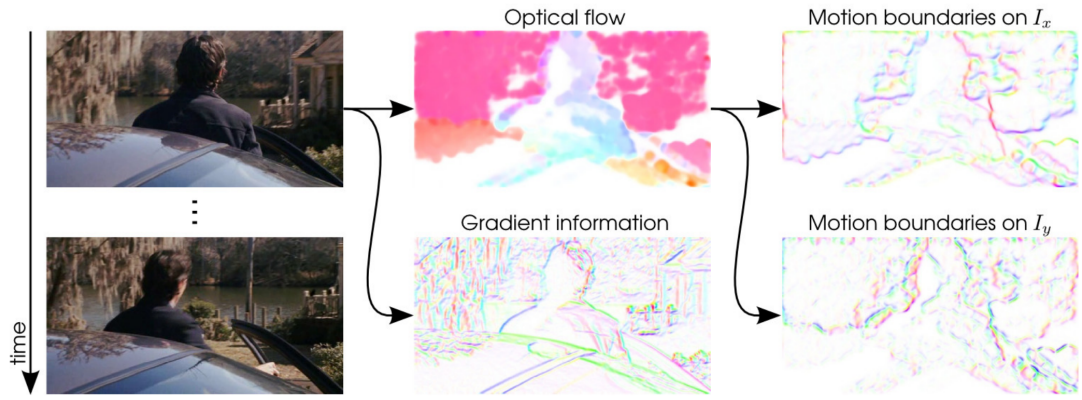


Figure 3: Visualization of the information captured by HOG, HOF and MBH on complete video frames. In each image, the orientation is given by color, the magnitude is given by saturation. [8]

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 3).

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. [8]

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)

[12]

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

[13]

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

[14]

3 Deep Learning Methods in Action Recognition

META: Review of approaches that use Deep Learning methods.

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

3.1 3D-Convolutional Networks

I.e. convolutional methods.

"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

3.1.1 3D Convolutional Neural Networks for Human Action Recognition – Ji et al. (2010/2013)

NOTE:Paper was first published in 2010, the publication of 2013 is more popular however.

In this work [15] the authors porpose 3D convolution for action recognition from video, which processes spatial as well as temporal information in a convolutional layer.

In regular convolutional neural networks 2D convolutions are applied in the convolutional layers to extract features from the feature maps in the previous layer. More formally in the notation of the authors, the value of feature map j in the ith layer at spatial position (x, y) is given by:

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

w_{ijm}^{pq} denotes the value at position (p, q) of the kernel, that performs convolution on feature map m of the previous layer, resulting in feature map j in layer i. P_i and Q_i

denote the dimensions of the kernel in x- and y-direction respectively. Tensor w therefore represents all kernels that produce the feature maps in layer i through convolution.

The two inner sums carry out the convolutional operation on feature map m of the previous layer, which is then combined with the results for the other feature maps by summation over m, added with a bias and fed into a non-linear function ($\tanh(\cdot)$) to result in the value of the current feature map.

Note: The convolutional operation used here is called cross-correlation, which differs from mathematical discrete convolutions in that the kernel is not flipped. This results in a non-commutative operation as described in chapter 9 of cite deep learning book ??

The authors propose an extension of 2D convolutions by using a three dimensional kernel. More formally, in the notation as above, the value of feature map j in the ith layer at position (x, y, z) 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 position (p, q, r), which performs convolution on the mth feature map of the previous layer. R_i denotes the dimension of the kernel in temporal direction.

Based on the 3D convolution, the authors design a neural network architecture, that takes an input of 7 frames of size 60x40.

The network is evaluated as part of an action detection and recognition system on the TRECVID (TREC Video Retrieval Evaluation) data, which consists of surveillance videos recorded at London Gatwick Airport.

The details of the architecture are described below:

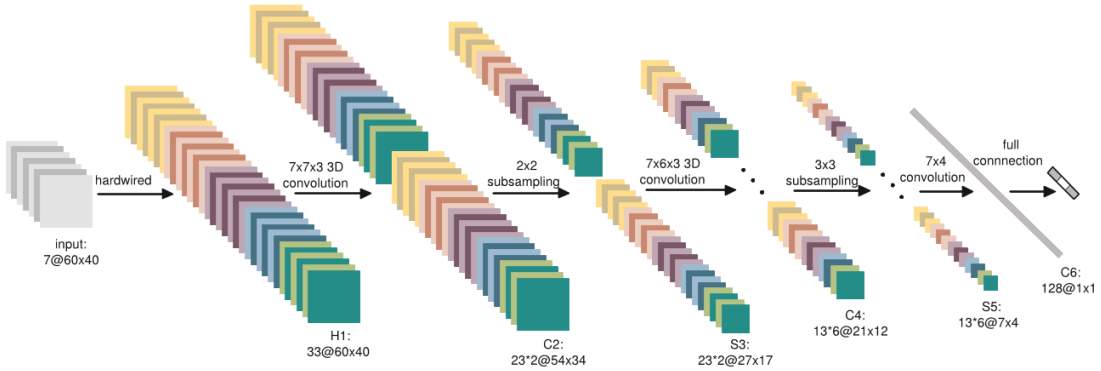


Figure 4: 3D CNN architecture developed for human action recognition on the TRECVID 2008 development dataset [15]

At first, hard wired kernels are applied to the input frames, which extract gray values, gradients along the horizontal and vertical direction in the frames and the optical flow between two consecutive frames. This results in 33 feature maps, organized in 5 different channels: gray, gradient-x, gradient-y, optflow-x and optflow-y.

In the first convolutional layer (C2) two 3D kernels with dimesions 7x7x3 (7x7 in the spatial dimension and 3 in the temporal dimension) are applied at each of the five channels separately. This results in 2x5 channels with a total of 46 feature maps. The first convolutional layer therefore necessitates $2\times5\times7\times7\times3$ (kernel-weights) $+2\times5$ (biases) $= 1480$ trainable parameters.

After subsampling layer S3, three different kernels with size 7x7x3 are applied to each of the 2x5 channels of the previous layer.

The last convolutional layer then performs, after 3x3 subsampling, 2D convolutions to obtain 128 feature maps of dimension 1x1. These feature maps can be seen as a 128 dimensional feature-vector representation of the input.

The resulting feature-vector is classified by a fully connected layer (into three different classes).

The architecture has 295.458 trainable parameters in total, which are initialized randomly and learned by online error back-propagation as in cite lecun 1998 ??. NOTE: Perhaps add comparison to recent image recognition networks here. ??

The authors tried other layouts but conclude that the one described above works best. The performance was evaluated on the TRECVID 2008 development dataset ?? and on the KTH dataset ??.

TRECVID:
The TRECVID 2008 development dataset contains 49-hour annotated surveillance videos from five different cameras on five days. The authors train their model to recognize three different action classes from the dataset: CellToEar, ObjectPut and Pointing. The other classes were used to generate negative training examples.

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 5: Number of samples per class from the TRECVID 2008 development dataset [15]

Since the dataset provides continuous videos with several persons in a real-world scence, the authors 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 other equally sized bounding boxes are sampled at the same location from three frames before and after the intial frame. Between those samples, one frame is omitted which results in a temporal step size of 2.

The contents of these 7 bounding boxes are stacked and taken as input to the 3D CNN architecture for classification.

Figure 6: Example scenes taken by different cameras with the results of the human detection and tracking [15]

In order to compare their 3D CNN model to other techniques, the authors also evaluate:

1. A frame based 2D CNN model
2. Extraction of dense SIFT features from the gray images of the 3D CNN input cube, which are then aggregated using the BoW-Paradigm and classified through a linear SVM
3. Extraction of dense SIFT features from the motion edge history images (MEHI) of the 3D CNN input cube, which are then aggregated as above.

The 3D CNN model outperfomed the other approaches on the TRECVID 2008 development dataset on all classes except Pointing, where the 2D frame-based CNN performed best. The authors note, that the positive example for Pointing are significantly larger than for the other classes and conclude, that their architecture performs best, when little positive examples are present.

KTH:
Additionally, the 3D CNN model was evaluated on the KTH dataset. It achieves an overall accuracy of 90.2%.

DISCUSSION: Authors cite the work of schindler and van gool from 2008 and state, that 5-7 frames are enough for action recognition. Schindler and van gool: 5-7 frames are enough to recognize SIMPLE actions.

3.1.2 Sequential Deep Learning for Human Action Recognition – Baccouche et al. (2011)

Baccouche et al. [16] identify two issues with the previous approaches for extending convolutional neural networks to the video domain, i.e. 3D convolutions as in [15] and [17]:

1. They rely on hand crafted inputs (hard wired preprocessing of the data to produce image gradients or 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 consisting of:

1. A convolutional spatio-temporal feature extractor network, based on 3D convolutions.
2. A recurrent neural network classifier, that incorporates LSTM cells [18], to classify the entire sequence of previously extracted spatio-temporal features.

The architecture is trained and evaluated on the KTH dataset.

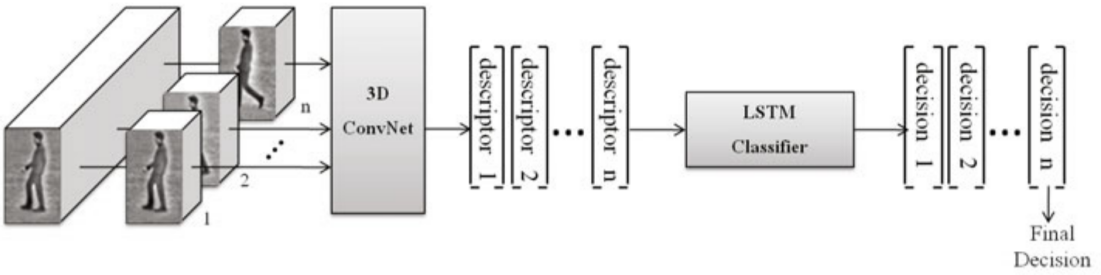


Figure 7: Overview of the two-steps architecture consisting of a 3D ConvNet and a RNN classifier. [16]

The ConvNet architecture of this approach is depicted in figure 7. It is comparable to the one of Ji et al. [15] in that it incorporates 3D convolutions, but works on raw input pixels instead of hand-crafted inputs.

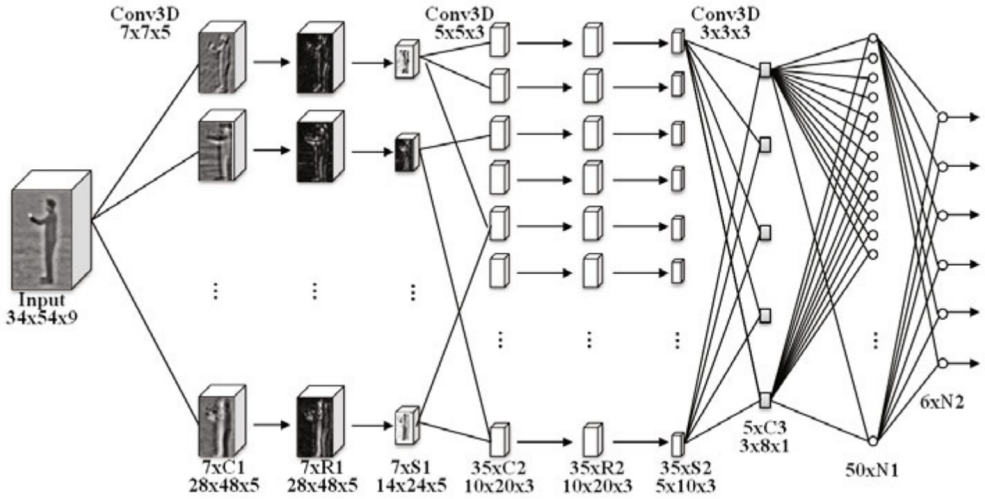


Figure 8: 3D ConvNet architecture for the extraction of spatio-temporal features [16]

Input to the network is formed by stacking 9 succesive frames of spatial size 34x54, which results in an input cube of dimension 34x54x9.

The configuration of it's layers is as follows:

1. First convolutional layers C1 computes 7 feature maps, by convolving a 3D 7x7x5 kernel for each feature map with the input cube.
2. Layer R1 and S1 perform rectification (building of the absolute value) and subsampling with a spatial factor of 2 respectively.
3. Second convolutional layer C2 computes 35 feature maps. Each of the 7 feature maps in layer S1 is connected to two different feature maps in C2 (resulting in 14 feature maps) and each pair of different feature maps of S1 is connected to one feature map in C2 (resulting in additional 21 feature maps, leaving a total of 35 feature maps).
4. Layer R1 and S2 are equivalent to R1 and S1.
5. Third (and last) convolutional layer C3 computes five features maps, which are fully connected to all feature maps in previous layer S2 by 3x3x3 kernels. These five feature maps have dimension 3x8x1, rendering the raw input encoded into a 120 dimensional feature vector.
6. For training the individual ConvNet model, the 120D feature vector is finally fed into two fully connected layers with 6 output neurons (one for each class of the KTH dataset).

The ConvNet model embeds a total of 17,169 trainable parameters, which is about 15 times less than the number of parameters trained by Ji et al. [15] (295,458 parameters).

The authors use the same training algorithm as Ji et al., which was proposed by cite lecun 1998 ??: online Backpropagation with momentum adapted to weight-sharing.

In order to take the temporal evolution of movements in a video into account, the authors propose to train a recurrent neural network with one hidden layer of LSTM cells for classifying sequences of previously extracted CNN feature-vector representations.

Several configurations were tested and 50 LSTM cells in the hidden layer were found to be a good compromise between training time and performance.

The output of the third convolutional layer C3 of the ConvNet architecture is fed into the recurrent neural network as input at each time step.

The LSTM cells in the hidden layer are fully connected with the input and have recurrent connections among each other.

The output layer is connected to the outputs of the LSTM cells.

The RNN is trained as part of the two-step architecture using online backpropagation through time with momentum.

The architecture is evaluated on the KTH dataset. To build the inputs to the network, the following simple pre-processing steps are applied: down-sampling by a factor of 2 horizontally and vertically, extraction of the bounding boxes around persons, application of 3D Local Contrast Normalization on a 7x7x7 neigbourhoood.

The authors find the 3D ConvNet model to yield a recognition rate of 91.04%, when the classification is done by majority voting over several short sub-sequences of the test-video (without using the LSTM classifier). Which is comparable to other deep learning approaches at that time and almost the same result as obtained by Ji et al. [15] (90.2%), although the model requires about 15 times less paramerters.

When using the LSTM classifier network, performance increases of about 3%. The combined two-steps architecture consistently outperformed other deep models at that time.

Evaluation scheme: five best results averages over 30 trials.

3.1.3 Large-scale Video Classification with Convolutional Neural Networks – Karpathy et al. (2014)

The author's contribution in [7] is four-fold:

1. They gather the Sports-1M dataset containing a collection of 1 million automatically annotated sports videos from YouTube.
2. They study multiple approaches for extending convolutional neural networks to incorporate spatio-temporal information.

3. They propose a multiresolution convolutional architecture with the goal of speeding up the training 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 significant increase in performance towards training the network on UCF-101 alone (transfer learning).

Here, the architectures will be reviewed (contributions 2. and 3.), for a description and review of the Sports-1M dataset and the transfer learning paradigm see section 4 of this work.

To obtain fixed-sized inputs for their architectures, the authors interpret an entire video as a set of short, fixed-sized video clips.

The authors first implement a baseline CNN architecture to classify videos based on a single frame. The model is similar to the winning network of the 2012 ImageNet Classification Challenge cite ??, but receives slightly smaller inputs (here 170x170x3 against 224x224x3 in the ImageNet model).

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 9.

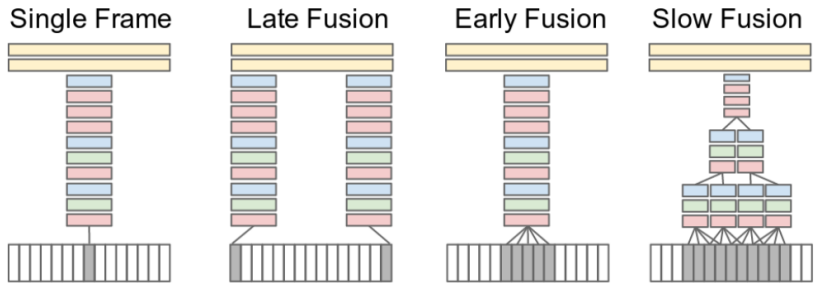


Figure 9: Different methods for fusing temporal with spatial information in convolutional neural network architectures. Red, green and blue denotes convolutional, normalization and pooling-layers. Grey colored inputs are single video frames. [7]

Early Fusion:
Temporal information is incorporated in the network on the pixel level by extending the convolutional kernel in the first layer to be of dimension $11 \times 11 \times 3 \times T$, where T is the temporal extend (here used: $T = 10$).

Late Fusion:
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 processes more spatio-temporal information of the input frames. These kinds of convolutions have also been applied in [15] and baccouche ??. The first convolutional layer uses a temporal extend of $T = 4$, while the second convolutional layer uses a temporal extend of $T = 2$, which allows the third layer to access the information of all 10 input frames.

Since the training time of a network heavily influences the amount of experiments that can be conducted on different hyperparameter settings, it is of great interest to reduce the training time for neural networks (for CNNs: in the order of weeks) while maintaining the accuracy.

The authors propose a multiresolution architecture as shown in figure 10.

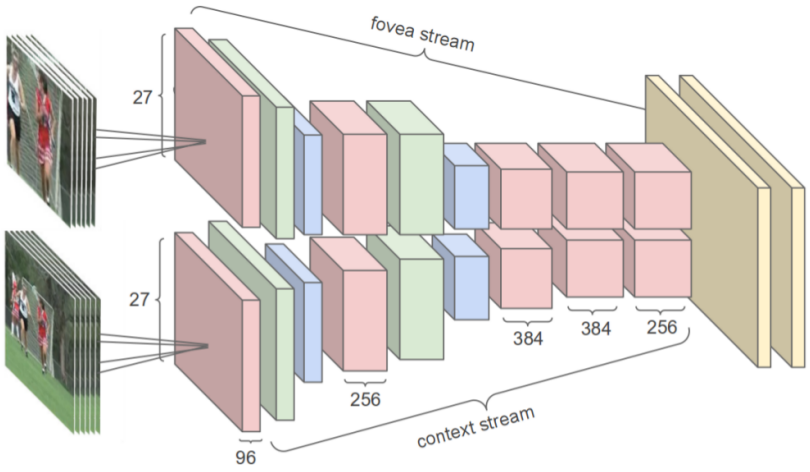


Figure 10: Multiresolution CNN architecture [7]

The overall multi-resolution network processes videos of size 178x178 pixel as input.

The context stream receives a downsampled version of the input frames, which has half the size of the original input (89x89 pixel). The fovea stream works on just the 89x89 pixel sized center of the original input frames.

Both streams are implemented as the same CNN architecture.

The outputs of both streams are fed into the first of two fully connected layers, where their information is merged and a class prediction is calculated.

The multiresolution architecture recquires half the input dimensionality compared to processing the raw 178x178 pixel input video in one stream.

Thereby the authors achieved a reduction in training time by a factor of 2-4.

The authors use downpour stochastic gradient descent cite ?? for training the networks on the Sports-1M dataset using a computing cluster. 70% of the dataset were used as training data, 10% as the validation set and 20% as the test set.

At testing 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 (with different crops and flips) and the result is averaged to produce a robust class prediction. The video-level predictions are computed from the clip-level predictions simply by averaging.

In addition to comparing different CNN architectures, the authors also implement a feature-histogram baseline approach by extracting multiple kinds of hand-crafted features from each video and aggregating them according to the bag-of-words paradigm. The resulting feature vector is classified using multilayer neural network, with rectified linear activation units.

The performance of the studied architectures 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 fraction of test samples, that had at least one of their class labels included in the top k predictions. [7]

The authors find their deep models to consistently outperform the hand-crafted feature based baseline.

Compared to each other, the deep models performed similarly well, despite their different convolutional architectures. The slow fusion model however performed best, by a small margin.

According to the authors interpretation, the "variation among different CNN architectures turns out to be 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.

"we find that a single-frame model already displays very strong performance, suggesting that local motion cues may not be critically important, even for a dynamic dataset such as Sports. An alternative theory is that more careful treatment of camera motion may be necessary (for example by extracting features in the local coordinate system of a tracked point, as seen in [25])."

3.1.4 Learning Spatiotemporal Features with 3D Convolutional Networks – Tran et al. (2015)

Tran et al. [19] propose to create a generic spatio-temporal feature extractor, by training a very deep 3D convolutional network on a large-scale action recognition dataset.

The activations of the network's last fully connected layer given an input video-clip build the descriptive feature-vector.

The authors evaluate their model and show that the extracted features, which they call *C3D*, are generic enough to be reused in other vision tasks without requiring task-specific fine tuning of the model.

?? Their model processes full video frames and does not require any pre-processing. It is therefore easily scalable to large datasets.

Reviewing the evaluation of several fusion methods for temporal information in CNNs made by Karpathy et al. [7], the authors conclude that 3D convolutions are best suited for preserving the temporal information in the processing stages of deep CNNs, because they result in an output volume keeping a temporal dimension.

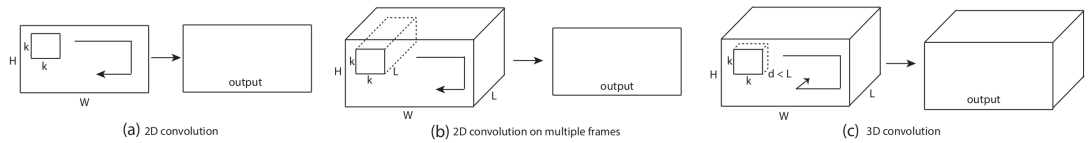


Figure 11: Dimensionality of convolutions: a) 2D convolution on a single image results in a 2D feature map, b) 2D convolution on multiple frames, interpreted as different channels, results in a 2D feature map, c) 3D convolution on multiple frames results in a 3D feature map preserving the temporal dimension [7]

The authors search for the best architecture with 3D convolutions by experimenting on the medium-scale dataset UCF-101, because using a large-scale dataset would be too time-consuming. Recent findings regarding 2D convolutional neural networks suggest that 3x3 convolution kernels yield best results in deep architectures [20]. The authors

therefore fix the spatial kernel size in their 3D convolutions to 3x3 and vary the temporal depth of the kernel to find the best architecture.

All trained network versions take inputs of fixed-size 128x171x16 (frame width x frame height x number of frames) in three channels (RGB).

The evaluation networks are designed as follows:

- 5 convolutional layers with 64, 128, 256, 256 and 256 filters from layer 1 to 5.
- Each convolutional layer is followed by a max-pooling layer with filter size 2x2x2 (except for the first layer, which has a filter size of 1x2x2 to not collapse the temporal information too early).
- Two fully connected layers with 2048 neuraons each and a softmax layer in the end to estimate action labels.

For finding the best architecture only the convolutional kernel depth d is varied by following two methods:

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).

Results of the evaluation of different kernel temporal depth on UCF-101 (split 1) are shown in figure 12:

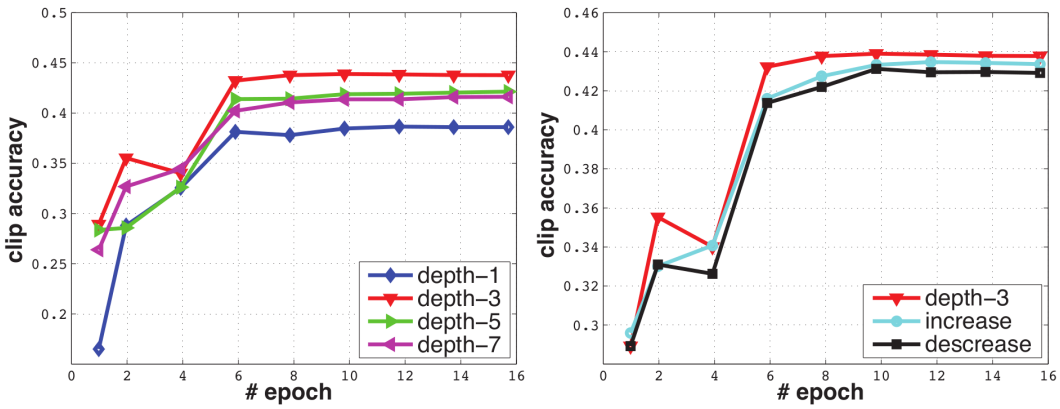


Figure 12: Clip accuracy for different temporal kernel depth over training epochs [19]

The results show, that temporal depth 3 performs best among the homogeneous and also among the varying networks. After also testing 5x5 spatial kernel size, which yielded

the same result, the authors conclude that 3x3x3 kernels are the best choice in 3D convolutional networks.

Using this result, the authors design the architecture for extracting C3D features with 3x3x3 convolutional kernels as follows: 8 convolutional layers, 5 pooling layers, two fully connected layers and a softmax output layer as can be seen in figure 13.

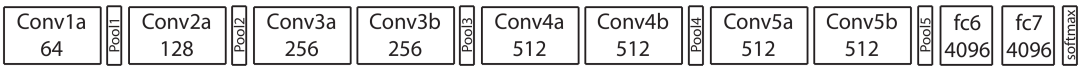


Figure 13: C3D architecture. Number of filters is given in each box. [19]

Training
The network is trained on the large-scale dataset Sports-1M to learn spatio-temporal features.

Five two second long clips are extracted from every training video and resized to the input dimensions of 128x171 pixel.

The training inputs are randomly extracted video volumes from these clips of size 112x112x16 (using temporal and spatial jittering).

Training is conducted with stochastic gradient descent with batch size of 30.

Testing
A single center crop of a testing-video is passed through the network to obtain a clip-preditction.

For obtaining video-predictions, the predictions for 10 clips which were extracted randomly from the test-video are being averaged.

The network trained from scratch yields top-5 accuracy at action recognition on the Sports-1M dataset of 84.4%. The authors also try to boost the performance by pre-training the model on an internal dataset (called I380K), which results in a top-5 accuracy of 85.5%.

The authors propose a standardized method to extract C3D features using their model and evaluate them on several video analysis tasks:

1. Action recognition on UCF-101 dataset:
   A single C3D net obtains an accuracy of 82.3%. Three networks combined obtain 85.2%. In order to outperform the results of Simonyan and Zisserman [21] or Ng et al. [22], C3D has to be combined with the improved dense trajecotries approach [12], which results in an accuracy of 90.4%.
2. Action Similarity Labeling (ASLAN [23]):
   C3D features classified with a linear SVM outperforms the state-of-the-art method [24] by 9.6%.

3. Scene Recognition on YUPENN and Maryland benchmark:
   Using C3D features outperforms the state-of-the-art method on both benchmarks [25].
4. Object Recognition on the Egocentric datset [26]:
   C3D features outperfrom the method of [26] but perform worse than an ImageNet baseline.

Additionally to good results on various video analysis datasets, the authors note several advantages of using their features:

1. They are simple to compute by making the pre-trained 3D CNN model available
2. The features are compact and discriminative, which makes them scalable to large-scale analysis tasks.
3. They are efficient to compute, because of the fast forward processing in neural networks.

3.1.5 Long-term Temporal Convolutions for Action Recognition – Varol et al. (2016)

Varol, Laptev, and Schmid [27] address, similar to Baccouche et al. [16], the common drawback of recent CNN extensions in action recognition, that class labels are learned from very short video subsequences only, i.e. a temporal extend of only 1-16 input frames can be processed by the architectures at a time.[15][7][19]

Instead of using a recurrent neural network for classifying convolutionally extracted spatio-temporal features, which takes the temporal evolution of features into account (as done by Baccouche et al. [16]), in this publication the authors study the effects of increasing the number of input frames of a 3D convolutional architecture on the performance in action recognition.

The authors call their approach long-term temporal convolutions. Their contribution in this work is two-fold:

1. Systematical evaluation of the influence of the temporal extend, i.e. the number of input frames $T = \{20, 40, 60, 80, 100\}$, on the performance of a 3D CNN architectures for video action recognition.
2. Confirming the importance of high-quality optical flow inputs, in order to learn accurate video features with a 3D CNN architecture for human action recognition.

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 58x58 or 71x71 pixels.

The authors' 3D CNN architecture is shown in figure 14.

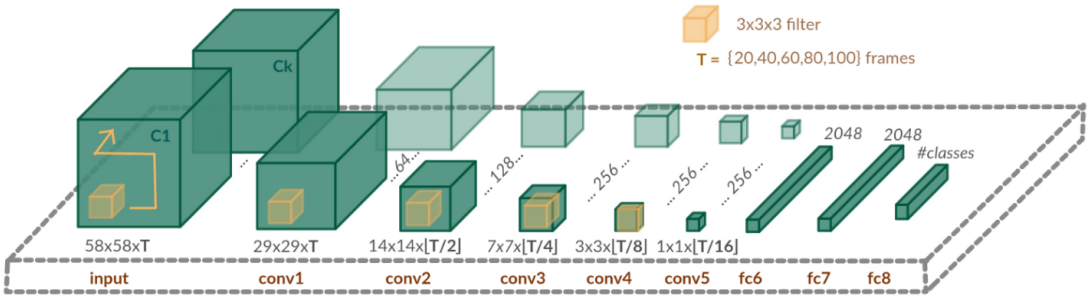


Figure 14: 3D CNN architecture for evaluating the influence of temporal extend on action recognition performance. [27]

Architectural details:

- 5 space-time convolutional layers with 64, 128, 256, 256 and 256 feature maps (also called 3D convolutions).
- 3x3x3 space-time convolutional kernels in every convolutional layer.
- After each convolutional layer: One layer of rectified linear units (ReLU) and one layer of max-pooling with filter size 2x2x2 except in the first layer where it is 2x2x1.
- 3 fully connected layers as classifier with sizes 2048, 2048 and number of classes. The fully connected layers also use rectified linear units as activation functions and a softmax layer at the end of the network.

As a baseline approach the authors first implement their architecture with inputs of size 112x112x16, beacause it can be directly compared to the work of Tran et al. [19]. The authors initially study the performance between the 16 frame baseline and a 60 frame network, which takes inputs of size 58x58x60 (spatial size has to be decreased to remain computationally tractable).

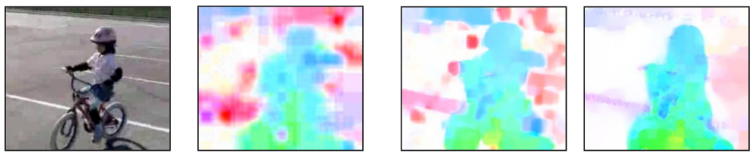
In order to then systematically investigate the influence of temporal extend on the action recognition performance, the authors implement their architecture with different numbers of input frames $T = \{20, 40, 60, 80, 100\}$ and different spatial resolutions $58 \times 58, 71 \times 71$.

Additionally the influence of using optical flow inputs on the performance, specifically the influence of different sources of optical flow, is evaluated. These namely are:

1. MPEG flow, which directly can be obtained from the video encoding. It is a fast alternative to regular optical flow estimators, but has low spatial resolution and is not available for all video frames.
2. Farneback optical flow estimator cite ??, which is fast, but calculates noisy flow fields.

3. Brox optical flow estimator cite ??, which generates highest quality optical fields but is slower than the other two methods.

Example flow estimations for all three methods are shown in figure 15.



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 15: Add caption here

The networks are trained on the UCF-101 and HMDB-51 dataset using stochastic gradient descent with negative log-likelihood criterion.

The authors generate training inputs by randomly sampling video subsequences with the desired spatial and temporal dimensions from the input videos, which in general have a higher resolution and are temporally longer than needed. The authors call this form of pre-processing *random clipping*.

An alternative method for creating suitable network inputs during training is called *multiscale cropping*. Smaller input volumes 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 rendomly selected factors for frame width and height from $\{1.0, 0.875, 0.75, 0.66\}$. Finally the input is flipped with a probability of 50%.

Dropout is used during training independent of the selected cropping method.

Details of the testing-procedure are:

1. The video under test is divided into sequences of length T with temporal stride 4, each called a clip.
2. For each video clip, 10 crops are being classified: The 4 corners, the center and their horizontal flips.
3. The overall video classification is obtained by averaging over crop scores and clip scores.

Two evaluation metrics are used to compare the networks:

1. Clip-accuracy: Each clip is assigned the class according to the highest softmax-score and the number of correctly classified clips is measured.
2. Video-accuracy (i.e. the standard evaluation protocol): The per-clip softmax scores are averaged and the maximum value defines the class label for the video. The

authors report their final resutlts by taking the mean over the results on the three test splits of the datasets.

Optical flow results:
The influence of optical flow quality on the action recognition performance is depicted in the table on the right of figure 15.

For this evaluation a network with 60 input-frames was trained on UCF-101 split 1 from scratch.

The authors find, that high quality optical flow, i.e. Brox flow, can boost the action recognition performance by nearly 20%.

Even using low-quality MPEG-flow outperforms regular RGB inputs.

The authors therefore conclude, that using high-quality optical flow inputs is critical for learning competitive human action features from video.

Data-augmentation and dropout results:
Results for the used data augmentation methods *Random Clipping*, *Multiscale Cropping* and *Dropout* are shown below in table 3. Best performance is obtained when combining all three methods.

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 data-augmentation methods and dropout on a 60 input-frame network, trained on UCF-101 (split 1) from scratch using Brox flow as input modality [27]

Comparison 16 frame against 60 frame input:
The 16 frame network is equivalent to the C3D architecture in [19] and therefore enables direct comparison to the long-term temporal convolution approach of this work (the 60 frame network).

The evaluation results on the UCF-101 dataset (split 1) for RGB and optical flow inputs as well as different data augmentations and dropout are shown in table 4 below.

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: Performance for 16 frame against 60 frame inputs, evaluated on UCF-101 (split 1) [27]

The results show that long-term temporal convolutions in the 60 frame network persistently outperform the 16 frame counterpart.

The authors note, that the complexity of the networks, i.e. the number of trainable parameters, is similar.

Results for the clip accuracy are more prominent, because video evaluation already averages and aggregates the available information over the whole video.

The authors repeat the comparison on the HMDB-51 (split 1) dataset and additionally evaluate the effects of pre-training the networks on the UCF-101 dataset, because HMDB-51 is significantly smaller than UCF-101.

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: Performance for 16 frame against 60 frame inputs, evaluated on HMDB-51 (split 1). Flow input, random clipping, multiscale cropping and 0.9 dropout are used. [27]

The results again indicate, that using long-term temporal convolutions leads to significant increase in action recognition accuracy.

Pre-training the networks yields significant improvement over training it from scratch.

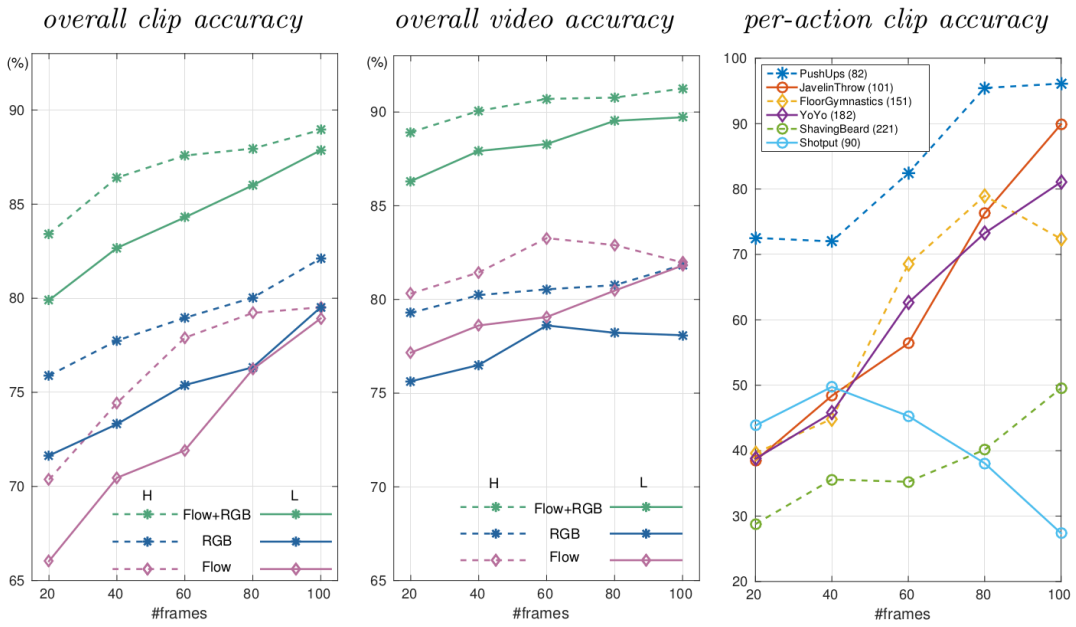


Figure 16: <+caption text+>

3.1.6 Summary and Comparison

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 Note: No need for action detection???

3.2 Multiple Stream Networks

The most successfull architecture at 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)

The authors propose a novel architecture for action recognition with two separate recognition streams (spatial- and temporal-stream) which are combined by late fusion.

The authors evaluate two different fusion methods: building the average of both network's outputs and training a linear multi-class SVM on stacked L_2-normalised softmax scores.

This approach is motivated by the two-streams hypothesis cite (??), according to which the human visual cortex contains two paths: the ventral stream for object recognition and the dorsal stream for recognising motion.

Both streams are implemented as deep CNNs, with the rectification activation function for all hidden units.

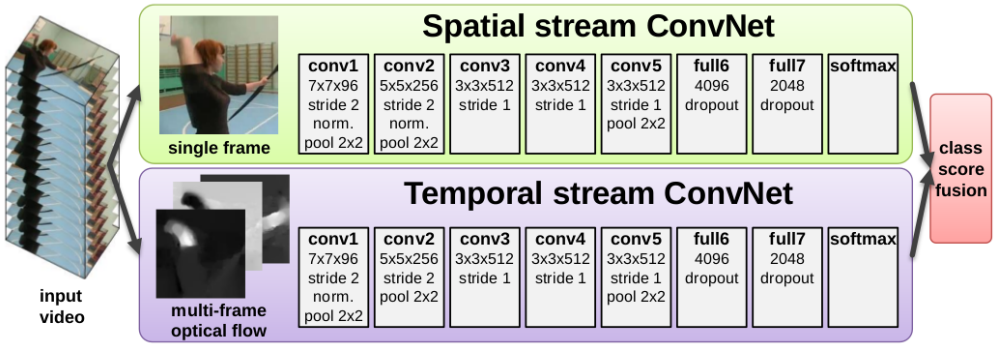


Figure 17: Two-stream architecture for video classification, depicting the spatial- and temporal-stream with implementation details as deep Convolutional Neural Network [21]

Spatial stream: Takes single video frames as input. Performs action recognition from still images and is fairly competitive on its own. Basically an image recognition architecture.

Advantage: Can be pre-trained using large amount of image data, here from the ImageNet challenge dataset cite (??).

Spatial part of a video, i.e. the individual static frames, convey information about the objects and persons in the scene.

Temporal stream: is trained to recognize actions from motions given in the form of dense optical flow.

The temporal part of a video, i.e. the change between frames, conveys information about the movement of the observer (camera) and the movement of objects in the scene.

The second normalisation layer was removed from the the temporal stream network in order to reduce memory usage.

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 $u, v \in \mathbb{N}$, $1 \leq u \leq w$ and $1 \leq v \leq h$ where w and h are the width and height of the video frames.

The horizontal and vertical components $d_t^x(u, v)$, $d_t^y(u, v)$ 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 $1 \leq k \leq L$:

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

Trajectory Stacking:
Instead of sampling at fixed locations in each frame, this methods samples the dense optical flow field along the 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 input volume can then be constructed by sampling the horizontal and vertical optical flow components along these trajectories.

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

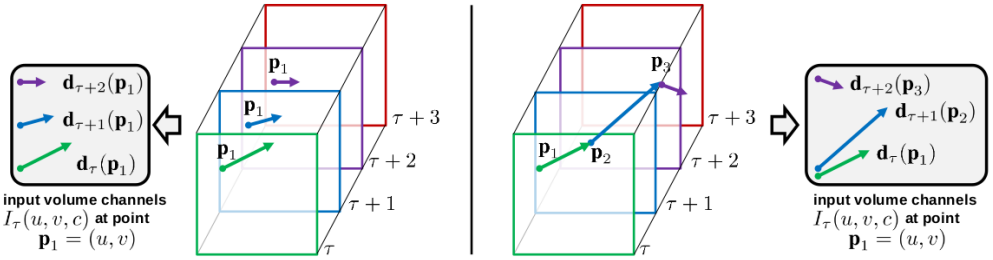


Figure 18: Construction of input volumes from multi-frame optical flow. Left: Optical Flow Stacking. Right: Trajectory Stacking [21]

Since the Convolutional Networks require fixed sized inputs, the authors sample a $224 \times 224 \times 2L$ subvolume from $I_\tau \in \mathbb{R}^{w \times h \times 2L}$ and use it as the temporal networks input.

By using optical flow, the authors explicitly incorporate a representation of motion in their action recognition architecture.

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 $L/2$ around frame τ in both directions.

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 $\mathbf{d}_t$.

An advantage of separating the spatial and the temporal stream is the possibility of pre-training the spatial net with large pre-existing image datasets (here ImageNet ILSVRC-2012 ??).

For training the temporal stream network with video-data, the authors address the general problem of video action recognition, that the existing datasets are too small in comparison to their image-dataset counterparts.

The authors use, according to the multi-task learning paradigm [28], the UCF-101 and the HMDB-51 dataset simultaneously for training the temporal stream network (see chapter ??).

By training a network on several tasks (here UCF-101 classification and HMDB-51 classification), the network learns more general video representations, since the second task acts as regulariser and more data can be utilized.

The authors implement this 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. For optical flow computation the authors use a fast implementation (0.06s per pair of frames) from the OpenCV toolbox. Despite it's speed, on-the-fly computation of optical-flow would be a bottleneck and is therefore pre-computed and stored for the complete datasets.

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.

The authors build their final design of the two-stream architecture by evaluating different setups for the spatial and temporal stream network on their own using UCF-101 (split 1).

Besides using two different dropout rate (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, because the network needs to be trained from scratch. 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.

Spatial-stream network: The authors decided on pre-trained the network on ILSVRC and fine-tuning only the last layer.

Temporal-stream network: Mean subtraction and stacking multiple optical flows 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 forwars 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.

(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) [21]

After having found the optimal configurations for the individual temporal-stream and spatial-stream networks, the authors evaluate different fusion methods (averaging and SVM) and find that fusion by SVM performs best. 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 three splits on UCF-101 and HMDB-51 [21]

The results in table 7 show, that fusion by SVM works best.

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

Ng et al. [22] 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 [16] and [27], 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 19):

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.

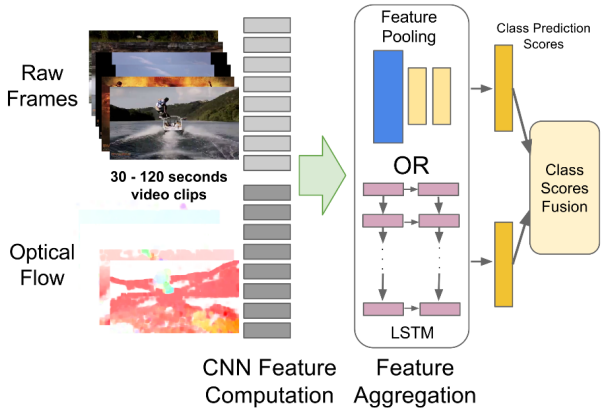


Figure 19: Overview of the appraoch taken by Ng et al. [22]

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 [29]
2. GoogLeNet [30]

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 **??**).

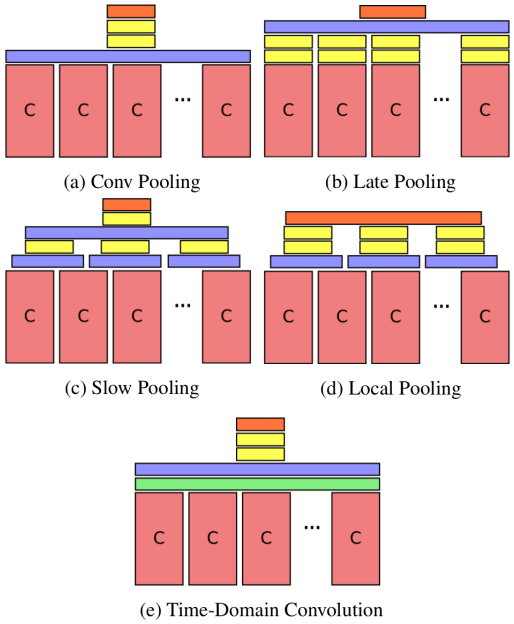


Figure 20: Feature pooling architectures with: Max-pooling layers (blue), time-domain-convolution layer (green), fully-connected layers (yellow), softmax layers (orange). [22]

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 [18] architecture with 5 layers containing 512 memory cells each is used to aggregate sequences of frame-level CNN activations.

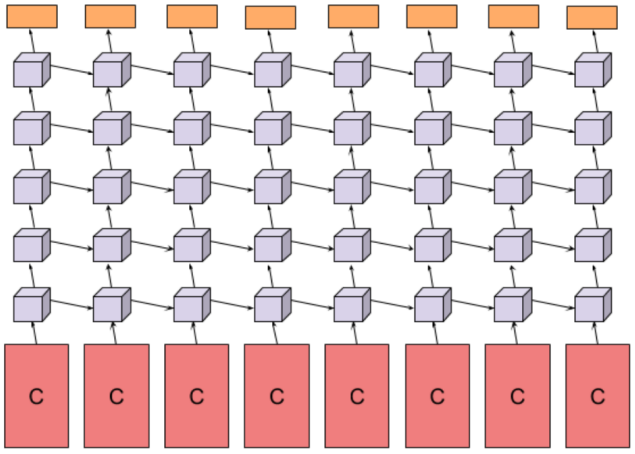


Figure 21: LSTM Architecture. Five layers of LSTM cells take the final CNN activations as inputs. The video class is predicted at every timestep by a softmax output layer. [22]

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 [21] 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. [22]

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 [21].

?? read conclusion!!

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

[31]

Wang et al. [31] aim at improving the Two-Stream ConvNet approach for action recognition[21] by leveraging the advances of very deep ConvNet architectures in image clas-

sification.

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[30] and VGGNet[20]. 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 [21] 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 [21]. 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 [21].

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 over-fitting.

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

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

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 [21], which isthe original two-stream approach. [31]

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)

[32]

Feichtenhofer, Pinz, and Zisserman [32] 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 [21] 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 22 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) [32]

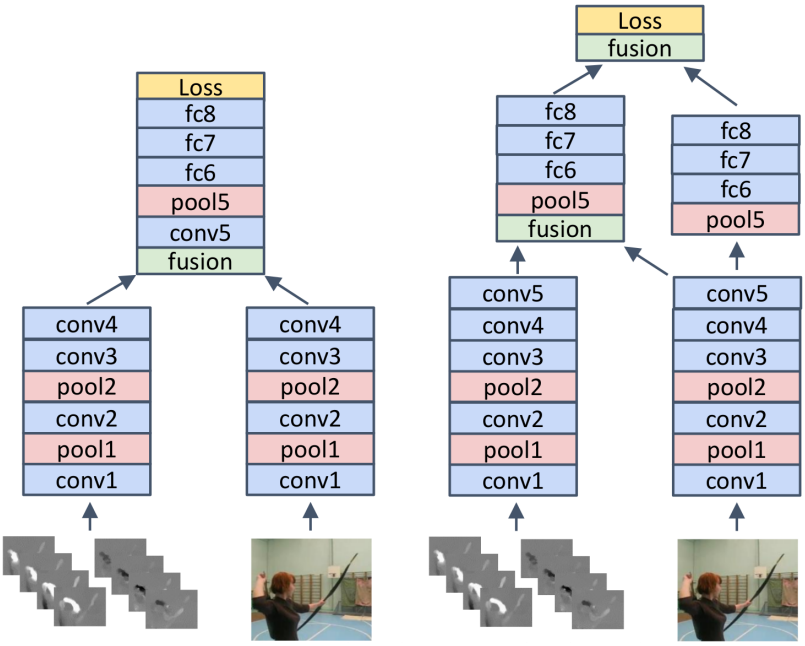


Figure 22: Placement examples of fusion layers for early combination of the spatial and temporal stream in two-stream architectures. (*left*) Only the merged stream is kept. (*right*) The spatial stream is kept after the first fusion layer and fused into it before the final layer [32]

The authors evaluate the spatial fusion methods in a two-stream architecture as implemented in the original approach [21]. 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 22 (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.

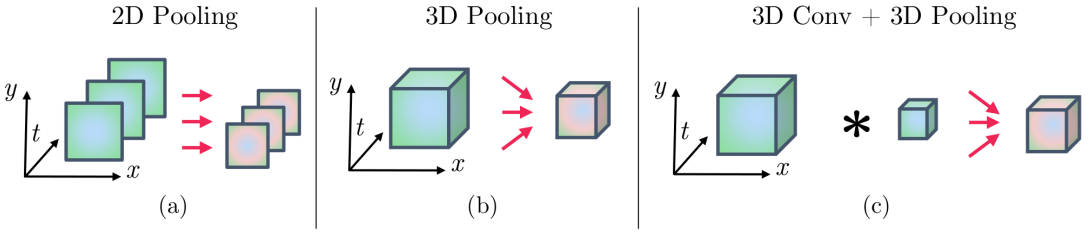


Figure 23: Methods for pooling feature maps along the temporal dimension. [32]

(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 [33] propose a novel method of building video representations by combining the advantageous porperties of hand-crafted features as in *Dense Trajectories* [12] with deeply learned features as in *Two-Stream Convolutional Networks* [21].

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 [12].
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 [12], 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 [34] 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 [33]

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.

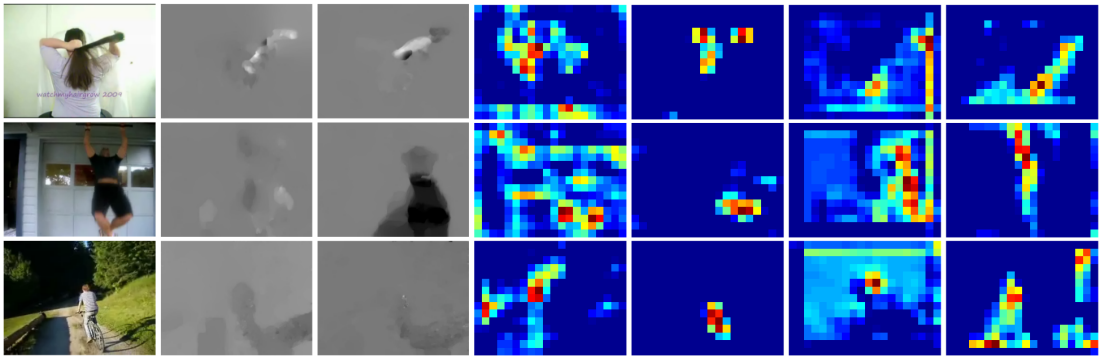
Both normalization methods were evaluated experimentally and the authors found, that combining the resulting video representations from both normalization methods by late fusion yields the best results.

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 24: 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) [33]

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 [35] 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 [36] (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 [21].

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 [21]). 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) [12] and two-stream ConvNets [21]. [33]

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

[37]

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)

[38]

Srivastava, Mansimov, and Salakhudinov [38] 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 25.

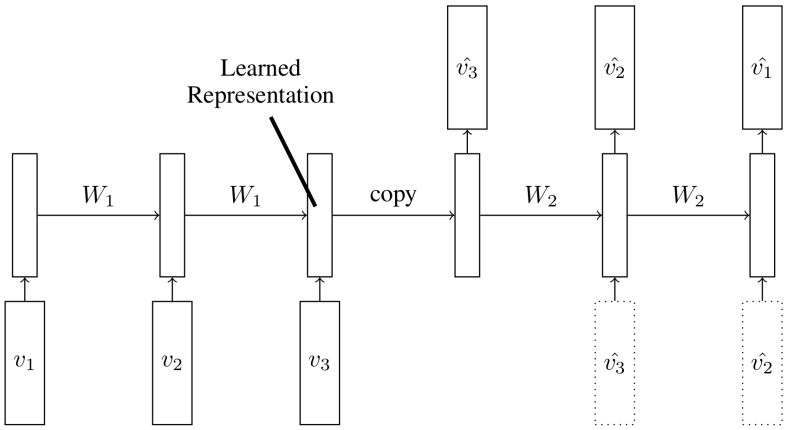


Figure 25: Auto-encoder model [38]

$W_i (i \in \mathbb{N})$ 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 26. This decoder LSTM network can again be unconditioned or conditional.

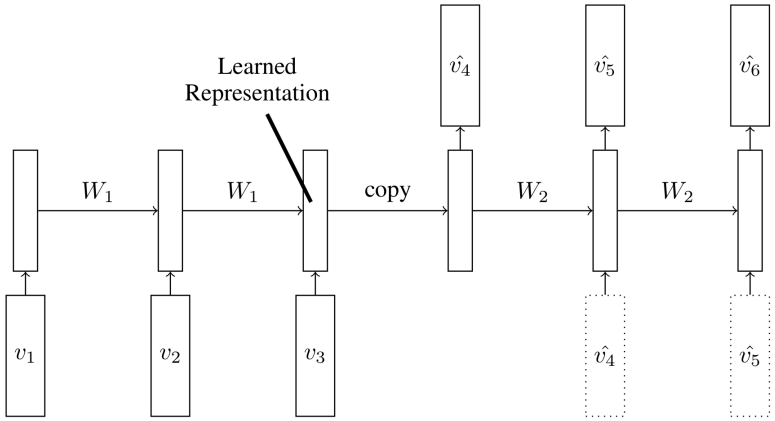


Figure 26: Future-predictor model [38]

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

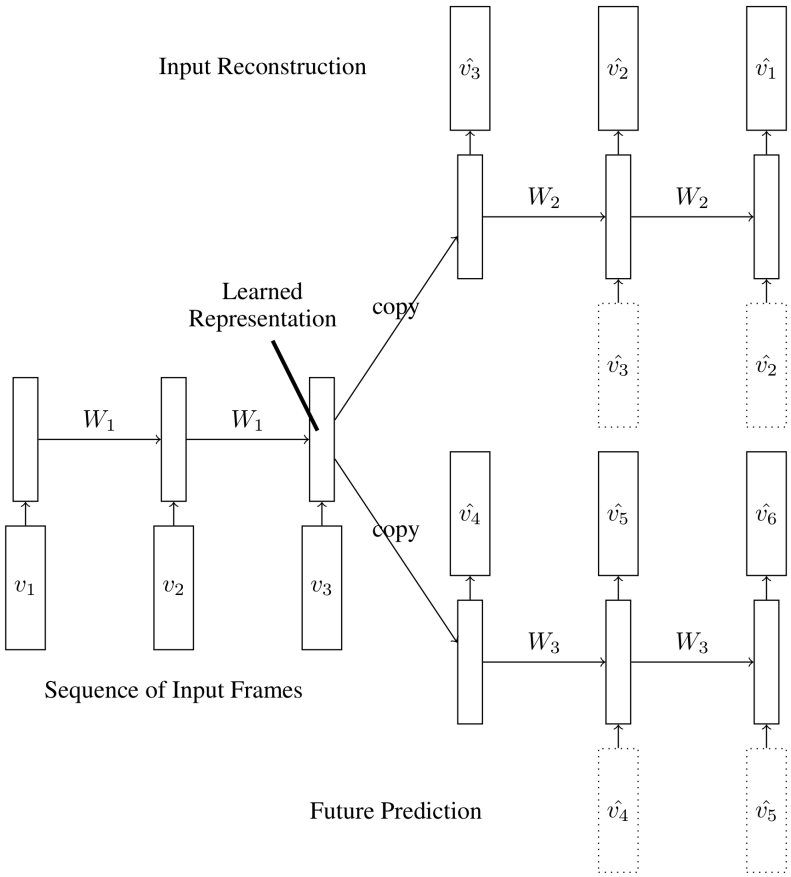


Figure 27: Composite model [38]

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 28.

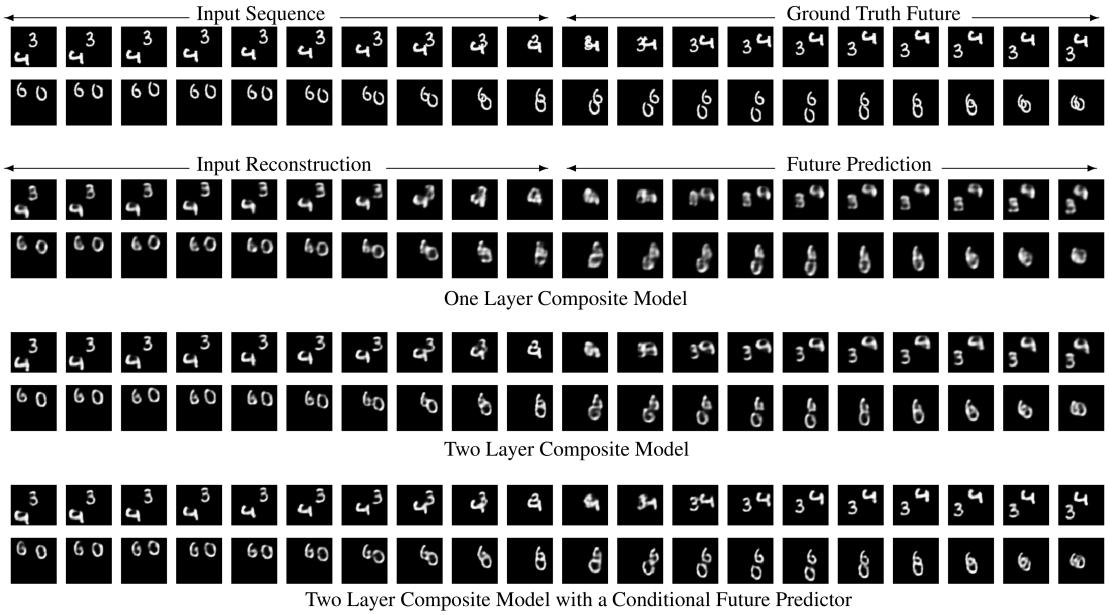


Figure 28: Input reconstruction and future predictions of composite models on a synthetic dataset of moving MNIST digits. [38]

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.

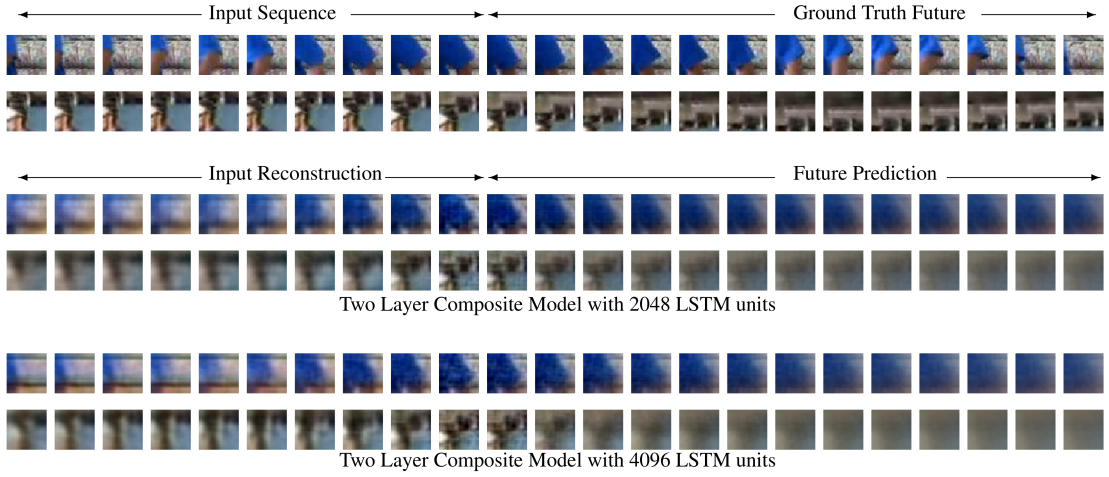


Figure 29: Image reconstruction and future prediction of a two-layer composite model with a different number of LSTM units in each layer. [38]

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 30). 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 [21], as inputs.

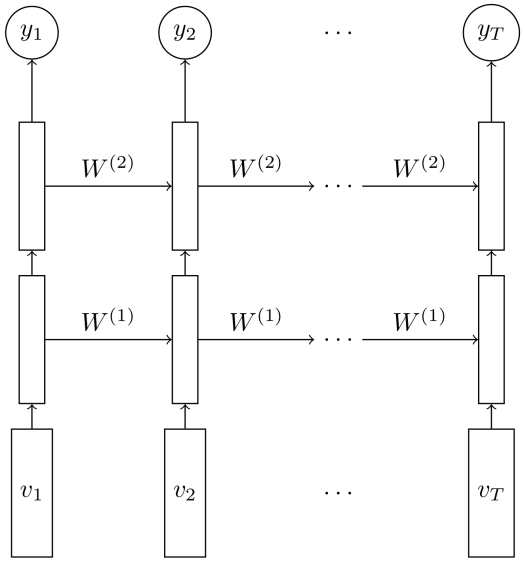


Figure 30: LSTM classifier applied on an input sequence $\{v_1, \cdots, v_T\}$ [38]

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)

[39]

Palasek and Patras [39] 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. [40] 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. [40]
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. [40]

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 31. 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 31. 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. [40]

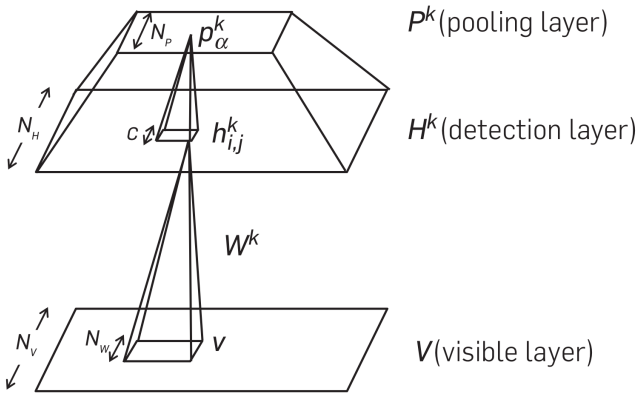


Figure 31: Architecture of the Convolutional Restricted Boltzmann Machine. The figure shows group k in the detection- and pooling-layer. [40]

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. [40] 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. [40]

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. [40]

Palasek and Patras [39] 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 32.

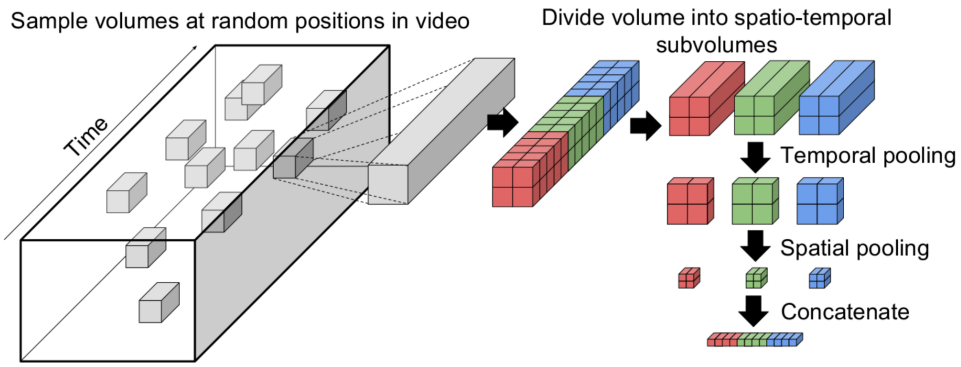


Figure 32: Approach for generating fixed sized video-representations using ConvDBN activations. An input video is handled as video-volume of stacked frames. [40]

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 32). [40]

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 [39] 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. [40]

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)

[41]

Misra, Zitnick, and Hebert [41] 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.

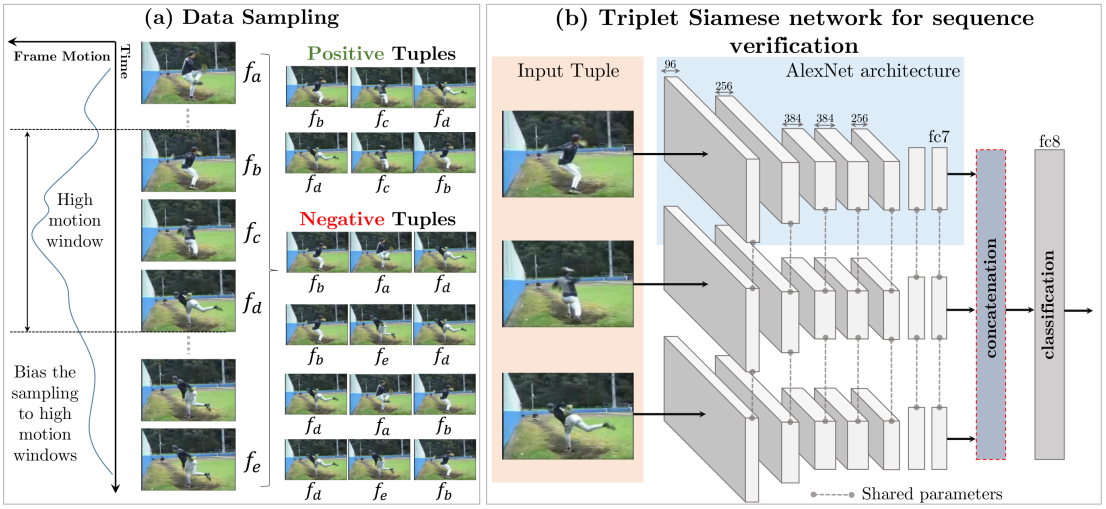


Figure 33: Sampling method of input sequences and triplet Convolutional Neural Network for temporal order verification [41]

Figure 33 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 **??**)

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. [41]

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

"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.

4.1 Review of Datasets for Human Action Classification

Review of the most important currently existing datasets, focus on newest ones (since 2013)

Reference dataset survey paper.

THUMOS is a large scale dataset (Feichtenhofer)

4.2 Alternative Benchmarks for Action Recognition Algorithms

4.3 Data Augmentation

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)

4.4 Inter-Dataset Approaches

4.4.1 Multi-task learning

4.4.2 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.

4.4.3 Unsupervised pre-training

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.

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] 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).

[9] 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).

[10] 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).

[11] 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).

[12] 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).

[13] 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).

[14] 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).

[15] 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).

[16] 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).

[17] 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).

[18] 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).

[19] 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).

[20] 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/a85a359be655f459516919138a46ae516621.pdf (visited on 11/10/2016).

[21] 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).

[22] 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).

[23] 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).

[24] Xiaojiang Peng et al. “Large Margin Dimensionality Reduction for Action Similarity Labeling”. In: *IEEE Signal Processing Letters* 21.8 (2014). 00011, pp. 1022–1025. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6807695 (visited on 11/11/2016).

[25] Christoph Feichtenhofer, Axel Pinz, and Richard P. Wildes. “Bags of Spacetime Energies for Dynamic Scene Recognition”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 00015. 2014, pp. 2681–2688. URL: http://www.cv-foundation.org/openaccess/content_cvpr_2014/html/Feichtenhofer_Bags_of_Spacetime_2014_CVPR_paper.html (visited on 11/11/2016).

[26] Xiaofeng Ren and Matthai Philipose. “Egocentric Recognition of Handled Objects: Benchmark and Analysis”. In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. 00056. IEEE, 2009, pp. 1–8. URL: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=5204360 (visited on 11/11/2016).

[27] 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).

[28] 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).

[29] 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).

[30] 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).

[31] 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).

[32] 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).

[33] 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).

[34] 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).

[35] 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).

[36] 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).

[37] 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).

[38] 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).

[39] 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).

[40] 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).

[41] 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).