ViPr Reading Group

Meeting# 02

AI&IC Lab, FCI

MMU Cyberjaya

02/05/2016

My Introduction

Saimunur Rahman

Graduate Research Assistant (JS)

Centre of Visual Computing

Multimedia University, Cyberjaya Campus

Facebook: fb.me/saimunur.rahman

Web: http://saimunur.github.io

Today's Agenda

- Talk on "**Action Recognition with Trajectory-Pooled Deep-Convolutional Descriptors**" by L. Wang, Y. Qiao, and X. Tang published at CVPR 2015.

Lets begin with some vision-based action recognition basics

Action Recognition

Machine interpretation of human actions in video

Inherent Complexity

A single activity can be performed in many ways

Video Source: YouTube

Many degrees of freedom

Large capability set, 206 bones ans 230 joints in total!!

Video Source: YouTube

View specific

Same activity can be viewed differently

Video Source: UCF-YouTube (Liu et al. 2009)

Subject dependency

Same activities can be performed differently by different people

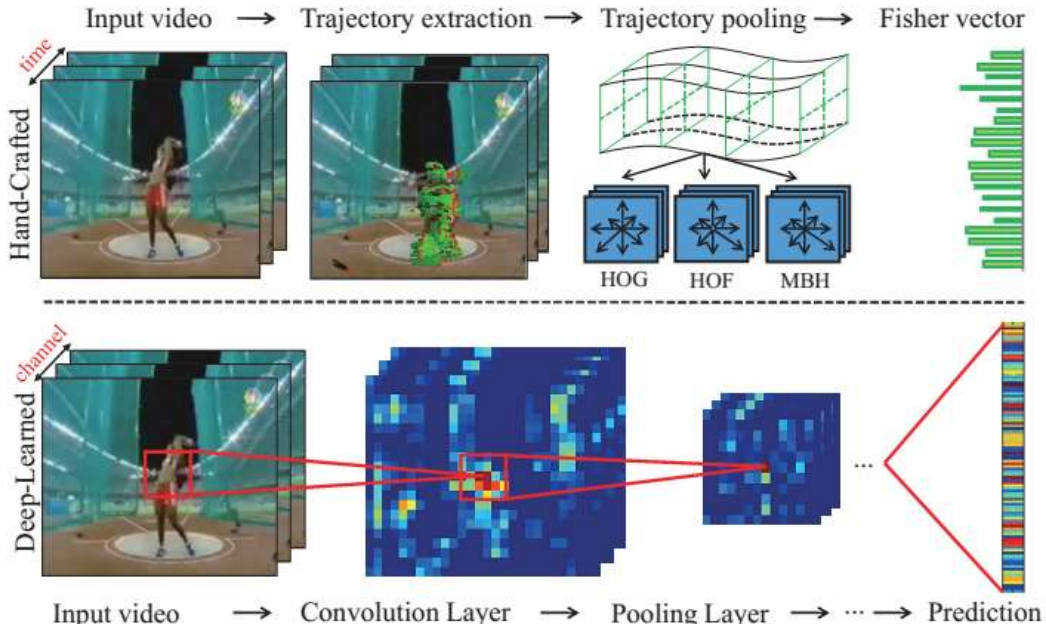
Video Source: UCF-YouTube (Liu et al. 2009)

Why it is even relevant?

Video source: YouTube

Research Trend in Action Recognition

- Hand-crafted method
  - Holistic or global method
  - Localized method
- Unsupervised method
  - Deep learning
- Fusion



Action Recognition with Trajectory-Pooled Deep-Convolutional Descriptors

Limin Wang[1,2],Yu Qiao[2], Xiaoou Tang[1,2]

[1]The Chinese University of Hong Kong

[2]Shenzhen Institutes of Advanced Technology

CVPR 2015 Poster

Total Citation : 40 (18 in 2016)

Main Idea

- Utilize deep architectures to learn conv. feature maps
- Apply trajectory based pooling to aggregate conv. features into effective descriptors
- Aims to combine the benefits of both hand-crafted and deep-learned features.

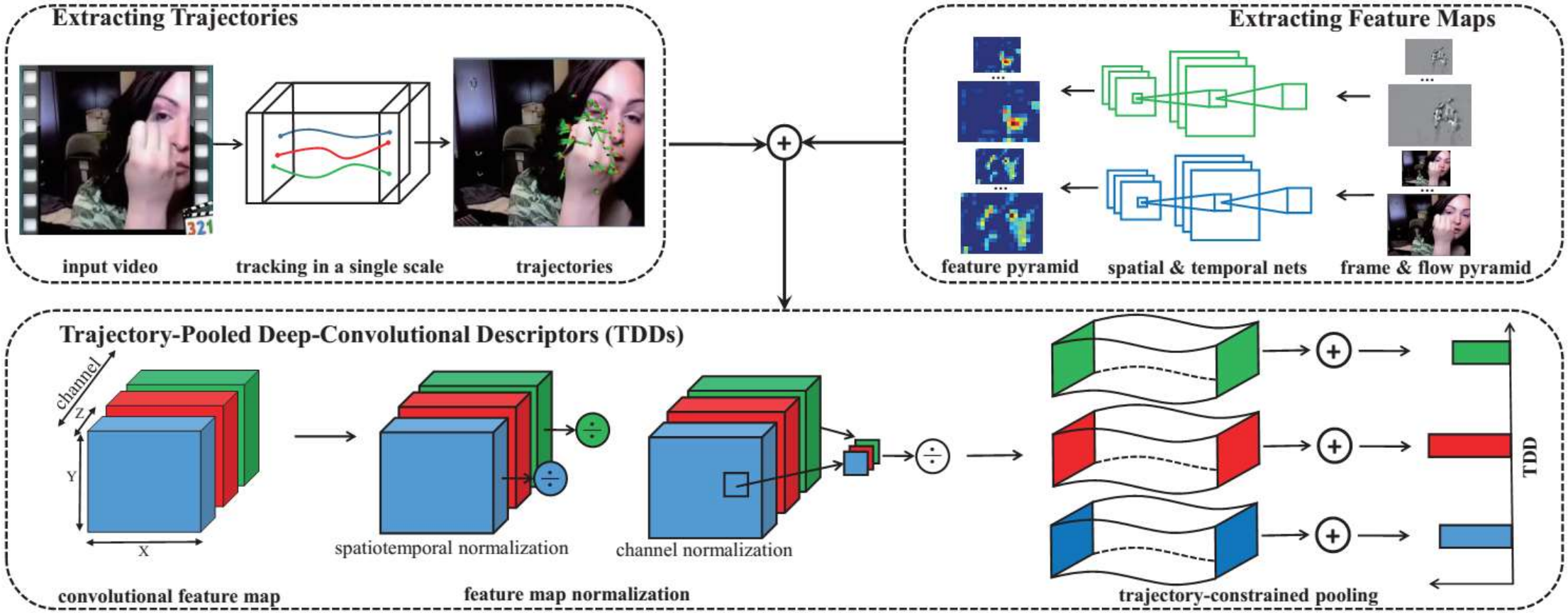
Motivations

- Hand-crafted methods are lack of discriminative capacity
- Current deep learning do not differentiate between spatial and temporal domain
  - Treat temporal dimension as feature channels when image trained ConvNet is use to model videos

Contributions

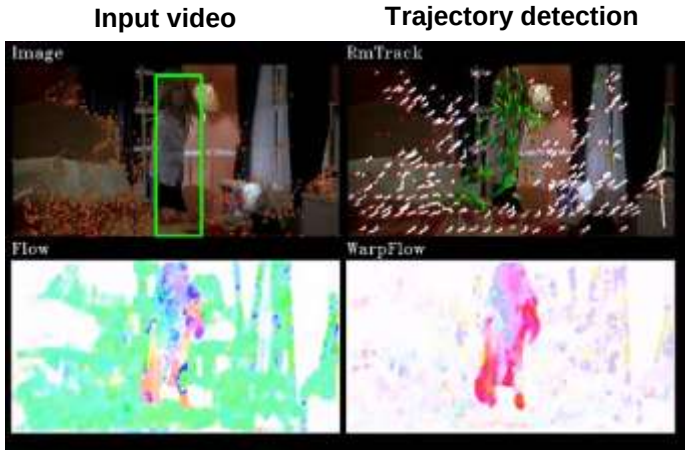
- Modified Two-stream CNN model [Simonyan and Zisserman, NIPS 2014] trained on UCF-101 [Soomro et al., CoRR 2012]
- Two CNN normalization method
- Thorough evaluation of later Convolution layers (Conv. 3,4,5)
- Multi-scale extension

Overview


Extracting Trajectories
input video
tracking in a single scale
trajectories
Extracting Feature Maps
feature pyramid
spatial & temporal nets
frame & flow pyramid
Trajectory-Pooled Deep-Convolutional Descriptors (TDDs)
channel
Z
Y
X
convolutional feature map
spatiotemporal normalization
channel normalization
feature map normalization
trajectory-constrained pooling
TDD

Trajectory extraction

- Used improved dense trajectory (iDT) [Wang et al., ICCV 13]
- Camera motion removal
  - Compute optical flow
  - Homography estimation using RANSAC [Fischler & Bolles. 1981]
    - SURF and Optical flow (OF) for similarity between two frames
  - Re-compute the optical flow – *warped flow*
- Trajectory estimation
  - Trajectories using dense trajectories [Wang et al. 11]
  - Track points with original spatial scale (results 2-3% less than multi-scale) [Wang et al. 11]



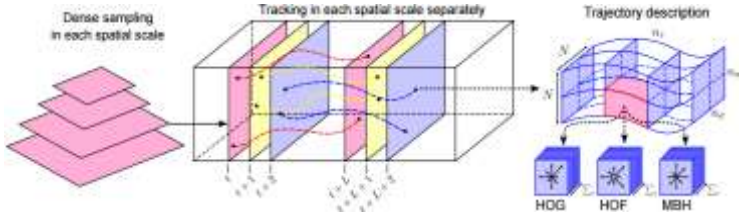
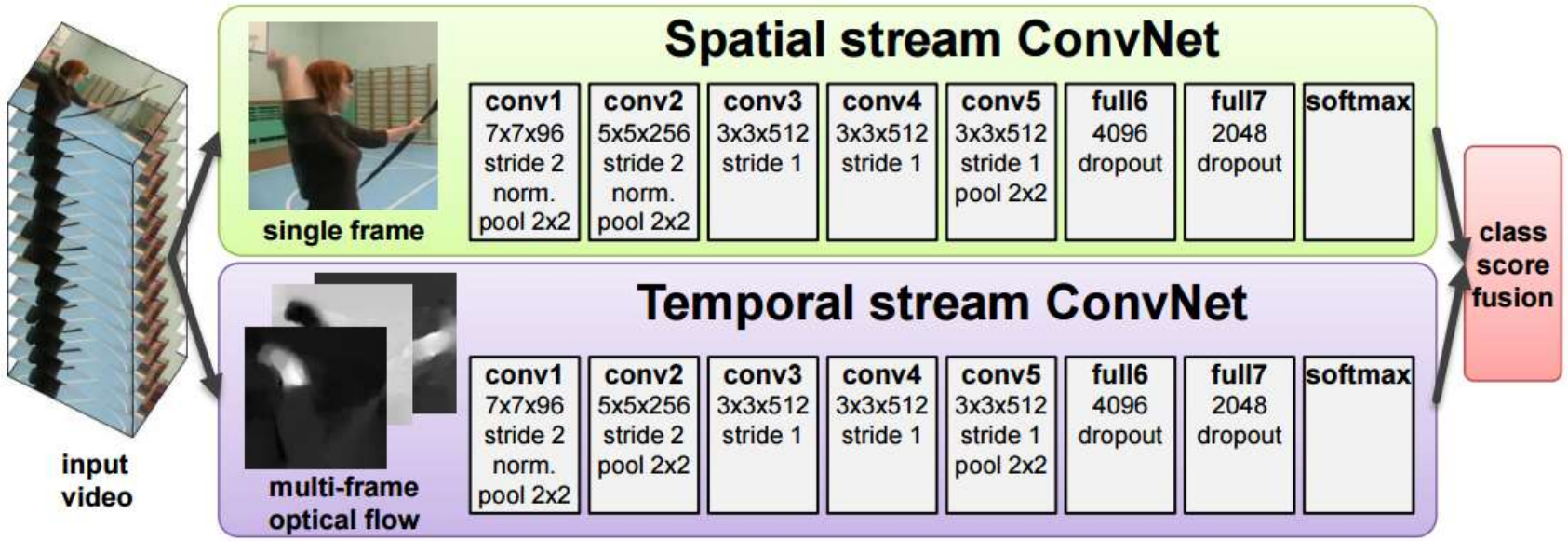


Image reproduced from Wang et al. 2013

Input video source: YouTube

Feature map extraction



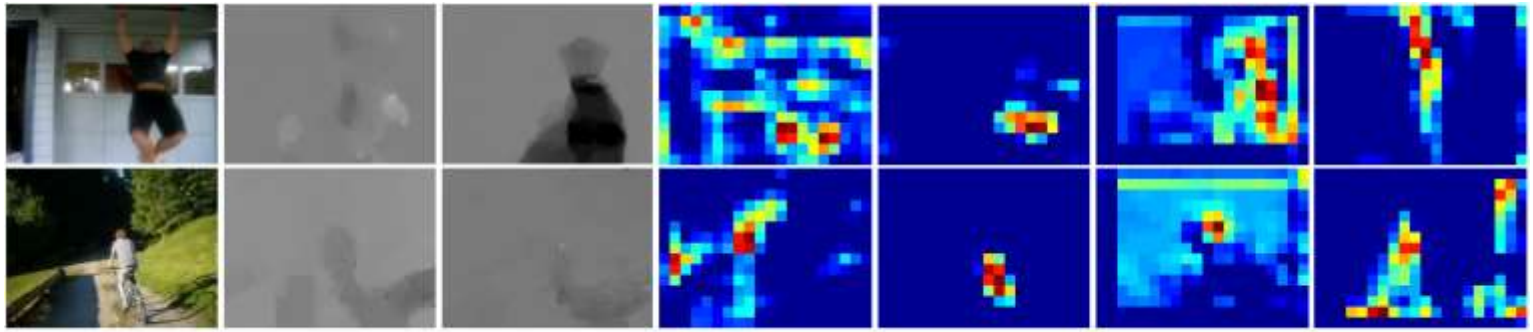
Two-stream network [Simonyan and Zisserman, NIPS 2014], Use CNN-M-2048 model [Chatfield et al, BMVC 2014]

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	-	-	-

Proposed network model of both spatial and temporal stream

Feature map extraction (2)

- Spatial-net: frame-by-frame
- Temporal-net: stack optical flow volume (one frame is replicated)
- Trajectory mapping:
  - Zero-padding $\lfloor k/2 \rfloor$, k is kernel size in conv and pooling
  - Trajectory point mapping: $(x, y, t) \rightarrow (r * x, r * y, t)$, r is feature map ratio w.r.t input image

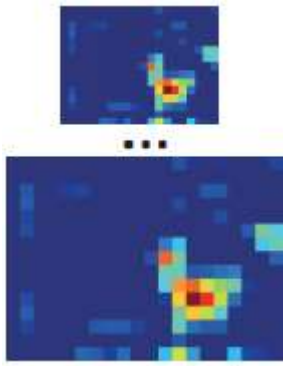


Trajectory pooled descriptor (TDD)

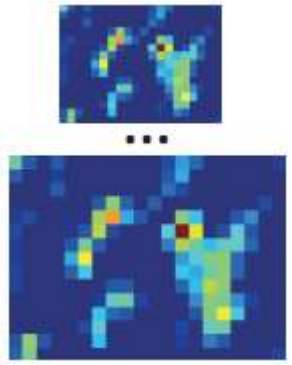
- Local trajectory-aligned descriptor computed in a 3D volume around the trajectory.
- The size is $N \times N \times P$ where, N is spatial size and P is traj. Length.
- Feature Normalization (Ensure everything is in same range and equ. Cont.)
  - Spatiotemporal Normalization: $\tilde{C}_{st}(x, y, t, n) = C(x, y, t, n)/max_{x,y,t}(x, y, t, n)$
  - Channel Normalization: $\tilde{C}_{st}(x, y, t, n) = C(x, y, t, n)/max_{n}(x, y, t, n)$
- TDD estimation is done by sum-pooling normalized channels over the trajectory: $D(T_k, \tilde{C}_m) = \sum_{p=1}^{P} \tilde{C}_m \left((r_m \times x_p^k), (r_m \times y_p^k), t^k \right)$

Multi-scale TDD

1. Multi-scale pyramid representations of video frames and optical flow fields.
2. Pyramid representations are fed into the two stream ConvNets for multi-scale feature map
3. Calculate multi-scale TDD: $(x, y, t) \rightarrow (r_m \times s \times x, r_m \times s \times y, t)$, s is the scale of features and $s = \frac{1}{2}, \frac{1}{\sqrt{2}}, 1, \sqrt{2}, 2$



Spatial net pyramid



Temporal net pyramid

Datasets

- HMDB51 [Kuehne et al., ICCV 2011]
  - 6, 766 video clips from 51 action categories
  - 3 splits for evaluation, each split has 70% training and 30% testing samples

51 action classes

Datasets

- UCF-101
  - 13, 320 video clips from 101 action categories
  - THUMOS13 challenge evaluation scheme with three training/testing splits

101 action classes

Implementation - ConvNet Training

- **Spatial Net**
  1. UCF-101 first split → resize frame to 256x256 → rand. crop 224x224 → rand. horizontal flip
  2. Pre-train the network with publicly available model from Chatfield et al. (BMVC 2014)
  3. Fine tune the model parameters on the UCF101 dataset (full dataset)

- **Temporal Net**
  1. 3D volume → resize to 256x256x.. → rand. crop 224x224x20 → rand. horizontal flip → selection of 10 frames (for performance and efficiency balancing)
  2. Train temporal net on UCF101 from scratch
  3. High dropout ratio for FC6, FC7 for improve the generalization capacity of trained model (Training Dataset is relatively small !!)

Implementation – Feature Encoding

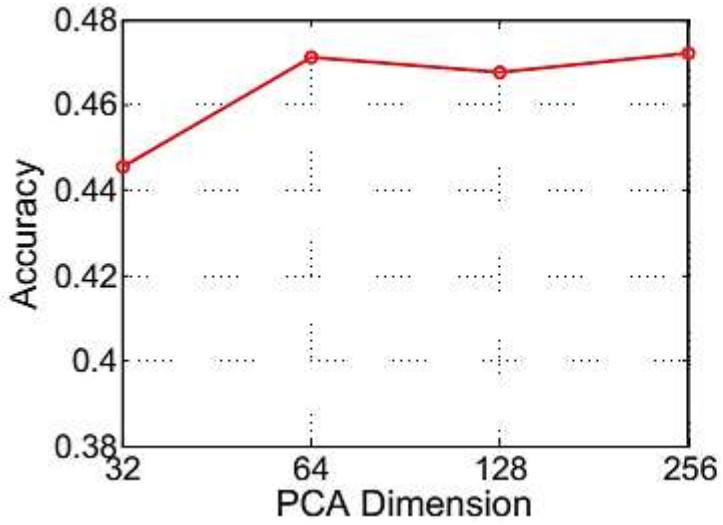
- Used fisher vector (FV) [Sanchez et al., IJCV 2013]
  - GMM clusters K = 256
  - PCA to reduce dimensionality D, FV is $2KD$ where D is feature (vector) dimension!!
- Linear SVM as the classifier ($C = 100$)

Experimental Results

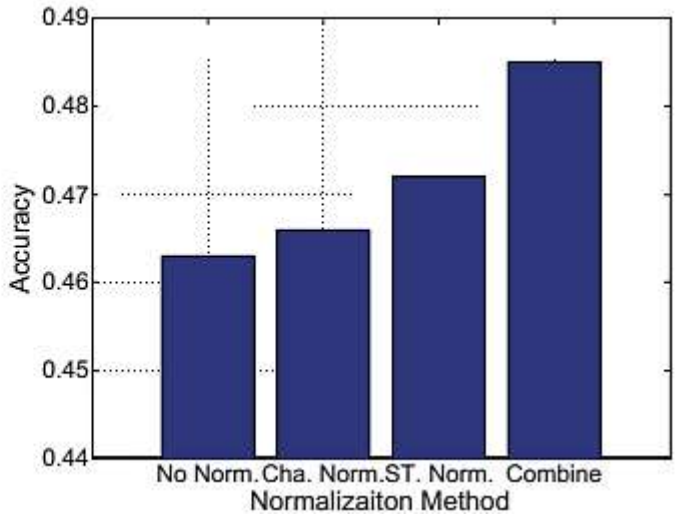
Algorithm	HMDB51	UCF101
HOG (Wang & Schmid, ICCV 13)	40.2%	72.4%
HOF (Wang & Schmid, ICCV 13)	48.9%	76.0%
MBH (Wang & Schmid, ICCV 13)	52.1%	80.8%
HOF+MBH (Wang & Schmid, ICCV 13)	54.7%	82.2%
iDT (Wang & Schmid, ICCV 13)	**57.2%**	**84.7%**
Spatial net (Simonyan and Zisserman, NIPS 2014)	40.5%	73.0%
Temporal net (Simonyan and Zisserman, NIPS 2014)	54.6%	83.7%
Two-stream ConvNets (Simonyan and Zisserman, NIPS 2014)	**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%**

- Shape is important!! See iDT vs. HOF+MBH
- Motion performance is better in 2-st. ConvNet
  - See Temporal Net
- Early Conv. Layer is better for both Net
- Spatial Conv. 4+5 is slightly better for UCF-101
- Temporal Conv. 4+5 is better for HMDB51
- iDT can further boost the TDD
  - 63.2% → **65.9%** (HMDB51)
  - 90.3% → **91.5%** (UCF-101)

Additional Exploration Experiments



Performance with PCA reduced dimension.



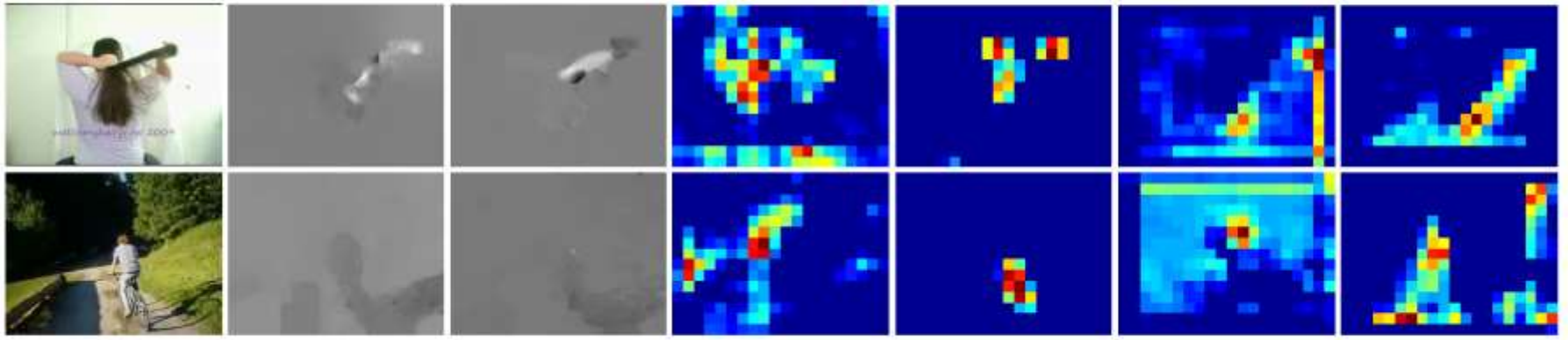
Comparison of different normalization methods.

Performance on HMDB51

ConvNet Layer performance

	Spatial ConvNets					Temporal ConvNets				
Convolutional layer	conv1	conv2	conv3	conv4	conv5	conv1	conv2	conv3	conv4	conv5
Recognition accuracy	24.1%	33.9%	41.9%	**48.5%**	**47.2%**	39.2%	50.7%	**54.5%**	**51.2%**	46.1%

Table 2. The performance of different layers of spatial nets and temporal nets on the HMDB51 dataset.



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

Figure 4. Examples of video frames, optical flow fields, and their corresponding feature maps of spatial nets and temporal nets.

- Conv1 and Conv2 are outputs of max pooling layers after convolution operations
- Conv3, Conv4 and Conv5 are outputs of RELU activations
- Observations: Earlier layers performs better than laters e.g. conv3 in Temporal ConvNet

Comparison with state-of-the-art

HMDB51		UCF101	
STIP+BoVW (HMDB51 Baseline)	23.0%	STIP+BoVW (UCF-101 Baseline)	43.9%
Motionlets (Wang et al., CVPR 2013)	42.1%	Deep Net (Karpathy et al., CVPR 2014)	63.3%
DT+BoVW (Wang et al., IJCV 2013)	46.6%	DT+VLAD (Cai et al., CVPR 2014)	79.9%
DT+MVSV (Cai et al., CVPR 2014)	55.9%	DT+MVSV (Cai et al., CVPR 2014)	83.5%
iDT+FV (Wang and Schmid, ICCV 2013)	57.2%	iDT+FV (Wang and Schmid, CVPRW 2013)	85.9%
iDT+HSV (Peng et al., CoRR 2014)	61.1%	iDT+HSV (Peng et al., CoRR 2014)	87.9%
Two Stream (Simonyan and Zisserman, NIPS 2014)	59.4%	Two Stream (Simonyan and Zisserman, NIPS 2014)	88.0%
TDD+FV	63.2%	TDD+FV	90.3%
Best Result	**65.9%**	**Best Result**	**91.5%**

Similar results with one-stream CNN on UCF-101: **91.1**% (Ma et al., arXiv 2016)

Conclusions

- An idea of exploiting 2D CNN models for action recognition
- Exploited raw image value and optical flow for model training
- Normalization of feature maps increase performance
- Single-trajectory features are good enough to achieve competitive perfm.
- Late Conv layers offers more discriminative features
- Handcrafted features can help to boost the feature performance

Few important information about TDD

- Spatial (pre-trained and fine-tuned) and Temporal model are available online
- Dense optical flow and trajectory code is also available online
- Ready-to-go main script (MatLab) for Linux is also available online
- For CNN the Caffe toolbox (Python) was used!!

Thank You