Video recognition

Boris Knyazev*

*Most of this research was conducted in 2015 while I was at VisionLabs company.
The research is supported by the Russian Ministry
of Science and Education grant RFMEFI57914X0071.

Why is it interesting today

~100M surveillance cameras worldwide

500k in London

130k in Moscow (video.mos.ru)

More automatic analysis = cheaper security, less human factor

More privacy issues

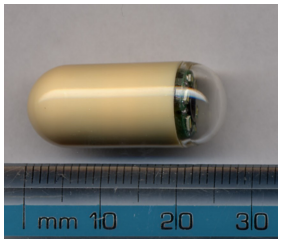
Why is it interesting today

~50k videos uploaded every hour

More automatic analysis = better video hosting service, fewer criminal content

More privacy issues

Why is it interesting today



Many specific video collections:

- job interviews and meetings
- Interrogations
- medical data
- others (bbc motion gallery, TV recordings)

Humans (and penguins?) take about 35-40% of all video pixels

[I. Laptev; CVLM'13]

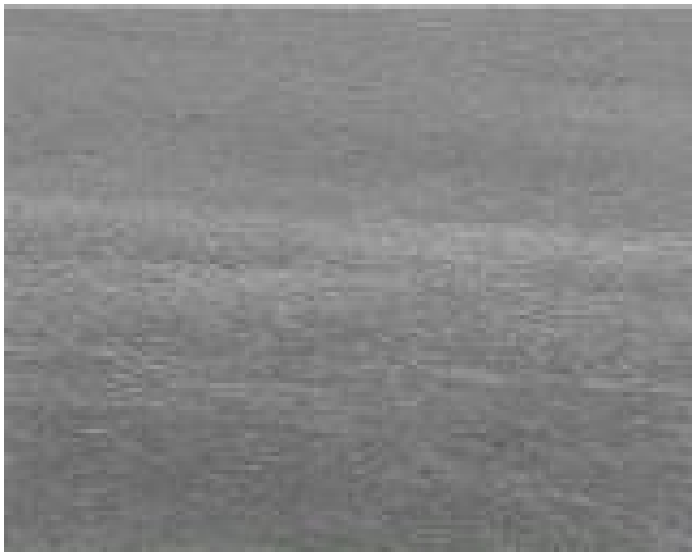
More automatic analysis = higher quality research

More privacy issues

Key challenges

- Complex **mi x tu r e** of **spatial** and **t e mpo r a l** variations within the same class of action
- More computational resources (e.g., 1 min = 240x320x3x1500 = 3.5×10^8 elements)
- Manual labeling is difficult
  - one might need to watch a video for several minutes to understand its class
  - some classes are very rare
- Verbs describing actions in videos are sometimes ambiguous

jogging

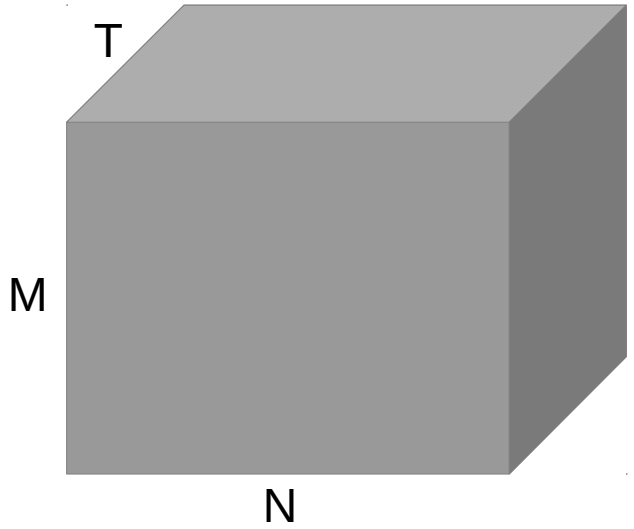


running

Inner Brow Raiser
Brow Lowerer

Inner Brow Raiser
Brow Lowerer
+ blink (in general, very difficult to count, frame by frame does not work)
+ **Outer** Brow Raiser

Problem formulation



$y_i = f(\mathbf{x_i})$

1
2
3
.
.
.
k-1
k

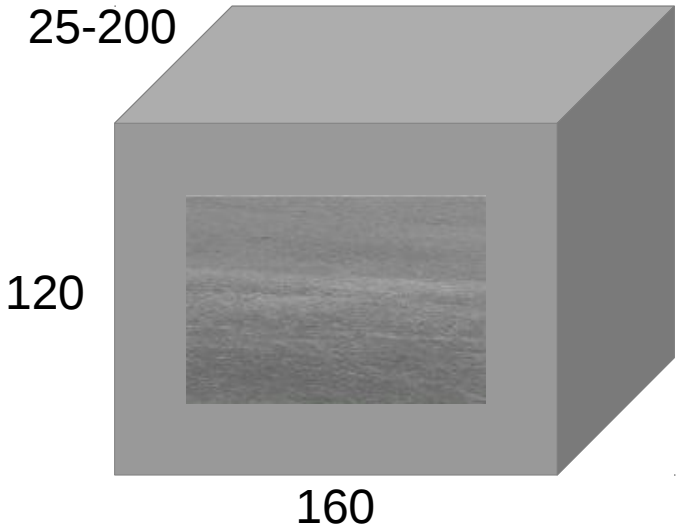
$\mathbf{x_i} \in \mathbb{R}^{M \times N \times colors \times T}$ - video

$y_i \in \{1, \ldots, k\}$ - labels (classes)

1. Train model ***f*** given pairs <$\mathbf{x}_i$,$\mathbf{y}_i$>.
2. Predict label $\mathbf{y}_j$ for previously unseen video $\mathbf{x}_j$ as good (in some sense) as possible.

Example - KTH dataset

[C. Schuldt, I. Laptev, B. Caputo; ICPR'04]



$y_i = f(\mathbf{x_i})$

Boxing
Handclapping
Handwaving
Jogging
Running
Walking

$\mathbf{x_i} \in \mathbb{R}^{M \times N \times colors \times T}$ - video

$y_i \in \{1, \ldots, k\}$ - labels (classes)

1532 training samples

1. Train model ***f*** given pairs <$\mathbf{x}_i$,$\mathbf{y}_i$>.
2. Predict label $\mathbf{y}_j$ for previously unseen video $\mathbf{x}_j$ as good (in some sense) as possible.

863 test samples

Example - UCF101 dataset

[K. Soomro, A. R. Zamir, M. Shah; CRCV-TR-12-01, 2012]

$y_i = f(\mathbf{x_i})$

ApplyEyeMakeup
Archery
BabyCrawling
BalanceBeam
.
.
.
Diving
.
.
.
PizzaTossing
.
.
.
SkyDiving
.
.
.
YoYo

101 classes

$\mathbf{x_i} \in \mathbb{R}^{M \times N \times colors \times T}$ - video

$y_i \in \{1, \ldots, k\}$ - labels (classes)

9.5k training samples

1. Train model ***f*** given pairs $<\mathbf{x}_i, \mathbf{y}_i>$.
2. Predict label $\mathbf{y}_j$ for previously unseen video $\mathbf{x}_j$ as good (in some sense) as possible.

3.8k test samples

Common methods

1. Bag of visual words (BoW, or bag of features)
2. Filters

    predefined filters

    unsupervised learning

    CNN

Bag of visual words (BoW)

Same as for images (e.g., [L Fei-Fei, P Perona; CVPR'05])

But, spatio-temporal words (word = cluster)

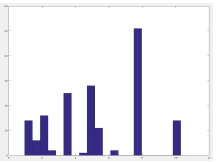
Feature Extraction

Feature Detection

Maximum variation in space (Harris) → Maximum variation in space and time:
Harris3D (STIP) [I.Laptev, 2005]

Histograms of gradients →HOG3D
Histograms of optical flow

Feature Descriptor



Clustering

Random subsets of features +
Randomized clustering algorithm

- kmeans
- GMM

Features Encoding

- VQ (Flann)
- Fisher

Normalization of histogroms

Concatenation of histograms in case of many descriptors

Classifcation

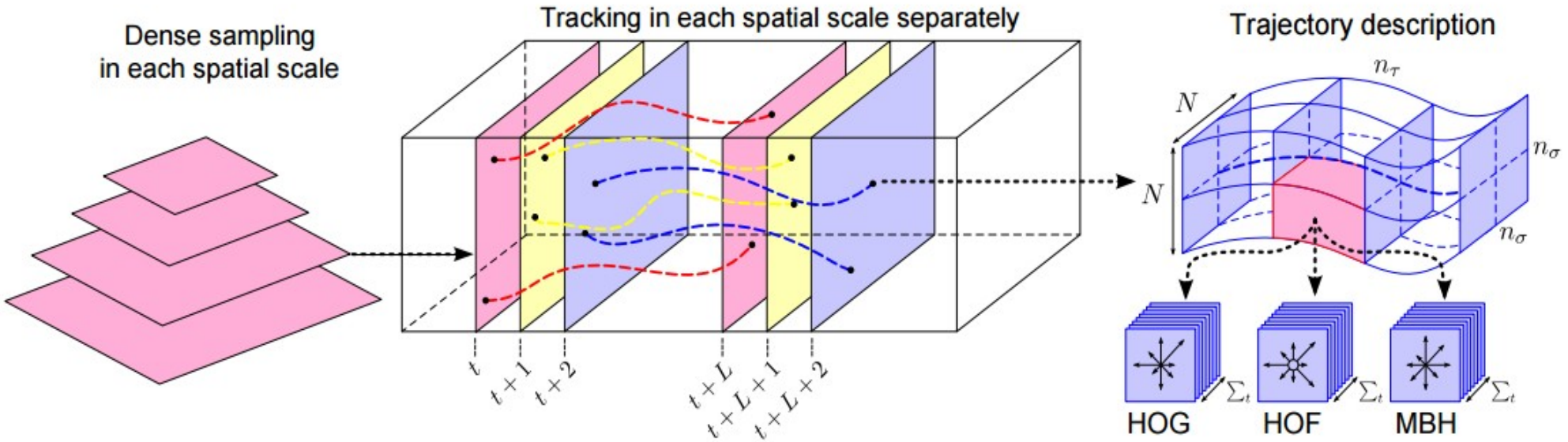
[I. Laptev, M. Marszałek, C. Schmid, B. Rozenfeld; CVPR'08]

Dense trajectories (DT) as feature detection and descriptor

[H. Wang, A. Kläser, C. Schmid, C.L. Liu; CVPR'11]

Both feature detection (motion trajectories) and feature descriptor composed of:

- [Trajectories+] HOG + HOF + MBHx + MBHy



A trajectory: 15 coordinates (x,y) corresponding to optical flow of pixels satisfying certain criteria

Pros:

High classification accuracy

Fast (about real time for 160x120 videos in KTH, 10-15 fps for 320x240 videos in UCF101)

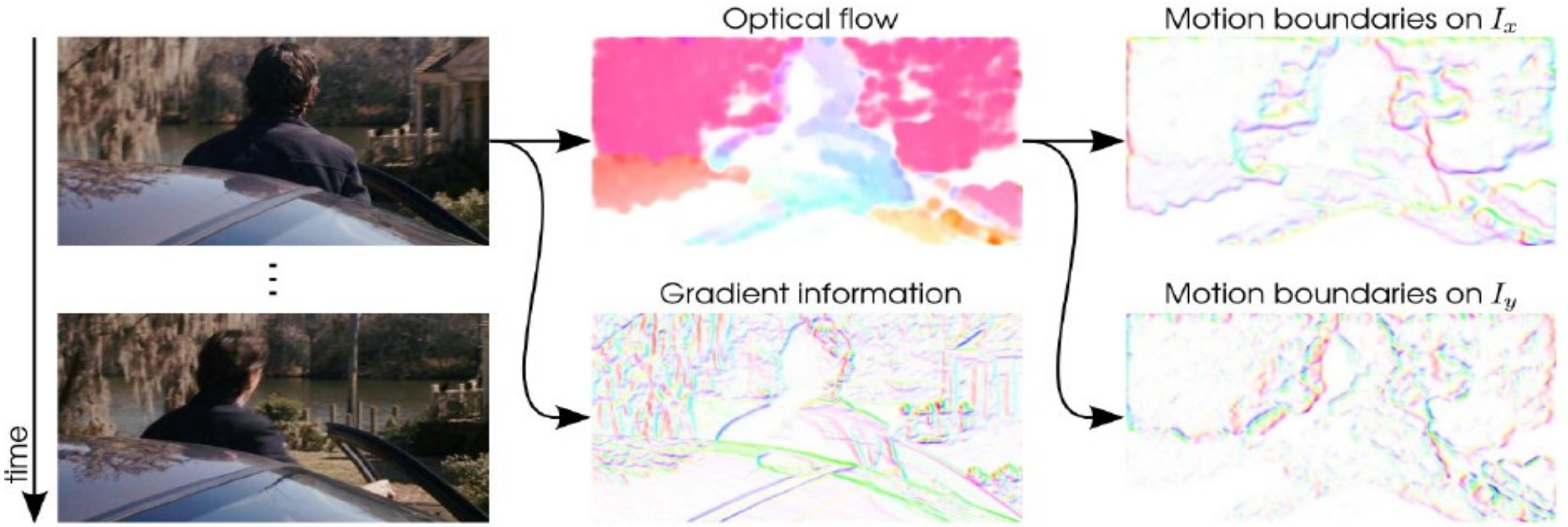
Cons:

Very dense → huge RAM and HDD requirements (descriptor size / video size ~ 100)

Sensitive to camera movements

HOG, HOF, and MBH descriptors

[H. Wang, A. Kläser, C. Schmid, C.L. Liu; CVPR'11]



Motion boundaries (MBH) highlight foreground motion (and dynamic texture) and suppress background (static texture) and camera motion

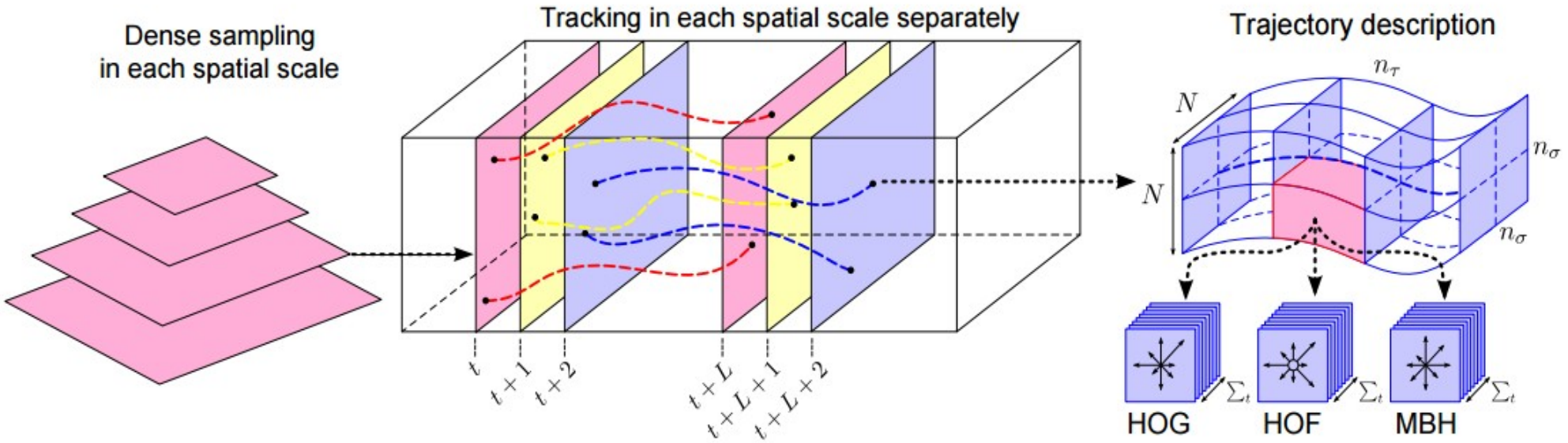
Video **x** of size 320x240x3x200 (~**1Mb** in MPEG-4)→ D of size nx426, n ~ 100k (~**150 Mb** in .txt)

Improved Dense trajectories (iDT)

[H. Wang, C. Schmid; ICCV'13]

Both feature detection (motion trajectories) and feature descriptor composed of:

- [Trajectories+] HOG + HOF + MBHx + MBHy



Pros:

High classification accuracy

Fast (about real time for 160×120 videos in KTH, 10-15 fps for 320×240 videos in UCF101)

Less sensitive to camera movements

Cons:

Dense → huge RAM and HDD requirements (descriptor size / video size **<** 100)

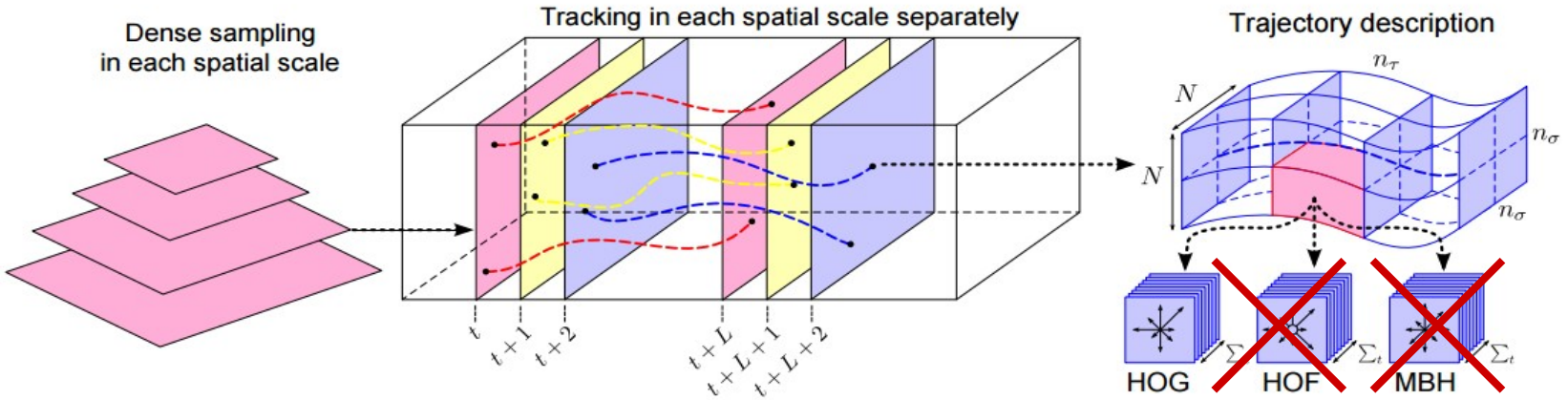
Can lose a battle to Neural Networks

Fast Dense trajectories (FDT)

[V. Kantorov, I. Laptev; CVPR'14]

Both feature detection (motion trajectories) and feature descriptor composed of:

- [Trajectories+] HOG + HOF + MBHx + MBHy



Pros:

Still relatively high classification accuracy

Very fast (>400 fps for 320×240 videos in UCF101, because it's simple decoding video. So, even 640×480 videos can be classified in real time on CPU)

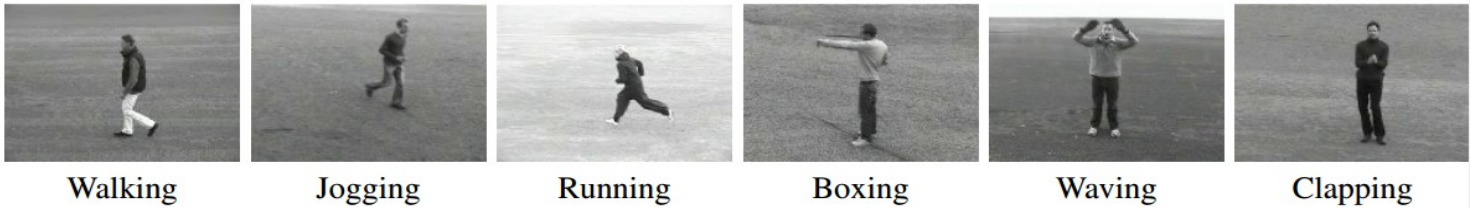
Cons:

Dense → huge RAM and HDD requirements (descriptor size / video size ~ 100)

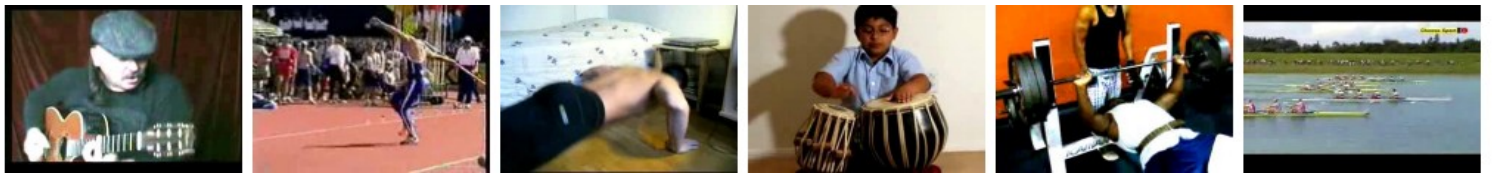
Videos must be encoded with MPEG-4

Some problems with codecs (so, some MPEG-4 videos must be re-encoded)

Datasets



KTH, 6 classes, resolution 160x120, grayscale, 1532+863 samples



UCF101, 101 classes, resolution 320x240, 3570+1530 samples (3 splits or folds)

HMDB51, 51 classes, resolution 320x240, 9538+3784 samples (3 splits or folds)

Hollywood2, 12 classes (multiple per sample), resolution 640x480 (on average), 823+884 samples

complexity

BoW: Results on KTH dataset

Descriptors

Detectors	HOG[1]	HOG[2]+HOF	HOG[2]	HOF	Cuboids	ESURF
Cuboids	90.0	88.7	82.3	88.2	89.1	-
Hessian	84.6	88.7	77.7	88.6	-	81.4
Dense	85.3	86.1	79.0	88.0	-	-
Harris3D (STIP)	89.0	91.8	80.9	**92.1**	-	-
Harris3D (STIP)	-	**91.8**	**82.4**	**91.9**	-	-

In my implementation

92.6 validation of *C* in SVM
92.9 validation of *A* in χ^2 kernel fo SVM
93.05 using rootsift norm and SVM onevsone

SVM with χ^2 kernel:

$$K(H_i, H_j) = exp(-\frac{1}{2A}\sum_{n=1}^{V}\frac{(h_{in}-h_{jn})^2}{h_{in}+h_{jn}})$$

$\mathbf{H}_i = \{h_{in}\}$ and $\mathbf{H}_j = \{h_{jn}\}$ - frequency histograms of spatio-temporal word occurrences and V - vocabulary size (V=4000). A is the mean value of distances between all training samples (can be validated). K-means to build the vocabulary.

[H. Wang, M. M. Ullah, A. Kläser, I. Laptev, C. Schmid; BMVC'09]

Confusion between 6 classes of KTH

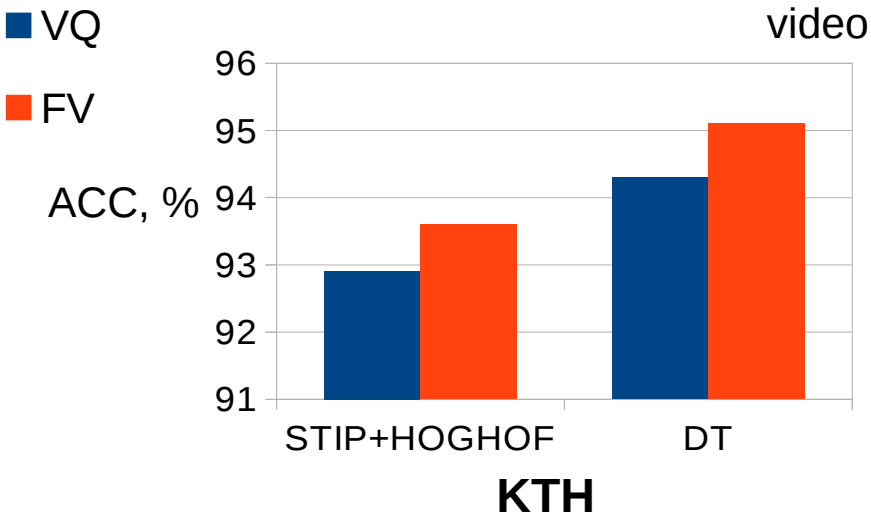
Prediction

True class

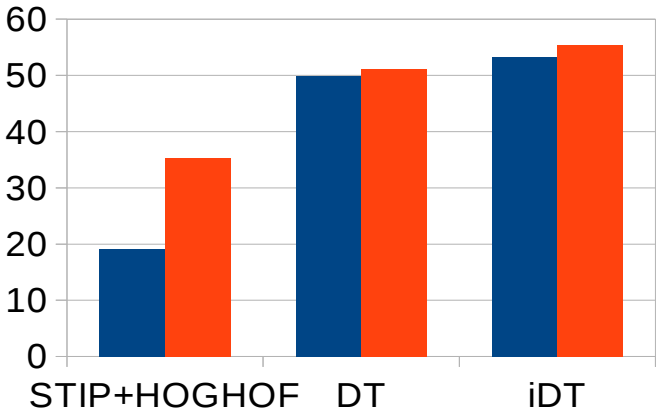
	Boxing	Handclapping	Handwaving	Jogging	Running	Walking
Boxing	**93.01**	0	0	0	0	**6.99**
Handclapping	0	**100**	0	0	0	0
Handwaving	0	**6.94**	**93.06**	0	0	0
Jogging	0	0	0	**94.44**	**4.17**	1.39
Running	0	0	0	**22.22**	**77.78**	0
Walking	0	0	0	0	0	**100**

Comparison of features and encoding methods: our results

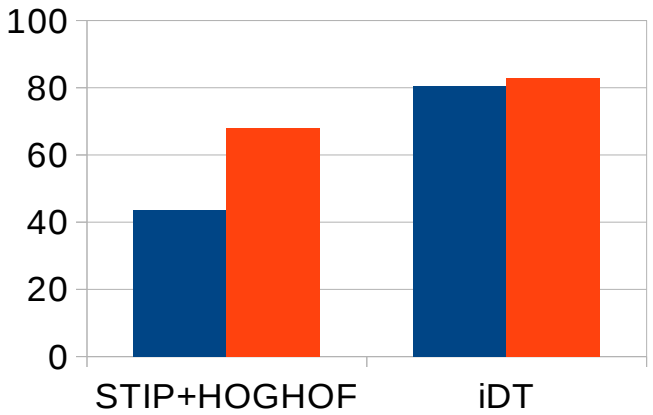
y axis: ACC (average classification accuracy among all test video samples), %



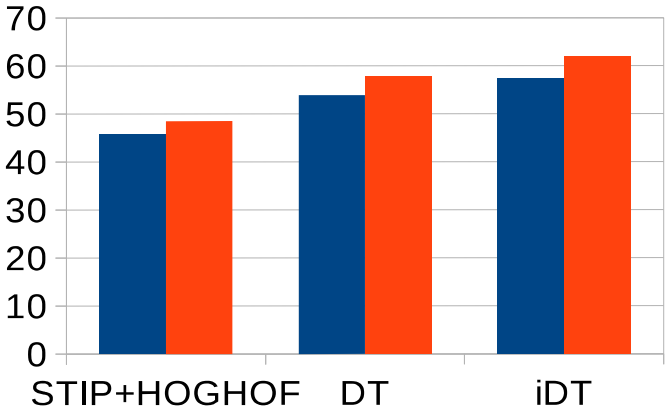
KTH



HMDB51

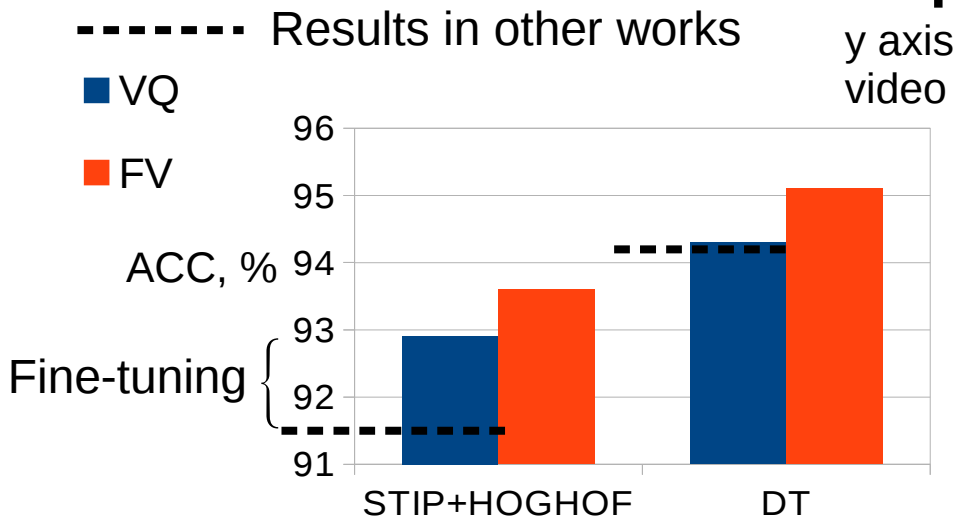


UCF101



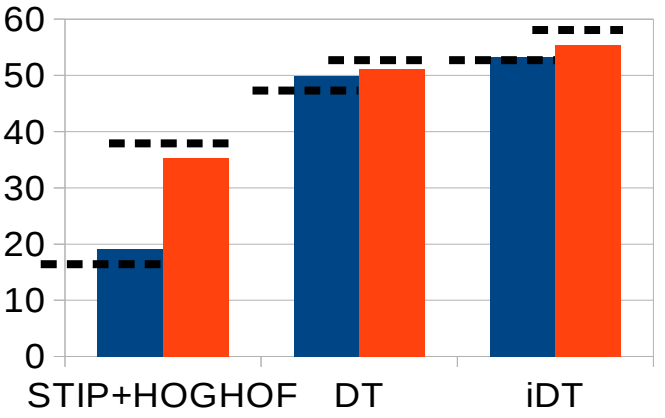
Hollywood2

Comparison of features and encoding methods: previous works

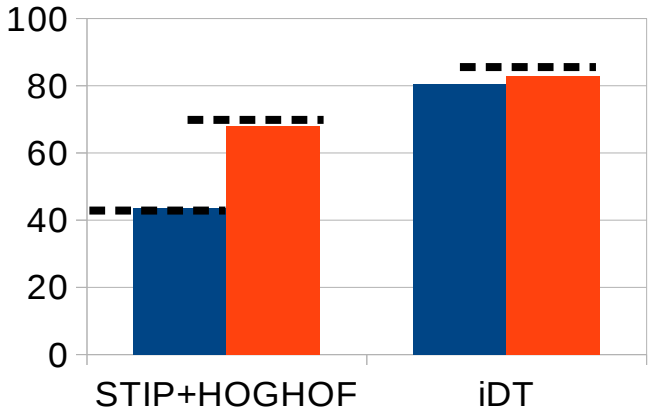


KTH

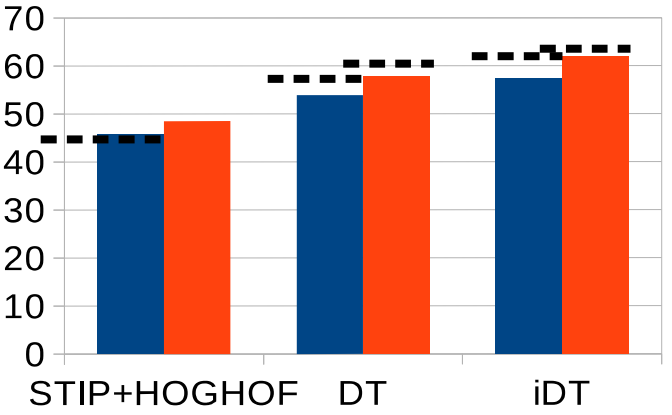
y axis: ACC (average classification accuracy among all test video samples), %



HMDB51



UCF101

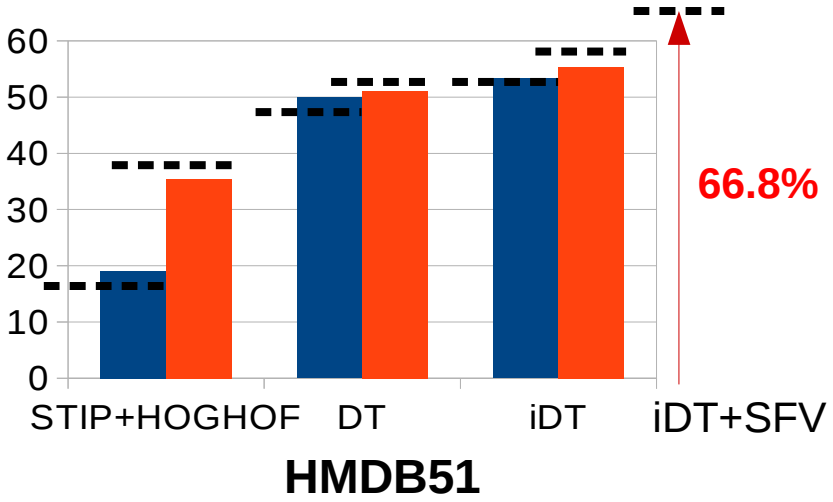


Hollywood2

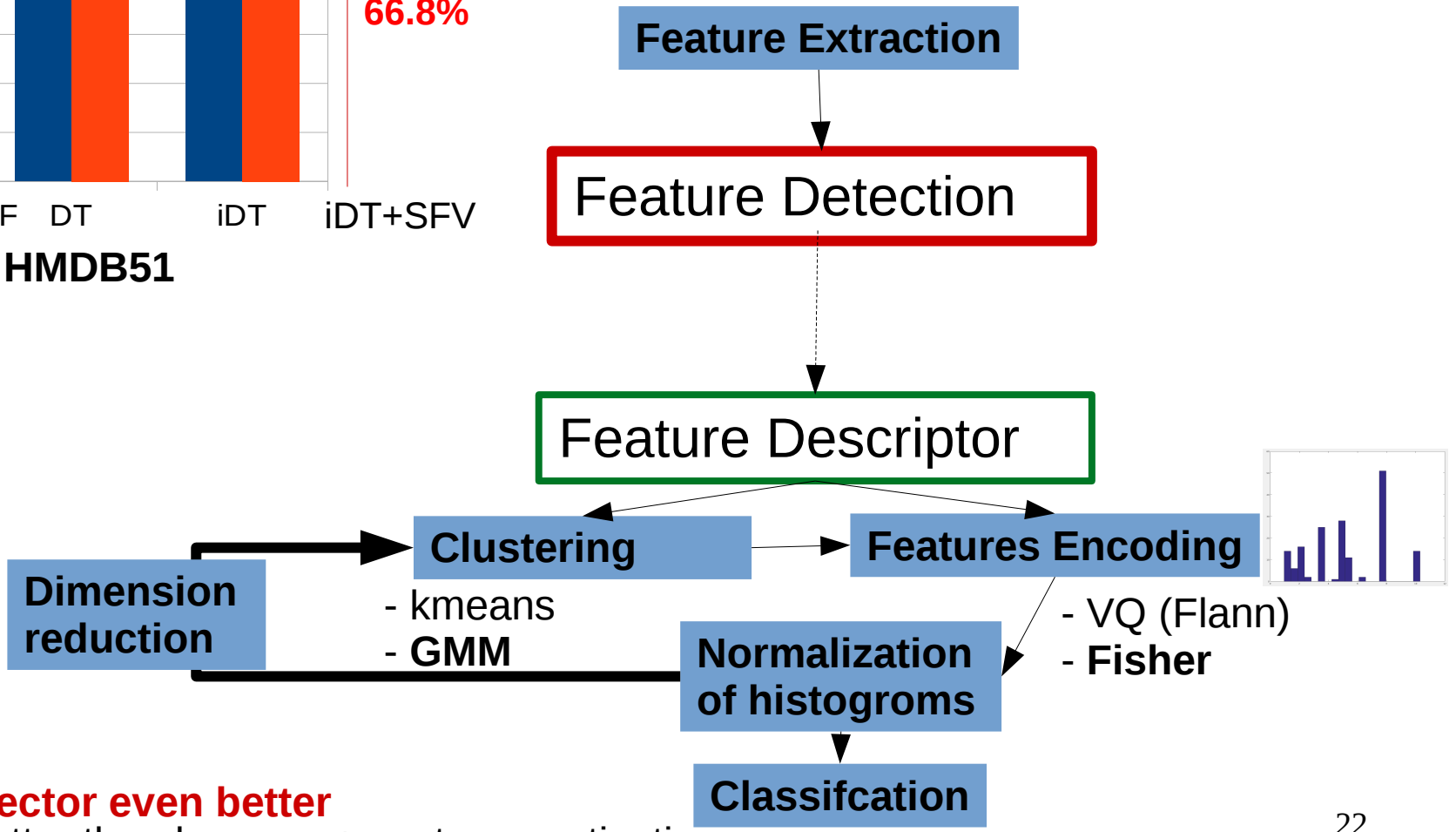
Fisher coding is better than kmeans + vector quantization
Dense trajectories are better than STIP+HOGHOF

IDT+ 2 layers of FV = iDT+SFV (Stacked Fisher Vectors)

[X. Peng, C.Q. Zou, Y. Qiao, Q. Peng; ECCV'14]



But, spatio-temporal words (word = cluster)

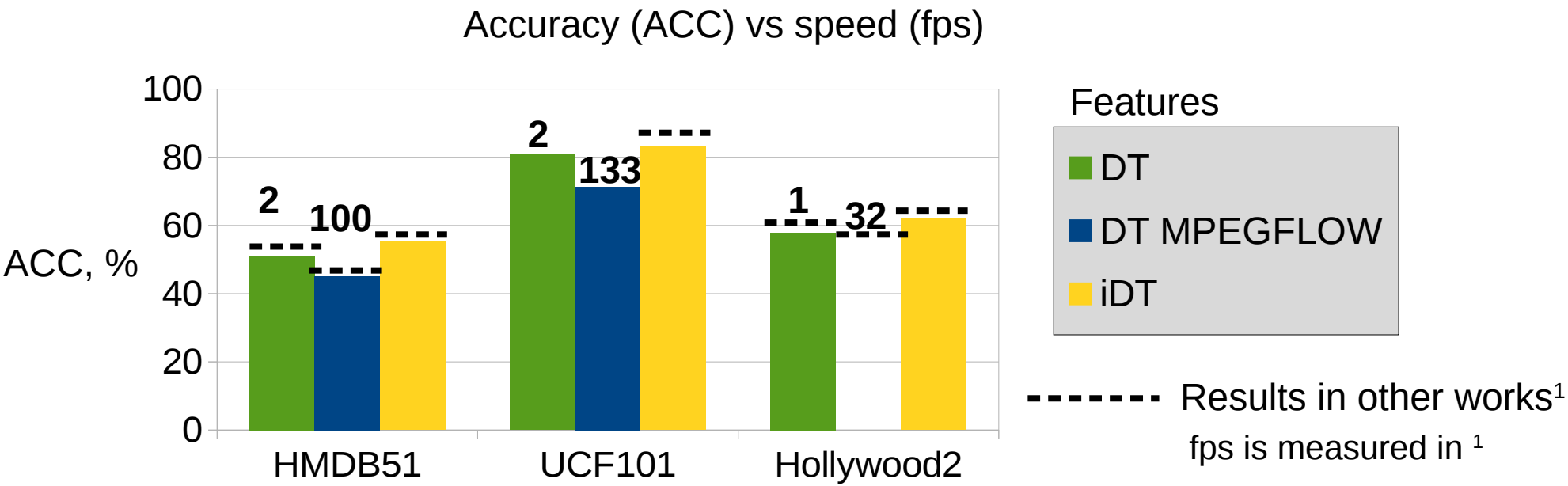


Stacked Fisher Vector even better

Fisher coding is better than kmeans + vector quantization

Dense trajectories are better than STIP+HOGHOF

Influence of replacing Farneback optical flow with “MPEG flow”

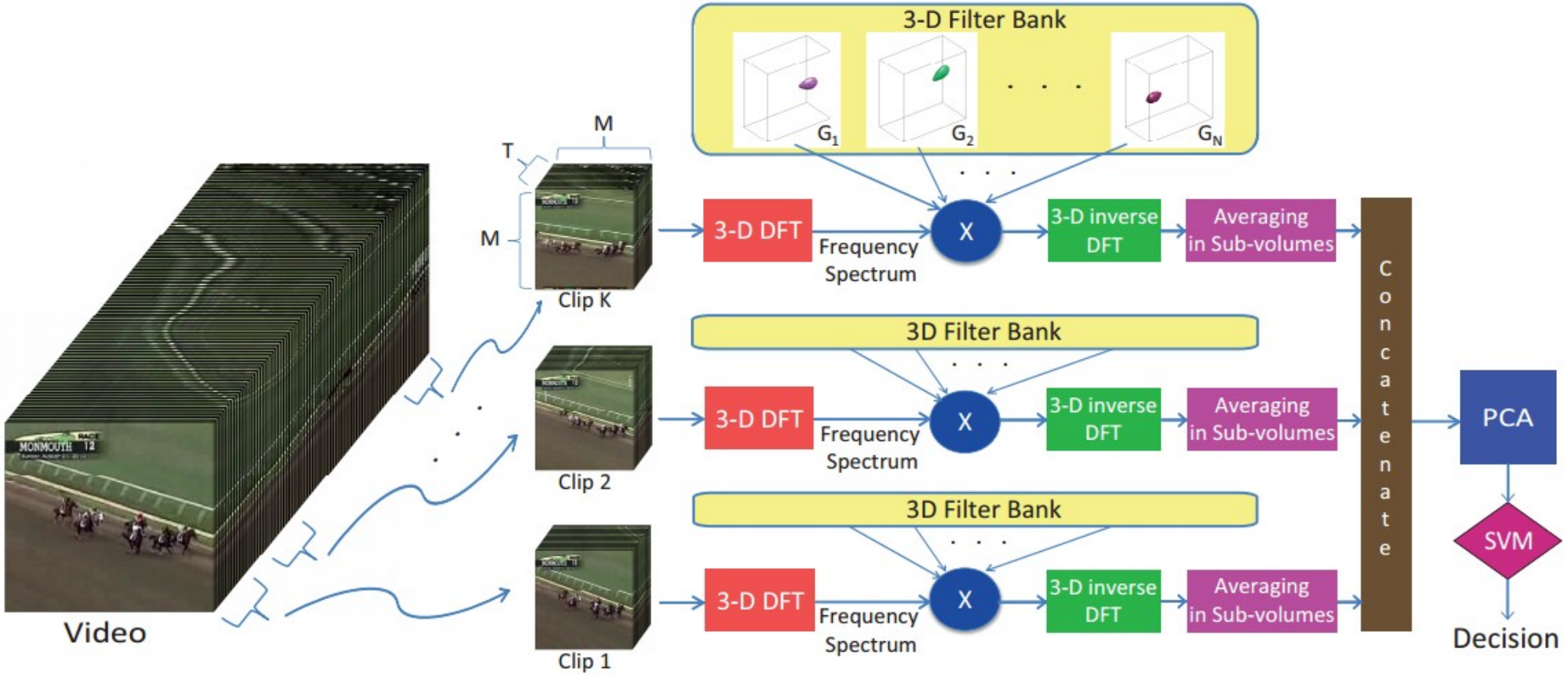


MPEG flow can be used depending on application requirements: accuracy vs speed

[1] [V. Kantorov, I. Laptev; CVPR'14]

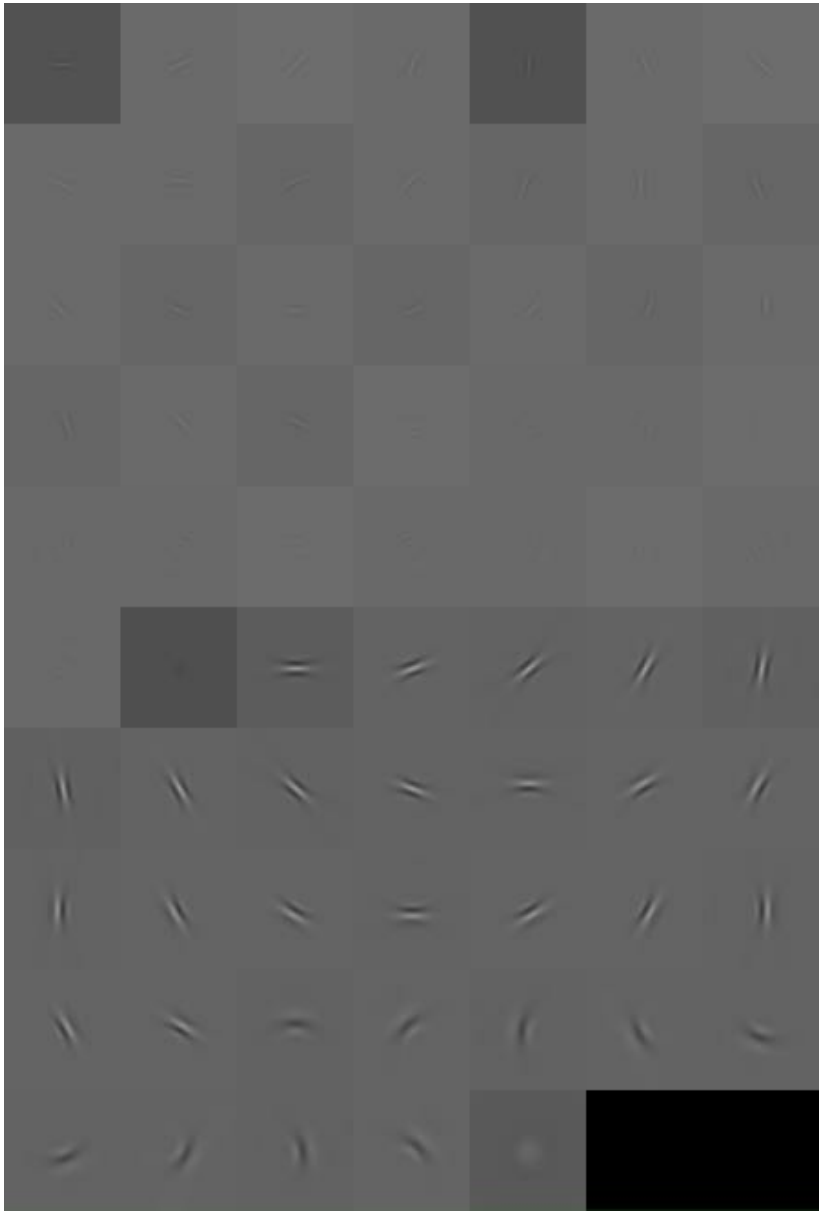
Filters: Global Video Descriptor "Gist3D"

[B. Solmaz,S. M. Assari, M. Shah; Machine Vision and Applications (MVA) '12]

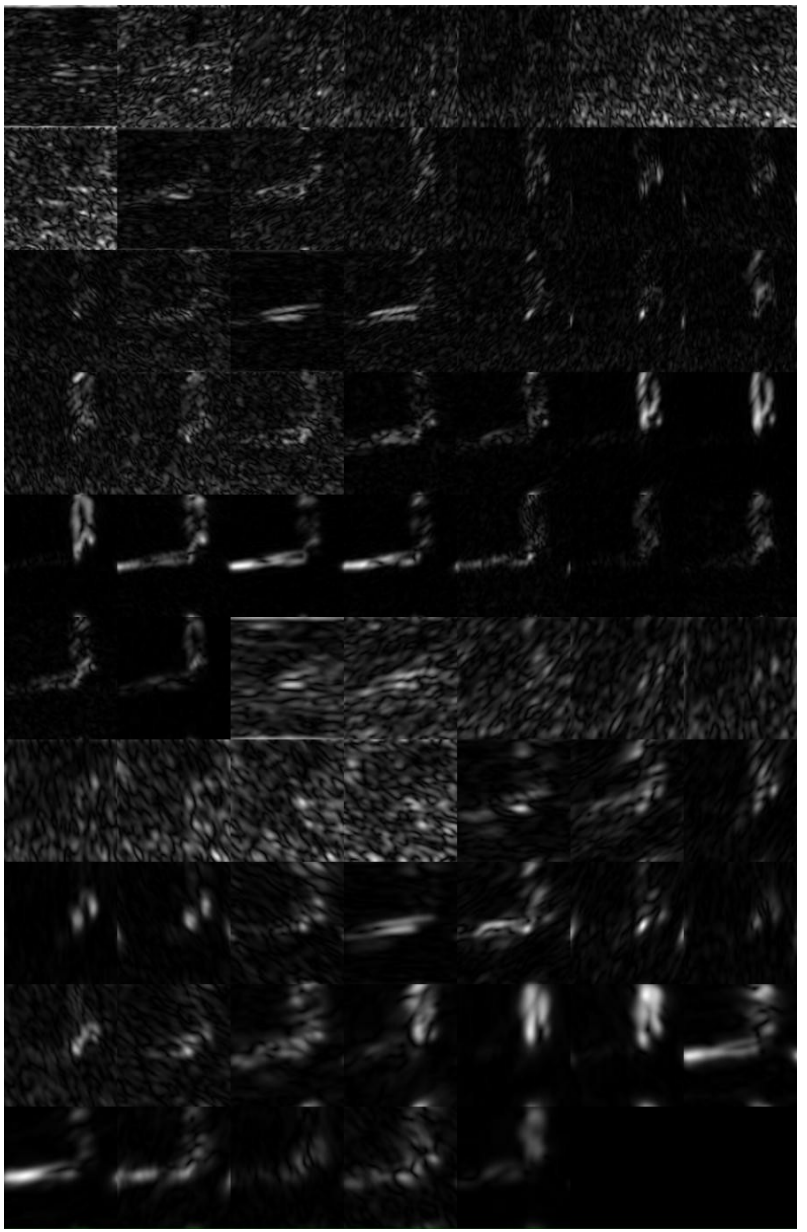


Gist3D filters and outputs

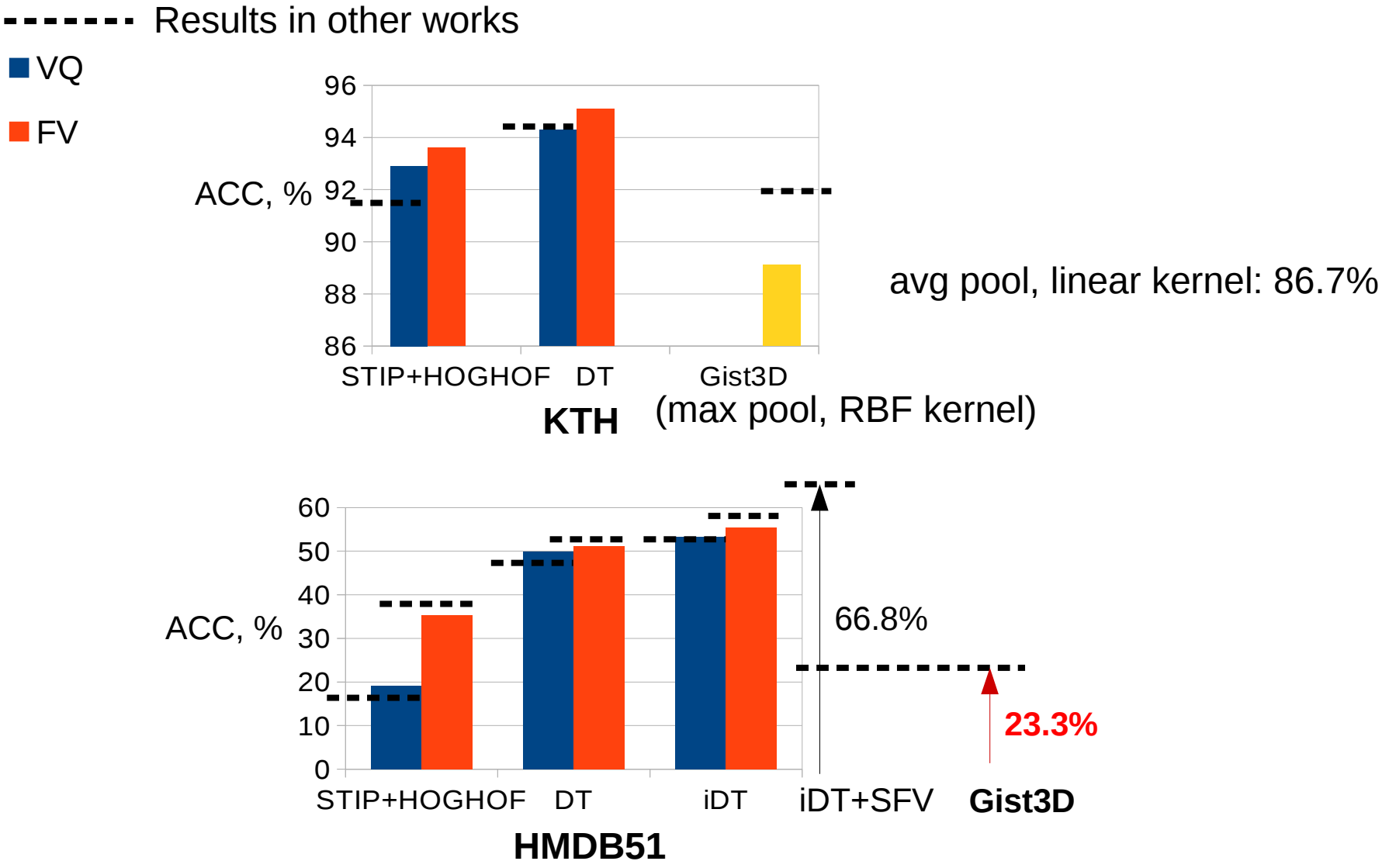
T=64



T=64



Comparison of features and encoding methods: Gist3D



3D Gabor filters are good for baseline, but need further improvement

Filters: Convolutional Networks

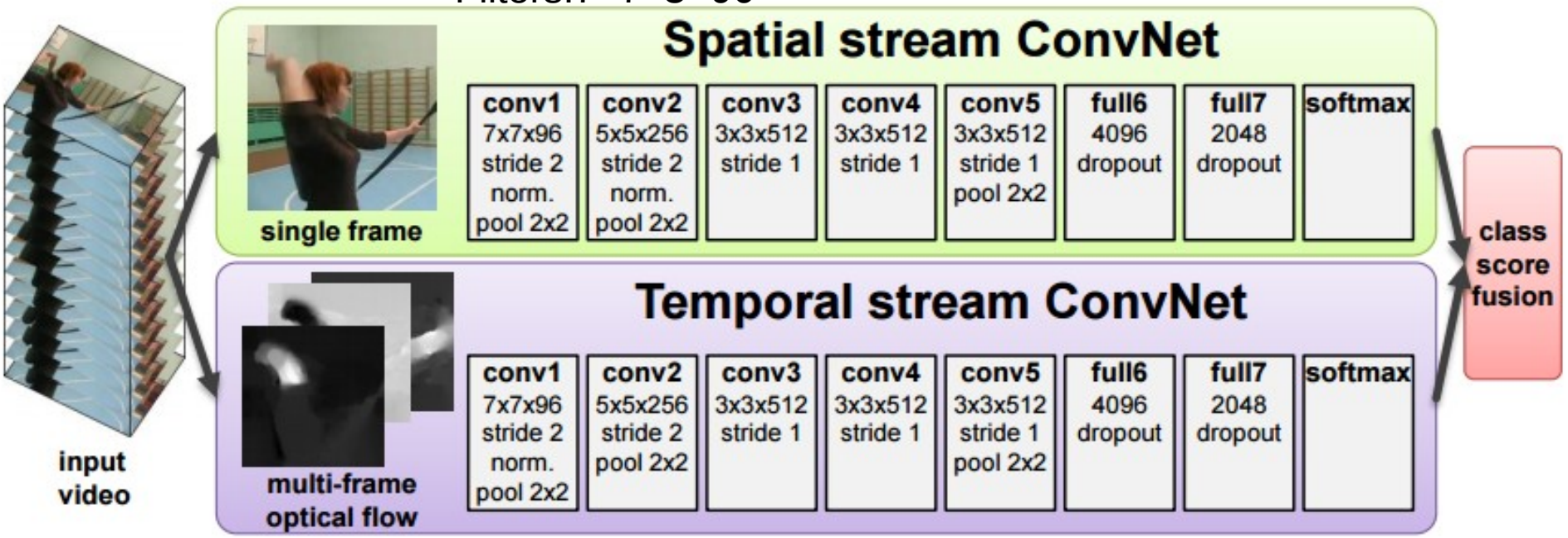
When applied to video recognition

- **Pros**
  - Proved to be great for image recognition (MNIST, CIFAR, ImageNet, etc.)
  - Networks trained on ImageNet or other huge datasets can be easily used to predict labels of single frames
  - Relatively easy to set up using modern libraries (Matconvnet, Caffe, Theano, Torch, etc.) with built-in architectures
  - Can be trained on different features inherent to videos (e.g., optical flow, MPEG motion vectors)
- **Cons**
  - Video datasets are Small! (exception: Sports-1M)
  - Resource consuming (at least 6GB of GPU memory is required)

    and can take several days of training (however, prediction can be fast)

Two-Stream Convolutional Networks

[K. Simonyan, A. Zisserman; NIPS'14]



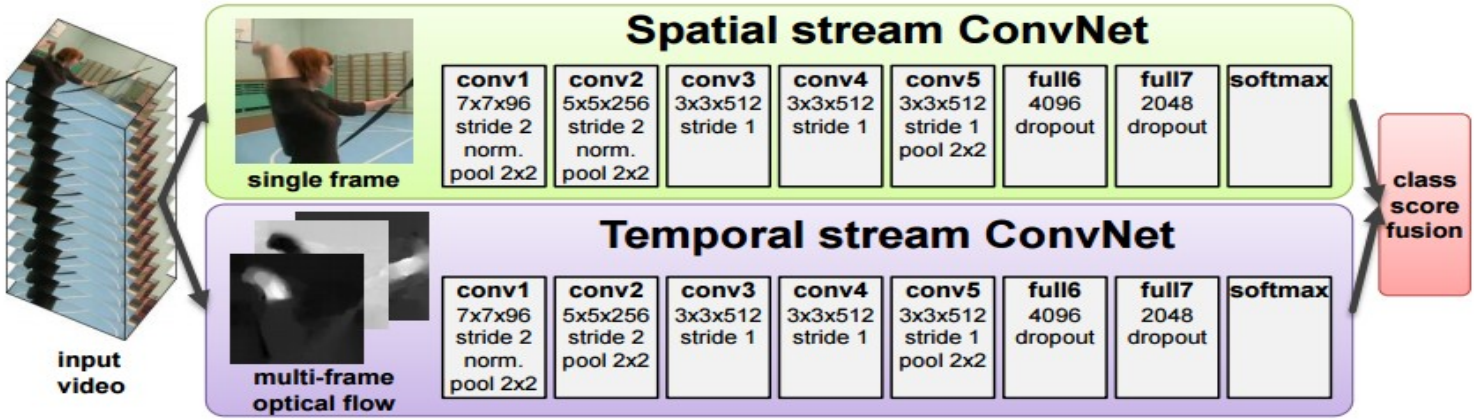
88.0% on UCF101

Spatial network: Pretrained Imagenet-vgg-m-2048 (13.5% top-5 error on ILSVRC-2012)
[K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman; BMVC'14]
Temporal network: same as spatial except for **conv1**.
High accuracy optical flow algorithm implemented on GPU in OpenCV
[T. Brox, A. Bruhn, N. Papenberg, J. Weickert; ECCV'04]

Two-Stream Convolutional Networks

Input: 224×224×***3***
Filters:7×7×***3***×96

[K. Simonyan, A. Zisserman; NIPS'14]



L optical flow frames
(*x*,*y* directions) → 2*L*

Input: 224×224×***2L***
Filters:7×7×***2L***×96

88.0% on UCF101

Spatial network: Pre-trained Imagenet-vgg-m-2048 (13.5% top-5 error on ILSVRC-2012)
Fine-tuning in case of UCF101 (about 1.7mln frames, label of frame = label of its video): **20K** iterations (1 iteration = 1 batch of 256 random 224×224 crops* of random frames), dropout = 0.5
Temporal network: same as spatial except for **conv1**
High accuracy optical flow algorithm implemented on GPU in OpenCV
[T. Brox, A. Bruhn, N. Papenberg, J. Weickert; ECCV'04]
Training from scratch in case of UCF101: **80K** iterations (1 iteration = 1 batch of 256 random frames), dropout = 0.9

Predict labels for both streams (UCF101): 25 frames with fixed interval, 10 crops as in *[A. Krizhevsky, I. Sutskever, and G. E. Hinton; NIPS'12], label of video=avg score of for frames

Two-Stream CNN: Results on UCF101

	UCF101 (split1), our results [1]	UCF101 (split1), other works [2]
Dense Trajectoris + FV	80.7	-
Dense Trajectoris with MPEG FLOW + FV	71.3	-
Improved Dense Trajectoris + FV	83.0 (84.3 all splits)	- (85.9 all splits)
Spatial CNN (from scratch, droupout = 0.5)		42.5
Spatial CNN (from scratch, droupout = 0.9)		52.3
Spatial CNN (Pre-trained+fine-tuning, droupout = 0.5)		70.8
Spatial CNN (Pre-trained + fine-tuning, droupout = 0.9)		**72.8**
Spatial CNN (Pre-trained + last layer, droupout = 0.5)	**70.7**	**72.7**
Spatial CNN (Pre-trained + last layer, droupout = 0.9)		59.9
Temporal CNN (optical flow L = 1, droupout = 0.9)		73.9
Temporal CNN (L= 5, droupout = 0.9), 20 patches		80.4
Temporal CNN (L=10, droupout = 0.9), 20 patches	77.5	**81.2** (81.0 all splits)
Temporal CNN (L=10, droupout = 0.9), 40 patches	**78.5**	- (80.1 all splits)

[1] Nvidia TITAN Black 6Gb (limit of 192 samples/batch for L=10), Framework: Matconvnet (Matlab)
[2] Nvidia TITAN -, Framework: Caffe

Two-Stream CNN: Training time on UCF101

	UCF101 (split1), our results [1]	UCF101 (split1), other works [2]	Training time (qualitative)
Dense Trajectoris with MPEG FLOW + FV	71.3	-	**4 hours, 12 core CPU**
Dense Trajectoris + FV	80.7	-	**20 hours, 12 core CPU**
Improved Dense Trajectoris + FV	83.0 (84.3 all)	- (85.9 all)	
Spatial CNN (Pre-trained + last layer, droupout = 0.5)	70.7	**72.7**	**10 hours training, 1 GPU[1] +SSD**
Temporal CNN (L=10, droupout = 0.9), 20 patches	77.5	0	**10 days training, 2 GPU[1] +SSD**
Temporal CNN (L=10, droupout = 0.9), 40 patches	78.5		

1 day training, 4 GPU[2] +SSD

Temporal CNN trained on MPEG motion vectors can be trained much faster, with a larger temporal receptive field, but leading to lower classification accuracy

[1] Nvidia TITAN Black 6Gb (limit of 192 images/batch for L=10), Framework: Matconvnet (Matlab)
[2] Nvidia TITAN -, Framework: Caffe
[K. Simonyan, A. Zisserman; NIPS'14]

Two-Stream CNN: Fusion on UCF101

	UCF101 (split1), our results	UCF101 (split1), other works
Dense Trajectoris with MPEG FLOW + FV	71.3	-
Dense Trajectoris + FV	80.7	-
Improved Dense Trajectoris + FV	83.0 (84.3 all)	- (85.9 all)
Spatial CNN (Pre-trained + last layer, droupout = 0.5)	70.7	**72.7**
Temporal CNN (L=10, droupout = 0.9), 20 patches	77.5[2]	**81.2** (81.0 all)
Temporal CNN (L=10, droupout = 0.9), 40 patches	78.5	- (80.1 all)
Spatial+Temporal CNN	82.7	87.0 (88.0 all)
Spatial+Temporal CNN + iDT	84.5	
Spatial+Temporal CNN + iDT + model weights	88.6	
TDD*		**90.3** all
TDD* + iDT		**91.5** all

*Trajectory-Pooled Deep-Convolutional Descriptors [L. Wang, Y. Qiao, X. Tang; CVPR'15]

Other datasets: Sports-1M

[A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei; CVPR'14]

- 1.2 million samples
- 487 classes of YouTube sports videos
  - ~1000-3000 videos/class
  - ~5% of the videos are multiclass
- Videos in the wild (unconstrained camera morions)
- Some videos are removing by users
  - → 1.1 million samples

Beyond Short Snippets: Deep Networks for Video Classification

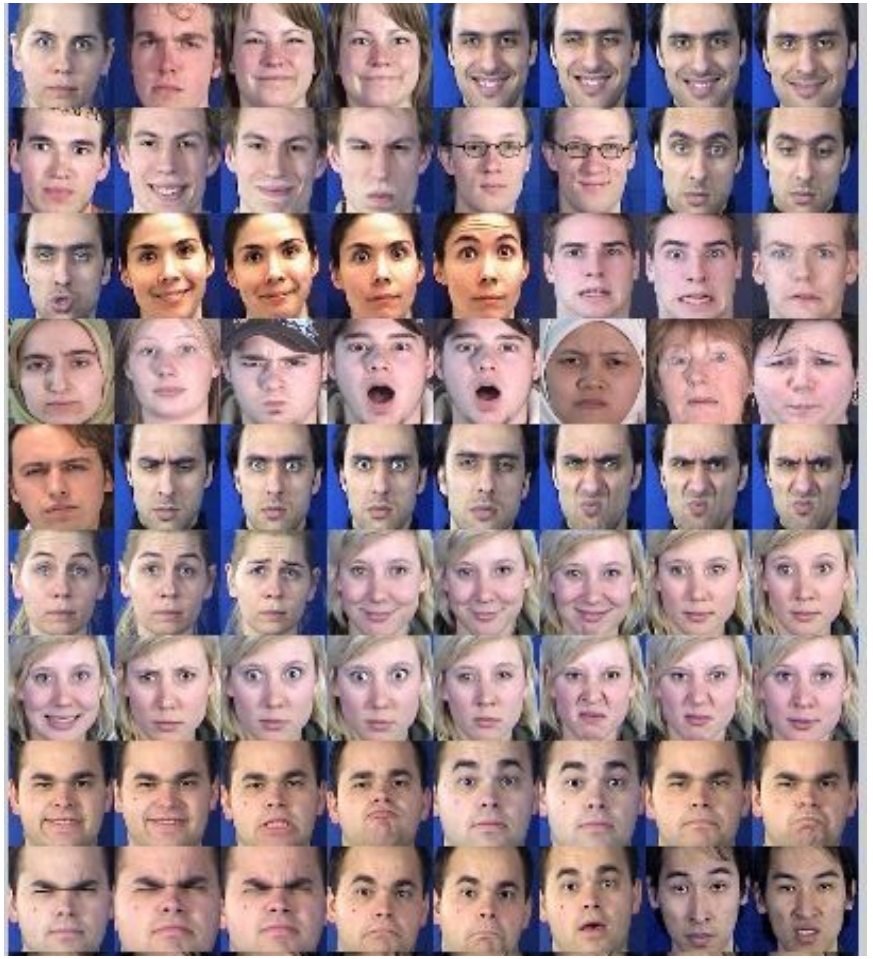
[J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, G. Toderici; CVPR'15]

- Recurrent neural network (Long Short Term Memory – LSTM) on the outputs of CNN
  - 90.5% (top5) on Sports-1M from 80.2% in [1]
  - 88.6% on UCF101 from 88% in [2]

[1] [A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, L. Fei-Fei; CVPR'14]
[2] [K. Simonyan, A. Zisserman; NIPS'14]

Other datasets: MMI

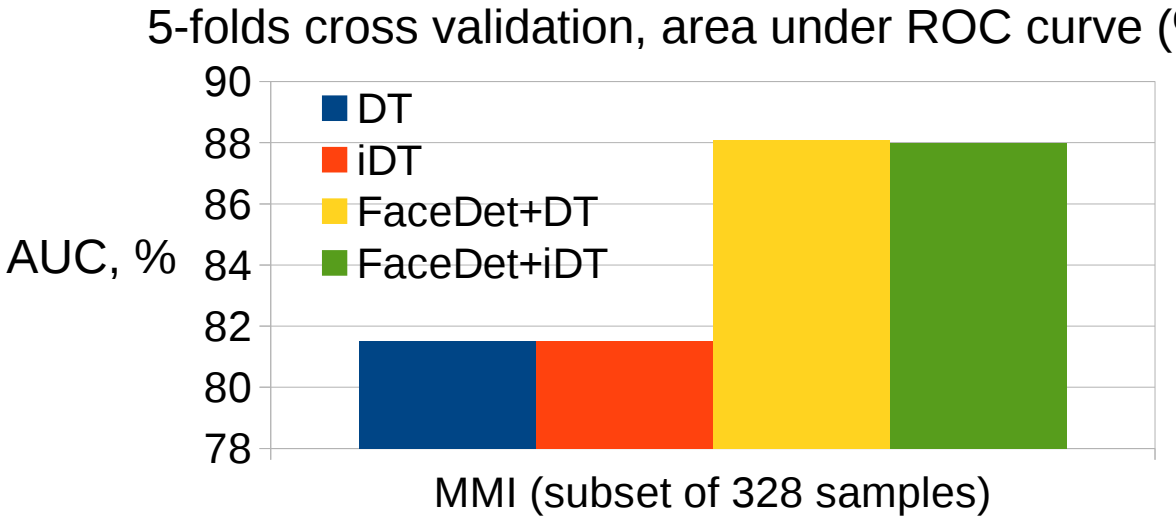
[M. Pantic, M. F. Valstar, R. Rademaker, L. Maat; ICME'05]

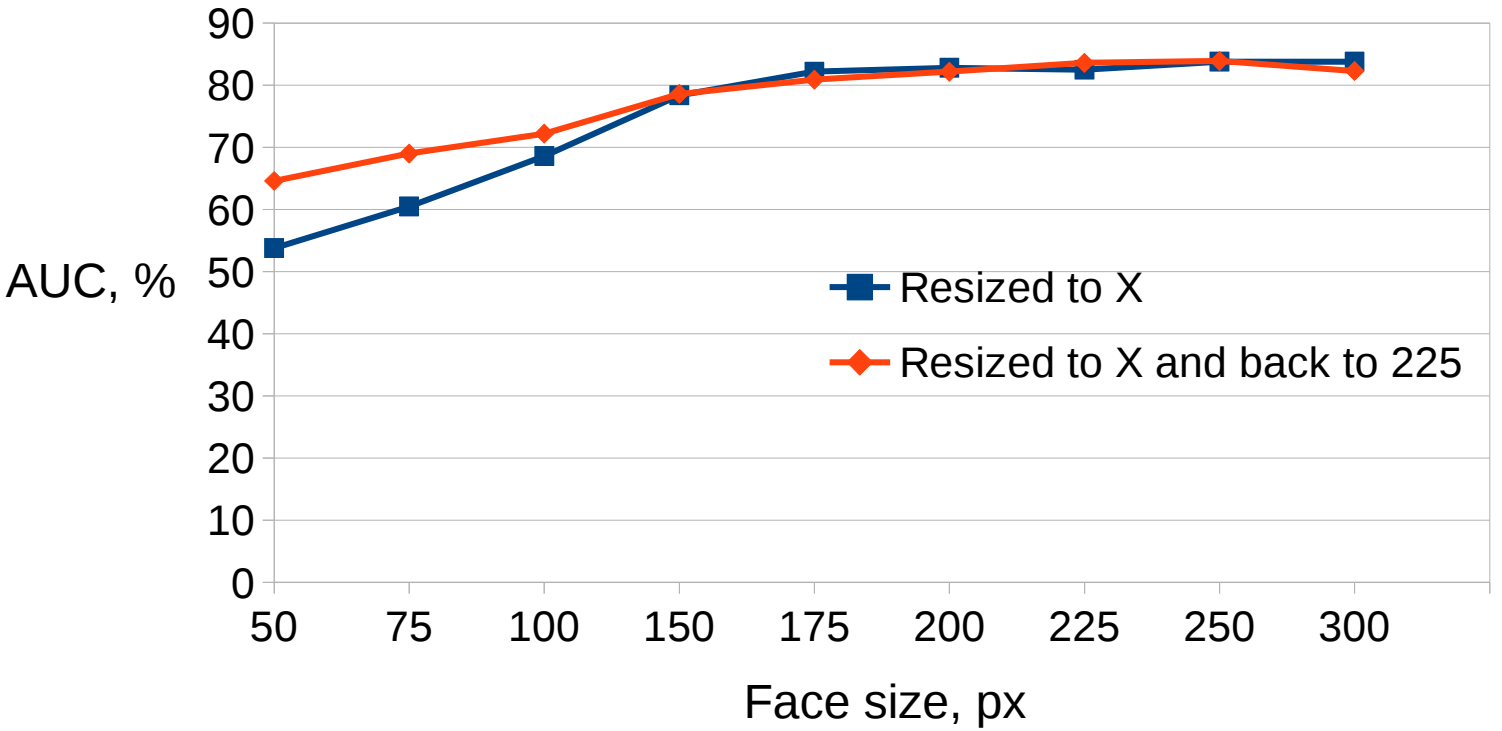


I get 1082 “correct” samples overall

- **Frame by frame annotation** for 328 video samples (~100 frames/sample)
- **Variaty** of action classses (34), gender, age, ethnicity, occlusions (glasses, hat, hairs)
- Many classes can present on one video
- **Frontal faces**
- Too few examples per class (10-20)
- Some samples are staged (professional actors)
- Some ambiguities and errors in annotations (partialy due to complexity of some classes)
- Difficult to compare with other works

MMI results


5-folds cross validation, area under ROC curve (%)
AUC, %
90
88
86
84
82
80
78
DT
iDT
FaceDet+DT
FaceDet+iDT
MMI (subset of 328 samples)


AUC, %
90
80
70
60
50
40
30
20
10
0
50
75
100
150
175
200
225
250
300
Resized to X
Resized to X and back to 225
Face size, px

Other datasets: Chalearn

S. Escalera, J. Gonzàlez, X. Baró, M. Reyes, O. Lopes, I. Guyon, V. Athistos, H.J. Escalante, "Multi-modal Gesture Recognition Challenge 2013: Dataset and Results", ICMI 2013.

(Imrpoved) Dense Trajectories can be applied to other sources of video data:
- infrared spectrum
- depth data
- etc.