



Boulder

Computer Vision; Video



[YouTube Playlist](#)

Maziar Raissi

Assistant Professor

Department of Applied Mathematics

University of Colorado Boulder

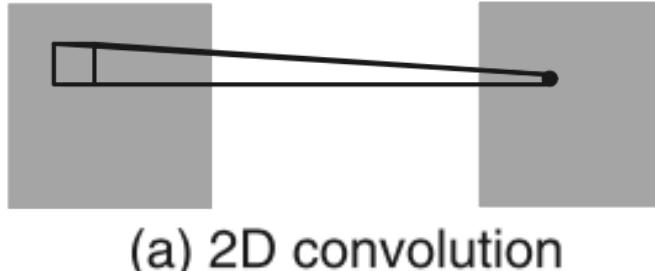
maziar.raissi@colorado.edu



Boulder

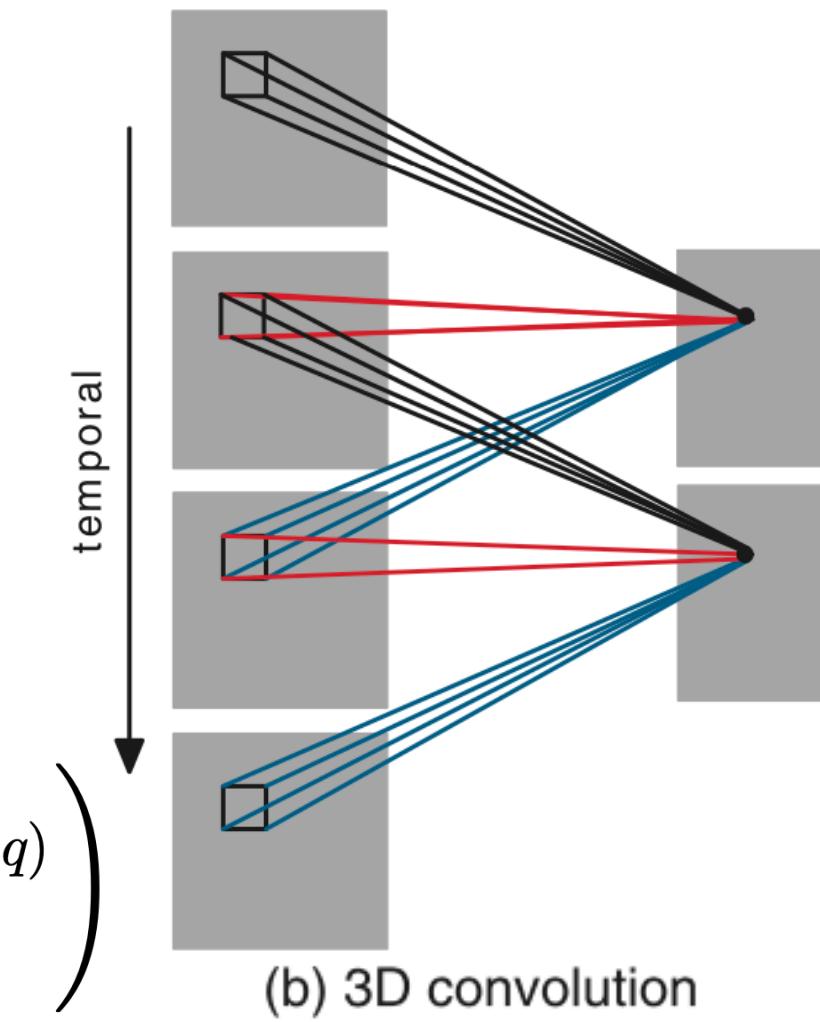
3D Convolutional Neural Networks for Human Action Recognition

- intelligent video surveillance
- customer attributes
- shopping behavior analysis



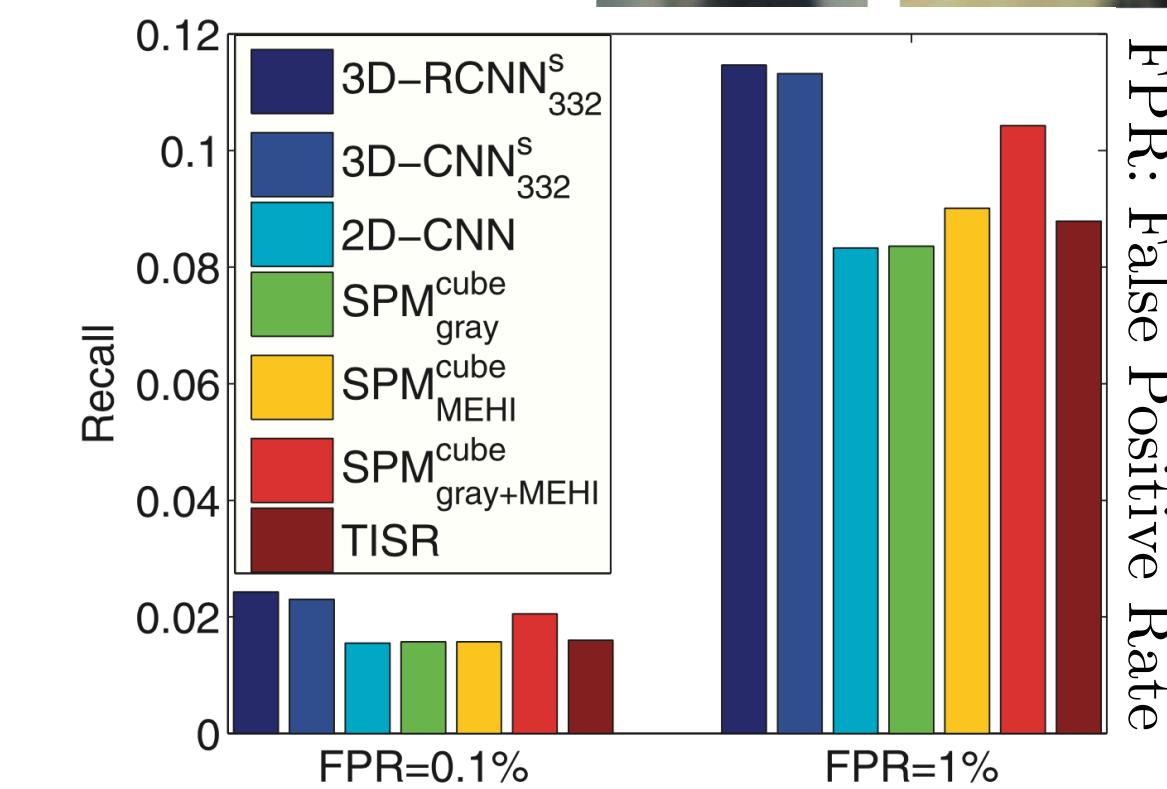
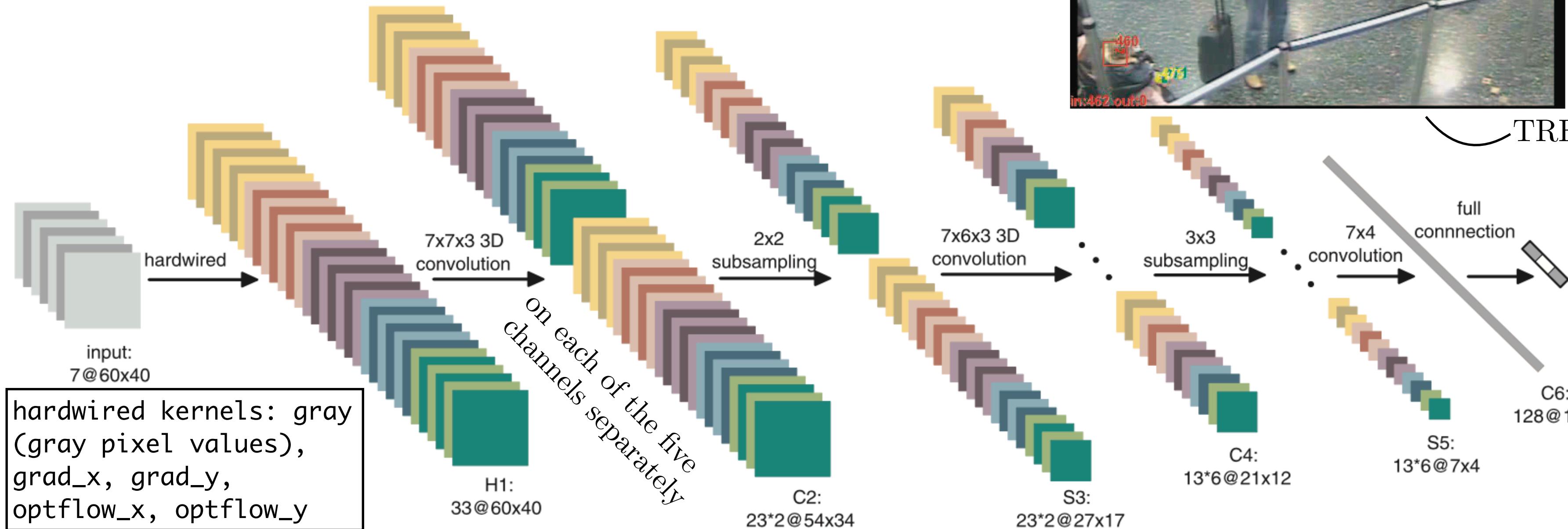
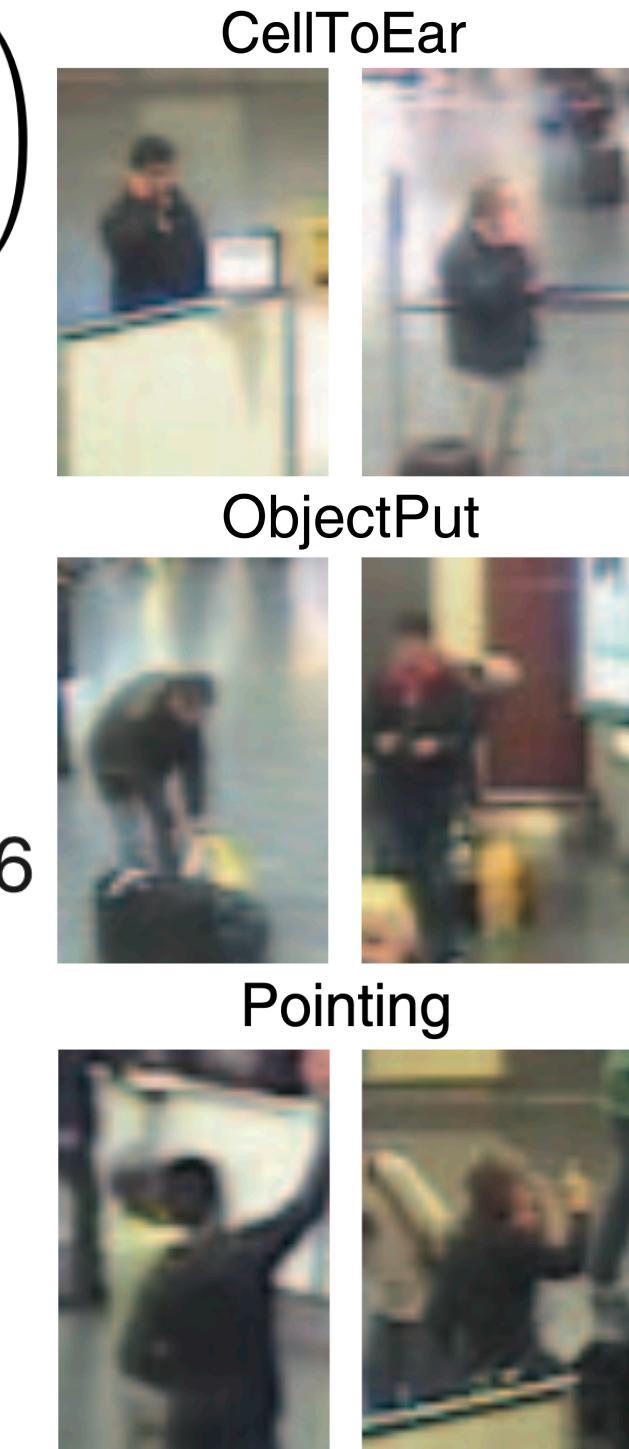
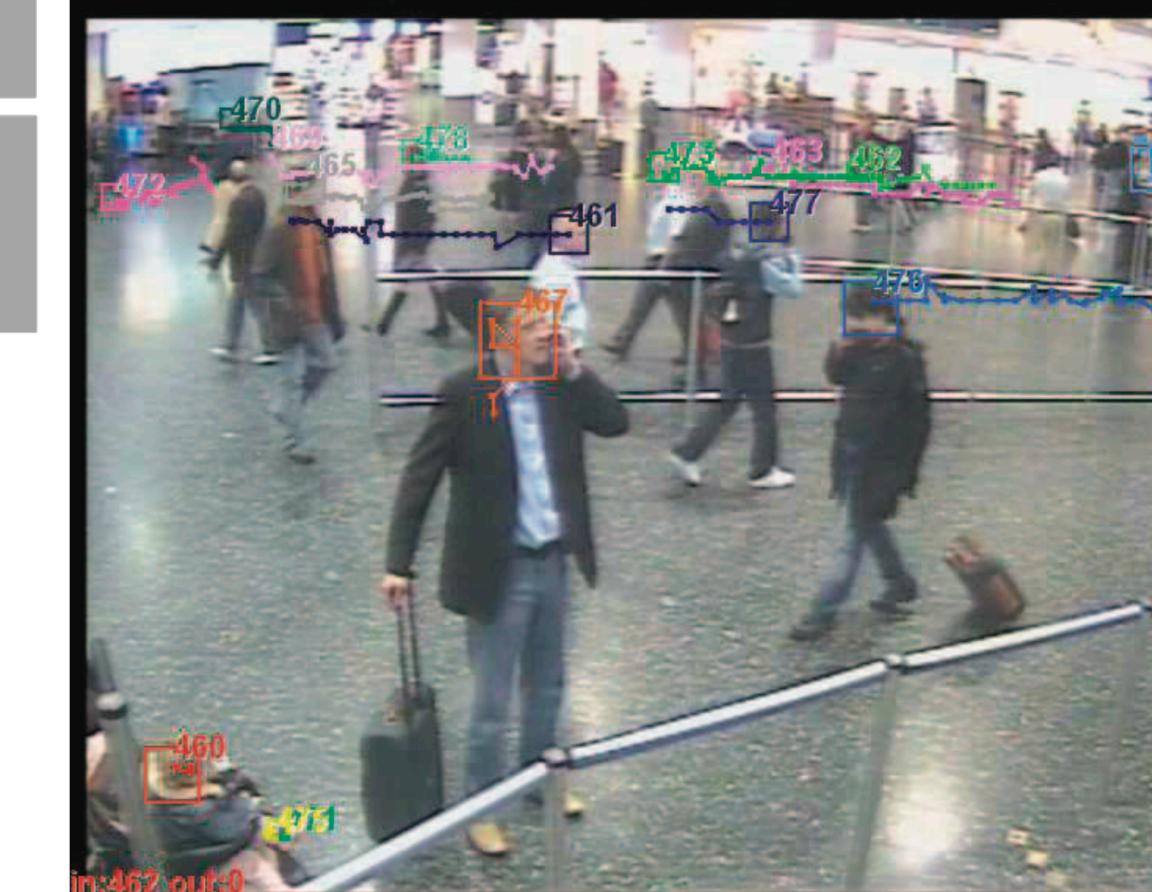
$v_{ij}^{xy} \rightarrow$ value at position (x, y) in the j -th feature map of the i -th layer

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



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

human detection and tracking





Boulder

Large-scale Video Classification with Convolutional Neural Networks



[YouTube Video](#)

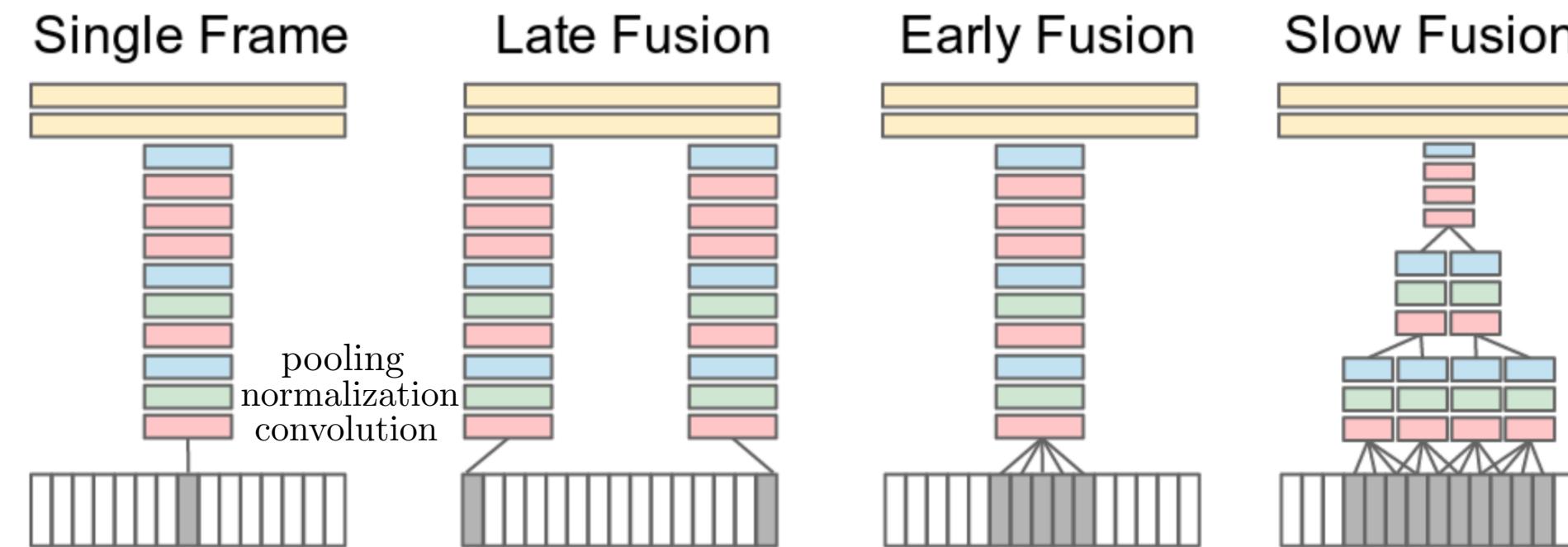
- search
- summarization

Sports-1M dataset: Consists of 1 million YouTube videos belonging to a taxonomy of 487 classes of sports

Context stream: Learns features on low-resolution frames

Fovea stream: Operates on the high-resolution middle portion of the frame

- UCF-101 dataset \leftarrow Transfer Learning
treat every video as a bag of short, fixed-sized clips
each clip contains several contiguous frames in time



Single Frame
 $170 \times 170 \times 3 \rightarrow$ input size
 $C(96, 11, 3) \triangleright N \triangleright P \triangleright C(256, 5, 1) \triangleright N \triangleright P \triangleright C(384, 3, 1) \triangleright C(384, 3, 1) \triangleright C(256, 3, 1) \triangleright P \triangleright FC(4096) \triangleright FC(4096)$
 $P \rightarrow 2 \times 2$ non-overlapping pooling
 $N \rightarrow$ Local Response Normalization

Early Fusion

$11 \times 11 \times 3 \times T \rightarrow$ filters on the first convolutional layer

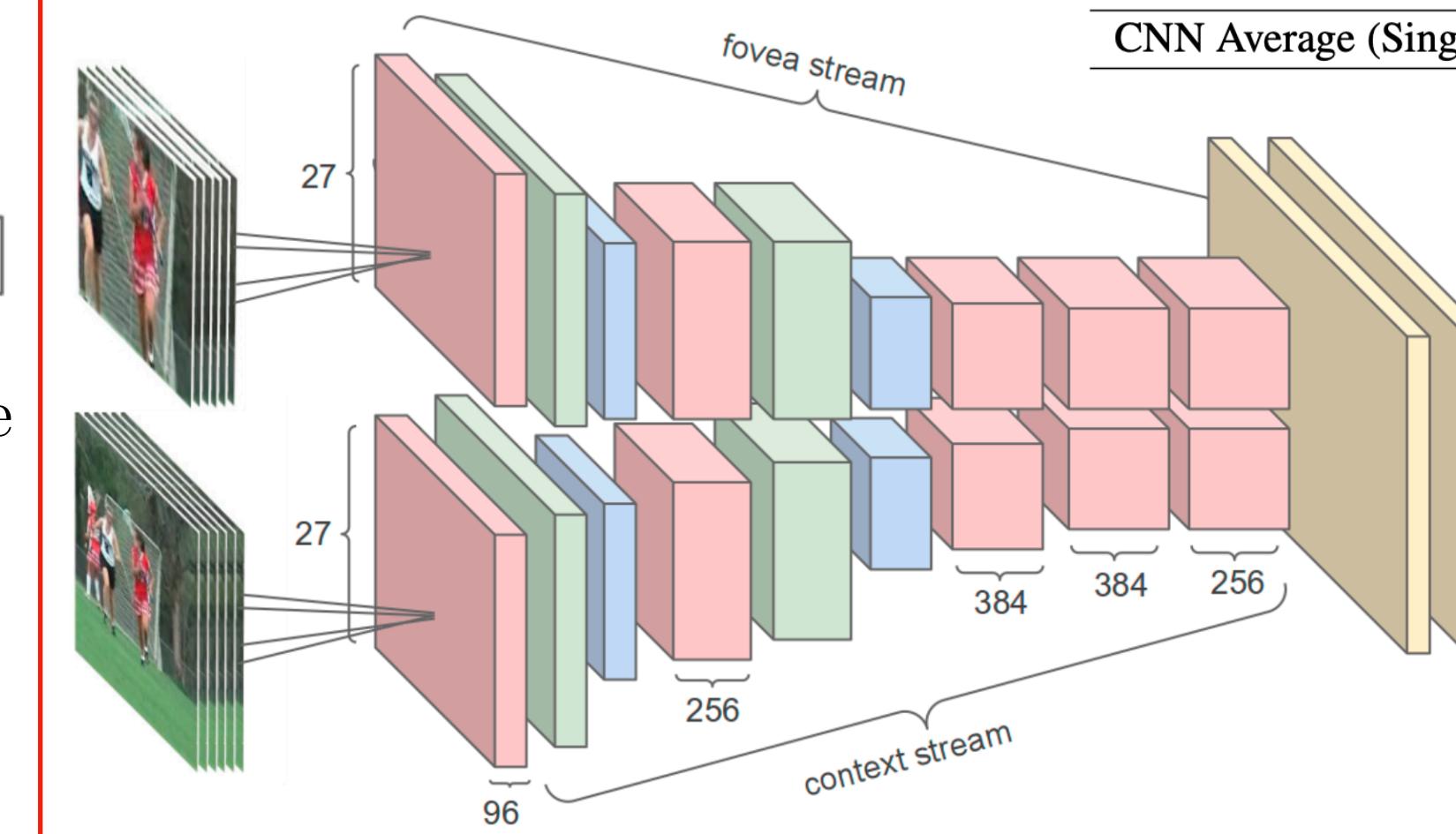
$T = 10$ ($\approx 1/3$ of a second)

Late Fusion

Place two separate single-frame networks up to last convolutional layer $C(256, 3, 1)$
with shared parameters a distance of 15 frames apart and then merge the two streams in the first fully connected layer

Multi-resolution CNNs

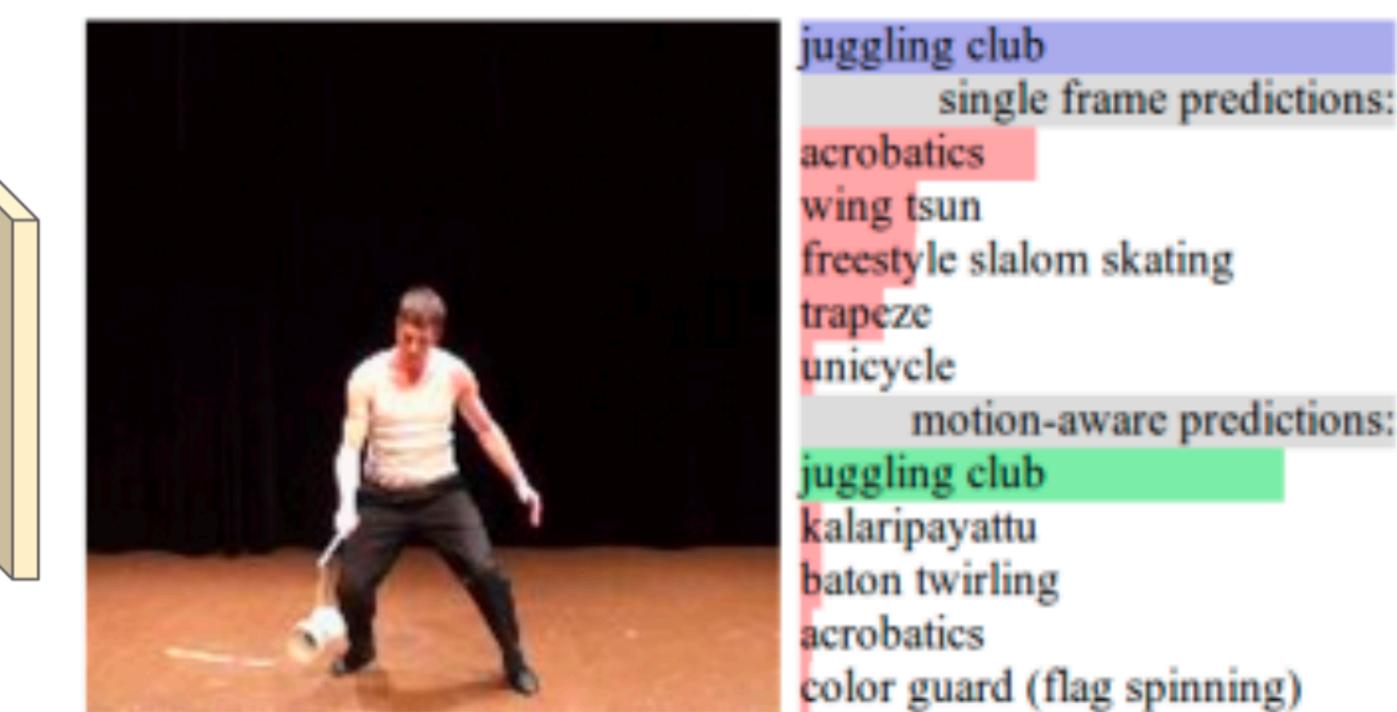
(context stream)
 $178 \times 178 \xrightarrow{27}$
 89×89 downsampled
 89×89 center crop
(fovea stream)



Slow Fusion

A balanced mix between the two approaches

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





Boulder

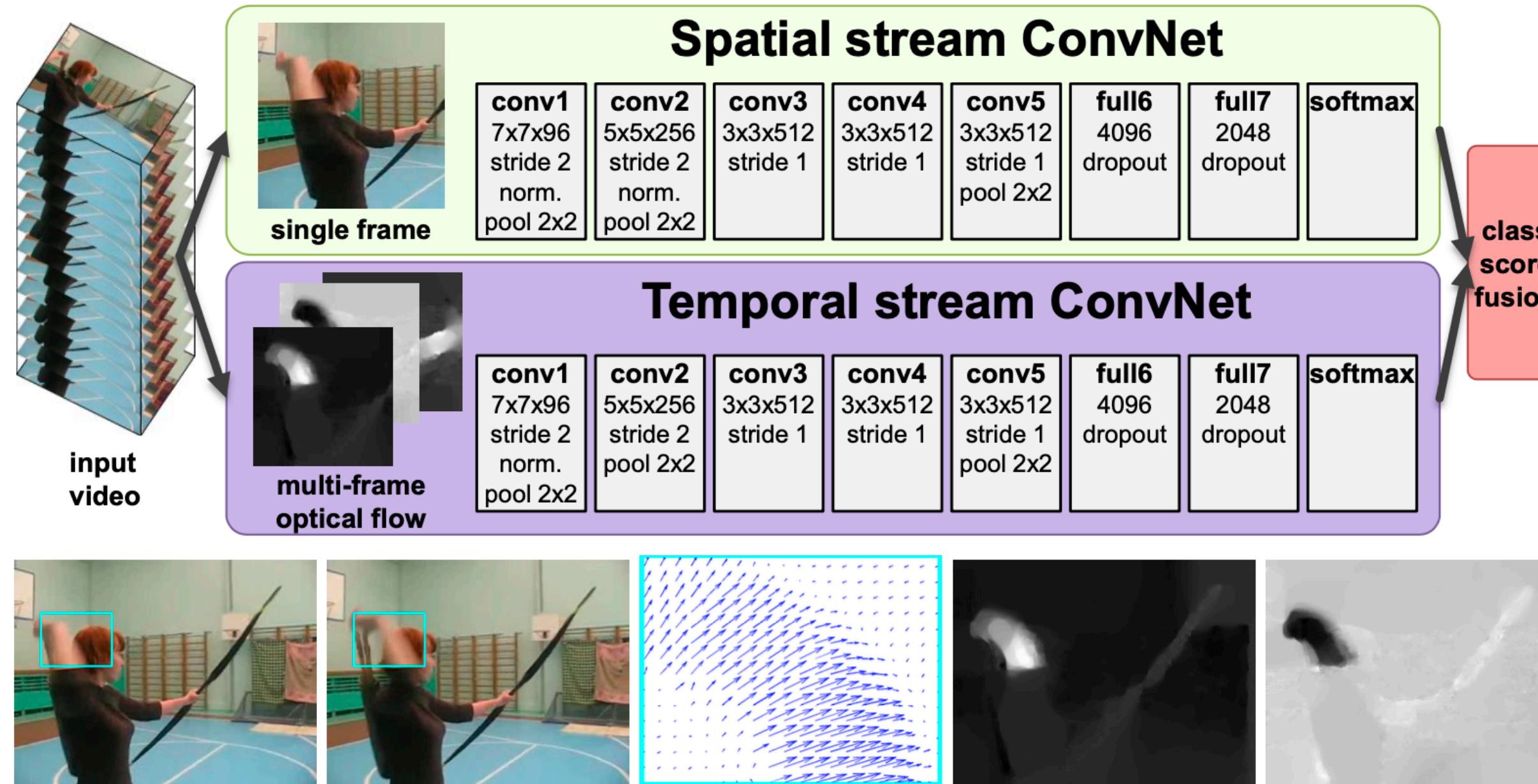
Two-Stream Convolutional Networks for Action Recognition in Videos



[YouTube Video](#)

- UCF-101
- HMDB-51

Video can naturally be decomposed into spatial and temporal components



A dense optical flow can be seen as a set of displacement vector fields d_t between the pairs of consecutive frames t and $t + 1$

$d_t(u, v) \rightarrow$ displacement vector at the point (u, v)

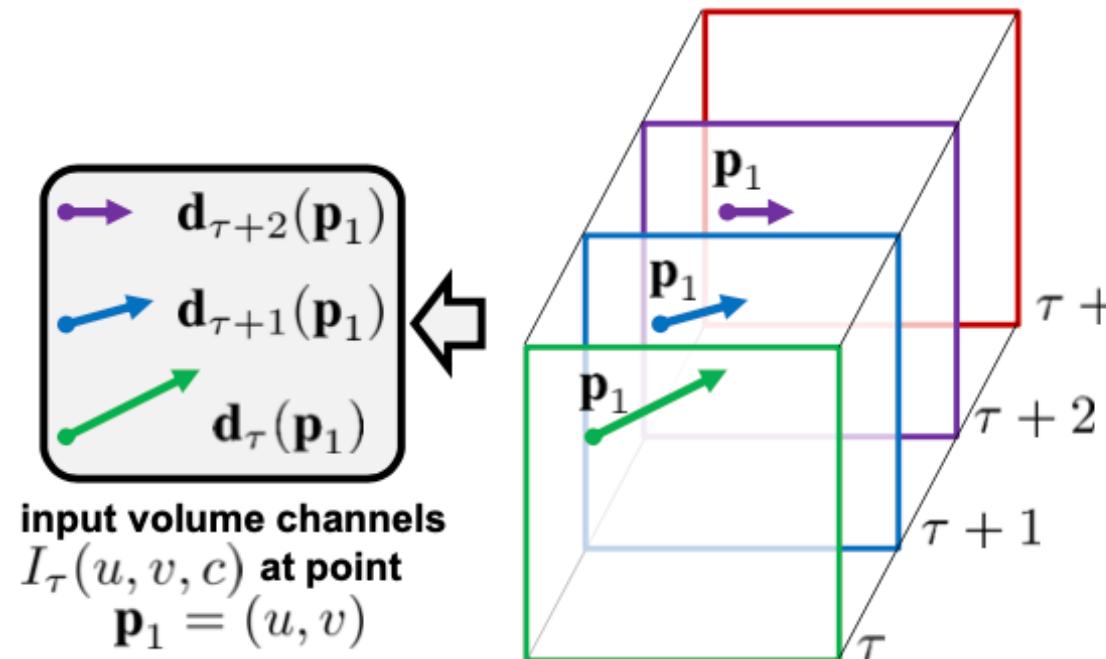
moves the point to the corresponding point in the following frame $t + 1$

$d_t^x, d_t^y \rightarrow$ horizontal and vertical components of the vector field

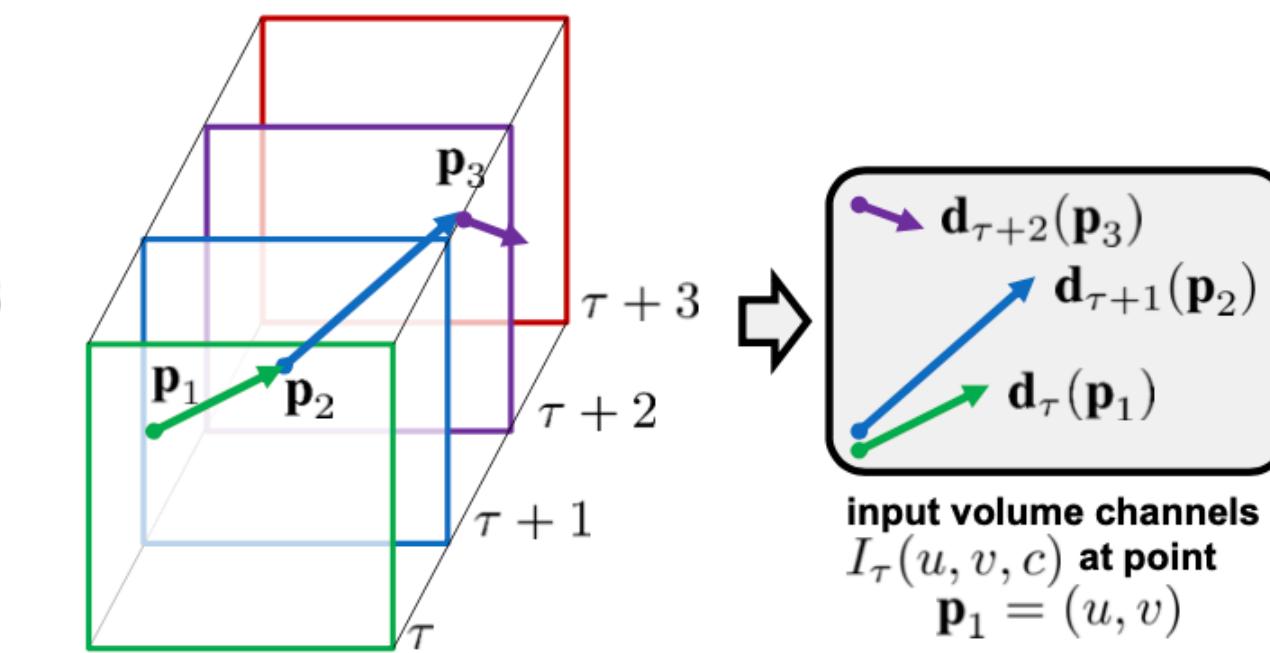
Optical flow stacking

Stack the flow channels d_t^x, d_t^y of L consecutive frames to form a total of $2L$ input channels

$$\begin{aligned}\tau &\rightarrow \text{an arbitrary frame} \\ I_\tau &\in \mathbb{R}^{w \times h \times 2L} \rightarrow \text{input volume} \\ I_\tau(u, v, 2k - 1) &= d_{\tau+k-1}^x(\mathbf{p}_k) \\ I_\tau(u, v, 2k) &= d_{\tau+k-1}^y(\mathbf{p}_k)\end{aligned}$$



$$\begin{aligned}\mathbf{p}_k &= \mathbf{p}_{k-1} + \mathbf{d}_{\tau+k-2}(\mathbf{p}_{k-1}), k > 1 \\ I_\tau(u, v, 2k - 1) &= d_{\tau+k-1}^x(\mathbf{p}_k) \\ I_\tau(u, v, 2k) &= d_{\tau+k-1}^y(\mathbf{p}_k)\end{aligned}$$



Bi-directional optical flow forward & backward flows

Training setting	Dropout ratio		Spatial ConvNet
	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%	

Temporal ConvNet	Input configuration		Mean subtraction	
	off	on		
Single-frame optical flow ($L = 1$)	-	73.9%		
Optical flow stacking (1) ($L = 5$)	-	80.4%		
Optical flow stacking (1) ($L = 10$)	79.9%	81.0%		
Trajectory stacking (2) ($L = 10$)	79.6%	80.2%		
Optical flow stacking (1) ($L = 10$), bi-dir.	-	81.2%		

Method	UCF-101	HMDB-51
Improved dense trajectories (IDT) [26, 27]	85.9%	57.2%
IDT with higher-dimensional encodings [20]	87.9%	61.1%
IDT with stacked Fisher encoding [21] (based on Deep Fisher Net [23])	-	66.8%
Spatio-temporal HMAX network [11, 16]	-	22.8%
“Slow fusion” spatio-temporal ConvNet [14]	65.4%	-
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%



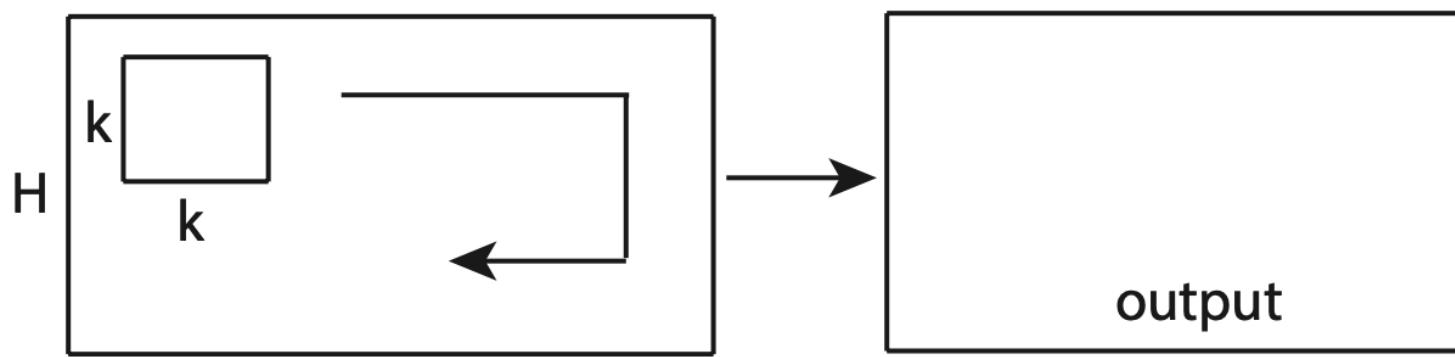
Boulder

Learning Spatiotemporal Features with 3D Convolutional Networks

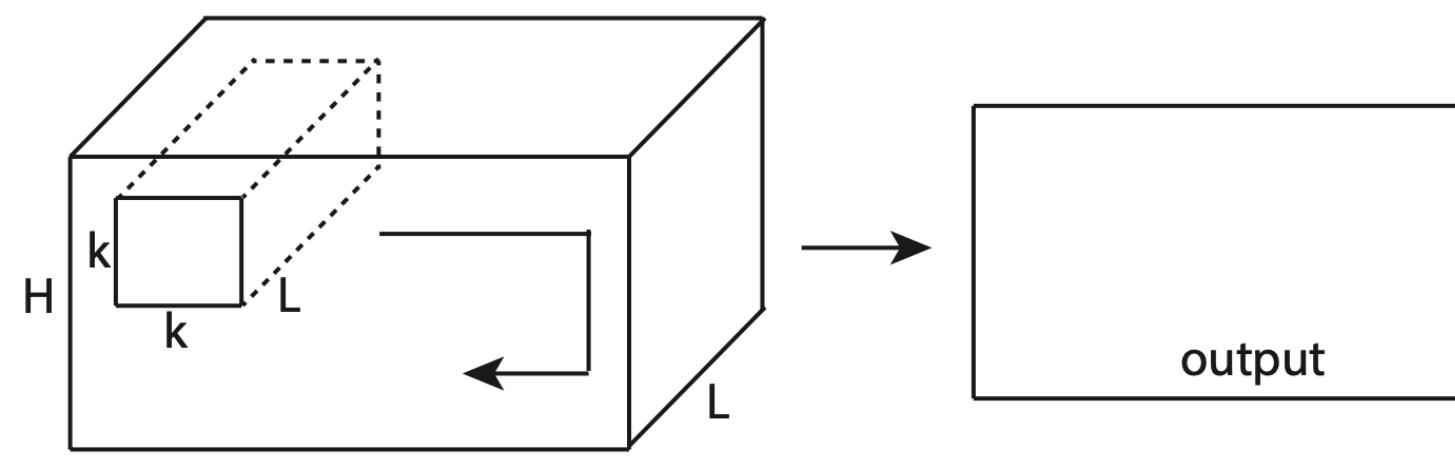


[YouTube Video](#)

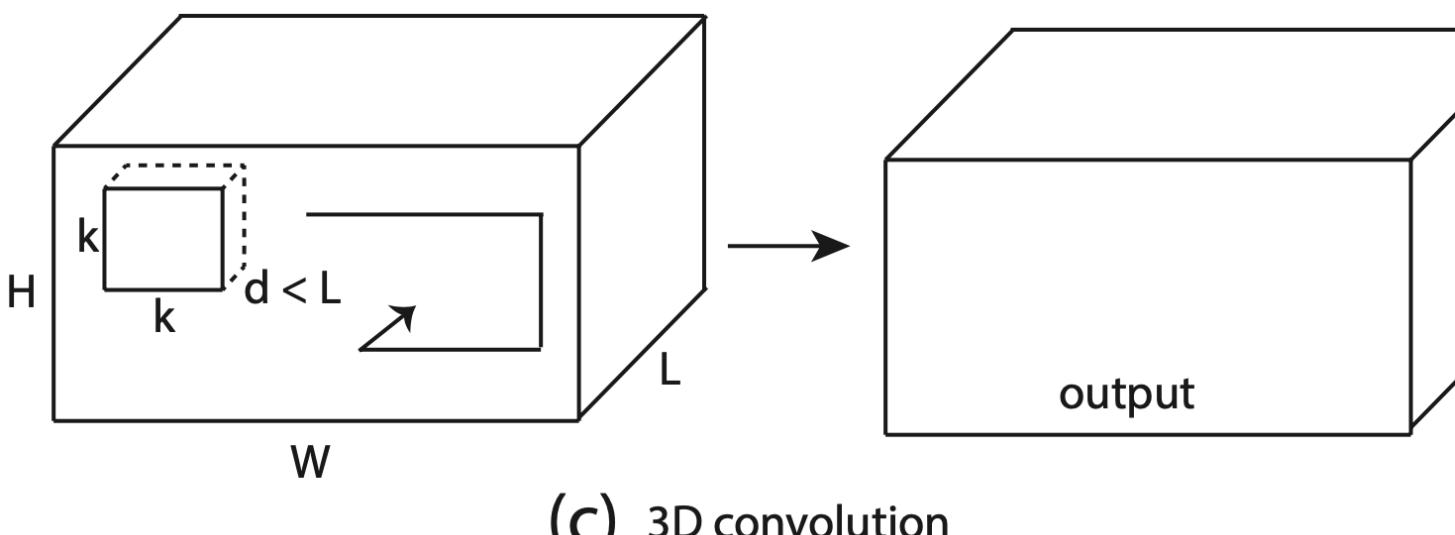
- search – recommendation – ranking
- action recognition
- abnormal event detection (anomaly detection)
- activity understanding
- video retrieval
- event & action detection



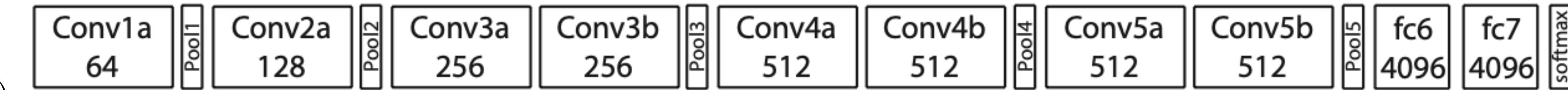
(a) 2D convolution



(b) 2D convolution on multiple frames



(c) 3D convolution



All 3D convolution kernels are $3 \times 3 \times 3$

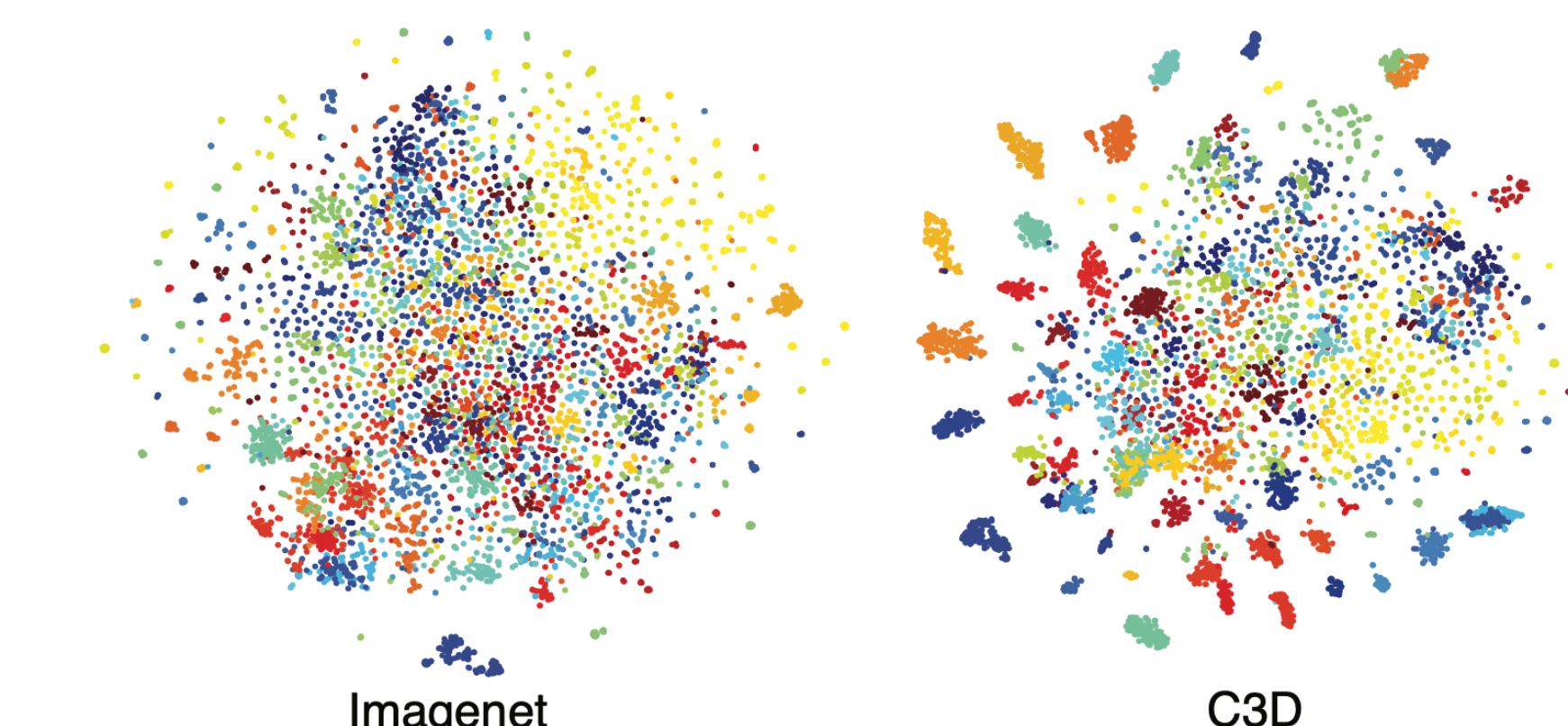
Dataset Task	Sport1M action recognition	UCF101 action recognition	ASLAN action similarity labeling	YUPENN scene classification	UMD scene classification	Object object recognition
Method	[29]	[39]([25])	[31]	[9]	[9]	[32]
Result	90.8	75.8 (89.1)	68.7	96.2	77.7	12.0
C3D	85.2	85.2 (90.4)	78.3	98.1	87.7	22.3

RGB



All possible features (e.g. optical flow, improved Dense Trajectory)

iDT



Imagenet

C3D

Each clip is visualized as a point

Method Usage	iDT CPU	Brox's CPU	Brox's GPU	C3D GPU
RT (hours)	202.2	2513.9	607.8	2.2
FPS	3.5	0.3	1.2	313.9
x Slower	91.4	1135.9	274.6	1

Runtime analysis on UCF101

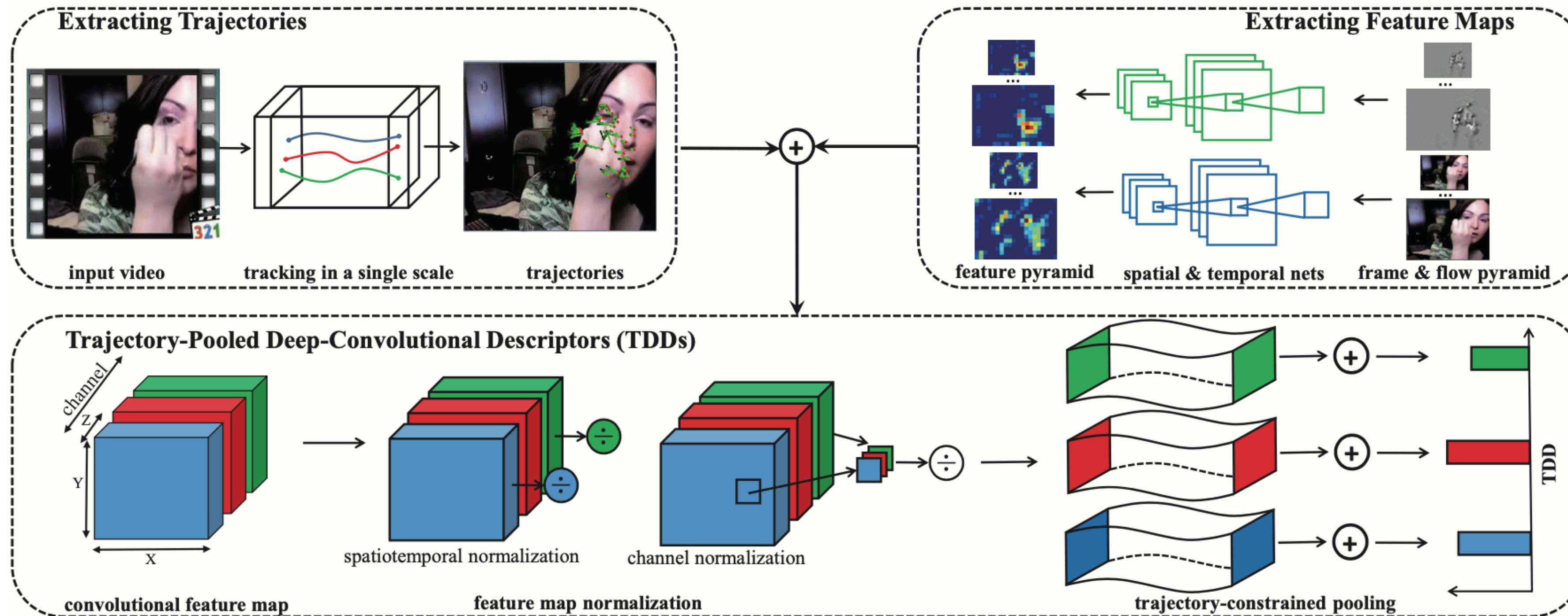
Method	Accuracy (%)
Imagenet + linear SVM	68.8
iDT w/ BoW + linear SVM	76.2
Deep networks [18]	65.4
Spatial stream network [36]	72.6
LRCN [6]	71.1
LSTM composite model [39]	75.8
C3D (1 net) + linear SVM	82.3
C3D (3 nets) + linear SVM	85.2
iDT w/ Fisher vector [31]	87.9
Temporal stream network [36]	83.7
Two-stream networks [36]	88.0
LRCN [6]	82.9
LSTM composite model [39]	84.3
Conv. pooling on long clips [29]	88.2
LSTM on long clips [29]	88.6
Multi-skip feature stacking [25]	89.1
C3D (3 nets) + iDT + linear SVM	90.4

Action recognition results on UCF101



Boulder

Action Recognition with Trajectory-Pooled Deep-Convolutional Descriptors



Algorithm	HMDB51	UCF101
HOG [31, 32]	40.2%	72.4%
HOF [31, 32]	48.9%	76.0%
MBH [31, 32]	52.1%	80.8%
HOF+MBH [31, 32]	54.7%	82.2%
iDT [31, 32]	57.2%	84.7%
Spatial net [24]	40.5%	73.0%
Temporal net [24]	54.6%	83.7%
Two-stream ConvNets [24]	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%

– video surveillance – human computer interaction – video content analysis

Trajectory-pooled Deep-convolutional Descriptor (TDD)

$V \rightarrow$ video

$T(V) = \{T_1, \dots, T_K\} \rightarrow$ set of trajectories

$T_k = \{(x_1^k, y_1^k, z_1^k), \dots, (x_P^k, y_P^k, z_P^k)\} \rightarrow$ trajectory k

$C(V) = \{C_1^s, \dots, C_M^s, C_1^t, \dots, C_M^t\} \rightarrow$ convolutional feature maps (Two-Stream CNNs)

$C_m^s \in \mathbb{R}^{H_m \times W_m \times L \times N_m} \rightarrow$ m -th layer feature map of spatial network

$C_m^t \in \mathbb{R}^{H_m \times W_m \times L \times N_m} \rightarrow$ m -th layer feature map of temporal network

$L \rightarrow$ duration of the video, $N_m \rightarrow$ number of channels

Trajectory Pooling

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

$r_m \rightarrow$ map size ratio with respective to input size

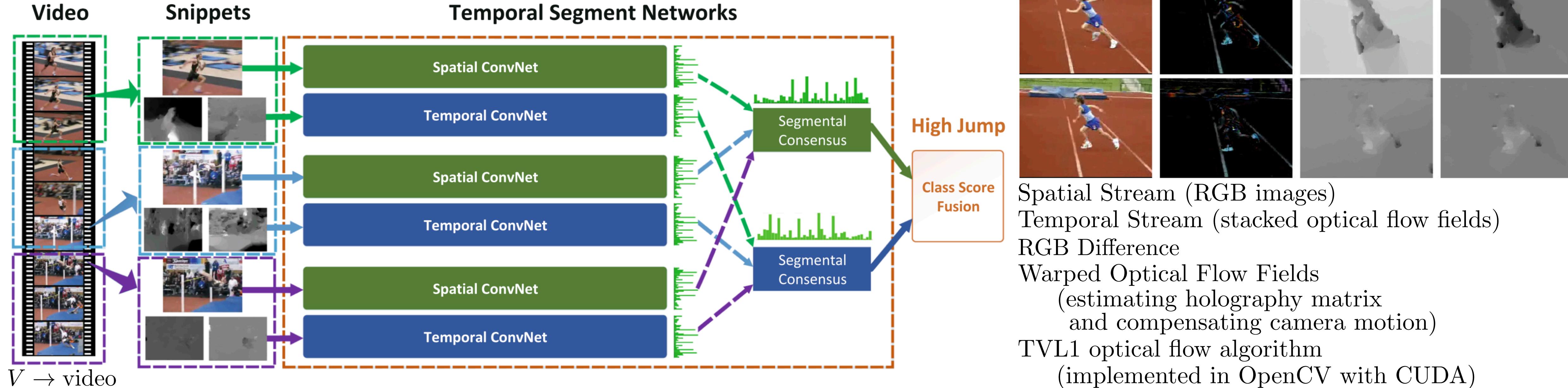
$(\cdot) \rightarrow$ rounding operation spatiotemporal normalization

$$\begin{aligned} \tilde{C}_m^a(x, y, z, n) &= C_m^a(x, y, z, n) / \max_{x, y, z} C_m^a(x, y, z, n) \\ \tilde{C}_m^a(x, y, z, n) &= C_m^a(x, y, z, n) / \max_n C_m^a(x, y, z, n) \end{aligned}$$

Temporal Segment Networks: Towards Good Practices for Deep Action Recognition

Security and behavior analysis

An extension of the two-stream convolutional networks to model long range temporal structure



$\{S_1, S_2, \dots, S_K\} \rightarrow K = 3$ segments of equal durations

$(T_1, T_2, \dots, T_K) \rightarrow \text{sequence of snippets}$

$T_k \rightarrow \text{snippet randomly sampled from its corresponding segment } S_k$

$\text{TSN}(T_1, T_2, \dots, T_K) = \mathcal{H}(\mathcal{G}(\mathcal{F}(T_1; \mathbf{W}), \mathcal{F}(T_2; \mathbf{W}), \dots, \mathcal{F}(T_K; \mathbf{W})))$

$\mathcal{F}(T_k; \mathbf{W}) \rightarrow \text{ConvNet (outputs class scores for all classes)}$

$\mathcal{G} \rightarrow \text{segmental consensus function}$

$\mathcal{H} \rightarrow \text{prediction function (softmax)}$

$\mathbf{G} = \mathcal{G}(\mathcal{F}(T_1; \mathbf{W}), \mathcal{F}(T_2; \mathbf{W}), \dots, \mathcal{F}(T_K; \mathbf{W}))$

$G_i = g(\mathcal{F}_i(T_1), \dots, \mathcal{F}_i(T_K)) \rightarrow \text{aggregation function (averaging)}$

$$\mathcal{L}(y, \mathbf{G}) = - \sum_{i=1}^C y_i \left(G_i - \log \sum_{j=1}^C \exp G_j \right)$$

UCF101 dataset (split 1)

Component	Basic Two-stream [1]	Cross-Modality pre-training
Accuracy	90.0 %	91.5
Partial BN with dropout		Temporal segment networks
92.0 %		93.5 %

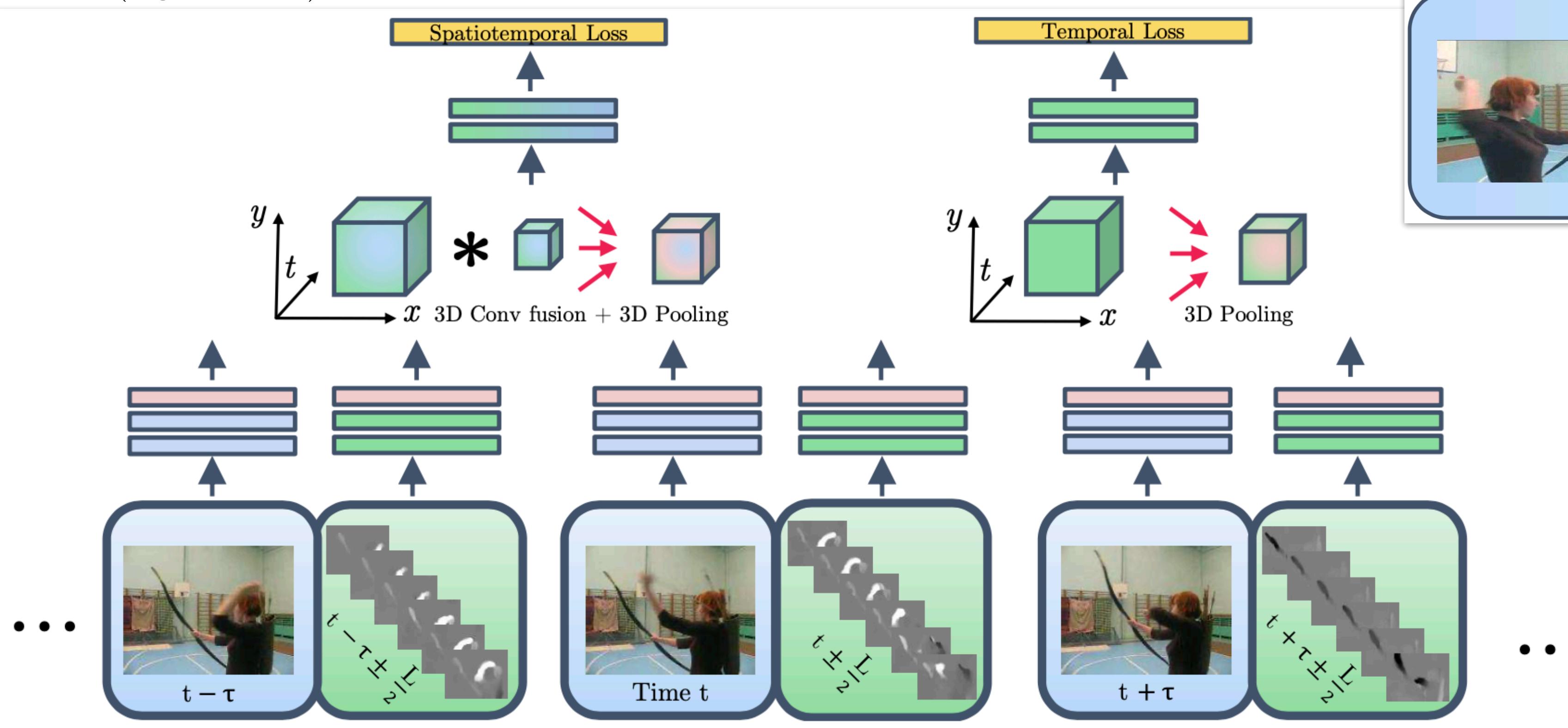


Boulder

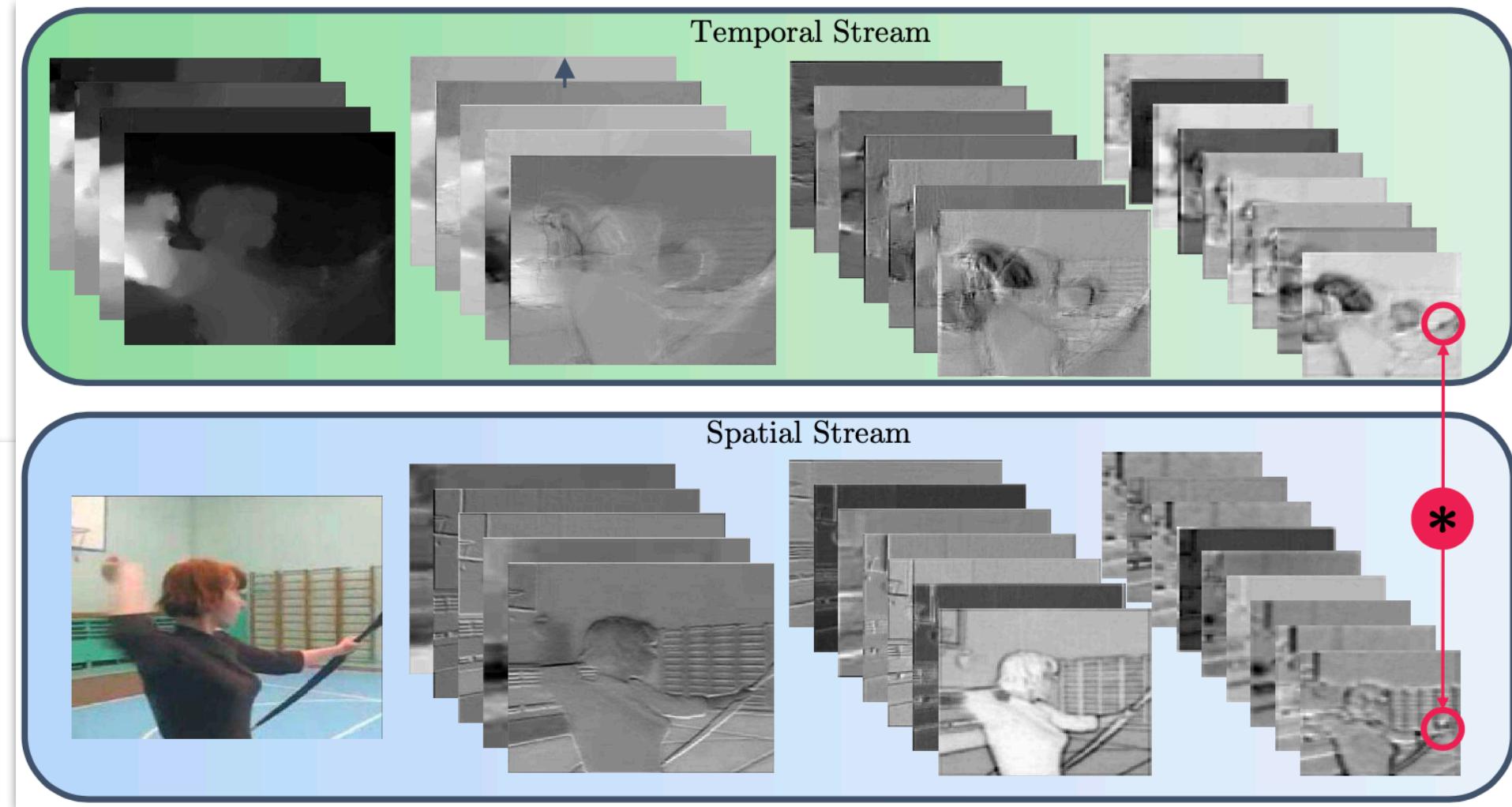
Convolutional Two-Stream Network Fusion for Video Action Recognition

Two main drawbacks of the two-stream architecture:

- unable learn the pixel-wise correspondences between spatial and temporal features (since fusion is only on the classification scores)
- limited in temporal scale as the spatial ConvNet operates only on single frames and the temporal ConvNet only on a stack of L temporally adjacent optical flow frames (e.g. $L = 10$).



$TL \rightarrow$ total temporal receptive field



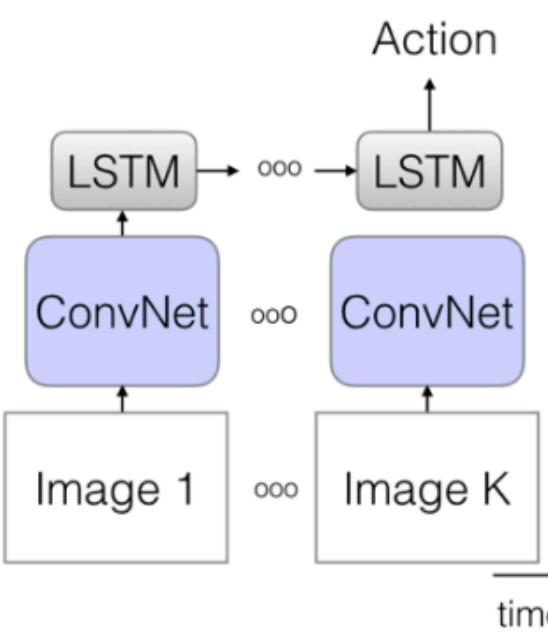
- How to fuse the two networks?
- Where to fuse the networks?
- How to fuse the networks temporally?

Method	UCF101	HMDB51
Spatiotemporal ConvNet [11]	65.4%	-
LRCN [6]	82.9%	-
Composite LSTM Model [25]	84.3%	44.0
C3D [30]	85.2%	-
Two-Stream ConvNet [22]	88.0%	59.4%
Factorized ConvNet [26]	88.1%	59.1%
Two-Stream Conv Pooling [17]	88.2%	-
Ours (S:VGG-16, T:VGG-M)	90.8%	62.1%
Ours (S:VGG-16, T:VGG-16, single tower after fusion)	91.8%	64.6%
Ours (S:VGG-16, T:VGG-16)	92.5%	65.4%

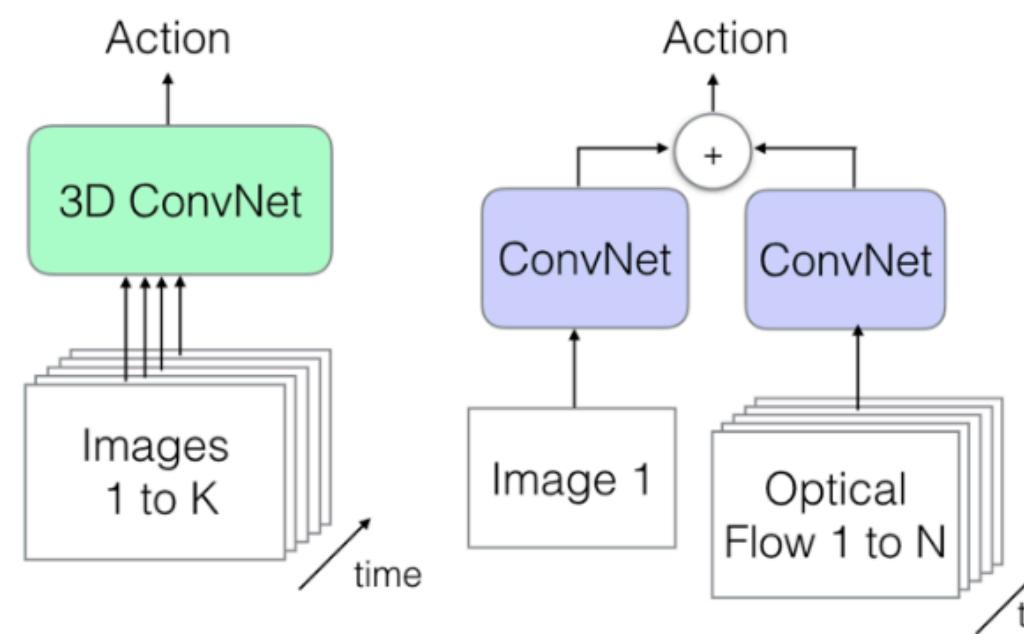
Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset


[YouTube Video](#)

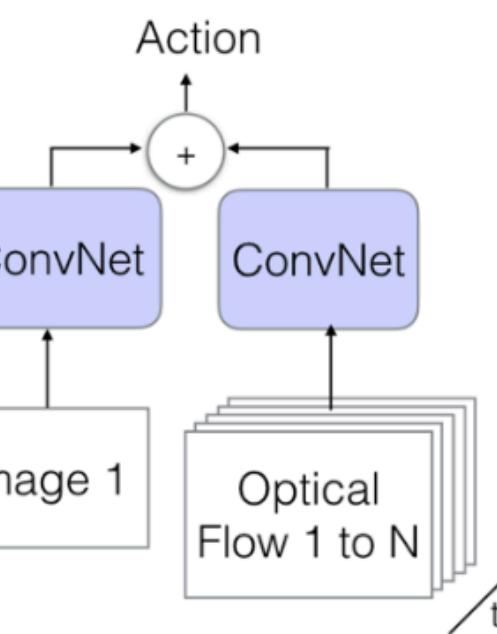
a) LSTM



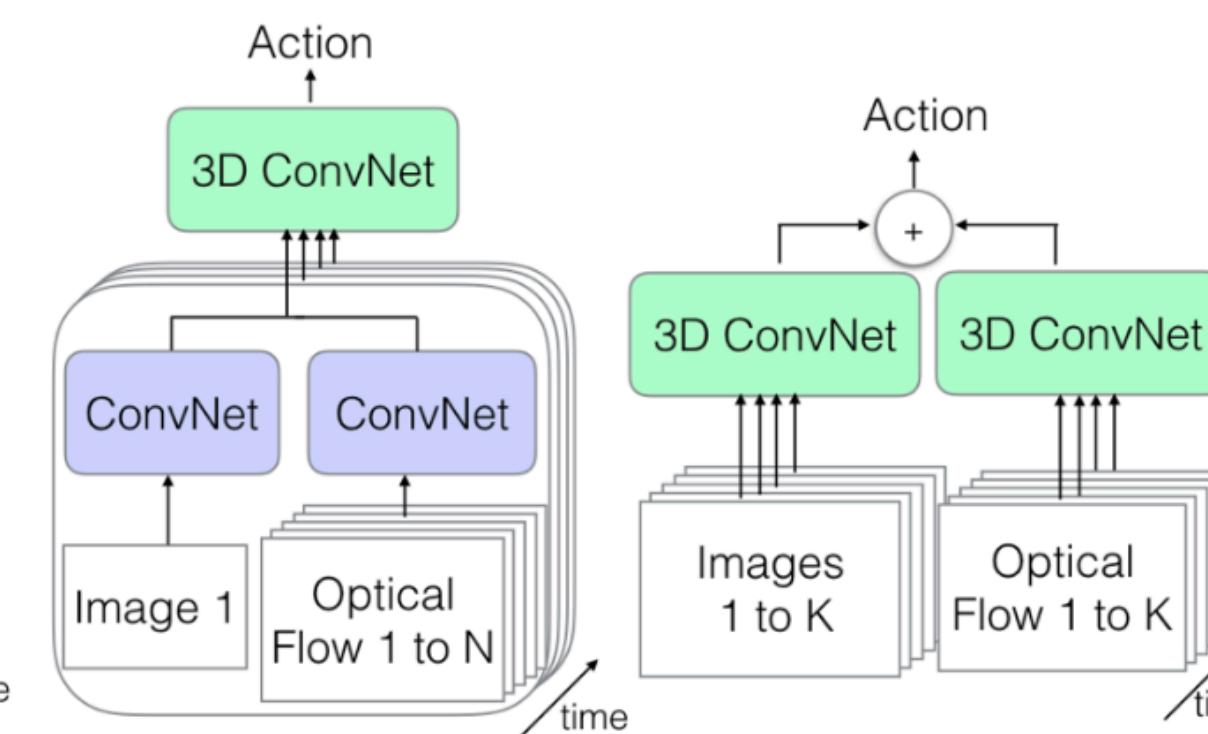
b) 3D-ConvNet



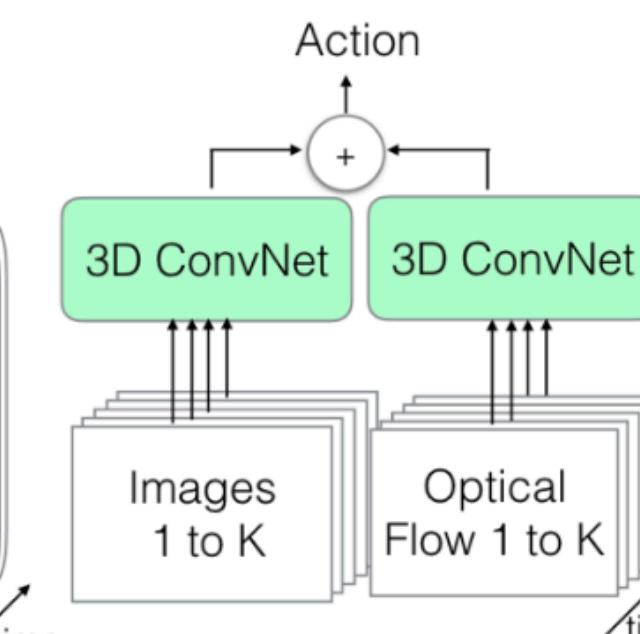
c) Two-Stream



d) 3D-Fused Two-Stream



e) Two-Stream 3D-ConvNet



Two-stream Inflated 3D ConvNet (I3D)

Repeating the weights of the 2D filters N times along the time dimension, and rescaling them by dividing by N .

Train the two networks separately and average their predictions at test time.

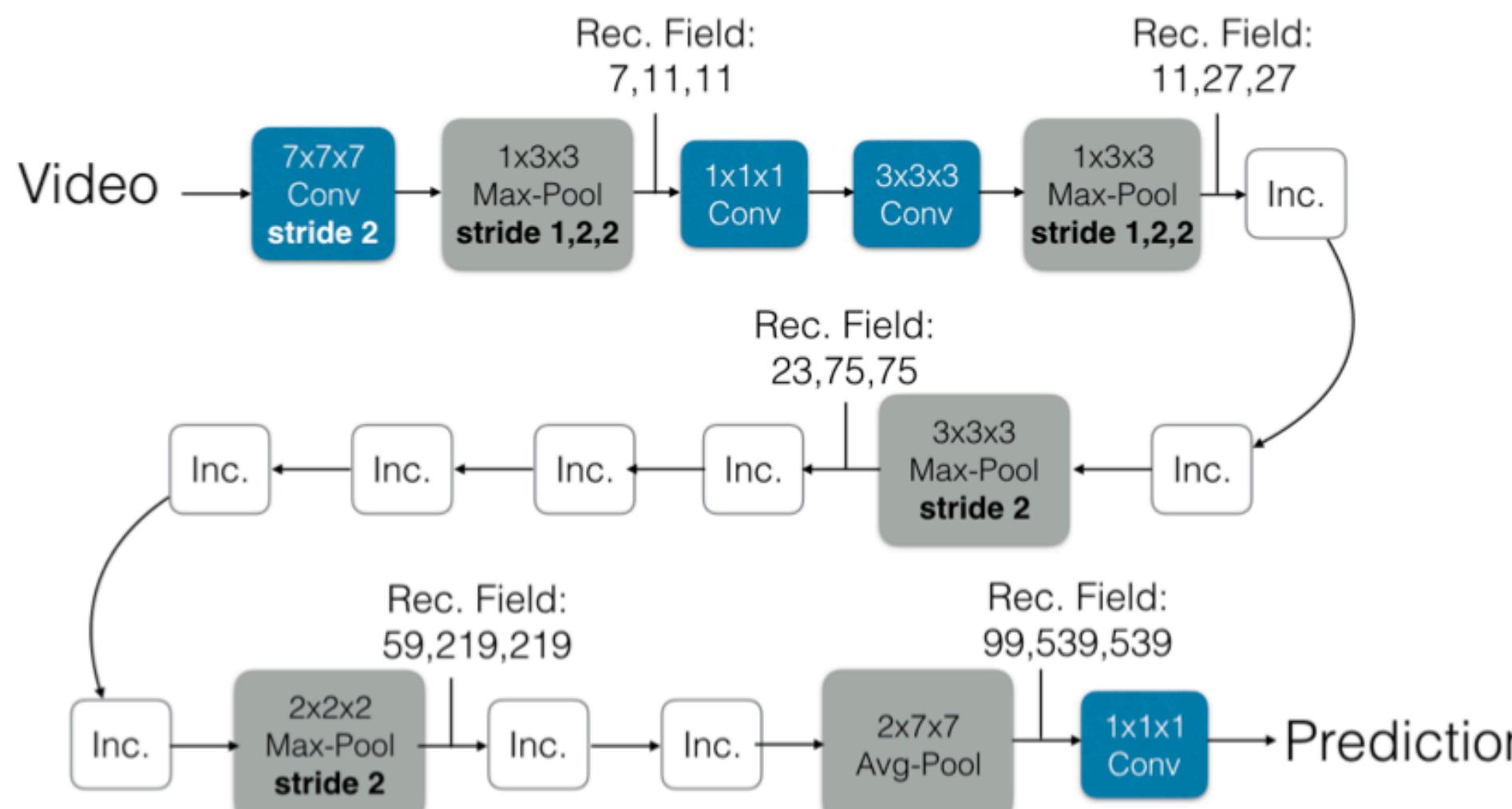
The Kinetics Human Action Video Dataset

The list of action classes covers: Person Actions (singular), e.g. drawing, drinking, laughing, punching; Person-Person Actions, e.g. hugging, kissing, shaking hands; and, Person-Object Actions, e.g. opening presents, mowing lawn, washing dishes.

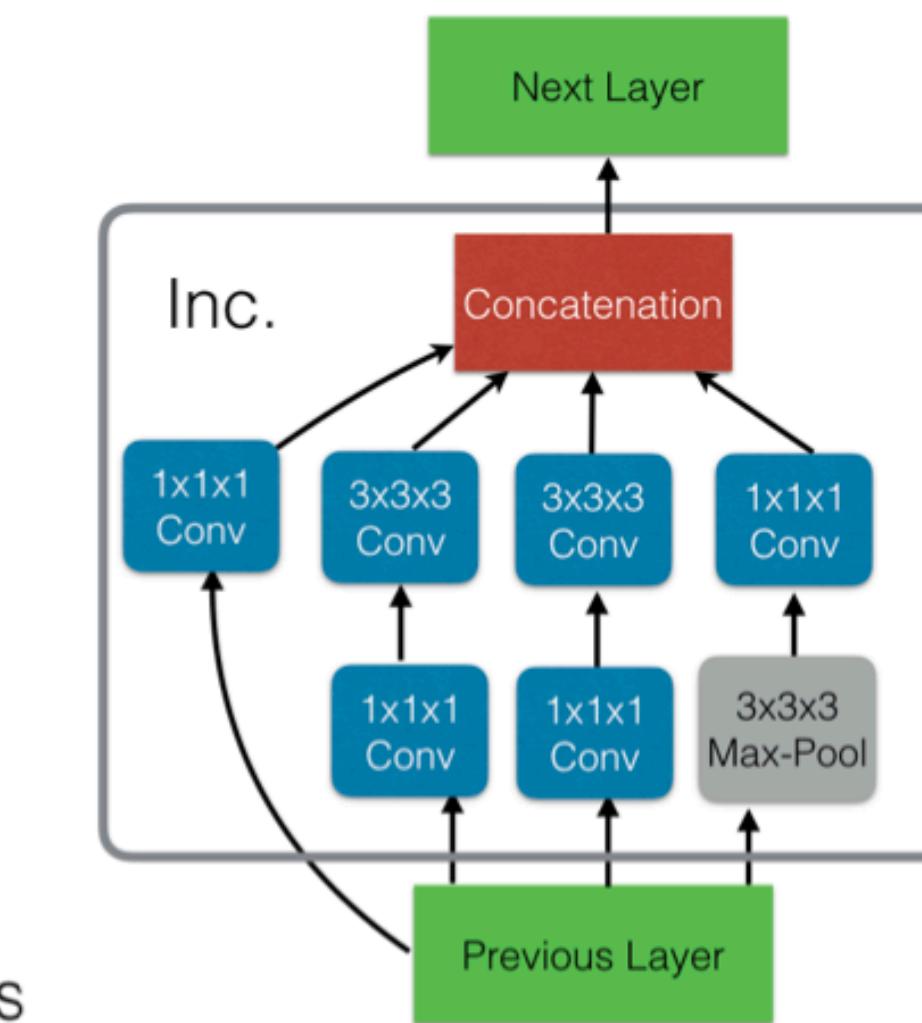
– 400 human action classes – 400 or more clips per class

$K \rightarrow$ total number of frames in a video
 $N \rightarrow$ a subset of neighboring frames of the video

Inflated Inception-V1



Inception Module (Inc.)



Model	UCF-101	HMDB-51
Two-Stream [25]	88.0	59.4
IDT [30]	86.4	61.7
Dynamic Image Networks + IDT [2]	89.1	65.2
TDD + IDT [31]	91.5	65.9
Two-Stream Fusion + IDT [8]	93.5	69.2
Temporal Segment Networks [32]	94.2	69.4
ST-ResNet + IDT [7]	94.6	70.3
Deep Networks [15], Sports 1M pre-training	65.2	-
C3D one network [29], Sports 1M pre-training	82.3	-
C3D ensemble [29], Sports 1M pre-training	85.2	-
C3D ensemble + IDT [29], Sports 1M pre-training	90.1	-
RGB-I3D, miniKinetics pre-training	91.8	66.4
RGB-I3D, Kinetics pre-training	95.4	74.5
Flow-I3D, miniKinetics pre-training	94.7	72.4
Flow-I3D, Kinetics pre-training	95.4	74.6
Two-Stream I3D, miniKinetics pre-training	96.9	76.3
Two-Stream I3D, Kinetics pre-training	97.9	80.2



Boulder

A Closer Look at Spatiotemporal Convolutions for Action Recognition

– Sports-1M – Kinetics – UCF101 – HMDB51

Convolutional residual blocks for video

$x \in \mathbb{R}^{3 \times L \times H \times W} \rightarrow$ input clip

$L \rightarrow$ number of frames in the clip

$z_i \rightarrow$ tensor computed by the i -th convolutional block in the residual network

$$z_i = z_{i-1} + \mathcal{F}(z_{i-1}; \theta_i)$$

$\mathcal{F} \rightarrow$ two convolutional layers with a ReLU activation function after each layer

Global average pooling at the end

R2D: 2D convolutions over the entire clip

Reshape x into a 3D tensor of size $3L \times H \times W$

$$z_i \in \mathbb{R}^{N_i \times H_i \times W_i}$$

$$N_{i-1} \times d \times d \rightarrow$$
 each filter size

Prevents temporal reasoning!

f-R2D: 2D convolutions over frames

frame-based R2D

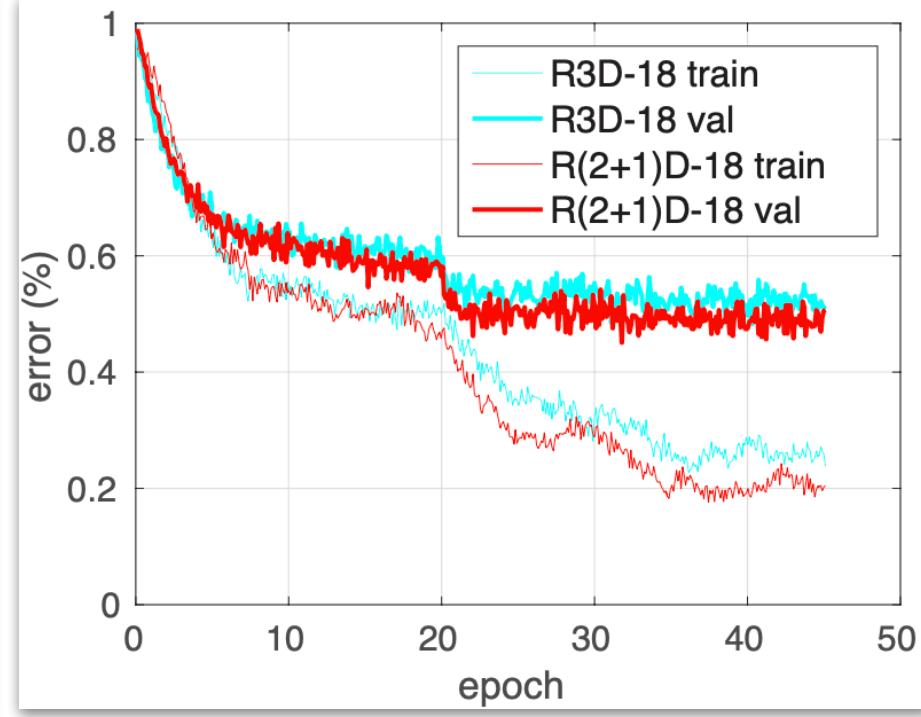
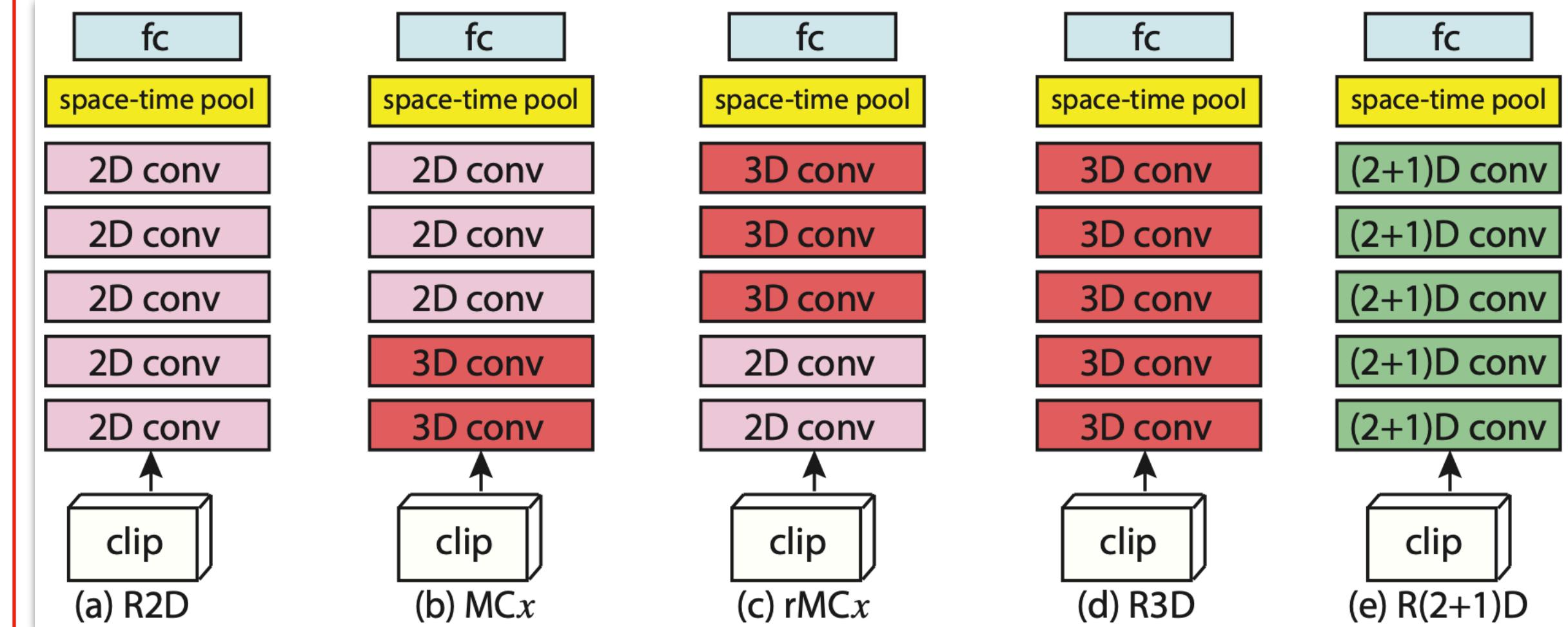
Processing independently the L frames via a series of 2D convolutional residual block. The same filters are applied to all L frames.

R3D: 3D convolutions

Preserves temporal information!

$$z_i \in \mathbb{R}^{N_i \times L \times H_i \times W_i}$$

$$N_{i-1} \times t \times d \times d \rightarrow$$
 filter size



MCx and rMCx: mixed 3D-2D convolutions

Mixed and Reversed Mixed Convolutions

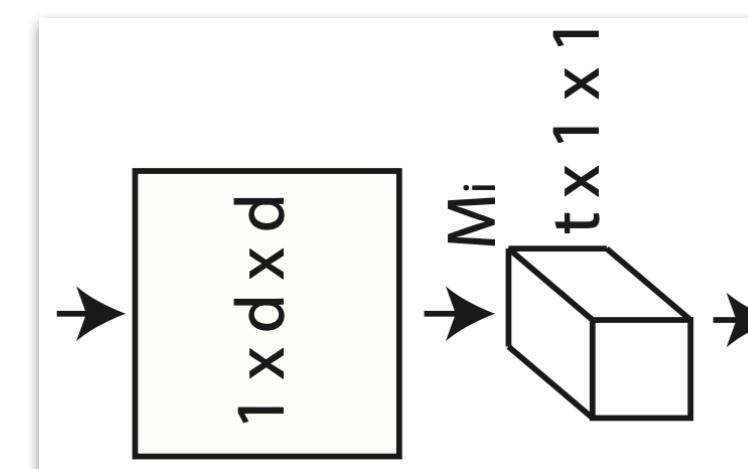
R(2+1)D: (2+1)D convolutions

A 2D convolution followed by a 1D convolution

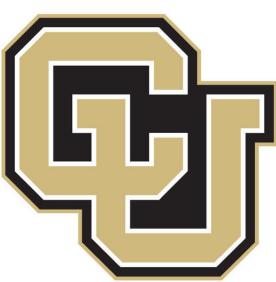
M_i 2D convolutional filters of size $N_{i-1} \times 1 \times d \times d$

N_i temporal convolutional filters of size $M_i \times t \times 1 \times 1$

$M_i = \lfloor \frac{td^2 N_{i-1} N_i}{d^2 N_{i-1} + t N_i} \rfloor \rightarrow$ similar number of parameters to a 3D convolution



Net	# params	Clip@1	Video@1	Clip@1	Video@1
Input			$8 \times 112 \times 112$		$16 \times 112 \times 112$
R2D	11.4M	46.7	59.5	47.0	58.9
f-R2D	11.4M	48.1	59.4	50.3	60.5
R3D	33.4M	49.4	61.8	52.5	64.2
MC2	11.4M	50.2	62.5	53.1	64.2
MC3	11.7M	50.7	62.9	53.7	64.7
MC4	12.7M	50.5	62.5	53.7	65.1
MC5	16.9M	50.3	62.5	53.7	65.1
rMC2	33.3M	49.8	62.1	53.1	64.9
rMC3	33.0M	49.8	62.3	53.2	65.0
rMC4	32.0M	49.9	62.3	53.4	65.1
rMC5	27.9M	49.4	61.2	52.1	63.1
R(2+1)D	33.3M	52.8	64.8	56.8	68.0

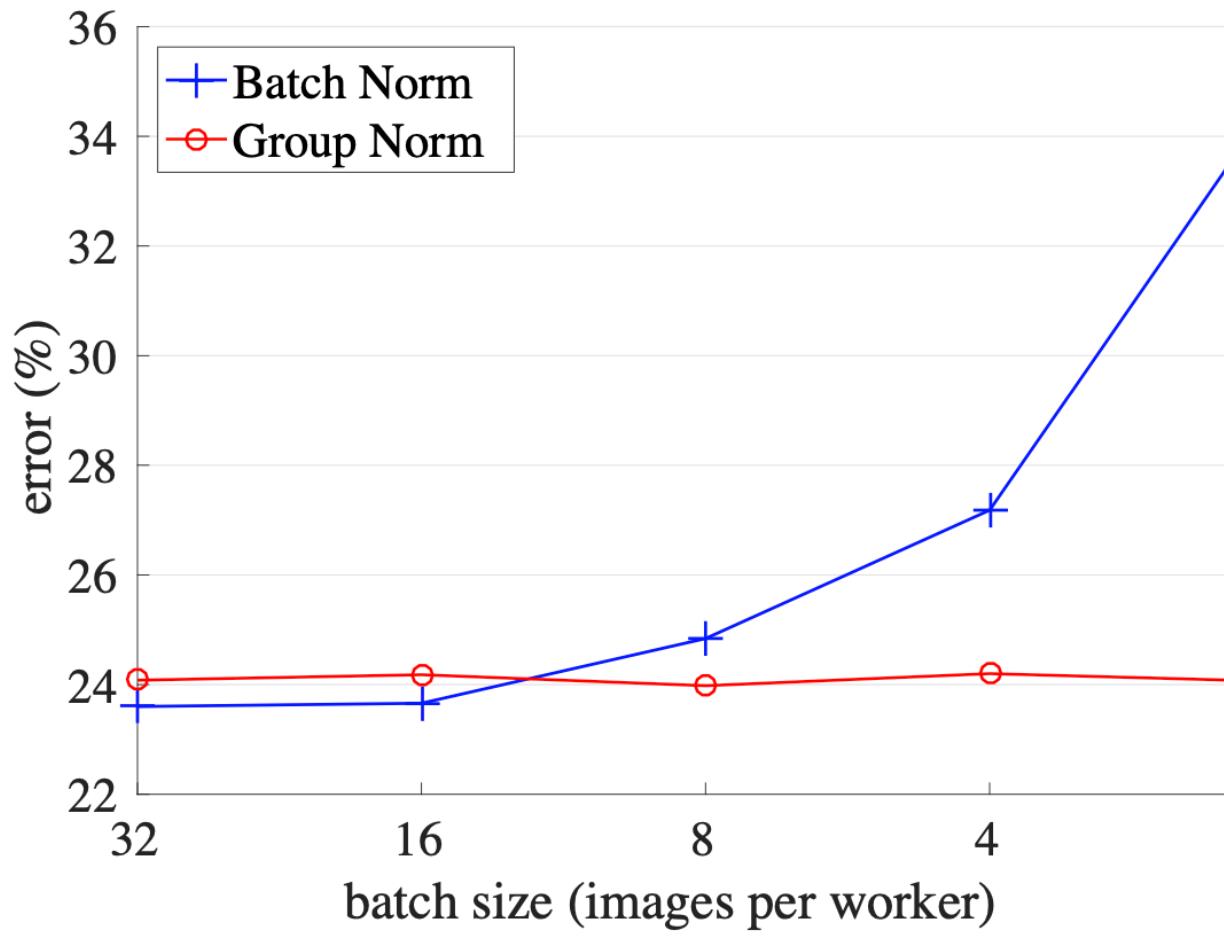


Boulder



[YouTube Video](#)

Group Normalization



$$\hat{x}_i = \frac{1}{\sigma_i} (x_i - \mu_i)$$

$x \rightarrow$ the feature computed by a layer

$i = (i_N, i_C, i_H, i_W) \rightarrow$ an index

$N \rightarrow$ batch axis

$C \rightarrow$ channel axis

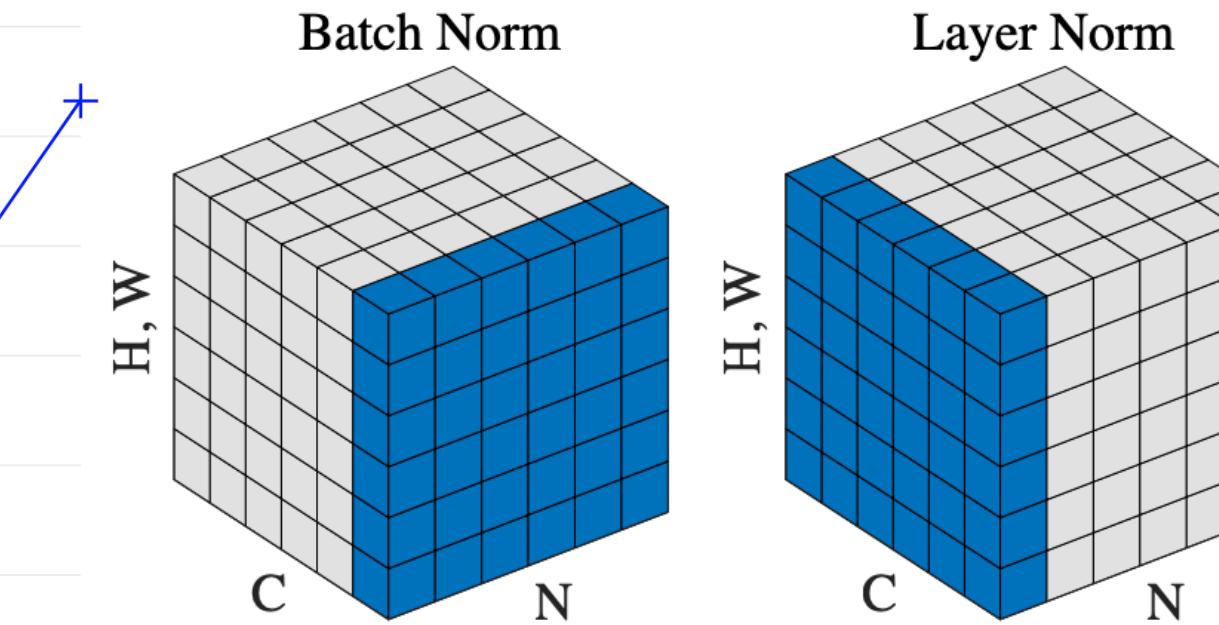
$H \rightarrow$ height axis

$W \rightarrow$ width axis

$$\mu_i = \frac{1}{m} \sum_{k \in \mathcal{S}_i} x_k, \quad \sigma_i = \sqrt{\frac{1}{m} \sum_{k \in \mathcal{S}_i} (x_k - \mu_i)^2 + \epsilon},$$

$\mathcal{S}_i \rightarrow$ set of pixels in which the mean and std are computed
 $|\mathcal{S}_i| = m$

Batch Norm $\Rightarrow \mu \& \sigma$ computed
 $\mathcal{S}_i = \{k : k_C = i_C\}$ along (N, H, W) axes



Layer Norm

$\mathcal{S}_i = \{k : k_N = i_N\}$
 $\Rightarrow \mu \& \sigma$ computed along (C, H, W) axes

Instance Norm

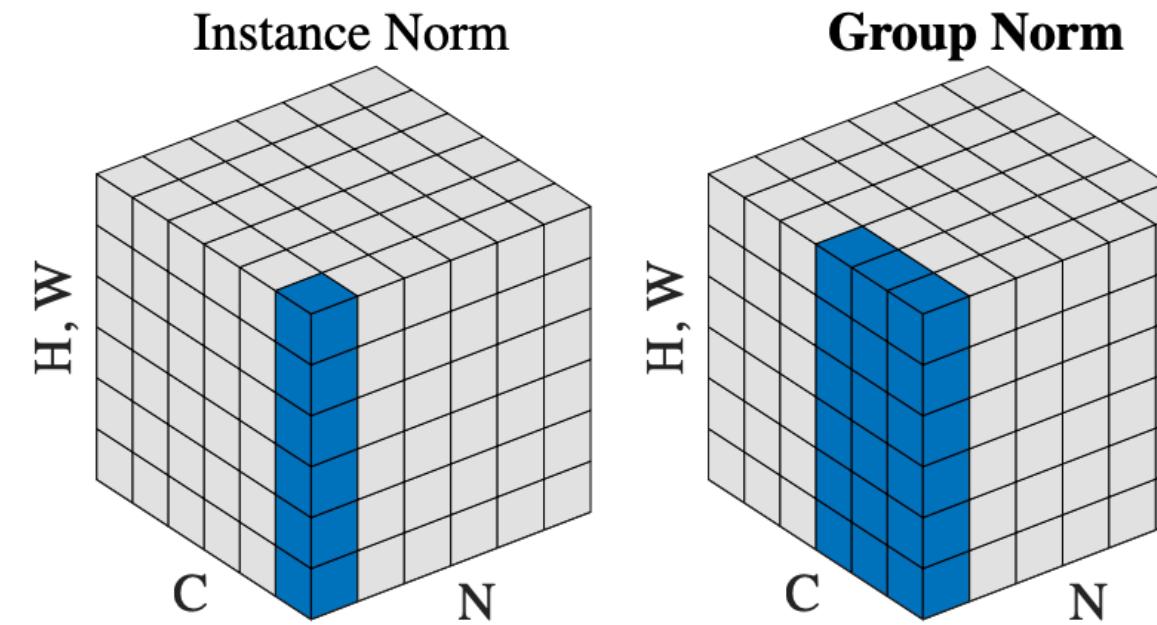
$\mathcal{S}_i = \{k : k_N = i_N, k_C = i_C\}$
 $\Rightarrow \mu \& \sigma$ computed along (H, W) axes for each sample and for each channel

Shift & Scale

$y_i = \gamma \hat{x}_i + \beta$
per channel γ and β
(indexed by i_C)

Group Norm

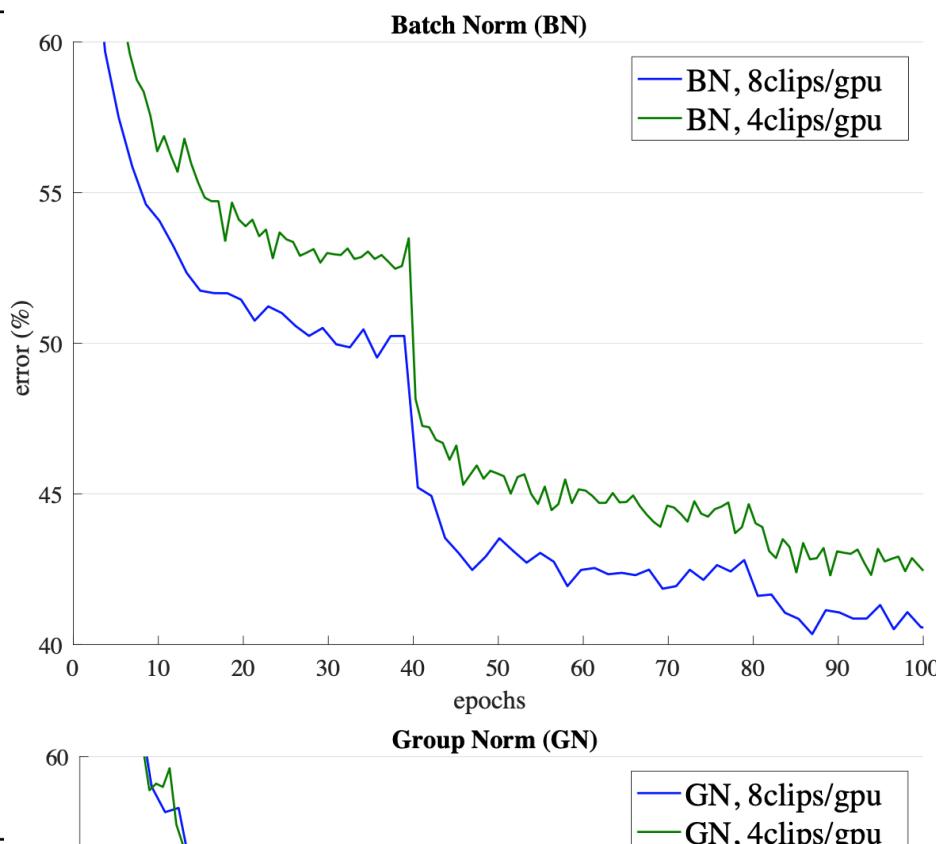
$\mathcal{S}_i = \{k : k_N = i_N, \left\lfloor \frac{k_C}{C/G} \right\rfloor = \left\lfloor \frac{i_C}{C/G} \right\rfloor\}$ $\Rightarrow \mu \& \sigma$ computed along (H, W) axes and a group of C/G channels
 $G = 32 \rightarrow$ number of groups
 $C/G \rightarrow$ number of channels per group



ImageNet

	BN	LN	IN	GN
val error	23.6	25.3	28.4	24.1
Δ (vs. BN)	-	1.7	4.8	0.5
batch size	32	16	8	4
BN	23.6	23.7	24.8	27.3
GN	24.1	24.2	24.0	24.1
Δ	0.5	0.5	-0.8	-3.1
				-10.6

Video Classification in Kinetics



```
def GroupNorm(x, gamma, beta, G, eps=1e-5):
    # x: input features with shape [N,C,H,W]
    # gamma, beta: scale and offset, with shape [1,C,1,1]
    # G: number of groups for GN
```

```
N, C, H, W = x.shape
x = tf.reshape(x, [N, G, C // G, H, W])
mean, var = tf.nn.moments(x, [2, 3, 4], keep_dims=True)
x = (x - mean) / tf.sqrt(var + eps)

x = tf.reshape(x, [N, C, H, W])

return x * gamma + beta
```

Detection and segmentation ablation results in COCO

backbone	AP ^{bbox}	AP ₅₀ ^{bbox}	AP ₇₅ ^{bbox}	AP ^{mask}	AP ₅₀ ^{mask}	AP ₇₅ ^{mask}
BN*	37.7	57.9	40.9	32.8	54.3	34.7
GN	38.8	59.2	42.2	33.6	55.9	35.4

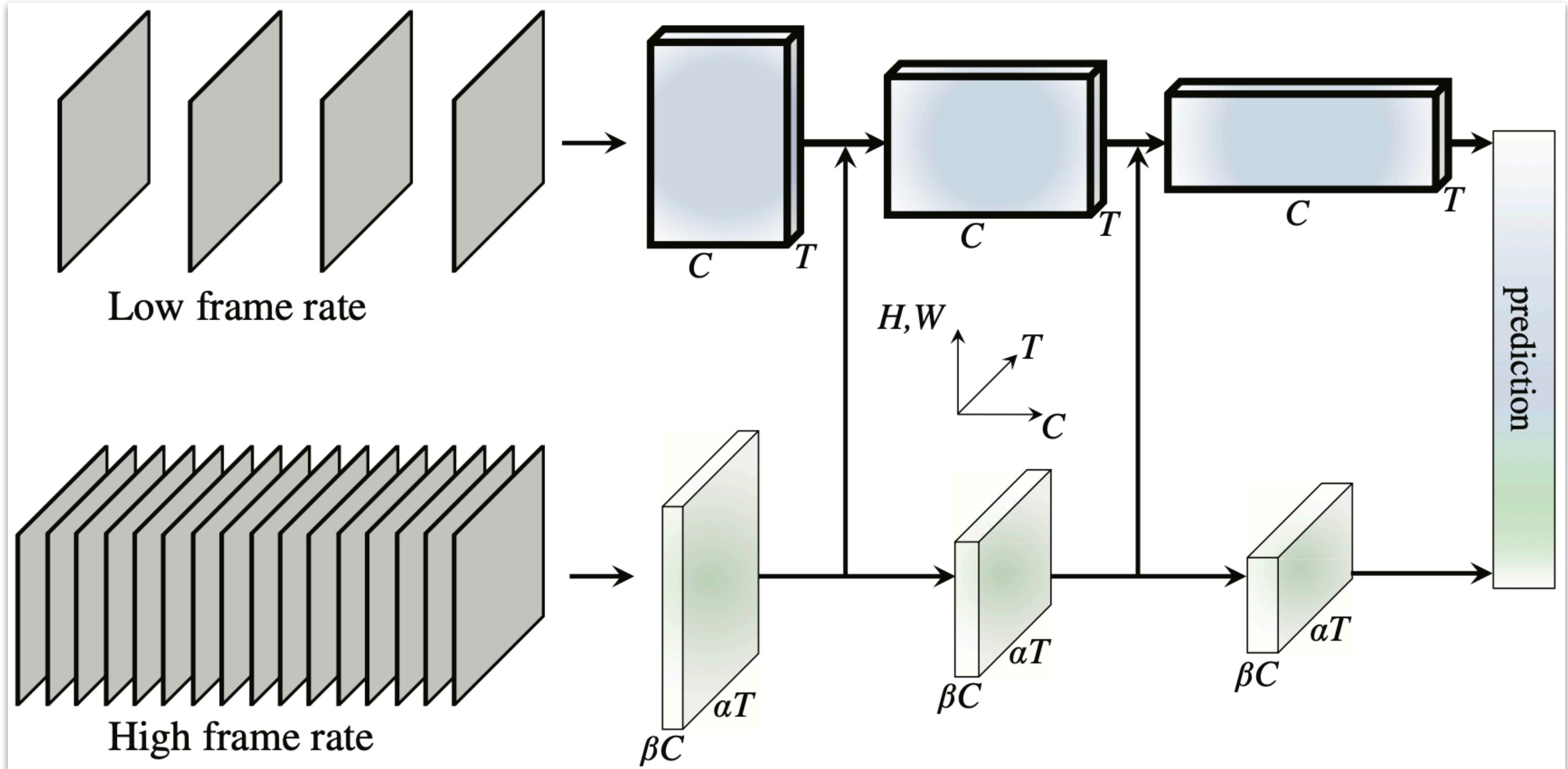
clip length	32	32	64
batch size	8	4	4
BN	73.3 / 90.7	72.1 / 90.0	73.3 / 90.8
GN	73.0 / 90.6	72.8 / 90.6	74.5 / 91.7



Boulder

SlowFast Networks for Video Recognition

– Kinetics – Charades – AVA dataset



$\alpha > 1 \rightarrow$ frame rate ratio between Fast and Slow pathways

$\beta < 1 \rightarrow$ ratio of Fast pathway channels to Slow ones

Fast pathway: high frame rate, high temporal resolution features (no temporal downsampling layers), and low channel capacity

Lateral connections

$\{T, S^2, C\}, \{\alpha T, S^2, \beta C\} \rightarrow$ shapes of the Slow and Fast pathways

Time-strided convolution: Perform a 3D convolution of a 5×1^2 kernel with $2\beta C$ output channels and stride = α . Concatenation is employed for fusion.

stage	Slow pathway	Fast pathway	output sizes $T \times S^2$
raw clip	-	-	64×224^2
data layer	stride 16, 1^2	stride $2, 1^2$	<i>Slow</i> : 4×224^2 <i>Fast</i> : 32×224^2
conv ₁	$1 \times 7^2, 64$ stride 1, 2^2	$5 \times 7^2, 8$ stride 1, 2^2	<i>Slow</i> : 4×112^2 <i>Fast</i> : 32×112^2
pool ₁	1×3^2 max stride 1, 2^2	1×3^2 max stride 1, 2^2	<i>Slow</i> : 4×56^2 <i>Fast</i> : 32×56^2
res ₂	$\left[\begin{array}{l} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{array} \right] \times 3$	$\left[\begin{array}{l} \underline{3} \times 1^2, 8 \\ \underline{1} \times 3^2, 8 \\ 1 \times 1^2, 32 \end{array} \right] \times 3$	<i>Slow</i> : 4×56^2 <i>Fast</i> : 32×56^2
res ₃	$\left[\begin{array}{l} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{array} \right] \times 4$	$\left[\begin{array}{l} \underline{3} \times 1^2, 16 \\ \underline{1} \times 3^2, 16 \\ 1 \times 1^2, 64 \end{array} \right] \times 4$	<i>Slow</i> : 4×28^2 <i>Fast</i> : 32×28^2
res ₄	$\left[\begin{array}{l} \underline{3} \times 1^2, 256 \\ \underline{1} \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{array} \right] \times 6$	$\left[\begin{array}{l} \underline{3} \times 1^2, 32 \\ \underline{1} \times 3^2, 32 \\ 1 \times 1^2, 128 \end{array} \right] \times 6$	<i>Slow</i> : 4×14^2 <i>Fast</i> : 32×14^2
res ₅	$\left[\begin{array}{l} \underline{3} \times 1^2, 512 \\ \underline{1} \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{array} \right] \times 3$	$\left[\begin{array}{l} \underline{3} \times 1^2, 64 \\ \underline{1} \times 3^2, 64 \\ 1 \times 1^2, 256 \end{array} \right] \times 3$	<i>Slow</i> : 4×7^2 <i>Fast</i> : 32×7^2
		global average pool, concat, fc	# classes

dimensions of kernels are denoted by $\{T \times S^2, C\}$ for temporal, spatial, and channel sizes. Strides are denoted as $\{\text{temporal stride}, \text{spatial stride}^2\}$. Here the speed ratio is $\alpha = 8$ and the channel ratio is $\beta = 1/8$. τ is 16. The green colors mark *higher* temporal resolution, and orange colors mark *fewer* channels, for the Fast pathway. Non-degenerate temporal filters are underlined. Residual blocks are shown by brackets. The backbone is ResNet-50.



Boulder

Learning Multi-Domain Convolutional Neural Networks for Visual Tracking

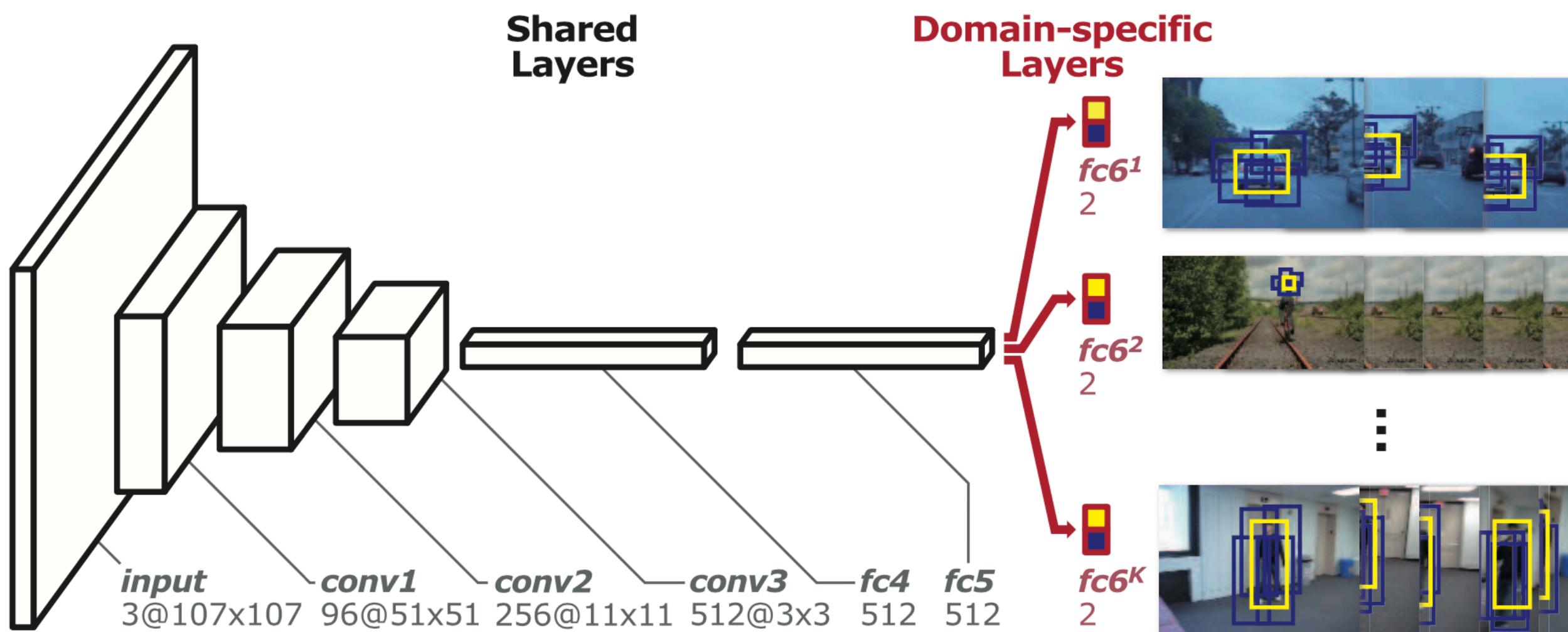
Multi-Domain Network (MDNet)

Each video is regarded as a separate domain!

Offline multi-domain learning

Collect 50 positive and 200 negative samples from every frame, where positive and negative examples have ≥ 0.7 and ≤ 0.5 IoU overlap ratios with ground-truth bounding boxes, respectively.

Yellow and blue bounding boxes denote the positive and negative samples in each domain, respectively.



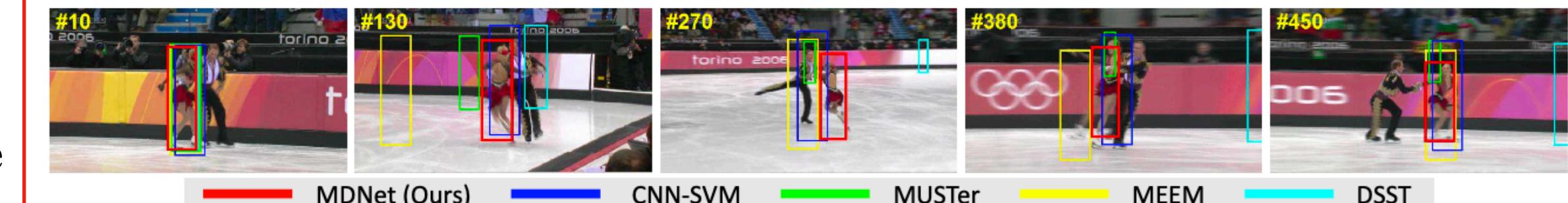
Online tracking algorithm

Fine-tune a new fc6 branch (corresponding to a new test video) and the shared fc4 and fc5 while keeping the conv layers fixed (on the first frame).

Input: Initial target state x_1

Output: Estimate target states x_t^*

<https://www.youtube.com/watch?v=zYM7G5qd090&t=102s>



$x_t^1, \dots, x_t^N \rightarrow N$ target candidates sampled around the previous target state x_{t-1}^*

$f^+(x_t^i), f^-(x_t^i) \rightarrow$ positive and negative scores

$x_t^* = \arg \max_{x_t^i} f^+(x_t^i) \rightarrow$ optimal target state

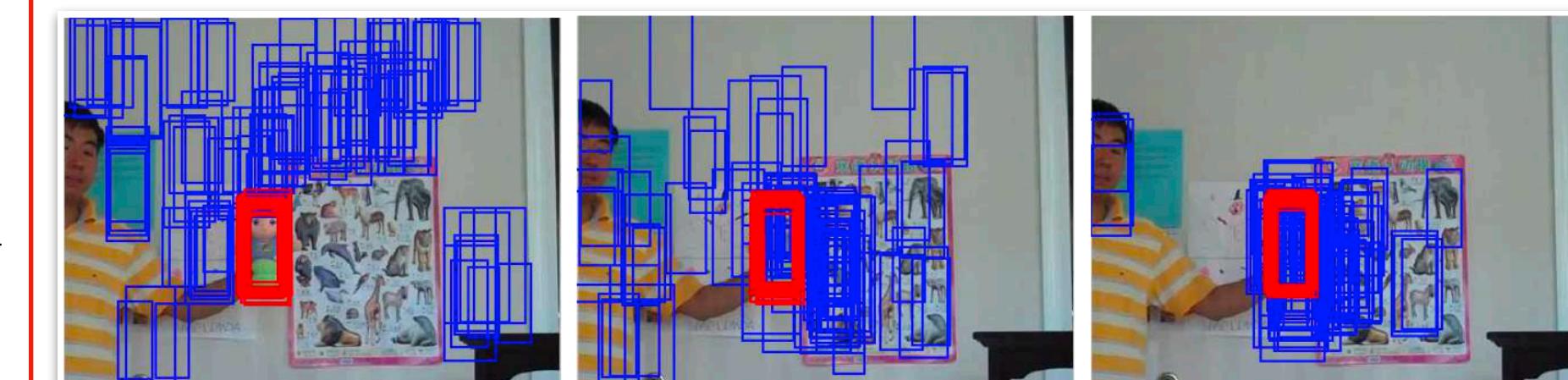
If $f^+(x_t^*) > 0.5$ then accept x_t^* as the target state and adjust it according to a simple linear bounding box regressor trained on the first frame only.

If $f^+(x_t^*) < 0.5$ then update fc4, fc5 and fc6.

Hard Minibatch Mining

Minibatch consists of M^+ positives and M_h^- hard negatives.

The hard negative examples are identified by testing $M^- (\gg M_h^-)$ negative samples and selecting the ones with top M_h^- scores.



The evaluation is based on two metrics: center location error and bounding box overlap ratio.



Boulder

Fully-Convolutional Siamese Networks for Object Tracking

<https://www.youtube.com/watch?v=zS6kil8DgDY>



Frame 1 (init.)

Frame 50

Frame 100

Frame 200

Deep Similarity Learning for Tracking

$z \rightarrow$ an exemplar image (i.e., initial appearance of the object)

$x \rightarrow$ candidate image of the same size

$f(z, x) \rightarrow$ return a high score if two images depict the same object
and a low score otherwise

To find the position of the object in a new image, we can then exhaustively test all possible locations and choose the candidate with the maximum similarity to the past appearance of the object.

$f \rightarrow$ learned from a dataset of videos with labeled object trajectories

$f(z, x) = g(\varphi(z), \varphi(x)) \rightarrow$ Siamese network

embedding

a simple distance of a similarity metric

Fully-Convolutional Siamese Architecture

$h(L_{k\tau}x) = L_\tau h(x) \iff h$ is fully convolutional with stride k

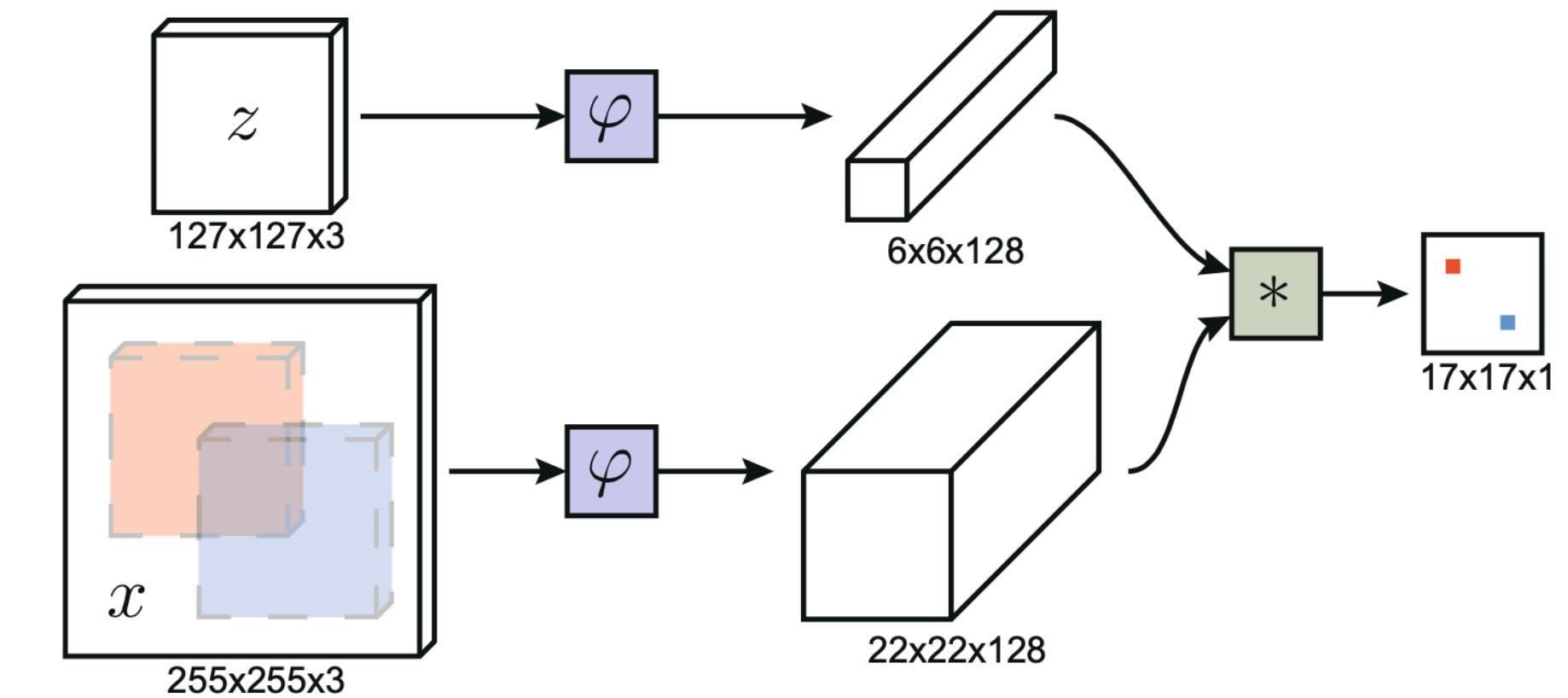
$(L_\tau x)[u] = x[u - \tau]$

$f(z, x) = \varphi(z) * \varphi(x) + b \rightarrow$ score map defined on a finite grid $\mathcal{D} \subset \mathbb{Z}^2$

convolutional embedding function

cross-correlation layer

Advantage: Instead of a candidate image of the same size we can use a much larger search image



During tracking, we use a search image centred at the previous position of the target. The position of the maximum score relative to the centre of the score map, multiplied by the stride of the network, gives the displacement of the target from frame to frame. Multiple scales are searched in a single forward-pass by assembling a mini-batch of scaled images.

Training with Large Search Images

$$L(y, v) = \frac{1}{|\mathcal{D}|} \sum_{u \in \mathcal{D}} \ell(y[u], v[u])$$

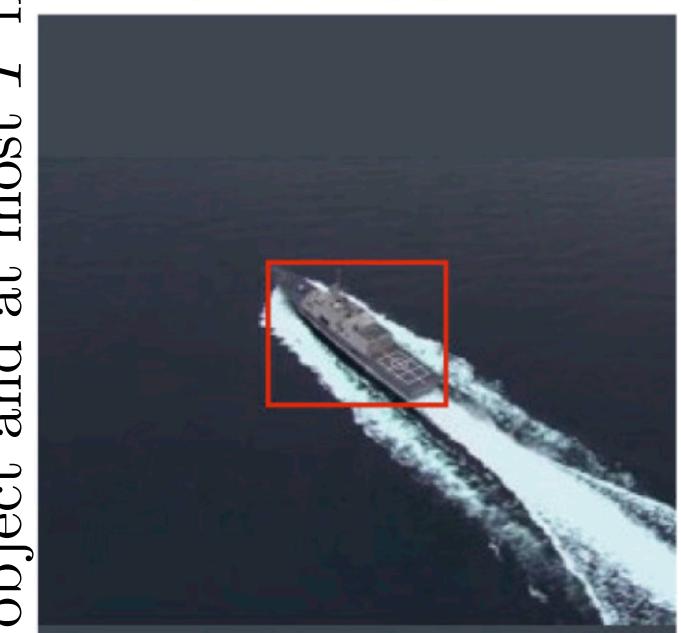
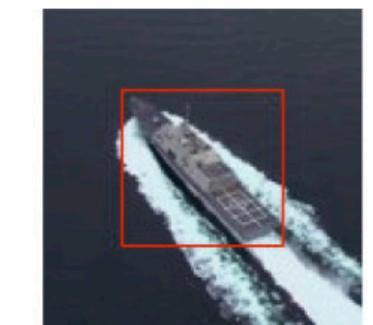
logistic loss

$$\ell(y, v) = \log(1 + \exp(-yv))$$

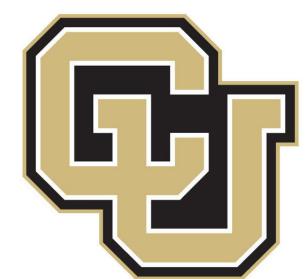
$$y[u] = \begin{cases} +1 & \text{if } k\|u - c\| \leq R \\ -1 & \text{otherwise.} \end{cases}$$

Dataset (%)	# Videos	# Objects	Accuracy	# Failures	Expected avg. overlap
2	88	60k	0.484	183	0.168
4	177	110k	0.501	160	0.192
8	353	190k	0.484	142	0.193
16	707	330k	0.522	132	0.219
32	1413	650k	0.521	117	0.234
100	4417	2m	0.524	87	0.274

Two frames of a video, both containing the object and at most T frames apart



ImageNet Video



Boulder



Questions?

[YouTube Playlist](#)
