Boulder

Computer Vision; Video

YouTube Playlist

Maziar Raissi

Assistant Professor

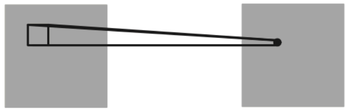
Department of Applied Mathematics

University of Colorado Boulder

maziar.raissi@colorado.edu

3D Convolutional Neural Networks for Human Action Recognition

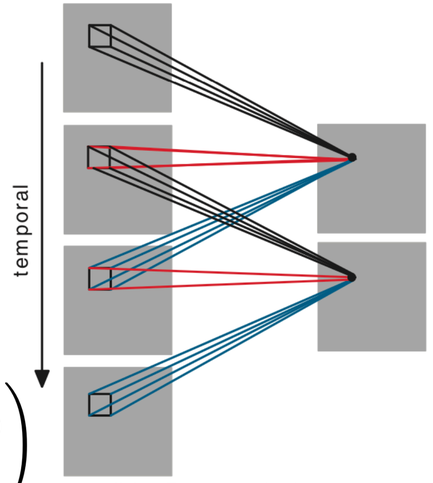
– intelligent video surveillance
– customer attributes
– shopping behavior analysis



(a) 2D convolution

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

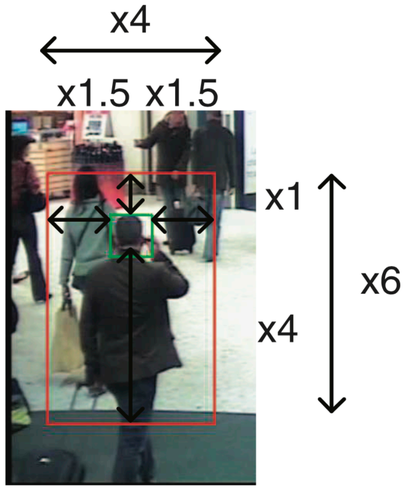
$$v_{ij}^{xy} = \tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1} w_{ijm}^{pq} v_{(i-1)m}^{(x+p)(y+q)}\right)$$



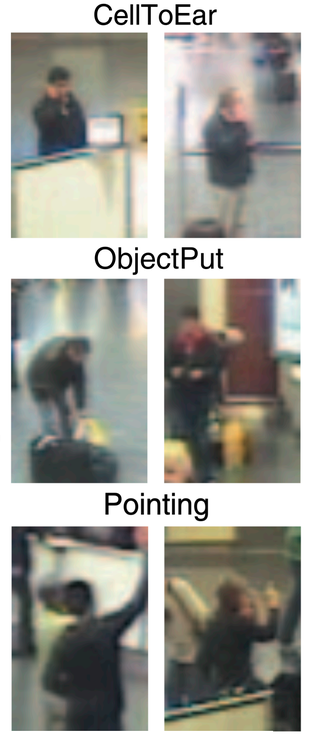
(b) 3D convolution

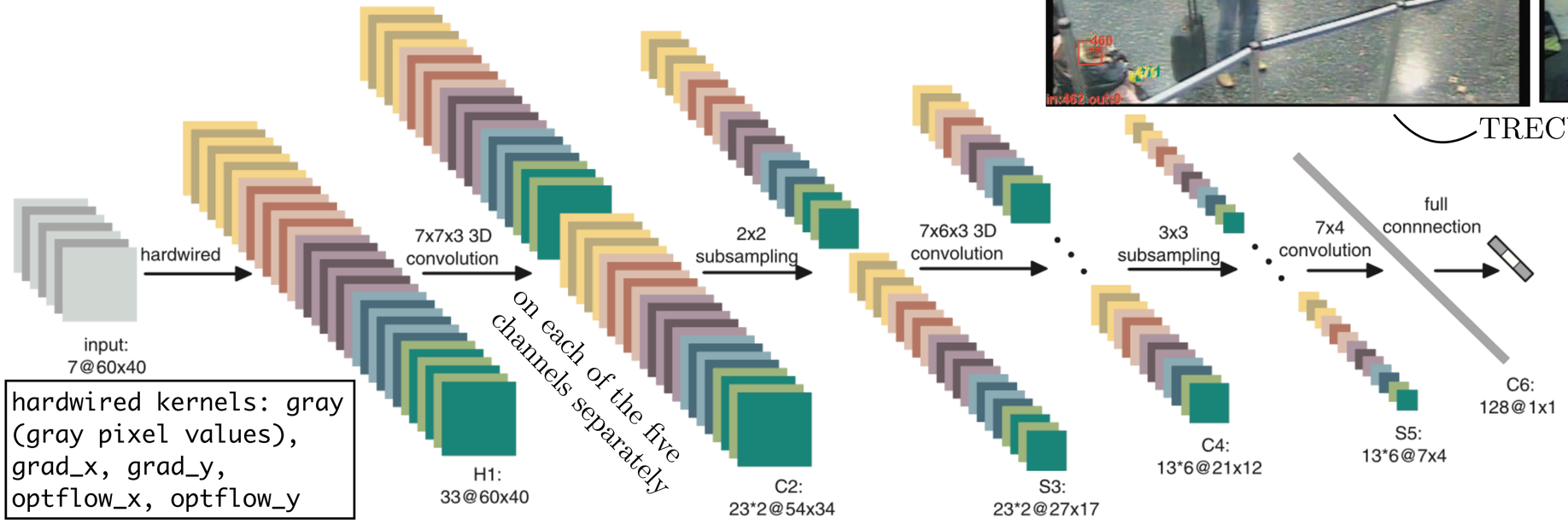
$$v_{ij}^{xyz} = \tanh\left(b_{ij} + \sum_{m}\sum_{p=0}^{P_i-1}\sum_{q=0}^{Q_i-1}\sum_{r=0}^{R_i-1} w_{ijm}^{pqr} v_{(i-1)m}^{(x+p)(y+q)(z+r)}\right)$$

human detection and tracking

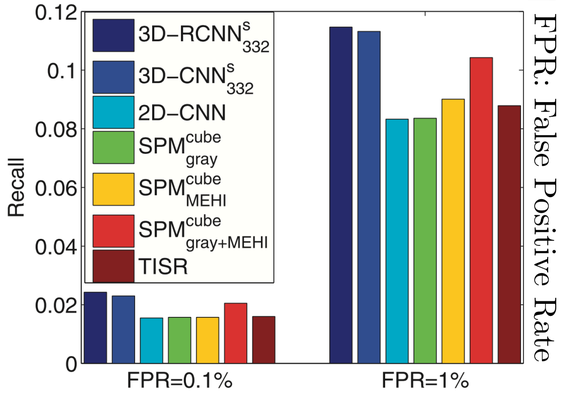


TRECVID dataset





hardwired kernels: gray (gray pixel values), grad_x, grad_y, optflow_x, optflow_y

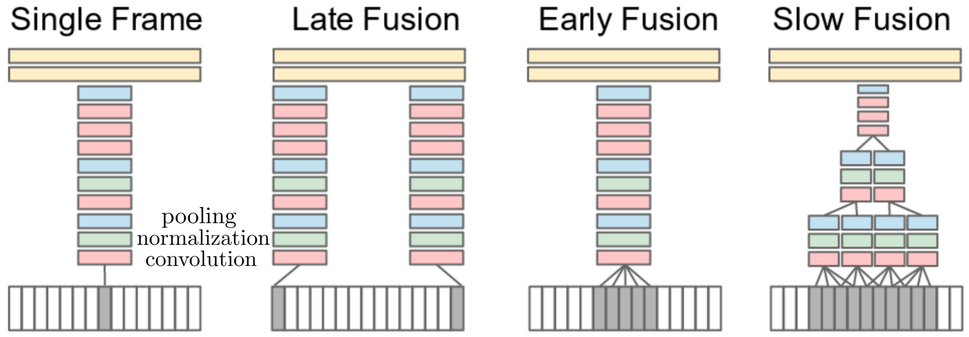


Ji, Shuiwang, et al. "3D convolutional neural networks for human action recognition." *IEEE transactions on pattern analysis and machine intelligence* 35.1 (2012): 221-231.

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 ← Transfer Learning
treat every video as a bag of short, fixed-sized clips
each clip contains several contiguous frames in time



Single Frame

convolution $\leftarrow C(d, f, s)$ (filters $f \times f$, stride)

$170 \times 170 \times 3 \rightarrow$ 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

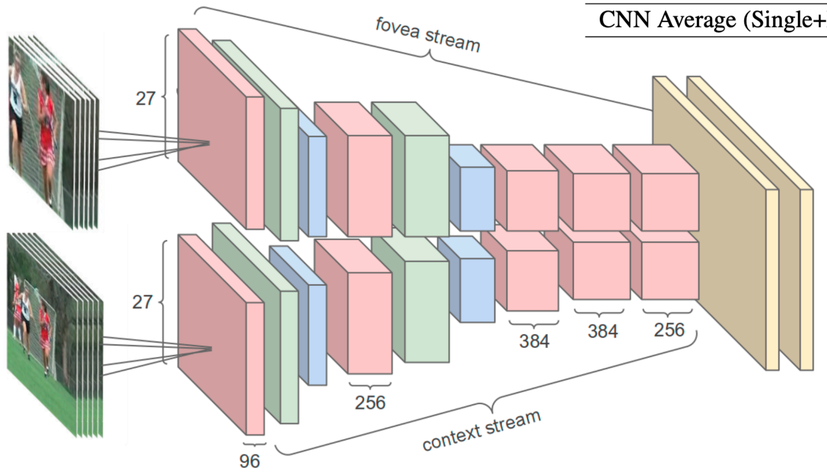
Slow Fusion

A balanced mix between the two approaches

Multi-resolution CNNs

178×178 → 89×89 downsampled (contex stream)
178×178 → 89×89 center crop (fovea stream)

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



juggling club
single frame predictions:
acrobatics
wing tsun
freestyle slalom skating
trapeze
unicycle
motion-aware predictions:
juggling club
kalaripayattu
baton twirling
acrobatics
color guard (flag spinning)

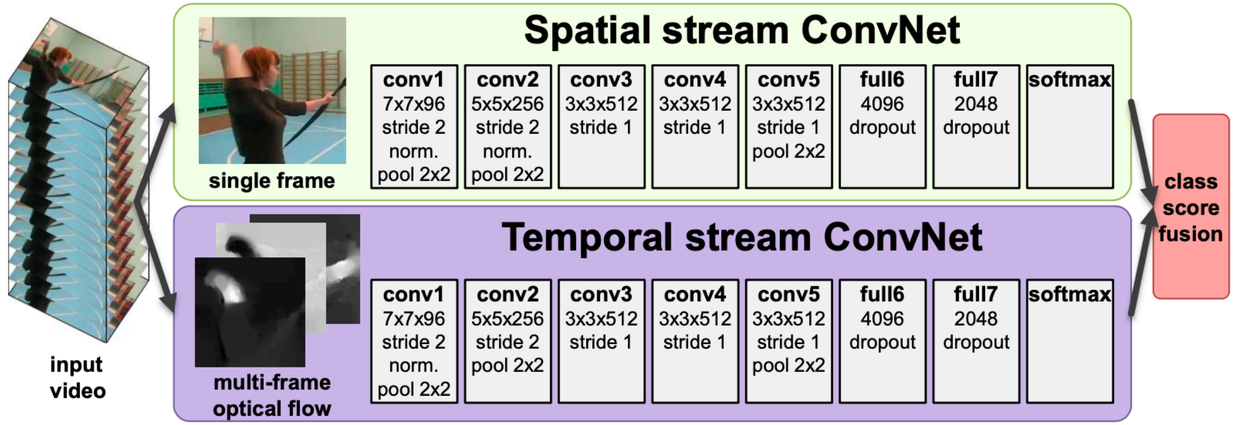
Karpathy, Andrej, et al. "Large-scale video classification with convolutional neural networks." *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*. 2014.

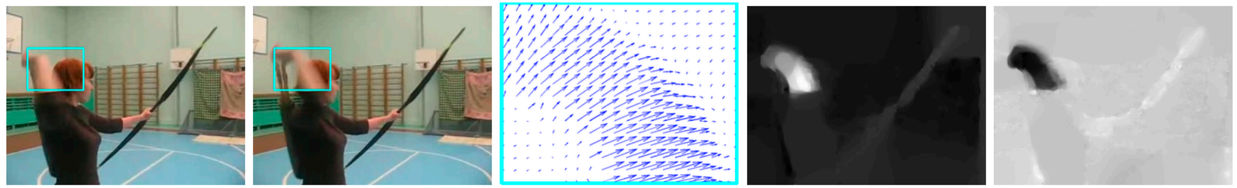
Two-Stream Convolutional Networks for Action Recognition in Videos

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

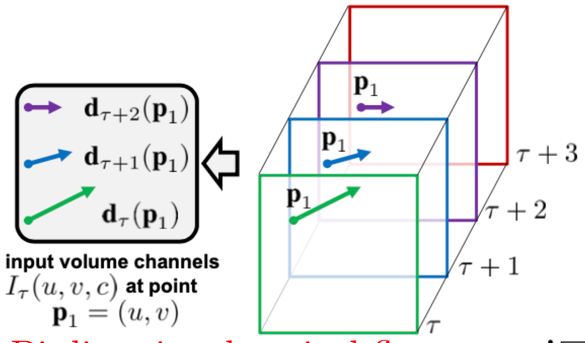
Optical flow stacking

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

$\tau \rightarrow$ an arbitrary frame

$I_\tau \in \mathbb{R}^{w \times h \times 2L} \rightarrow$ input volume

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

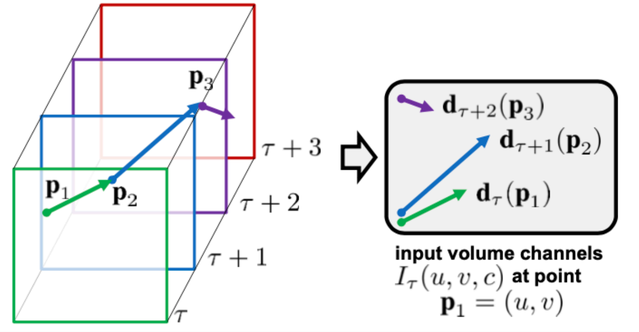


Bi-directional optical flow

forward & backward flows

Trajectory stacking

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



Spatial ConvNet.

Training setting	Dropout ratio 0.5	Dropout ratio 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	Mean subtraction: 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%**

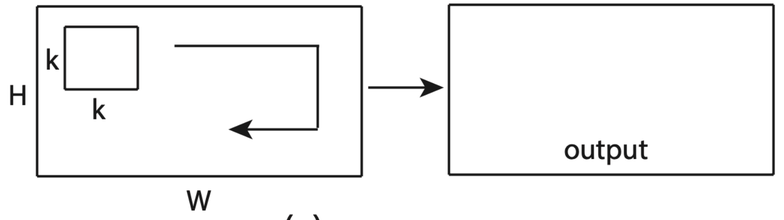
Method	UCF-101	HMDB-51
Improved dense trajectories (IDT) [26, 27]	85.9%	57.2%
IDT with higher-dimensional encodings [20]	**87.9%**	61.1%
IDT with stacked Fisher encoding [21] (based on Deep Fisher Net [23])	-	**66.8%**
Spatio-temporal HMAX network [11, 16]	-	22.8%
"Slow fusion" spatio-temporal ConvNet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

Simonyan, Karen, and Andrew Zisserman. "Two-stream convolutional networks for action recognition in videos." *Advances in neural information processing systems*. 2014.

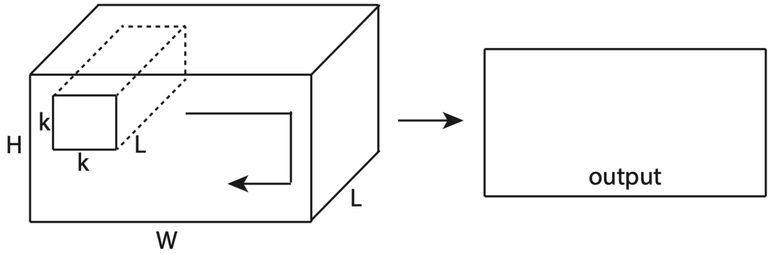
Learning Spatiotemporal Features with 3D Convolutional Networks

YouTube Video

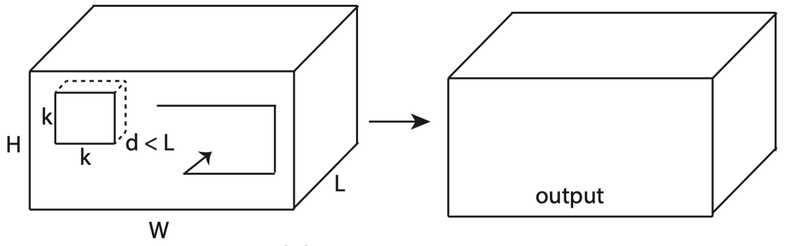
– search – recommendation – ranking
– action recognition
– abnormal event detection (anomaly detection)
– activity understanding
– video retrieval
– event & action detection



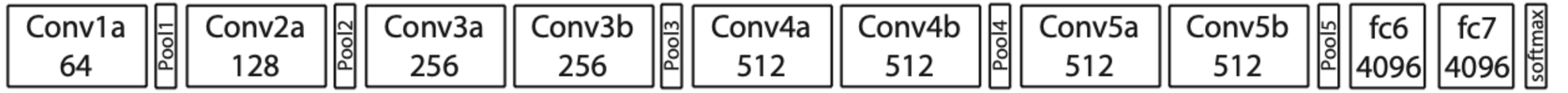
(a) 2D convolution



(b) 2D convolution on multiple frames



(c) 3D convolution

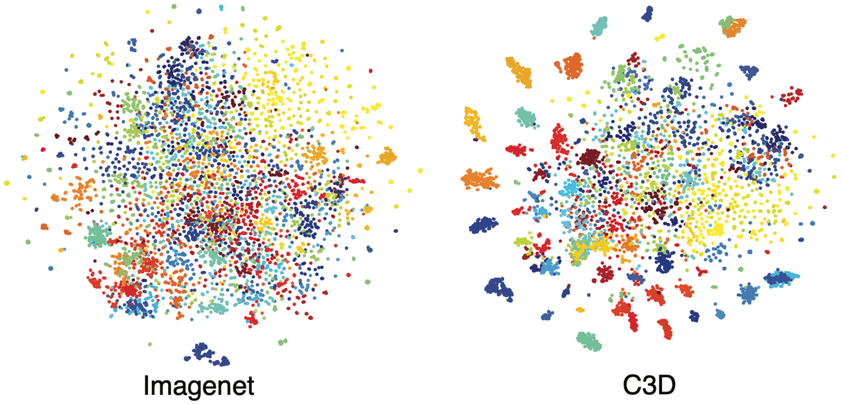


All 3D convolution kernels are $3 \times 3 \times 3$

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



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

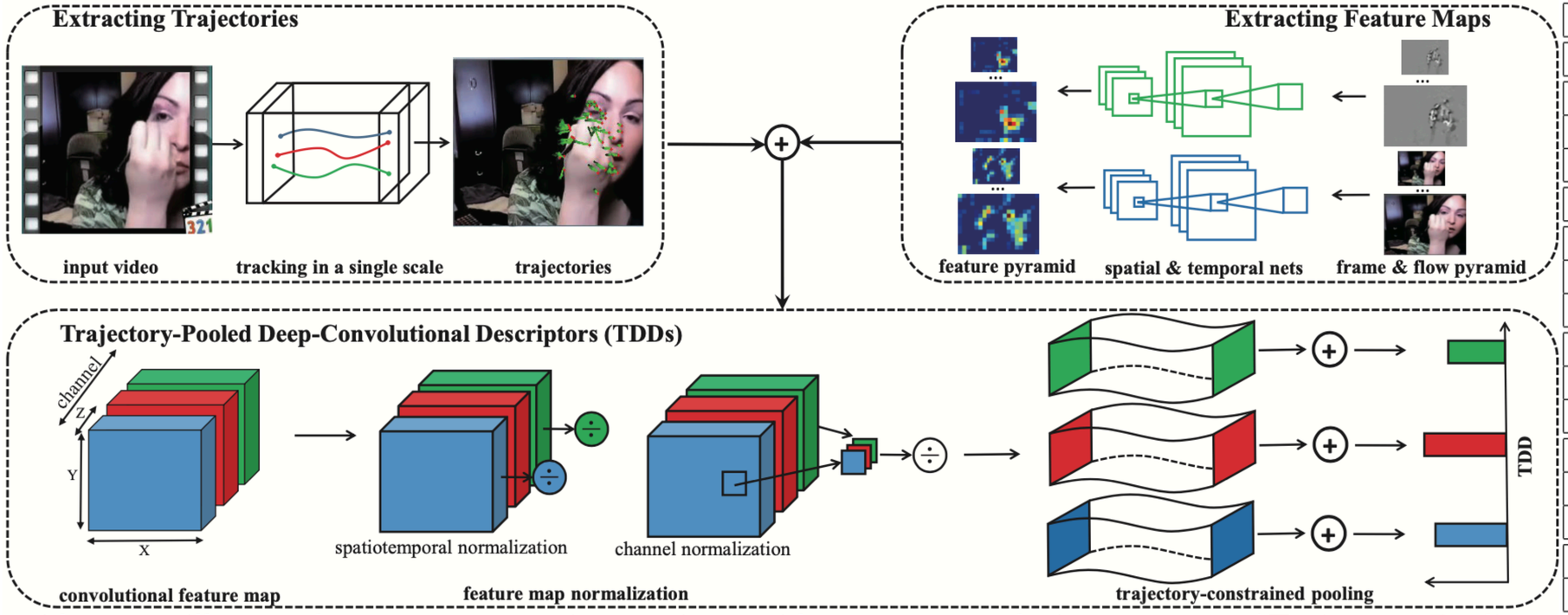
Runtime analysis on UCF101

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

Tran, Du, et al. "Learning spatiotemporal features with 3d convolutional networks." *Proceedings of the IEEE international conference on computer vision*. 2015.

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

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

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

$C(V) = \{C_1^s, \ldots, C_M^s, C_1^t, \ldots, C_M^t\} \rightarrow$ 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

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

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

$\overline{(\cdot)} \rightarrow$ rounding operation

spatiotemporal normalization

$$\widetilde{C}_m^a(x, y, z, n) = C_m^a(x, y, z, n) / \max_{x,y,z} C_m^a(x, y, z, n)$$

channel

$$\widetilde{C}_m^a(x, y, z, n) = C_m^a(x, y, z, n) / \max_{n} C_m^a(x, y, z, n)$$

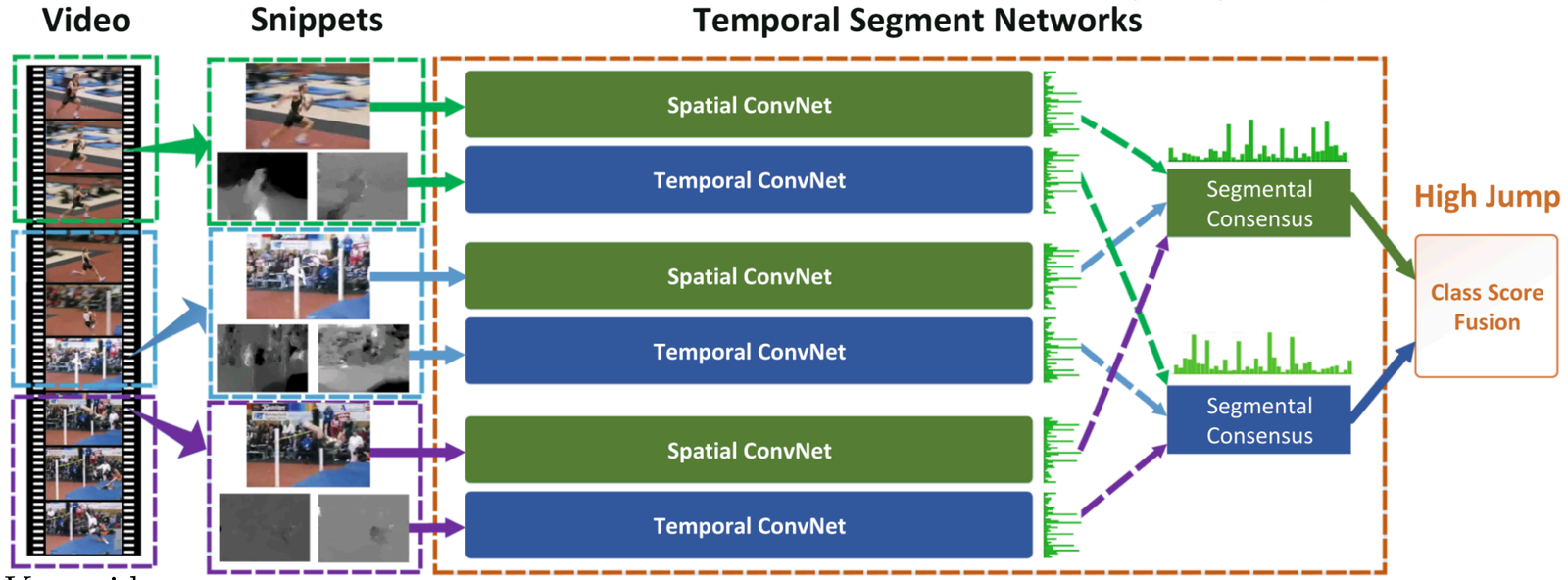
Wang, Limin, Yu Qiao, and Xiaoou Tang. "Action recognition with trajectory-pooled deep-convolutional descriptors." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

Temporal Segment Networks: Towards Good Practices for Deep Action Recognition

Boulder

Security and behavior analysis

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



$V \rightarrow$ video

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

$(T_1, T_2, \ldots, T_K) \rightarrow$ sequence of snippets

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

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

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

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

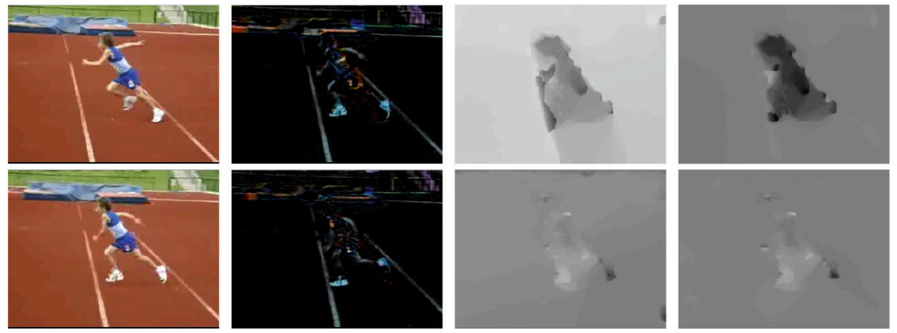
$\mathcal{H} \rightarrow$ prediction function (softmax)

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

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

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

Network Inputs



Spatial Stream (RGB images)

Temporal Stream (stacked optical flow fields)

RGB Difference

Warped Optical Flow Fields
(estimating holography matrix and compensating camera motion)

TVL1 optical flow algorithm
(implemented in OpenCV with CUDA)

UCF101 dataset (split 1)

Component	Basic Two-stream [1]	Cross-Modality pre-training
Accuracy	90.0 %	91.5

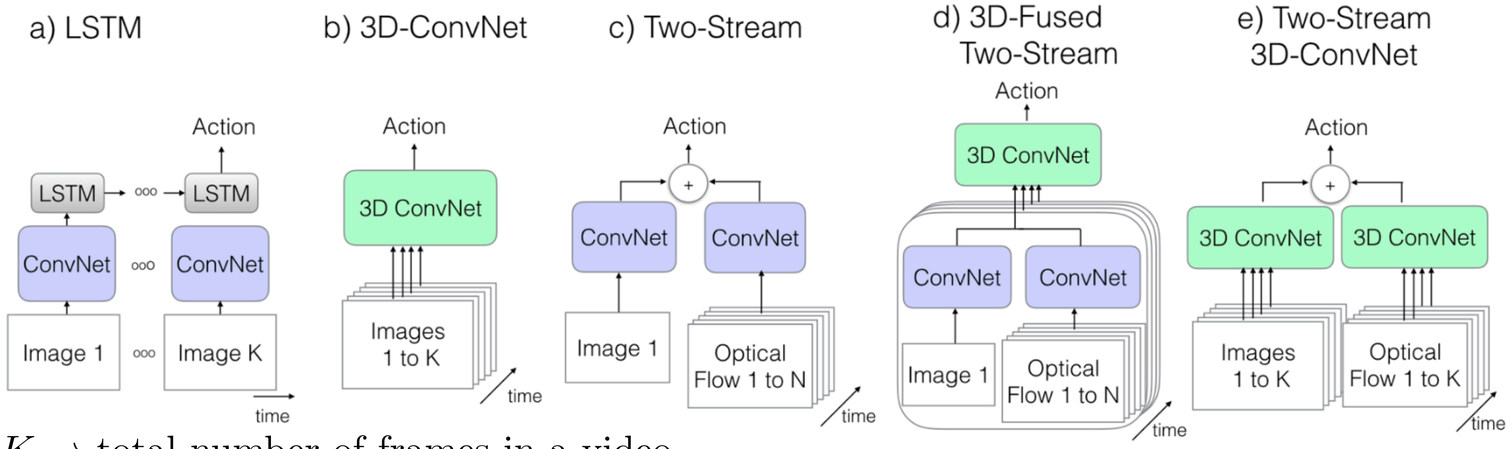
Partial BN with dropout	Temporal segment networks
92.0 %	93.5 %

Wang, Limin, et al. "Temporal segment networks: Towards good practices for deep action recognition." *European conference on computer vision*. Springer, Cham, 2016.

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

Boulder

YouTube Video



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

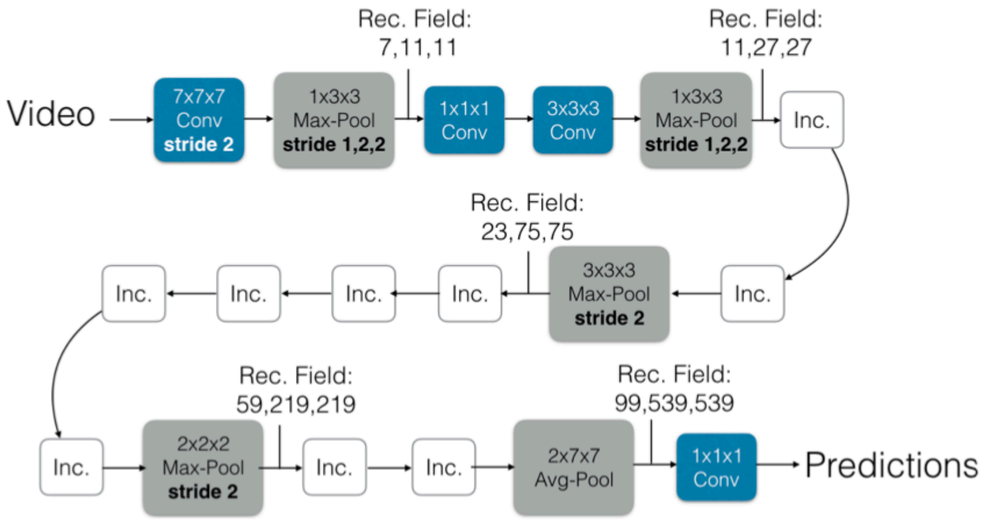
Two-stream Inflated 3D ConvNet (I3D)
Repeating the weights of the 2D filters N times along the time dimension, and rescaling them by dividing by N.
Train the two networks separately and
average their predictions at test time.

The Kinetics Human Action Video Dataset

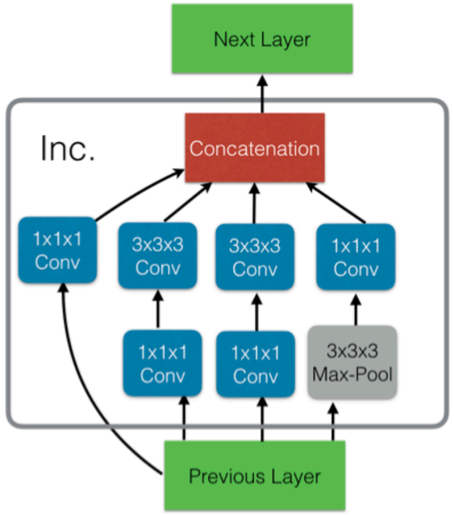
```
The list of action classes covers: Person Actions
(singular), e.g. drawing, drinking, laughing, punching;
Person-Person Actions, e.g. hugging, kissing, shaking
hands; and, Person-Object Actions, e.g. open- ing
presents, mowing lawn, washing dishes.
```

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

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**

Carreira, Joao, and Andrew Zisserman. "Quo vadis, action recognition? a new model and the kinetics dataset." *proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017.

Non-local Neural Networks

YouTube Video

– Kinetics and Charades datasets

$$y_i = \frac{1}{\mathcal{C}(x)} \sum_j f(x_i, x_j) g(x_j) \rightarrow \text{a generic non-local operation}$$

$i \rightarrow$ index of an output position (in space, time, or spacetime)

$j \rightarrow$ index that enumerates all possible positions

$x \rightarrow$ input signal (image, sequence, video; often their features)

$y \rightarrow$ output signal of the same size as x

$f \rightarrow$ scalar-valued pairwise function representing relationship such as affinity between i and all j

$g \rightarrow$ unary function computing a representation of the input signal at the position j

$\mathcal{C} \rightarrow$ normalizing factor

Instantiations

$g(x_j) = W_g x_j$

$f(x_i, x_j) = e^{x_i^T x_j} \rightarrow$ Gaussian

$\mathcal{C}(x) = \sum_j f(x_i, x_j)$

$f(x_i, x_j) = e^{\theta(x_i)^T \phi(x_j)} \rightarrow$ Embedded Gaussian

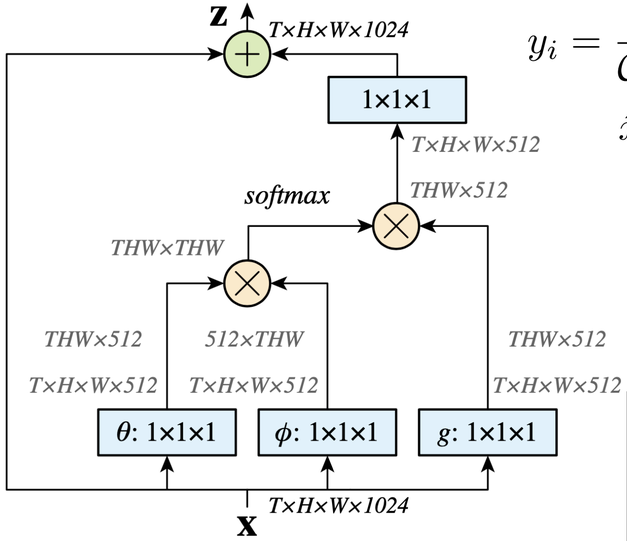
$\theta(x_i) = W_\theta x_i \qquad \phi(x_j) = W_\phi x_j$

$f(x_i, x_j) = \theta(x_i)^T \phi(x_j) \rightarrow$ Dot product

$\mathcal{C}(x) = N \rightarrow$ number of positions in x

$f(x_i, x_j) = \text{ReLU}(w_f^T [\theta(x_i), \phi(x_j)]) \rightarrow$ Concatenation

$z_i = W_z y_i + x_i \rightarrow$ non-local block

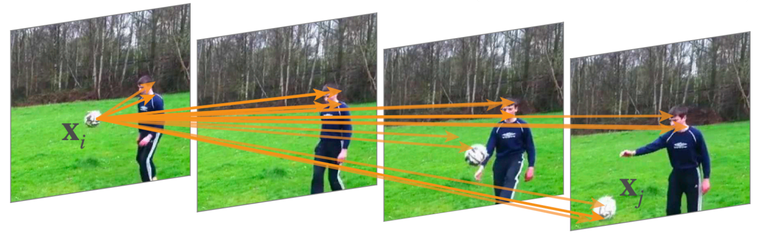


Baseline ResNet-50 C2D model for video

layer		output size
conv_1	7×7, 64, stride 2, 2, 2	16×112×112
pool_1	3×3×3 max, stride 2, 2, 2	8×56×56
res_2	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix} \times 3$	8×56×56
pool_2	3×1×1 max, stride 2, 1, 1	4×56×56
res_3	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix} \times 4$	4×28×28
res_4	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix} \times 6$	4×14×14
res_5	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix} \times 3$	4×7×7
global average pool, fc		1×1×1

$$y_i = \frac{1}{\mathcal{C}(\hat{x})} \sum_j f(x_i, \hat{x}_j) g(\hat{x}_j) \rightarrow \text{reduce computation}$$

$\hat{x} \rightarrow$ subsampled version of x (e.g., by pooling)



model, R50	top-1	top-5
C2D baseline	71.8	89.7
Gaussian	72.5	90.2
Gaussian, embed	72.7	**90.5**
dot-product	**72.9**	90.3
concatenation	72.8	**90.5**

model		top-1	top-5
R50	C2D baseline	71.8	89.7
	I3D	73.3	90.7
	NL I3D	**74.9**	**91.6**
R101	C2D baseline	73.1	91.0
	I3D	74.4	91.1
	NL I3D	**76.0**	**92.1**

Inflated 3D ConvNet (I3D)

model		top-1	top-5
R50	baseline	71.8	89.7
	space-only	72.9	90.8
	time-only	73.1	90.5
	spacetime	**73.8**	**91.0**
R101	baseline	73.1	91.0
	space-only	74.4	91.3
	time-only	74.4	90.5
	spacetime	**75.1**	**91.7**

keypoint detection

model	AP^{kp}	AP^{kp}_{50}	AP^{kp}_{75}
R101 baseline	65.1	86.8	70.4
NL, +4 in head	66.0	87.1	71.7
NL, +4 in head, +1 in backbone	**66.5**	**87.3**	**72.8**

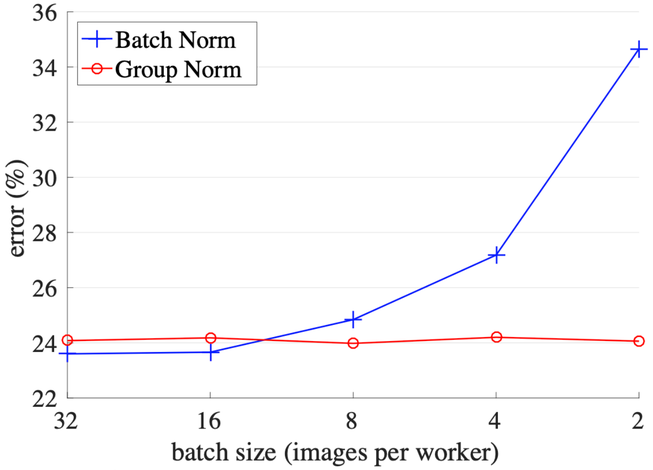
Adding 1 non-local block to Mask R-CNN for COCO

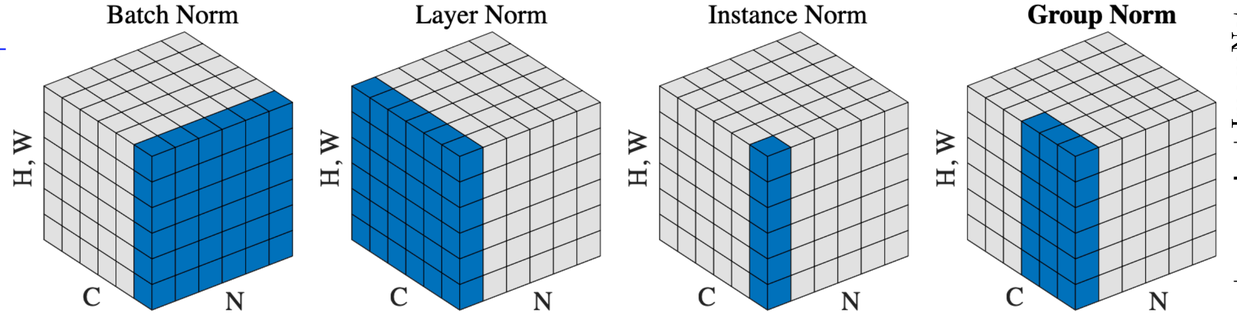
method		AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
R50	baseline	38.0	59.6	41.0	34.6	56.4	36.5
	+1 NL	**39.0**	**61.1**	**41.9**	**35.5**	**58.0**	**37.4**
R101	baseline	39.5	61.4	42.9	36.0	58.1	38.3
	+1 NL	**40.8**	**63.1**	**44.5**	**37.1**	**59.9**	**39.2**
X152	baseline	44.1	66.4	48.4	39.7	63.2	42.2
	+1 NL	**45.0**	**67.8**	**48.9**	**40.3**	**64.4**	**42.8**

Wang, Xiaolong, et al. "Non-local neural networks." Proceedings of the IEEE conference on computer vision and pattern recognition. 2018.

Group Normalization

YouTube Video





ImageNet

	BN	LN	IN	GN
val error	**23.6**	25.3	28.4	24.1
△ (*vs.* BN)	-	*1.7*	*4.8*	***0.5***

batch size	32	16	8	4	2
BN	**23.6**	**23.7**	24.8	27.3	34.7
GN	24.1	24.2	**24.0**	**24.2**	**24.1**
△	*0.5*	*0.5*	*-0.8*	*-3.1*	*-10.6*

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

$x \rightarrow$ 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

$\mathcal{S}_i = \{k : k_C = i_C\}$ $\implies$ μ & σ computed along (N, H, W) axes

Layer Norm

$\mathcal{S}_i = \{k : k_N = i_N\}$ $\implies$ μ & σ computed along (C, H, W) axes

Instance Norm

$\mathcal{S}_i = \{k : k_N = i_N, k_C = i_C\}$ $\implies$ μ & σ 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

$\mathcal{S}_i = \{k : k_N = i_N, \left\lfloor \frac{k_C}{C/G} \right\rfloor = \left\lfloor \frac{i_C}{C/G} \right\rfloor\}$ $\implies$ μ & σ computed along (H, W) axes and a group of C/G channels

$G = 32 \rightarrow$ number of groups

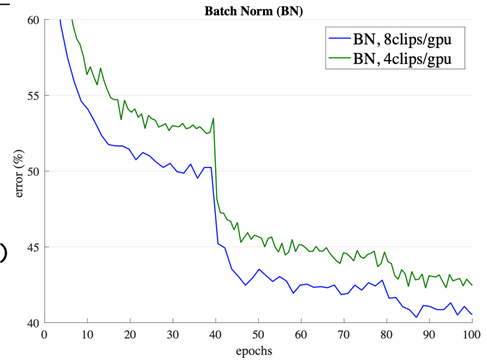
$C/G \rightarrow$ number of channels per group

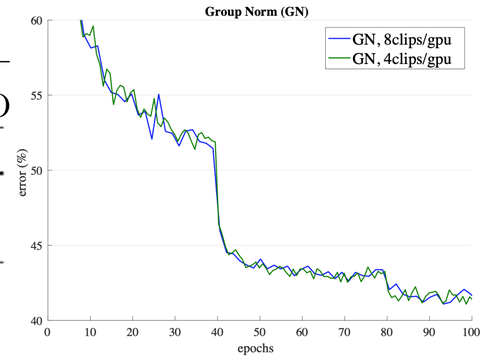
```
def GroupNorm(x, gamma, beta, G, eps=1e−5):
    # x: input features with shape [N,C,H,W]
    # gamma, beta: scale and offset, with shape [1,C,1,1]
    # G: number of groups for GN

    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^{bbox}_{50}	AP^{bbox}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
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**

Video Classification in Kinetics





clip length batch size	32 8	32 4	64 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**

Wu, Yuxin, and Kaiming He. "Group normalization." Proceedings of the European conference on computer vision (ECCV). 2018.

Learning Multi-Domain Convolutional Neural Networks for Visual Tracking

Boulder

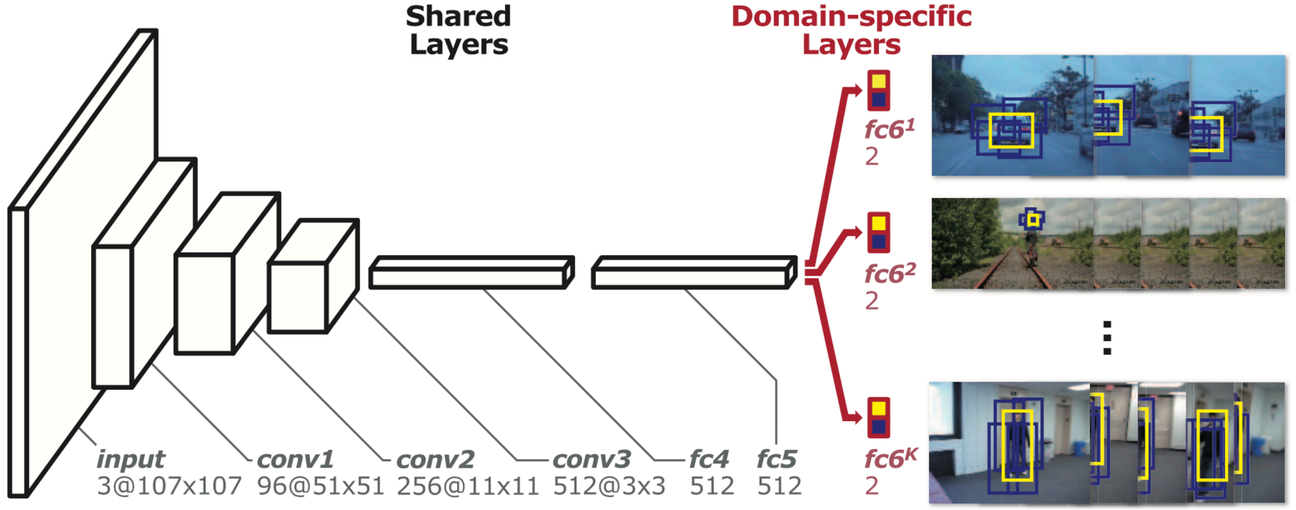
Multi-Domain Network (MDNet)

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.



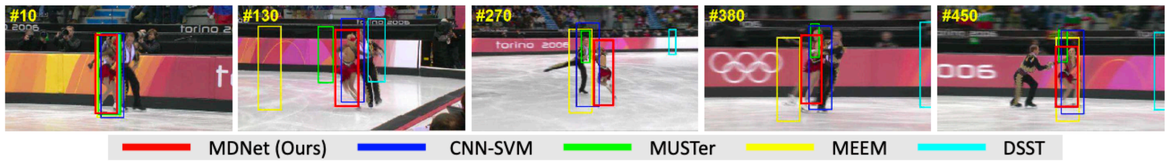
Online tracking algorithm

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



$x_t^1, \ldots, x_t^N \rightarrow$ 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

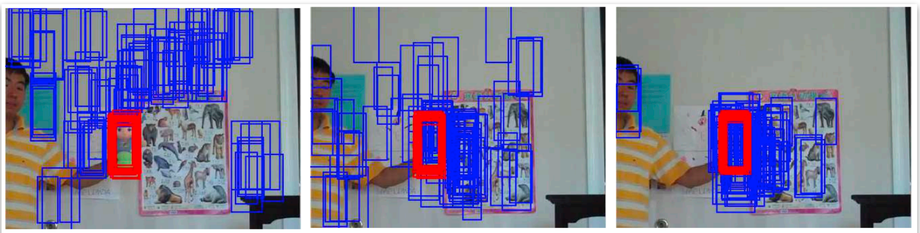
If $f^+(x_t^*) > 0.5$ then accept x_t^* as the target state and adjust it according to a simple linear bounding box regressor trained on the first frame only.

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.



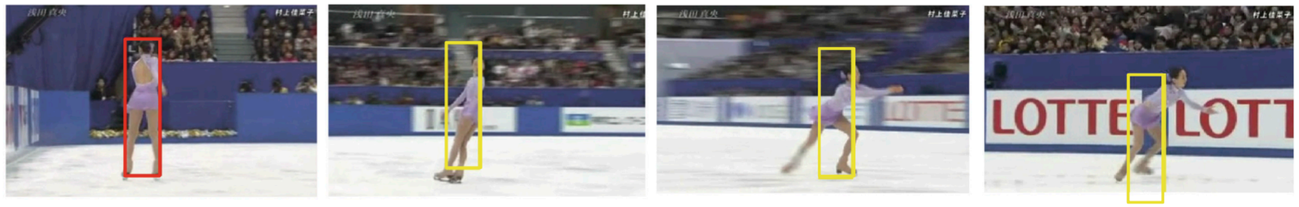
(a) 1st minibatch (b) 5th minibatch (c) 30th minibatch

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

Nam, Hyeonseob, and Bohyung Han. "Learning multi-domain convolutional neural networks for visual tracking." *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016.

Fully-Convolutional Siamese Networks for Object Tracking

https://www.youtube.com/watch?v=zS6kiI8DgDY



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
- $\varphi \rightarrow$ embedding
- $g \rightarrow$ a simple distance of a similarity metric

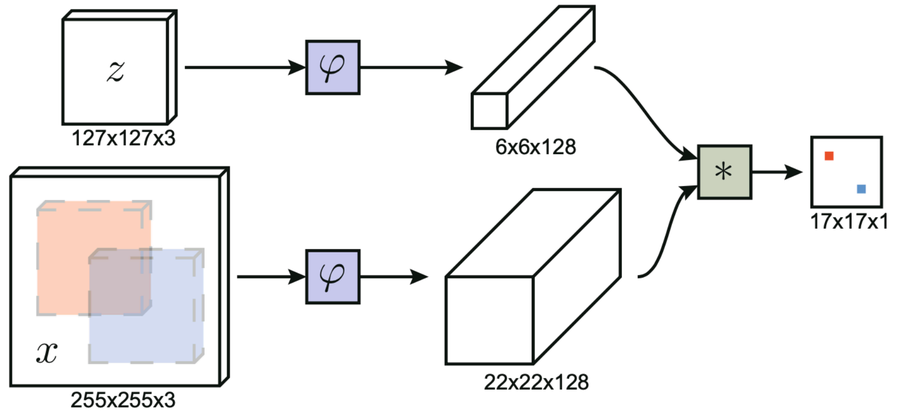
Fully-Convolutional Siamese Architecture

$h(L_{k\tau}x) = L_\tau h(x) \iff h$ 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 $\mathcal{D} \subset \mathbb{Z}^2$
- $\varphi \rightarrow$ convolutional embedding function
- $* \rightarrow$ 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

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

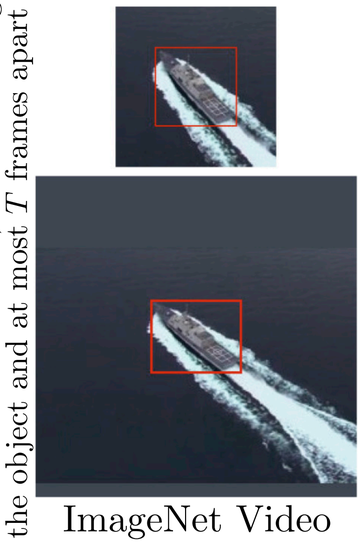
$\ell \rightarrow$ logistic loss

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

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

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

measured on VOT-15

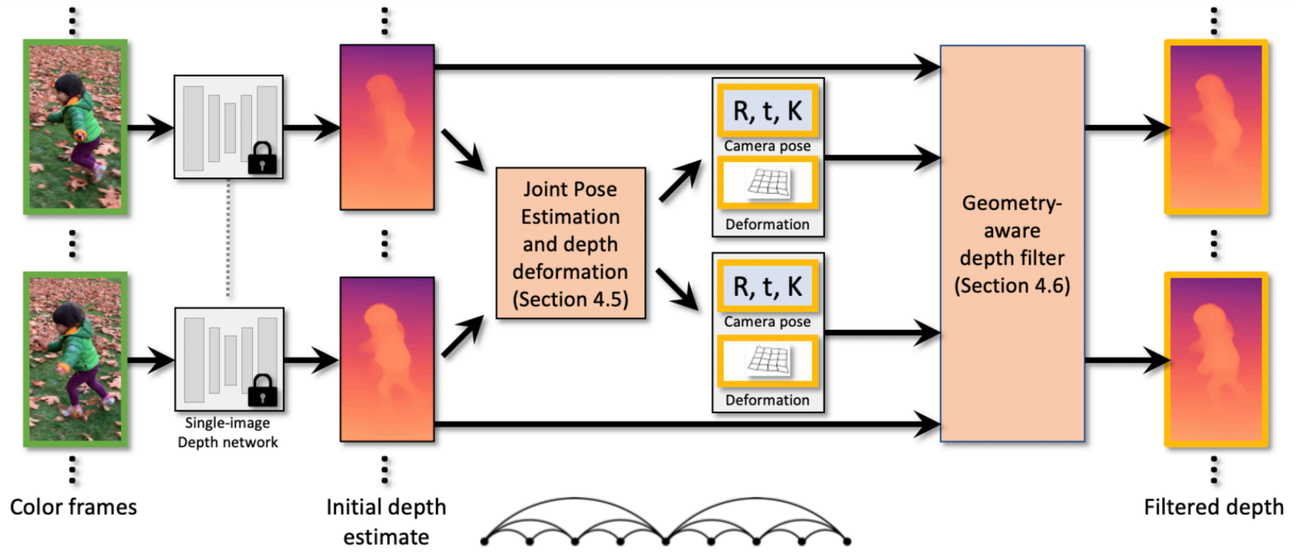


Two frames of a video, both containing the object and at most T frames apart

ImageNet Video

Bertinetto, Luca, et al. "Fully-convolutional siamese networks for object tracking." *European conference on computer vision*. Springer, Cham, 2016.

Robust Consistent Video Depth Estimation



$P = \{(i,j) : |i-j| = k, i \bmod k = 0, k = 1, 2, 4, \ldots\}$ → set of frame pairs

$f_{i\to j}$ → dense optical flow field
mapping a pixel in frame i to its corresponding location in frame j

$m^{\text{flow}}_{i\to j}$ → binary mask (forward-backward consistent flow pixels)

m^{dyn}_i → binary segmentation mask (Mask R-CNN)
indicating likely dynamic pixels ("person", "animal", "vehicle")

Depth Estimation

p → 2D pixel coordinate

$c_i(p) = s_i d_i(p)\tilde{p}$ → 3D coordinate in frame's i's 3D camera coordinate system

s_i → per-frame scale coefficient

d_i → CNN-estimated depth map

$\tilde{p} = [p_x, p_y, 1]^T$ → homogenous-augmented pixel coordinate

$c_{i\to j}(p) = K_j R_j^T (R_i K_i^{-1} c_i(p) + t_i - t_j)$ → projection of $c_i(p)$ into the camera coordinate system of another frame j

R_i, R_j → rotation of frames i and j

t_i, t_j → translation of frames i and j

K_i, K_j → intrinsics of frames i and j

$$\arg\min_{\theta^{\text{depth}}} \sum_{(i,j)\in P} \sum_{p\in m^{\text{flow}}_{i\to j}} \mathcal{L}^{\text{repro}}_{i\to j}(p), \quad \text{s.t. fixed } \theta^{\text{cam}}$$

$\mathcal{L}^{\text{repro}}_{i\to j}(p) = \mathcal{L}^{\text{sim}}_j(c_{i\to j}(p), c_j(f_{i\to j}(p)))$ → reprojection loss

$\theta^{\text{cam}} = \{R_i, t_i, K_i, s_i\}$ → camara parameters

θ^{depth} → depth CNN parameters

Joint Pose Estimation and Depth Deformation

Reverse the roles of θ^{depth} and θ^{cam}!

Replace s_i with $\phi_i(p) = \sum_k b_k(p) s_i^k$.

$\phi_i(p)$ → depth deformation model

s_i^k → scale factors (defined on a regular grid across the image)

$b_k(p)$ → bilinear coefficients

Geometry-aware Depth Filter

$$d_i^{\text{final}}(p) = \sum_{q\in N(p)} \sum_{j=i-\tau}^{i+\tau} z_{j\to i}(\tilde{f}_{i\to j}(q)) w_{i\to j}(q)$$

$z_{j\to i}$ → reprojected depth (scalar z-component of $c_{j\to i}$)

$\tilde{f}$ → flow field between far frames
(chaining flow maps between consecutive frames)

$w_{i\to j}(q) = \exp(-\lambda_f \mathcal{L}^{\text{ratio}}(c_i(p), c_{j\to i}(\tilde{f}_{i\to j}(q))))$

– Sintel –DAVIS – Cellphone Vides Datasets

$$\mathcal{L}^{sim}(a,b) = \mathcal{L}^{spatial}(a,b) + \mathcal{L}^{ratio}(a,b)$$

$$\mathcal{L}^{spatial}(a,b) = \left\| \frac{a_{xy}}{a_z} - \frac{b_{xy}}{b_z} \right\|_2^2$$

$$\mathcal{L}^{ratio}(a,b) = \frac{\max(a_z, b_z)}{\min(a_z, b_z)} - 1$$

Kopf, Johannes, et al. "Robust Consistent Video Depth Estimation." *ArXiv:2012.05901 [Cs]*, Dec. 2020. *arXiv.org*, http://arxiv.org/abs/2012.05901.

Boulder

Questions?

YouTube Playlist