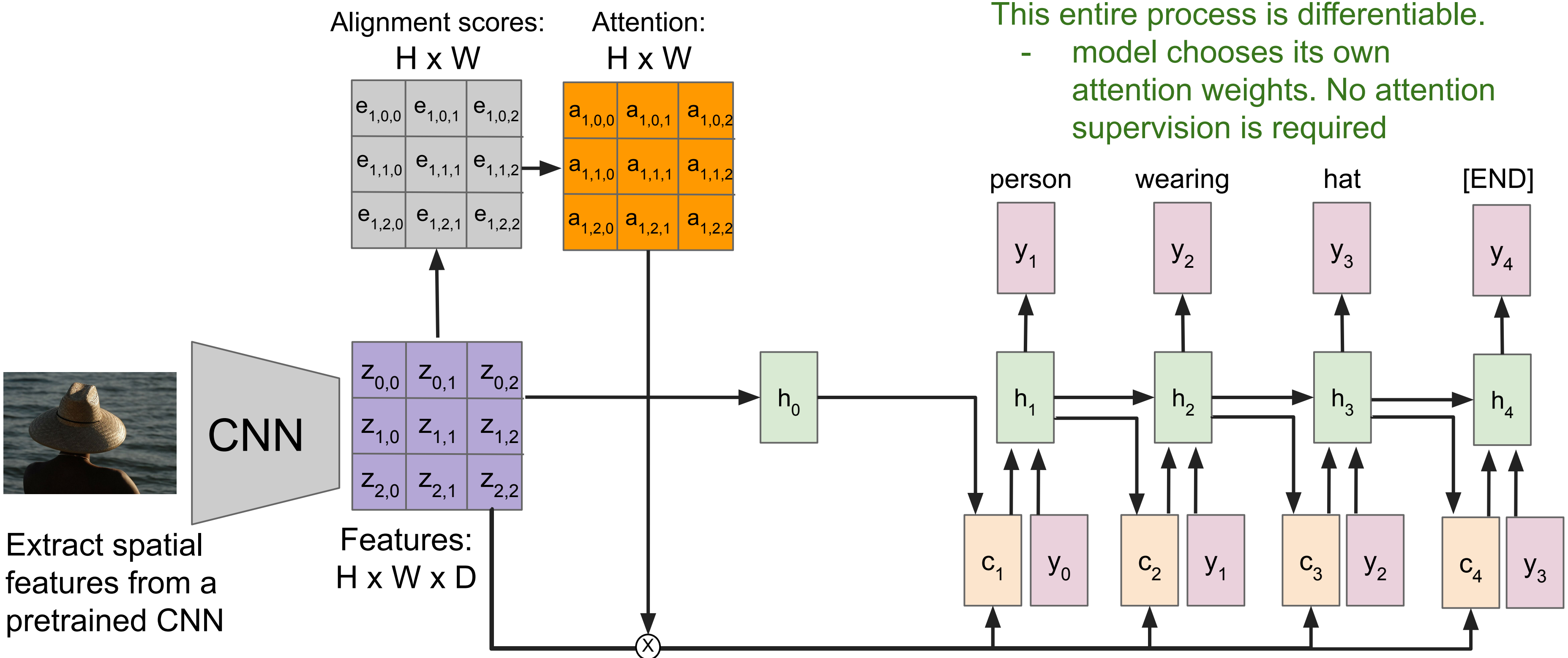
Lecture 10:
Video Understanding

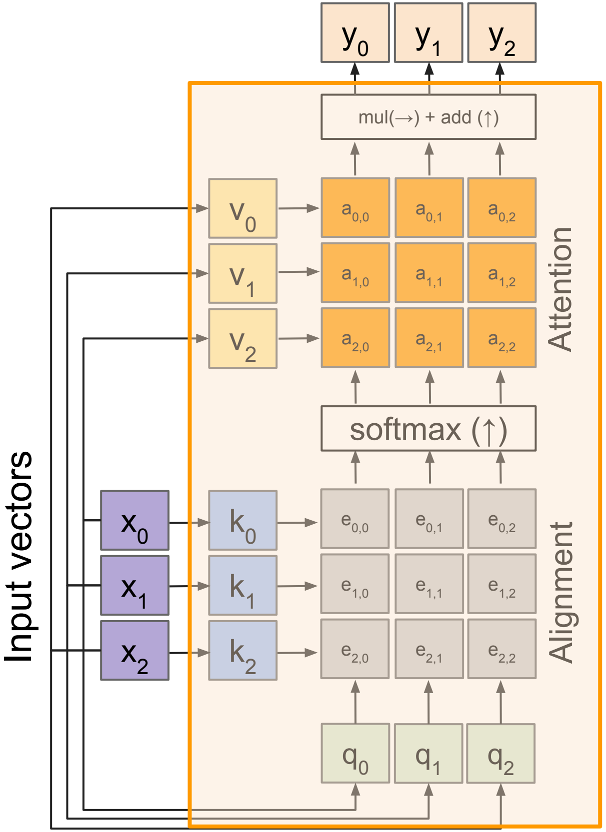
Last time: Image Captioning with RNNs and Attention

This entire process is differentiable.
- model chooses its own attention weights. No attention supervision is required



Xu et al, "Show, Attend and Tell: Neural Image Caption Generation with Visual Attention", ICML 2015

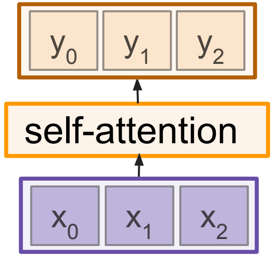
Last time: Self-Attention



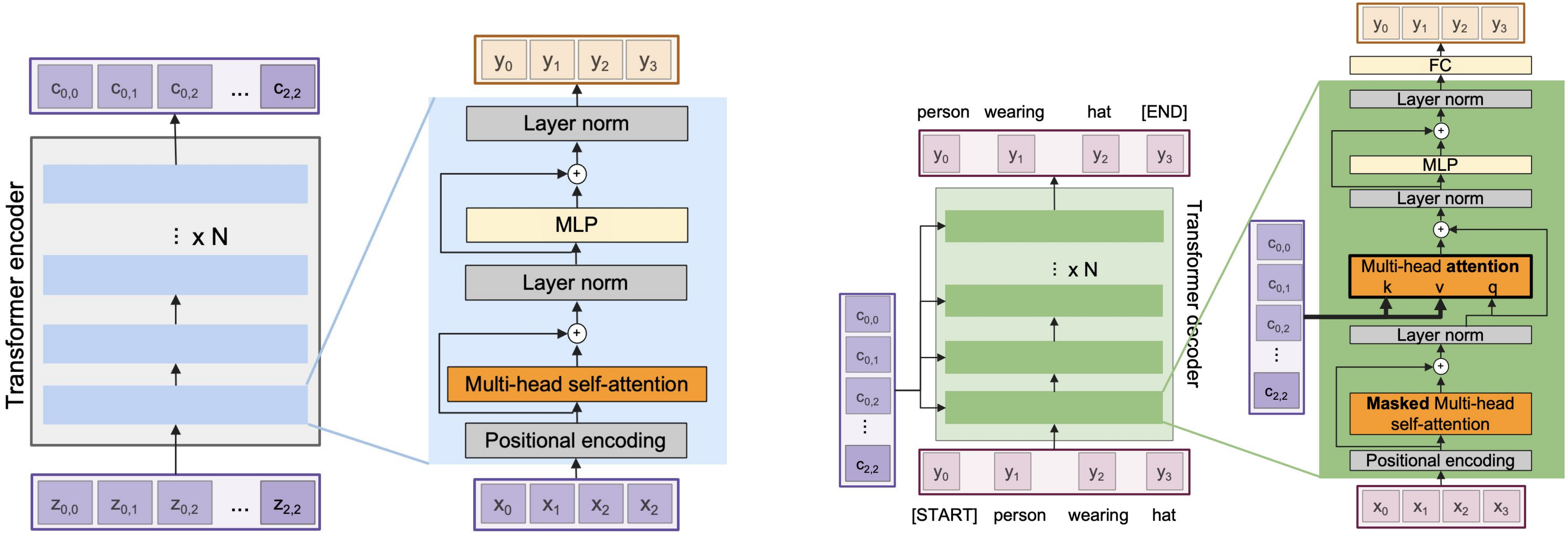
Outputs:
context vectors: $\mathbf{y}$ (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W}_k$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W}_v$
Query vectors: $\mathbf{q} = \mathbf{x}\mathbf{W}_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: $\mathbf{x}$ (shape: N x D)



Last time: Transformer



Encoder

Decoder

Recall: (2D) Image classification

This image by Nikita is licensed under CC-BY 2.0

(assume given a set of possible labels)
{dog, cat, truck, plane, ...}

→ cat

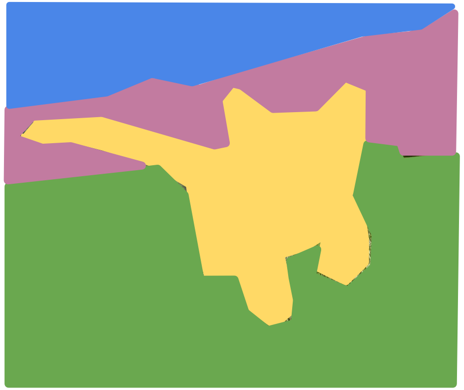
Next Lecture: (2D) Detection and Segmentation

Classification

CAT

No spatial extent

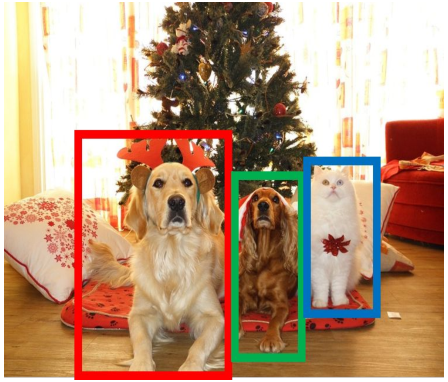
Semantic Segmentation



GRASS, CAT, TREE, SKY

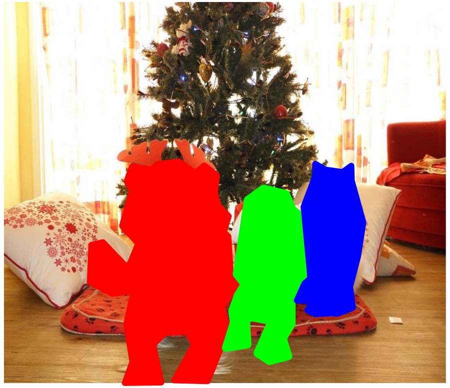
No objects, just pixels

Object Detection



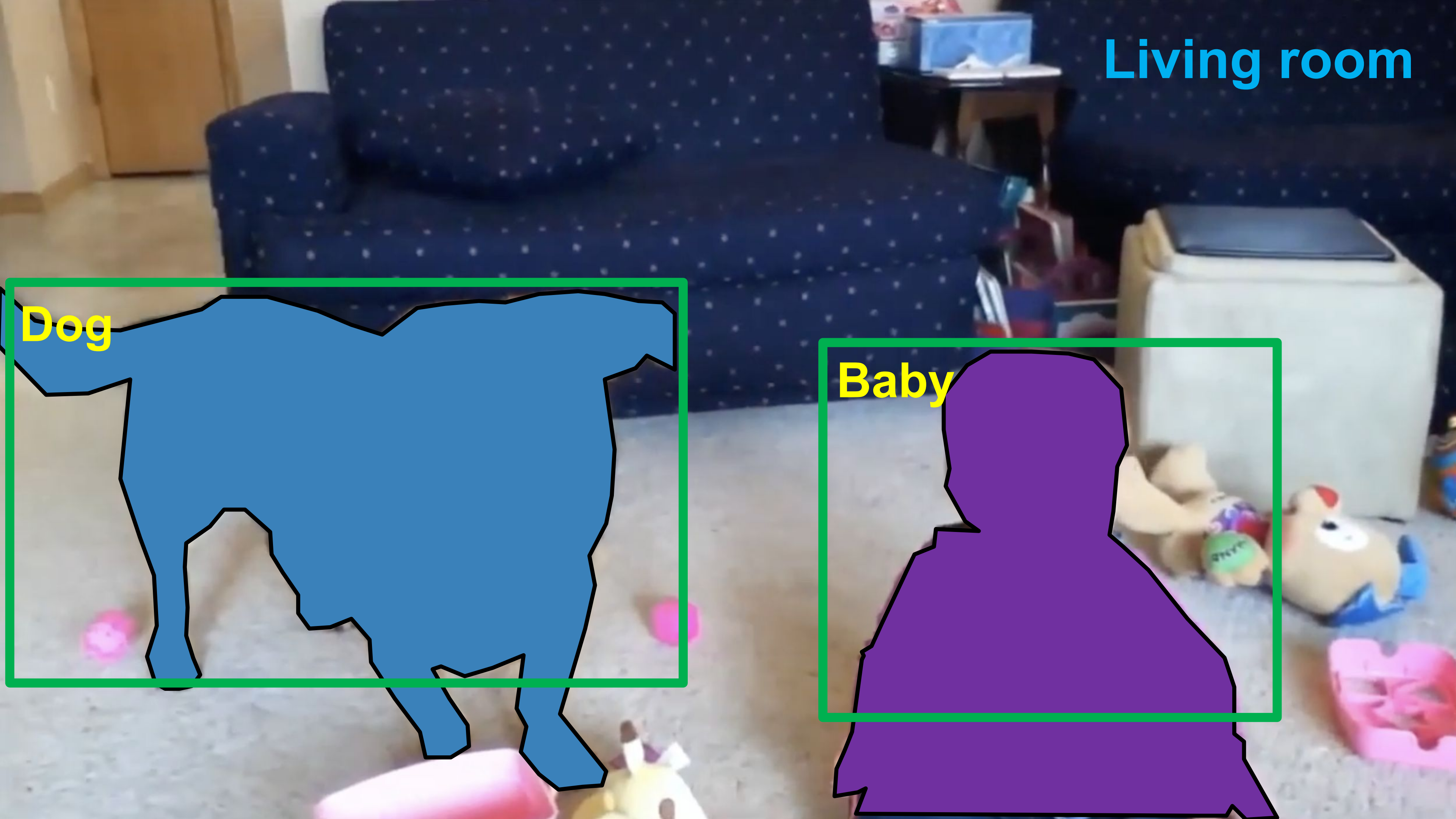
DOG, DOG, CAT

Instance Segmentation



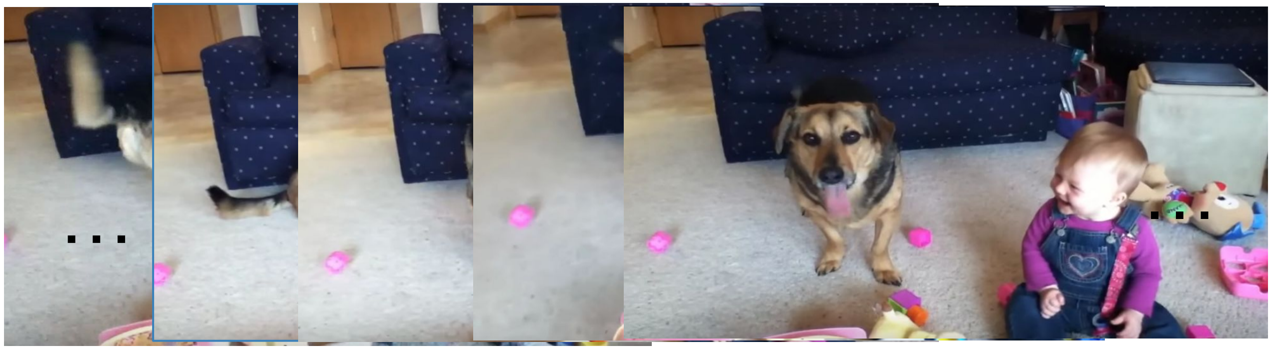
DOG, DOG, CAT

Multiple Objects


Living room
Dog
Baby

Today: **Video** = 2D + Time

A video is a **sequence** of images
4D tensor: T x 3 x H x W
(or 3 x T x H x W)



Example task: **Video Classification**

Input video:
T x 3 x H x W

Swimming
Running
Jumping
Eating
Standing

Running video is in the public domain

Slide credit: Justin Johnson

Example task: **Video Classification**

Images: Recognize **objects**

Dog
Cat
Fish
Truck

Videos: Recognize **actions**

Swimming
Running
Jumping
Eating
Standing

Running video is in the public domain

Slide credit: Justin Johnson

Problem: Videos are big!

Input video:
T x 3 x H x W

Videos are ~30 frames per second (fps)

Size of uncompressed video
(3 bytes per pixel):

SD (640 x 480): **~1.5 GB per minute**
HD (1920 x 1080): **~10 GB per minute**

Slide credit: Justin Johnson

Problem: Videos are big!

Input video:
T x 3 x H x W

Videos are ~30 frames per second (fps)

Size of uncompressed video
(3 bytes per pixel):

SD (640 x 480): **~1.5 GB per minute**
HD (1920 x 1080): **~10 GB per minute**

Solution: Train on short **clips:** low fps and low spatial resolution
e.g. T = 16, H=W=112
(3.2 seconds at 5 fps, 588 KB)

Slide credit: Justin Johnson

Training on Clips

Raw video: Long, high FPS

Slide credit: Justin Johnson

Training on Clips

Raw video: Long, high FPS

Training: Train model to classify short **clips** with low FPS

Slide credit: Justin Johnson

Training on Clips

Raw video: Long, high FPS

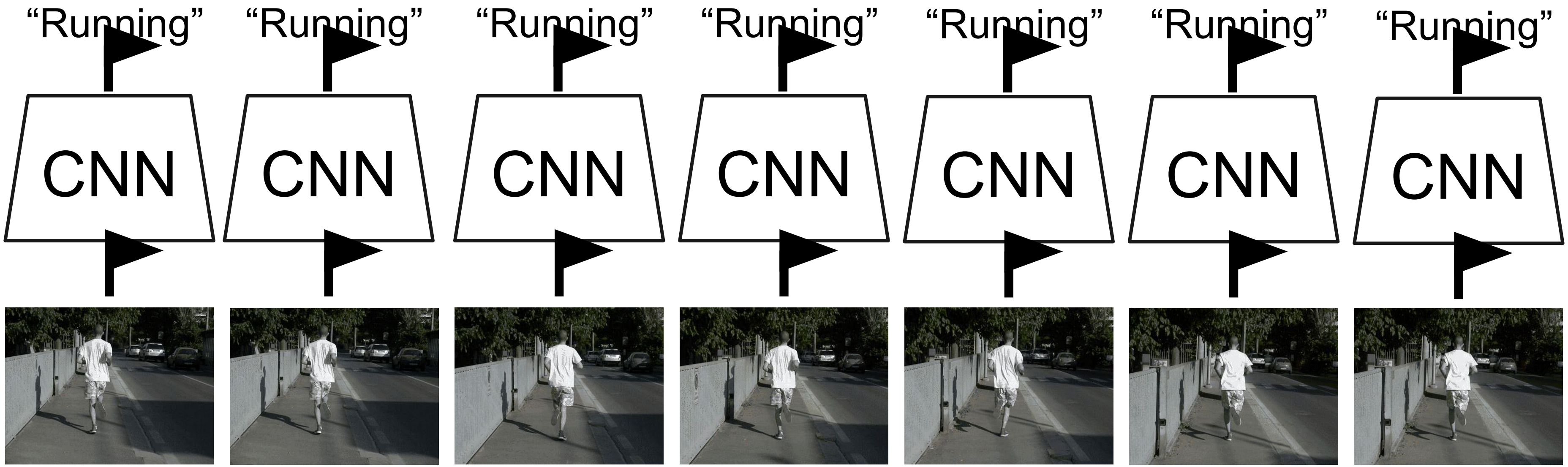
Training: Train model to classify short **clips** with low FPS

Testing: Run model on different clips, average predictions

Slide credit: Justin Johnson

Video Classification: Single-Frame CNN

Simple idea: train normal 2D CNN to classify video frames independently!
(Average predicted probs at test-time)
Often a **very** strong baseline for video classification



Slide credit: Justin Johnson

Video Classification: Late Fusion (with FC layers)

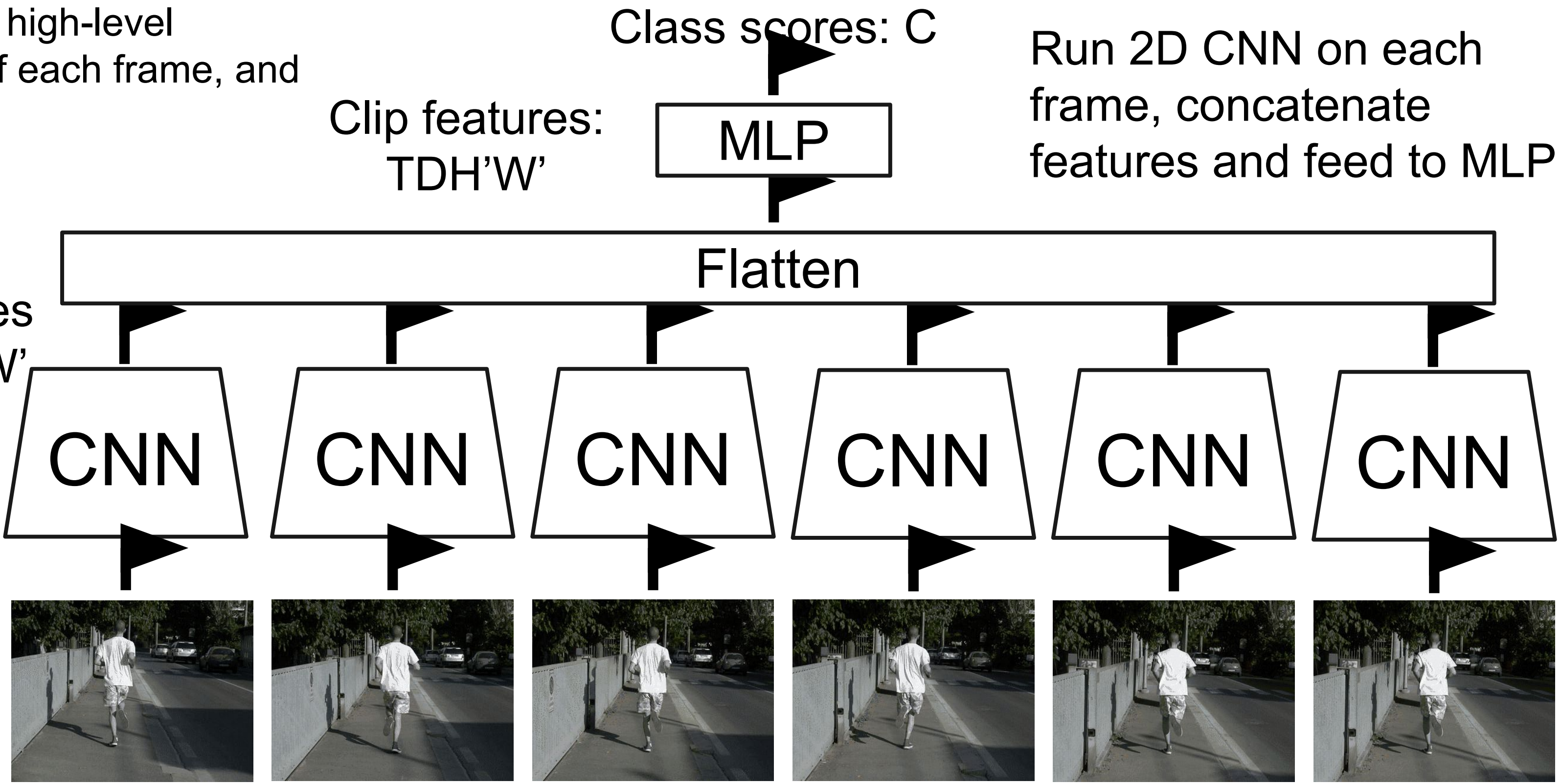
Intuition: Get high-level appearance of each frame, and combine them

Run 2D CNN on each frame, concatenate features and feed to MLP

Frame features
T x D x H' x W'

2D CNN on each frame

Input:
T x 3 x H x W

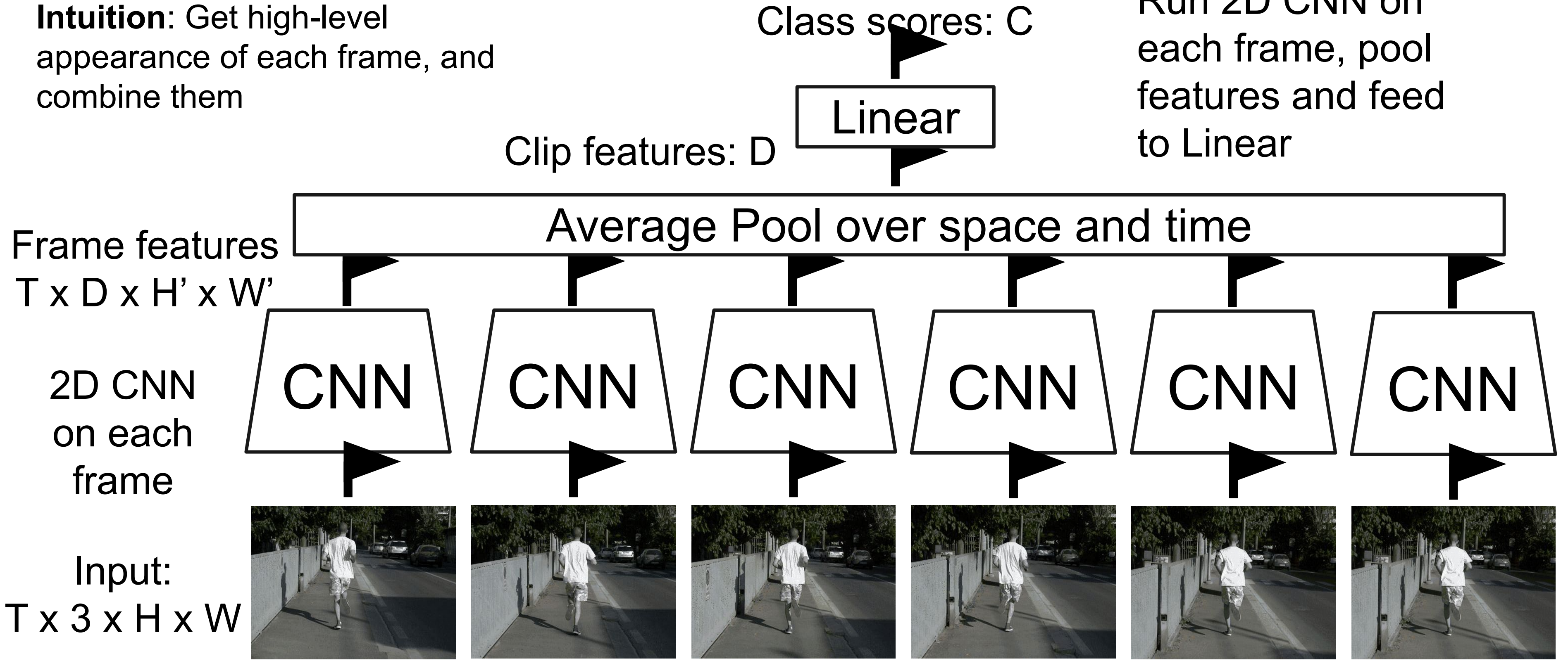


Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Slide credit: Justin Johnson

Video Classification: Late Fusion (with pooling)

Intuition: Get high-level appearance of each frame, and combine them



Run 2D CNN on each frame, pool features and feed to Linear

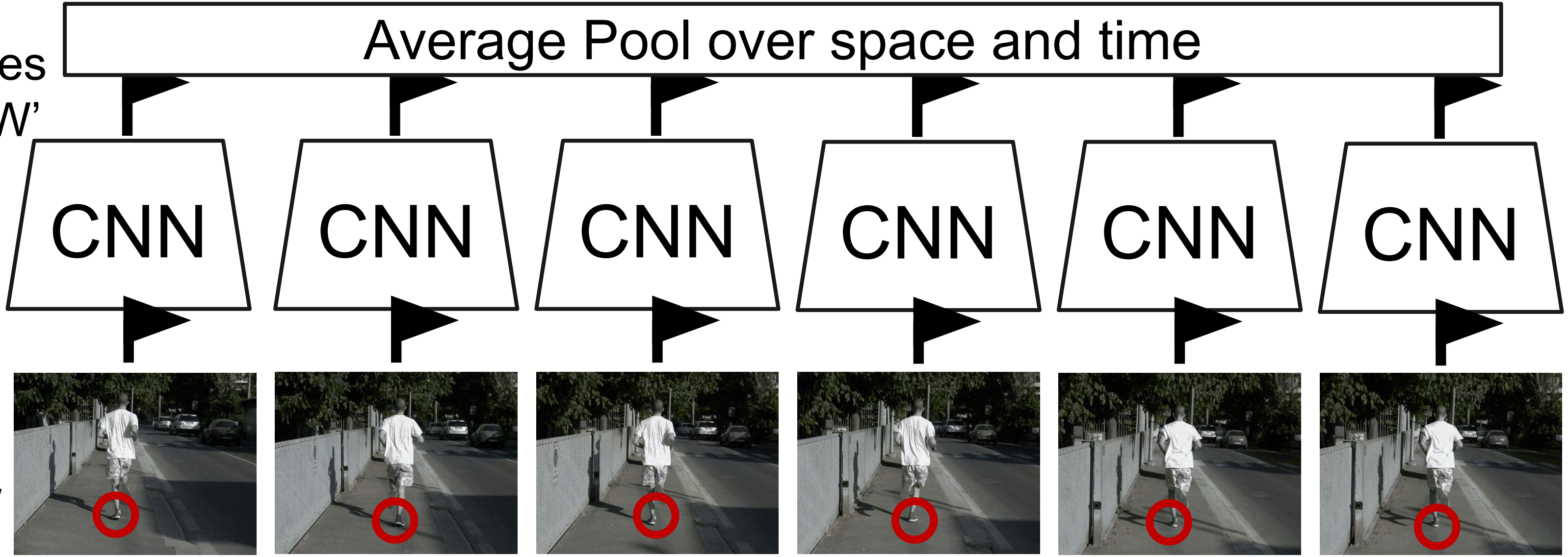
Slide credit: Justin Johnson

Video Classification: Late Fusion (with pooling)

Intuition: Get high-level appearance of each frame, and combine them

Problem: Hard to compare low-level motion between frames

Run 2D CNN on each frame, pool features and feed to Linear

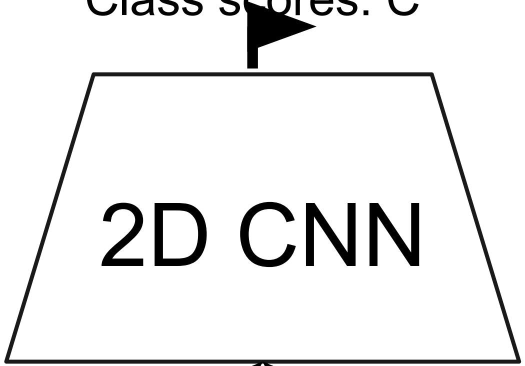


Slide credit: Justin Johnson

Video Classification: Early Fusion

Intuition: Compare frames with very first conv layer, after that normal 2D CNN

Class scores: C

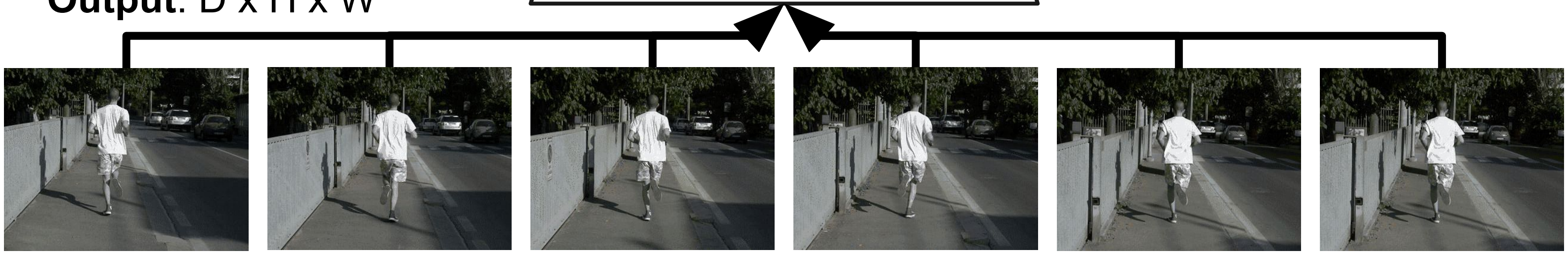


Rest of the network is standard 2D CNN

First 2D convolution collapses all temporal information:
Input: 3T x H x W
Output: D x H x W

Reshape:
3T x H x W

Input:
T x 3 x H x W



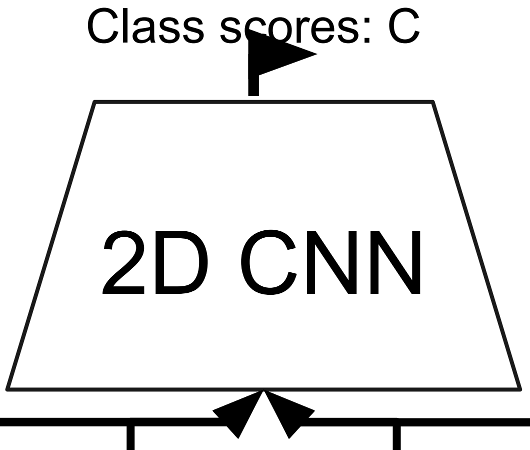
Karpathy et al, “Large-scale Video Classification with Convolutional Neural Networks”, CVPR 2014

Slide credit: Justin Johnson

Video Classification: Early Fusion

Intuition: Compare frames with very first conv layer, after that normal 2D CNN

Problem: One layer of temporal processing may not be enough!

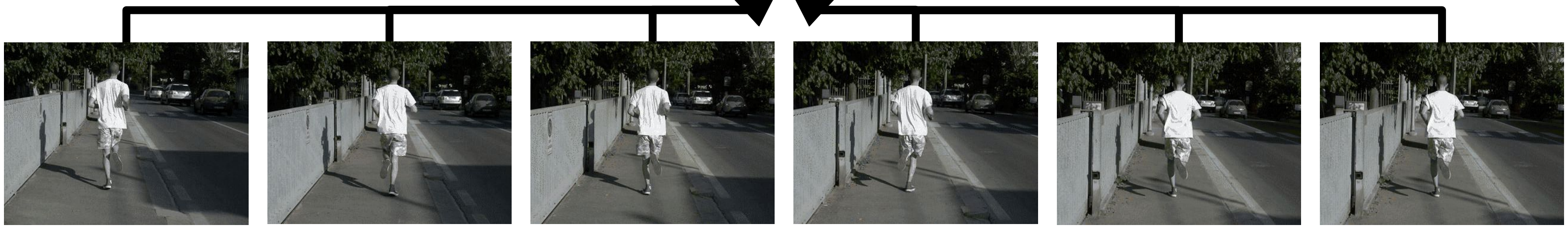


Rest of the network is standard 2D CNN

First 2D convolution collapses all temporal information:
Input: 3T x H x W
Output: D x H x W

Reshape:
3T x H x W

Input:
T x 3 x H x W

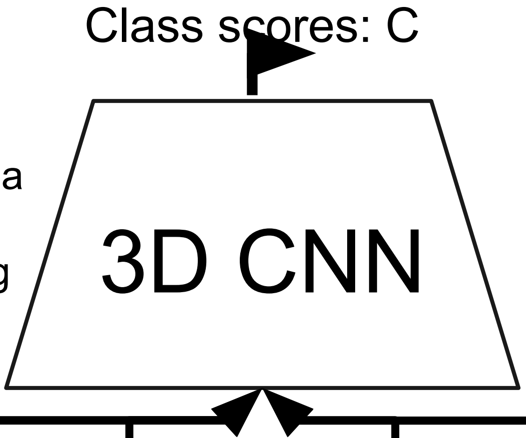


Karpathy et al, “Large-scale Video Classification with Convolutional Neural Networks”, CVPR 2014

Slide credit: Justin Johnson

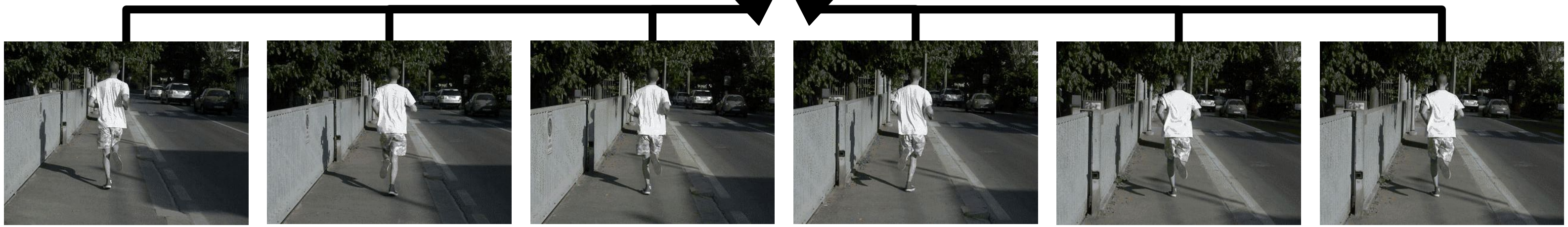
Video Classification: 3D CNN

Intuition: Use 3D versions of convolution and pooling to slowly fuse temporal information over the course of the network



Each layer in the network is a 4D tensor: D x T x H x W
Use 3D conv and 3D pooling operations

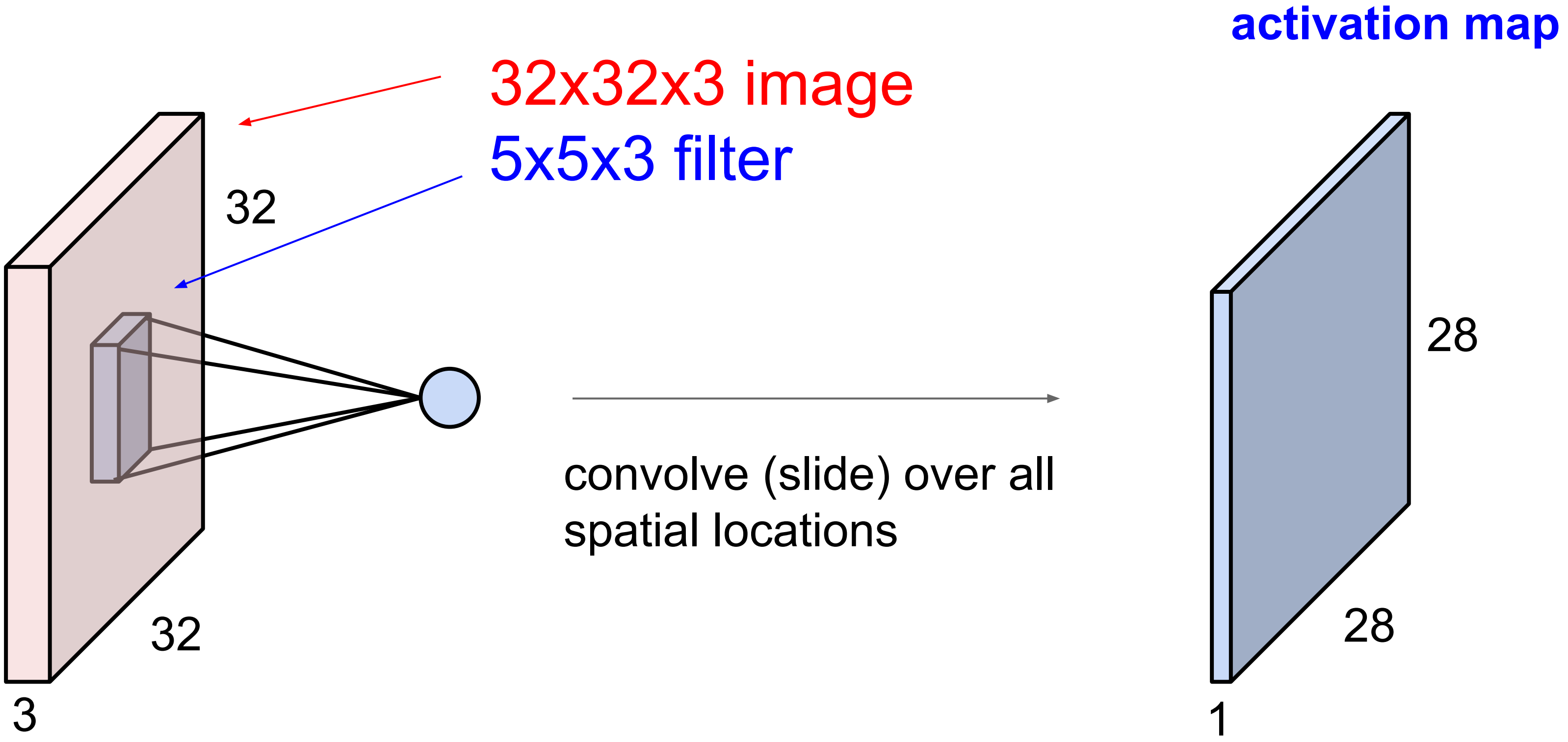
Input:
3 x T x H x W



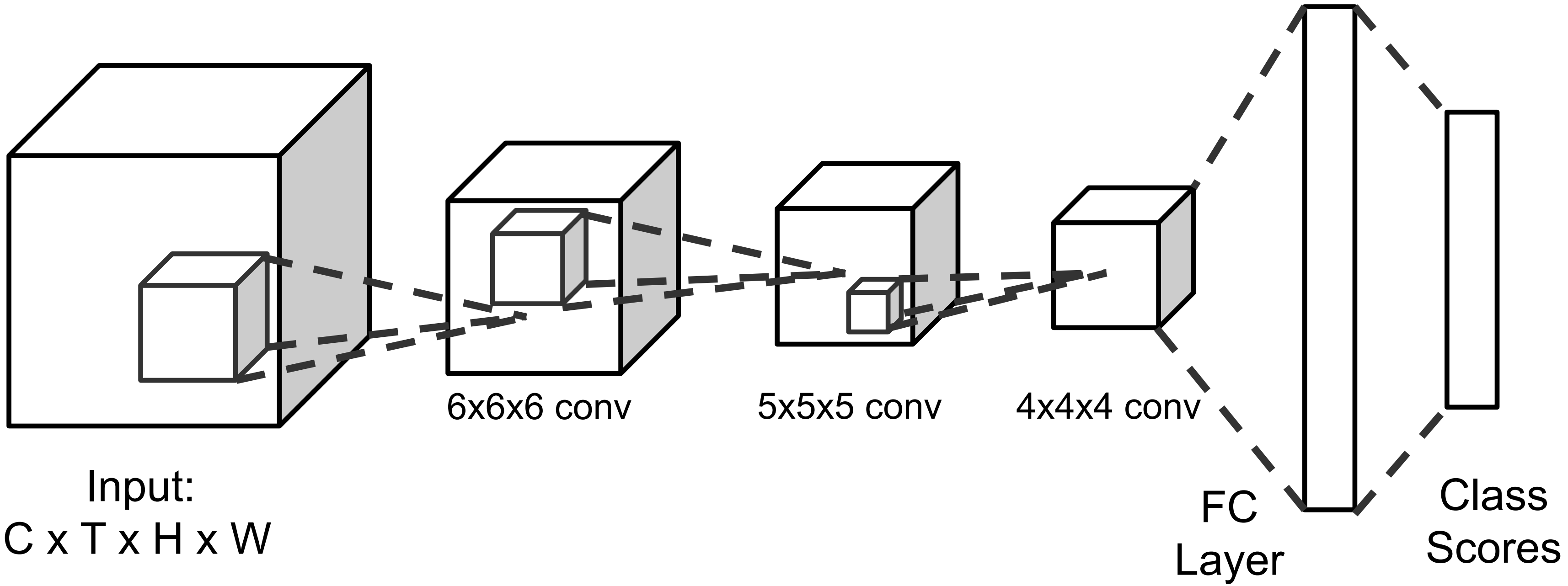
Ji et al, “3D Convolutional Neural Networks for Human Action Recognition”, TPAMI 2010 ; Karpathy et al, “Large-scale Video Classification with Convolutional Neural Networks”, CVPR 2014

Slide credit: Justin Johnson

Convolution Layer



3D Convolution



Early Fusion vs Late Fusion vs 3D CNN

Late Fusion

Layer	Size (C x T x H x W)	Receptive Field (T x H x W)
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3

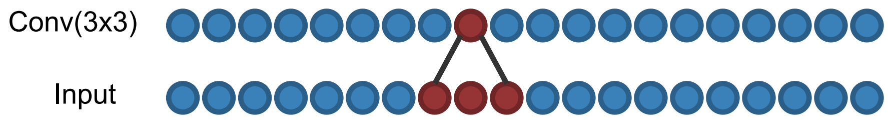
(Small example architectures, in practice much bigger)

Slide credit: Justin Johnson

Early Fusion vs Late Fusion vs 3D CNN

Late Fusion

Layer	Size (C x T x H x W)	Receptive Field (T x H x W)
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3



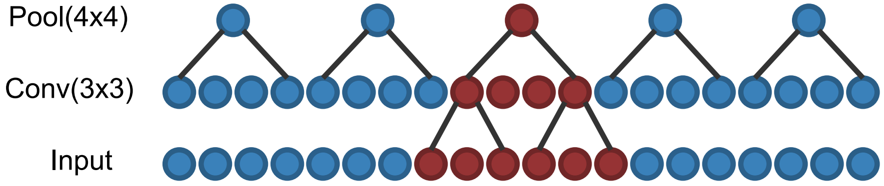
(Small example architectures, in practice much bigger)

Slide credit: Justin Johnson

Early Fusion vs Late Fusion vs 3D CNN

Late Fusion

Layer	Size (C x T x H x W)	Receptive Field (T x H x W)
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3
Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6



(Small example architectures, in practice much bigger)

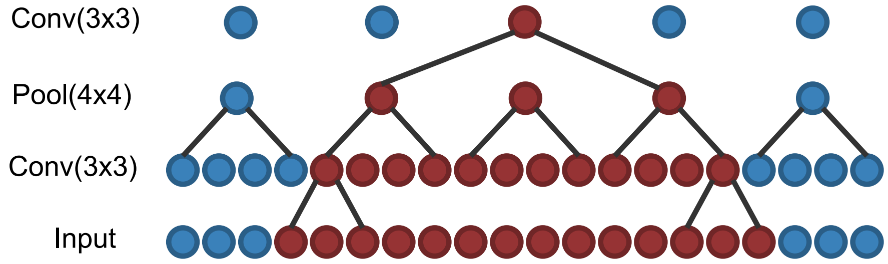
Slide credit: Justin Johnson

Early Fusion vs Late Fusion vs 3D CNN

Late Fusion

Layer	Size (C x T x H x W)	Receptive Field (T x H x W)
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3
Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6
Conv2D(3x3, 12->24)	24 x 20 x 16 x 16	1 x 14 x 14

Build slowly in space



(Small example architectures, in practice much bigger)

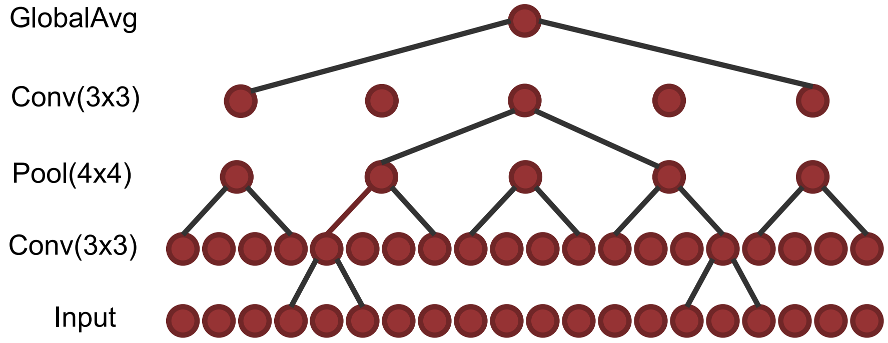
Slide credit: Justin Johnson

Early Fusion vs Late Fusion vs 3D CNN

Late Fusion

Layer	Size (C x T x H x W)	Receptive Field (T x H x W)
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3
Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6
Conv2D(3x3, 12->24)	24 x 20 x 16 x 16	1 x 14 x 14
GlobalAvgPool	24 x 1 x 1 x 1	20 x 64 x 64

Build slowly in space,
All-at-once in time at end



(Small example architectures, in practice much bigger)

Slide credit: Justin Johnson

Early Fusion vs Late Fusion vs 3D CNN

	Layer	Size (C x T x H x W)	Receptive Field (T x H x W)
Late Fusion	Input	3 x 20 x 64 x 64	
	Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3
	Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6
	Conv2D(3x3, 12->24)	24 x 20 x 16 x 16	1 x 14 x 14
	GlobalAvgPool	24 x 1 x 1 x 1	20 x 64 x 64
Early Fusion	Input	3 x 20 x 64 x 64	
	Conv2D(3x3, 3*20->12)	12 x 64 x 64	20 x 3 x 3
	Pool2D(4x4)	12 x 16 x 16	20 x 6 x 6
	Conv2D(3x3, 12->24)	24 x 16 x 16	20 x 14 x 14
	GlobalAvgPool	24 x 1 x 1	20 x 64 x 64

Build slowly in space,
All-at-once in time at end

Build slowly in space,
All-at-once in time at start

(Small example architectures, in practice much bigger)

Slide credit: Justin Johnson

Early Fusion vs Late Fusion vs 3D CNN

	Layer	Size (C x T x H x W)	Receptive Field (T x H x W)	
Late Fusion	Input	3 x 20 x 64 x 64		Build slowly in space, All-at-once in time at end
	Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3	
	Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6	
	Conv2D(3x3, 12->24)	24 x 20 x 16 x 16	1 x 14 x 14	
	GlobalAvgPool	24 x 1 x 1 x 1	20 x 64 x 64	
Early Fusion	Input	3 x 20 x 64 x 64		Build slowly in space, All-at-once in time at start
	Conv2D(3x3, 3*20->12)	12 x 64 x 64	20 x 3 x 3	
	Pool2D(4x4)	12 x 16 x 16	20 x 6 x 6	
	Conv2D(3x3, 12->24)	24 x 16 x 16	20 x 14 x 14	
	GlobalAvgPool	24 x 1 x 1	20 x 64 x 64	
3D CNN	Input	3 x 20 x 64 x 64		Build slowly in space, Build slowly in time "Slow Fusion"
	Conv3D(3x3x3, 3->12)	12 x 20 x 64 x 64	3 x 3 x 3	
	Pool3D(4x4x4)	12 x 5 x 16 x 16	6 x 6 x 6	
	Conv3D(3x3x3, 12->24)	24 x 5 x 16 x 16	14 x 14 x 14	
	GlobalAvgPool	24 x 1 x 1	20 x 64 x 64	

(Small example architectures, in practice much bigger)

Slide credit: Justin Johnson

Early Fusion vs Late Fusion vs 3D CNN

What is the difference?

	Layer	Size (C x T x H x W)	Receptive Field (T x H x W)
Late Fusion	Input	3 x 20 x 64 x 64	
	Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3
	Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6
	Conv2D(3x3, 12->24)	24 x 20 x 16 x 16	1 x 14 x 14
	GlobalAvgPool	24 x 1 x 1 x 1	20 x 64 x 64
Early Fusion	Input	3 x 20 x 64 x 64	
	Conv2D(3x3, 3*20->12)	12 x 64 x 64	20 x 3 x 3
	Pool2D(4x4)	12 x 16 x 16	20 x 6 x 6
	Conv2D(3x3, 12->24)	24 x 16 x 16	20 x 14 x 14
	GlobalAvgPool	24 x 1 x 1	20 x 64 x 64
3D CNN	Input	3 x 20 x 64 x 64	
	Conv3D(3x3x3, 3->12)	12 x 20 x 64 x 64	3 x 3 x 3
	Pool3D(4x4x4)	12 x 5 x 16 x 16	6 x 6 x 6
	Conv3D(3x3x3, 12->24)	24 x 5 x 16 x 16	14 x 14 x 14
	GlobalAvgPool	24 x 1 x 1	20 x 64 x 64

Late Fusion: Build slowly in space,
All-at-once in time at end

Early Fusion: Build slowly in space,
All-at-once in time at start

3D CNN: Build slowly in space,
Build slowly in time
"Slow Fusion"

(Small example architectures, in practice much bigger)

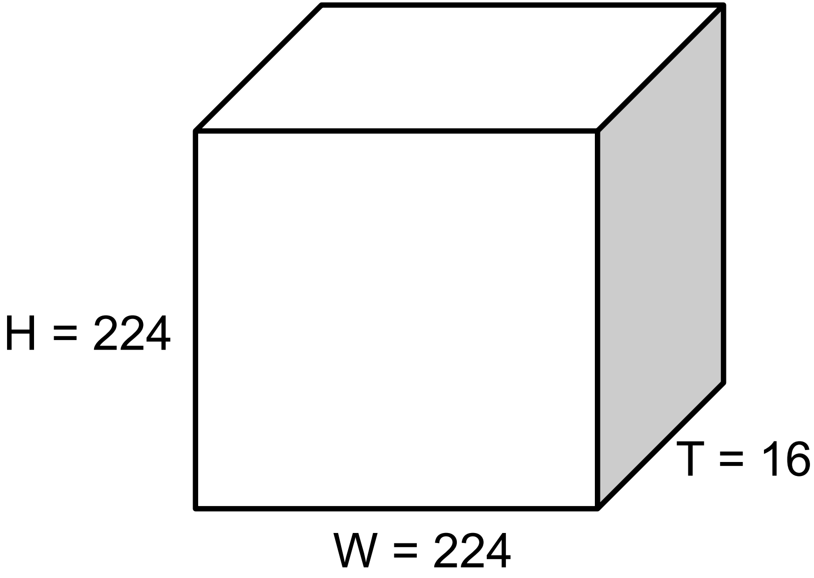
Slide credit: Justin Johnson

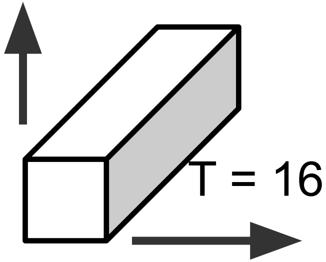
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)

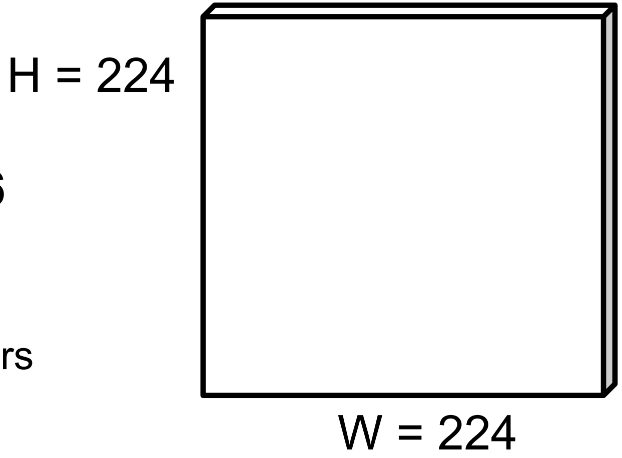
Weight:
C_{out} x C_{in} x T x 3 x 3
Slide over x and y

Output:
C_{out} x H x W
2D grid with C_{out} – dim feat at each point





C_{out} different filters



Slide credit: Justin Johnson

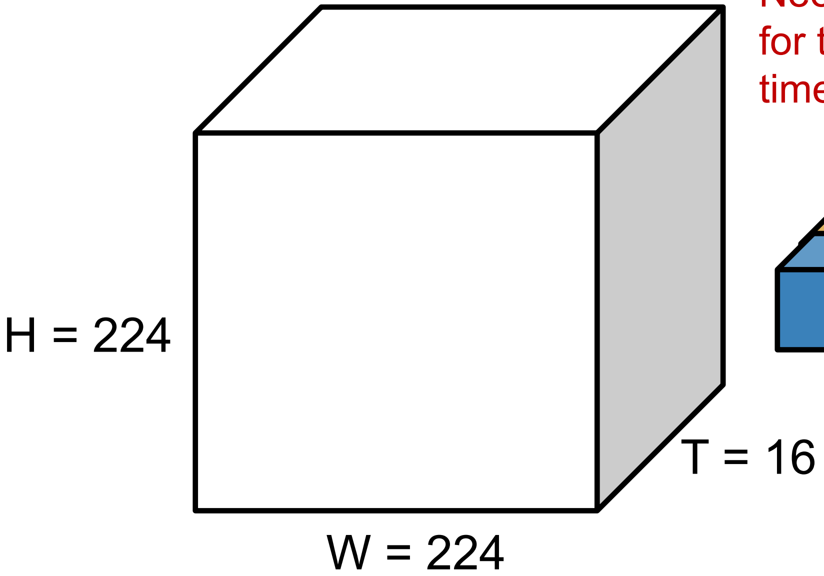
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

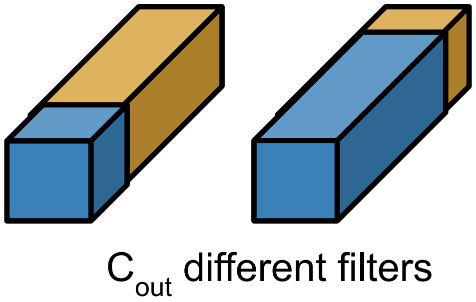
Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)

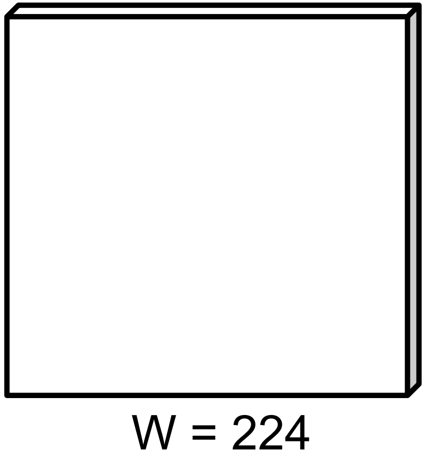
Weight:
C_{out} x C_{in} x T x 3 x 3
Slide over x and y

No temporal shift-invariance!
Needs to learn separate filters for the same motion at different times in the clip

Output:
C_{out} x H x W
2D grid with C_{out} –dim feat at each point







Slide credit: Justin Johnson

2D Conv (Early Fusion) vs 3D Conv (3D CNN)

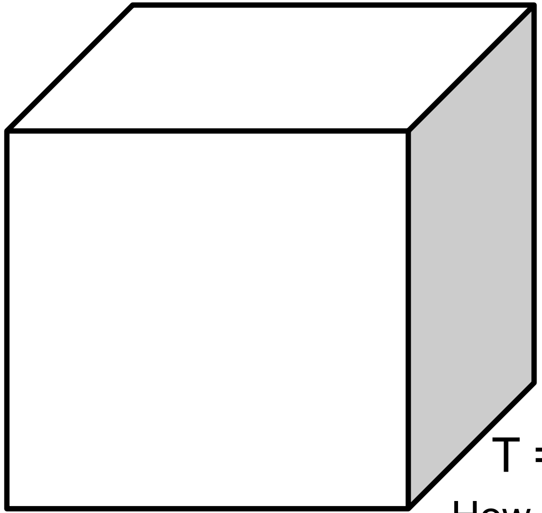
Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)

Weight:
C_{out} x C_{in} x T x 3 x 3
Slide over x and y

No temporal shift-invariance! Needs to learn separate filters for the same motion at different times in the clip

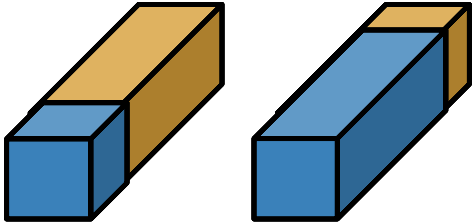
Output:
C_{out} x H x W
2D grid with C_{out} –dim feat at each point

H = 224



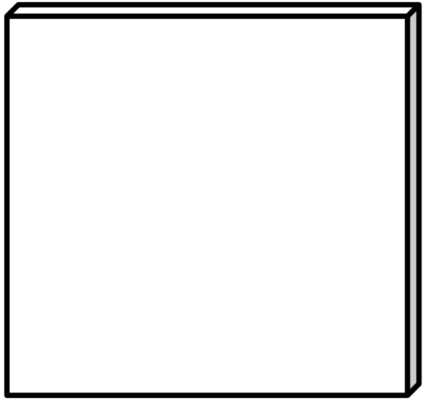
T = 16

W = 224



C_{out} different filters

How to recognize **blue** to **orange** transitions anywhere in space and time?



W = 224

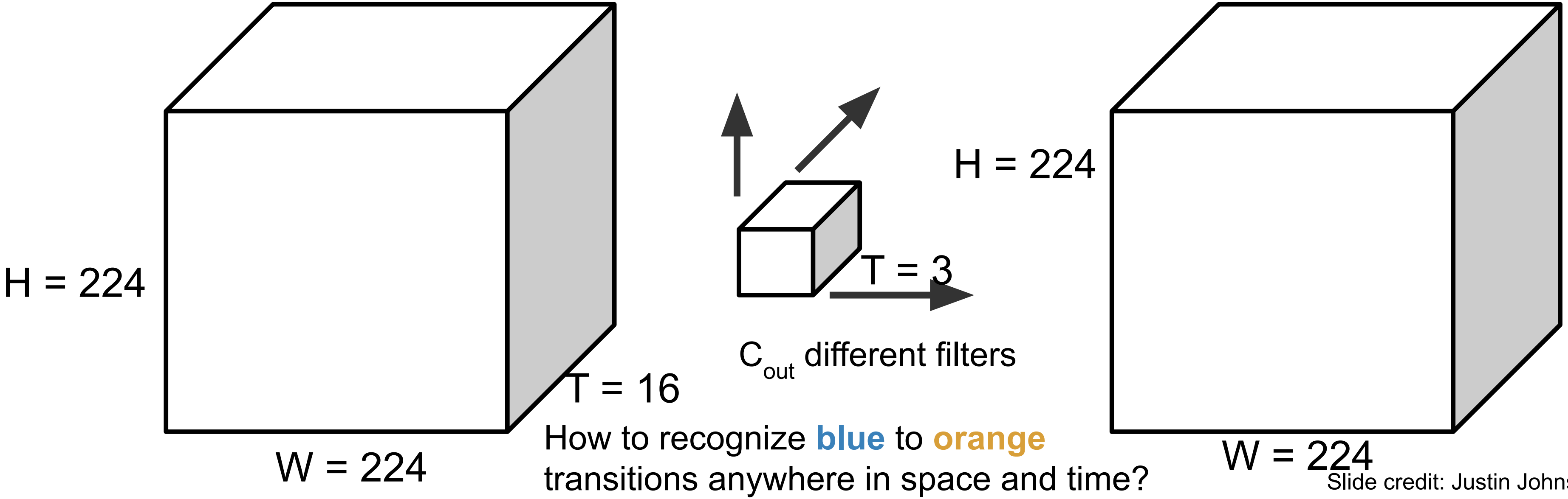
Slide credit: Justin Johnson

2D Conv (Early Fusion) vs 3D Conv (3D CNN)

Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)

Weight:
C_{out} x C_{in} x 3 x 3 x 3
Slide over x and y

Output:
C_{out} x T x H x W
3D grid with C_{out}–dim feat at each point



C_{out} different filters

How to recognize **blue** to **orange** transitions anywhere in space and time?

Slide credit: Justin Johnson

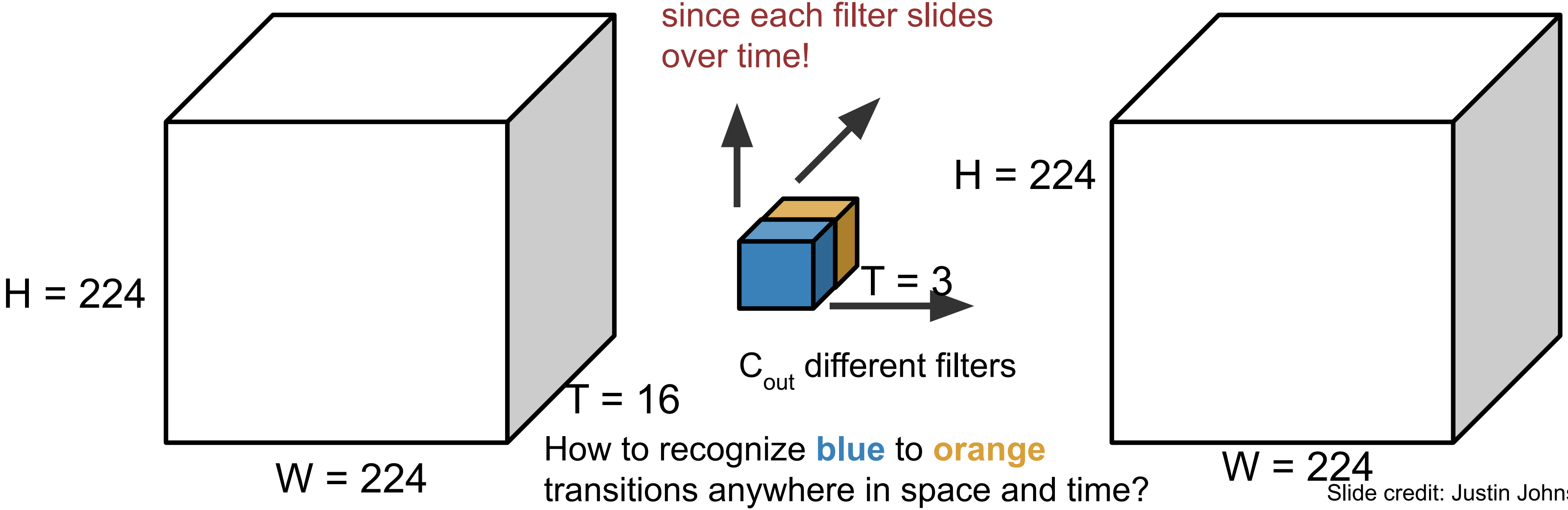
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)

Weight:
C_{out} x C_{in} x 3 x 3 x 3
Slide over x and y

Temporal shift-invariant since each filter slides over time!

Output:
C_{out} x T x H x W
3D grid with C_{out}–dim feat at each point



How to recognize **blue** to **orange** transitions anywhere in space and time?

Slide credit: Justin Johnson

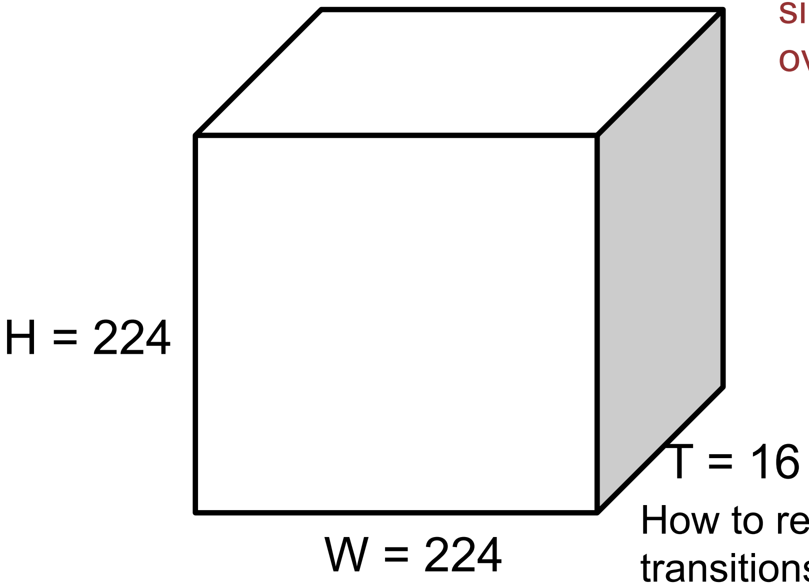
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

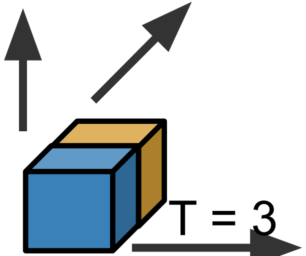
Input: C_{in} x T x H x W
(3D grid with C_{in}-dim feat at each point)

Weight:
C_{out} x C_{in} x 3 x 3 x 3
Slide over x and y

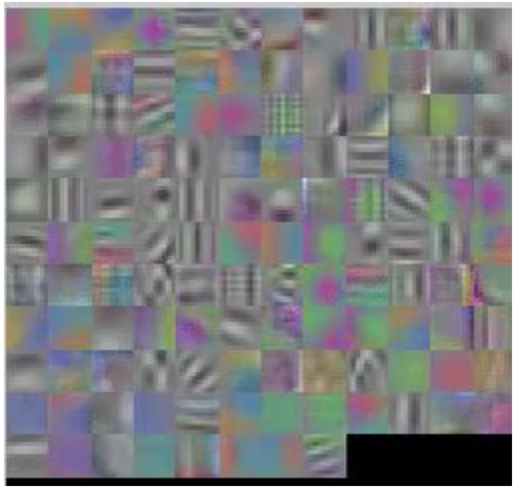
First-layer filters have shape 3 (RGB) x 4 (frames) x 5 x 5 (space)
Can visualize as video clips!

Temporal shift-invariant since each filter slides over time!





C_{out} different filters



How to recognize **blue** to **orange** transitions anywhere in space and time?

Slide credit: Justin Johnson

Example Video Dataset: **Sports-1M**

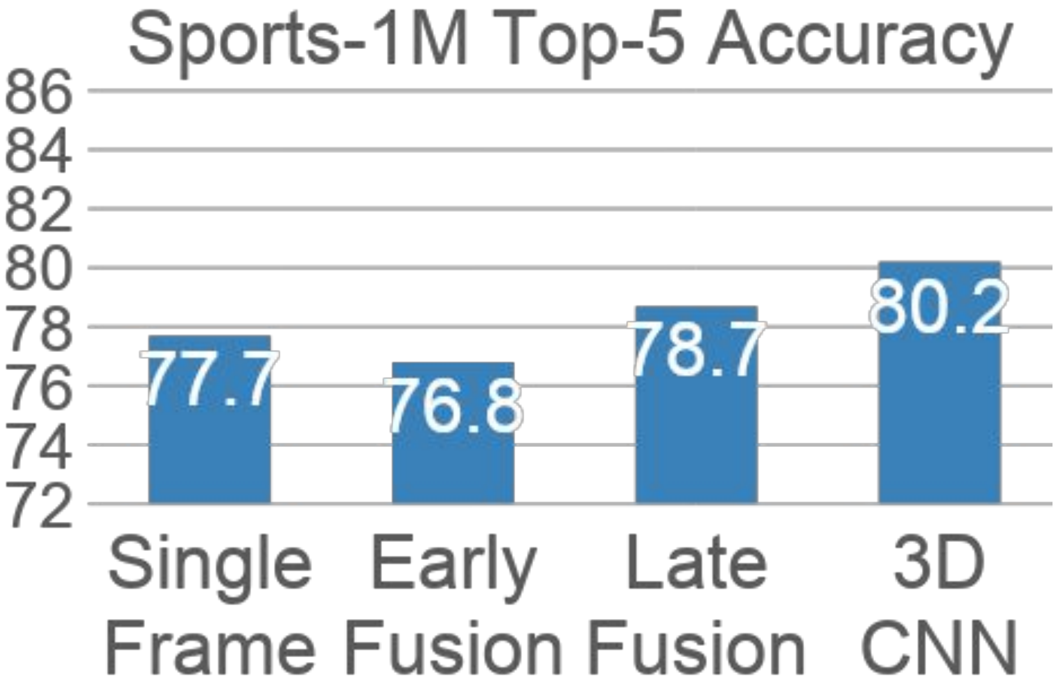
1 million YouTube videos annotated with labels for 487 different types of sports

Ground Truth
Correct prediction
Incorrect prediction

Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Slide credit: Justin Johnson

Early Fusion vs Late Fusion vs 3D CNN



Single Frame model works well – always try this first!

3D CNNs have improved a lot since 2014!

Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Slide credit: Justin Johnson

C3D: The VGG of 3D CNNs

3D CNN that uses all 3x3x3 conv and 2x2x2 pooling (except Pool1 which is 1x2x2)

Released model pretrained on Sports-1M: Many people used this as a video feature extractor

Layer	Size
Input	3 x 16 x 112 x 112
Conv1 (3x3x3)	64 x 16 x 112 x 112
Pool1 (1x2x2)	64 x 16 x 56 x 56
Conv2 (3x3x3)	128 x 16 x 56 x 56
Pool2 (2x2x2)	128 x 8 x 28 x 28
Conv3a (3x3x3)	256 x 8 x 28 x 28
Conv3b (3x3x3)	256 x 8 x 28 x 28
Pool3 (2x2x2)	256 x 4 x 14 x 14
Conv4a (3x3x3)	512 x 4 x 14 x 14
Conv4b (3x3x3)	512 x 4 x 14 x 14
Pool4 (2x2x2)	512 x 2 x 7 x 7
Conv5a (3x3x3)	512 x 2 x 7 x 7
Conv5b (3x3x3)	512 x 2 x 7 x 7
Pool5	512 x 1 x 3 x 3
FC6	4096
FC7	4096
FC8	C

Tran et al, "Learning Spatiotemporal Features with 3D Convolutional Networks", ICCV 2015

C3D: The VGG of 3D CNNs

3D CNN that uses all 3x3x3 conv and 2x2x2 pooling
(except Pool1 which is 1x2x2)

Released model pretrained on Sports-1M: Many people used this as a video feature extractor

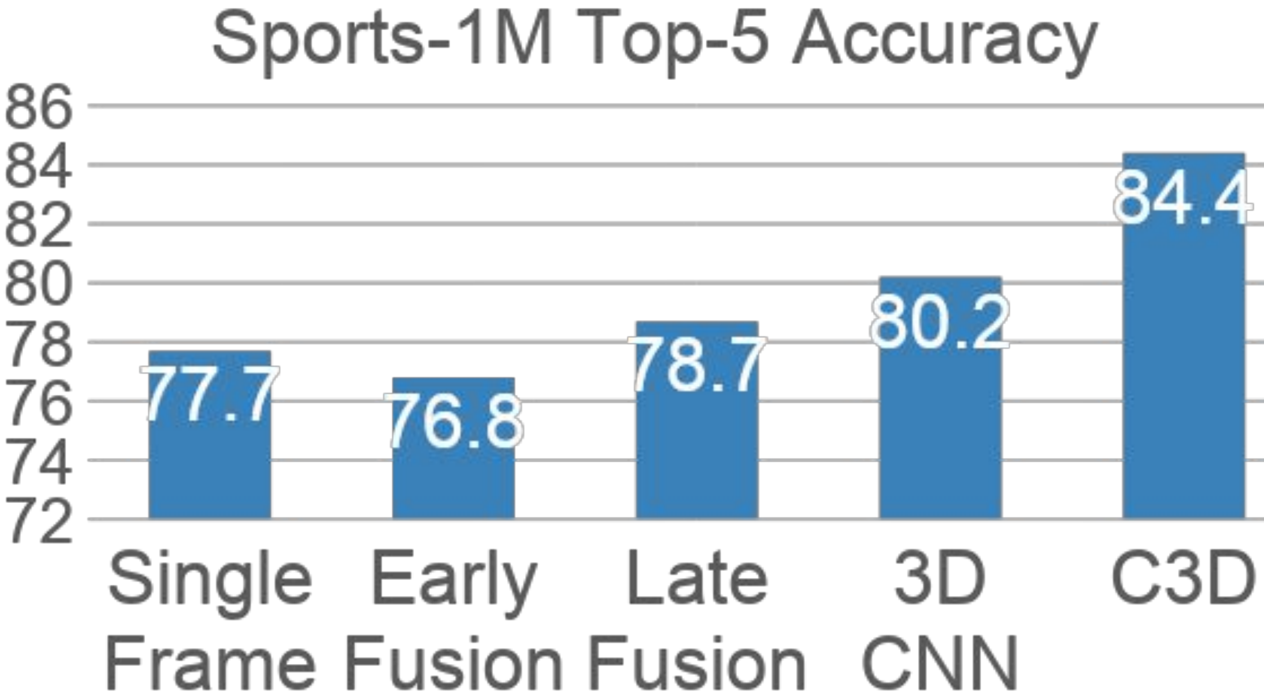
Problem: 3x3x3 conv is very expensive!
AlexNet: 0.7 GFLOP
VGG-16: 13.6 GFLOP
C3D: **39.5 GFLOP (2.9x VGG!)**

Layer	Size	MFLOPs
Input	3 x 16 x 112 x 112	
Conv1 (3x3x3)	64 x 16 x 112 x 112	1.04
Pool1 (1x2x2)	64 x 16 x 56 x 56	
Conv2 (3x3x3)	128 x 16 x 56 x 56	11.10
Pool2 (2x2x2)	128 x 8 x 28 x 28	
Conv3a (3x3x3)	256 x 8 x 28 x 28	5.55
Conv3b (3x3x3)	256 x 8 x 28 x 28	11.10
Pool3 (2x2x2)	256 x 4 x 14 x 14	
Conv4a (3x3x3)	512 x 4 x 14 x 14	2.77
Conv4b (3x3x3)	512 x 4 x 14 x 14	5.55
Pool4 (2x2x2)	512 x 2 x 7 x 7	
Conv5a (3x3x3)	512 x 2 x 7 x 7	0.69
Conv5b (3x3x3)	512 x 2 x 7 x 7	0.69
Pool5	512 x 1 x 3 x 3	
FC6	4096	0.51
FC7	4096	0.45
FC8	C	0.05

Tran et al, "Learning Spatiotemporal Features with 3D Convolutional Networks", ICCV 2015

Slide credit: Justin Johnson

Early Fusion vs Late Fusion vs 3D CNN

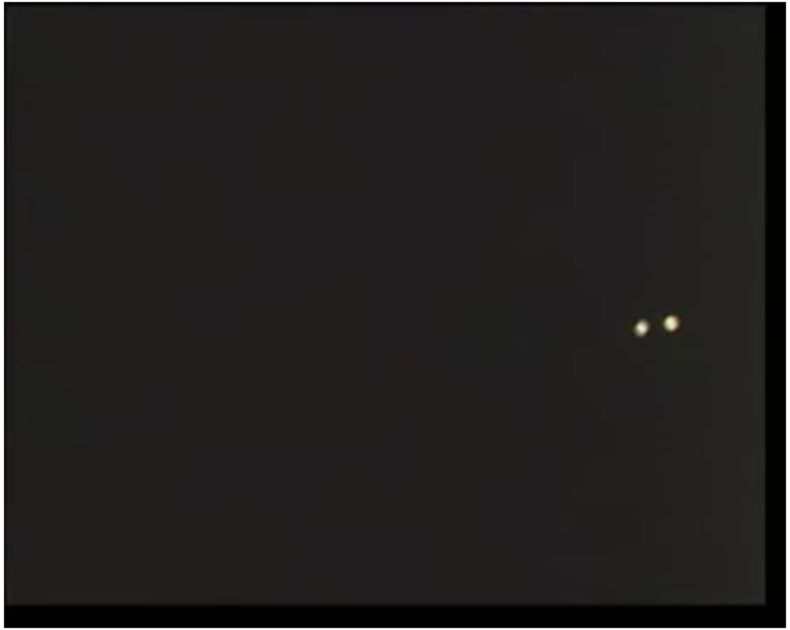


Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014
Tran et al, "Learning Spatiotemporal Features with 3D Convolutional Networks", ICCV 2015

Slide credit: Justin Johnson

Recognizing Actions from Motion

We can easily recognize actions using only **motion information**

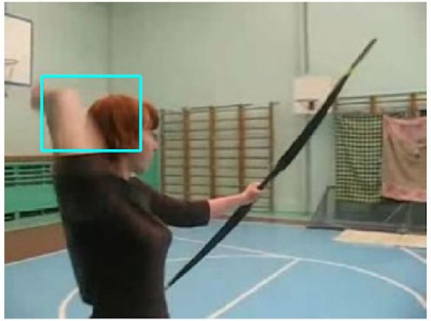


Johansson, "Visual perception of biological motion and a model for its analysis." Perception & Psychophysics. 14(2):201-211. 1973.

Slide credit: Justin Johnson

Measuring Motion: Optical Flow

Image at frame t



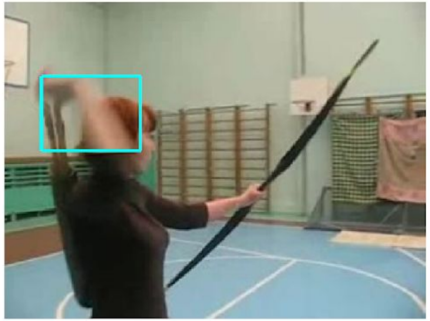


Image at frame t+1

Simonyan and Zisserman, “Two-stream convolutional networks for action recognition in videos”, NeurIPS 2014

Slide credit: Justin Johnson

Measuring Motion: Optical Flow

Image at frame t

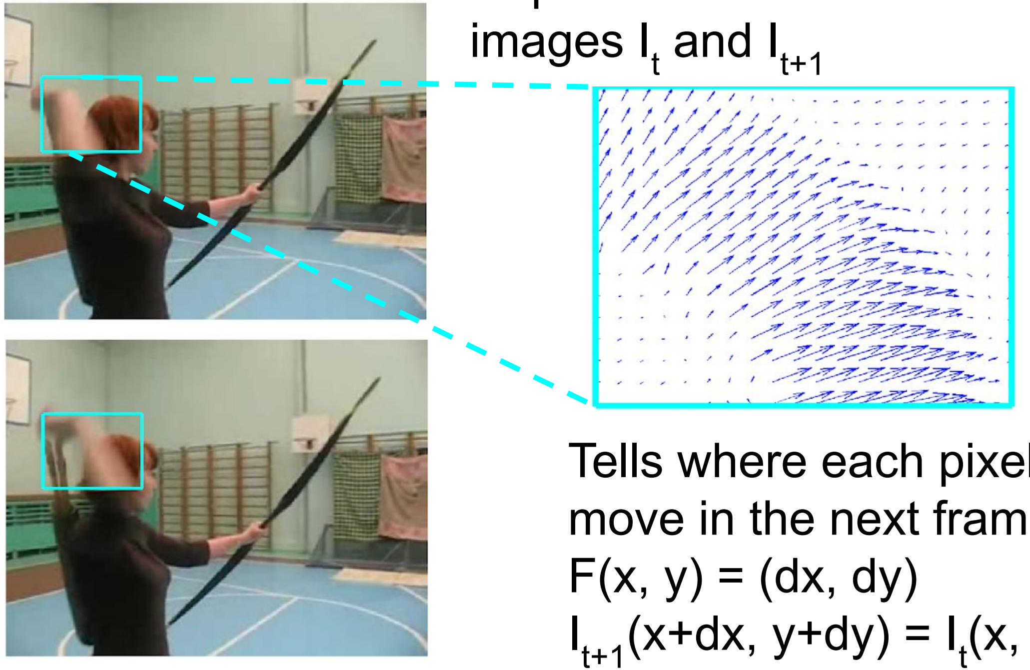


Image at frame t+1

Optical flow gives a displacement field F between images I_t and I_{t+1}

Tells where each pixel will move in the next frame:

$F(x, y) = (dx, dy)$

$I_{t+1}(x+dx, y+dy) = I_t(x, y)$

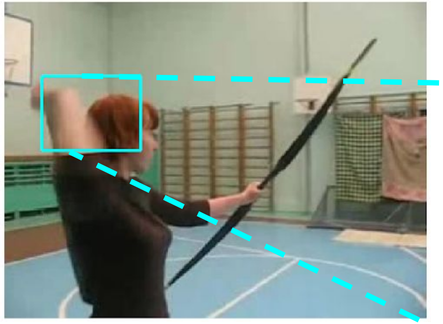
Simonyan and Zisserman, “Two-stream convolutional networks for action recognition in videos”, NeurIPS 2014

Slide credit: Justin Johnson

Measuring Motion: Optical Flow

Optical Flow highlights **local motion**

Image at frame t



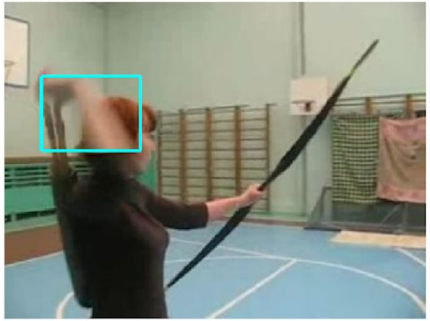
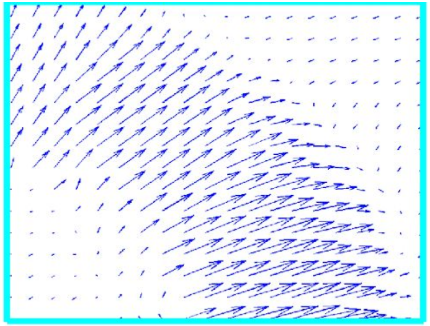


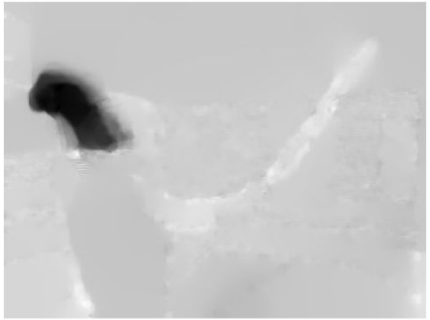
Image at frame t+1

Optical flow gives a displacement field F between images I_t and I_{t+1}



Tells where each pixel will move in the next frame:
$F(x, y) = (dx, dy)$
$I_{t+1}(x+dx, y+dy) = I_t(x, y)$

Horizontal flow dx



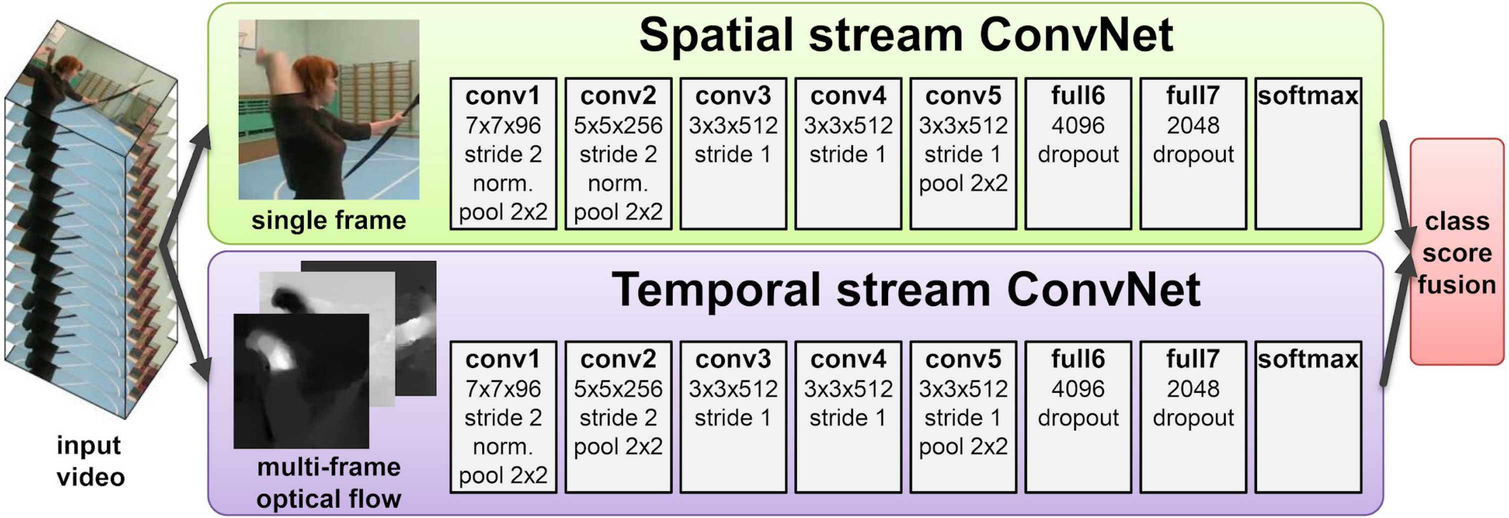
Vertical Flow dy

Slide credit: Justin Johnson

Simonyan and Zisserman, “Two-stream convolutional networks for action recognition in videos”, NeurIPS 2014

Separating Motion and Appearance: Two-Stream Networks

Input: Single Image
3 x H x W



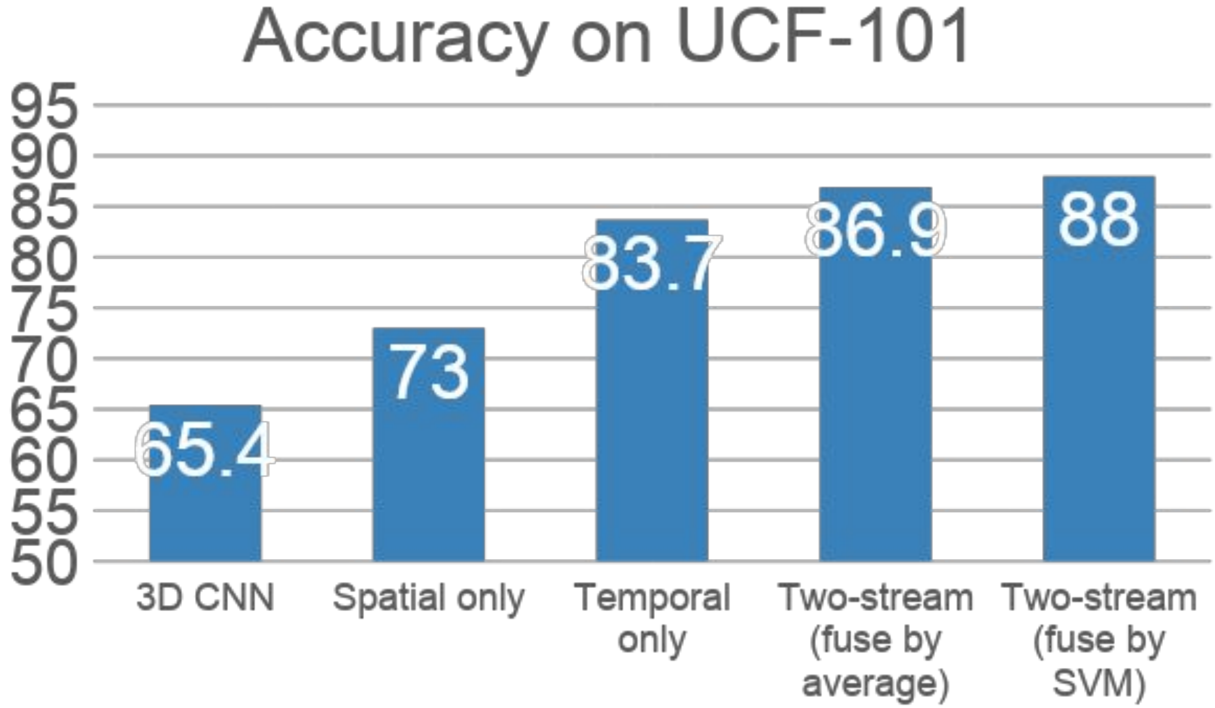
Input: Stack of optical flow:
[2*(T-1)] x H x W

Early fusion: First 2D conv
processes all flow images

Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014

Slide credit: Justin Johnson

Separating Motion and Appearance: Two-Stream Networks

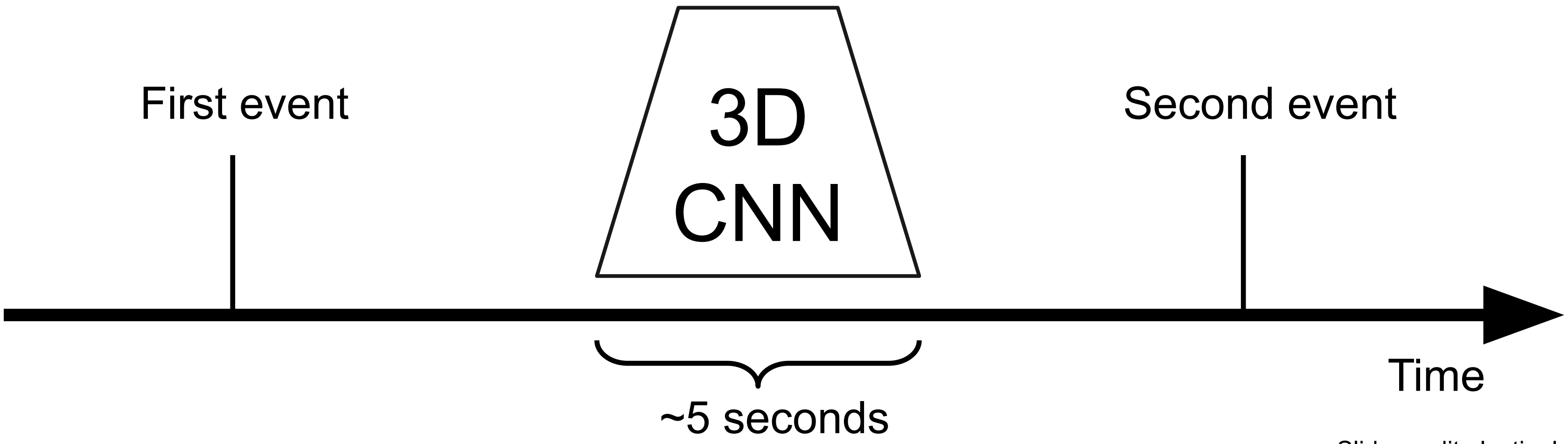


Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014

Slide credit: Justin Johnson

Modeling long-term temporal structure

So far all our temporal CNNs only model local motion between frames in very short clips of ~2-5 seconds. What about long-term structure?

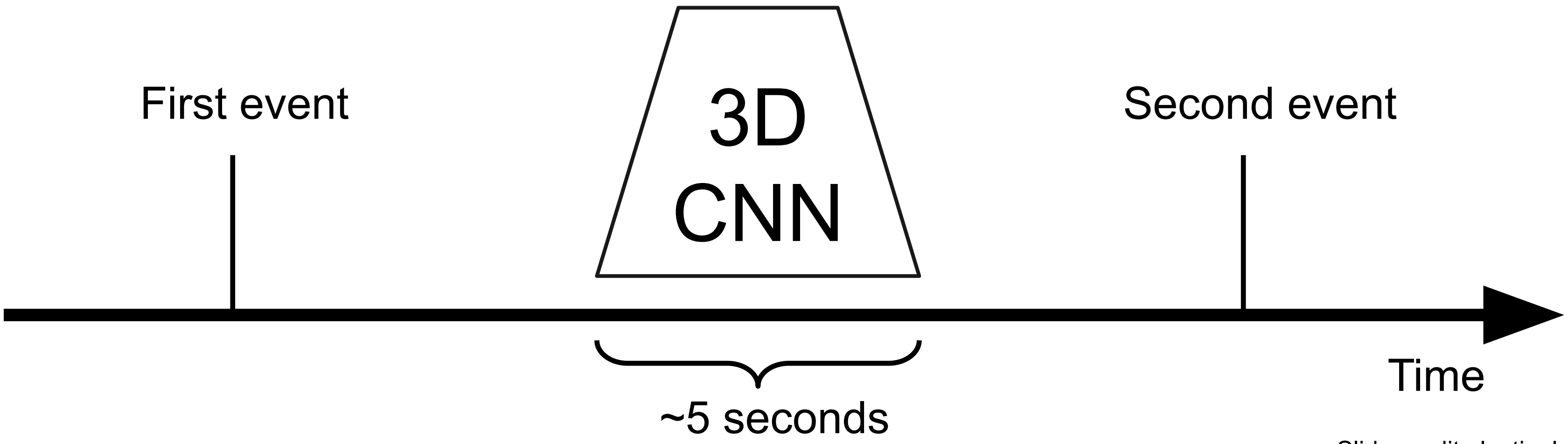


Slide credit: Justin Johnson

Modeling long-term temporal structure

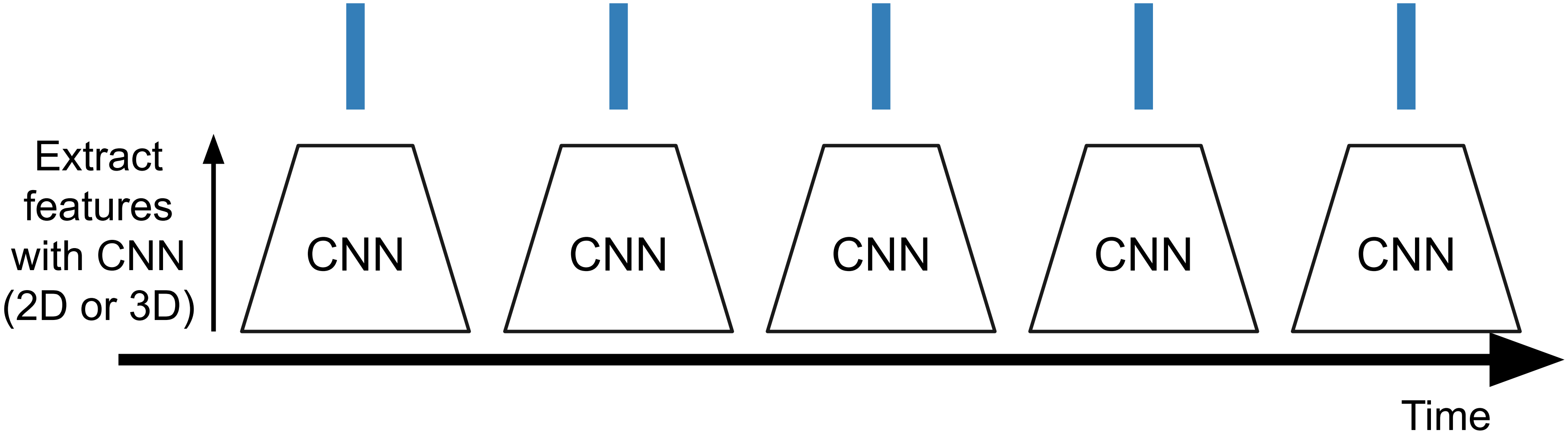
So far all our temporal CNNs only model local motion between frames in very short clips of ~2-5 seconds. What about long-term structure?

We know how to handle sequences! How about recurrent networks?



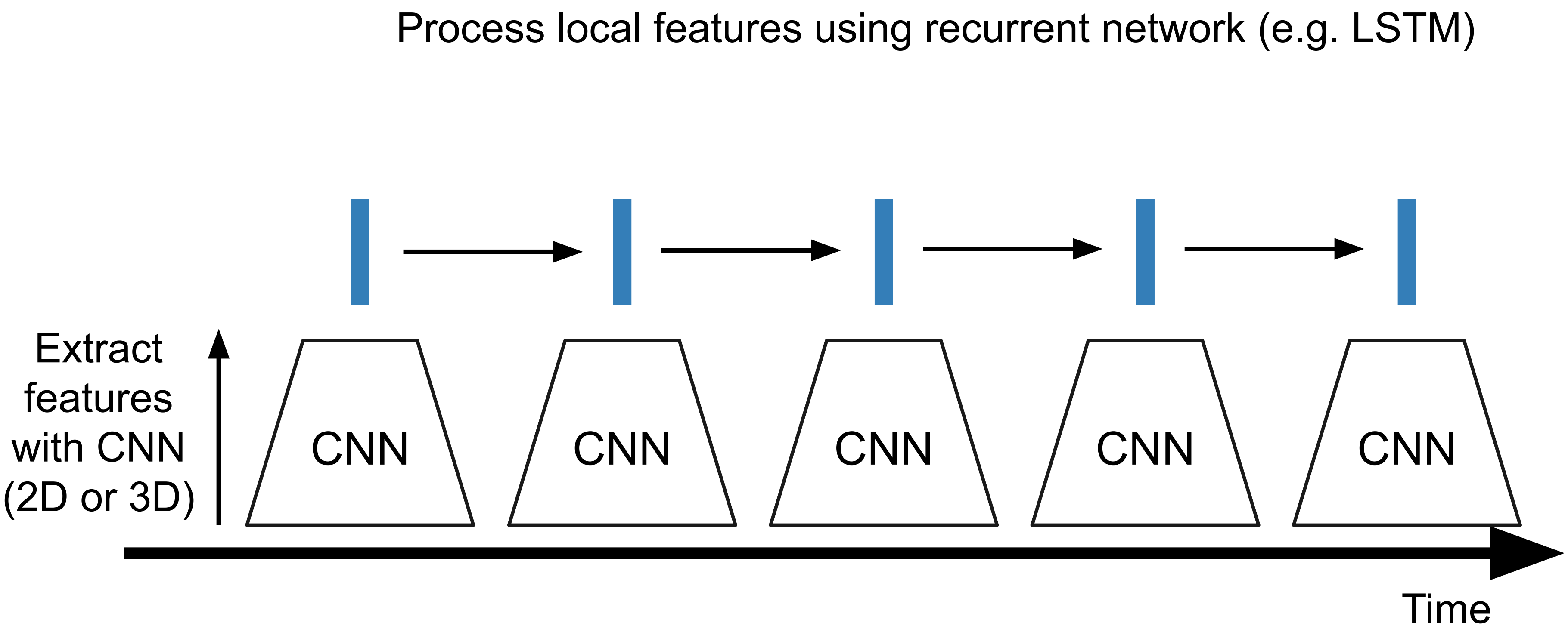
Slide credit: Justin Johnson

Modeling long-term temporal structure



Slide credit: Justin Johnson

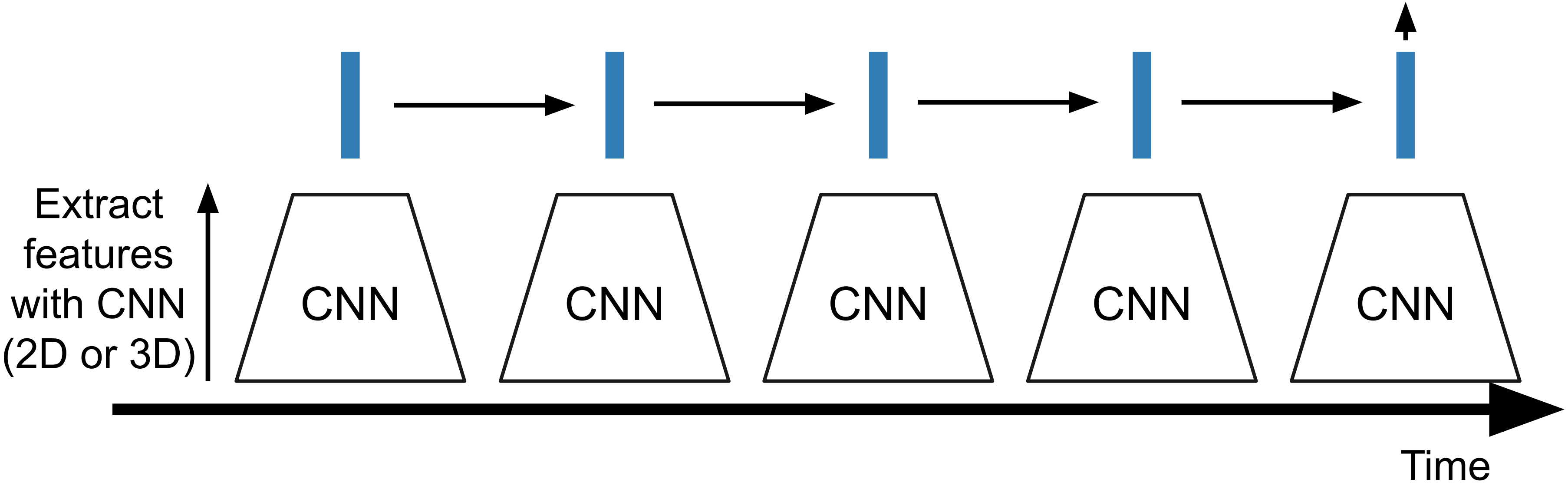
Modeling long-term temporal structure



Slide credit: Justin Johnson

Modeling long-term temporal structure

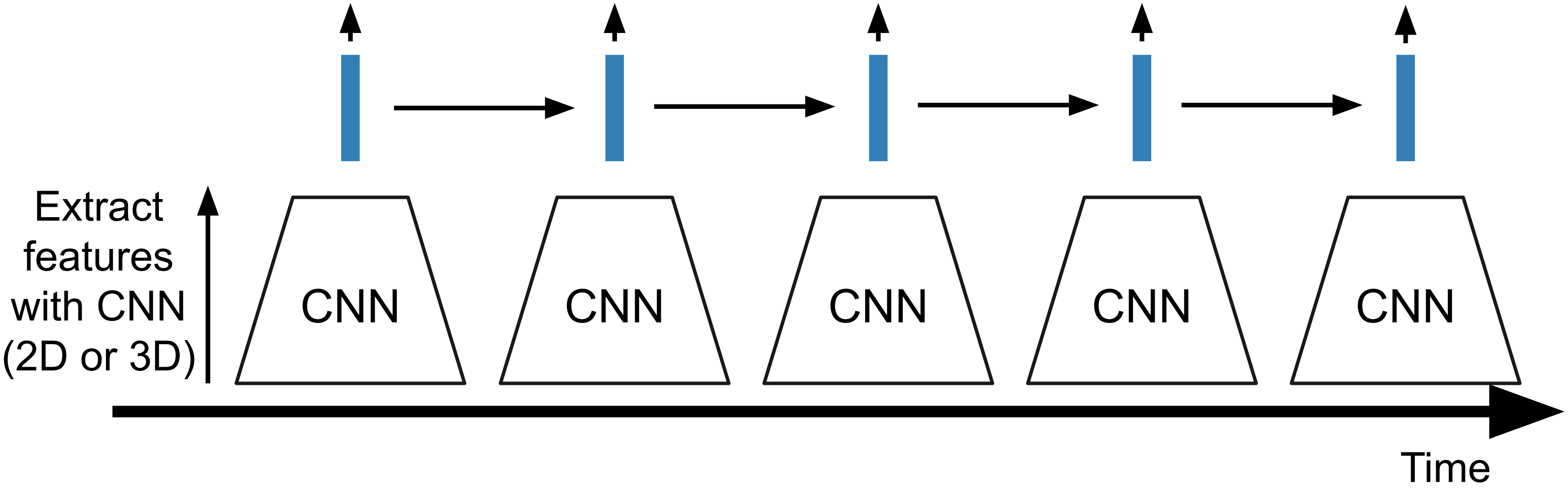
Process local features using recurrent network (e.g. LSTM)
Many to one: One output at end of video



Slide credit: Justin Johnson

Modeling long-term temporal structure

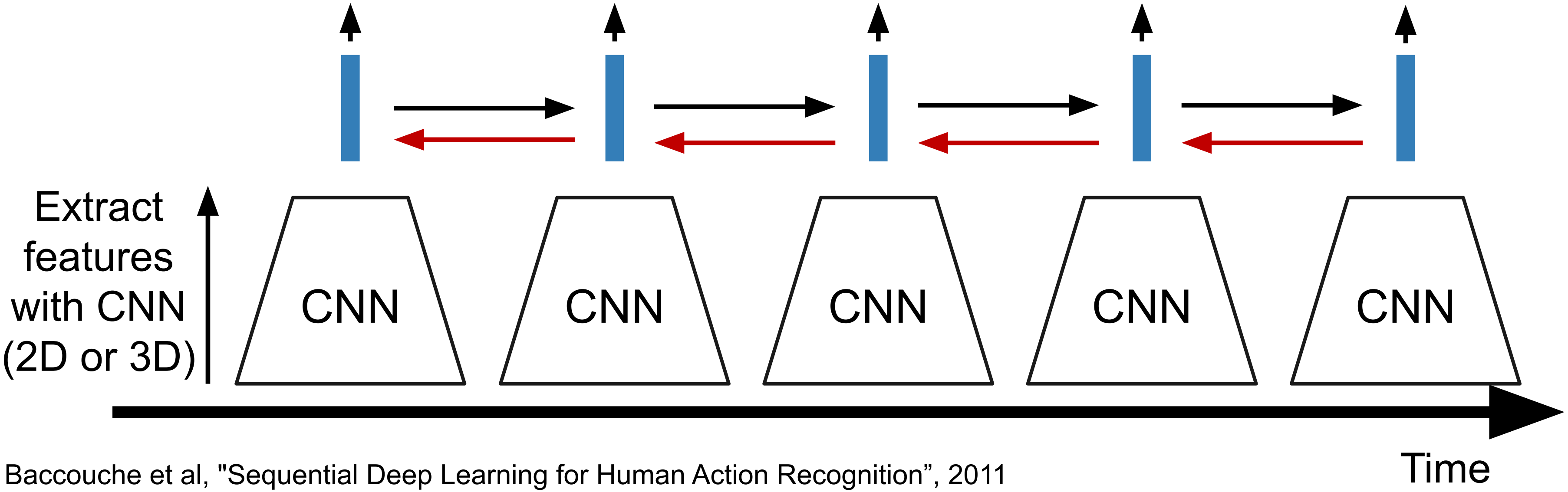
Process local features using recurrent network (e.g. LSTM)
Many to many: one output per video frame



Slide credit: Justin Johnson

Modeling long-term temporal structure

Sometimes don’t backprop to CNN to save memory; pretrain and use it as a feature extractor

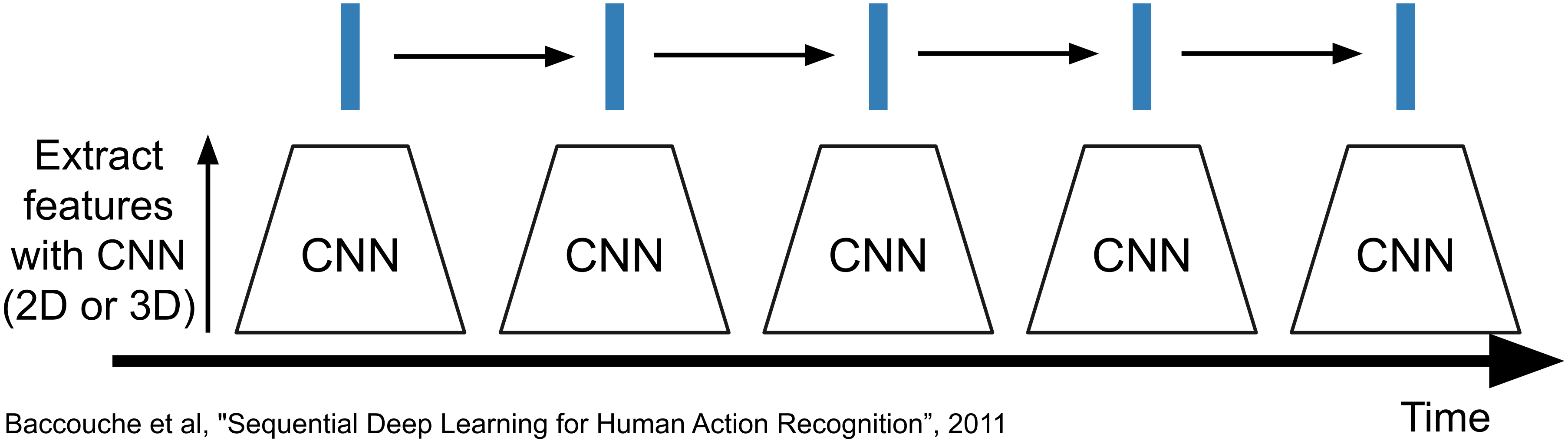


Baccouche et al, "Sequential Deep Learning for Human Action Recognition”, 2011
Donahue et al, “Long-term recurrent convolutional networks for visual recognition and description”, CVPR 2015

Slide credit: Justin Johnson

Modeling long-term temporal structure

Inside CNN: Each value is a function of a fixed temporal window (local temporal structure)
Inside RNN: Each vector is a function of all previous vectors (global temporal structure)
Can we merge both approaches?

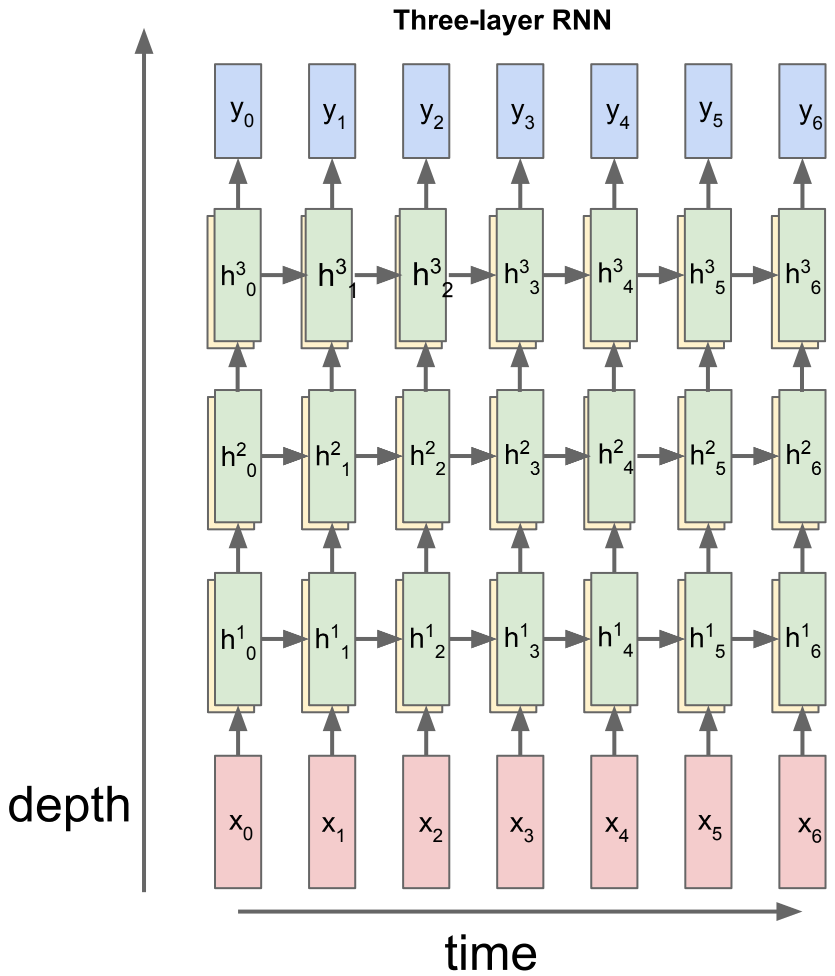


Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011
Donahue et al, "Long-term recurrent convolutional networks for visual recognition and description", CVPR 2015

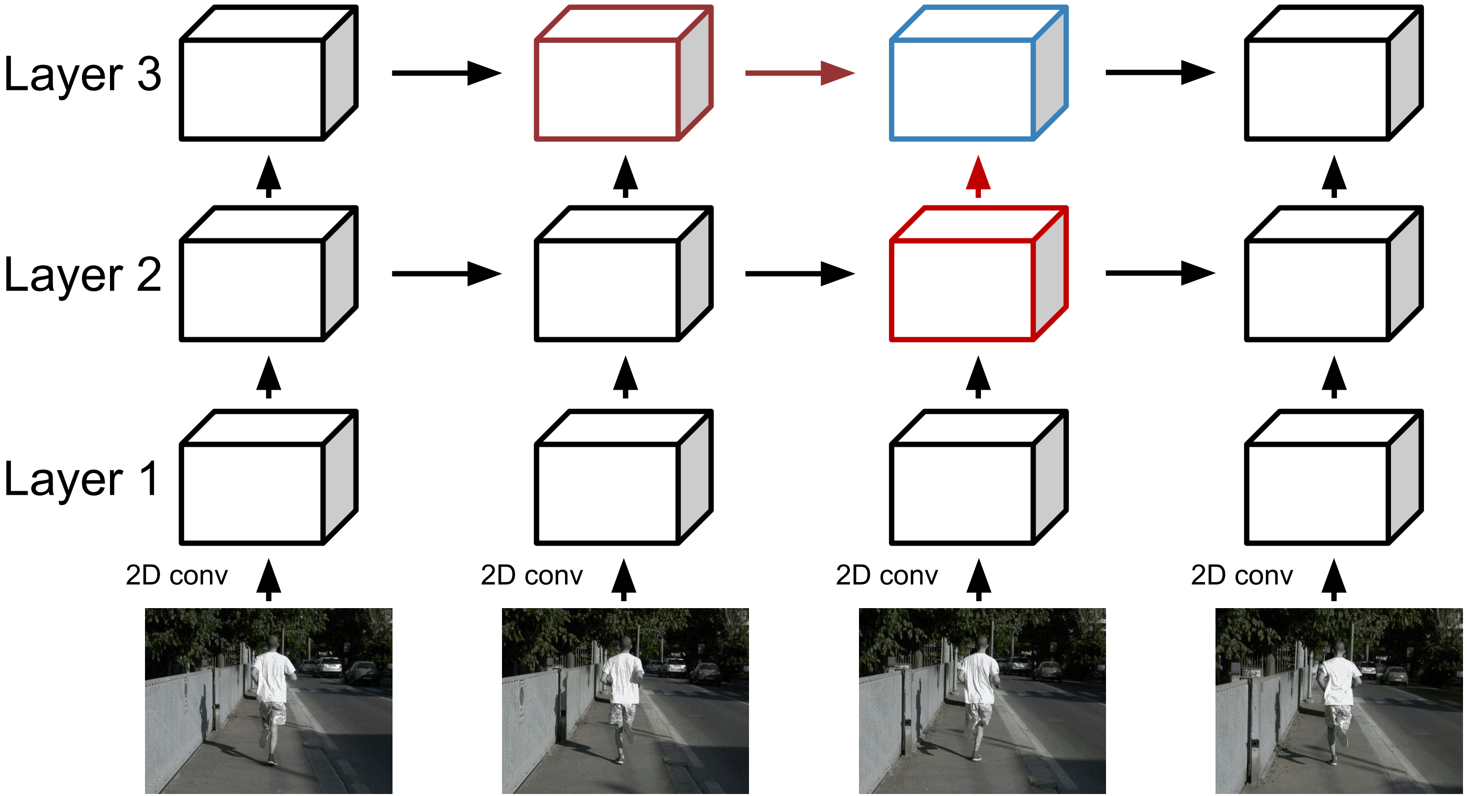
Slide credit: Justin Johnson

Recall: Multi-layer RNN

We can use a similar structure to process videos!



Recurrent Convolutional Network



Entire network uses 2D feature maps: C x H x W

Each depends on two inputs:
1. Same layer, previous timestep
2. Prev layer, same timestep

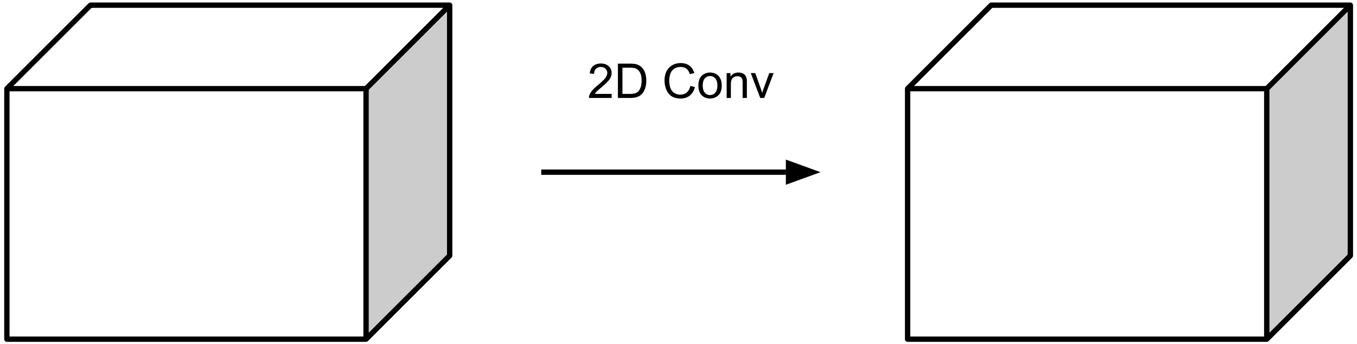
Use different weights at each layer, share weights across time

Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

Slide credit: Justin Johnson

Recurrent Convolutional Network

Normal 2D CNN:



Input features:
C x H x W

Output features:
C x H x W

Slide credit: Justin Johnson

Recurrent Convolutional Network

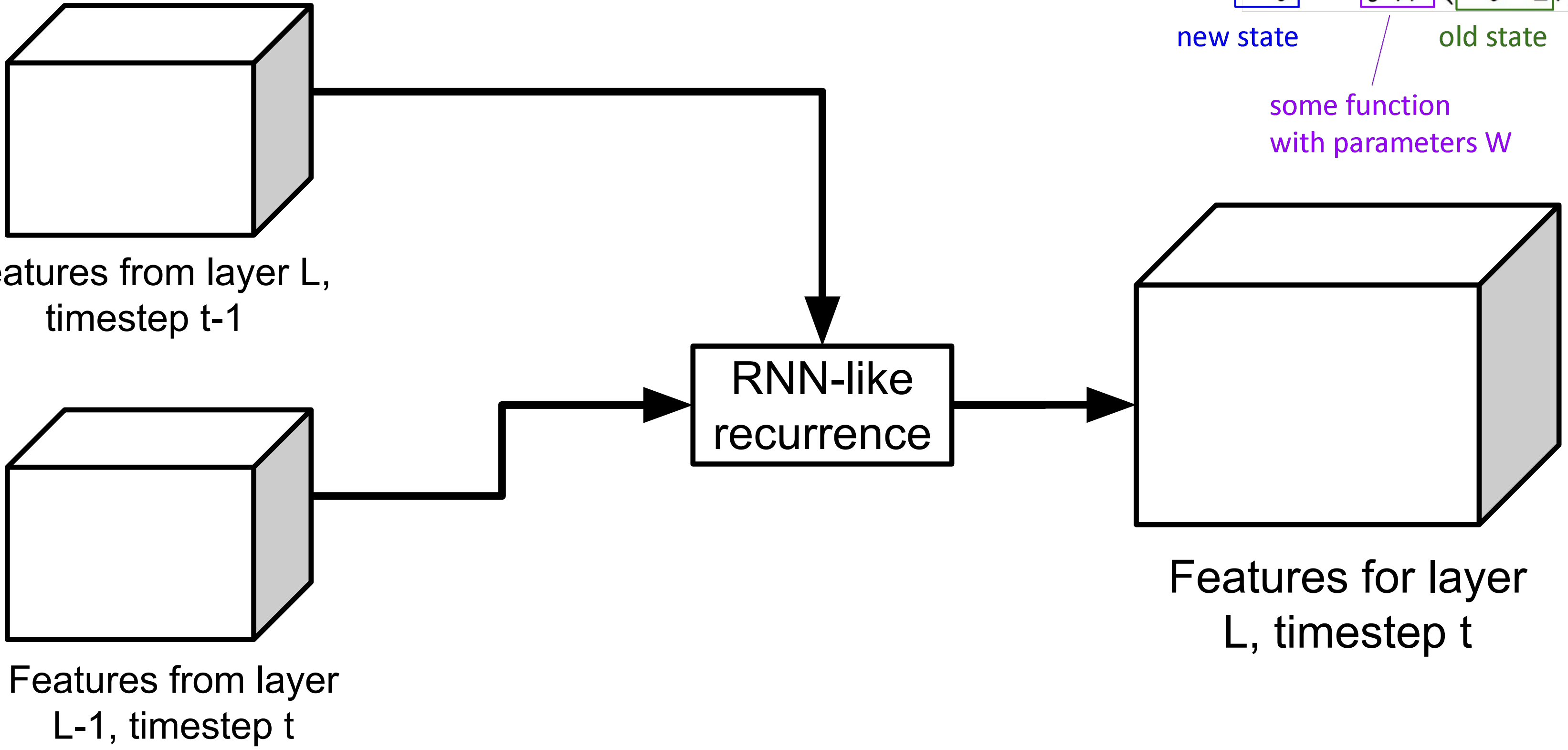
Recall: Recurrent Network

$$\boxed{h_t} = \boxed{f_W}(\boxed{h_{t-1}}, \boxed{x_t})$$

new state

old state

some function
with parameters W



Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

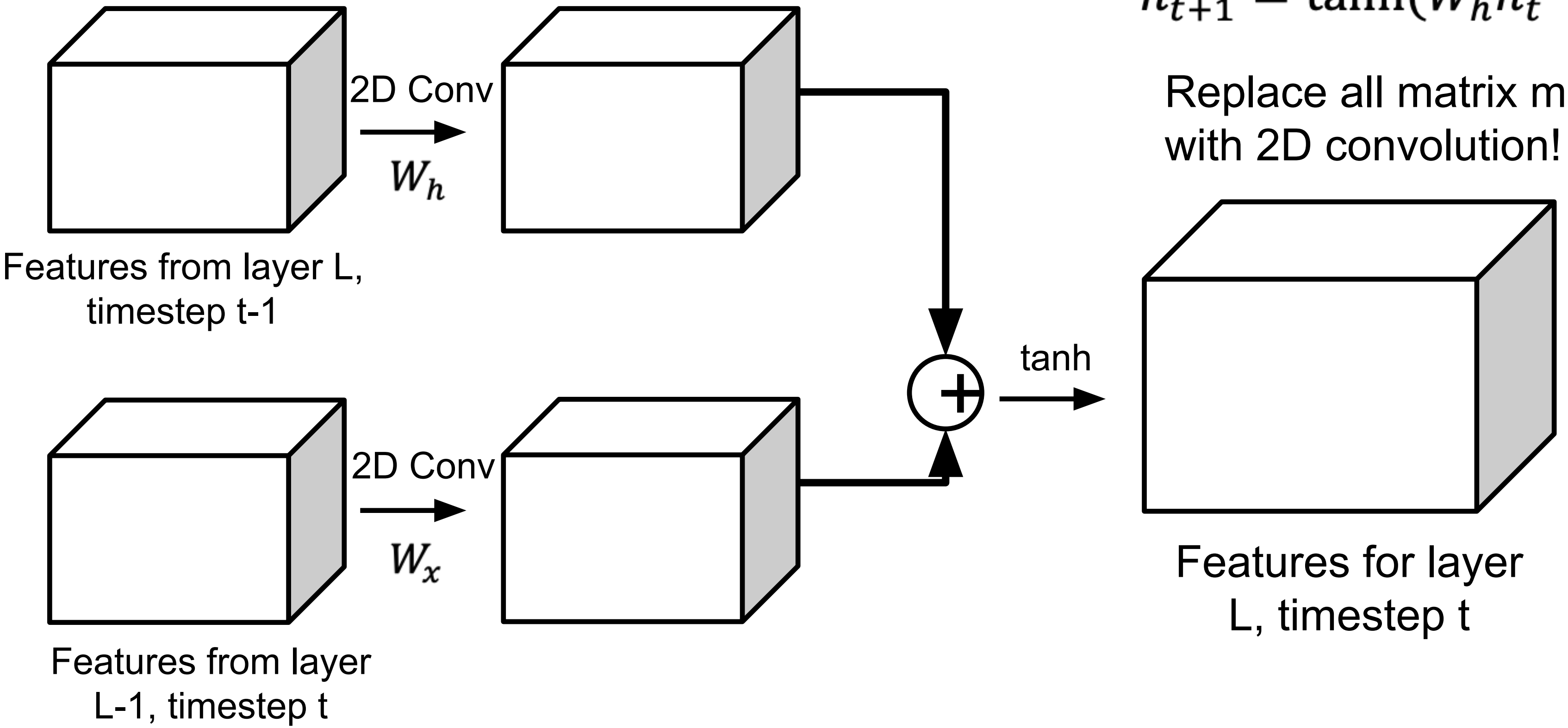
Slide credit: Justin Johnson

Recurrent Convolutional Network

Recall: Vanilla RNN

$$h_{t+1} = \tanh(W_h h_t + W_x x)$$

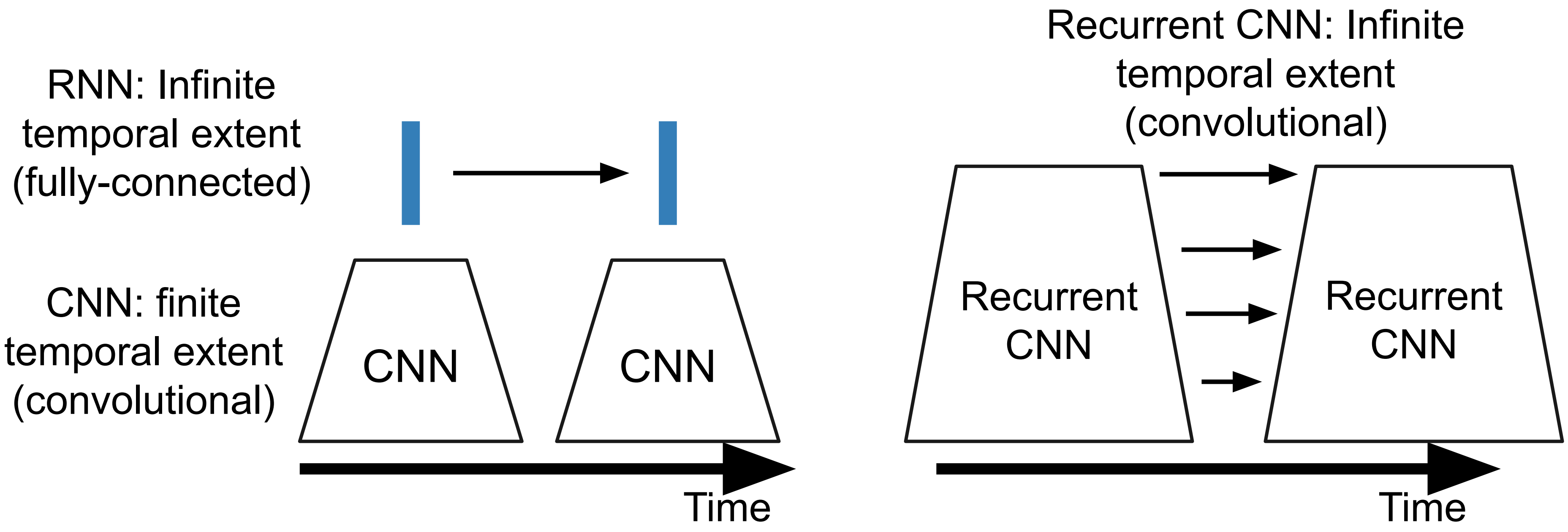
Replace all matrix multiply with 2D convolution!



Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

Slide credit: Justin Johnson

Modeling long-term temporal structure



Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011
Donahue et al, "Long-term recurrent convolutional networks for visual recognition and description", CVPR 2015

Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

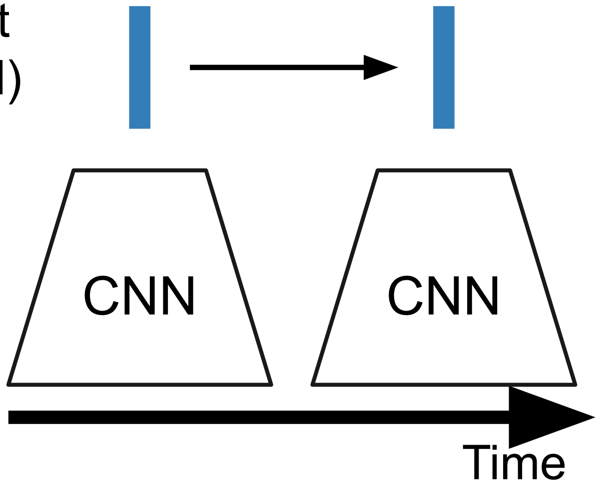
Slide credit: Justin Johnson

Modeling long-term temporal structure

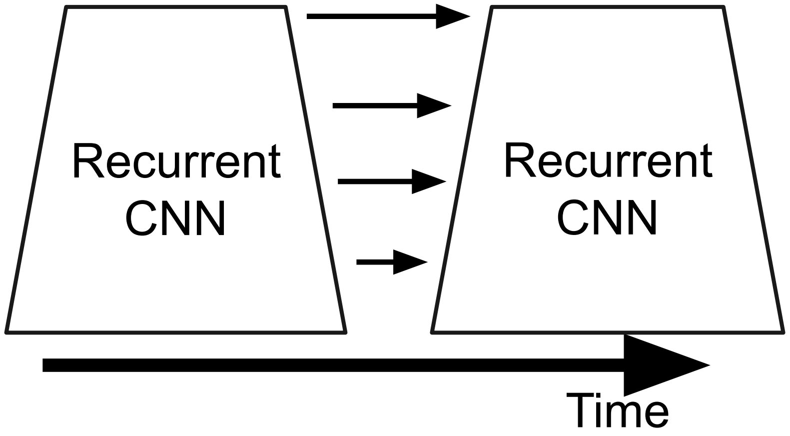
Problem: RNNs are slow for long sequences (can't be parallelized)

RNN: Infinite temporal extent (fully-connected)

CNN: finite temporal extent (convolutional)



Recurrent CNN: Infinite temporal extent (convolutional)

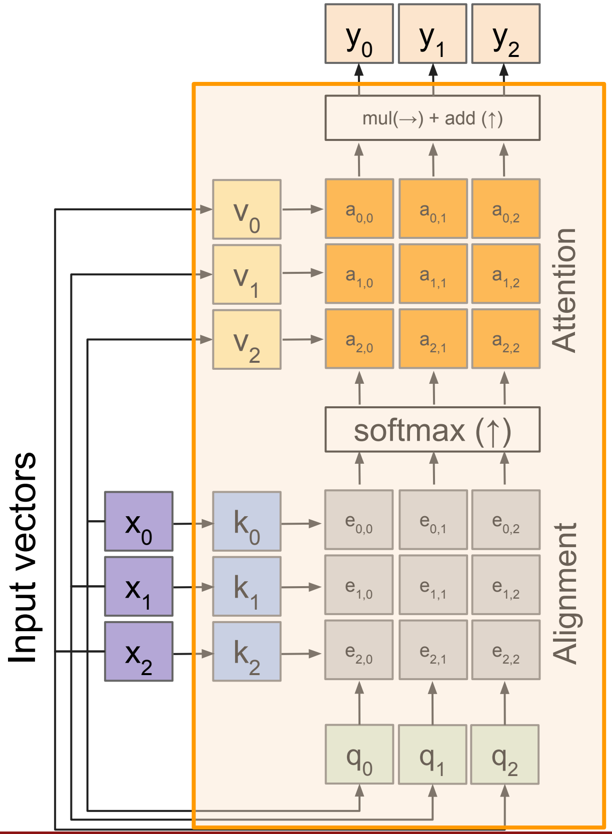


Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011
Donahue et al, "Long-term recurrent convolutional networks for visual recognition and description", CVPR 2015

Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

Slide credit: Justin Johnson

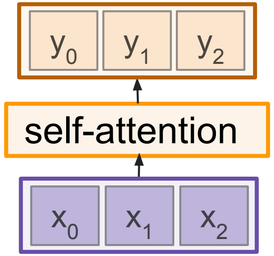
Recall: Self-Attention



Outputs:
context vectors: **y** (shape: D_v)

Operations:
Key vectors: $\mathbf{k} = \mathbf{x}\mathbf{W}_k$
Value vectors: $\mathbf{v} = \mathbf{x}\mathbf{W}_v$
Query vectors: $\mathbf{q} = \mathbf{x}\mathbf{W}_q$
Alignment: $e_{i,j} = q_j \cdot k_i / \sqrt{D}$
Attention: $\mathbf{a} = \text{softmax}(\mathbf{e})$
Output: $y_j = \sum_i a_{i,j} v_i$

Inputs:
Input vectors: **x** (shape: N x D)

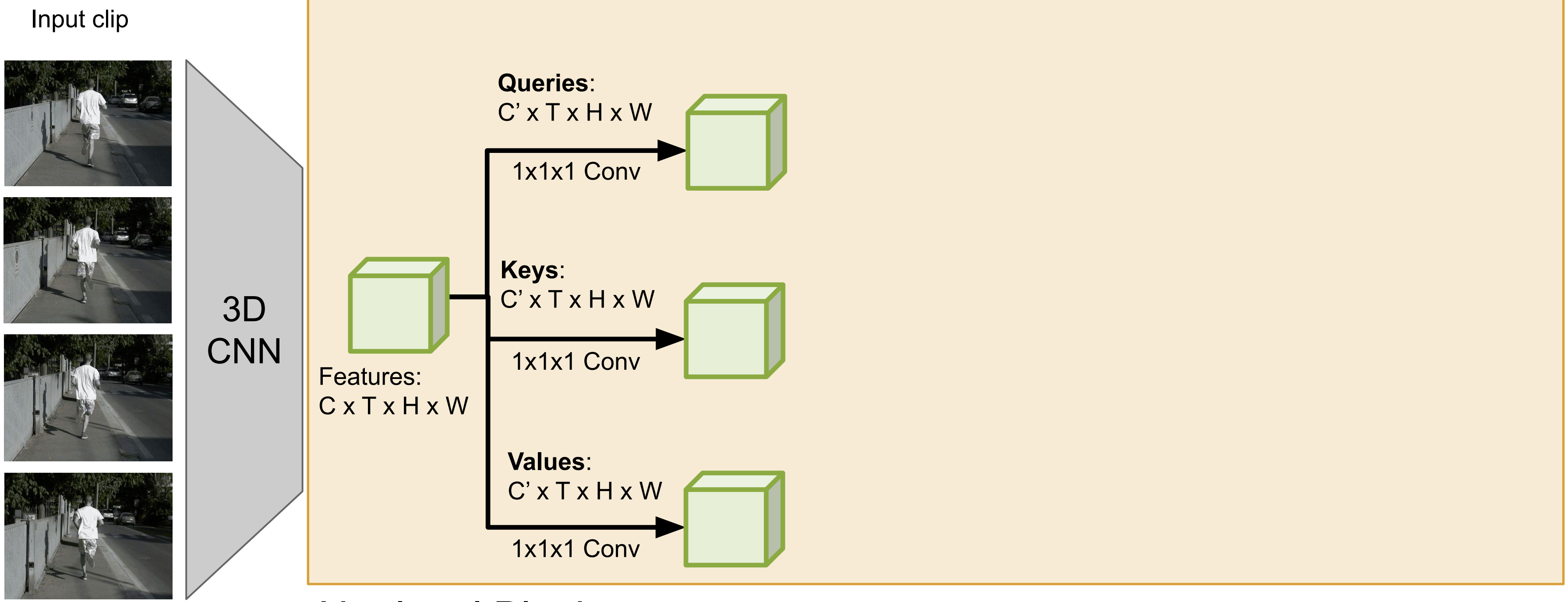


Spatio-Temporal Self-Attention (Nonlocal Block)

Wang et al, "Non-local neural networks", CVPR 2018

Slide credit: Justin Johnson

Spatio-Temporal Self-Attention (Nonlocal Block)

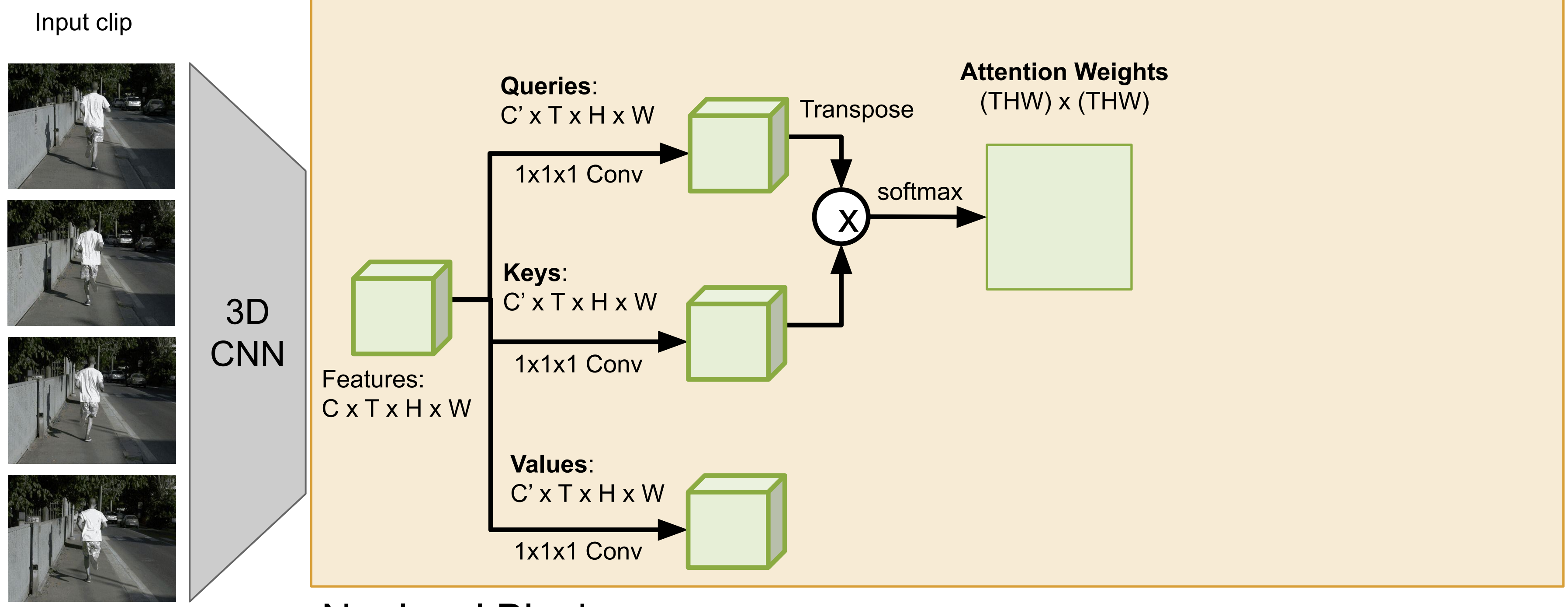


Nonlocal Block

Wang et al, “Non-local neural networks”, CVPR 2018

Slide credit: Justin Johnson

Spatio-Temporal Self-Attention (Nonlocal Block)

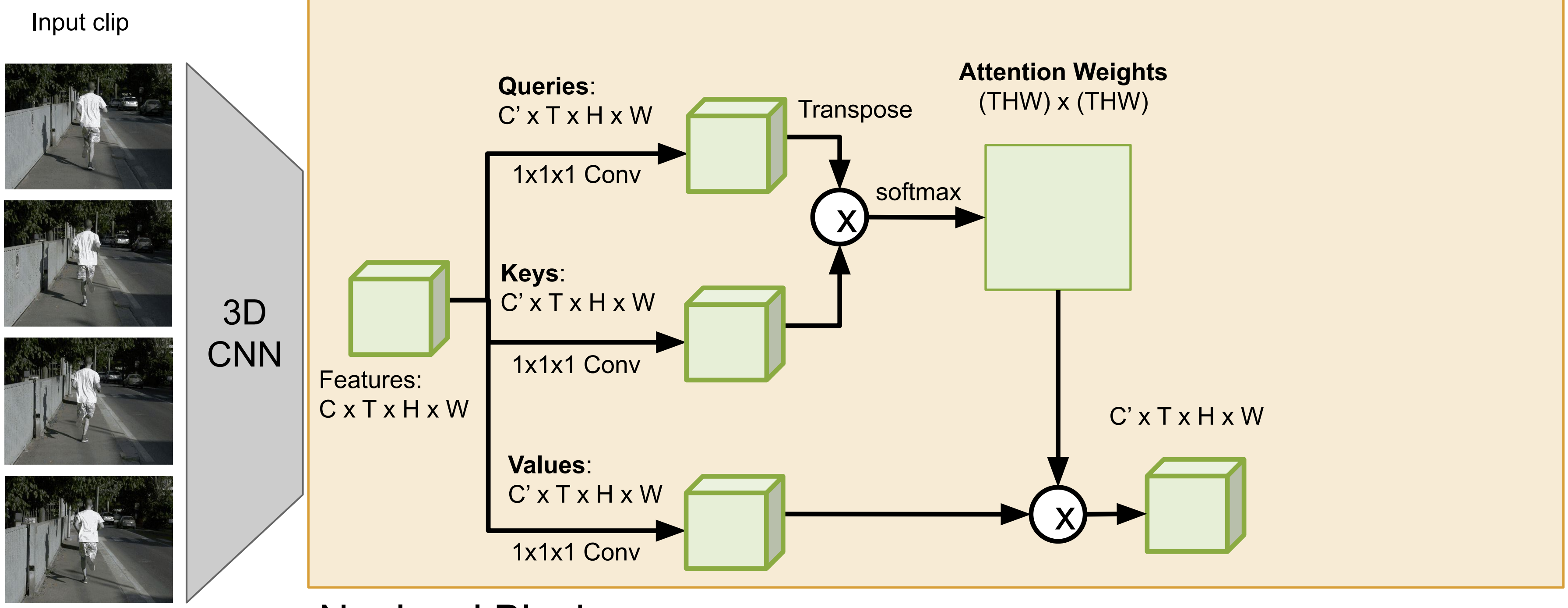


Nonlocal Block

Wang et al, "Non-local neural networks", CVPR 2018

Slide credit: Justin Johnson

Spatio-Temporal Self-Attention (Nonlocal Block)

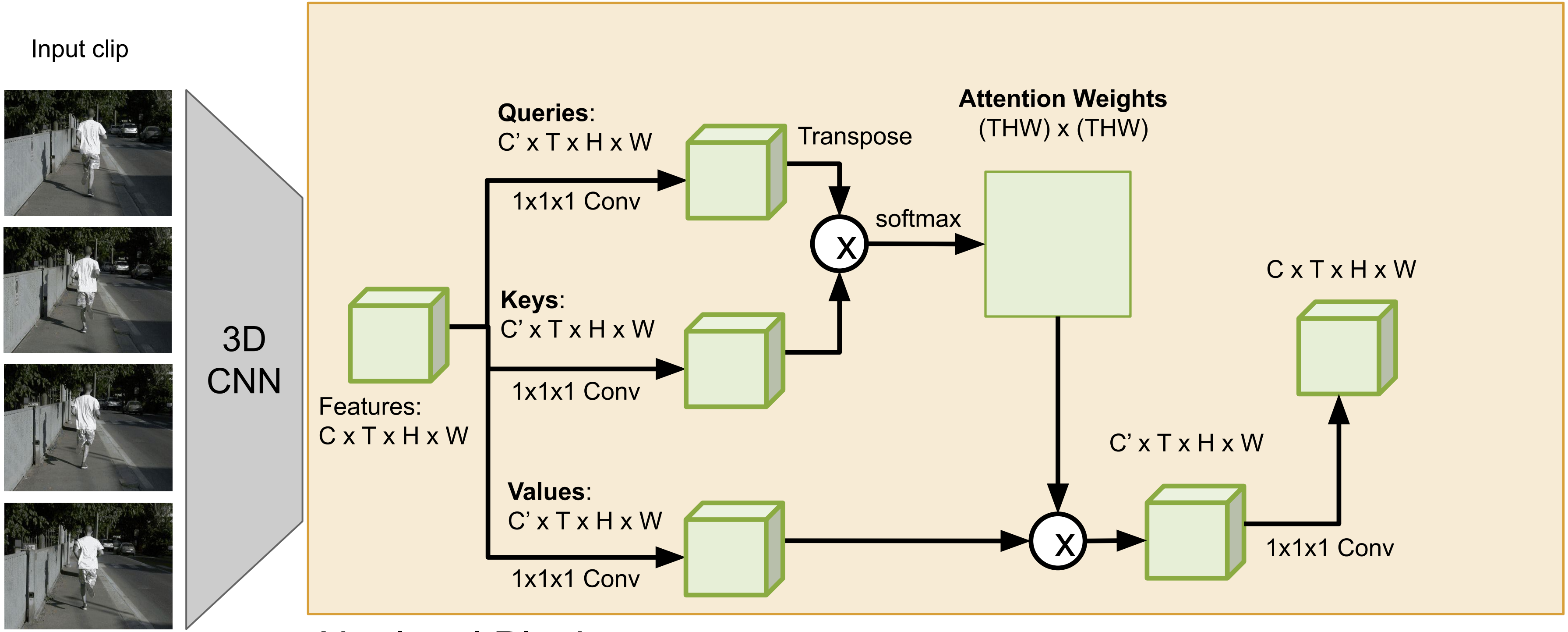


Nonlocal Block

Wang et al, "Non-local neural networks", CVPR 2018

Slide credit: Justin Johnson

Spatio-Temporal Self-Attention (Nonlocal Block)

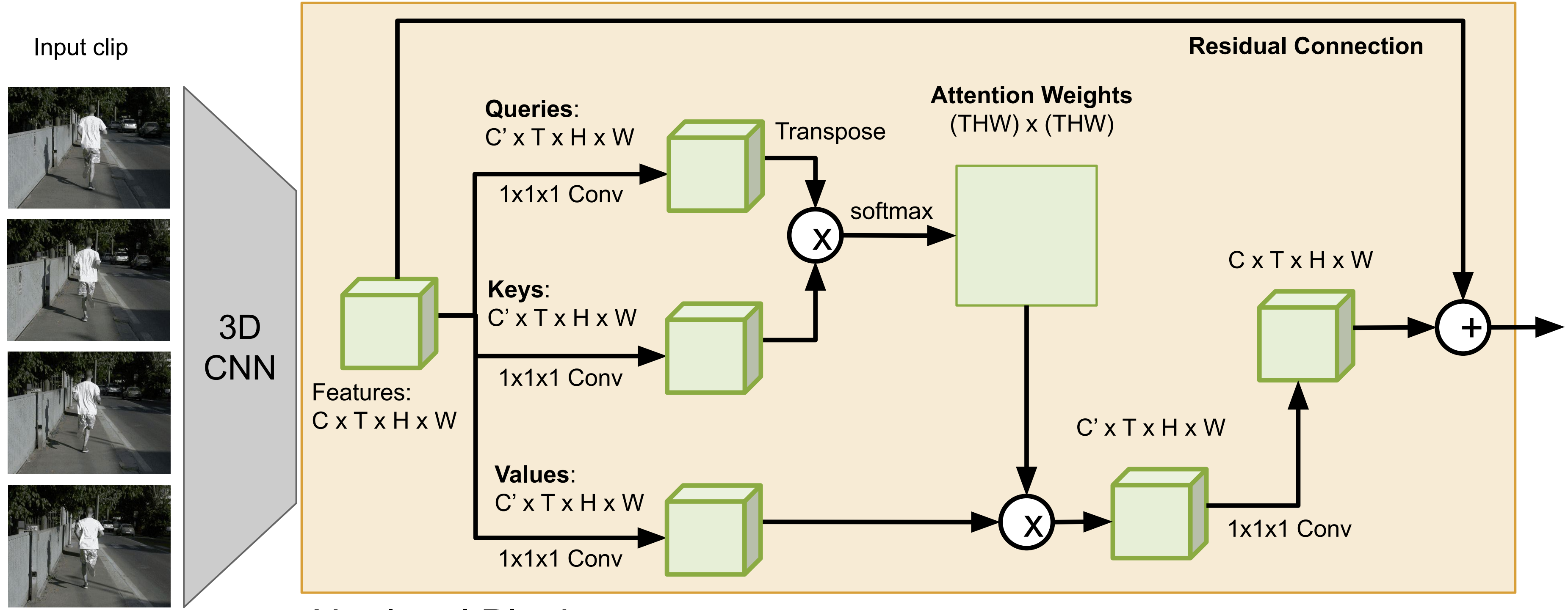


Nonlocal Block

Wang et al, "Non-local neural networks", CVPR 2018

Slide credit: Justin Johnson

Spatio-Temporal Self-Attention (Nonlocal Block)



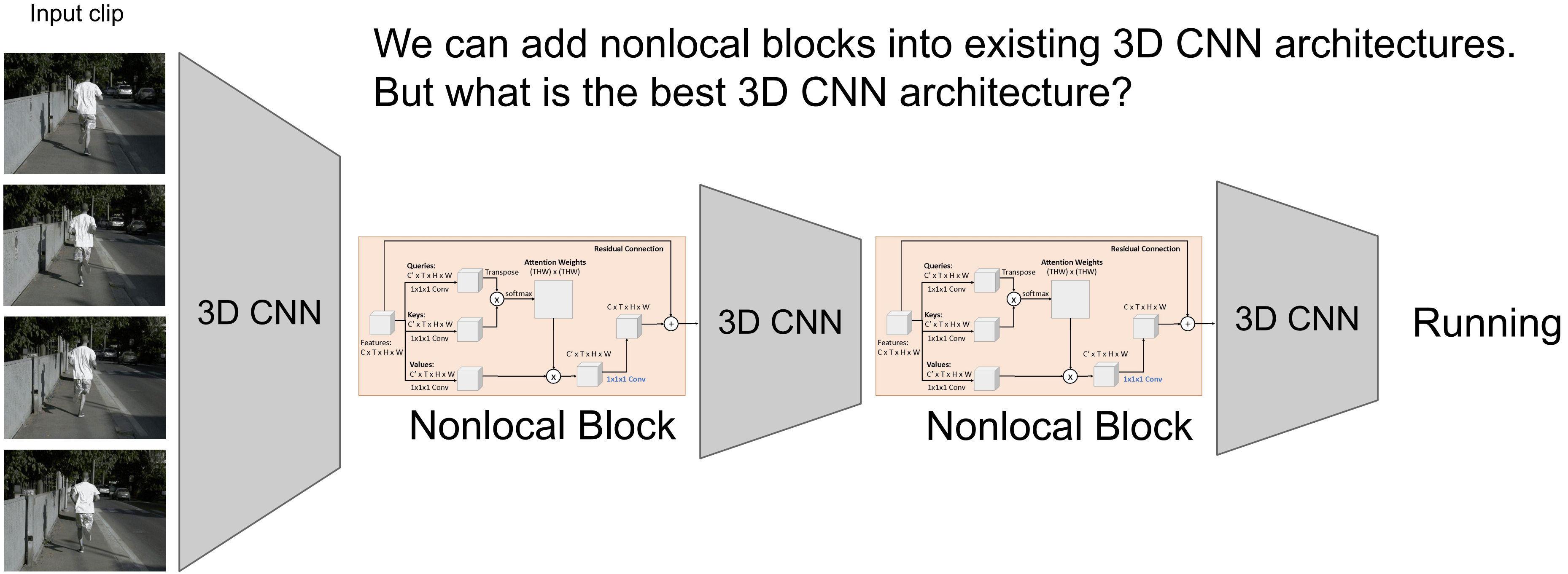
Nonlocal Block

Wang et al, "Non-local neural networks", CVPR 2018

Slide credit: Justin Johnson

Spatio-Temporal Self-Attention (Nonlocal Block)

We can add nonlocal blocks into existing 3D CNN architectures.
But what is the best 3D CNN architecture?



Wang et al, "Non-local neural networks", CVPR 2018

Slide credit: Justin Johnson

Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images. Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D K_h x K_w conv/pool layer with a 3D K_t x K_h x K_w version

Carreira and Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset", CVPR 2017

Slide credit: Justin Johnson

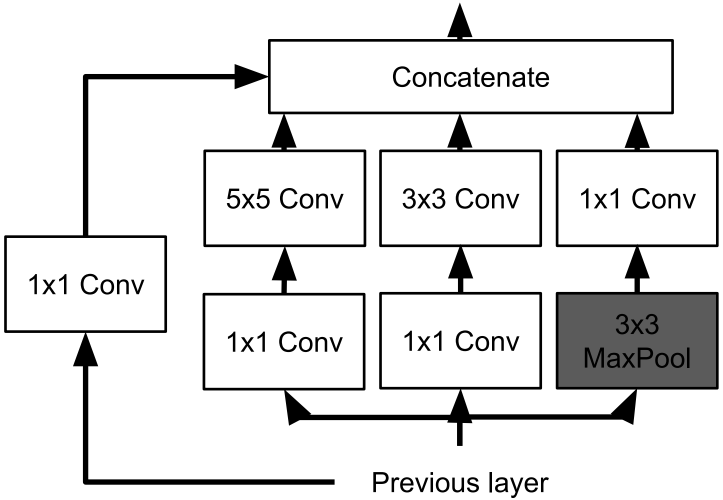
Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images. Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D K_h x K_w conv/pool layer with a 3D K_t x K_h x K_w version

Inception Block: Original



Carreira and Zisserman, “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset”, CVPR 2017

Slide credit: Justin Johnson

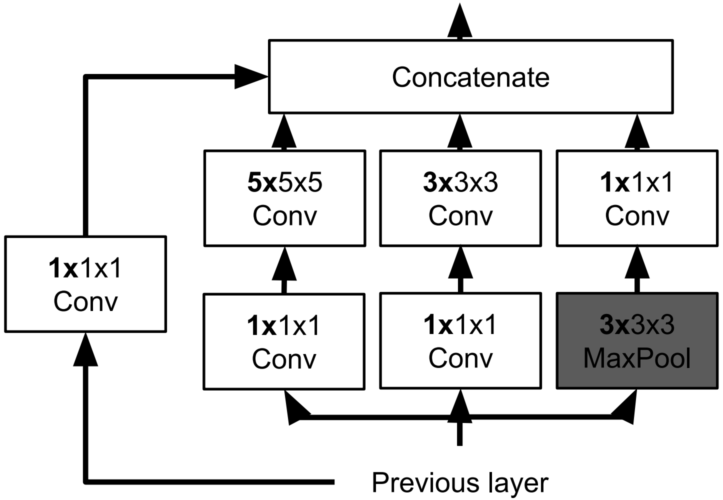
Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images. Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D K_h x K_w conv/pool layer with a 3D K_t x K_h x K_w version

Inception Block: Inflated



Carreira and Zisserman, “Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset”, CVPR 2017

Slide credit: Justin Johnson

Inflating 2D Networks to 3D (I3D)

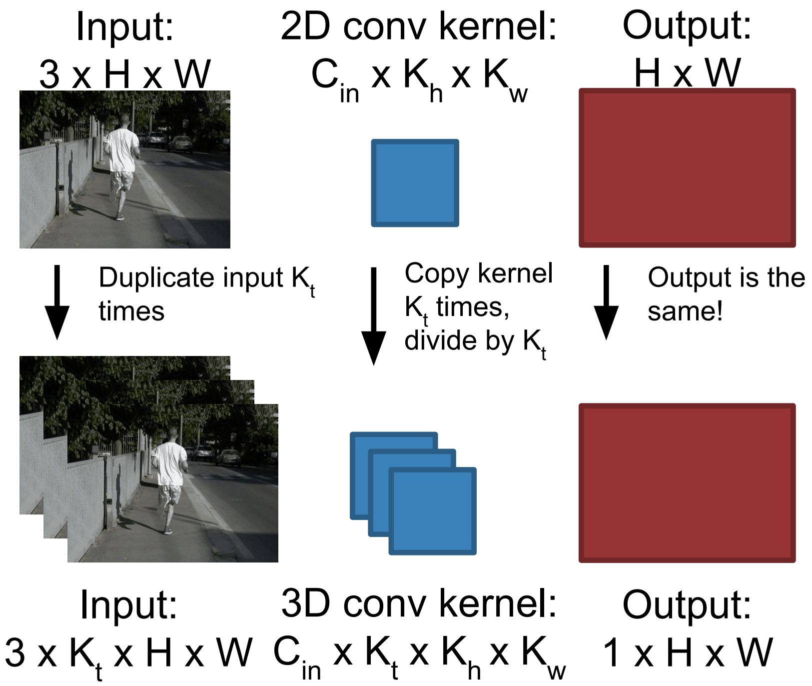
There has been a lot of work on architectures for images. Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D K_h x K_w conv/pool layer with a 3D K_t x K_h x K_w version

Can use weights of 2D conv to initialize 3D conv: copy K_t times in space and divide by K_t

This gives the same result as 2D conv given "constant" video input



Slide credit: Justin Johnson

Carreira and Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset", CVPR 2017

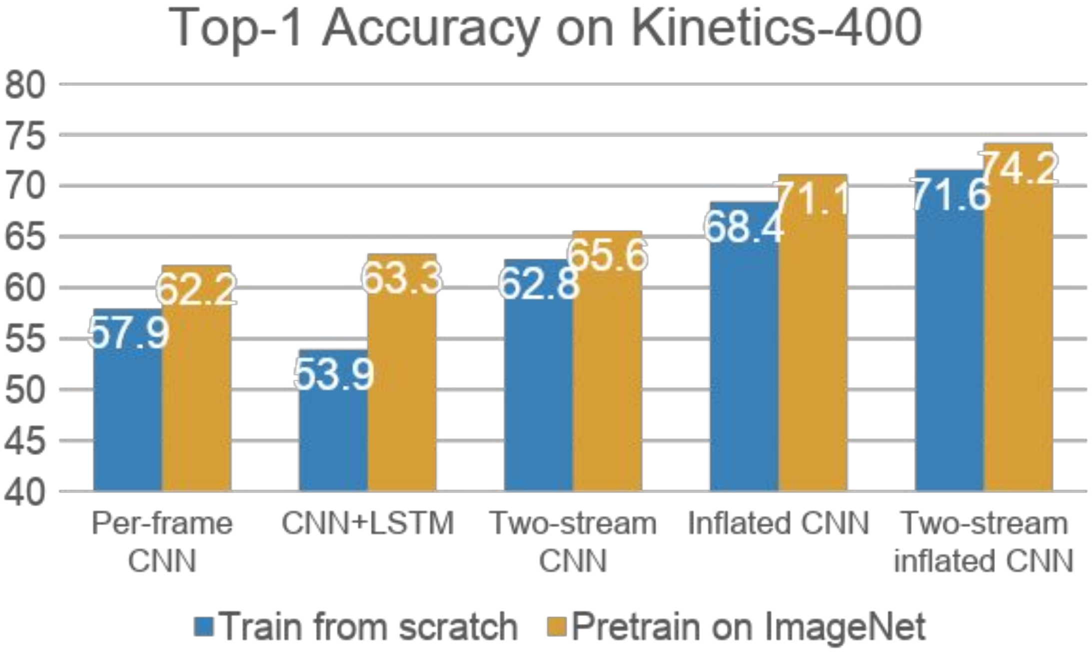
Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images. Can we reuse image architectures for video?

Idea: take a 2D CNN architecture.

Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version

Can use weights of 2D conv to initialize 3D conv: copy K_t times in space and divide by K_t
This gives the same result as 2D conv given "constant" video input

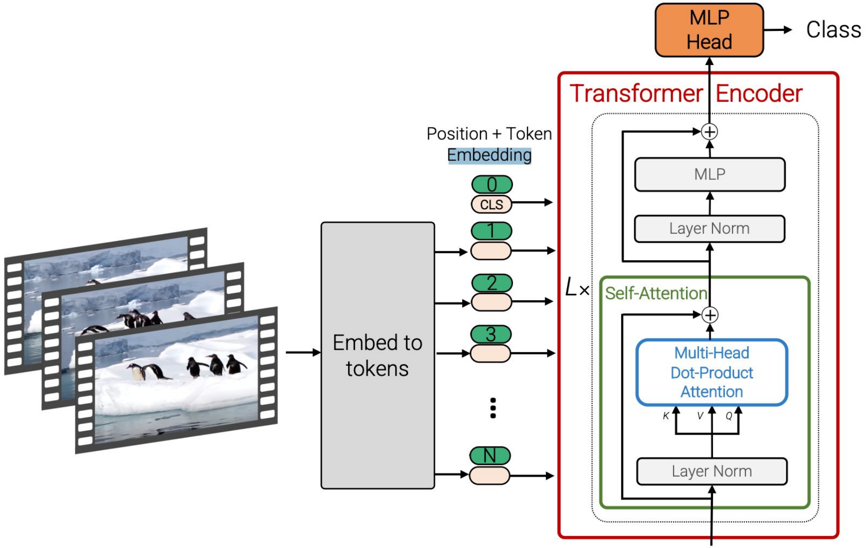


All using Inception CNN

Carreira and Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset", CVPR 2017

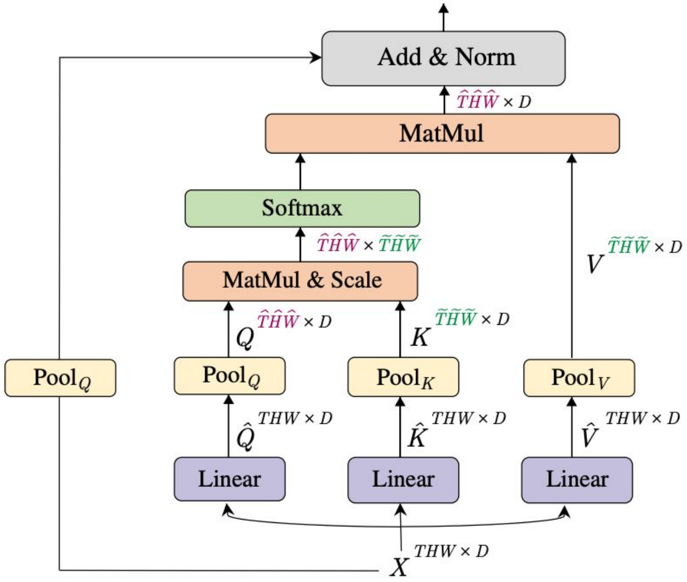
Vision Transformers for Video

Factorized attention: Attend over space / time



Bertasius et al, "Is Space-Time Attention All You Need for Video Understanding?", ICML 2021
Arnab et al, "ViViT: A Video Vision Transformer", ICCV 2021
Neimark et al, "Video Transformer Network", ICCV 2021

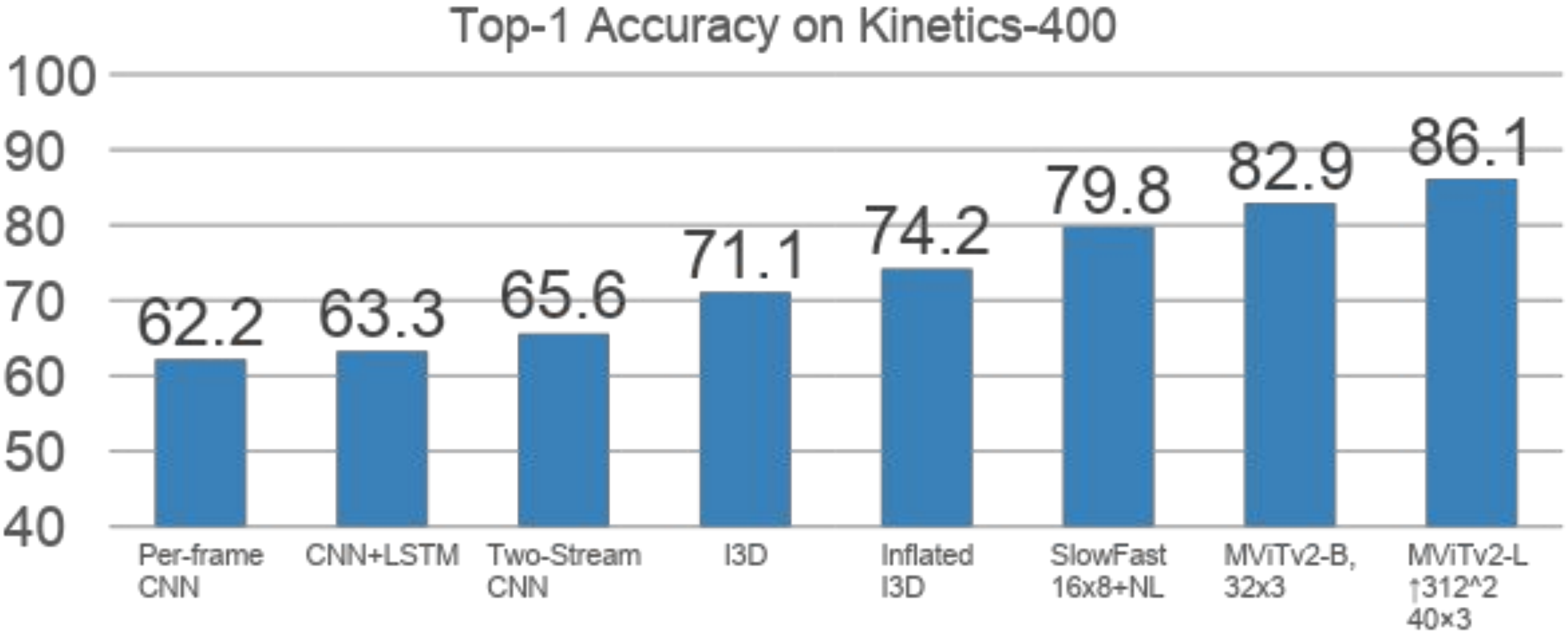
Pooling module: Reduce number of tokens



Fan et al, "Multiscale Vision Transformers", ICCV 2021
Li et al, "MViTv2: Improved Multiscale Vision Transformers for Classification and Detection", CVPR 2022

Slide credit: Justin Johnson

Vision Transformers for Video

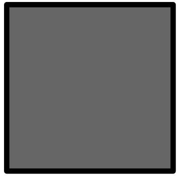


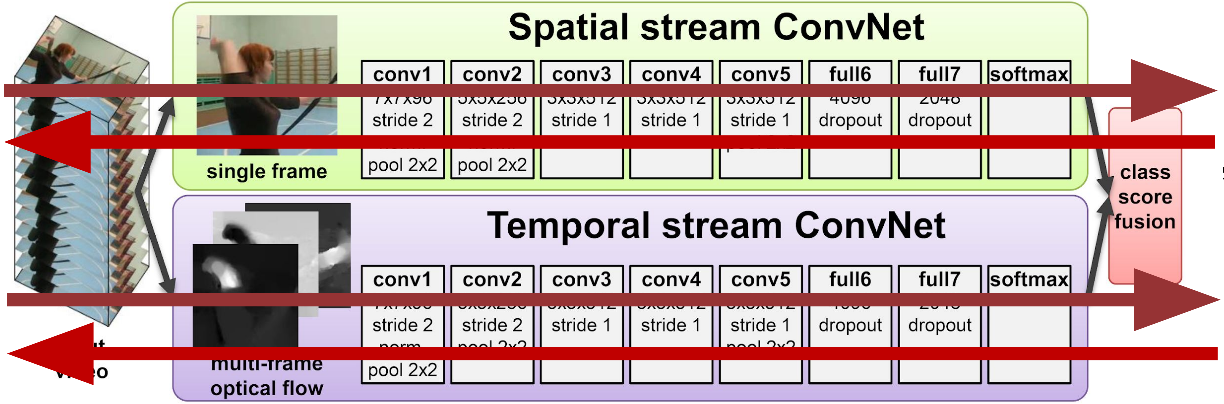
Li et al, "MViTv2: Improved Multiscale Vision Transformers for Classification and Detection", CVPR 2022

Slide credit: Justin Johnson

Visualizing Video Models

Image

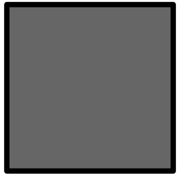




Forward: Compute class score

Backward: Compute gradient

”weightlifting” score



Flow

Add a term to encourage spatially smooth flow; tune penalty to pick out “slow” vs “fast” motion

Figure credit: Simonyan and Zisserman, “Two-stream convolutional networks for action recognition in videos”, NeurIPS 2014
Feichtenhofer et al, “What have we learned from deep representations for action recognition?”, CVPR 2018
Feichtenhofer et al, “Deep insights into convolutional networks for video recognition?”, IJCV 2019.

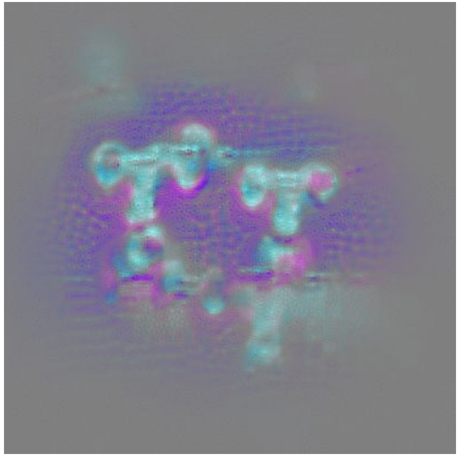
Slide credit: Justin Johnson

Can you guess the action?

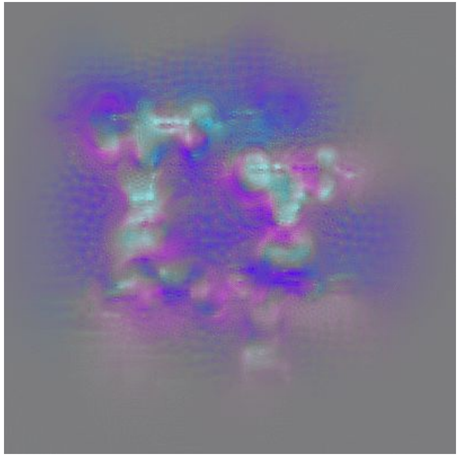
Feichtenhofer et al, "What have we learned from deep representations for action recognition?", CVPR 2018
Feichtenhofer et al, "Deep insights into convolutional networks for video recognition?", IJCV 2019.
Slide credit: Christoph Feichtenhofers

Appearance

"Slow" motion



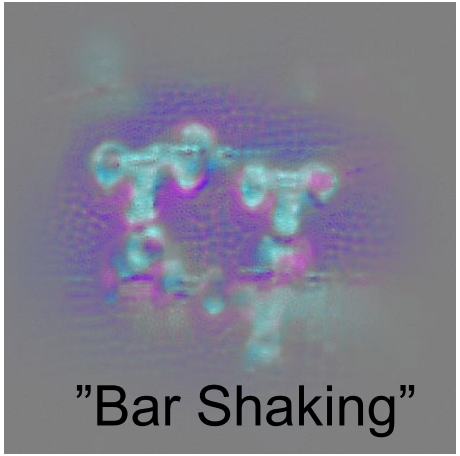
"Fast" motion

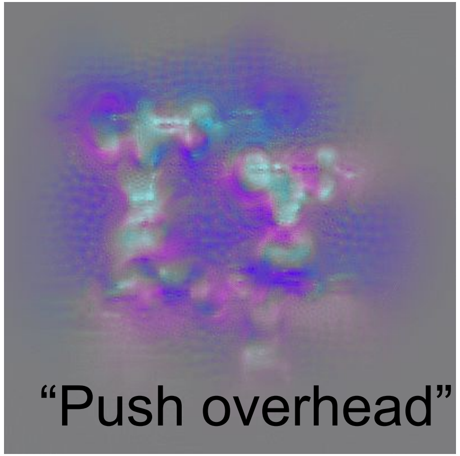


Slide credit: Justin Johnson

Can you guess the action? Weightlifting

Appearance	"Slow" motion	"Fast" motion

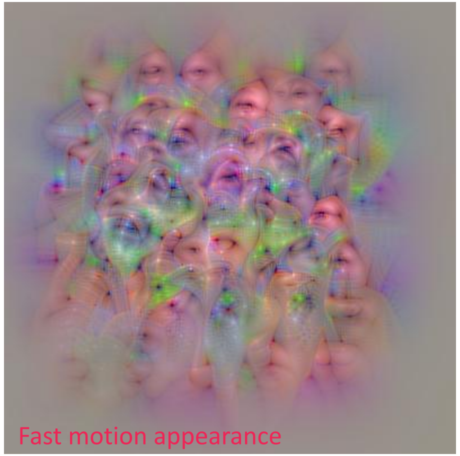




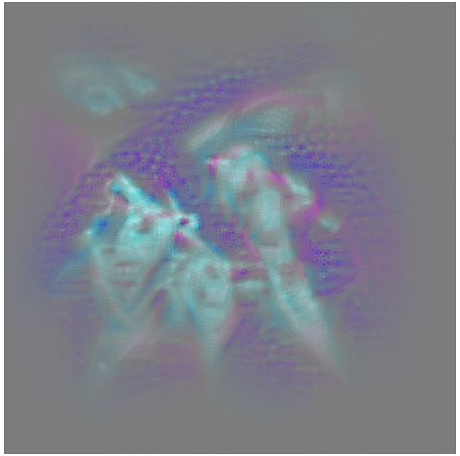
Slide credit: Justin Johnson

Can you guess the action?

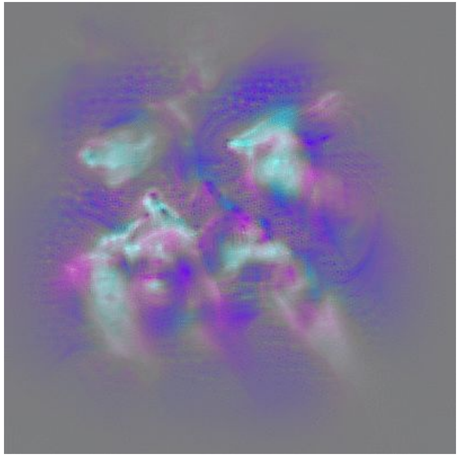
Appearance



"Slow" motion



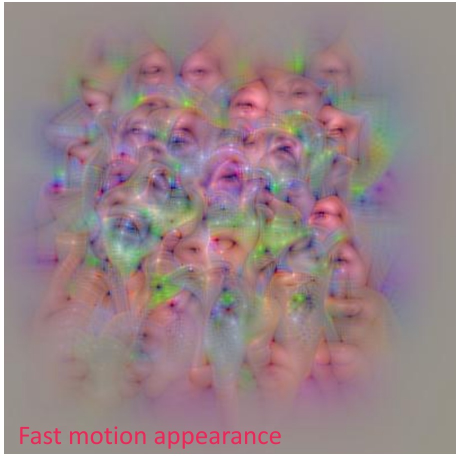
"Fast" motion



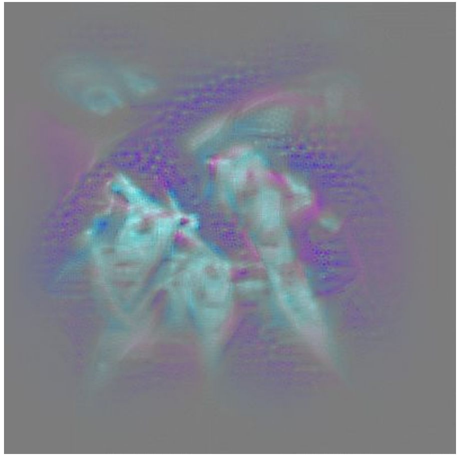
Slide credit: Justin Johnson

Can you guess the action?   Apply Eye Makeup

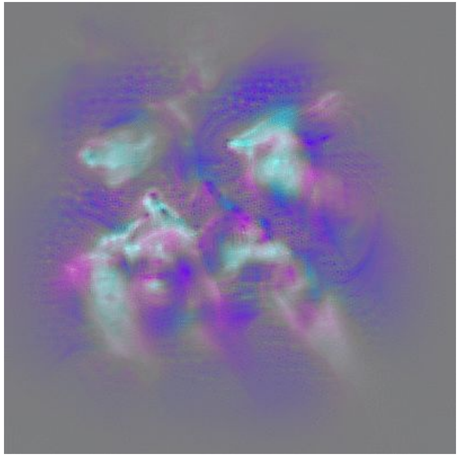
Appearance



“Slow” motion



“Fast” motion



Slide credit: Justin Johnson

So far: Classify short clips

Videos: Recognize **actions**

Swimming
Running
Jumping
Eating
Standing

Slide credit: Justin Johnson

Temporal Action Localization

Given a long untrimmed video sequence, identify frames corresponding to different actions

Can use architecture similar to Faster R-CNN: first generate **temporal proposals** then **classify**

Chao et al, " Rethinking the Faster R-CNN Architecture for Temporal Action Localization", CVPR 2018

Slide credit: Justin Johnson

Spatio-Temporal Detection

Given a long untrimmed video, detect all the people in both space and time and classify the activities they are performing. Some examples from AVA Dataset:

Gu et al, “AVA: A Video Dataset of Spatio-temporally Localized Atomic Visual Actions”, CVPR 2018

Slide credit: Justin Johnson

Today: Temporal Stream

3D CNN, Two-Stream Neural Network, Spatial-Temporal Self-Attention……

Ba Ba Ba

...

Video source: BBC

(McGurk & McDonald 1976)


BBC TWO
Fa Fa Fa ...
Video source: BBC
(McGurk & McDonald 1976)


Video source: BBC
(McGurk & McDonald 1976)

Visually-guided audio source separation

[Gao et al. ECCV 2018, Afouras et al. Interspeech'18, Gabby et al. Interspeech'18, Owens & Efros ECCV'18, Ephrat et al. SIGGRAPH'18, Zhao et al. ECCV 2018, Gao & Grauman ICCV 2019, Zhao et al. ICCV 2019, Xu et al. ICCV 2019, Gan et al. CVPR 2020, Gao et al. CVPR 2021]

Gao et al., VisualVoice, CVPR 2021

Gao et al., VisualVoice, CVPR 2021

Separated voice for the right speaker

Gao et al., VisualVoice, CVPR 2021

Musical instruments source separation

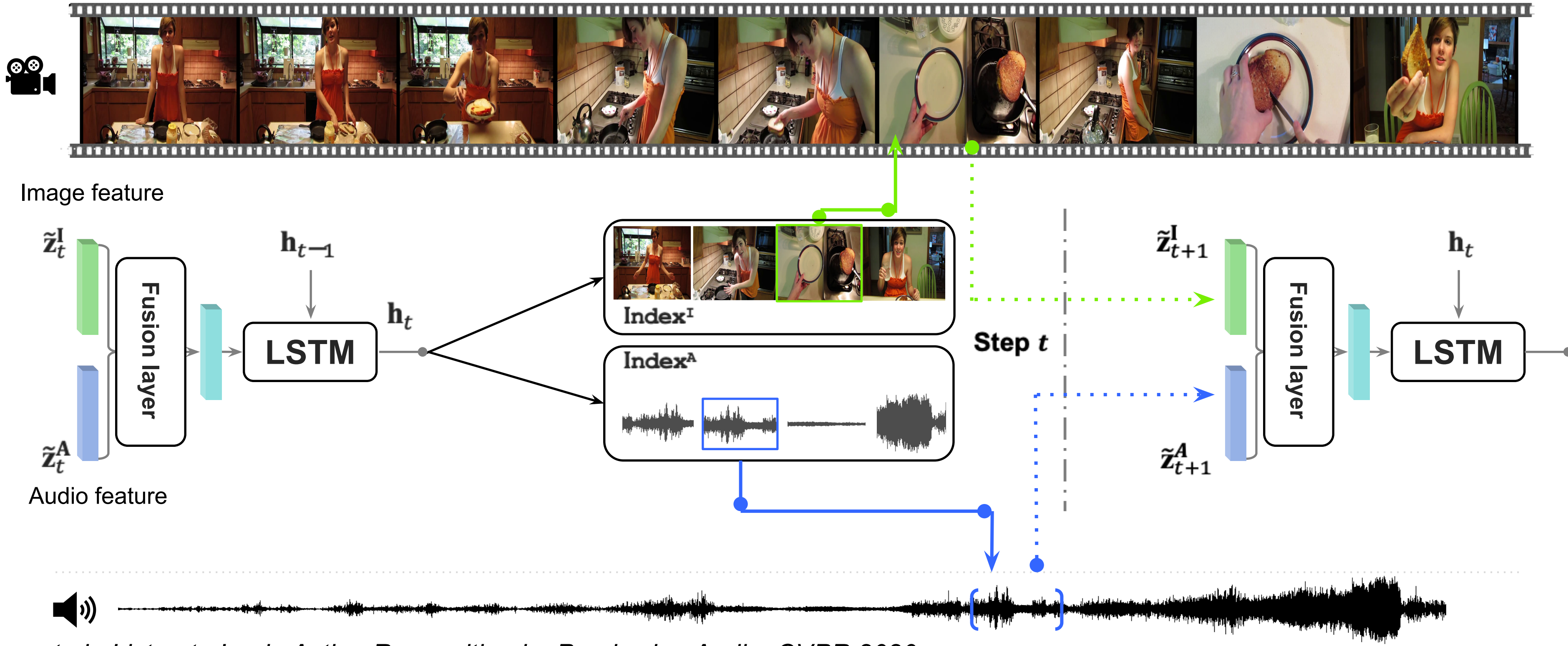
Train on 100,000 unlabeled multi-source video clips, then separate audio for novel video.

original video
(before separation)

object detections:
violin & flute

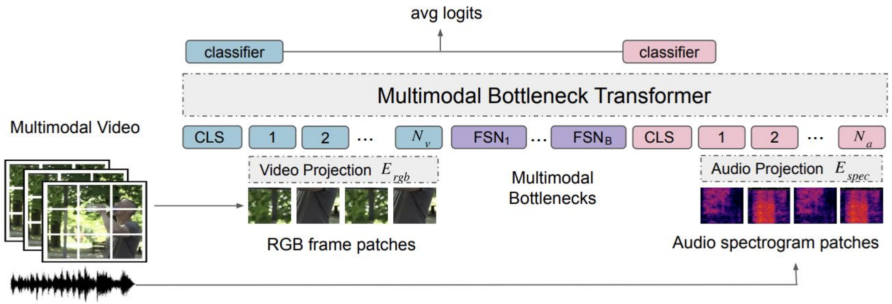
Gao & Grauman, Co-Separating Sounds of Visual Objects, ICCV 2019

Audio as a preview mechanism for efficient action recognition in untrimmed videos

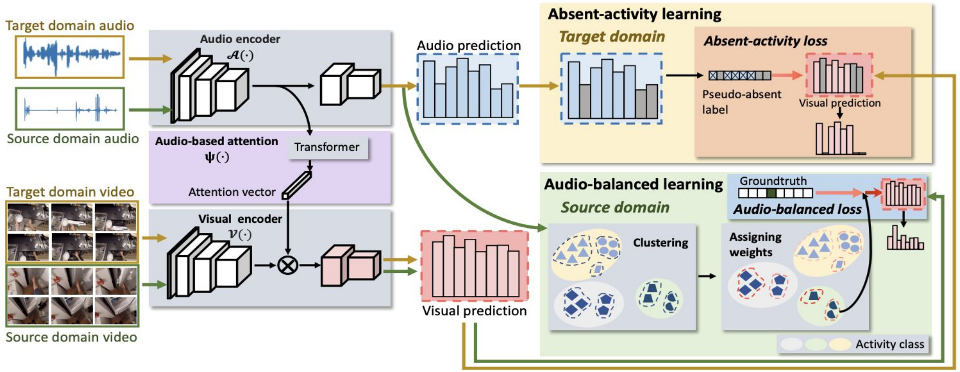


Gao et al., Listen to Look: Action Recognition by Previewing Audio, CVPR 2020

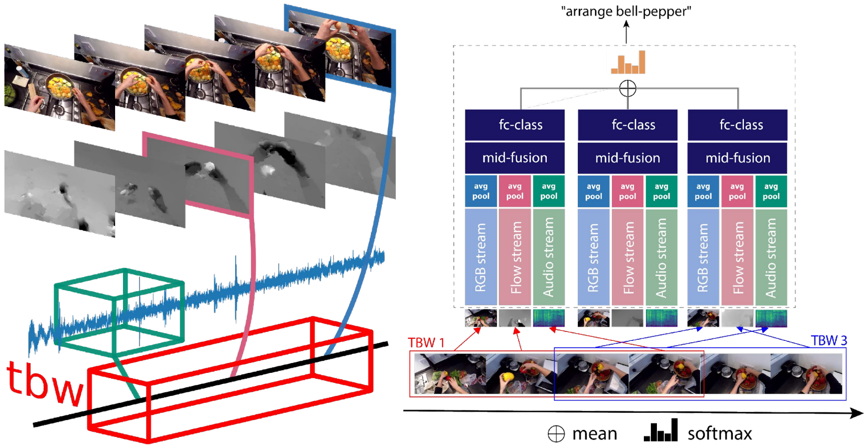
Multimodal Video Understanding



Attention Bottlenecks for Multimodal Fusion, Nagrani et al. NeurIPS 2021

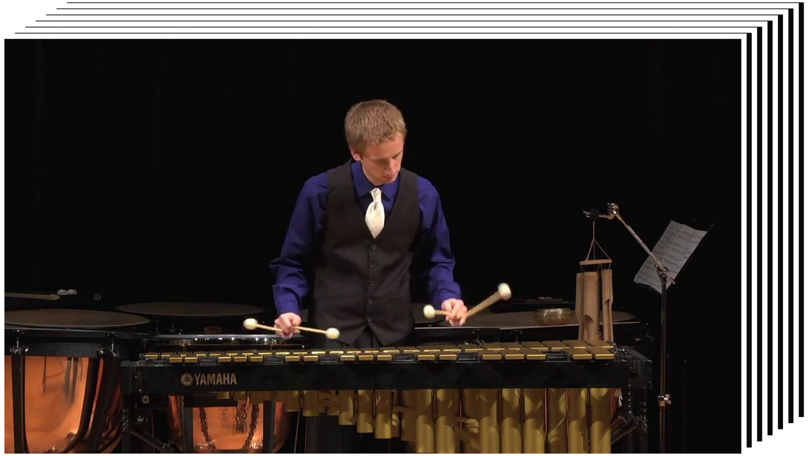


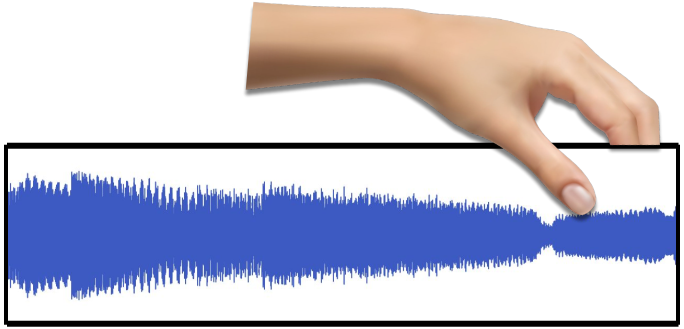
Audio-Adaptive Activity Recognition Across Video Domains, Yunhua et al. CVPR 2022



EPIC-Fusion: Audio-Visual Temporal Binding for Egocentric Action Recognition, Kazakos et al., ICCV 2019

Learning audio-visual synchronization



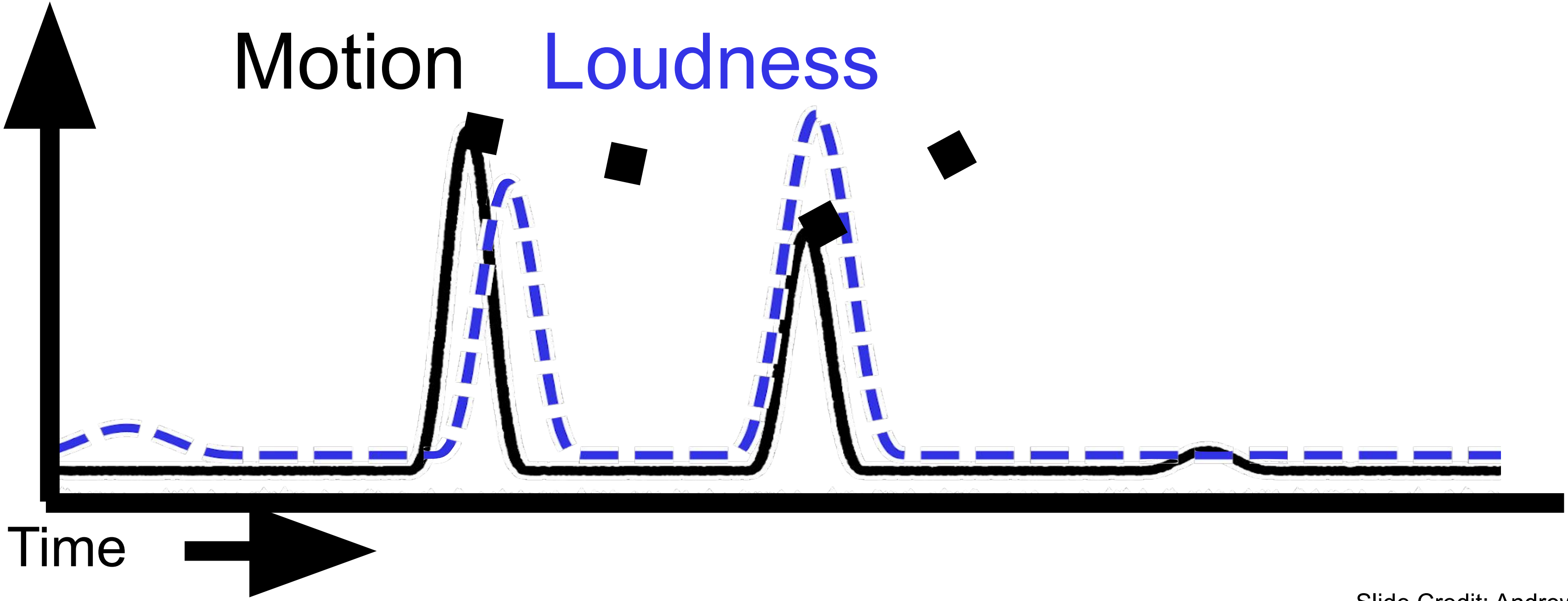


Owens & Efros, Audio-visual scene analysis with self-supervised multisensory features, ECCV 2018
Korbar et al., Co-training of audio and video representations from self-supervised temporal synchronization, NeurIPS 2018

Learning audio-visual synchronization

Owens & Efros, Audio-visual scene analysis with self-supervised multisensory features, ECCV 2018

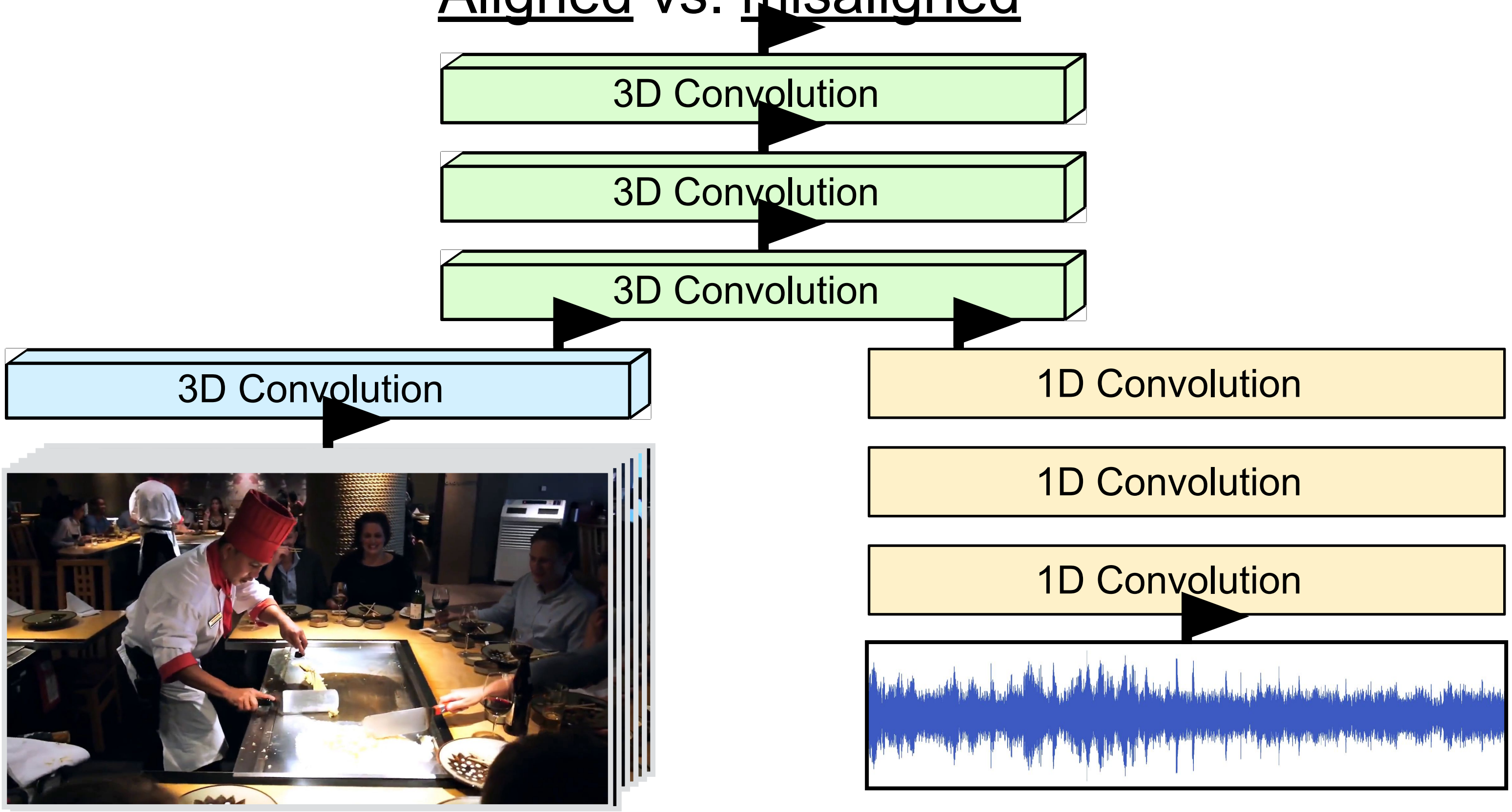
Learning audio-visual synchronization



Slide Credit: Andrew Owens

Learning audio-visual synchronization

Aligned vs. misaligned



Owens & Efros, Audio-visual scene analysis with self-supervised multisensory features, ECCV 2018

Top responses in test set

Owens & Efros, Audio-visual scene analysis with self-supervised multisensory features, ECCV 2018

Sound source localization

Owens & Efros, Audio-visual scene analysis with self-supervised multisensory features, ECCV 2018
Arandjelović and Zisserman, ECCV 2018; Senocak et al. CVPR 2018; Kidron et al. CVPR 2005 …

Next time: Object Detection and Image Segmentation