

# Lecture 12: Video Understanding

# Administrative

- Project milestone due May 7<sup>th</sup> Saturday 11:59pm PT
- Check Ed and course website for requirements

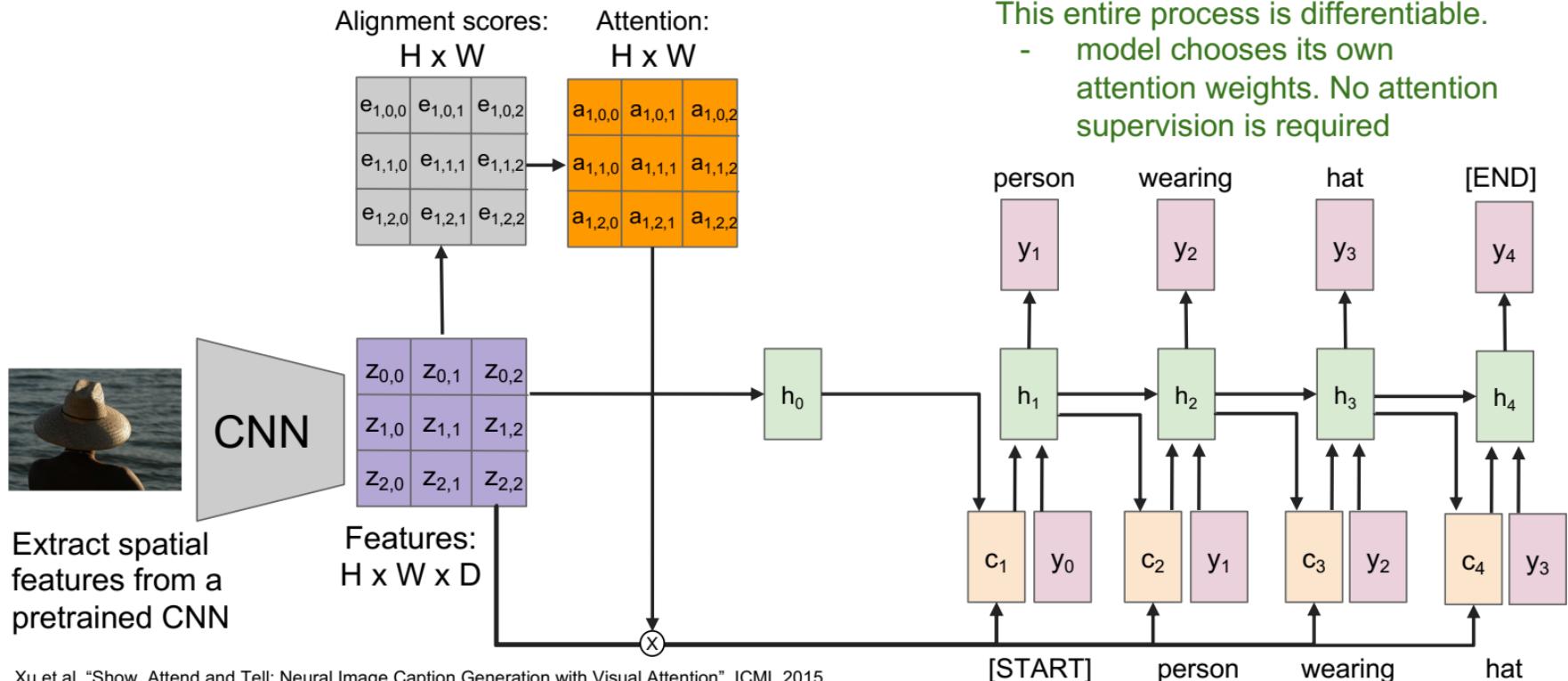
# Administrative

- In-Class Midterm next Tuesday May 6<sup>th</sup> 1:30-3:00pm PT  
Check Ed for details!
- Check sample midterm and solutions  
Midterm review session tomorrow!
- 20% of your grade

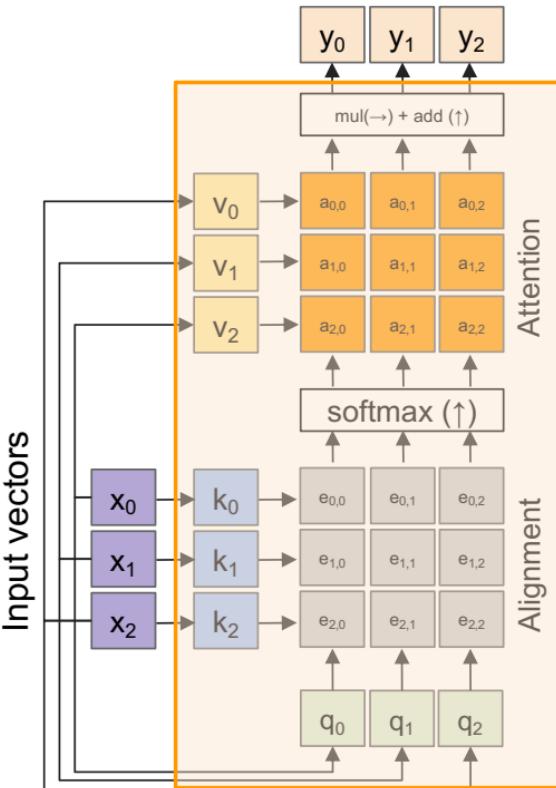
# Administrative

- Assignment 3 will be released on May 6<sup>th</sup> after the midterm

# Last time: Image Captioning with RNNs and Attention



# Last time: Self-Attention



**Outputs:**

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

**Operations:**

Key vectors:  $\mathbf{k} = \mathbf{x}\mathbf{W}_k$

Value vectors:  $\mathbf{v} = \mathbf{x}\mathbf{W}_v$

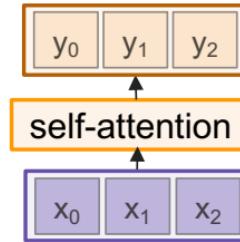
Query vectors:  $\mathbf{q} = \mathbf{x}\mathbf{W}_q$

Alignment:  $e_{i,j} = \mathbf{q}_j \cdot \mathbf{k}_i / \sqrt{D}$   
Attention:  $\mathbf{a} = \text{softmax}(\mathbf{e})$

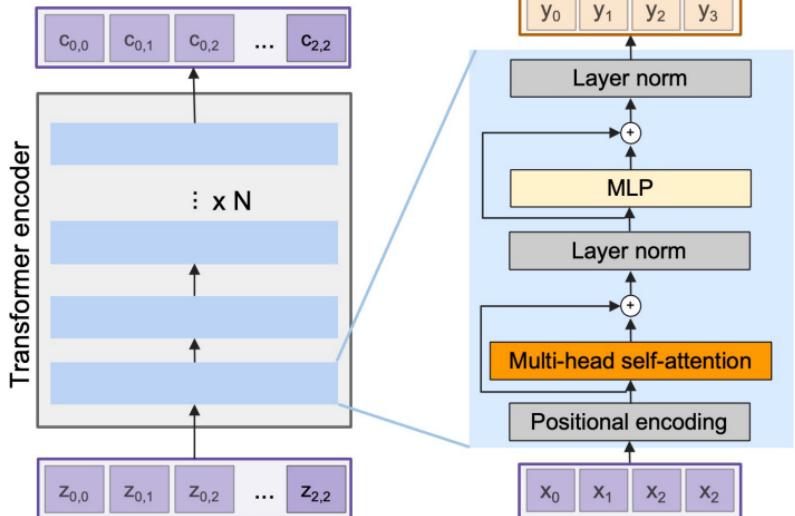
Output:  $y_j = \sum_i a_{i,j} \mathbf{v}_i$

**Inputs:**

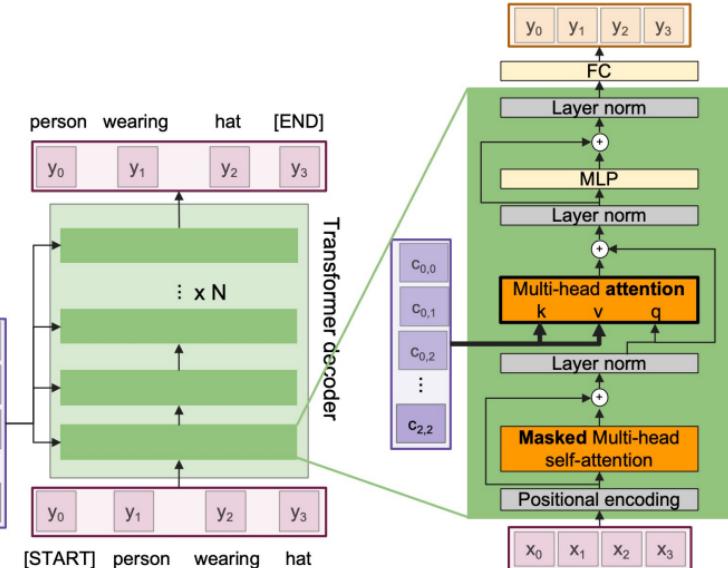
Input vectors:  $\mathbf{x}$  (shape:  $N \times D$ )



# Last time: Transformer

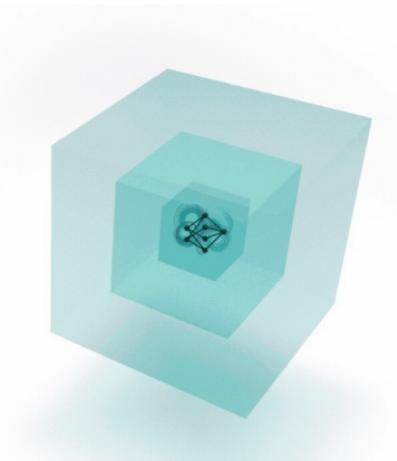


Encoder



Decoder

# Last time: Large foundation models



OPT-175B: 175 billion-parameter language model

Open sourced:

<https://github.com/facebookresearch/metaseq/tree/main/projects/OPT>

OPT: Open Pre-trained Transformer Language Models, Zhang et al. 2022

<https://ai.facebook.com/blog/democratizing-access-to-large-scale-language-models-with-opt-175b/>

# 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

# Recall: (2D) Detection and Segmentation

Classification



CAT

No spatial extent

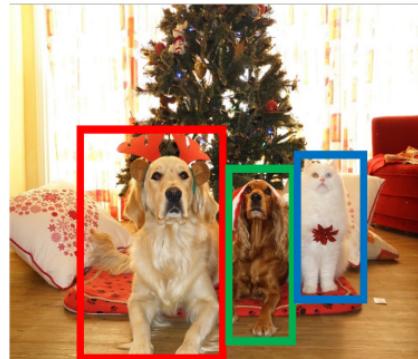
Semantic Segmentation



GRASS, CAT,  
TREE, SKY

No objects, just pixels

Object Detection



DOG, DOG, CAT

Multiple Objects

Instance Segmentation



DOG, DOG, CAT

This image is CC0 public domain

# Living room

Dog

Baby



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

A video is a **sequence** of images

4D tensor:  $T \times 3 \times H \times W$   
(or  $3 \times T \times H \times W$ )



[This image](#) is CC0 public domain

# Example task: Video Classification



Swimming  
**Running**  
Jumping  
Eating  
Standing

Input video:

$T \times 3 \times H \times W$

# Example task: Video Classification



Images: Recognize **objects**



Dog  
**Cat**  
Fish  
Truck



Videos: Recognize **actions**



Swimming  
**Running**  
Jumping  
Eating  
Standing

# Problem: Videos are big!

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**

Input video:

$T \times 3 \times H \times W$

Slide credit: Justin Johnson

# Problem: Videos are big!

Videos are ~30 frames per second (fps)



Input video:  
 $T \times 3 \times H \times W$

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



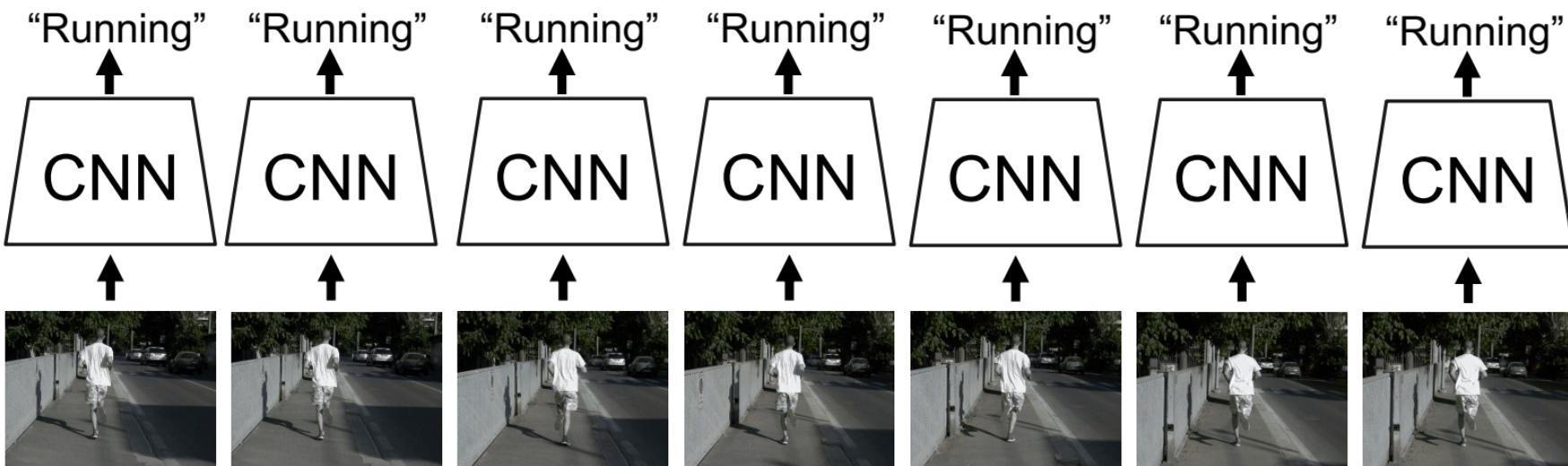
**Testing:** Run model on different clips, average predictions



Slide credit: Justin Johnson

# Video Classification: Single-Frame CNN

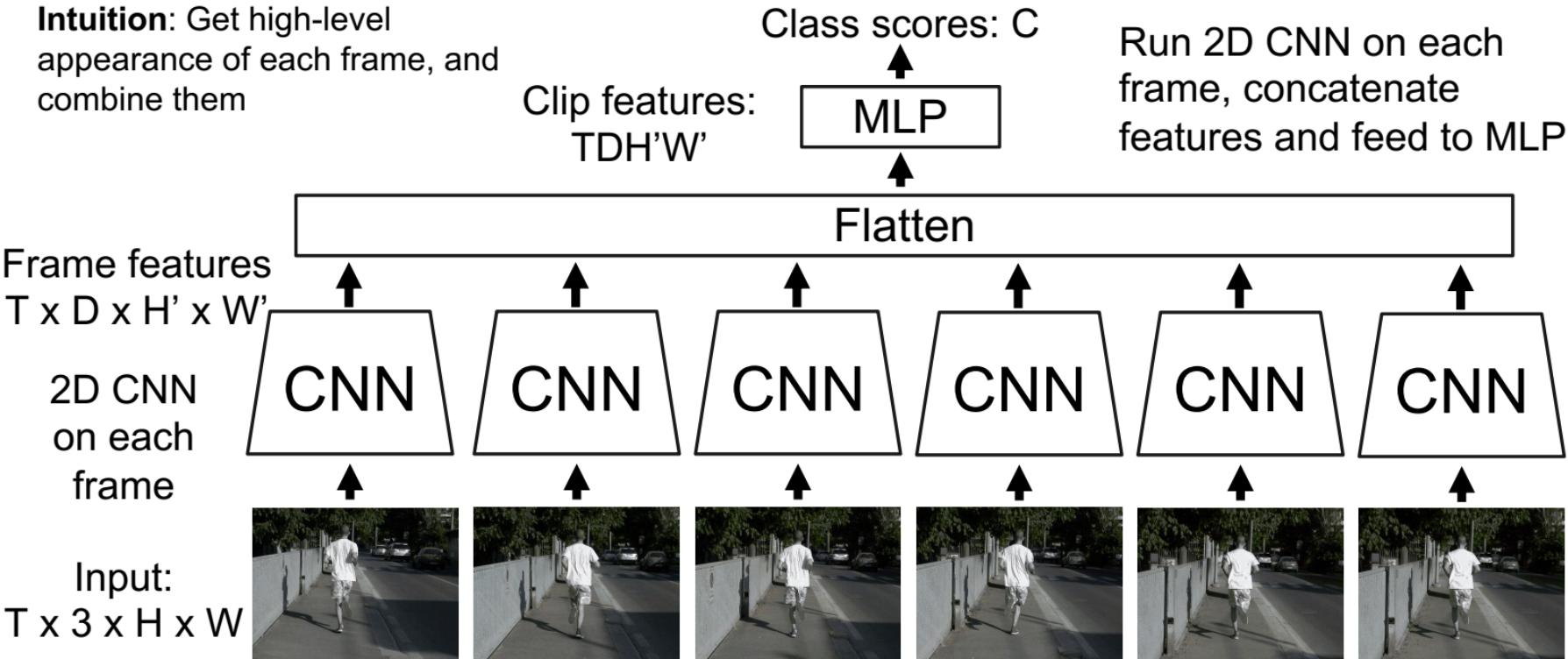
Simple idea: train normal 2D CNN to classify video frames independently!  
(Average predicted probs at test-time)  
Often a **very** strong baseline for video classification



Slide credit: Justin Johnson

# Video Classification: Late Fusion (with FC layers)

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

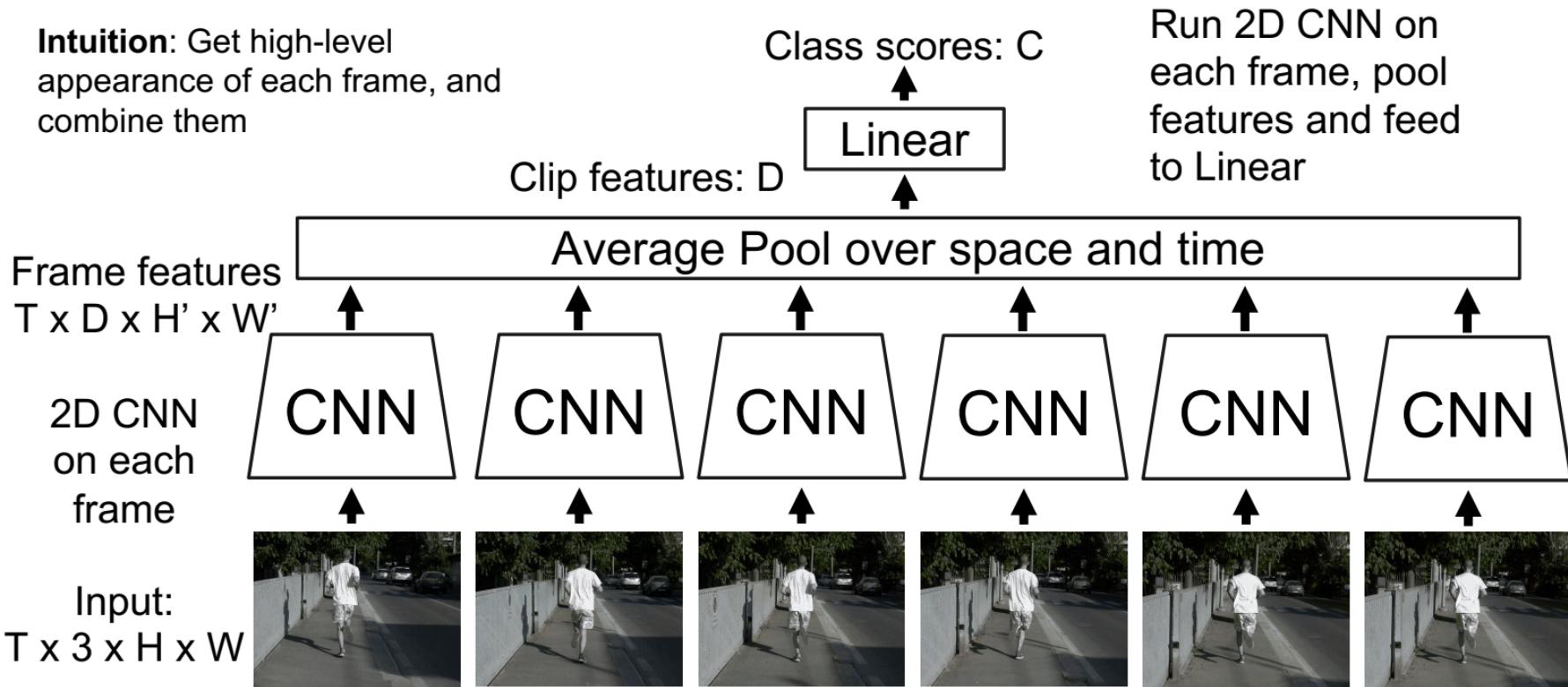


Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

Slide credit: Justin Johnson

# Video Classification: Late Fusion (with pooling)

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

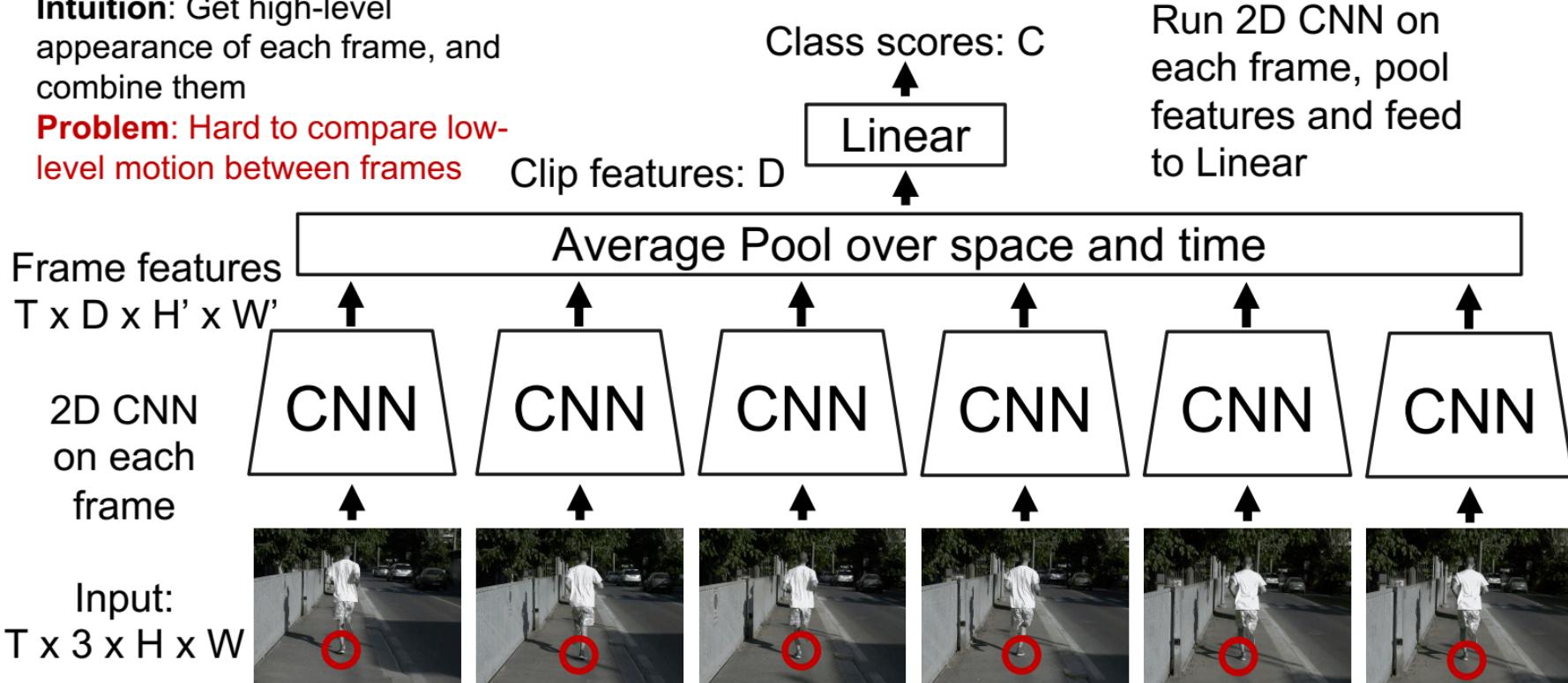


Slide credit: Justin Johnson

# 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



Slide credit: Justin Johnson

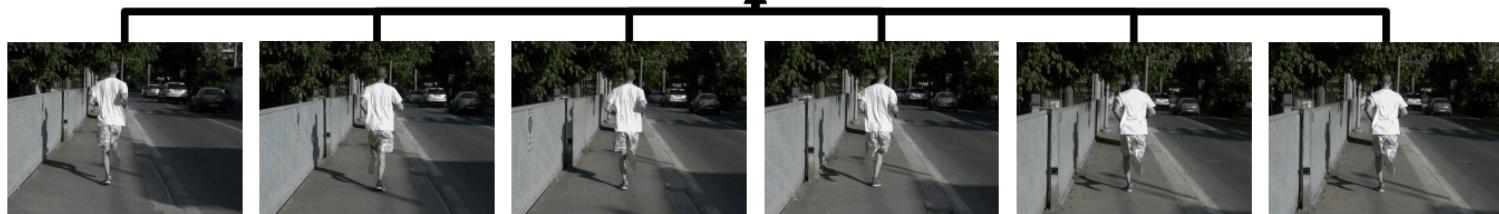
# Video Classification: Early Fusion

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

Reshape:  
 $3T \times H \times W$

Input:  
 $T \times 3 \times H \times W$

First 2D convolution collapses all temporal information:  
**Input:**  $3T \times H \times W$   
**Output:**  $D \times H \times W$



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

# 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!

First 2D convolution collapses all temporal information:

**Input:**  $3T \times H \times W$

**Output:**  $D \times H \times W$

Reshape:  
 $3T \times H \times W$

**Input:**  
 $T \times 3 \times H \times W$

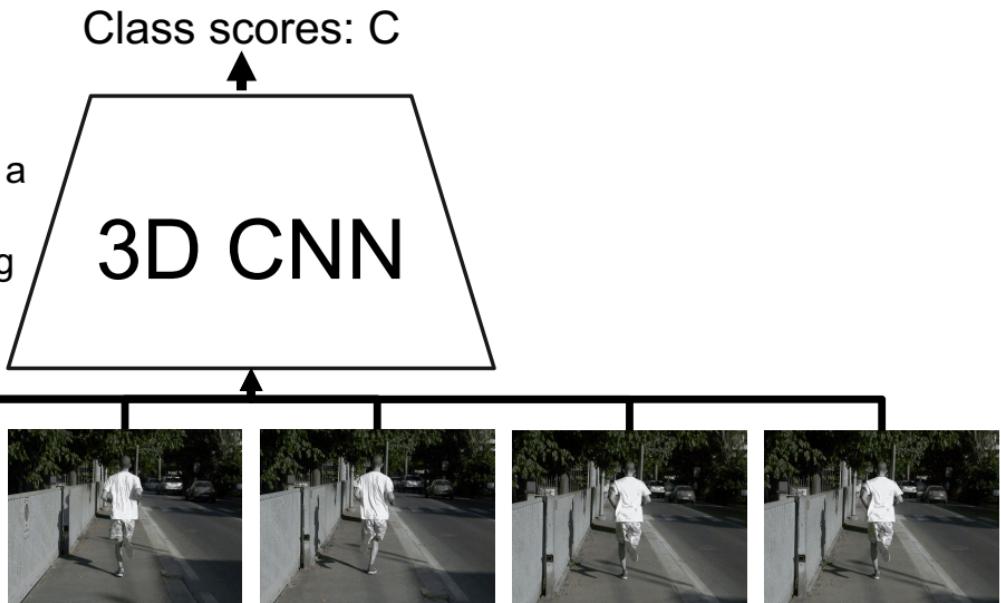


Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

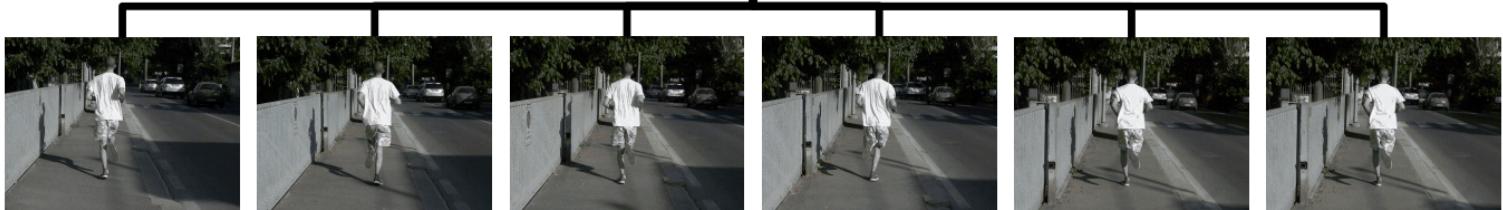
# Video Classification: 3D CNN

**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:  $D \times T \times H \times W$   
Use 3D conv and 3D pooling operations



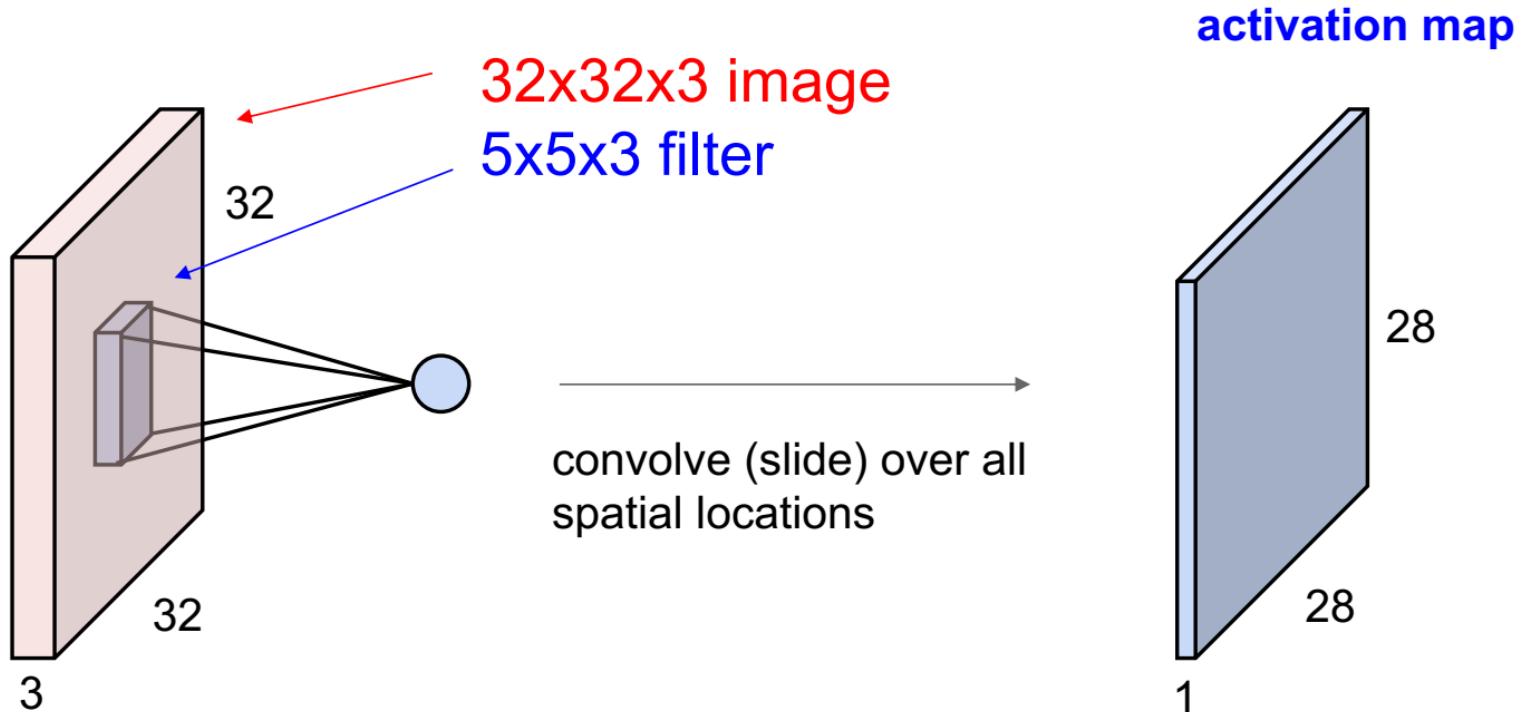
**Input:**  
 $3 \times T \times H \times W$



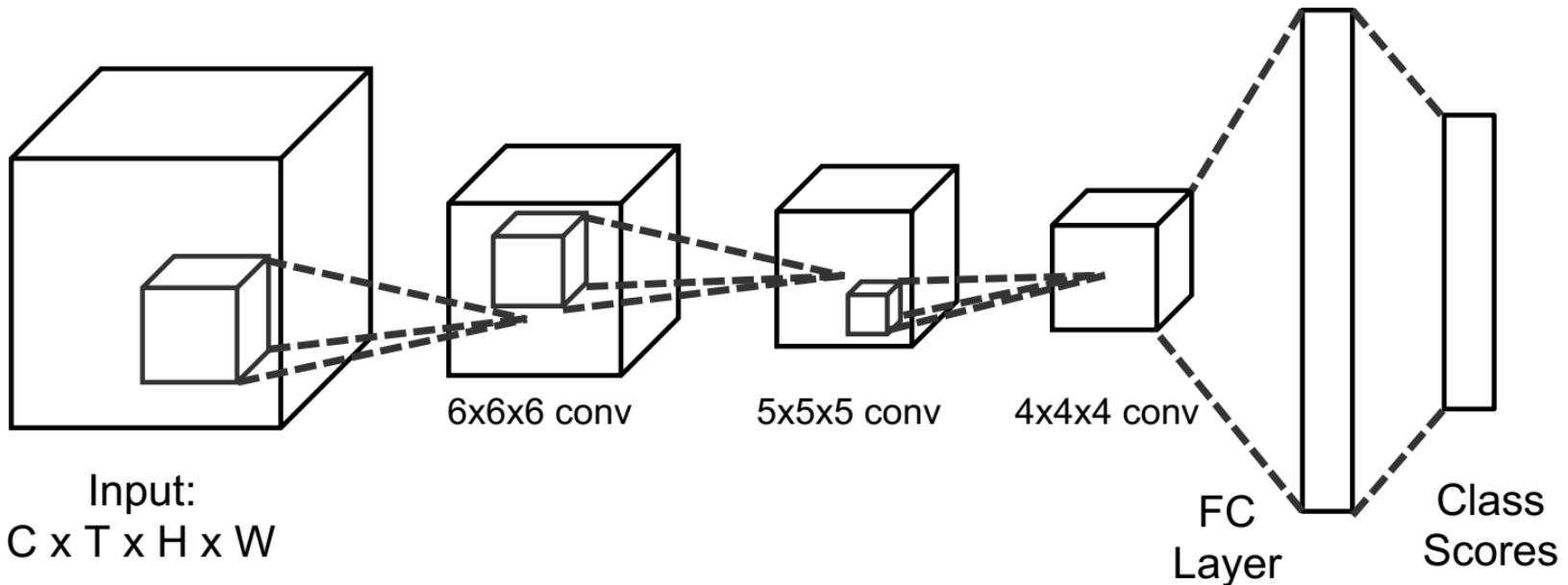
Ji et al, "3D Convolutional Neural Networks for Human Action Recognition", TPAMI 2010 ; Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014

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

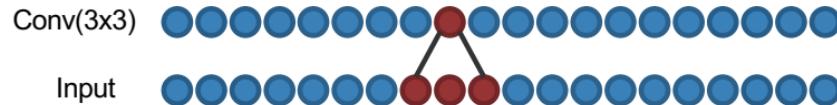
(Small example  
architectures, in  
practice much  
bigger)

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



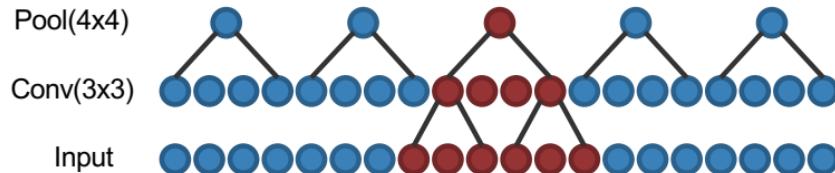
(Small example  
architectures, in  
practice much  
bigger)

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)

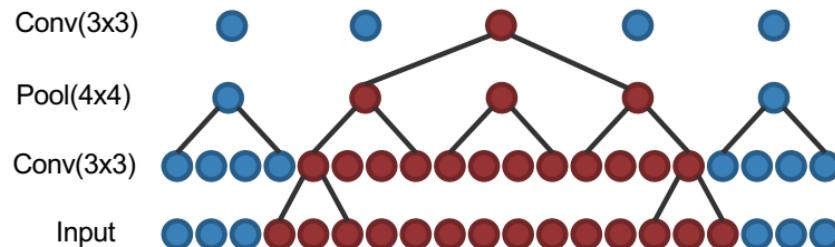
Slide credit: Justin Johnson

# 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	$3 \times 20 \times 64 \times 64$	
Conv2D(3x3, 3->12)	$12 \times 20 \times 64 \times 64$	$1 \times 3 \times 3$
Pool2D(4x4)	$12 \times 20 \times 16 \times 16$	$1 \times 6 \times 6$
Conv2D(3x3, 12->24)	$24 \times 20 \times 16 \times 16$	$1 \times 14 \times 14$

Build slowly in space



(Small example  
architectures, in  
practice much  
bigger)

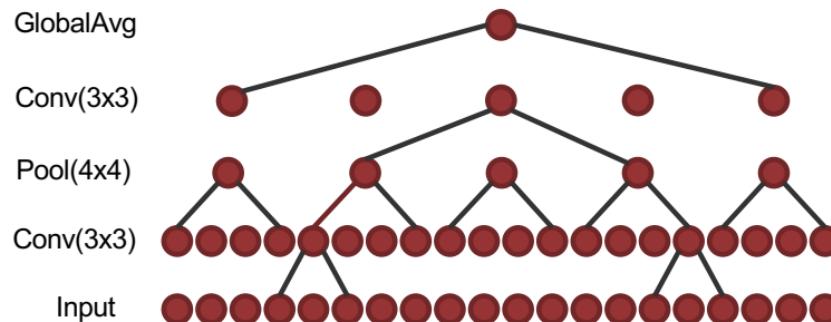
Slide credit: Justin Johnson

# 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	$3 \times 20 \times 64 \times 64$	
Conv2D(3x3, 3->12)	$12 \times 20 \times 64 \times 64$	$1 \times 3 \times 3$
Pool2D(4x4)	$12 \times 20 \times 16 \times 16$	$1 \times 6 \times 6$
Conv2D(3x3, 12->24)	$24 \times 20 \times 16 \times 16$	$1 \times 14 \times 14$
GlobalAvgPool	$24 \times 1 \times 1 \times 1$	$20 \times 64 \times 64$

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

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

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 start

(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

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 start

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

Build slowly in space,  
Build slowly in time  
"Slow Fusion"

(Small example  
architectures, in  
practice much  
bigger)

Slide credit: Justin Johnson

# Early Fusion vs Late Fusion vs 3D CNN

What is the difference?

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

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 start

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

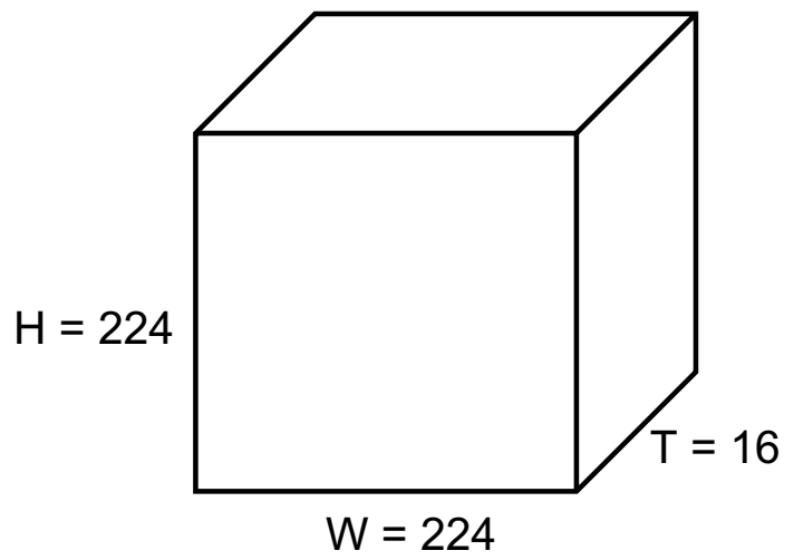
Build slowly in space,  
Build slowly in time  
"Slow Fusion"

(Small example architectures, in practice much bigger)

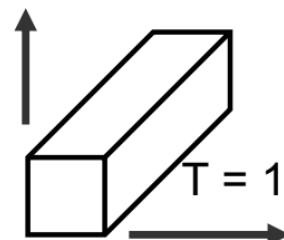
Slide credit: Justin Johnson

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

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

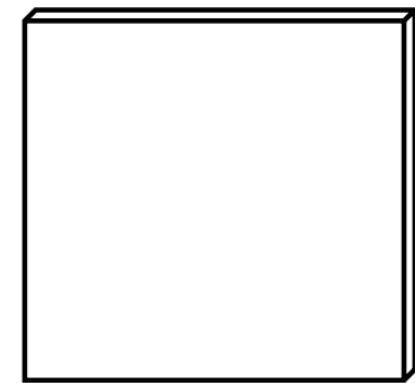


**Weight:**  
 $C_{out} \times C_{in} \times T \times 3 \times 3$   
Slide over x and y



$C_{out}$  different filters

**Output:**  
 $C_{out} \times H \times W$   
2D grid with  $C_{out}$  – dim  
feat at each point



Slide credit: Justin Johnson

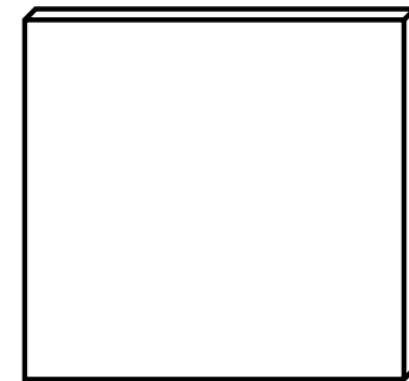
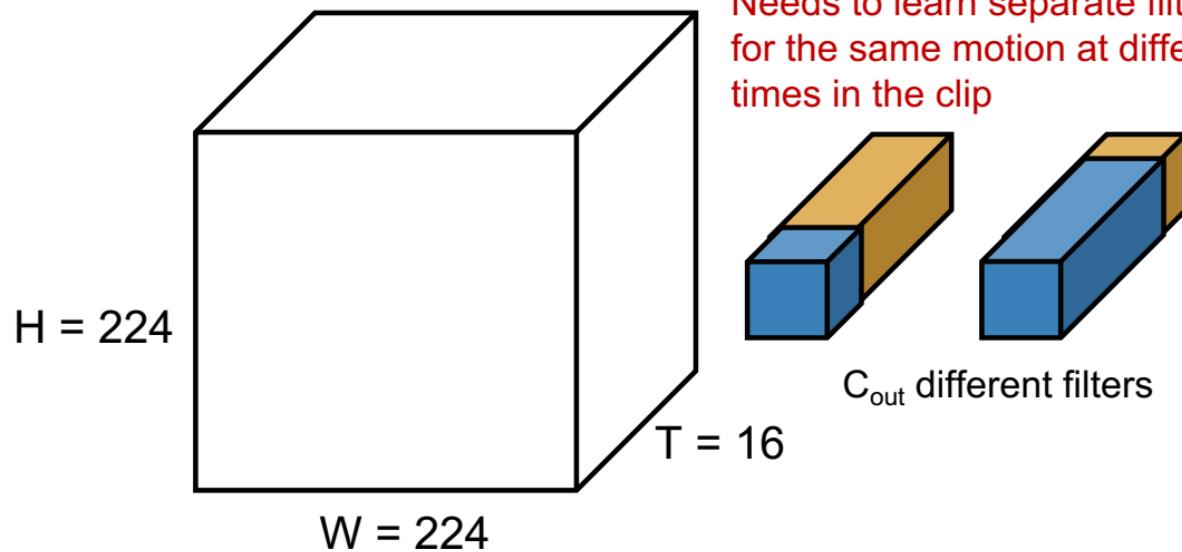
# 2D Conv (Early Fusion) vs 3D Conv (3D CNN)

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

**Weight:**  
 $C_{out} \times C_{in} \times T \times 3 \times 3$   
Slide over x and y

**Output:**  
 $C_{out} \times H \times W$   
2D grid with  $C_{out}$ -dim  
feat at each point

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



Slide credit: Justin Johnson

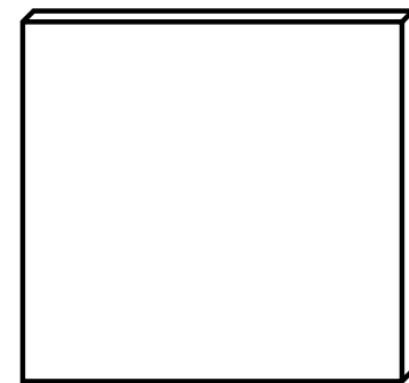
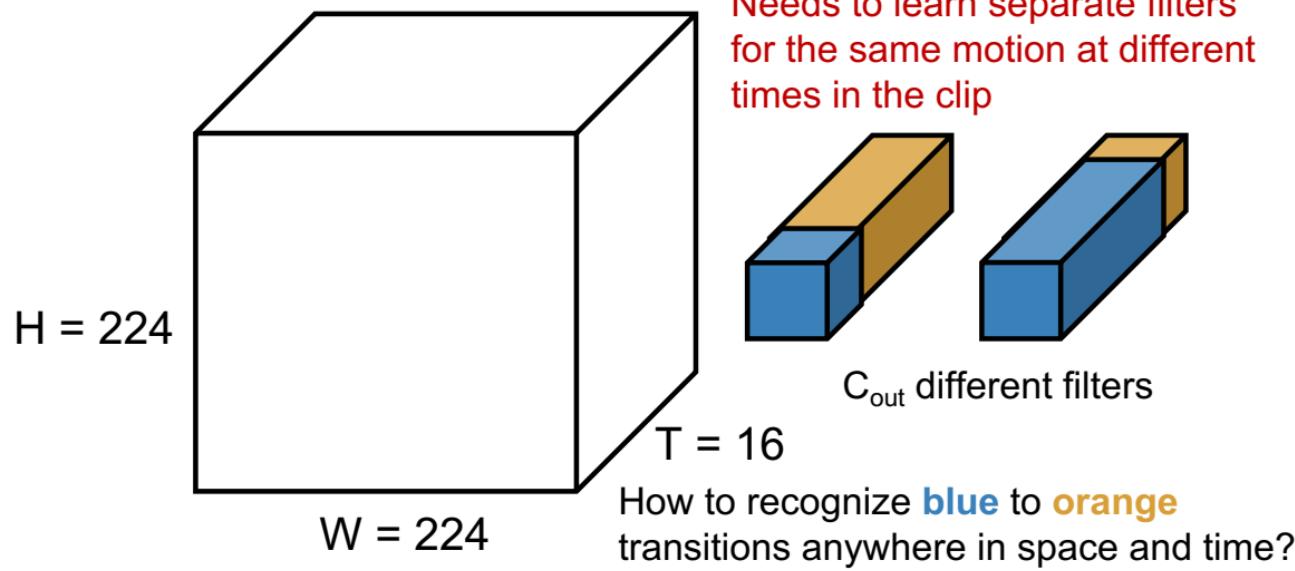
# 2D Conv (Early Fusion) vs 3D Conv (3D CNN)

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

**Weight:**  
 $C_{out} \times C_{in} \times T \times 3 \times 3$   
Slide over x and y

**Output:**  
 $C_{out} \times H \times W$   
2D grid with  $C_{out}$ -dim  
feat at each point

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



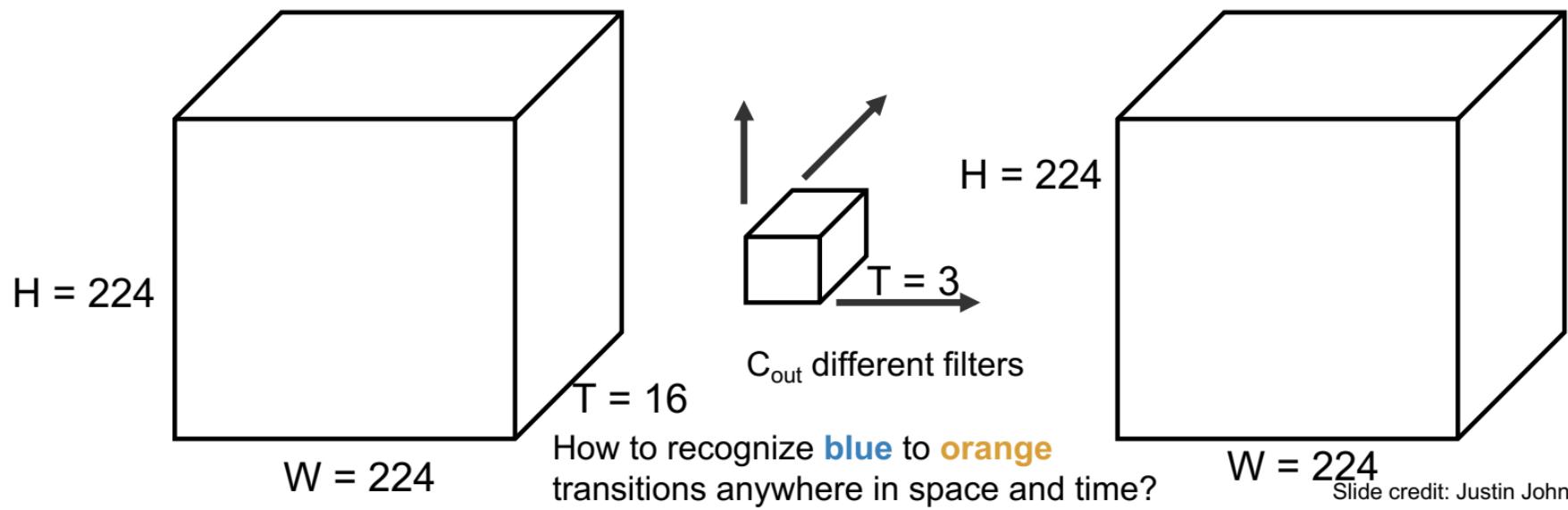
Slide credit: Justin Johnson

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

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

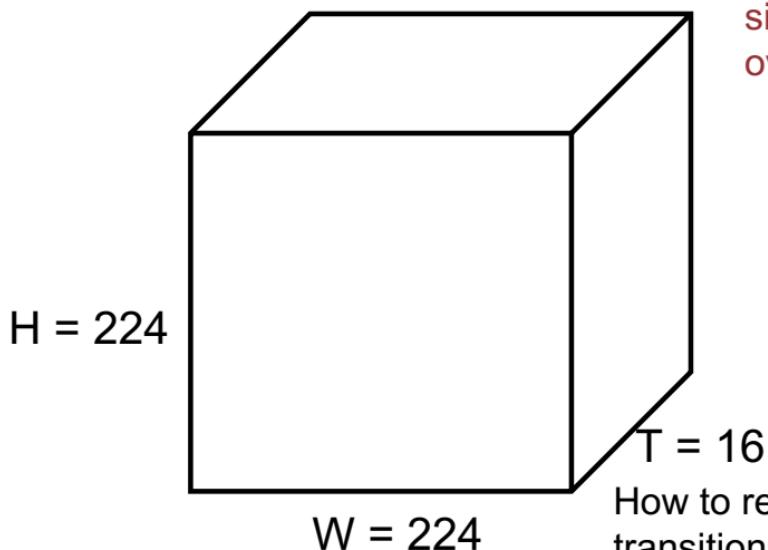
**Weight:**  
 $C_{out} \times C_{in} \times 3 \times 3 \times 3$   
Slide over x and y

**Output:**  
 $C_{out} \times T \times H \times W$   
3D grid with  $C_{out}$ -dim  
feat at each point



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

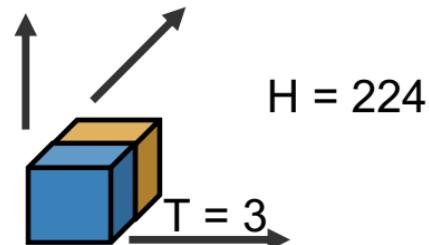
**Input:**  $C_{in} \times T \times H \times W$   
(3D grid with  $C_{in}$ -dim  
feat at each point)



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

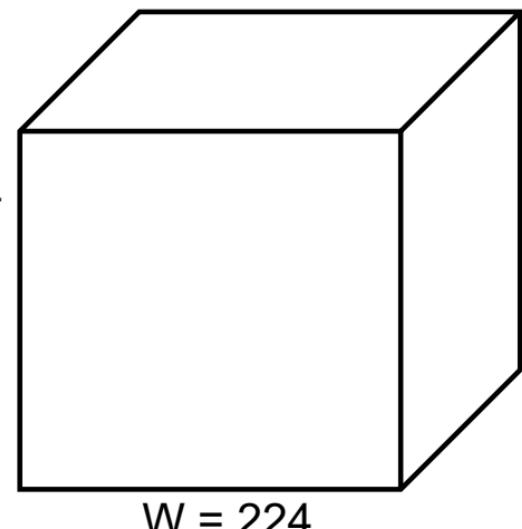
**Weight:**  
 $C_{out} \times C_{in} \times 3 \times 3 \times 3$   
Slide over x and y

Temporal shift-invariant  
since each filter slides  
over time!



$C_{out}$  different filters

**Output:**  
 $C_{out} \times T \times H \times W$   
3D grid with  $C_{out}$ -dim  
feat at each point



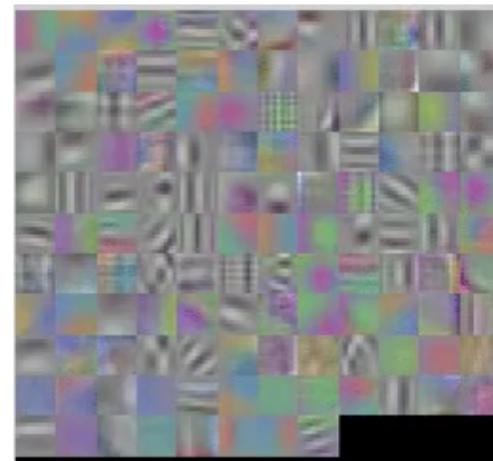
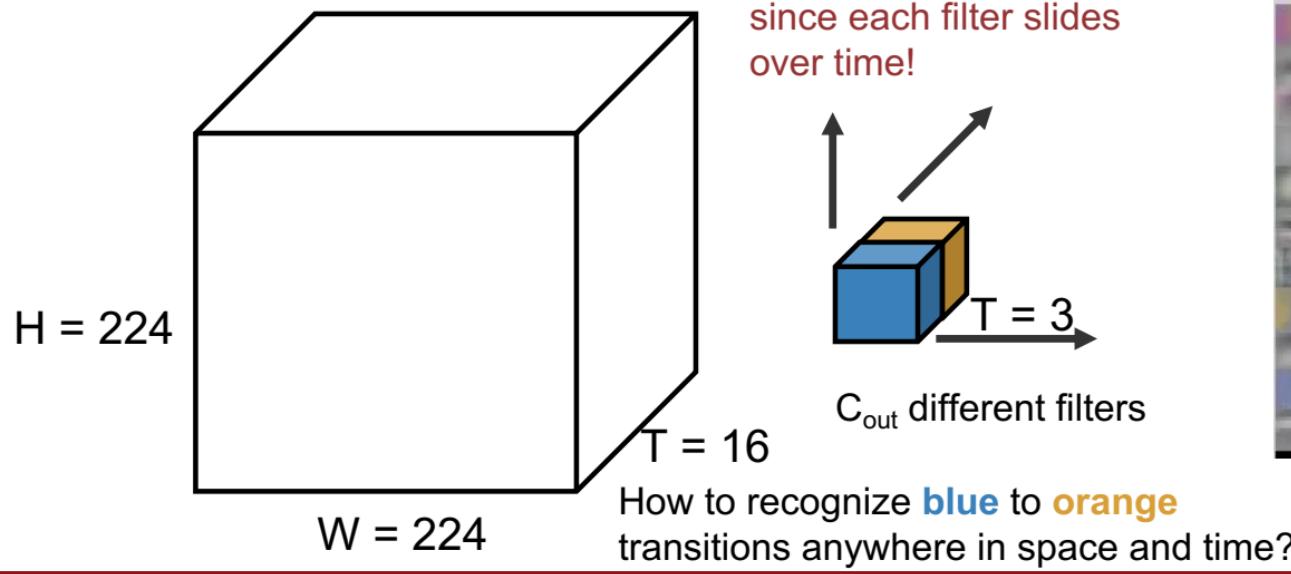
Slide credit: Justin Johnson

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

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

**Weight:**  
 $C_{out} \times C_{in} \times 3 \times 3 \times 3$   
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!



Slide credit: Justin Johnson

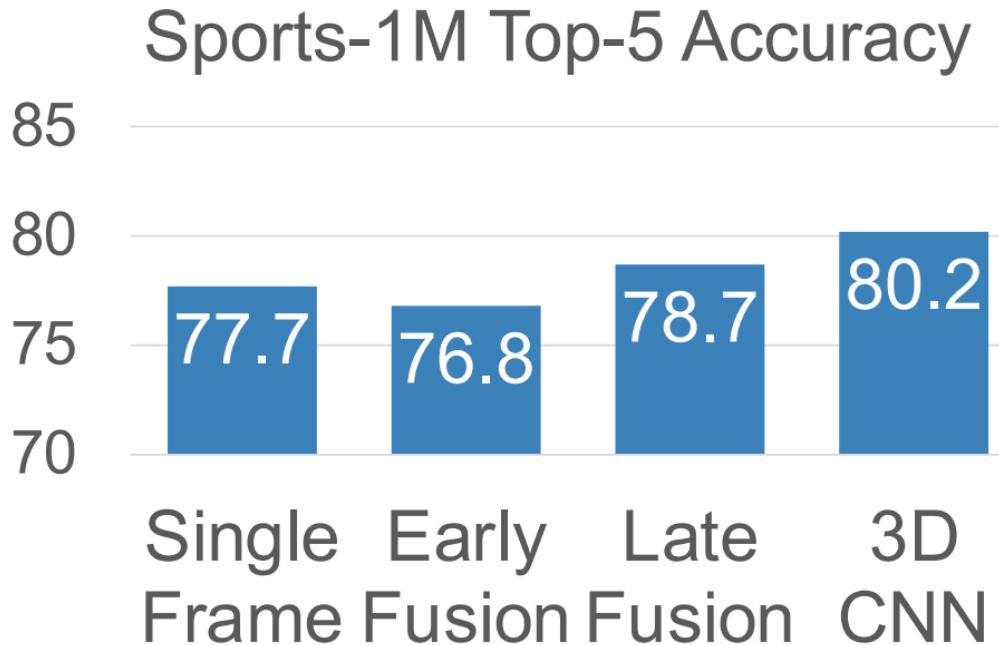
# Example Video Dataset: Sports-1M



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

**Ground Truth**  
**Correct prediction**  
**Incorrect prediction**

# 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!

# 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

# 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

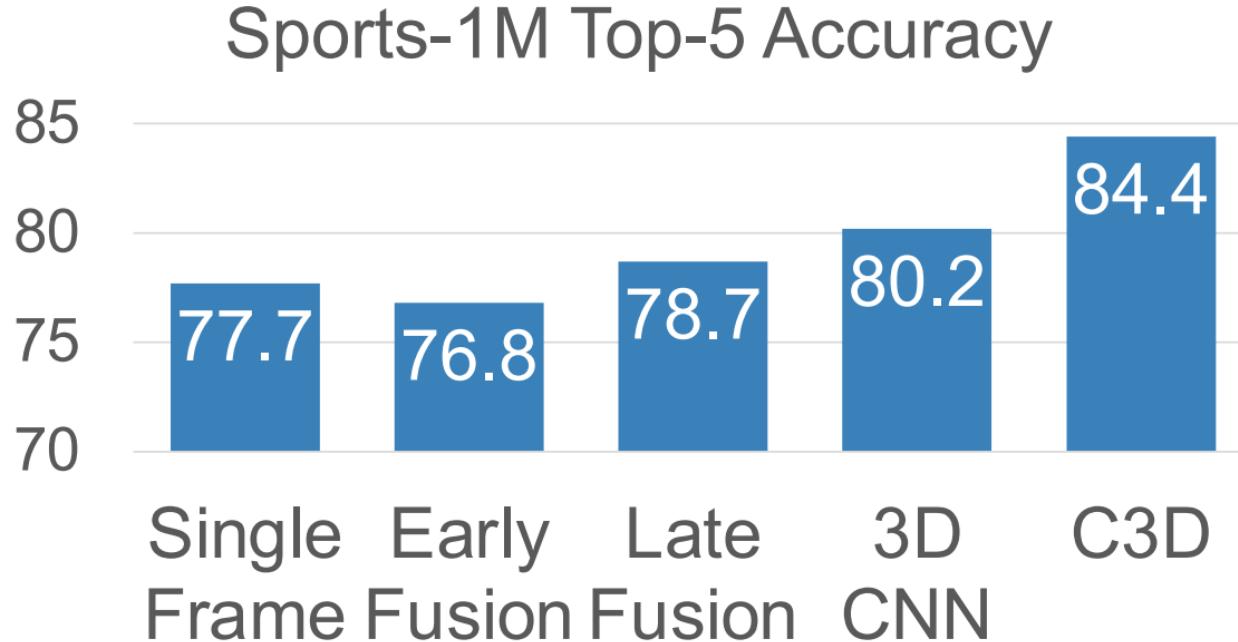
VGG-16: 13.6 GFLOP

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

Slide credit: Justin Johnson

# Early Fusion vs Late Fusion vs 3D CNN

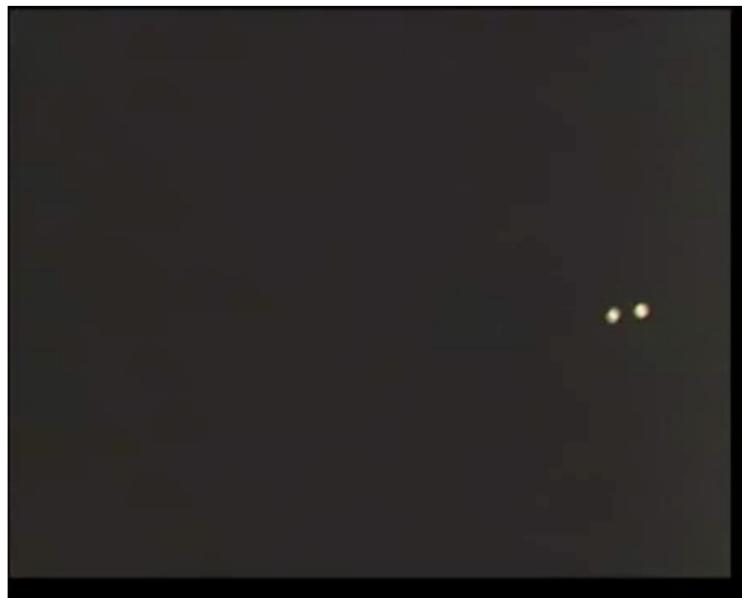


Karpathy et al, "Large-scale Video Classification with Convolutional Neural Networks", CVPR 2014  
Tran et al, "Learning Spatiotemporal Features with 3D Convolutional Networks", ICCV 2015

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

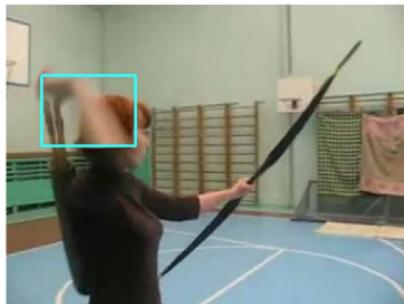
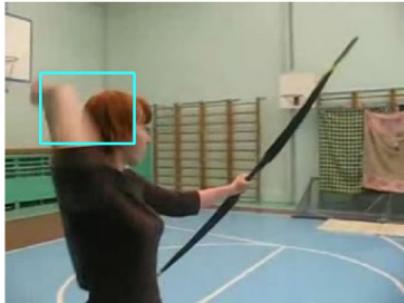


Image at frame t+1

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

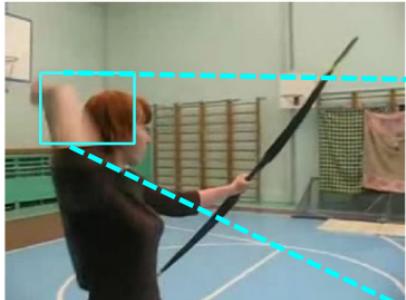
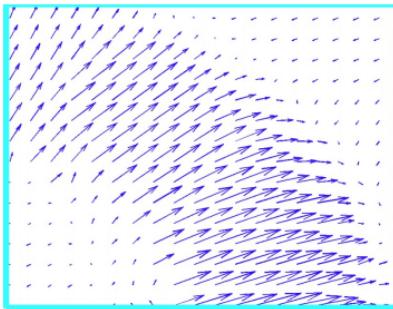


Image at frame t+1

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

# Measuring Motion: Optical Flow

Image at frame t

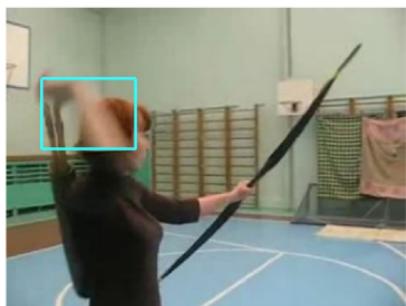
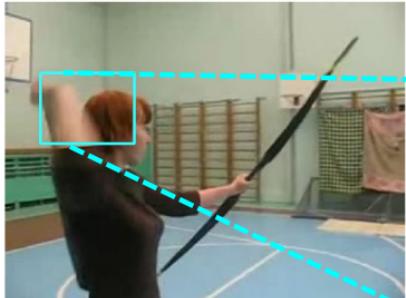
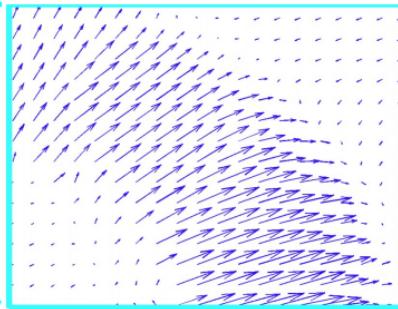


Image at frame t+1

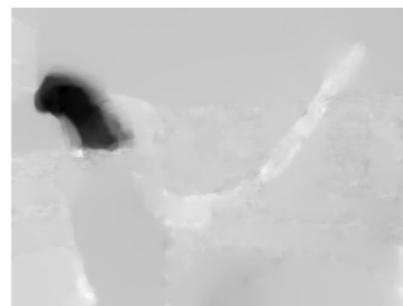
Optical flow gives a displacement field  $F$  between images  $I_t$  and  $I_{t+1}$



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

Optical Flow highlights local motion

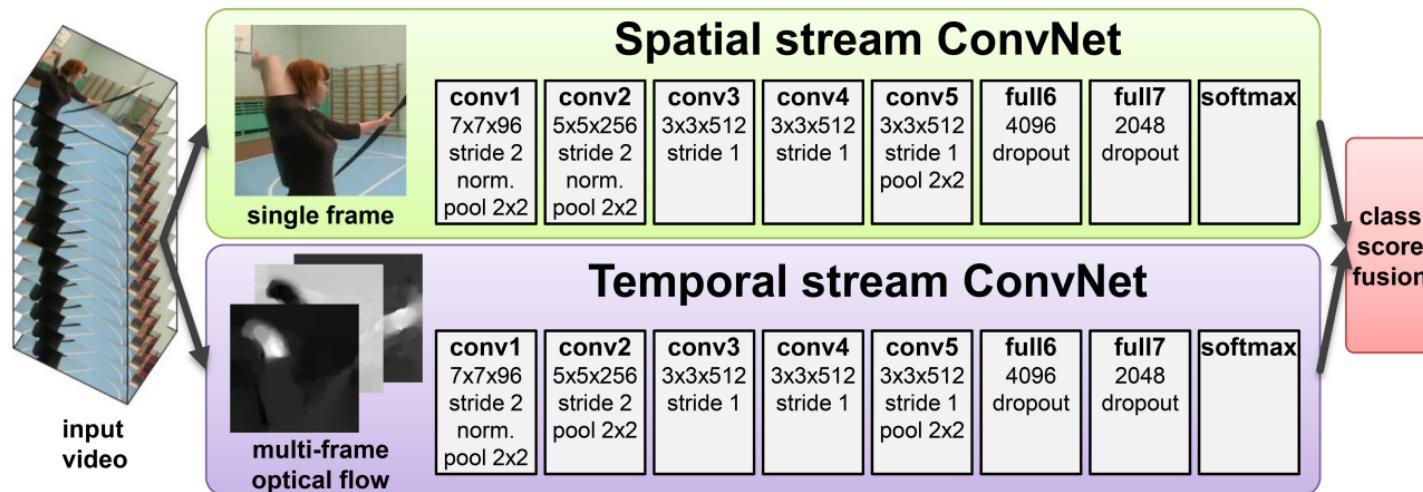
Horizontal flow



Vertical Flow dy

# Separating Motion and Appearance: Two-Stream Networks

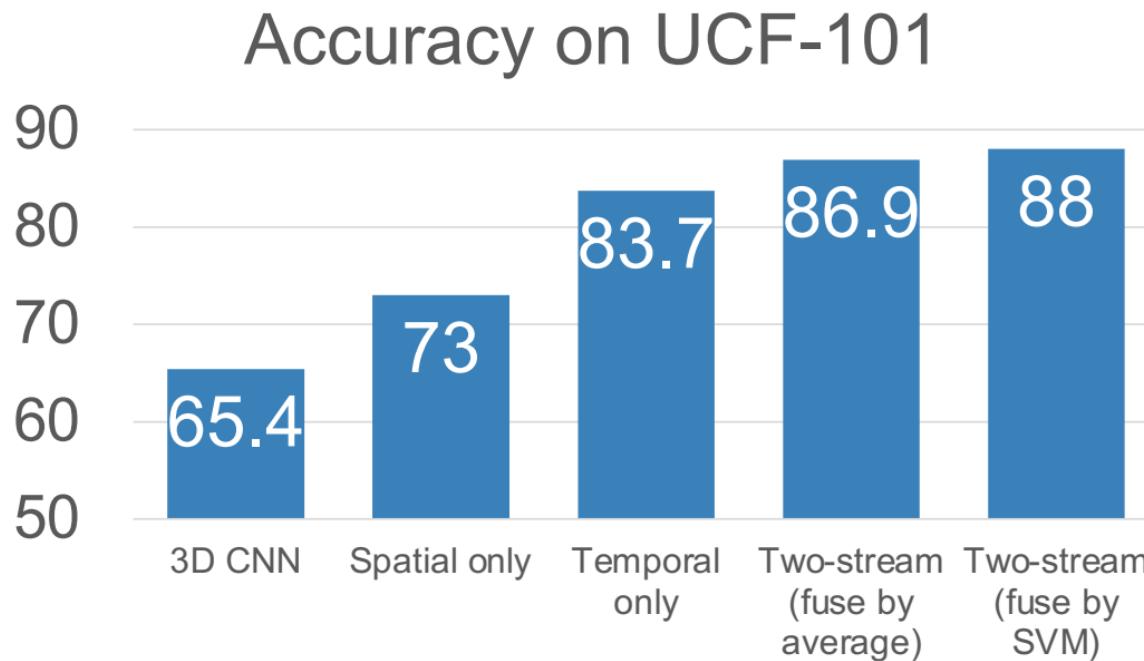
**Input:** Single Image  
 $3 \times H \times W$



**Input:** Stack of optical flow:  
 $[2^*(T-1)] \times H \times W$

**Early fusion:** First 2D conv  
processes all flow images

# Separating Motion and Appearance: Two-Stream Networks

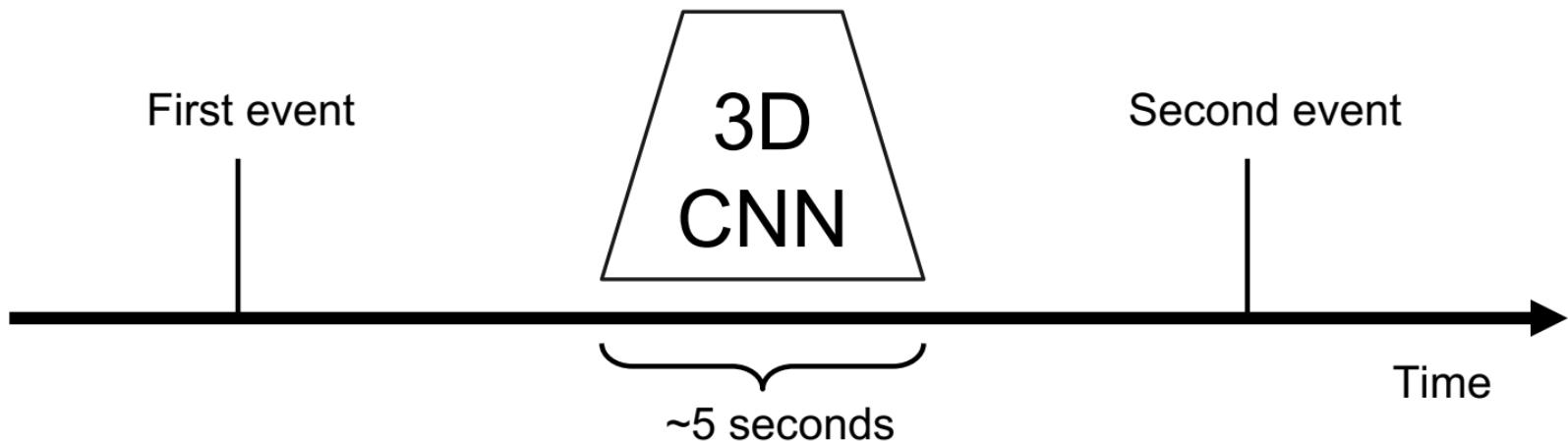


Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014

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?

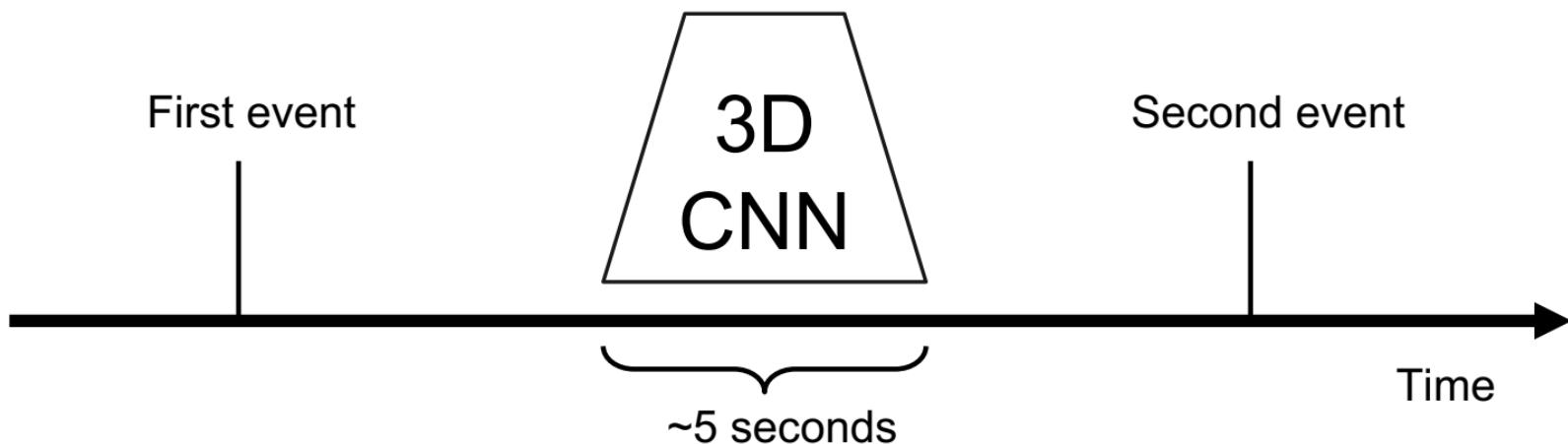


Slide credit: Justin Johnson

# Modeling long-term temporal structure

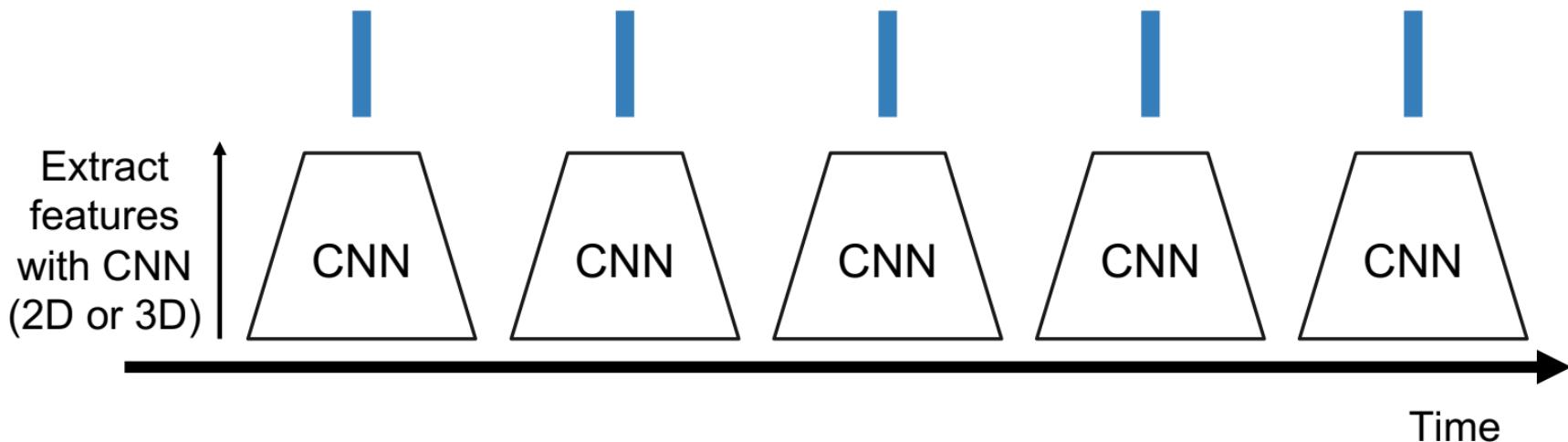
So far all our temporal CNNs only model local motion between frames in very short clips of ~2-5 seconds. What about long-term structure?

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



Slide credit: Justin Johnson

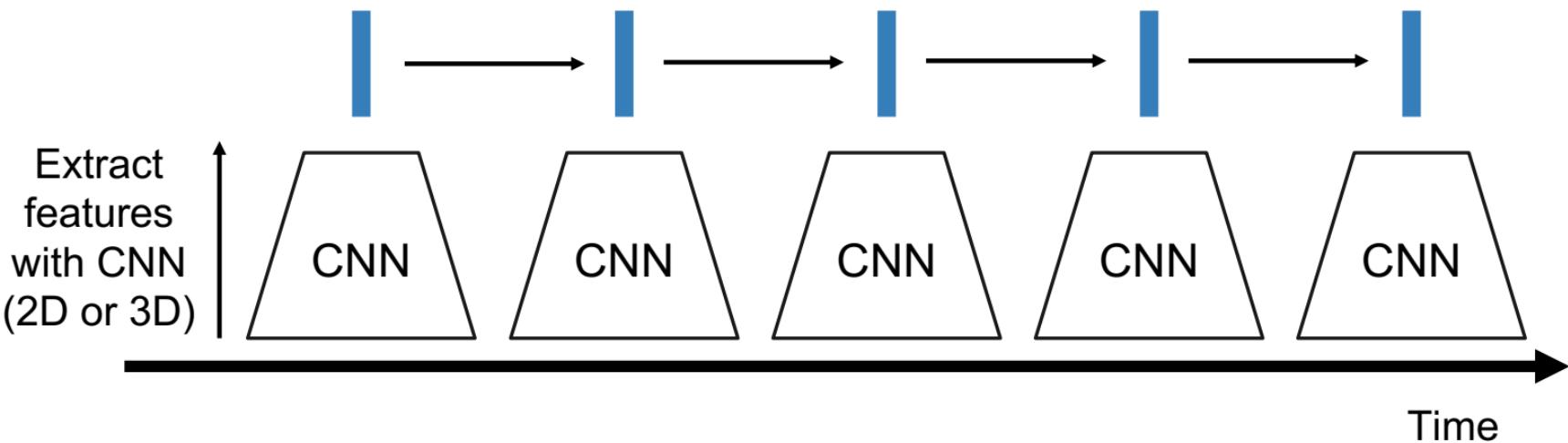
# Modeling long-term temporal structure



Slide credit: Justin Johnson

# Modeling long-term temporal structure

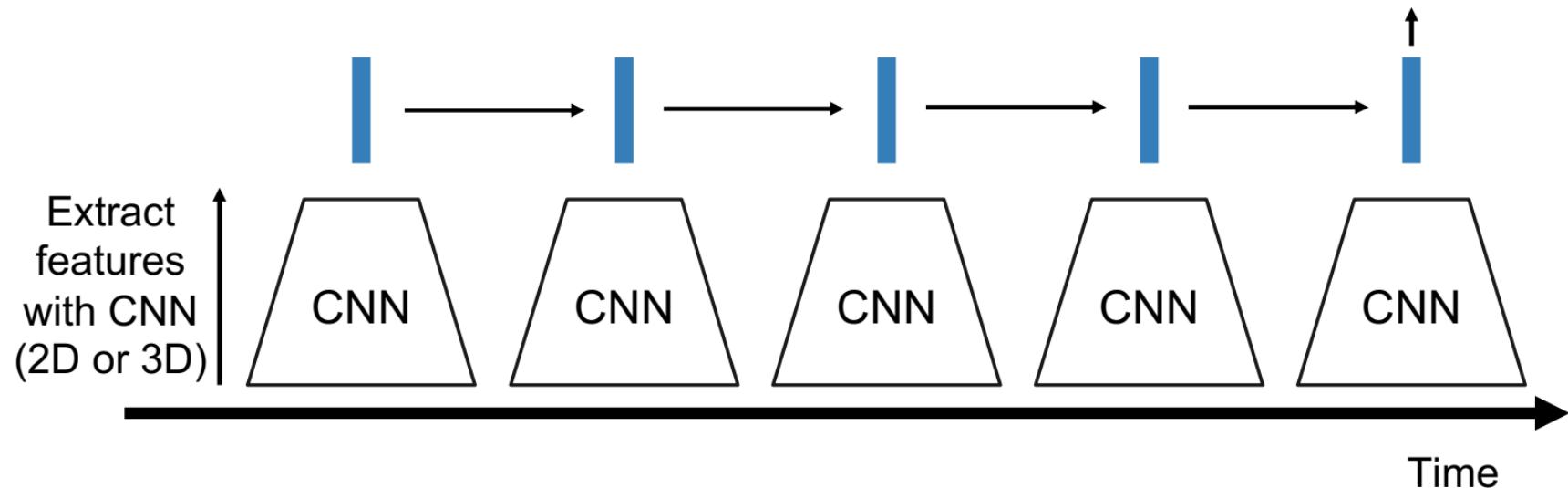
Process local features using recurrent network (e.g. LSTM)



Slide credit: Justin Johnson

# Modeling long-term temporal structure

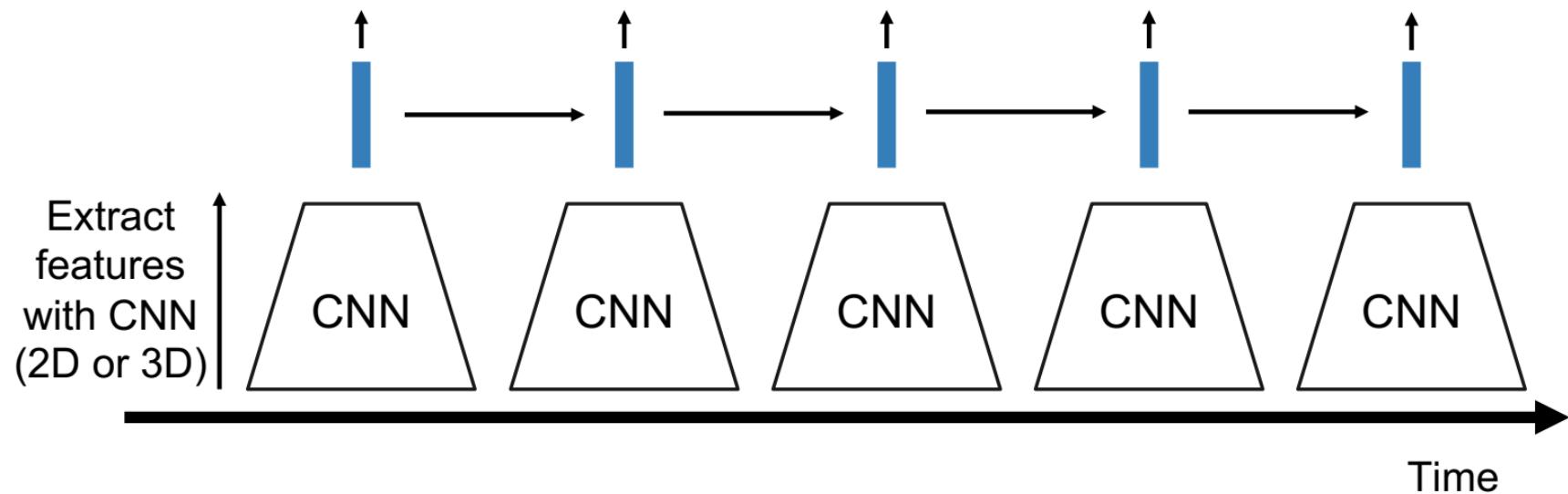
Process local features using recurrent network (e.g. LSTM)  
Many to one: One output at end of video



Slide credit: Justin Johnson

# Modeling long-term temporal structure

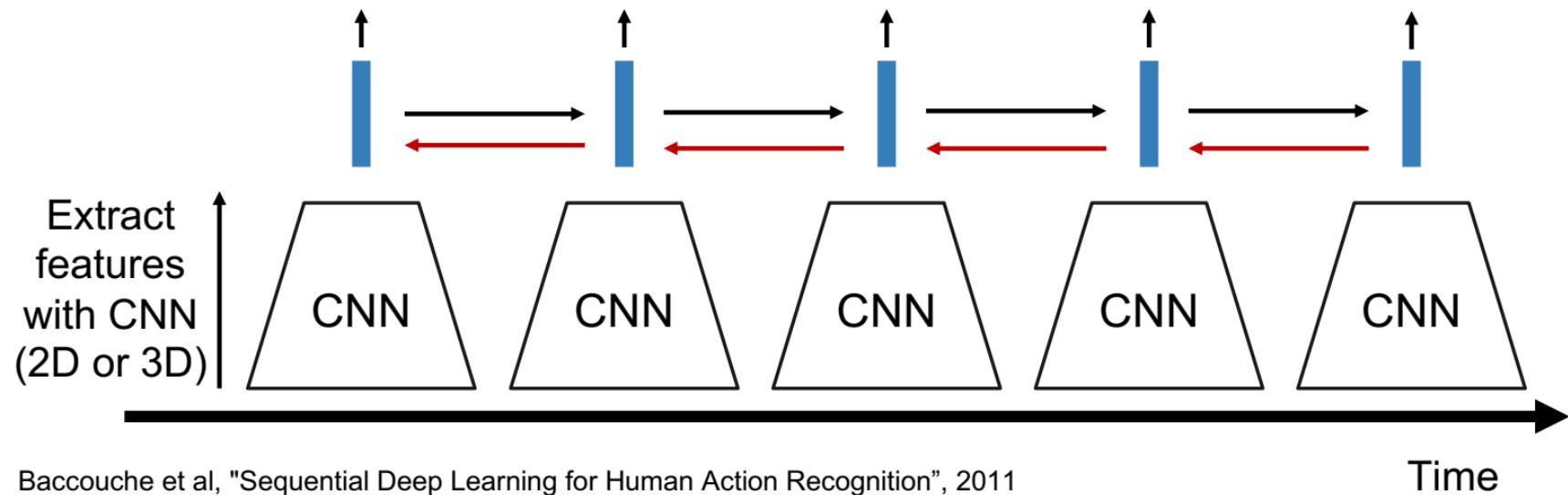
Process local features using recurrent network (e.g. LSTM)  
Many to many: one output per video frame



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

Time

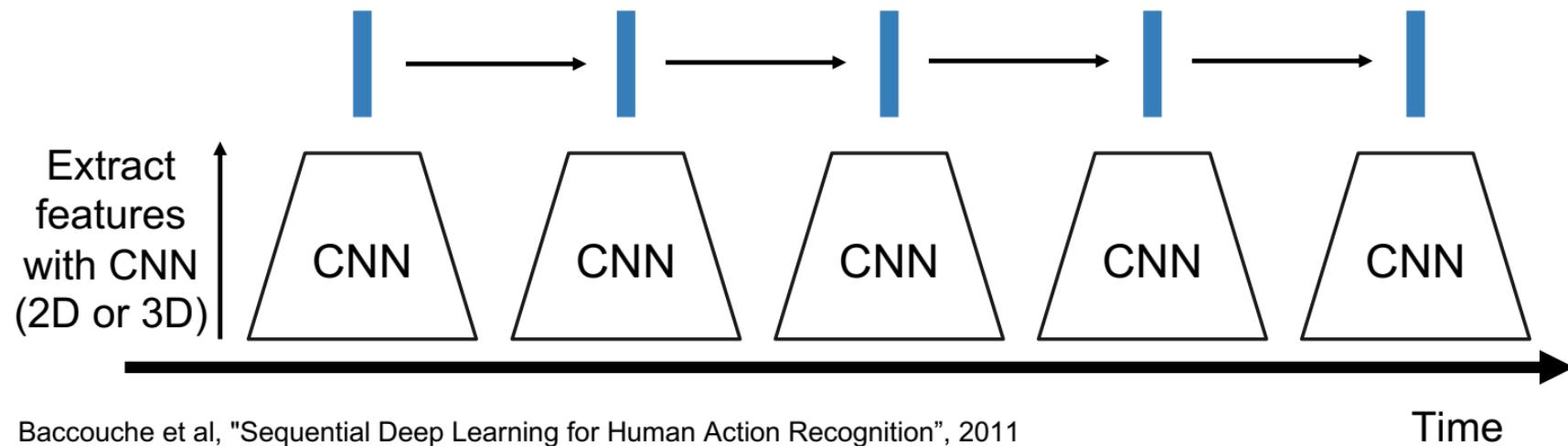
Slide credit: Justin Johnson

# 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?



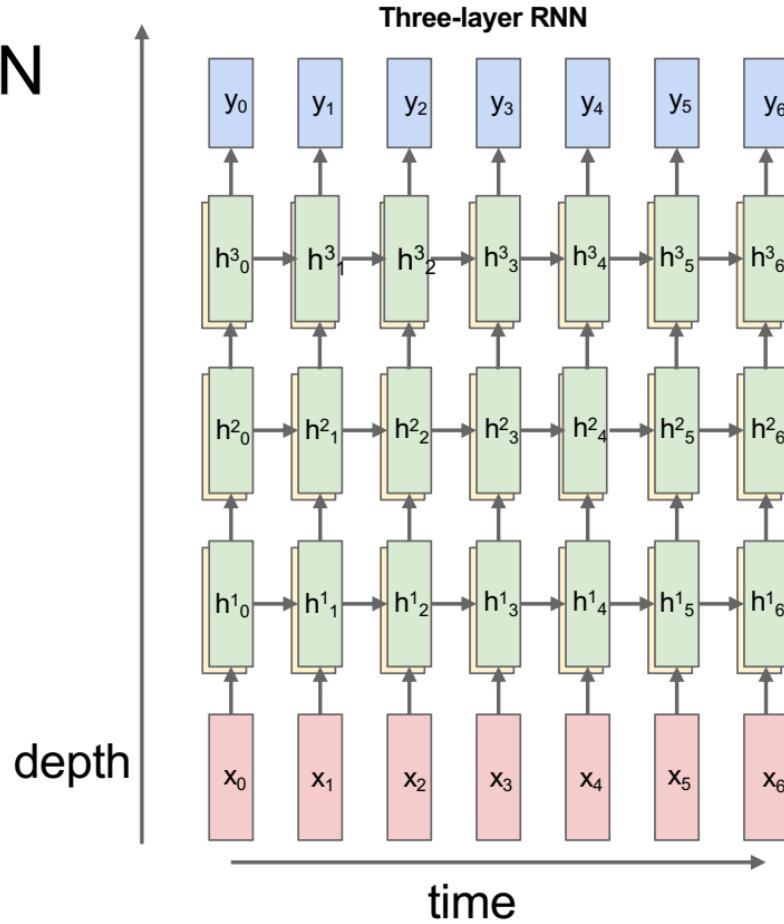
Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011

Donahue et al, "Long-term recurrent convolutional networks for visual recognition and description", CVPR 2015

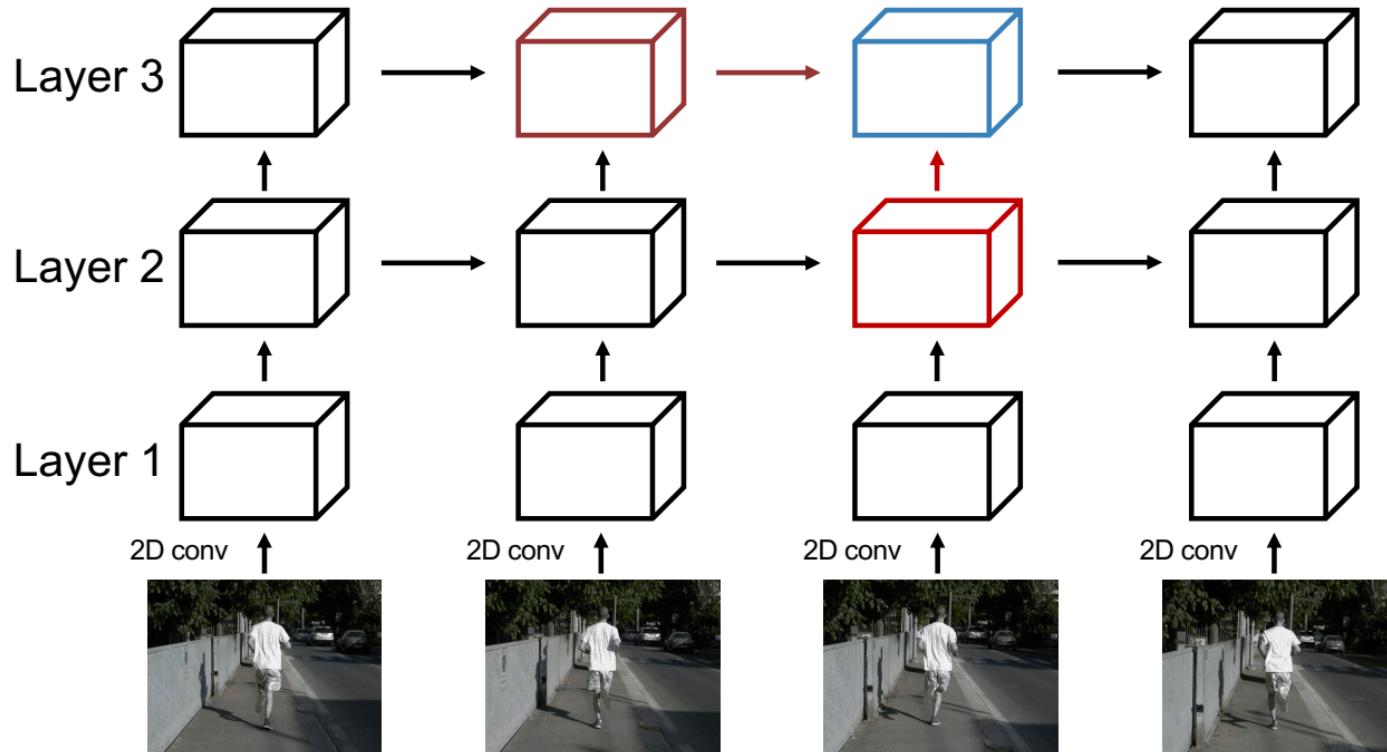
Slide credit: Justin Johnson

# Recall: Multi-layer RNN

We can use a similar structure to process videos!



# Recurrent Convolutional Network



Entire network uses 2D feature maps:  
 $C \times H \times W$

Each depends on two inputs:

1. Same layer, previous timestep
2. Prev layer, same timestep

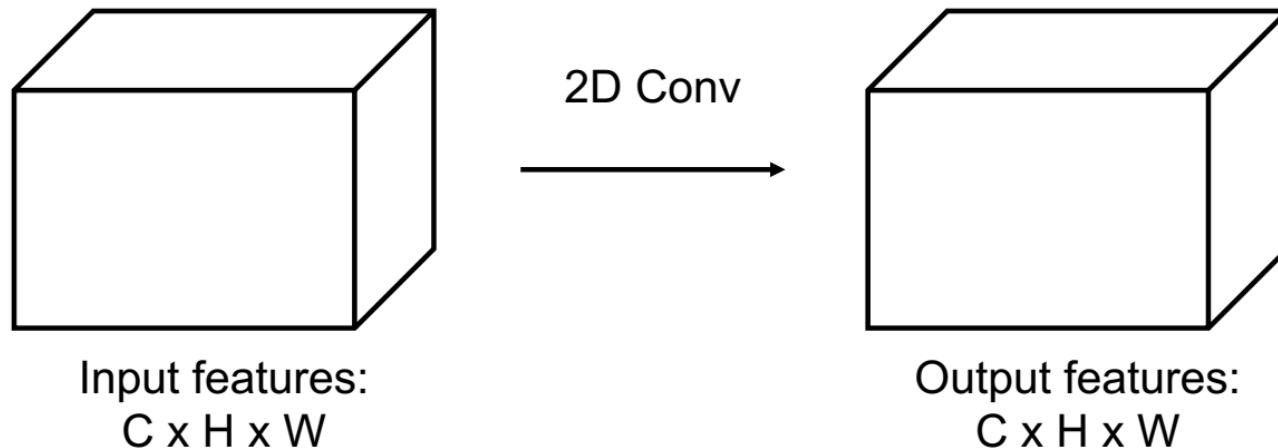
Use different weights at each layer, share weights across time

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

Slide credit: Justin Johnson

# Recurrent Convolutional Network

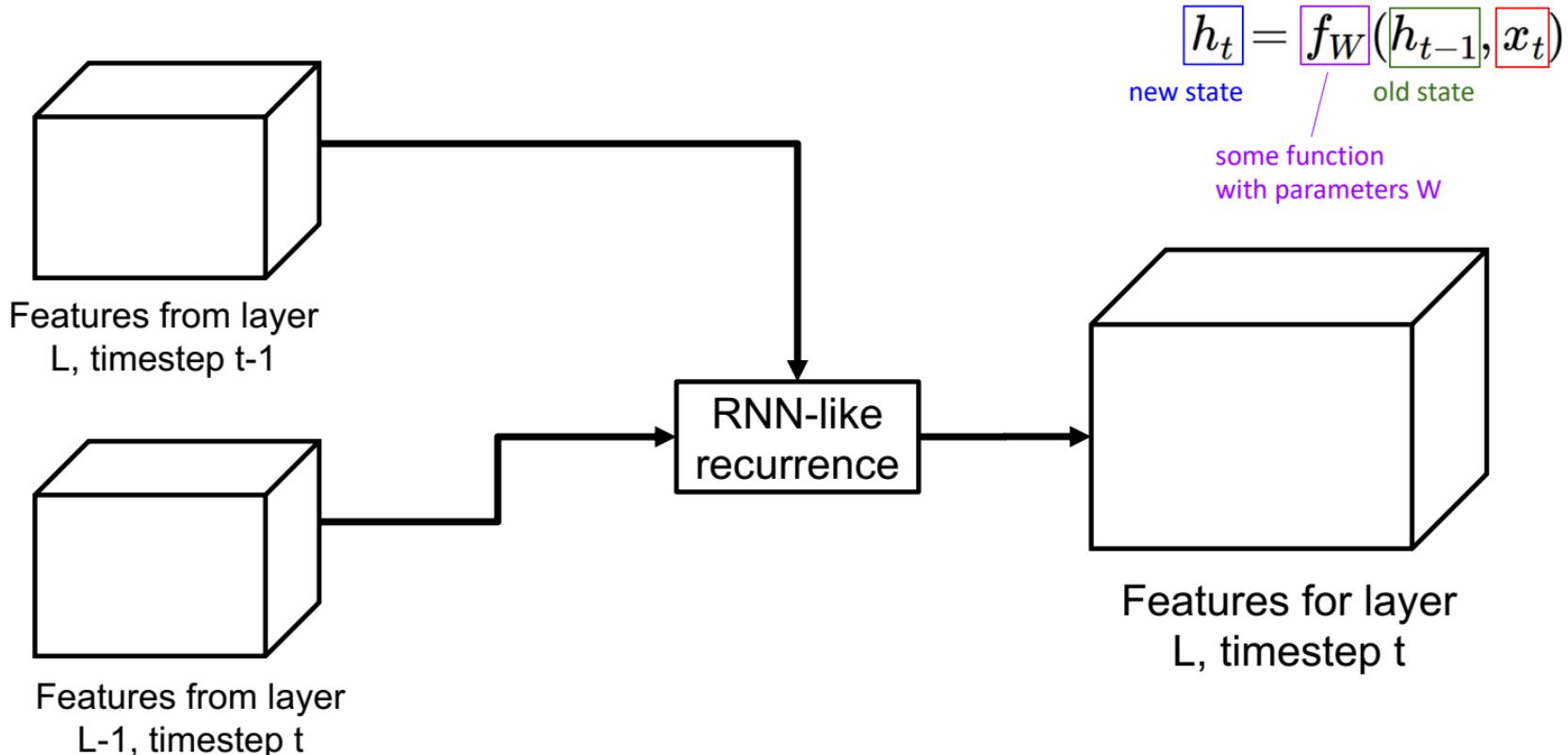
Normal 2D CNN:



Slide credit: Justin Johnson

# Recurrent Convolutional Network

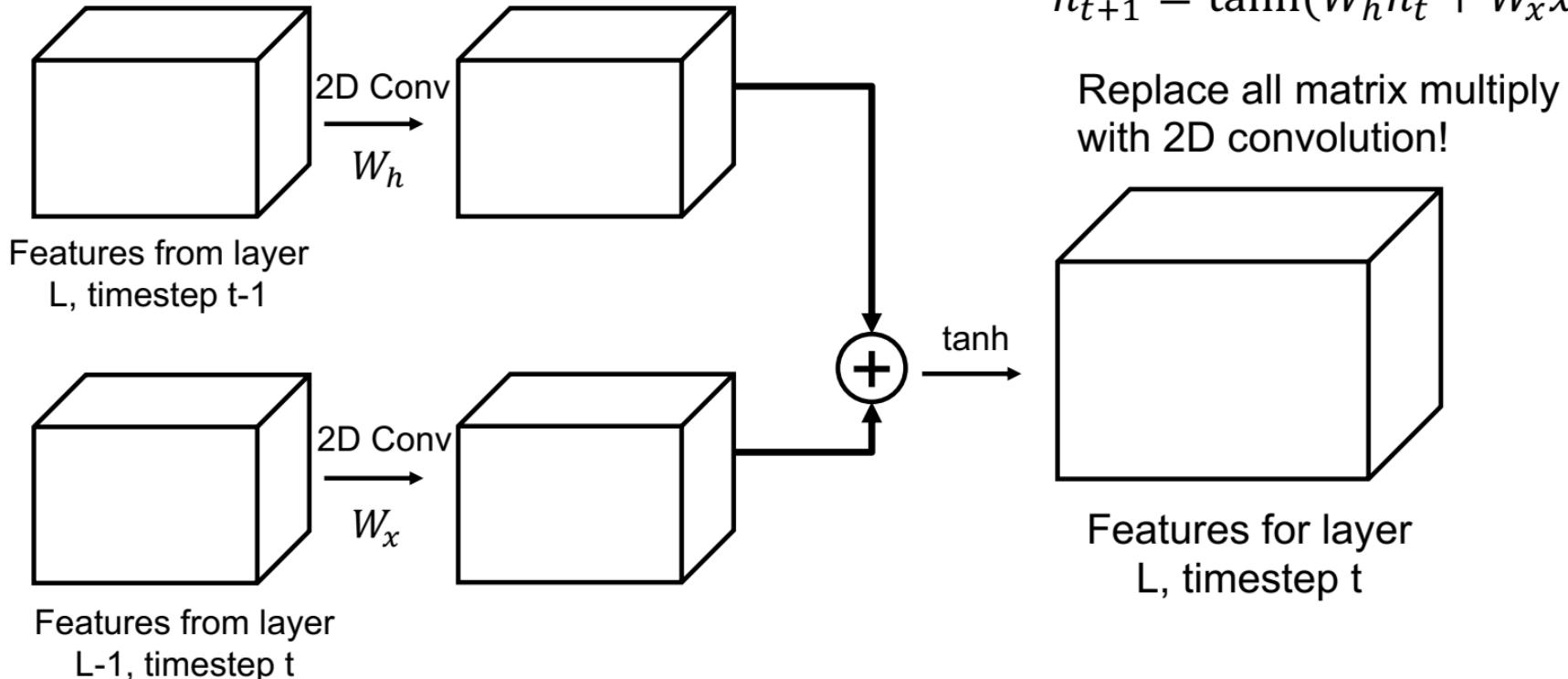
Recall: Recurrent Network



# Recurrent Convolutional Network

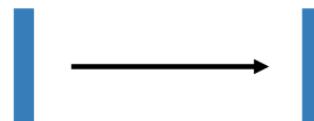
Recall: Vanilla RNN

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

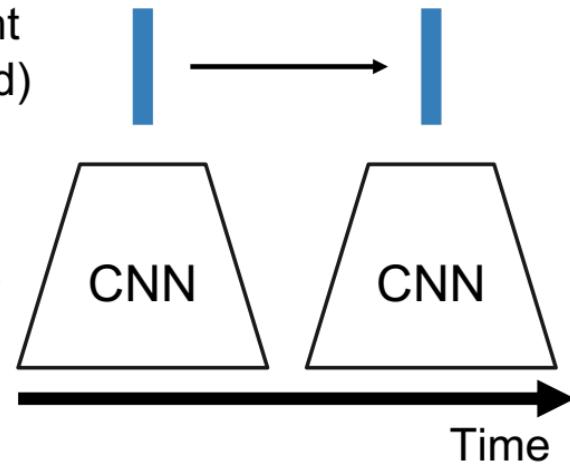


# Modeling long-term temporal structure

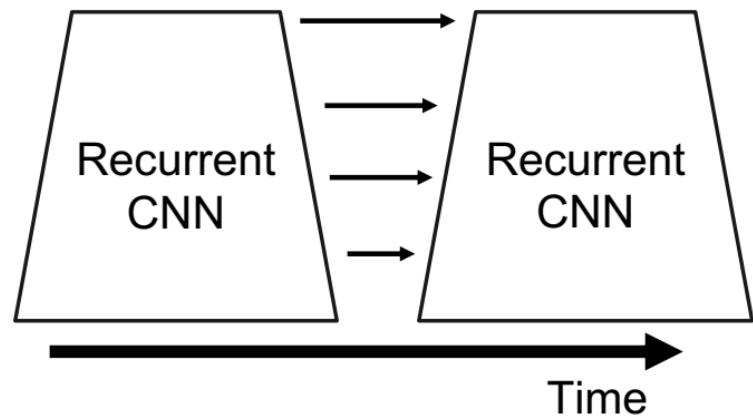
RNN: Infinite  
temporal extent  
(fully-connected)



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

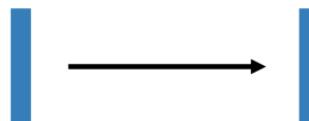
Ballas et al, "Delving Deeper into Convolutional Networks for Learning Video Representations", ICLR 2016

Slide credit: Justin Johnson

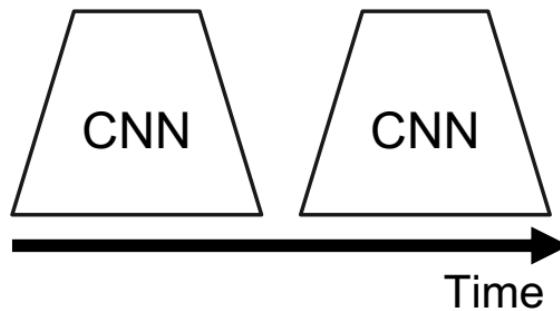
# Modeling long-term temporal structure

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

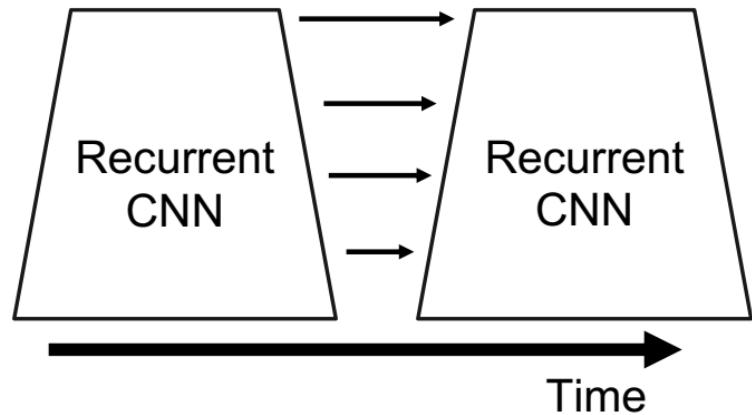
RNN: Infinite temporal extent (fully-connected)



CNN: finite temporal extent (convolutional)



Recurrent CNN: Infinite temporal extent (convolutional)



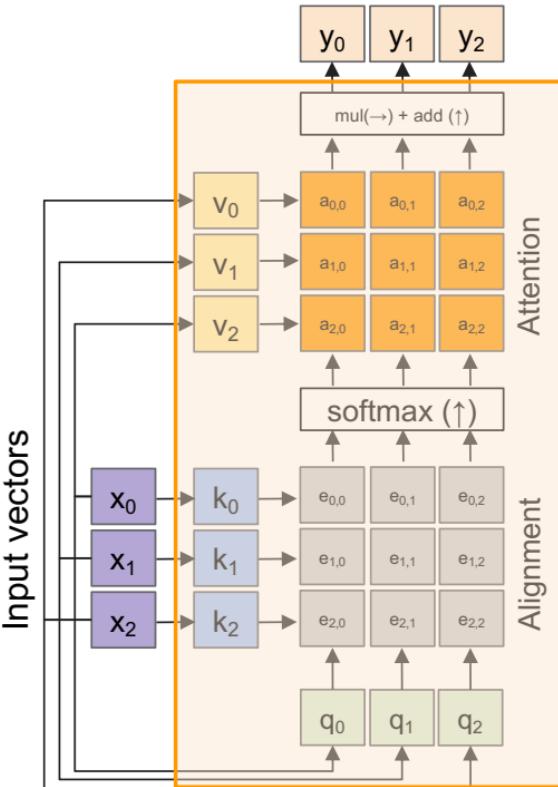
Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011

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:  $\mathbf{y}$  (shape:  $D_v$ )

**Operations:**

Key vectors:  $\mathbf{k} = \mathbf{x}\mathbf{W}_k$

Value vectors:  $\mathbf{v} = \mathbf{x}\mathbf{W}_v$

Query vectors:  $\mathbf{q} = \mathbf{x}\mathbf{W}_q$

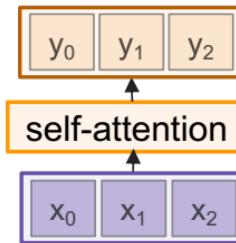
Alignment:  $\mathbf{e}_{i,j} = \mathbf{q}_j \cdot \mathbf{k}_i / \sqrt{D}$

Attention:  $\mathbf{a} = \text{softmax}(\mathbf{e})$

Output:  $\mathbf{y}_j = \sum_i \mathbf{a}_{i,j} \mathbf{v}_i$

**Inputs:**

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



# Spatio-Temporal Self-Attention (Nonlocal Block)

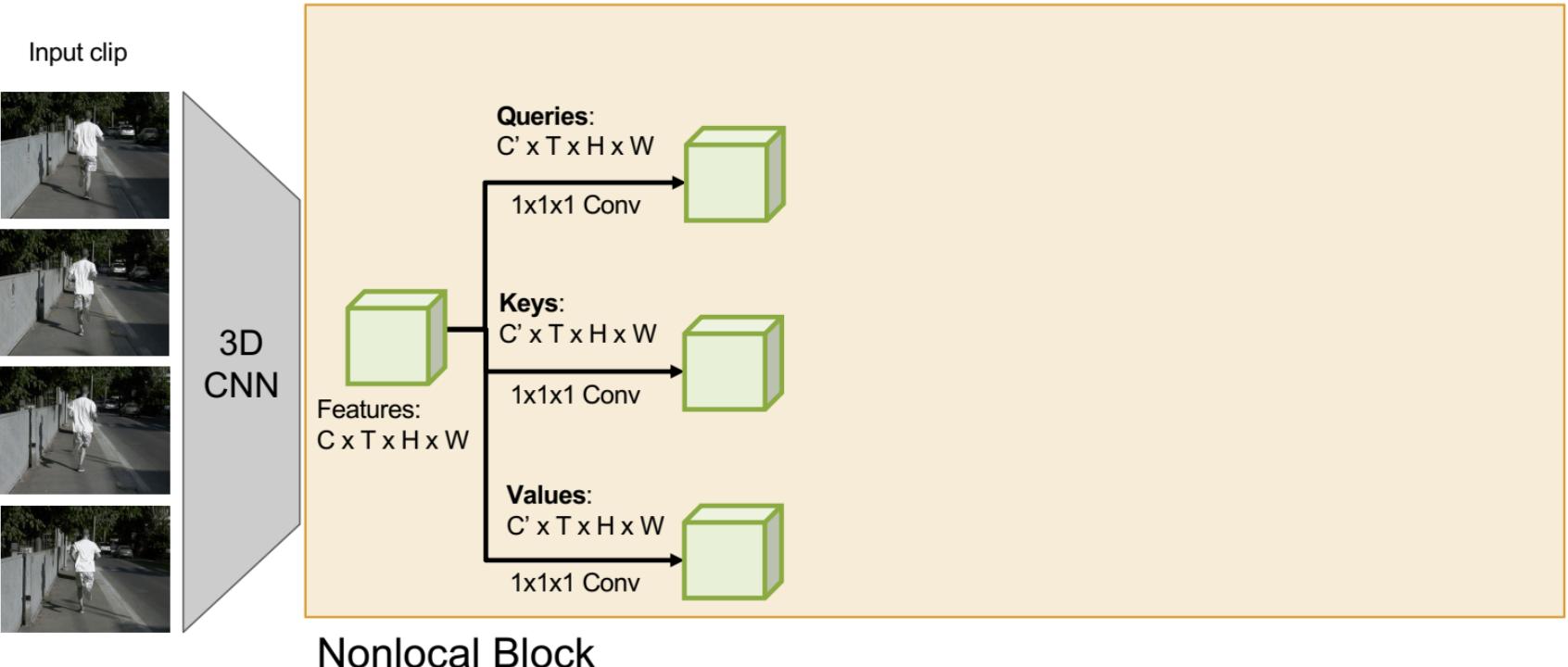
Input clip



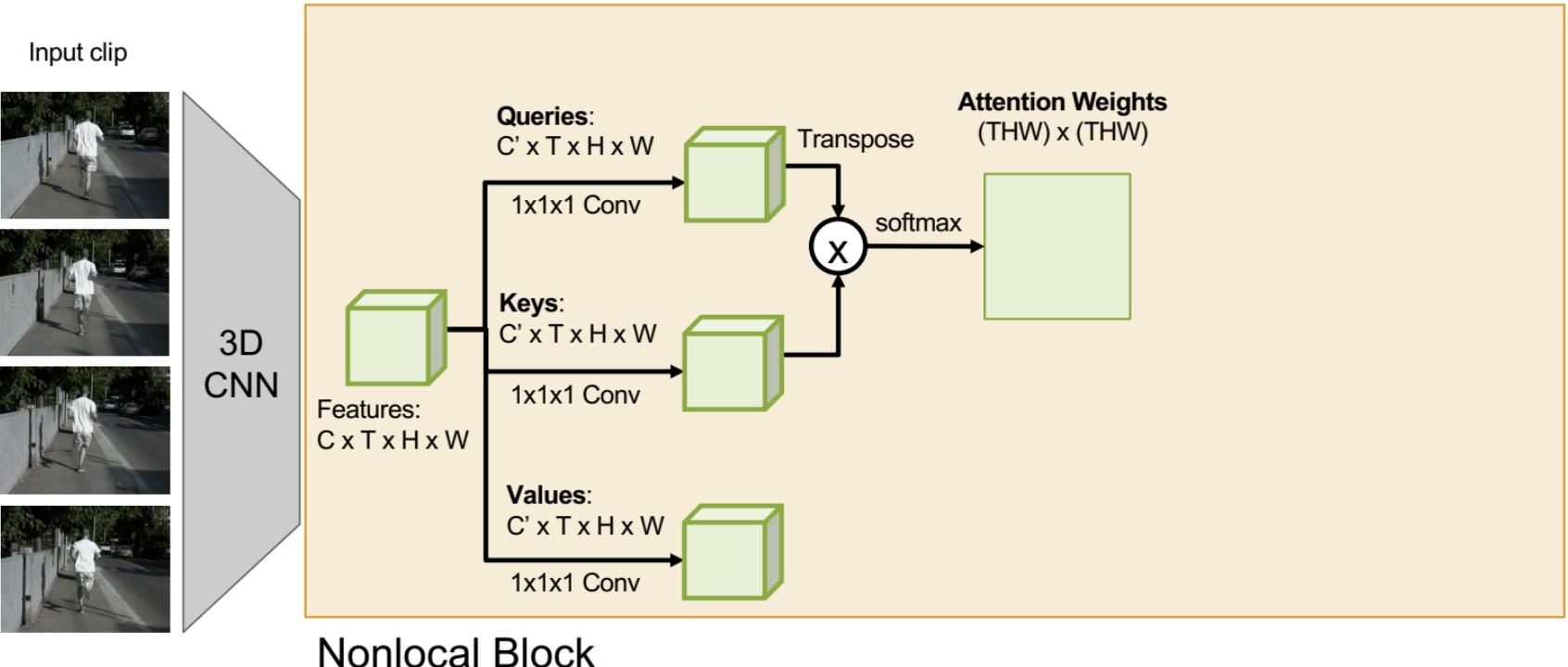
Features:  
 $C \times T \times H \times W$

Nonlocal Block

# Spatio-Temporal Self-Attention (Nonlocal Block)

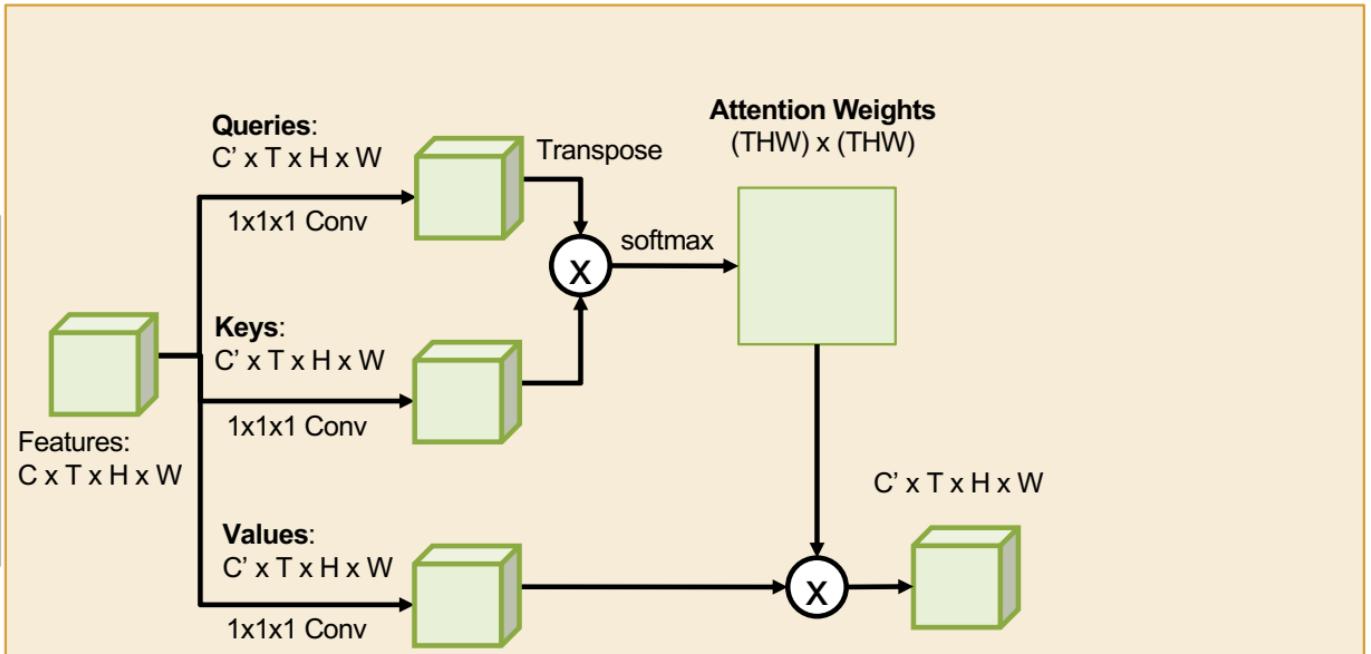


# Spatio-Temporal Self-Attention (Nonlocal Block)



# Spatio-Temporal Self-Attention (Nonlocal Block)

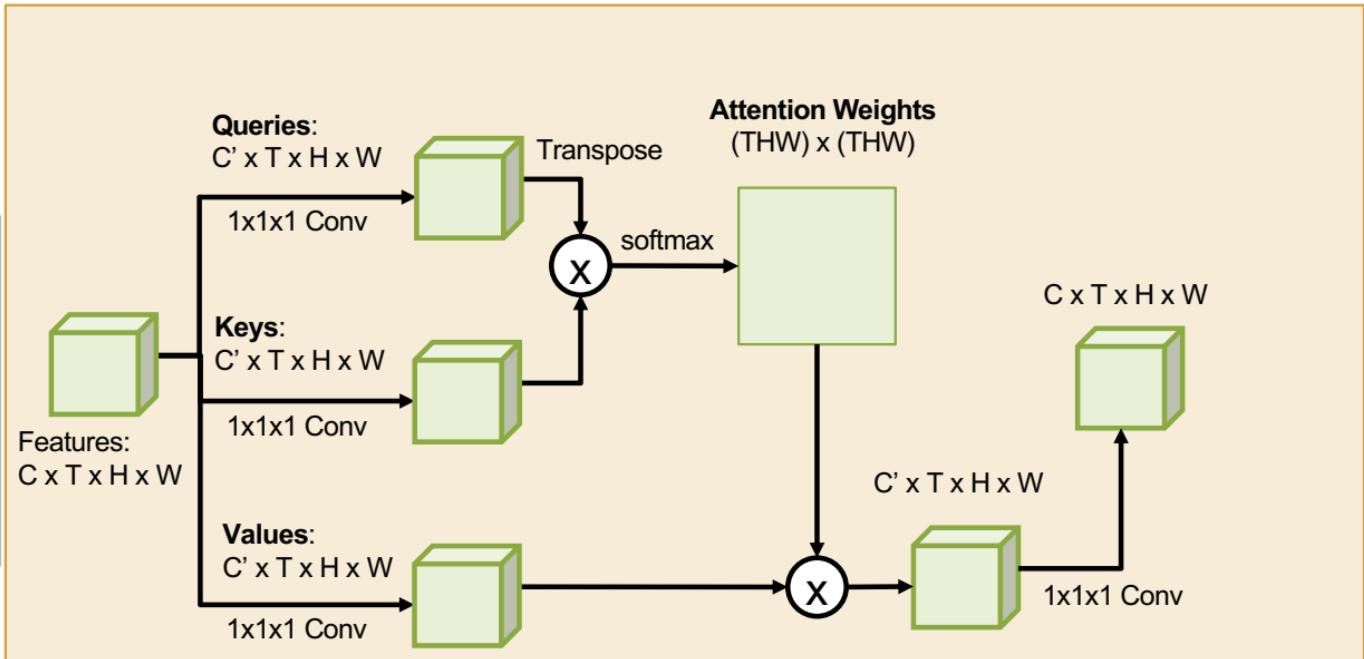
Input clip



Nonlocal Block

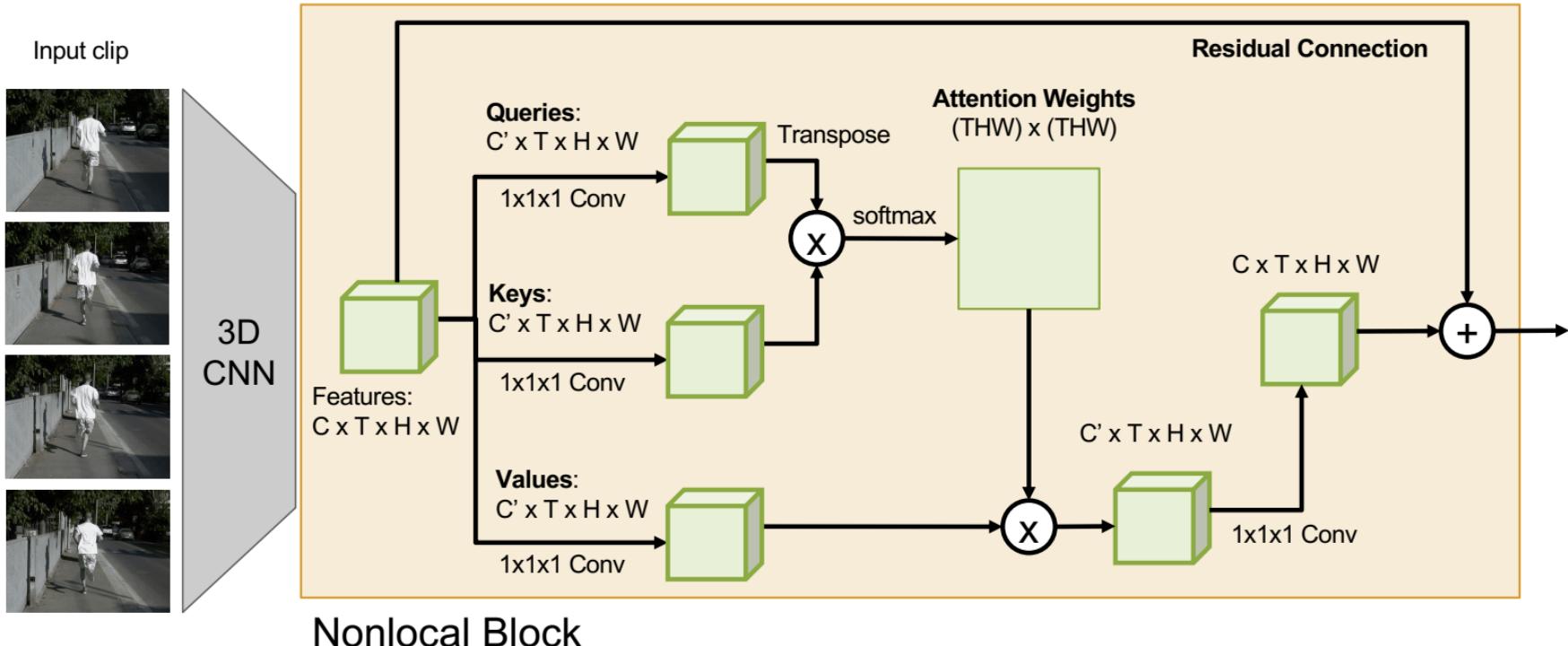
# Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip



Nonlocal Block

# Spatio-Temporal Self-Attention (Nonlocal Block)



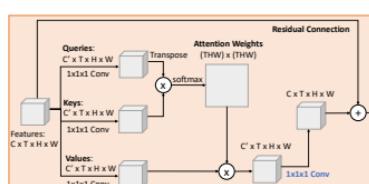
# Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip



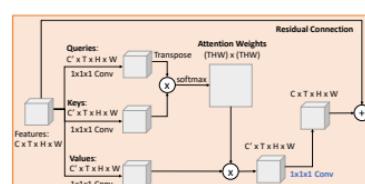
3D CNN

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



Nonlocal Block

3D CNN



Nonlocal Block

3D CNN

Running

# 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.

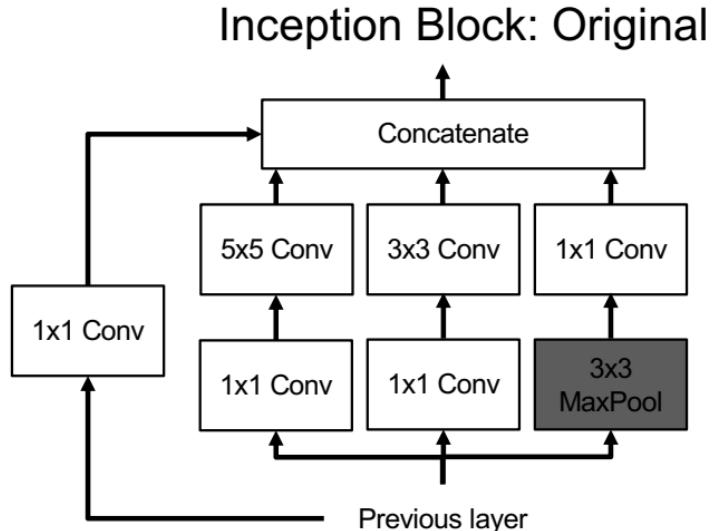
Replace each 2D  $K_h \times K_w$  conv/pool layer with a 3D  $K_t \times K_h \times K_w$  version

# 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

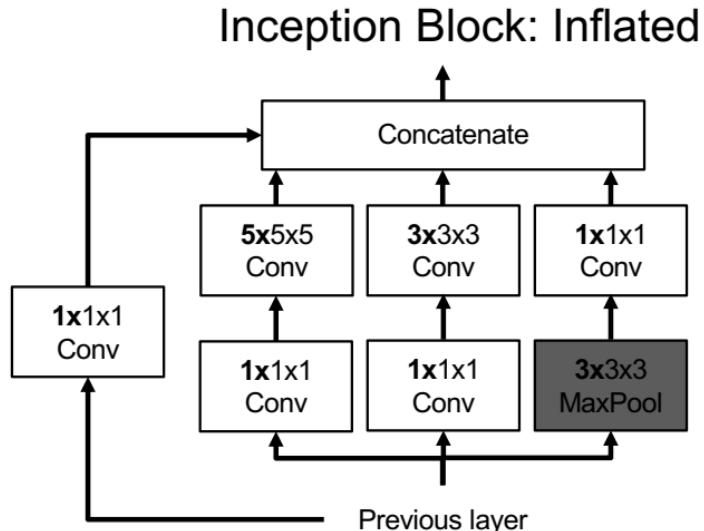


# Inflating 2D Networks to 3D (I3D)

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



# Inflating 2D Networks to 3D (I3D)

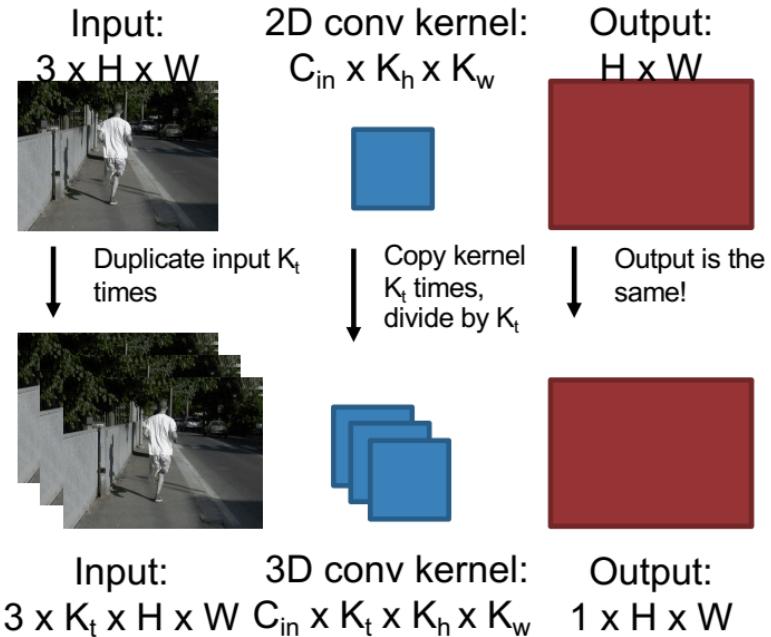
There has been a lot of work on architectures for images. Can we reuse image architectures for video?

**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



# Inflating 2D Networks to 3D (I3D)

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

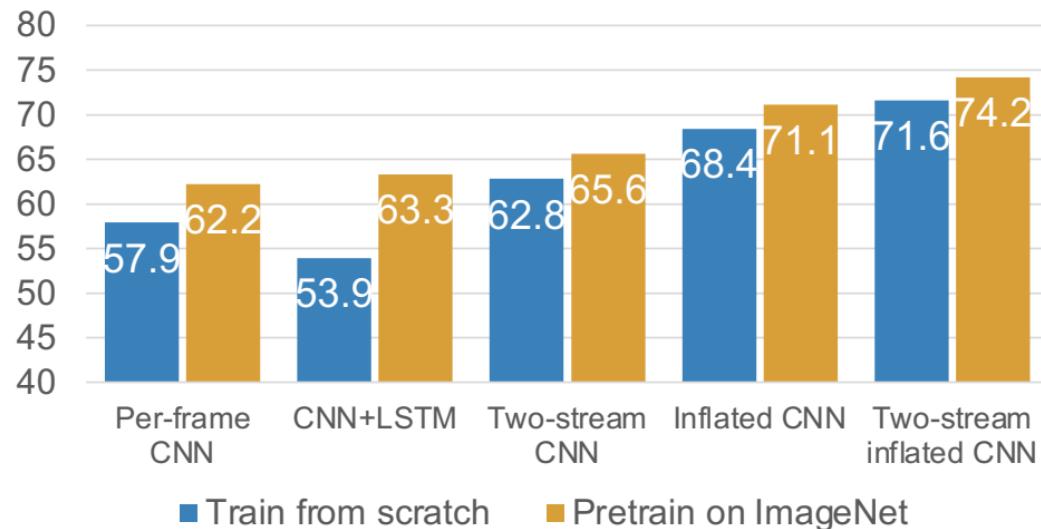
Idea: take a 2D CNN architecture.

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  $K_t$

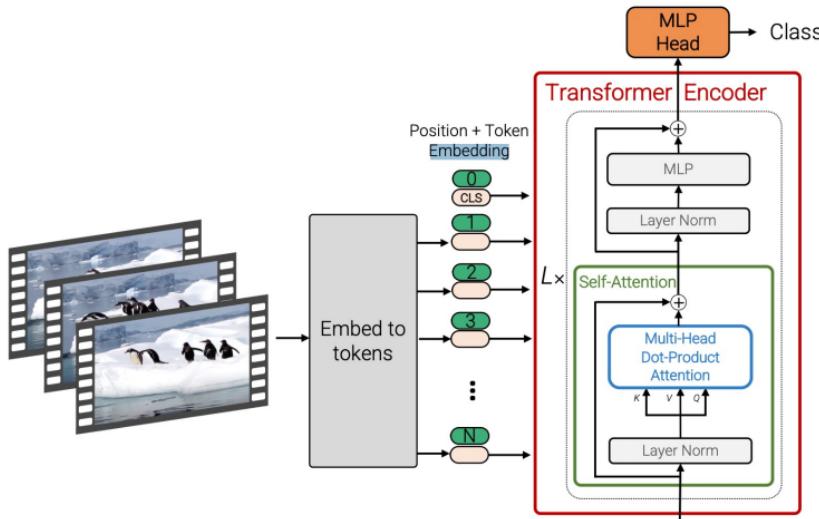
This gives the same result as 2D conv given “constant” video input

Top-1 Accuracy on Kinetics-400

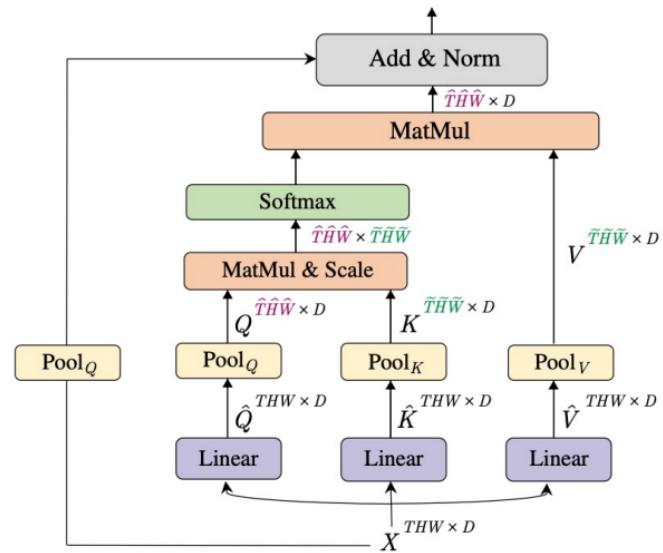


# Vision Transformers for Video

Factorized attention: Attend over space / time



Pooling module: Reduce number of tokens



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

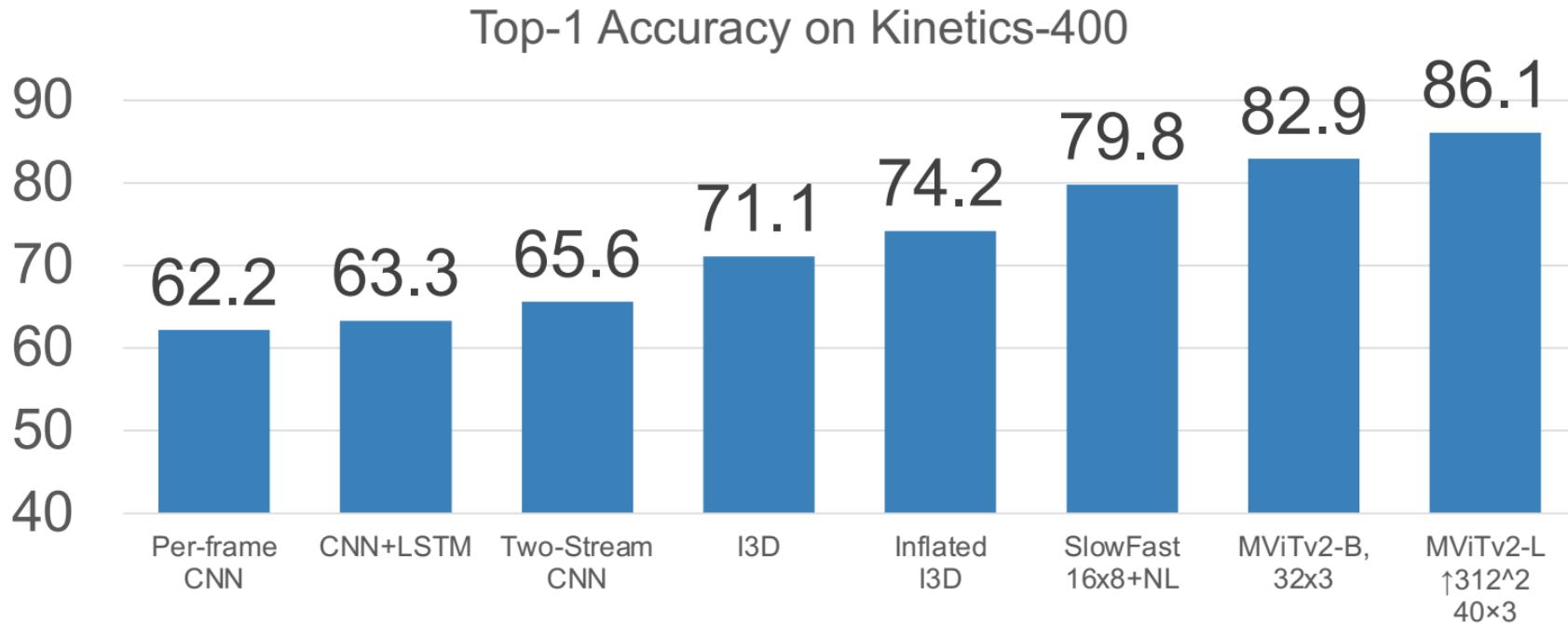
Neimark et al, "Video Transformer Network", ICCV 2021

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

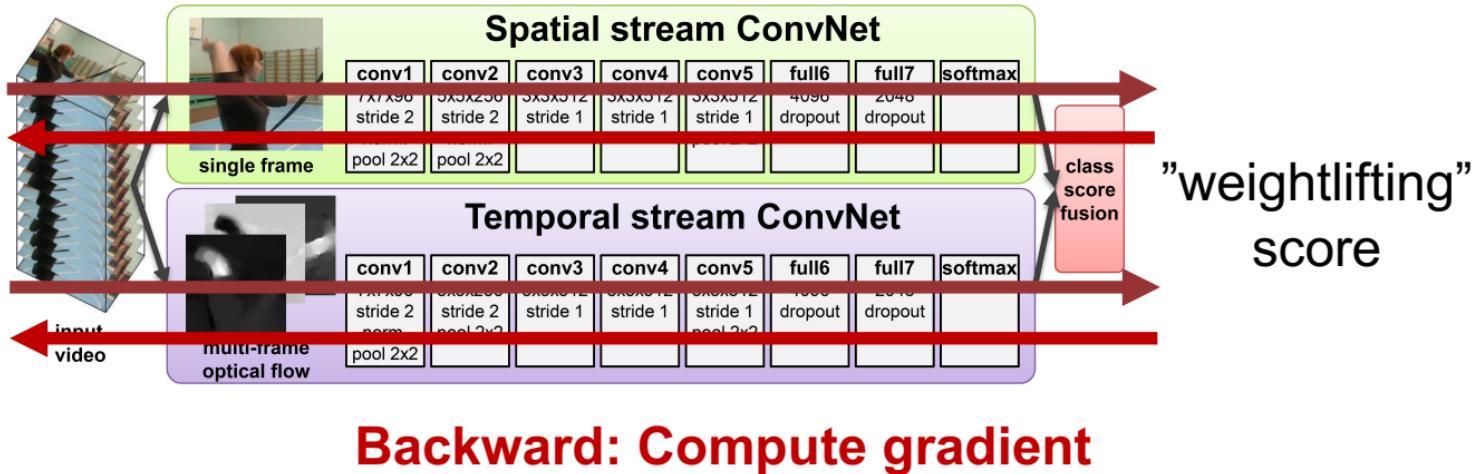
Slide credit: Justin Johnson

# Visualizing Video Models

Image



## Forward: Compute class score



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

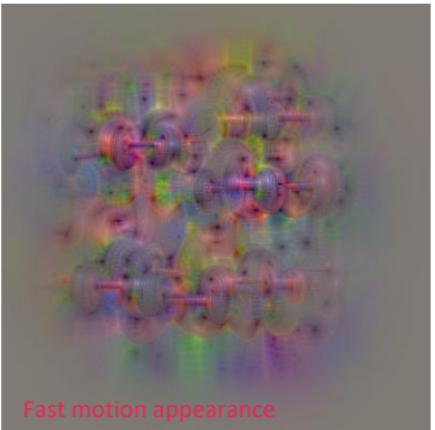
Feichtenhofer et al, “What have we learned from deep representations for action recognition?”, CVPR 2018  
Feichtenhofer et al, “Deep insights into convolutional networks for video recognition?”, IJCV 2019.

Slide credit: Justin Johnson

# Can you guess the action?

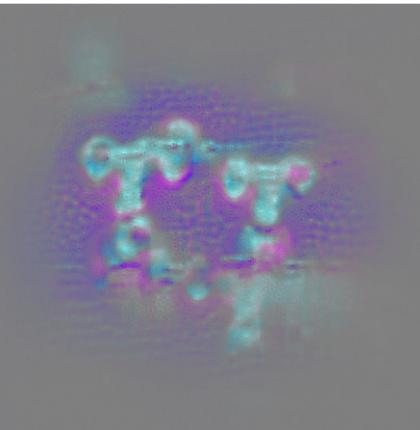
Feichtenhofer et al, "What have we learned from deep representations for action recognition?", CVPR 2018  
Feichtenhofer et al, "Deep insights into convolutional networks for video recognition?", IJCV 2019.  
Slide credit: Christoph Feichtenhofers

Appearance

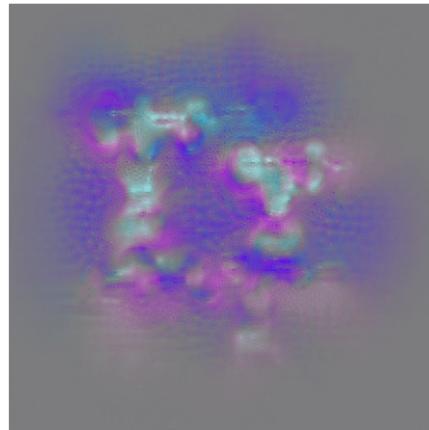


Fast motion appearance

"Slow" motion



"Fast" motion

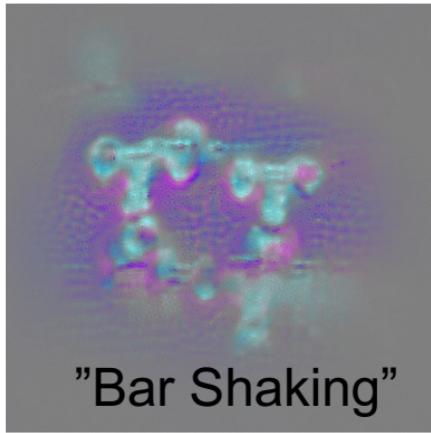


# Can you guess the action? Weightlifting

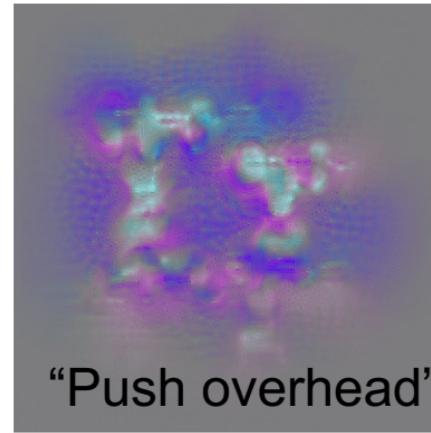
Appearance



“Slow” motion

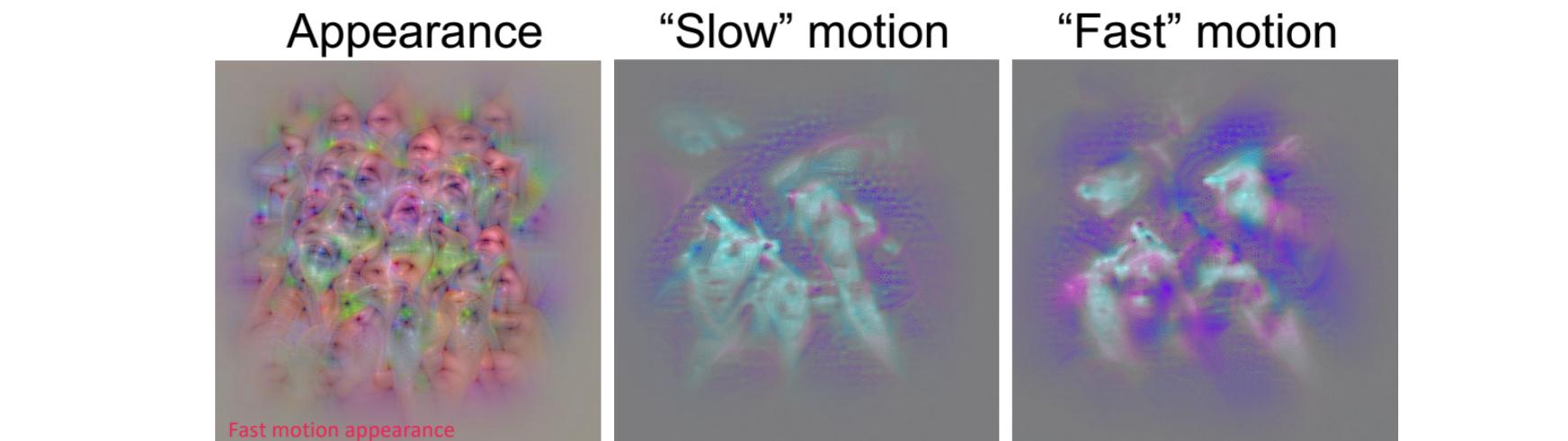


“Fast” motion



Slide credit: Justin Johnson

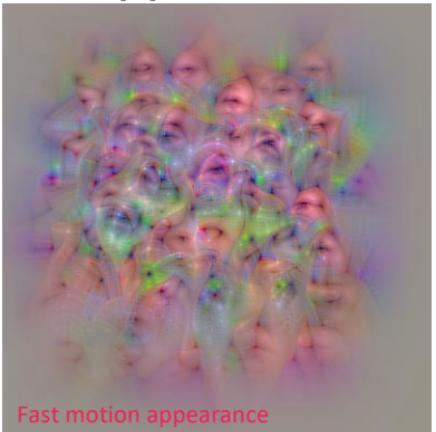
# Can you guess the action?



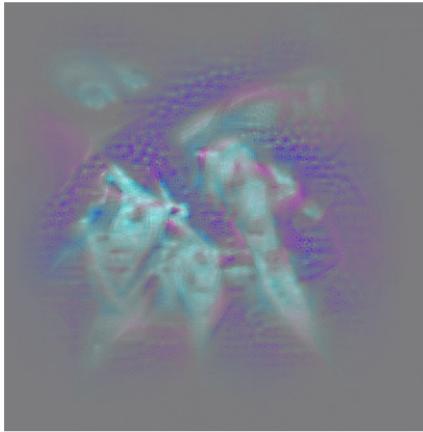
Slide credit: Justin Johnson

# Can you guess the action? Apply Eye Makeup

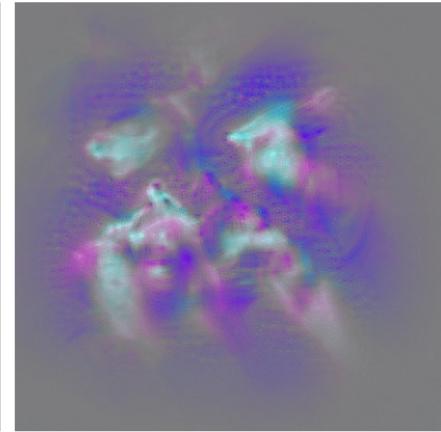
Appearance



“Slow” motion



“Fast” motion



Slide credit: Justin Johnson

# So far: Classify short clips



Videos: Recognize **actions**



Swimming  
**Running**  
Jumping  
Eating  
Standing

# Temporal Action Localization

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



Running

Jumping

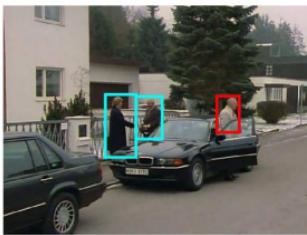
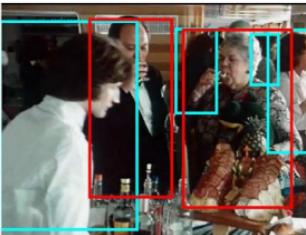
Can use architecture similar to Faster R-CNN:  
first generate **temporal proposals** then **classify**

# Spatio-Temporal Detection

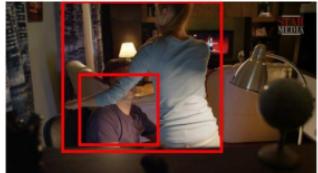
Given a long untrimmed video, detect all the people in both space and time and classify the activities they are performing.  
Some examples from AVA Dataset:



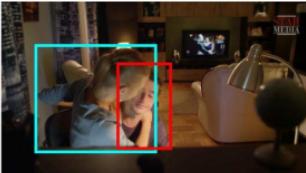
clink glass → drink



open → close



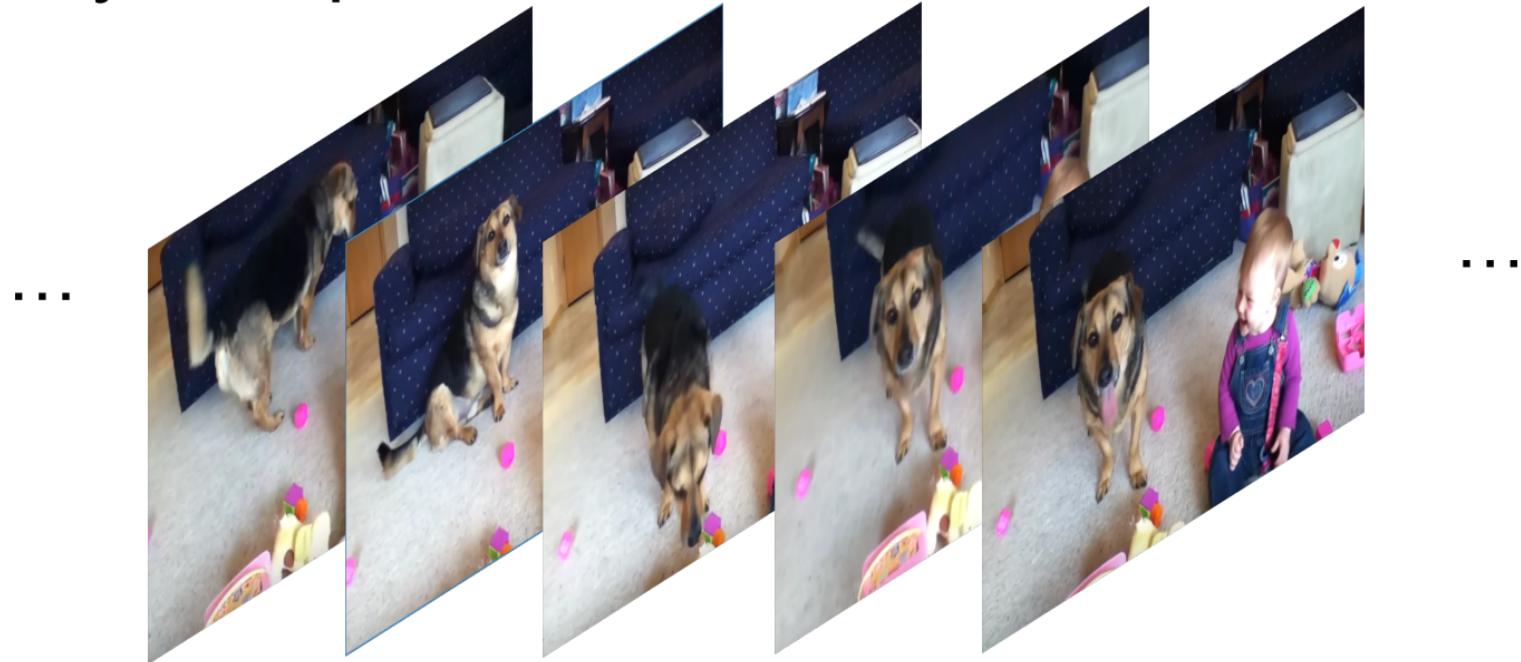
grab (a person) → hug



look at phone → answer phone

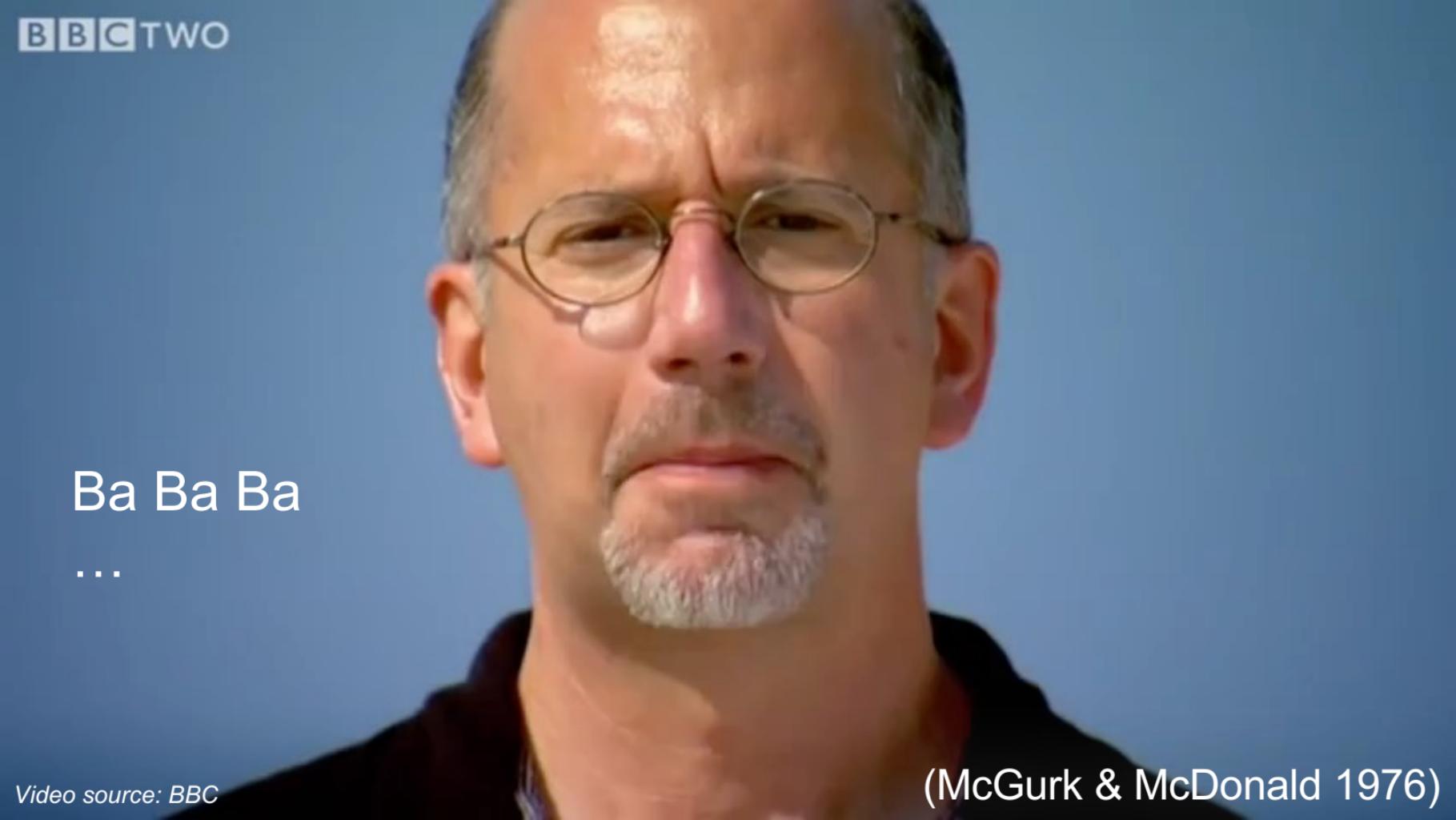


# Today: Temporal Stream



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





Ba Ba Ba

...

(McGurk & McDonald 1976)

Fa Fa Fa ...

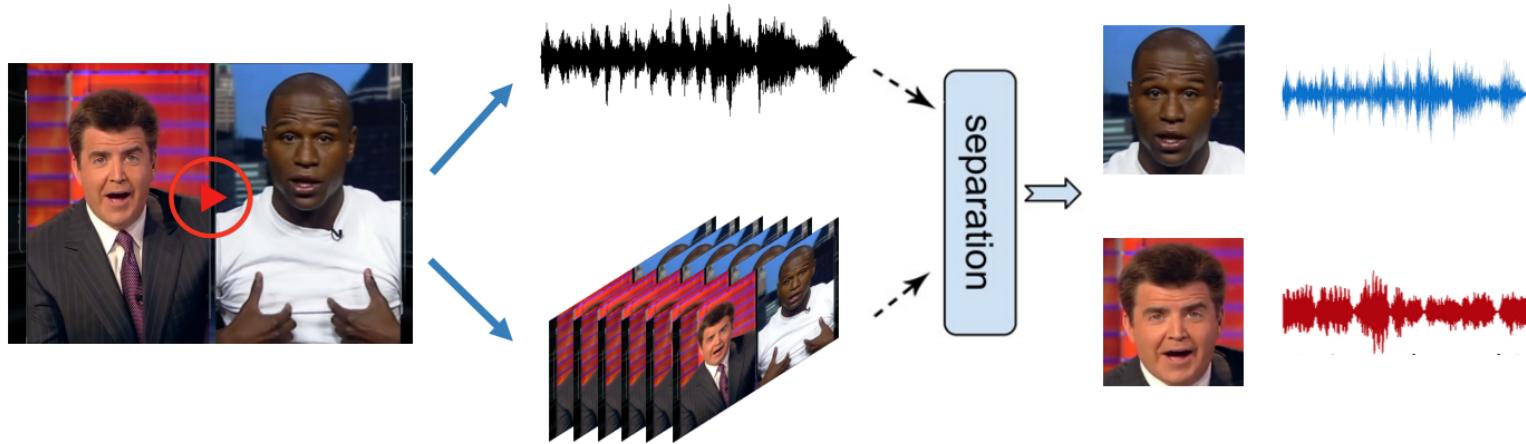
(McGurk & McDonald 1976)



Video source: BBC

(McGurk & McDonald 1976)

# Visually-guided audio source separation



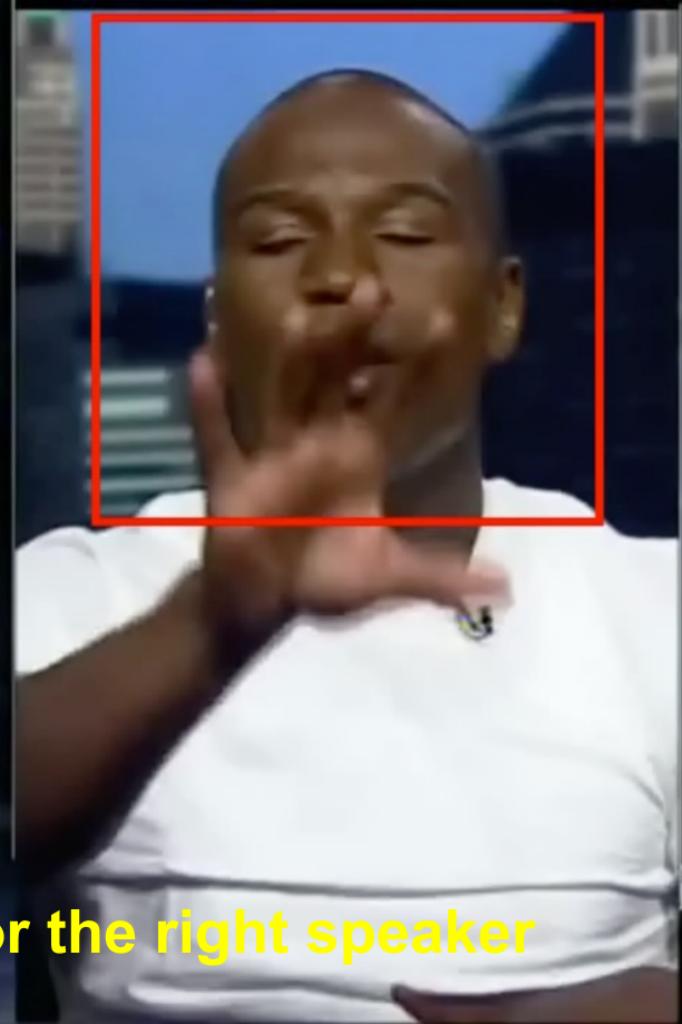
[Gao et al. ECCV 2018, Afouras et al. Interspeech'18, Gabby et al. Interspeech'18, Owens & Efros ECCV'18, Ephrat et al. SIGGRAPH'18, Zhao et al. ECCV 2018, Gao & Grauman ICCV 2019, Zhao et al. ICCV 2019, Xu et al. ICCV 2019, Gan et al. CVPR 2020, Gao et al. CVPR 2021]



**Speech mixture**



**Separated voice for the left speaker**



**Separated voice for the right speaker**

# Musical instruments source separation

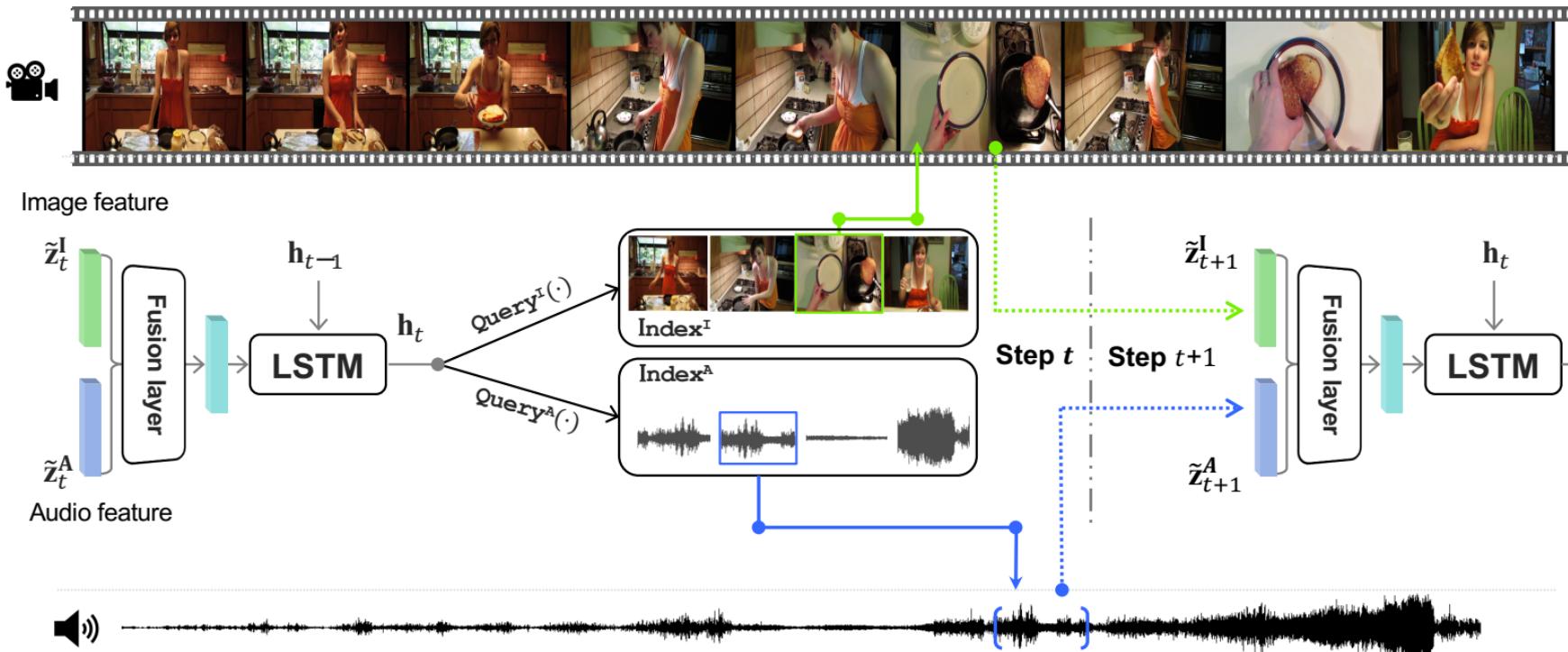
Train on 100,000 unlabeled multi-source video clips,  
then separate audio for novel video.



original video  
(before separation)

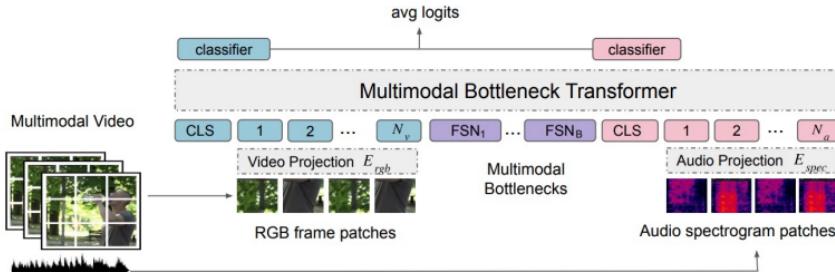
object detections:  
violin & flute

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

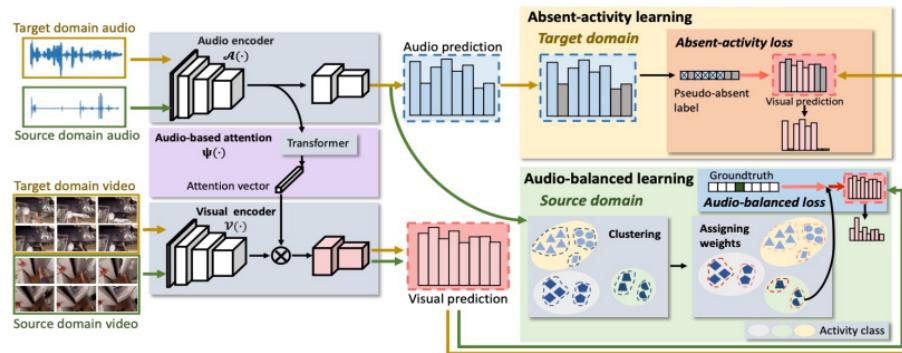


Gao et al., Listen to Look: Action Recognition by Previewing Audio, CVPR 2020

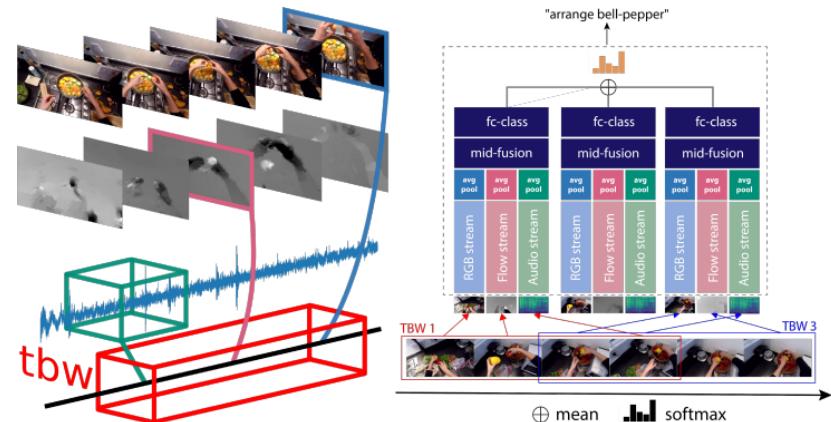
# Multimodal Video Understanding



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

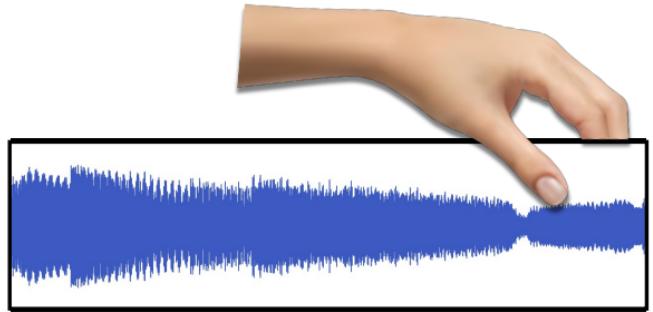
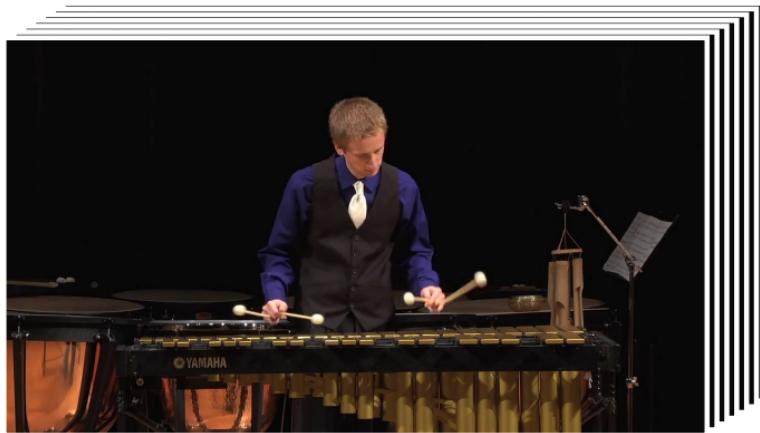


Audio-Adaptive Activity Recognition Across Video Domains, Yunhua et al., CVPR 2022



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

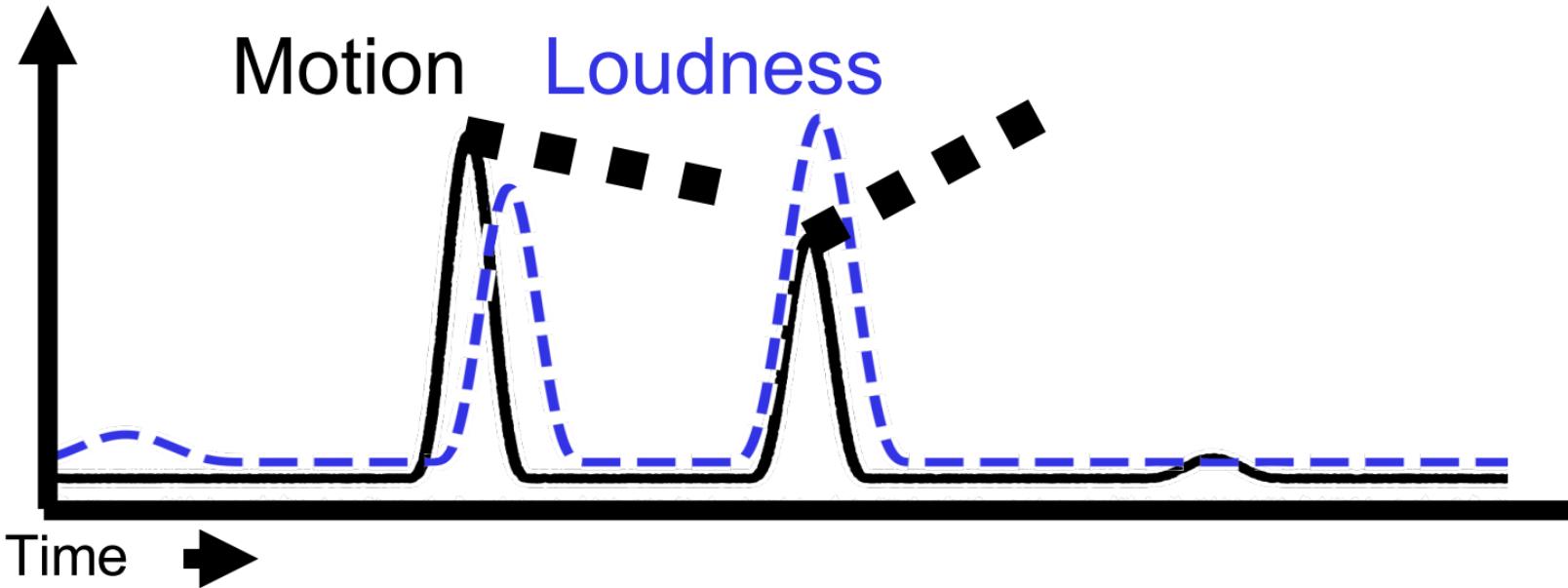
Korbar et al., *Co-training of audio and video representations from self-supervised temporal synchronization*, NeurIPS 2018

# Learning audio-visual synchronization



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

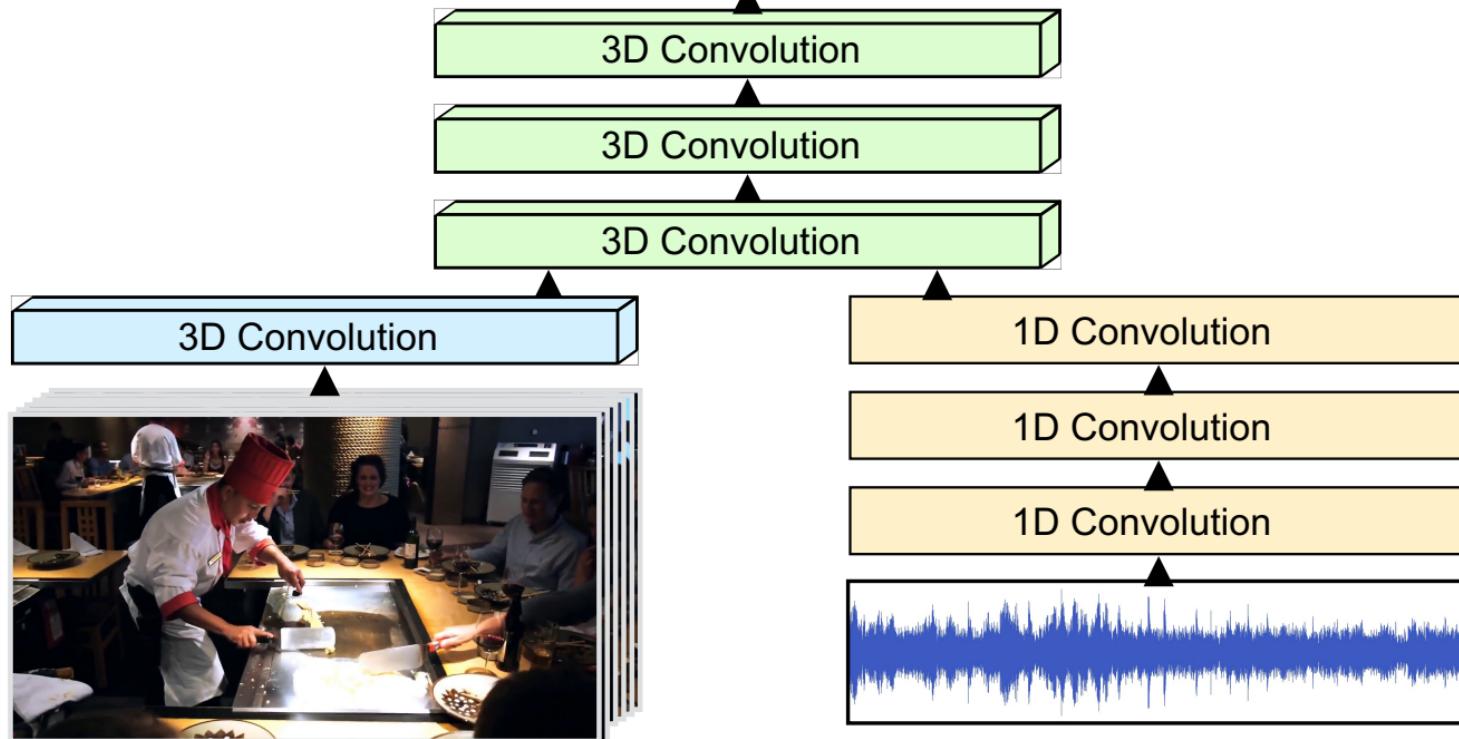
# Learning audio-visual synchronization



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

Top responses per category  
(speech examples omitted)



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 ...

# CS231n: Deep Learning for Computer Vision

- Deep Learning Basics (Lecture 2 – 4)
- • Perceiving and Understanding the Visual World (Lecture 5 – 12)
- • Reconstructing and Interacting with the Visual World (Lecture 13 – 16)
- Human-Centered Artificial Intelligence (Lecture 17 – 18)

# Next time: Generative Models