Lecture 3
Videos

Video Models

Many video models:

Single-frame CNN (Try this first!)

Late fusion

Early fusion

3D CNN / C3D

Two-stream networks

CNN + RNN

Convolutional RNN

Spatio-temporal self-attention

SlowFast networks

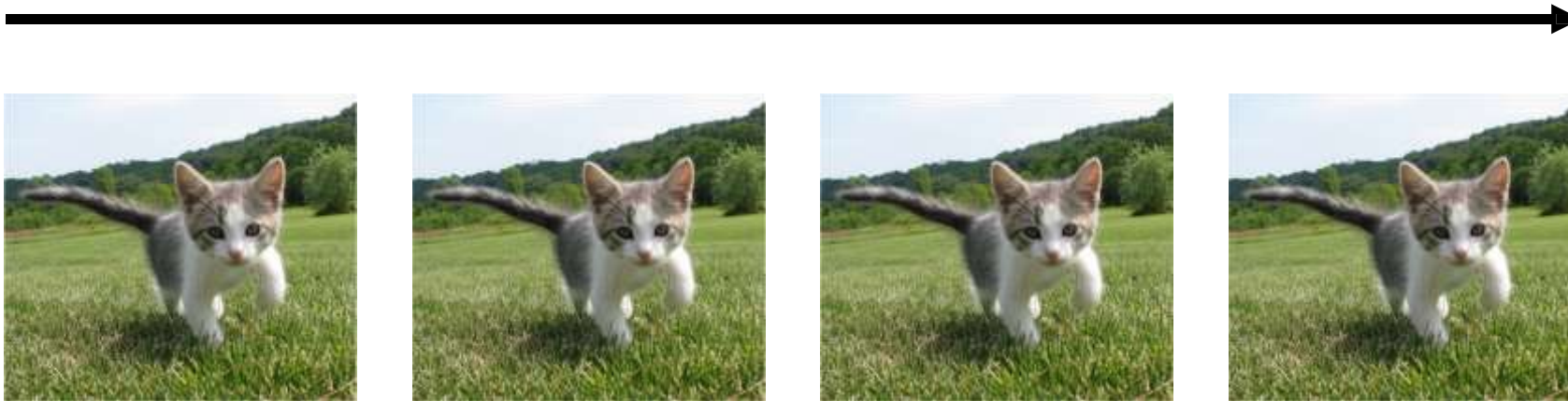# Video

- Sequences of frames
- 30 frames per second

Sequences of Images

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

Example task: Video Classification

Images: Recognize **objects**

Dog
Cat
Fish
Truck

Videos: Recognize **actions**

Swimming
Running
Jumping
Eating
Standing

Action recognition

- Given a video
  - Recognize which action is present

Action recognition
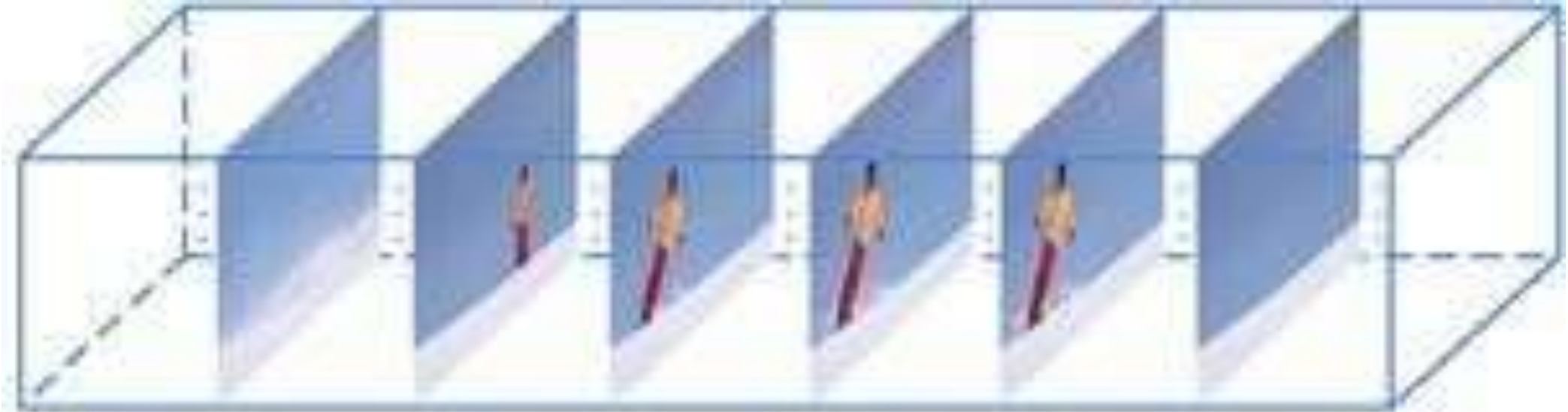
- Variations
  - Multiple instances

Action recognition

- Variations
  - Multiple action

Action recognition

- Variations
  - Trimmed/untrimmed
  - Temporal action localization

Action detection

- Spatio-temporal localization

Action recognition – UCF101

Cycling

Diving

Golf Swinging

Riding

Volleyball

Basketball Shooting

Swinging

Tennis Swinging

Action detection - VIRAT

Video segmentation

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 x 3 x H x W

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**

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)

Input video:
T x 3 x H x W

Training on Clips

Raw video: Long, high FPS

Training on Clips

Raw video: Long, high FPS

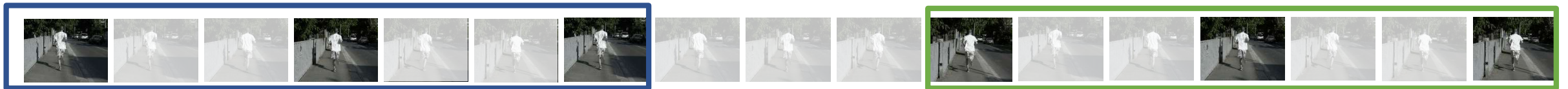**Training**: Train model to classify short **clips** with low FPS

Training on Clips

Raw video: Long, high FPS

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

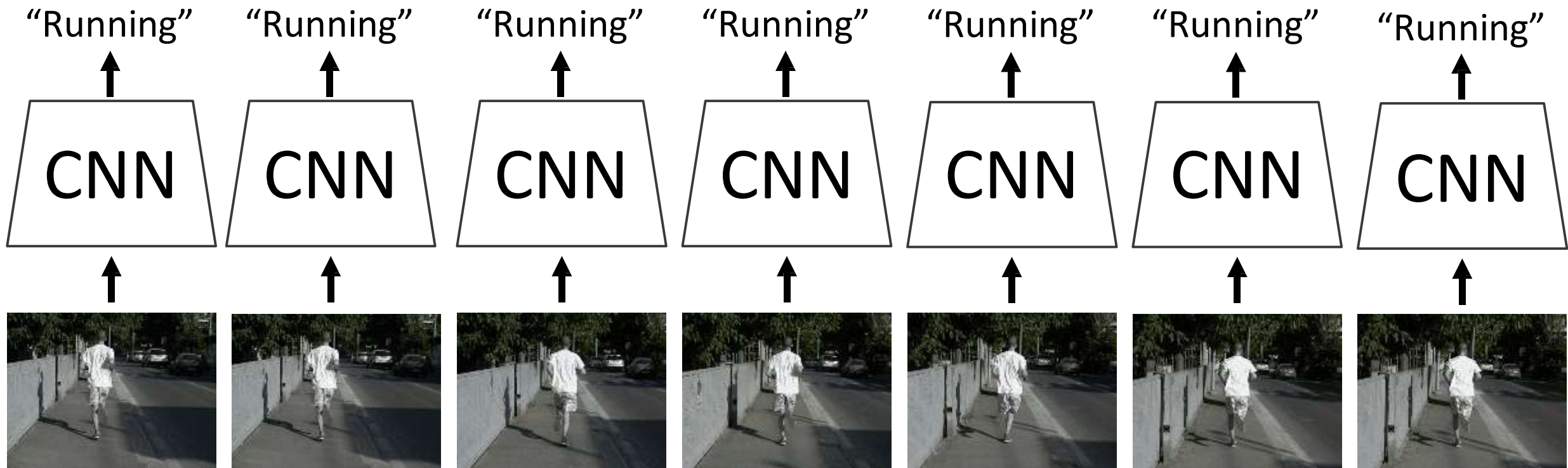**Testing**: Run model on different clips, average predictions

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

Video Classification: Late Fusion (with FC layers)

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

Class scores: C

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

MLP

Clip features: TDH'W'

Flatten

Frame features
T x D x H' x W'

2D CNN on each frame

CNN CNN CNN CNN CNN CNN

Input:
T x 3 x H x W

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

Video Classification: Late Fusion (with pooling)

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

Class scores: C

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

Linear

Clip features: D

Average Pool over space and time

Frame features
T x D x H' x W'

2D CNN on
each frame

CNN CNN CNN CNN CNN CNN

Input:
T x 3 x H x W

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

Class scores: C

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

Clip features: D

Linear

Average Pool over space and time

Frame features
T x D x H' x W'

2D CNN on each frame

CNN   CNN   CNN   CNN   CNN   CNN
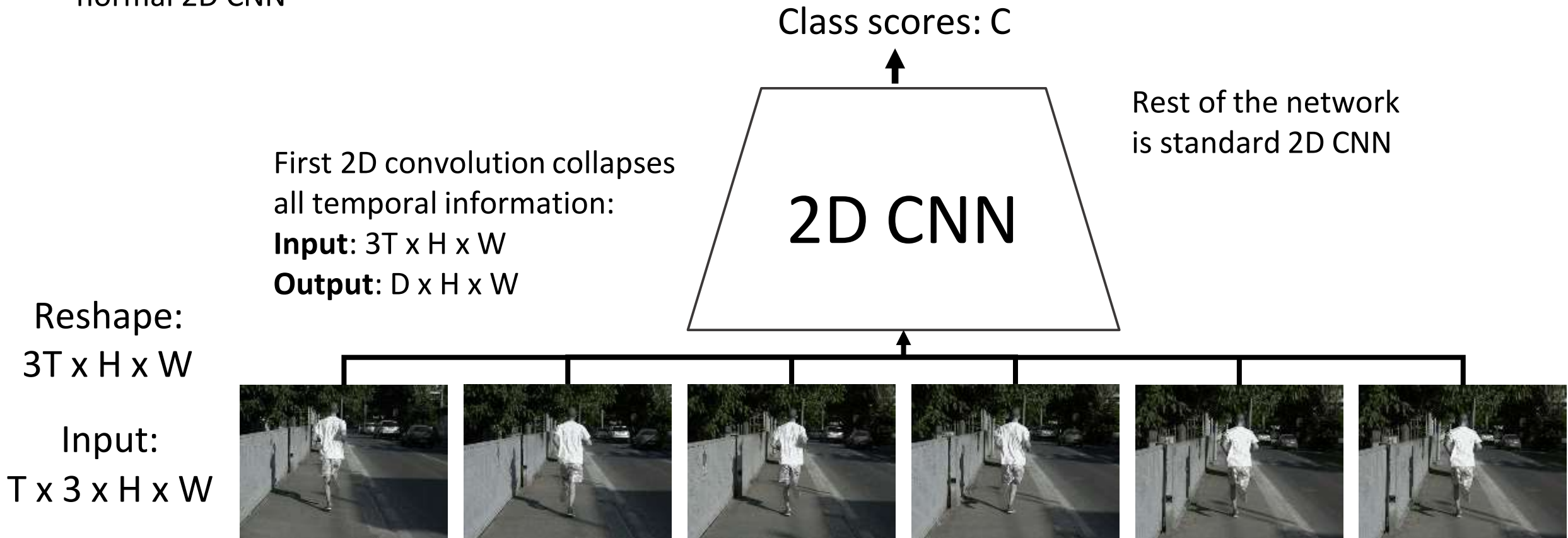
Input:
T x 3 x H x W

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

2D CNN

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

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 x H x W
Output: D x H x W

Class scores: C

Rest of the network is standard 2D CNN

2D CNN

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

Video Classification: 3D CNN

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

Class scores: C

↑

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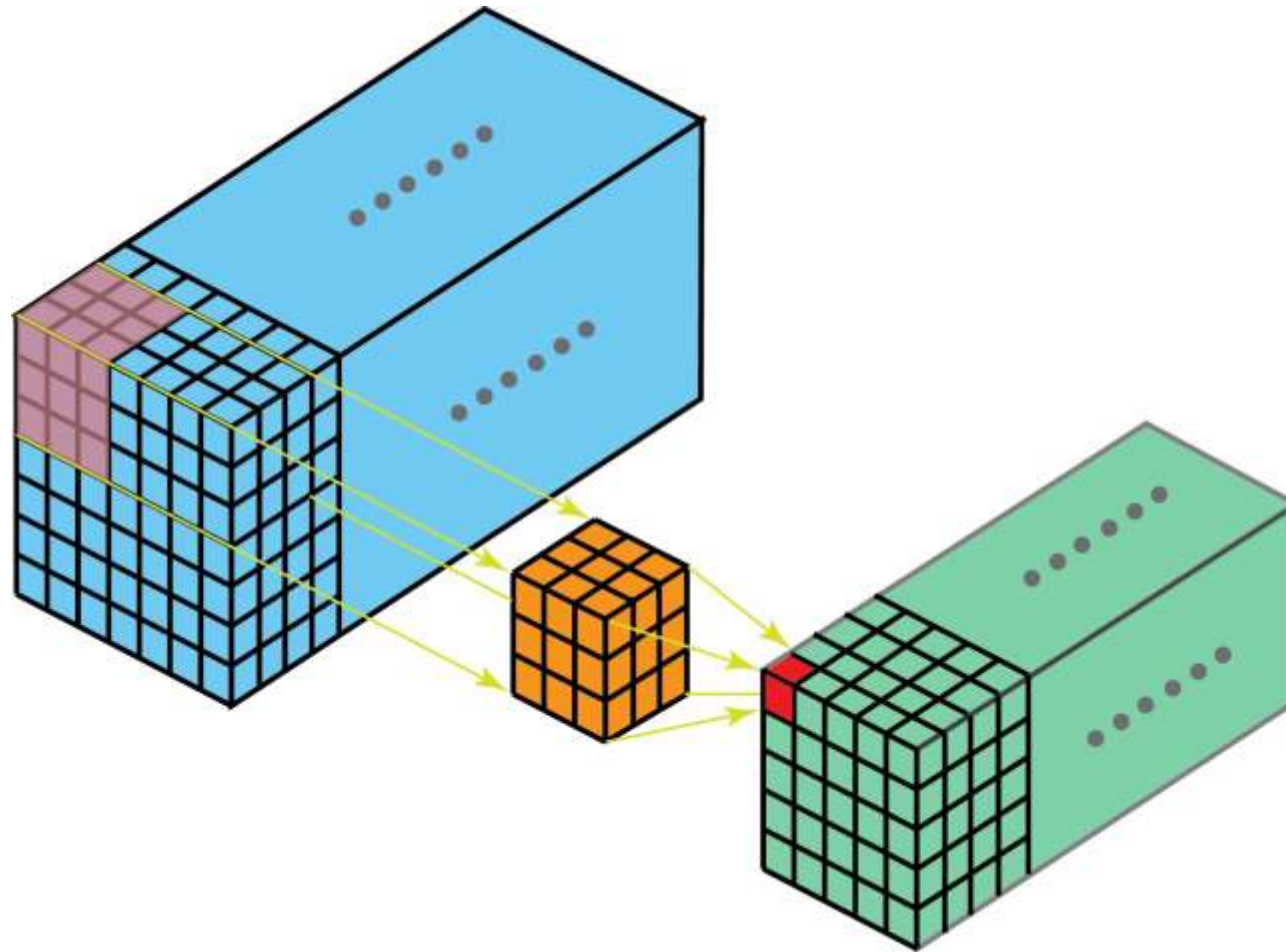4D tensor: D x T x H x W
Use 3D conv and 3D pooling
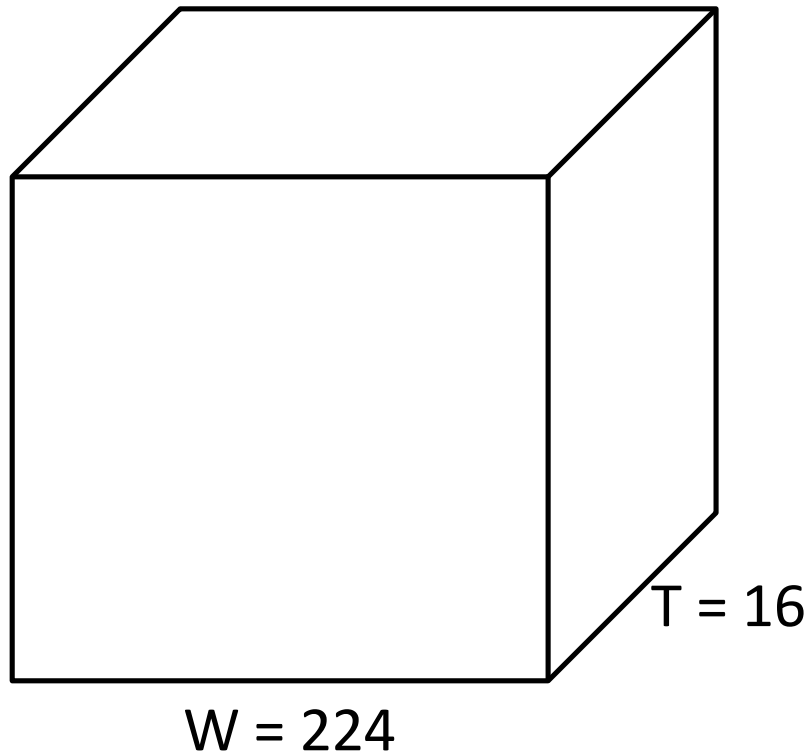operations

3D CNN

↑

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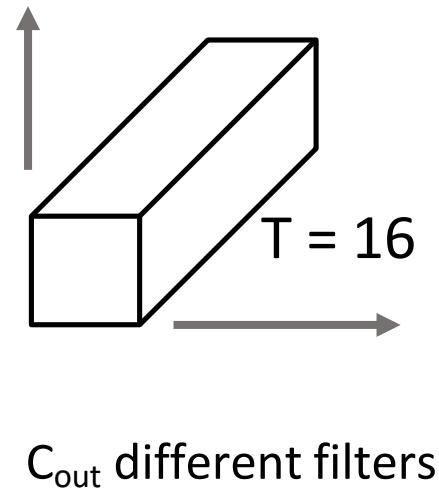
3D convolution
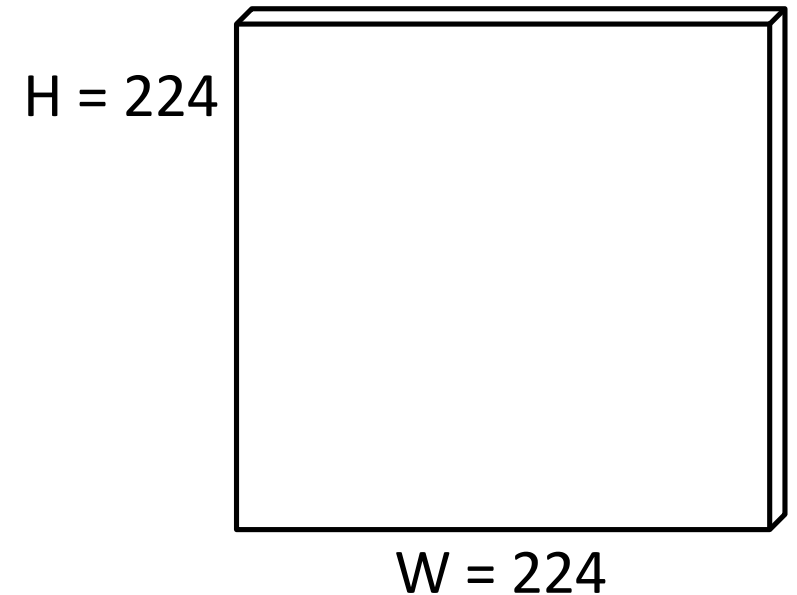
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

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

H = 224

W = 224

T = 16

T = 16

C_{out} different filters

H = 224

W = 224

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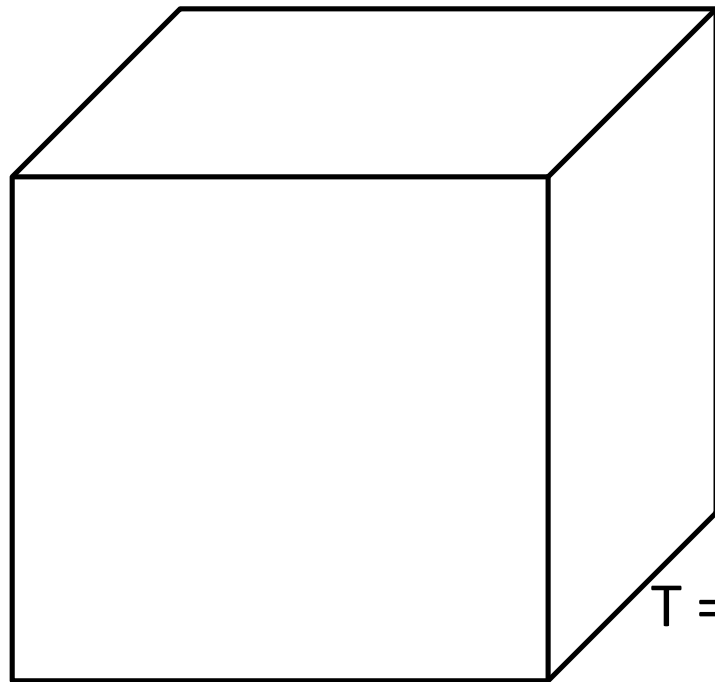
Output:
C_{out} x H x W
2D grid with C_{out} –dim
feat at each point

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

H = 224
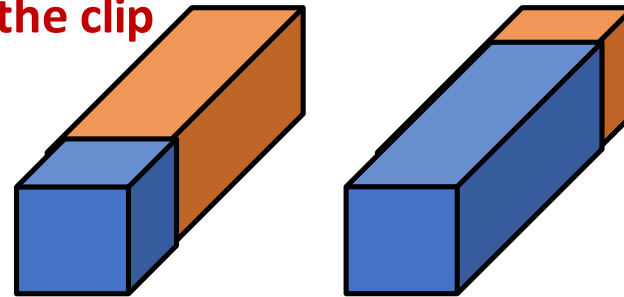
T = 16

C_{out} different filters

W = 224

W = 224

2D Conv (Early Fusion) vs <u>3D Conv (3D CNN)</u>

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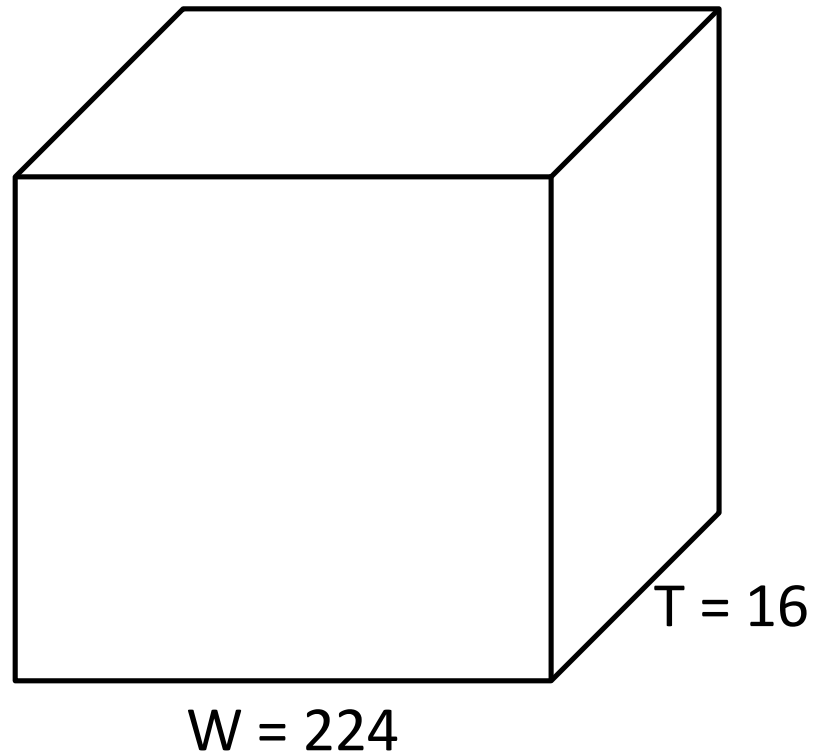(3D grid with C_{in}-dim
feat at each point)

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

H = 224

T = 3

H = 224

T = 16

C_{out} different filters

W = 224

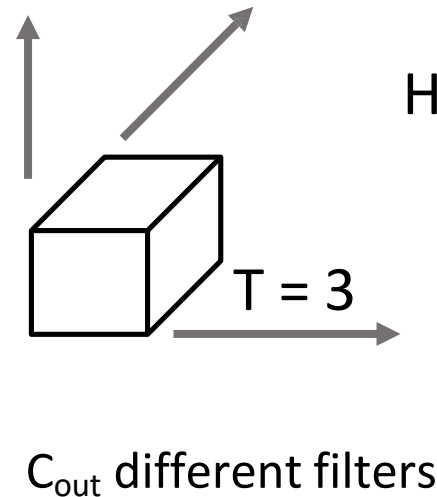W = 224

2D Conv (Early Fusion) vs <u>3D Conv (3D CNN)</u>

Input: C_{in} x T x H x W
(3D grid with C_{in}-dim
feat at each point)
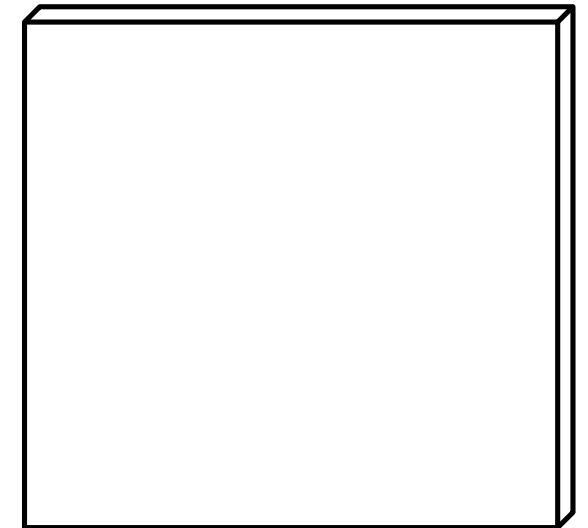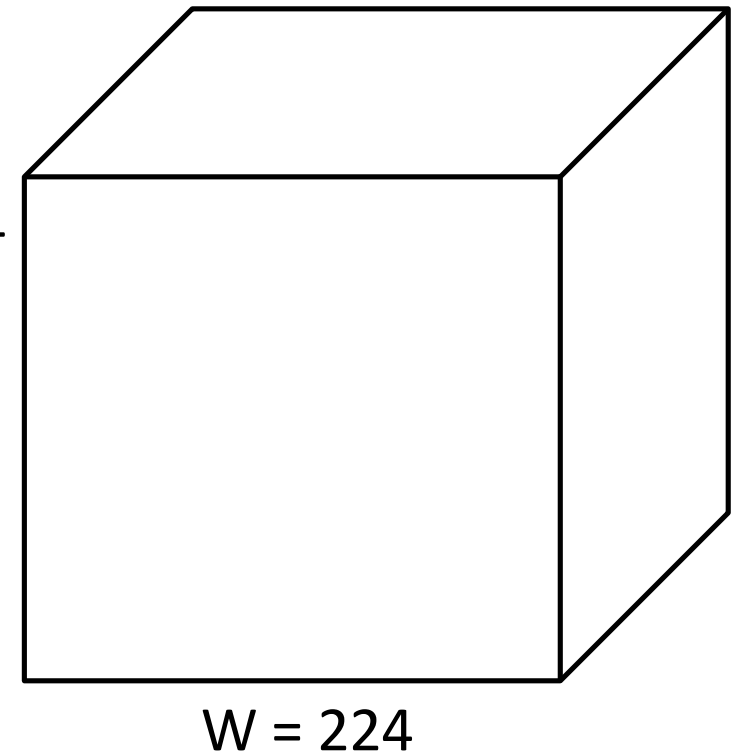
Weight:
C_{out} x C_{in} x 3 x 3 x 3
Slide over x and y

Output:
C_{out} x T x H x W
3D grid with C_{out}–dim
feat at each point
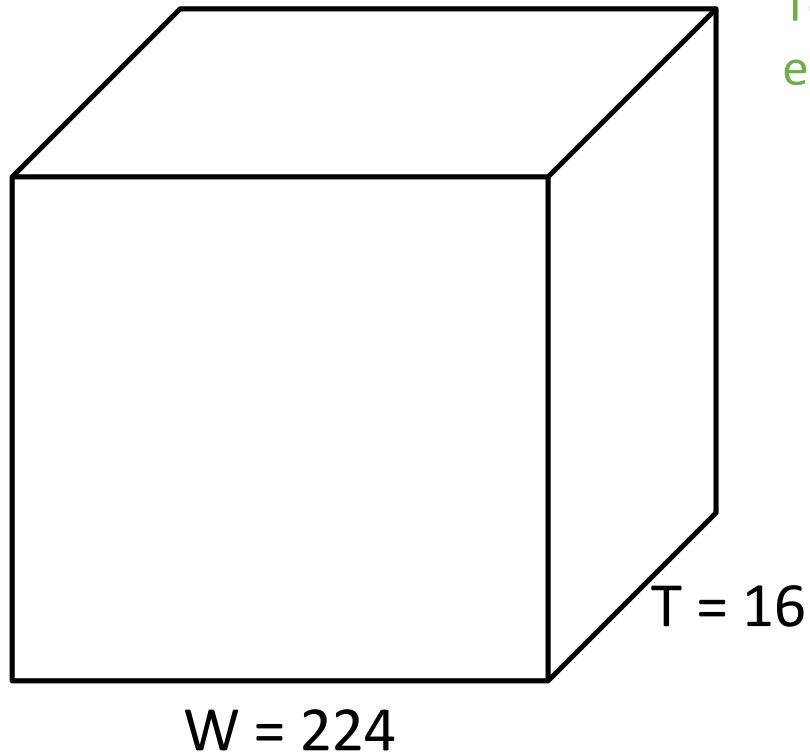
Temporal shift-invariant since
each filter slides over time!
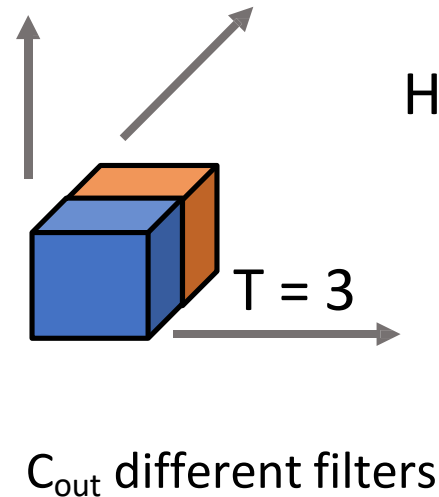
H = 224

T = 3

H = 224

T = 16

C_{out} different filters

W = 224

W = 224
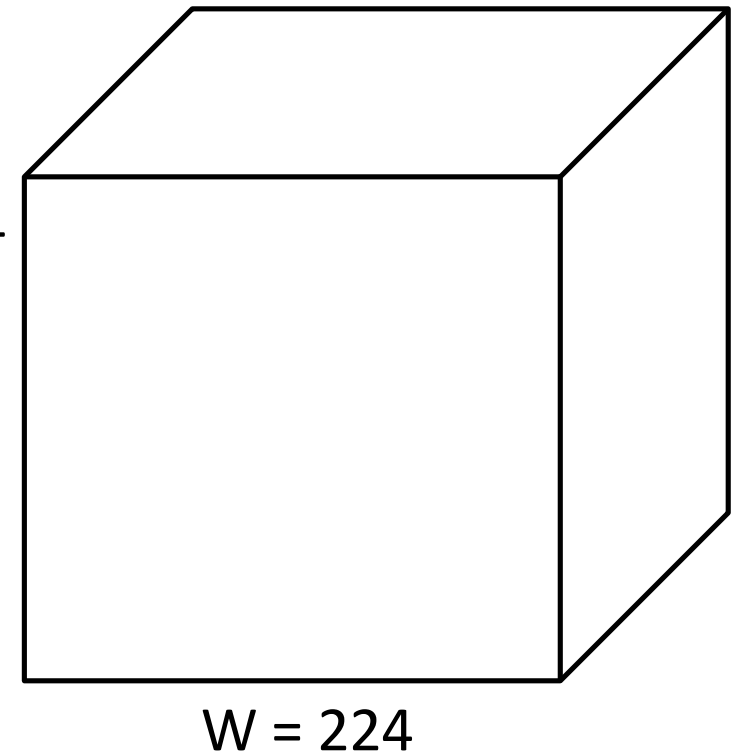
Early Fusion vs Late Fusion vs 3D CNN

Sports-1M Top-5 Accuracy
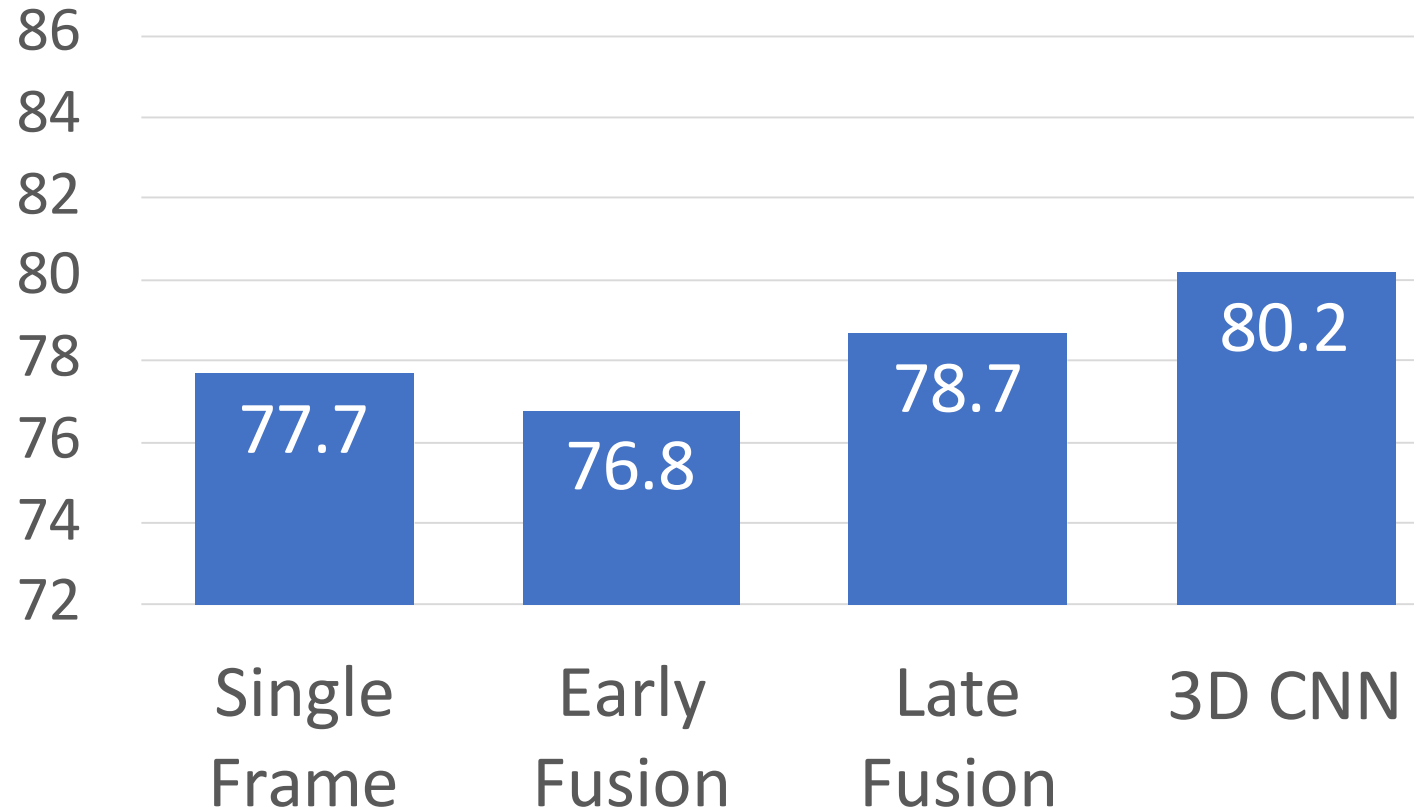
Single Frame model works well – always try this first!

3D CNNs have improved a lot since 2014!

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

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

Early Fusion vs Late Fusion vs 3D CNN

Sports-1M Top-5 Accuracy

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
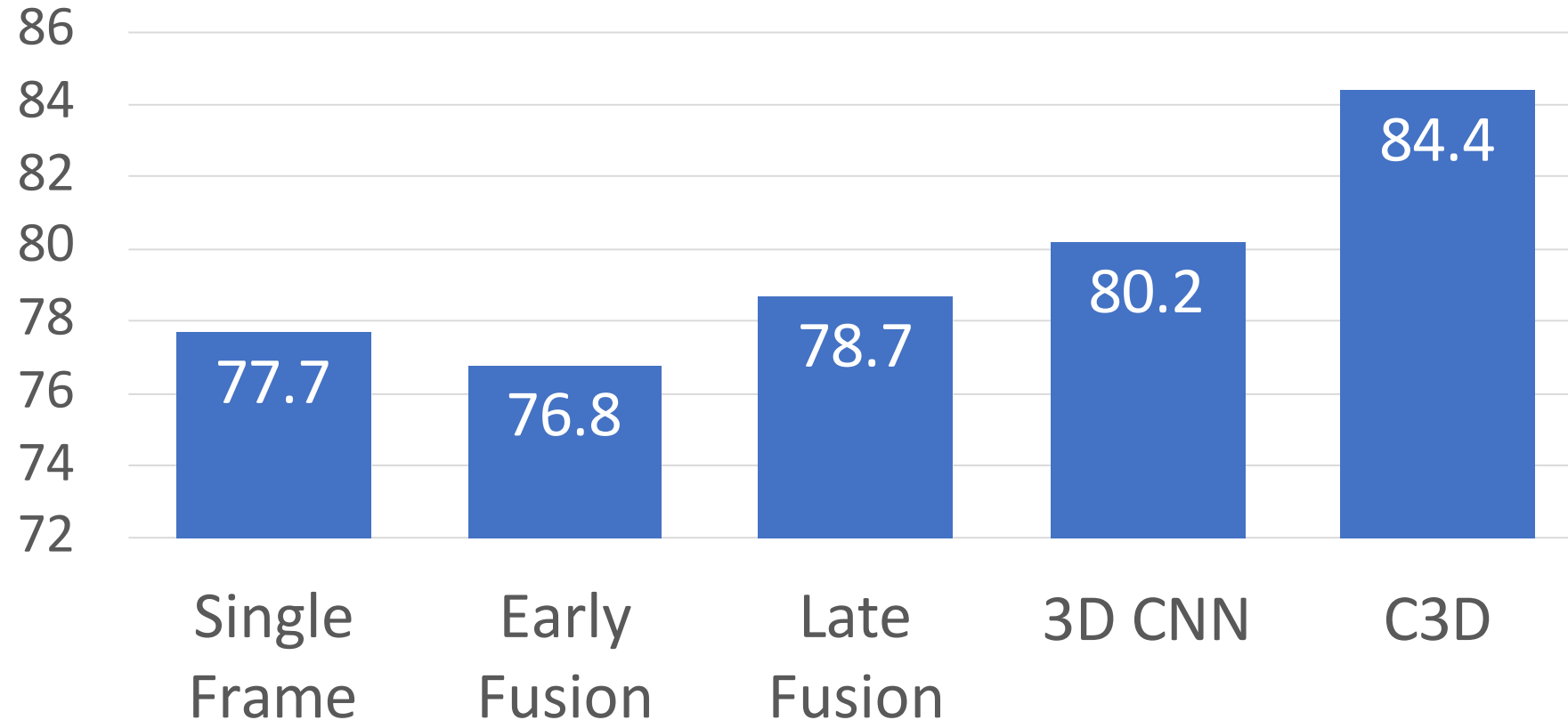
C3D: The VGG of 3D CNNs

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!) for a forward pass**

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

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

Recognizing Actions from Motion

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

https://www.youtube.com/watch?v=1F5ICP9SYLU&ab_channel=BioMotionLab

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

Measuring Motion: Optical Flow
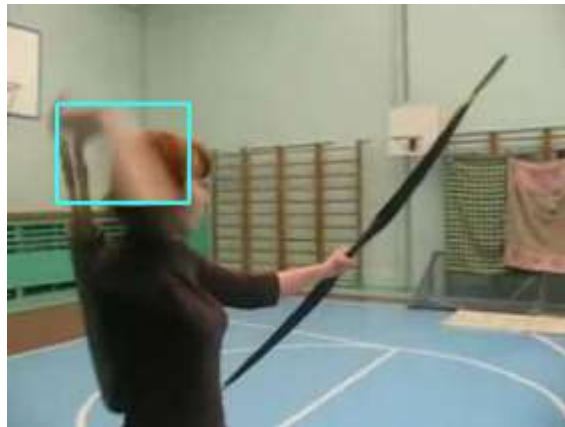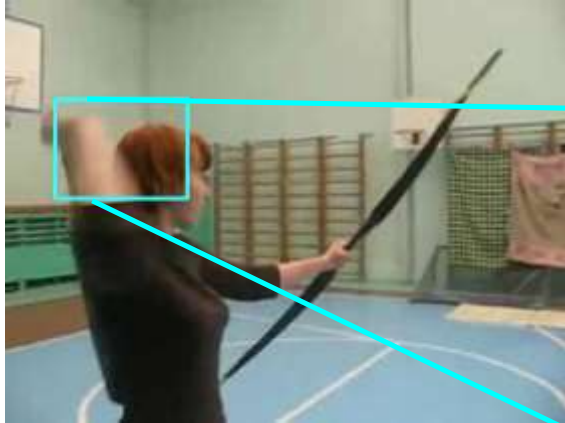
Image at frame t

Image at frame t+1

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

Measuring Motion: Optical Flow

Image at frame t

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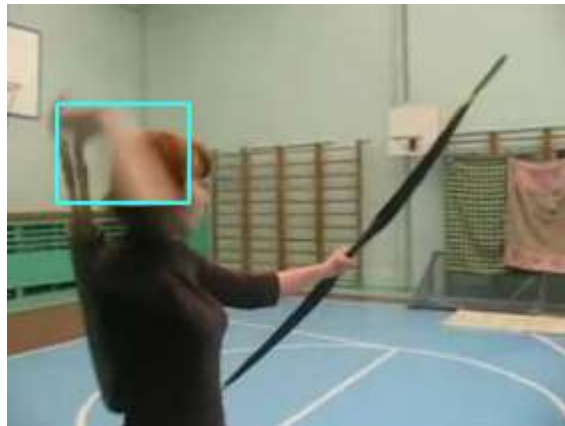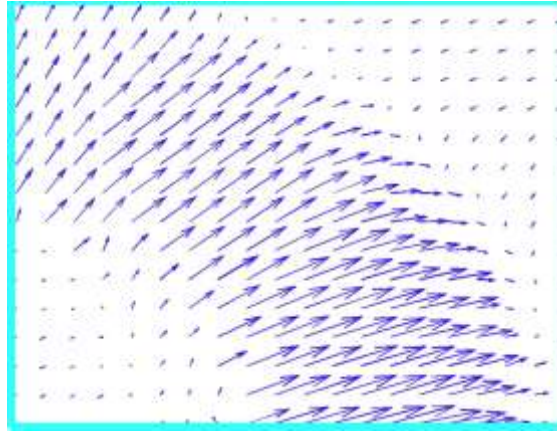

Image at frame t+1

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

Measuring Motion: Optical Flow

Optical Flow highlights **local motion**
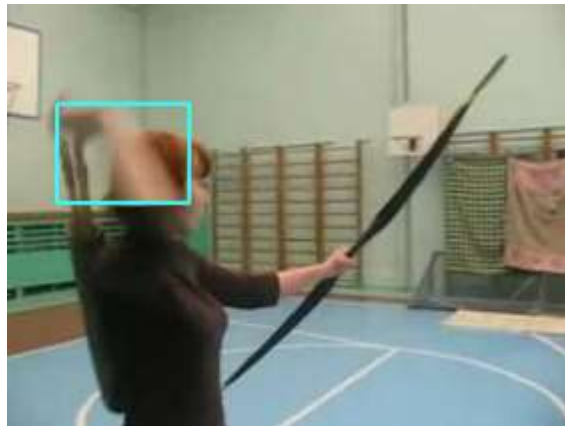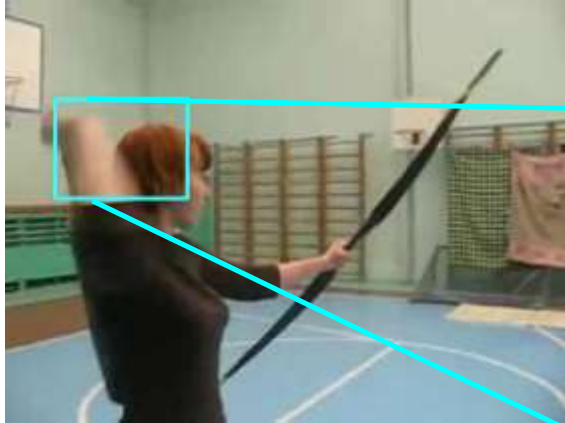
Image at frame t

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

Image at frame t+1

Horizontal flow dx

Vertical Flow dy

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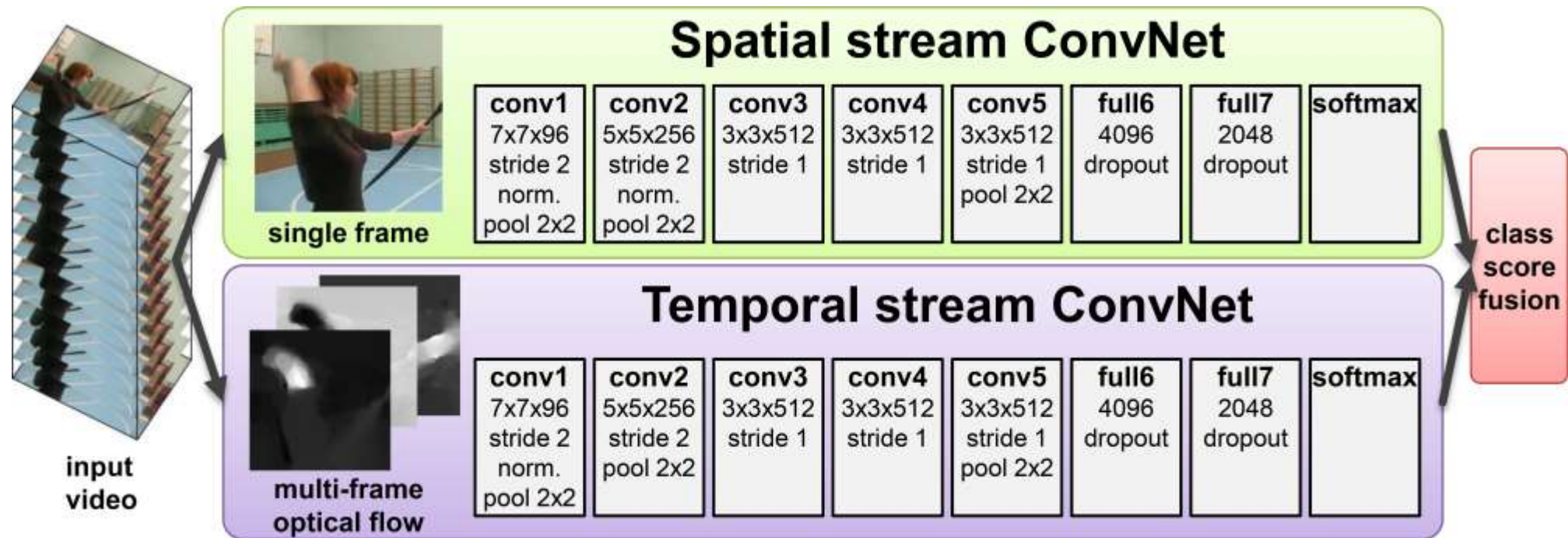
Input: Stack of optical flow:
[2*(T-1)] x H x W

Early fusion: First 2D conv
processes all flow images

Simonyan and Zisserman, "Two-stream convolutional networks for action recognition in videos", NeurIPS 2014
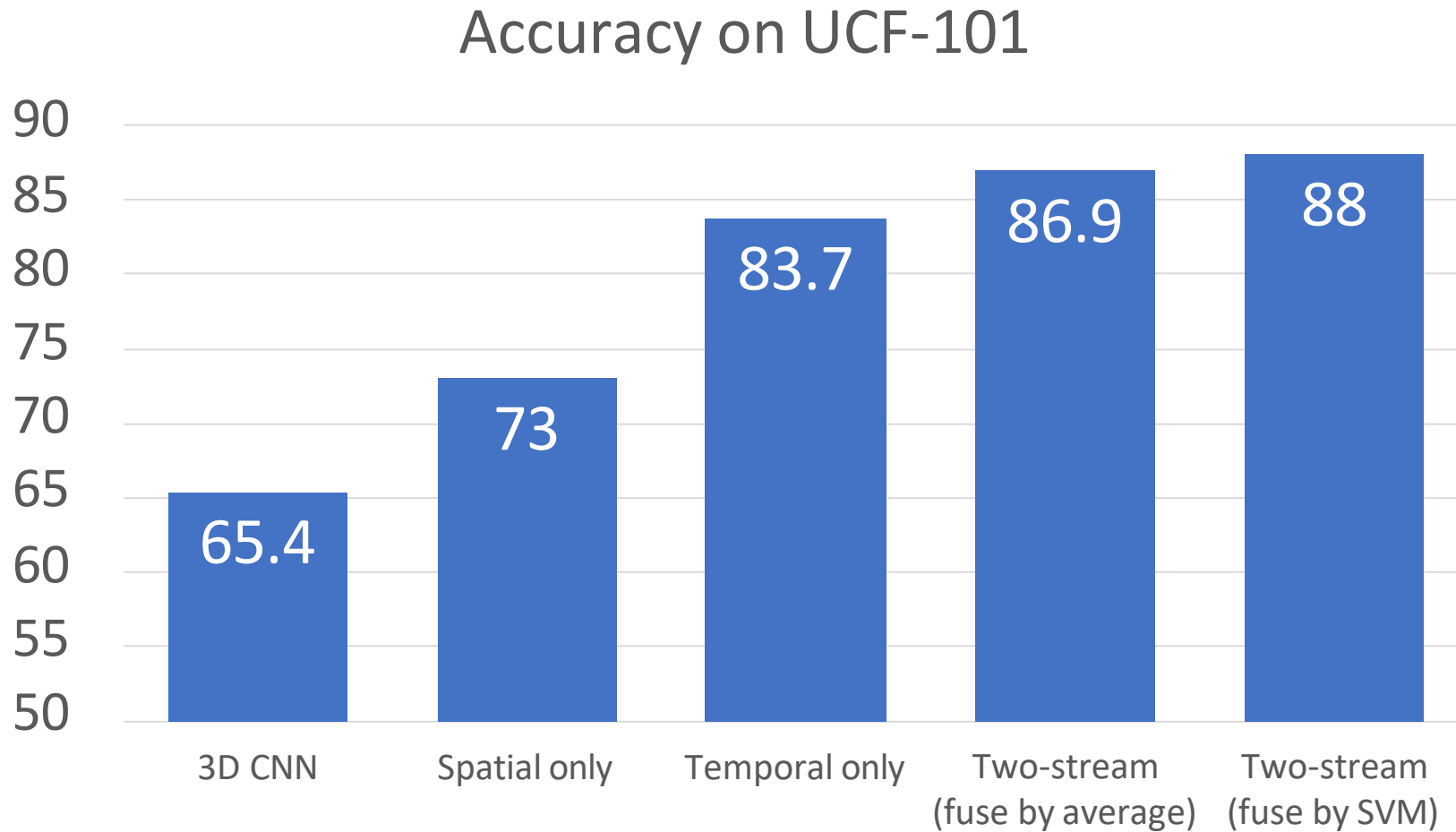
Separating Motion and Appearance: Two-Stream Networks

Accuracy on UCF-101

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

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?
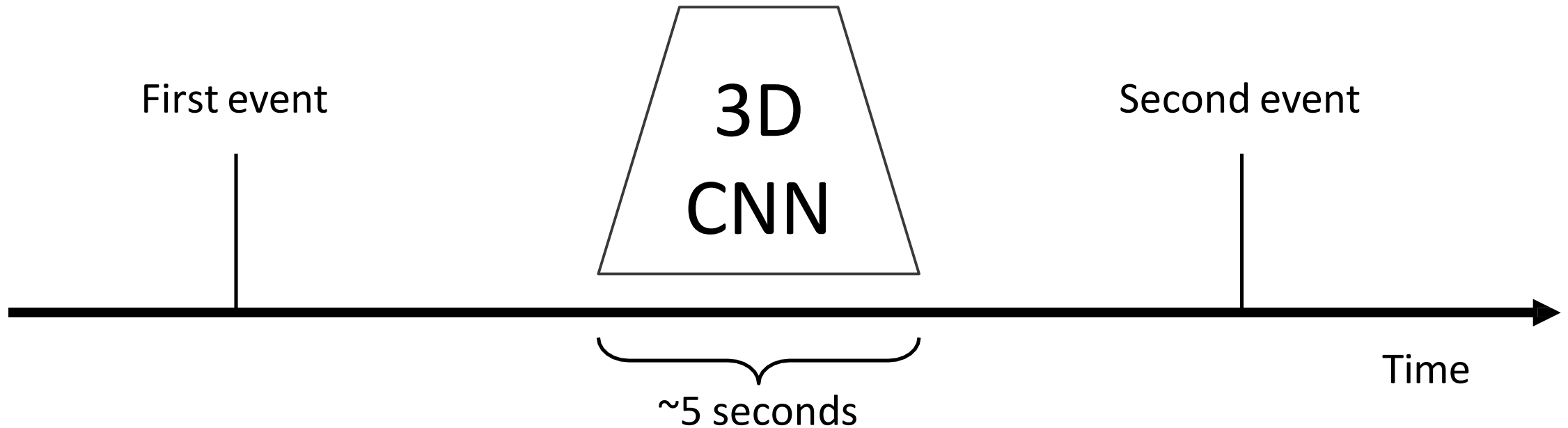
Modeling long-term temporal structure

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?
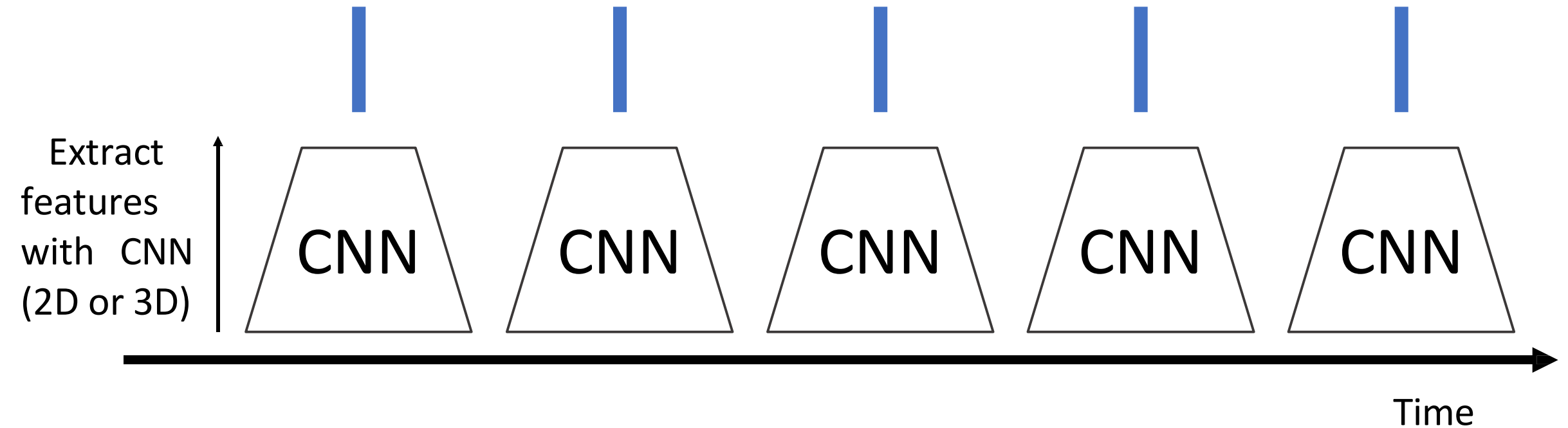
First event

3D CNN

Second event

Time

~5 seconds

Modeling long-term temporal structure

Extract features with CNN (2D or 3D)

Time

Modeling long-term temporal structure

Process local features using recurrent network (e.g. LSTM)
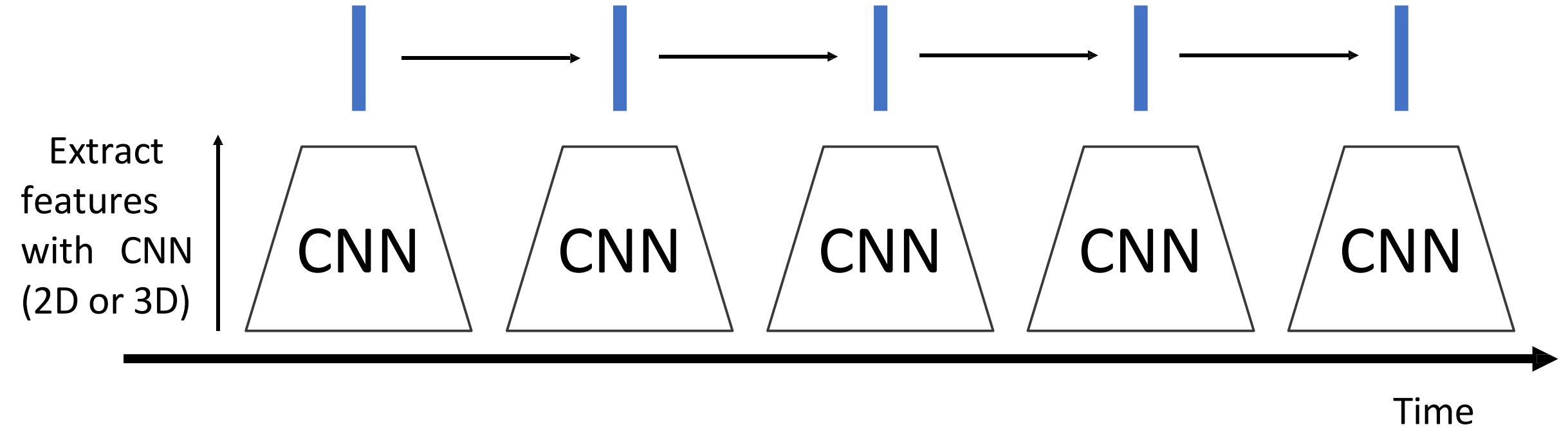
Modeling long-term temporal structure

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

Modeling long-term temporal structure

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

Extract features with CNN (2D or 3D)

Time

Modeling long-term temporal structure

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

Extract features with CNN (2D or 3D)

Used 3D CNNs and LSTMs in 2011! Way ahead of its time

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

Time

Modeling long-term temporal structure

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

Extract features with CNN (2D or 3D)

Used 3D CNNs and LSTMs in 2011! Way ahead of its time

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

Time

Modeling long-term temporal structure

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

Extract features with CNN (2D or 3D)

Time

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

Modeling long-term temporal structure

Sometimes don't backprop to CNN to save memory; pretrain and use it as a feature extractor
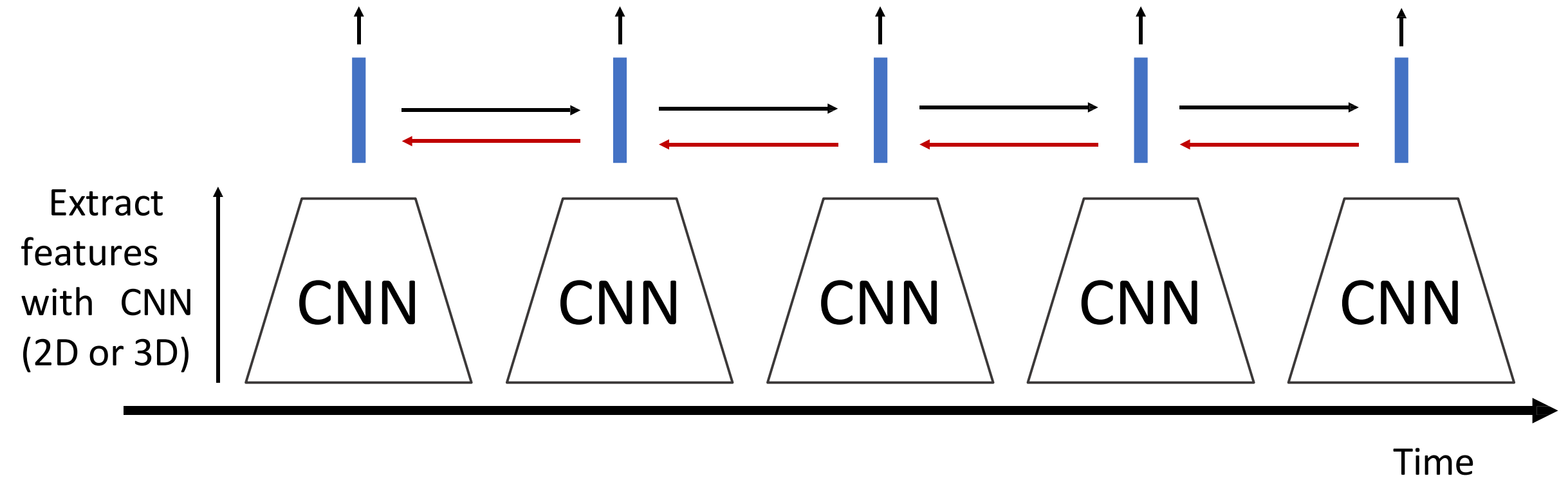
Baccouche et al, "Sequential Deep Learning for Human Action Recognition", 2011
Donahue et al, "Long-term recurrent convolutional networks for visual recognition and description", CVPR 2015

Modeling long-term temporal structure

Inside CNN: Each value 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?
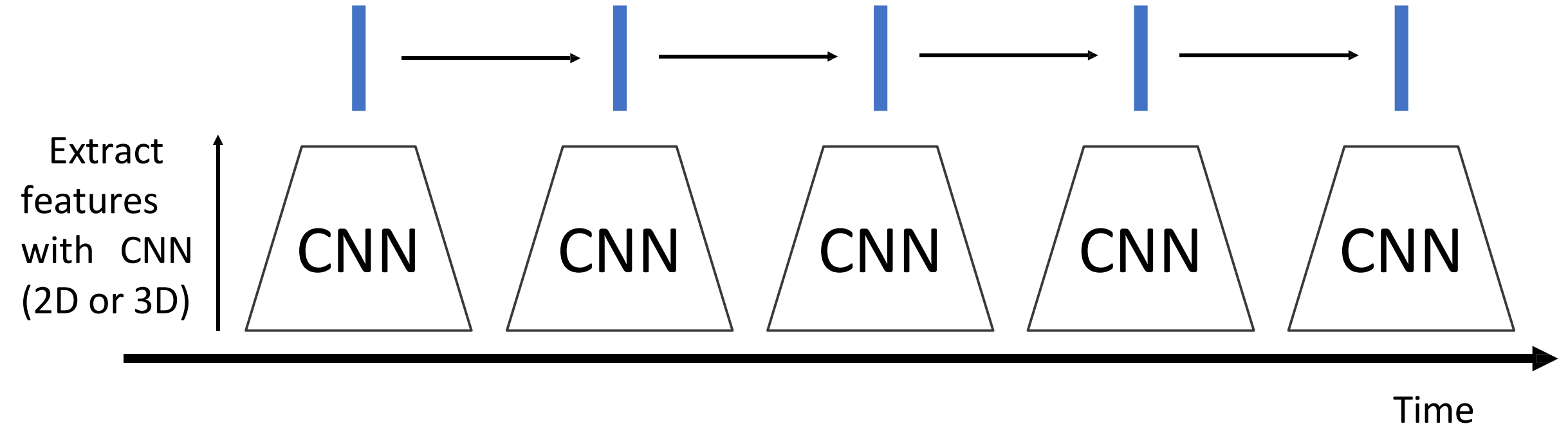
Extract features with CNN (2D or 3D)

Time

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

# Recall: Multi-layer RNN

We can use a
similar structure to
process videos!

Three-layer RNN

depth

time

Recurrent Convolutional Network

Layer 3

Layer 2

Layer 1

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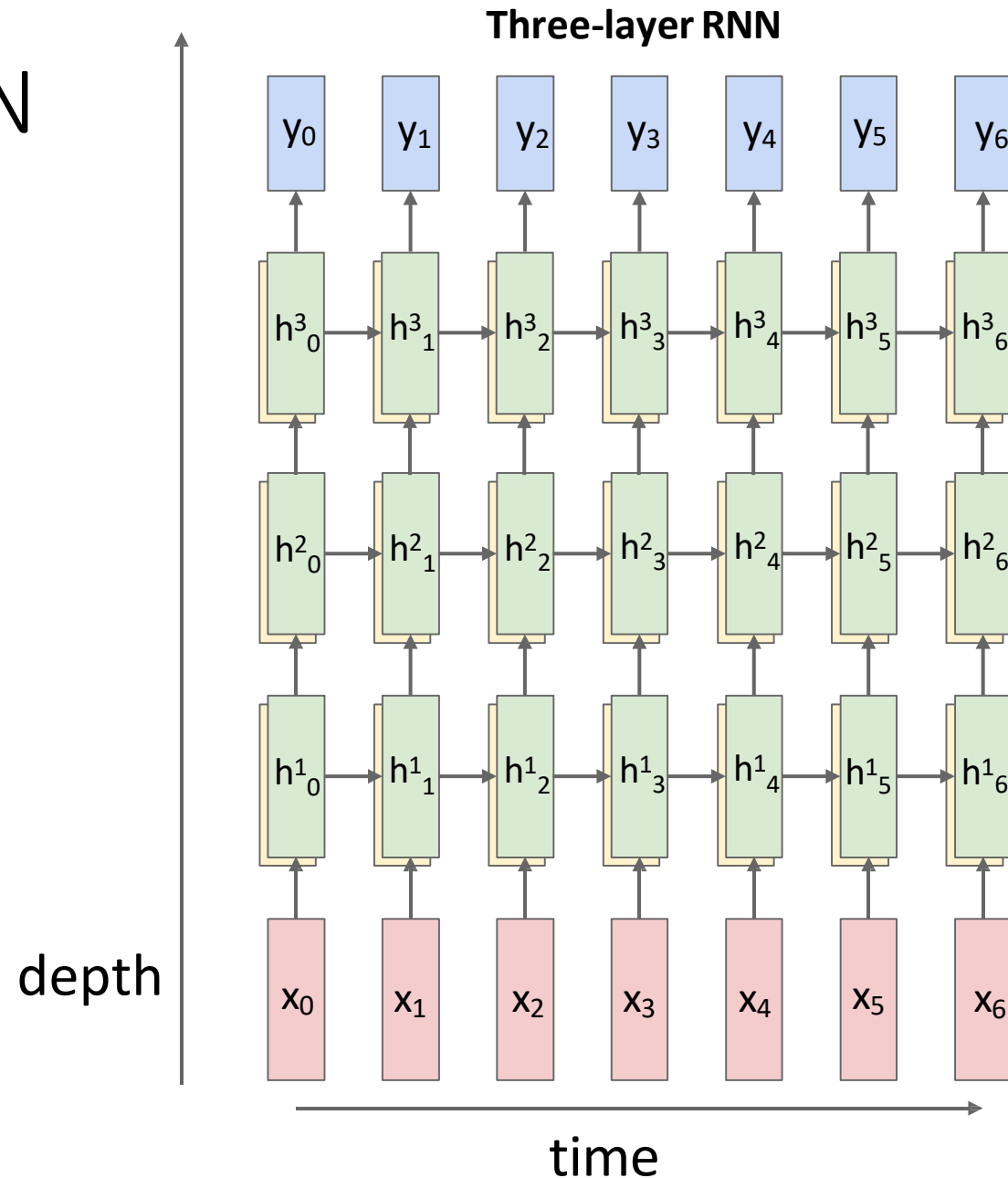
Entire network uses 2D feature maps: C x H x W

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

Recurrent Convolutional Network

Normal 2D CNN:

2D Conv

Input features:
C x H x W

Output features:
C x H x W
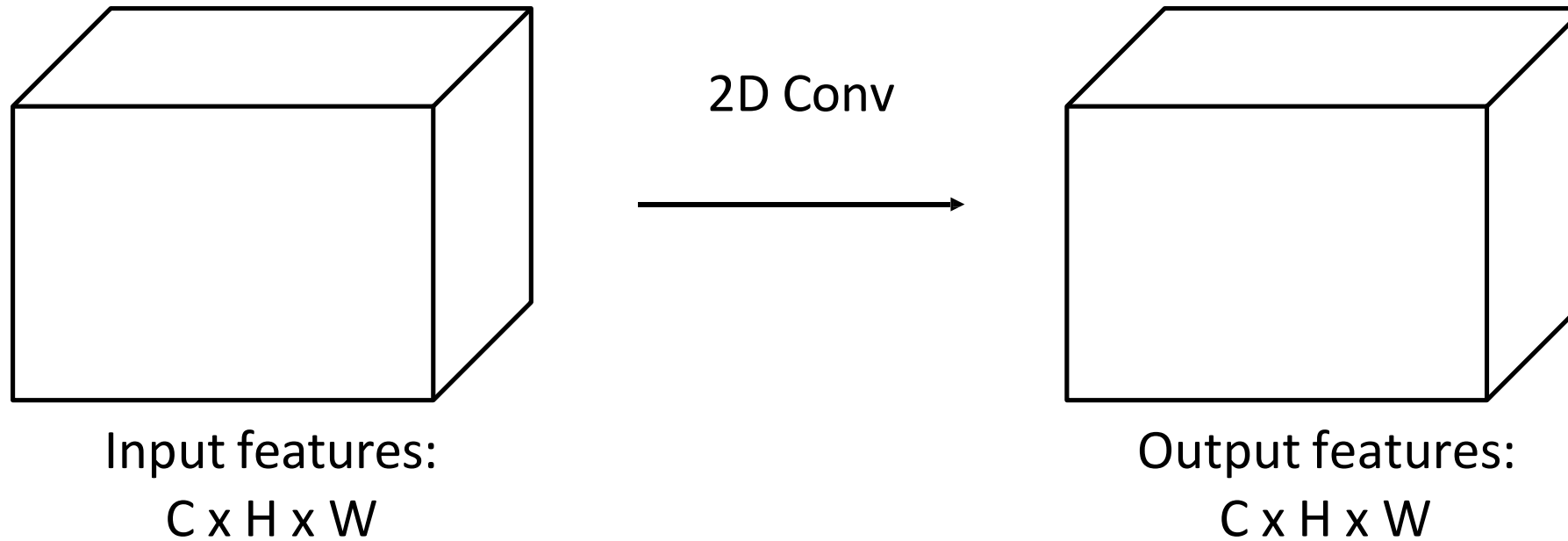
Recurrent Convolutional Network
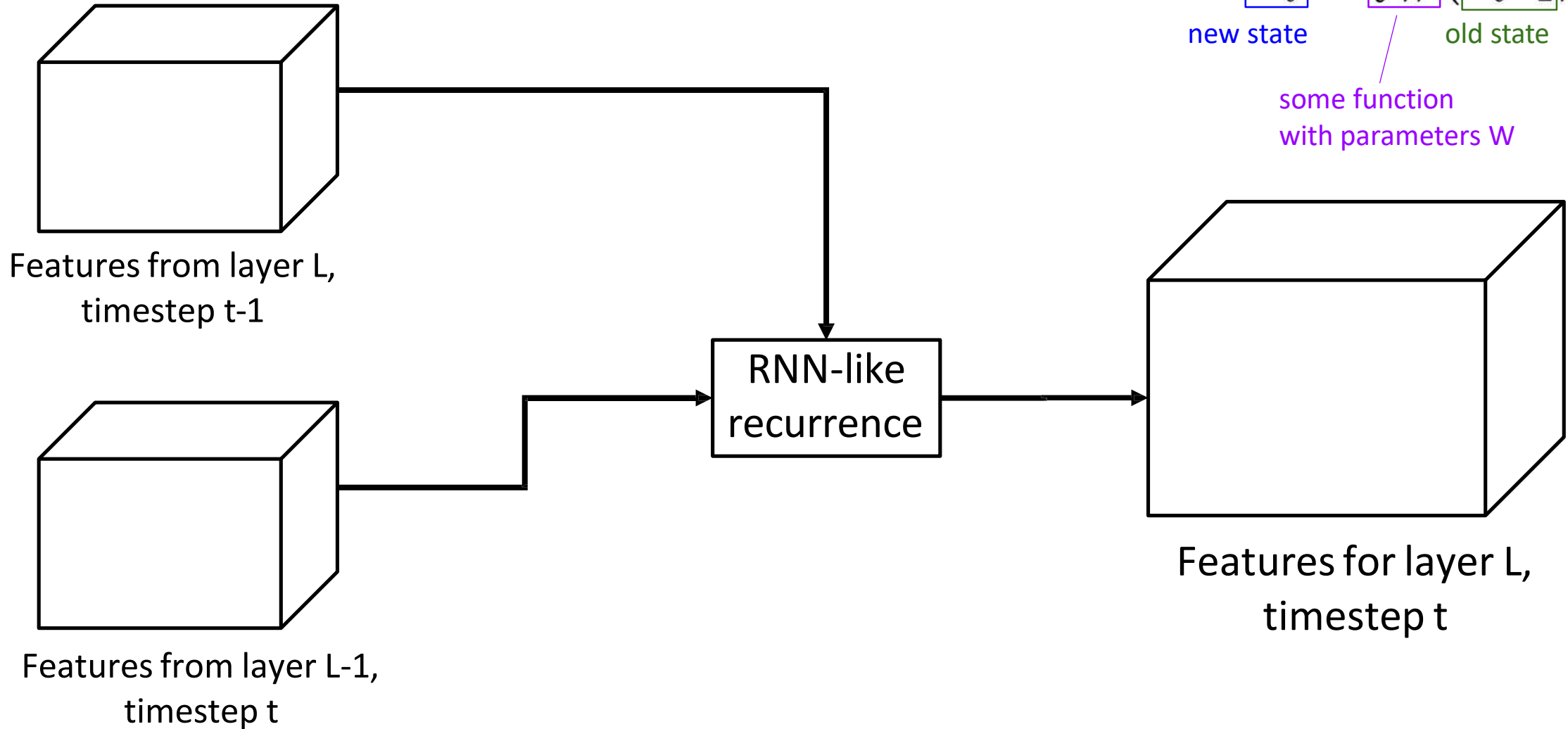
Recall: Recurrent Network

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

new state   old state

some function
with parameters W

Features from layer L,
timestep t-1

Features from layer L-1,
timestep t

RNN-like
recurrence

Features for layer L,
timestep t

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

Recurrent Convolutional Network
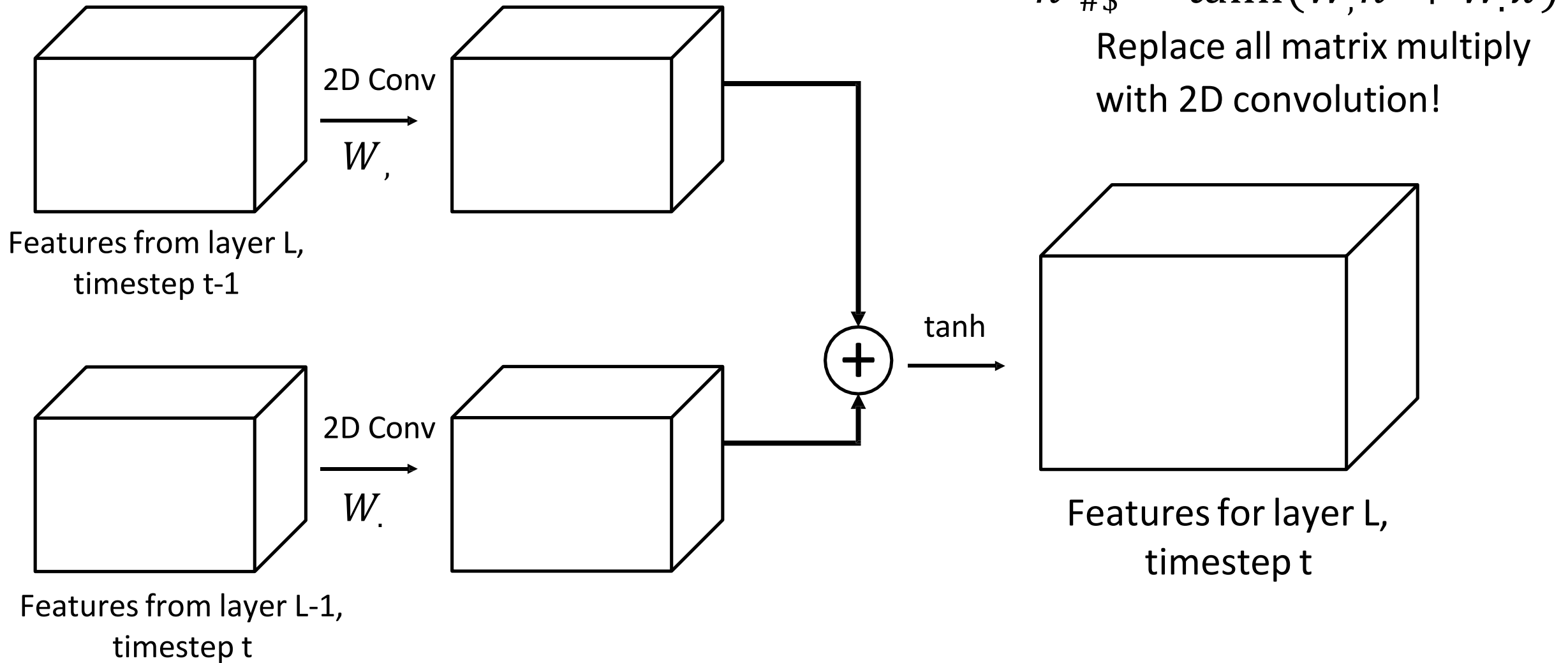
Recall: Vanilla RNN

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

Replace all matrix multiply with 2D convolution!
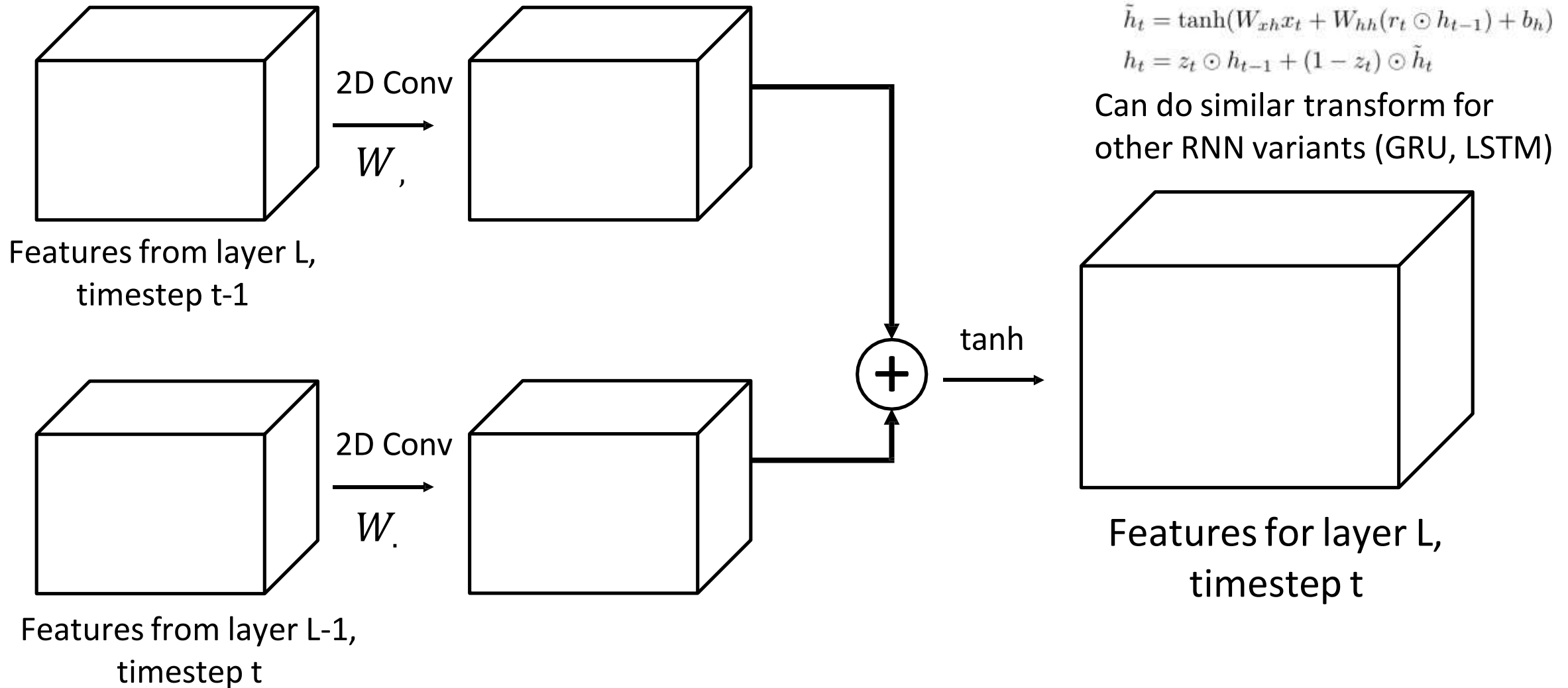
2D Conv
W_h

Features from layer L, timestep t-1

2D Conv
W_x

Features from layer L-1, timestep t

$\oplus$ → tanh →

Features for layer L, timestep t

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

Recurrent Convolutional Network

Features from layer L,
timestep t-1

2D Conv
$W_,$

2D Conv
$W_.$

Features from layer L-1,
timestep t

tanh

Features for layer L,
timestep t

Recall: GRU

$$r_t = \sigma(W_{xr}x_t + W_{hr}h_{t-1} + b_r)$$
$$z_t = \sigma(W_{xz}x_t + W_{hz}h_{t-1} + b_z)$$
$$\tilde{h}_t = \tanh(W_{xh}x_t + W_{hh}(r_t \odot h_{t-1}) + b_h)$$
$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t$$

Can do similar transform for
other RNN variants (GRU, LSTM)

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

Modeling long-term temporal structure

RNN: Infinite temporal extent (fully-connected)

CNN: finite temporal extent (convolutional)

Time

Recurrent CNN: Infinite temporal extent (convolutional)
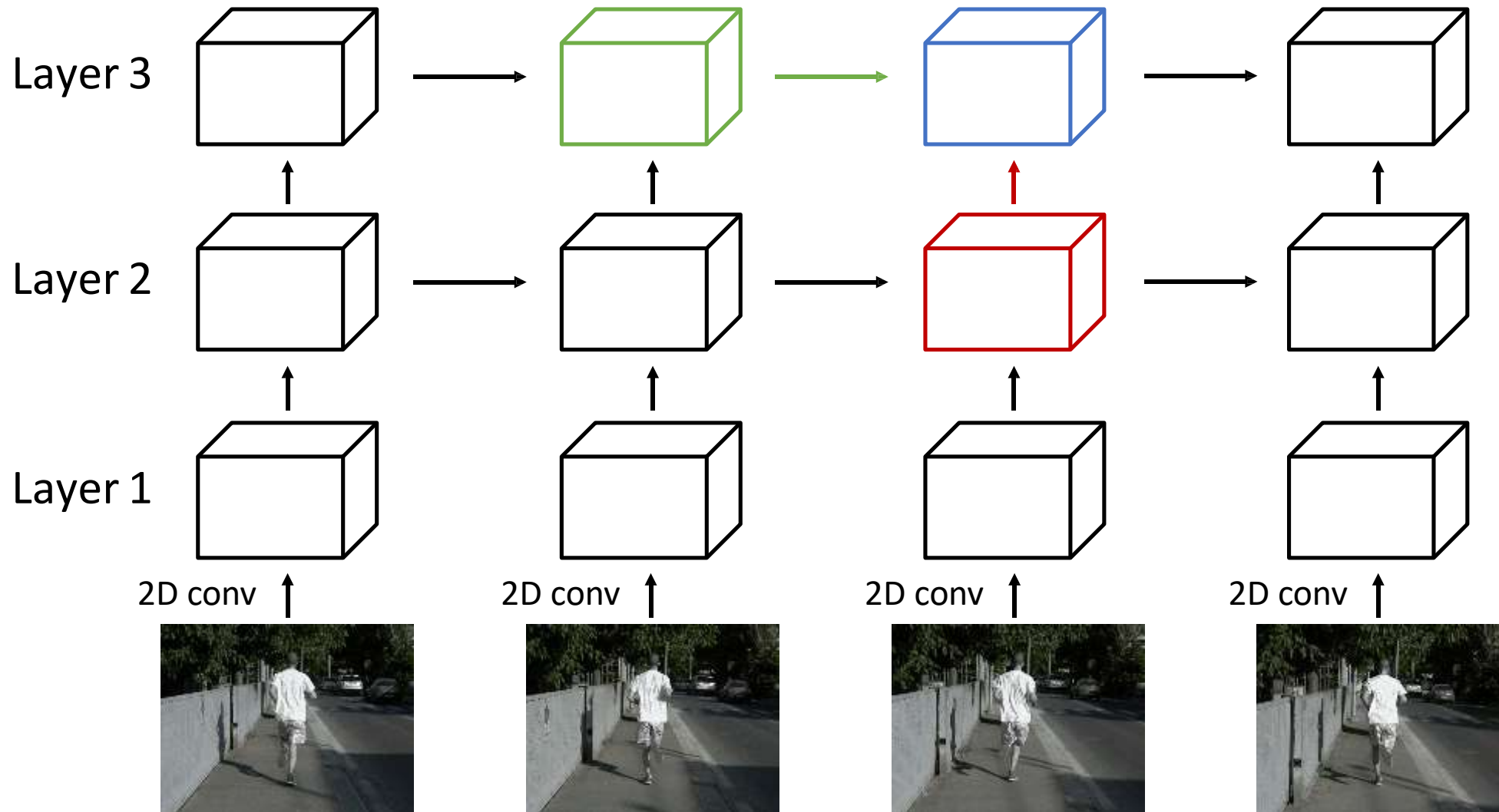
Time

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

Modeling long-term temporal structure

Problem: RNNs are slow for long sequences (can't be parallelized)
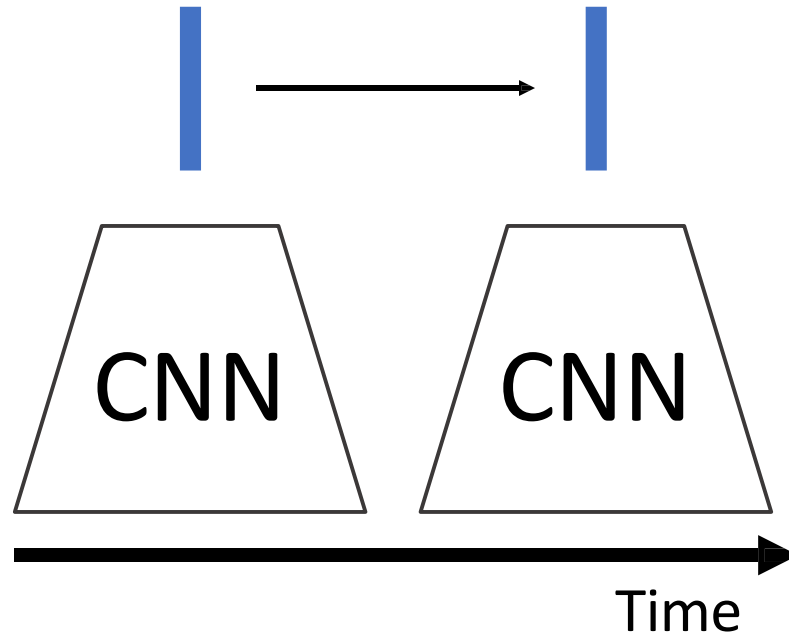
RNN: Infinite temporal extent (fully-connected)

CNN: finite temporal extent (convolutional)

Time

Recurrent CNN: Infinite temporal extent (convolutional)

Recurrent CNN → Recurrent CNN
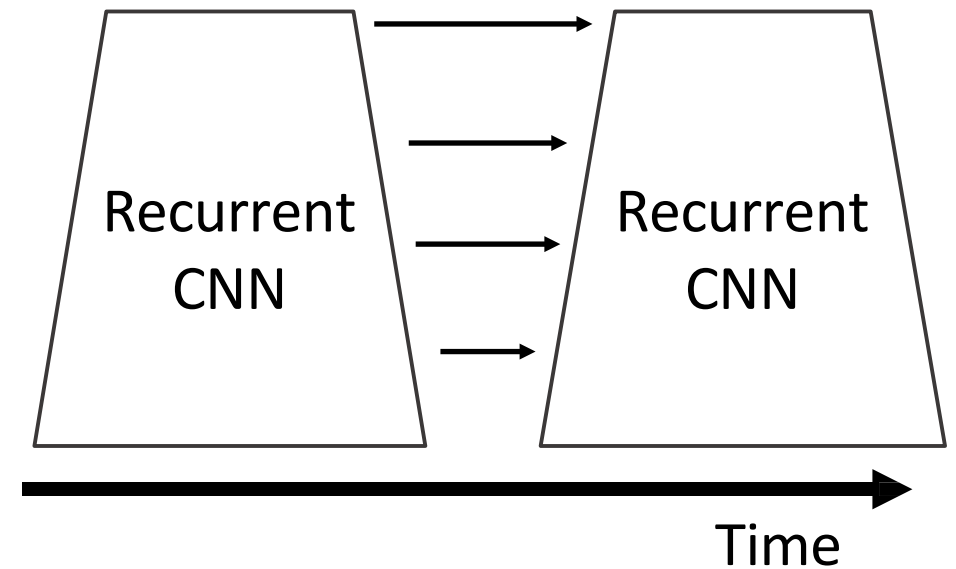
Time

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

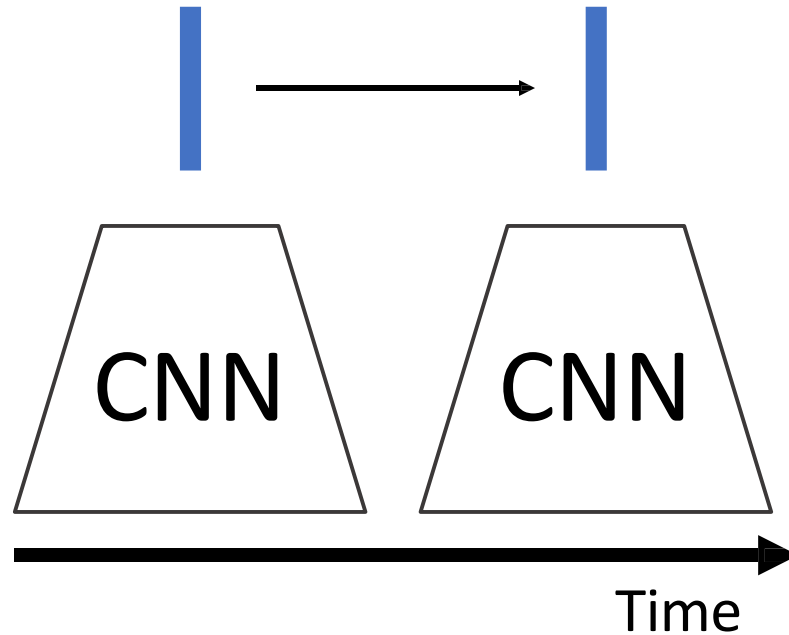# Recall: Different ways of processing sequences
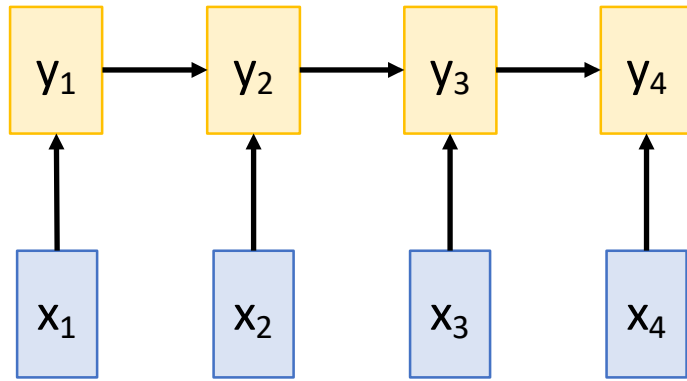
Recurrent Neural Network

Works on **Ordered Sequences**
(+) Good at long sequences: After one RNN layer, h_T "sees" the whole sequence
(-) Not parallelizable: need to compute hidden states sequentially
In video: CNN+RNN, or recurrent CNN
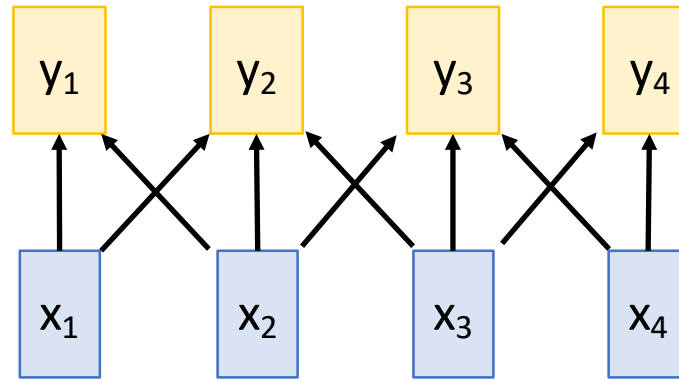
1D Convolution

Works on **Multidimensional Grids**
(-) Bad at long sequences: Need to stack many conv layers for outputs to "see" the whole sequence
(+) Highly parallel: Each output can be computed in parallel
In video: 3D convolution

Recall: Different ways of processing sequences
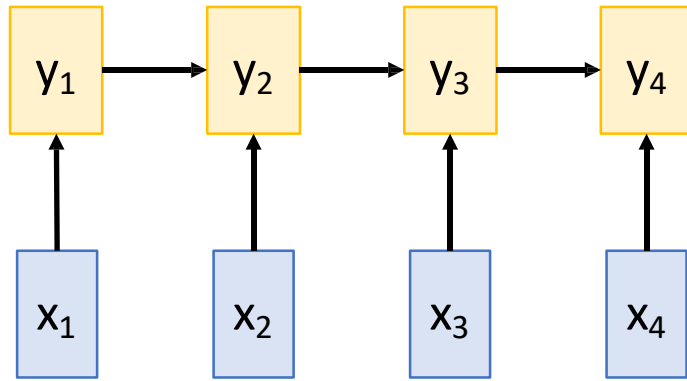
Recurrent Neural Network

Works on **Ordered Sequences**
(+) Good at long sequences: After one RNN layer, h_T "sees" the whole sequence
(-) Not parallelizable: need to compute hidden states sequentially
In video: CNN+RNN, or recurrent CNN
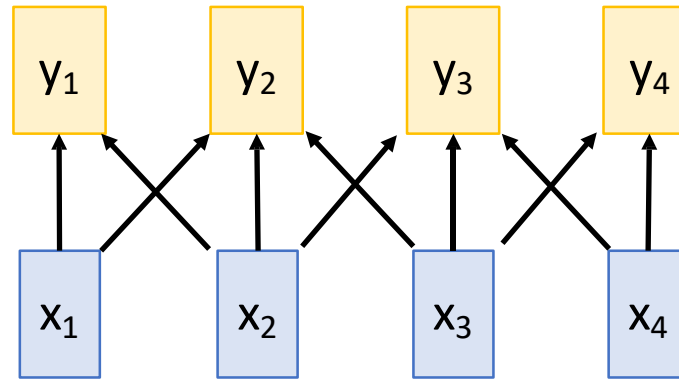
1D Convolution

Works on **Multidimensional Grids**
(-) Bad at long sequences: Need to stack many conv layers for outputs to "see" the whole sequence
(+) Highly parallel: Each output can be computed in parallel
In video: 3D convolution

Self-Attention
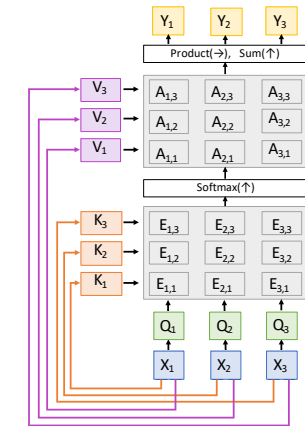
Works on **Sets of Vectors**
(-) Good at long sequences: after one self-attention layer, each output "sees" all inputs!
(+) Highly parallel: Each output can be computed in parallel
(-) Very memory intensive
In video: ????

Recall: Self-Attention

Input: Set of vectors $x_1, ..., x_N$

Keys, Queries, Values: Project each x to a key, query, and value using linear layer

Affinity matrix: Compare each pair of x, (using scaled dot-product between keys and values) and normalize using softmax

Output: Weighted sum of values, with weights given by affinity matrix

Features in 3D CNN: C x T x H x W Interpret as a set of THW vectors of dim C
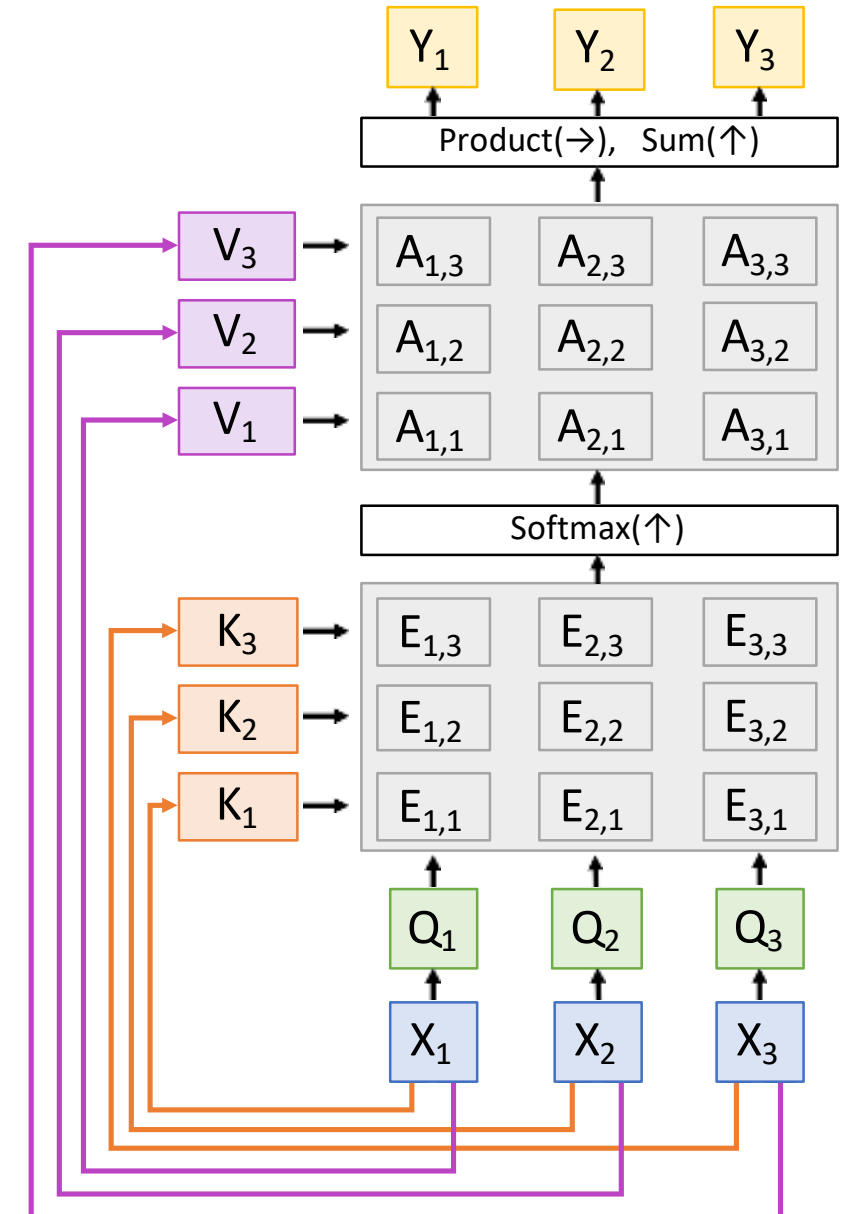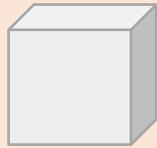
Vaswani et al, "Attention is all you need", NeurIPS 2017

Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip

3D CNN

Features:
C x T x H x W

Nonlocal Block

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

Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip

3D CNN

Features:
C x T x H x W

Queries:
C' x T x H x W

1x1x1 Conv

Keys:
C' x T x H x W

1x1x1 Conv

Values:
C' x T x H x W

1x1x1 Conv

Nonlocal Block

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

Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip

3D CNN

Nonlocal Block

Features: C x T x H x W

Queries: C' x T x H x W

1x1x1 Conv

Keys: C' x T x H x W

1x1x1 Conv

Values: C' x T x H x W

1x1x1 Conv

Transpose

X

softmax

Attention Weights (THW) x (THW)

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

Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip

3D CNN

Features:
C x T x H x W

Queries:
C' x T x H x W

1x1x1 Conv

Transpose

Keys:
C' x T x H x W

1x1x1 Conv

Values:
C' x T x H x W

1x1x1 Conv

Attention Weights
(THW) x (THW)

softmax

C' x T x H x W

Nonlocal Block

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

Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip

3D CNN

Features:
C x T x H x W

Queries:
C' x T x H x W

1x1x1 Conv

Transpose

Attention Weights
(THW) x (THW)

X softmax

Keys:
C' x T x H x W

1x1x1 Conv

C x T x H x W

Values:
C' x T x H x W

1x1x1 Conv

C' x T x H x W

X

1x1x1 Conv

Nonlocal Block

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

Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip

3D CNN

Features:
C x T x H x W

Queries:
C' x T x H x W

1x1x1 Conv

Transpose

Keys:
C' x T x H x W

1x1x1 Conv

Values:
C' x T x H x W

1x1x1 Conv

X

softmax

Attention Weights
(THW) x (THW)

X

C' x T x H x W

1x1x1 Conv

C x T x H x W

+

Residual Connection

Nonlocal Block

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

Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip

3D CNN

Features: C x T x H x W

Queries: C' x T x H x W
1x1x1 Conv

Keys: C' x T x H x W
1x1x1 Conv

Values: C' x T x H x W
1x1x1 Conv

Transpose

Residual Connection

Attention Weights (THW) x (THW)

softmax

C' x T x H x W
1x1x1 Conv

C x T x H x W

Nonlocal Block

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

Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip

3D CNN

Residual Connection

Queries:
C' x T x H x W

1x1x1 Conv

Transpose

Attention Weights
(THW) x (THW)

softmax

X

Keys:
C' x T x H x W

1x1x1 Conv

Features:
C x T x H x W

C x T x H x W

Values:
C' x T x H x W

1x1x1 Conv

C' x T x H x W

X

1x1x1 Conv

+

Nonlocal Block

Trick: Initialize **last conv** to 0, then entire block computes identity. Can insert into existing 3D CNNs

In practice, actually insert BatchNorm layer after final conv, and initialize scale parameter of BN layer to 0 rather than setting conv weight to 0

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

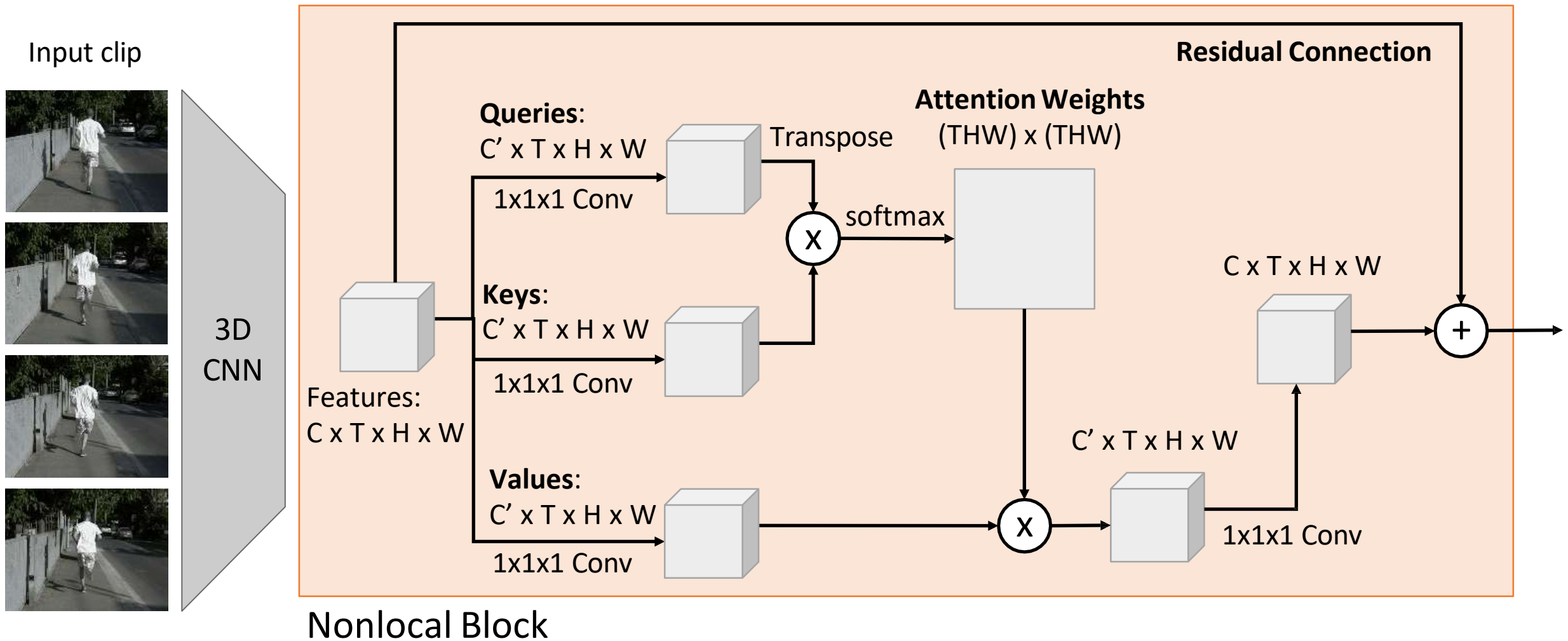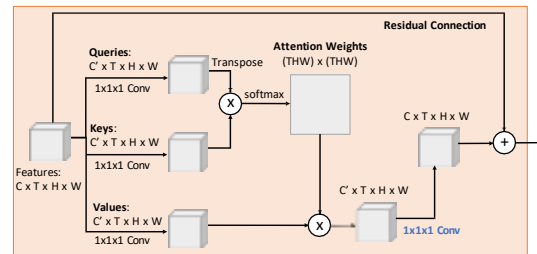# Spatio-Temporal Self-Attention (Nonlocal Block)

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
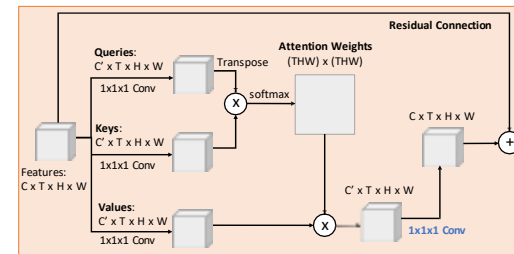
Nonlocal Block

3D CNN

Running

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

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 (InceptionNet).

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

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

Inflating 2D Networks to 3D (I3D)

There has been a lot of work on architectures for images.
Can we reuse image architectures for video?
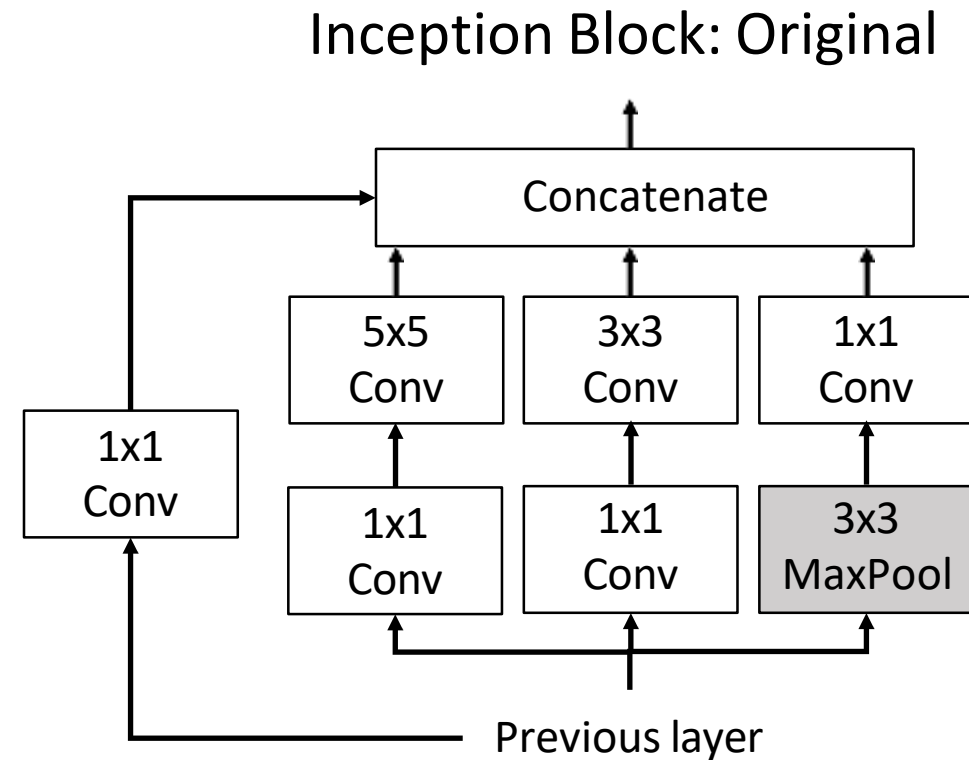
Idea: take a 2D CNN architecture.

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

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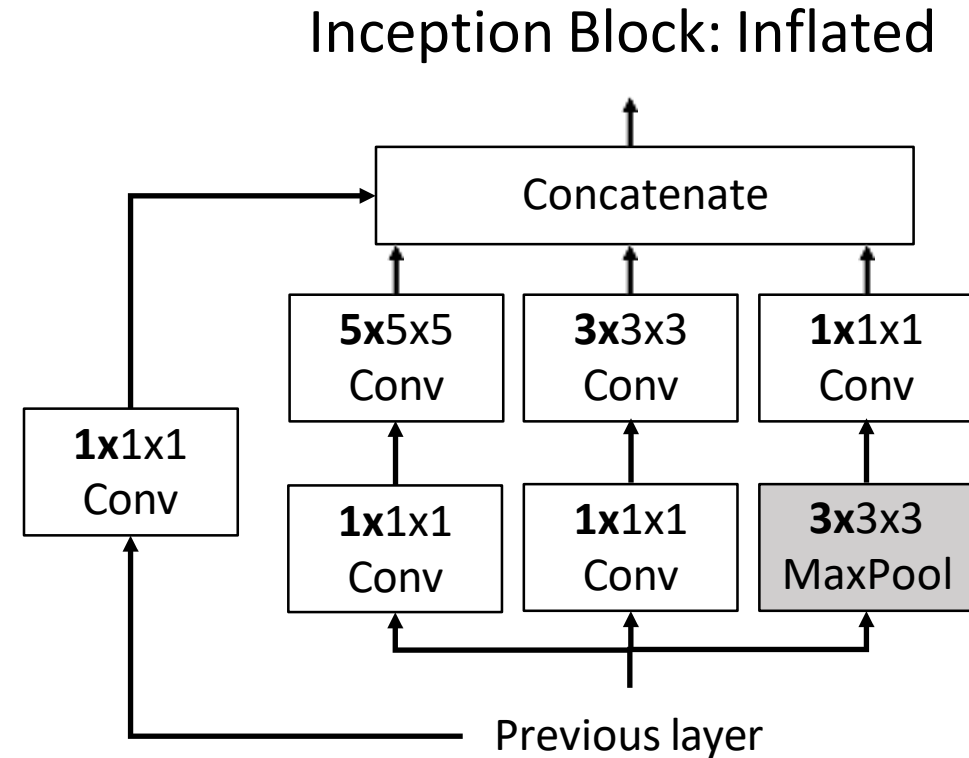
Can use weights of 2D conv to initialize 3D conv: copy K_t times in space and divide by K_t
This gives the same result as 2D conv given "constant" video input

Input:
3 x H x W

2D conv kernel:
C_{in} x K_h x K_w

Output:
H x W

Duplicate input K_t times

Copy kernel K_t times, divide by K_t

Output is the same!

Input:
3 x K_t x H x W

3D conv kernel:
C_{in} x K_t x K_h x K_w

Output:
1 x H x W

Carreira and Zisserman, "Quo Vadis, Action Recognition? A New Model and the Kinetics Dataset", CVPR 2017
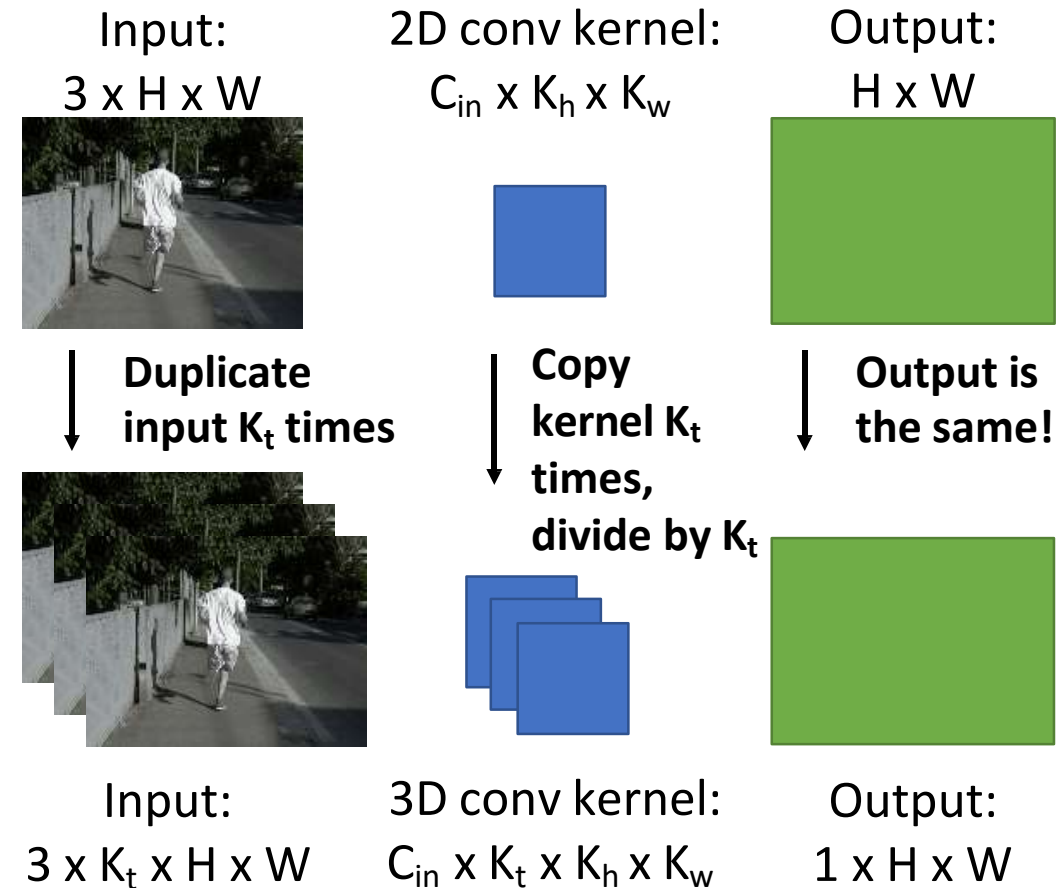
Inflating 2D Networks to 3D (I3D)

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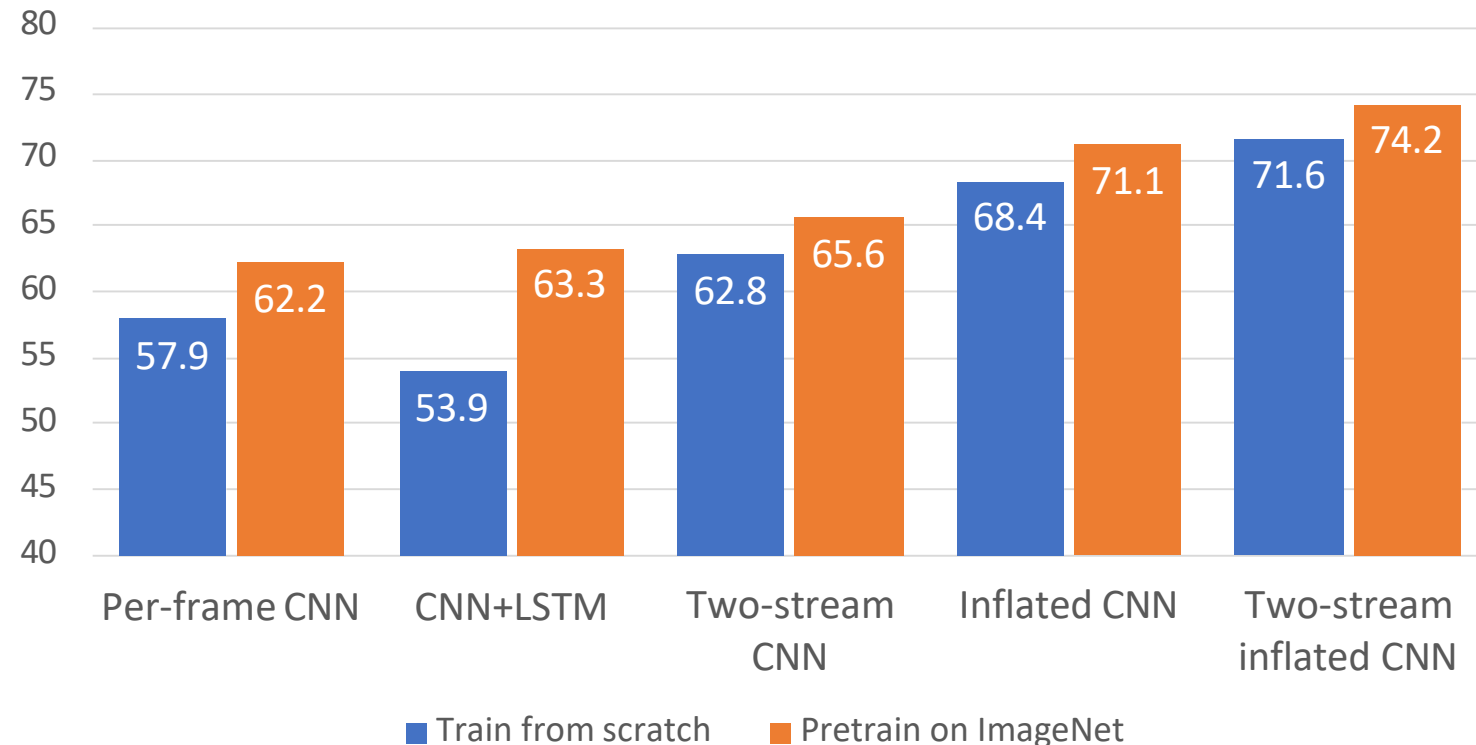Can we reuse image architectures for video?

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

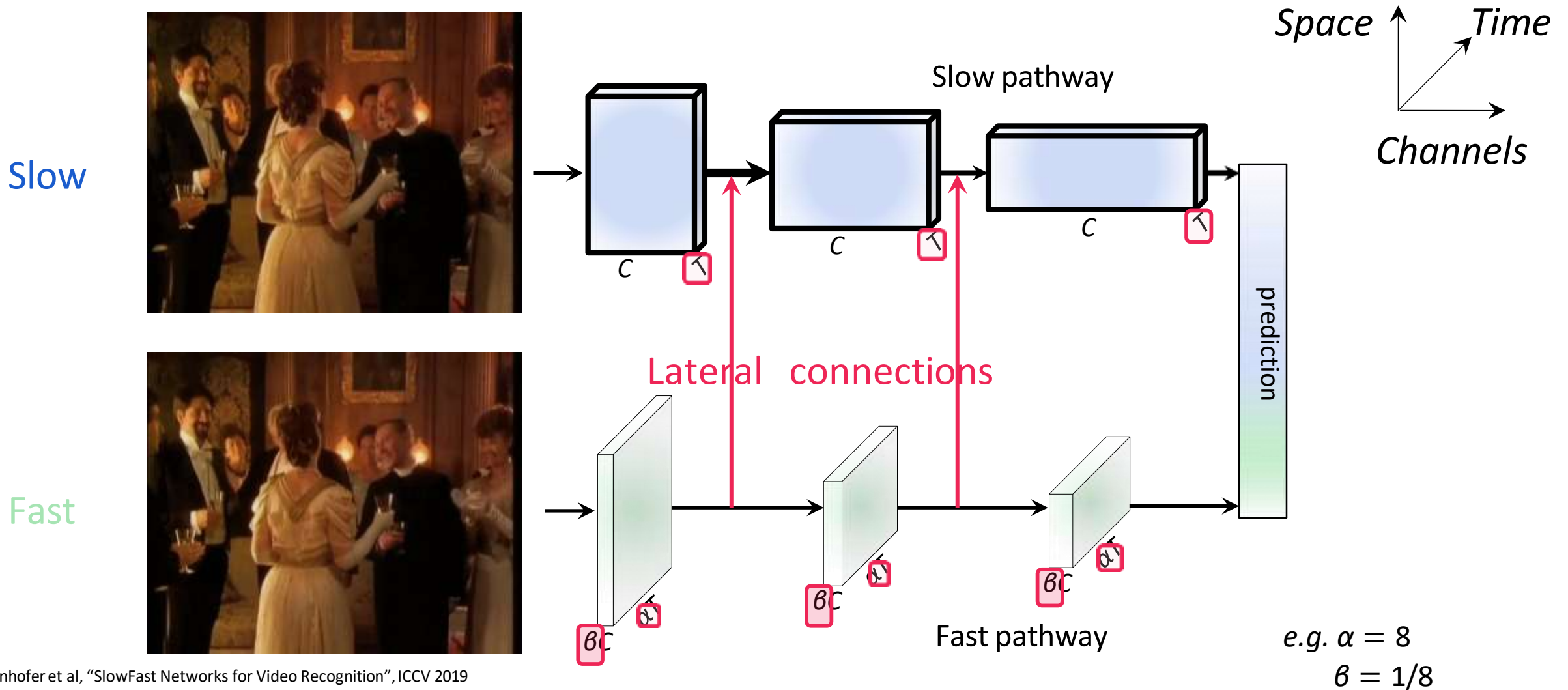Can use weights of 2D conv to initialize 3D conv: copy K_t times in space and divide by K_t
This gives the same result as 2D conv given "constant" video input

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

Top-1 Accuracy on Kinetics-400

	Train from scratch	Pretrain on ImageNet
Per-frame CNN	57.9	62.2
CNN+LSTM	53.9	63.3
Two-stream CNN	62.8	65.6
Inflated CNN	68.4	71.1
Two-stream inflated CNN	71.6	74.2

All using Inception CNN

Treating time and space differently: SlowFast Networks

Slow

Fast

Slow pathway

Space ↗ *Time*

Channels

Lateral connections

Fast pathway

prediction

c τ

c τ

c τ

βc $\alpha \tau$

βc $\alpha \tau$

βc $\alpha \tau$

e.g. $\alpha = 8$

$\beta = 1/8$

Feichtenhofer et al, "SlowFast Networks for Video Recognition", ICCV 2019
Slide credit: Christoph Feichtenhofer

Treating time and space differently: SlowFast Networks

- Dimensions are $\{T \times S^2, C\}$
- Strides are {temporal, spatial^2}
- The backbone is ResNet-50
- Residual blocks are shown by brackets
- Non-degenerate temporal filters are underlined
- Here the speed ratio is $\alpha = 8$ and the channel ratio is $\beta = 1/8$
- Orange numbers mark fewer channels, for the Fast pathway
- Green numbers mark higher temporal resolution of the Fast pathway
- No temporal *pooling* is performed throughout the hierarchy

stage	Slow pathway	Fast pathway	output sizes $T \times S^2$
raw clip	-	-	64×224^2
data layer	stride 16, 1^2	stride **2**, 1^2	Slow : 4×224^2 Fast : 32×224^2
conv$_1$	1×7^2, 64 stride 1, 2^2	$\underline{5 \times 7^2}$, 8 stride 1, 2^2	Slow : 4×112^2 Fast : 32×112^2
pool$_1$	1×3^2 max stride 1, 2^2	1×3^2 max stride 1, 2^2	Slow : 4×56^2 Fast : 32×56^2
res$_2$	$\begin{bmatrix} 1 \times 1^2, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} \underline{3 \times 1^2}, 8 \\ 1 \times 3^2, 8 \\ 1 \times 1^2, 32 \end{bmatrix} \times 3$	Slow : 4×56^2 Fast : 32×56^2
res$_3$	$\begin{bmatrix} 1 \times 1^2, 128 \\ 1 \times 3^2, 128 \\ 1 \times 1^2, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} \underline{3 \times 1^2}, 16 \\ 1 \times 3^2, 16 \\ 1 \times 1^2, 64 \end{bmatrix} \times 4$	Slow : 4×28^2 Fast : 32×28^2
res$_4$	$\begin{bmatrix} \underline{3 \times 1^2}, 256 \\ 1 \times 3^2, 256 \\ 1 \times 1^2, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} \underline{3 \times 1^2}, 32 \\ 1 \times 3^2, 32 \\ 1 \times 1^2, 128 \end{bmatrix} \times 6$	Slow : 4×14^2 Fast : 32×14^2
res$_5$	$\begin{bmatrix} \underline{3 \times 1^2}, 512 \\ 1 \times 3^2, 512 \\ 1 \times 1^2, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} \underline{3 \times 1^2}, 64 \\ 1 \times 3^2, 64 \\ 1 \times 1^2, 256 \end{bmatrix} \times 3$	Slow : 4×7^2 Fast : 32×7^2
global average pool, concate, fc			# classes

Feichtenhofer et al, "SlowFast Networks for Video Recognition", ICCV 2019
Slide credit: Christoph Feichtenhofer

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**

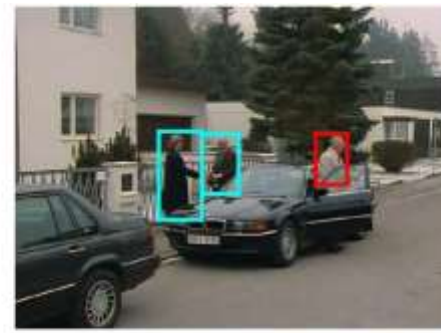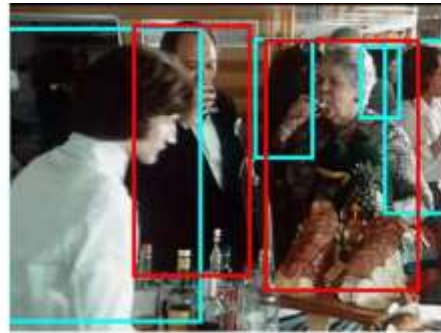Chao et al, " Rethinking the Faster R-CNN Architecture for Temporal Action Localization", CVPR 2018

Spatio-Temporal Detection

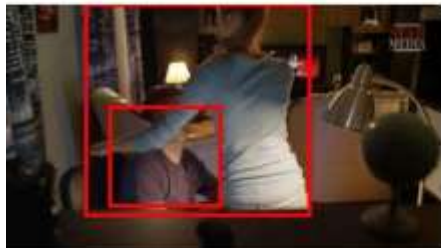Given a long untrimmed video, detect all the people in space and time and classify the activities they are performing
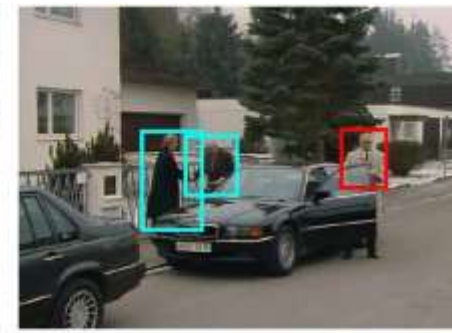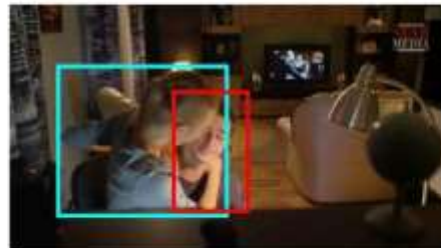Some examples from AVA Dataset:
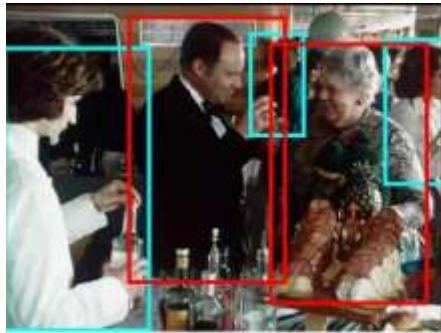
clink glass → drink

open → close

grab (a person) → hug

look at phone → answer phone

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

Recap: Video Models

Many video models:
Single-frame CNN (Try this first!)
Late fusion
Early fusion
3D CNN / C3D
Two-stream networks
CNN + RNN
Convolutional RNN
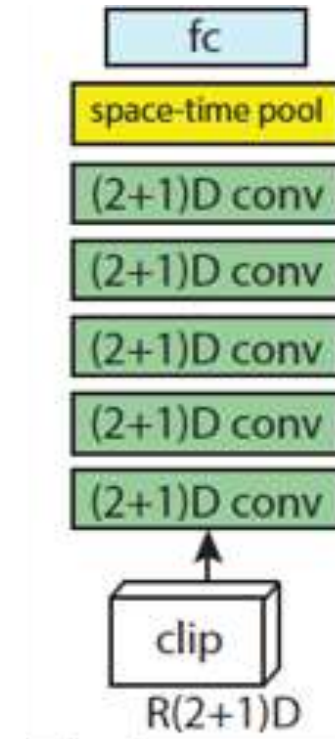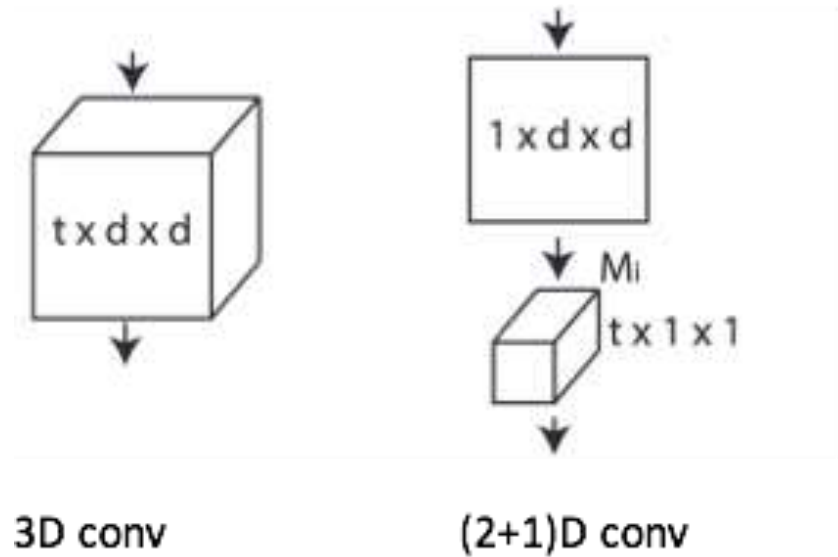Spatio-temporal self-attention
SlowFast networks (current SoTA)

Next time:
Generative Models, part 1
Generative Adversarial Networks

R(2+1)D
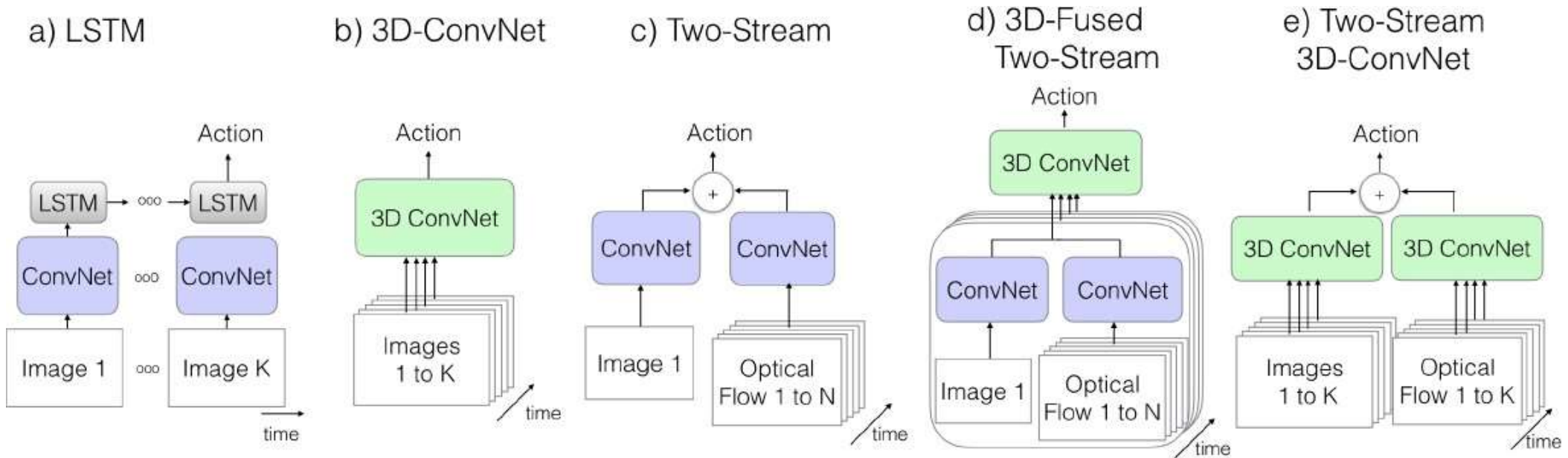

3D conv   (2+1)D conv   R(2+1)D

Practical aspects

- Pretrain on large datasets
    - ImageNet, Kinetics, Sports1M, Youtube8M

- Finetune of the target dataset

- Loading image weights for video
    - ImageNet

Video classification variants

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