3D Computer Vision

Video Understanding

Emanuele Colonna — 20/05/2025

Who am I?

colonnaemanuele.github.io

Emanuele Colonna - PhD Student

Recall - Image Task

Classification

CAT

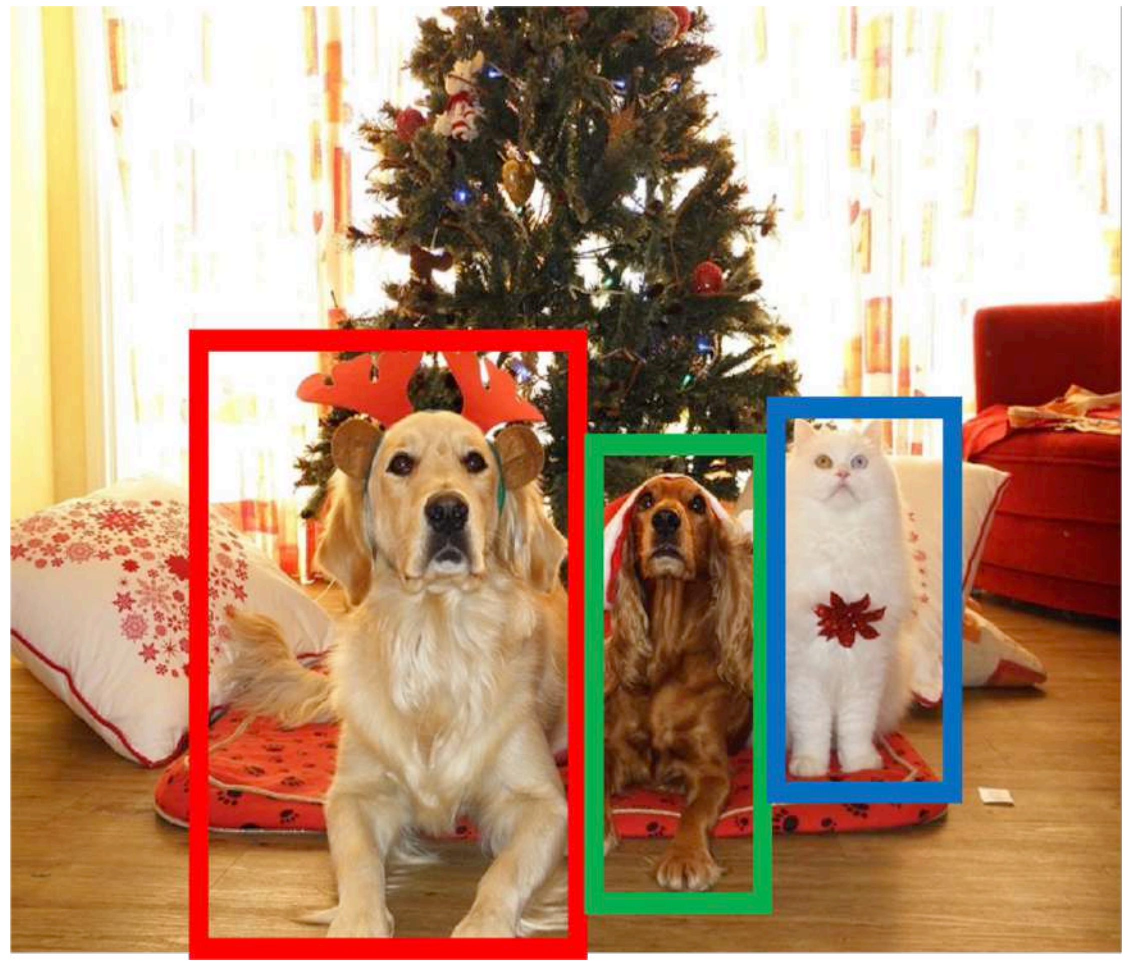
No spacial extent

Semantic Segmentation

GRASS, CAT, TREE, SKY

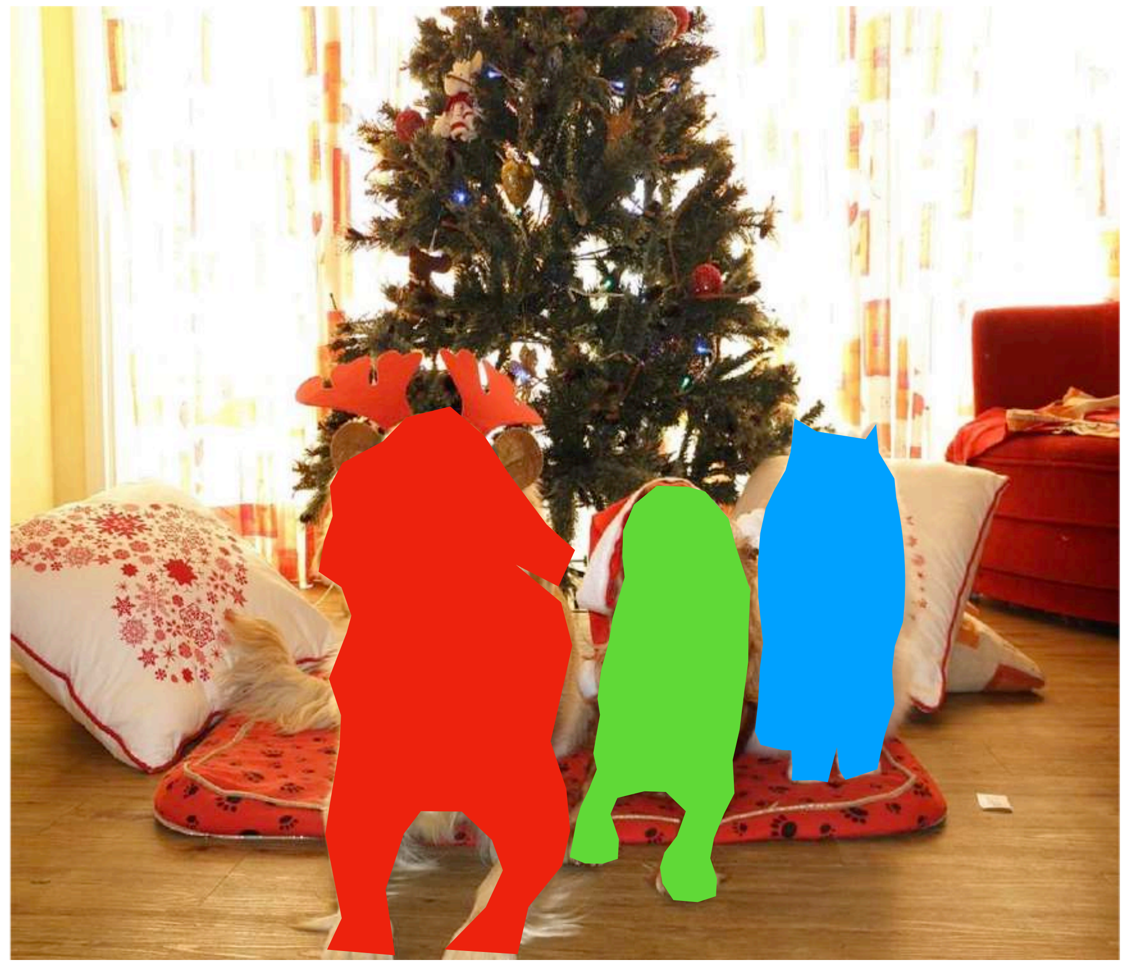
No object, just pixels

Object Detection



DOG, DOG, CAT

Instance Segmentation



DOG, DOG, CAT

Multiple Objects

Videos

Image with Time

A video is a sequence of images

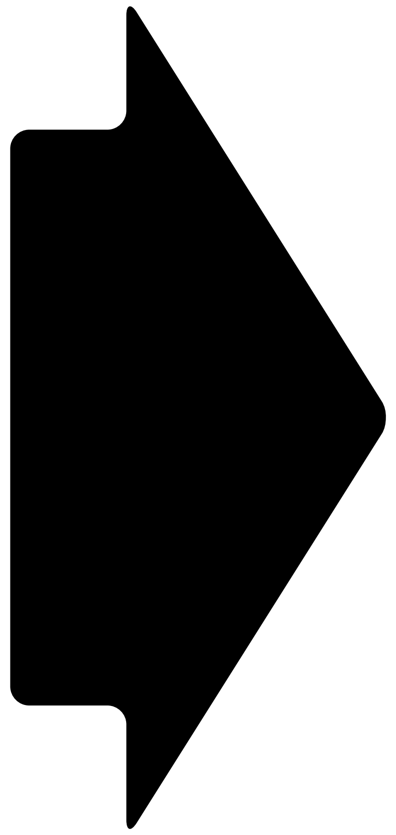
4D Tensor: T x 3 x H x W

Example Task: ?

Input: T x 3 x H x W

Example Task: Video Classification

Input: T x 3 x H x W



Swimming

Running

Jumping

Eating

Problem!

Video are a series of images!

Input: T x 3 x H x W

Video are ~30 frames per second

Size of uncompressed video (3 bytes for pixel)

SD (640x480) -> ~1.5GB/minut
HD (1920x1080) -> ~10GB/minut

Possible Solution?

Solution

Spatio-Temporal reduction

Input: T x 3 x H x W

Video are ~30 frames for second

SD (640x480) -> ~1.5GB/minut
HD (1920x1080) -> ~10GB/minut

Low fps and low spatial resolution

T = 16 H=W=112

3.2 seconds at 5fps -> 588KB

Solution - Train on Clips

Raw video - Long, high FPS

Training - short clip with low FPS

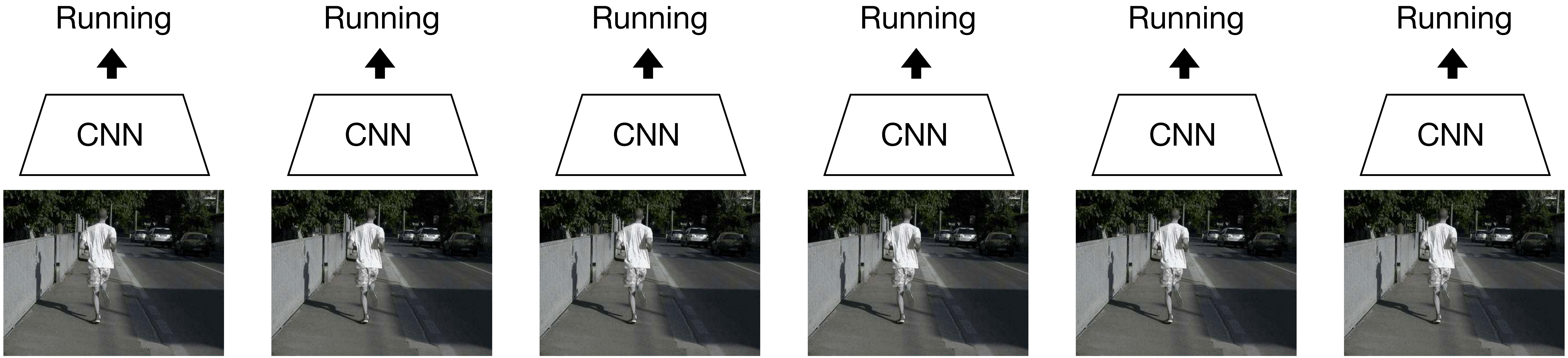
Testing - different clips and average predictions

Single-Frame CNN

Simple Idea -> train 2D CNN to classify each frame video.

Strong Baseline for video classification

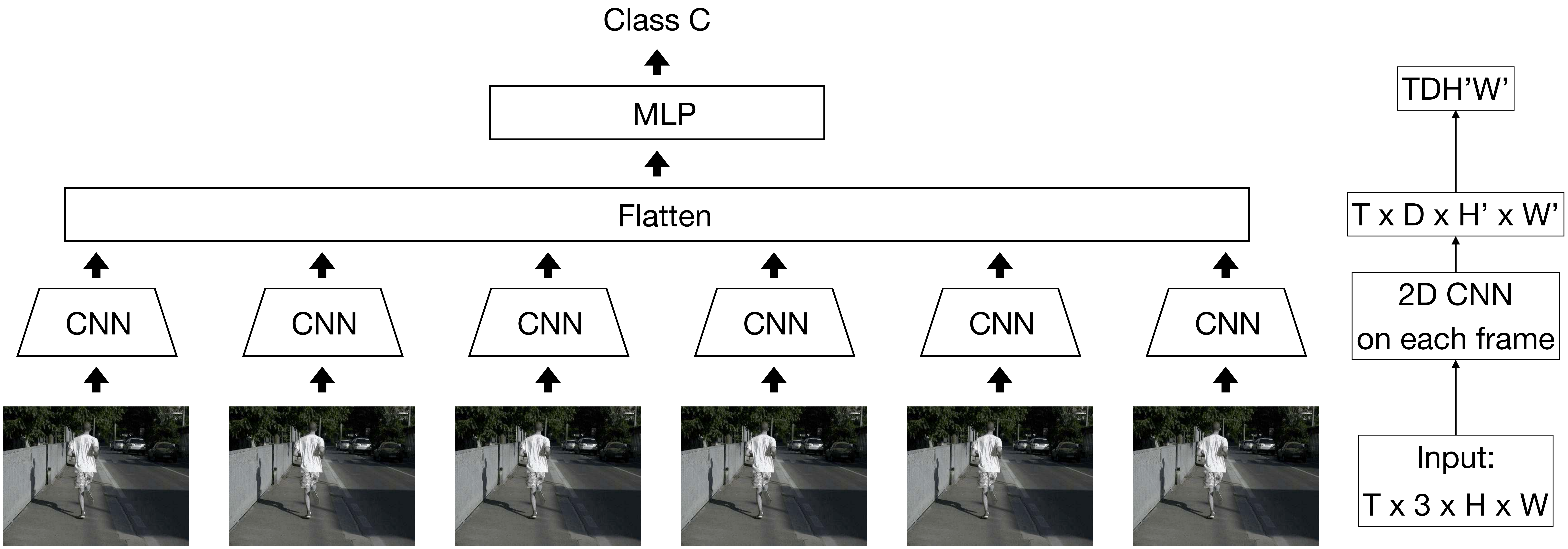
Very computational expensive method



Late Fusion

Get High-Level appearance of each frame, and combine them.

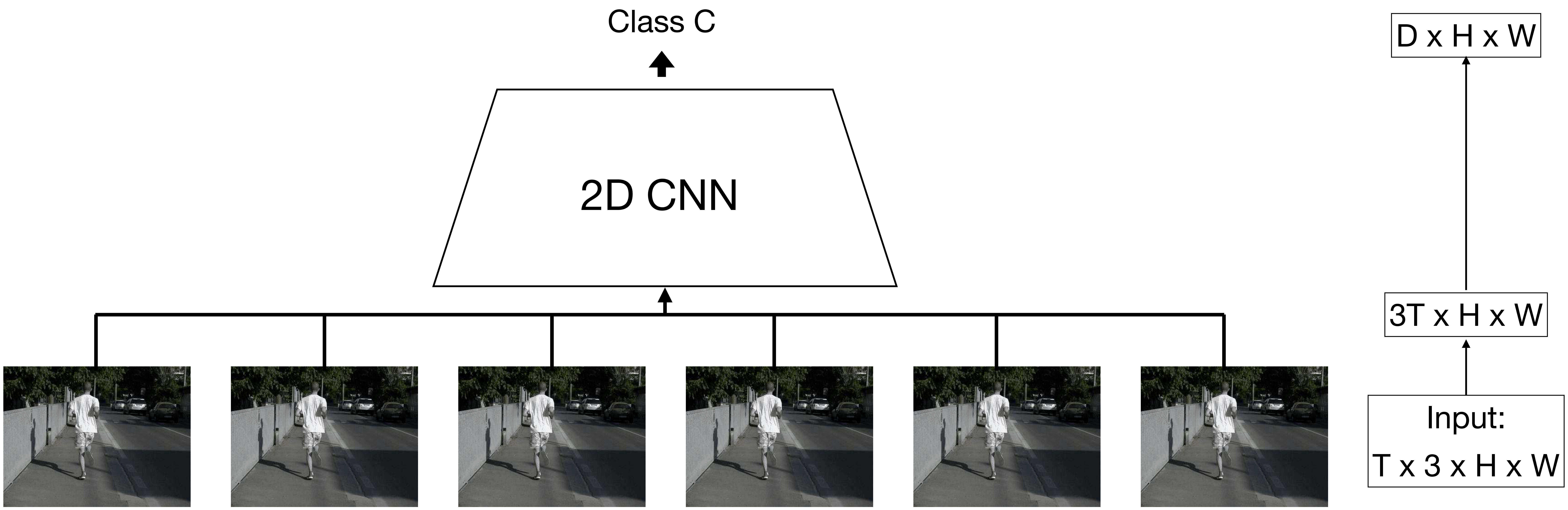
Hard to compare low-level motion between frames



Early Fusion

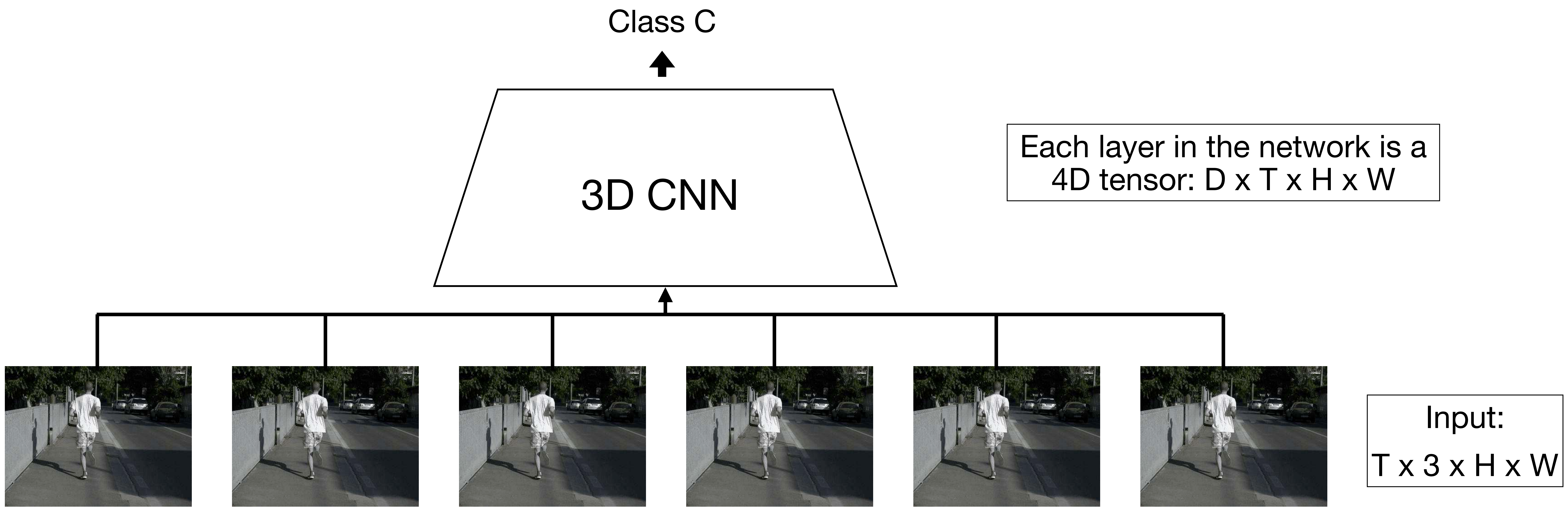
Compare frames with very first conv layer, after that normal 2D CNN

One layer of temporal processing may not be enough



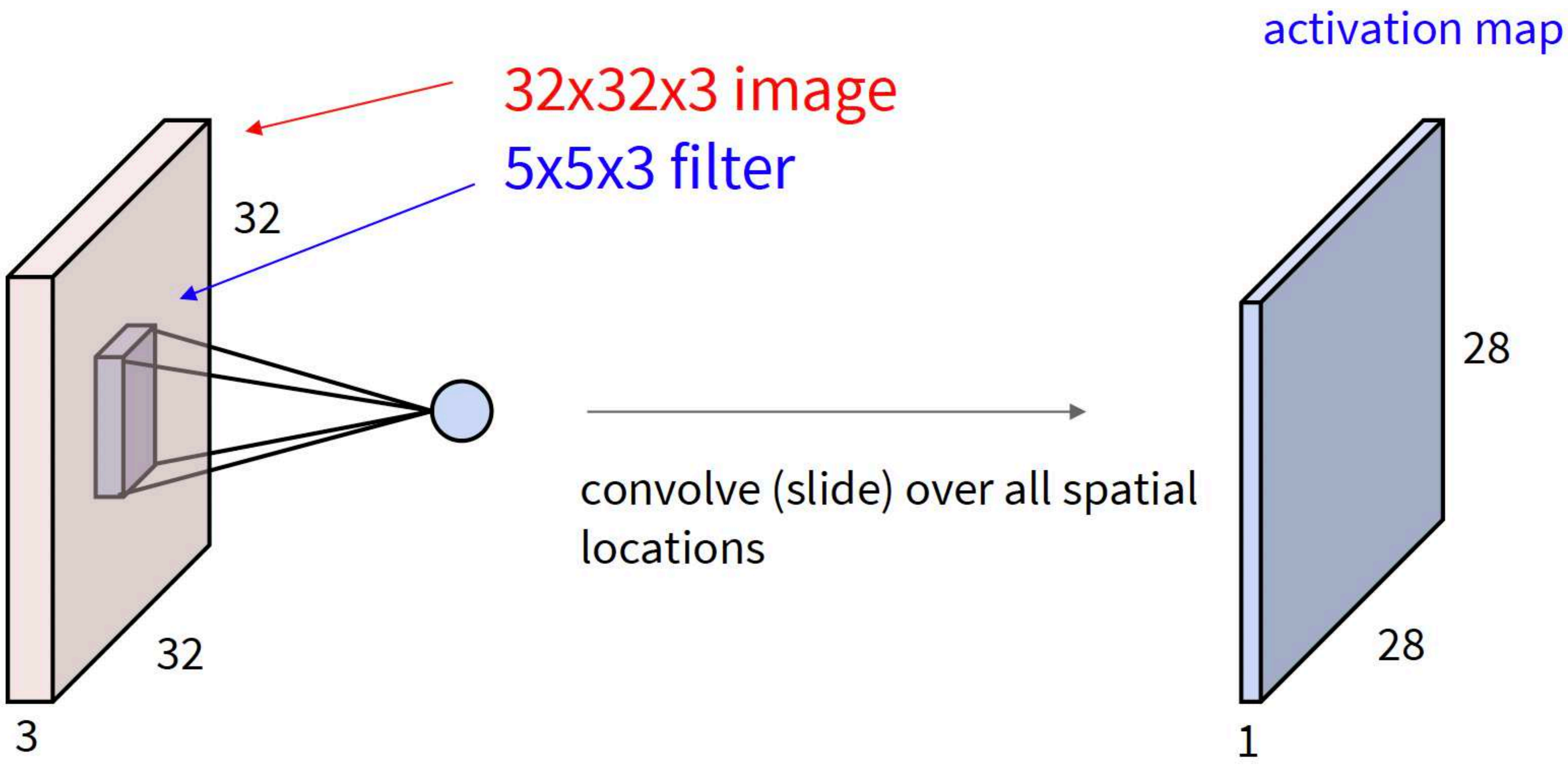
3D CNN

3D versions of conv ad pooling to slowly fuse temporal information over the course of the network



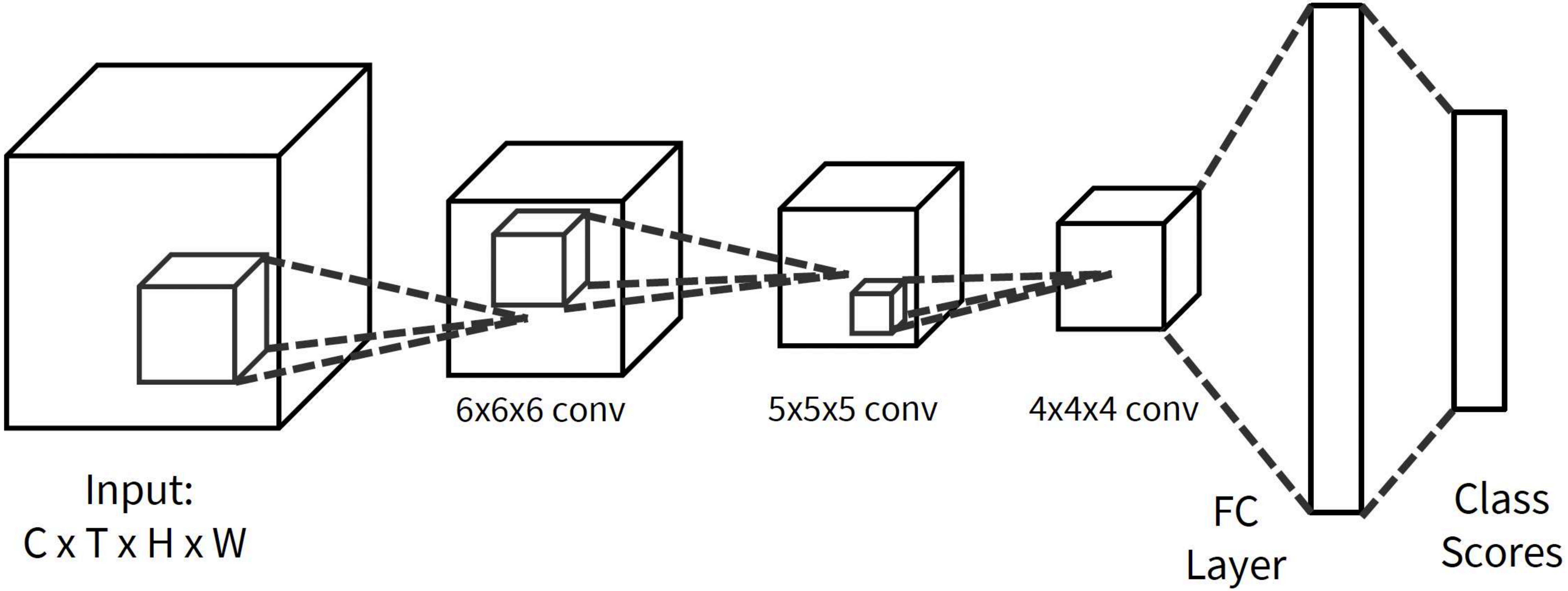
Difference with time

2D Convolution



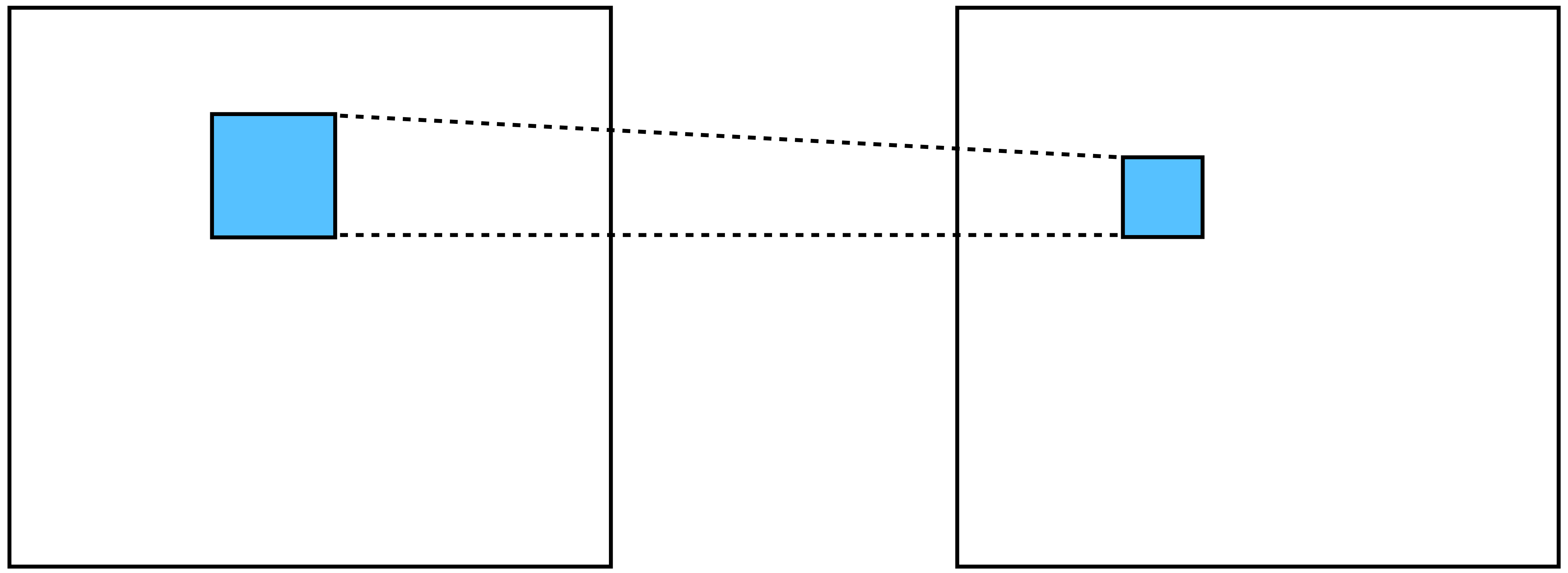
Difference with time

3D Convolution

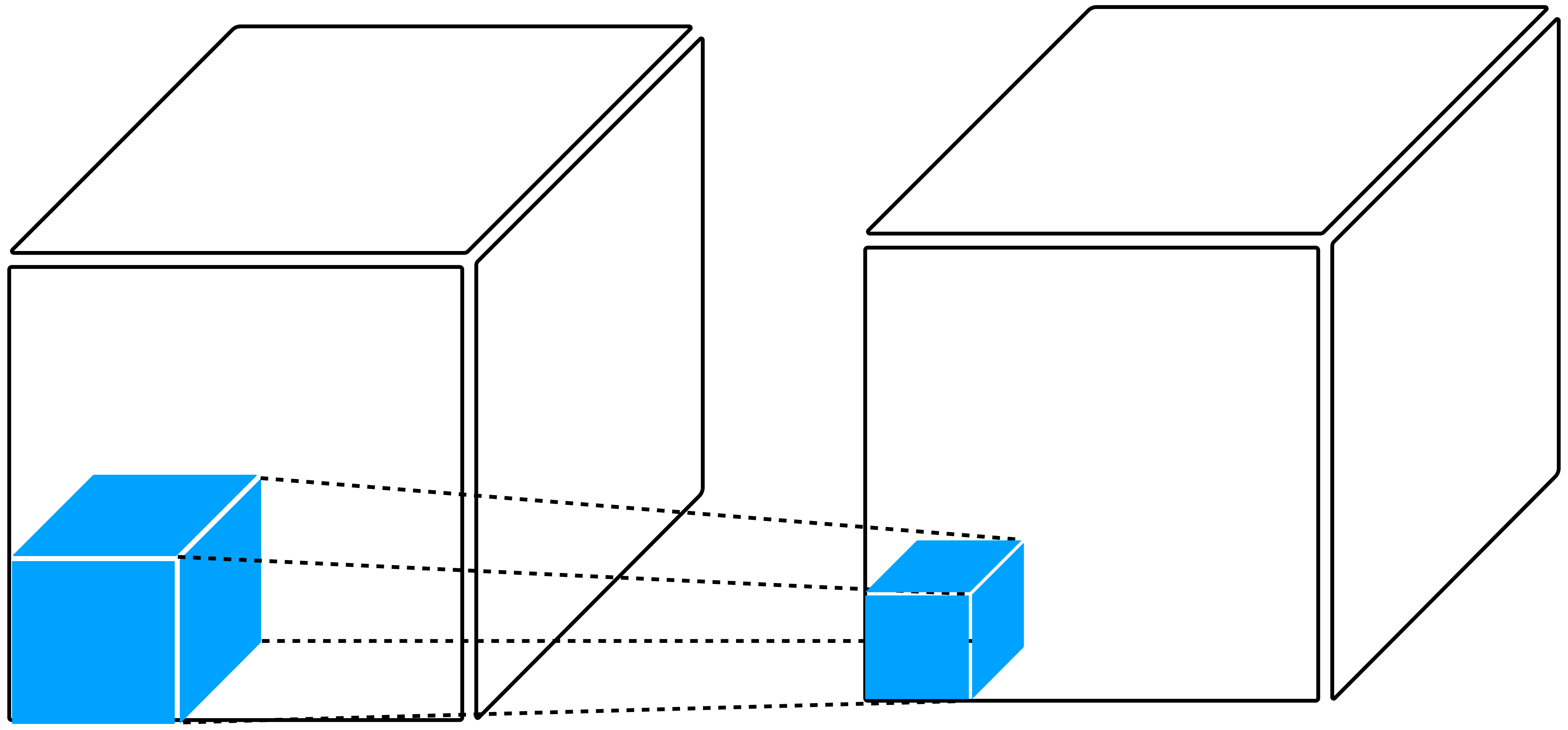


Difference with time

2D Conv vs 3D Conv



2D Convolution

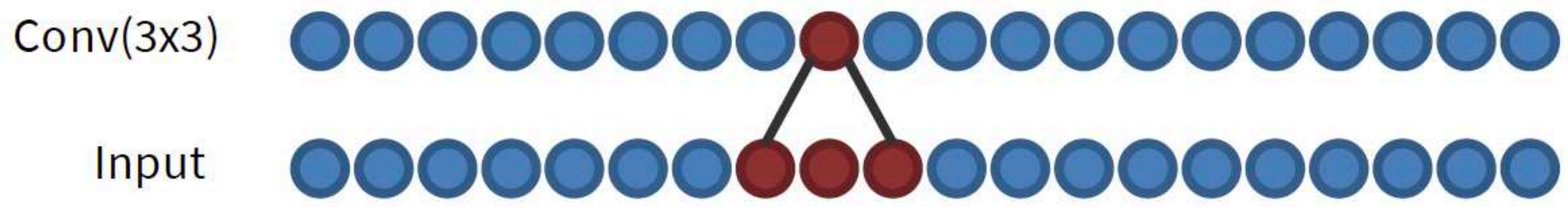


3D Convolution

Early Fusion Vs Late Fusion Vs 3D CNN

Late Fusion

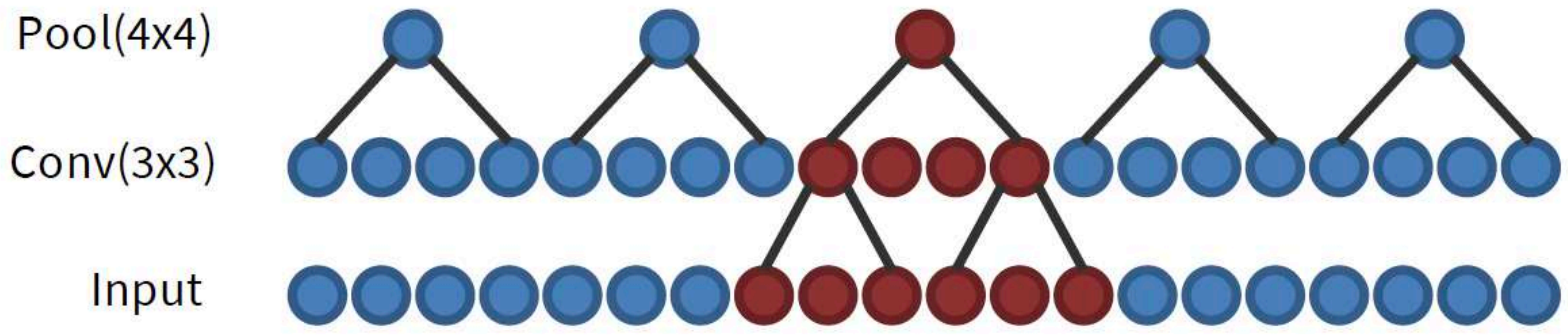
Layer	Size **(C x T x H x W)**	Receptive Field **(T x H x W)**
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3



Early Fusion Vs Late Fusion Vs 3D CNN

Late Fusion

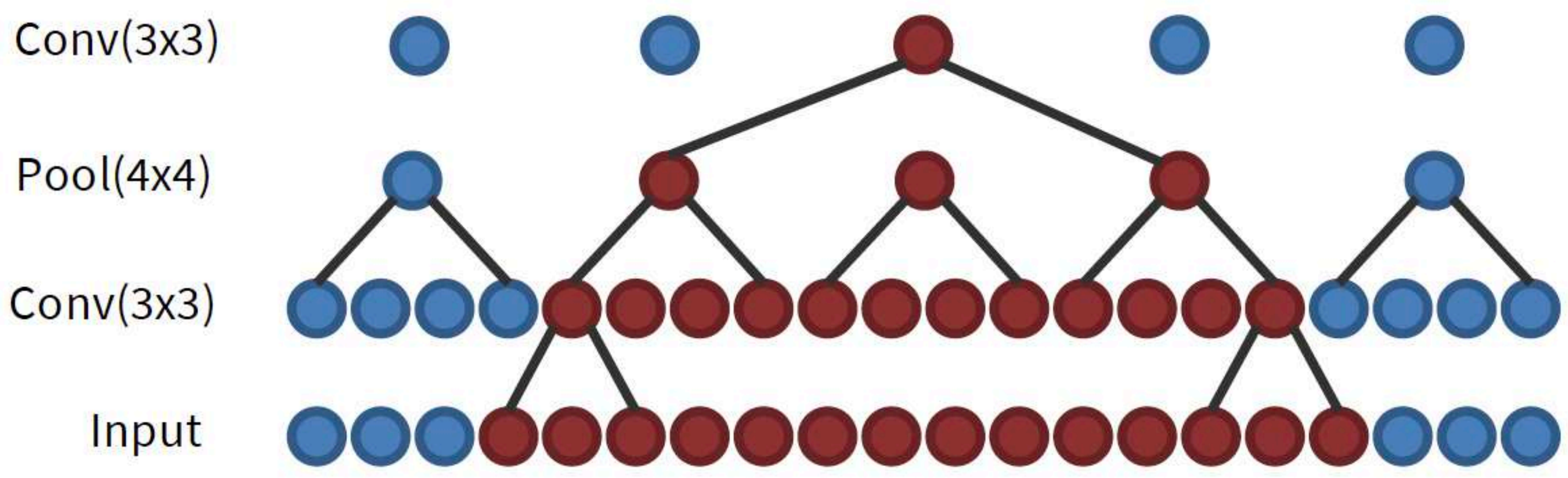
Layer	Size **(C x T x H x W)**	Receptive Field **(T x H x W)**
Input	3 x 20 x 64 x 64	
Conv2D(3x3, 3->12)	12 x 20 x 64 x 64	1 x 3 x 3
Pool2D(4x4)	12 x 20 x 16 x 16	1 x 6 x 6



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

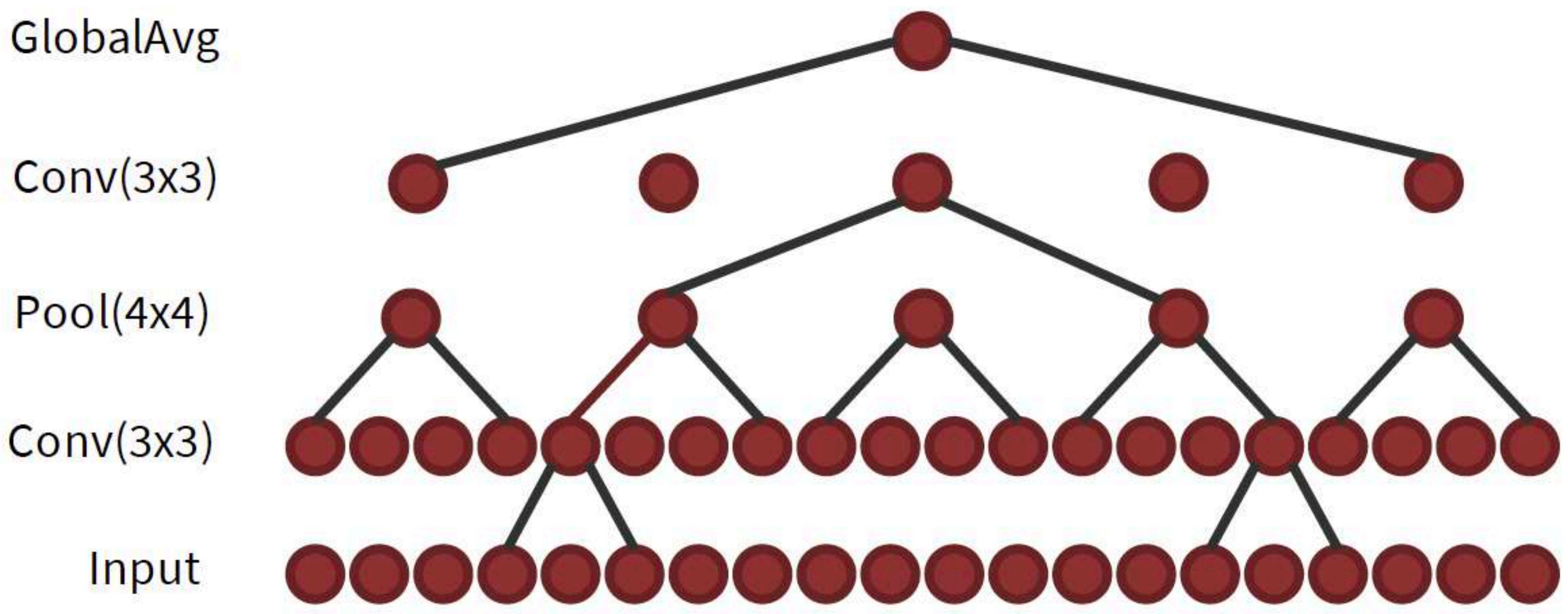


Early Fusion Vs Late Fusion Vs 3D CNN

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 Vs Late Fusion Vs 3D CNN

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

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

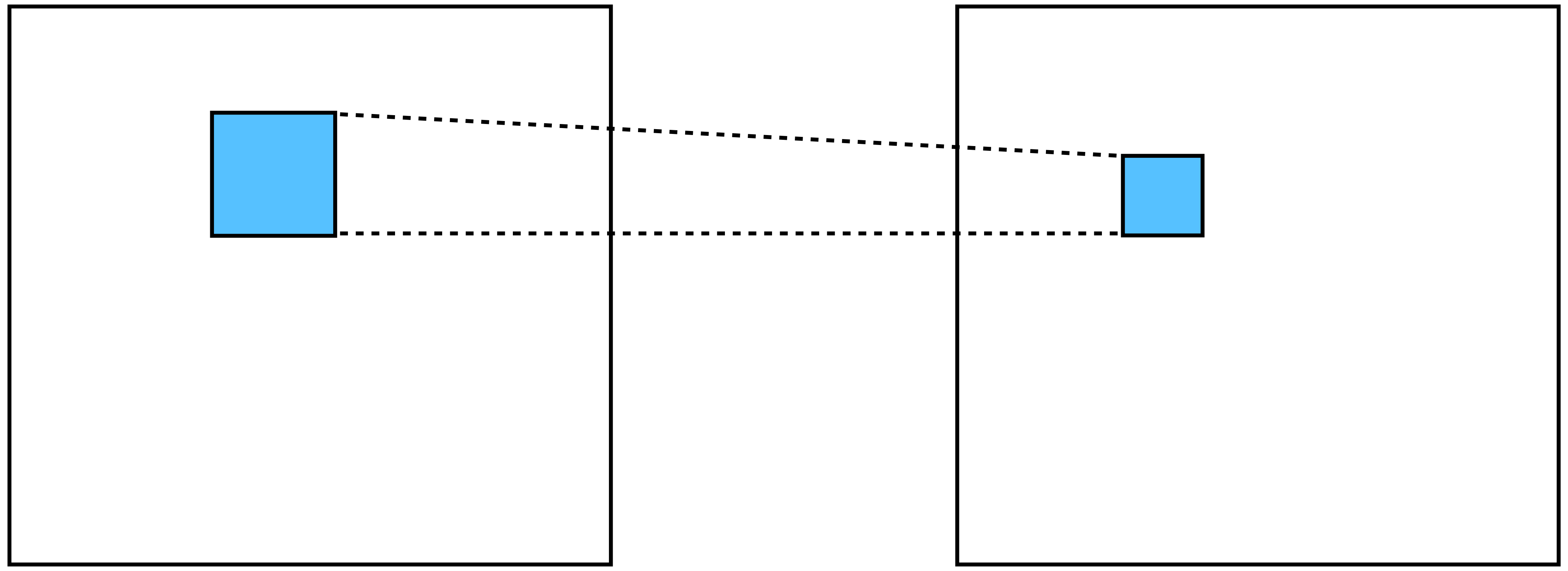
Early Fusion: Build slowly in space,
All at once in time at start

3D CNN: Build slowly in space and build slowly in time

→ **Slow Fusion**

Difference with time

2D Conv vs 3D Conv



2D Convolution

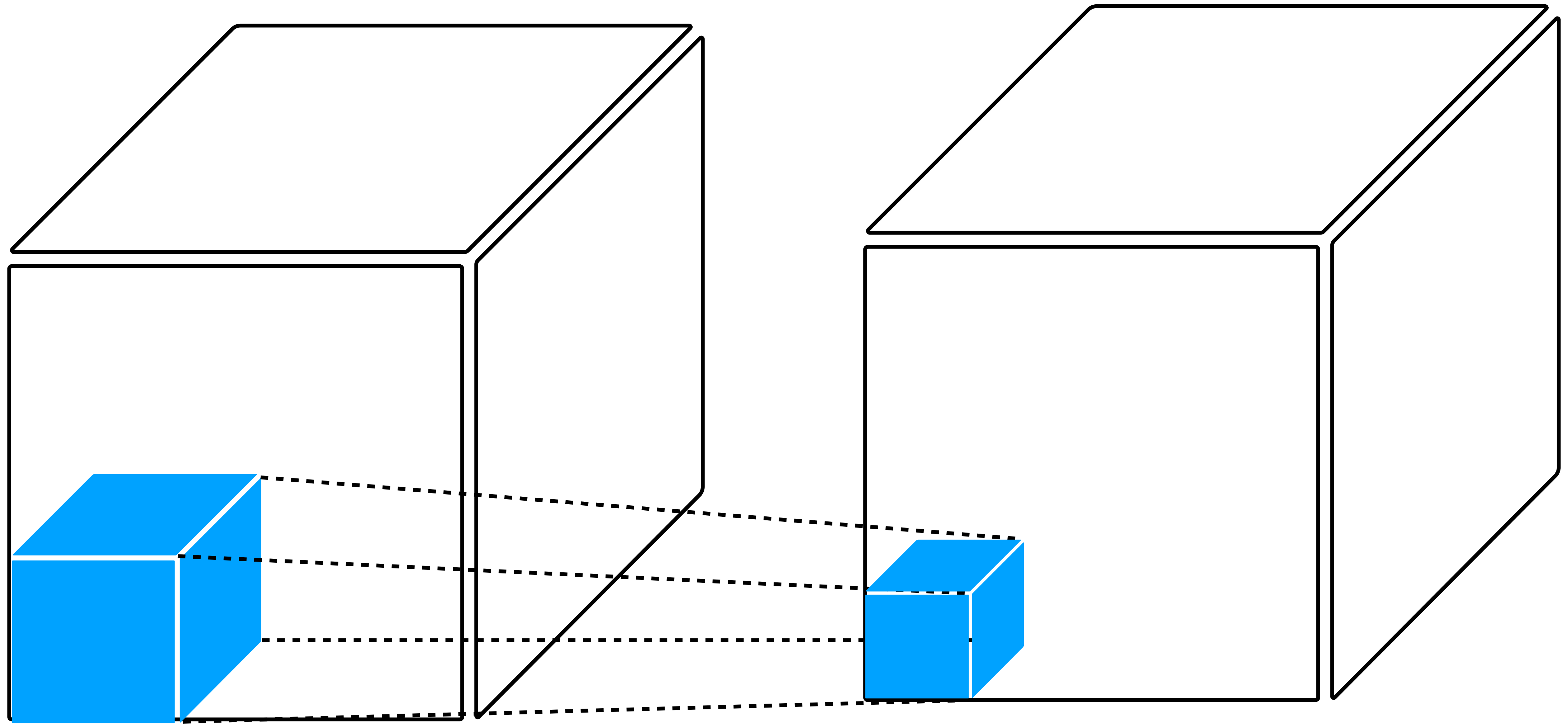
No temporal shift-invariance!

Input

C_{in} x T x H x W

Output

C_{out} x H x W



3D Convolution

Temporal shift-invariance since each filter slides over time!

Input

C_{in} x T x H x W

Output

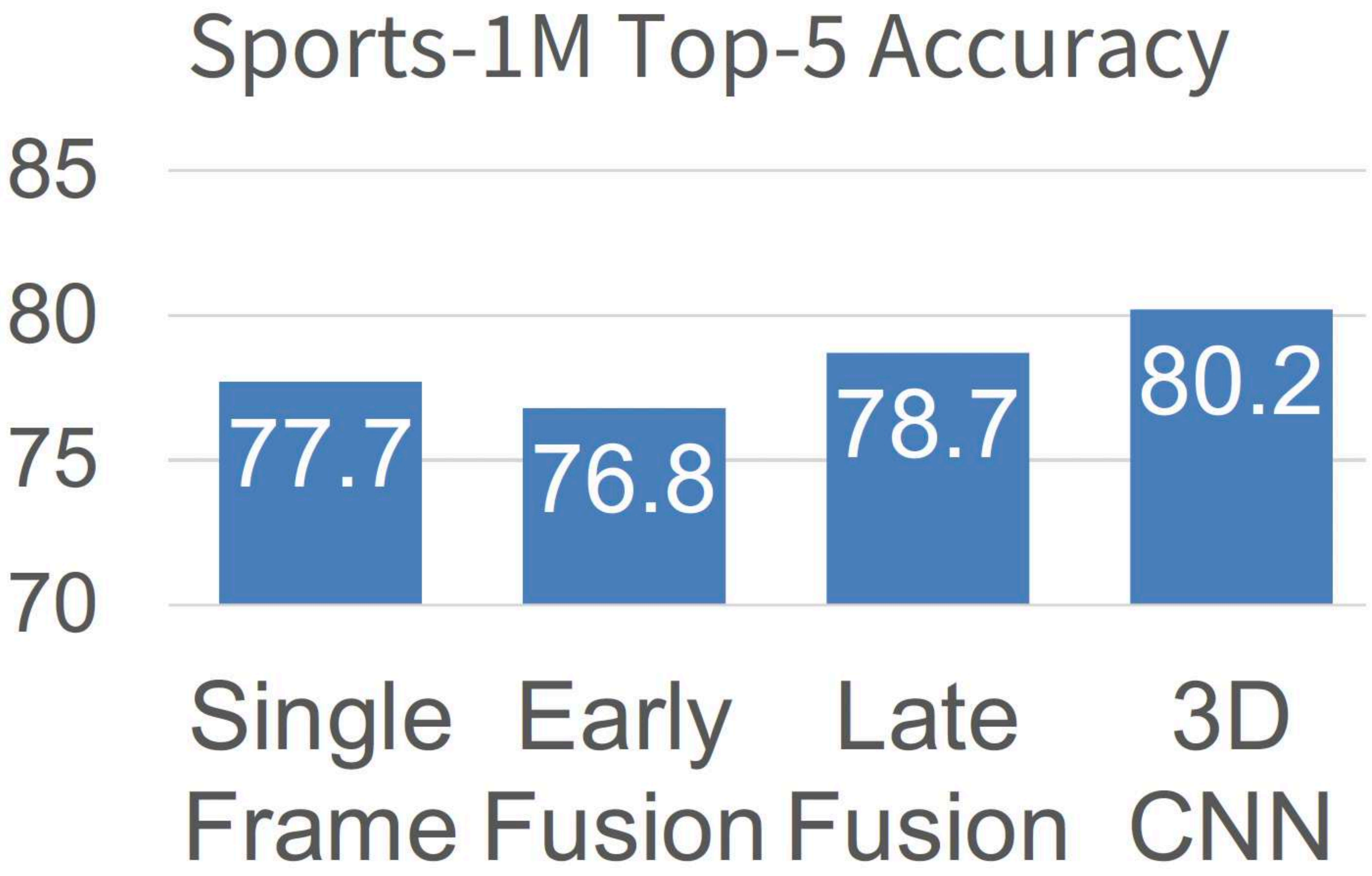
C_{out} x T x H x W

Example Video Dataset

Sport-1M

1 milion YouTube videos annotated with 487 labels of different type of sports

Early Fusion Vs Late Fusion Vs 3D CNN



3x3x3 conv is very expensive!

3D CNNs have improved a lot since 2014!

C3D - The VGG of 3D CNNs

3D CNN that uses all:

- 3x3x3 convolution
- 2x2x2 pooling

Released model retrained 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 convolution

- 2x2x2 pooling

Released model retrained on Sports-1M

Many people used this as a video feature extractor

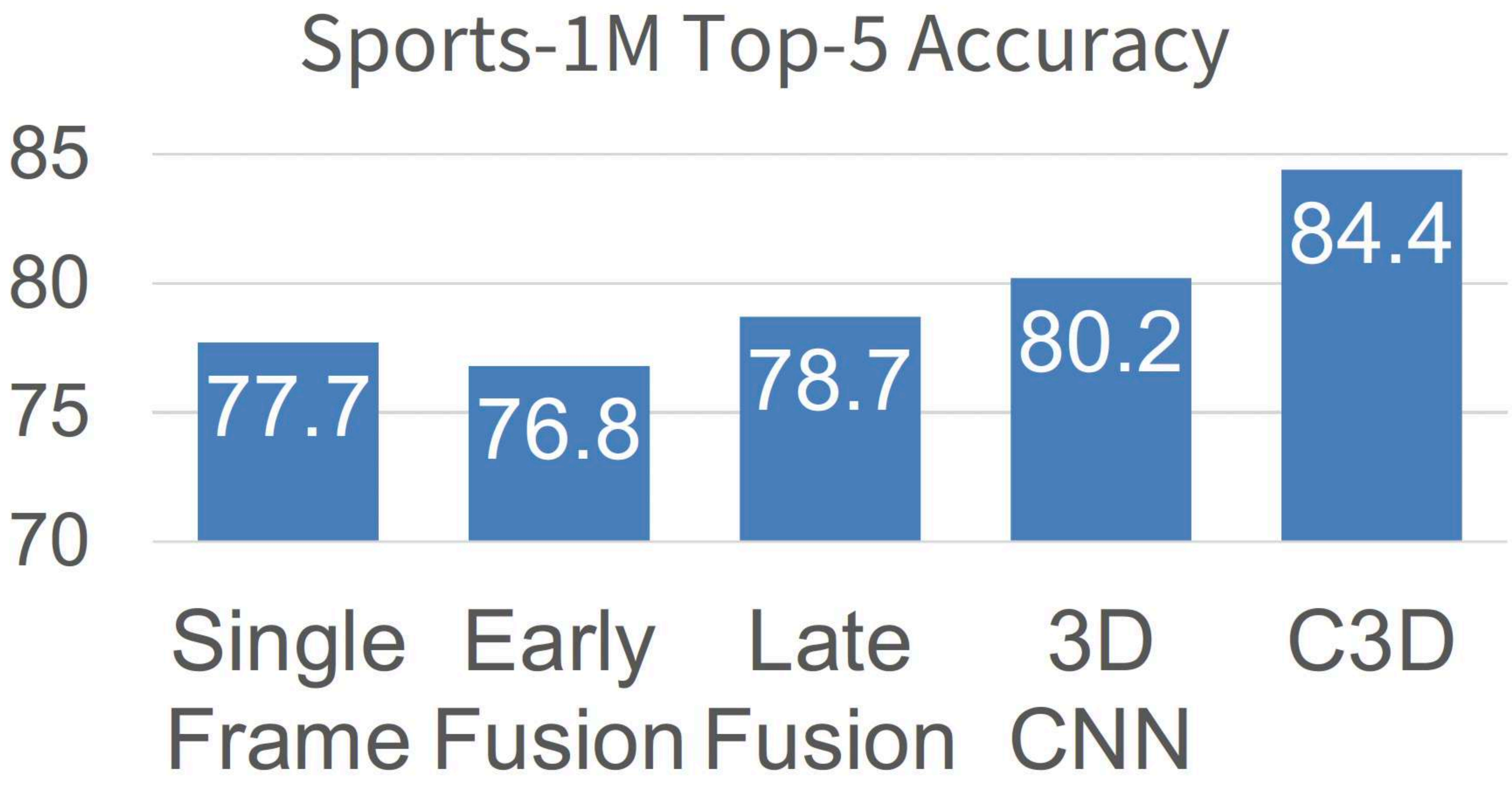
AlexNet -> 0.7 GFLOP

VGG16 -> 13.6 GFLOP

C3D -> 39.5 GFLOP

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

Early Fusion Vs Late Fusion Vs 3D CNN



Recognizing Actions form Motion

Easily recognize actions using only motion information

Optical Flow

Separating Motion and Appearance

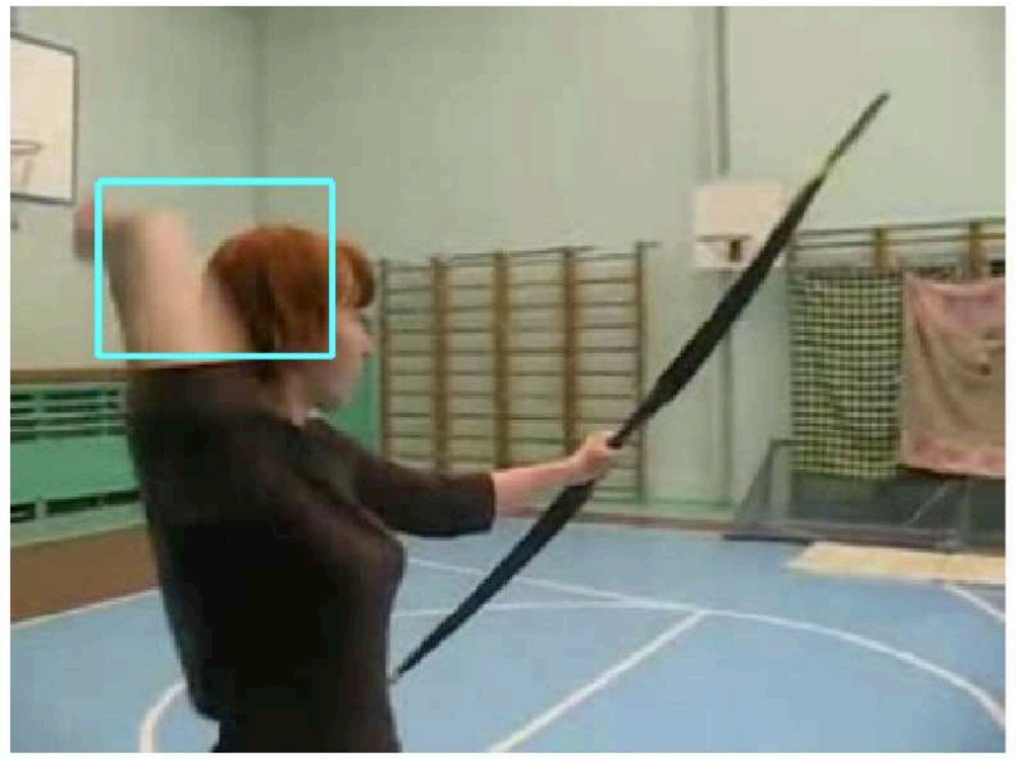


Image at frame t

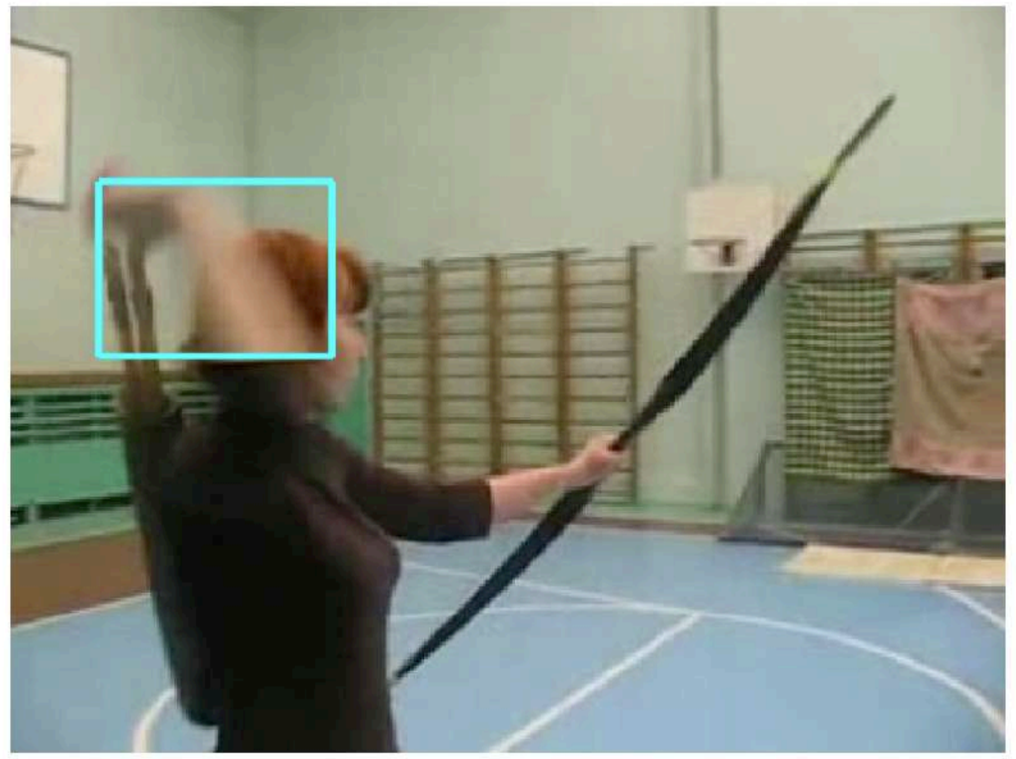


Image at frame t + 1

Optical Flow

Separating Motion and Appearance

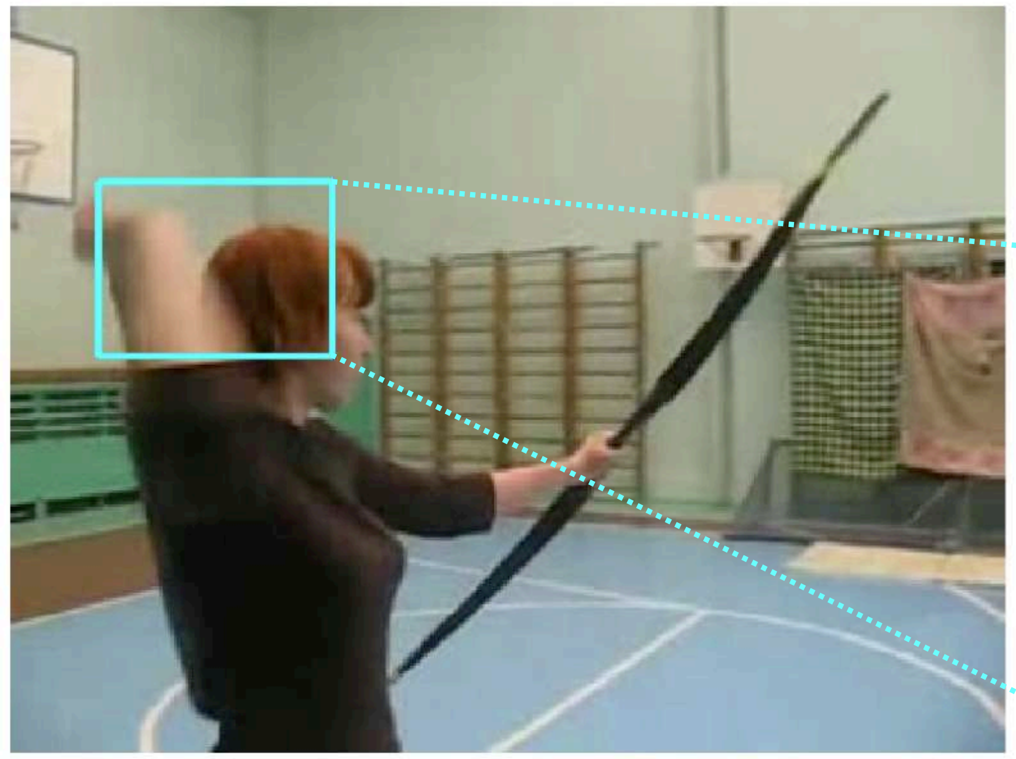


Image at frame t

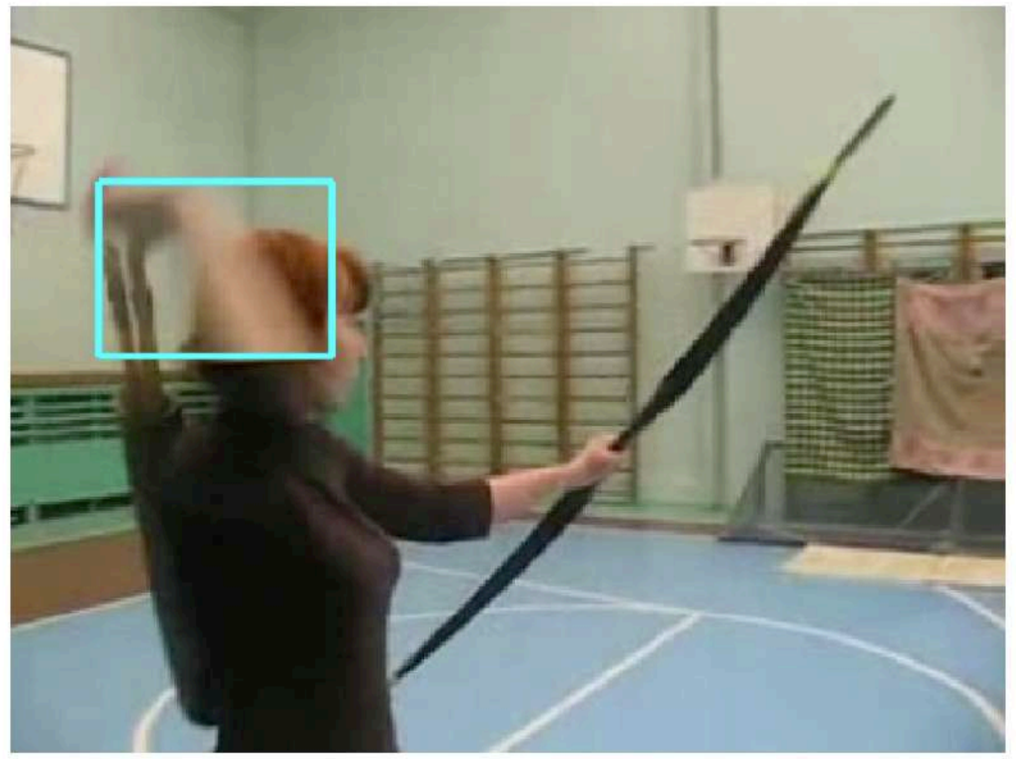
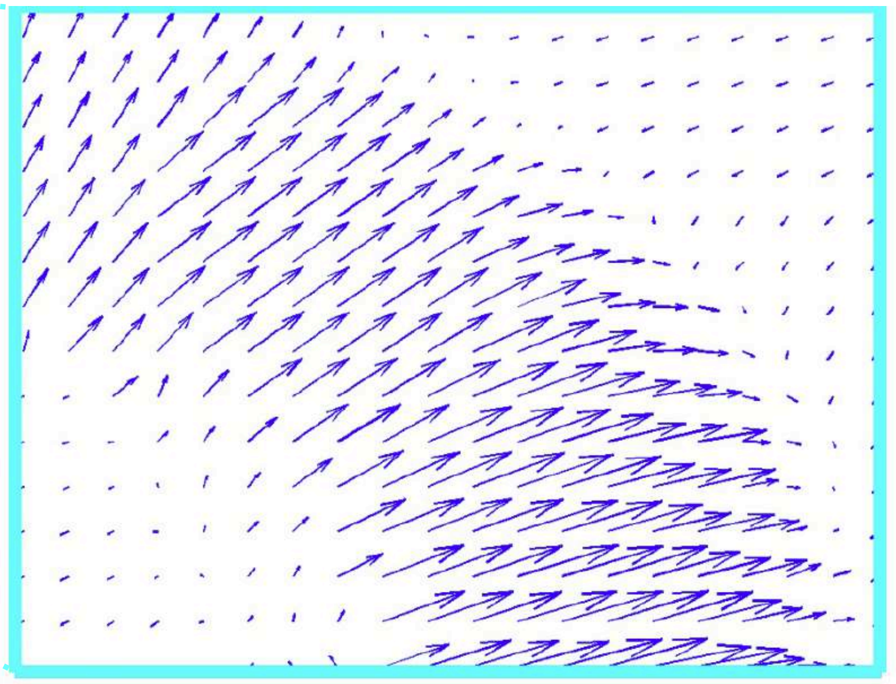


Image at frame t + 1



Displacement filed F between images I_t and I_{t+1}

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

Separating Motion and Appearance

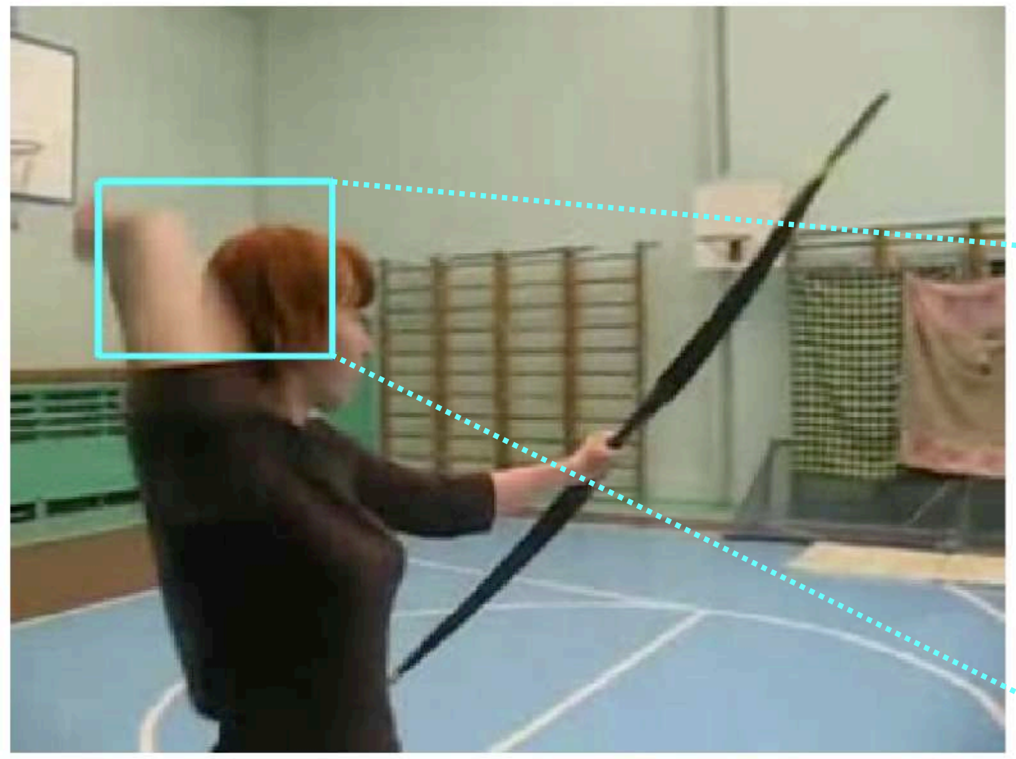


Image at frame t

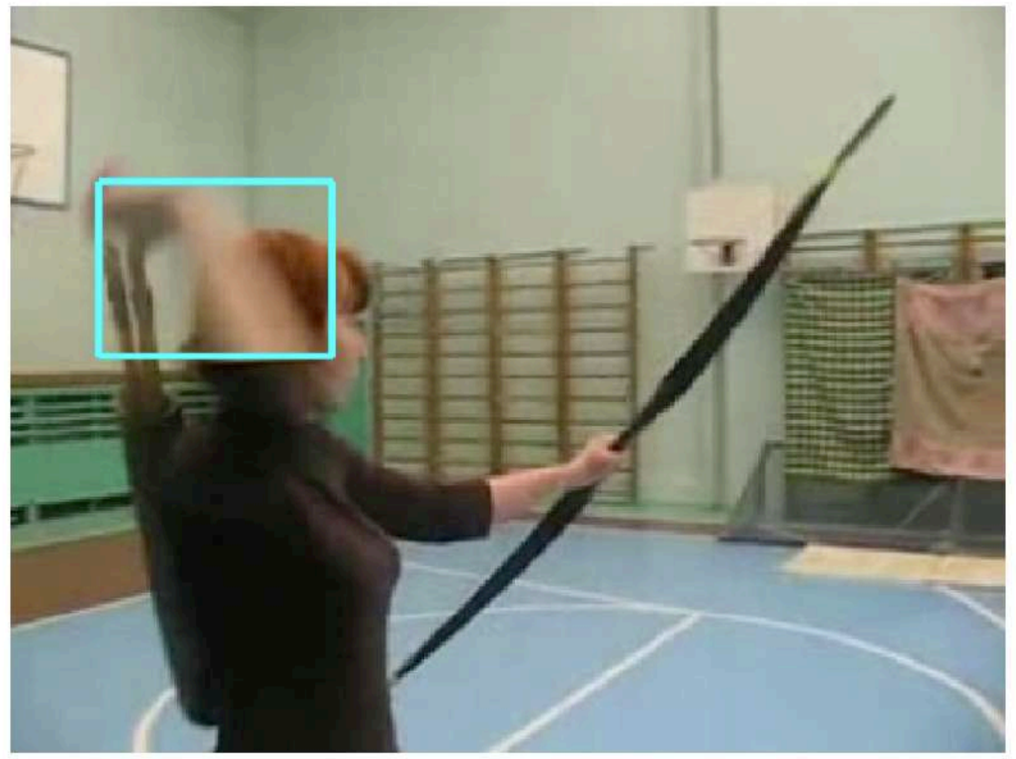
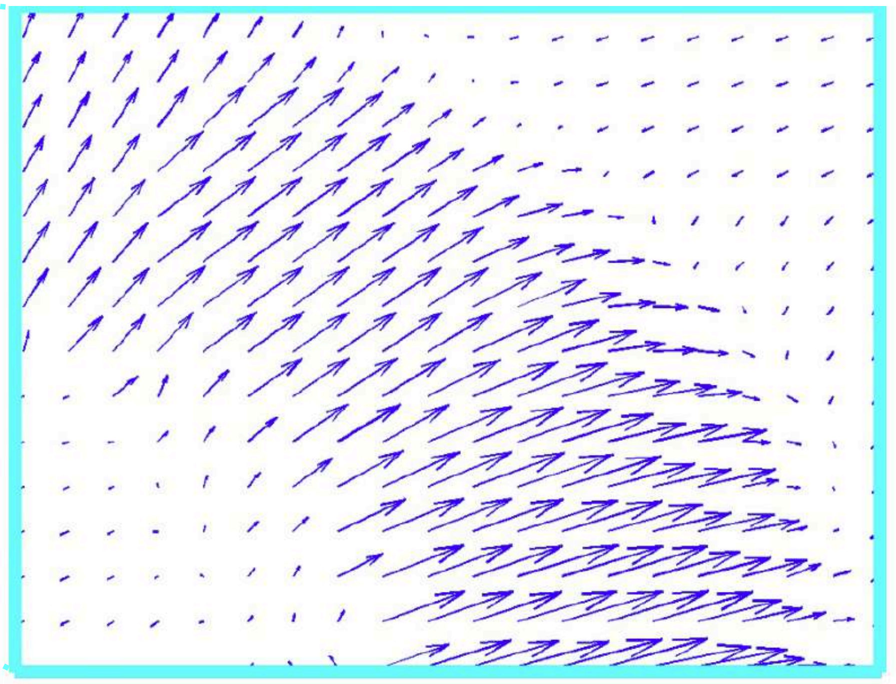


Image at frame t + 1

Displacement filed F
between images I_t and I_{t+1}



Where each pixel will move in the next frame:

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

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

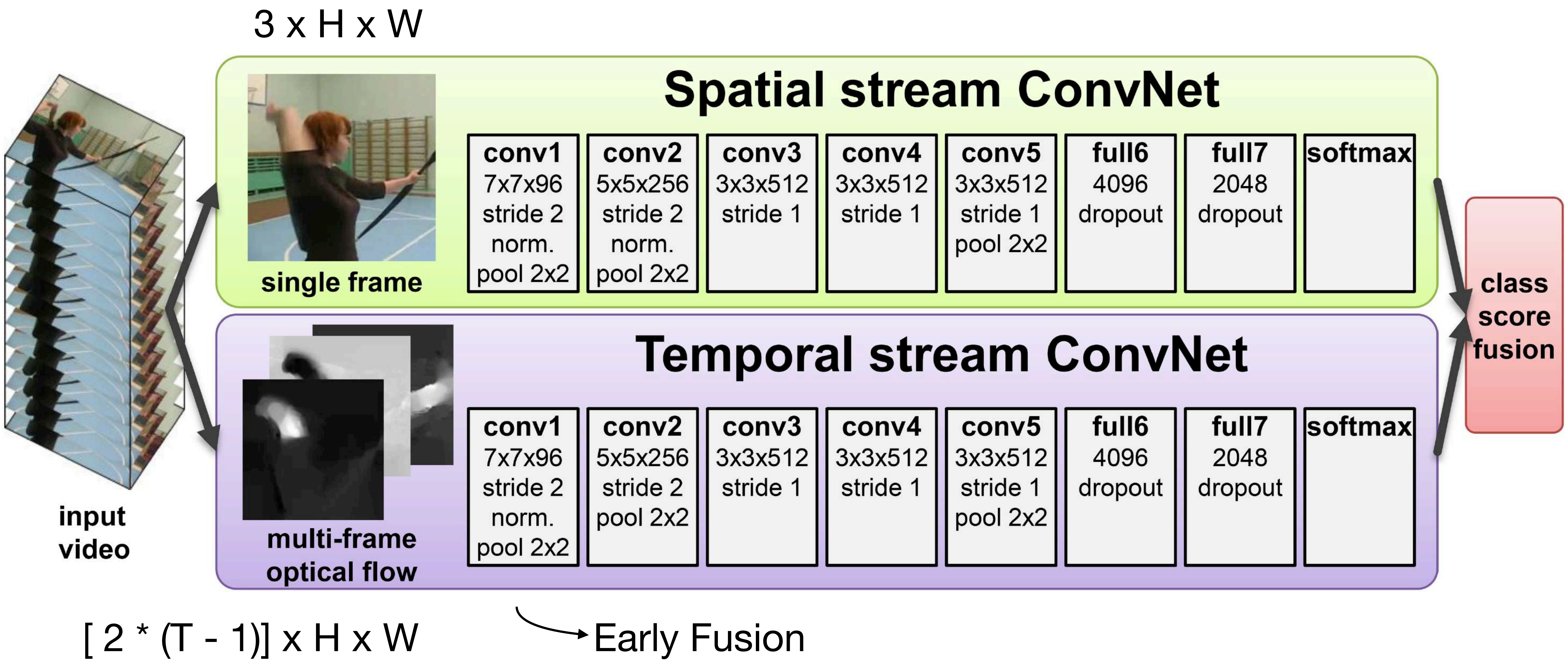
Local Motion

Horizontal flow dx

Vertical flow dy

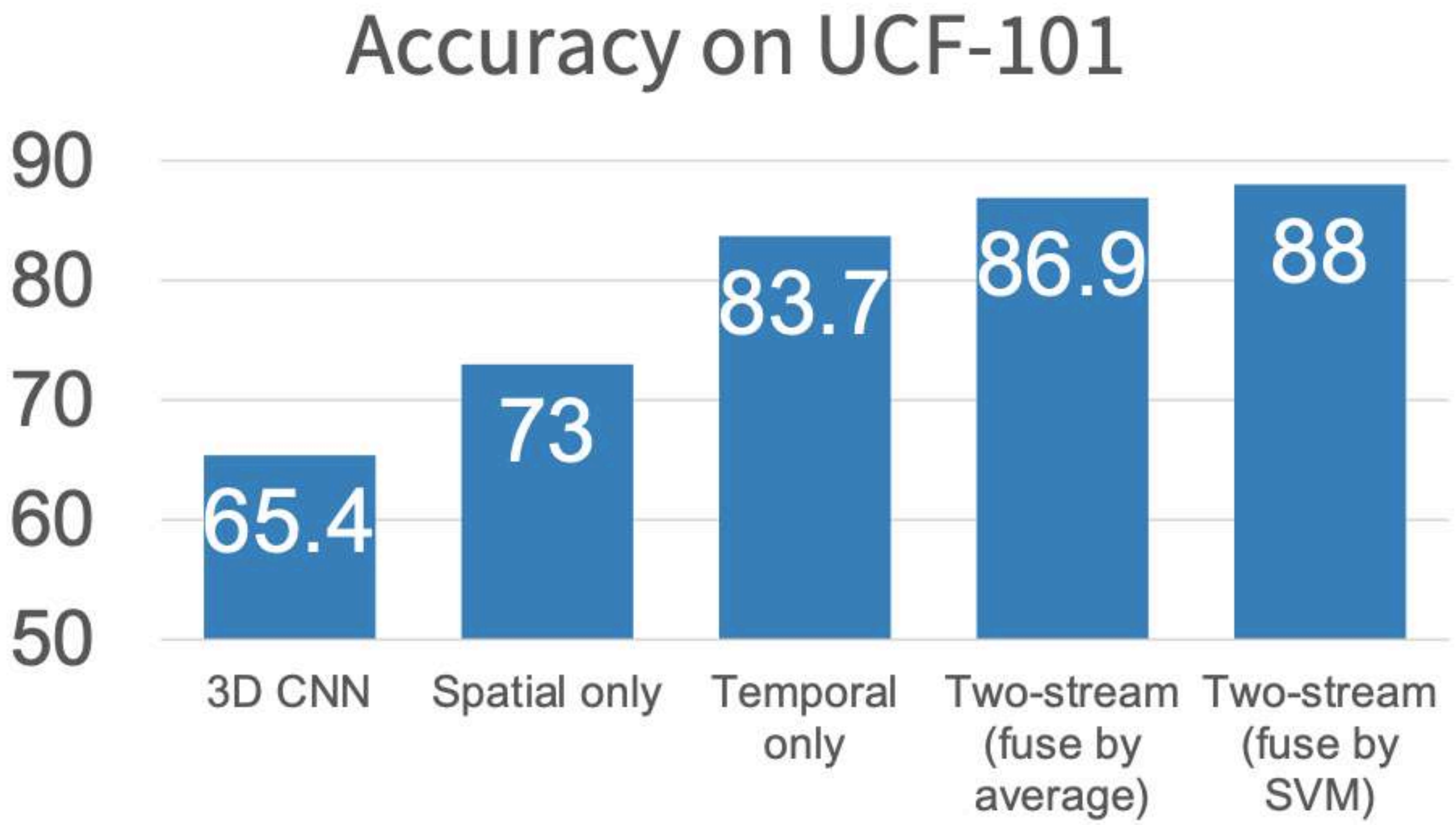
Two-Stream Networks

Separating Motion and Appearance



Two-Stream Networks

Separating Motion and Appearance

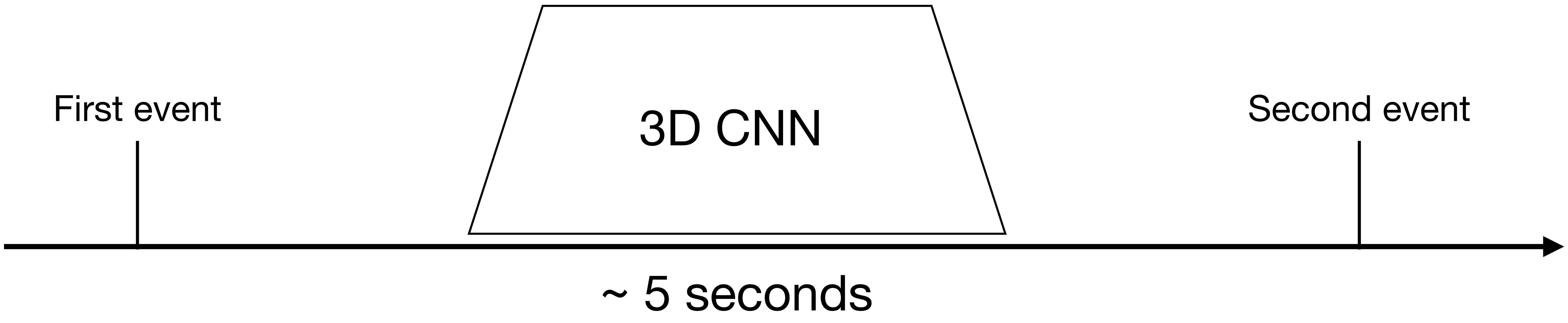


Problem!

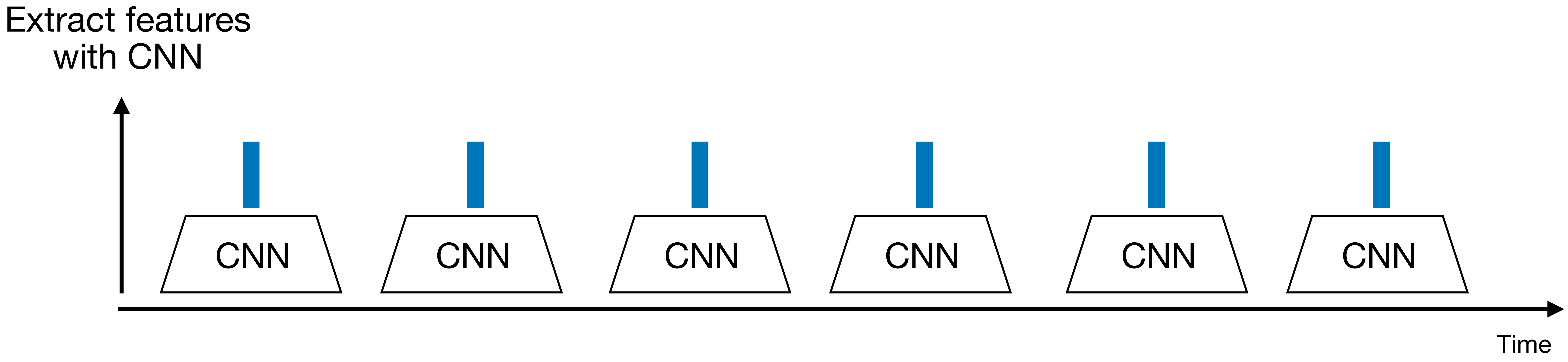
What about long-term structure?

Theoretically we know how to handle sequences

How about Recurrent Networks?



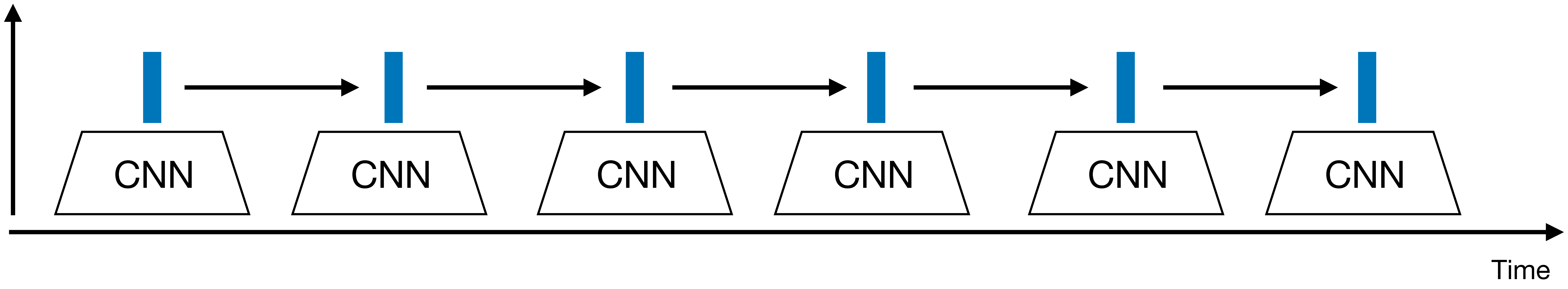
Modeling long-time temporal structure



Modeling long-time temporal structure

Process local features using recurrent network (es. LSTM)

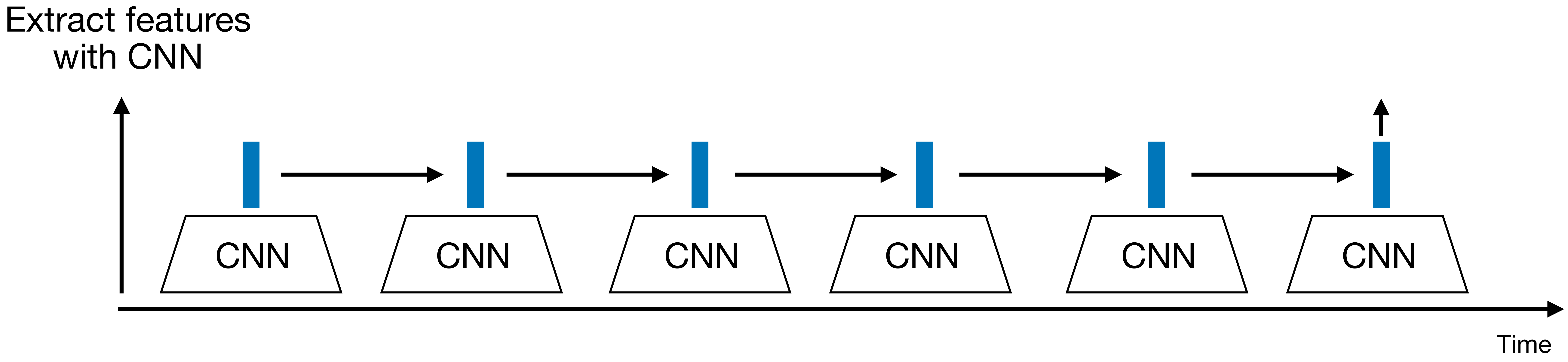
Extract features
with CNN



Modeling long-time temporal structure

Process local features using recurrent network (es. LSTM)

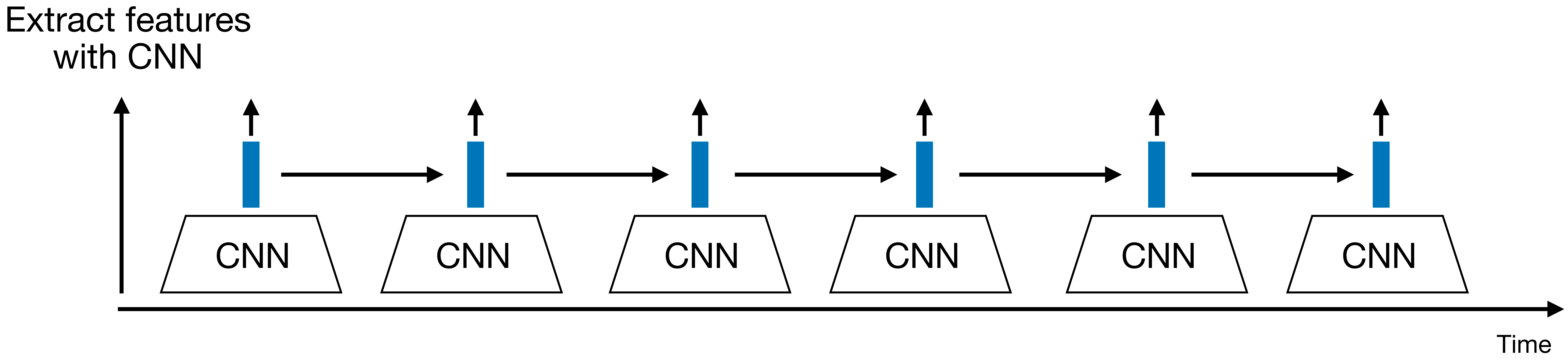
Many to one -> one output at end of video



Modeling long-time temporal structure

Process local features using recurrent network (es. LSTM)

Many to many -> one output per video frame

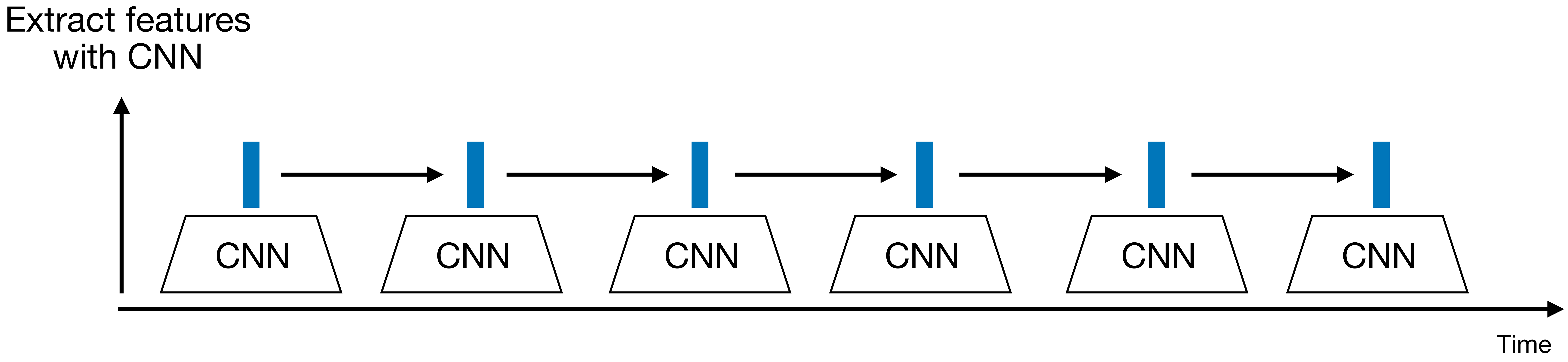


Modeling long-time temporal structure

Inside CNN -> Each value is a function of a fixed temporal window

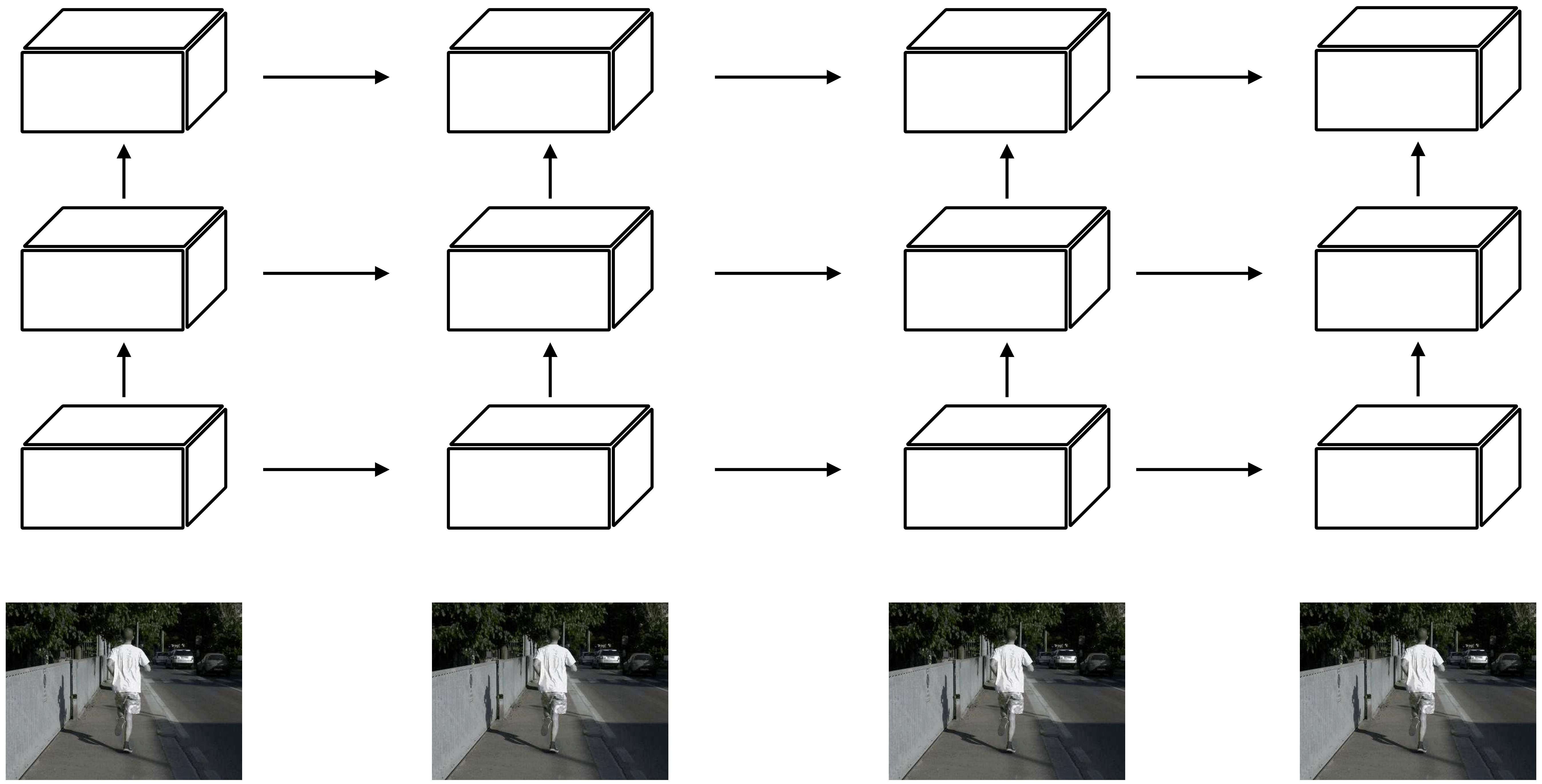
Inside RNN -> Each vector is a function of all previous vectors

Can we merge both approaches?



Recurrent Convolutional Network

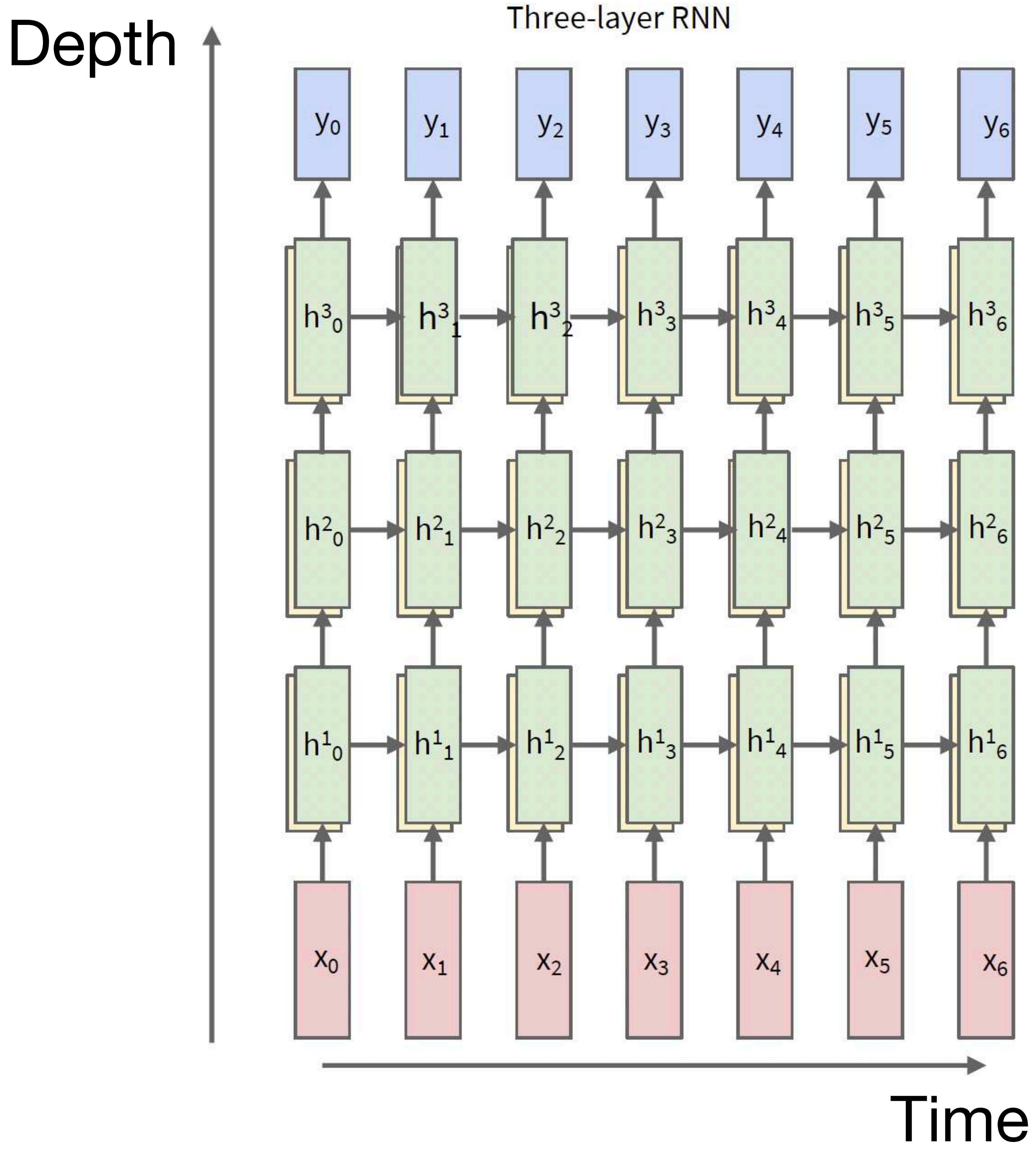
Entire network uses 2D feature maps -> C x H x W



Use different weights at each layer

Share weights across time

Multi-layer RNN



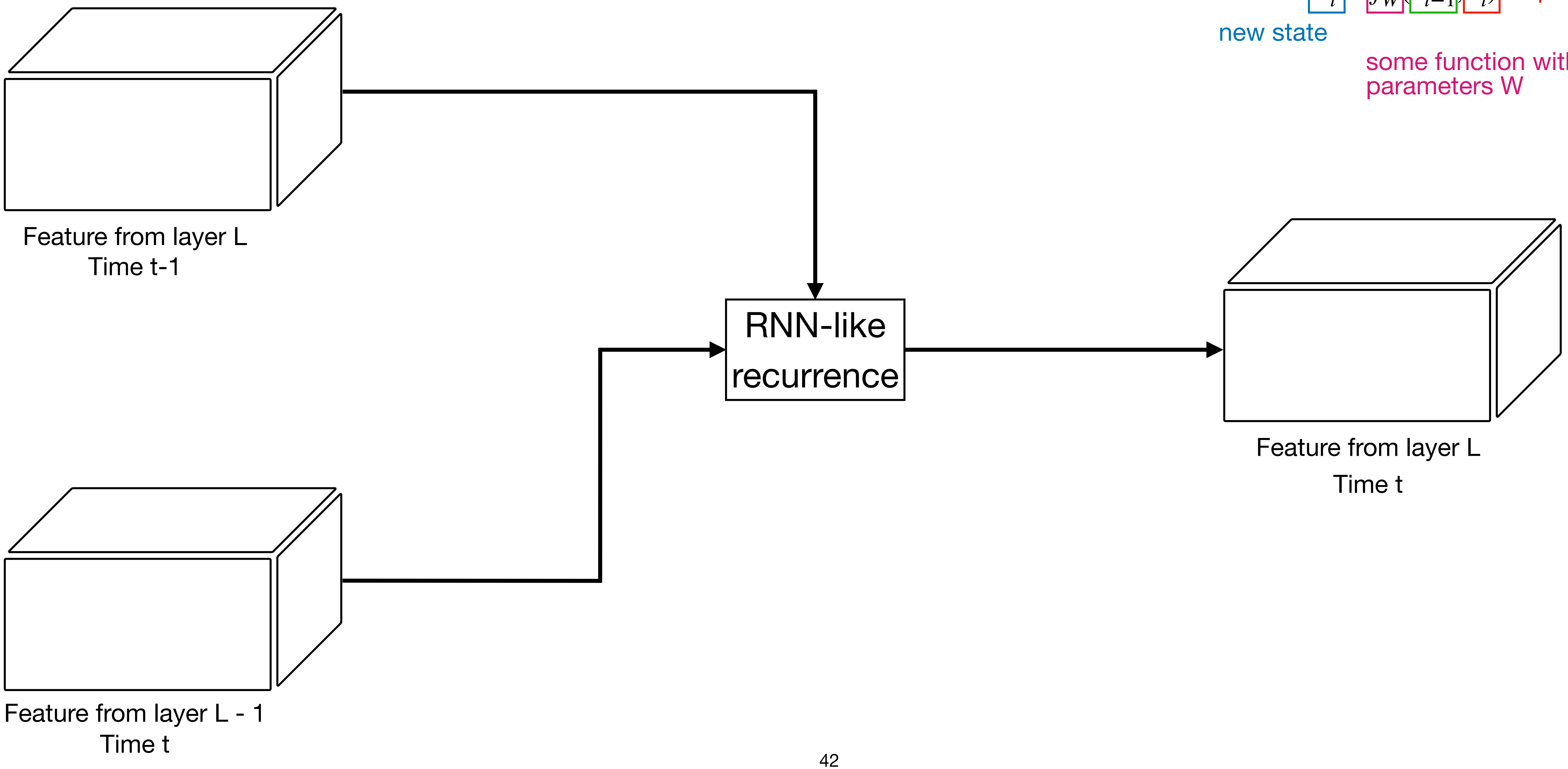
Recurrent Convolutional Network

old state

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

new state

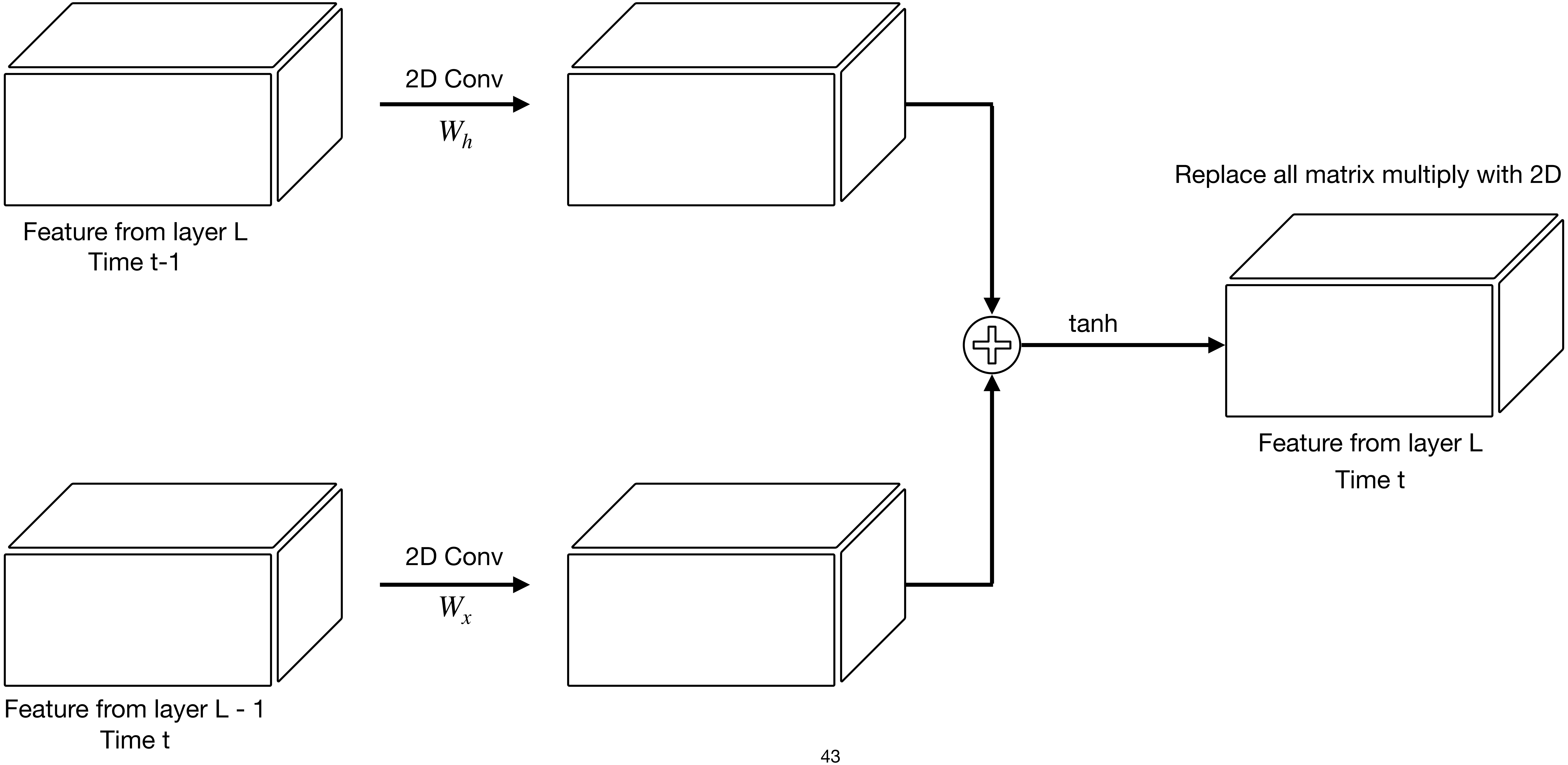
some function with parameters W



Recurrent Convolutional Network

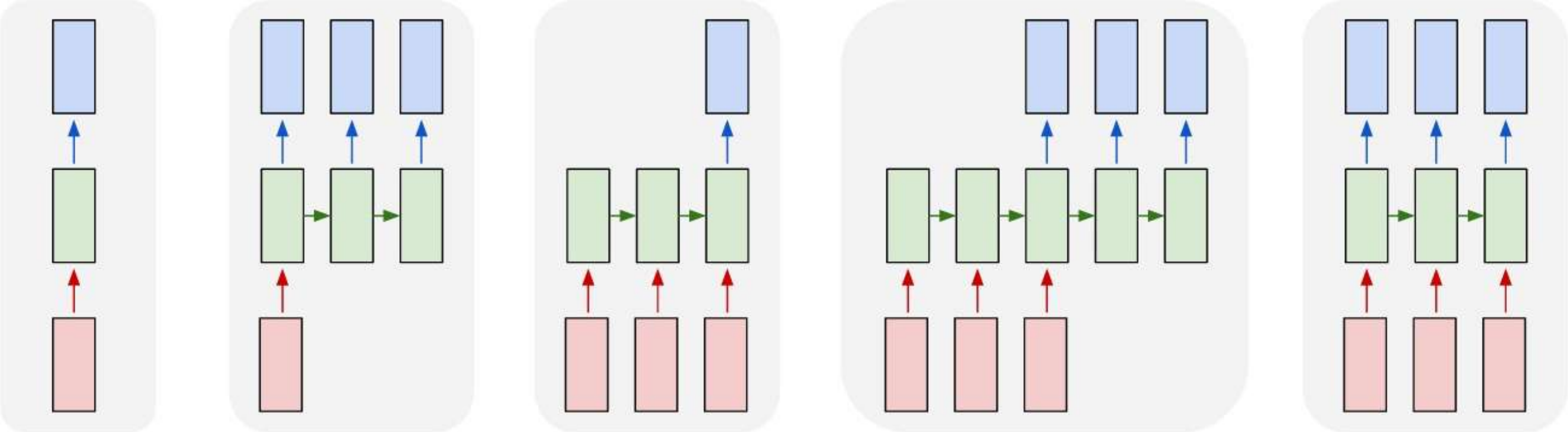
Recall: $h_{t=1} = \tanh(W_h h_t + W_x x)$

Replace all matrix multiply with 2D convolution

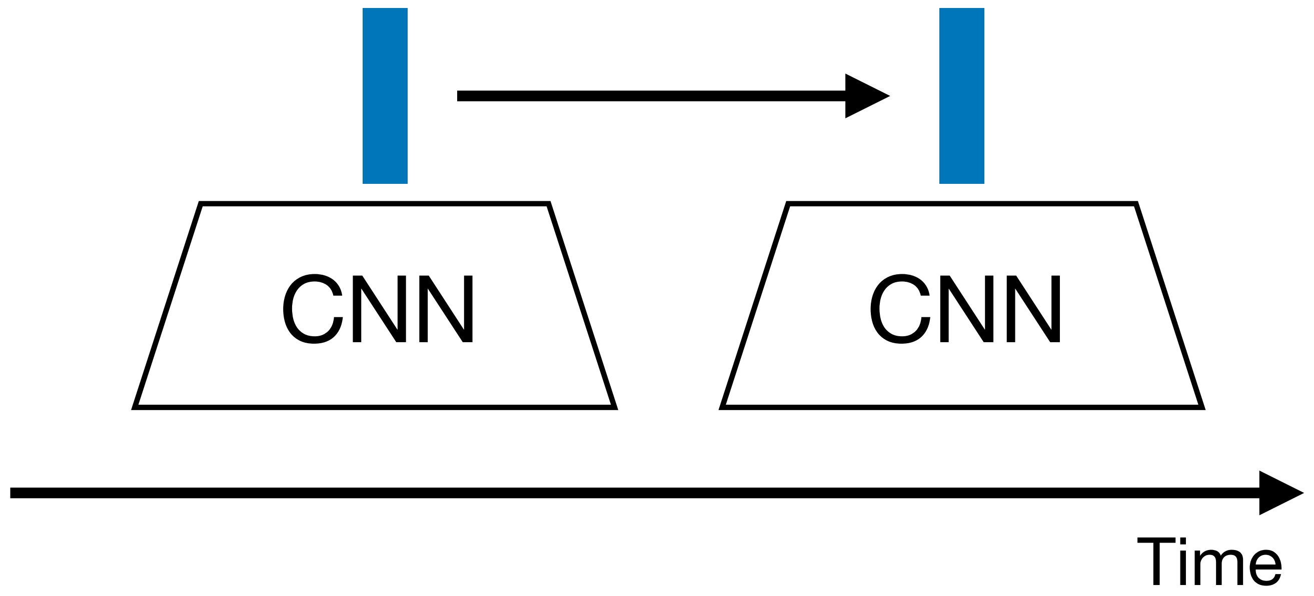


Modeling long-time temporal structure

Infinite temporal extent

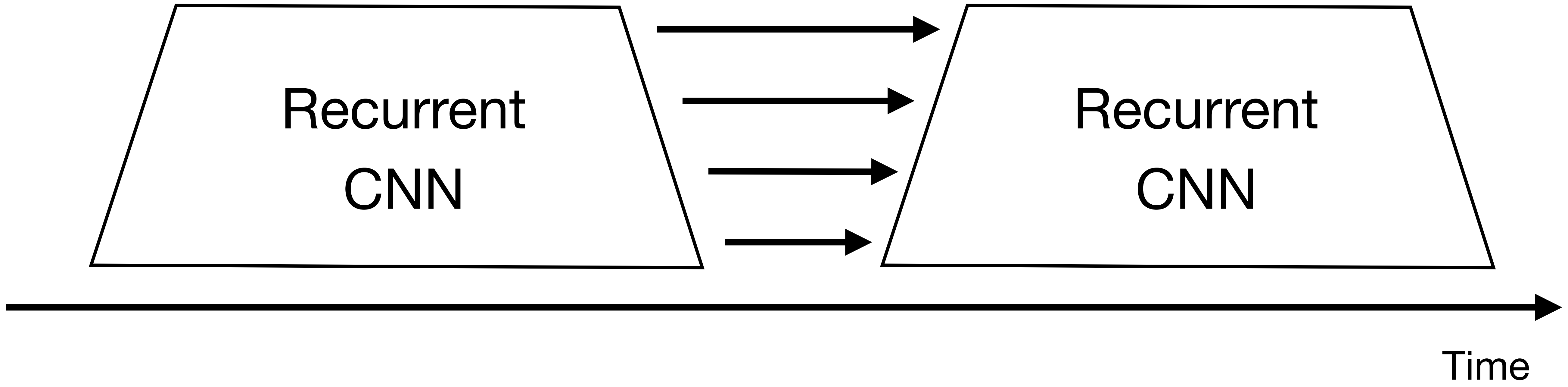


Finite temporal extent

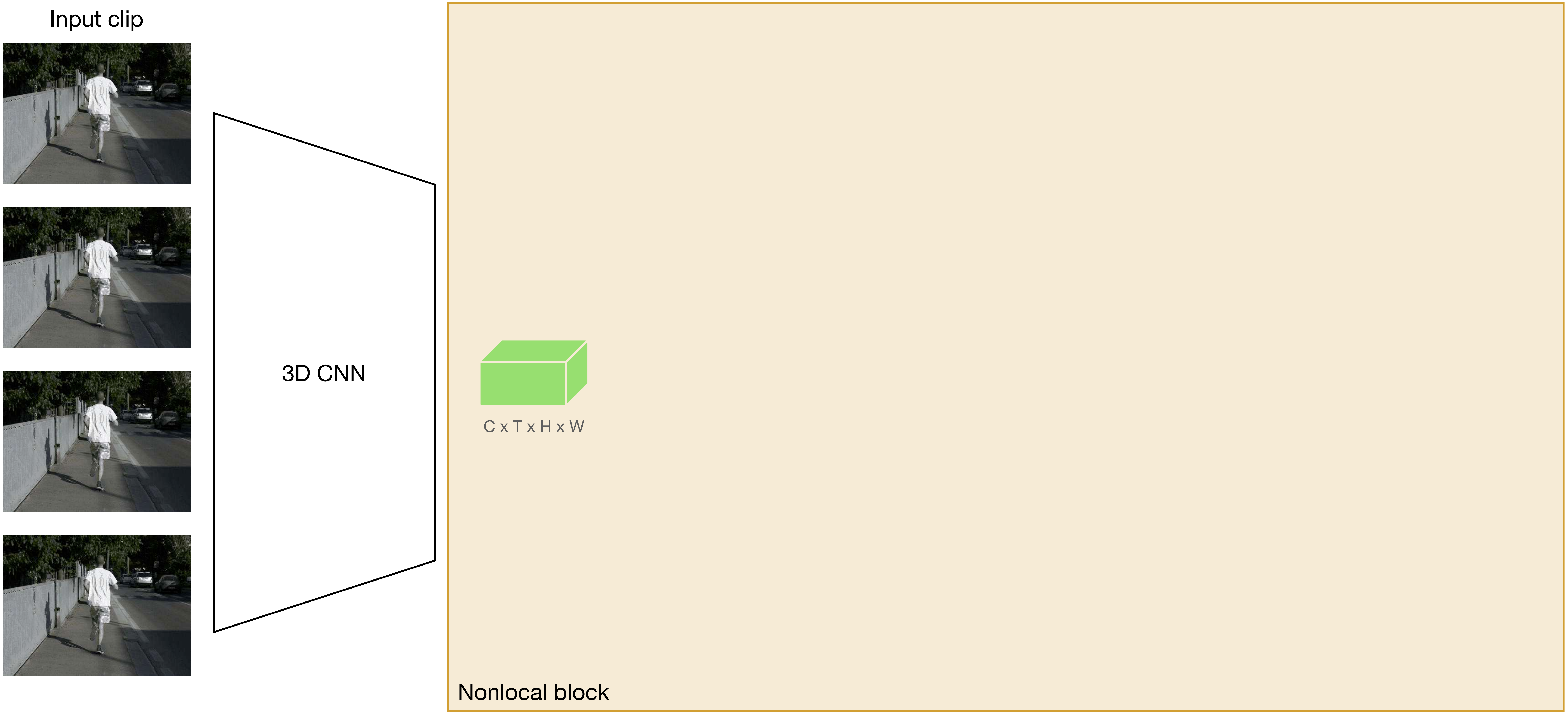


RNNs are slow for long sequences

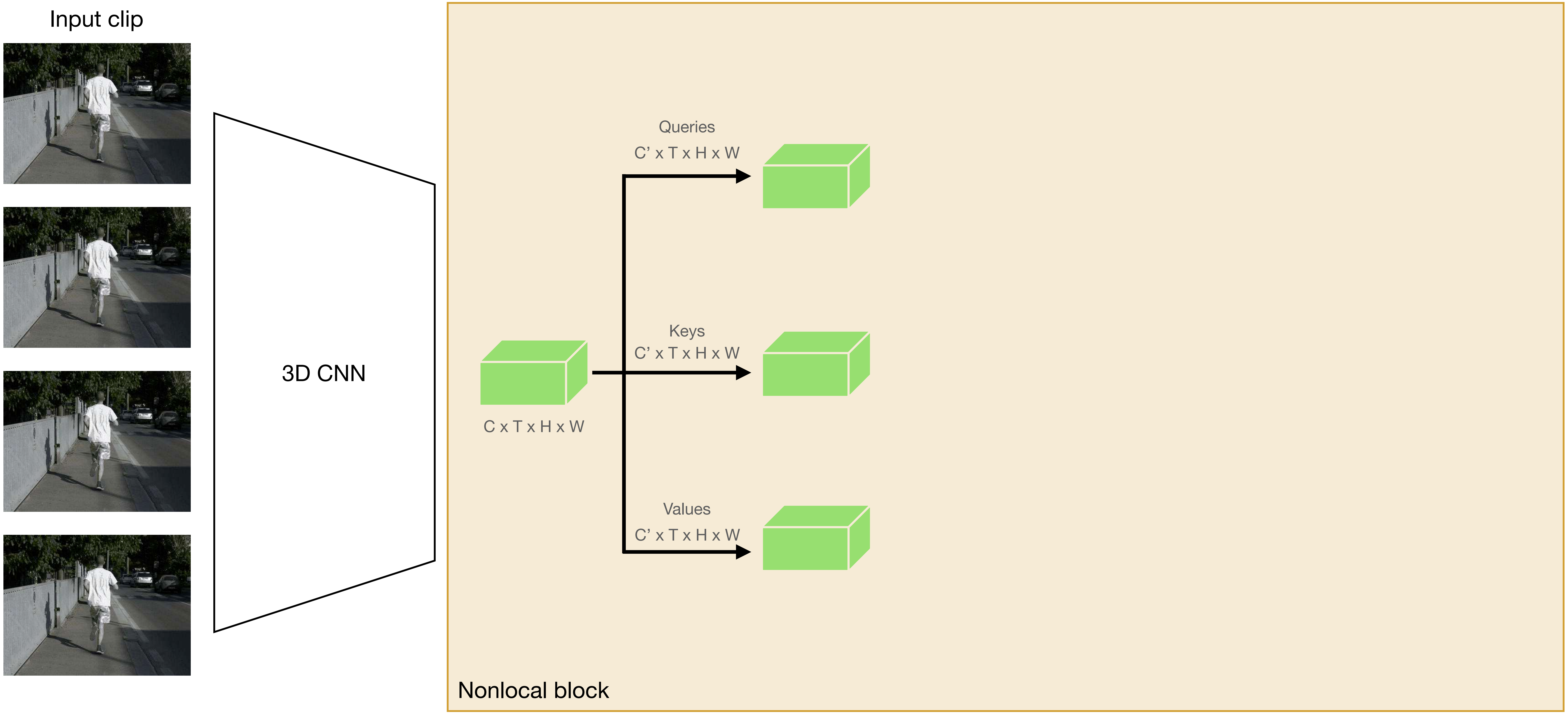
Infinite temporal extent



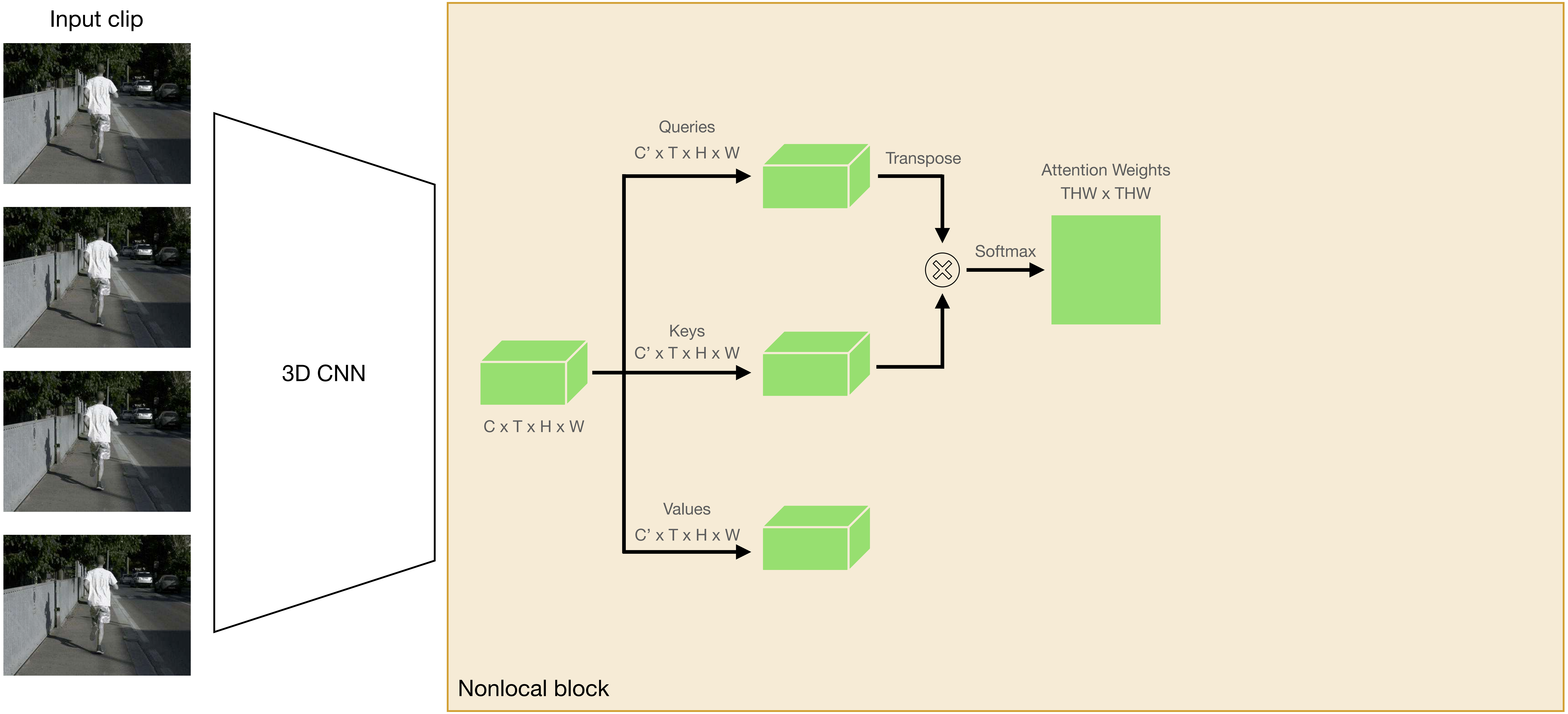
Spatio-Temporal Self-Attention



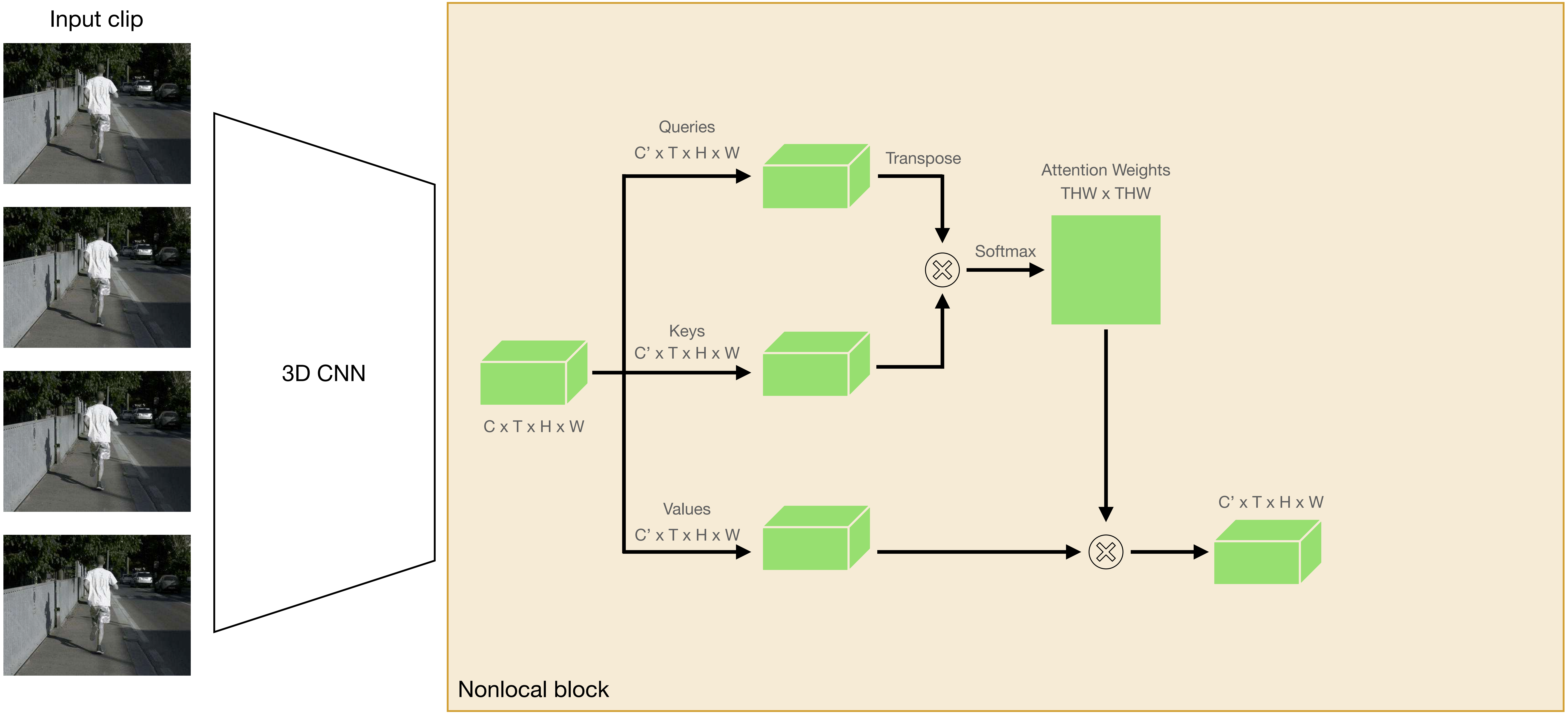
Spatio-Temporal Self-Attention



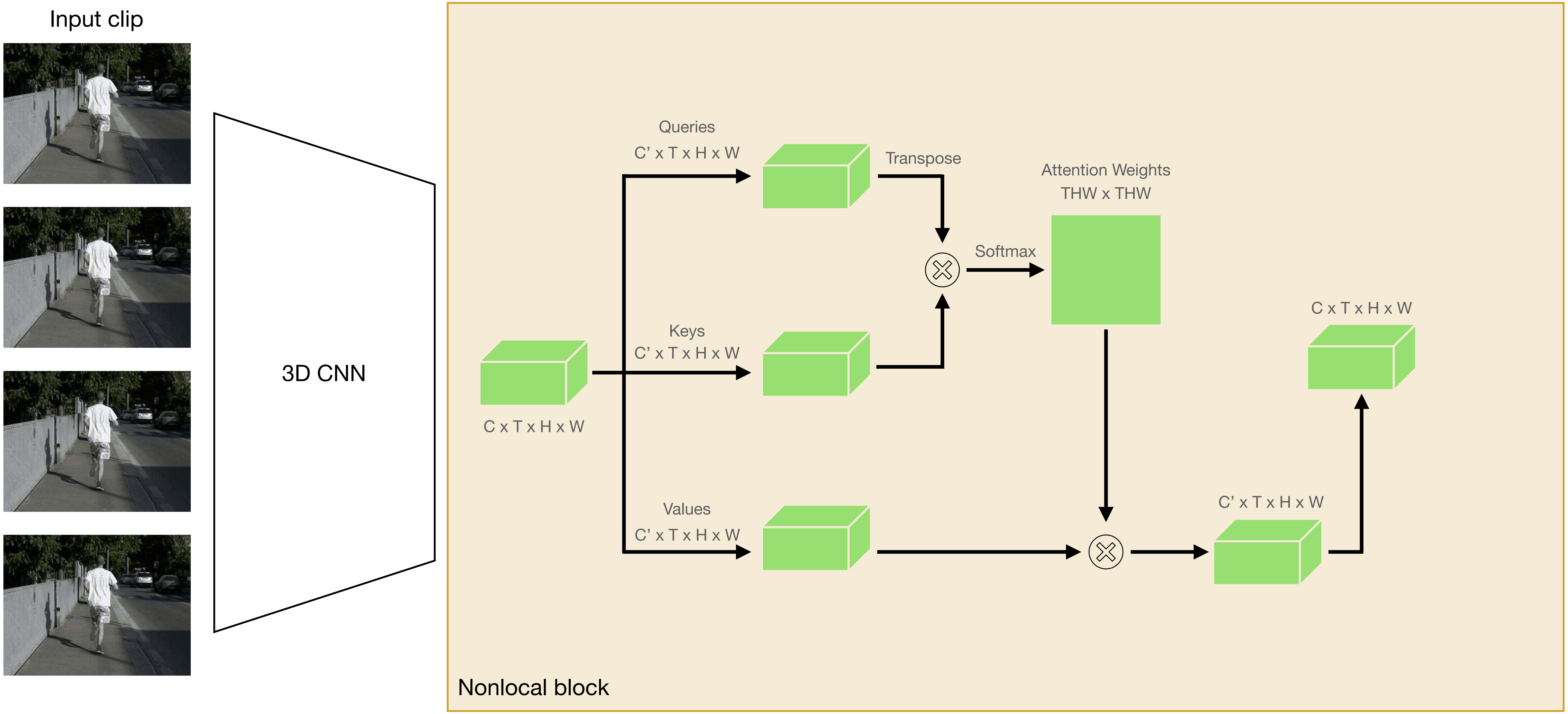
Spatio-Temporal Self-Attention



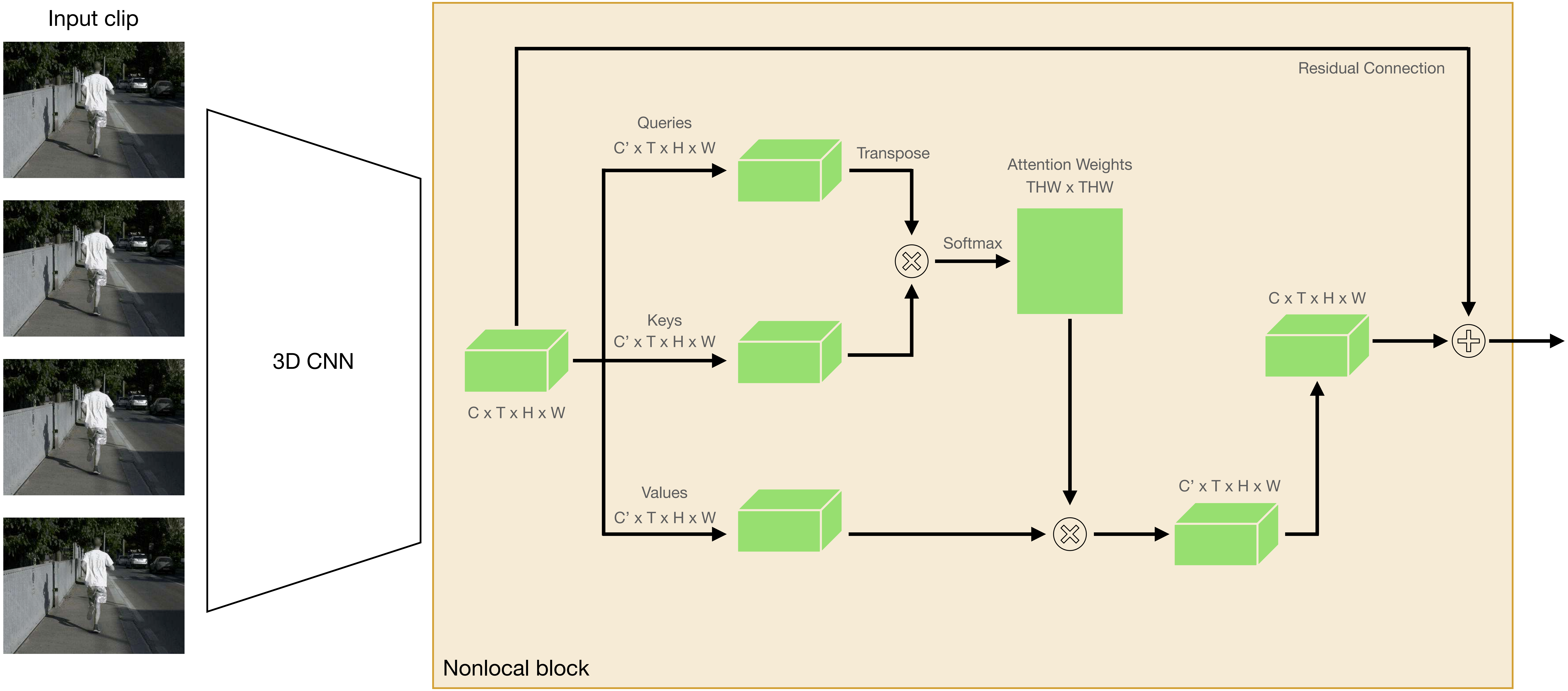
Spatio-Temporal Self-Attention



Spatio-Temporal Self-Attention

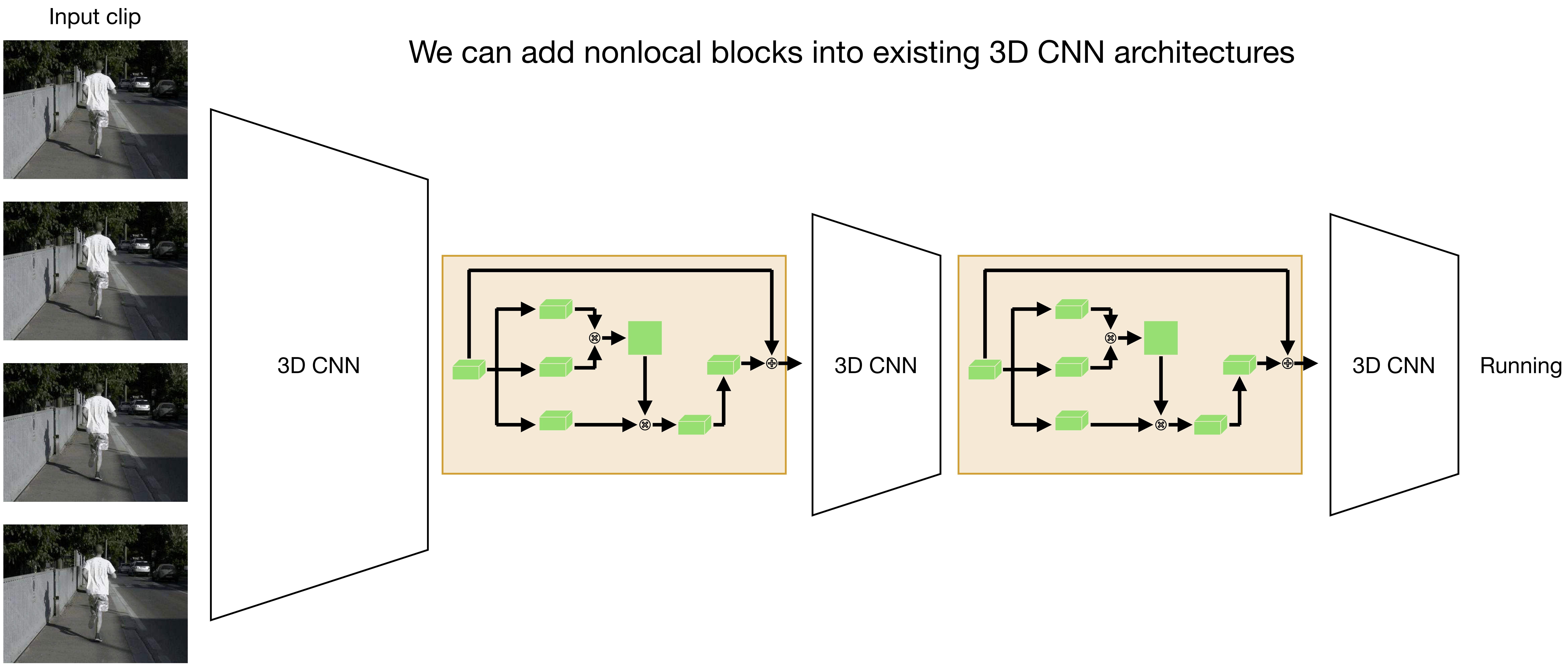


Spatio-Temporal Self-Attention



Spatio-Temporal Self-Attention

We can add nonlocal blocks into existing 3D CNN architectures

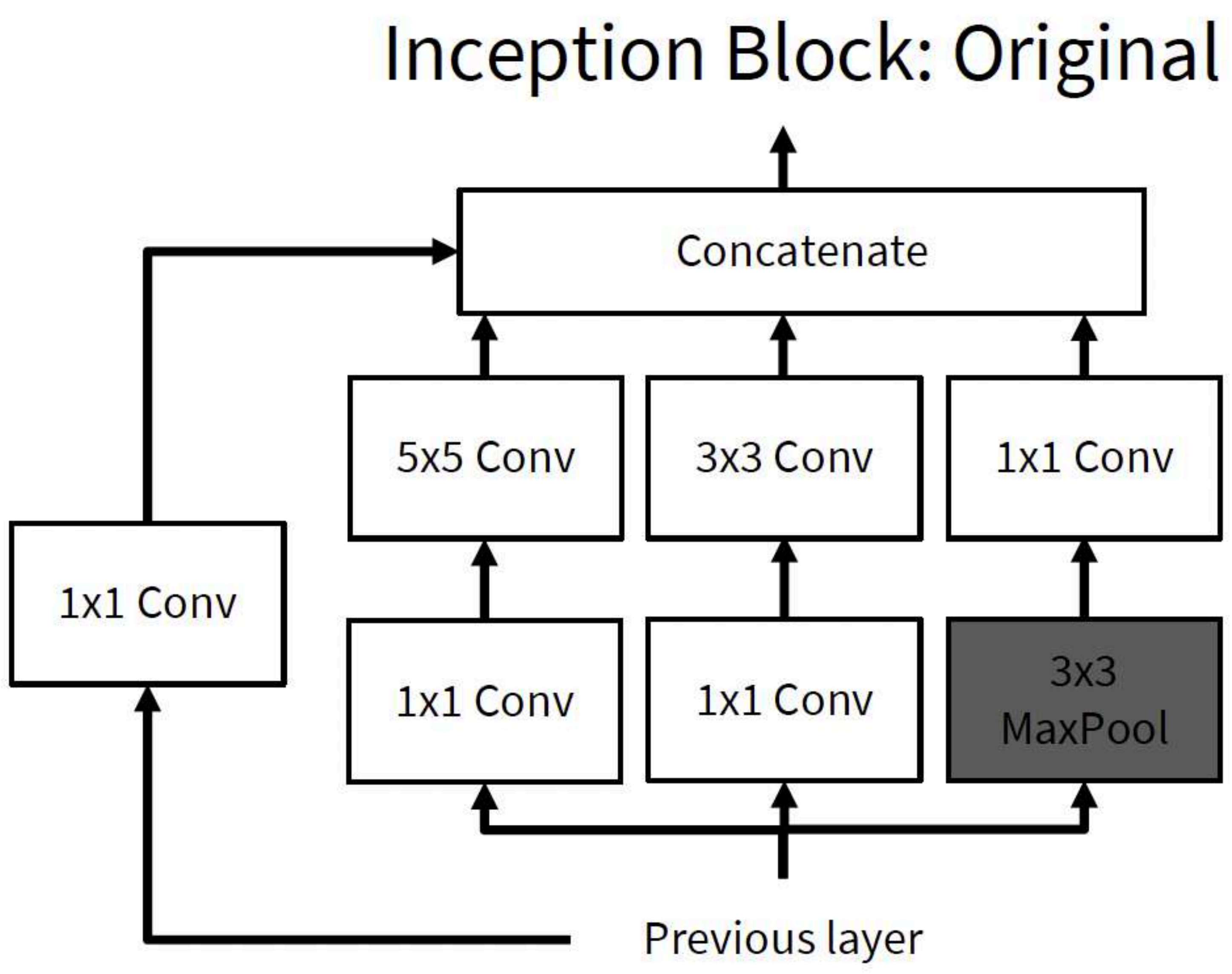


Inflating 2D Networks to 3D

Can we reuse image architectures for video?

Idea: take a 2D CNN architecture

Replace each 2D K_h x K_W conv/pool layer



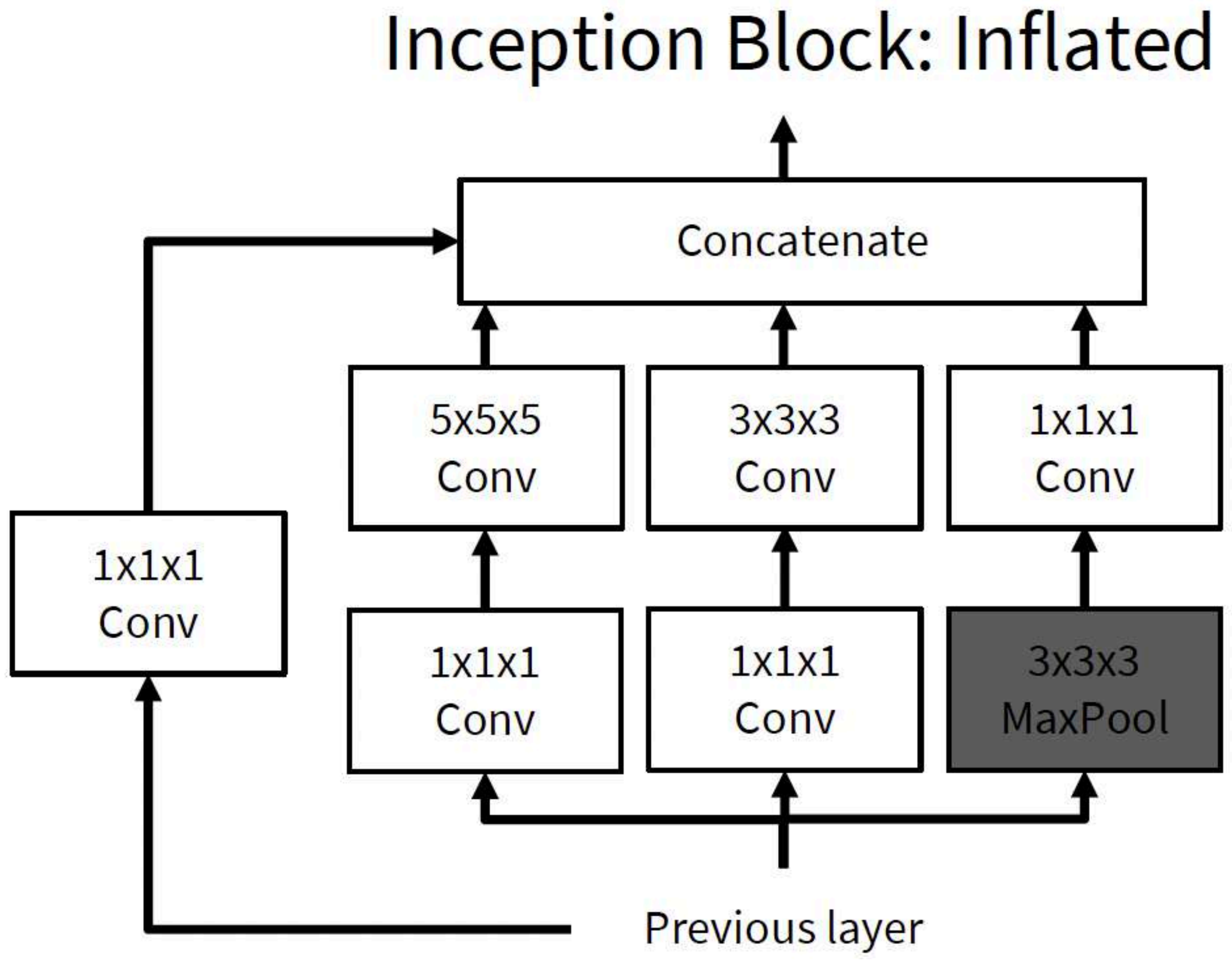
Inflating 2D Networks to 3D

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 3D K_t x K_h x K_W version



Inflating 2D Networks to 3D

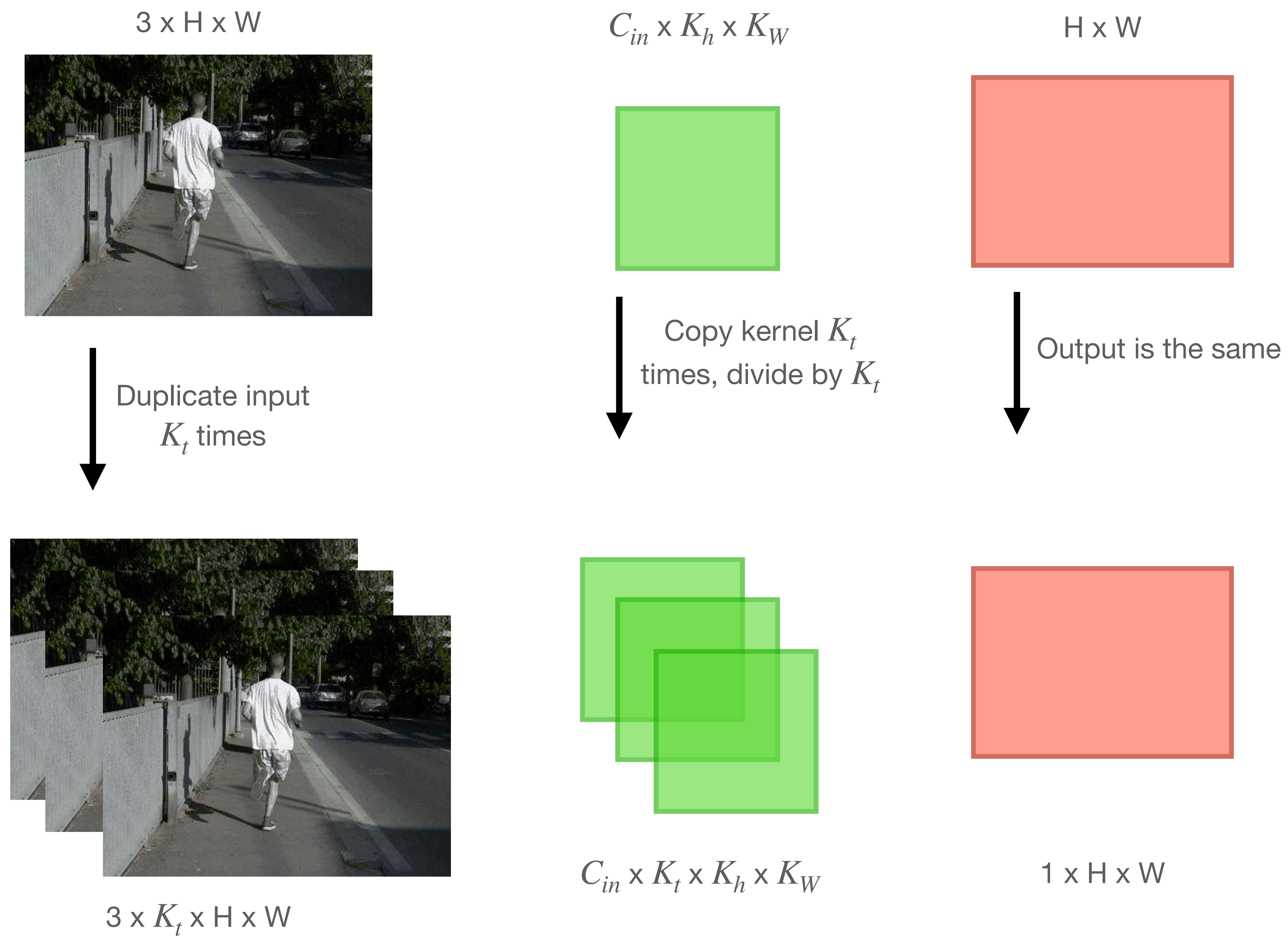
Can we reuse image architectures for video?

Idea: take a 2D CNN architecture

Replace each 2D conv/pool layer with 3D version

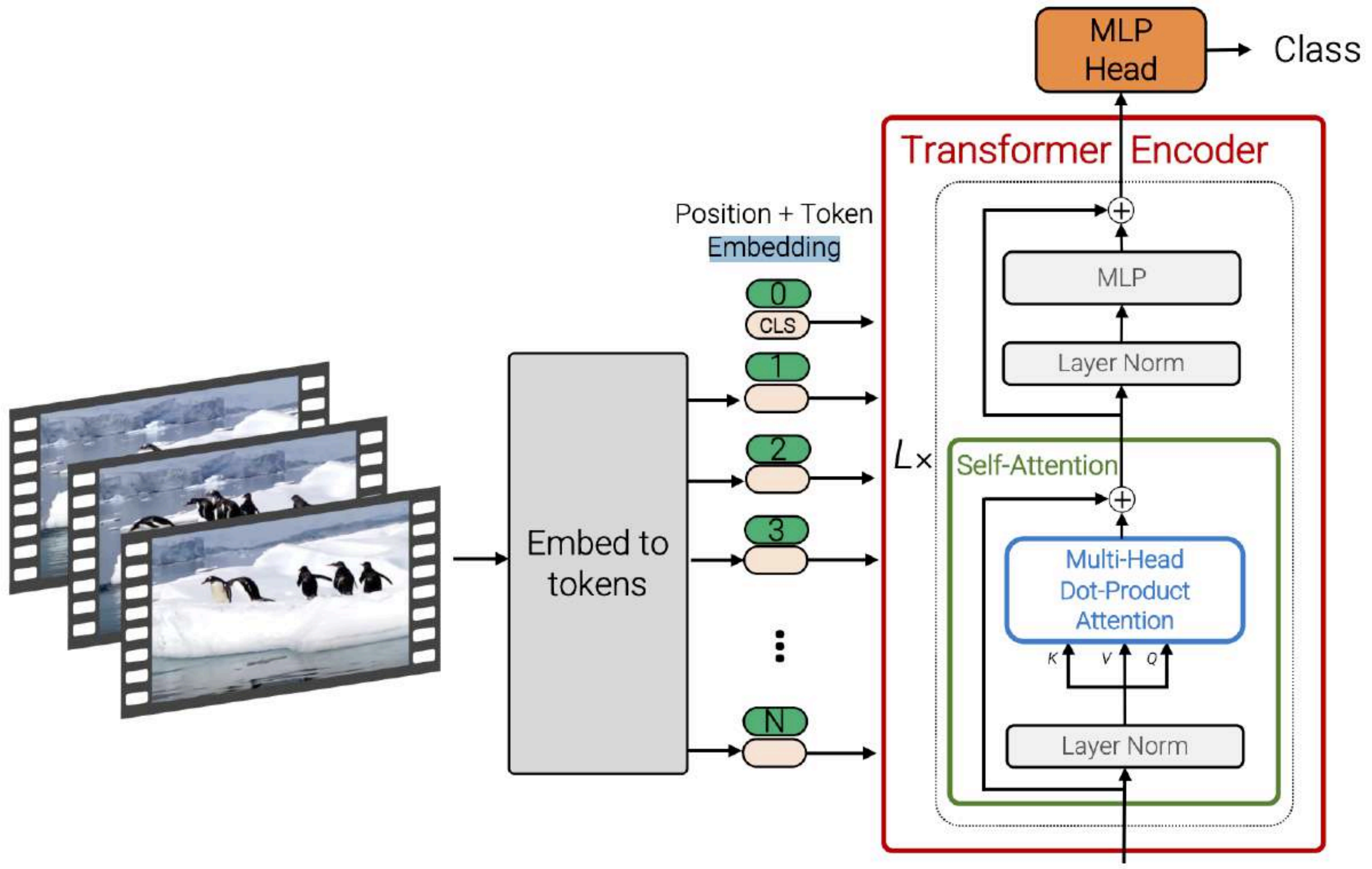
Can use weights of 2D convolution to initialize 3D conv -> copy K_t time in space and divide by K_t

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



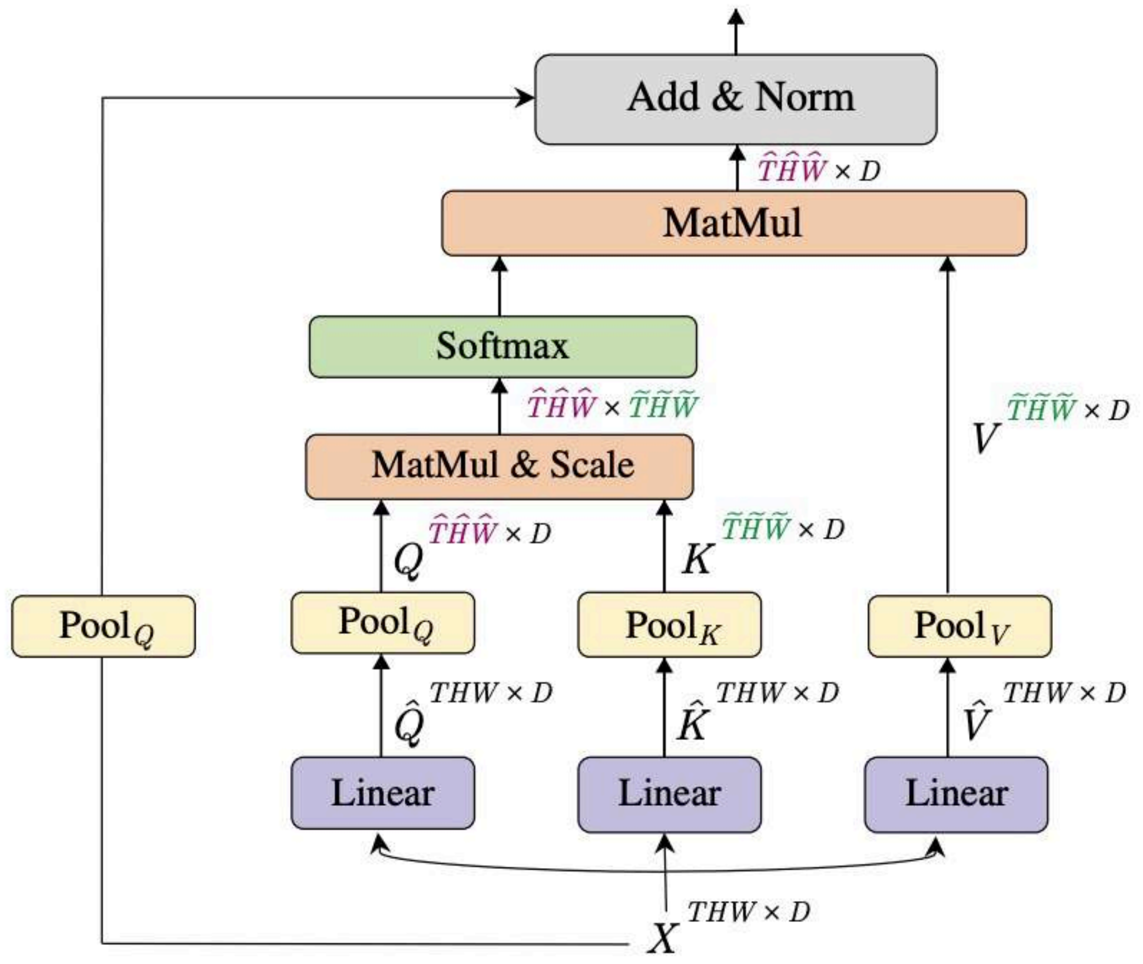
Vision Transformers for Video

Factorized attention
Attend over space/time



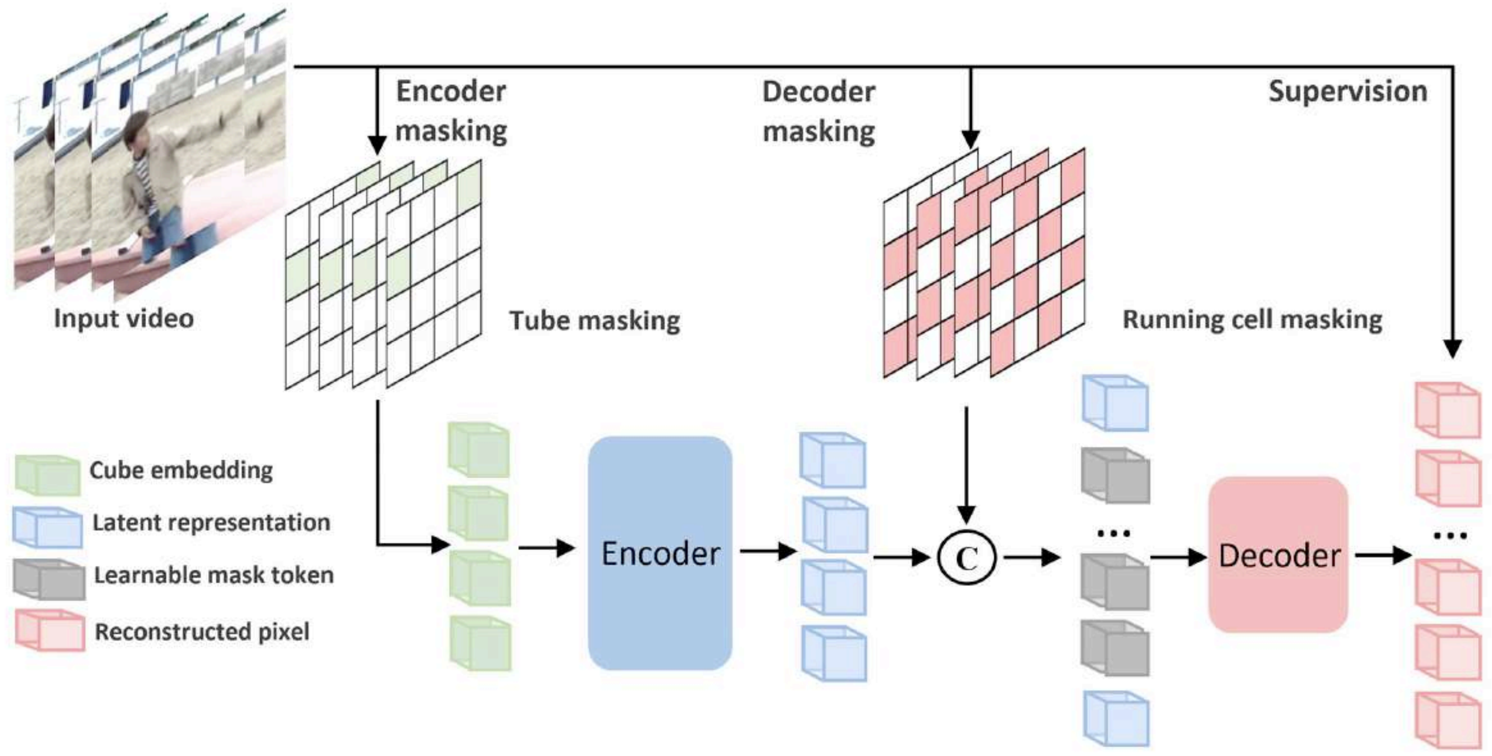
Bertasius et al, “Is Space-Time Attention All You Need for Video Understanding?”, ICML 2021
Arnab et al, “ViViT: A Video Vision Transformer”, ICCV 2021
Neimark et al, “Video Transformer Network”, ICCV 2021

Pooling module
Reduce number of tokens



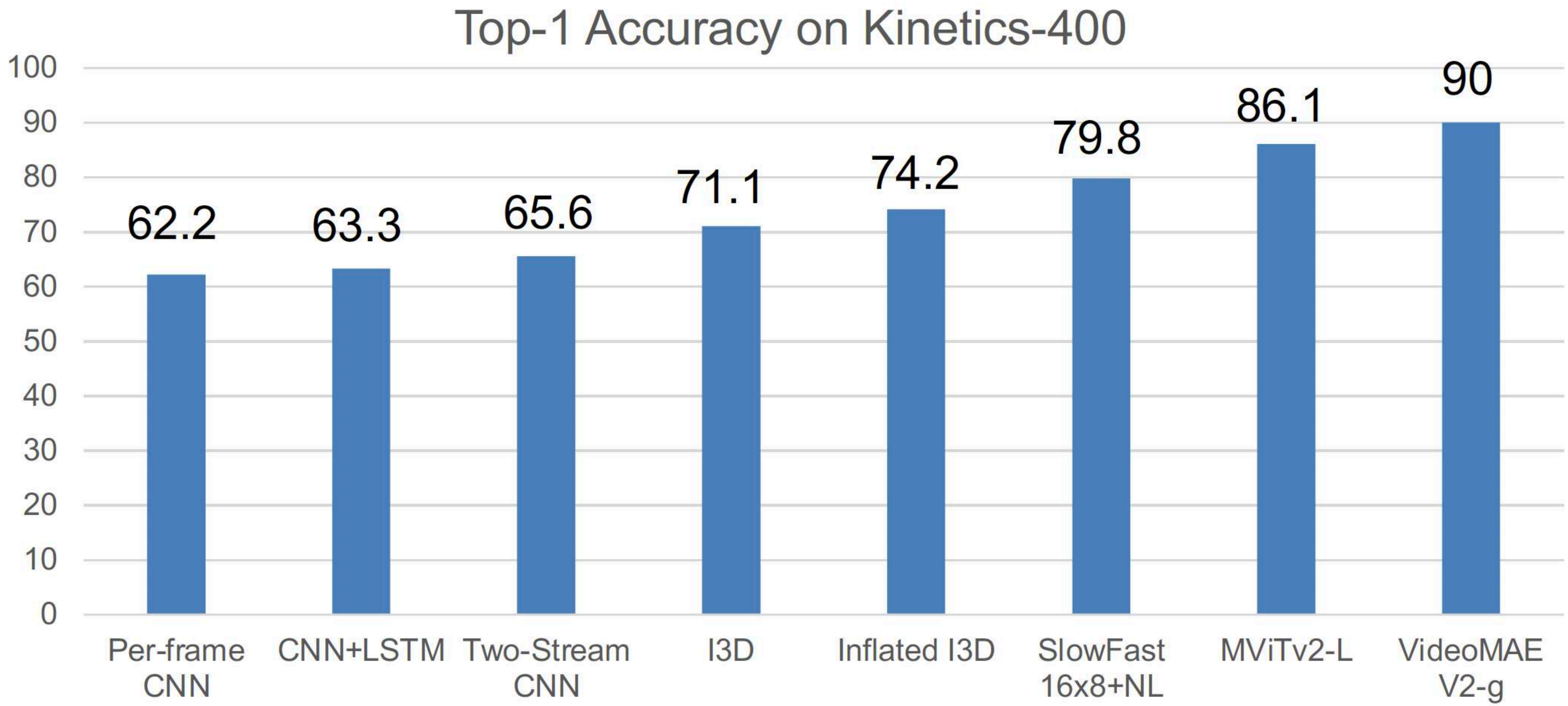
Fan et al, “Multiscale Vision Transformers”, ICCV 2021
Li et al, “MViTv2: Improved Multiscale Vision Transformers for Classification and Detection”, CVPR 2022

Video masked autoencoders
Efficient scalable pretraning



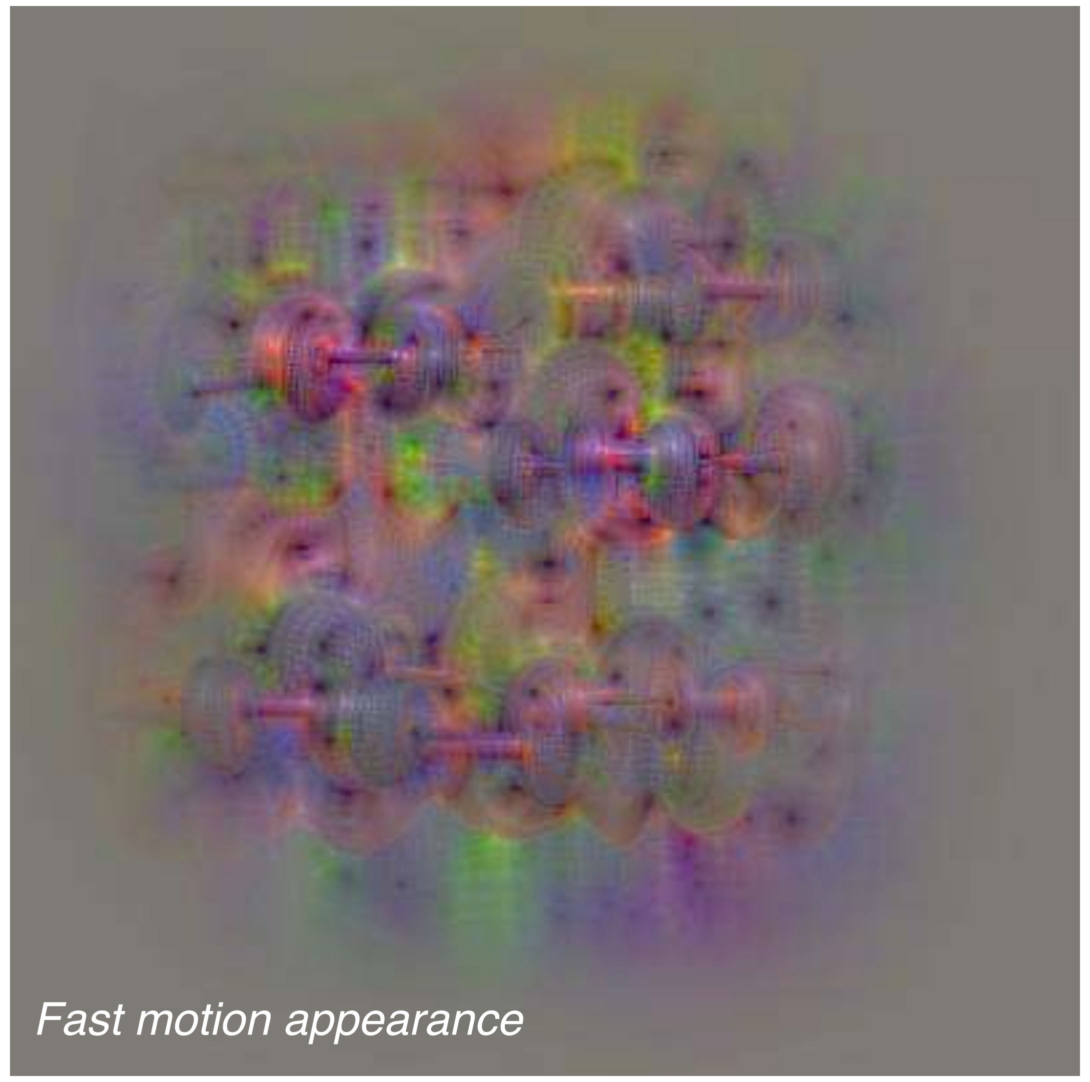
Wang et al. VideoMAE V2: Scaling Video Masked Autoencoders with Dual Making. CVPR 2023.
Tong et al. Video MAE: Masked Autoencoders are Data-Efficient Learners for Self-Supervised Video Pre-Training. NeurIPS 2022.
Feichtenhofer et al. Masked autoencoders as spatiotemporal learners. NeurIPS 2022.

Vision Transformers for Video

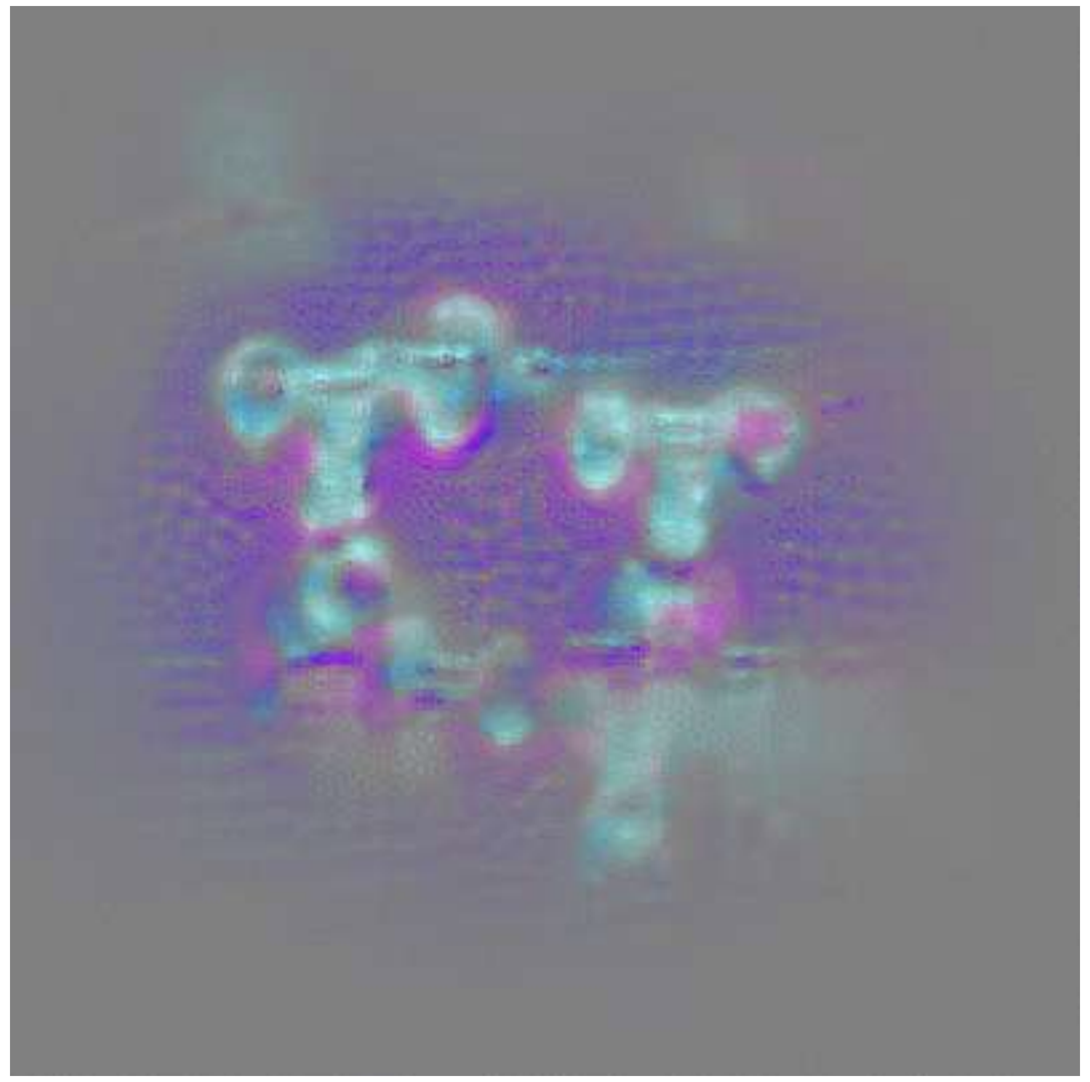


Can you guess the action? **Weightlifting**

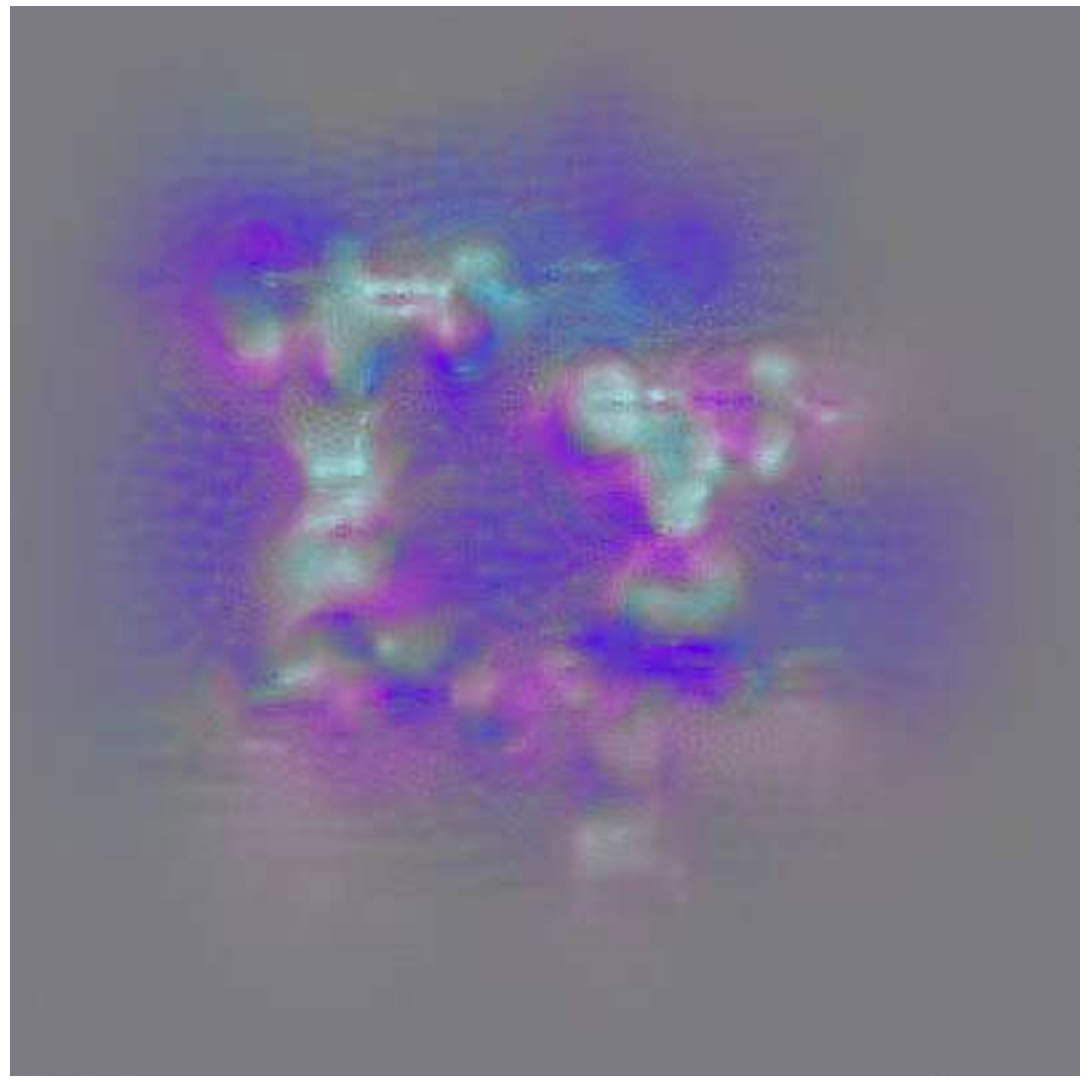
Appearance



Slow motion

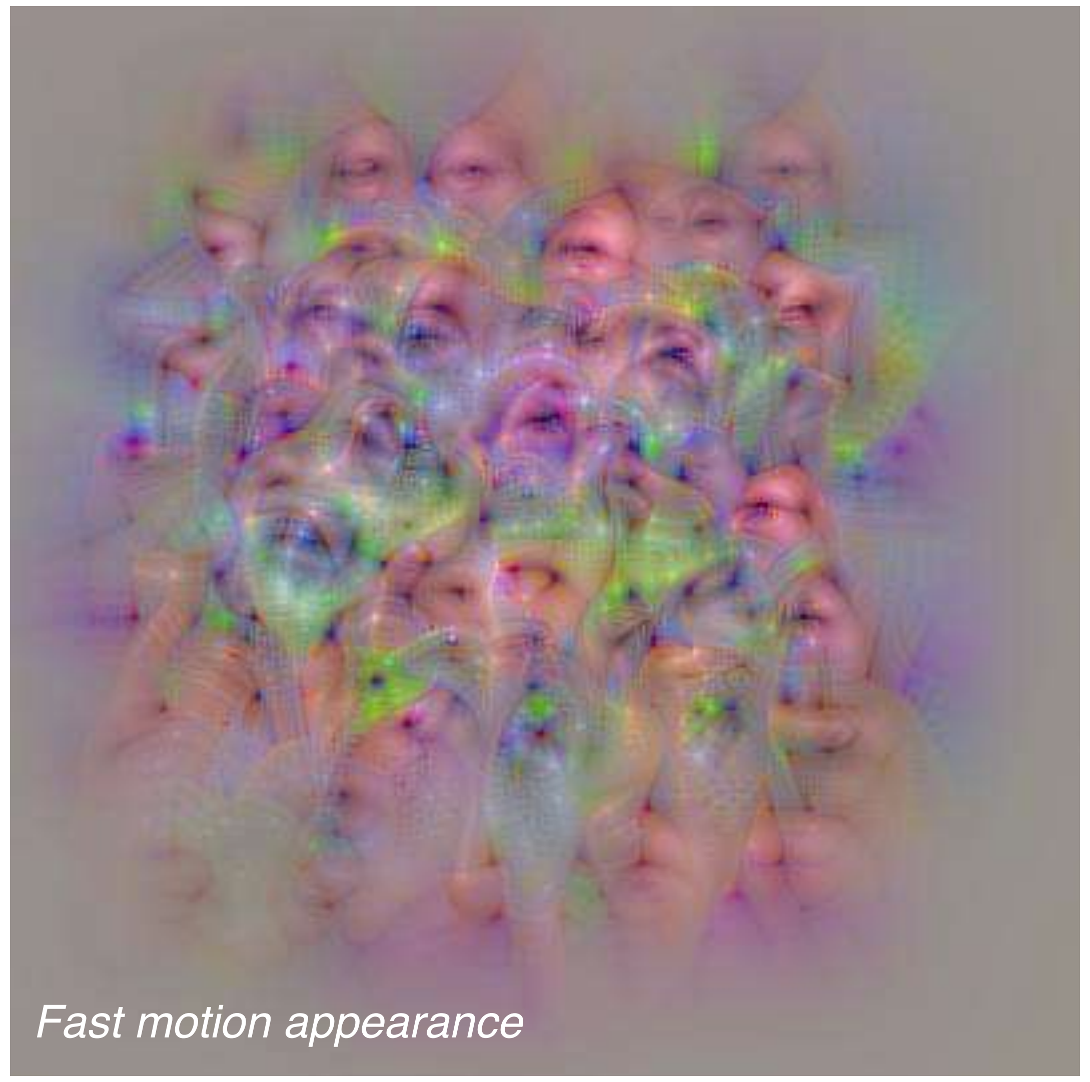


Fast motion



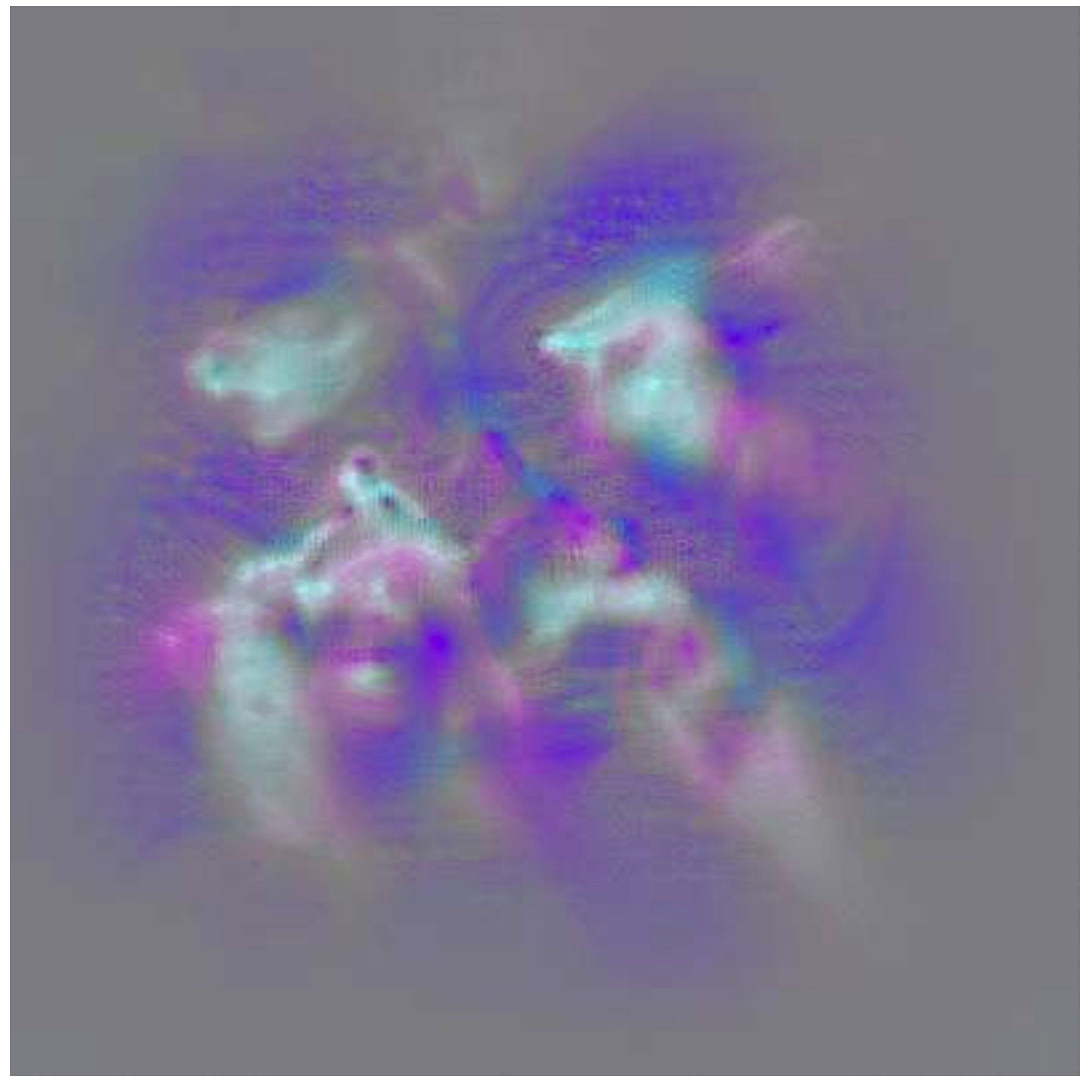
Can you guess the action? **Apply eye makeup**

Appearance

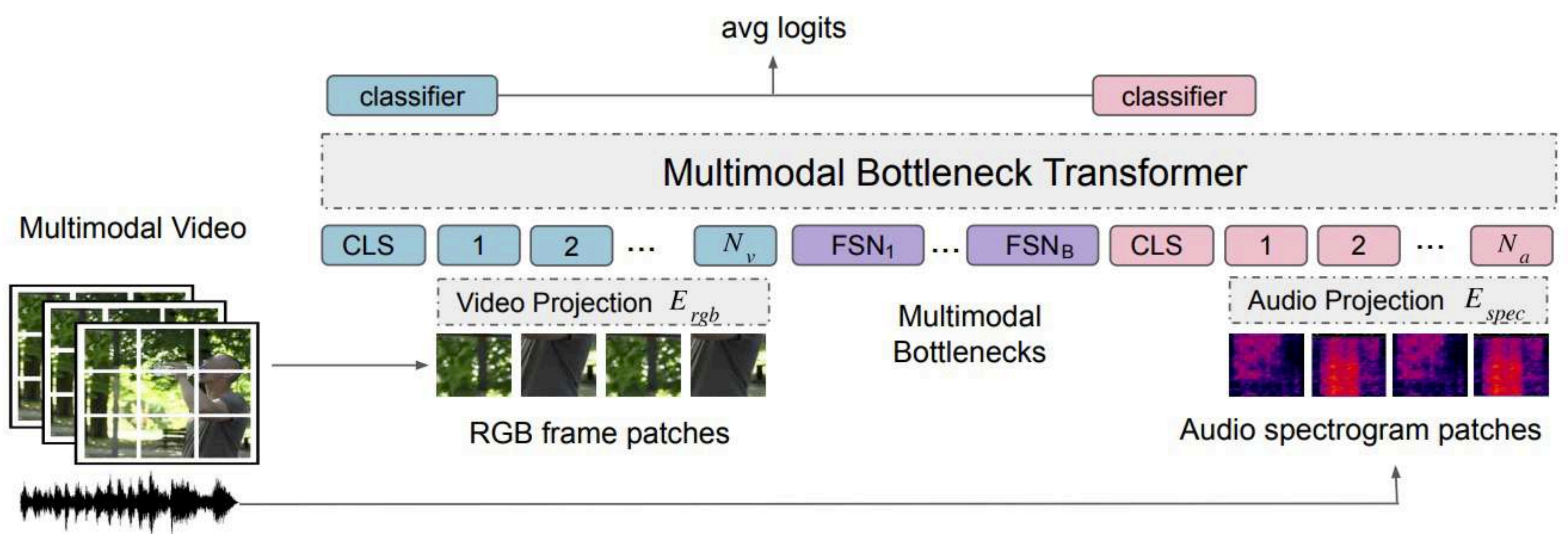


Slow motion

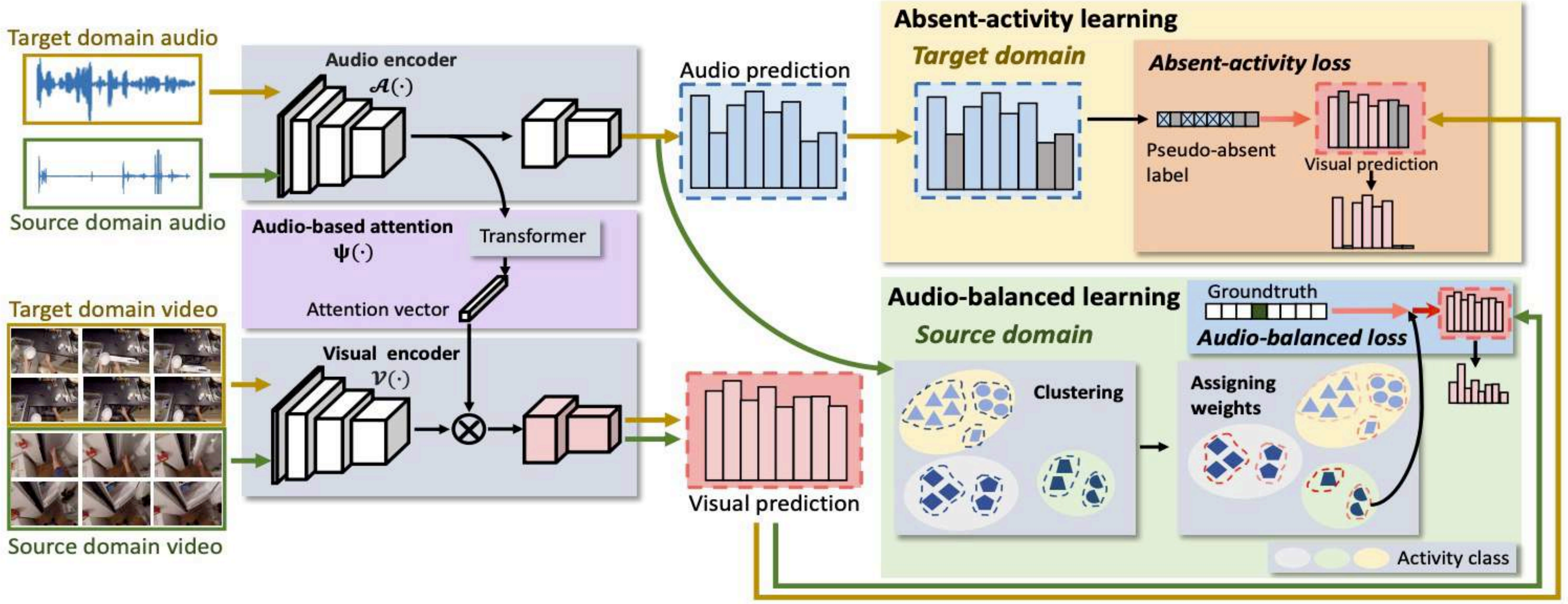
Fast motion



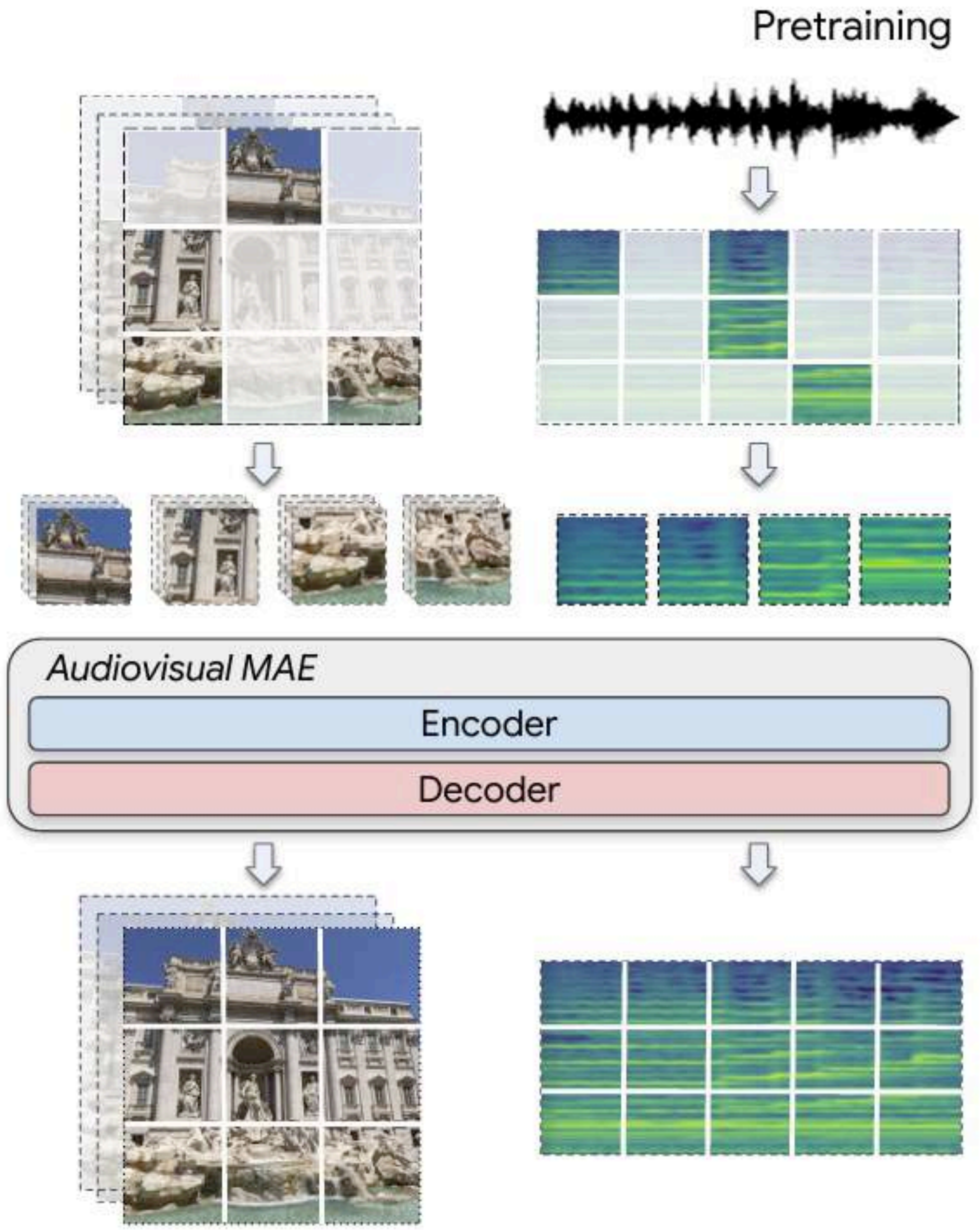
Video Understanding with Audio



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

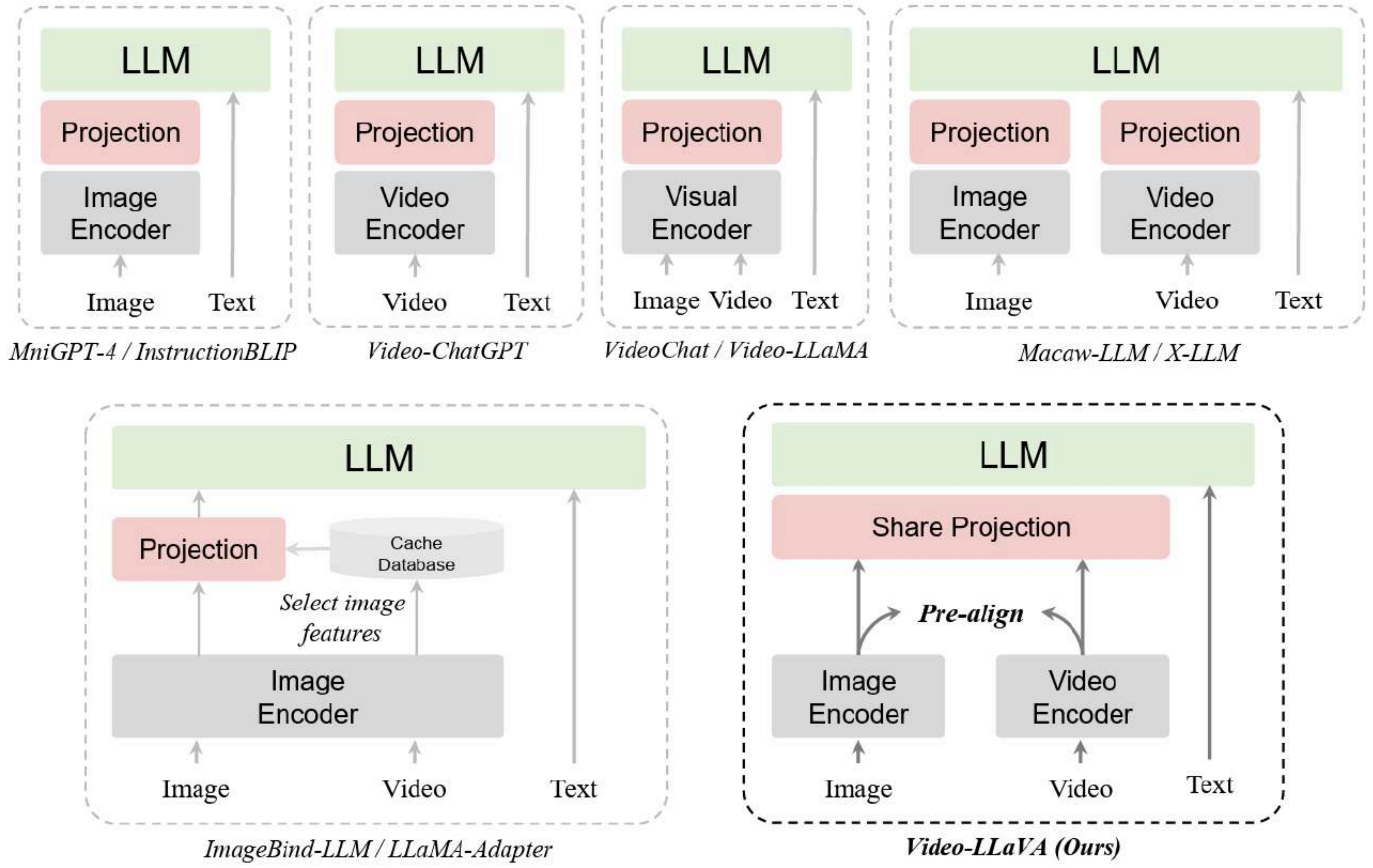


Audio-Adaptive Activity Recognition Across Video Domains,
Zhang et al. CVPR 2022

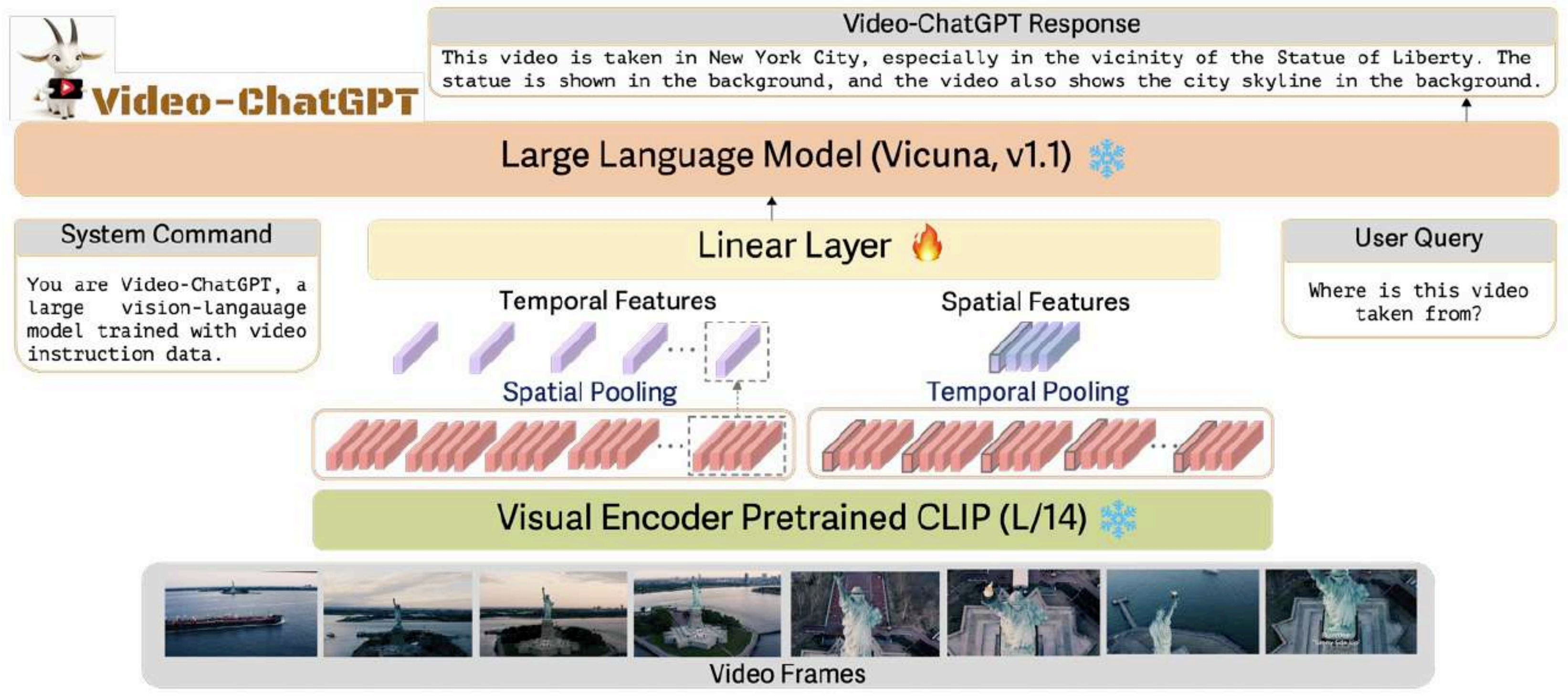


Audio-Visual Masked Autoencoders. Georgescu et al. ICCV 2023.

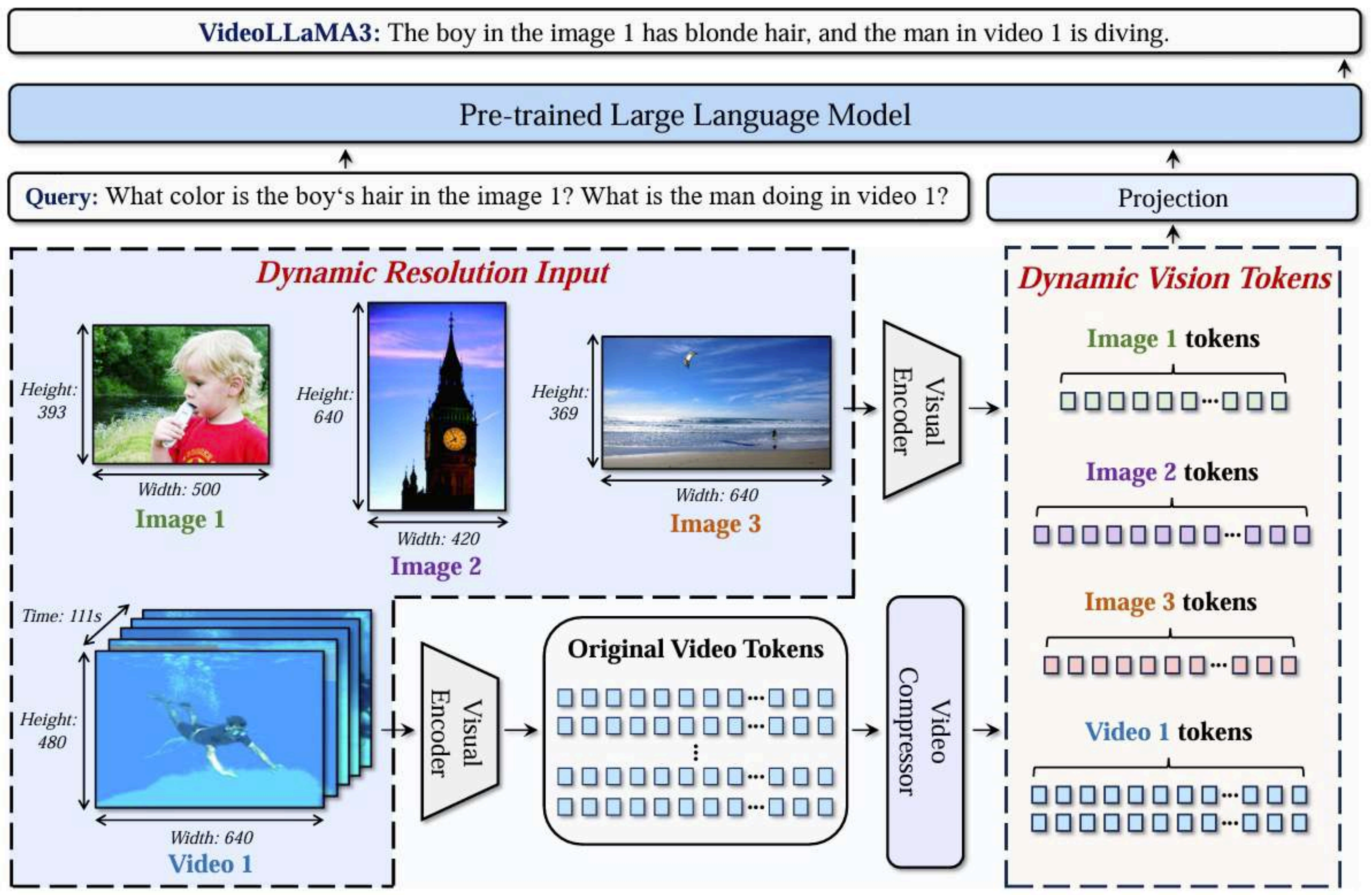
Video Understanding with (M)LLMs



Video-LLaVA: Learning United Visual Representations by Alignment Before Projection.
Lin et al. EMNLP 2024



Video-ChatGPT: Towards Detailed Video Understanding via Large Vision and Language Models.
Maaz et al. ACL 2024.



VideoLLaMa3: Frontier Multimodal Foundation Models for Video Understanding.
Zhang et al. arXiv 2025


That’s all folks!