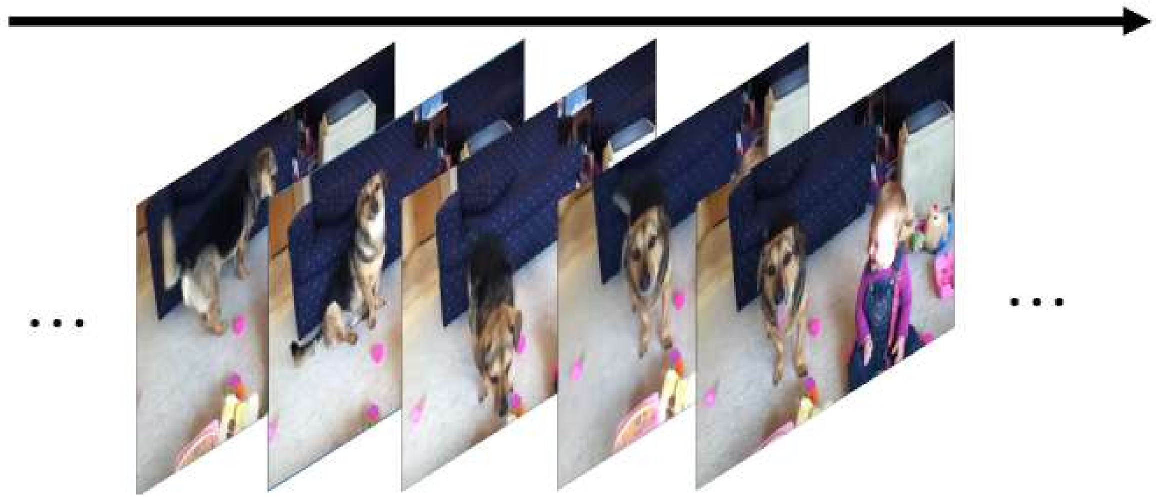
Visión Computacional

Ivan Sipiran

Video: 2D + Tiempo

Video como secuencia de imágenes
Tensor 4D: T x 3 x H x W



Video: Clasificación

Input video:
T x 3 x H x W

Swimming
Running
Jumping
Eating
Standing

Video

Input video:
T x 3 x H x W

Los videos son muy grandes!

~30 frames por segundo

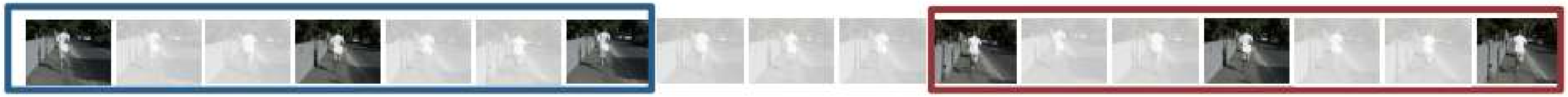
Tamaño de video descomprimido: 3 bytes por píxel

SD(640 x 480): ~1.5GB por minuto
HD(1920 x 1080): ~10GB por minuto

Entrenamiento sobre Clips

Video crudo: largo, alto FPS

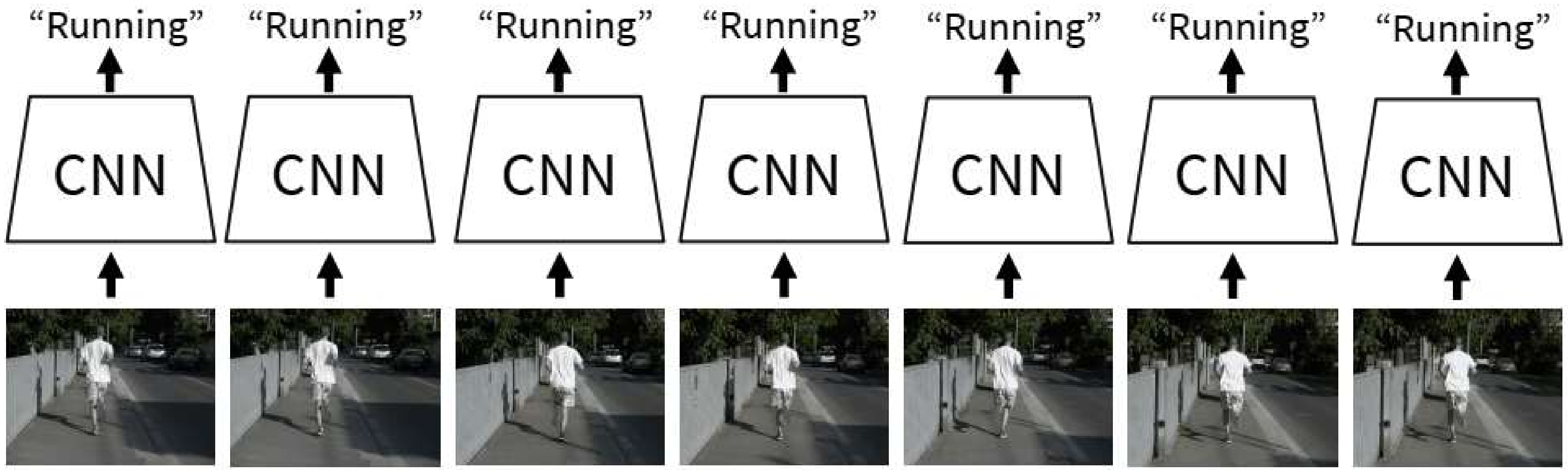
Entrenar modelo para clasificar clips cortos con bajo FPS



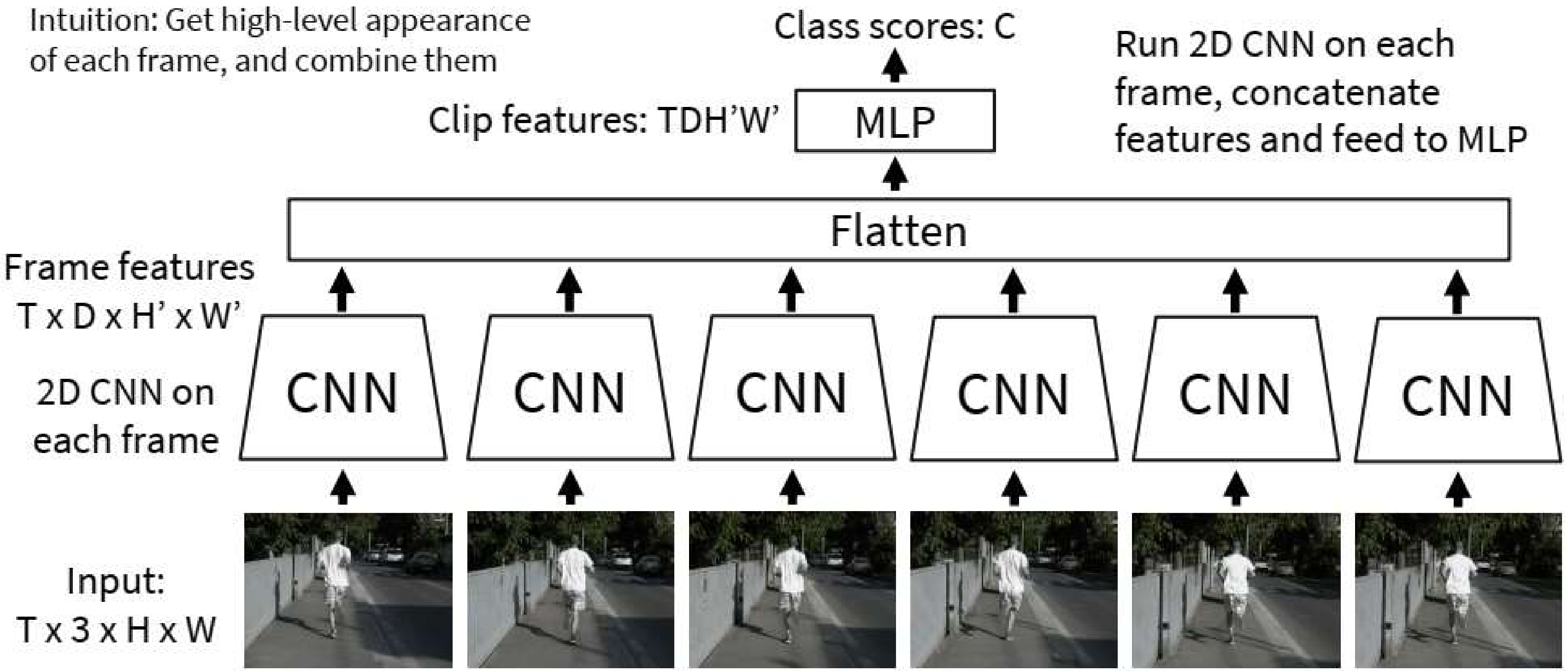
Test sobre otros clips

Clasificación de Videos: Single-frame CNN

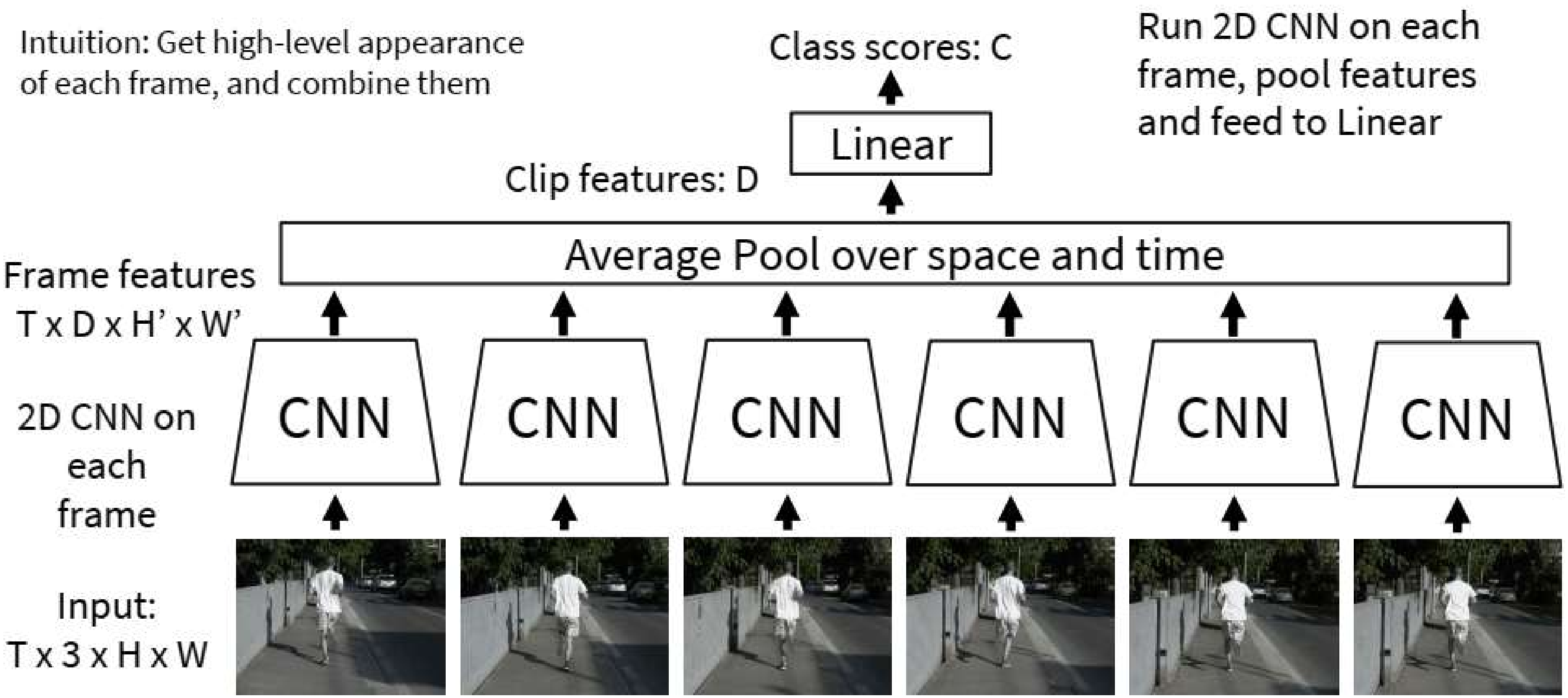
Idea simple: Entrenar un CNN normal para clasificar frames independientes
Durante test, se promedian las salidas de varios frames
Es un baseline bueno para clasificación de videos



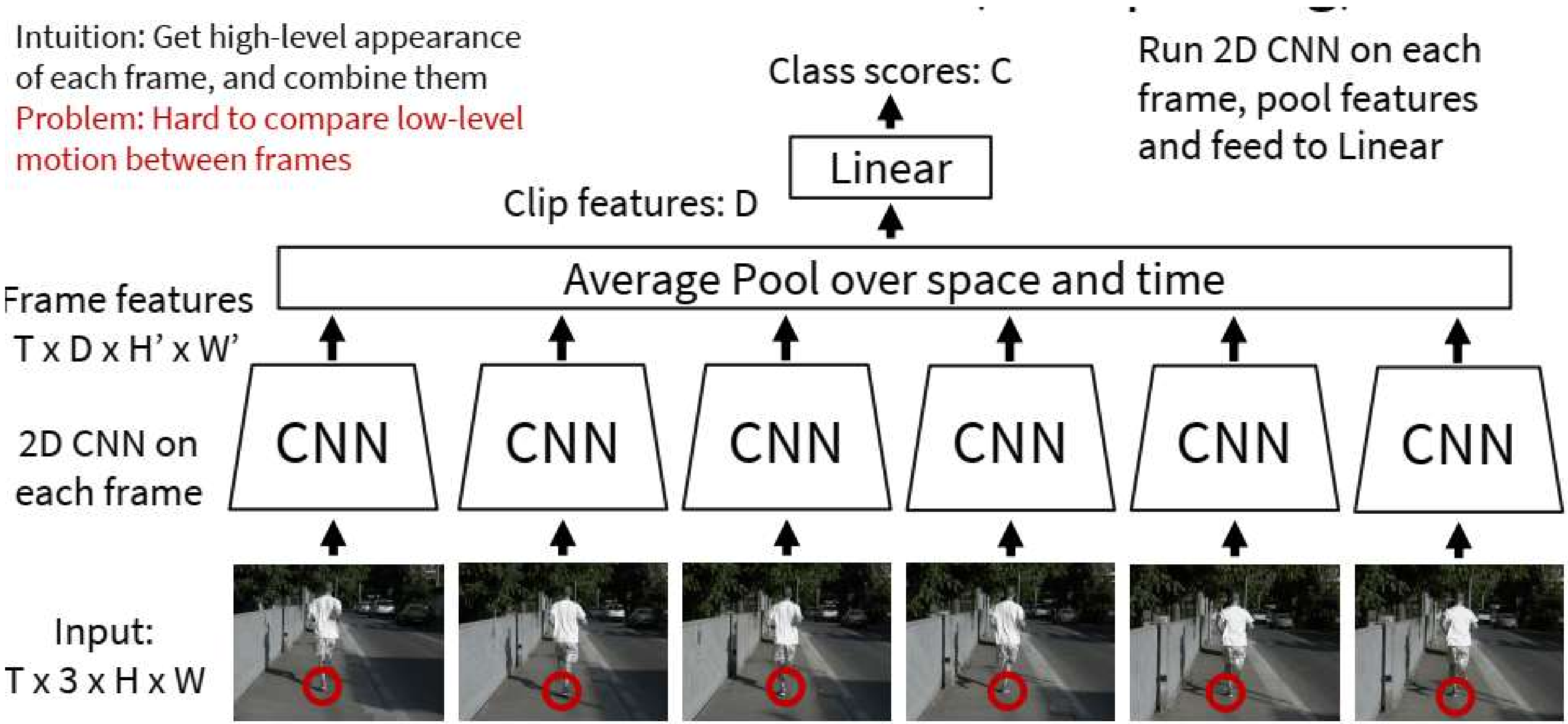
Clasificación de Videos: Late Fusion



Clasificación de Videos: Late Fusion (pooling)

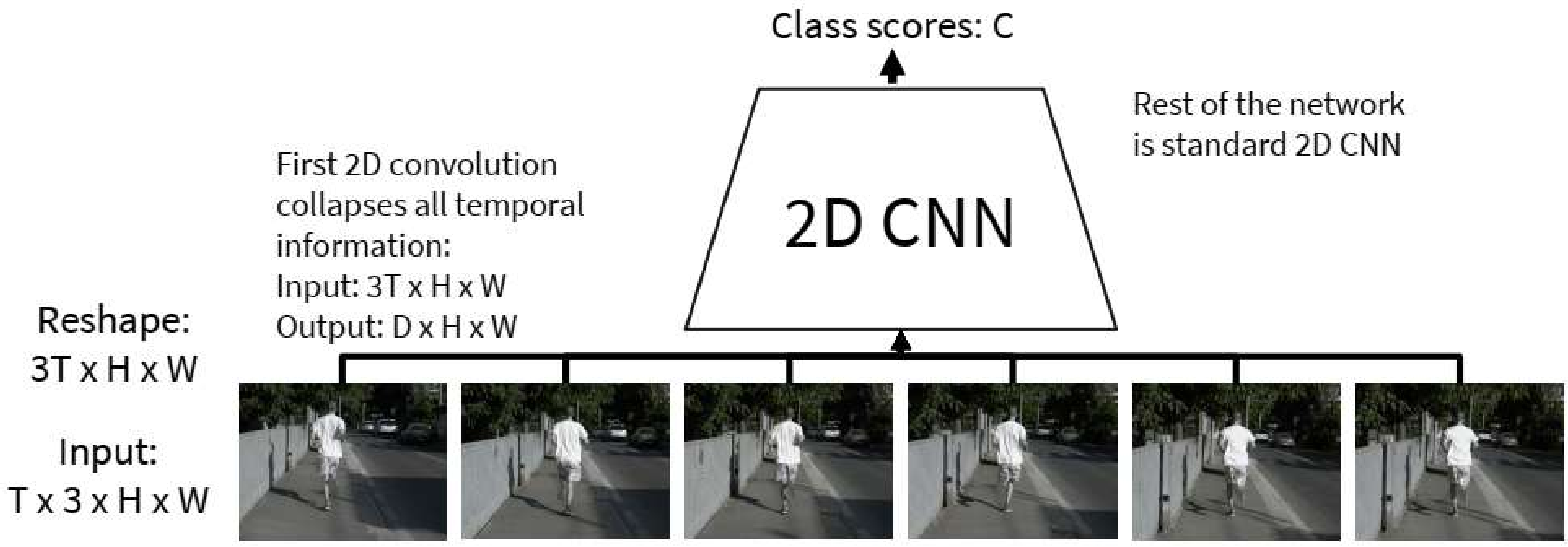


Clasificación de Videos: Late Fusion (pooling)



Clasificación de Videos: Early Fusion

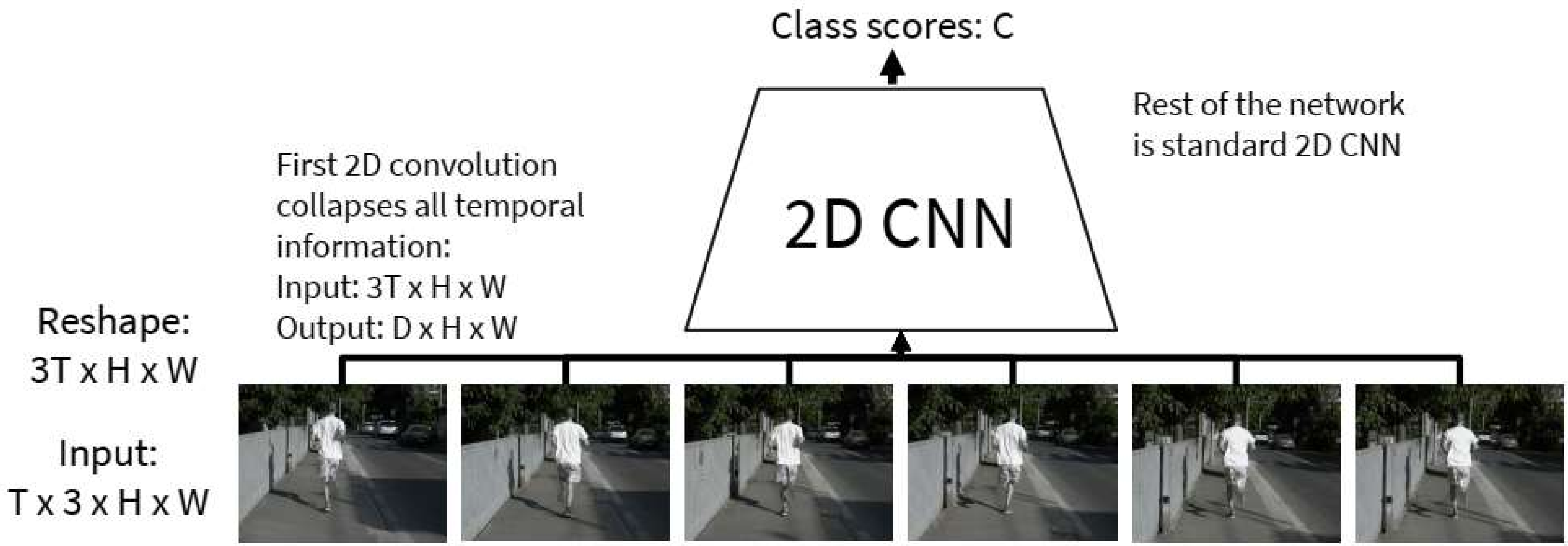
Comparar frames con primeras capas convolucionales, después pasar por CNN



Clasificación de Videos: Early Fusion

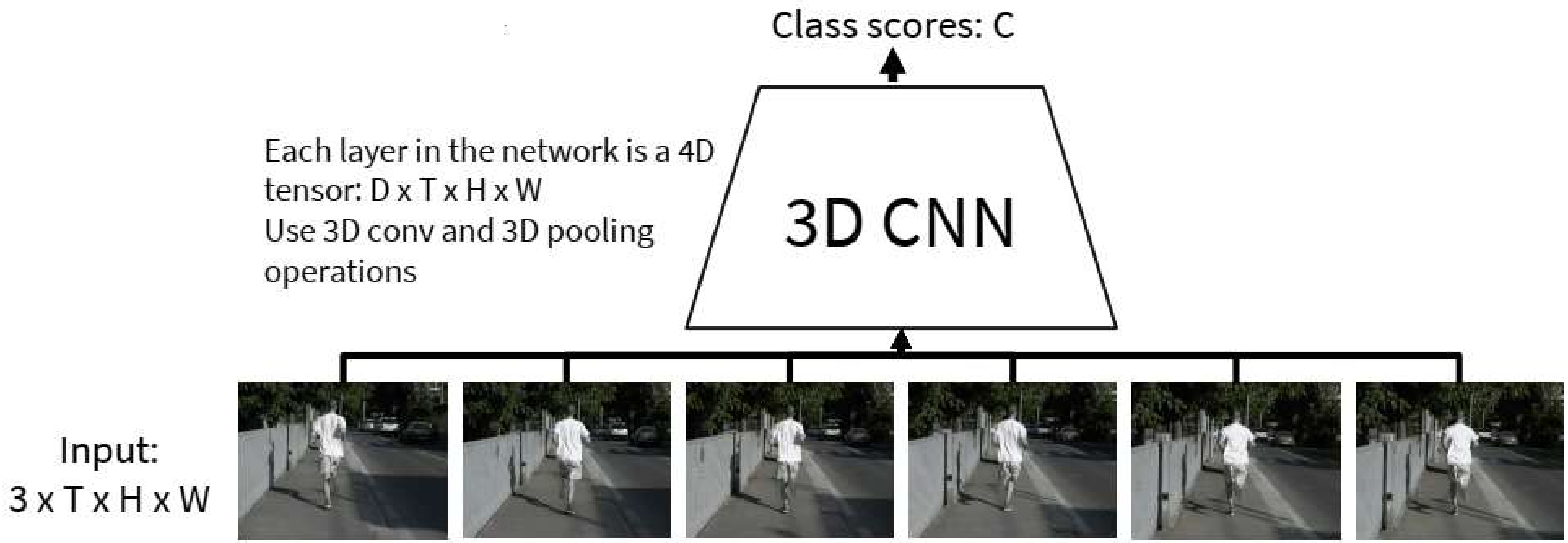
Comparar frames con primeras capas convolucionales, después pasar por CNN

Problema: un solo procesamiento temporal puede ser insuficiente

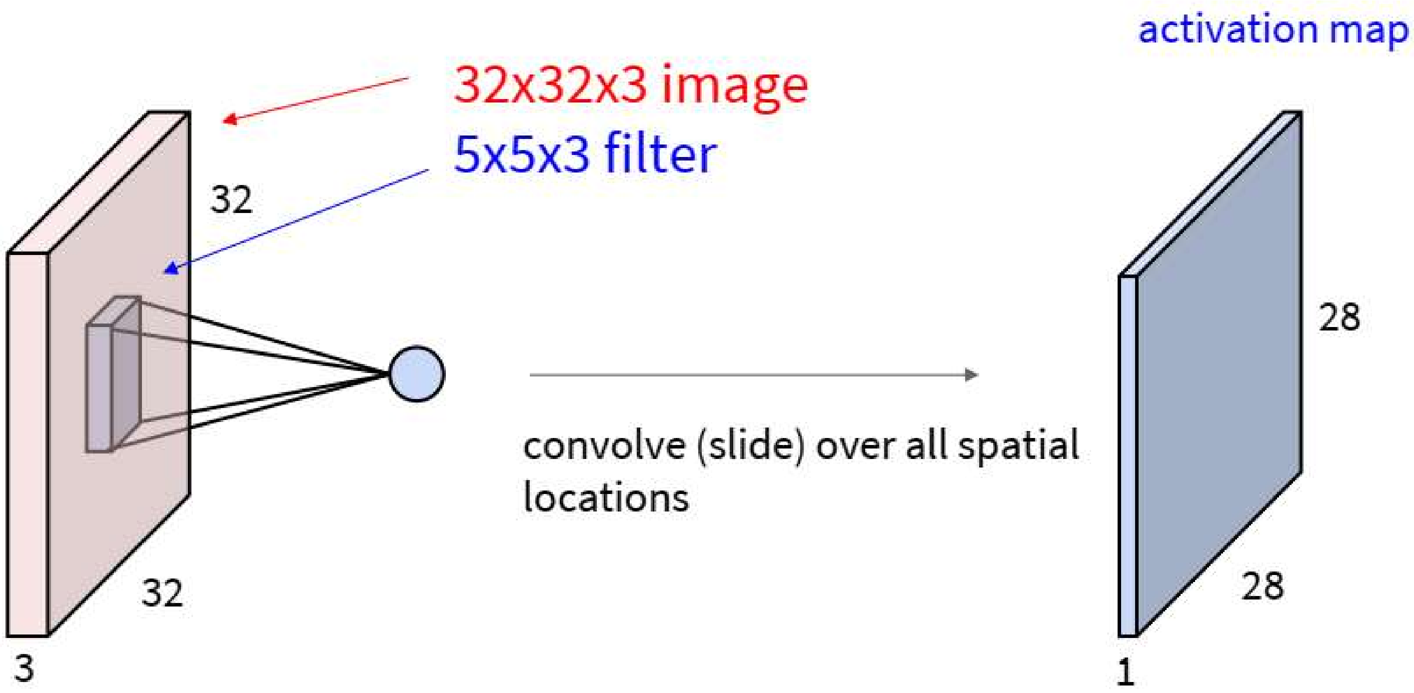


Clasificación de Videos: 3D CNN

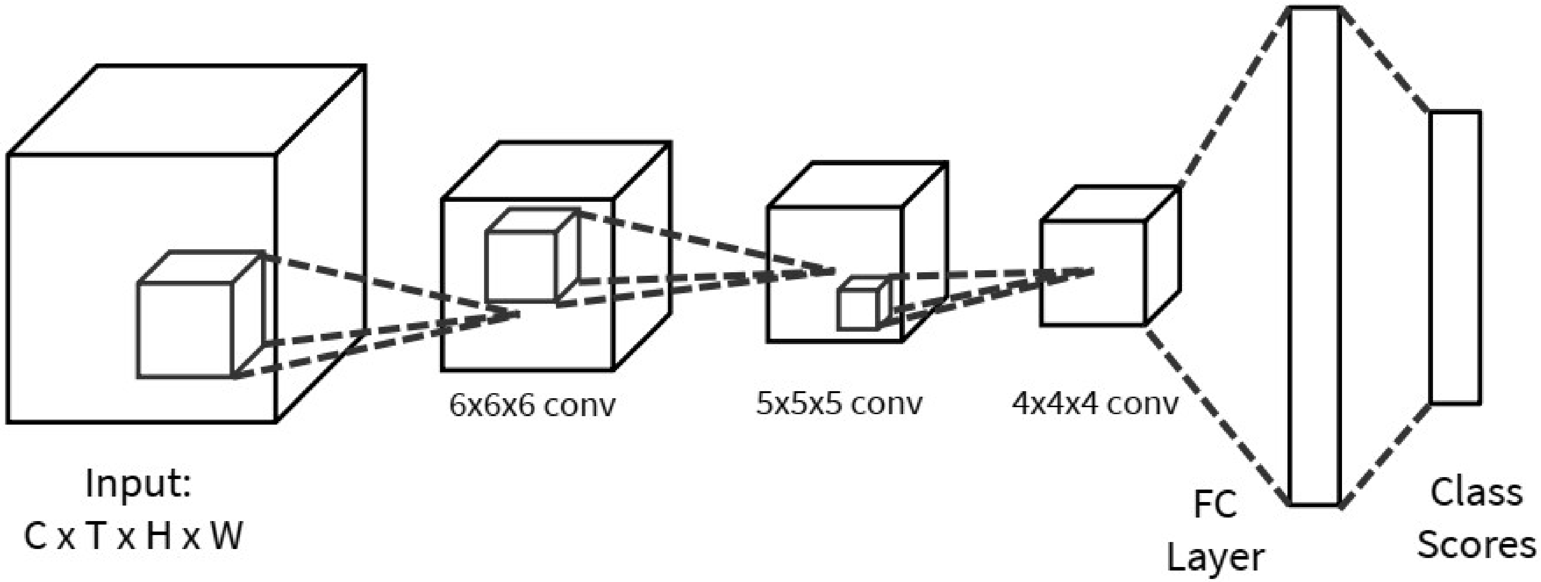
Usar versión 3D de convolución y pooling para fusionar información temporal



Capa Convolucional



Convolución 3D



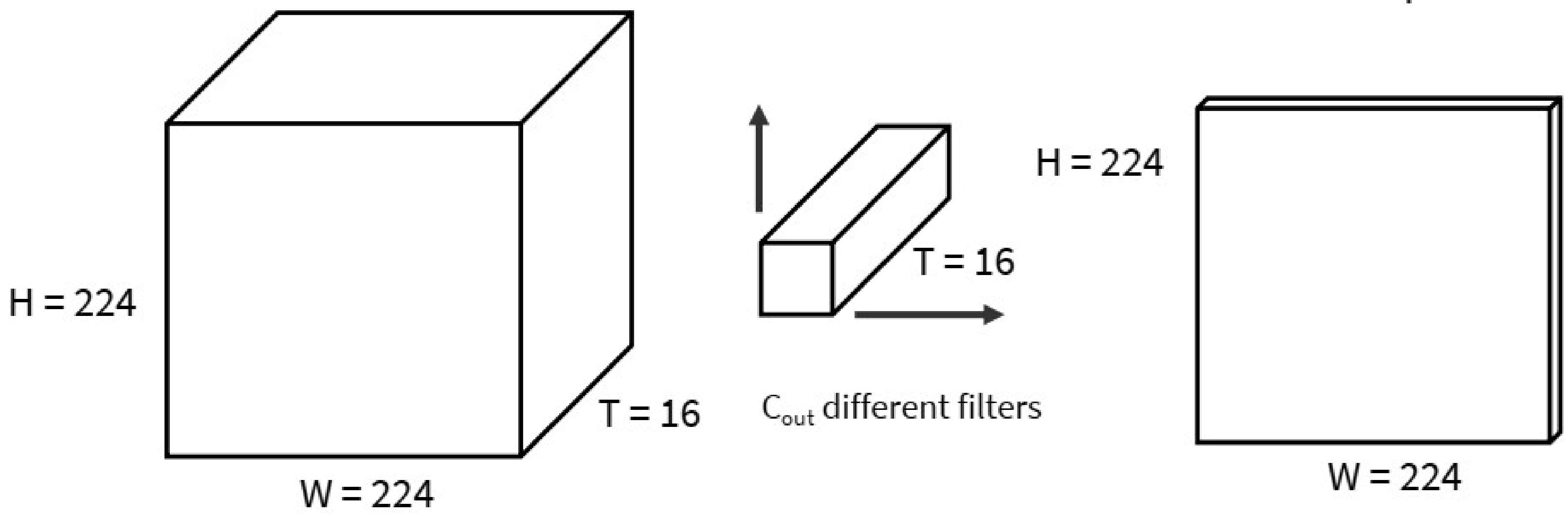
Convolución 3D

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



Convolución 3D

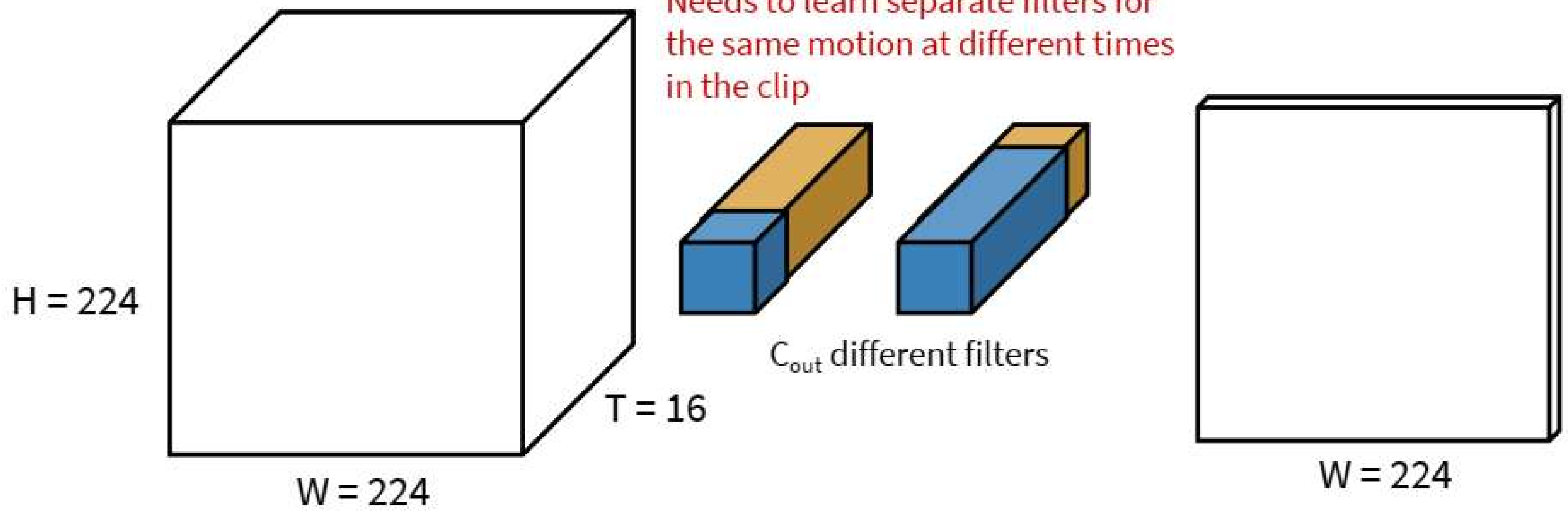
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

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



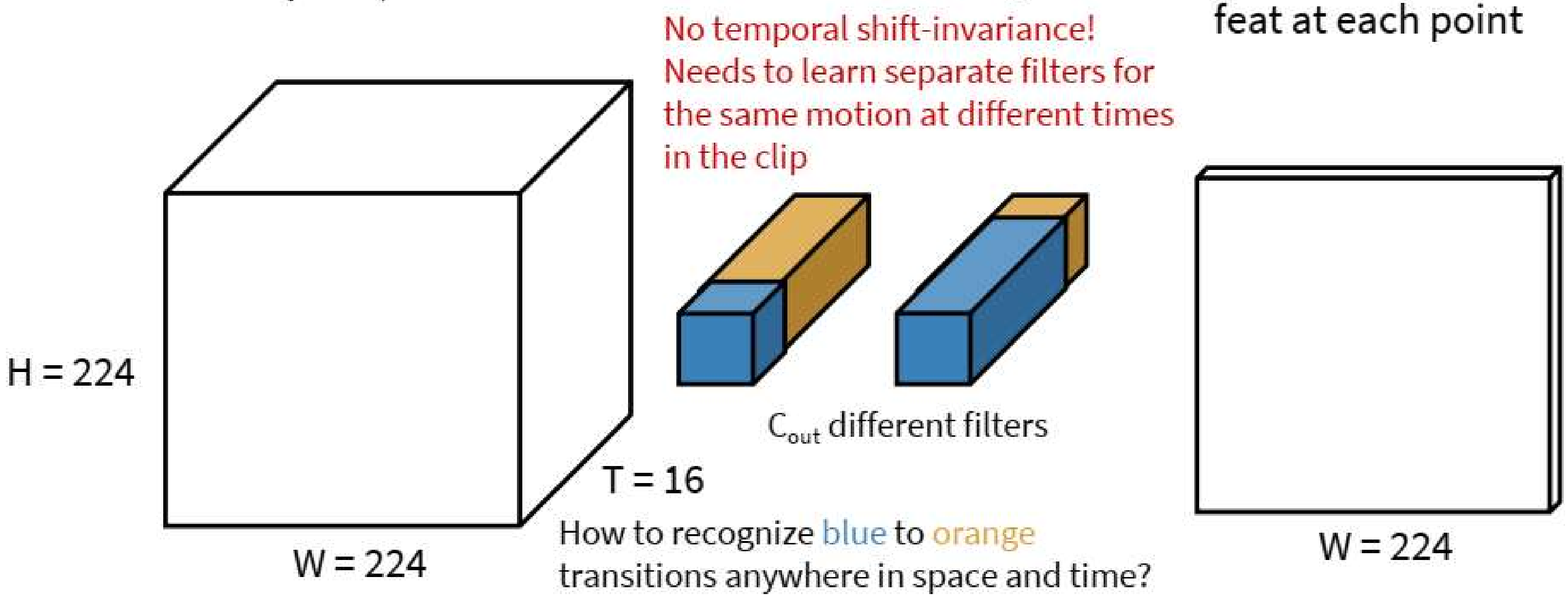
Convolución 3D

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



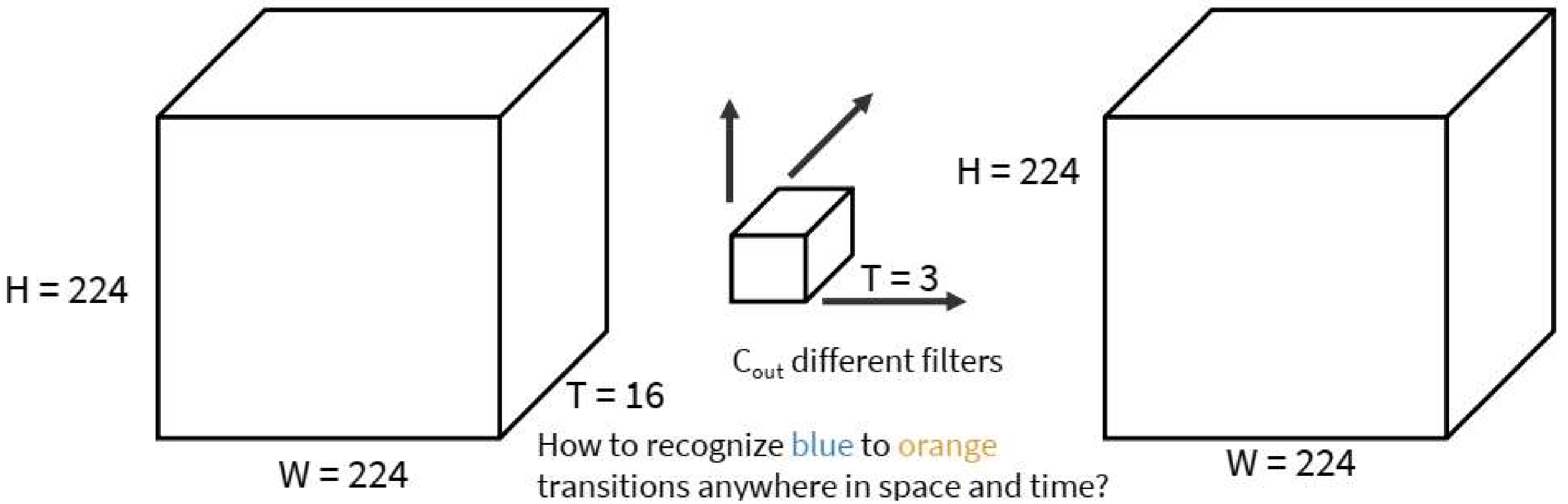
Convolución 3D

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

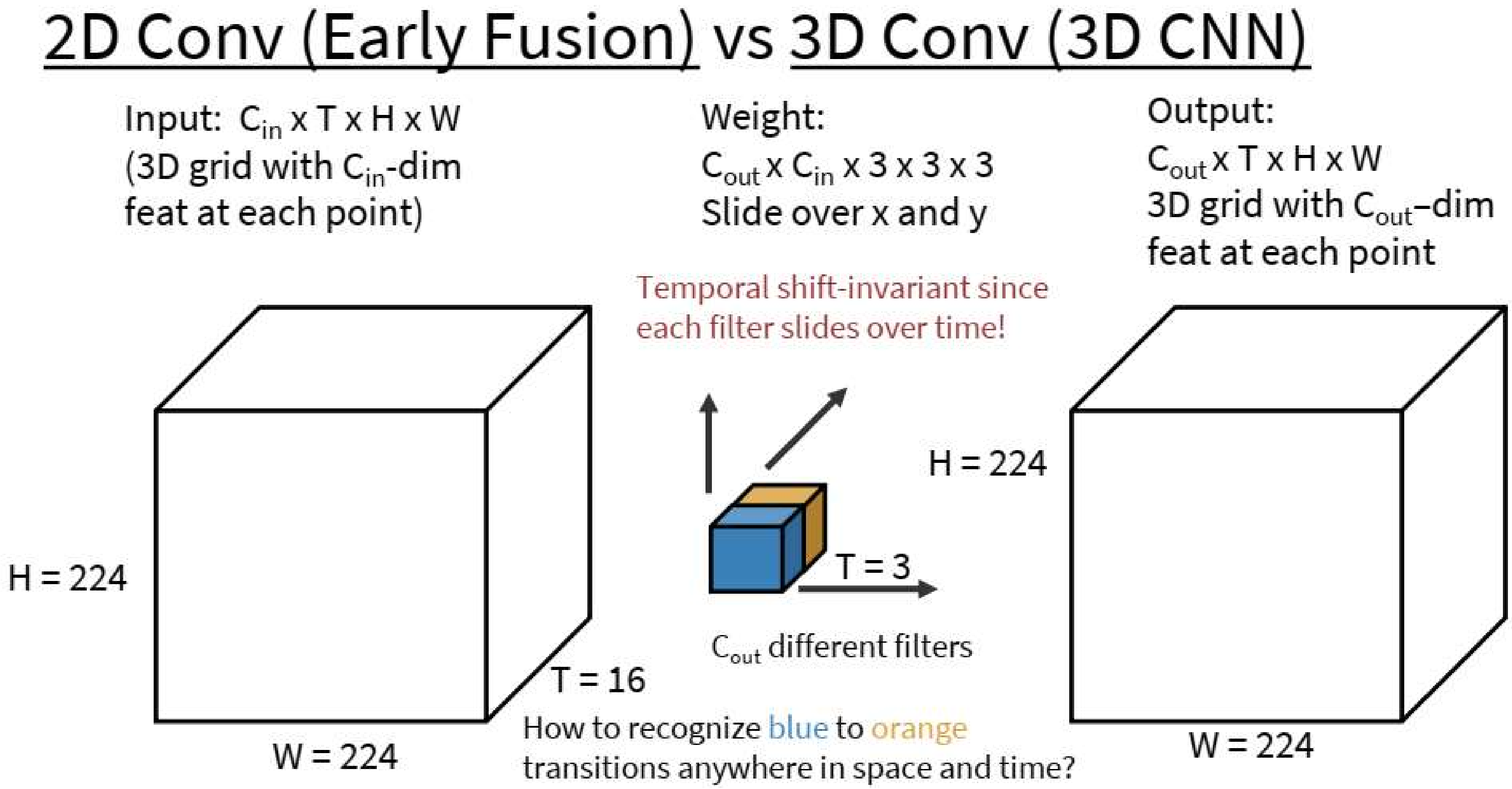
Input: C_{in} x T x H x W
(3D grid with C_{in}-dim
feat at each point)

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

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



Convolución 3D

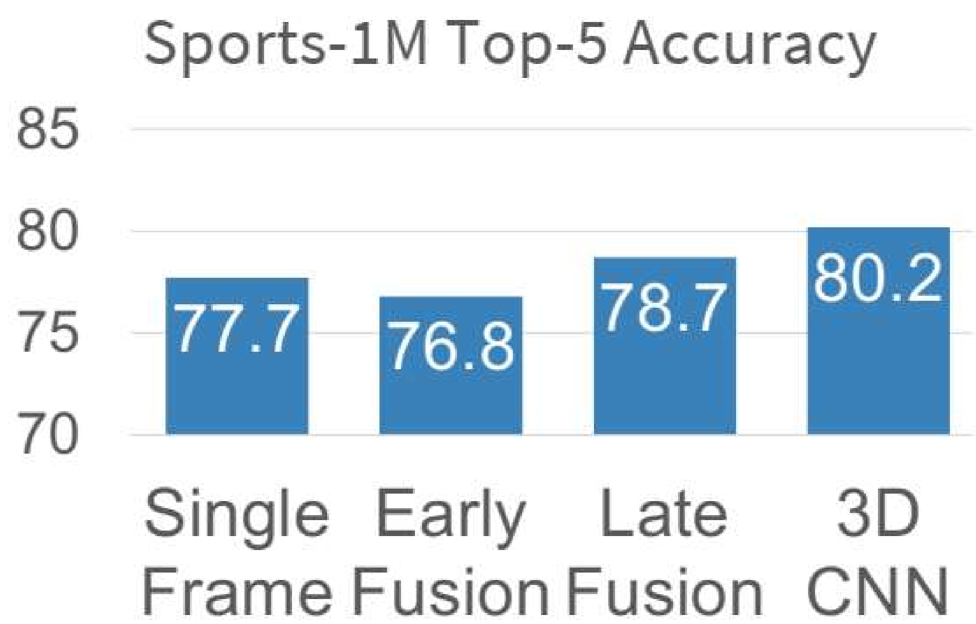


Video Dataset – Sport 1M

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

Ground Truth
Correct prediction
Incorrect prediction

Video Dataset – Sport 1M



C3D: El VGG de 3D CNN

3D CNN que usa solo convoluciones 3x3x3 y pooling 2x2x2

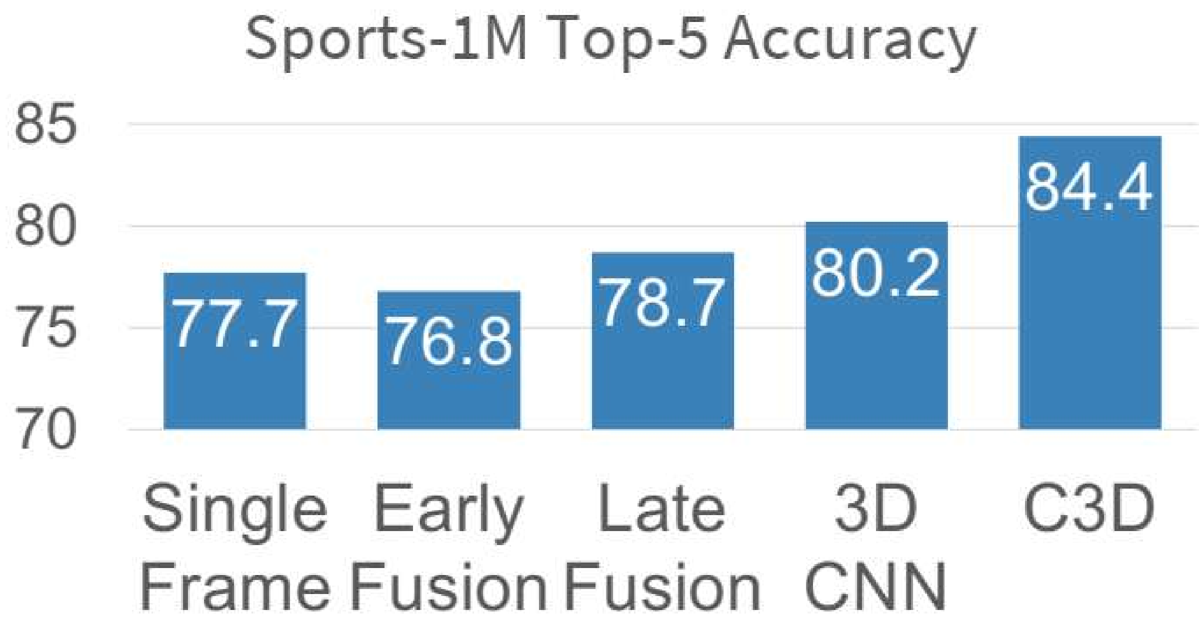
Modelo pre-entrenado en Sport-1M: se usa como feature extractor

Problema: convolución 3x3x3 es muy costoso

AlexNet: 0.7 GFLOP
VGG-16: 13.6 GFLOP
C3D: 39.5 GFLOP (~3x VGG)

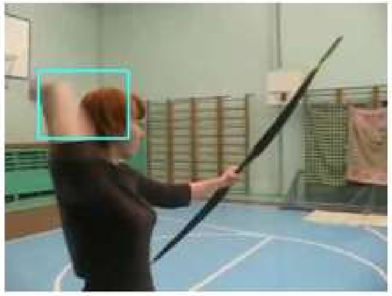
Layer	Size
Input	3 x 16 x 112 x 112
Conv1 (3x3x3)	64 x 16 x 112 x 112
Pool1 (1x2x2)	64 x 16 x 56 x 56
Conv2 (3x3x3)	128 x 16 x 56 x 56
Pool2 (2x2x2)	128 x 8 x 28 x 28
Conv3a (3x3x3)	256 x 8 x 28 x 28
Conv3b (3x3x3)	256 x 8 x 28 x 28
Pool3 (2x2x2)	256 x 4 x 14 x 14
Conv4a (3x3x3)	512 x 4 x 14 x 14
Conv4b (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: El VGG de 3D CNN



Midiendo el movimiento

Image at frame t



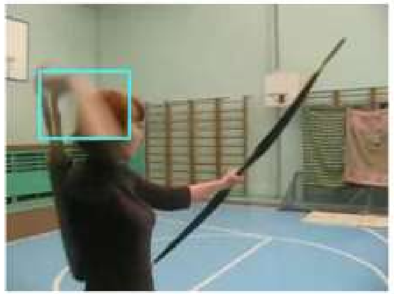
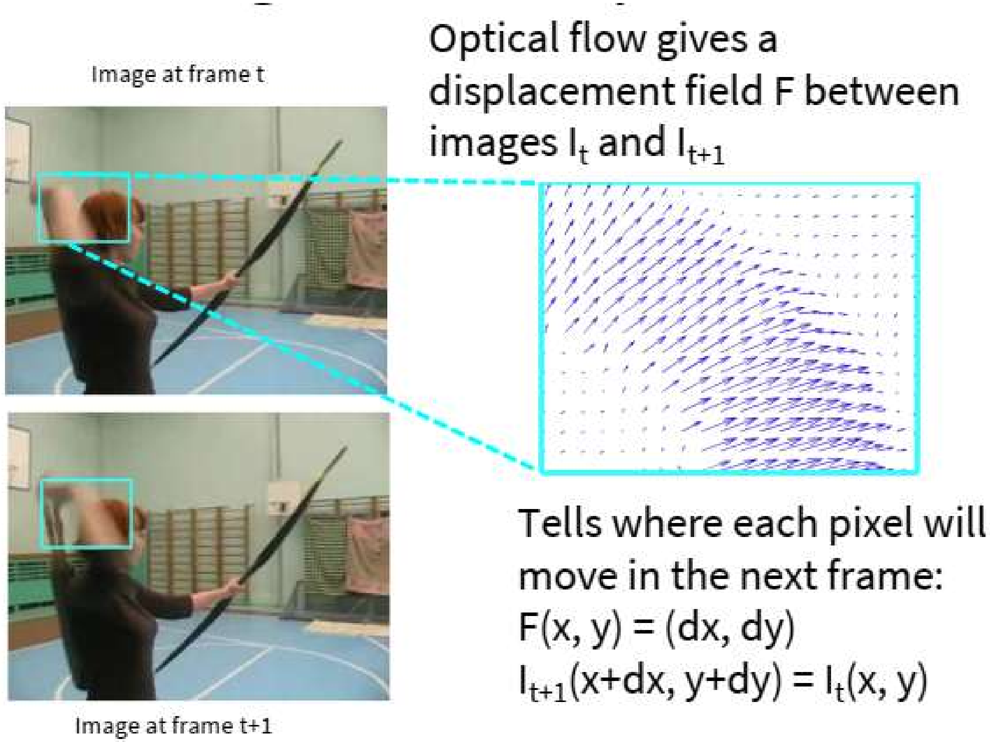


Image at frame t+1

Midiendo el movimiento



Midiendo el movimiento

Image at frame t

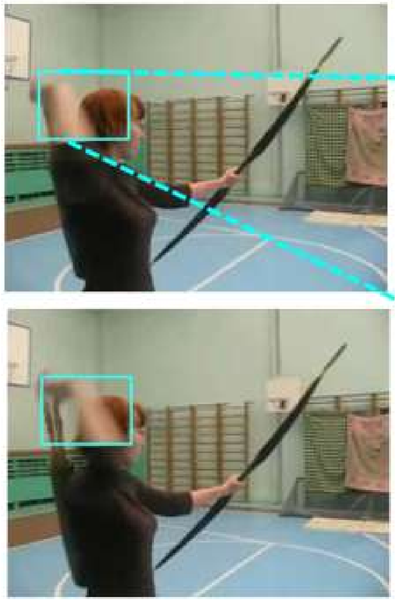
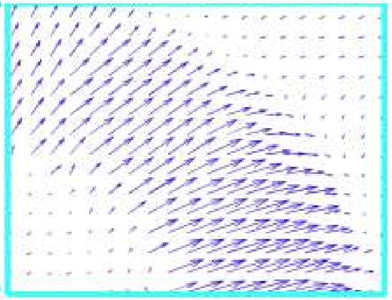


Image at frame t+1

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



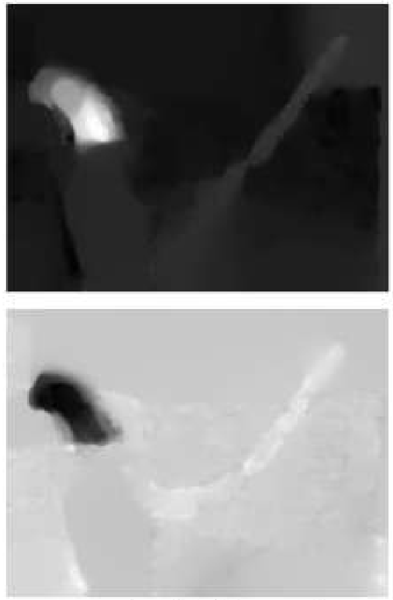
Tells where each pixel will move in the next frame:

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

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

Optical Flow highlights local motion

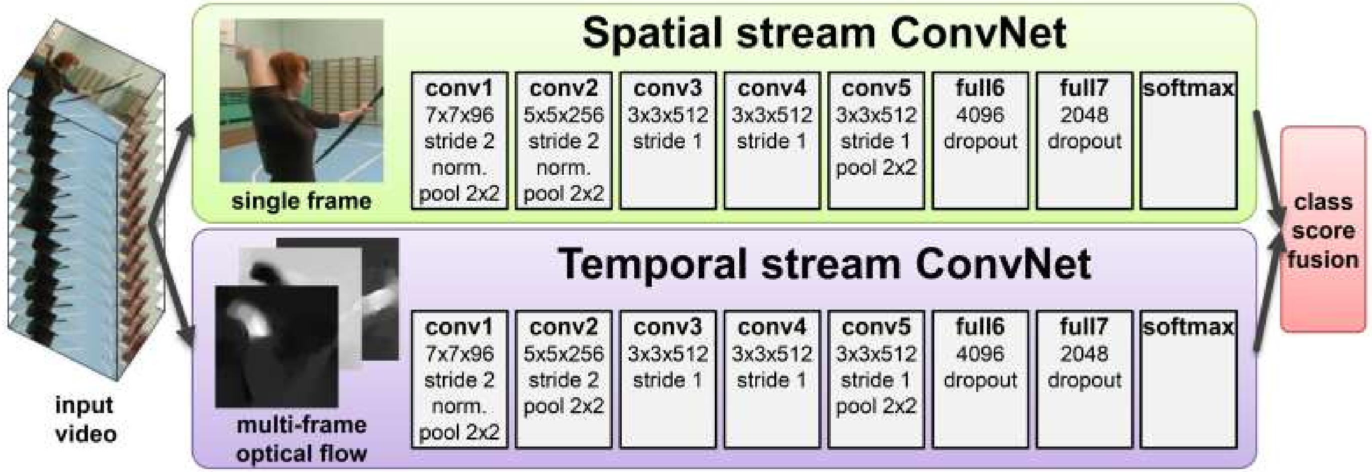
Horizontal flow dx



Vertical Flow dy

Movimiento y Apariencia

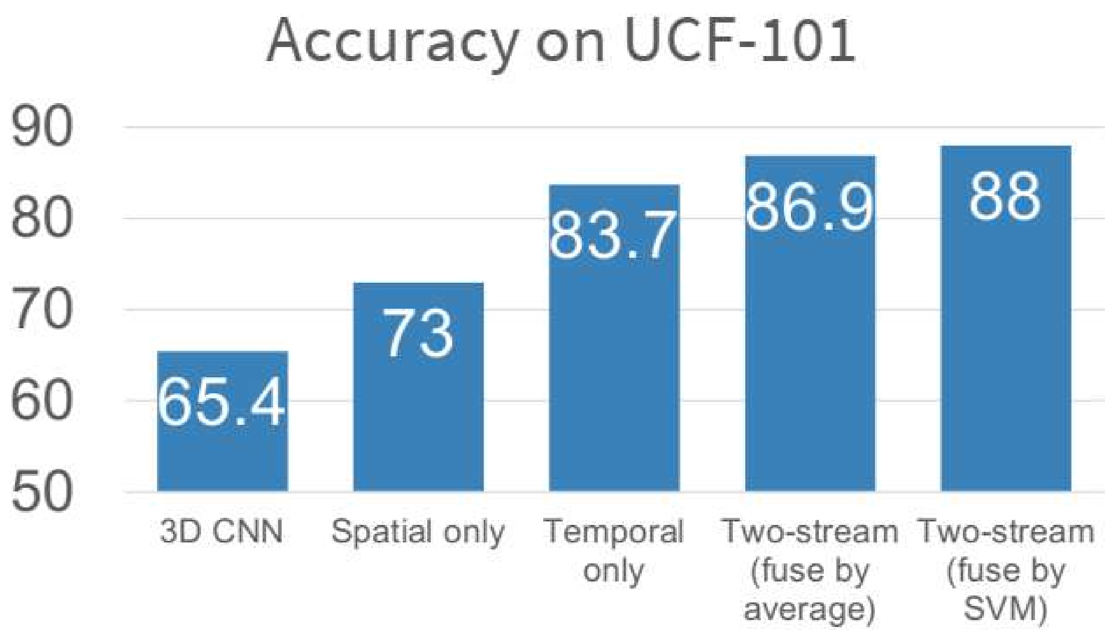
Input: Single Image
3 x H x W



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

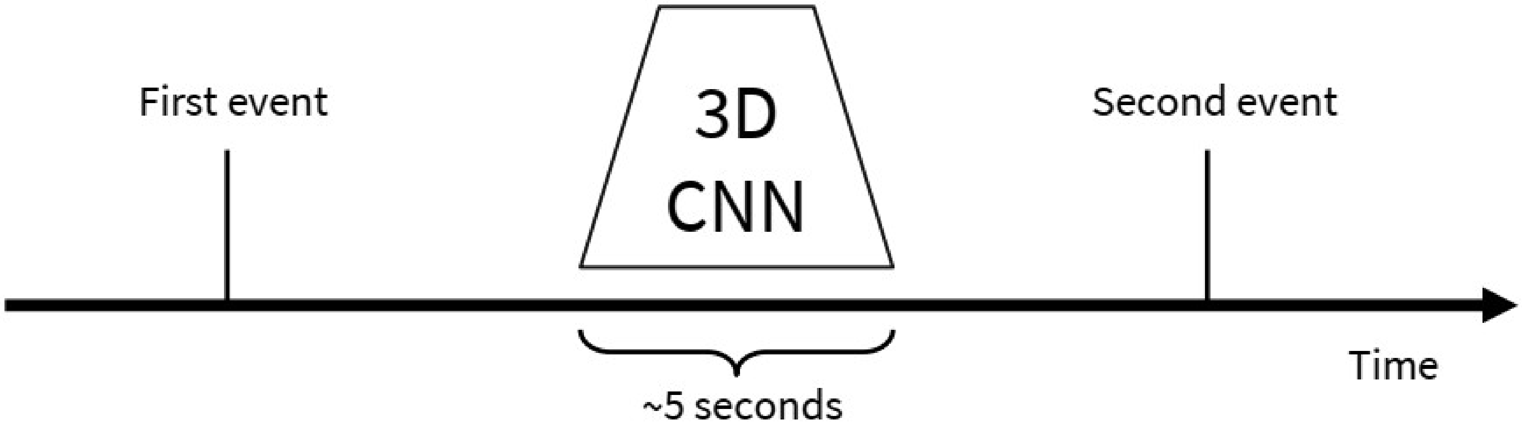
Early fusion: First 2D conv
processes all flow images

Movimiento y Apariencia

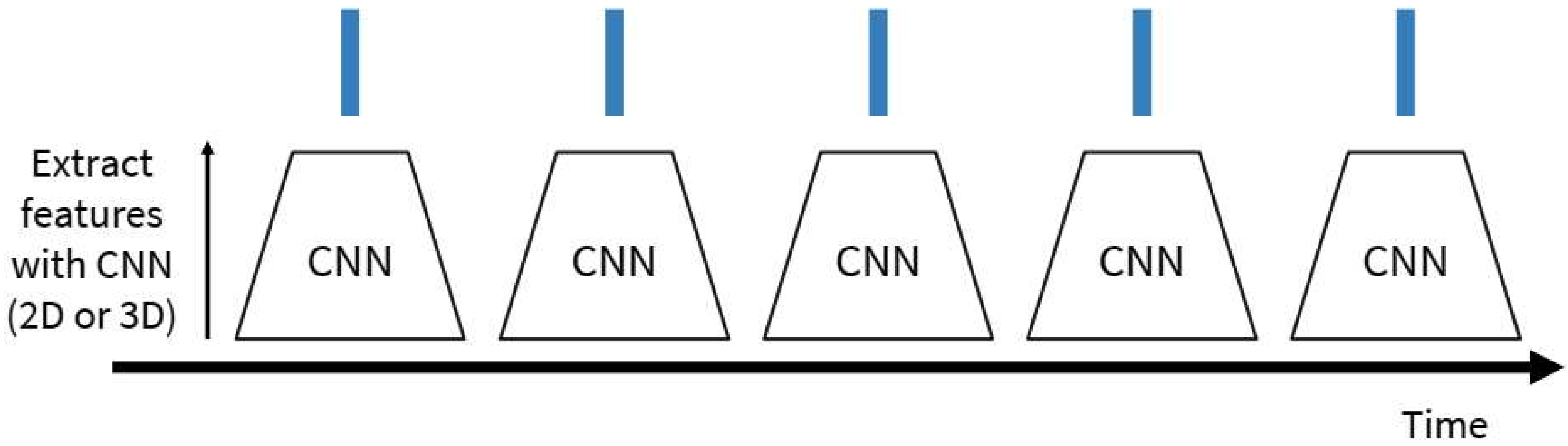


Modelando estructura temporal

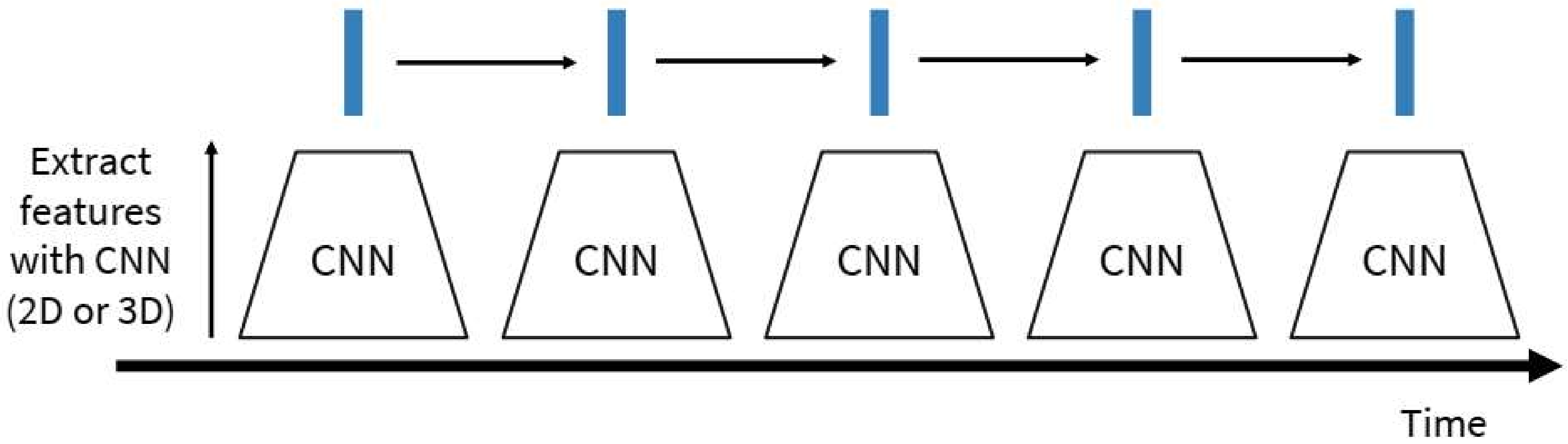
So far all our temporal CNNs only model local motion between frames in very short clips of ~2-5 seconds. What about long-term structure?



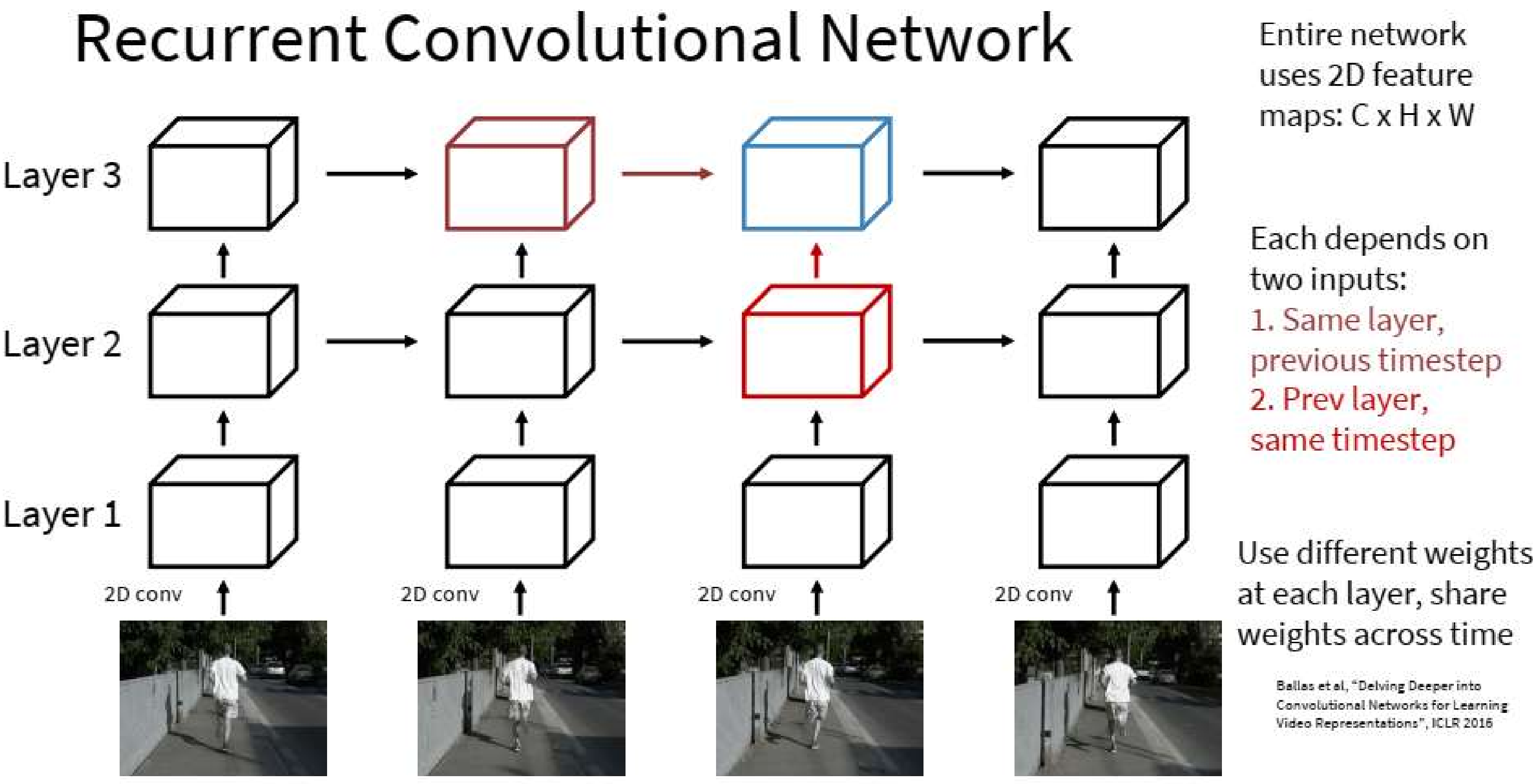
Modelando estructura temporal



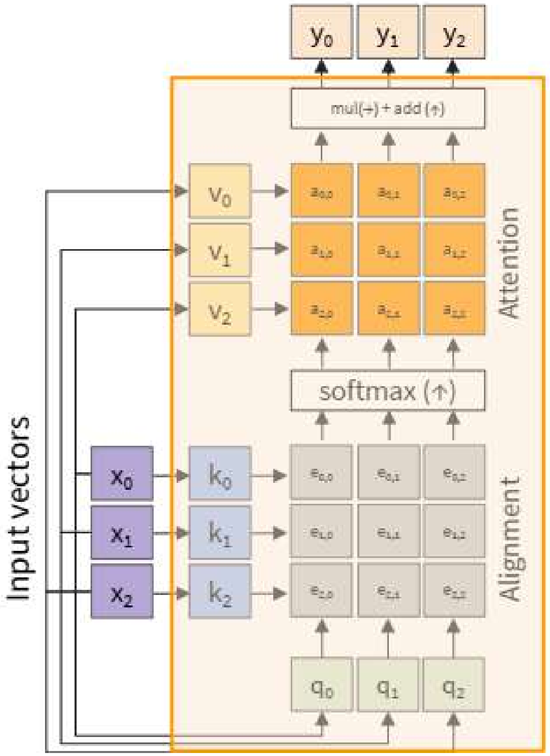
Modelando estructura temporal



Modelando estructura temporal



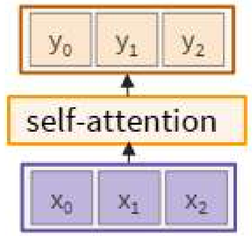
Self-attention



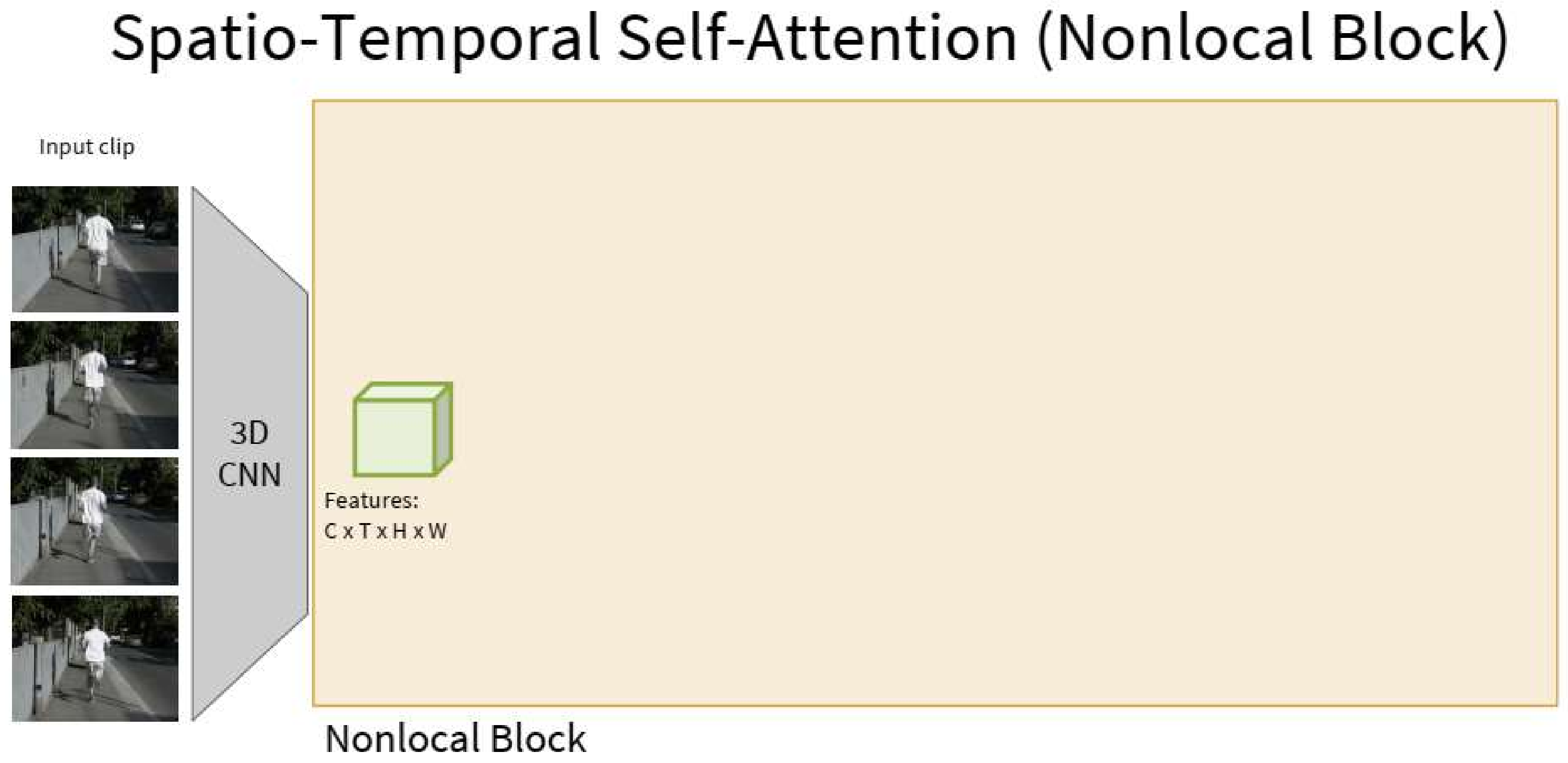
Outputs:
context vectors: y (shape: D_v)

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

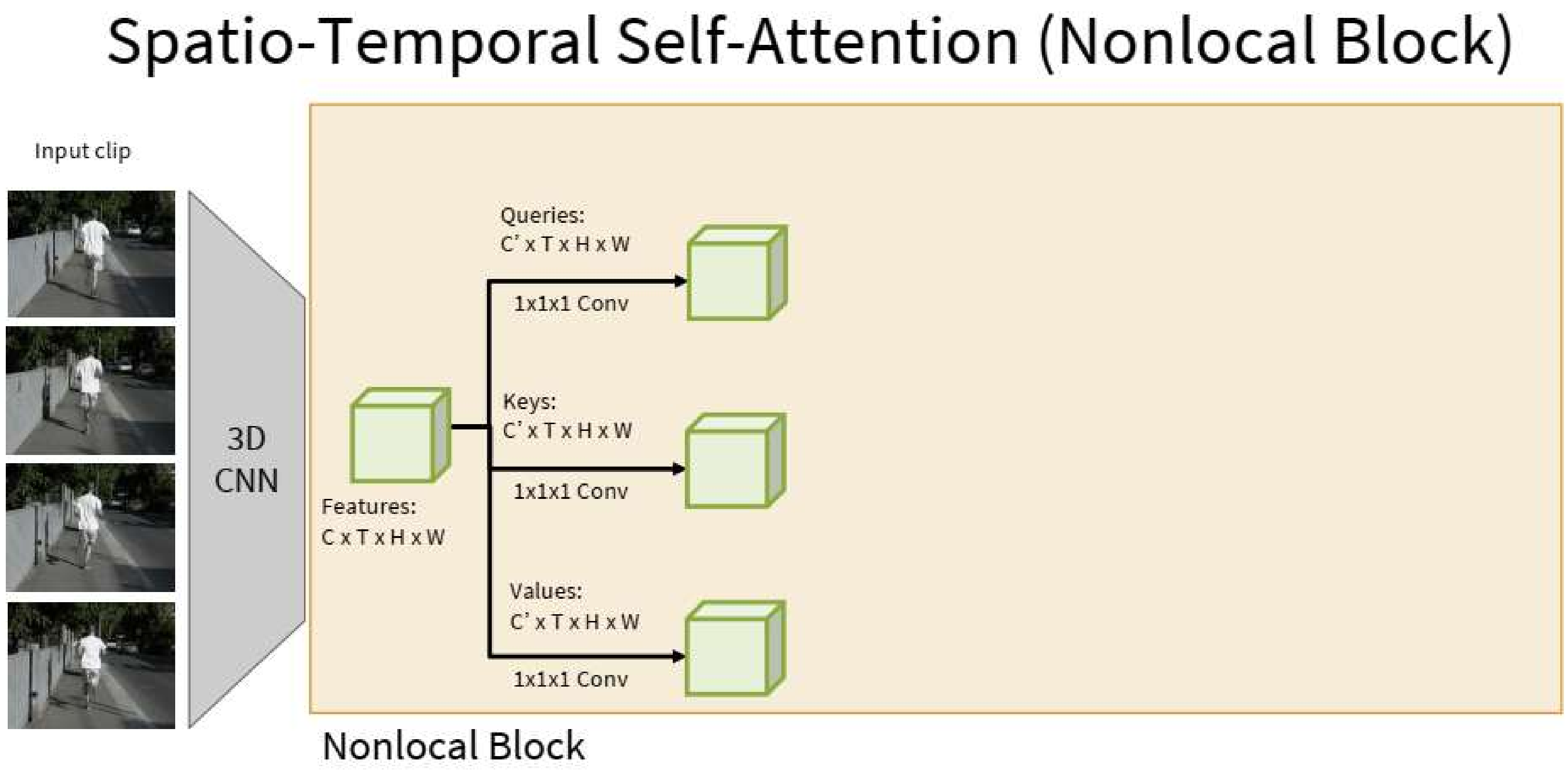
Inputs:
Input vectors: x (shape: N x D)



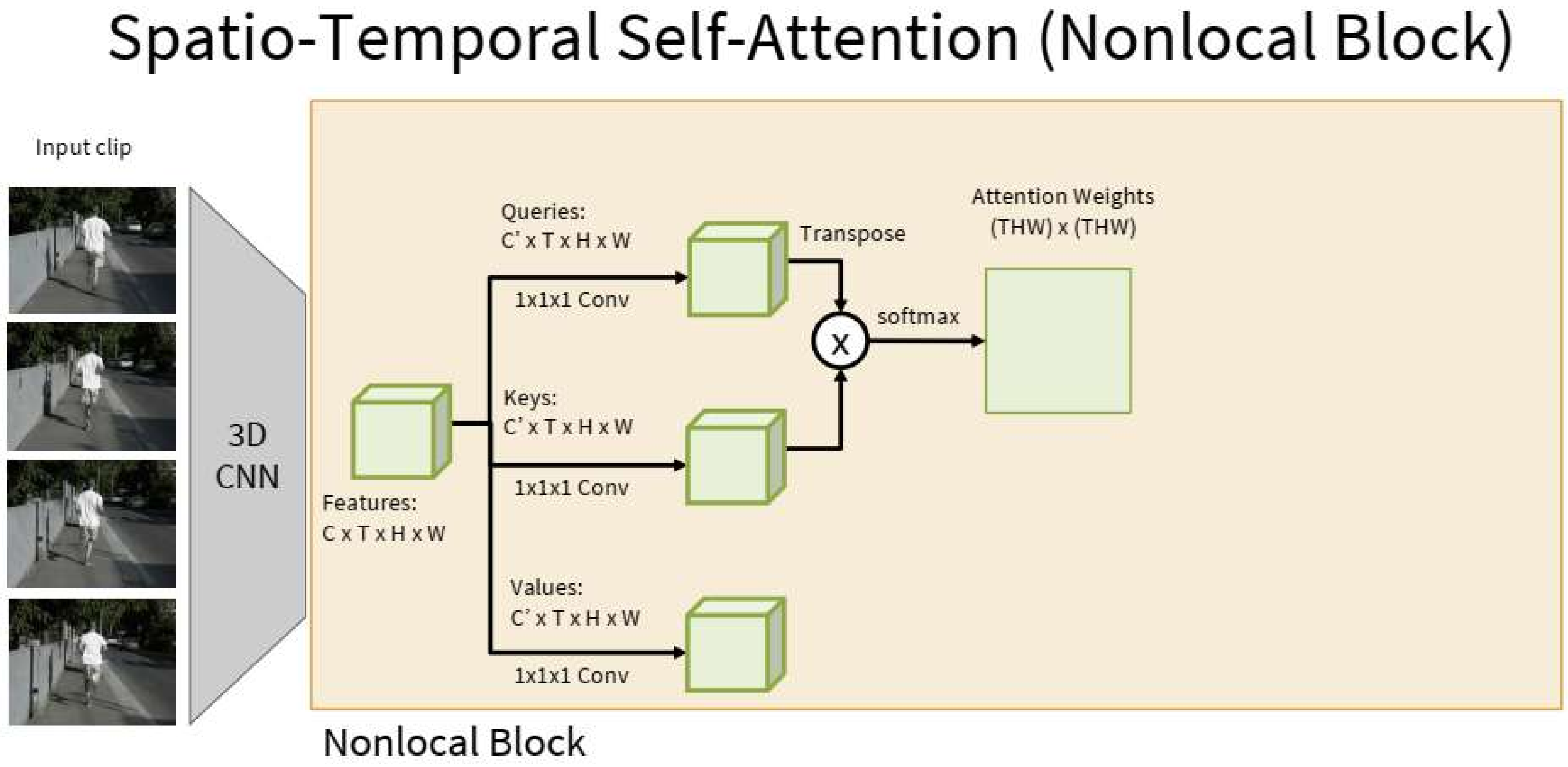
Self-attention



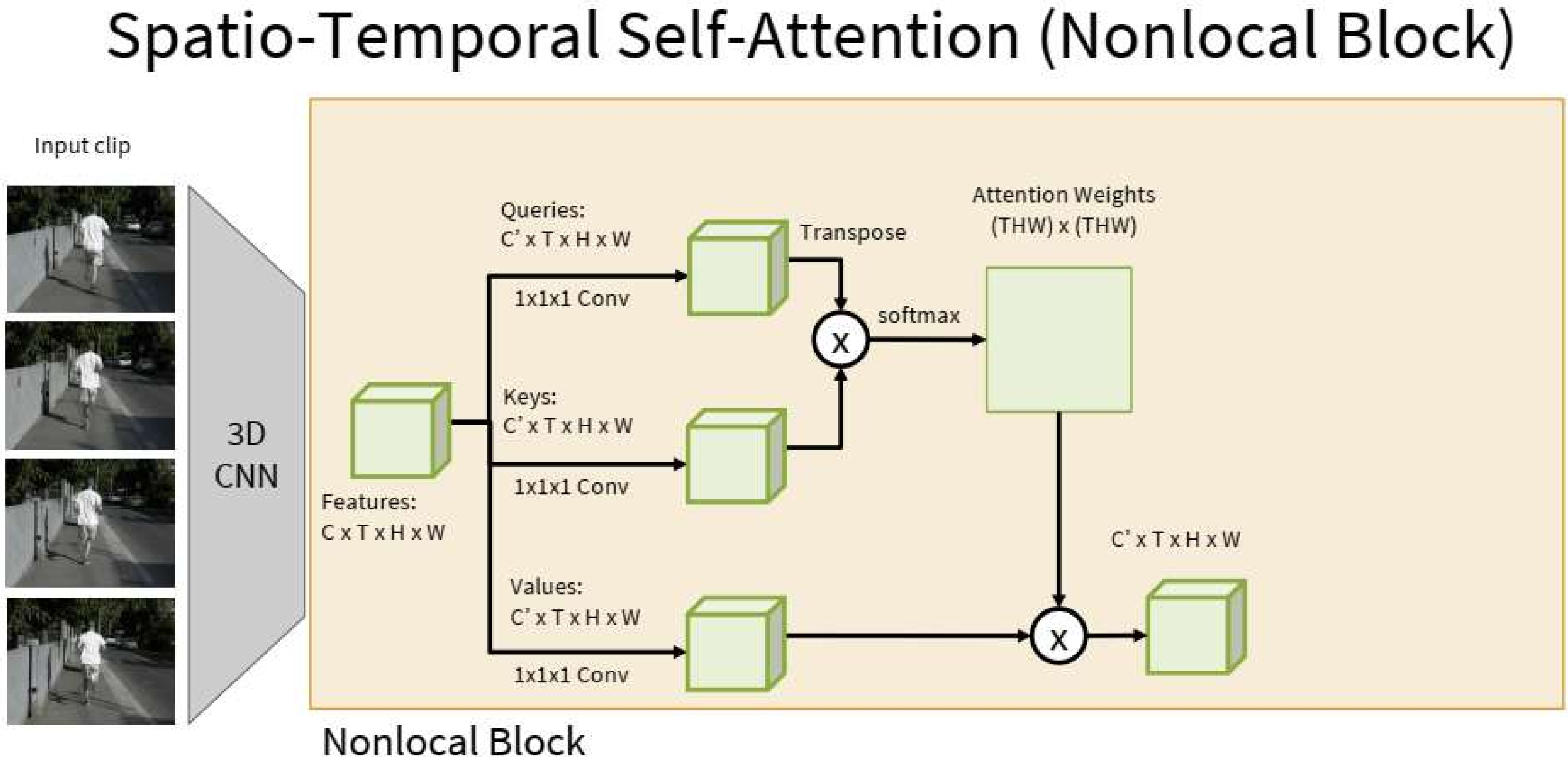
Self-attention



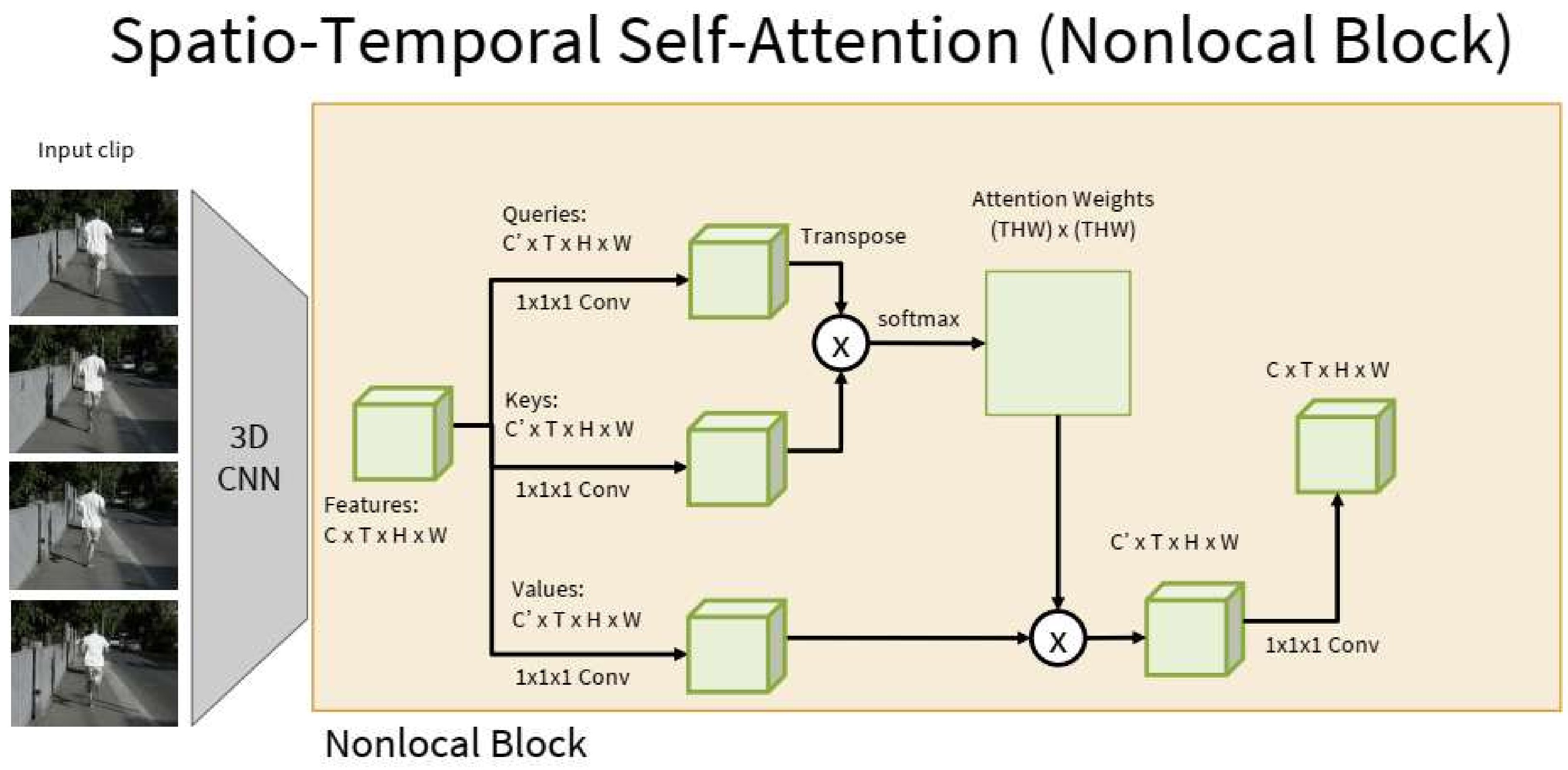
Self-attention



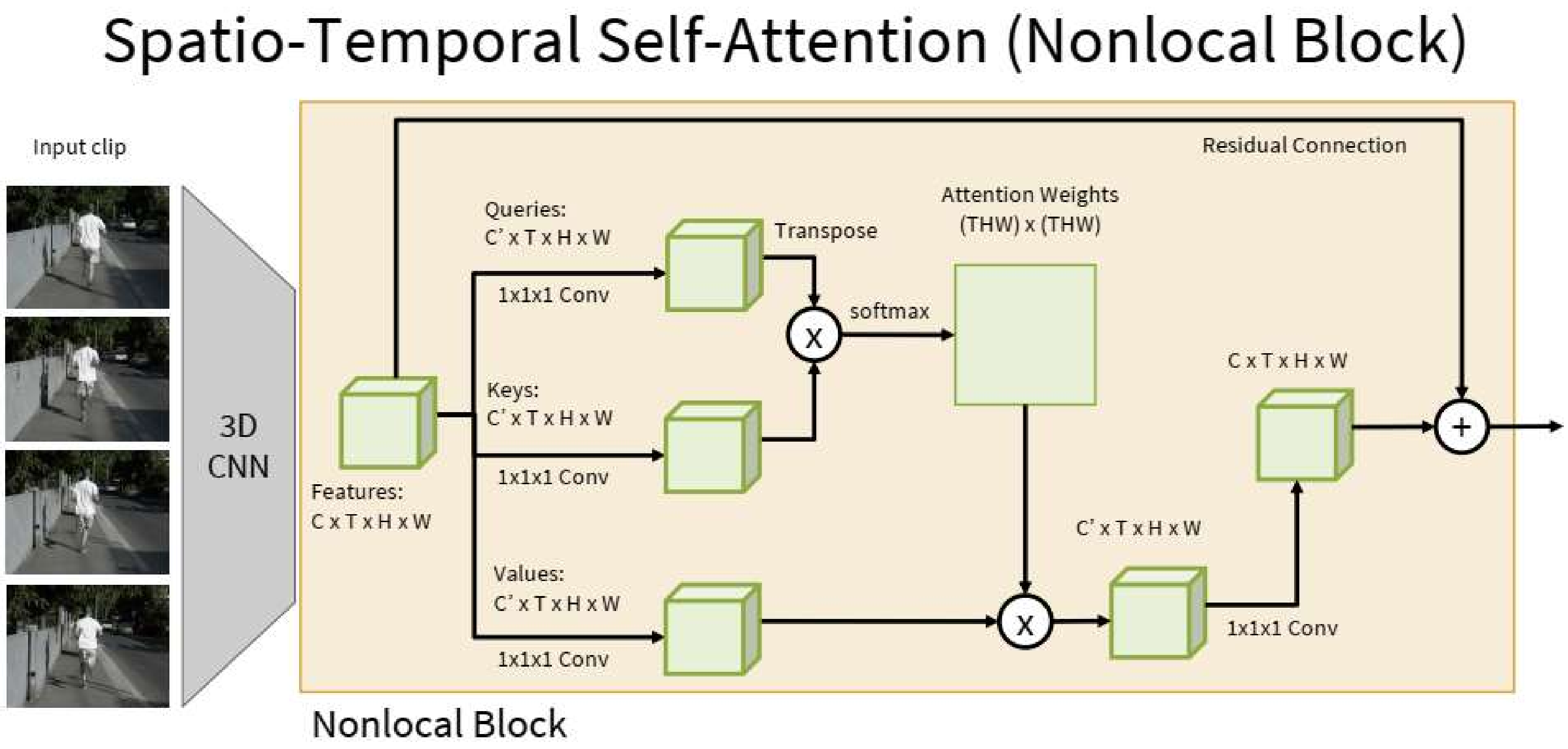
Self-attention



Self-attention



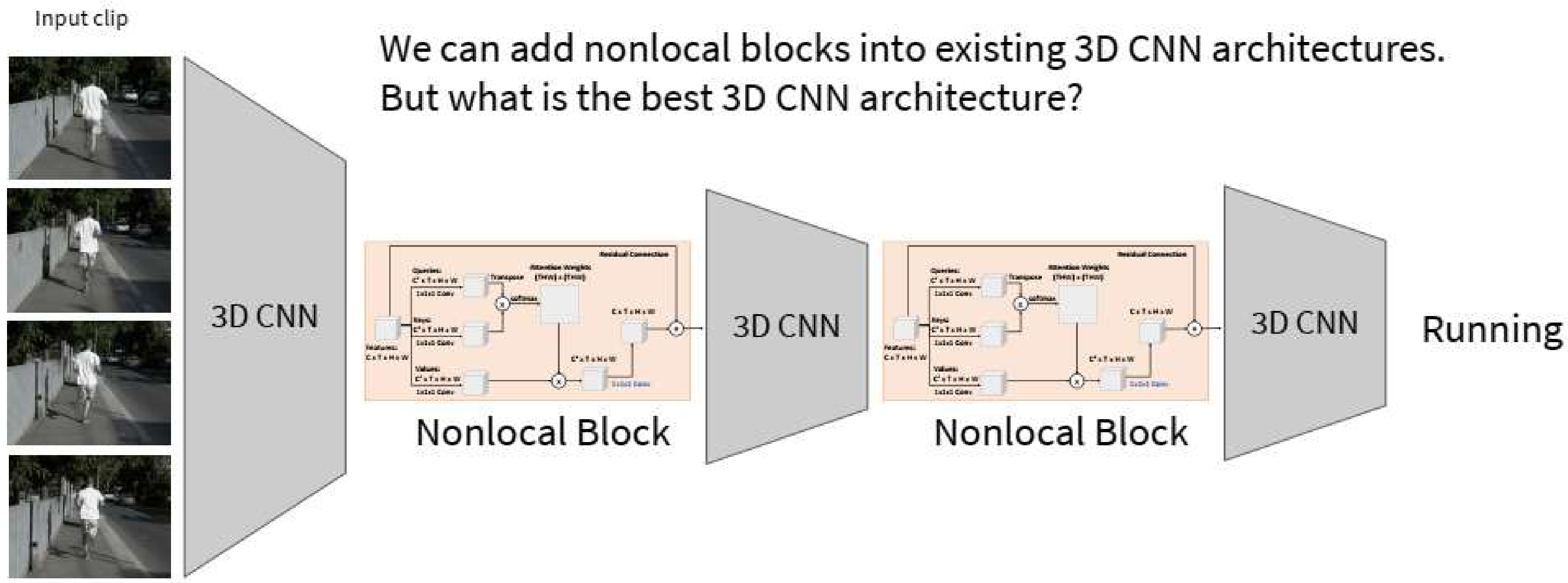
Self-attention



Self-attention

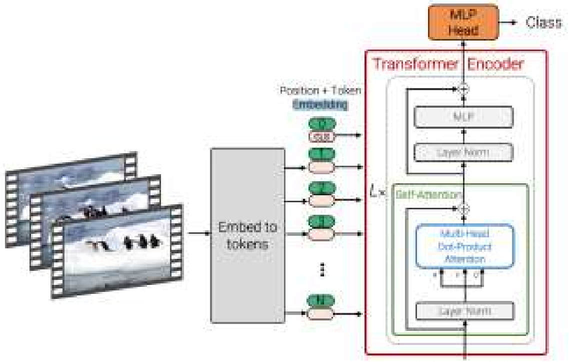
Spatio-Temporal Self-Attention (Nonlocal Block)

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



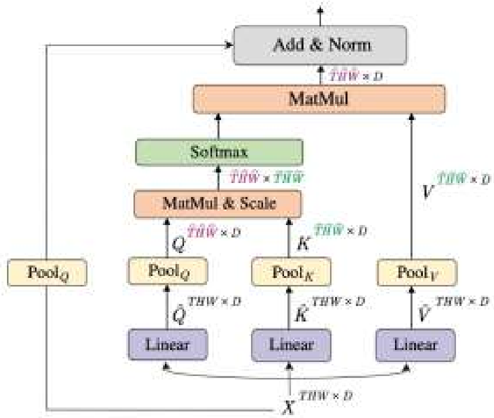
Video Transformer

Factorized attention:
Attend over space / time



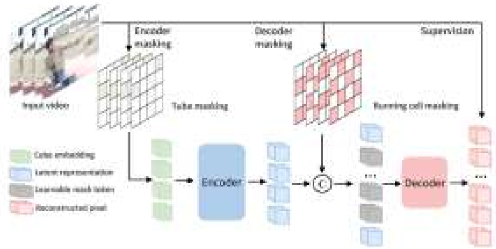
Bertasius et al, "Is Space-Time Attention All You Need for Video Understanding?", ICML 2021
Arnab et al, "ViViT: A Video Vision Transformer", ICCV 2021
Neimark et al, "Video Transformer Network", ICCV 2021

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 pretraining



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.

Video Transformer

Vision Transformers for Video