Visione Artificiale

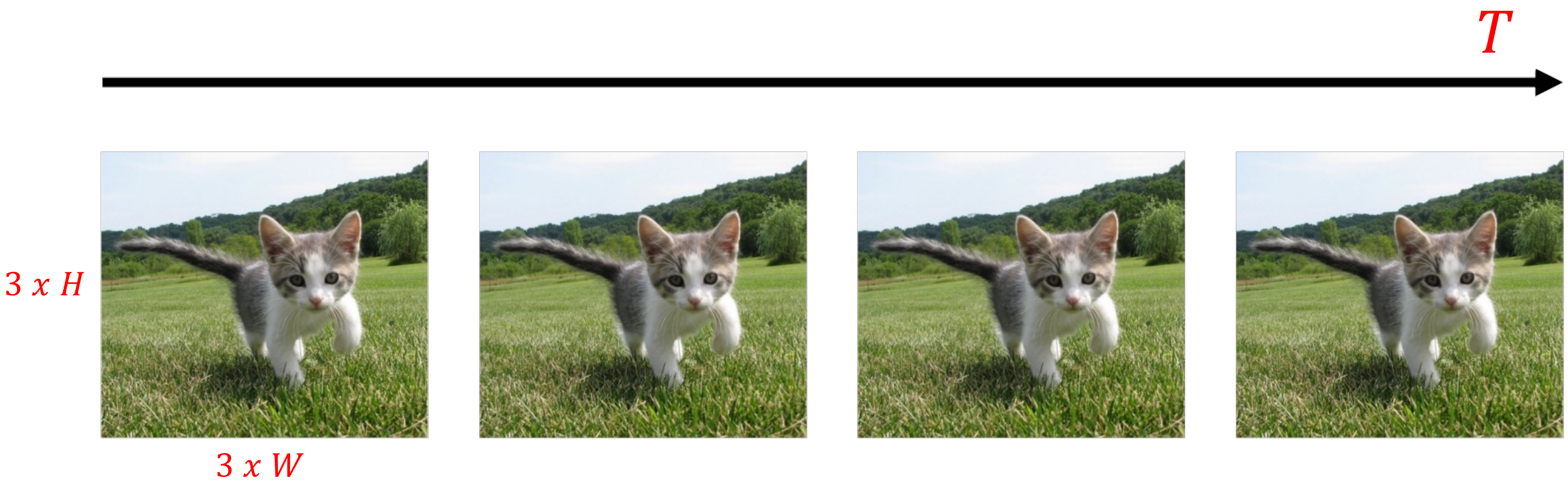
Raffaella Lanzarotti

Video

Video representation

- ✓ **Video = 2D + Time**
- ✓ A video is a **sequence** of images **4D tensor**:

$$T \ x \ 3 \ x \ H \ x \ W \quad \text{or} \quad 3 \ x \ T \ x \ H \ x \ W$$



Video Classification

Video sequence

Input dims: $T \ x \ 3 \ x \ H \ x \ W$

Swimming
Jumping
Eating
Standing
Running

Example task: Video Classification

Images: Recognize **objects**

Dog
Cat
Fish
Truck

Videos: Recognize **actions**

Swimming
Running
Jumping
Eating
Standing

Problem: Videos are big!

Input video:
T x 3 x H x W

- ✓ Videos are ~30 frames per second (fps)
- ✓ Size of uncompressed video
- ✓ (3 bytes per pixel):
  - SD (640 x 480): **~1.5 GB per minute**
  - HD (1920 x 1080): ~**10 GB per minute**

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

Training on Clips

- ✓ **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)
- ✓ **Raw video**: Long, high FPS

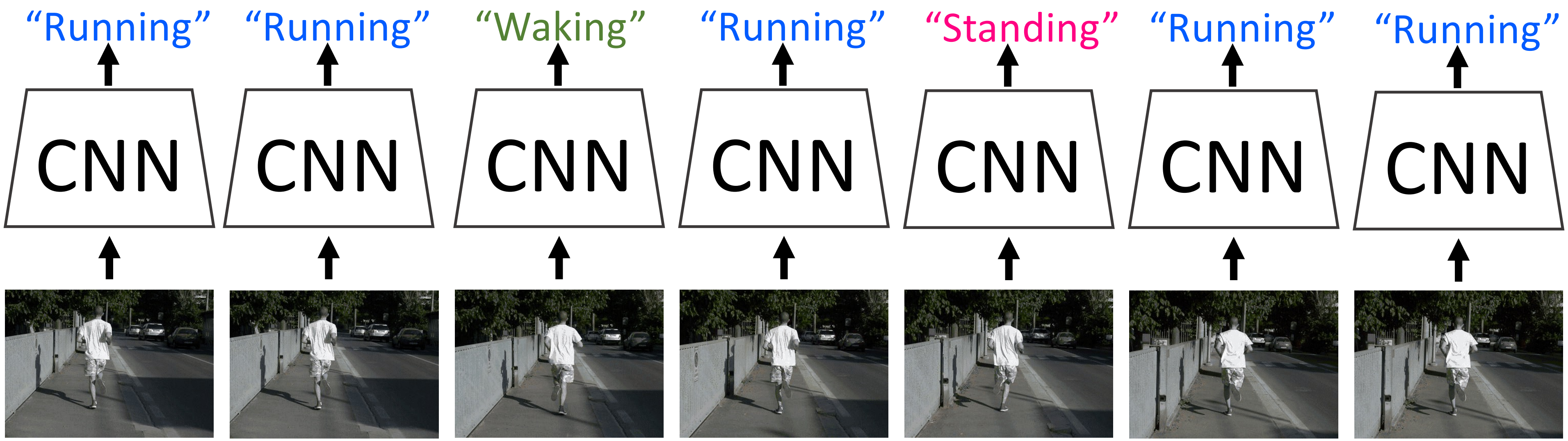
- ✓ **Training**: Train model to classify short clips with low FPS

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

Video classification

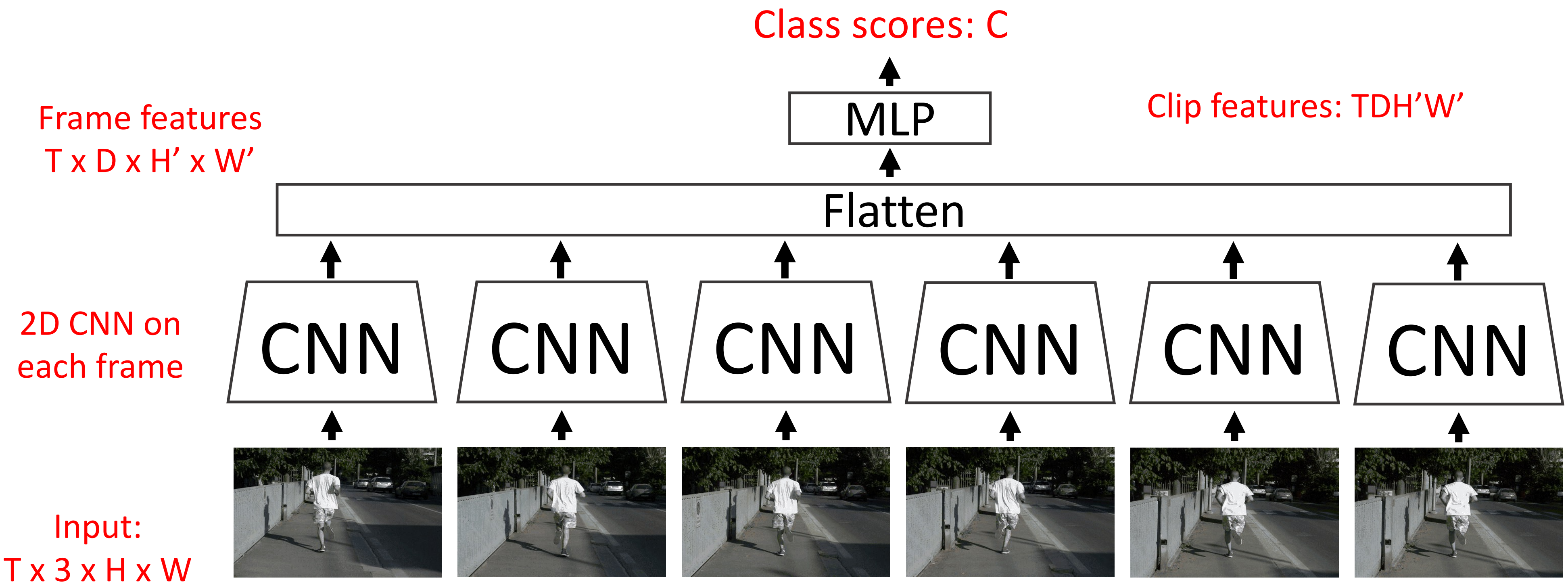
Video Classification: Single-Frame CNN

- ✓ **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)

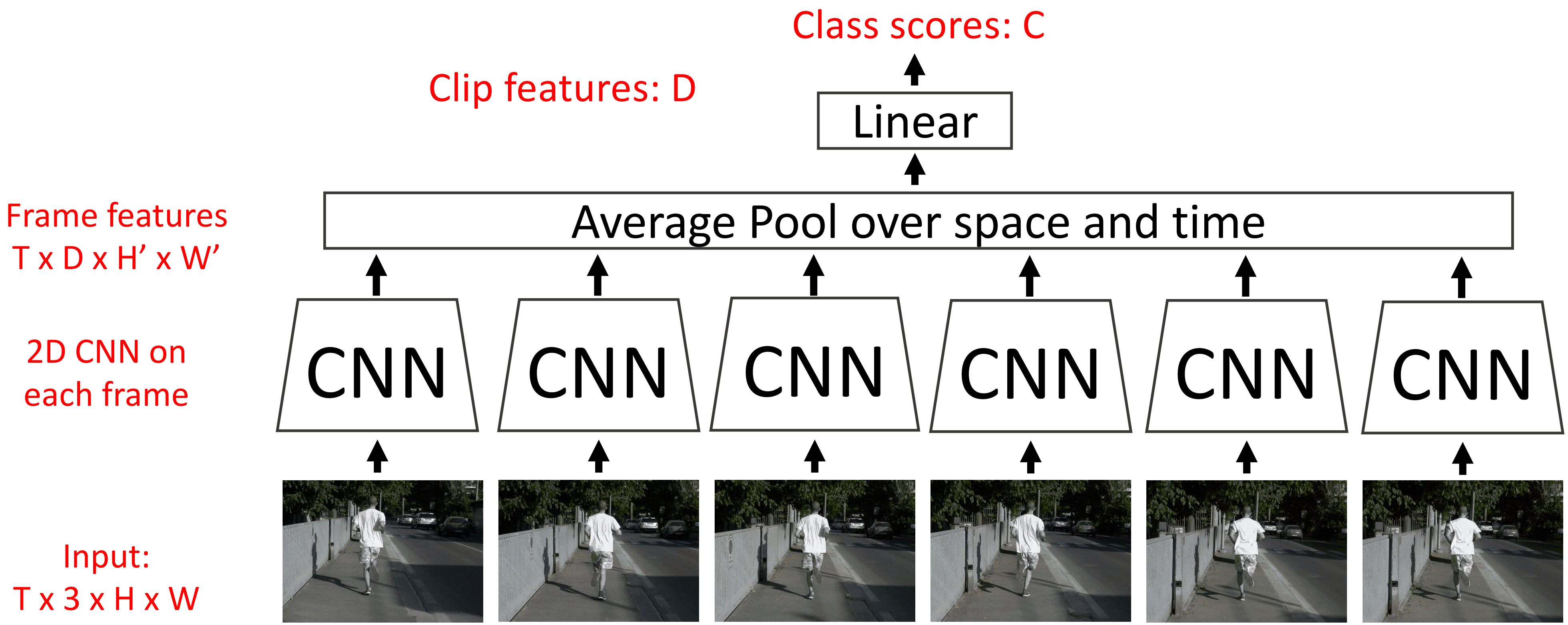
- ✓ **Intuition**: Get high-level appearance of each frame, and combine them
- ✓ Run **2D CNN** on **each frame**, **concatenate features** and feed to **MLP**



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

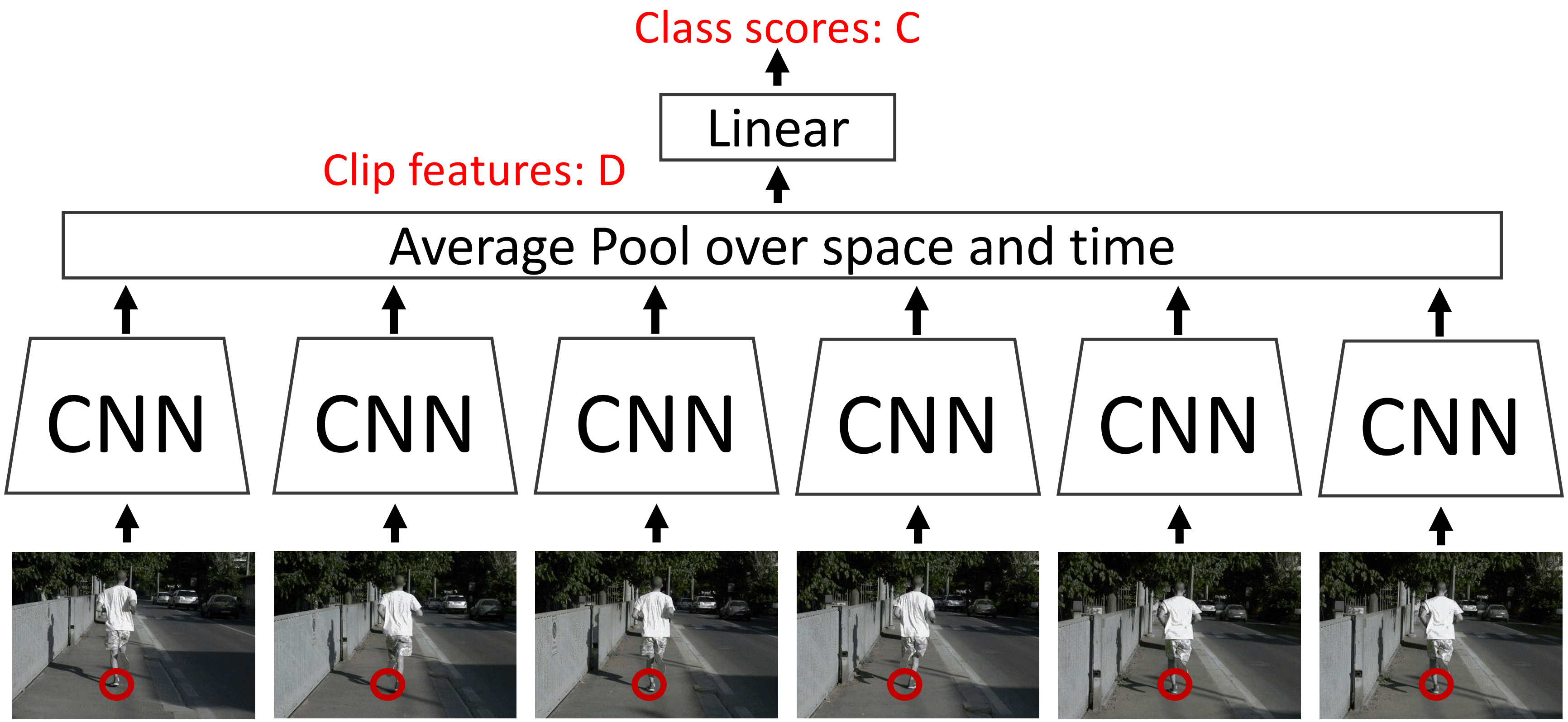
Video Classification: Late Fusion (with pooling)

- ✓ **Intuition**: Get high-level appearance of each frame, and combine them
- ✓ Run **2D CNN** on **each frame, concatenate features** and feed to **Linear**



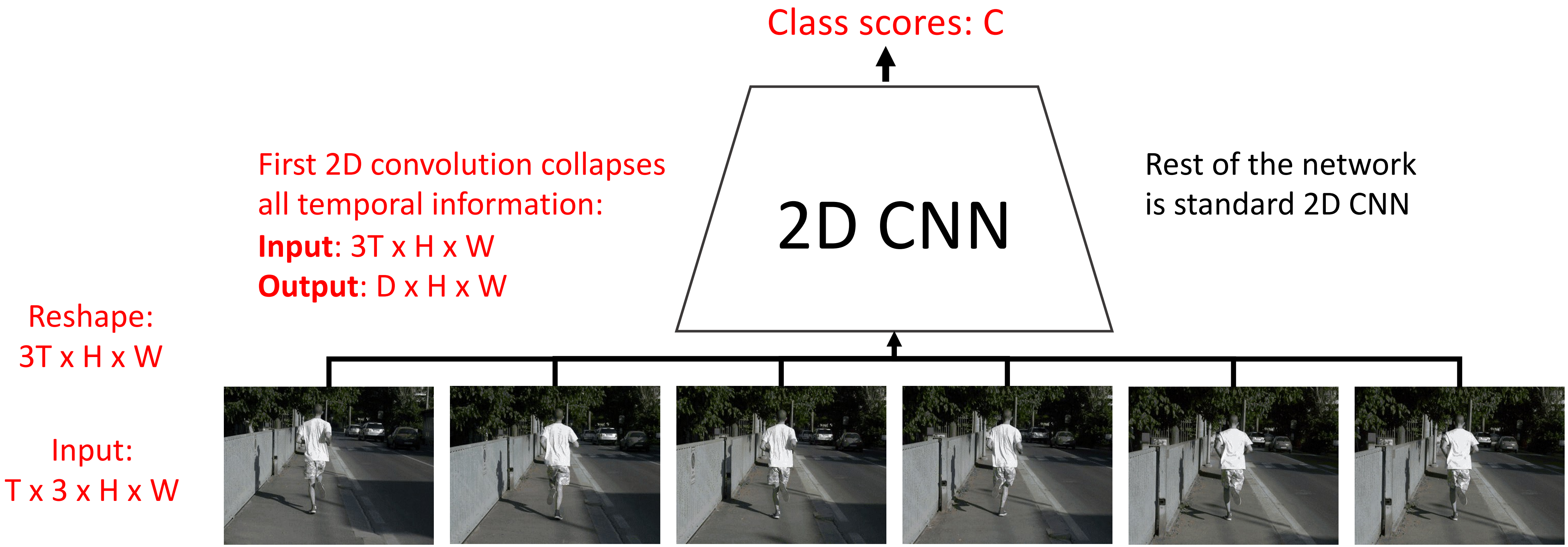
Video Classification: Late Fusion (with pooling)

Problem: Hard to compare low-level motion between frames!



Video Classification: Early Fusion

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



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

Video Classification: Early Fusion

Problem: One layer of temporal processing may not be enough

Class scores: C

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

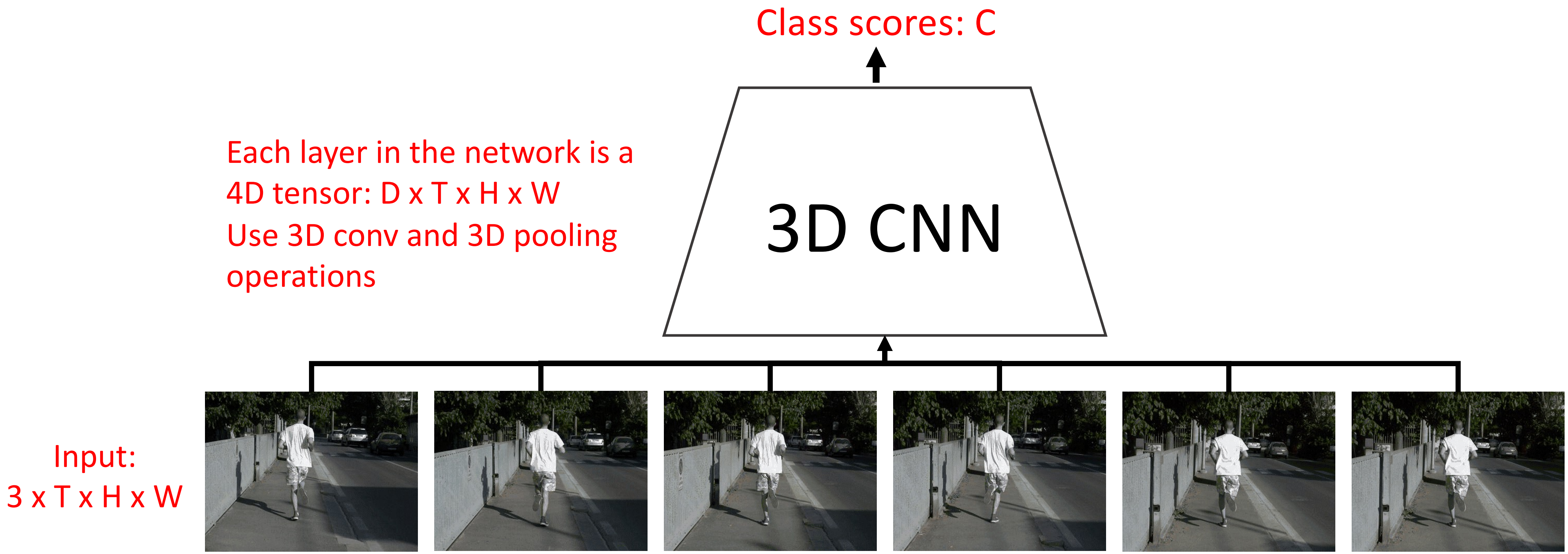
2D CNN

Rest of the network is standard 2D CNN

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



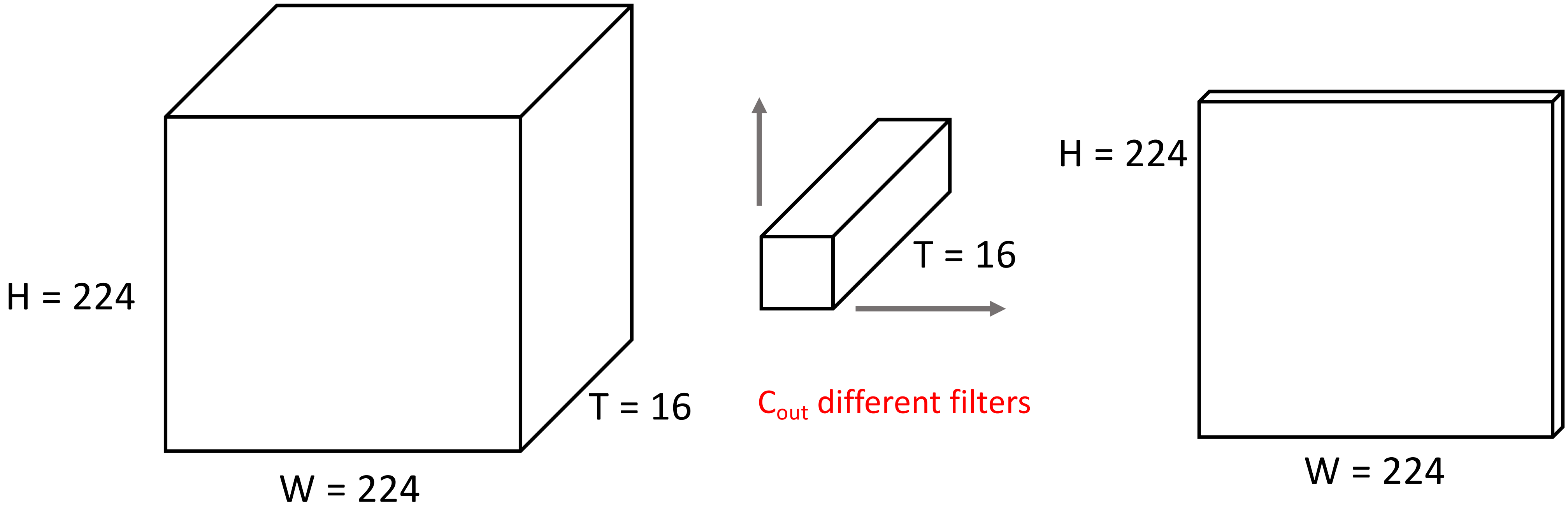
Ji et al, “3D Convolutional Neural Networks for Human Action Recognition”, TPAMI 2010 ; Karpathy et al, “Large-scale Video Classification with Convolutional Neural Networks”, CVPR 2014

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

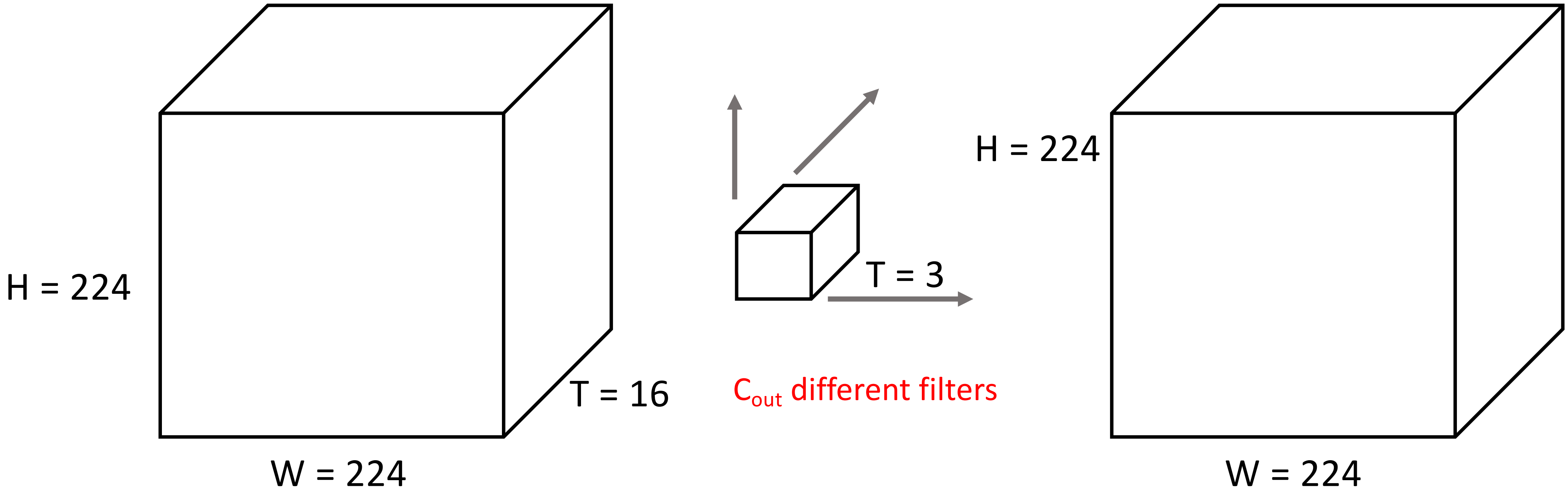


2D Conv (Early Fusion) vs 3D Conv (3D CNN)

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

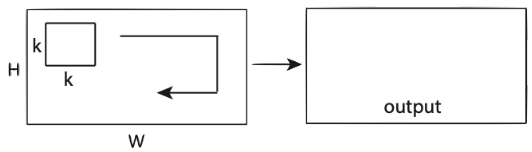
Weight:
C_{out} x C_{in} x 3 x 3 x 3
Slide over x and y

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

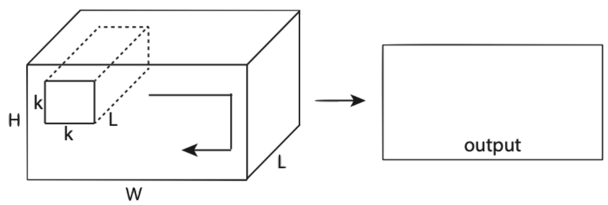


Rationale

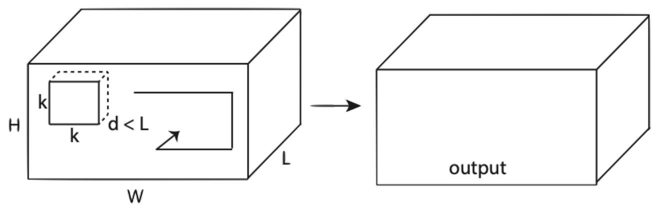
- ✓ In **3D ConvNets**, **convolution** and **pooling** operations are performed **spatio-temporally** while in **2D ConvNets** they are done only **spatially**
- ✓ **2D convolution** applied on an image will output an **image**, 2D convolution applied on **multiple images** (treating them as different channels) also results in an image
- ✓ **2D ConvNets loose temporal** information of the input signal right after every convolution operation
- ✓ Only **3D convolution preserves** the **temporal information** of the input signals resulting in an **output volume**



2D convolution



2D convolution on multiple frames

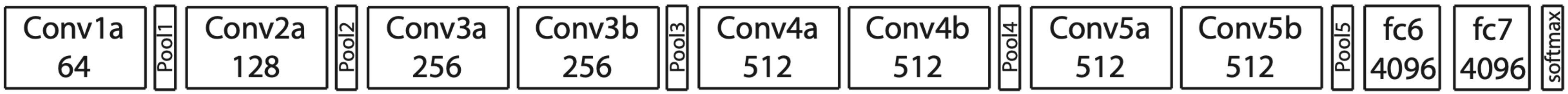


3D convolution

C3D architecture

✓ **C3D architecture**

1. C3D net **has 8 convolution**, **5 max-pooling**, and **2 fully connected layers**, followed by a **softmax** output layer.
2. All 3D convolution **kernels are 3 × 3 × 3** with stride 1 in both spatial and temporal dimensions. Number of filters are denoted in each box.
3. The 3D pooling layers are denoted from **pool1** to **pool5**
4. All **pooling kernels are 2 × 2 × 2**, except for pool1 is 1 × 2 × 2
5. Each fully connected layer has 4096 output units.



Tran, Du, et al. "Learning Spatiotemporal Features with 3D Convolutional Networks." *2015 IEEE International Conference on Computer Vision (ICCV)*, 2015

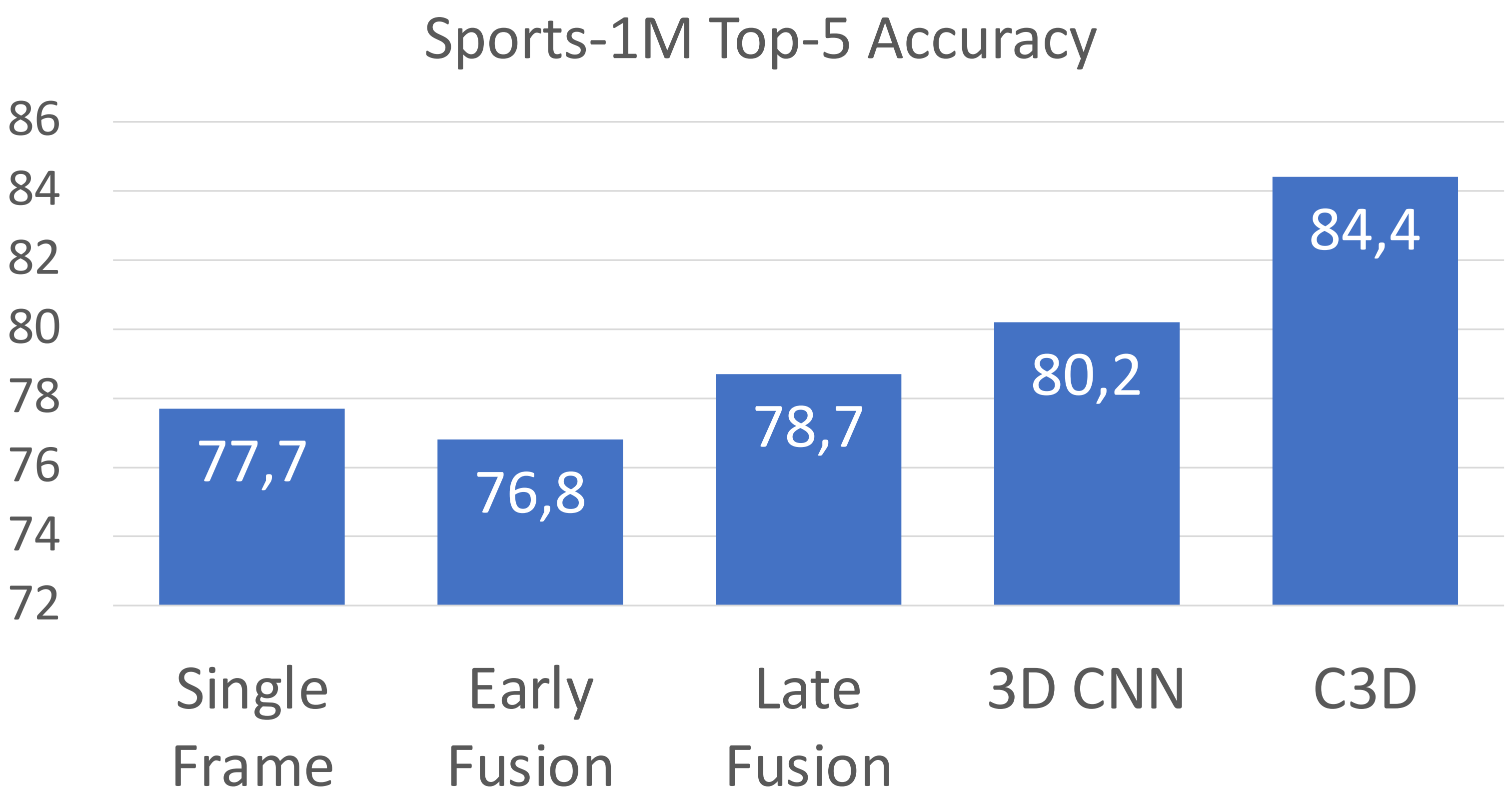
Example Video Dataset: Sports-1M

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

Ground Truth
Correct prediction
Incorrect prediction

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

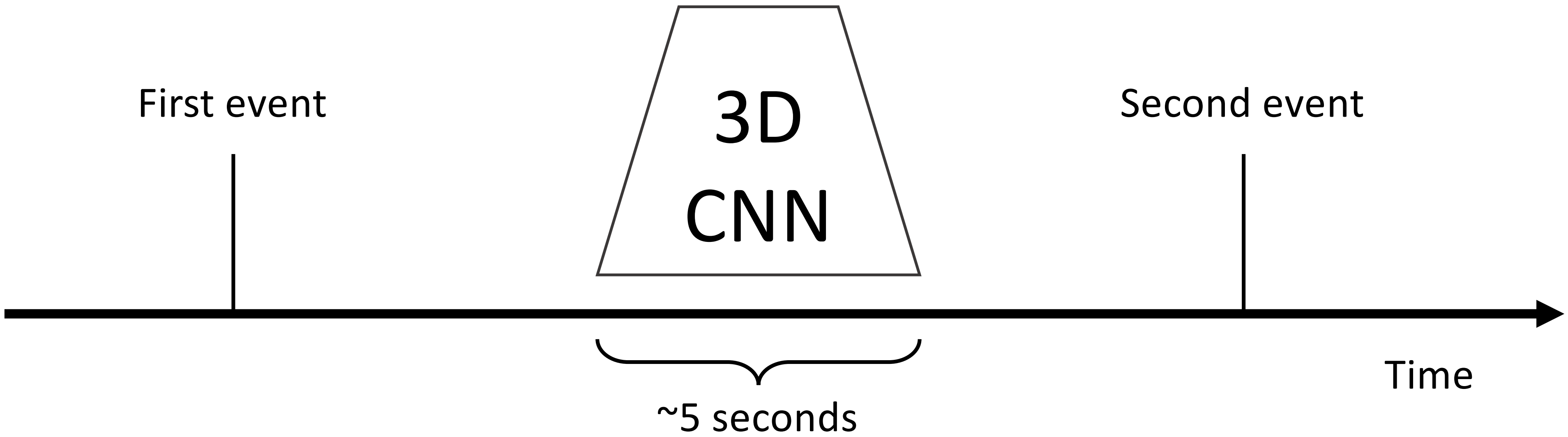
Early Fusion vs Late Fusion vs 3D CNN



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

Modeling long-term temporal structure

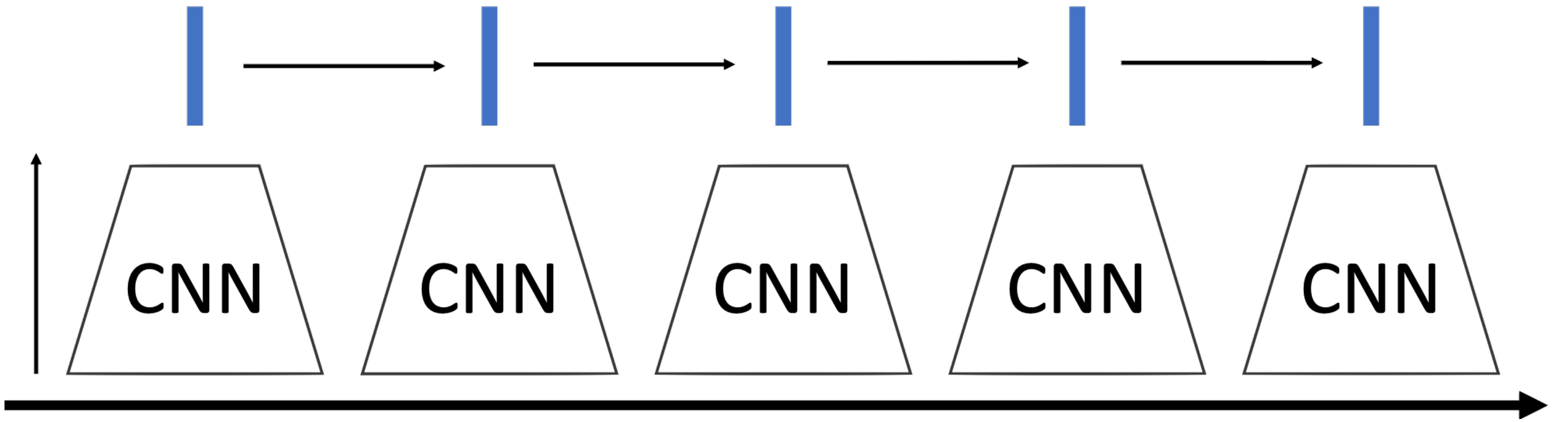
- ✓ **Temporal CNNs** only model local motion between frames in **very short clips** of ~2-5 seconds
- ✓ What about **long-term structure**?



Modeling long-term temporal structure

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

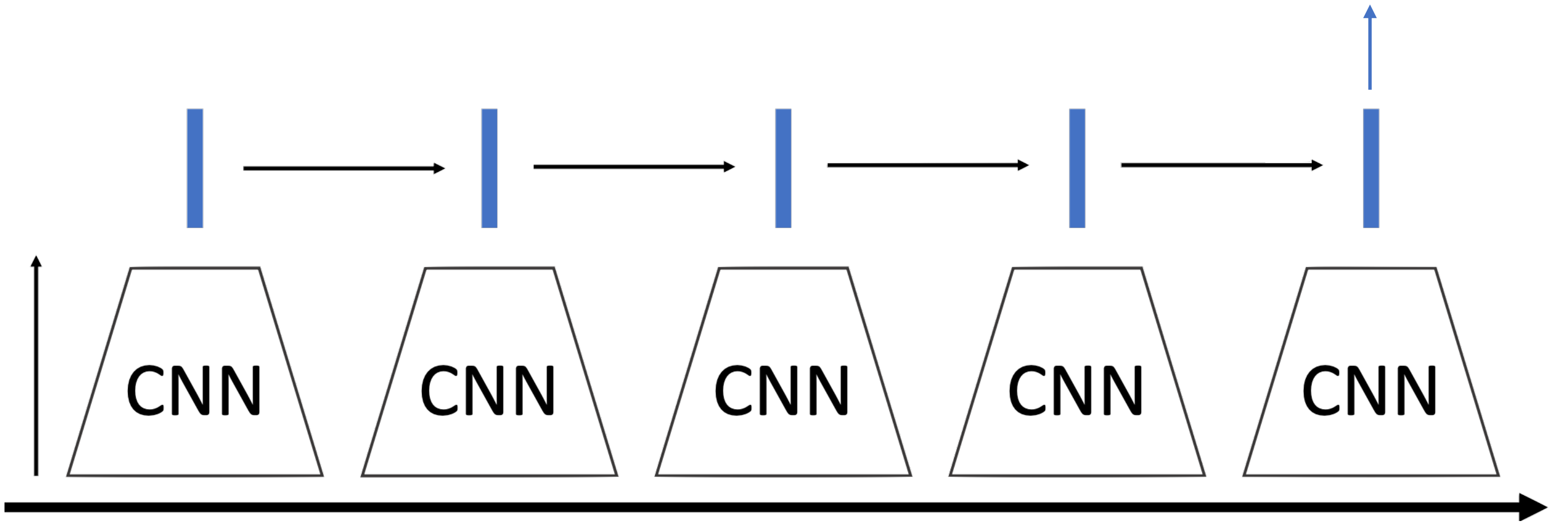
Extract features with CNN (2D or 3D)



Modeling long-term temporal structure

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

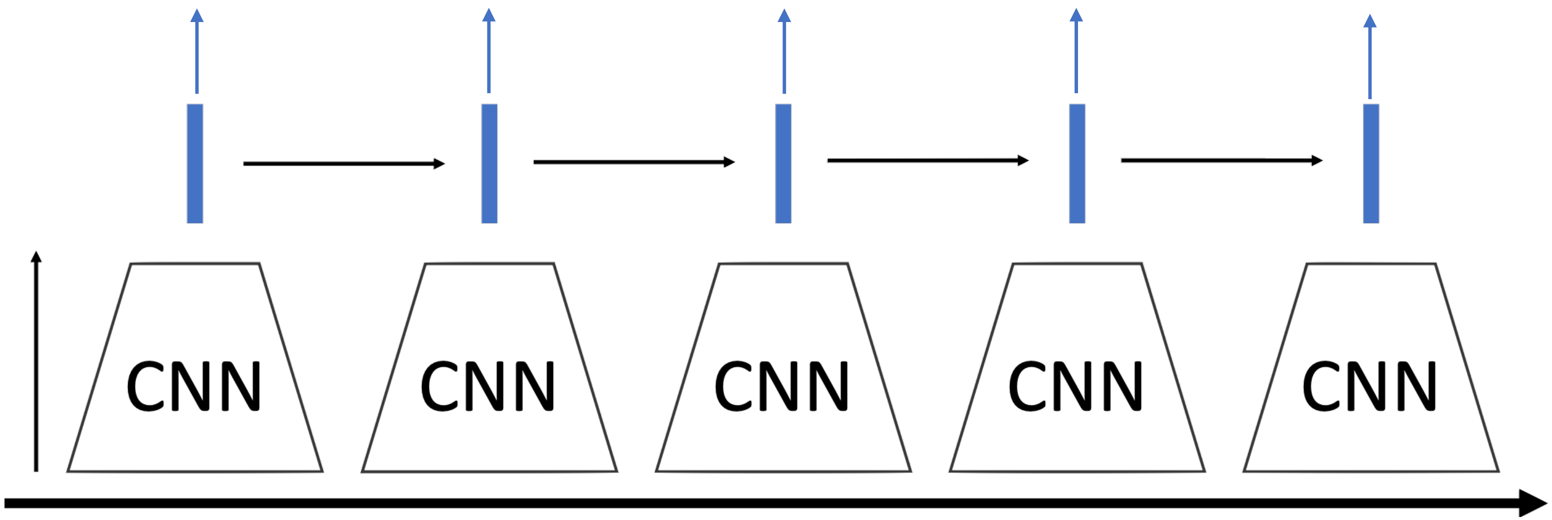
Extract features with CNN (2D or 3D)



Modeling long-term temporal structure

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

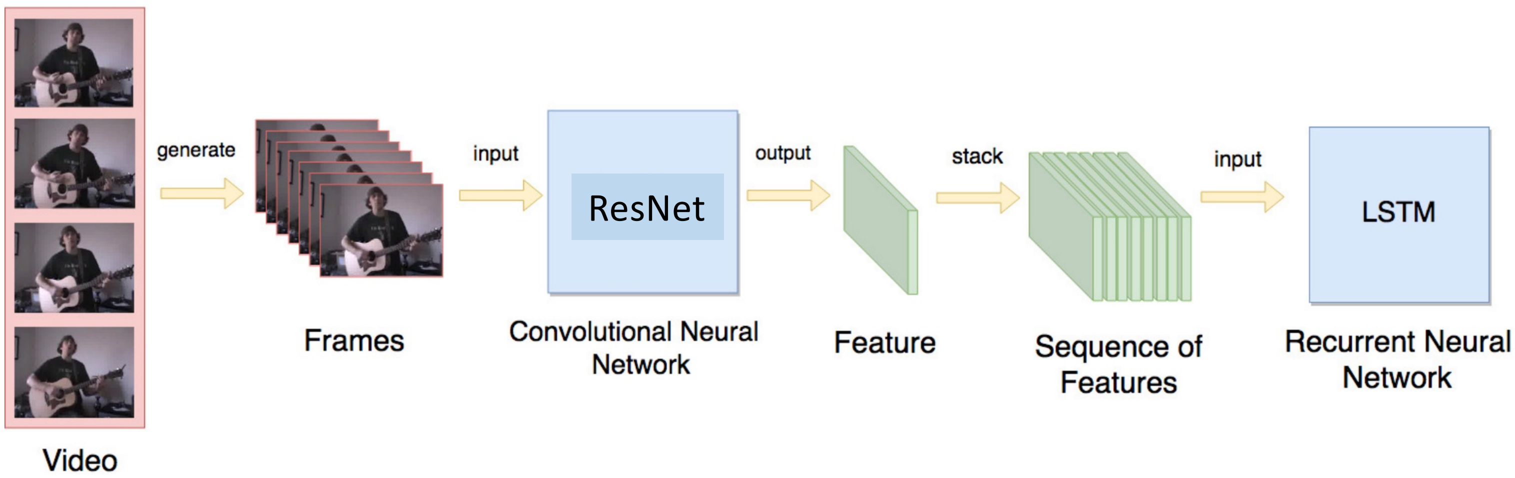
Extract
features
with CNN
(2D or 3D)



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

Design choice

- ✓ The **final layers** of a **CNN** contain important **properties** related to a **frame**
- ✓ Represented as an **array** of values **given** by a **prediction** for each RGB **frame** for the video clips
- ✓ **LSTM layers** are composed by units which consider what previously happens and a **memory state** to generate a prediction
- ✓ The **output** of the **LSTM** layer is passed to **Dense** layers followed by a final **softmax** layer



Demo

(Inspired by: https://github.com/eriklindernoren/Action-Recognition)

- ✓ **Download** the code folder `Action-Recognition-master` from my github (*.py original files, *.ipynb rearranged notebooks)
- ✓ **Download** the dataset using my `download_ucf101.sh`:

```
$ cd data/
$ bash download_ucf101.sh
$ unrar x UCF101.rar
$ unzip ucfTrainTestlist.zip
```

- ✓ Before extracting frames, **remove** the corrupted videos:
  - v_Surfing_g07_c02.avi
  - v_Surfing_g02_c03.avi
- ✓ **Extract** frames:

```
$ python3 extract_frames.py
```

Demo

- ✓ Download the model ConvLSTN_90.pth from: repository
- ✓ Save it in the folder `model_checkpoints`
- ✓ Now you can **run** the `test` or the `test_on_video`

- ✓ You can also **learn** new models running the notebook `train`

- ✓ **Inspect** the notebook `model`

Attention mechanism & Transformers

On the blackboard …

Vision Transformers (ViTs)