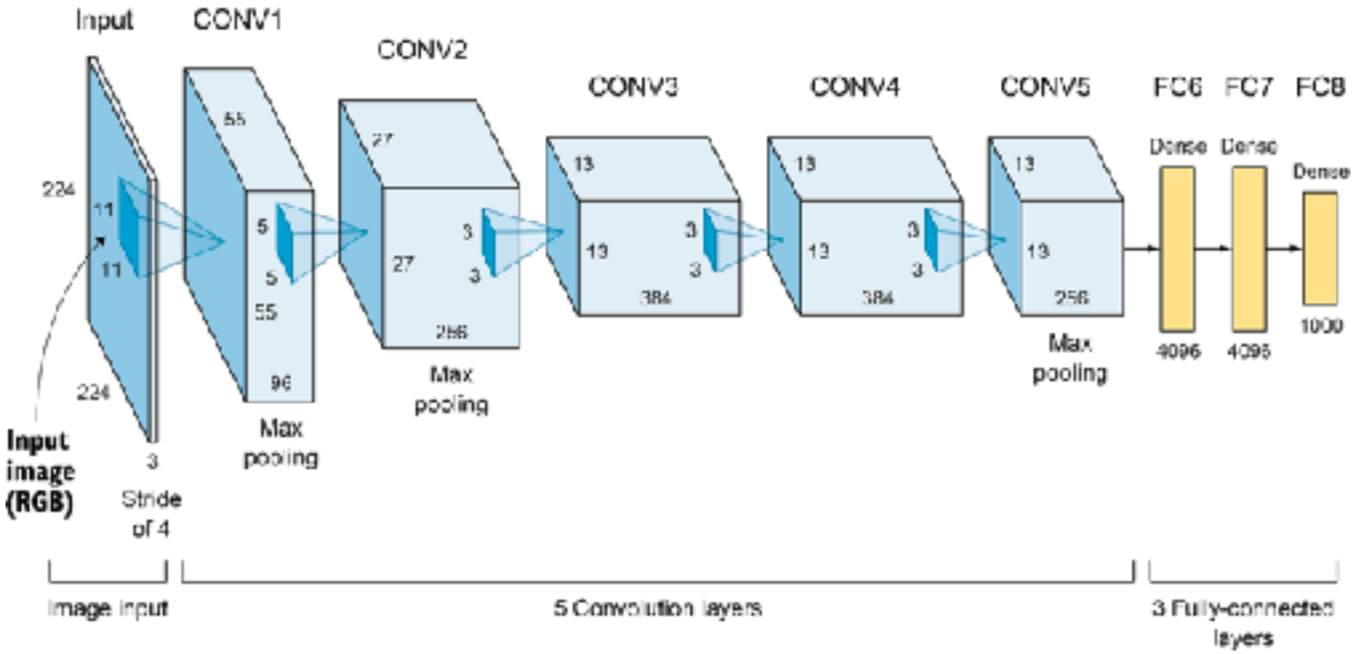
Learning Spatiotemporal Features with 3D Convolutional Networks

ICCV 2015

Du Tran, Lubomir Bourdev, Rob Fergus,
Lorenzo Torresani, Manohar Paluri

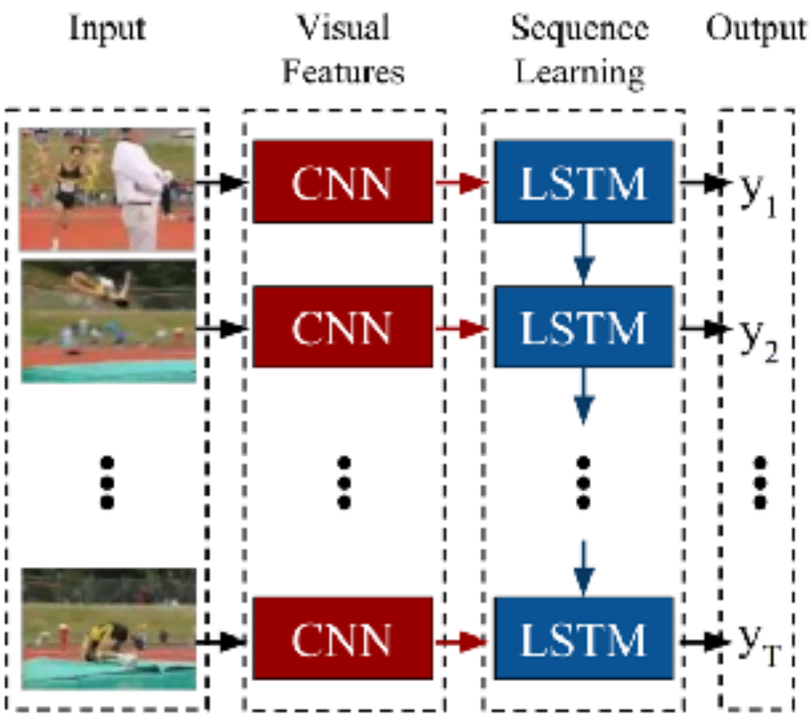
Motivation

Traditional 2D CNNs are not designed to learn spatiotemporal features from video inputs.



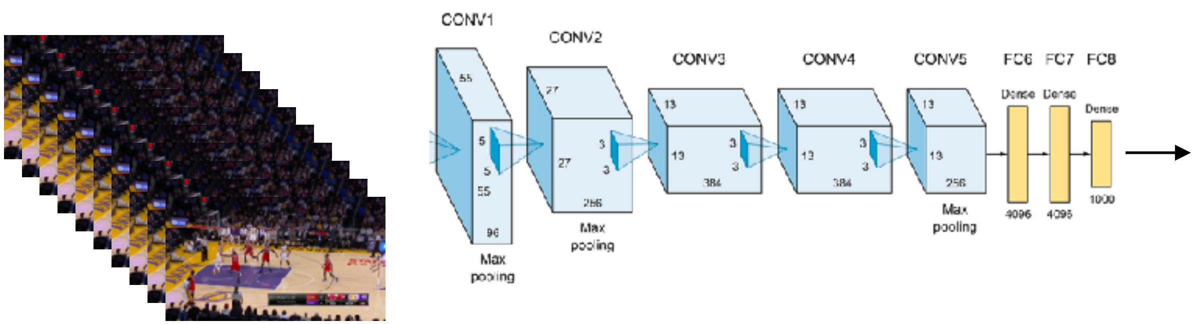
Motivation

CNN + LSTM approaches are not very effective when applied on large-scale datasets.



3D Convolutional Networks

- Deep 3-dimensional convolutional networks (3D CNNs) for spatiotemporal feature learning from video inputs.

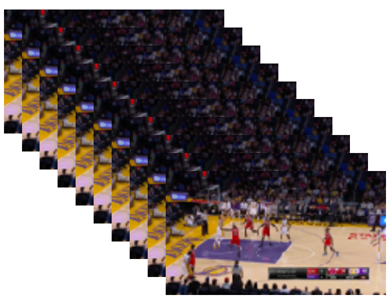


Playing Basketball: 0.89

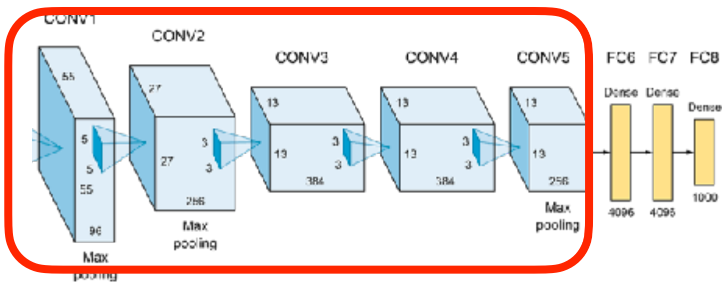
video frames 1, 2, … , L

3D Convolutional Networks

- Deep 3-dimensional convolutional networks (3D CNNs) for spatiotemporal feature learning from video inputs.



video frames 1, 2, … , L

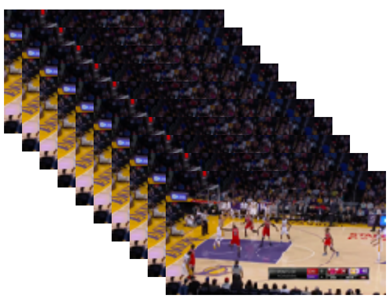


Playing Basketball: 0.89

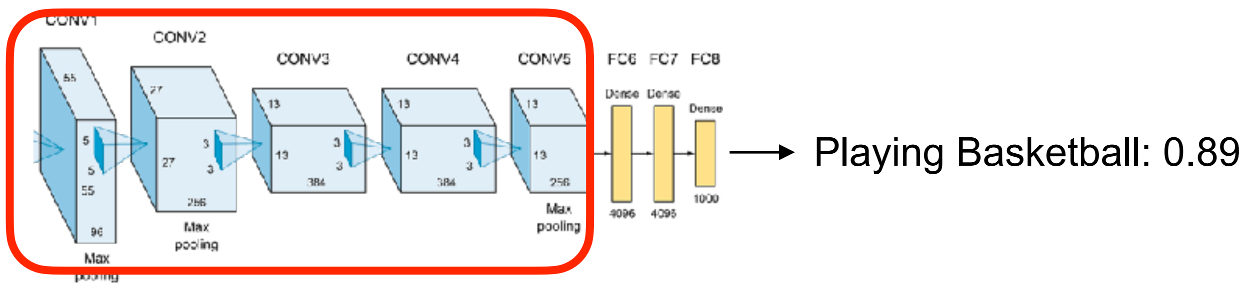
Instead of using 2D convolutions, we will use 3D convolutions in every convolutional layer inside the network.

3D Convolutional Networks

- Deep 3-dimensional convolutional networks (3D CNNs) for spatiotemporal feature learning from video inputs.



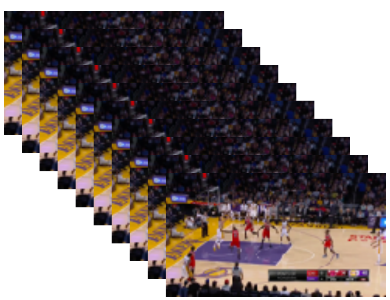
video frames 1, 2, … , L



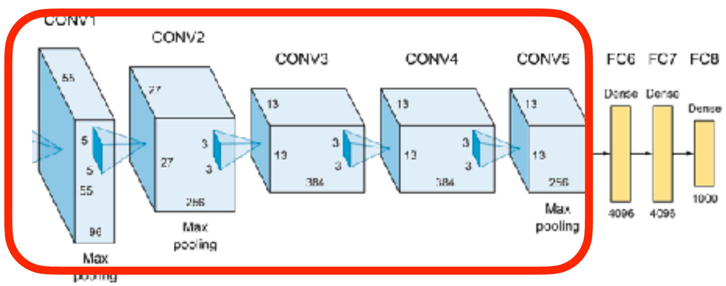
Instead of 3 x 3 convolutions we will use 3 x 3 x 3 convolutions.

3D Convolutional Networks

- Deep 3-dimensional convolutional networks (3D CNNs) for spatiotemporal feature learning from video inputs.



video frames 1, 2, … , L



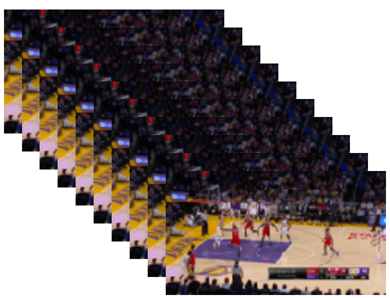
→ Playing Basketball: 0.89

Instead of 3 x 3 convolutions we will use 3 x 3 x 3 convolutions.

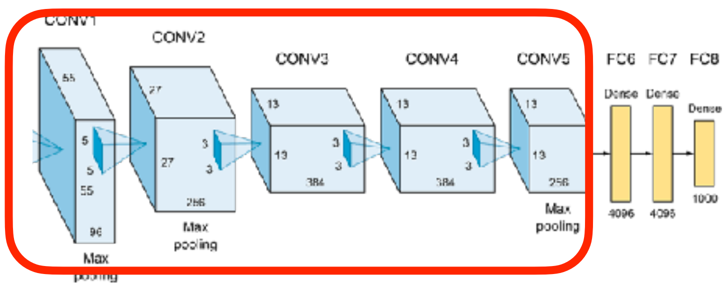
The additional dimension in the convolutional kernel will allow us to learn spatiotemporal information from multiple frames.

3D Convolutional Networks

- Deep 3-dimensional convolutional networks (3D CNNs) for spatiotemporal feature learning from video inputs.



video frames 1, 2, … , L



Playing Basketball: 0.89

Instead of 3 x 3 convolutions we will use 3 x 3 x 3 convolutions.

Temporal Modeling

Spatial Modeling

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$$f = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 5 & 6 & 7 & 8 \\ \hline 9 & 10 & 11 & 12 \\ \hline 13 & 14 & 15 & 16 \\ \hline \end{array}$$

2D convolutional filter

$$g = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$h_{ij} = \sum_{m=0}^{M} \sum_{n=0}^{N} g_{mn} f_{(i-m)(j-n)} \qquad \begin{array}{l} i = 1, 2, ..., H \\ j = 1, 2, ..., W \end{array}$$

A sliding window operation across the entire grid $f \in \mathbb{R}^{H \times W}$.

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$$f = \begin{array}{|c|c|c|c|} \hline 1 & 2 & 3 & 4 \\ \hline 5 & 6 & 7 & 8 \\ \hline 9 & 10 & 11 & 12 \\ \hline 13 & 14 & 15 & 16 \\ \hline \end{array}$$

2D convolutional filter

$$g = \begin{array}{|c|c|c|} \hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$h_{ij} = \sum_{m=0}^{M} \sum_{n=0}^{N} g_{mn} f_{(i-m)(j-n)} \qquad \begin{array}{l} i = 1, 2, ..., H \\ j = 1, 2, ..., W \end{array}$$

A sliding window operation across the entire grid $f \in \mathbb{R}^{H \times W}$.

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$$f = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 5 & 6 & 7 & 8 \\ \hline 9 & 10 & 11 & 12 \\ \hline 13 & 14 & 15 & 16 \\ \hline \end{array}$$

2D convolutional filter

$$g = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$h_{ij} = \sum_{m=0}^{M} \sum_{n=0}^{N} g_{mn} f_{(i-m)(j-n)} \qquad \begin{array}{l} i = 1, 2, ..., H \\ j = 1, 2, ..., W \end{array}$$

A sliding window operation across the entire grid $f \in \mathbb{R}^{H \times W}$.

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$$f = \begin{array}{|c|c|c|c|}\hline 1 & 2 & 3 & 4 \\ \hline 5 & 6 & 7 & 8 \\ \hline 9 & 10 & 11 & 12 \\ \hline 13 & 14 & 15 & 16 \\ \hline \end{array}$$

2D convolutional filter

$$g = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

$$h_{ij} = \sum_{m=0}^{M} \sum_{n=0}^{N} g_{mn} f_{(i-m)(j-n)} \qquad \begin{array}{l} i = 1, 2, ..., H \\ j = 1, 2, ..., W \end{array}$$

A sliding window operation across the entire grid $f \in \mathbb{R}^{H \times W}$.

2D Convolution

a 2D grid of values that we want to convolve (e.g. an image)

$$f = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}$$

2D convolutional filter

$$g = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

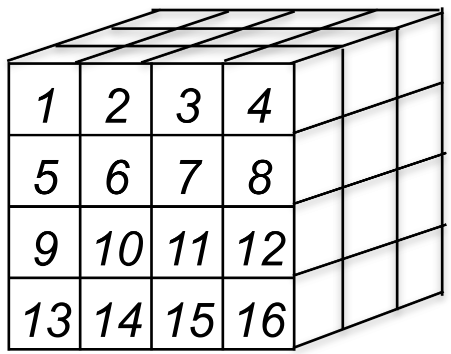
$$h_{ij} = \sum_{m=0}^{M} \sum_{n=0}^{N} g_{mn} f_{(i-m)(j-n)} \qquad \begin{array}{l} i = 1, 2, ..., H \\ j = 1, 2, ..., W \end{array}$$

A sliding window operation across the entire grid $f \in \mathbb{R}^{H \times W}$.

3D Convolution

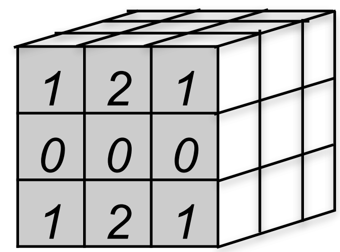
a 3D grid of values that we want to convolve (e.g. a video)

$f =$



3D convolutional filter

$g =$



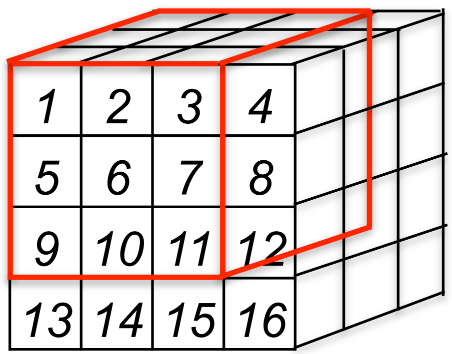
$$h_{ijk} = \sum_{m=0}^{M} \sum_{n=0}^{N} \sum_{l=0}^{L} g_{mnk} f_{(i-m)(j-n)(k-l)}$$

A sliding volume operation across the entire 3D grid $f \in \mathbb{R}^{H \times W \times T}$.

3D Convolution

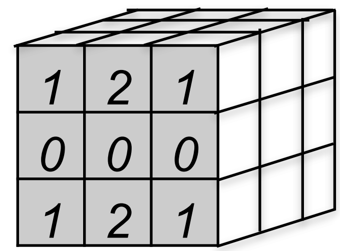
a 3D grid of values that we
want to convolve (e.g. a video)

$f =$



3D convolutional filter

$g =$



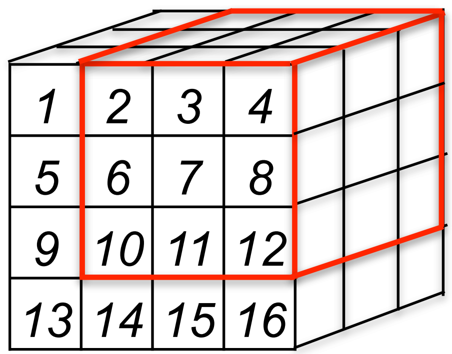
$$h_{ijk} = \sum_{m=0}^{M} \sum_{n=0}^{N} \sum_{l=0}^{L} g_{mnk} f_{(i-m)(j-n)(k-l)}$$

A sliding volume operation across the
entire 3D grid $f \in \mathbb{R}^{H \times W \times T}$.

3D Convolution

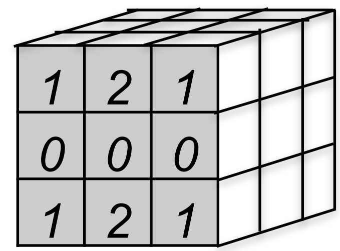
a 3D grid of values that we want to convolve (e.g. a video)

$f =$



3D convolutional filter

$g =$



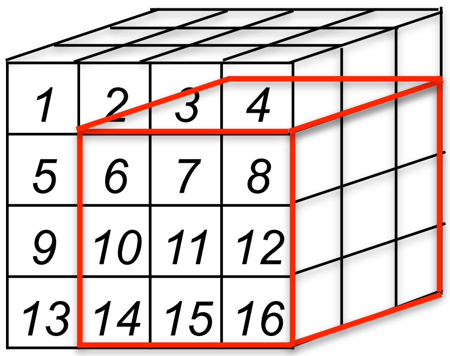
$$h_{ijk} = \sum_{m=0}^{M} \sum_{n=0}^{N} \sum_{l=0}^{L} g_{mnk} f_{(i-m)(j-n)(k-l)}$$

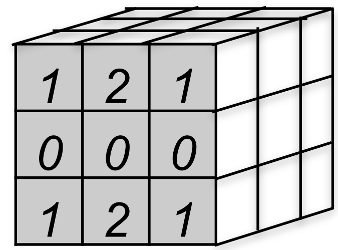
A sliding volume operation across the entire 3D grid $f \in \mathbb{R}^{H \times W \times T}$.

3D Convolution

a 3D grid of values that we want to convolve (e.g. a video)

3D convolutional filter

$f =$ 

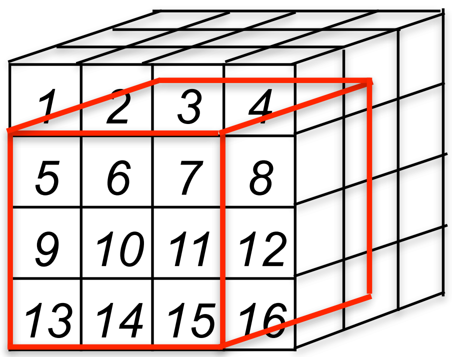
$g =$ 

$$h_{ijk} = \sum_{m=0}^{M} \sum_{n=0}^{N} \sum_{l=0}^{L} g_{mnk} f_{(i-m)(j-n)(k-l)}$$

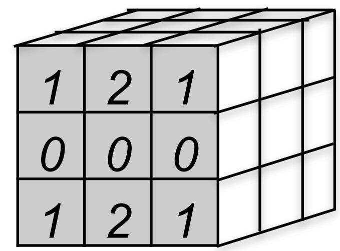
A sliding volume operation across the entire 3D grid $f \in \mathbb{R}^{H \times W \times T}$.

3D Convolution

a 3D grid of values that we want to convolve (e.g. a video)

$f =$ 

3D convolutional filter

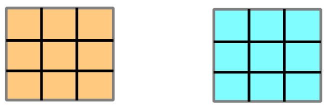
$g =$ 

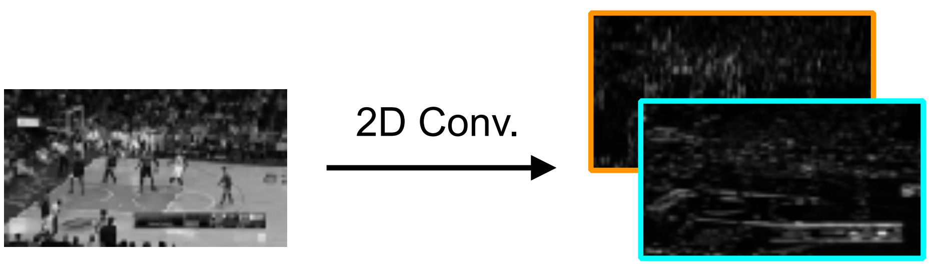
$$h_{ijk} = \sum_{m=0}^{M} \sum_{n=0}^{N} \sum_{l=0}^{L} g_{mnk} f_{(i-m)(j-n)(k-l)}$$

A sliding volume operation across the entire 3D grid $f \in \mathbb{R}^{H \times W \times T}$.

2D Convolution

Learnable **3** x **3** Convolutional Kernels **(Spatial)**



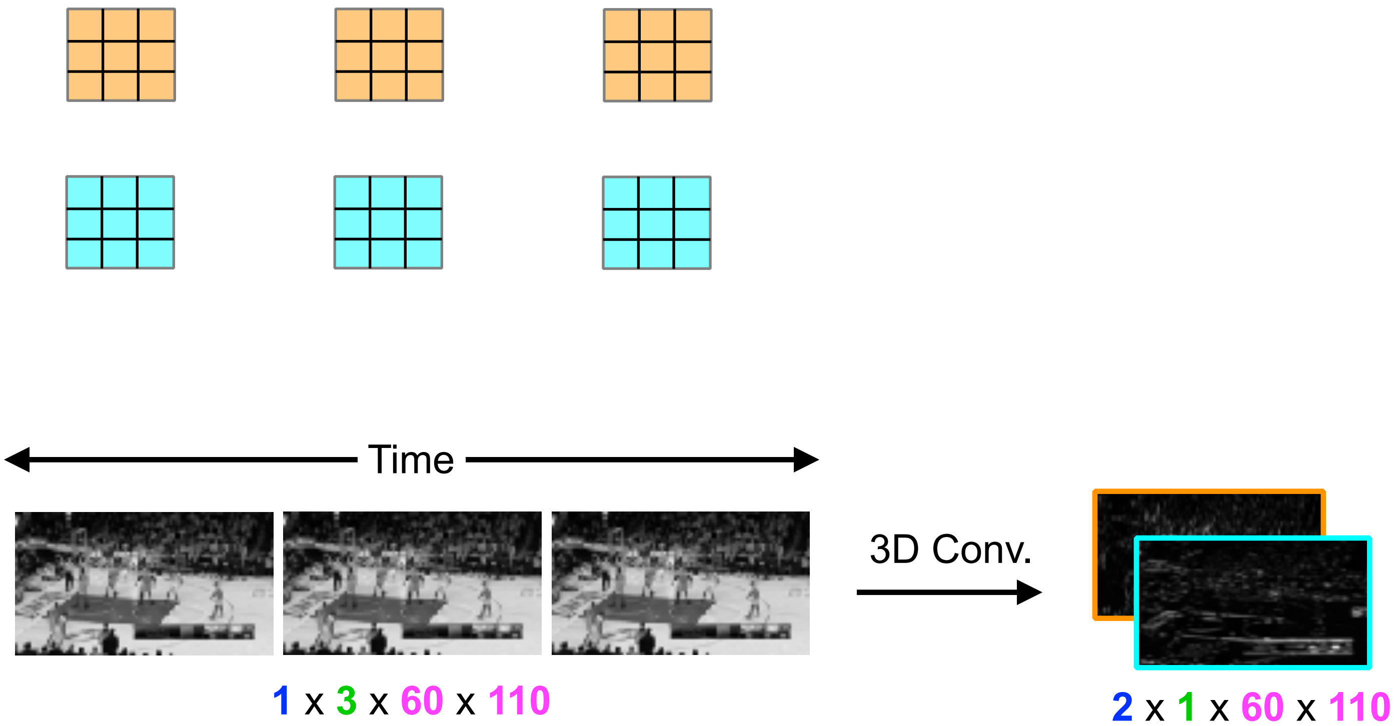


1 x **60** x **110**

2 x **60** x **110**

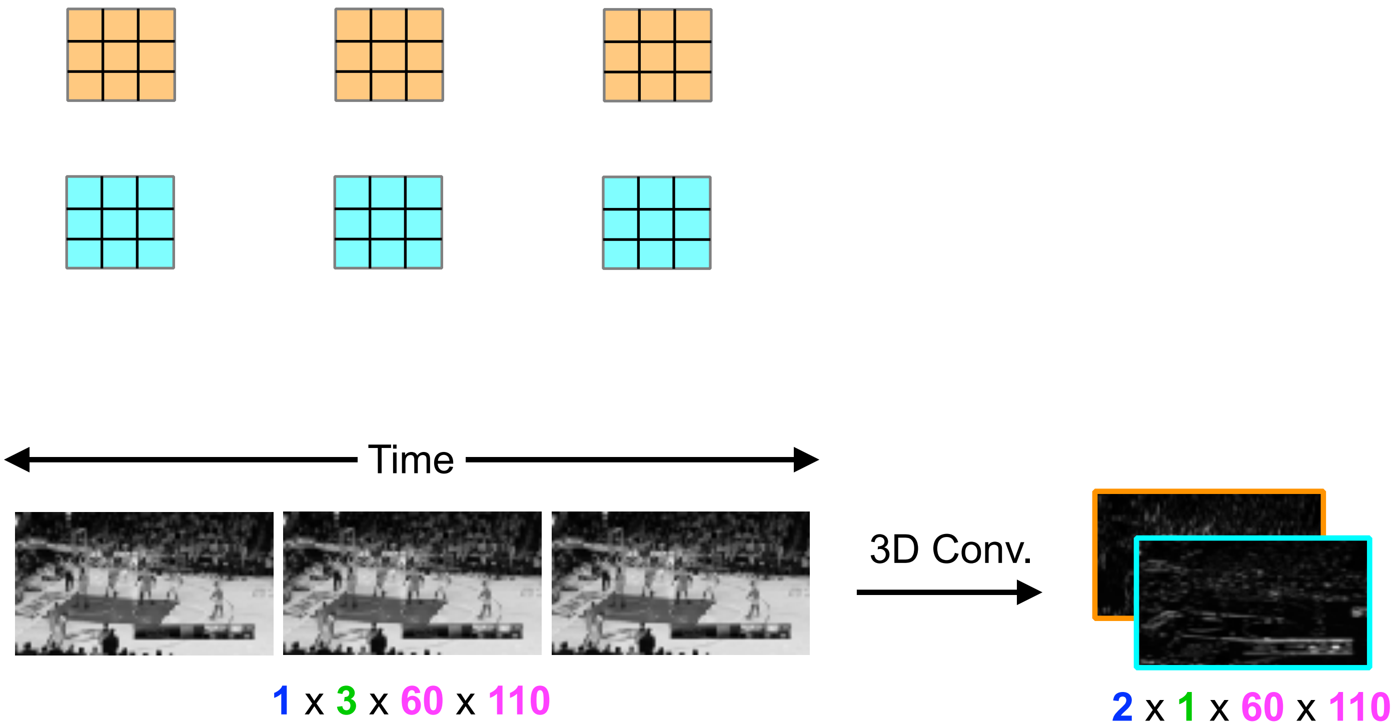
3D Convolution

Learnable **3** x **3** x **3** Convolutional Kernels (**Temporal**, **Spatial**)



3D Convolution

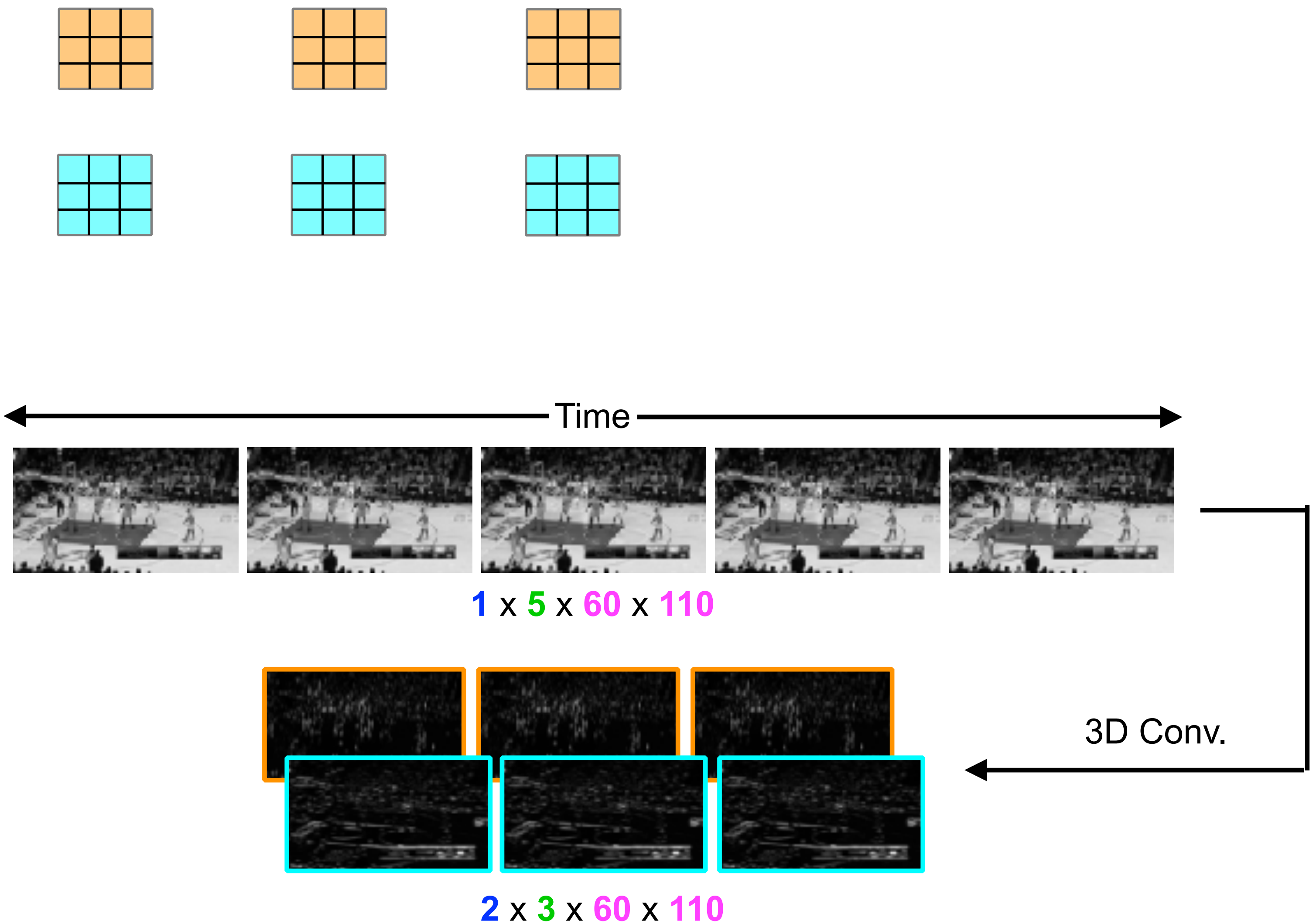
Learnable 3 x 3 x 3 Convolutional Kernels (Temporal, Spatial)



3D convolution enables learning temporal information from the video.

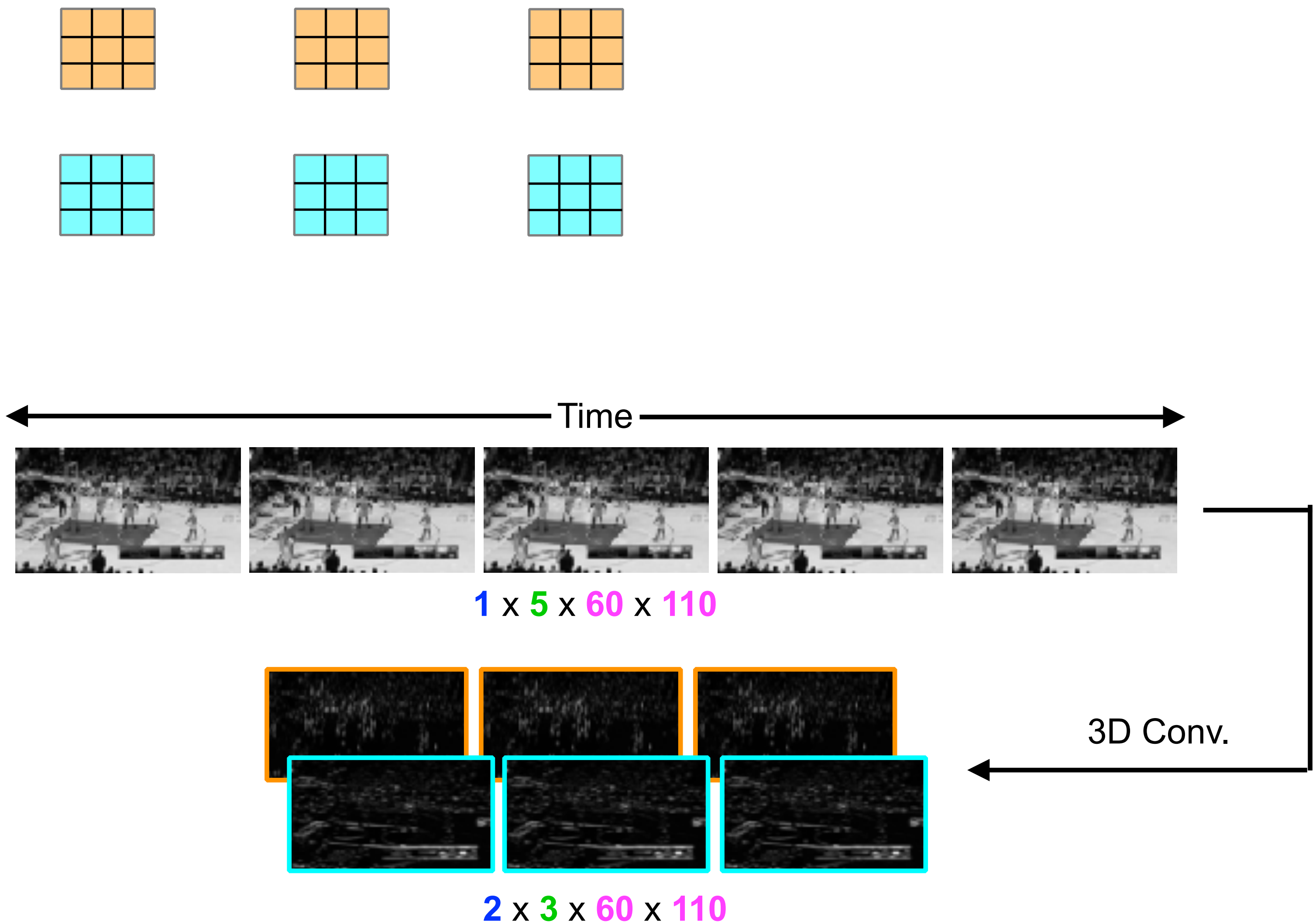
3D Convolution

Learnable 3 x 3 x 3 Convolutional Kernels (Temporal, Spatial)



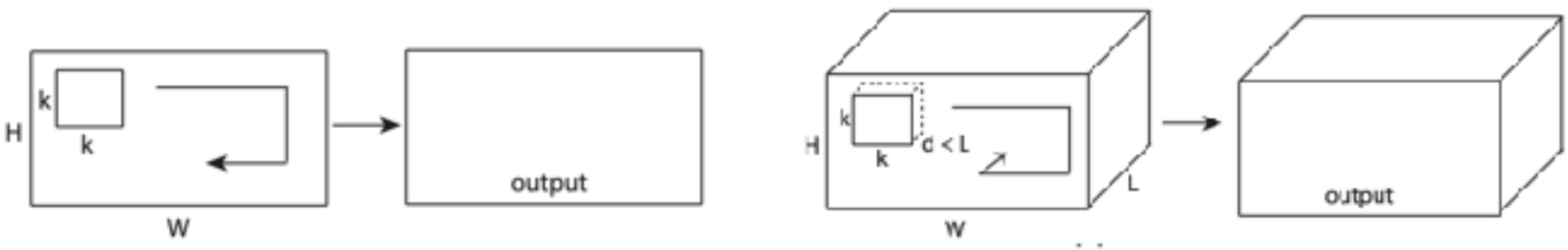
3D Convolution

Learnable 3 x 3 x 3 Convolutional Kernels (Temporal, Spatial)



3D Convolution

- Applying 2D convolution produces a 2D grid.
- Applying 3D convolution yields a 3D volume that preserves temporal information.

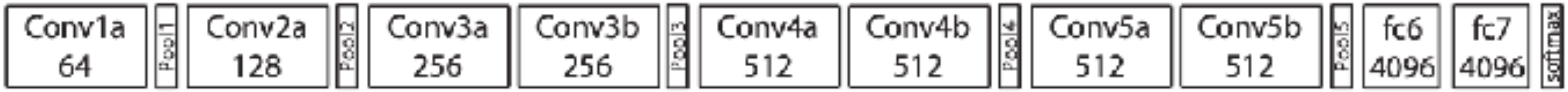


a) 2D convolution

b) 3D convolution

3D CNN Architecture

- Eight 3D convolutional layers consisting of three dimensional 3x3x3 convolutional kernels.
- Five max-pooling layers.
- Two fully connected layers followed by a softmax output layer.

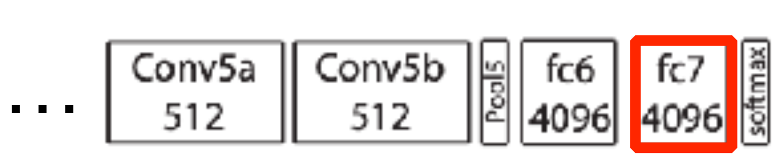


Sports-1M (Pre-training)

- 1 million YouTube videos annotated with 487 classes.
- The 3D CNN model is pre-trained on Sports-1M to predict one of the 487 sports categories for each video.

Transfer Learning to UCF-101

- UCF-101 consists of 13,320 videos from 101 action categories.
- A pretrained 3D CNN model is finetuned on UCF-101.
- A linear classifier (e.g., SVM) is trained on top of 3D CNN features from the last fully connected layer.



a) Feature Extraction using a 3D CNN

b) UCF-101

Results on UCF-101

- Performance evaluated as action recognition accuracy.

Method	Accuracy (%)
Imagenet + linear SVM	68.8
iDT w/ BoW + linear SVM	76.2
Deep networks [18]	65.4
Spatial stream network [36]	72.6
LRCN [6]	71.1
LSTM composite model [39]	75.8
C3D (1 net) + linear SVM	82.3
C3D (3 nets) + linear SVM	**85.2**
iDT w/ Fisher vector [31]	87.9
Temporal stream network [36]	83.7
Two-stream networks [36]	88.0
LRCN [6]	82.9
LSTM composite model [39]	84.3
Conv. pooling on long clips [29]	88.2
LSTM on long clips [29]	88.6
Multi-skip feature stacking [25]	89.1
C3D (3 nets) + iDT + linear SVM	**90.4**

Results on UCF-101

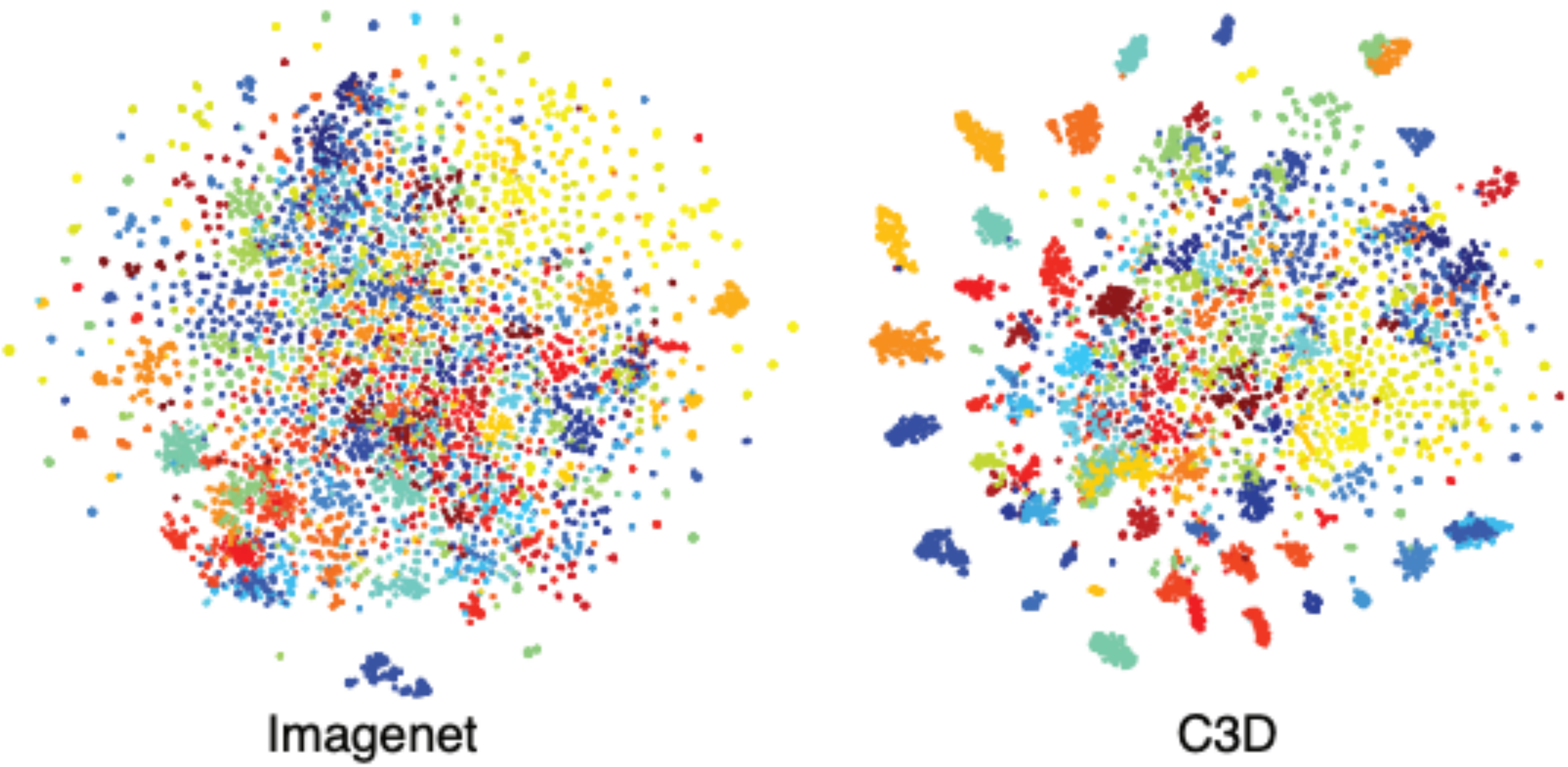
- Performance evaluated as action recognition accuracy.

Method	Accuracy (%)
Imagenet + linear SVM	68.8
iDT w/ BoW + linear SVM	76.2
Deep networks [18]	65.4
Spatial stream network [36]	72.6
LRCN [6]	71.1
LSTM composite model [39]	75.8
C3D (1 net) + linear SVM	82.3
C3D (3 nets) + linear SVM	**85.2**
iDT w/ Fisher vector [31]	87.9
Temporal stream network [36]	83.7
Two-stream networks [36]	88.0
LRCN [6]	82.9
LSTM composite model [39]	84.3
Conv. pooling on long clips [29]	88.2
LSTM on long clips [29]	88.6
Multi-skip feature stacking [25]	89.1
C3D (3 nets) + iDT + linear SVM	**90.4**

C3D is better than all prior approaches, including LRCN

Results on UCF-101

- Feature embedding visualizations of Imagenet and C3D on UCF101 using t-SNE.



Runtime Analysis

- Runtime comparison between C3D and prior action recognition methods.

Method Usage	iDT CPU	Brox's CPU	Brox's GPU	C3D GPU
RT (hours)	202.2	2513.9	607.8	2.2
FPS	3.5	0.3	1.2	313.9
x Slower	91.4	1135.9	274.6	1

Runtime Analysis

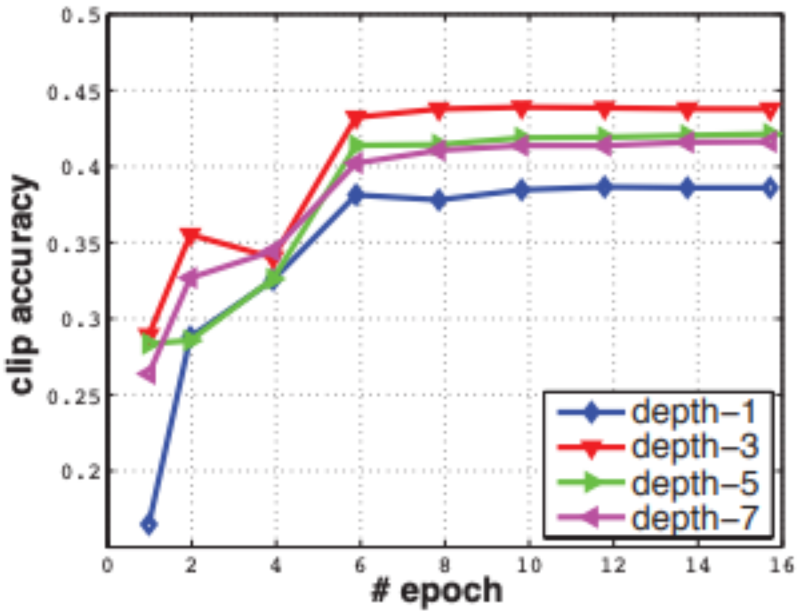
- Runtime comparison between C3D and prior action recognition methods.

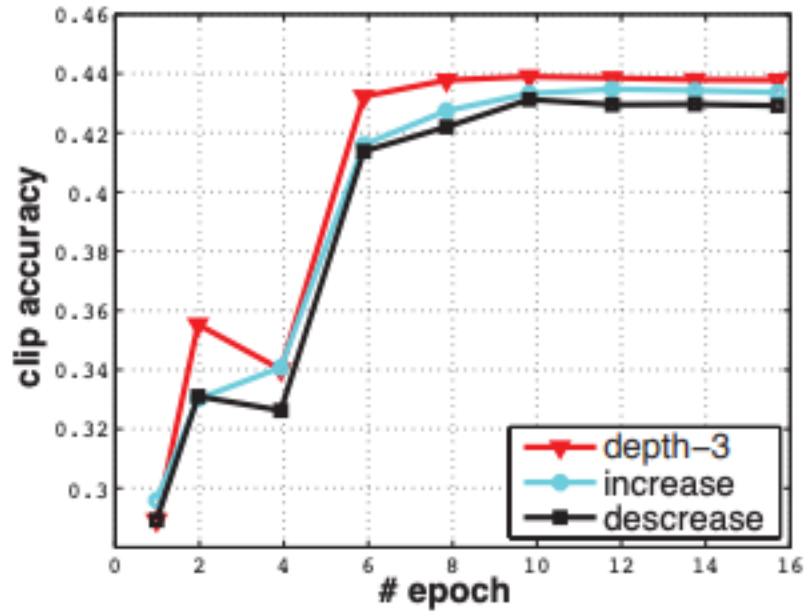
Method Usage	iDT CPU	Brox's CPU	Brox's GPU	C3D GPU
RT (hours)	202.2	2513.9	607.8	2.2
FPS	3.5	0.3	1.2	313.9
x Slower	91.4	1135.9	274.6	1

C3D is 91x faster than iDT and 274x faster than Brox's GPU implementation.

Ablations

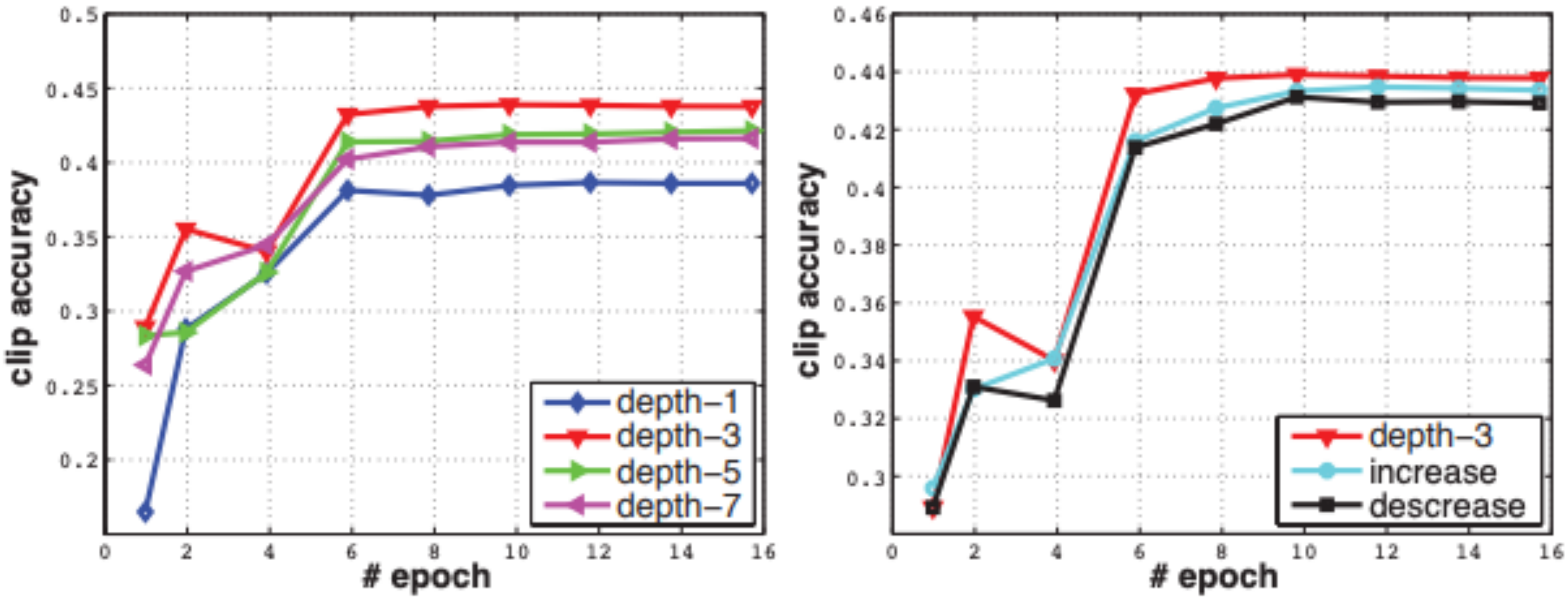
- Comparison between different variants of 3D CNN architectures.





Ablations

- Comparison between different variants of 3D CNN architectures.



Based on these results, 3x3x3 is the best kernel choice for 3D CNNs.

Summary

- A simple, yet effective 3D CNN architecture for large-scale spatiotemporal feature learning.
- Better accuracy and inference run-time than prior hand-crafted or optical flow-based action recognition methods.
- 3D CNNs have much larger learning capacity than 2D CNNs.
- 3D CNNs are much more costly than 2D CNNs.

Discussion Points

- Why are 3x3x3 convolutional kernels most effective? Are they sufficient for learning temporal video dynamics?
- Other variants of 3D CNNs were proposed before this paper. Considering this, why was this paper so impactful?

First Assignment

- The reading list is posted here.
- Select the following:
    1. Seven 30min or 45min papers for standard paper presentations (marked **red** and **purple** in the schedule). Any combo of the papers suffice (e.g., five 30min & two 45min papers, all 30min papers, etc.)
    2. Three 20min papers for paper battles (marked **green** in the schedule).
- Make sure that the papers that you selected will **NOT** be presented by me.
- Rank the papers in each of these lists in descending order of preference (from highest to lowest) and upload them to Canvas **by Sunday, Aug 27th, 11:59 PM** (please include paper IDs in your lists!!).
- I will then update the website with the paper assignments.

Second Assignment

- Complete the paper critique for paper [5] SlowFast Networks for Video Recognition.
- Upload it to Canvas by **1 PM on Wednesday, August 30th**.