Lecture 10: Video Understanding
Last time: Image Captioning with RNNs and Attention

Extract spatial features from a pretrained CNN

Alignment scores: H x W

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_{1,0,0}$</td>
<td>$e_{1,0,1}$</td>
<td>$e_{1,0,2}$</td>
</tr>
<tr>
<td>$e_{1,1,0}$</td>
<td>$e_{1,1,1}$</td>
<td>$e_{1,1,2}$</td>
</tr>
<tr>
<td>$e_{1,2,0}$</td>
<td>$e_{1,2,1}$</td>
<td>$e_{1,2,2}$</td>
</tr>
</tbody>
</table>

Attention: H x W

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$a_{1,0,0}$</td>
<td>$a_{1,0,1}$</td>
<td>$a_{1,0,2}$</td>
</tr>
<tr>
<td>$a_{1,1,0}$</td>
<td>$a_{1,1,1}$</td>
<td>$a_{1,1,2}$</td>
</tr>
<tr>
<td>$a_{1,2,0}$</td>
<td>$a_{1,2,1}$</td>
<td>$a_{1,2,2}$</td>
</tr>
</tbody>
</table>

Features: H x W x D

$Z_{0,0}$ $Z_{0,1}$ $Z_{0,2}$

$Z_{1,0}$ $Z_{1,1}$ $Z_{1,2}$

$Z_{2,0}$ $Z_{2,1}$ $Z_{2,2}$

This entire process is differentiable.
- model chooses its own attention weights. No attention supervision is required

**Last time:** Self-Attention

**Inputs:**
- Input vectors: $x$ (shape: $N \times D$)

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

**Outputs:**
- Context vectors: $y$ (shape: $D_v$)

**Self-attention**
Last time: Transformer

Encoder

Decoder
Recall: (2D) Image classification

(assume given a set of possible labels) {dog, cat, truck, plane, ...}

This image by Nikita is licensed under CC-BY 2.0
Next Lecture: (2D) Detection and Segmentation

Classification

Semantic Segmentation

Object Detection

Instance Segmentation

CAT

GRASS, CAT, TREE, SKY

DOG, DOG, CAT

DOG, DOG, CAT

No spatial extent

No objects, just pixels

Multiple Objects

Fei-Fei Li, Yunzhu Li, Ruohan Gao

Lecture 10 - 6

May 4, 2023
Today: Video = 2D + Time

A video is a sequence of images
4D tensor: $T \times 3 \times H \times W$
(or $3 \times T \times H \times W$)
Example task: **Video Classification**

Input video: $T \times 3 \times H \times W$

Swimming
Running
Jumping
Eating
Standing
Example task: Video Classification

Images: Recognize objects

- Dog
- Cat
- Fish
- Truck

Videos: Recognize actions

- Swimming
- Running
- Jumping
- Eating
- Standing

Running video is in the public domain

Slide credit: Justin Johnson
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
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)
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

"Running" CNN "Running" CNN "Running" CNN "Running" CNN "Running" CNN "Running" CNN "Running" CNN

Slide credit: Justin Johnson
Video Classification: Late Fusion (with FC layers)

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

Frame features

\[ T \times D \times H' \times W' \]

2D CNN on each frame

\[ \text{CNN} \]

Input:

\[ T \times 3 \times H \times W \]

Clip features:

\[ TDH'W' \]

Flatten

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

Class scores: \( C \)

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

Slide credit: Justin Johnson
**Video Classification: Late Fusion (with pooling)**

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

- **Frame features**
  - \( T \times D \times H' \times W' \)

- **2D CNN on each frame**

- **Average Pool over space and time**

- **Clip features:** \( D \)

- **Class scores:** \( C \)

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

Input:
- \( T \times 3 \times H \times W \)

Slide credit: Justin Johnson
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

- **Input:** $T \times 3 \times H \times W$
- **Frame features:** $T \times D \times H' \times W'$
- **2D CNN on each frame**
- **Average Pool over space and time**
- **Clip features:** $D$
- **Linear**
- **Class scores:** $C$
- **Run 2D CNN on each frame, pool features and feed to Linear**

Slide credit: Justin Johnson
Video Classification: Early Fusion

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

First 2D convolution collapses all temporal information:

- **Input**: $3T \times H \times W$
- **Output**: $D \times H \times W$

**Input**: $T \times 3 \times H \times W$

**Reshape**: $3T \times H \times W$

**Class scores**: $C$

Rest of the network is standard 2D CNN

---

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

Slide credit: Justin Johnson
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**: $3T \times H \times W$
- **Output**: $D \times H \times W$

Rest of the network is standard 2D CNN

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

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

Slide credit: Justin Johnson
Video Classification: 3D CNN

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

Each layer in the network is a 4D tensor: \( D \times T \times H \times W \)
Use 3D conv and 3D pooling operations.

Input: \( 3 \times T \times H \times W \)

Class scores: \( C \)

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

Slide credit: Justin Johnson
Convolution Layer

- 32x32x3 image
- 5x5x3 filter
- Convolve (slide) over all spatial locations
- Activation map

Fei-Fei Li, Yunzhu Li, Ruohan Gao
3D Convolution

Input: $C \times T \times H \times W$

- 6x6x6 conv
- 5x5x5 conv
- 4x4x4 conv

Class Layer

Class Scores
## Early Fusion vs Late Fusion vs 3D CNN

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size (C x T x H x W)</th>
<th>Receptive Field (T x H x W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>3 x 20 x 64 x 64</td>
<td>1 x 3 x 3</td>
</tr>
<tr>
<td>Conv2D(3x3, 3-&gt;12)</td>
<td>12 x 20 x 64 x 64</td>
<td>1 x 3 x 3</td>
</tr>
</tbody>
</table>

(Small example architectures, in practice much bigger)

Slide credit: Justin Johnson
Early Fusion vs Late Fusion vs 3D CNN

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size (C x T x H x W)</th>
<th>Receptive Field (T x H x W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>3 x 20 x 64 x 64</td>
<td></td>
</tr>
<tr>
<td>Conv2D(3x3, 3-&gt;12)</td>
<td>12 x 20 x 64 x 64</td>
<td>1 x 3 x 3</td>
</tr>
</tbody>
</table>

Late Fusion

Conv(3x3) (Small example architectures, in practice much bigger)

Slide credit: Justin Johnson
Early Fusion vs Late Fusion vs 3D CNN

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size (C x T x H x W)</th>
<th>Receptive Field (T x H x W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>3 x 20 x 64 x 64</td>
<td></td>
</tr>
<tr>
<td>Conv2D(3x3, 3-&gt;12)</td>
<td>12 x 20 x 64 x 64</td>
<td>1 x 3 x 3</td>
</tr>
<tr>
<td>Pool2D(4x4)</td>
<td>12 x 20 x 16 x 16</td>
<td>1 x 6 x 6</td>
</tr>
</tbody>
</table>

Late Fusion

(Small example architectures, in practice much bigger)

Slide credit: Justin Johnson
## Early Fusion vs Late Fusion vs 3D CNN

### Late Fusion

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size (C x T x H x W)</th>
<th>Receptive Field (T x H x W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>3 x 20 x 64 x 64</td>
<td></td>
</tr>
<tr>
<td>Conv2D(3x3, 3-&gt;12)</td>
<td>12 x 20 x 64 x 64</td>
<td>1 x 3 x 3</td>
</tr>
<tr>
<td>Pool2D(4x4)</td>
<td>12 x 20 x 16 x 16</td>
<td>1 x 6 x 6</td>
</tr>
<tr>
<td>Conv2D(3x3, 12-&gt;24)</td>
<td>24 x 20 x 16 x 16</td>
<td>1 x 14 x 14</td>
</tr>
</tbody>
</table>

Build slowly in space

(Small example architectures, in practice much bigger)

Slide credit: Justin Johnson

---

Fei-Fei Li, Yunzhu Li, Ruohan Gao  
Lecture 10 - 29  
May 4, 2023
Early Fusion vs Late Fusion vs 3D CNN

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size (C x T x H x W)</th>
<th>Receptive Field (T x H x W)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>3 x 20 x 64 x 64</td>
<td>1 x 3 x 3</td>
</tr>
<tr>
<td>Conv2D(3x3, 3-&gt;12)</td>
<td>12 x 20 x 64 x 64</td>
<td>1 x 3 x 3</td>
</tr>
<tr>
<td>Pool2D(4x4)</td>
<td>12 x 20 x 16 x 16</td>
<td>1 x 6 x 6</td>
</tr>
<tr>
<td>Conv2D(3x3, 12-&gt;24)</td>
<td>24 x 20 x 16 x 16</td>
<td>1 x 14 x 14</td>
</tr>
<tr>
<td>GlobalAvgPool</td>
<td>24 x 1 x 1 x 1</td>
<td>20 x 64 x 64</td>
</tr>
</tbody>
</table>

Build slowly in space, All-at-once in time at end

(Small example architectures, in practice much bigger)

Slide credit: Justin Johnson
## Early Fusion vs Late Fusion vs 3D CNN

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size (C x T x H x W)</th>
<th>Receptive Field (T x H x W)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Late Fusion</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>3 x 20 x 64 x 64</td>
<td></td>
</tr>
<tr>
<td>Conv2D(3x3, 3-&gt;12)</td>
<td>12 x 20 x 64 x 64</td>
<td>1 x 3 x 3</td>
</tr>
<tr>
<td>Pool2D(4x4)</td>
<td>12 x 20 x 16 x 16</td>
<td>1 x 6 x 6</td>
</tr>
<tr>
<td>Conv2D(3x3, 12-&gt;24)</td>
<td>24 x 20 x 16 x 16</td>
<td>1 x 14 x 14</td>
</tr>
<tr>
<td>GlobalAvgPool</td>
<td>24 x 1 x 1 x 1</td>
<td>20 x 64 x 64</td>
</tr>
<tr>
<td><strong>Early Fusion</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>3 x 20 x 64 x 64</td>
<td></td>
</tr>
<tr>
<td>Conv2D(3x3, 3*20-&gt;12)</td>
<td>12 x 64 x 64</td>
<td>20 x 3 x 3</td>
</tr>
<tr>
<td>Pool2D(4x4)</td>
<td>12 x 16 x 16</td>
<td>20 x 6 x 6</td>
</tr>
<tr>
<td>Conv2D(3x3, 12-&gt;24)</td>
<td>24 x 16 x 16</td>
<td>20 x 14 x 14</td>
</tr>
<tr>
<td>GlobalAvgPool</td>
<td>24 x 1 x 1</td>
<td>20 x 64 x 64</td>
</tr>
</tbody>
</table>

Build slowly in space, All-at-once in time at end

Build slowly in space, All-at-once in time at start

(Small example architectures, in practice much bigger)

Slide credit: Justin Johnson
# Early Fusion vs Late Fusion vs 3D CNN

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size $(C \times T \times H \times W)$</th>
<th>Receptive Field $(T \times H \times W)$</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Late Fusion</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>$3 \times 20 \times 64 \times 64$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv2D(3x3, 3-&gt;12)</td>
<td>$12 \times 20 \times 64 \times 64$</td>
<td>$1 \times 3 \times 3$</td>
<td>Build slowly in space, All-at-once in time at end</td>
</tr>
<tr>
<td>Pool2D(4x4)</td>
<td>$12 \times 20 \times 16 \times 16$</td>
<td>$1 \times 6 \times 6$</td>
<td></td>
</tr>
<tr>
<td>Conv2D(3x3, 12-&gt;24)</td>
<td>$24 \times 20 \times 16 \times 16$</td>
<td>$1 \times 14 \times 14$</td>
<td></td>
</tr>
<tr>
<td>GlobalAvgPool</td>
<td>$24 \times 1 \times 1 \times 1$</td>
<td>$20 \times 64 \times 64$</td>
<td></td>
</tr>
<tr>
<td><strong>Early Fusion</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>$3 \times 20 \times 64 \times 64$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv2D(3x3, 3*20-&gt;12)</td>
<td>$12 \times 64 \times 64$</td>
<td>$20 \times 3 \times 3$</td>
<td>Build slowly in space, All-at-once in time at start</td>
</tr>
<tr>
<td>Pool2D(4x4)</td>
<td>$12 \times 16 \times 16$</td>
<td>$20 \times 6 \times 6$</td>
<td></td>
</tr>
<tr>
<td>Conv2D(3x3, 12-&gt;24)</td>
<td>$24 \times 16 \times 16$</td>
<td>$20 \times 14 \times 14$</td>
<td></td>
</tr>
<tr>
<td>GlobalAvgPool</td>
<td>$24 \times 1 \times 1$</td>
<td>$20 \times 64 \times 64$</td>
<td></td>
</tr>
<tr>
<td><strong>3D CNN</strong></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>$3 \times 20 \times 64 \times 64$</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Conv3D(3x3x3, 3-&gt;12)</td>
<td>$12 \times 20 \times 64 \times 64$</td>
<td>$3 \times 3 \times 3$</td>
<td>Build slowly in space, Build slowly in time &quot;Slow Fusion&quot; (Small example architectures, in practice much bigger)</td>
</tr>
<tr>
<td>Pool3D(4x4x4)</td>
<td>$12 \times 5 \times 16 \times 16$</td>
<td>$6 \times 6 \times 6$</td>
<td></td>
</tr>
<tr>
<td>Conv3D(3x3x3, 12-&gt;24)</td>
<td>$24 \times 5 \times 16 \times 16$</td>
<td>$14 \times 14 \times 14$</td>
<td></td>
</tr>
<tr>
<td>GlobalAvgPool</td>
<td>$24 \times 1 \times 1$</td>
<td>$20 \times 64 \times 64$</td>
<td></td>
</tr>
</tbody>
</table>
# Early Fusion vs Late Fusion vs 3D CNN

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size</th>
<th>Receptive Field</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Late Fusion</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>$3 \times 20 \times 64 \times 64$</td>
<td></td>
</tr>
<tr>
<td>Conv2D(3x3, 3-&gt;12)</td>
<td>$12 \times 20 \times 64 \times 64$</td>
<td>$1 \times 3 \times 3$</td>
</tr>
<tr>
<td>Pool2D(4x4)</td>
<td>$12 \times 20 \times 16 \times 16$</td>
<td>$1 \times 6 \times 6$</td>
</tr>
<tr>
<td>Conv2D(3x3, 12-&gt;24)</td>
<td>$24 \times 20 \times 16 \times 16$</td>
<td>$1 \times 14 \times 14$</td>
</tr>
<tr>
<td>GlobalAvgPool</td>
<td>$24 \times 1 \times 1 \times 1$</td>
<td>$20 \times 64 \times 64$</td>
</tr>
<tr>
<td><strong>Early Fusion</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>$3 \times 20 \times 64 \times 64$</td>
<td></td>
</tr>
<tr>
<td>Conv2D(3x3, 3*20-&gt;12)</td>
<td>$12 \times 64 \times 64$</td>
<td>$20 \times 3 \times 3$</td>
</tr>
<tr>
<td>Pool2D(4x4)</td>
<td>$12 \times 16 \times 16$</td>
<td>$20 \times 6 \times 6$</td>
</tr>
<tr>
<td>Conv2D(3x3, 12-&gt;24)</td>
<td>$24 \times 16 \times 16$</td>
<td>$20 \times 14 \times 14$</td>
</tr>
<tr>
<td>GlobalAvgPool</td>
<td>$24 \times 1 \times 1$</td>
<td>$20 \times 64 \times 64$</td>
</tr>
<tr>
<td><strong>3D CNN</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td>$3 \times 20 \times 64 \times 64$</td>
<td></td>
</tr>
<tr>
<td>Conv3D(3x3x3, 3-&gt;12)</td>
<td>$12 \times 20 \times 64 \times 64$</td>
<td>$3 \times 3 \times 3$</td>
</tr>
<tr>
<td>Pool3D(4x4x4)</td>
<td>$12 \times 5 \times 16 \times 16$</td>
<td>$6 \times 6 \times 6$</td>
</tr>
<tr>
<td>Conv3D(3x3x3, 12-&gt;24)</td>
<td>$24 \times 5 \times 16 \times 16$</td>
<td>$14 \times 14 \times 14$</td>
</tr>
<tr>
<td>GlobalAvgPool</td>
<td>$24 \times 1 \times 1$</td>
<td>$20 \times 64 \times 64$</td>
</tr>
</tbody>
</table>

**What is the difference?**

- **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, Build slowly in time "Slow Fusion"

(Small example architectures, in practice much bigger)

Slide credit: Justin Johnson
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

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

**Weight:** \( C_{out} \times C_{in} \times T \times 3 \times 3 \) Slide over \( x \) and \( y \)

**Output:** \( C_{out} \times H \times W \) 2D grid with \( C_{out} \)-dim feat at each point

Slide credit: Justin Johnson
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

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

**Weight:**
- \( C_{\text{out}} \times C_{\text{in}} \times T \times 3 \times 3 \) Slide over x and y
- \( C_{\text{out}} \) different filters

**Output:**
- \( C_{\text{out}} \times H \times W \) 2D grid with \( C_{\text{out}} \)-dim feat at each point

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

\[ H = 224 \]
\[ T = 16 \]
\[ W = 224 \]

Slide credit: Justin Johnson
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

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

Weight:
\( C_{out} \times C_{in} \times T \times 3 \times 3 \)
Slide over x and y

Output:
\( C_{out} \times H \times W \)
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

How to recognize blue to orange transitions anywhere in space and time?

Slide credit: Justin Johnson
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

**Input:** $C_{\text{in}} \times T \times H \times W$

(3D grid with $C_{\text{in}}$-dim feat at each point)

**Weight:**

$C_{\text{out}} \times C_{\text{in}} \times 3 \times 3 \times 3$

Slide over x and y

**Output:**

$C_{\text{out}} \times T \times H \times W$

3D grid with $C_{\text{out}}$-dim feat at each point

How to recognize blue to orange transitions anywhere in space and time?

Slide credit: Justin Johnson
**2D Conv (Early Fusion) vs 3D Conv (3D CNN)**

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

**Weight:**  
\( C_{out} \times C_{in} \times 3 \times 3 \times 3 \)  
Slide over x and y

**Output:**  
\( C_{out} \times T \times H \times W \)  
3D grid with \( C_{out} \)-dim feat at each point

How to recognize blue to orange transitions anywhere in space and time?

Slide credit: Justin Johnson
2D Conv (Early Fusion) vs 3D Conv (3D CNN)

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

**Weight:**
$C_{out} \times C_{in} \times 3 \times 3 \times 3$
Slide over $x$ and $y$

First-layer filters have shape $3 \times 4 \times 5 \times 5$
Can visualize as video clips!

How to recognize **blue** to **orange**
transitions anywhere in space and time?

Temporal shift-invariant since each filter slides over time!
Example Video Dataset: **Sports-1M**

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

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

Slide credit: Justin Johnson
Early Fusion vs Late Fusion vs 3D CNN

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

Single Frame model works well – always try this first!

3D CNNs have improved a lot since 2014!

Slide credit: Justin Johnson
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

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>3 x 16 x 112 x 112</td>
</tr>
<tr>
<td>Conv1 (3x3x3)</td>
<td>64 x 16 x 112 x 112</td>
</tr>
<tr>
<td>Pool1 (1x2x2)</td>
<td>64 x 16 x 56 x 56</td>
</tr>
<tr>
<td>Conv2 (3x3x3)</td>
<td>128 x 16 x 56 x 56</td>
</tr>
<tr>
<td>Pool2 (2x2x2)</td>
<td>128 x 8 x 28 x 28</td>
</tr>
<tr>
<td>Conv3a (3x3x3)</td>
<td>256 x 8 x 28 x 28</td>
</tr>
<tr>
<td>Conv3b (3x3x3)</td>
<td>256 x 8 x 28 x 28</td>
</tr>
<tr>
<td>Pool3 (2x2x2)</td>
<td>256 x 4 x 14 x 14</td>
</tr>
<tr>
<td>Conv4a (3x3x3)</td>
<td>512 x 4 x 14 x 14</td>
</tr>
<tr>
<td>Conv4b (3x3x3)</td>
<td>512 x 4 x 14 x 14</td>
</tr>
<tr>
<td>Pool4 (2x2x2)</td>
<td>512 x 2 x 7 x 7</td>
</tr>
<tr>
<td>Conv5a (3x3x3)</td>
<td>512 x 2 x 7 x 7</td>
</tr>
<tr>
<td>Conv5b (3x3x3)</td>
<td>512 x 2 x 7 x 7</td>
</tr>
<tr>
<td>Pool5</td>
<td>512 x 1 x 3 x 3</td>
</tr>
<tr>
<td>FC6</td>
<td>4096</td>
</tr>
<tr>
<td>FC7</td>
<td>4096</td>
</tr>
<tr>
<td>FC8</td>
<td>C</td>
</tr>
</tbody>
</table>

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!)

---

<table>
<thead>
<tr>
<th>Layer</th>
<th>Size</th>
<th>MFLOPs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input</td>
<td>3 x 16 x 112 x 112</td>
<td></td>
</tr>
<tr>
<td>Conv1 (3x3x3)</td>
<td>64 x 16 x 112 x 112</td>
<td>1.04</td>
</tr>
<tr>
<td>Pool1 (1x2x2)</td>
<td>64 x 16 x 56 x 56</td>
<td></td>
</tr>
<tr>
<td>Conv2 (3x3x3)</td>
<td>128 x 16 x 56 x 56</td>
<td>11.10</td>
</tr>
<tr>
<td>Pool2 (2x2x2)</td>
<td>128 x 8 x 28 x 28</td>
<td></td>
</tr>
<tr>
<td>Conv3a (3x3x3)</td>
<td>256 x 8 x 28 x 28</td>
<td>5.55</td>
</tr>
<tr>
<td>Conv3b (3x3x3)</td>
<td>256 x 8 x 28 x 28</td>
<td>11.10</td>
</tr>
<tr>
<td>Pool3 (2x2x2)</td>
<td>256 x 4 x 14 x 14</td>
<td></td>
</tr>
<tr>
<td>Conv4a (3x3x3)</td>
<td>512 x 4 x 14 x 14</td>
<td>2.77</td>
</tr>
<tr>
<td>Conv4b (3x3x3)</td>
<td>512 x 4 x 14 x 14</td>
<td>5.55</td>
</tr>
<tr>
<td>Pool4 (2x2x2)</td>
<td>512 x 2 x 7 x 7</td>
<td></td>
</tr>
<tr>
<td>Conv5a (3x3x3)</td>
<td>512 x 2 x 7 x 7</td>
<td>0.69</td>
</tr>
<tr>
<td>Conv5b (3x3x3)</td>
<td>512 x 2 x 7 x 7</td>
<td>0.69</td>
</tr>
<tr>
<td>Pool5</td>
<td>512 x 1 x 3 x 3</td>
<td></td>
</tr>
<tr>
<td>FC6</td>
<td>4096</td>
<td>0.51</td>
</tr>
<tr>
<td>FC7</td>
<td>4096</td>
<td>0.45</td>
</tr>
<tr>
<td>FC8</td>
<td>C</td>
<td>0.05</td>
</tr>
</tbody>
</table>

Tran et al., “Learning Spatiotemporal Features with 3D Convolutional Networks”, ICCV 2015

Slide credit: Justin Johnson
Early Fusion vs Late Fusion vs 3D CNN

Sports-1M Top-5 Accuracy

- Single Frame: 77.7
- Early Fusion: 76.8
- Late Fusion: 78.7
- 3D CNN: 80.2
- C3D: 84.4

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

Slide credit: Justin Johnson
Recognizing Actions from Motion

We can easily recognize actions using only motion information.
Measuring Motion: Optical Flow

Image at frame t

Image at frame t+1

Measuring Motion: Optical Flow

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:

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

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


Slide credit: Justin Johnson
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:

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

$I_{t+1}(x+dx, y+dy) = I_t(x, y)$
Separating Motion and Appearance: Two-Stream Networks

**Input:** Single Image
3 x H x W

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

**Early fusion:** First 2D conv processes all flow images


Slide credit: Justin Johnson
Separating Motion and Appearance: Two-Stream Networks

Accuracy on UCF-101

- 3D CNN: 65.4%
- Spatial only: 73%
- Temporal only: 83.7%
- Two-stream (fuse by average): 86.9%
- Two-stream (fuse by SVM): 88%


Slide credit: Justin Johnson
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?

Slide credit: Justin Johnson
Modeling long-term temporal structure

Extract features with CNN (2D or 3D)

Time

CNN  CNN  CNN  CNN  CNN
Modeling long-term temporal structure

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

Extract features with CNN (2D or 3D)

CNN → CNN → CNN → CNN → CNN → CNN

Time

Slide credit: Justin Johnson
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)

Slide credit: Justin Johnson
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)

CNN → CNN → CNN → CNN → CNN → CNN

Time
Modeling long-term temporal structure

Sometimes don’t backprop to CNN to save memory; pretrain and use it as a feature extractor

Extract features with CNN (2D or 3D)

Modeling long-term temporal structure

Inside CNN: Each value is 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?

Recall: Multi-layer RNN

We can use a similar structure to process videos!
Recurrent Convolutional Network

Entire network uses 2D feature maps: \(C \times H \times W\)

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


Slide credit: Justin Johnson
Recurrent Convolutional Network

Normal 2D CNN:

Input features: \( C \times H \times W \)

\[ \text{2D Conv} \]

Output features: \( C \times H \times W \)
Recall: Recurrent Network

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

new state

old state

some function with parameters $W$

Features from layer $L-1$, timestep $t-1$

Features from layer $L$, timestep $t$

RNN-like recurrence

Features for layer $L$, timestep $t$


Slide credit: Justin Johnson
Recall: Vanilla RNN

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

Replace all matrix multiply with 2D convolution!


Slide credit: Justin Johnson
Modeling long-term temporal structure

RNN: Infinite temporal extent (fully-connected)

CNN: finite temporal extent (convolutional)


Recurrent CNN: Infinite temporal extent (convolutional)

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

Slide credit: Justin Johnson
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)

Recurrent CNN: Infinite temporal extent (convolutional)

---


Slide credit: Justin Johnson
Recall: Self-Attention

Inputs:
Input vectors: $x$ (shape: $N \times D$)

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

Outputs:
context vectors: $y$ (shape: $D_v$)

Self-attention

Input vectors:
$x_0, x_1, x_2$

Key vectors:
$k_0, k_1, k_2$

Value vectors:
$v_0, v_1, v_2$

Query vectors:
$q_0, q_1, q_2$

Alignment:
$e_{0,0}, e_{0,1}, e_{0,2}, e_{1,0}, e_{1,1}, e_{1,2}, e_{2,0}, e_{2,1}, e_{2,2}$

Output vectors:
$y_0, y_1, y_2$
Spatio-Temporal Self-Attention (Nonlocal Block)

3D CNN

Input clip

Features: \(C \times T \times H \times W\)

Nonlocal Block

Wang et al, “Non-local neural networks”, CVPR 2018

Slide credit: Justin Johnson
Spatio-Temporal Self-Attention (Nonlocal Block)

Features: $C \times T \times H \times W$

Queries: $C' \times T \times H \times W$

Keys: $C' \times T \times H \times W$

Values: $C' \times T \times H \times W$

1x1x1 Conv

Input clip

3D CNN

Nonlocal Block

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

Slide credit: Justin Johnson
Spatio-Temporal Self-Attention (Nonlocal Block)

Input clip

3D CNN

Features: $C \times T \times H \times W$

Queries: $C' \times T \times H \times W$

Keys: $C' \times T \times H \times W$

Values: $C' \times T \times H \times W$

1x1x1 Conv

1x1x1 Conv

1x1x1 Conv

Transpose

softmax

Attention Weights

$(THW) \times (THW)$

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

Slide credit: Justin Johnson
Spatio-Temporal Self-Attention (Nonlocal Block)

**Features:**
- $C \times T \times H \times W$

**Queries:**
- $C' \times T \times H \times W$

**Keys:**
- $C' \times T \times H \times W$

**Values:**
- $C' \times T \times H \times W$

**1x1x1 Conv**

**Transpose**

**softmax**

**Attention Weights**
- $(THW) \times (THW)$

**1x1x1 Conv**

**Nonlocal Block**

Wang et al, “Non-local neural networks”, CVPR 2018

Slide credit: Justin Johnson
Spatio-Temporal Self-Attention (Nonlocal Block)

Features: $C \times T \times H \times W$

Queries: $C' \times T \times H \times W$

Keys: $C' \times T \times H \times W$

Values: $C' \times T \times H \times W$

1x1x1 Conv

$1 \times 1 \times 1$ Conv

$1 \times 1 \times 1$ Conv

Attention Weights

$(THW) \times (THW)$

Input clip

3D CNN

Nonlocal Block

Wang et al, “Non-local neural networks”, CVPR 2018

Slide credit: Justin Johnson
Spatio-Temporal Self-Attention (Nonlocal Block)

3D CNN

Features: \( C \times T \times H \times W \)

Queries: \( C' \times T \times H \times W \)

Keys: \( C' \times T \times H \times W \)

Values: \( C' \times T \times H \times W \)

1x1x1 Conv

Transpose

softmax

Attention Weights \((THW) \times (THW)\)

1x1x1 Conv

Residual Connection

\( C \times T \times H \times W \)

\( C' \times T \times H \times W \)

1x1x1 Conv

Wang et al, “Non-local neural networks”, CVPR 2018

Slide credit: Justin Johnson
Spatio-Temporal Self-Attention (Nonlocal Block)

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

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.

Replace each 2D $K_h \times K_w$ conv/pool layer with a 3D $K_t \times K_h \times K_w$ version

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

---

**Inceptio Block: Original**

- 5x5 Conv
- 3x3 Conv
- 1x1 Conv

---

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

Slide credit: Justin Johnson
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.

Inception Block: Inflated

---

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

Fei-Fei Li, Yunzhu Li, Ruohan Gao
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

Slide credit: Justin Johnson
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
Vision Transformers for Video

Factorized attention: Attend over space / time

Pooling module: Reduce number of tokens

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

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

Slide credit: Justin Johnson
Vision Transformers for Video

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

Slide credit: Justin Johnson
Visualizing Video Models

Add a term to encourage spatially smooth flow; tune penalty to pick out “slow” vs “fast” motion

Figure credit: Simonyan and Zisserman, “Two-stream convolutional networks for action recognition in videos”, NeurIPS 2014
Feichtenhofer et al, “What have we learned from deep representations for action recognition?”, CVPR 2018

Slide credit: Justin Johnson
Can you guess the action?

Appearance  “Slow” motion  “Fast” motion

Feichtenhofer et al, “What have we learned from deep representations for action recognition?”, CVPR 2018
Slide credit: Christoph Feichtenhofer

Slide credit: Justin Johnson
Can you guess the action?  Weightlifting

Appearance  “Slow” motion  “Fast” motion

“Bar Shaking”  “Push overhead”

Slide credit: Justin Johnson
Can you guess the action?

Appearance  
“Slow” motion  
“Fast” motion

Slide credit: Justin Johnson
Can you guess the action? Apply Eye Makeup

Appearance

“Slow” motion

“Fast” motion

Fast motion appearance

Slide credit: Justin Johnson
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 both space and time and classify the activities they are performing.

Some examples from AVA Dataset:

- clink glass $\rightarrow$ drink
- open $\rightarrow$ close
- grab (a person) $\rightarrow$ hug
- look at phone $\rightarrow$ answer phone


Slide credit: Justin Johnson
Today: Temporal Stream

3D CNN, Two-Stream Neural Network, Spatial-Temporal Self-Attention……
Ba Ba Ba

(McGurk & McDonald 1976)
Fa Fa Fa Fa ... (McGurk & McDonald 1976)
Visually-guided audio source separation

Speech mixture

Gao et al., VisualVoice, CVPR 2021
Separated voice for the left speaker

Gao et al., VisualVoice, CVPR 2021
Separated voice for the right speaker

Gao et al., VisualVoice, CVPR 2021
Musical instruments source separation

Train on 100,000 unlabeled multi-source video clips, then separate audio for novel video.

Gao & Grauman, Co-Separating Sounds of Visual Objects, ICCV 2019
Audio as a preview mechanism for efficient action recognition in untrimmed videos

Gao et al., Listen to Look: Action Recognition by Previewing Audio, CVPR 2020
**Multimodal Video Understanding**

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

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

EPIC-Fusion: Audio-Visual Temporal Binding for Egocentric Action Recognition, Kazakos et al., ICCV 2019
Learning audio-visual synchronization

Owens & Efros, Audio-visual scene analysis with self-supervised multisensory features, ECCV 2018
Korbar et al., Co-training of audio and video representations from self-supervised temporal synchronization, NeurIPS 2018
Learning audio-visual synchronization

Owens & Efros, Audio-visual scene analysis with self-supervised multisensory features, ECCV 2018
Learning audio-visual synchronization

Motion

Loudness

Time

Slide Credit: Andrew Owens
Learning audio-visual synchronization

Aligned vs. misaligned

3D Convolution
3D Convolution
3D Convolution

3D Convolution

1D Convolution
1D Convolution
1D Convolution

Owens & Efros, Audio-visual scene analysis with self-supervised multisensory features, ECCV 2018
Top responses in test set

Owens & Efros, Audio-visual scene analysis with self-supervised multisensory features, ECCV 2018
Sound source localization

Top responses per category
(speech examples omitted)

Dribbling basketball

Owens & Efros, Audio-visual scene analysis with self-supervised multisensory features, ECCV 2018
Arandjelović and Zisserman, ECCV 2018; Senocak et al. CVPR 2018; Kidron et al. CVPR 2005 ...
Next time: Object Detection and Image Segmentation