Action Recognition with CNNs

CIS 680
Action Recognition

**Goal:** Recognize an activity from a given video.

**Input:** A Video

**Output:** A Basketball Activity
Long Short Term Memory Network

LSTM Summary:

- An extension of traditional CNNs that allows to model temporal relationships (e.g. videos, text sequences, etc).
Long Short Term Memory Network

LSTM Summary:

- An extension of traditional CNNs that allows to model temporal relationships (e.g. videos, text sequences, etc).

- This is achieved via a memory mechanism that allows the network to remember what has happened in the past.
Long Short Term Memory Network

**LSTM Summary:**

- An extension of traditional CNNs that allows to model temporal relationships (e.g. videos, text sequences, etc).

- This is achieved via a memory mechanism that allows the network to remember what has happened in the past.

- Note that standard CNNs process each input (e.g. frame in the video) separately completely forgetting what has happened before.
Long Short Term Memory Network

\( \hat{y}_{t-1} \)
\[ h_{t-1} \]

\( C_{t-2} \)
\[ h_{t-2} \]
\[ x_{t-1} \]

\( C_{t-1} \)
\[ h_{t-1} \]
\[ x_t \]

\( C_t \)
\[ h_t \]
\[ x_{t+1} \]

\( \hat{y}_t \)
\[ h_t \]

\( \hat{y}_{t+1} \)
\[ h_{t+1} \]
Long Short Term Memory Network

\[ \hat{y}_{t-1} \]
\[ h_{t-1} \]

\[ \hat{y}_t \]
\[ h_t \]

\[ \hat{y}_{t+1} \]
\[ h_{t+1} \]

\[ C_{t-2} \]
\[ h_{t-2} \]
\[ x_{t-1} \]

\[ C_{t-1} \]
\[ h_{t-1} \]
\[ x_t \]

\[ C_t \]
\[ h_t \]
\[ x_{t+1} \]
Notation:

- elementwise multiplication
- concatenation
- sigmoid function
- tanh function
- elementwise summation
Long Short Term Memory Network

\[ C_{t-1} \]

Previous Memory Cell State

\[ h_{t-1} \]

\[ x_t \]
Long Short Term Memory Network
Long Short Term Memory Network

Current Input

\[ x_t \]

\[ h_{t-1} \]
Long Short Term Memory Network

The Forget Gate:

\[ f_t = \sigma(W_f[h_{t-1}, x_t]^T) \]
The Forget Gate:

$$f_t = \sigma(W_f[h_{t-1}, x_t]^T)$$
Long Short Term Memory Network

The Input Gate:

\[ i_t = \sigma(W_i[h_{t-1}, x_t]^T) \]
Long Short Term Memory Network

\[ \begin{align*}
C_{t-1} & \quad f_t \\
\cdot & \quad \sigma \\
W_f[h_{t-1}, x_t]^T & \quad W_i[h_{t-1}, x_t]^T \\
\cdot & \quad \sigma \\
W_c[h_{t-1}, x_t]^T & \quad \cdot \\
\cdot & \quad \text{tanh}
\end{align*} \]

\[ a_t = \text{tanh}(W_c[h_{t-1}, x_t]^T) \]

New Candidate Cell Values:
Long Short Term Memory Network

\[ C_{t-1} \]

\[ h_{t-1} \]

\[ x_t \]

\[ f_t \]

\[ i_t \]

\[ a_t \]

\[ C_t \]

\[ h_t \]

Time \( t \)

\[ W_f[h_{t-1}, x_t]^T \]

\[ W_i[h_{t-1}, x_t]^T \]

\[ W_c[h_{t-1}, x_t]^T \]

\[ \sigma \]

\[ \sigma \]

\[ \tanh \]
Long Short Term Memory Network

Memory Cell Update:

\[ C_t = f_t \odot C_{t-1} + i_t \odot a_t \]
Memory Cell Update:

\[ C_t = f_t \odot C_{t-1} + i_t \odot a_t \]
Long Short Term Memory Network

Memory Cell Update:

\[ C_t = f_t \odot C_{t-1} + i_t \odot a_t \]

What new information to add
The Output Gate:

\[ o_t = \sigma(W_o[h_{t-1}, x_t]^T) \]
Long Short Term Memory Network

Output:

\[ h_t = o_t \odot \tanh(C_t) \]
Long Short Term Memory Network

\[ C_{t-1} \]

\[ h_{t-1} \]

\[ x_t \]

\[ C_t \]

\[ h_t \]

\[ W_f[h_{t-1}, x_t]^T \]

\[ W_i[h_{t-1}, x_t]^T \]

\[ W_c[h_{t-1}, x_t]^T \]

\[ W_o[h_{t-1}, x_t]^T \]

\[ f_t \]

\[ i_t \]

\[ a_t \]

\[ o_t \]

\[ \sigma \]

\[ \sigma \]

\[ \sigma \]

\[ \sigma \]

\[ \text{tanh} \]

\[ \text{tanh} \]

\[ \text{tanh} \]

\[ \text{tanh} \]
Long Short Term Memory Network

Forward Pass:

\[ \hat{y}_{t-1} \]
\[ h_{t-1} \]
\[ x_{t-1} \]
\[ C_{t-2} \]

Time t-1

\[ h_{t-2} \]

\[ C_{t-1} \]

Time t

\[ h_{t-1} \]

\[ x_t \]

\[ x_t \]

\[ C_t \]

Time t+1

\[ h_t \]

\[ h_t \]

\[ x_{t+1} \]

\[ h_{t+1} \]
Long Short Term Memory Network

Backward Pass:

\[
\frac{\partial L}{\partial \hat{y}_{t-1}} \quad \frac{\partial L}{\partial \hat{y}_t} \quad \frac{\partial L}{\partial \hat{y}_{t+1}}
\]

\[
\frac{\partial L}{\partial h_{t-1}} \quad \frac{\partial L}{\partial h_t} \quad \frac{\partial L}{\partial h_{t+1}}
\]

\[
\frac{\partial L}{\partial W} \quad \frac{\partial L}{\partial W} \quad \frac{\partial L}{\partial W}
\]

Time t-1 \quad Time t \quad Time t+1

\[
\frac{\partial L}{\partial C_{t-2}} \quad \frac{\partial L}{\partial C_{t-1}} \quad \frac{\partial L}{\partial C_t}
\]

\[
\frac{\partial L}{\partial h_{t-2}} \quad \frac{\partial L}{\partial h_{t-1}} \quad \frac{\partial L}{\partial h_t}
\]
Long Short Term Memory Network

\[
C_{t-1} \quad f_t \quad i_t \quad a_t \quad o_t \quad C_t
\]

\[
h_{t-1} \quad W_f[h_{t-1}, x_t]^T \quad W_i[h_{t-1}, x_t]^T \quad W_c[h_{t-1}, x_t]^T \quad W_o[h_{t-1}, x_t]^T
\]

\[\frac{\partial L}{\partial h_t}\]
Long Short Term Memory Network

Forward: \[ h_t = o_t \odot \tanh(C_t) \]

Backward: \[ \frac{\partial L}{\partial o_t} = \frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial o_t} = \frac{\partial L}{\partial h_t} \odot \tanh(C_t) \]
Long Short Term Memory Network

\[ C_{t-1} \]

\[ h_{t-1} \]

\[ x_t \]

\[ C_t \]

\[ \frac{\partial L}{\partial h_t} \]

Forward: \[ h_t = o_t \odot \tanh(C_t) \]

Backward: \[
\frac{\partial L}{\partial C_t} = \frac{\partial L}{\partial h_t} \frac{\partial h_t}{\partial C_t} = \frac{\partial L}{\partial h_t} \odot o_t \odot (1 - \tanh^2(C_t))
\]
Long Short Term Memory Network

Forward: \[ C_t = f_t \odot C_{t-1} + i_t \odot a_t \]

Backward: \[ \frac{\partial L}{\partial C_{t-1}} = \frac{\partial L}{\partial C_t} \frac{\partial C_t}{\partial C_{t-1}} = \frac{\partial L}{\partial C_t} \odot f_t \]
Long Short Term Memory Network

\[
C_t = f_t \odot C_{t-1} + i_t \odot a_t
\]

**Forward:**

**Backward:**

\[
\frac{\partial L}{\partial a_t} = \frac{\partial L}{\partial C_t} \frac{\partial C_t}{\partial a_t} = \frac{\partial L}{\partial C_t} \odot i_t
\]
Long Short Term Memory Network

\[
\begin{align*}
C_t &= f_t \odot C_{t-1} + i_t \odot a_t \\
\frac{\partial L}{\partial i_t} &= \frac{\partial L}{\partial C_t} \frac{\partial C_t}{\partial i_t} = \frac{\partial L}{\partial C_t} \odot a_t
\end{align*}
\]
Long Short Term Memory Network

\[ C_t = f_t \odot C_{t-1} + i_t \odot a_t \]

**Forward:**

\[ \frac{\partial L}{\partial h_t} \]

**Backward:**

\[ \frac{\partial L}{\partial f_t} = \frac{\partial L}{\partial C_t} \frac{\partial C_t}{\partial f_t} = \frac{\partial L}{\partial C_t} \odot C_{t-1} \]
Long Short Term Memory Network

\[ C_{t-1} \]

\[ h_{t-1} \]

\[ x_t \]

\[ \sigma(\cdot) = \frac{1}{1 + \exp(-\cdot)} \]

**Forward:**
\[ \sigma(z) = \frac{1}{1 + \exp(-z)} \]

**Backward:**
\[ \frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z)) \]
Long Short Term Memory Network

\[
C_{t-1} \xrightarrow{f_t} \sigma \xrightarrow{W_f[h_{t-1}, x_t]^T} C_t \xleftarrow{C_t}
\]

\[
h_t-1 \xrightarrow{i_t} \sigma \xrightarrow{W_i[h_{t-1}, x_t]^T} a_t \xrightarrow{W_c[h_{t-1}, x_t]^T} \xrightarrow{W_o[h_{t-1}, x_t]^T} \frac{\partial L}{\partial h_t}
\]

Forward: \( \tanh(z) \)

Backward: \[ \frac{\partial \tanh(z)}{\partial z} = 1 - \tanh^2(z) \]
**Long Short Term Memory Network**

### Forward:

\[ z_t = W[h_{t-1}, x_t]^T = \begin{bmatrix} W_f \\ W_i \\ W_c \\ W_o \end{bmatrix} [h_{t-1}, x_t]^T = \begin{bmatrix} f_t \\ i_t \\ c_t \\ o_t \end{bmatrix} \]

### Backward:

\[ \frac{\partial L}{\partial W} = \frac{\partial L}{\partial z_t} \frac{\partial z_t}{\partial W} = \frac{\partial L}{\partial z_t} [h_{t-1}, x_t] \]
Long Short Term Memory Network

\[ C_{t-1} \]

\[ C_t \]

\[ h_{t-1} \]

\[ x_t \]

**Forward:**

\[ z_t = W[h_{t-1}, x_t]^T = \begin{bmatrix} W_f & W_i & W_c & W_o \end{bmatrix} [h_{t-1}, x_t]^T = \begin{bmatrix} W_f \hat{f}_t \hat{i}_t \hat{a}_t \hat{o}_t \end{bmatrix} \]

**Backward:**

\[ \frac{\partial L}{\partial h_{t-1}} = \frac{\partial L}{\partial z_t} \frac{\partial z_t}{\partial h_{t-1}} = W^T \frac{\partial L}{\partial z_t} \]
Long Short Term Memory Network

\[ C_t \]

\[ h_{t-1} \]

\[ x_t \]

\[ W_f[h_{t-1}, x_t]^T \]

\[ W_i[h_{t-1}, x_t]^T \]

\[ W_c[h_{t-1}, x_t]^T \]

\[ W_o[h_{t-1}, x_t]^T \]

\[ f_t \]

\[ i_t \]

\[ a_t \]

\[ o_t \]

\[ \sigma \]

\[ \sigma \]

\[ \sigma \]

\[ \tanh \]

\[ \partial L / \partial h_t \]
1. For each timestep $t = T \ldots 1$, let $\frac{\partial L}{\partial h_t}$ be the summed gradient from the loss layer and from LSTM unit $t + 1$. 
1. For each timestep $t = T \ldots 1$, let $\frac{\partial L}{\partial h_t}$ be the summed gradient from the loss layer and from LSTM unit $t + 1$.

• Compute: $\frac{\partial L}{\partial C_t} = \frac{\partial L}{\partial h_t} \odot o_t \odot (1 - \tanh^2(C_t)) + \frac{\partial L}{\partial C_{t+1}} \odot f_{t+1}$
1. For each timestep $t = T \ldots 1$, let $\frac{\partial L}{\partial h_t}$ be the summed gradient from the loss layer and from LSTM unit $t + 1$.

- Compute: 
  \[
  \frac{\partial L}{\partial C_t} = \frac{\partial L}{\partial h_t} \odot o_t \odot (1 - \tanh^2(C_t)) + \frac{\partial L}{\partial C_{t+1}} \odot f_{t+1}
  \]

- Compute gradients for each gate:
  \[
  \frac{\partial L}{\partial f_t} = \frac{\partial L}{\partial C_t} \odot C_{t-1}, \quad \frac{\partial L}{\partial i_t} = \frac{\partial L}{\partial C_t} \odot a_t, \quad \frac{\partial L}{\partial a_t} = \frac{\partial L}{\partial C_t} \odot i_t, \quad \frac{\partial L}{\partial o_t} = \frac{\partial L}{\partial h_t} \odot \tanh(C_t)
  \]
1. For each timestep \( t = T \ldots 1 \), let \( \frac{\partial L}{\partial h_t} \) be the summed gradient from the loss layer and from LSTM unit \( t + 1 \).

- Compute: \( \frac{\partial L}{\partial C_t} = \frac{\partial L}{\partial h_t} \odot o_t \odot (1 - \tanh^2(C_t)) + \frac{\partial L}{\partial C_t+1} \odot f_{t+1} \)

- Compute gradients for each gate:

\[
\begin{align*}
\frac{\partial L}{\partial f_t} &= \frac{\partial L}{\partial C_t} \odot C_{t-1} \\
\frac{\partial L}{\partial i_t} &= \frac{\partial L}{\partial C_t} \odot a_t \\
\frac{\partial L}{\partial a_t} &= \frac{\partial L}{\partial C_t} \odot i_t \\
\frac{\partial L}{\partial o_t} &= \frac{\partial L}{\partial h_t} \odot \tanh(C_t)
\end{align*}
\]

- Let \( z_t = W[h_{t-1}, x_t]^T = \begin{bmatrix} W_f & \hat{f}_t \\ W_i & \hat{i}_t \\ W_c & \hat{a}_t \\ W_o & \hat{o}_t \end{bmatrix} \begin{bmatrix} h_{t-1}, x_t \end{bmatrix} = \begin{bmatrix} \hat{f}_t \\ \hat{i}_t \\ \hat{a}_t \\ \hat{o}_t \end{bmatrix} \rightarrow \begin{align*}
\frac{\partial L}{\partial \hat{f}_t} &= \frac{\partial L}{\partial f_t} \sigma(f_t)(1 - \sigma(f_t)) \\
\frac{\partial L}{\partial \hat{i}_t} &= \frac{\partial L}{\partial i_t} \sigma(i_t)(1 - \sigma(i_t)) \\
\frac{\partial L}{\partial \hat{a}_t} &= \frac{\partial L}{\partial a_t} (1 - \tanh^2(a_t)) \\
\frac{\partial L}{\partial \hat{o}_t} &= \frac{\partial L}{\partial o_t} \sigma(o_t)(1 - \sigma(o_t))
\end{align*}\]
1. For each timestep $t = T \ldots 1$, let $\frac{\partial L}{\partial h_t}$ be the summed gradient from the loss layer and from LSTM unit $t + 1$.

- Compute: $\frac{\partial L}{\partial C_t} = \frac{\partial L}{\partial h_t} \odot o_t \odot (1 - \tanh^2(C_t)) + \frac{\partial L}{\partial C_{t+1}} \odot f_{t+1}$

- Compute gradients for each gate:

\[
\begin{align*}
\frac{\partial L}{\partial f_t} &= \frac{\partial L}{\partial C_t} \odot C_{t-1} \\
\frac{\partial L}{\partial i_t} &= \frac{\partial L}{\partial C_t} \odot a_t \\
\frac{\partial L}{\partial a_t} &= \frac{\partial L}{\partial C_t} \odot i_t \\
\frac{\partial L}{\partial o_t} &= \frac{\partial L}{\partial h_t} \odot \tanh(C_t)
\end{align*}
\]

- Let $z_t = W[h_{t-1}, x_t]^T = \begin{bmatrix} W_f & W_i & W_c & W_o \end{bmatrix} [h_{t-1}, x_t]^T = \begin{bmatrix} \hat{f}_t \\ \hat{i}_t \\ \hat{a}_t \\ \hat{o}_t \end{bmatrix}$

\[
\begin{align*}
\frac{\partial L}{\partial \hat{f}_t} &= \frac{\partial L}{\partial f_t} \sigma(f_t)(1 - \sigma(f_t)) \\
\frac{\partial L}{\partial \hat{i}_t} &= \frac{\partial L}{\partial i_t} \sigma(i_t)(1 - \sigma(i_t)) \\
\frac{\partial L}{\partial \hat{a}_t} &= \frac{\partial L}{\partial a_t} (1 - \tanh^2(a_t)) \\
\frac{\partial L}{\partial \hat{o}_t} &= \frac{\partial L}{\partial o_t} \sigma(o_t)(1 - \sigma(o_t))
\end{align*}
\]

- Compute gradients: $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial z_t} [h_{t-1}, x_t]$ $\frac{\partial L}{\partial h_{t-1}} = W^T \frac{\partial L}{\partial z_t}$
1. For each timestep $t = T \ldots 1$, let $\frac{\partial L}{\partial h_t}$ be the summed gradient from the loss layer and from LSTM unit $t+1$.

   • Compute: 
     $\frac{\partial L}{\partial C_t} = \frac{\partial L}{\partial h_t} \odot o_t \odot (1 - \tanh^2(C_t)) + \frac{\partial L}{\partial C_{t+1}} \odot f_{t+1}$

   • Compute gradients for each gate:
     $\frac{\partial L}{\partial f_t} = \frac{\partial L}{\partial C_t} \odot C_{t-1}$
     $\frac{\partial L}{\partial i_t} = \frac{\partial L}{\partial C_t} \odot a_t$
     $\frac{\partial L}{\partial a_t} = \frac{\partial L}{\partial C_t} \odot i_t$
     $\frac{\partial L}{\partial o_t} = \frac{\partial L}{\partial h_t} \odot \tanh(C_t)$

   • Let $z_t = W[h_{t-1}, x_t]^T = \begin{bmatrix} W_f \\ W_i \\ W_c \\ W_o \end{bmatrix} [h_{t-1}, x_t]^T = \begin{bmatrix} \hat{f}_t \\ \hat{i}_t \\ \hat{a}_t \\ \hat{o}_t \end{bmatrix}$

   • Compute gradients: 
     $\frac{\partial L}{\partial W} = \frac{\partial L}{\partial z_t} [h_{t-1}, x_t]$  
     $\frac{\partial L}{\partial h_{t-1}} = W^T \frac{\partial L}{\partial z_t}$

2. Accumulate the gradients from all time steps and update the parameters: 
   $W = W - \alpha \frac{\partial L}{\partial W}$
Long Short Term Memory Network

**Issues with LSTMs:**

- LSTM reasoning is typically done on the high-level visual appearance features.
- The network is not really learning motion cues, which are critical for accurate action prediction.
Two Stream CNNs

Simonyan et al. (NIPS 2016):

- The first stream is a standard CNN operating on RGB images.
- The second stream operates on optical flow images between two adjacent frames in the video.
- The predictions from two streams are fused/averaged at the end.
Two Stream CNNs

Simonyan et al. (NIPS 2016):

• The flow is computed between every pair of adjacent frames in a video.
Two Stream CNNs

Simonyan et al. (NIPS 2016):

- The flow is computed between every pair of adjacent frames in a video.
Two Stream CNNs

Simonyan et al. (NIPS 2016):
• The flow is computed between every pair of adjacent frames in a video.
Two Stream CNNs

Simonyan et al. (NIPS 2016):

• The flow is computed between every pair of adjacent frames in a video.
Two Stream CNNs

Simonyan et al. (NIPS 2016):

- The flow is computed between every pair of adjacent frames in a video.
Two Stream CNNs

Results:

- Two-stream architecture significantly outperforms traditional CNNs and prior methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>Spatial</th>
<th>Temporal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatio-temporal HMAX network [11, 16]</td>
<td>-</td>
<td>22.8%</td>
</tr>
<tr>
<td>“Slow fusion” spatio-temporal ConvNet [14]</td>
<td>65.4%</td>
<td>-</td>
</tr>
<tr>
<td>Spatial stream ConvNet</td>
<td>73.0%</td>
<td>40.5%</td>
</tr>
<tr>
<td>Temporal stream ConvNet</td>
<td>83.7%</td>
<td>54.6%</td>
</tr>
<tr>
<td>Two-stream model (fusion by averaging)</td>
<td>86.9%</td>
<td>58.0%</td>
</tr>
<tr>
<td>Two-stream model (fusion by SVM)</td>
<td>88.0%</td>
<td>59.4%</td>
</tr>
</tbody>
</table>
Two Stream CNNs

Results:

• Two-stream architecture significantly outperforms traditional CNNs and prior methods.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Accuracy (%)</td>
<td>-</td>
<td>22.8%</td>
<td>73.0%</td>
<td>83.7%</td>
<td>86.9%</td>
<td>88.0%</td>
</tr>
<tr>
<td>Confidence (%)</td>
<td>-</td>
<td>-</td>
<td>40.5%</td>
<td>54.6%</td>
<td>58.0%</td>
<td>59.4%</td>
</tr>
</tbody>
</table>

Motion cues are much more informative than visual appearance cues!
Two Stream CNNs

Results:

- Two-stream architecture significantly outperforms traditional CNNs and prior methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
<th>F1 Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Spatio-temporal HMAX network [11, 16]</td>
<td>-</td>
<td>22.8%</td>
</tr>
<tr>
<td>“Slow fusion” spatio-temporal ConvNet [14]</td>
<td>65.4%</td>
<td>-</td>
</tr>
<tr>
<td>Spatial stream ConvNet</td>
<td>73.0%</td>
<td>40.5%</td>
</tr>
<tr>
<td>Temporal stream ConvNet</td>
<td>83.7%</td>
<td>54.6%</td>
</tr>
<tr>
<td>Two-stream model (fusion by averaging)</td>
<td>86.9%</td>
<td>58.0%</td>
</tr>
<tr>
<td>Two-stream model (fusion by SVM)</td>
<td>88.0%</td>
<td>59.4%</td>
</tr>
</tbody>
</table>
Two Stream CNNs

Results:

- Two-stream architecture significantly outperforms traditional CNNs and prior methods.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>-</td>
<td>65.4%</td>
<td>73.0%</td>
<td>83.7%</td>
<td>86.9%</td>
<td>88.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Knowing how to fuse the outputs from both streams matters.
Two Stream CNNs

How to do the fusion?:
• Fusing at the end
Two Stream CNNs

How to do the fusion?:

• Fusing at the end

• Good: Predictions from both streams are completely independent from each other, which provides diversity.
How to do the fusion?:
• Fusing at the end

- Bad: If two streams worked together they might achieve better results than if they worked independently.
Two Stream CNNs

How to do the fusion?:
• Fusing in the middle

• Good: both streams can learn to complement each other
Two Stream CNNs

How to do the fusion?:

• Fusing in the middle

• Bad: The more informative stream may start dominating, which would make the other stream useless.
Two Stream CNNs

How to do the fusion?:

• Hybrid approach

• Good: Enhances diversity of the predictions, while also allowing both streams to work together.
Two Stream CNNs

How to do the fusion?:
- Hybrid approach
- Bad: More parameters
Two Stream CNNs

How to do the fusion?:

<table>
<thead>
<tr>
<th>Fusion Method</th>
<th>Fusion Layer</th>
<th>Acc.</th>
<th>#layers</th>
<th>#parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum [22]</td>
<td>Softmax</td>
<td>85.6%</td>
<td>16</td>
<td>181.42M</td>
</tr>
<tr>
<td>Sum (ours)</td>
<td>Softmax</td>
<td>85.94%</td>
<td>16</td>
<td>181.42M</td>
</tr>
<tr>
<td>Max</td>
<td>ReLU5</td>
<td>82.70%</td>
<td>13</td>
<td>97.31M</td>
</tr>
<tr>
<td>Concatenation</td>
<td>ReLU5</td>
<td>83.53%</td>
<td>13</td>
<td>172.81M</td>
</tr>
<tr>
<td>Bilinear [15]</td>
<td>ReLU5</td>
<td>85.05%</td>
<td>10</td>
<td>6.61M+SVM</td>
</tr>
<tr>
<td>Sum</td>
<td>ReLU5</td>
<td>85.20%</td>
<td>13</td>
<td>97.31M</td>
</tr>
<tr>
<td>Conv</td>
<td>ReLU5</td>
<td>85.96%</td>
<td>14</td>
<td>97.58M</td>
</tr>
</tbody>
</table>

Fusion in the middle works almost the same as the fusion in the end, but it requires significantly less parameters!
Two Stream CNNs

How to do the fusion?:

<table>
<thead>
<tr>
<th>Fusion Layers</th>
<th>Accuracy</th>
<th>#layers</th>
<th>#parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReLU2</td>
<td>82.25%</td>
<td>11</td>
<td>91.90M</td>
</tr>
<tr>
<td>ReLU3</td>
<td>83.43%</td>
<td>12</td>
<td>93.08M</td>
</tr>
<tr>
<td>ReLU4</td>
<td>82.55%</td>
<td>13</td>
<td>95.48M</td>
</tr>
<tr>
<td>ReLU5</td>
<td>85.96%</td>
<td>14</td>
<td>97.57M</td>
</tr>
<tr>
<td>ReLU5 + FC8</td>
<td>86.04%</td>
<td>17</td>
<td>181.68M</td>
</tr>
<tr>
<td>ReLU3 + ReLU5 + FC6</td>
<td>81.55%</td>
<td>17</td>
<td>190.06M</td>
</tr>
</tbody>
</table>

Hybrid approach works best, but requires almost twice as many parameters.
Two Stream CNNs

How to do the fusion?:

<table>
<thead>
<tr>
<th>Fusion Layers</th>
<th>Accuracy</th>
<th>#layers</th>
<th>#parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReLU2</td>
<td>82.25%</td>
<td>11</td>
<td>91.90M</td>
</tr>
<tr>
<td>ReLU3</td>
<td>83.43%</td>
<td>12</td>
<td>93.08M</td>
</tr>
<tr>
<td>ReLU4</td>
<td>82.55%</td>
<td>13</td>
<td>95.48M</td>
</tr>
<tr>
<td><strong>ReLU5</strong></td>
<td>85.96%</td>
<td>14</td>
<td><strong>97.57M</strong></td>
</tr>
<tr>
<td>ReLU5 + FC8</td>
<td>86.04%</td>
<td>17</td>
<td>181.68M</td>
</tr>
<tr>
<td>ReLU3 + ReLU5 + FC6</td>
<td>81.55%</td>
<td>17</td>
<td>190.06M</td>
</tr>
</tbody>
</table>

Given the tradeoff between accuracy and complexity, the clear winner is the mid-fusion technique (i.e. fusing at ReLU 5)
Two Stream CNNs

**Issues with two stream CNNs:**

- Many more parameters than the LSTM or standard CNNs
- Requires pre-computing optical flow ahead of time, which is annoying and can be slow.
- Why not learn flow features the same way as we learn visual appearance features?
2D convolution:

\[ h = g \ast f \]

- \( g \) - convolutional weights of size MxN
- \( f \) - the values in a 2D grid that we want to convolve

\[ h_{ij} = \sum_{m=0}^{M} \sum_{n=0}^{N} g_{mn} f(i-m)(j-n) \]

A sliding window operation across the entire grid \( f \).
### 3D Convolutional Networks

#### 2D convolution:

\[ f = \begin{bmatrix}
 1 & 2 & 3 & 4 \\
 5 & 6 & 7 & 8 \\
 9 & 10 & 11 & 12 \\
 13 & 14 & 15 & 16 \\
\end{bmatrix} \quad \quad g = \begin{bmatrix}
 1 & 2 & 1 \\
 0 & 0 & 0 \\
 1 & 2 & 1 \\
\end{bmatrix} \]

\[ h = g \ast f = \begin{bmatrix}
 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 0 & 0 & 0 & 0 \\
 0 & 0 & 1 & 2 & 3 & 4 \\
 0 & 0 & 5 & 6 & 7 & 8 \\
 0 & 0 & 9 & 10 & 11 & 12 \\
 0 & 0 & 13 & 14 & 15 & 16 \\
\end{bmatrix} \]
2D convolution: 

\[ f = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \quad g = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \]

\[ h = g \ast f = \begin{bmatrix} 1 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 2 & 3 & 4 \\ 0 & 0 & 5 & 6 & 7 & 8 \\ 0 & 0 & 9 & 10 & 11 & 12 \\ 0 & 0 & 13 & 14 & 15 & 16 \end{bmatrix} = \begin{bmatrix} 1 \end{bmatrix} \]
3D Convolutional Networks

2D convolution:

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

\[ g = \begin{array}{c}
1 \\
0 \\
1 \\
\end{array} \]

\[ h = g \ast f = \]

\[ \begin{array}{cccc}
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 \\
\end{array} \]

\[ \begin{array}{cccc}
1 & 4 \\
\end{array} \]
3D Convolutional Networks

2D convolution:

\[ f = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \quad g = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \]

\[ h = g \ast f = \begin{bmatrix} 0 & 0 & 1 & 2 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 1 & 3 & 4 \\ 0 & 0 & 5 & 6 & 7 & 8 & 0 \\ 0 & 0 & 9 & 10 & 11 & 12 & 0 \\ 0 & 0 & 13 & 14 & 15 & 16 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 8 \end{bmatrix} \]
3D Convolutional Networks

2D convolution:

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

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

\[ h = g \ast f = \]

\[ \begin{bmatrix}
0 & 0 & 0 & 1 & 2 & 3 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 2 & 3 & 4 & 4 \\
0 & 0 & 5 & 6 & 7 & 8 & 8 \\
0 & 0 & 9 & 10 & 11 & 12 & 12 \\
0 & 0 & 13 & 14 & 15 & 16 & 16 \\
\end{bmatrix} \]
3D Convolutional Networks

2D convolution:

\[ f = \begin{bmatrix}
    1 & 2 & 3 & 4 \\
    5 & 6 & 7 & 8 \\
    9 & 10 & 11 & 12 \\
    13 & 14 & 15 & 16 \\
\end{bmatrix} \quad g = \begin{bmatrix}
    1 & 2 & 1 \\
    0 & 0 & 0 \\
    1 & 2 & 1 \\
\end{bmatrix} \]

\[ h = g \ast f = \begin{bmatrix}
    0 & 0 & 0 & 0 & 0 & 0 \\
    0 & 0 & 1 & 2 & 3 & 4 \\
    0 & 0 & 5 & 6 & 7 & 8 \\
    0 & 0 & 9 & 10 & 11 & 12 \\
    0 & 0 & 13 & 14 & 15 & 16 \\
\end{bmatrix} = \begin{bmatrix}
    1 & 4 & 8 & 12 \\
    5 \\
\end{bmatrix} \]
3D Convolutional Networks

2D convolution:

\[ f = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix}, \quad g = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \]

\[ h = g \ast f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 2 & 3 & 4 \\ 0 & 1 & 0 & 2 & 5 & 1 & 6 & 7 & 8 \\ 0 & 0 & 0 & 9 & 10 & 11 & 12 \\ 0 & 0 & 0 & 13 & 14 & 15 & 16 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 8 & 12 \\ 5 & 16 \end{bmatrix} \]
3D Convolutional Networks

2D convolution:

\[ f = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \quad g = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \]

\[ h = g \ast f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 4 \\ 0 & 0 & 1 & 2 & 6 & 7 \\ 0 & 0 & 9 & 10 & 11 & 12 \\ 0 & 0 & 13 & 14 & 15 & 16 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 8 & 12 \\ 5 & 16 & 24 \end{bmatrix} \]
3D Convolutional Networks

2D convolution:

\[ f = \begin{bmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16
\end{bmatrix} \quad g = \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{bmatrix} \]

\[ h = g \ast f = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 2 & 3 & 4 \\
0 & 0 & 5 & 6 & 7 & 8 \\
0 & 0 & 9 & 10 & 11 & 12 \\
0 & 0 & 13 & 14 & 15 & 16
\end{bmatrix} = \begin{bmatrix}
1 & 4 & 8 & 12 \\
5 & 16 & 24 & 28
\end{bmatrix} \]
3D Convolutional Networks

2D convolution:

\[ f = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \quad \quad g = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \]

\[ h = g \ast f = \]

\[ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 2 & 1 & 2 & 3 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 5 & 6 & 7 & 8 \\ 1 & 2 & 1 & 9 & 10 & 11 & 12 \\ 0 & 0 & 13 & 14 & 15 & 16 \end{bmatrix} = \]

\[ \begin{bmatrix} 1 & 4 & 8 & 12 \\ 5 & 16 & 24 & 28 \\ 10 \end{bmatrix} \]
### 3D Convolutional Networks

#### 2D convolution:

\[
f = \begin{bmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16 \\
\end{bmatrix}, \quad g = \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
1 & 2 & 1 \\
\end{bmatrix}
\]

\[
h = g \ast f = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 2 & 1 & 2 & 3 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 5 & 6 & 7 & 8 \\
0 & 0 & 9 & 10 & 11 & 12 \\
0 & 0 & 13 & 14 & 15 & 16 \\
\end{bmatrix}
\]
3D Convolutional Networks

2D convolution:

\[ f = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \quad g = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \]

\[ h = g \ast f = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 2 & 3 & 4 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 5 & 6 & 7 & 8 \\ 0 & 0 & 9 & 10 & 11 & 12 \\ 0 & 0 & 13 & 14 & 15 & 16 \end{bmatrix} = \begin{bmatrix} 1 & 4 & 8 & 12 \\ 5 & 16 & 24 & 28 \\ 10 & 32 & 48 \end{bmatrix} \]
3D Convolutional Networks

2D convolution:

\[
\begin{align*}
f &= \begin{pmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16
\end{pmatrix} \\
g &= \begin{pmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{pmatrix}
\end{align*}
\]

\[
h = g \ast f = \begin{pmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 12 & 2 & 14 \\
0 & 0 & 5 & 6 & 7 & 8 \\
0 & 0 & 9 & 10 & 11 & 12 \\
0 & 0 & 13 & 14 & 15 & 16
\end{pmatrix}
\]
# 3D Convolutional Networks

## 2D convolution:

$$f = \begin{bmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16
\end{bmatrix} \quad g = \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{bmatrix}
$$

$$h = g \ast f = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 2 & 3 & 4 \\
0 & 0 & 5 & 6 & 7 & 8 \\
0 & 0 & 9 & 10 & 11 & 12 \\
0 & 0 & 13 & 14 & 15 & 16
\end{bmatrix} = \begin{bmatrix}
1 & 4 & 8 & 12 \\
5 & 16 & 24 & 28 \\
10 & 32 & 48 & 56 \\
18
\end{bmatrix}$$
2D convolution:

\[
\begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16
\end{array}
\quad
\begin{array}{c}
1 & 2 & 1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{array}
\]

\[
h = g \ast f =
\]

\[
\begin{array}{cccccccc}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 2 & 3 & 4 & 0 & 0 \\
0 & 0 & 5 & 6 & 7 & 8 & 0 & 0 \\
0 & 0 & 9 & 10 & 11 & 12 & 0 & 0 \\
0 & 0 & 13 & 14 & 15 & 16 & 0 & 0
\end{array}
\quad
\begin{array}{cccc}
1 & 4 & 8 & 12 \\
5 & 16 & 24 & 28 \\
10 & 32 & 48 & 56 \\
18 & 56 & 0 & 0
\end{array}
\]
3D Convolutional Networks

2D convolution:

\[ f = \begin{bmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16 \\
\end{bmatrix} \quad g = \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
1 & 2 & 1 \\
\end{bmatrix} \]

\[ h = g \ast f = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 2 & 3 & 4 & 0 & 0 & 5 & 6 & 7 & 8 & 0 & 0 & 0 \\
0 & 0 & 9 & 10 & 11 & 12 & 0 & 0 & 13 & 14 & 15 & 16 & 0 & 0 & 0 \\
\end{bmatrix} = \begin{bmatrix}
1 & 4 & 8 & 12 \\
5 & 16 & 24 & 28 \\
10 & 32 & 48 & 56 \\
18 & 56 & 80 & \end{bmatrix} \]
3D Convolutional Networks

**2D convolution:**

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

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

\[
h = g \ast f = \begin{bmatrix}
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 2 & 3 & 4 \\
0 & 0 & 5 & 16 & 27 & 38 \\
0 & 0 & 9 & 10 & 11 & 12 \\
0 & 0 & 13 & 14 & 15 & 16
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 4 & 8 & 12 \\
5 & 16 & 24 & 28 \\
10 & 32 & 48 & 56 \\
18 & 56 & 80 & 88
\end{bmatrix}
\]
3D Convolutional Networks

3D convolution:

$h = g \ast f$

- convolutional weights of size $M \times N \times K$
- the values in a 3D grid that we want to convolve

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

A sliding volume operation across the entire grid $f$ in 3D.
3D Convolutional Networks

3D convolution:

\[ f = \begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16 \\
\end{array} \quad g = \begin{array}{ccc}
1 & 2 & 1 \\
0 & 0 & 0 \\
1 & 2 & 1 \\
\end{array} \]
3D Convolutional Networks

3D convolution:

\[ f = \begin{bmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16
\end{bmatrix} \quad g = \begin{bmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{bmatrix} \]
3D Convolutional Networks

3D convolution:

\[ f = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{bmatrix} \quad \text{and} \quad g = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \]
3D Convolutional Networks

3D convolution:

\[ f = \begin{array}{cccc}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16 \\
\end{array} \quad g = \begin{array}{ccc}
1 & 2 & 1 \\
0 & 0 & 0 \\
1 & 2 & 1 \\
\end{array} \]
3D Convolutional Networks

3D convolution:

\[ f = \begin{pmatrix}
1 & 2 & 3 & 4 \\
5 & 6 & 7 & 8 \\
9 & 10 & 11 & 12 \\
13 & 14 & 15 & 16
\end{pmatrix} \quad \text{and} \quad g = \begin{pmatrix}
1 & 2 & 1 \\
0 & 0 & 0 \\
1 & 2 & 1
\end{pmatrix} \]
3D Convolutional Networks

2D vs 3D Convolution:

- Applying 2D convolution (on a single frame, or on multiple frames) results in an image.
- Applying 3D convolution yields a 3D volume that preserves temporal information.
3D Convolutional Networks

2D vs 3D Convolution:

• Applying 2D convolution (on a single frame, or on multiple frames) results in an image.

• Applying 3D convolution yields a 3D volume that preserves temporal information.

• Backpropagation is still applied in a similar fashion, except that gradients have to be computed in a 3D volume, instead of a 2D window.
3D Convolutional Networks

**3D Convolution:**
- Learned 3D filters

Learns to recognize salient motion
# 3D Convolutional Networks

## 3D Convolution:

- Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imagenet + linear SVM</td>
<td>68.8</td>
</tr>
<tr>
<td>iDT w/ BoW + linear SVM</td>
<td>76.2</td>
</tr>
<tr>
<td>Deep networks [18]</td>
<td>65.4</td>
</tr>
<tr>
<td>Spatial stream network [36]</td>
<td>72.6</td>
</tr>
<tr>
<td>LRCN [6]</td>
<td>71.1</td>
</tr>
<tr>
<td>LSTM composite model [39]</td>
<td>75.8</td>
</tr>
<tr>
<td><strong>C3D</strong> (1 net) + linear SVM</td>
<td>82.3</td>
</tr>
<tr>
<td><strong>C3D</strong> (3 nets) + linear SVM</td>
<td>85.2</td>
</tr>
</tbody>
</table>
3D Convolutional Networks

3D Convolution:

- Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imagenet + linear SVM</td>
<td>68.8</td>
</tr>
<tr>
<td>iDT w/ BoW + linear SVM</td>
<td>76.2</td>
</tr>
<tr>
<td>Deep networks [18]</td>
<td>65.4</td>
</tr>
<tr>
<td>Spatial stream network [36]</td>
<td>72.6</td>
</tr>
<tr>
<td>LRCN [6]</td>
<td>71.1</td>
</tr>
<tr>
<td>LSTM composite model [39]</td>
<td>75.8</td>
</tr>
<tr>
<td><strong>C3D (1 net) + linear SVM</strong></td>
<td><strong>82.3</strong></td>
</tr>
<tr>
<td><strong>C3D (3 nets) + linear SVM</strong></td>
<td><strong>85.2</strong></td>
</tr>
</tbody>
</table>

Significantly outperforms standard CNNs
3D Convolutional Networks

3D Convolution:

- Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imagenet + linear SVM</td>
<td>68.8</td>
</tr>
<tr>
<td>iDT w/ BoW + linear SVM</td>
<td>76.2</td>
</tr>
<tr>
<td>Deep networks [18]</td>
<td>65.4</td>
</tr>
<tr>
<td>Spatial stream network [36]</td>
<td>72.6</td>
</tr>
<tr>
<td>LRCN [6]</td>
<td>71.1</td>
</tr>
<tr>
<td>LSTM composite model [39]</td>
<td>75.8</td>
</tr>
<tr>
<td><strong>C3D (1 net) + linear SVM</strong></td>
<td>82.3</td>
</tr>
<tr>
<td><strong>C3D (3 nets) + linear SVM</strong></td>
<td>85.2</td>
</tr>
</tbody>
</table>

Significantly outperforms handcrafted features
3D Convolutional Networks

3D Convolution:
• Results

<table>
<thead>
<tr>
<th>Method</th>
<th>Accuracy (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Imagenet + linear SVM</td>
<td>68.8</td>
</tr>
<tr>
<td>iDT w/ BoW + linear SVM</td>
<td>76.2</td>
</tr>
<tr>
<td>Deep networks [18]</td>
<td>65.4</td>
</tr>
<tr>
<td>Spatial stream network [36]</td>
<td>72.6</td>
</tr>
<tr>
<td><strong>LRCN [6]</strong></td>
<td>71.1</td>
</tr>
<tr>
<td>LSTM composite model [39]</td>
<td>75.8</td>
</tr>
<tr>
<td><strong>C3D (1 net) + linear SVM</strong></td>
<td><strong>82.3</strong></td>
</tr>
<tr>
<td><strong>C3D (3 nets) + linear SVM</strong></td>
<td><strong>85.2</strong></td>
</tr>
</tbody>
</table>

Significantly outperforms LSTM
3D Convolutional Networks

3D Convolution:
• Learned Visual Representation
3D Convolutional Networks

3D Convolution:
- Learned Visual Representation

Learned feature representation in 3D CNNs separates different action classes much better than the features in 2D CNNs
3D Convolutional Networks

3D Convolution:

• Qualitative Results
Conclusion

• Standard CNNs are not very well suited for temporal tasks because they have no memory and because everything is processed in a 2D fashion.
Conclusion

• Standard CNNs are not very well suited for temporal tasks because they have no memory and because everything is processed in a 2D fashion.

• LSTMs can remember events from the past but cannot learn motion features very well.
Conclusion

• Standard CNNs are not very well suited for temporal tasks because they have no memory and because everything is processed in a 2D fashion.

• LSTMs can remember events from the past but cannot learn motion features very well

• Two stream networks incorporate both motion and visual appearance cues, but optical flow has to be extracted manually
Conclusion

- Standard CNNs are not very well suited for temporal tasks because they have no memory and because everything is processed in a 2D fashion.

- LSTMs can remember events from the past but cannot learn motion features very well

- Two stream networks incorporate both motion and visual appearance cues, but optical flow has to be extracted manually

- 3D convolutional networks can learn motion features automatically using 3D convolution.