Two-Stream Convolutional Networks for Action Recognition in Videos

Karen Simonyan
Andrew Zisserman

Cemil Zalluhoğlu
Introduction

• Aim
  • Extend deep Convolution Networks to action recognition in video.

• Motivation
  • Deep Convolutional Networks (ConvNets) work very well for image recognition
  • It is less clear what is the right deep architecture for video recognition

• Main Contribution
  • Two separate recognition stream
    • Spatial stream – appearance recognition ConvNet
    • Temporal stream – motion recognition ConvNet
    • Both streams are implemented as ConvNets
Introduction

- Proposed architecture is related to the two-streams hypothesis
  - the human visual cortex contains two pathways:
    - The ventral stream (which performs object recognition)
    - The dorsal stream (which recognises motion)
Two-stream architecture for video recognition

• The spatial part, in the form of individual frame appearance, carries information about scenes and objects depicted in the video.

• The temporal part, in the form of motion across the frames, conveys the movement of the observer (the camera) and the objects.
Two-stream architecture for video recognition

Figure 1: Two-stream architecture for video classification.
Two-stream architecture for video recognition

- Each stream is implemented using a deep ConvNet, softmax scores of which are combined by late fusion.

- Two fusion methods:
  - averaging
  - training a multi-class linear SVM on stacked $L2$-normalised softmax scores as features.
The Spatial stream ConvNet

- Predicts action from still images - image classification
- Operates on individual video frames
- The static appearance by itself is a useful clue, due to some actions are strongly associated with particular objects

- Since a spatial ConvNet is essentially an image classification architecture,
- Build upon the recent advances in large-scale image recognition methods
- pre-train the network on a large image classification dataset, such as the ImageNet challenge dataset.
The Temporal stream ConvNet

- Optical flow
- Input of the ConvNet model is stacking optical flow displacement fields between several consecutive frames
- This input describes the motion between video frames

Figure 2: **Optical flow.** (a),(b): a pair of consecutive video frames with the area around a moving hand outlined with a cyan rectangle. (c): a close-up of dense optical flow in the outlined area; (d): horizontal component $d^x$ of the displacement vector field (higher intensity corresponds to positive values, lower intensity to negative values). (e): vertical component $d^y$. Note how (d) and (e) highlight the moving hand and bow. The input to a ConvNet contains multiple flows (Sect. 3.1).
ConvNet input configurations (1)

• Optical flow stacking

  A dense optical flow can be seen as a set of displacement vector fields

  ➢ $\mathbf{d}_t$: displacement vector fields between the pairs of consecutive frames $t$ and $t+1$

  ➢ $\mathbf{d}_t(u,v)$: denote the displacement vector at the point $(u,v)$ in frame $t$, which moves the point to the corresponding point in the following frame $t+1$.

  ➢ $d_t^x$, $d_t^y$: horizontal and vertical components of the vector field

• The input volume of ConvNet is $I_T \in \mathbb{R}^{w \times h \times 2L}$ $w$ and $h$ be the width and height of a video, $L$ is number of consecutive frames, $2L$ comes from ($d_t^x$ and $d_t^y$)

$$I_T(u,v,2k-1) = d_{t+k-1}^x(u,v),$$
$$I_T(u,v,2k) = d_{t+k-1}^y(u,v), \quad u = [1; w], v = [1; h], k = [1; L].$$
ConvNet input configurations (2)

• **Trajectory stacking**
  • Inspired by trajectory-based descriptors
  • replaces the optical flow, sampled at the same locations across several frames, with the flow, sampled along the motion trajectories

\[
I^x_\tau(u, v, 2k - 1) = d^x_{\tau+k-1}(p_k), \\
I^y_\tau(u, v, 2k) = d^y_{\tau+k-1}(p_k), \quad u = [1; w], v = [1; h], k = [1; L].
\]

where \( p_k \) is the \( k \)-th point along the trajectory, which starts at the location \((u, v)\) in the frame \( \tau \) and is defined by the following recurrence relation:

\[ p_1 = (u, v); \quad p_k = p_{k-1} + d_{\tau+k-2}(p_{k-1}), \quad k > 1. \]
ConvNet input configurations (3)

Figure 3: ConvNet input derivation from the multi-frame optical flow. Left: optical flow stacking (1) samples the displacement vectors $\mathbf{d}$ at the same location in multiple frames. Right: trajectory stacking (2) samples the vectors along the trajectory. The frames and the corresponding displacement vectors are shown with the same colour.

\[ I_{r}(u, v, 2k - 1) = d_{r+k-1}^{x}(u, v), \]
\[ I_{r}(u, v, 2k) = d_{r+k-1}^{y}(u, v), \quad u = [1; w], v = [1; h], k = [1; L]. \]

(2)

where $\mathbf{p}_{k}$ is the $k$-th point along the trajectory, which starts at the location $(u, v)$ in the frame $r$ and is defined by the following recurrence relation:

\[ \mathbf{p}_{1} = (u, v); \quad \mathbf{p}_{k} = \mathbf{p}_{k-1} + d_{r+k-2}(\mathbf{p}_{k-1}), \quad k > 1. \]
ConvNet input configurations (4)

• **Bi-directional optical flow**
  Construct an input volume $I\tau$ by stacking $L/2$ forward flows between frames $\tau$ and $\tau + L/2$ and $L/2$ backward flows between frames $\tau - L/2$ and $\tau$. The input $I\tau$ thus has the same number of channels $(2L)$ as before.

• **Mean flow subtraction**
  For camera motion compensation, from each displacement field $d$, subtract its mean vector.

• **Architecture**
  - ConvNet requires a fixed-size input, we sample a $224 \times 224 \times 2L$
  - The hidden layers configuration remains largely the same as that used in the spatial net
ConvNet input configurations (5)

• Visualisation of learnt convolutional filters

• Spatial derivatives capture how motion changes in space
• Temporal derivatives capture how motion changes in time
Multi-task learning

- The temporal ConvNet needs to be trained on video data unlike the spatial ConvNet.
- Training is performed on the UCF-101 and HMDB-51 datasets, which have only: 9.5K and 3.7K videos respectively.
- Each dataset is a separate task.
- ConvNet architecture is modified. It has two softmax classification layers on top of the last fully-connected layer:
  - One softmax layer computes HMDB-51 classification scores, the other one – the UCF-101 scores.
  - Each of the layers is equipped with its own loss function, which operates only on the videos, coming from the respective dataset.
- The overall training loss is computed as the sum of the individual tasks’ losses, and the network weight derivatives can be found by back-propagation.
Implementation details

- **ConvNets configuration**
  - CNN-M-2048 architecture is similar to Zeiler and Fergus network.
  - All hidden weight layers use the rectification (ReLU) activation function;
  - Maxpooling is performed over $3 \times 3$ spatial windows with stride 2.
  - CNN architecture by using 5 convolution layers and 3 fully connected layers.

- The only difference between spatial and temporal ConvNet configurations: the second normalisation layer of temporal ConvNet is removed to reduce memory consumption.
Implementation details (2)

• **Training**
  - Spatial net training; 224 × 224 sub-image is randomly cropped from the selected frame.
  - Temporal net training; optical flow is computed, a fixed-size 224 × 224 × 2L input is randomly cropped and flipped.
  - The learning rate is initially set to 10⁻².
  - Namely, when training a ConvNet from scratch, the rate is changed to 10⁻³ after 50K iterations, then to 10⁻⁴ after 70K iterations, and training is stopped after 80K iterations.
  - In the fine-tuning scenario, the rate is changed to 10⁻³ after 14K iterations, and training stopped after 20K iterations.

• **Multi-GPU training**
  - Training a single temporal ConvNet takes 1 day on a system with 4 NVIDIA Titan cards, which constitutes a 3.2 times speed-up over single-GPU training.

• **Optical Flow**
  - Pre-computed the flow before training.
Evaluation (1)

• **Datasets and evaluation protocol**
  • UCF-101 contains 13K videos (180 frames/video on average), annotated into 101 action classes;
  • HMDB-51 includes 6.8K videos of 51 actions

• The evaluation protocol is the same for both datasets:
  • the organisers provide three splits into training and test data
  • the performance is measured by the mean classification accuracy across the splits.

  • Each UCF-101 split contains 9.5K training videos; an HMDB-51 split contains 3.7K training videos.
  • We begin by comparing different architectures on the first split of the UCF-101 dataset.
  • For comparison with the state of the art, we follow the standard evaluation protocol and report the average accuracy over three splits on both UCF-101 and HMDB-51.
Evaluation (2)

- Spatial ConvNets:

<table>
<thead>
<tr>
<th>Training setting</th>
<th>Dropout ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.5</td>
</tr>
<tr>
<td>From scratch</td>
<td>42.5%</td>
</tr>
<tr>
<td>Pre-trained + fine-tuning</td>
<td>70.8%</td>
</tr>
<tr>
<td>Pre-trained + last layer</td>
<td><strong>72.7%</strong></td>
</tr>
</tbody>
</table>
Evaluation (3)

• Temporal ConvNets:

<table>
<thead>
<tr>
<th>Input configuration</th>
<th>Mean subtraction</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>off</td>
</tr>
<tr>
<td>Single-frame optical flow ($L = 1$)</td>
<td>-</td>
</tr>
<tr>
<td>Optical flow stacking (1) ($L = 5$)</td>
<td>-</td>
</tr>
<tr>
<td>Optical flow stacking (1) ($L = 10$)</td>
<td>79.9%</td>
</tr>
<tr>
<td>Trajectory stacking (2) ($L = 10$)</td>
<td>79.6%</td>
</tr>
<tr>
<td>Optical flow stacking (1) ($L = 10$), bi-dir.</td>
<td>-</td>
</tr>
</tbody>
</table>
Evaluation (4)

• Multi-task learning of temporal ConvNets

Table 2: Temporal ConvNet accuracy on HMDB-51 (split 1 with additional training data).

<table>
<thead>
<tr>
<th>Training setting</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Training on HMDB-51 without additional data</td>
<td>46.6%</td>
</tr>
<tr>
<td>Fine-tuning a ConvNet, pre-trained on UCF-101</td>
<td>49.0%</td>
</tr>
<tr>
<td>Training on HMDB-51 with classes added from UCF-101</td>
<td>52.8%</td>
</tr>
<tr>
<td>Multi-task learning on HMDB-51 and UCF-101</td>
<td>55.4%</td>
</tr>
</tbody>
</table>
Evaluation (5)

- Two-stream ConvNets

<table>
<thead>
<tr>
<th>Spatial ConvNet</th>
<th>Temporal ConvNet</th>
<th>Fusion Method</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pre-trained + last layer</td>
<td>bi-directional</td>
<td>averaging</td>
<td>85.6%</td>
</tr>
<tr>
<td>Pre-trained + last layer</td>
<td>uni-directional</td>
<td>averaging</td>
<td>85.9%</td>
</tr>
<tr>
<td>Pre-trained + last layer</td>
<td>uni-directional, multi-task</td>
<td>averaging</td>
<td>86.2%</td>
</tr>
<tr>
<td>Pre-trained + last layer</td>
<td>uni-directional, multi-task</td>
<td>SVM</td>
<td>87.0%</td>
</tr>
</tbody>
</table>

Table 3: Two-stream ConvNet accuracy on UCF-101 (split 1).
Evaluation (6)

- Multi-task learning of temporal ConvNets

Table 4: **Mean accuracy (over three splits)** on UCF-101 and HMDB-51.

<table>
<thead>
<tr>
<th>Method</th>
<th>UCF-101</th>
<th>HMDB-51</th>
</tr>
</thead>
<tbody>
<tr>
<td>Improved dense trajectories (IDT) [26, 27]</td>
<td>85.9%</td>
<td>57.2%</td>
</tr>
<tr>
<td>IDT with higher-dimensional encodings [20]</td>
<td>87.9%</td>
<td>61.1%</td>
</tr>
<tr>
<td>IDT with stacked Fisher encoding [21] (based on Deep Fisher Net [23])</td>
<td>-</td>
<td>66.8%</td>
</tr>
<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><strong>88.0%</strong></td>
<td><strong>59.4%</strong></td>
</tr>
</tbody>
</table>
Conclusions

• Temporal stream performs very well
• Two stream deep ConvNet idea
• Temporal and Spatial streams are complementary
  • Two-stream architecture outperforms a single-stream one