**Introduction**

- **Goal:** Design new features sharing merits of both hand-crafted and deep-learned features for video representation.
- **Existing works:**
  - Improved trajectories [1]: (i) Extracting trajectories. (ii) Pooling local features along trajectories (HOG, HOF, MBH).
  - Two-stream ConvNets [2]: (i) Stacking frames or optical flow fields. (ii) Learning features for classification with CNNs.

**Our idea:** Trajectory-Pooled Deep-Convolutional Descriptors (TDD):
- (i) We exploit deep architectures to learn discriminative convolutional feature maps.
- (ii) We perform trajectory-constrained pooling to aggregate these convolutional feature maps into effective descriptors.

**Advantages:**
- TDDs are automatically learned and contain high discriminative capacity compared with those hand-crafted features.
- TDDs take account of the intrinsic characteristics of temporal dimension and introduce the strategies of trajectory-constrained sampling and pooling.

**Improved Trajectories Revisited**

- **Improved trajectories:**
  - Densely sampling a set of points and tracking them by media filtering: $P = \{(x_i, y_i, t_{i+1}) = (x_i, y_i) + (x_{i+1} - x_i)\}\forall i$.
  - Camera motion estimation: determining a homography matrix by using SURF feature matching and optical flow matching.
  - Camera motion estimation is capable of rectifying the optical flow fields and removing the trajectories of background.

- **IDTs for TDDs**
  - Given a video $V$, we obtain a set of trajectories: $T(V) = \{T_1, T_2, \ldots, T_k\}$.
  - $T_k$ denotes the $k$th trajectory in the original spatial scale: $T_k = \{(x_{k,1}, y_{k,1}), (x_{k,2}, y_{k,2}), \ldots, (x_{k,P}, y_{k,P})\}$, where $(x, y)$ is the pixel position, and $P$ is the length of trajectory ($P = 15$).

**Deep Convolutional Descriptors**

- **Convolutions:**
  - We choose the two-stream ConvNets, which is composed of spatial nets and temporal nets.
  - Spatial nets capture static appearance cues and are trained on single frame images ($224 \times 224 \times 3$).
  - Temporal nets describe the dynamic motion information and are trained on the stacking of optical flow fields ($224 \times 224 \times 20$).

- **Convolutional feature maps:**
  - We use two-stream ConvNets as generic feature extractors: $C(V) = \{C_{w,1}, C_{w,2}, \ldots, C_{w,1}, C_{w,2}, \ldots, C_{w,1}\}$, where $C_{w,1} \in \mathbb{R}^{M \times (W \times H \times L \times C)}$ is the $w$th feature map.
  - We conduct zero padding of the layer’s input with size $[k/2]$ before each convolutional or pooling layer, with kernel size $k$.
  - A point with video coordinates $(x_p, y_p, t)$ will be centered on $(r \times x_p, r \times y_p, t)$ in convolutional maps, where $r$ is map size ratio.

- **Trajectory-pooled descriptors:**
  - Two normalization methods:
    - Spatial-temporal normalization: $C_{s}(x, y, z, a) = C(x, y, z, a) / \max_{x,y,z,a} C_{s}$.
    - Channel normalization: $C_{c}(x, y, z, a) = C(x, y, z, a) / \max_{x,y,z,a} C_{c}$.
  - Sum pooling along trajectory: $D(T_k, C_k) = \sum_{i \in I} C_k(x_{i}, y_{i}, t_{i}, z_{i}, a_{i})$.

  - **Multi-scale TDD extension:** We construct multi-scale pyramid representations of video frames and optical flow fields, which are transformed into multi-scale convolutional feature maps by ConvNets.

**Experimental Results**

- **Comparison of TDD with iDT features [1] and two-stream ConvNets [2].**

- **Temporal nets**
  - Two normalization methods:
    - Trajectory-pooled descriptors
      - Deep Convolutional Descriptors

- **Spatial nets**
  - Two normalization methods:
    - Trajectory-pooled descriptors
      - Deep Convolutional Descriptors

- **Performance Comparison of different layers of spatial nets and temporal nets.**

**References**