Segmental Spatiotemporal CNNs for Fine-Grained Action Segmentation

Colin Lea(✉), Austin Reiter, René Vidal, and Gregory D. Hager

Johns Hopkins University, Baltimore, USA
clea1@jhu.edu, {areiter,hager}@cs.jhu.edu, rvidal@cis.jhu.edu

Abstract. Joint segmentation and classification of fine-grained actions is important for applications of human-robot interaction, video surveillance, and human skill evaluation. However, despite substantial recent progress in large-scale action classification, the performance of state-of-the-art fine-grained action recognition approaches remains low. We propose a model for action segmentation which combines low-level spatiotemporal features with a high-level segmental classifier. Our spatiotemporal CNN is comprised of a spatial component that represents relationships between objects and a temporal component that uses large 1D convolutional filters to capture how object relationships change across time. These features are used in tandem with a semi-Markov model that captures transitions from one action to another. We introduce an efficient constrained segmental inference algorithm for this model that is orders of magnitude faster than the current approach. We highlight the effectiveness of our Segmental Spatiotemporal CNN on cooking and surgical action datasets for which we observe substantially improved performance relative to recent baseline methods.

1 Introduction

New spatiotemporal feature representations [1,2] and massive datasets like ActivityNet [3] has catalyzed progress towards large-scale action recognition in recent years. In the large-scale case, the goal is to classify diverse actions like skiing and basketball, so it is often advantageous to capture contextual cues like the background appearance. In contrast, despite active development on fine-grained action recognition (e.g. [4–8]) progress has been comparatively modest. Many of these models do not capture the nuances necessary for recognizing fine-grained actions such as subtle changes in object location.

In this paper we provide a methodology built around the idea of modeling object states, their relationships, and how they change over time. Our goal is to temporally segment a video and to classify each of its constituent actions. We target goal-driven activities performed in a situated environment, like a kitchen, where a static camera captures a user who performs dozens of actions. For concreteness, refer to the sub-sequence depicted in Fig. 1: A user places a tomato onto a cutting board, cuts it with a knife, and places it into a salad bowl. This is part of a much longer salad preparation sequence. There are many

© Springer International Publishing AG 2016
B. Leibe et al. (Eds.): ECCV 2016, Part III, LNCS 9907, pp. 36–52, 2016.
DOI: 10.1007/978-3-319-46487-9_3

applications of this task including in industrial manufacturing [9,10], surgical training [11–13], and general human activity analysis (e.g. cooking, sports) [4,6, 14–17].

We introduce a Spatiotemporal CNN (ST-CNN) which encodes low-level visual information and a semi-Markov model that captures high-level temporal information. The spatial component of the ST-CNN is a variation on VGG [18] designed for fine-grained tasks which encodes object state, location, and inter-object relationships. Our network is smaller than models like VGG [18] and AlexNet [19] and induces more spatial invariance. This model diverges from recent fine-grained models which typically use holistic approaches to model the scene.

The temporal component of the ST-CNN captures how object relationships change over the course of an action. In the tomato cutting example the `cut` action changes the tomato's state from *whole* to *diced* and the `place` action requires moving the tomato from location *cutting board* to *bowl*. Each action is represented as a linear combination of shared temporal convolutional filters. The probability of an action at any given time is computed using 1D convolutions over the spatial activations. These filters are on the order of 10 s long and explicitly capture mid-range motion patterns.

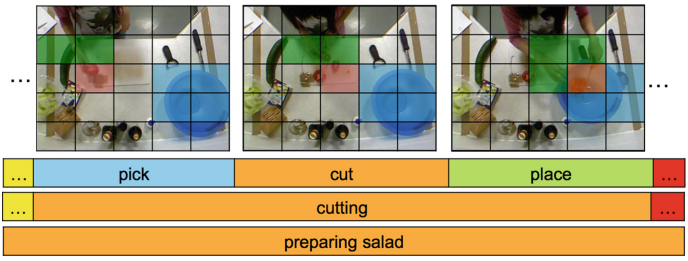

Fig. 1. Our model encodes object relationships and how these relationships change temporally. (top) Latent *hand* and *tomato* regions are highlighted in different colors on images from the 50 Salads dataset. (bottom) We evaluate on multiple label granularities that model fine-grained or coarse-grained actions. (Color figure online)

The segmental component jointly segments and classifies actions using a Semi-Markov Conditional Random Field [20] that encodes pairwise transitions between action segments. This model offers two benefits over traditional time series models like linear chain Conditional Random Fields (CRFs) and Recurrent Neural Networks. Features are computed segment-wise, as opposed to per-frame, and we condition the action at each segment on the previous segment instead of the previous frame. Traditionally these models have higher computational complexity than their frame-wise alternatives. In this work we introduce a new constrained inference algorithm that is one to three order of magnitude faster than the common inference technique.

Despite a large number of action recognition datasets in the computer vision community, few are sufficient for modeling fine-grained segmentation and classification. We apply our approach to two datasets: University of Dundee 50 Salads [21], which is in the cooking domain, and the JHU-ISI Surgical Assessment Working Set (JIGSAWS) [11], which is in the surgical robotics domain. Both of these datasets have reasonable amounts of data, interesting task granularity, and realistic task variability. On these datasets, our model substantially outperforms popular methods such as Dense Trajectories, spatial Convolutional Neural Networks, and LSTM-based Recurrent Neural Networks.

In summary, our contributions are:

- We develop a Spatiotemporal CNN that captures object relationships and how relationships change over time.
- We introduce an efficient algorithm for segmental inference that is one to three orders of magnitude faster than the common approach.
- We substantially outperform recent methods for fine-grained recognition on two challenging datasets.

2 Related Work

Holistic Features: Holistic methods using spatiotemporal features with a bag of words representation are standard for large-scale [1,22–24] and fine-grained [4–6,23,25] action analysis. The typical baseline represents a given clip using Improved Dense Trajectories (IDT) [1] with a histogram of dictionary elements [4] or a Fisher Vector encoding [1]. Dense Trajectories concatenate HOG, HOF, and MBH texture descriptors along optical flow trajectories to characterize small spatiotemporal patches. Empirically they perform well on large-scale tasks, in part because of their ability to capture background detail (e.g. sport arena versus mountaintop). However, for fine-grained tasks the image background is often constant so it is more important to model objects and their relationships. These are typically not modeled in holistic approaches. Furthermore, the typical image patch size for IDT (neighborhood = 32px, cell size = 2px) is too small to extract high-level object information.

Large-Scale Action Classification: While recent work has extended CNN models to video [2,24,26–30], often results are only superior when concatenated with IDT features [24,28,29]. These models improve over holistic methods by encoding spatial and temporal relationships within an image. Several papers (e.g. [2,26,30]) have proposed models to fuse spatial and temporal techniques. While each achieve state of the art, their models are only marginally better than IDT baselines. Our approach is similar in that we propose a spatiotemporal CNN, but our temporal filters are applied in 1D and are much longer in duration.

From Large-Scale Detection to Fine-Grained Segmentation: Despite success in classification, large-scale approaches are inadequate for tasks like

action localization and detection which are more similar to fine-grained segmentation. In the 2015 THUMOS large-scale action recognition challenge[1], the top team fused IDT and CNN approaches to achieve 70 % mAP on classification. However, the top method only achieves 18 % (overlap ≥0.5) for localization. Heilbron *et al.* [3] found similar results on ActivityNet with 11.9 % (overlap ≥0.2). This suggests that important methodological changes are necessary for identifying and localizing actions regardless of fine-grained or large-scale.

Moving to fine-grained recognition, recent work has combined holistic methods with human pose or object detection. On MPII Cooking Rohrbach *et al.* [4] combine IDT with pose features to get a detection score of 34.5 % compared to 29.5 % without pose features. Cheron *et al.* [7] show that if temporal segmentation on MPII is known then CNN-based pose features achieve 71.4 % mAP. While this performance is comparatively high, classification is a much easier problem than detection. Object-centric methods (e.g. [5,6,8]), first detect the identity and location of objects in an image. Ni *et al.* [8] achieve 54.3 % mAP on MPII Cooking and 79 % on the ICPR 2012 Kitchen Scene Context-based Gesture Recognition dataset. While performance is state of the art, their method requires learning object models from a large number of manual annotations. In our work we learn a latent object representation without object annotations. Lastly, on Georgia Tech Egocentric Activities Li *et al.* [6] use object, egocentric, and hand features to achieve 66.8 % accuracy for action classification versus an IDT baseline of 39.8 %. Their features are similar to IDT but they use a recent hand-detection method to find the regions of most importance in each image.

Temporal Models: Several papers have used Conditional Random Fields for action segmentation and classification (e.g. [13,23,31,32]). CRFs offer a principled approach for combining multiple energy terms like segment-wise unaries and pairwise action transitions. Most of these approaches have been applied to simpler activities like recognizing *walking* versus *bending* versus *drawing* [31]. In each of the listed cases, segments are modeled with histograms of holistic features. In our work segments are modeled using spatiotemporal CNN activations.

Recently, there has been significant interest in Recurrent Neural Networks (RNNs), specifically those using Long Short Term Memory (LSTM) (e.g. [30,33,34]). LSTM implicitly learns how latent states transition between actions through the use of gating mechanisms. While their performance is often impressive, they are blackbox models that are hard to interpret. In contrast, the temporal component of our CNN explicitly learns how latent states transition and is easy to interpret and visualize. It is more similar to those in speech recognition (e.g. [35,36]) which learn phonemes using 1D convolutional filters or in robotics which learn sensor-based action primitives [37]. For completeness we compare our model with LSTM.

3 Spatiotemporal Model

In this section we introduce the spatial and temporal components of our ST-CNN. The input is a video including a color image and a motion image

[1] THUMOS Challenge: http://www.thumos.info/.

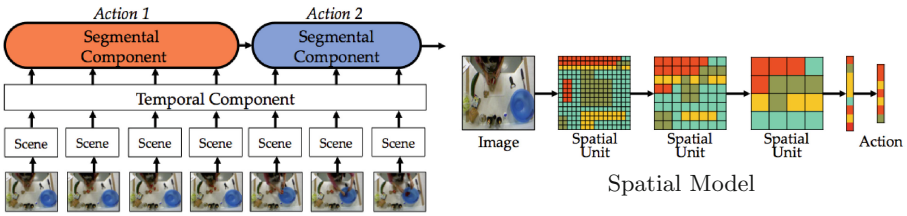

Fig. 2. (left) Our model contains three components. The spatial, temporal, and segmental units encode object relationships, how those relationships change, and how actions transition from one to another. (right) The spatial component of our model.

for each frame. The output is a vector of action probabilities at every frame. Figure 2 (left) depicts the full Segmental Spatiotemporal model.

3.1 Spatial Component

In this section, we introduce a CNN topology, inspired by VGG [18], that by construction captures object state and location in fine-grained actions. First we introduce the mathematical framework, as depicted in Fig. 2 (right), and then highlight differences between our approach and other CNNs. For a recent introduction to CNNs see [38].

For each timestep t there is an image pair $I_t = \{I_t^c, I_t^m\}$ where I_t^c is a color image and I_t^m is a Motion History Image [39]. The motion image captures when an object has moved into or out of a region and is computed by taking the difference between frames across a 2 second window. Other work (e.g. [27]) has shown success using optical flow as a motion image. We found that optical flow was insufficient for capturing small hand motions and was noisy due to the video compression.

The input is decomposed into an $N \times N$ grid of non-overlapping regions indexed by $i \in \{1, \ldots, R\}$. For each region, feature vector r_i, encodes object location and state and is computed by applying a series of spatial convolutional units over that part of the image. Each spatial unit, indexed by $l \in \{1, \ldots, L\}$, consists of a convolution layer with F_l filters of size 3×3, a Rectified Linear Unit (ReLU), and 3×3 max pooling. In Fig. 2 (right), each colored block in the third spatial unit corresponds to a feature vector r_i in that region.

A fully connected layer with F_{fc} states captures relationships between regions and their corresponding objects. For example, a state may produce a high score for *tomato* in the region with the cutting board and *knife* in the region next to it. Let $r \in \mathbb{R}^{R \cdot F_L}$ be the concatenation of all features $\{r_i\}_{i=1}^{R}$ and $h \in \mathbb{R}^{F_{fc}}$ be the fully connected states. The state is a function of weight matrix $W^{(0)}$ and biases $b^{(0)}$.[2]

[2] For notational clarity we denote all weight matrices as $W^{(\cdot)}$ and bias vectors $b^{(\cdot)}$ to reduce the number of variables.

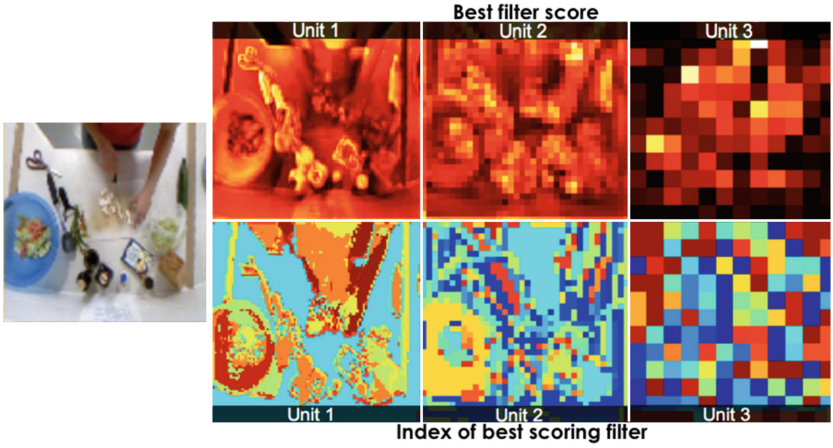

Fig. 3. The user is chopping vegetables. The top images show the best filter activations after each convolutional unit from the CNN. The activations around the cutting board and bowl are high (yellow) whereas in unimportant regions are low (black/red). The bottom images indicate which filter gave the highest activation for each region. Each color corresponds to a different filter index. (Color figure online)

$$h = \text{ReLU}(W^{(0)} r + b^{(0)}) \tag{1}$$

Ideally, the spatial and temporal components of our CNN should be trained jointly, however, this requires an exhorbitant amount of GPU memory, so the spatial model is trained and then the temporal model is trained. As such, we train the spatial component with auxiliary labels, z. We define z_t to be the ground truth action label for each timestep and compute the probability, $\hat{z}_t$, of that frame being each action using the softmax function:

$$\hat{z}_t = \text{softmax}(W^{(1)} h + b^{(1)}) \tag{2}$$

Note, that $\hat{z}_t$ is computed solely for training purposes. The input to the temporal component is the latent activations h_t.

Figure 3 shows example CNN activations after each spatial unit. The top row shows the sum of all filter activations after that layer and the bottom row shows the color corresponding to the best scoring filter at that location. We find that these filters are similar to mid-level object detectors. Notice the relevant objects in the image and the regions corresponding to the action all have high activations and different best-scoring filters.

Relationships to Other CNNs: Our network is inspired by models like VGG and AlexNet but differs in important ways. Like VGG, we employ a sequence of spatial units with common parameters like filter size. However, we found that using two consecutive convolution layers in each spatial unit has negligible impact on performance. Normalization layers, like in AlexNet, did not improve performance either. Overall our network is shallower, has fewer spatial

regions, and contains only one fully connected layer. In addition, common data-augmentation techniques like image rotation and translation have a negative impact on our performance. These augmentations tend to introduce unwanted spatial and rotational invariances which are important for our applications.

We performed cross validation using one to seven spatial units and grid sizes from 1×1 to 9×9 and found three spatial units with a 3×3 grid achieved the best results. By contrast, for image classification, deep networks tend to use at least four spatial units and have larger grid counts. VGG uses a 7×7 grid and Alexnet uses a 12×12 grid. A low spatial resolution naturally induces more spatial invariance which is useful when there is limited amounts of training data. To contrast, if the grid resolution is larger, there needs to be more training data to capture all object configurations. We compare performance our model with a pre-trained VGG network in the results.

3.2 Temporal Component

Temporal convolutional filters capture how the scene changes over the course of an action. These filters capture properties like the scene configuration at the beginning or end of an action and different ways users perform the same action.

For time t and video duration T let $H = \{h_t\}_{t=1}^{T}$ be the set of spatial features and $y_t \in \{1, \ldots, C\}$ be an action label. For convenience we define $H_{t:t+d}$ to be a sequence of features from time t to $t+d-1$. We learn F_e temporal filters $W^{(2)} = \{W_1^{(2)}, \ldots, W_{F_e}^{(2)}\}$ with biases $b^{(2)} = \{b_1^{(2)}, \ldots, b_{F_e}^{(2)}\}$ shared across actions. Each filter is of duration d such that $W_i^{(2)} \in \mathbb{R}^{F_{fc} \times d}$. The activation for the i-th filter at time t is given by a 1D convolution between the spatial features $H_{t:t+d}$ and the temporal filters using a ReLU non-linearity:

$$a_{t,i} = \text{ReLU}(W_i^{(2)} * H_{t:t+d} + b_i^{(2)}) = \text{ReLU}(\sum_{t'=0}^{d-1} W_{i,t'}^{(2)} H_{t+t'} + b_i^{(2)}) \quad (3)$$

A score vector $s_t \in \mathbb{R}^C$ is a function of weight vectors $W^{(3)} \in \mathbb{R}^{F_e \times C}$ and biases $b^{(3)}$ with the softmax function:

$$s_t = \text{softmax}(W^{(3)} a_t + b^{(3)}) \quad (4)$$

We choose filter lengths that span 10 s of video. This is much larger than those used in related work (e.g. [2,30]). Qualitatively we find these filters capture states, transitions between states, and attributes like action duration. In principle we could create a deep temporal model. In preliminary experiments we found that multiple layers did not improve performance, however, it is worth further exploration.

3.3 Learning

We learn parameters $W = \{W^0, W^1, W^2, W^3\}$, $b = \{b^0, b^1, b^2, b^3\}$, and the spatial convolutional filters with the cross entropy loss function. We minimize the

spatial network and temporal networks independently using ADAM [40], a recent method for stochastic optimization. Dropout regularization was used on fully connected layers.

Parameters such as grid size, number of filters, and non-linearity functions were chosen using cross validation. We use $F = \{64, 96, 128\}$ filters in the three corresponding spatial units and $F_{fc} = 256$ fully connected states. We used Keras [41], a library of deep learning tools, to implement our model.

4 Segmental Model

We jointly segment and classify actions with a constrained variation on the Semi-Markov CRF (SM-CRF) [20] using the activations from the spatiotemporal CNN and a pairwise term that captures action-to-action transitions between segments.

Let tuple $P_j = (y_j, t_j, d_j)$ be the jth action segment where y_j is the action label, t_j is the start time, and d_j is the segment duration. There is a sequence of M segments $P = \{P_1, \ldots, P_M\}$ for $0 < M \leq T$ such that the start of segment j coincides with the end of the previous segment $t_j = t_{j-1} + d_{j-1}$ and the durations sum to the total time $\sum_{i=1}^{M} d_i = T$. Given scores $S = \{s_1, \ldots, s_T\}$ we infer segments P that maximize the energy $E(S, P)$ for the whole video using segment function $f(\cdot)$:

$$E(S, P) = \sum_{j=1}^{M} f(S, y_{j-1}, y_j, t_j, d_j) \tag{5}$$

This model is a Conditional Random Field where $\Pr(P|S) \propto \exp(E(S, P))$. Our segment function contains transition matrix $A \in \mathbb{R}^{C \times C}$ and the sum of spatiotemporal CNN scores across a segment:

$$f(S, y_{j-1}, y_j, t_j, d_j) = A_{y_{j-1}, y_j} + \sum_{t=t_j}^{t_j + d_j - 1} S_t^{y_j} \tag{6}$$

Each element A_{y_{j-1}, y_j} of our pairwise term models the probability of transitioning from action y_{j-1} at segment $j-1$ to y_j at segment j. A is given by the log probabilities computed directly from the training data.

4.1 Segmental Inference

The inference method proposed by Sarawagi and Cohen [20], and rediscovered by Pirsiavash and Ramanan [23] for Segmental Regular Grammars, solves the following discrete optimization problem:

$$P^* = \underset{P_1, \ldots, P_M}{\arg\max}\ E(S, P) \quad s.t. \sum_{i=1}^{M} d_i = T \qquad \text{and} \quad 0 < M \leq T \tag{7}$$

Sarawagi and Cohen introduced an algorithm, which we refer to as Segmental Viterbi, that extends the traditional Viterbi method to the problem of joint segmentation and classification. The optimal labeling is computed by recursively computing the best score, $V_{t,c}$, for each time step t and class c where t corresponds to the ending time for a segment with duration d:

$$V_{t,c} = \max_{\substack{d \in \{1 \ldots D\} \\ c' \in \mathcal{Y}/c}} V_{t-d,c'} + f(S, c', c, t, d) \tag{8}$$

Their forward pass using our energy is shown in Algorithm 1. The optimal labels are recovered by backtracking through the matrix.

Their approach is inherently frame-wise: for each frame, they compute scores for all possible segment durations, current labels, and previous labels. In the naive case this results in an algorithm of complexity $O(T^2C^2)$ because the segments can be of arbitrary length. If the maximum segment duration, D, is bounded then complexity is reduced to $O(TDC^2)$.

We introduce an algorithm that is inherently segmental which is applicable for a broad range of energy functions so long as the energy is decomposable across frames in each segment. For each segment, we maximize over the start times, current labels, and previous labels. Instead of optimizing over segment durations, we optimize over the number of segments in a sequence. This can be computed efficiently formulating it as a constrained optimization problem where K is an upper bound on the number of segments:

$$P^* = \underset{P_1, \ldots, P_M}{\arg\max}\ E(S, P) \quad s.t. \quad 0 < M \leq K \tag{9}$$

In all cases, we set K based on the maximum number of segments in the training split. The best score for the kth segment starting at time t is given by $\bar{V}^k_{t,c}$. The recursive update is

$$\bar{V}^k_{t,c} = \max_{c' \in \mathcal{Y}} \bar{V}^{k'}_{t-1,c'} + A_{c',c} + S_{t,c} \tag{10}$$

where, if staying in the same segment ($c = c'$), then $k' = k$ and $A_{c',c} = 0$, otherwise $k' = k - 1$ and $A_{c',c}$ is the pairwise term.

The forward pass for our method is shown in Algorithm 2. Similar to Segmental Viterbi, the optimal labeling is found by backtracking through $\bar{V}$.

Algorithm 1. Seg. Viterbi [20]

for $t = 1 : T$ **do**
  for $c = 1 : C$ **do**
    $v_{cur} = V_{t-1,c}$
    $v_{prev} = \max_{c' \in \mathcal{Y}/c} \bar{V}^{k-1}_{t-1,c'} + A_{c',c}$
    $V_{t,c} = \max(v_{cur}, v_{prev}) + S^c_t$

Algorithm 2. Our algorithm

for $k = 1 : K$ **do**
  for $c = 1 : C$ **do**
    for $t = 1 : T$ **do**
      $v_{cur} = \bar{V}^k_{t-1,c}$
      $v_{prev} = \max_{c' \in \mathcal{Y}/c} \bar{V}^{k-1}_{t-1,c'} + A_{c',c}$
      $\bar{V}^k_{t,c} = \max(v_{cur}, v_{prev}) + S^c_t$

The complexity of our algorithm is $O(KTC^2)$. Assuming $K < D$ our solution is $\frac{D}{K}$ times more efficient than Segmental Viterbi. Clearly it becomes more efficient as the ratio of the number of segments K to the maximum segment duration D decreases. In most practical applications, K is much smaller than D. In the evaluated datasets there is a speedup of one to three orders of magnitude. Note, however, our method requires more memory than Segmental Viterbi. Ours has space complexity $O(KTC)$ whereas Segmental Viterbi has $O(TC)$. Typically $K << T$ so the increase in memory is easily manageable on any modern computer.

5 Experimental Setup

Historically, most action recognition datasets were developed for classifying individual actions using pre-trimmed clips. Recent datasets for fine-grained recognition have been developed to classify many actions, however they often contain too few users or an insufficient amount of data to learn complex models. MPII Cooking [4] has a larger number of videos but some actions are rarely performed. Specifically, seven actions are performed fewer than ten times each. Furthermore there is gratuitous use of a background class because it was labeled for (sparse) action detection instead of (dense) action segmentation. Georgia Tech Egocentric Activities [42] has 28 videos across seven tasks. Unfortunately, the actions in each task are independent thus there are only three videos to train on and one for evaluation. Furthermore the complexities of egocentric video are beyond the scope of this work. We use datasets from the ubiquitous computing and surgical robotics communities which contain many instances of each action.

University of Dundee 50 Salads: Stein and McKenna introduced 50 Salads [21] for evaluating fine-grained action recognition in the cooking domain. We believe this dataset provides great value to the computer vision community due to the large number of action instances per class, the high quality labels, plethora of data, and multi-modal sensors (RGB, depth, and accelerometers).

This dataset includes 50 instances of salad preparation where each of the 25 users makes a salad in two different trials. Videos are annotated at four levels of granularity. The coarsest level ("high") consists of labels *cut and mix ingredients*, *prepare dressing*, and *serve salad*. At the second tier ("mid") there are 17 fine-grained actions like *add vinegar*, *cut tomato*, *mix dressing*, *peel cucumber*, *place cheese into bowl*, and *serve salad*. At the finest level ("low") there are 51 actions indicating the start, middle, and end of the previous 17 actions. For each granularity there is also a *background* class.

A fourth granularity ("eval"), suggested by [21], consolidates some object-specific actions like *cutting a tomato* and *cutting a cucumber* into object-agnostic actions like *cutting*. Actions include *add dressing*, *add oil*, *add pepper*, *cut*, *mix dressing*, *mix ingredients*, *peel*, *place*, *serve salad on plate*, and *background*. These labels coincide with the tools instrumented with accelerometers.

JHU-ISI Gesture and Skill Assessment Working Set (JIGSAWS): JIGSAWS [11] was developed for recognizing actions in robotic surgery training

tasks like suturing, needle passing, and knot tying. In this work we evaluate using the suturing task which includes 39 trials of synchronized video and robot kinematics data collected from a daVinci medical robot. The video is captured from an overhead endoscopic camera and depicts two tools and the training task apparatus. The suturing task consists of 10 fine-grained actions such as *insert needle into skin*, *tie a knot*, *transfer needle between tools*, and *drop needle at finish*. Videos last about two minutes and contain 15 to 37 action instances per video. Users perform low-level actions in significantly different orders. We evaluate using Leave One User Out as suggested in [11]. Most prior work on this dataset focuses on the kinematics data which consists of positions, velocities, and robot joint information. We compare against [13] who provide video-only results. Their approach uses holistic features with a Markov Semi-Markov CRF.

Metrics: We evaluate on segmental and frame-wise metrics as suggested by [37] for the 50 Salads dataset. The first measures segment *cohesion* and the latter captures overall *coverage*.

The segmental metric evaluates the ordering of actions but not the specific timings. The motivation is that in many applications there is high uncertainty in the location of temporal boundaries. For example, different annotators may have different interpretations of when an action starts or ends. As such, the precise location of temporal boundaries may be inconsequential. This score, $A_{edit}(P, P^*)$, is computed using the Levenshtein distance which is a function of segment insertions, deletions, and substitutions [43]. Let the ground truth segments be $P = \{P_1, \ldots, P_M\}$ and predicted segments be $P^* = \{P_1^*, \ldots, P_N^*\}$. The number of edits is normalized by the maximum of M and N. For clarity we show the score $(1 - A_{edit}(P, P^*)) \times 100$ which is from 0 to 100.

Frame-wise accuracy measures the percentage of correct frames in a sequence. Let $y = \{y_1, \ldots, y_T\}$ be the true labels and $y^* = \{y_1^*, \ldots, y_T^*\}$ be the predicted labels. The score is a function of each frame: $A_{acc}(y, y^*) = \frac{1}{T} \sum_{t=1}^{T} \mathbf{1}(y_t = y_t^*)$.

We also include action classification results which assume temporal segmentation is known. These use the accuracy metric applied to segments instead of individual frames.

Baselines: We evaluate two spatial baselines on both datasets using Improved Dense Trajectories and a pre-trained VGG network, and one temporal baseline using a Recurrent Neural Network with LSTM. For the classification results, the (known) start and end times are fed into the segmental model to predict each class.

The Dense Trajectory (IDT) baseline is comparable to Rohrbach *et al.* [4] on the MPII dataset. We extract IDT, create a KMeans dictionary ($k = 2000$), and aggregate the dictionary elements into a locally normalized histogram with a sliding window of 30 frames. We only use one feature type, HOG, because it outperformed all other feature types or their combination. This may be due to the large dimensionality of IDT and relatively low number of samples from our training sets. Note that it took 18 h to compute IDT features on 50 Salads compared to less than 5 h for the CNN features using a Nvidia Titan X graphics card.

Table 1. 50 salads

Spatial models	Edit	Accuracy
VGG	7.58	38.30
IDT	16.77	54.28
S-CNN	**24.10**	**66.64**
Spatiotemporal	Edit	Accuracy
S-CNN + LSTM	58.84	66.30
ST-CNN	60.98	71.37
ST-CNN + Seg	**62.06**	**72.00**

Table 2. JIGSAWS

Spatial models	Edit	Accuracy
VGG	**24.29**	45.91
IDT	8.45	53.92
S-CNN	20.10	**67.25**
Spatiotemporal	Edit	Accuracy
[13] STIPS+CRF	-	71.78
S-CNN + LSTM	54.07	68.37
ST-CNN	59.89	71.21
ST-CNN + Seg	**66.56**	**74.22**

Table 3. 50 salads granularity analysis

Labels	Classes	Edit	Acc.	Classif.
Low	52	29.30	44.13	39.67
Mid	18	48.94	58.06	63.49
Eval	10	66.44	72.71	86.63
High	4	83.2	92.43	95.14

Table 4. Speedup analysis

Labels	Dur	#Segs	Speedup
Low	2289	65	35×
Mid	3100	25	124×
Eval	3100	24	129×
High	11423	6	1902×
JIGSAWS	1107	37	30×

For our spatial-only results, we classify the action at each time step with a linear Support Vector Machine using the features from IDT, VGG, or our spatial CNN. These results highlight how effective each model is at representing the scene and are not meant to be state of the art. The CNN baseline uses the VGG network [18] pretrained on Imagenet. We use the activations from FC6, the first of VGG's three fully connected layers, as the features at each frame.

In addition we compare our temporal model to a Recurrent Neural Network with LSTM using our spatial CNN as input. The LSTM baseline was implemented in Keras and uses one LSTM layer with 64 latent states.

6 Results and Discussion

Tables 1 and 2 show performance using Dense Trajectories (IDT), VGG, LSTM, and our models. S-CNN, ST-CNN, and ST-CNN + Seg refer to the spatial, spatiotemporal, and segmental components of our model. These 50 Salads results are on the "eval" granularity. Our full model has 27.8 % better accuracy on 50 Salads and 37.6 % better accuracy on JIGSAWS relative to the IDT baseline. Figure 4 shows example predictions using each component of our model.

Spatial Model: Our results are consistent with the claim that holistic methods like IDT are insufficient for fine-grained action segmentation. Interestingly, we

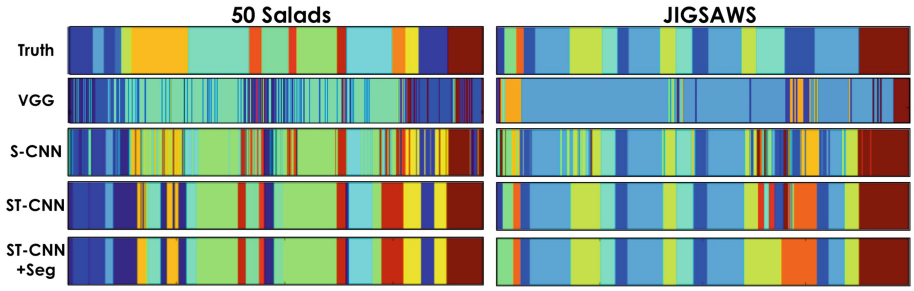

Fig. 4. The plots on top depict the ground truth action predictions for a given video. Each color corresponds to a different class label. Subsequent rows show predictions using VGG, S-CNN, ST-CNN, and ST-CNN + Seg. (Color figure online)

also see that the VGG results are also relatively poor which could be due to the data augmentation to train the model. While our results are still insufficient for many practical applications the accuracy of our spatial model is at least 12 % better than IDT and 21 % better than VGG on both datasets. Note that the edit score is very low for all of these models. This is not surprising because each model only uses local temporal information which results in many oscillations in predictions, as shown in Fig. 4.

Many actions in 50 Salads, like cutting, require capturing small hand motions. We visualized IDT[3] and found it does not detect many tracklets for these actions. In contrast, when the user performs actions like placing ingredients in the bowl, IDT generates thousands of tracklets. Even though the IDT features are normalized we find this is still problematic. We visualized our method, for example as shown in Fig. 3, and qualitatively found it is better at capturing details necessary for finer motions.

Temporal Model: The spatiotemporal model (*ST-CNN*) outperforms the spatial model (*S-CNN*) on both datasets. The effect on edit score is substantial and likely due to the large temporal filters. Aside from modeling temporal evolution these have a byproduct of smoothing out predictions. By visualizing these features we see they tend to capture different phases of an action like the start or finish. In contrast, while LSTM substantially improves edit score over the spatial model it has a negligible impact on accuracy. LSTM is capable of learning how actions transition across time, however, it does not appear that it sufficiently captures this information. Due to the complex nature of this method, we were not able to visualize the internal parameters in a meaningful way.

Segmental Model: The segmental model provides a notable improvement on JIGSAWS but only a modest improvement on 50 Salads. By visualizing the results we see that the segmental model helps in some cases and hurts in others. For example, when the predictions oscillate (like in Fig. 4 (right)) the segmental model provides a large improvement. However, sometimes the model smooths

[3] Visualization was performed using the public software from Wang and Schmid [1].

over actions that are short in duration. Future work should look at incorporating additional cues such as duration to better model each action class.

Action Granularity: Table 3 shows performance on all four action granularities from 50 Salads using our full model. Columns 3 and 4 show scores for segmental and frame-wise metrics on the action segmentation task and the last shows action classification accuracies which assume temporal segmentation is known. While performance decreases as the number of classes increases, results degrade sub-linearly with each additional class. Some errors at the finer levels are likely due to temporal shifts in the predictions. Given the high accuracy at the courser levels, future work should look at modeling finer granularities using by modeling actions hierarchically.

Other Results: Lea *et al.* [37] showed results using the instrumented kitchen tools on 50 Salads. Their model achieves an edit score of 58.46 % and accuracy of 81.75 %. They also achieve state of the art performance on JIGSAWS with 78.91 % edit and 83.45 % accuracy. Note we do not expect to achieve as high performance from video. These results used domain-specific sensors which are well suited to each application but may not be practical for real-world deployment. To contrast, video is much more practical for deployment but is more complicated to model.

Our classification accuracy on JIGSAWS is 90.47 %. This is notably higher than the state of the art [12] which achieves 81.17 % using a video-based linear dynamical system model and also better than their hybrid approach using video and kinematics which achieves 86.56 %. For joint segmentation and classification the improvement over the state of the art [13] is modest. These surgical actions can be recognized well using position and velocity information [37], thus our ability to capture object relationships may be less important on this dataset.

Speedup: Table 4 shows the speedup of our inference algorithm compared to Segmental Viterbi on all 50 Salads and JIGSAWS label sets. One practical implication is that our algorithm scales readily to full-length videos. On 50 Salads, Segmental Viterbi takes 2 h to compute high-level predictions for all trials compared to a mere 4 s using ours.

7 Conclusion

In this paper we introduced a segmental spatiotemporal CNN that substantially outperforms popular methods like Dense Trajectories, pre-trained spatial CNNs, and temporal models like Recurrent Neural Networks with LSTM. Furthermore, our approach takes less time to compute features than IDT, less time to train than LSTM, and performs inference more efficiently than traditional Segmental methods. We hope the insights from our discussions are useful for highlighting the nuances of fine-grained action recognition.

Acknowledgements. This work was funded in part by grants NIH R01HD87133-01, ONR N000141310116, and an Intuitive Surgical Research Grant.

References

1. Wang, H., Schmid, C.: Action recognition with improved trajectories. In: IEEE International Conference on Computer Vision (ICCV) (2013)
2. Sun, L., Jia, K., Yeung, D.Y., Shi, B.: Human action recognition using factorized spatio-temporal convolutional networks. In: IEEE International Conference on Computer Vision (ICCV) (2015)
3. Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J.: ActivityNet: a large-scale video benchmark for human activity understanding. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
4. Rohrbach, M., Rohrbach, A., Regneri, M., Amin, S., Andriluka, M., Pinkal, M., Schiele, B.: Recognizing fine-grained and composite activities using hand-centric features and script data. Int. J. Comput. Vis. (IJCV) **119**, 346–373 (2015)
5. Fathi, A., Rehg, J.M.: Modeling actions through state changes. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2013)
6. Li, Y., Ye, Z., Rehg, J.M.: Delving into egocentric actions. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
7. Cheron, G., Laptev, I., Schmid, C.: P-CNN: pose-based CNN features for action recognition. In: IEEE International Conference on Computer Vision (ICCV) (2015)
8. Ni, B., Paramathayalan, V.R., Moulin, P.: Multiple granularity analysis for fine-grained action detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
9. Vo, N.N., Bobick, A.F.: From stochastic grammar to Bayes network: probabilistic parsing of complex activity. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
10. Hawkins, K.P., Bansal, S., Vo, N.N., Bobick, A.F.: Anticipating human actions for collaboration in the presence of task and sensor uncertainty. In: IEEE International Conference on Robotics and Automation (ICRA) (2014)
11. Gao, Y., Vedula, S.S., Reiley, C.E., Ahmidi, N., Varadarajan, B., Lin, H.C., Tao, L., Zappella, L., Béjar, B., Yuh, D.D., et al.: JHU-ISI gesture and skill assessment working set (JIGSAWS): a surgical activity dataset for human motion modeling. In: MICCAI Workshop: Modeling and Monitoring of Computer Assisted Interventions (M2CAI) (2014)
12. Zappella, L., Haro, B.B., Hager, G.D., Vidal, R.: Surgical gesture classification from video and kinematic data. Med. Image Anal. **17**, 732–745 (2013)
13. Tao, L., Zappella, L., Hager, G.D., Vidal, R.: Surgical gesture segmentation and recognition. In: Medical Image Computing and Computer Assisted Intervention (MICCAI), pp. 339–346 (2013)
14. Lei, J., Ren, X., Fox, D.: Fine-grained kitchen activity recognition using RGB-D. In: ACM Conference on Ubiquitous Computing (UbiComp) (2012)
15. Morariu, V.I., Davis, L.S.: Multi-agent event recognition in structured scenarios. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2011)
16. Koppula, H.S., Gupta, R., Saxena, A.: Learning human activities and object affordances from RGB-D videos. Intl. J. Robot. Res. (IJRR) **32**(8), 951–970 (2013)
17. van Kasteren, T., Englebienne, G., Kröse, B.J.: Activity recognition using semi-Markov models on real world smart home datasets. J. Ambient Intell. Smart Environ. **2**, 311–325 (2010)
18. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: International Conference Learning Representations (ICLR) (2015)

19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: ImageNet classification with deep convolutional neural networks. In: Pereira, F., Burges, C., Bottou, L., Weinberger, K. (eds.) Advances in Neural Information Processing Systems (NIPS) (2012)
20. Sarawagi, S., Cohen, W.W.: Semi-Markov conditional random fields for information extraction. In: Advances in Neural Information Processing Systems (NIPS) (2004)
21. Stein, S., McKenna, S.J.: Combining embedded accelerometers with computer vision for recognizing food preparation activities. In: ACM Conference on Ubiquitous Computing (UbiComp) (2013)
22. Wang, H., Kläser, A., Schmid, C., Liu, C.L.: Action recognition by dense trajectories. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2011)
23. Pirsiavash, H., Ramanan, D.: Parsing videos of actions with segmental grammars. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
24. Jain, M., van Gemert, J.C., Snoek, C.G.M.: What do 15,000 object categories tell us about classifying and localizing actions? In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
25. Pishchulin, L., Andriluka, M., Schiele, B.: Fine-grained activity recognition with holistic and pose based features. In: German Conference on Pattern Recognition (GCPR) (2014)
26. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Fei-Fei, L.: Large-scale video classification with convolutional neural networks. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2014)
27. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: Advances in Neural Information Processing Systems (NIPS), pp. 568–576 (2014)
28. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3D convolutional networks. In: The IEEE International Conference on Computer Vision (ICCV) (2015)
29. Peng, X., Schmid, C.: Encoding feature maps of cnns for action recognition. In: CVPR, THUMOS Challenge 2015 Workshop (2015)
30. Ng, J.Y., Hausknecht, M.J., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: deep networks for video classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
31. Shi, Q., Cheng, L., Wang, L., Smola, A.: Human action segmentation and recognition using discriminative semi-Markov models. Intl. J. Comput. Vis. (IJCV) **93**, 22–32 (2011)
32. Tang, K., Fei-Fei, L., Koller, D.: Learning latent temporal structure for complex event detection. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2012)
33. Vinyals, O., Toshev, A., Bengio, S., Erhan, D.: Show and tell: a neural image caption generator. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
34. Donahue, J., Hendricks, L.A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., Darrell, T.: Long-term recurrent convolutional networks for visual recognition and description. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015)
35. Hannun, A.Y., Case, C., Casper, J., Catanzaro, B.C., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., Ng, A.Y.: Deep speech: scaling up end-to-end speech recognition. CoRR abs/1412.5567 (2014)

36. Abdel-Hamid, O., Deng, L., Yu, D.: Exploring convolutional neural network structures and optimization techniques for speech recognition. In: INTERSPEECH International Speech Communication Association (2013)
37. Lea, C., Vidal, R., Hager, G.D.: Learning convolutional action primitives for fine-grained action recognition. In: IEEE International Conference on Robotics and Automation (ICRA) (2016)
38. Ian Goodfellow, Y.B., Courville, A.: Deep Learning. MIT Press (2016, in preparation). http://www.deeplearningbook.org/
39. Davis, J., Bobick, A.: The representation and recognition of action using temporal templates. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (1997)
40. Kingma, D.P., Ba, J.: Adam: a method for stochastic optimization. In: International Conference Learning Representations (ICLR) (2014)
41. Chollet, F.: Keras (2015). https://github.com/fchollet/keras
42. Fathi, A., Farhadi, A., Rehg, J.M.: Understanding egocentric activities. In: IEEE International Conference on Computer Vision (ICCV) (2011)
43. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**, 31–88 (2001)