A Multi-Stream Bi-Directional Recurrent Neural Network for Fine-Grained Action Detection

Bharat Singh
U. of Maryland
bharat@cs.umd.edu

Tim K. Marks Michael Jones Oncel Tuzel
Mitsubish Electric Research Labs (MERL)
{tmarks, mjones, oncel}@merl.com

Ming Shao
Northeastern University
mingshao@ccs.neu.edu

Abstract

We present a multi-stream bi-directional recurrent neural network for fine-grained action detection. Recently, two-stream convolutional neural networks (CNNs) trained on stacked optical flow and image frames have been successful for action recognition in videos. Our system uses a tracking algorithm to locate a bounding box around the person, which provides a frame of reference for appearance and motion and also suppresses background noise that is not within the bounding box. We train two additional streams on motion and appearance cropped to the tracked bounding box, along with full-frame streams. Our motion streams use pixel trajectories of a frame as raw features, in which the displacement values corresponding to a moving scene point are at the same spatial position across several frames. To model long-term temporal dynamics within and between actions, the multi-stream CNN is followed by a bi-directional Long Short-Term Memory (LSTM) layer. We show that our bi-directional LSTM network utilizes about 8 seconds of the video sequence to predict an action label. We test on two action detection datasets: the MPII Cooking 2 Dataset, and a new MERL Shopping Dataset that we introduce and make available to the community with this paper. The results demonstrate that our method significantly outperforms state-of-the-art action detection methods on both datasets.

1. Introduction

In this paper, we present an approach for detecting actions in videos. *Action detection* refers to the problem of localizing temporally and spatially every occurrence of each action from a known set of action classes in a long video sequence. This is in contrast to most of the previous work in video activity analysis, which has focused on the problem of *action recognition* (also called action classification). In action recognition, a temporally segmented clip of a video is given as input, and the task is to classify it as one of N known actions. For action recognition, temporal localization is not required, as each video clip is *trimmed* to contain precisely the full duration (from start to finish) of one action. Furthermore, action recognition algorithms do not need to consider the case that a presented clip might not contain any of the known actions. In general, action detection is more difficult than action recognition. However, it is worth overcoming that difficulty because action detection is also much more relevant to real-world applications.

In this work, we focus on fine-grained action detection. We use the term *fine-grained* in the same sense as [20] to indicate that the differences among the classes of actions to be detected are small. For example, in a cooking scenario, detecting similar actions such as chopping, grating, and peeling constitutes fine-grained action detection.

We propose a method for fine-grained action detection in long video sequences, based on a Multi-Stream Bi-Directional Recurrent Neural Network (MSB-RNN). We call our neural network *multi-stream* because it begins with a convolutional neural network (CNN) that has four streams: two different streams of information (motion and appearance) for each of two different spatial frames (full-frame and person-centric). The video that is input to the network is split into a sequence of brief (6-frame-long) chunks. The multi-stream network output is a sequence of high-level representations of these chunks. These are input to bi-directional long short-term memory (LSTM) [8, 6] units to analyze long-term temporal dynamics.

Previous deep learning approaches use features that are computed over the full spatial extent of the video frame. We show the importance of using a tracked bounding box around the person to compute features relative to the location of the person, in addition to full-frame features, to provide both location-independent and location-dependent information. Unlike some previous work that represents motion information using a sequence of flow fields [22], we instead use a sequence of corresponding pixel displacements that we call *pixel trajectories*, as illustrated in Figure 3. The advantage of pixel trajectories is that the displacements for a moving point in the scene are represented at the same pixel location across several frames. We analyze the relative im-

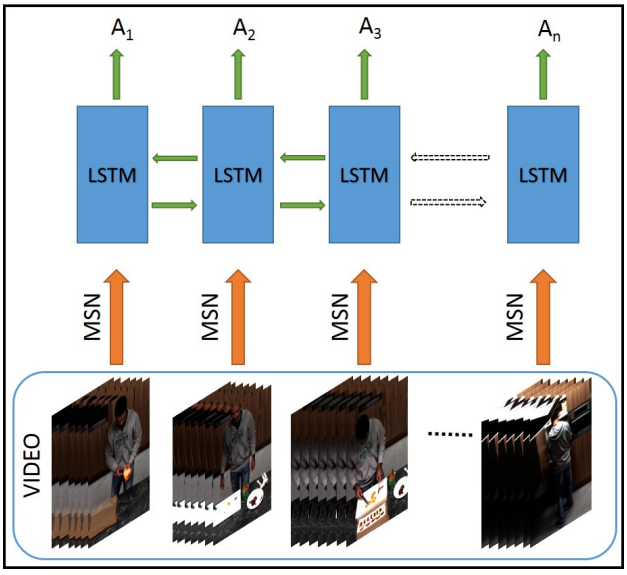

Figure 1. Framework for our approach. Short chunks of a video are given to a multi-stream network (MSN) to create a representation for each chunk. The sequence of these representations is then given to a bi-directional LSTM, which is used to predict the action label, A_i. Details of the multi-stream network are shown in Fig. 2.

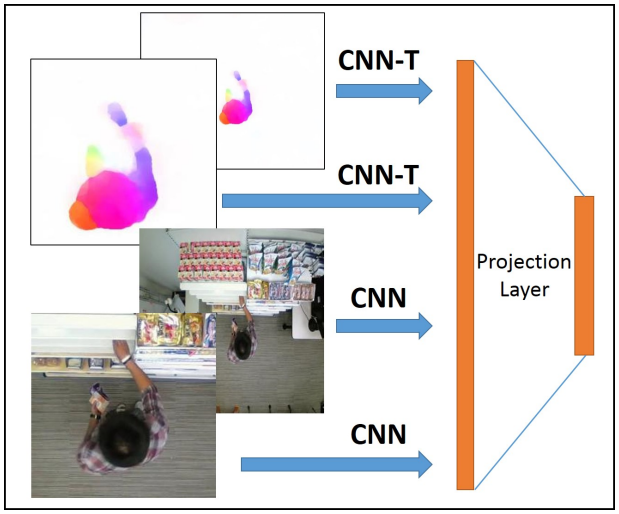

Figure 2. Figure depicting our multi-stream network (MSN). The multi-stream network uses two different streams of information (motion and appearance) for each of two different spatial croppings (full-frame and person-centric) to analyze short chunks of video. One network (CNN-T) computes features on pixel trajectories (motion), while the other (CNN) computes features on RGB channels (appearance).

portance of each of these components using two different datasets. The first is the MPII Cooking 2 Dataset [21], and the second is a new dataset we introduce containing overhead videos of people shopping from grocery-store shelves. Our results on the MPII Cooking 2 Dataset represent a significant improvement over the previous state of the art.

Our work includes the following novel contributions:

- We demonstrate the effectiveness of a bi-directional LSTM for the action detection task. It should be noted that although LSTMs have been used before for action recognition and sentence generation, we are the first to analyze the importance of LSTMs for action detection. Furthermore, since our LSTM layer is trained on full-length videos containing multiple actions (not just trimmed clips of individual actions), it can learn interactions among temporally neighboring actions.
- We train a multi-stream convolutional network that consists of two 2-stream networks, demonstrating the importance of using both full-frame and person-centric cropped video. We use pixel trajectories rather than stacked optical flow as input to the motion streams, leading to a significant improvement in results.
- We introduce a new action detection dataset, which we release to the community with this publication.

2. Related Work

Early work that can be considered action detection includes methods that detect walking people by analyzing simple appearance and motion patterns [26, 2]. Several algorithms have been proposed since then for detecting actions using space time interest points [33], multiple instance learning [9], or part-based models [25, 10]. By adding another dimension (time) to object proposals, action proposals have also been used for detection [11, 32].

Until recently, the standard pipeline for most video analysis tasks such as action recognition, event detection, and video retrieval was to compute hand-crafted features such as Histogram of Oriented Gradients (HOG), Motion Boundary Histogram (MBH), and Histogram of Optical Flow (HOF) along improved dense trajectories [28], create a Fisher vector for each video clip, then perform classification using support vector machines. In fact, shallow architectures using Fisher vectors still give state-of-the-art results for action/activity recognition [17, 29, 21]. Wang et al. [29] showed that results improved when hand-crafted features were replaced by deep features that were computed by convolutional neural networks whose inputs were images and stacked optical flow along trajectories. In [22], a two-stream network was proposed in which video frames and stacked optical flow fields (computed over a few frames) were fed to a deep neural network for action recognition. A similar architecture was used for spatial localization of actions [5] in short video clips. However, these networks did not learn long-term sequence information from videos.

Since recurrent neural networks can learn long-term sequence information in a data-driven fashion, they have recently gained traction in the action recognition community

[1, 3, 15]. In [1], a 3D convolutional neural network followed by an LSTM classifier was successful at classifying simple actions. LSTMs have shown improved performance over a two-stream network for action recognition [3, 15]. Recently, bi-directional LSTMs were also successful in skeletal action recognition [4]. However, even after using LSTMs, deep learning methods perform only slightly better than fisher vectors built on hand-crafted features for many action recognition tasks [15].

Although substantial progress has been made in action recognition [17, 22, 29, 21], not as much work has been done in action detection (spatio-temporal localization of actions in longer videos). The major focus in action detection has been on using high level semantic information to improve performance, rather than making use of bottom up cues. Using annotations for the objects being interacted with [16, 21] or enforcing the grammar of the high level activity being performed [13, 18] is generally helpful, though these approaches may require learning extra detectors for objects and having prior knowledge about high-level activities. Sun et al. [24] used LSTMs for action detection although their focus was on leveraging web images to help with video action detection. Their paper did not analyze the importance of LSTMs as we do here.

For fine-grained action detection, extracting trajectories from spatio-temporal regions of interest or using hand-trajectories has shown significantly improved performance [16, 21]. In recent work in generating sentences from images, LSTM networks with attention models [31, 14] learn to focus on salient regions in an image to generate captions for the image. Since motion and actor location are important clues for knowing where an action is happening, we were inspired by these methods to add our network's two person-centric streams, which capture information from regions of video that are salient due to actor motion.

3. Approach

Our framework is shown in Figs. 1 and 2. First, we train four independent convolutional neural networks, each based on the VGG architecture [23], to perform the task of action classification when given as input a single small chunk (6 consecutive frames) of video. As shown in Fig. 2, two of the networks (one each for images and motion) are trained on chunks of full-frame video, so that the spatial context of the action being performed is preserved. The other two networks (one each for images and motion) are trained on frames that have been cropped to a tracked bounding box. These cropped frames provide actions with a reference frame, which helps in classifying them. After these four networks have been trained, we learn a fully-connected projection layer on top of all four fc7 layer outputs, to create a joint representation for these independent streams. This multi-stream network (MSN) is shown in Fig. 2. As illustrated in Fig. 1, the multi-stream network is provided with full-length video (arranged as a temporal sequence of 6-frame chunks), and the corresponding temporal sequence of outputs of the projection layer is then fed into an LSTM network running in two directions. We use a fully connected layer on top of each directional LSTM's hidden states, followed by a softmax layer, to obtain an intermediate score corresponding to each action. Finally, the scores for the two LSTMs are averaged to get action-specific scores.

There are multiple components in an action detection pipeline that are critical for achieving good performance. In this task, we need a model that captures both spatial and long-term temporal information that are present in a video. Person tracks (bounding boxes) provide a reference frame that make many actions easier to learn by removing location variation from the input representation. Some actions, however, are location dependent. For scenes shot using a static camera, as in our test datasets, these actions always occur at the same image location. For example, washing/rinsing are almost always done near the sink, and opening a door would most likely be performed near a refrigerator or a cupboard. For these reasons, we train two separate deep networks each on pixel trajectories and image appearance. The first network is trained on the entire frame to preserve the global spatial context. The second network is trained on cropped boxes from the tracker to reduce background noise and to provide a person-centric reference frame for trajectories and image regions. To capture short-term temporal information, we use pixel trajectories, in which each moving scene point is in positional correspondence with itself across several frames. This alignment enables pixel trajectories to capture richer motion information than stacked optical flow fields. Since actions can be of any duration, our method uses LSTMs to learn the duration and long-term temporal context of actions in a data-driven fashion. Our results demonstrate that LSTMs are quite effective in learning long-term temporal context for fine-grained action detection.

3.1. Tracking for Fine-Grained Action Detection

To provide a bounding box around the person for the location-independent appearance and motion streams, any good person-tracking algorithm could be used. In this paper, we use a simple state-based tracker to spatially localize actions in a video with a single actor. Keeping the size (chosen manually) of the tracked bounding box fixed, we update its position so that the magnitude of flow inside the box is maximized. If the magnitude is below a threshold, the location is not updated (when the person is not moving, the bounding box is stationary). Initially, if no actor is present, the bounding box is arbitrarily placed. The location of the bounding box is updated only after a video chunk (6 frames) is processed and flow/appearance features

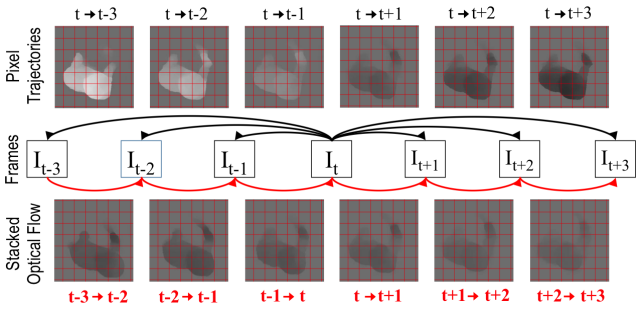

Figure 3. The middle row of squares represent a sequence of frames, and the arrows indicate the pairs of frames between which optical flow is computed for both pixel trajectories (arrows above the frames) and stacked optical flow (arrows below the frames). The top and bottom rows show the y-component of optical flow computed for pixel trajectories (*top row*) and stacked flow (*bottom row*). In pixel trajectories, note that only the intensity changes, while the spatial layout of the image stays the same. Thus, only a single convolution layer in time is sufficient for learning motion features for a pixel. In stacked optical flow, however, the spatial correspondence between pixels is lost. For example, the back of the head (lowest point of the silhouette) moves up and to the left in subsequent images of stacked optical flow.

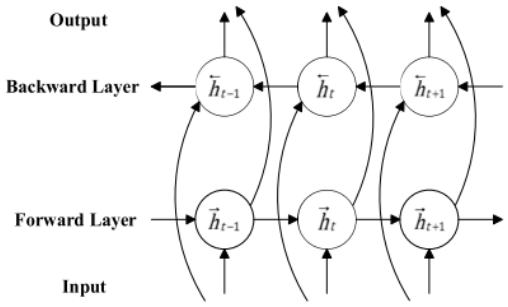

Figure 4. Connections depicting architecture of a bi-directional LSTM [7]. The circular nodes represent LSTM cells.

are computed relative to it, to ensure that the bounding box is stationary over the 6 frames of a chunk. Our simple tracking method can be effectively applied when the camera is stationary and we have a reasonable estimate about the size of the actor. This is a practical assumption for many videos taken at retail stores, individual homes, or in a surveillance setting where fine-grained action detection is likely to be used. For more difficult tracking situations, a more sophisticated tracker would be needed.

3.2. Training of Flow Networks

Stacking optical flow as an input to the deep network has been a standard practice in the literature to train motion-based networks [22, 29, 30]. However, in stacked optical flow, the motion vectors corresponding to a particular moving point in the scene (e.g., the tip of a finger) change their pixel location from one frame to the next. Thus, the convolutional neural network needs to learn the spatial movement of optical flow for classifying an action. The complete motion information could be learned by the network at a higher layer, but that would require more parameters and data to learn. An alternate representation for motion in a sequence of frames is to compute flow from a central frame, t, to each of the K previous and K subsequent frames (we use $K = 3$). This representation, which we call *pixel trajectories*, is illustrated and compared with stacked optical flow in Figure 3. In all $2K$ frames of a pixel trajectory, the flow values from each point to the corresponding point in frame t are all located at the point's location in frame t. As shown in Figure 3, in pixel trajectories, only the intensity of the optical flow image changes (its location is fixed). Thus, the network can learn a temporal filter for each pixel more easily than from stacked flow fields.

Now, for each pixel in frame t, we have the complete motion information in a short window of time. To learn motion patterns for each pixel, a $1 \times 2K$ convolutional kernel can produce a feature map for the movement of each pixel. In contrast, a network layer that inputs stacked optical flow (using, e.g., a $3 \times 3 \times 2K$ kernel on stacked optical flow) will not be able to learn motion patterns using the first convolutional layer for pixels that have a displacement of more than 3 pixels over $2K$ frames. A similar method to pixel trajectories was mentioned in [22], but there it yielded slightly worse performance than stacked optical flow, likely because it was applied on moving camera videos where trajectories are less reliable. For fine-grained action detection with a stationary camera, however, we demonstrate that pixel trajectories perform better than stacked flow for both datasets (see Table 2).

3.3. Training on Long Sequences using a Bi-Directional LSTM Network

We now provide a brief background of Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) cells [8]. Given an input sequence, $\mathbf{x} = (x_1, \ldots, x_T)$, an RNN uses a hidden state representation $\mathbf{h} = (h_1, \ldots, h_T)$ so that it can map the input $\mathbf{x}$ to the output sequence $\mathbf{y} = (y_1, \ldots, y_T)$. To compute this representation, it iterates through the following recurrence equations:

$$h_t = g(W_{xh}x_t + W_{hh}h_{t-1} + b_h), \quad y_t = g(W_{hy}h_t + b_z),$$

where g is an activation function, W_{xh} is the weight matrix which maps the input to the hidden state, W_{hh} is the transition matrix between hidden states at two adjacent time steps, W_{hy} is a matrix which maps the hidden state h to the output y, and b_h and b_z are bias terms.

The weight update equations for an LSTM cell are as

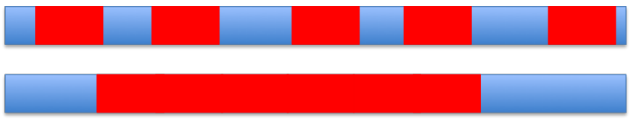

Figure 5. The bottom row shows the output of a method that produces contiguous segments. The top row shows another method, which generates the same proportion of non-contiguous segments.

follows:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \\
g_t &= \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
c_t &= f_t c_{t-1} + i_t\ g_t \\
h_t &= o_t\ \tanh(c_t)
\end{aligned}
$$

where σ is a sigmoid function, tanh is the hyperbolic tangent function, and i_t, f_t, o_t, and c_t are the *input gate*, *forget gate*, *output gate*, and *memory cell* activation vectors, respectively. The forget gate f_t decides when (and which) information should be cleared from the memory cell c_t. The input gate i_t decides when (and which) new information should be incorporated into the memory. The $\tanh$ layer g_t generates a candidate set of values which will be added to the memory cell if the input gate allows it. Based on the output of the forget gate f_t, input gate i_t, and the new candidate values g_t, the memory cell c_t is updated. The output gate o_t controls which information in the memory cell should be used as a representation for the hidden state. Finally, the hidden state is represented as a product between a function of the memory cell state and the output gate.

In action recognition datatsets (e.g., UCF 101), video clips are temporally trimmed to start and end at the start and end times of each action, and are generally short in length (e.g., from 2–20 seconds). Hence, in the action recognition task, there is not enough long-term context to be learned in a data-driven fashion. This long-term context could include properties such as the expected duration of an action, which action follows or precedes an action, or long-term motion patterns that extend beyond action boundaries. In an action recognition setting, an LSTM network has little access to the longer-term temporal context. However, in fine-grained action detection, videos are typically on the order of minutes or hours. Thus, LSTM networks are more suited to this task as they are designed to model long-term temporal dynamics in a sequence.

Action recognition involves assigning a single label to a video sequence, while in action detection we need to assign a label per frame. Consider a video sequence consisting of 100 frames where each frame has the same label. Even if a 2-stream network predicts correct labels only for 50 frames, it is likely that it would assign the correct label to the complete video sequence in the task of action recognition. For action detection, however, if the 50 correctly predicted frames are not contiguous, it would generate many action segments, and all but one of them would be assigned as false positives. A bi-directional LSTM (B-LSTM) on top of a 2-stream network would be more likely to produce contiguous action segments, and would thus have fewer false positives for action detection when compared to a 2-stream network (see Figure 5). However, in such cases, B-LSTM would not show any improvement over a 2-stream network for action recognition, because recognition performance does not change even if the predicted labels are fragmented.

Bi-directional LSTM networks [6], illustrated in Figure 4, integrate information from the future as well as the past to make a prediction for each chunk in the video sequence. Therefore, they are expected to be better at predicting the temporal boundaries of an action as compared to a unidirectional LSTM. In this work, the forward and backward LSTM networks each give softmax scores for each action class, and we average the softmax predictions of the two LSTM networks to obtain the score for each action. While training these networks on long sequences, back-propagation through time can only be done up to a fixed number of steps, using a short sequence of chunks. To preserve long-term context, we retain the hidden state of the last element in the previous sequence when training on the subsequent sequence.

4. Results

4.1. Datasets

We evaluate our method on two datasets: the MPII Cooking 2 Dataset [21], and the new MERL Shopping Dataset that we collected and are releasing to the community. The MPII Cooking 2 Dataset consists of 273 video sequences that vary in length from 40 seconds to 40 minutes, with a total of 2.8 million frames. The videos are labeled with the start and end times of fine-grained actions from 67 action classes (6 of them are not part of the test set). Actions such as "smell," "screw open," and "take out" may be as brief as one-half of a second, while actions such as "grate," "peel," and "stir" can last as long as a few minutes. There is also significant intra-class variation in the duration of an action.

Our MERL Shopping Dataset consists of 96 two-minute videos, shot by a static overhead HD camera, of people shopping from grocery-store shelving units that we set up in a lab space. There are 32 subjects, each of whom is in 3 videos collected on different days. Videos are labeled with the start and end times of fine-grained actions from 5 different action classes: "Reach to Shelf," "Retract from Shelf," "Hand in Shelf," "Inspect Product," and "Inspect Shelf." We divide this dataset into three partitions: 60 training videos, 9 validation videos, and 27 test videos. For each subject, all

Figure 6. Images for various actions in the MERL Shopping Dataset. We show images corresponding to different actions such as "retract from shelf," "inspect product," "hand in shelf," and "inspect shelf."

three videos of that subject are in only one of the three partitions. Although the number of videos in this dataset is less than in MPII Cooking 2, there are many action instances per video, so the number of frames per action class is high ($\sim$30,000). In this dataset, the duration of an action ranges from one-half of a second to on the order of a minute. We show examples of frames from this dataset in Figure 6. The version we make public will have more videos, as well as revised labels for existing videos. We will release our results for the larger version along with the dataset.

Although our algorithm performs both temporal and spatial localization of actions, only temporal accuracy is evaluated in order to be consistent with the MPII Cooking 2 evaluation protocol.

4.2. Implementation Details

We sample each video at 15 frames per second. We then extract optical flow between each frame (sampled every 6 frames) and its 6 neighboring frames ($K = 3$ each to the left and right). This provides pixel trajectories for each pixel. Epic flow is used to compute optical flow [19], as it gives reliable flow even for large movements. We then use our tracker to obtain bounding boxes for each video. Finally, all full-size image frames and cropped frames are resized to 256×256. Pixel trajectories are resized to 224×224. For training of frame-based networks (the appearance stream), we fine-tune VGG net [23] using Caffe [12]. From each 6-frame chunk of video, we use a single image frame for the appearance streams. We encode two sets of 6 optical flow fields (one stack each for x- and y-direction) as pixel trajectories for the motion stream. While training motion networks, we change the conv_1 filter of VGG to a $1 \times 2K$ kernel, which only performs convolution in time. We project the four fc7 layers of the multi-stream network using a fully connected layer to a 200-dimensional vector. This 200-dimensional vector is given to two LSTM layers (one forward and one backward in time) with 60 hidden units each. Finally, a softmax classifier is trained on each LSTM's hidden units, and the softmax predictions of both LSTM networks are averaged to get the action scores for each class. While training LSTMs for detection, we use the entire video sequence, so this also includes a background class. We use the same architecture for both datasets. Since the four networks that make up our multi-stream network cannot all fit in GPU (Tesla K40) memory at once, we train each network independently. To train the LSTM networks, we use the implementation provided in [3].

Since mean Average Precision (mAP) is the standard measure used to evaluate action detection in past work, we need to produce a ranked list of action clips, along with a start frame, an end frame, and a score associated with each clip. Midpoint hit criterion is used to evaluate detection as done in [21]. This means that the midpoint of the detected interval should lie within the ground-truth interval in the test video. If a second detection fires within the same ground-truth interval, that second detection is considered a false positive. We use the evaluation code used in [20].

To obtain segments for each action class, we start with an initial threshold. We apply this threshold to the output score (average of the two LSTM softmax outputs) that was assigned to each 6-frame chunk of video by our MSB-RNN network. We group the above-threshold chunks into connected components, each of which represents one detection, which we refer to as a clip (defined by its start and end time). The initial threshold will give us some number of detections. If the number of detections is less than m for a class, we lower the threshold further until we get m unique clips. To get the next set of clips, we lower the threshold until we get $2m$ unique clips. If a new action clip intersects with any clip in the previous set, we discard the new clip. We keep on doubling the size of the next set until we obtain 2500 unique clips. In our experiments, we set $m = 5$. Each clip consists of some number of consecutive 6-frame chunks of video, each of which is assigned an output score (average of the two LSTM softmax outputs) by our MSB-RNN system. We assign a score to each clip by max-pooling the output scores of all of the chunks in the clip. Since the validation set in the MPII Cooking 2 Dataset does not contain every action class in the dataset, we adopt this method because it enables us to obtain a ranked list of detections without requiring us to select detection thresholds for each action class. We use the same process on the MERL Shop-

Method	mAP
Hand-cSIFT [21]	10.5%
Hand-trajectories [21]	21.3%
Hand-cSIFT+Hand-trajectories [21]	26.0%
Dense Trajectories [27, 21]	29.5%
Two-Stream Network [22]	30.18%
DT+Hand-trajectories+cSIFT [21]	34.5%
MSB-RNN	41.2%

Table 1. Comparison of performance of our MSB-RNN system with previous action detection methods on the MPII Cooking 2 dataset. Mean Average Precision (mAP) is reported.

ping Dataset. We replicate the output labels for each chunk 6 times to get per-frame labels for evaluation. The above process is similar to non-maximal suppression, as we rank confident detections at the top and do not include less confident detections in the ranked list whose subsets have already been detected.

4.3. Experiments

In Table 1 we show that our MSB-RNN obtains an mAP of 41.2% on the MPII Cooking 2 Dataset, outperforming the previous state of the art's mAP of 34.5%. Note that the 34.5% reported in [21] is a very strong baseline. Dense trajectories are still known to give state-of-the-art performance in fine-grained action recognition and detection, and [21] uses a combination of dense trajectories along with the additional hand-centric color-SIFT and Hand-trajectories features. Our implementation of the two-stream network [22] (just our two full-frame streams, without our person-centric streams, and without the LSTMs) yields an mAP of 30.18% on this dataset, which is only slightly better than the performance of using improved dense trajectories alone.

Pixel Trajectories

In Table 2, we compare the effectiveness of variations of the person-centric (cropped) appearance stream ("Frame"), and the person-centric motion stream using either pixel trajectories ("Trajectories") or stacked optical flow ("Stacked OF"). We evaluate mAP for two versions of each network: when the stream is followed by a unidirectional LSTM layer, and when the LSTM is omitted and replaced by a softmax layer. Using pixel trajectories instead of stacked flow improves performance both with and without LSTM, on both the MPII Cooking 2 (MPII 2) and MERL Shopping (Shop) datasets, making pixel trajectories a clear winner over stacked flow for action detection. For all three types of streams on both datasets, the LSTM layer produces a large improvement.

Multi-Stream Network

In Table 3 we compare the performance of our multi-stream network (using both full-frame and person-centric bounding boxes) with that of a two-stream network (full-frame only). Including a person-centric reference frame improves performance on both datasets. If we use B-LSTM only on full-frame features (Two-Stream + BLSTM, not shown in result tables), the performance drops by 3.4% on the MERL Shopping Dataset and 1.8% on the MPII Cooking 2 dataset compared to MSB-RNN. We report results for individual actions of the Shopping dataset in Table 4.

Method	MPII2	MPII2 LSTM	Shop	Shop LSTM
Stacked OF	21.31%	27.36%	55.29%	71.70%
Trajectories	22.35%	29.51%	57.88%	73.06%
Frame	24.72%	28.77%	40.02%	63.26%

Table 2. Evaluating individual components of our MSB-RNN system. Mean average Precision (mAP) is reported. For both datasets, MPII Cooking 2 and Shopping dataset, pixel trajectories outperform stacked flow (both with and without a subsequent LSTM layer). For all three stream types and both datasets, incorporating the LSTM layer greatly improves performance.

Method	MPII 2	Shop
Two-Stream [22]	30.18%	65.21%
Multi-Stream	33.38%	69.08%
Multi-Stream LSTM $\rightarrow$	38.03%	77.24%
Multi-Stream LSTM $\leftarrow$	37.43%	75.08%
MSB-RNN	41.22%	80.31%

Table 3. Performance comparison of multi-stream vs. two-stream network. Performance when multi-stream network is followed by each unidirectional LSTM or by their bi-directional combination (MSB-RNN). mAP is reported.

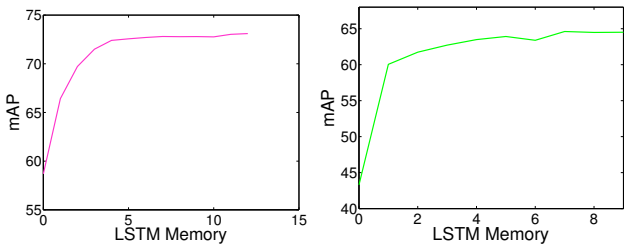

Figure 7. Mean Average Precision (mAP) for pixel trajectory network (*left*) and frame network (*right*) on the Shopping Dataset, when followed by a unidirectional LSTM layer with restricted memory duration. LSTM memory duration (x-axis) is expressed as a number of 6-frame chunks. Note that the LSTM network can effectively remember as far as 10 chunks (4 seconds) into the past.

LSTM

Tables 3 and 4 also compare the performance of our multi-stream network when followed by a forward unidirectional LSTM, a backward unidirectional LSTM, or the bi-directional LSTM (which is our complete MSB-RNN system). Each unidirectional LSTM provides a significant

Action	Two Stream	MSN	LSTM ←	LSTM →	MSB-RNN
Reach To Shelf	75.95%	80.8%	84.39%	84.86%	89.74%
Retract From Shelf	74.53%	77.71%	81.45%	84.61%	90.47%
Hand In Shelf	52.48%	54.13%	59.69%	68.73%	65.56%
Inspect Product	67.56%	75.68%	79.29%	78.6%	82.7%
Inspect Shelf	55.52%	57.09%	70.57%	70.31%	73.09%
Mean	65.21%	69.08%	75.08%	77.42%	80.31%

Table 4. Results for each action class in the Shopping Dataset using various network configurations.

boost, and including bi-directional LSTMs (MSB-RNN) yields an even larger improvement, because it provides more temporal context than a unidirectional LSTM. The results in these tables and Table 2 clearly show that the LSTM layer is the most important factor contributing to our system's improved performance over previous methods.

These observations led us to explore in more detail why using an LSTM layer improves performance by such a large margin. We conducted two experiments to analyze the contributions of an LSTM layer to our system.

How long does the LSTM remember?

In the first experiment, we use the trained model and analyze the effective temporal duration of the LSTM layer's memory. For this experiment, we clear the memory of the LSTM layer at different time steps using a continuation sequence indicator. A continuation sequence indicator is 0 at the beginning of a sequence and 1 otherwise. Thus, we can set every kth indicator to 0 for clearing the memory, if we are interested in an LSTM which remembers history from the past k chunks in the sequence. However, this would abruptly reduce the memory of the element at the beginning of the sequence to zero. To avoid this problem, we generate k different continuation indicator sequences, shifted by one element, to limit the memory of the LSTM layer to k time steps. Thus, when a prediction is made for an element, we choose the output from the sequence whose continuation indicator was set to zero k time steps before. In Figure 7, we plot the mAP when an LSTM is used on top of frame or pixel trajectory features on the Shopping Dataset. We observe that performance improves as we increase the memory duration for LSTM. It is quite encouraging that the unidirectional LSTM layer can make effective use of as many as 10 preceding chunks in a sequence. Thus, a bi-directional LSTM would use a context of 20 chunks in a sequence while making a prediction. In the context of a video, where each chunk comprises 6 frames of a video (sampled at 15 frames per second), this sequence length would correspond to 8 seconds. Thus, the bi-directional LSTM improves action detection performance by a large margin, by incorporating information from about 8 seconds of temporal context. Many actions last less than 8 seconds, and actions that last longer than that are likely to have a recurring pattern that can be captured in 8 seconds.

Learning transitions between actions

The first experiment (above) demonstrates that an LSTM can remember long-term temporal information. In the second experiment, we explore whether the LSTM can also learn information from the transitions between different actions in a video sequence. Recent works train an LSTM network on trimmed video sequences [3, 15]. Thus, they cannot learn long-term context that extends beyond the start or end of an action. Therefore, we conducted our second experiment, in which the continuation indicators are set to 0 (while training only) whenever an action starts or ends. This simulates training on trimmed video sequences, instead of a continuous video sequence that includes many actions. We observe that training on trimmed clips drops the performance from 77.24% to 75.51% on the Shopping Dataset and from 38.03% to 36.22% on the MPII Cooking 2 dataset (using a unidirectional LSTM). This confirms our hypothesis that training networks on long video sequences is beneficial as compared to training on temporally clipped videos of individual actions.

5. Conclusion

In this paper, we showed that using a multi-stream network that augments full-frame image features with features from a bounding box surrounding the actor is useful in fine-grained action detection. We showed that for this task, pixel trajectories give better results than stacked optical flow due to their location correspondence. We showed that to capture long-term temporal dynamics within and between actions, a bi-directional LSTM is highly effective. We also provided an analysis of how long an LSTM network can remember information in the action detection scenario. Finally, our results represent a significant step forward in accuracy on a difficult publicly available dataset (MPII Cooking 2), as well as on a new MERL Shopping Dataset that we are releasing with the publication of this paper.

Acknowledgement

The authors acknowledge the University of Maryland supercomputing resources http://www.it.umd.edu/hpcc made available for conducting the research reported in this paper.

References

[1] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt. Sequential deep learning for human action recognition. In *Human Behavior Understanding*, pages 29–39. Springer, 2011. 3

[2] R. Cutler and L. S. Davis. Robust real-time periodic motion detection, analysis, and applications. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):781–796, 2000. 2

[3] J. Donahue, L. A. Hendricks, S. Guadarrama, M. Rohrbach, S. Venugopalan, K. Saenko, and T. Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3, 6, 8

[4] Y. Du, W. Wang, and L. Wang. Hierarchical recurrent neural network for skeleton based action recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1110–1118, 2015. 3

[5] G. Gkioxari and J. Malik. Finding action tubes. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 759–768, 2015. 2

[6] A. Graves, S. Fernández, and J. Schmidhuber. Bidirectional lstm networks for improved phoneme classification and recognition. In *Artificial Neural Networks: Formal Models and Their Applications (ICANN)*, pages 799–804. Springer, 2005. 1, 5

[7] A. Graves, A.-r. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6645–6649, 2013. 4

[8] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1, 4

[9] Y. Hu, L. Cao, F. Lv, S. Yan, Y. Gong, and T. S. Huang. Action detection in complex scenes with spatial and temporal ambiguities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 128–135, 2009. 2

[10] A. Jain, A. Gupta, M. Rodriguez, and L. S. Davis. Representing videos using mid-level discriminative patches. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2571–2578, 2013. 2

[11] M. Jain, J. Van Gemert, H. Jégou, P. Bouthemy, and C. G. Snoek. Action localization with tubelets from motion. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 740–747, 2014. 2

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *Proceedings of the 22nd ACM International Conference on Multimedia*, pages 675–678, 2014. 6

[13] H. Kuehne, A. Arslan, and T. Serre. The language of actions: Recovering the syntax and semantics of goal-directed human activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 780–787, 2014. 3

[14] V. Mnih, N. Heess, A. Graves, et al. Recurrent models of visual attention. In *Advances in Neural Information Processing Systems*, pages 2204–2212, 2014. 3

[15] J. Y.-H. Ng, M. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 3, 8

[16] B. Ni, V. R. Paramathayalan, and P. Moulin. Multiple granularity analysis for fine-grained action detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 756–763, 2014. 3

[17] X. Peng, L. Wang, X. Wang, and Y. Qiao. Bag of visual words and fusion methods for action recognition: Comprehensive study and good practice. *arXiv preprint arXiv:1405.4506*, 2014. 2, 3

[18] H. Pirsiavash and D. Ramanan. Parsing videos of actions with segmental grammars. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 612–619, 2014. 3

[19] J. Revaud, P. Weinzaepfel, Z. Harchaoui, and C. Schmid. EpicFlow: Edge-Preserving Interpolation of Correspondences for Optical Flow. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015. 6

[20] M. Rohrbach, S. Amin, M. Andriluka, and B. Schiele. A database for fine grained activity detection of cooking activities. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1194–1201, 2012. 1, 6

[21] M. Rohrbach, A. Rohrbach, M. Regneri, S. Amin, M. Andriluka, M. Pinkal, and B. Schiele. Recognizing fine-grained and composite activities using hand-centric features and script data. *International Journal of Computer Vision*, 2015. 2, 3, 5, 6, 7

[22] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in Neural Information Processing Systems*, pages 568–576, 2014. 1, 2, 3, 4, 7

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3, 6

[24] C. Sun, S. Shetty, R. Sukthankar, and R. Nevatia. Temporal localization of fine-grained actions in videos by domain transfer from web images. In *Proceedings of the 23rd ACM Conference on Multimedia*, pages 371–380, 2015. 3

[25] Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2642–2649, 2013. 2

[26] P. Viola, M. J. Jones, and D. Snow. Detecting pedestrians using patterns of motion and appearance. In *IEEE 9th International Conference on Computer Vision (ICCV)*, pages 734–741, 2003. 2

[27] H. Wang, A. Kläser, C. Schmid, and C.-L. Liu. Dense trajectories and motion boundary descriptors for action recognition. *International Journal of Computer Vision*, 103(1):60–79, 2013. 7

[28] H. Wang and C. Schmid. Action recognition with improved trajectories. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3551–3558, 2013. 2

[29] L. Wang, Y. Qiao, and X. Tang. Action recognition with trajectory-pooled deep-convolutional descriptors. In *IEEE*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 4305–4314, 2015. 2, 3, 4

[30] L. Wang, Y. Xiong, Z. Wang, and Y. Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015. 4

[31] K. Xu, J. Ba, R. Kiros, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *Proceedings of the 32nd International Conference on Machine Learning*, 2015. 3

[32] G. Yu and J. Yuan. Fast action proposals for human action detection and search. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1302–1311, 2015. 2

[33] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2442–2449, 2009. 2