

2016

Learning activity progression in LSTMs for activity detection and early detection

S Ma, L Sigal, S Sclaroff. 2016. "Learning Activity Progression in LSTMs for Activity Detection and Early Detection." Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR).
<https://hdl.handle.net/2144/26690>

"Downloaded from OpenBU. Boston University's institutional repository."

Learning Activity Progression in LSTMs for Activity Detection and Early Detection

Shugao Ma
Boston University
shugaoma@bu.edu

Leonid Sigal
Disney Research
lsigal@disneyresearch.com

Stan Sclaroff
Boston University
sclaroff@bu.edu

Abstract

In this work we improve training of temporal deep models to better learn activity progression for activity detection and early detection tasks. Conventionally, when training a Recurrent Neural Network, specifically a Long Short Term Memory (LSTM) model, the training loss only considers classification error. However, we argue that the detection score of the correct activity category, or the detection score margin between the correct and incorrect categories, should be monotonically non-decreasing as the model observes more of the activity. We design novel ranking losses that directly penalize the model on violation of such monotonicities, which are used together with classification loss in training of LSTM models. Evaluation on ActivityNet shows significant benefits of the proposed ranking losses in both activity detection and early detection tasks.

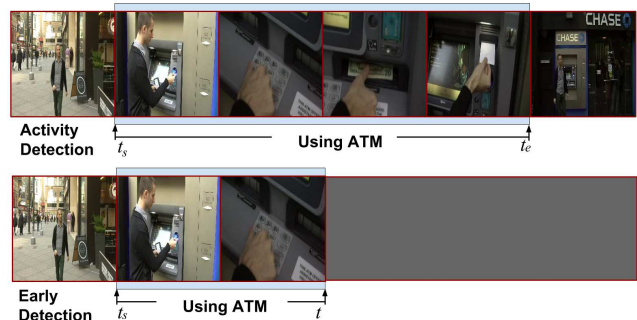


Figure 1: We study two problems: activity detection and early detection. For activity detection, we detect the category of the activity and its start and end point. For early detection, we need to detect the category and the start point of an activity after observing only a fraction of the activity. This example sequence contains the activity *using ATM*.

1. Introduction

In this work we study human activity detection and early detection in videos (Fig. 1). For activity detection, we detect segments of human activities in a video sequence, recognizing the activities’ categories and detecting their start and end points. For early detection, we detect the activity segment after observing only a fraction of the activity.

Automatic detection of human activities, in videos, has many potential applications, such as video understanding and retrieval, automatic video surveillance, and human-computer interaction. Further, for many applications, such as human-robot interaction it is desirable to detect the activity as early as possible [5, 19], to make the interaction more natural, e.g., deploying a robot to help an elderly patient stand up before he/she is upright and is at risk of a fall.

Activity detection in realistic settings is quite challenging. There is high variability in the viewpoint from which the activity is observed, the actors and their appearance, as well as the execution and overall duration of the activities (see Fig. 6). This is particularly true for relatively long

and complex activities. For example, the activity “making pasta” typically entails *cutting vegetables, setting a pot on the fire, boiling water, boiling pasta noodles, cooking pasta sauce, and combining pasta with sauce*. To better detect, i.e., recognize and temporally localize such activities, we argue that it is critically important for the learned detector to model the activities’ temporal progression.

Recurrent Neural Network (RNN) models are particularly helpful in this context: the prediction at each time instant is based not only on the observations at that time instant, but also on the previous model hidden states that provide temporal context for the progression of the activity. More specifically, in the Long Short Term Memory (LSTM), a type of RNN, *memory* is used to capture useful patterns of previous observations, and is used in addition to the previous hidden states to provide longer-range context (e.g., as compared to HMMs) for the current prediction.

While RNN models are powerful, using only classification loss in training such models typically fails to properly penalize incorrect predictions, i.e., the prediction error is penalized the same no matter how much context the model

has already processed. For example, given a video of the activity *making pasta*, to output the activity class label *preparing coffee* after the detector sees the activity up to *combining pasta with sauce* should be penalized more than the same error when the detector only sees activity up to *boiling water*.

The above mentioned defect in training RNN models is especially critical for activity detection. Unlike conventional applications of RNNs in machine translation and speech recognition, in which specific output such as words or phonemes continue for a relatively short time, human activities such as *making pasta* may continue for a relatively long period, *e.g.*, several minutes or thousands of video frames. It is thus very important for the model to learn the progression patterns of the activities in training.

In this work, we introduce novel *ranking losses* within the RNN learning objective so that the trained model better captures progression of activities. These ranking losses are computed for the prediction at each time point, while also taking into consideration the past predictions starting from the very beginning of the activity.

The intuition for our formulation is shown in Fig. 2. As the detector sees more of an activity, it should: (1) become more confident of the correct activity category, *i.e.*, output a higher detection score for the correct category as the action progresses, and (2) become more confident of the absence of incorrect categories, *i.e.*, the detection score margin between the correct and incorrect categories should be non-decreasing as the action progresses.

Thus, we introduce two explicit constraints in RNN training. The first is a ranking loss on the detection score of the correct category, which constrains the detection score of the correct category to be monotonically non-decreasing as the activity progress. The second is a ranking loss on the detection score margin between the correct activity category and all other categories, which constrains that this discriminative margin is monotonically non-decreasing.

In summary, we make the following contributions:

- We propose formulations for ranking loss on the detection score and on the discriminative margin to better learn models for human activity progression.
- We implement our proposed ranking losses in training LSTM models, and show significant improvements over LSTM model trained only with classification loss in the tasks of activity detection and early detection.
- We achieve start-of-the-art performance for activity detection and early detection on a large-scale video dataset: ActivityNet [4].

2. Related Work

A topic closely related to human activity detection is human action recognition. In action recognition, the input video clip is (manually) trimmed so that it only contains

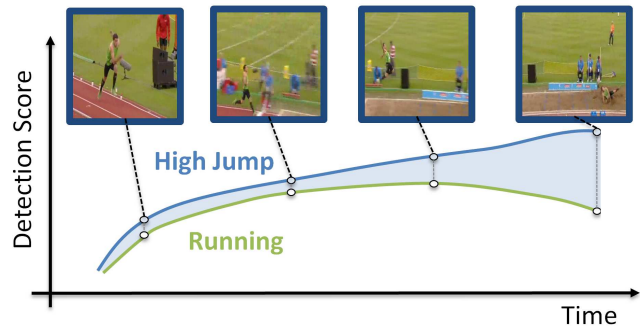


Figure 2: As the detector sees more of the activity, it should become more confident of the presence of the correct category and absence of incorrect categories. This example sequence contains a *high jump*. The blue curve is the detection score of the correct category, which is encouraged to be non-decreasing. The green curve is the detection score of an incorrect category *running*, whose margin with respect to the correct category (shaded light blue area) is encouraged to be non-decreasing.

video frames depicting a human action, and the goal is to correctly recognize the action category. Many past works focus on this topic, *e.g.*, encoding video clips using a bag-of-words representation over local space-time features and training SVM classifiers [11, 13, 23], or modeling human actions as space-time structures [1, 10, 12, 17, 24, 25]. In [6, 7], Convolutional Neural Networks (CNNs) with space-time convolutional filters are trained to capture space-time patterns from training videos. In [20], separate CNNs are trained for a spatial stream (*i.e.*, video frames) and motion stream (*i.e.*, optical flow fields) and the features from both CNNs are concatenated to train an action classifier. LSTM models are explored in [14, 26] for recognizing human actions. In contrast to these works, we not only recognize activities but also detect their start and end time points.

Human action detection is also a well studied problem. In [8], simple actions are represented as space-time shapes that are matched against over-segmented space-time video volumes. In [28], action detection entailed searching for 3D subvolumes of space-time invariant points. In [10, 22], human actions are modeled as space-time structures, using deformable part models [2]. In [15, 18] discriminative hand-centric features are explored for fine grained activity detection in cooking, *i.e.*, relatively short sub-activities such as *chop* and *fill*. In [3], the detector is trained on CNN features extracted from the action tubes in space-time; however, evaluation is on relatively short video clips (*i.e.*, several hundred frames) of relatively short actions. In [27] an LSTM is trained that takes CNN features of multiple neighboring frames as input to detect actions at every frame; while their model is similar to ours, they focus on detecting

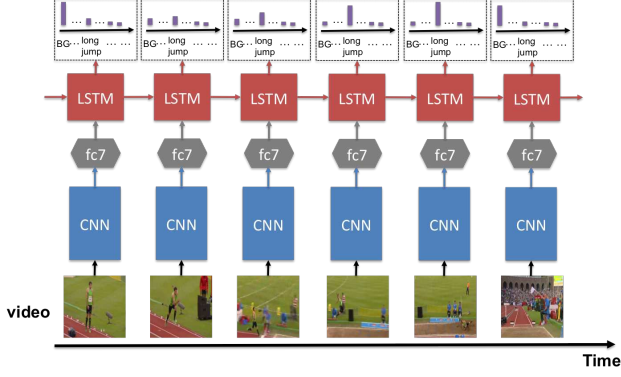


Figure 3: **Model overview.** At each video frame, the model first computes CNN features (illustrated as *fc7*) and then the features are fed into the LSTM to compute detection scores of activities and non-activity (*BG* in the figure).

simple actions such as *stand up* that last only for a few video frames, and the training loss accounts only for classification errors. In this work, we focus on accurately localizing activities that are long and complex by learning and enforcing activity progression as part of LSTM learning objective.

Early recognition of human action or activities, *i.e.*, recognizing human actions or activities given partial observations, has also been studied in previous works *e.g.*, by using dynamic bag-of-words of space-time features [19], by modeling actions as a sparse sequence of key-frames [17], or by using compositional kernels to hierarchically capture relationships between partial observations [9]. In [5] a structured output SVM is used for recognizing and also temporally localizing events given partial observations. Compared to [5], which is evaluated on lab collected videos of simple human actions, we use deep learning techniques to solve this problem on large scale realistic video dataset of human activities which are often long and complex.

3. Model Overview

Fig. 3 illustrates our model for activity detection. It contains two major components: a CNN that computes visual features from each video frame, and an LSTM with a linear layer that computes activity detection scores based on the CNN features of the current frame and the hidden states and memory of the LSTM from the previous time step. We adopt the VGG19 [21] CNN architecture, whose output of the second fully connected layer (*fc7*) is fed into the LSTM. We use the LSTM described in [16] that applies dropout on non-recurrent connections. A similar model has been used in [27] for detecting relatively short actions. Our key contributions are in exploring the rank losses, during training, that encourage monotonicity in detection score and margin produced by the model as a training activity progresses.

4. Learning Activity Progression

To accurately detect the complete duration of human activities, especially for relatively long and complex ones, it is important for the model to capture the progression patterns of activities during training. An RNN only implicitly considers progression via the context that is passed along time in the form of the previous hidden state and, in LSTM, memory as well. We introduce ranking loss into the learning objective, to explicitly capture activity progression globally from the activity start to the current time:

$$\mathcal{L}^t = \mathcal{L}_c^t + \lambda_r \mathcal{L}_r^t, \quad (1)$$

where \mathcal{L}_c^t and \mathcal{L}_r^t are the classification loss and the ranking loss, respectively, at the current time t , and λ_r is a positive scalar constant that controls relative term contribution.

Usually, for training deep models, the cross entropy loss is used to formulate \mathcal{L}_c^t :

$$\mathcal{L}_c^t = -\log p_t^{y_t}, \quad (2)$$

where y_t is the ground truth activity category of the training video sequence at the t -th video frame, and $p_t^{y_t}$ is the detection score of the ground truth label y_t for the t -th frame, *i.e.*, the softmax output of the model.

We explore two formulations of the ranking loss, \mathcal{L}_r^t . The first constrains the model to output a non-decreasing detection score for the correct category throughout the duration of the activity. Our second ranking loss constrains the output of the model to have non-decreasing discriminative margin: at any point in the activity, the margin between the detection score of the correct category and the maximum detection score among all other categories should be non-decreasing. Detailed formulations of these two ranking losses are given below. For easier reading, we use \mathcal{L}_s^t and \mathcal{L}_m^t to denote ranking loss on the detection score and margin, respectively. While these two ranking losses are different, they are related. Note that the output of softmax layer of the LSTM sums to 1, so \mathcal{L}_s^t considers the margin between $p_t^{y_t}$ and $\sum_{y' \neq y_t} p_t^{y'}$, whereas \mathcal{L}_m^t considers the margin between $p_t^{y_t}$ and $\max_{y' \neq y_t} p_t^{y'}$. We will discuss this more at the end of Section 5.6.

4.1. Ranking Loss on Detection Score

Ideally we want the activity detector to produce monotonically non-decreasing detection scores for the correct activity category as the detector sees more of the activity (Fig. 2). To this end, we introduce the ranking loss \mathcal{L}_s^t into the learning objective at time step t as:

$$\mathcal{L}_s^t = \max(0, -\delta_t \cdot (p_t^{y_{t-1}} - p_t^{*})), \quad (3)$$

where δ_t is set to 1 if $y_{t-1} = y_t$, *i.e.*, when there is no activity transition from $t-1$ to t according to ground truth

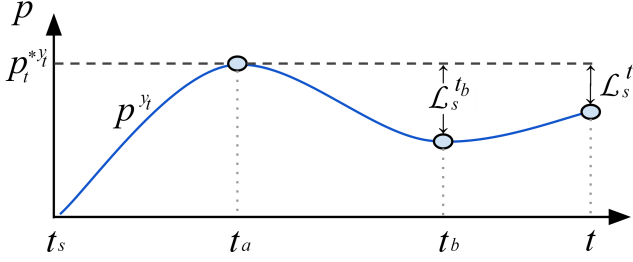


Figure 4: Detection score p^{y_t} (blue curve) of an activity y_t spanning $[t_s, t]$. $p_{t_b}^{y_t}$ and $p_t^{y_t}$ are smaller than $p_{t_a}^{y_t}$ (which is also $p_t^{*y_t}$ in this example), violating the monotonicity of the detection score, so $\mathcal{L}_s^{t_b}$ and \mathcal{L}_s^t are non-zero.

labeling (e.g. $\delta_t = 1$ for t_a, t_b and t in Fig. 4); otherwise, δ_t is set to -1 .

In Eq. (3) p_t^* is computed as:

$$p_t^* = \begin{cases} p_t^{*y_t}, & \text{if } \delta_t = 1, \\ 0, & \text{otherwise,} \end{cases} \quad (4)$$

where

$$p_t^{*y_t} = \max_{t' \in [t_s, t-1]} p_{t'}^{y_t}. \quad (5)$$

$$t_s = \min\{t' \mid y_{t'} = y_t, \forall t' \in [t_s, t]\}, \quad (6)$$

where t_s is the starting point of the current activity y_t , and $p_t^{*y_t}$ is the highest previous detection score in $[t_s, t-1]$ (illustrated by the dashed horizontal line in Fig. 4).

In other words, if there is no activity transition at time t , i.e., $y_t = y_{t-1}$, then we want the current detection score to be no less than any previous detection score for the same activity, computing the ranking loss as:

$$\mathcal{L}_s^t = \max(0, p_t^{*y_t} - p_t^{y_t}). \quad (7)$$

On the other hand, if an activity transition happens at time t , i.e., $y_t \neq y_{t-1}$, we want the detection score of the previous activity to drop to zero at t and compute the ranking loss as:

$$\mathcal{L}_s^t = p_t^{y_{t-1}}. \quad (8)$$

Fig. 4 shows the detection scores p^{y_t} (the blue curve) of an activity y_t spanning $[t_s, t]$. In $[t_a + 1, t]$, the detection scores are smaller than $p_{t_a}^{y_t}$, violating the monotonicity of the detection score, so the ranking losses in this period are non-zero, e.g. $\mathcal{L}_s^{t_b}$ and \mathcal{L}_s^t as shown in the figure.

One may be tempted to simply require $p_t^{y_t}$ to be no less than $p_{t-1}^{y_t}$ when there is no activity transition, replacing Eq. 7 with:

$$\mathcal{L}_s^t = \max(0, p_{t-1}^{y_t} - p_t^{y_t}). \quad (9)$$

However, as shown in Fig. 4, in this situation, the ranking loss will be zero in $[t_b + 1, t_c]$ even though the monotonicity of detection score is also violated.

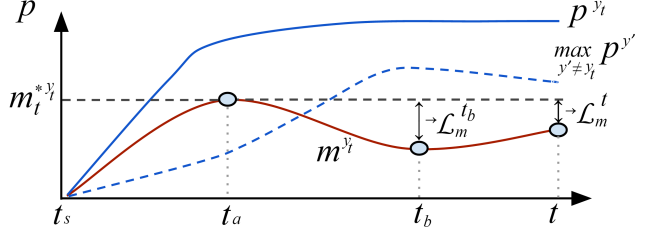


Figure 5: Discriminative margin m^{y_t} (red curve) of an activity y_t spanning $[t_s, t]$. The margin m^{y_t} is computed as the difference between the ground truth activity detection scores p^{y_t} (blue curve) and the maximum detection scores $\max_{y' \neq y_t} p^{y'}$ (dashed blue curve) of all incorrect activity categories at each time point in $[t_s, t]$. $m_{t_b}^{y_t}$ and $m_t^{y_t}$ are smaller than $m_{t_a}^{y_t}$ (which is also $m_t^{*y_t}$), violating the monotonicity of the margin, so $\mathcal{L}_m^{t_b}$ and \mathcal{L}_m^t are non-zero.

4.2. Ranking Loss on Discriminative Margin

When more of an activity is observed, the detector should become more confident in discriminating between the correct category vs. the incorrect categories. We guide the training of our model to achieve such behavior by implementing the following ranking loss:

$$\mathcal{L}_m^t = \max(0, -\delta_t \cdot (m_t^{y_{t-1}} - m_t^*)). \quad (10)$$

where m_t^y is the discriminative margin of an activity label y at time step t (the blue point on the red curve at time t in Fig. 5), computed as:

$$m_t^y = p_t^y - \max\{p_t^{y'} \mid \forall y' \in \mathcal{Y}, y' \neq y\}, \quad (11)$$

where \mathcal{Y} is the set of all activity category labels. The m_t^* in Eq. 10 is computed as:

$$m_t^* = \begin{cases} m_t^{*y_t}, & \text{if } \delta_t = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (12)$$

where $m_t^{*y_t}$ is computed as:

$$m_t^{*y_t} = \max_{t' \in [t_s, t-1]} m_{t'}^{y_t}, \quad (13)$$

i.e., the largest previous discriminative margin of the current activity y_t that started at t_s (illustrated by the dashed horizontal line in Fig. 5).

In other words, when there is no activity transition at t , we want the current discriminative margin to be no less than any previous margin in the same activity, computing the ranking loss as:

$$\mathcal{L}_m^t = \max(0, m_t^{*y_t} - m_t^{y_t}). \quad (14)$$

If an activity transition happens at time t , we want the discriminative margin of the previous activity to drop and compute the ranking loss as:

$$\mathcal{L}_m^t = m_t^{y_{t-1}}. \quad (15)$$

Fig. 5 illustrates the discriminative margins (red curve) m^{y_t} of the current activity y_t spanning $[t_s, t]$. The margin m^{y_t} is equal to the difference between the detection scores p^{y_t} of the correct category y_t (blue curve) and the maximum of the detection scores $\max_{y' \neq y_t} p^{y'}$ for the incorrect categories (dashed blue curve). Note that within the time interval $[t_a + 1, t]$, the margins are smaller than m^{y_t} , violating the monotonicity; consequently, the ranking losses are non-zero within the interval $[t_a + 1, t]$. Also note that simply requiring the current margin to be less than that of the previous timestep is insufficient, which will result in zero ranking loss in interval $[t_b + 1, t]$ in Fig. 5.

4.3. Training

In training, we compute the gradient of the ranking loss with respect to the softmax output at each time step:

$$\frac{\partial \mathcal{L}^t}{\partial p_t^y} = \frac{\partial \mathcal{L}_c^t}{\partial p_t^y} + \lambda_r \frac{\partial \mathcal{L}_r^t}{\partial p_t^y} \quad (16)$$

which is then back propagated through time to compute the gradients with respect to the model parameters. At a non-activity frame, if the previous frame has activity, *i.e.*, an activity to non-activity transition happens, the ranking loss at this frame is computed according to Eq. 8 or Eq. 15; otherwise, if the previous frame is also a non-activity frame, the ranking loss is fixed to 0. Although \mathcal{L}_s^t and \mathcal{L}_m^t are also functions of p_t^y for $t' < t$, *i.e.*, the softmax output of previous time steps, to simplify computation, we do not compute and back propagate the gradients of the ranking loss with respect to these variables.

5. Experiments

We evaluate our formulation on a large-scale, realistic activity dataset: ActivityNet [4]. Using our proposed ranking losses in training significantly improves performance in both the activity detection and early activity detection tasks.

5.1. Dataset

The ActivityNet [4] dataset comprises 28K videos of 203 activity categories collected from YouTube. Fig. 6 shows sample frames from video sequences of this dataset. The lengths of the videos range from several minutes to half an hour. The total length of the whole dataset is 849 hours. A single video may contain multiple activities and often also contains periods with none of the annotated activities. On average, 1.4 activities are annotated per video. The activity category, along with the start and end point of each activity are annotated by crowd-workers, leading to some annotation noise. Many of the videos are shot by amateurs in uncontrolled environments, and variances within the same activity category are often large. More importantly, many activities are relatively long and complex, and the viewpoint



Figure 6: Each row contains sample frames of one example video sequence in ActivityNet. Frames with green borders contain the activities labeled on the left. Note the significant viewpoint and foreground object changes within the activities *Using ATM*, *Sailing* and *Preparing Pasta*.

and foreground objects may change significantly within the same activity, *e.g.*, *Using ATM* and *Preparing pasta* shown in Fig. 6. Given these challenges, it is important that the model learns the progression of activities for accurate activity detection and early detection.

The authors of ActivityNet use one fourth of the dataset as a *validation set*, but have not released the *test set* used in their paper.¹ In our experiments, we use the validation set as our test set, and we split the remaining videos into one fifth for validation and four fifths for training. To reduce computational cost, we temporally down-sample the videos to 6 frames per second for all our experiments.

5.2. Model Training

For the CNN component (see Fig. 3), we first use training video frames of ActivityNet to fine-tune a VGG19 model [21] that is pre-trained on ImageNet. The output dimension of the softmax layer is 204, which corresponds to the 203 activities plus one additional class corresponding to non-activity. We set the learning batch size to 32. The learning rate starts at 10^{-4} and is divided by 10 after every 40K iterations. The fine-tuning stops at 120K iterations.

For LSTM training, the output of the second fully connected layer ($fc7$) of the fine-tuned VGG19 model is used as input to the LSTM. We use learning batches of 64 sequences, where each sequence comprises 100 frames. Back propagation through time is performed for 20 time steps. The momentum and weight decay are set to 0.9 and 0.0005 respectively. The learning rate starts at 0.01 and is divided

¹According to communication with the authors of [4], this test split is kept confidential for use in a future challenge.

by 10 after every 20K iterations. Training stops after 50K iterations. In this training phase, the CNN $fc7$ layer is also further trained together with the LSTM but with a lower starting learning rate of 10^{-4} .

5.3. Experimental Setup

In testing, we run the model across the whole input sequence and output activity detection scores at each input frame. We reset the LSTM memory whenever the model predicts non-activity, which we find slightly improves performance. For both activity detection and early detection, we detect video segments of activities from an input video sequence. To achieve this, we first classify each video frame to the activity category for which the detection score is the highest at this frame. Note that non-activity is treated simply as a special category. We then find continuous video frame segments that are classified to belong to the same activity category; this produces the initial detection spans. Finally, we iteratively merge the detection spans that are temporally close (less than 20 frames apart in our experiments). The score of each detection is then computed as the mean of the detection scores of all its video frames.

Following [4], we use the mAP (mean average precision) in evaluating performance. A detection is a true positive if: 1) its IOU (intersection-over-union) of temporal duration with a ground truth activity is above the IOU threshold, and 2) its activity label is equal to the ground truth activity label. If multiple detections overlap with the same ground truth activity, only the one with the longest duration is treated as a true positive. All the other detections are false positives. For evaluating performance on early detection, we split each input test sequence into multiple sequences so that each new sequence contains the non-activity segment (if there is any) before a test activity, and a portion of the test activity.

We evaluate the performance of four models: i) the fine-tuned VGG19 CNN model; ii) the LSTM model shown in Fig. 3 trained with the classification loss only (Eq. 2); iii) the LSTM-s model trained with both the classification loss and ranking loss on the detection score (Eq. 3); iv) the LSTM-m model trained with classification loss and ranking loss on the discriminative margin (Eq. 10). In LSTM-s and LSTM-m, the weight for ranking loss (λ_r in Eq. 1) is empirically set to 6, according to performance on our validation set. We find that using a combination of both ranking losses in training offers no further improvement over using just one so we do not include results for this in the paper.

5.4. Activity Detection

Table 1 shows the activity detection performance of the evaluated models under different IOU thresholds, α . The results of Heilbron *et al.* [4] are produced on their test set, which is not publicly available; therefore, their results are not directly comparable to ours. Heilbron *et al.* [4] use a

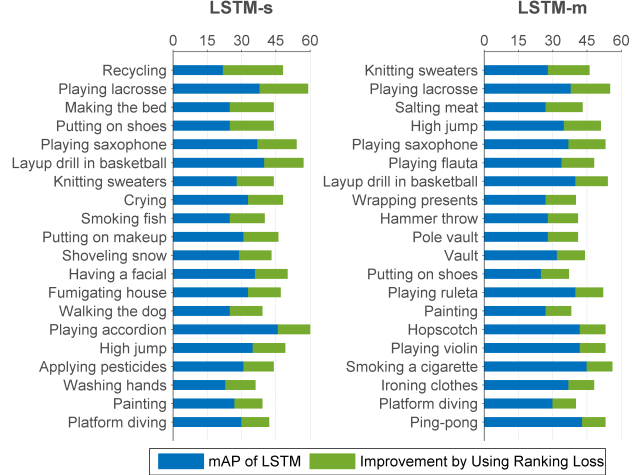


Figure 7: Top 20 activity categories for which the detection performance improved the most by using either LSTM-s or LSTM-m in training.

sliding window approach to detect activities in the video sequences, where the temporal lengths of the sliding windows are empirically selected and fixed. In our approach the length of each detection is automatically determined as described in Section 5.3.

The LSTM models greatly outperform the CNN model. This demonstrates the benefit of using a recurrent neural network model in activity detection. Both of the proposed ranking losses are beneficial in training a better LSTM model for activity detection: significant improvements are achieved over the LSTM model trained only using classification loss. For LSTM-s the improvements are consistently around 4.1~5.9% at all IOU thresholds. Note that the relative improvement of LSTM-m and LSTM-s over LSTM increases when requiring the detection to more accurately overlap with ground truth, *e.g.*, growing from 12.3% when $\alpha = 0.1$ to 16.7% when $\alpha = 0.8$ with LSTM-s. This shows that the proposed ranking-losses are even more useful in applications where accurate temporal localization is required.

Fig. 7 shows the top 20 activities for which the detection performance are improved the most by using ranking loss (LSTM-s or LSTM-m) in training (IOU threshold $\alpha = 0.5$). It is interesting to note that using the proposed ranking losses, detection performance improves both for relatively simple activities such as *playing saxophone* and for relatively complex activities such as *high jump*. This shows that the proposed ranking losses may improve the detection of various types of activities.

5.5. Activity Early Detection

In this experiment, the goal is to recognize and also temporally localize partially observed activities. Table 2 shows the detection performances when we only observe 3/10,

Table 1: Activity detection performance measured in mAP at different IOU thresholds α . Note that the results of Heilbron *et al.* [4] are produced on their own test split that is unavailable to us, so their results are not directly comparable to ours.

Model	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$
Heilbron <i>et al.</i> [4]	12.5%	11.9%	11.1%	10.4%	9.7%	-	-	-
CNN	30.1%	26.9%	23.4%	21.2%	18.9%	17.5%	16.5%	15.8%
LSTM	48.1%	44.3%	40.6%	35.6%	31.3%	28.3%	26.0%	24.6%
LSTM-m	52.6%	48.9%	45.1%	40.1%	35.1%	31.8%	29.1%	27.2%
LSTM-s	54.0%	50.1%	46.3%	41.2%	36.4%	33.0%	30.4%	28.7%

Table 2: Activity early detection performance at different IOU thresholds (α), when *only* 3/10 of each activity is observed.

Model	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	$\alpha = 0.6$	$\alpha = 0.7$	$\alpha = 0.8$
CNN	27.0%	23.4%	20.4%	17.2%	14.6%	12.3%	11.0%	10.3%
LSTM	49.5%	44.7%	38.8%	33.9%	29.6%	25.6%	23.5%	22.4%
LSTM-m	52.6%	47.9%	41.5%	36.2%	31.4%	27.1%	24.8%	23.5%
LSTM-s	55.1%	50.3%	44.0%	38.9%	34.1%	29.8%	27.4%	26.1%

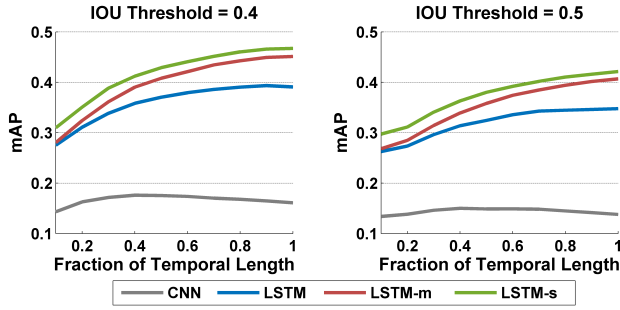


Figure 8: Activity early detection performance plotted as a function of the observed fraction of each test activity.

i.e., approximately the first third, of each testing activity. The LSTM models greatly outperform the CNN model on the early detection task. Moreover, the LSTM models trained with the proposed ranking losses (LSTM-s or LSTM-m) clearly outperform the LSTM model trained only with classification loss. For instance, with LSTM-s, the absolute improvements are consistently around 5.6~3.7% at all IOU thresholds α , with relative improvement increasing from 11.3% at $\alpha = 0.1$ to 16.5% at $\alpha = 0.8$.

Fig. 8 shows the performance of early detection when the observed fraction of each test activity increases from 0.1 to 1 with IOU threshold fixed at 0.4 or 0.5. All LSTM models greatly outperform the CNN, no matter how much of each activity is observed. Both ranking losses, LSTM-m and LSTM-s, outperform LSTM. Although the increase in detection performance slows down after observing approximately half of each activity, the performance gap between LSTM-s (LSTM-m) and LSTM increases as more of each activity is observed. More interestingly, LSTM-s significantly outperforms LSTM even when we only observe a small fraction of each activity, *e.g.*, one tenth. This could be

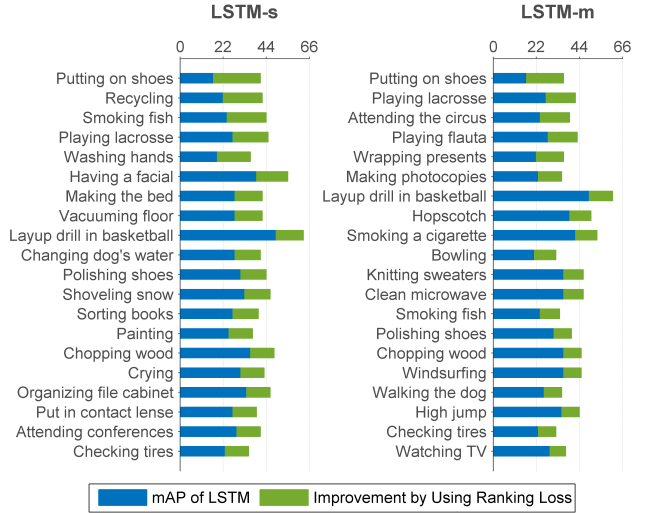


Figure 9: Top 20 activity categories for which the early detection performance improved the most by using either LSTM-s or LSTM-m in training. Only the first 3/10 of each test activity is observed. The IOU threshold $\alpha = 0.5$.

quite useful for applications that require detecting activities as early as possible.

Fig. 9 lists the top 20 activity categories for which the early detection performance improves the most when using either of the proposed ranking losses in training. Interestingly, among these activities, some may have relatively little visual content change across the whole duration of the activity, such as *Playing lacrosse*, whereas others may undergo significant visual content change, such as *Layup drill in basketball*. This suggests that the benefits of the proposed ranking losses are applicable to various types of activities in the task of early detection.

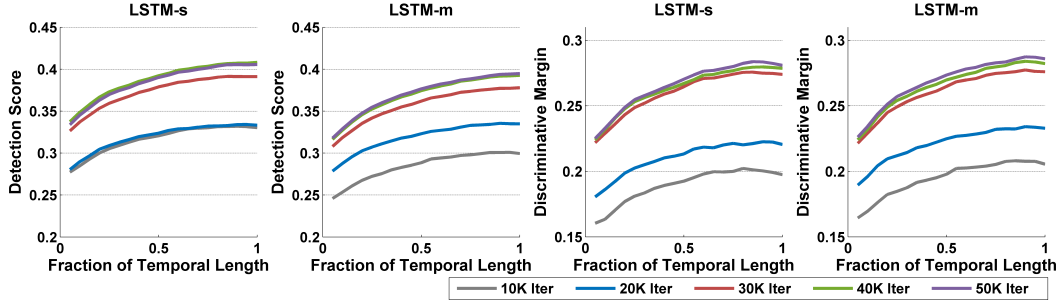


Figure 10: Mean curves of the detection score and the discriminative margin as function of time over all test activity sequences produced by snapshots of the LSTM-m and LSTM-s models trained after 10K, 20K, 30K, 40K and 50K iterations.

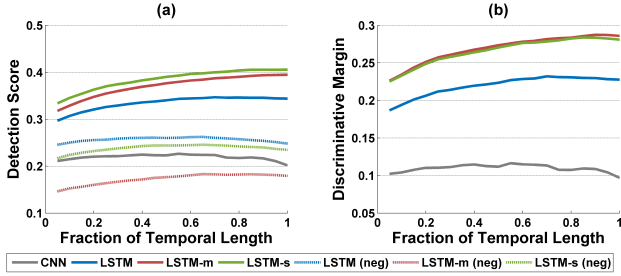


Figure 11: Mean curves of (a) the detection score and (b) the discriminative margin, as function of time over all test sequences for CNN, LSTM, LSTM-m and LSTM-s. The mean curves of the detection score for the *worst negative category*, i.e., negative activity category with the highest detection score, are also shown as the dashed curves.

5.6. Effects of the Ranking Losses

We now analyze what effects the proposed ranking losses introduce over the evolving time scale of model training. We first analyze how the detection score of the correct activity category and the discriminative margin (Eq. 11) change as we train LSTM-m and LSTM-s. We compute the detection scores and the discriminative margins at every frame in each test sequence using snapshots of the LSTM models trained after 10K, 20K, 30K, 40K and 50K iterations. This produces for each activity sequence a curve of the detection score (or discriminative margin) as a function of time. We normalize the curves so that each has a length of 20 points, and finally compute the mean curve over the whole test set. Fig. 10 shows the mean curves. For both models, the mean curves are approximately non-decreasing, and such monotonicity becomes more apparent as we train for more iterations. The absolute values of the detection scores and discriminative margins increase as we train for more iterations, but converge after roughly 40K training iterations.

Fig. 11 compares the mean curves of the detection score and discriminative margin produced by LSTM-s, LSTM-m and LSTM trained after 50K iterations, as well as the

CNN model. The mean curves of LSTM-s and LSTM-m (solid green curves and solid red curves) for both the detection score and discriminative margin are significantly higher than those of LSTM (blue curves). The LSTM-s and LSTM-m curves also show a more apparent monotone increasing trend compared to LSTM, which tends to be flat after approximately the first half of the activity. We also show the mean detection score curves for the *worst negative category*, i.e., the negative activity category with the highest detection score for LSTM, LSTM-s and LSTM-m using the dashed curves. The curves of LSTM-m and LSTM-s for the *worst negative category* are lower than that of LSTM.

It is interesting to note that each of the proposed ranking losses has useful impacts on both the detection score and the discriminative margin, despite the fact that they are either computed based on detection scores only or discriminative margins only. This conforms to our intuition that encouraging a non-decreasing detection score may help in producing a non-decreasing discriminative margin and vice versa. Also note that LSTM-s produces higher detection scores for the correct category than LSTM-m, while LSTM-m pushes the detection scores of the *worst negative category* significantly lower, as shown Fig. 11 (a). In practice one can use either of these ranking loss formulations, depending on the application, e.g., selecting the best one via cross-validation.

6. Conclusion

We improve training of the LSTM model to better learn activity progression. We introduce two novel formulations for ranking loss in LSTM training, designed to encourage consistent scoring and margin for detecting the correct activity as more of the activity sequence is observed. We gain significant performance improvements in activity detection and early detection on ActivityNet. In future work, we plan to conduct further in-depth study of the relative advantages of the two ranking losses.

Acknowledgments. This work was supported in part by NSF grant 1029430 and a gift from NVIDIA.

References

- [1] W. Brendel and S. Todorovic. Learning spatiotemporal graphs of human activities. In *ICCV*, 2011. 2
- [2] P. F. Felzenszwalb, R. B. Girshick, D. A. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *TPAMI*, 32(9):1627–1645, 2010. 2
- [3] G. Gkioxari and J. Malik. Finding action tubes. In *CVPR*, 2015. 2
- [4] F. C. Heilbron, V. Escorcia, B. Ghanem, and J. C. Niebles. Activitynet: A large-scale video benchmark for human activity understanding. In *CVPR*, 2015. 2, 5, 6, 7
- [5] M. Hoai and F. De la Torre. Max-margin early event detectors. *IJCV*, 107(2):191–202, 2014. 1, 3
- [6] S. Ji, W. Xu, M. Yang, and K. Yu. 3d convolutional neural networks for human action recognition. *TPAMI*, 35(1):221–231, 2013. 2
- [7] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 2
- [8] Y. Ke, R. Sukthankar, and M. Hebert. Event detection in crowded videos. In *ICCV*, 2007. 2
- [9] Y. Kong and Y. Fu. Max-margin action prediction machine. *TPAMI*, PP(99):1–1, 2015. 3
- [10] T. Lan, Y. Wang, and G. Mori. Discriminative figure-centric models for joint action localization and recognition. In *ICCV*, 2011. 2
- [11] I. Laptev, M. Marszałek, C. Schmid, and B. Rozenfeld. Learning realistic human actions from movies. In *CVPR*, 2008. 2
- [12] S. Ma, L. Sigal, and S. Sclaroff. Space-time tree ensemble for action recognition. In *CVPR*, 2015. 2
- [13] S. Ma, J. Zhang, N. Ikizler-Cinbis, and S. Sclaroff. Action recognition and localization by hierarchical space-time segments. In *ICCV*, 2013. 2
- [14] J. Y. Ng, M. J. Hausknecht, S. Vijayanarasimhan, O. Vinyals, R. Monga, and G. Toderici. Beyond short snippets: Deep networks for video classification. *CoRR*, abs/1503.08909, 2015. 2
- [15] B. Ni, V. R. Paramathayalan, and P. Moulin. Multiple granularity analysis for fine-grained action detection. In *CVPR*, 2014. 2
- [16] V. Pham, C. Kermorvant, and J. Louradour. Dropout improves recurrent neural networks for handwriting recognition. *CoRR*, 2013. 3
- [17] M. Raptis and L. Sigal. Poselet key-framing: A model for human activity recognition. In *CVPR*, 2013. 2, 3
- [18] 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. *CoRR*, abs/1502.06648, 2015. 2
- [19] M. Ryoo. Human activity prediction: Early recognition of ongoing activities from streaming videos. In *ICCV*, 2011. 1, 3
- [20] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. In *NIPS*, 2014. 2
- [21] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *ICLR*, 2015. 3, 5
- [22] Y. Tian, R. Sukthankar, and M. Shah. Spatiotemporal deformable part models for action detection. In *CVPR*, 2013. 2
- [23] H. Wang and C. Schmid. Action recognition with improved trajectories. In *ICCV*, 2013. 2
- [24] L. Wang, Y. Qiao, and X. Tang. Video action detection with relational dynamic-poselets. In *ECCV*, 2014. 2
- [25] Y. Wang and G. Mori. Hidden part models for human action recognition: Probabilistic versus max margin. *TPAMI*, 33(7):1310–1323, 2011. 2
- [26] Z. Wu, X. Wang, Y. Jiang, H. Ye, and X. Xue. Modeling spatial-temporal clues in a hybrid deep learning framework for video classification. *CoRR*, abs/1504.01561, 2015. 2
- [27] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and F. Li. Every moment counts: Dense detailed labeling of actions in complex videos. *CoRR*, 2015. 2, 3
- [28] J. Yuan, Z. Liu, and Y. Wu. Discriminative subvolume search for efficient action detection. In *CVPR*, 2009. 2