Context-Aware Action Detection in Untrimmed Videos Using Bidirectional LSTM

Jaideep Singh Chauhan
Indian Institute of Technology, Kharagpur
Kharagpur, West Bengal, India
Email: jaideepiit2@gmail.com

Yang Wang
Department of Computer Science
University of Manitoba, Canada
Email: ywang@cs.umanitoba.ca

***Abstract*—We consider the problem of action detection in untrimmed videos. We argue that the contextual information in a video is important for this task. Based on this intuition, we design a network using a bidirectional Long Short Term Memory (Bi-LSTM) model that captures the contextual information in videos. Our model includes a modified loss function which enforces the network to learn action progression, and a backpropagation in which gradients are weighted on the basis of their origin on the temporal scale. LSTMs are good at capturing the long temporal dependencies, but not so good at modeling local temporal features. In our model, we use a 3-D Convolutional Neural Network (3-D ConvNet) for capturing the local spatio-temporal features of the videos. We perform a comprehensive analysis on the importance of learning the context of the video. Finally, we evaluate our work on two action detection datasets, namely ActivityNet and THUMOS'14. Our method achieves competitive results compared with the existing approaches on both datasets.**

***Keywords*-action detection, LSTM, video analysis**

I. Introduction

Understanding actions in videos is an important task in computer vision. A lot of previous work [1], [2] has focused on action classification. Impressive results have been achieved in classifying trimmed videos (lasting a few seconds). Action classification only provides a coarse description of a video. Real-world applications often require localizing actions in untrimmed videos, which is a particularly challenging task since untrimmed videos [3], [4] have actions which are long, complex and come with a lot of inter-class variations. In this paper, we consider action detection in an untrimmed video where the goal is to localize the temporal extent of the action in the video.

A popular approach for action detection is based on frame-wise classification [5], [6]. Frame-wise classifiers use either hand-crafted features [7], [8] or deep features [9], [2] learned using Convolutional Neural Networks (ConvNets). These features are good at capturing the local spatio-temporal characteristics of the activities occurring in the videos but fail to learn the long-term dependencies.

Recurrent Neural Networks (RNNs) provide a potential solution for modeling the long-term temporal dependencies. They have been successfully used in various sequence labeling problems, such as sequence generation [10], machine translation [11], image and video captioning [12], etc. RNNs model temporal dependencies by maintaining hidden states over time. The prediction at each time step is a function of previous hidden states and the current input. In our work, we use a popular variant of RNN called the Bidirectional Long Short Term Memory (Bi-LSTM). Bi-LSTM uses memory states to control the flow of information in both directions in the network.

In this work, we tackle action detection using deep features learned by a 3D Convolutional Neural Network (3D-ConvNet). These features are fed to a modified Bi-LSTM for the temporal localization of the actions. In real-world videos, there are often multiple long activities separated by periods of non-activity. As an LSTM sees more frames of the video, it should become more confident in its predictions. To capture this intuition, our Bi-LSTM uses a loss function suggested in [13] to enforce the network to learn action progression. This loss function penalizes the network if it does not give a monotonically increasing score for the ground-truth class as the video progresses. To further improve the model, we introduce a linear bias on the gradients during backpropagation. Again, this is based on the intuition that as the model sees more of the activity, it should become more confident in its prediction. Thus the gradients encountered further on the temporal scale should be more informative.

This paper makes the following contributions:

- We introduce a modified Bidirectional LSTM network for action detection and show its effectiveness in learning the context of the video.
- We show the importance of learning the context and progression in predicting the overall class by comparing models with and without learning the context.
- We perform extensive evaluations on two publicly available untrimmed video datasets and demonstrate the effectiveness of our approach.

II. Related Work

Traditional action recognition pipelines capture the local spatio-temporal information using hand-crafted feature descriptors, such as Histogram of Oriented Gradients (HOG) [14], [7], Histogram of Optical Flow (HOF) [7] and Motion Boundary Histograms [8]. These features are then encoded to represent the video using some encoding techniques, such as bag of Words (BoW) [7] or Fisher Vector (FV) [15]. The

video-level features are then fed to standard classifiers for the recognition task.

Recently, ConvNet based methods have been shown to outperform previous methods using hand-crafted features. ConvNets have achieved the state-of-the-art performance on many image based tasks such as image classification [16], [17], scene labeling [18] and object detection [19]. There have been several attempts to adopt ConvNet in action detection. Simonyan *et al.*[20] use a two-stream ConvNet with one stream for the spatial and the other stream for the temporal features. The temporal ConvNet uses multi-frame optical flow for training whereas the spatial ConvNet is trained on RGB images. They classify the action but do not localize it in the video. Karpathy *et al.*[9] explore several approaches which exploit the local spatio-temporal features by extending the connectivity of ConvNets from the spatial domain to the temporal domain. Their method only slightly outperforms single-frame classification based methods. This shows the difficulty in learning temporal dependencies in long videos.

Several 3-D ConvNet based approaches [21], [22], [23] have been proposed for action recognition. 3-D ConvNet extends 2-D ConvNet by adding an extra temporal dimension. In [2], it is shown that 3-D ConvNets are more appropriate for learning local spatio-temporal features than 2-D ConvNets. The limitation of 3-D ConvNets is that although they are extremely effective for learning short dependencies, they do not work well for the longer dependencies that are common in long untrimmed videos.

RNNs provide a potential solution to the problem of action detection in long untrimmed videos. In particular, a variant of RNN called LSTM [24] has been shown to be very effective in learning long term dependencies. Deep features learned using 3-D ConvNets followed by a LSTM classifier have recently shown promise in video detection tasks for long videos [25], [26], [27]. 3-D ConvNets tend to learn the local spatio-temporal features whereas LSTMs learn the long temporal dependencies. [13], [28] use the combination of LSTM and ConvNet to detect actions at every frame. [29] uses a bi-directional LSTM for action recognition. [13] learns the progression in videos by enforcing it using a modified loss function. Our model is similar to theirs. But in addition to learning the progression, we also incorporate the local spatio-temporal dependencies by using features learned by a 3-D ConvNet on short clips of the whole video.

III. Approach

Our framework is shown in Figure 1. We divide an input video into a sequence of short clips, where each clip has 16 frames. These clips are then passed to a 3-D Convolutional Network called C3D [2]. C3D takes in a $3 \times 16 \times 128 \times 171$ (channels $\times$ frames $\times$ width $\times$ height) video clip as the input and produce a 4096 dimensional feature vector. The feature vectors of the short clips are then fed to a Bi-LSTM consisting of a forward and a backward running LSTM. Both LSTMs are connected to the same output layer. We apply a softmax layer on top of the output layer to get the probability scores for each activity class (and a no-activity class) for each clip. Finally we combine the scores of all the clips to get the activity class for the whole video by taking a weighted average of the scores. The clips where the activity is happening are identified as those with the highest scores for the class assigned to the whole video.

A. Bi-LSTM Architecture

For an input sequence $x = (x_1, ..., x_T)$, a standard RNN model computes the hidden states $h = (h_1, ..., h_T)$ and the output sequence $y = (y_1, ., y_T)$. The recurrence equations in RNN are shown in Eq. 1 and Eq. 2 :

$$h_t = \Gamma(W_{xh} \cdot x_t + W_{hh} \cdot h_{t1} + b_h) \tag{1}$$

$$y_t = \Gamma(W_{hy} \cdot h_t + b_z) \tag{2}$$

Here Γ is a nonlinear activation function. W_{xh} is a matrix which maps the input to the hidden state. W_{hh} captures the relationship between the hidden states in adjacent time steps. W_{hy} maps the hidden state at a time step to the output. A popular variant of the RNN model is called the Long Short Term Memory (LSTM) model. LSTM implements *memory* using three gates which control the flow of information from the previous time steps and the current input. The update equations for LSTMs are as follows :

$$i_t = \sigma(W_{xi} \cdot x_t + W_{hi} \cdot h_t + W_{ci} \cdot c_t + b_i) \tag{3}$$

$$f_t = \sigma(W_{xf} \cdot x_t + W_{hf} \cdot h_t + W_{cf} \cdot c_t + b_f) \tag{4}$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot \tanh(W_{xc} \cdot x_t + W_{hc} \cdot h_{t-1} + b_c) \tag{5}$$

$$o_t = \sigma(W_{xo} \cdot x_t + W_{ho} \cdot h_t + W_{co} \cdot c_t + b_o) \tag{6}$$

$$h_t = o_t \cdot tanh(c_t) \tag{7}$$

where σ is the element-wise sigmoid function and i, f, c and o are the gates (input gate, forget gate, cell activation, output gate) which control the flow of memory from previous time steps.

For a given clip, other clips (either come before or after this clip) provide contextual information about the video. In our work, we would like to incorporate this contextual information to a RNN cell when it processes a clip. An elegant solution is to use Bi-LSTM. Bi-LSTM has two LSTMs running in opposite directions providing contextual information both from the future and the past. To implement Bi-LSTM, we need to calculate two different hidden states. This calculation is done as follows :

$$h_t^f = \Gamma(W_{xh}^f \cdot x_t + W_{hh}^f \cdot h_{t1}^f + b_h^f) \tag{8}$$

$$h_t^b = \Gamma(W_{xh}^b \cdot x_t + W_{hh}^b \cdot h_{t1}^b + b_h^b) \tag{9}$$

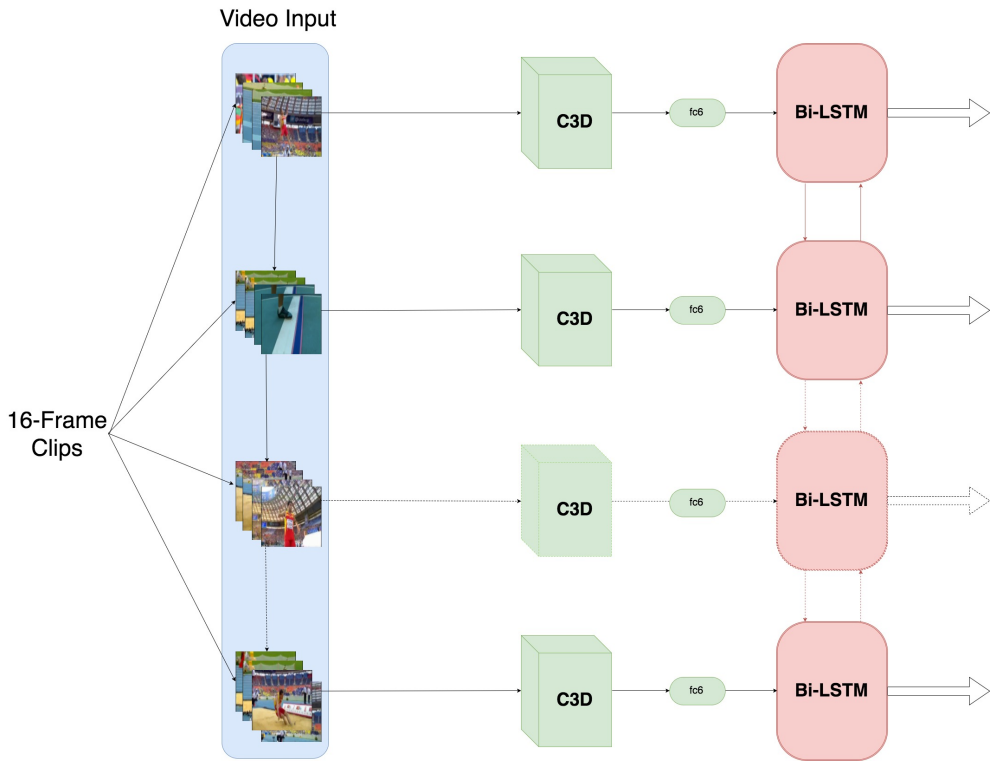


Figure 1. Illustration of the architecture of our model. (Left) A video is divided into a sequence of 16-frame clips. (Middle) The C3D network is used to convert each clip into a feature vector. (Right) Finally, the feature vector of each clip is fed to a Bi-LSTM model. The Bi-LSTM consists of two LSTMs running in forward and backward directions.

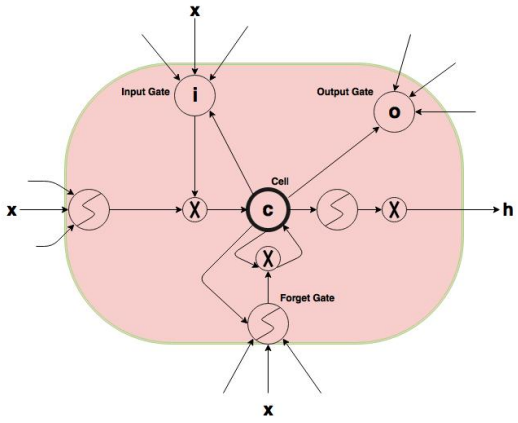


Figure 2. An illustration of an LSTM cell. It has an input gate **i**, an output gate **o**, a forget gate **f** and a memory cell **c**.

where the subscript f is for forward running hidden states and b is for backward running hidden states. Then we combine the two hidden states to get the output y_T as follows :

$$y_T = \nabla(U[g \cdot h_t^f, (1-g) \cdot h_t^b] + c) \quad (10)$$

In Eq. 10, we introduce a term g which we call the *gain term*. This term controls the relative contributions of the two LSTMs on the final output. The intuition is that the hidden states of either LSTMs know more about the video when it has seen more video clips. Thus we weight the hidden states with the term g which is linearly interpolated between 1 and 0 for the clip from T to 0. We concatenate the hidden states of both LSTMs using the gain term g. The concatenated vector is used to predict the final output. On top of this output layer, we apply a softmax layer to obtain the scores for each clip.

B. Bi-LSTM Training

In this section, we introduce several modifications of the standard Bi-LSTM for our problem. First, we use a loss function which enforces Bi-LSTM to learn the progression of an action in the video. This allows Bi-LSTM to better capture the contextual information in the video. Second, we propose to weight the gradients in backpropagation according to where the gradients originate along the time scale. Gradients encountered further on the temporal scale hold are more informative than the ones before them. This speeds up the convergence and improves the performance of the model.

1) Loss Function: The standard classification loss commonly used in LSTM learning does not explicitly ensure that the action progression in the videos is properly learned.

Standard LSTMs only implicitly learn the context in long videos through previous hidden states. To properly capture the action progression in videos, a ranking loss function was introduced in [13] for unidirectional LSTMs. Before we discuss the loss function, we like to point out that using Bi-LSTM is a potential solution for learning the context as well, since it uses both the future and past hidden states in calculating the output. Thus it is interesting to see the effect of the ranking loss function on the Bi-LSTMs in learning action progression. As we will show in the experiments, these two strategies complement each other and the final model using both achieves the best performance.

The new loss function is modeled on two principles [13]. First, LSTM gathers more info as it sees more frames in the video. Thus the score given to the ground-truth activity class should not decrease as the model observes more of the video. Second, the margin of the scores between the ground-truth action class and other classes should not decrease either as the model observes more of the video. Based on these two principles, the loss function is defined as follows:

$$L^s = L^s_c + \Omega L^s_r \tag{11}$$

Our video is divided into 16-frame long clips. They are arranged as a sequence $S = (s_1, s_2, ..., s_N)$. The subscript in Eq.11 denotes this sequence. Here L^s_c is the standard cross entropy classification loss function. L^s_r is the ranking loss function and Ω is a parameter that controls the relative contributions of these two losses. L^s_r consists of two terms L^s_{det} and L^s_{margin} based on the two principles defined above. Here L^s_{det} is the detection score loss function and L^s_{margin} is the margin loss function. The two terms are defined as follows.

In order to define the *detection score* loss function L^s_{det}, let us first introduce some notations for convenience. For a given clip s, we use y^s to denote the ground-truth action label of this clip. We use the term *time sequence* to refer to a continuous sequence of clips $(s_{i-n}, ..., s_i, ..., s_{i+n})$ for which the ground-truth activity class is same. We use S_s to denote the starting position of the current time sequence defined as follows:

$$s_s = \min\{s_{\prime} | y_{s_{\prime}} = y_s \forall s_{\prime} \in (s_s, s)\} \tag{12}$$

We use $p^{y_s}_s$ to denote the detection score for the ground-truth class of the clip s. We define $p^{*y_s}_s$ to be the highest detection score encountered for the ground-truth class of the current *time sequence* since the beginning of the time sequence. It is mathematically defined as:

$$p^{*y_s}_s = \max\{p^{*y_s}_s | \forall s \in [s_s, s-1]\} \tag{13}$$

Then we can define the *detection score* loss function L^s_{det} as follows:

$$L^s_{det} = \begin{cases} \max(0, p^{*y_s}_s - p^{y_s}_s) & \text{if } y_s = y_{s-1} \\ p^{y_{s-1}}_s & \text{otherwise} \end{cases} \tag{14}$$

The intuition of Eq.14 is as follows. If there is no activity transition between the clips s and $s-1$, the loss is equal to the difference between the highest encountered detection score in the *time sequence* and the detection score for the current clip s for the ground-truth class. Otherwise, if there is an activity transition, the detection score for the ground class of the previous step $(s-1)$ is taken as the loss for the current step (s).

Similarly, the *margin* loss function L^s_{margin} penalizes the model if the margin between the correct and the incorrect class is decreasing as it observes more frames. It is implemented as follows :

$$L^s_{margin} = \begin{cases} \max(0, m^{*y_s}_s - m^{y_s}_s) & \text{if } y_s = y_{s-1} \\ m^{y_{s-1}}_s & \text{otherwise} \end{cases} \tag{15}$$

In Eq.15, $m^{y_s}_s$ is defined as the *discriminative margin*, which is the difference between the detection score of the ground-truth class and the highest scoring class other than the ground-truth class for the current clip s. It is calculated as follows :

$$m^{y_s}_s = p^y_s - \max\{p^{y_{\prime}}_s | \forall y_{\prime} \in Y, y_{\prime} \neq y\} \tag{16}$$

In Eq.15, $m^{*y_s}_s$ is analogous to $p^{*y_s}_s$ in Eq. 14. In other words, it is the highest *discriminative margin* score which has been encountered in the current *time sequence* for the ground-truth class. It is calculated as follows :

$$m^{*y_s}_s = \max\{m^{*y_s}_s | \forall s \in [s_s, s-1]\} \tag{17}$$

where s_s denotes the start of the current *time sequence*, defined in Eq. 12.

Intuitively, Eq. 15 means that if there is no activity transition between clips s and $s-1$, the margin loss would be the difference between the highest encountered margin score in the *time sequence* and the margin score for the current clip s for the ground-truth class. Otherwise, if there is an activity transition, the margin score for the ground-truth class of the previous step $(s-1)$ is taken as the loss for the current step (s).

2) Training: When calculating the loss, we encounter three cases: (i) Previous and current clips have the same activity for which the loss is calculated using the first halves of Eqs. 14 and 15. (2) Previous clip has an activity, but current clip does not (i.e. background). Then the loss is calculated using the second halves of Eqs. 14 and 15. (iii) Previous time step does not have any activity label for which the loss would be 0. For training the model, we need to backpropagate the gradient of the loss function over the softmax output as follows :

$$\frac{\partial L_s}{\partial p^y_s} = \frac{\partial L^c_s}{\partial p^y_s} + g \cdot \Omega \frac{\partial L^r_s}{\partial p^y_s} \tag{18}$$

which can be backpropagated independently in the two opposite running LSTMs as the hidden states of the two are not dependent on each other.

The term g in Eq.18 is the gain term which we have introduced to weigh the gradients on temporal scale. The intuition is that when LSTM sees more of the activity, it becomes more aware of the context of the video. As a result, the gradients further down the temporal scale hold more information than the ones occurring early. The value of the gain term g is linearly interpolated between 0 to 1 for clip sequence 0 to S in the forward running LSTM, and between 1 to 0 for clip sequence 0 to S in the backward running LSTM.

IV. Experiments

In this section, we first introduce the two datasets used in the experiments. Then we discuss our experimental setup on these two datasets. Finally, we present our experimental results and compare our approach with other baselines.

A. Datasets

We evaluate our method on two publicly available datasets: THUMOS'14 [3] and ActivityNet [4]. Sample frames from videos for both datasets are shown in Figure 3.

1) ActivityNet Dataset: The ActivityNet [4] dataset consists of 10024, 4926 and 5044 videos in training, validation and test sets, respectively. These videos are untrimmed and contain activities belonging to one of the 200 different classes. We use a split ratio of $4:1$ to divide the train dataset for training and cross-validation purposes. Evaluation is done on the validation dataset. Multiple instances of a class may be present in a video and there are time intervals which do not have any annotated activities. On average, 1.4 activities are present per video. In total, it amounts to 849 hours of untrimmed videos with 68.8 hours of annotations. This is a particularly challenging dataset as the videos are taken from YouTube, so the videos have large variations.

2) THUMOS'14 Dataset: The THUMOS'14 dataset consists of 2765 trimmed, 200 and 213 untrimmed videos in the training, validation and test sets, respectively. As the training set only consists of trimmed videos, we only use it for fine-tuning the C3D architecture. The validation set is divided into training and validation with a split ratio of $4:1$. The evaluation is done on the test set. The dataset contains 20 activity classes. Each video may contain multiple activities belonging to the same class.

B. Experimental Setup

For both datasets, we extract the frames from the videos at 5fps due to memory constraints and resize them to 128×171. Each video is divided into a collection of 16 frame long clips. C3D [2] is used as the feature descriptor for the clips. We fine-tune the pre-trained C3D on our datasets. The output of the second fully connected layer (fc6) of C3D is taken as the feature vector (4096 dimension) that describes the clip.

1) Experiment on ActivityNet: The original C3D network is pre-trained on the Sports-1M dataset [9]. We first fine-tune the C3D network on the training videos in the ActivityNet dataset. For fine-tuning, we freeze the first two convolutional layers of the network and introduce a softmax layer with an output of 201 dimensions (200 classes + 1 non-activity class). A learning rate of 10^{-4} for 20 epochs is used. This is followed by a decay by a factor of 10 for every 5 epochs. We stop the fine-tuning at 30 epochs. The next step is to train the Bi-LSTM network. We use a batch size of 64 with each instance in the batch being a sequence of 64 vectors of length of 4096. We begin with a learning rate of 10^{-3} for the first 20 epochs, then decay it periodically by a factor of 10 for the next 40 epochs. Along with the Bi-LSTMs, we also train the fc6 layer of the C3D but with a small constant learning factor of 10^{-5}.

2) Experiment on THUMOS'14: On the THUMOS'14 dataset, we also fine-tune C3D on the training set in the same way as ActivityNet except that the softmax layer has 102 dimensions (101 classes + 1 non-activity class). Since THUMOS'14 is a smaller dataset compared with ActivityNet and is more similar to the Sports-1M dataset, it requires less fine-tuning than ActivityNet. We use a constant learning rate of 10^{-5} over 15 epochs. For THUMOS'14, we also train the fc6 layer along with the Bi-LSTMs with a constant learning rate of 10^{-5}.

C. Results

We show our results in terms of the mean Average Precision (mAP). For a detection to be true positive, the Intersection-Over-Union (IOU) between the temporal prediction and the ground-truth annotation should be above a threshold (α), and the class labels should match. This is called mAP@α and is a standard metric used to evaluate action detection methods in the literature [28], [13].

1) Results on THUMOS'14 Dataset: Table I shows mAP@α of our results compared with other baselines. First, we compare our results with the top 3 performers of the THUMOS'14 challenge [5], [6], [30]. We outperform them with a margin of 10.7 for the $\alpha = 0.1$. Next, we compare our model with a recent work in [31]. It uses proposals and a localization network for detection. Our results are comparable to [31] even though we do not use proposals.

Figure 4(a) shows a comparison between models with and without learning context on THUMOS'14. We train LSTM and Bi-LSTM models with and without the modified loss function, and also with and without the modified backpropagation (weighted backpropagation using gain term g). From the comparisons, we can make three main observations. First, Bi-LSTM models in general perform better than LSTM models. For example, when $\alpha = 0.1$, the margin between the best performing Bi-LSTM and LSTM models is 8.6%. Second, the modified loss helps improving the results. Bi-LSTM trained with the modified loss outperforms the one

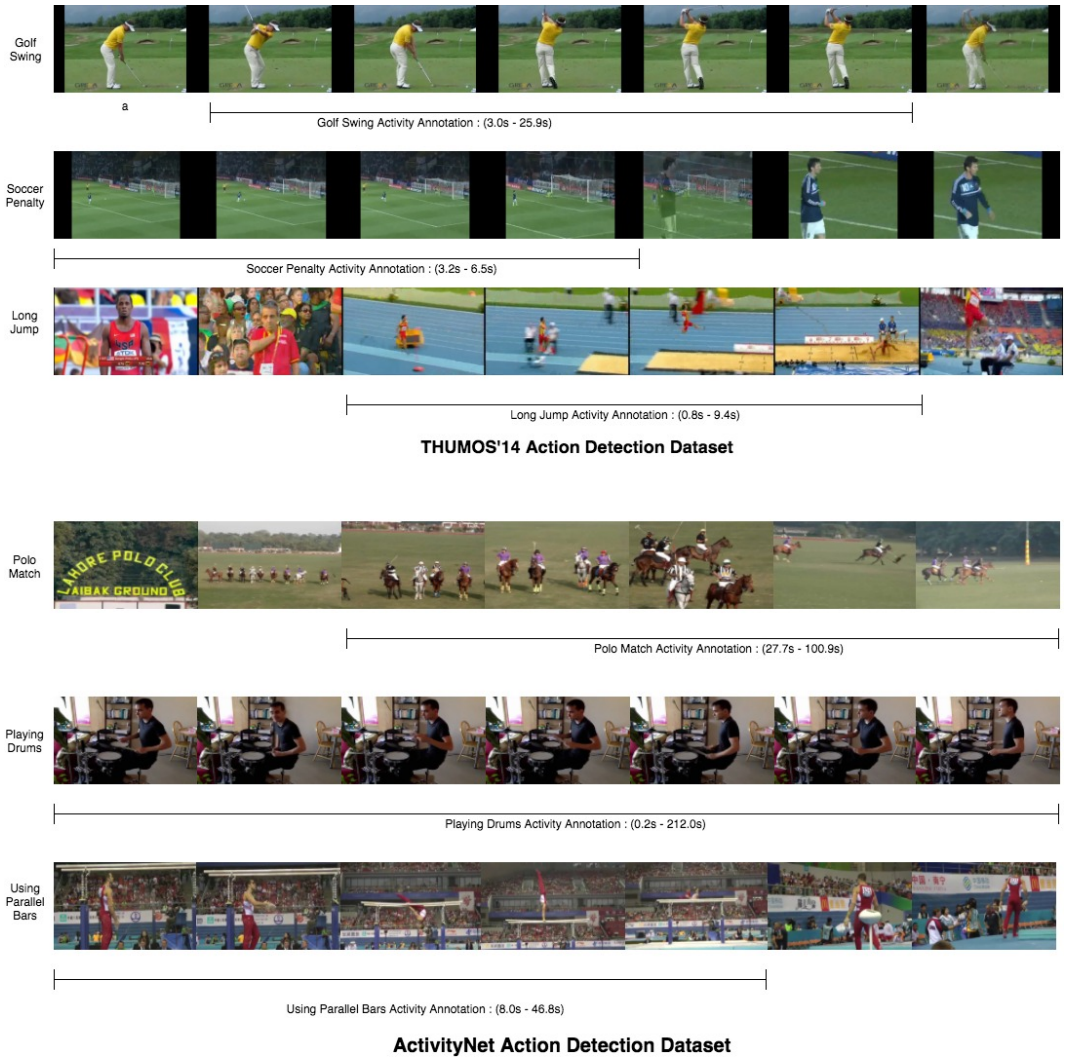


Figure 3. The first three rows are example frames from the videos taken from the THUMOS'14 dataset. This dataset has 20 classes corresponding to different sport activities, such as "Golf swings", "High Jumps", etc. The last three rows are example frames from the ActivityNet Dataset. The dataset consists of 200 classes of human activities, such as "Playing Drums", "Polo Matches", etc. In both datasets, ground-truth annotations are provided to indicate the start and end time of each action in a video, as illustrated in the figure.

without this loss by a margin of 2.4% for $\alpha = 0.1$. Third, the weighted backpropagation also improves the performance. Bi-LSTM trained with the modified backpropagation outperforms the alternative by a margin of 1.5% for $\alpha = 0.1$. The best performing model is a Bi-LSTM which is trained using modified backpropagation and modified loss.

2) Results on ActivityNet: Table II shows the results on ActivityNet. We compare our results with UPC [32]. Their network uses deep features learned through a 3D-ConvNet and a LSTM setup, so their work is similar to ours. For $\alpha = 0.5$, we outperform the method in [32] on validation set by 2.3%.

Figure 4(b) shows a comparison between models with and without learning context on ActivityNet. We train LSTMs and Bi-LSTMs with and without modified loss, also with and without the modified backpropagation. These results show similar conclusions. The worst performing model is the LSTM network without the modified loss or backpropagation. The best performing model is the Bi-LSTM model with the modified loss and backpropagation (margin between the two being 15.9% for $\alpha = 0.1$). Bi-LSTM networks in general outperform LSTM networks with similar training routines. With the best Bi-LSTM network outperforming the best LSTM network by 10.6% for $\alpha = 0.1$.

V. Conclusion

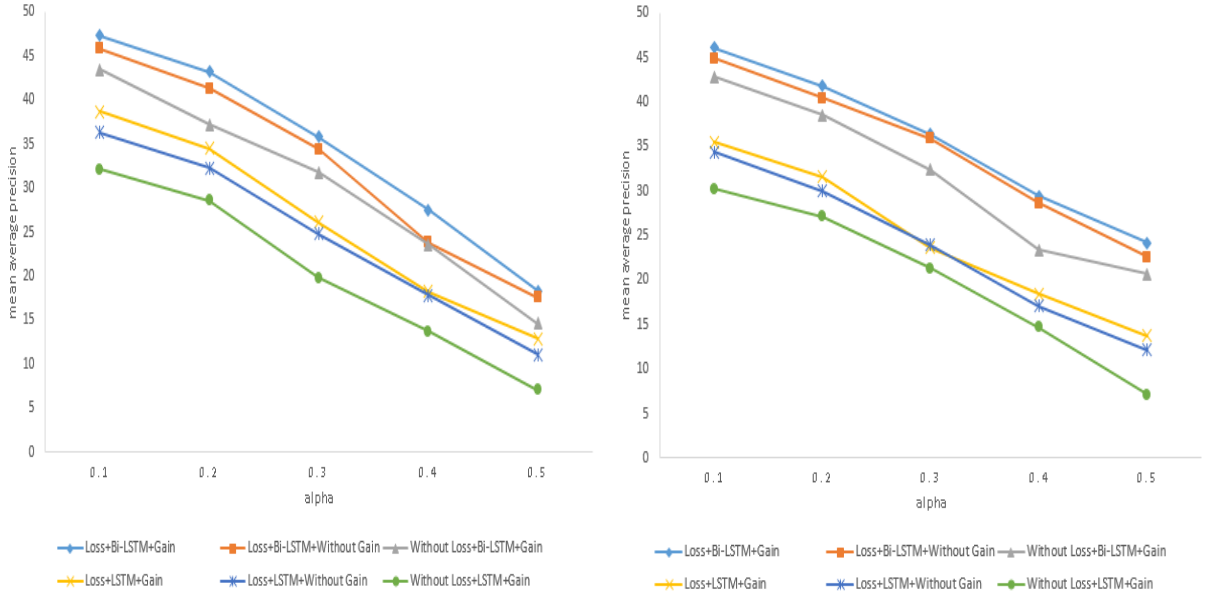
We have introduced a Bi-LSTM model for action detection in untrimmed videos. Our model which explicitly learns

Table I
EXPERIMENTAL RESULTS ON THE THUMOS'14 DATASET. WE USE MAP@α TO MEASURE THE PERFORMANCE OF EACH METHOD AND REPORT THE RESULTS FOR DIFFERENT VALUES OF α.

Models	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$
Karman *et al.*[5]	4.6	3.4	2.1	1.4	0.9
Wang *et al.*[6]	18.2	17.0	14.0	11.7	8.3
Oneata *et al.*[30]	36.6	33.6	27.0	20.8	14.4
Shou *et al.*[31]	47.7	43.5	36.3	28.7	19.0
Ours	47.3	43.2	35.8	27.5	18.3

Table II
EXPERIMENTAL RESULTS ON THE ACTIVITYNET DATASET. WE USE MAP@α TO MEASURE THE PERFORMANCE OF EACH METHOD AND REPORT THE RESULTS FOR DIFFERENT VALUES OF α.

Models	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$
Caba *et al.*[4]	12.5	11.9	11.1	10.4	9.7
UPC *et al.*[32]	-	-	-	-	22.5
Ours	**46.1**	**41.8**	**36.4**	**29.5**	**24.2**



(a) THUMOS'14 (b) ActivityNet

Figure 4. Comparison of different variants of our method on THUMOS'14 and ActivityNet. On each dataset, we compare different methods by varying these three design choices: (1) LSTM vs Bi-LSTM; (2) modified loss vs standard loss; (3) modified backpropagation vs standard backpropatation.

the contextual information in the video. We have demonstrated the importance of learning the contextual information for action detection through a comparative study. We find that Bi-LSTMs are better suited for learning the context of videos than LSTMs and hence are better at the task of detection. We achieve competitive results on two benchmark datasets.

ACNOWLEDGMENT

This work was done during Jaideep Singh Chauhan's internship at the University of Manitoba funded by the MITACS Globalink program. Yang Wang is supported by grants from NSERC. We thank NVIDIA for the GPU donations used in this work.

REFERENCES

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

[2] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in *IEEE International Conference on Computer Vision*, 2015, pp. 4489–4497.

[3] Y. Jiang, J. Liu, A. R. Zamir, G. Toderici, I. Laptev, M. Shah, and R. Sukthankar, "Thumos challenge: Action recognition with a large number of classes," 2014.

[4] F. Caba Heilbron, V. Escorcia, B. Ghanem, and J. Carlos Niebles, "Activitynet: A large-scale video benchmark

for human activity understanding," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 961–970.

[5] S. Karaman, L. Seidenari, and A. Del Bimbo, "Fast saliency based pooling of fisher encoded dense trajectories," in *ECCV THUMOS Workshop*, vol. 1, no. 2, 2014, p. 5.

[6] L. Wang, Y. Qiao, and X. Tang, "Action recognition and detection by combining motion and appearance features," *THUMOS14 Action Recognition Challenge*, vol. 1, no. 2, p. 2, 2014.

[7] I. Laptev, M. Marszalek, C. Schmid, and B. Rozenfeld, "Learning realistic human actions from movies," in *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*. IEEE, 2008, pp. 1–8.

[8] N. Dalal, B. Triggs, and C. Schmid, "Human detection using oriented histograms of flow and appearance," in *European Conference on Computer Vision*. Springer, 2006, pp. 428–441.

[9] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in *IEEE conference on Computer Vision and Pattern Recognition*, 2014, pp. 1725–1732.

[10] A. Graves, "Generating sequences with recurrent neural networks," *arXiv:1308.0850*, 2013.

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv:1406.1078*, 2014.

[12] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *IEEE conference on Computer Vision and Pattern Recognition*, 2015, pp. 3156–3164.

[13] S. Ma, L. Sigal, and S. Sclaroff, "Learning activity progression in lstms for activity detection and early detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1942–1950.

[14] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1. IEEE, 2005, pp. 886–893.

[15] F. Perronnin, J. Sánchez, and T. Mensink, "Improving the fisher kernel for large-scale image classification," *European Conference on Computer Vision*, pp. 143–156, 2010.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[17] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *International Conference on Learning Representations*, 2015.

[18] C. Farabet, C. Couprie, L. Najman, and Y. LeCun, "Learning hierarchical features for scene labeling," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 8, pp. 1915–1929, 2013.

[19] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "Overfeat: Integrated recognition, localization and detection using convolutional networks," *arXiv:1312.6229*, 2013.

[20] K. Simonyan and A. Zisserman, "Two-stream convolutional networks for action recognition in videos," in *Advances in Neural Information Processing Systems*, 2014, pp. 568–576.

[21] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Sequential deep learning for human action recognition," in *International Workshop on Human Behavior Understanding*. Springer, 2011, pp. 29–39.

[22] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 1, pp. 221–231, 2013.

[23] V. Escorcia, F. C. Heilbron, J. C. Niebles, and B. Ghanem, "Daps: Deep action proposals for action understanding," in *European Conference on Computer Vision*. Springer, 2016, pp. 768–784.

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] G. Gkioxari and J. Malik, "Finding action tubes," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 759–768.

[26] J. Yue-Hei 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*, 2015, pp. 4694–4702.

[27] M. Baccouche, F. Mamalet, C. Wolf, C. Garcia, and A. Baskurt, "Action classification in soccer videos with long short-term memory recurrent neural networks," *International Joint Conferece on Artifical Neural Networks*, pp. 154–159, 2010.

[28] S. Yeung, O. Russakovsky, N. Jin, M. Andriluka, G. Mori, and L. Fei-Fei, "Every moment counts: Dense detailed labeling of actions in complex videos," *International Journal of Computer Vision*, pp. 1–15, 2015.

[29] A. Ullah, A. Jamil, K. Muhammad, M. Sajjad, and S. W. Baik, "Action recognition in video sequences using deep bi-directional lstm with cnn features," *IEEE Access*, vol. 6, pp. 1155–1166, 2017.

[30] D. Oneata, J. Verbeek, and C. Schmid, "The LEAR submission at thumos 2014," 2014.

[31] Z. Shou, D. Wang, and S.-F. Chang, "Temporal action localization in untrimmed videos via multi-stage cnns," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 1049–1058.

[32] A. Montes, A. Salvador, and X. Giro-i Nieto, "Temporal activity detection in untrimmed videos with recurrent neural networks," *arXiv:1608.08128*, 2016.