Detailed Action Identification in Baseball Game Recordings

Abstract

This research introduces MLB-YouTube, a new and complex dataset created for nuanced activity recognition in baseball videos. This dataset is structured to support two types of analysis: one for classifying activities in segmented videos and another for detecting activities in unsegmented, continuous video streams. This study evaluates several methods for recognizing activities, focusing on how they capture the temporal organization of activities in videos. This evaluation starts with categorizing segmented videos and progresses to applying these methods to continuous video feeds. Additionally, this paper assesses the effectiveness of different models in the challenging task of forecasting pitch velocity and type using baseball broadcast videos. The findings indicate that incorporating temporal dynamics into models is beneficial for detailed activity recognition.

1 Introduction

Action recognition, a significant problem in computer vision, finds extensive use in sports. Professional sporting events are extensively recorded for entertainment, and these recordings are invaluable for subsequent analysis by coaches, scouts, and media analysts. While numerous game statistics are currently gathered manually, the potential exists for these to be replaced by computer vision systems. Systems like PITCHf/x and Statcast have been employed by Major League Baseball (MLB) to automatically record pitch speed and movement, utilizing a network of high-speed cameras and radar to collect detailed data on each player. Access to much of this data is restricted from the public domain.

This paper introduces MLB-YouTube, a novel dataset that includes densely annotated frames of activities extracted from broadcast baseball videos. Unlike many current datasets for activity recognition or detection, our dataset emphasizes fine-grained activity recognition. The differences between activities are often minimal, primarily involving the movement of a single individual, with a consistent scene structure across activities. The determination of activity is based on a single camera perspective. This study compares various methods for temporal feature aggregation, both for classifying activities in segmented videos and for detecting them in continuous video streams.

2 Related Work

The field of activity recognition has garnered substantial attention in computer vision research. Initial successes were achieved with hand-engineered features such as dense trajectories. The focus of more recent studies has shifted towards the application of Convolutional Neural Networks (CNNs) for activity recognition. Two-stream CNN architectures utilize both spatial RGB frames and optical flow frames. To capture spatio-temporal characteristics, 3D XYT convolutional models have been developed. The development of these advanced CNN models has been supported by large datasets such as Kinetics, THUMOS, and ActivityNet.

Several studies have investigated the aggregation of temporal features for the purpose of activity recognition. Research has compared several pooling techniques and determined that both Long Short-

.

Term Memory networks (LSTMs) and max-pooling across entire videos yielded the best outcomes. It has been discovered that pooling intervals from varying locations and durations is advantageous for activity recognition. It was demonstrated that identifying and classifying key sub-event intervals can lead to better performance.

Recently, segment-based 3D CNNs have been employed to capture spatio-temporal data concurrently for activity detection. These methods depend on the 3D CNN to capture temporal dynamics, which typically span only 16 frames. Although longer-term temporal structures have been explored, this was usually accomplished with temporal pooling of localized features or (spatio-)temporal convolutions with extended fixed intervals. Recurrent Neural Networks (RNNs) have also been applied to represent transitions in activity between frames.

3 MLB-YouTube Dataset

We have compiled an extensive dataset from 20 baseball games of the 2017 MLB postseason, available on YouTube, totaling over 42 hours of video. Our dataset includes two main parts: segmented videos intended for activity recognition and continuous videos designed for activity classification. The dataset's complexity is amplified by the fact that it originates from televised baseball games, where a single camera perspective is shared among various activities. Additionally, there is minimal variance in motion and appearance among different activities, such as swinging a bat versus bunting. In contrast to datasets like THUMOS and ActivityNet, which encompass a broad spectrum of activities with diverse settings, scales, and camera angles, our dataset features activities where a single frame might not be adequate to determine the activity.

The minor differences between a ball and a strike are illustrated in Figure 3. Differentiating between these actions requires identifying whether the batter swings or not, detecting the umpire's signal (Figure 4) for a strike, or noting the absence of a signal for a ball. This is further complicated because the batter or catcher can obstruct the umpire, and each umpire has their unique style of signaling a strike.

Our dataset for segmented video analysis comprises 4,290 clips. Each clip is annotated for multiple baseball actions, including swing, hit, ball, strike, and foul. Given that a single clip may contain several activities, this is considered a multi-label classification task. Table 1 presents the complete list of activities and their respective counts within the dataset. Additionally, clips featuring a pitch were annotated with the type of pitch (e.g., fastball, curveball, slider) and its speed. Furthermore, a collection of 2,983 hard negative examples, where no action is present, was gathered. These instances include views of the crowd, the field, or players standing idly before or after a pitch. Examples of activities and hard negatives are depicted in Figure 2.

Our continuous video dataset includes 2,128 clips, each lasting between 1 and 2 minutes. Every frame in these videos is annotated with the baseball activities that occur. On average, each continuous clip contains 7.2 activities, amounting to over 15,000 activity instances in total.

Table 1: Activity classes and their instance counts in the segmented MLB-YouTube dataset.

Activity	Count
No Activity	2983
Ball	1434
Strike	1799
Swing	2506
Hit	1391
Foul	718
In Play	679
Bunt	24
Hit by Pitch	14

4 Segmented Video Recognition Approach

We investigate different techniques for aggregating temporal features in segmented video activity recognition. In segmented videos, the classification task is simpler because each frame corresponds to an activity, eliminating the need for the model to identify the start and end of activities. Our methods are based on a CNN that generates a per-frame or per-segment representation, derived from standard two-stream CNNs using deep CNNs like I3D or InceptionV3.

Given video features v of dimensions $T \times D$, where T represents the video's temporal length and D is the feature's dimensionality, the usual approach for feature pooling involves max- or mean-pooling across the temporal dimension, followed by a fully-connected layer for video clip classification, as depicted in Fig. 5(a). This approach, however, yields a single representation for the entire video, losing temporal information. An alternative is to employ a fixed temporal pyramid with various lengths, as shown in Fig 5(b), dividing the video into intervals of lengths 1/2, 1/4, and 1/8, and max-pooling each. The pooled features are concatenated, creating a $K \times D$ representation, where K is the number of intervals in the temporal pyramid, and a fully-connected layer classifies the clip.

We also explore learning temporal convolution filters to aggregate local temporal structures. A kernel of size $L \times 1$ is applied to each frame, enabling each timestep representation to incorporate information from adjacent frames. After applying max-pooling to the output of the temporal convolution, a fully-connected layer is used for classification, as illustrated in Fig. 5(c).

While temporal pyramid pooling retains some structure, the intervals are fixed and predetermined. Previous studies have shown that learning the sub-interval to pool is beneficial for activity recognition. These learned intervals are defined by three parameters: a center g, a width σ, and a stride δ, parameterizing N Gaussians. Given the video length T, the positions of the strided Gaussians are first calculated as:

$$g_n = 0.5 - \frac{T - (g_n + 1)}{N - 1} \quad for n = 0, 1, \ldots, N-1$$

$$p_{t,n} = g_n + (t - 0.5T + 0.5)\frac{1}{\delta} \quad for t = 0, 1, \ldots, T-1$$

The filters are then generated as:

$$F_m[i, t] = \frac{1}{Z_m} \exp\left(-\frac{(t - \mu_{i,m})^2}{2\sigma_m^2}\right) \quad i \in \{0, 1, \ldots, N-1\}, t \in \{0, 1, \ldots, T-1\}$$

where Z_m is a normalization constant.

We apply these filters F to the $T \times D$ video representation through matrix multiplication, yielding an $N \times D$ representation that serves as input to a fully-connected layer for classification. This method is shown in Fig 5(d).

Additionally, we compare a bi-directional LSTM with 512 hidden units, using the final hidden state as input to a fully-connected layer for classification. We frame our tasks as multi-label classification and train these models to minimize binary cross-entropy:

$$L(v) = \sum_c z_c \log(p(c|G(v))) + (1 - z_c) \log(1 - p(c|G(v)))$$

where $G(v)$ is the function that pools the temporal information, and z_c is the ground truth label for class c.

5 Activity Detection in Continuous Videos

Detecting activities in continuous videos poses a greater challenge. The goal here is to classify each frame according to the activities occurring. Unlike segmented videos, continuous videos feature multiple sequential activities, often interspersed with frames of inactivity. This necessitates that the model learn to identify the start and end points of activities. As a baseline, we train a single fully-connected layer to serve as a per-frame classifier, which does not utilize temporal information beyond that contained in the features.

We adapt the methods developed for segmented video classification to continuous videos by implementing a temporal sliding window approach. We select a fixed window duration of L features, apply max-pooling to each window (similar to Fig. 5(a)), and classify each pooled segment. This approach is extended to temporal pyramid pooling by dividing the window of length L into segments of lengths $L/2$, $L/4$, and $L/8$, resulting in 14 segments per window. Max-pooling is applied to each segment, and the pooled features are concatenated, yielding a $14 \times D$-dimensional representation for each window, which is then used as input to the classifier.

For temporal convolutional models in continuous videos, we modify the segmented video approach by learning a temporal convolutional kernel of length L and convolving it with the input video features. This operation transforms input of size $T \times D$ into output of size $T \times D$, followed by a per-frame classifier. This enables the model to aggregate local temporal information.

To extend the sub-event model to continuous videos, we follow a similar approach but set $T = L$ in Eq. 1, resulting in filters of length L. The $T \times D$ video representation is convolved with the sub-event filters F, producing an $N \times D \times T$-dimensional representation used as input to a fully-connected layer for frame classification.

The model is trained to minimize per-frame binary classification:

$$L(v) = \sum_{t,c} z_{t,c} \log(p(c|H(v_t))) + (1 - z_{t,c}) \log(1 - p(c|H(v_t)))$$

where v_t is the per-frame or per-segment feature at time t, $H(v_t)$ is the sliding window application of one of the feature pooling methods, and $z_{t,c}$ is the ground truth class at time t.

A method to learn 'super-events' (i.e., global video context) has been introduced and shown to be effective for activity detection in continuous videos. This approach involves learning a set of temporal structure filters modeled as N Cauchy distributions. Each distribution is defined by a center x_n and a width γ_n. Given the video length T, the filters are constructed by:

$$x_n = \frac{(T-1)(\tanh(x'_n) + 1)}{2}$$

$$f_n(t) = \frac{1}{Z_n} \frac{\gamma_n}{\pi((t - x_n)^2 + \gamma_n^2)} \exp(1 - 2|\tanh(\gamma'_n)|)$$

where Z_n is a normalization constant, $t \in \{1, 2, \ldots, T\}$, and $n \in \{1, 2, \ldots, N\}$.

The filters are combined with learned per-class soft-attention weights A, and the super-event representation is computed as:

$$S_c = \sum_n A_{c,n} \sum_t f_n(t) \cdot v_t$$

where v is the $T \times D$ video representation. These filters enable the model to focus on relevant intervals for temporal context. The super-event representation is concatenated to each timestep and used for classification. We also experiment with combining the super- and sub-event representations to form a three-level hierarchy for event representation.

6 Experiments

6.1 Implementation Details

For our base per-segment CNN, we utilize the I3D network, pre-trained on the ImageNet and Kinetics datasets. I3D has achieved state-of-the-art performance on segmented video tasks, providing a reliable feature representation. We also employ a two-stream version of InceptionV3, pre-trained on Imagenet and Kinetics, as our base per-frame CNN for comparison. InceptionV3 was chosen for its depth compared to previous two-stream CNNs. Frames were extracted at 25 fps, and TVL1 optical flow was computed and clipped to $[-20, 20]$. For InceptionV3, features were computed every 3 frames (8 fps), while for I3D, every frame was used, with I3D having a temporal stride of 8, resulting in 3 features per second (3 fps). Models were implemented in PyTorch and trained using the Adam optimizer with a learning rate of 0.01, decayed by a factor of 0.1 every 10 epochs, for a total of 50 epochs.

6.2 Segmented Video Activity Recognition

We initially conducted binary pitch/non-pitch classification for each video segment. This task is relatively straightforward due to the distinct differences between pitch and non-pitch frames. The results, detailed in Table 2, reveal minimal variation across different features or models.

Table 2: Performance on segmented videos for binary pitch/non-pitch classification.

Model	RGB	Flow	Two-stream
InceptionV3	97.46	98.44	98.67
InceptionV3 + sub-events	98.67	98.73	99.36
I3D	98.64	98.88	98.70
I3D + sub-events	98.42	98.35	98.65

6.2.1 Multi-label Classification

We assessed various temporal feature aggregation methods by calculating the mean average precision (mAP) for each video clip, a standard metric for multi-label classification. Table 4 compares the performance of these methods. All methods surpass mean/max-pooling, highlighting the importance of preserving temporal structure for activity recognition. Fixed temporal pyramid pooling and LSTMs show some improvement. Temporal convolution offers a more significant performance boost but requires substantially more parameters (see Table 3). Learning sub-events, as per previous research, yields the best results. While LSTMs and temporal convolutions have been used before, they need more parameters and perform less effectively, likely due to overfitting. Moreover, LSTMs necessitate sequential processing of video features, whereas other methods can be fully parallelized.

Table 3: Additional parameters required for models when added to the base model (e.g., I3D or Inception V3).

Model	# Parameters
Max/Mean Pooling	16K
Pyramid Pooling	115K
LSTM	10.5M
Temporal Conv	31.5M
Sub-events	36K

Table 4: Mean Average Precision (mAP) results on segmented videos for multi-label classification. Learning sub-intervals for pooling is found to be crucial for activity recognition.

Method	RGB	Flow	Two-stream
Random	16.3	16.3	16.3
InceptionV3 + mean-pool	35.6	47.2	45.3
InceptionV3 + max-pool	47.9	48.6	54.4
InceptionV3 + pyramid	49.7	53.2	55.3
InceptionV3 + LSTM	47.6	55.6	57.7
InceptionV3 + temporal conv	47.2	55.2	56.1
InceptionV3 + sub-events	56.2	62.5	62.6
I3D + mean-pool	42.4	47.6	52.7
I3D + max-pool	48.3	53.4	57.2
I3D + pyramid	53.2	56.7	58.7
I3D + LSTM	48.2	53.1	53.1
I3D + temporal conv	52.8	57.1	58.4
I3D + sub-events	55.5	61.2	61.3

Table 5 shows the average precision for each activity class. Learning temporal structure is particularly beneficial for frame-based features (e.g., InceptionV3), which capture less temporal information

compared to segment-based features (e.g., I3D). Sub-event learning significantly aids in detecting strikes, hits, foul balls, and hit-by-pitch events, which exhibit changes in video features post-event. For instance, after a hit, the camera often tracks the ball's trajectory, while after a hit-by-pitch, it follows the player to first base, as illustrated in Fig. 6 and Fig. 7.

Table 5: Per-class average precision for segmented videos using two-stream features in multi-label activity classification. Utilizing sub-events to discern temporal intervals of interest proves advantageous for activity recognition.

Method	Ball	Strike	Swing	Hit	Foul	In Play	Bunt	Hit by Pitch
Random	21.8	28.6	37.4	20.9	11.4	10.3	1.1	4.5
InceptionV3 + max-pool	60.2	84.7	85.9	80.8	40.3	74.2	10.2	15.7
InceptionV3 + sub-events	66.9	93.9	90.3	90.9	60.7	89.7	12.4	29.2
I3D + max-pool	59.4	90.3	87.7	85.9	48.1	76.1	14.3	18.2
I3D + sub-events	62.5	91.3	88.5	86.5	47.3	75.9	16.2	21.0

6.2.2 Pitch Speed Regression

Estimating pitch speed from video frames is an exceptionally difficult problem, as it requires the network to pinpoint the pitch's start and end, and derive the speed from a minimal signal. The baseball, often obscured by the pitcher, travels at speeds over 100mph and covers 60.5 feet in approximately 0.5 seconds. Initially, with frame rates of 8fps and 3fps, only 1-2 features captured the pitch in mid-air, proving insufficient for speed determination. Utilizing the 60fps rate available in YouTube videos, we recalculated optical flow and extracted RGB frames at this higher rate. Employing a fully-connected layer with a single output for pitch speed prediction and minimizing the L1 loss between predicted and actual speeds, we achieved an average error of 3.6mph. Table 6 compares different models, and Fig. 8 illustrates the sub-events learned for various speeds.

Table 6: Results for pitch speed regression on segmented videos, reporting root-mean-squared errors.

Method	Two-stream
I3D	4.3 mph
I3D + LSTM	4.1 mph
I3D + sub-events	3.9 mph
InceptionV3	5.3 mph
InceptionV3 + LSTM	4.5 mph
InceptionV3 + sub-events	3.6 mph

6.2.3 Pitch Type Classification

We conducted experiments to determine the feasibility of predicting pitch types from video, a task made challenging by pitchers' efforts to disguise their pitches from batters and the subtle differences between pitches, such as grip and rotation. We incorporated pose data extracted using OpenPose, utilizing heatmaps of joint and body part locations as input to a newly trained InceptionV3 CNN. Pose features were considered due to variations in body mechanics between different pitches. Our dataset includes six pitch types, with results presented in Table 7. LSTMs performed worse than the baseline, likely due to overfitting, whereas learning sub-events proved beneficial. Fastballs were the easiest to detect (68% accuracy), followed by sliders (45%), while sinkers were the most difficult (12%).

6.3 Continuous Video Activity Detection

We evaluate models extended for continuous videos using per-frame mean average precision (mAP), with results shown in Table 8. This setting is more challenging than segmented videos, requiring the model to identify activity start and end times and handle ambiguous negative examples. All models improve upon the baseline per-frame classification, confirming the importance of temporal information. Fixed temporal pyramid pooling outperforms max-pooling, while LSTM and temporal

Table 7: Accuracy of pitch type classification using I3D for video inputs and InceptionV3 for pose heatmaps.

Method	Accuracy
Random	17.0%
I3D	25.8%
I3D + LSTM	18.5%
I3D + sub-events	34.5%
Pose	28.4%
Pose + LSTM	27.6%
Pose + sub-events	36.4%

convolution appear to overfit. Convolutional sub-events, especially when combined with super-event representation, significantly enhance performance, particularly for frame-based features.

Table 8: Performance on continuous videos for multi-label activity classification (per-frame mAP).

Method	RGB	Flow	Two-stream
Random	13.4	13.4	13.4
I3D	33.8	35.1	34.2
I3D + max-pooling	34.9	36.4	36.8
I3D + pyramid	36.8	37.5	39.7
I3D + LSTM	36.2	37.3	39.4
I3D + temporal conv	35.2	38.1	39.2
I3D + sub-events	35.5	37.5	38.5
I3D + super-events	38.7	38.6	39.1
I3D + sub+super-events	38.2	39.4	40.4
InceptionV3	31.2	31.8	31.9
InceptionV3 + max-pooling	31.8	34.1	35.2
InceptionV3 + pyramid	32.2	35.1	36.8
InceptionV3 + LSTM	32.1	33.5	34.1
InceptionV3 + temporal conv	28.4	34.4	33.4
InceptionV3 + sub-events	32.1	35.8	37.3
InceptionV3 + super-events	31.5	36.2	39.6
InceptionV3 + sub+super-events	34.2	40.2	40.9

7 Conclusion

This paper introduces MLB-YouTube, a novel and challenging dataset designed for detailed activity recognition in videos. We conduct a comparative analysis of various recognition techniques that employ temporal feature pooling for both segmented and continuous videos. Our findings reveal that learning sub-events to pinpoint temporal regions of interest significantly enhances performance in segmented video classification. In the context of activity detection in continuous videos, we establish that incorporating convolutional sub-events with a super-event representation, creating a three-level activity hierarchy, yields the most favorable outcomes.