This ECCV 2018 paper, provided here by the Computer Vision Foundation, is the author-created version.
The content of this paper is identical to the content of the officially published ECCV 2018
LNCS version of the paper as available on SpringerLink: https://link.springer.com/conference/eccv

Teaching Machines to Understand Baseball Games: Large-Scale Baseball Video Database for Multiple Video Understanding Tasks

Minho Shim*[0000−0002−9637−4909], Young Hwi Kim*[0000−0001−6863−8551],
Kyungmin Kim*[0000−0002−5167−0683], and Seon Joo Kim[0000−0001−8512−216X]

Yonsei University
{minhoshim,younghwikim,kyungminkim,seonjookim}@yonsei.ac.kr

Abstract. A major obstacle in teaching machines to understand videos is the lack of training data, as creating temporal annotations for long videos requires a huge amount of human effort. To this end, we introduce a new large-scale baseball video dataset called the BBDB, which is produced semi-automatically by using play-by-play texts available online. The BBDB contains 4200 hours of baseball game videos with 400k temporally annotated activity segments. The new dataset has several major challenging factors compared to other datasets: 1) the dataset contains a large number of visually similar segments with different labels. 2) It can be used for many video understanding tasks including video recognition, localization, text-video alignment, video highlight generation, and data imbalance problem. To observe the potential of the BBDB, we conducted extensive experiments by running many different types of video understanding algorithms on our new dataset. The database is available at `https://sites.google.com/site/eccv2018bbdb/`

Keywords: Video Understanding · Large-Scale Video Dataset · Action Recognition · Temporal Localization

1 Introduction

As from the old saying "Seeing is believing," paintings, photos, and videos are all produced to deliver what humans see. The ultimate goal of computer vision is to make machines to understand those visual media, and due to the rapid progress in the deep learning technology, we have now reached a point where we can teach a machine to understand a single image fairly well.

Among the visual media, videos are the most comprehensive media that most resembles how we as humans perceive the visual world. However, making machines to understand videos is still very challenging due to the additional temporal dimension. Videos include varying length of events and separating between different classes of similar actions require better understanding of the motion. For example, classifying

* indicates equal contribution

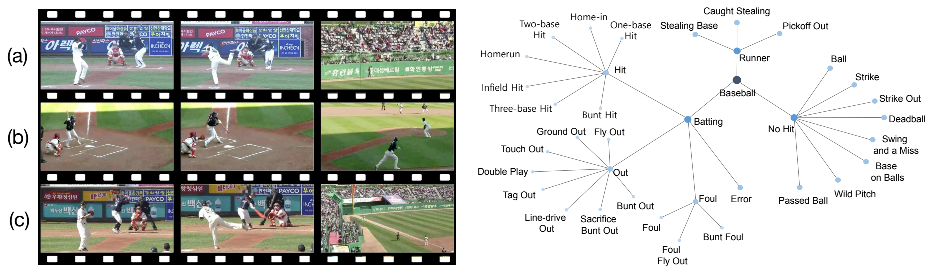

Fig. 1. *(left)* Understanding baseball videos: (a), (b), and (c) show a few samples from homerun, infield hit, and foul sequence, respectively. It is simply not enough to recognize a few frames or a discrete set of actions like hitting or running, to fully understand a baseball game or in general any video. *(right)* Semantic class hierarchy based on the baseball rule.

between walking and swimming may be relatively easy as the classification can rely solely on the visual features. For such a problem, just one single image may be enough to carry out the classification task. However, separating between similar activities like walking and running requires good motion features on top of the visual features. In addition, inferring the temporal progression of an event is another factor that needs to be accounted in video recognition. By looking at a man running, a system cannot easily determine whether it will end up with a vault or a long jump.

The goal of this paper is to introduce a new large-scale video database to promote the research in video understanding. Many video datasets have been introduced already [34,21,28,14,3,52,24,33,26,42,25,22,15,37,1,16,41,7]. However, most of the existing databases fall short when trying to learn the minute difference between similar actions. The class labels in the existing database are quite distinct such as vacuuming floor, cleaning toilet, and cleaning windows. While they provide good data to learn good visual and motion features for video analysis, we are more interested in providing data with much more similarity.

Thus we present a new large-scale video dataset called the BBDB (stands for **B**ase**b**all **D**ata**b**ase). BBDB contains 4,200+ hours (more than 500 million frames) of baseball game videos with 400k+ temporally annotated activity segments. The temporal boundaries have been annotated by making use of the play-by-play texts available online with minimal human validation, reducing the huge amount of human labor necessary for the labeling process. We categorize the actions in baseball games into 30 classes that include strike pitch, ball pitch, single/double/triple hits, homerun, etc (Fig. 1). What sets our database apart from the previous datasets is the visual and the motion similarities between the classes. A strike pitch and a ball pitch are visually similar and differentiating between the two can be difficult even for humans.

Our dataset can be used to solve other interesting problems in computer vision in addition to video recognition and localization. Class-imbalance is inherently imposed in our dataset; e.g. home-run is rare compared to strike and ball. Learning with imbalanced class sizes is a very important problem in machine learning that have yet been looked at in depth. Our dataset provides a natural opportunity to tackle this imbalance problem.

We have also collected corresponding highlights for each game in the dataset. This could be used for video highlight generation research.

2 Related Work

Database. Numerous video databases have been introduced to boost up the capability of video understanding models. Datasets for the action recognition [42,25,22,15] have been widely used, but those benchmarks provide temporally trimmed video around the action, limiting the practical use for various video understanding tasks.

Sports-1M [21] and YouTube-8M [1] introduce untrimmed video datasets, providing more complete data for realistic video understanding. KTH [34], THUMOS'15 [14], ActivityNet [3], and Charades [37] provide untrimmed videos with temporal locations, but most classes are visually distinct so that inferring an action class may depend on a few frames rather than understanding the whole sequence. Also, [14,3] contain only a small number of action instances per video; 1.1 and 1.41 annotations on average, respectively. MultiTHUMOS [52] is an extended version of THUMOS with the goal of providing multi-labeled annotation. While the dataset includes fine-grained classes such as basketball dunk, dribble, and guard, the number of videos of those classes is 420 on average, which still falls short of training a deep neural network for understanding visually similar classes. Relevant datasets [33,26] capturing fine-grained human actions exist, but those datasets consist of relatively short videos. Meanwhile, [16,41,7] provide useful benchmarks for the video summarization tasks. These datasets ask annotators to score units of video clips depending on annotators' criteria of importance. The labeling process requires many annotators per video, resulting in small size dataset.

Our large-scale video dataset BBDB provides more than 4,200 hours of videos. Out of 30 activity classes in our dataset, 23 classes have more than 1000 video clips and the other 7 classes also have more than 400 clips on average. Furthermore, an average length of the untrimmed videos in our dataset is 3.6 hours, making our benchmark more challenging as the models have to understand longer sequence of events. Each video is accompanied by its corresponding highlight video, and therefore our database can be used for the video summarization or highlight generation as well. Finally, our dataset is collected through a semi-automatic process with minimal human effort, so it is easy to scale up the size of the dataset.

Action Recognition. Before the deep learning era, handcrafted motion features like the improved dense trajectories [48] were widely used to extract appearance and motion features. One popular approach to learn spatio-temporal representation is to exploit 3D convolution. In the early stage, [45,20,46] applied simple 3D convolution network on action recognition. Recently, deeper 3D ConvNets [4,32,47,17] are proposed showing the state-of-the-art performance by inflating the well-known 2D networks (e.g. ResNet) into 3D. Another branch of video representation learning is the two-stream method [39] consisting of two complementary networks, appearance and motion network. Variations of the two-stream [12,10,11] were also introduced, exploring various ways of fusing the two streams. Aforementioned methods leverage only a fixed length of frames and a video level representation is obtained by computing the average score of segments. To model long-term temporal information, [9,29,44] employed RNN on top of the CNN and more sophisticated schemes such as TSN [50] have been proposed to watch the entire video during training.

Handling Imbalanced Dataset. Real data are imbalanced by nature. We see people walking all the time but rarely see a person back-flipping. Being able to deal with this problem of imbalance in the number of data per class is an important problem in machine learning [18], but it has yet been explored extensively. Classic approaches include heuristic sampling or adjusting cost functions to reflect the frequency of classes [5,23]. One could also use the focal loss [27] that has been proposed recently, which adjusts the cross entropy loss based on whether a class is well classified or not.

Temporal Action Localization. Temporal action localization is the problem of extracting target video segments in untrimmed videos. The basic approach for this problem [13,49] is to divide clips by a sliding window, extract features from the clips, and pass it to a classifier. Various deep learning based solutions [54,53,36] have also been introduced to solve this problem. The precision of the temporal action localization task is still low compared to other tasks. It has been pointed out that the main reason for the lack of precision is the lack of data, due to the difficulty of annotating the dataset [54].

Text-Video Alignment. Collecting dense annotations of action is costly and time-consuming. Several approaches were proposed to learn temporal localization in a weakly supervised manner. The goal is to label each frame with the corresponding action label, given only the sequence of actions without exact time stamps. An extended CTC framework [19] was proposed to evaluate all the possible alignments, enforced to be consistent with inter-frame visual similarities. Another approach [2] formulated the problem as a convex relaxation on discriminative clustering under ordering constraints.

3 Baseball Database

The goal of our Baseball Database (BBDB) is to provide a challenging benchmark for higher level understanding of videos. Previous datasets have focused on literal human actions such as running, and jumping. Only a few datasets have elaborate labels on videos; e.g. dense detailed labels [52,37], or dense caption [24]. When a sports game is analyzed with a visual recognition system trained on those simple actions, retrieved sequences of human actions will not be comprehensive enough to understand the game. This is because a sports game is a series of events, which can only be explained by a combination of action, sequential, and semantic information.

Constructing a large video dataset is challenging, especially when tasks require annotated temporal boundaries. Labeling videos involves tremendous amount of human effort, such that automating such process is one of the goals of an action detection algorithm. Labels of BBDB are collected in a semi-automatic manner making use of the play-by-play broadcasts available online. With this strategy, we could dramatically reduce human labor and create precise temporal annotations.

Among different video domains, baseball has a number of advantages over others. First, baseball has less anomalies due to its clear-cut rules and abundant statistics. This is why the play-by-play texts from the broadcasters can be utilized to generate precise segment locations. The rules are also crucial in the validation step, to analyze whether a system correctly understood the events and the underlying rules; e.g. strike-out can only

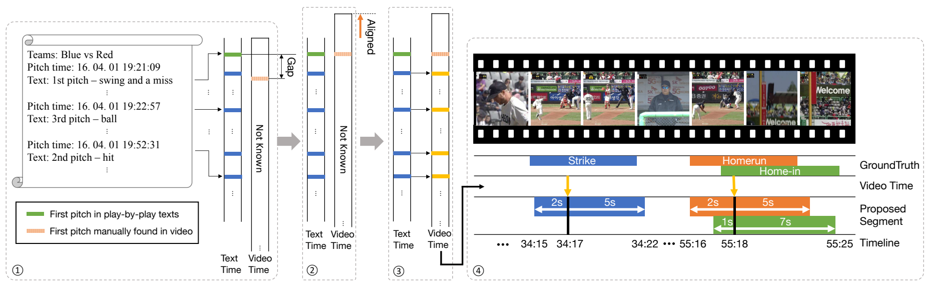

Fig. 2. Illustration of how BBDB has been collected. Left-top scroll is an example of play-by-play texts. ① From the texts, extract date and time of every pitch in a sorted order. In the video, manually find the first pitching moment. ② Using the obtained gap, align the first pitch of the video with the first pitch of the ordered text times. ③ Since text time and video time are aligned, all pitch times can be transferred from text time to video time. ④ The temporal boundaries are set with the pitch times and predetermined length of each action.

come after two strike counts. Second, thousands of new games are played every year, so the database can be easily expanded over time with minimal costs. Finally, baseball, as one of the most popular sports, has a lot of practical applications as well as high demand for automatic analysis tools.

3.1 Database Collection

Nowadays, full videos of baseball games can be found in online video archives, accompanied by play-by-play texts. Those text broadcasts (Fig. 2) include broad information about each game like the participating teams and players, and most importantly, every game activities and time stamps of pitches. With the provided pitch times, which are in absolute time with date, we can semi-automatically align the game video with the texts by just manually calculating the relative difference between the first pitch in the video and the text broadcast. Then, action segments can be extracted from videos based on pitch times and predetermined length of each action.

To make sure that the labels collected with our method are correct, we incorporate a review process for each game by checking the correctness of last few segments and their labels. If the gap is consistent throughout the game, the last segment will have correct boundaries. This is why annotators only need to check a few of last segments in each game. If there exists an inconsistency on the label (time annotation) of the last segment, the whole game is discarded in our database to ensure that none of the incorrect labels get included in the dataset.

To evaluate the semi-automatic labeling, we compared the semi-automatically labeled data with human annotation on 20 games. The details are explained in Section 4. On average, annotating 3 hours of video takes around 4 hours while our semi-automatic method takes about 5 minutes per game. This is because of the difficulty in temporally annotating untrimmed videos; after observing an event, an annotator has to seek back

Table 1. Comparison with other recognition, temporal localization (detection), and summarization video datasets. Our BBDB has peculiarities dedicated for new challenges of video analytics. † denotes the number of temporally annotated instances, available only for databases with detection tasks. ‡Database with large #instances/#videos ratio can be used as sequential alignment database, since text-video alignment is trained without temporal boundary annotations. Values are referenced from each dataset or [24] otherwise.

Dataset	#instances†/#videos	Avg.Len.	Untrimmed	Detection	Sequence‡	Summary
UCF101 [42]	/13k	7s	-	-	-	-
HMDB51 [25]	/7k	3s	-	-	-	-
Sth-Sth-V2 [15]	/220k	4s	-	-	-	-
Kinetics [22]	/306k	10s	-	-	-	-
SumMe [16]	/25	240s	-	-	-	✓
TvSum [41]	/50	150s	-	-	-	✓
VSUMM [7]	/100	180s	-	-	-	✓
Hollywood2 [28]	/4k	20s	✓	-	-	-
Sports-1M [21]	/1.1M	300s	✓	-	-	-
Youtube-8M [1]	/8.3M	230s	✓	-	-	-
KTH [34]	2.4k/600	20s	✓	✓	-	-
THUMOS'15 [14]	23.1k/21k	4s	✓	✓	△	-
MultiTHUMOS [52]	39k/400	270s	✓	✓	△	-
ActivityNet [3]	28k/20k	180s	✓	✓	△	-
Charades [37]	67k/9.8k	30s	✓	✓	✓	-
MPII cooking [33]	5.6k/44	600s	✓	✓	✓	-
TUM Breakfast [26]	11k/2k	140s	✓	✓	✓	-
BBDB (ours)	**405k**/1k	**13,000s**	✓	✓	✓	✓

to find the starting or ending point of the event. Repeating this process causes the labeling to take more time than the video's duration. However, our collection method only requires to find a gap, and to easily validate by observing a few last segments.

3.2 Properties

The BBDB is a challenging dataset for two key properties: 1) the dataset contains a large number of visually similar segments with different labels, that have evidently distinguishable differences. In baseball, strike and ball are basically the same pitching action. However, the last position of the ball, referee's movement, and on-screen graphics are distinctive cues. To solve this problem, a system must be more than just a action-category classifier, by taking more temporal information and semantics into account. 2) The number of segment instances for each class is imbalanced. Since thousands of baseball games are used for the dataset, this imbalance is naturally imposed and statistically meaningful.

Videos. The initial version of BBDB contains 1,172 full baseball game videos. It is split into three sets; training set with 703 videos, validation set with 234 videos, and test set

with 235 videos. In total, the dataset contains 4,254 hours of baseball. Each video is either 480p or 720p, and it is mostly in 30 fps, with a few 60 fps videos. Game duration ranges from 120 minutes to 350 minutes.
Highlights. Beside full game videos, we have also collected highlight videos that corresponds to each game in the dataset. Automatic generation of sports highlight is easier to evaluate compared to highlights of general user created videos, since sports highlight has less ambiguity about 'what is important' criteria. Even though those highlights are still cherry-picked by human experts, rare events like homerun or double-play are definitely more important than the others. BBDB can also serve as a good highlight database, providing a challenge of creating highlights not only through visual understanding but also by understanding storylines.
Annotations. The BBDB contains 404,964 annotated segments over 30 baseball activities. This is 345 activity instances per video on average. Labeled classes are not written as discrete human actions, but as baseball activities e.g. strike, ball, and homerun. Derived annotations comprise lexicon of labels. The lexicon has a tree structure to help semantically subdivide labels into groups (Fig. 1). The structure can be utilized to deal with disproportion of the number of instances per action class, as well as the visual similarity between two or more activities over the temporal domain.
Comparisons. For a brief comparison with other datasets, refer to Table 1. The BBDB has unparallel number of temporally annotated segments, with long full game videos. This enables the dataset to be used not only for temporal localization tasks ('Detection' in the table), but also for the text-video alignment. The alignment task does not use the temporal boundaries of the segments but the order of segments. So higher $\frac{\#\text{instances}}{\#\text{videos}}$ ratio gives more sequential information useful for the alignment task. Additionally, the BBDB has incomparable number of videos to be used for summarization or highlight generation task compared to other video summarization datasets.

Currently, we utilize a set of words as labels to evaluate the dataset. Those labels can be easily extended to sophisticated captions by making use of rich information in the play-by-play texts. The texts even contain information about rosters, number of inning, whether a player is on the first base or not, and so on. Ultimately, automatic commentary system can be established with those captions.

4 Database Evaluation

To validate our semi-automatically collected dataset, we evaluated the dataset against manually annotated videos. We randomly selected 20 baseball games in the dataset, and manually annotated about 7,000 temporal boundaries (7k is already comparable to other localization datasets).

Results of the comparison between the semi-automatic and the manual labels are shown in Table 2. For IoU threshold ≤ 0.5, the semi-automatic label shows very accurate results. While the accuracy drops for thresholds over 0.5, it is still quite accurate considering that the state-of-the-art methods do not reach even 30 on these measures. In addition, a high IoU threshold is a very strict measure that is sensitive to fine difference in boundaries. On the class *strike*, for example, the semi-automatic labeling makes a segment between one second before starting pitch and one second after the catcher

Table 2. Precision of semi-automatic labeling against human labeling on the BBDB. IoU threshold ranges from 0.3 to 0.7.

IoU threshold	0.3	0.4	0.5	0.6	0.7
BBDB	98.8	97.0	93.5	75.6	60.1

throws back to catcher. However, human annotators have their own standard for making segments, e.g. between right after starting pitch and right before the catcher throws back. These minor differences in the boundaries do not affect the ultimate goal of our vision task, since all of them are visually correct. Note that labeling the data with many annotators will inherently have difference as exact start and finish time can be different from person to person. Compared to manual labeling, the semi-automatic labels will have more consistency. Therefore, we conclude that the semi-automatic labeling is accurate enough with its own standard for making boundaries.

5 Video Understanding Algorithms

We evaluate several video understanding methods using our dataset. We first explain the methods in this section.

5.1 Action Recognition

IDT+FV. Dense trajectories [48] features include local descriptors like histogram of oriented gradients (HOG), optical flow (HOF), and motion boundary histogram (MBH). Albeit computationally expensive, Fisher vector [31] encoding of those descriptors has been used for video classification, and showed better results compared to their contemporary bag-of-words features. We used HOG, HOF, MBH descriptors and followed the configuration of feature encoding in [48]. Due to storage limitation and feasibility, all videos are set to 3 fps and frames are resized with fixed height of 240 pixels, while width is resized maintaining aspect ratio. The extracted IDT from training set is about 17TB even after resizing and reducing fps. Then, we trained 1-vs-rest SVM classifiers over Fisher vectors of all annotated segments of the training set. The classifiers are trained with Stochastic Gradient Descent (SGD) since the training data is over 10TB and cannot fit into memory.
Single Frame. We utilize the 16-layer VGG model [40] to see how a single frame based video classification and frame-level detection works. This network is originally trained on ImageNet [8] for image classification, so it provides comparisons with other systems that takes temporal changes into account. We fine-tuned the model on frames from training set segments over 30 classes.
Optical Flow Stacking. In action recognition, a flow of movement can be an important cue. This method only utilizes the optical flow information so that we can see how motion impacts the classification performance. First, we extract optical flows from clips for every 5 frame and normalize it to [0,255], which allows storing the optical flow as

an image. Then, the optical flow network is trained on a stack of 10 optical flow frames. We utilize ResNet-50 model to train motion stream network.

Two-Stream. Appearance and movement are complementary. Two-stream method utilizes both appearance and optical flow, so we can see how well two information bring together a synergy. The network designs for extracting the spatial and the temporal features are the same as Single frame and Optical flow stacking. For aggregating the results of the two networks, we average the softmax output and obtain the final result.

CNN+GRU. One of the useful tool to utilize the temporal information is recurrent neural network (RNN). In action recognition task, combinations of CNN and RNN [9] are widely used. We used 5 CNN layers to extract spatial features for each frame, and those features are fed into a RNN layer to make temporal features. We select the gated recurrent unit (GRU) [6] for its efficiency. The extracted temporal features are used as the input to a fully-connected layer and the softmax layer for the classification.

C3D. C3D is a 3D convolution network architecture in the early stage of 3D ConvNet. C3D consists of 8 convolution of 3 x 3 x 3 kernels, 5 max-pooling, and 2 fully-connected layers. Its input is 16 sequential frames so that the model directly learns spatio-temporal representations within 16 frames. We train C3D from scratch not from the pre-trained weights on Sport-1M, since all the clips point to 'Baseball' class at the very first stage leading the network to stay in a local minima.

I3D. I3D was first introduced in [4]. Unlike previous 3D ConvNets, I3D inflates not only 2D kernels into 3D, but also the 2D weight values pre-trained on ImageNet into 3D. Specifically, each $t \times k \times k$ 3D kernel is initialized by pre-trained $k \times k$ weights repeating t times along the time dimension and rescaling by $1/t$. We follow the 3D network architecture used in [51].

5.2 Handling Class Imbalance

To address the class imbalance in our dataset (Fig. 5), we ran experiments with five methods. All following experiments is based on the CNN+GRU model in 5.1, with variations in data selection, class structure, and the loss function.

Naïve Training. As the baseline of this experiment, we kept the dataset untouched and trained using CNN+GRU model.

Oversampling. Due to the highly imbalanced distribution among classes, we first randomly pick a class, and then retrieve a video in that class. This allows the model to learn each video class with equal chance. We set this method as our default setting, applying to all the action recognition experiments except Naïve training mentioned above.

Hierarchical Classification. We can hierarchically divide 30 classes as shown in Fig. 1. The distribution of the subsets are relatively balanced compared to that of the whole set. Following the hierarchy of the dataset, we trained the first level 3-class classifier, the batting-level 4-class classifier, and the last subset classifier.

Class Weight Adjustment. When a distribution among the classes is known, an additional weight can be imposed to loss values. Following [43], we tried setting a weight for each class according to the ratio of total number of samples and the number of samples from the class.

Focal Loss [27]**.** To prevent the cross entropy being overwhelmed [27] by severe class imbalance, we tried another approach by adding the focal loss balancing factor so the network focuses more on poorly classified examples:

$$\mathcal{FL} = -\sum_{i=1}^{c} (1 - p_i)^{\gamma} y_i \log(p_i) \tag{1}$$

where c, i denotes the number of classes and class indices respectively, $y_i \in \{0, 1\}$ specifies the ground-truth class, and p_i denotes class prediction probabilities. The parameter γ controls the rate at which well-classified examples are down-weighted.

5.3 Temporal Localization

A temporal localization task is to predict the start and end points of events as well as the corresponding class of each event. There are various designs of methods to localize segments. One way is to first propose candidate segments and then to classify those segments. The main drawback of this approach is losing the preciseness on the temporal domain. Another method is to evaluate every frame before grouping neighboring frames with high predicted probabilities. In this case, grouping can be exceedingly heuristic and maintaining temporal information around each frame becomes the main challenge.

We use Single frame model trained in Section 5.1 to evaluate the temporal localization task. We also evaluate using Convolutional-De-Convolutional (CDC) filter [35]. Most of settings are the same as [35], but without using pre-trained weights from Sports-1M dataset [21] for the same reason as C3D in Section 5.1. Using those two models, every frame in full game videos in the test set is fed into the models to produce class probabilities. Then, we use the sliding window approach [30] with window sizes of 5, 6, $\cdots$, 15, and 16 seconds, sliding with a stride of 1/3 second over the predicted probabilities. The detected windows are non-maximum suppressed based on each window's maximum class probability, to remove any overlapping detection. All videos are set to 3 fps before evaluation for equal comparison.

5.4 Text-Video Alignment

In this task, we apply the method of ordering constrained discriminative clustering (OCDC) [2] on our dataset. The method uses the idea of discriminative clustering with an order of actions as the constraint. OCDC solves alignment problem by jointly learning a classifier per each action. A loss function for discriminative clustering is a square loss function and we use a linear classifier to make the objective function quadratic. This allows us to apply a convex relaxation of our problem using Frank-Wolf Algorithm. We use HOF descriptor encoded by Fisher vector as frame-level representation.

6 Experimental Results

6.1 Action Recognition

In this section, we evaluate the action recognition methods with our BBDB dataset. Each clipped segments in the test set is evaluated to produce prediction probabilities, or

Table 3. Evaluations of technical approaches applied to BBDB. l denotes the maximum sequence length of GRU, and Jac stands for Jaccard measure.

Method	mAP	Input # frames	
Action Recognition (Oversampling)		Training	Testing
IDT + FV [48]	23.6	1 rgb	25 rgb
Single frame [40]	35.0	1 rgb	25 rgb
Optical flow stacking [39]	36.9	10 flow	250 flow
Two-stream [39]	42.3	1 rgb, 10 flow	25 rgb, 250 flow
C3D [46]	40.2	16 rgb	160 rgb
I3D [4]	44.2	32 rgb	320 rgb
CNN+GRU [9] ($l = 64$)	36.2	$\leq$ 64 rgb	$\leq$ 64 rgb
CNN+GRU ($l = 128$)	52.8	$\leq$ 128 rgb	$\leq$ 128 rgb
CNN+GRU ($l = 256$)	**62.8**	$\leq$ 256 rgb	$\leq$ 256rgb
Handling Imbalance ($l = 256$)			
CNN+GRU (Naïve)	67.0		
CNN+GRU (Oversampling)	62.8		
CNN+GRU (Hierarchical)	50.4		
CNN+GRU+FL [27] ($\gamma = 2$)	61.6		
CNN+GRU (Weight Adjustment)	55.3		
Per-frame Labeling		Training	Testing
Single frame	9.25	1 rgb	1 rgb
CDC [35]	**23.3**	32 rgb	32rgb
Text-Video Alignment	Jac		
OCDC [2]	7.0		

confidence scores in the case of the SVM classifier. All clips are sorted with the scores to compute the average precision (AP), then the APs are averaged with the number of classes to compute mean AP (mAP). Table 3 shows mAP of each method.

Exploiting both the appearance and the motion information shows better performance than using only one of them. C3D shows slightly lower performance compared to the Two-stream network, but I3D outperforms Two-stream, which reveals the importance of network depth, initialization method, and temporal resolution. CNN+GRU with maximum sequence length of $l = 256$ shows a large performance improvement compared to other methods.

Our experimental results are different from other works, where it has been shown that Two-stream or a 3D Convolution based networks usually work better than the CNN+GRU. We believe this is due to the difference in the nature of the dataset. Due to the visual similarity among the classes in our dataset, the classifier should take fine-grained features, e.g. trajectory of the ball, runner's direction, or referee's actions. CNN-based models take a limited number of frames as input during training, which is too short to express longer sequences. However, the RNN structure takes the whole frames

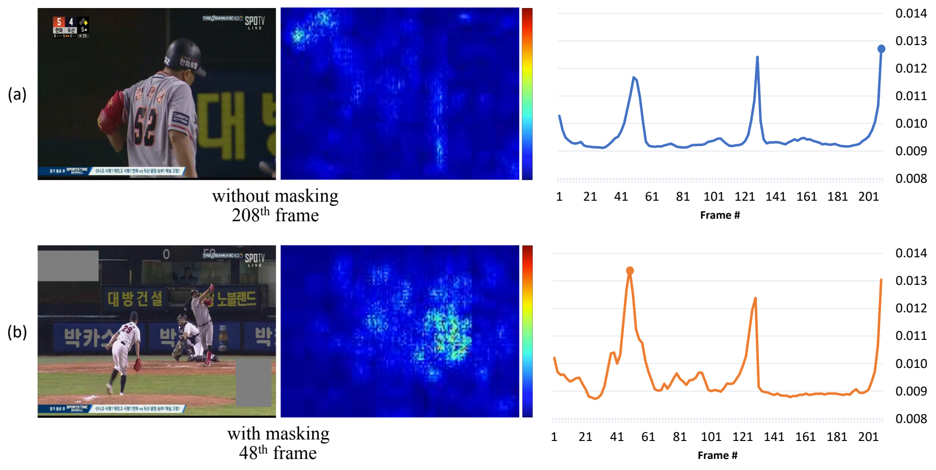

Fig. 3. Saliency on the class *foul* with corresponding video frame: saliency of CNN+GRU trained (a) without masking, (b) with on-screen socreboard masking. 48th and 208th frame are the most responsible moments for the *foul* clip with and without masking, respectively (marked as dots on the line graphs).

of the segment as input, so it is able to consider the fine-grained motion information better. In the same context, GRUs with shorter sequence length of $l = 64, 128$ show worse performances.

We also observed the relationship among the classes (Fig. 6). Ball and strike classes are the most confusing part, even though those classes have a lot of training examples. It shows that distinguishing a ball from a strike is challenging. There are more classes that are easily confused, but most extreme cases include tag out, error, and bunt out/hit classes. In a baseball game, these situations may occur simultaneously with other situations leading to the confusion.

6.2 Saliency Analysis

As an attempt to understand what the neural net is looking at when recognizing actions, we provide further analysis through the saliency map [38]. Instead of computing the saliency map in the spatial domain as is done in [38], we extend the idea to the temporal domain to see where and when the network focuses to recognize actions.

A saliency result for CNN+GRU is shown in Fig. 3(a). From the results, it is clear that the network is relying heavily on the scoreboard to make a decision. So the network learned to cheat as the easiest and the most accurate way to classify a strike or a ball event is to actually look at the scoreboard rather than the motions.

To prevent the network from cheating, we trained the network with the scoreboard masked out. A saliency results with the scoreboard masked is shown in Fig. 3(b). After masking, the network focuses more on the motions to recognize actions. With the score board masked out, the accuracy for recognizing the *ball* event went down from 0.908 to

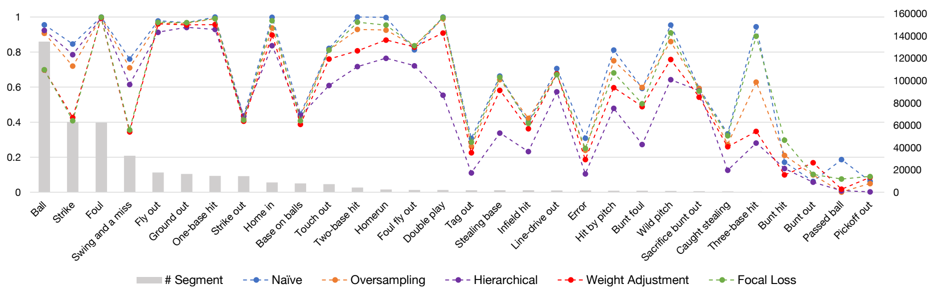

Fig. 4. APs (left axis) of classes computed with imbalance handling approaches; Naïve, Oversampling, Hierarchical classification, and Focal loss. Bars indicate the number of segments for each class (right axis). Points connected with dashed lines show AP for each class, and the dashed lines are drawn for illustration to easily seek between points. Focal loss has achieved better performance than oversampling in small-quantity classes, by improving 3.9% on average over 19 classes. However, it lost significant performance over 3 large-quantity classes by 29.2%.

0.719, and the *strike* event from 0.720 to 0.394. For the events on which the scoreboard does not provide additional information, the accuracy was similar with the masking.

6.3 Handling Class Imbalance

For handling the class imbalance, APs for every class are shown in Fig. 4 since mAPs in Table 3 can be misleading when understanding and comparing the imbalance handling methods. Oversampling failed to address the imbalance, getting worse AP in every class compared to the naïvely trained network. The gap was bigger in classes with very large and small number of samples like strike, and passed-ball. We suppose this is because the severe imbalance between classes gave the network less chance to learn the visual similarities between difficult classes like ball and strike.

Hierarchical classification showed the worst results among imbalance handling approaches. The irreversible property of the hierarchical structure mainly affected the performance. For instance, if *hit by pitch* is classified into runner or batting category, it loses any chance of being detected as *hit by pitch*. The results also present that the top classifiers could not distinguish the semantic differences of the topmost categories.

The network trained with the focal loss is based on the default oversampling technique, and it showed improvements throughout most of classes with less samples. Improvement was 3.9% on average, over 19 classes with less samples. However, the focal loss actually failed to discriminate visually similar classes like ball, strike, and swing-and-miss; losing 29.2% on average over 3 classes with large number of samples. We can infer the focusing factor $(1 - p_i)^\gamma$ kept the network from learning difficult classes by cutting down the loss. We also tried different γ values of 0.5, 1, 2, and 5 but they showed little differences.

Class weight adjustment shows the second worst performance. This is expected since setting weight according to the number of classes is a more pre-hoc way of tuning loss values, compared to focal loss that balances the loss while training.

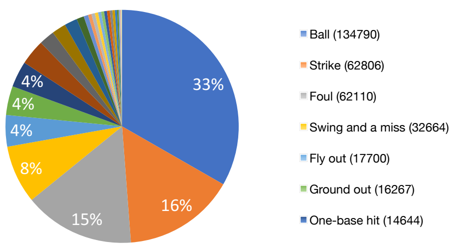

Fig. 5. Distribution of 30 classes in BBDB.

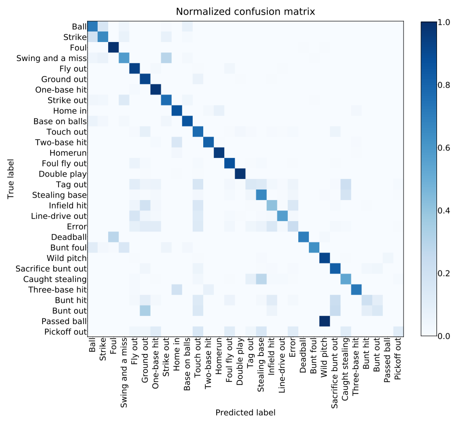

Fig. 6. Confusion matrix for CNN+GRU result. Ball and strike classes are confused each other. Tag out, error, bunt out/hit and those not having enough train data do not work correctly.

Table 4. Temporal localization mAP on BBDB. IoU threshold ranges from 0.3 to 0.7.

IoU threshold	0.3	0.4	0.5	0.6	0.7
Single frame	9.96	7.86	3.44	2.48	1.62
CDC [35]	**26.1**	**22.2**	**11.3**	**9.54**	**6.07**

6.4 Temporal Localization

Table 4 shows temporal activity localization mAP. The proposed temporal boundary is counted as true, when the boundary's Intersection-over-Union (IoU) with ground truth is bigger than the threshold. Compared with the Single frame model, the CDC shows better performance both in per-frame labeling (Table 3) and temporal localization tasks.

6.5 Text-Video Alignment

We apply OCDC on parts of the test set to show the expandability of BBDB on Text-Video Alignment. It is meaningful only on benchmarks where the number of instances per videos is large. Since representations of one simple video is too large, we divide fullgames by the end of the innings. Even with the shortened videos, OCDC results in 7.0 by Jaccard measure, which is a relatively low accuracy over the result on a subset of Hollywood2 dataset [2], which is around 45.

7 Conclusion

We have introduced our very large-scale BBDB with annotation made with minimal human labor. BBDB can be applied to video understanding tasks like action recognition, temporal action localization, text-video alignment, video highlight generation, and data imbalance problem. BBDB has a great deal of visual and motion similarity between the classes, and the class distribution follows natural statistics. We plan to develop novel video understanding algorithms using our BBDB and extend it to other video domains.
Acknowledgements. This work was supported by Samsung Research Funding Center of Samsung Electronics under Project Number SRFC-IT1701-01.

References

1. Abu-El-Haija, S., Kothari, N., Lee, J., Natsev, P., Toderici, G., Varadarajan, B., Vijayanarasimhan, S.: Youtube-8m: A large-scale video classification benchmark. CoRR **abs/1609.08675** (2016)
2. Bojanowski, P., Lajugie, R., Bach, F., Laptev, I., Ponce, J., Schmid, C., Sivic, J.: Weakly supervised action labeling in videos under ordering constraints. In: ECCV (2014)
3. Caba Heilbron, F., Escorcia, V., Ghanem, B., Carlos Niebles, J.: Activitynet: A large-scale video benchmark for human activity understanding. In: CVPR (2015)
4. Carreira, J., Zisserman, A.: Quo vadis, action recognition? A new model and the kinetics dataset. In: CVPR (2017)
5. Chawla, N.V., Japkowicz, N., Kotcz, A.: Editorial: Special issue on learning from imbalanced data sets. SIGKDD Explor. Newsl. **6**(1), 1–6 (2004)
6. Chung, J., Gulcehre, C., Cho, K., Bengio, Y.: Empirical evaluation of gated recurrent neural networks on sequence modeling. In: NIPS Deep Learning and Representation Learning Workshop (2014)
7. De Avila, S.E.F., Lopes, A.P.B., da Luz Jr, A., de Albuquerque Araújo, A.: VSUMM: A mechanism designed to produce static video summaries and a novel evaluation method. Pattern Recognition Letters **32**(1), 56 – 68 (2011)
8. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: CVPR (2009)
9. Donahue, J., Hendricks, L.A., Guadarrama, S., Rohrbach, M., Venugopalan, S., Darrell, T., Saenko, K.: Long-term recurrent convolutional networks for visual recognition and description. In: CVPR (2015)
10. Feichtenhofer, C., Pinz, A., Wildes, R.P.: Spatiotemporal residual networks for video action recognition. In: NIPS (2016)
11. Feichtenhofer, C., Pinz, A., Wildes, R.P.: Spatiotemporal multiplier networks for video action recognition. In: CVPR (2017)
12. Feichtenhofer, C., Pinz, A., Zisserman, A.: Convolutional two-stream network fusion for video action recognition. In: CVPR (2016)
13. Gaidon, A., Harchaoui, Z., Schmid, C.: Actom sequence models for efficient action detection. In: CVPR (2011)
14. Gorban, A., Idrees, H., Jiang, Y.G., Roshan Zamir, A., Laptev, I., Shah, M., Sukthankar, R.: THUMOS challenge: Action recognition with a large number of classes. `http://www.thumos.info/` (2015)
15. Goyal, R., Kahou, S.E., Michalski, V., Materzynska, J., Westphal, S., Kim, H., Haenel, V., Fründ, I., Yianilos, P., Mueller-Freitag, M., Hoppe, F., Thurau, C., Bax, I., Memisevic, R.: The 'something something' video database for learning and evaluating visual common sense. In: ICCV (2017)
16. Gygli, M., Grabner, H., Riemenschneider, H., Van Gool, L.: Creating summaries from user videos. In: ECCV (2014)
17. Hara, K., Kataoka, H., Satoh, Y.: Can spatiotemporal 3D CNNs retrace the history of 2D CNNs and ImageNet? In: CVPR (2018)
18. He, H., Garcia, E.A.: Learning from imbalanced data. IEEE Trans. on Knowl. and Data Eng. **21**(9), 1263–1284 (2009)
19. Huang, D.A., Fei-Fei, L., Niebles, J.C.: Connectionist temporal modeling for weakly supervised action labeling. In: ECCV (2016)
20. Ji, S., Xu, W., Yang, M., Yu, K.: 3d convolutional neural networks for human action recognition. PAMI **35**(1), 221–231 (2013)

21. Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., Li, F.: Large-scale video classification with convolutional neural networks. In: CVPR (2014)
22. Kay, W., Carreira, J., Simonyan, K., Zhang, B., Hillier, C., Vijayanarasimhan, S., Viola, F., Green, T., Back, T., Natsev, P., Suleyman, M., Zisserman, A.: The kinetics human action video dataset. CoRR **abs/1705.06950** (2017)
23. Krawczyk, B.: Learning from imbalanced data: open challenges and future directions. Progress in Artificial Intelligence **5**(4), 221–232 (2016)
24. Krishna, R., Hata, K., Ren, F., Fei-Fei, L., Niebles, J.C.: Dense-captioning events in videos. In: ICCV (2017)
25. Kuehne, H., Jhuang, H., Garrote, E., Poggio, T., Serre, T.: HMDB: a large video database for human motion recognition. In: ICCV (2011)
26. Kuehne, H., Arslan, A., Serre, T.: The language of actions: Recovering the syntax and semantics of goal-directed human activities. In: CVPR (2014)
27. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: ICCV (2017)
28. Marszałek, M., Laptev, I., Schmid, C.: Actions in context. In: CVPR (2009)
29. Ng, J.Y.H., Hausknecht, M.J., Vijayanarasimhan, S., Vinyals, O., Monga, R., Toderici, G.: Beyond short snippets: Deep networks for video classification. In: CVPR (2015)
30. Oneata, D., Verbeek, J., Schmid, C.: The lear submission at thumos 2014. In: ECCV THUMOS Workshop (2014)
31. Perronnin, F., Sánchez, J., Mensink, T.: Improving the fisher kernel for large-scale image classification. In: ECCV (2010)
32. Qiu, Z., Yao, T., Mei, T.: Learning spatio-temporal representation with pseudo-3d residual networks. In: ICCV (2017)
33. Rohrbach, M., Amin, S., Andriluka, M., Schiele, B.: A database for fine grained activity detection of cooking activities. In: CVPR (2012)
34. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local svm approach. In: ICPR (2004)
35. Shou, Z., Chan, J., Zareian, A., Miyazawa, K., Chang, S.F.: CDC: Convolutional-de-convolutional networks for precise temporal action localization in untrimmed videos. In: CVPR (2017)
36. Shou, Z., Wang, D., Chang, S.F.: Temporal action localization in untrimmed videos via multi-stage cnns. In: CVPR (2016)
37. Sigurdsson, G.A., Varol, G., Wang, X., Farhadi, A., Laptev, I., Gupta, A.: Hollywood in homes: Crowdsourcing data collection for activity understanding. In: ECCV (2016)
38. Simonyan, K., Vedaldi, A., Zisserman, A.: Deep inside convolutional networks: Visualising image classification models and saliency maps. In: ICLR Workshop (2014)
39. Simonyan, K., Zisserman, A.: Two-stream convolutional networks for action recognition in videos. In: NIPS (2014)
40. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. In: ICLR (2015)
41. Song, Y., Vallmitjana, J., Stent, A., Jaimes, A.: Tvsum: Summarizing web videos using titles. In: CVPR (2015)
42. Soomro, K., Roshan Zamir, A., Shah, M.: UCF101: A dataset of 101 human actions classes from videos in the wild. In: CRCV-TR-12-01 (2012)
43. Sozykin, K., Protasov, S., Khan, A., Hussain, R., Lee, J.: Multi-label class-imbalanced action recognition in hockey videos via 3d convolutional neural networks. In: IEEE/ACIS SNPD (2018)
44. Sun, L., Jia, K., Chen, K., Yeung, D., Shi, B.E., Savarese, S.: Lattice long short-term memory for human action recognition. In: ECCV (2017)

45. Taylor, G.W., Fergus, R., LeCun, Y., Bregler, C.: Convolutional learning of spatio-temporal features. In: ECCV (2010)
46. Tran, D., Bourdev, L., Fergus, R., Torresani, L., Paluri, M.: Learning spatiotemporal features with 3d convolutional networks. In: ICCV (2015)
47. Tran, D., Ray, J., Shou, Z., Chang, S., Paluri, M.: Convnet architecture search for spatiotemporal feature learning. CoRR **abs/1708.05038** (2017)
48. Wang, H., Schmid, C.: Action Recognition with Improved Trajectories. In: ICCV (2013)
49. Wang, L., Qiao, Y., Tang, X.: Action recognition and detection by combining motion and appearance features. In: ECCV THUMOS Workshop (2014)
50. Wang, L., Xiong, Y., Wang, Z., Qiao, Y., Lin, D., Tang, X., Val Gool, L.: Temporal segment networks: Towards good practices for deep action recognition. In: ECCV (2016)
51. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: CVPR (2018)
52. Yeung, S., Russakovsky, O., Jin, N., Andriluka, M., Mori, G., Fei-Fei, L.: Every moment counts: Dense detailed labeling of actions in complex videos. IJCV **126**(2), 375–389 (2018)
53. Yeung, S., Russakovsky, O., Mori, G., Fei-Fei, L.: End-to-end learning of action detection from frame glimpses in videos. In: CVPR (2016)
54. Yuan, J., Ni, B., Yang, X., Kassim, A.A.: Temporal action localization with pyramid of score distribution features. In: CVPR (2016)