Towards Real-Time Detection and Tracking of Basketball Players using Deep Neural Networks

David Acuna
University of Toronto
davidj@cs.toronto.edu

Abstract

Online multi-player detection and tracking in broadcast basketball videos are significant challenging tasks. In this environments, the target distributions are highly non-linear, and the varying number of objects creates complex interactions with overlap and ambiguities. In this paper, we present a real-time multi-person detection and tracking framework that is able to perform detection and tracking of basketball players on sequences of videos. Our framework is based on YOLOv2, a state-of-the-art real-time object detection system, and SORT, an object tracking framework based on data association and state estimation techniques. For training and testing, we use a given subset of the NCAA Basketball Dataset. As part of the bonus, we trained a two-layer LSTM to do action recognition.

1 Introduction

The ability to visually detect and track multiples persons across a scene has been a long standing challenge within the Computer Vision and Machine Learning communities. In terms of sports analytics, automatic player detection and tracking is critical for team tactics, player activity analysis, camera planning and even enjoyment in broadcast sports videos.

Compared with multiple object tracking (MOT) in other scenes, multiple player tracking in sports video is much more difficult due to the following reasons: (1) players in the same team are always visually similar ; (2) sports players often interact with others in complex ways and (3) the occlusions are much more frequent and severe. All of these issues together have posed quite a great challenge to the tracking system, which requires not only reliable observations but also a sophisticated tracking strategy to make the system robust [1].

As a result, many algorithms have been proposed to deal with the multi-player detection and tracking problem in sport games. For example, in [2], the authors use mixture particle filters and Adaboost to detect and track hockey players using video sequences. Similarly, Zhu et al propose in [3] a detection and tracking framework for broadcast sports video that uses support vector machine (SVM) and particle filters [4]. The main limitation with those approaches is that in some way or another they require a selection of hand-crafted features, which most of the time, makes their scalability to even similar sports really challenging.

Recent years, however, have seen an extraordinary success of Deep Neural Networks in areas such as Computer Vision and Natural Language Processing [5; 6]. Starting with AlexNet [5] , a deep Convolutional Neural Network (CNN) trained to classify more than a million of images , we have seen more and more models that uses Deep Neural Networks to perform image classification and object recognition tasks.

Player detection and tracking frameworks have also been influenced by neural networks. They have seen a significantly improve in their performance by incorporating CNNs to their architecture pipelines. For instance, Ramanathan et al propose in [7] a detection and tracking architecture

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

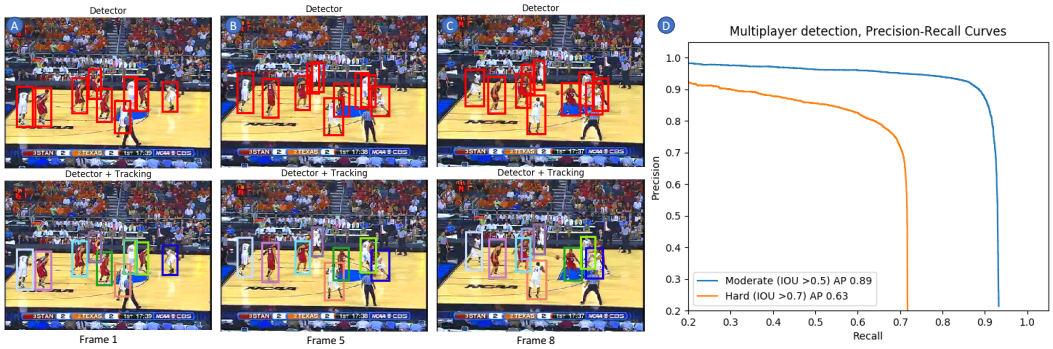

Figure 1: a, b, c) Performance of the presented framework during a sequence of frames. d) Precision and Recall curves for the online multi-player detector. A short video illustrating the performance of the architecture in the test set can be found here `http://goo.gl/ZPWStU`

composed of a CNN-based multibox detector from [8] and KLT tracker from [9]. With this approach, they outperformed state-of-the-art methods for event classification and detection on the NCAA basketball dataset.

In this paper, we present an online detection and tracking framework, that is composed by a bounding box detector and a tracking algorithm. The detector is based on YOLOv2 [10], a state-of-the-art real-time object detection system. The tracking algorithm, SORT [11], is primarily targeted towards online tracking, and it uses combination of simple techniques such as Kalman Filter [12] and Hungarian algorithm [13]. The presented approach was tested in the freely available NCAA basketball dataset [7].

2 Model Architecture

The architecture of our model constitutes of two parts: (1) a detection network and (2) a tracking algorithm. It is important to highlight that the accuracy of the detector is of paramount importance and it is identified as a key factor influencing the tracking performance.

2.1 Player detection

The architecture of our player detector was based on YOLOv2 [10]. YOLOv2 is an improved version of the darknet YOLO project [14; 15]. This model has been proved to produce state-of-the-art results in terms of both accuracy and speed. At 67 FPS, YOLOv2 gets 76.8 mAP on VOC 2007, and at 40 FPS, it gets 78.6 mAP, outperforming state-of-the-art methods like Faster RCNN [16] with ResNet[17] and SSD [18] while still running significantly faster [10].

The main idea behind YOLO is that the model only looks once to the image. In order to do that, the authors frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A neural network predicts bounding boxes and class probabilities directly from full images in one evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. Note that, YOLO divides the input image into an $S \times S$ grid. If the center of an object falls into a grid cell, that grid cell is responsible for detecting that object.

The loss function of the end-to-end model must then simultaneously solve the object detection and object classification tasks. This function simultaneously penalizes incorrect object detections as well

as considers what the best possible classification would be. Equation 1 shows the loss function.

$$\begin{aligned}\mathcal{L} = \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \Big[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \Big] \\ + \lambda_{coord} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \Big[\big(\sqrt{w_i} - \sqrt{\hat{w}_i}\big)^2 + \big(\sqrt{h_i} - \sqrt{\hat{h}_i}\big)^2 \Big] \\ + \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \Big(C_i - \hat{C}_i \Big)^2 + \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \Big(C_i - \hat{C}_i \Big)^2 \\ + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{obj} \sum_{c \in \text{classes}} \Big(p_i(c) - \hat{p}_i(c) \Big)^2 \end{aligned} \quad (1)$$

Note from equation 1 that λ_{coord} and λ_{noobj} are two independent hyperparameters. The first one increases the loss from bounding box coordinate predictions, and the second decreases the loss from confidence predictions for boxes that don't contain objects. Similarly, $\mathbb{1}_{i}^{obj}$ denotes if an object appears in cell i and $\mathbb{1}_{ij}^{obj}$ denotes that the jth bounding box predictor in cell i is responsible for that prediction.

In order to use YOLOv2 in the NCAA basketball dataset, we need to change the number of filters in the last Convolutional layer. Concretely, the number of filters need to be reduced to 30. The reason for that is that YOLO predicts 5 boxes with 5 coordinates each (x, y, w, h, confidence) and x number of classes per box. In this work, we are only interested in predicting the class 'player', then the need to readjust the number of filters to 30.

At the end of the detection pipeline, we also introduce non-maximal suppression to fix multiple detections in objects near the border of multiple cells. While not critical to performance as it is for R-CNN or DPM, non-maximal suppression adds slight an improvement in terms of AP [14; 10].

2.2 The tracking framework

The tracking algorithm that we use is a lean implementation of a tracking-by-detection framework where objects are detected each frame (using our modification of YOLOv2) and represented as bounding boxes. Since the main purpose of this work is to end up with an online detection and tracking pipeline, our tracker only relies on detection from the previous and the current frame.

Following the same idea presented in SORT [11], we approximate the inter-frame displacements of each object with a linear constant velocity model which is independent of other objects and camera motion. Concretely, the state of each target is modeled as: $\mathbf{x} = [u, v, s, r, \hat{u}, \hat{v}, \hat{s}]^T$, where u and v represent the horizontal and vertical pixel location of the centre of the target, and s and r represent the scale (area) and the aspect ratio of the target's bounding box respectively. The velocity components are solved optimally via a Kalman filter framework [12]. If no detection is associated to the target, its state is simply predicted without correction using the linear velocity model.

Another important step in this tracking framework is to correctly associate the players between frames. For that, an assignment cost matrix is computed as the intersection-over-union (IOU) distance between each detection and all predicted bounding boxes from the existing targets. The assignment is then solved optimally using the Hungarian algorithm. Additionally, a minimum IOU is imposed to reject assignments where the detection to target overlap is less than IOUmin.

For player tracking, it is also important the creation and deletion of tracking identities. To solve that, SORT creates an object identity when there is a detection with an overlap less than IOUmin. On the other hand, tracks are terminated if they are not detected for $TLost$ frames.

3 Experiments and Results

We evaluated the performance of our detection and tracking framework on the NCAA basketball dataset. Starting with the detector, we split the dataset in 8787 annotated frames for training, and 1000 annotated frames for testing.

For training, we reused the learned weights of the first layers of YOLOv2 (Darknet-19 [15]) but we randomly initialized the weights of the last Convolutional layers. The learning rate was set to 0.001, we used stochastic gradient descent and a polynomial rate decay with a power of 4, weight decay of 0.0005 and momentum of 0.9. We also need to resize the mini-batch size to 12 as we didn't have enough computational resources to process greater mini-batches. λ_{coord} and λ_{noobj} were empirically determined to be 5 and .5 respectively. The threshold for the NMS step was set to be 0.45, we tried several values, but we obtained the best performance in the testing set with this one.

Note that we didn't use a validation set. The reason is that training the detector was an extremely expensive task for our computational resources and we couldn't train several times in order to find the best hyper-parameters. That said, we selected them based on the results of previous works that use the same detector/architecture [15; 14; 10].

For evaluation, we compute the precision-recall curves (figure 1d). This curves were computed following the PASCAL criteria [19] for object detection. However, in addition to the standard PASCAL criteria (IOU>0.5), we also evaluated the model following a slightly harder criteria where we required an IOU>0.7. Figure 1d shows the performance of the detector, we obtained an $AP = 0.89$ for the standard case (IOU>0.5) and 0.63 for the harder one.

In order to find the hyperparameters of the tracking algorithm and test the output of the whole framework, we use 20 clips where each clip corresponds to approximately 20 frames of a segment of the game. These clips were carrefully selected so that they correspond to unseen data for both the detector and the tracker. The hyper-parameters of the tracking algorithm, namely the initial Kalman filter covariances, IOUmin, and TLost, were determined using the same approaches presented in [12; 11]. Figure 1a, 1b and 1c illustrate a sequences of detected and tracked players in one of the unseen clips. The colors of the different bounding boxes, in the case of the detection + tracking output, define the same player throughout the clip.

4 Limitations

Although our framework is able to accurately detect and track basketball players, we noticed that there are several situation when particular players are lost. During our experiments, this mostly happened at the end of the offensive action. At that time, the majority of the players occlude each other. However, when we checked the output of the detector, in most of the cases, we realized that all of the players were correctly detected. We believe that further improvement on the tracking side can help resolve this problem. In terms of the detector, the majority of the failure cases correspond to occlusion of one or two players. It is important to highlight, however, that those cases were extremely difficult to detect even for us as human beings.

5 Conclusion

We have presented a novel on-line multi-detection and tracking framework that is able to accurately detect and track basketball players by just looking at broadcast videos. Our detection and tracking pipeline was composed by YOLOv2, a real-time state-of-the-art detector, and SORT a simple but accurate tracking by detection algorithm. We evaluated the performance of our detector using the PASCAL VOC criteria and we showed the accurate performance of the whole pipeline.

6 Bonus

The main idea behind the bonus is to do action recognition on the given subset of clips. As commented before, these clips are a subset of the 14548 annotated clips corresponding to the NCAA Basketball Dataset. In particular, the annotated clips correspond to one of the 11 possible actions (e.g shot,pass) and their length is at most 20 consecutive frames.

We believe that temporal information is of paramount importance in order to do action recognition. Therefore, similar to [7], we model it using a Recurrent Neural Network (RNN). Concretely, we use a two-layer LSTM [20] trained on top of the features extracted from ResNet[17].

In order to do that, we used the pre-trained version of ResNet-18 available in PyTorch, we removed the last fully connected layer of ResNet and used the rest of the architecture as an encoder. Then, we added two LSTM layers with 256 hidden units and a fully connected layer on the last hidden state (256 to 11). The sequence length was set to 20 and in the cases where the length of the clip was less than 20 frames, we padded it with 0. The total number of given clips was 550. Therefore, we randomly picked 450 to be our training set , 50 to be our validation set and 50 to be our test-set. We used mini-batches of 10 due to the limitations on terms of computational resources. The optimizer used was Adam and the learning rate was set to be 0.001.

During training, we realized that the model heavily over-fitted the training data just after a few epochs. Therefore, we added a dropout of 0.5 and used data augmentation techniques. Concretely, we used Random Crops and RandomHorizontalFlip. With this techniques we obtained a slightly increased in the accuracy, however, the best accuracy we could obtained was around 33% in the test set.

We hypothesize that the main reason why this happened is because the given dataset was too small (i.e only 550 sequences) and that amount of data was not enough, resulting in a training set heavily over-fitted. We believe that training the same architecture using the whole NCAA Dataset will give more promising results.

References

[1] X. Jiang, Z. Liu, and Y. Wang, *Tracking Multiple Players in Beach Volleyball Videos*, pp. 65–71. Singapore: Springer Singapore, 2016.

[2] K. Okuma, A. Taleghani, N. de Freitas, J. J. Little, and D. G. Lowe, *A Boosted Particle Filter: Multitarget Detection and Tracking*, pp. 28–39. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004.

[3] G. Zhu, C. Xu, Q. Huang, and W. Gao, "Automatic multi-player detection and tracking in broadcast sports video using support vector machine and particle filter," in *Multimedia and Expo, 2006 IEEE International Conference on*, pp. 1629–1632, IEEE, 2006.

[4] M. S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, "A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking," *IEEE Transactions on signal processing*, vol. 50, no. 2, pp. 174–188, 2002.

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[6] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pp. 6645–6649, IEEE, 2013.

[7] V. Ramanathan, J. Huang, S. Abu-El-Haija, A. N. Gorban, K. Murphy, and L. Fei-Fei, "Detecting events and key actors in multi-person videos," *CoRR*, vol. abs/1511.02917, 2015.

[8] C. Szegedy, A. Toshev, and D. Erhan, "Deep neural networks for object detection," in *Advances in Neural Information Processing Systems*, pp. 2553–2561, 2013.

[9] C. J. Veenman, M. J. Reinders, and E. Backer, "Resolving motion correspondence for densely moving points," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 1, pp. 54–72, 2001.

[10] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," 2016.

[11] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, "Simple online and realtime tracking," *CoRR*, vol. abs/1602.00763, 2016.

[12] R. E. Kalman, "A new approach to linear filtering and prediction problems," *ASME Journal of Basic Engineering*, 1960.

[13] H. W. Kuhn and B. Yaw, "The hungarian method for the assignment problem," *Naval Res. Logist. Quart*, pp. 83–97, 1955.

[14] D. Impiombato, S. Giarrusso, T. Mineo, O. Catalano, C. Gargano, G. La Rosa, F. Russo, G. Sottile, S. Billotta, G. Bonanno, S. Garozzo, A. Grillo, D. Marano, and G. Romeo, "You Only Look Once: Unified, Real-Time Object Detection," *Nuclear Instruments and Methods in Physics Research, Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, vol. 794, pp. 185–192, 2015.

[15] J. Redmon, "Darknet: Open source neural networks in c." `http://pjreddie.com/darknet/`, 2013–2016.

[16] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," *Nips*, pp. 1–10, 2015.

[17] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," *Arxiv.Org*, vol. 7, no. 3, pp. 171–180, 2015.

[18] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015.

[19] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, pp. 98–136, Jan. 2015.

[20] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.