Electronic Letters on Computer Vision and Image Analysis 20(1):99-116, 2021

Deep Learning Based Automated Sports Video Summarization using YOLO

Chakradhar Guntuboina[1], Aditya Porwal[2], Preet Jain[1] and Hansa Shingrakhia[3]

[1] *Electronics and Communication Engineering Department, IITE, Indus University, Ahmedabad, India*
[2] *PG Student, Electrical and Computer Engineering Department, University of Calgary, Calgary, Canada*
[3] *Assistant Professor, Electronics and Communication Engineering Department, IITE, Indus University, Ahmedabad, India*

Received 12th of August 2020; accepted 14th of May 2021

Abstract

This paper proposes a computationally inexpensive method for automatic key-event extraction and subsequent summarization of sports videos using scoreboard detection. A database consisting of 1300 images was used to train (using transfer learning) a supervised-learning based object detection algorithm, YOLO (You Only Look Once). Then, for each frame of the video, once the scoreboard was detected using YOLO, the scoreboard was cropped out of the image. After this, image processing techniques were applied on the cropped scoreboard to reduce noise and false positives. Finally, the processed image was passed through an OCR (Optical Character Recognizer) to get the score. A rule-based algorithm was run on the output of the OCR to generate the timestamps of key-events based on the game. The proposed method is best suited for people who want to analyse the games and want precise timestamps of the occurrence of important events. The performance of the proposed design was tested on videos of Bundesliga, English Premier League, ICC WC 2019, IPL 2019, and Pro Kabaddi League. An average F1 Score of 0.979 was achieved during the simulations. The algorithm is trained on five different classes of three separate games (Soccer, Cricket, Kabaddi). The design is implemented using python 3.7.

Key Words: Sports video, Image detection, Image processing, Optical Character Recognizer (OCR), You Only Look Once (YOLO), Intersection Over Union (IOU), Region of Image (ROI), Mean Average Precision (mAP), Scoreboard, Key-events.

1 Introduction

Sports is a major part of the entertainment industry, and with the rise in internet accessibility in recent years, its popularity and viewership have only increased. A majority of those people find it difficult to devote time to watch full-length matches of their preferred sport in their busy lives and, as such, most individuals prefer to watch a shorter (summarized) version of the match. Manual summarization or highlight generation is a tedious and time-consuming process requiring professional video editing tools which puts an upper bound on the amount of video footage that can be summarized in a given amount of time. Moreover, video editing in

Correspondence to: guntuboinachakradhar.16.ec@iite.indusuni.ac.in

Recommended for acceptance by Angel D. Sappa
https://doi.org/10.5565/rev/elcvia.1286
ELCVIA ISSN: 1577-5097
Published by Computer Vision Center / Universitat Autònoma de Barcelona, Barcelona, Spain

general has a very steep learning curve meaning that not everyone - given access to a full-length video - can summarize a sports video without the necessary skills that are required for it. As a natural consequence of this, a lot of research work has been done to automatically summarize the videos into important video clips which are significantly shorter than full-length sports videos and thus enable the viewers to enjoy the match without having to watch the entire match. Traditional works relied on image processing, audio signal processing, computer vision techniques, etc. for video summarization. But with the recent resurgence of neural networks owing to the exponential growth of computational power, a whole new way of extracting highlights using deep learning has emerged. The objective of this paper is to exploit the computational power by using image detection and classification techniques to summarize almost any sports video. The proposed methodology in brief: use image detection and localization techniques to determine the location of the scoreboard in a video frame, crop the scoreboard, run OCR over the cropped scoreboard, and compare the score with the previous score to detect important events in the video.

2 Literature

Some of the previous work in this field include using ORB (Oriented Fast, Rotated Brief) to detect the BRS (Bowler Run-up Sequence) in cricket and classify it as a highlight [1]. Pushkar Shukla et al [2] have combined event-driven and excitement-based methods to generate highlights for cricket. In [3], they have used the intensity profile of umpire's gestures to detect the occurrence of an event. A. Javed et al [4] have used the fact that replays are generally preceded and followed by gradual transition effects to extract the replays and then pass it through an ELM (Extreme Learning Machine) for the detection of key events (Boundary, Six, Wicket). Using 3D CNNs (3 Dimensional Convolutional Neural Networks) and LSTM (Long Short-Term Memory) to extract features from a soccer video for summarization [5] & [6]. Shingrakhia et al [7, 8] have mentioned in their research work about video summarization techniques using novel deep learning algorithms such as SGRNN-AM (Stacked Gated Recurrent Neural Network with Attention Module) and HDNN-EPO (Hybrid Deep Neural Network with Emperor Penguin Optimization) for improved performance on cricket videos with 96% accuracy and precision. J. Yu et al [9] have designed an architecture that detects events and proposes temporal boundaries to the events to form what they call stories i.e. the event clip and some additional context around it. And while these are all novel methods, they all focus on a single sport for summarization and would translate poorly to other sports. Then there are those that try to generalize the summarization for multiple sports like P. Kathirvel et al [10] who have used referee's whistle sound as a cue for extraction of highlights. This would fail in games that don't require a whistle or when the whistle is inaudible due to background noise etc. In [11], Antonio Tejero-de-Pablos et al have proposed learning players' actions using deep neural networks and use that to detect important events applied to a game called Kendo (Japanese fencing). This would translate poorly to games where there are a lot of players like in cricket, soccer (football), etc. In this paper, we present a computationally inexpensive method to extract event-based highlights which could work on any game which has a scoreboard. The idea is to use a state-of-the-art object detection model called YOLO and train it to detect the scoreboard of the desired game. Then we run the model over images extracted from the video at the rate of one image per second and run an OCR over it to keep track of the score. If the score changes by some predefined delta (for example, in cricket, the delta would be set to 4/6 for the score variable and set to 1 for the wicket variable because a difference (increment) of either 4 or 6 in the score indicates that an event has occurred, similarly, a difference of 1 in wicket indicates an event and as such must appear in the summary of the match. In case of football, the delta would be set to 1 for both the score variables (team 1's score and team 2's score) and whenever the score changes by the aforementioned delta, an event is record. Similarly, for different games the delta would be set to whatever score change is required for the algorithm to detect that change as an event), then an event is detected at that frame. The simplicity of our model is such that with only 250-300 images of the scoreboard for training, anyone can deploy this on any desired sports video with the domain knowledge of the sport to fix the predetermined delta. The rest of the paper is organized as follows. Section 3 gives information about YOLO and why we have chosen YOLO, Section 4 presents the methodology, Section 5 presents the results &

performance of our model and then finally, Section 6 presents concluding remarks and limitations/future work for the model.

3 You Only Look Once

3.1 A Brief Comparison of Object Detection Algorithms

There are a number of object detection algorithms such as R-CNN [12], Faster R-CNN [13], YOLO. In this section, we discuss the reason(s) behind choosing YOLO as our go-to algorithm. This is done by briefing other algorithms, their limitations, and how YOLO counters/overcome those limitations. Region-based Convolutional Neural Networks (R-CNN): Focus is on dividing the image into 2000 small regions and then finding the object in each region, an offset is also added to improve the accuracy. Limitations: Takes a huge amount of time to classify 2000 sections. Not practical for real-time detections as each image takes approximately 47 seconds to process (classify). Faster R-CNN: The network itself learns the region proposals. It is a lot faster than R-CNN and Fast R-CNN [14] but still when it comes to real-time performance the algorithm is not suitable. It can be observed that almost every algorithm is based on three tasks to achieve object detection: 1) Use Region Proposal method to generate potential bounding boxes in an image. 2) Run the classifier on these boxes. 3) After classification, perform post-processing to tighten the boundaries of the bounding boxes, remove duplicates. These tasks prove to be very complex and hard to optimize at the same time as each of the components are required to be trained separately. This is where YOLO comes in. YOLO: This algorithm is different from all the traditional object detection algorithm in a way that it considers object detection as a regression problem rather than a classification problem. A single convolutional neural network is used to predict the bounding box as well as the probabilities. A prediction having the threshold (Maximum IOU) is considered as detection and the rest is discarded. The speed is of the order of 45 Frames per second, the fastest available algorithm as of now. Limitation: The algorithm cannot detect small objects because of the spatial constraints of the algorithm. Another limitation of the algorithm is that it lacks in accuracy compared to another algorithm such as Faster R-CNN.

3.2 How YOLO Works

The algorithm divides the input image into an S × S grid. If the centre of an object falls into a grid cell, that grid cell is responsible for detecting that object. Each grid cell predicts a fixed number of boundary boxes and confidence scores for those boxes to locate the object.
Training data is a bounding box along with the class label(s). This is referred to as 'ground truth box',

$$b = [b_{\mathrm{x}}, b_{\mathrm{y}}, b_{\mathrm{h}}, b_{\mathrm{w}}, \text{class name (or number)}] \tag{1}$$

where b_x, b_y is the midpoint of annotated bounding box and b_h, b_w is height and width of box.
Output or prediction is bounding box b along with class c for an image i. Formally:

$$y = [p_{\mathrm{c}}, b_{\mathrm{x}}, b_{\mathrm{y}}, b_{\mathrm{h}}, b_{\mathrm{w}}, c_{\mathrm{n}}] \tag{2}$$

where b_x, b_y is the midpoint of annotated bounding box. b_h, b_w is height and width of box. p_c - The probability of having class(es) c in 'box' b. c_n is a vector that corresponds to the output classes of the model (i.e. classes of objects that the model can detect) and the subscript n represents the number of neurons in the output layer of the model (i.e. the number of object classes that the model can detect) and is equal to the length of vector c (for example, say n = 2 which means that the output layer of the model has 2 neurons i.e. the model can detect 2 classes of objects (let's call them c_1 and c_2) therefore c_n = [c_1, c_2] where c_1 refers to the object class 1 and c_2 refers to object class 2, now, say the model detected object c_1 in a given image, then the value of the vector c_n would be equal to [1, 0] representing that object c_1 was detected and object c_2 was not. Similarly, if n = 4, then c_n becomes [c_1, c_2, c_3, c_4] and the object c_3 was detected, therefore c_n = [0, 0, 1, 0].

YOLO uses a linear activation function for the final layer and the following leaky rectified linear activation for the rest of the layers:

$$\phi(x) = \begin{cases} x & \text{if } x > 0 \\ 0.1 * x & \text{otherwise} \end{cases} \tag{3}$$

YOLO uses sum-squared error between the predictions and the ground truth to calculate the loss. The loss function comprises of:

- The classification loss - If an object is detected, the classification loss at each cell is the squared error of the class conditional probabilities for each class.
- The localization loss - Measures the errors in the predicted boundary box locations and sizes. Only the box responsible for detecting the object is considered.
- The confidence loss - Computes the loss associated with the confidence score for each bounding box predictor.

The final loss adds localization, confidence and classification losses together.

$$\begin{aligned} Loss = (\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \Omega_{\text{ij}}^{\text{obj}} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2] + \\ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \Omega_{\text{ij}}^{\text{obj}} [\sqrt{(w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2] + \\ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \Omega_{\text{ij}}^{\text{obj}} [(C_i - \hat{C}_i)^2] + \\ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \Omega_{\text{ij}}^{\text{noobj}} [(C_i - \hat{C}_i)^2] + \\ \sum_{i=0}^{S^2} \Omega_{\text{i}}^{\text{obj}} \sum_{c \epsilon classes} [(p_i(c) - \hat{p}_i(c))^2]) \end{aligned} \tag{4}$$

where,
$\Omega_{\text{i}}^{\text{obj}}$ = 1, if an object appears in cell i, otherwise 0.
$\hat{p}_i(c)$ denotes the conditional class probability for class c in cell i.
$\Omega_{\text{ij}}^{\text{obj}}$ = 1, if the j$^{\text{th}}$ boundary box in cell i is responsible for detecting the object, otherwise 0.
λ_{coord} increases the weight for the loss in the boundary box coordinates.
$\hat{C}_i$ is the box confidence score of the box j in the cell i.
$\Omega_{\text{ij}}^{\text{noobj}}$is the compliment of $\Omega_{\text{ij}}^{\text{obj}}$.
λ_{noobj} weights down the loss when detecting the background.

For a detailed explanation of how the algorithm works refer to [15].

4 Methodology

4.1 Training the YOLO Model

Approximately 250 to 300 images (for a single class, in total about 1300 images were used) containing score-boards of varying scores were used to train the models [16]. Approximately 200 to 240 images of the 300

images of each class were used as training data and the remaining images were used as testing/validation data. Put simply, the data was split in the ratio of 8:2 or 80% to 20% where 80% of the data was used for training and 20% for validation. The model was trained for about 11 epochs (by 11 epochs, all the models had a loss of less than 1) with the batch size set to 4 and IOU set to 0.5. In total 3 models were trained (using the pre-trained model provided by ImageAI and applying transfer learning on it), 1 for each sport (Cricket, Soccer, Kabaddi). The models were trained until the loss got down below 1. This is another advantage that YOLO provides in that it requires significantly fewer images to train on an object compared to other object detection algorithms. Since the only things that change in an image of a scoreboard are the digit fields, using 250 images for one type of scoreboard yields extremely good accuracy.

4.2 Feeding the Images to the Model

Using OpenCV we can extract one image per second (for example, for a 30FPS video, 1 out of the 30 frames in a second) from the video sequence and feed it into the model. The model outputs whether there is a scoreboard in the image or not and, if there is, it will return the coordinates of the scoreboard on the image. The rate of feeding the images is kept at 1 image per second of video sequence so that the possible error in the output timestamp of an event and the actual timestamp of the occurrence of the event is at most 1 second. While a lower rate would reduce the computational complexity and increase performance even further, it would also increase this max possible error, for example, a rate of 0.2 images per second from the video sequence i.e. 1 image per 5 seconds of video sequence would result in a max possible error of 5 seconds which is not acceptable.

4.3 Reading the Score from the Scoreboard

If the model finds the scoreboard on the image, the scoreboard will then be cropped out of the image. Then the following image processing operations will be performed on the resulting image of the scoreboard: convert the image to binary using thresholding, find contours on the image and define an ROI (region of image) around each individual contour, crop out these regions from the image to create separate images each containing 1 digit or character. All of these regions will then be fed to a trained OCR [17] which will return the score in the current frame. There are many OCRs available online but none of them had satisfactory results in reading the scoreboard due to the fact that after cropping, the image of the scoreboard becomes very small, and as such its resolution also decreases. Hence, we trained a convolutional neural network (consisting of a single hidden layer) on the digits of the 5 classes that we have used in this project. Doing this meant that the classification accuracy of the OCR went from unsatisfactory to almost 100%. The custom model has 1 CNN layer, with filter size 40, 128 neuron in hidden layer, kernel size of 3x3 and the model was trained using the adam optimizer and the sparse categorical cross entropy loss function. The model has 14 neurons in the output layer corresponding to the 14 classes of characters that it can recognize, these classes are the digits (0, 1, 2, 3, 4, 5, 6, 7, 8, 9) and the following special characters that were present in the training scoreboards: '|', 'P', '-' and ':' (The characters '|', '-' and ':' are separators in the scoreboard while the character 'P' is to indicate powerplay in cricket).

4.4 Steps to Minimize False Positive in OCR

1. Before processing the video, 10 frames (each taken 30 seconds apart) are extracted from the video and fed to YOLO. The output locations of these 10 frames are averaged to get a tighter bound on the scoreboard.

2. A threshold on the minimum size of individual contours is used to eliminate granular noise.

3. Regular Expression pattern matching is used to eliminate random scores such as 12-3-0.

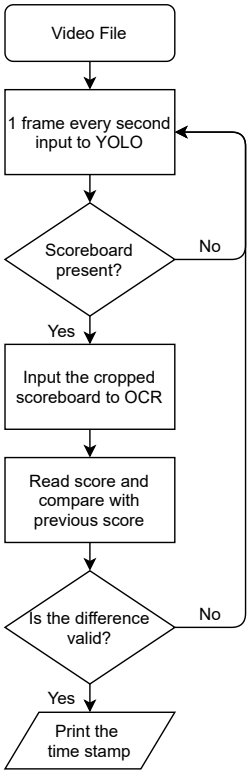

Figure 1: Flow Chart of the Algorithm

4.5 RegEx

A RegEx, or Regular Expression, is a special sequence of characters that helps you match or find other strings or sets of strings, using a specialized syntax held in a pattern [18]. RegEx can be used to check if a string contains the specified pattern of characters. Regular expressions are used extensively for pattern matching everywhere. For example in online forms, a regular expression is generally used to verify that the email id entered by the user is a valid email id like so: r'([a-zA-Z0-9._])+@(gmail|outlook|apple|live|hotmail|yahoo)(.com)'. This will match a string containing [a-zA-Z0-9._] - any combination of characters a to z (upper and lower case), digits, periods and underscores followed by the @ character followed by (gmail|outlook|apple|live|hotmail|yahoo) - the email domains Gmail, Outlook, Apple, Live, Hotmail, yahoo followed by .com. If a match of the pattern is found in the string input by the user, then it is considered a valid email id. RegEx is also used to check the strength and validity of a password, for example, r'[a-zA-Z0-9@#_&]{8,}' can be used to only accept a password that is at least 8 characters long and contains only a-z (upper and lower case) digits and special characters @,#,_, and &. Similarly, a phone number can be verified using r'(\+91)?(\d{10}|\d{8})'. This will match a string that contains only digits, is either 10 or 8 characters long, and has an optional +91 at the front.

4.6 Algorithm to Detect an Event

The score outputted by the OCR will be compared to the score from the previous frame and the difference of the scores is used to determine if there is an event at the current frame or not according to some predefined rules which are different for different sport. For example - in cricket, if the difference in runs is among 6, 4, 3, 5, or if the difference in wickets is 1 then it is considered as an event and the timestamp of that particular frame is outputted along with the type of the event which has occurred. Using these timestamps, highlights may be extracted using a static number of frames before and after the event detection (for example, 7 seconds before and 7 seconds after the event).

5 Results

5.1 Performance of Trained Scoreboard Detection Models

The individual YOLO models (3 models, 1 for each class of game) were tested on images of their respective class of game using an IOU value of 0.5. These images were not part of the training dataset and were of varying resolution (between 360p and 1080p) to thoroughly examine the image detection capabilities of the models. The results are shown in Table 1.

Model	Input Images	Images with a detection	Best mAP
Kabaddi	647	639	1.00
Soccer	639	600	0.93
Cricket	341	341	0.98

Table 1: Performance of Trained Detection Models

5.2 Performance on Video Footage

The proposed method was tested on video footage of kabaddi, soccer, and cricket amounting to a total of 6hours and 35minutes. The performance metric used here are Precision, Recall, and F1 Score with the IOU value set to 0.5.
Some terms related to precision, recall, and accuracy (the following definitions have been derived from [19]):

- True Positives – These are the correctly predicted positive values which means that the actual class (ground truth) is ‘scoreboard’ and the predicted class is also ‘scoreboard’.
- True Negatives - These are the correctly predicted negative values which means that the actual class (ground truth) is ‘not scoreboard’ and predicted class is also ‘not scoreboard’.
- False Positives - These are the wrongly predicted positive values which means that the actual class (ground truth) is ‘no scoreboard’ and predicted class is ‘scoreboard’.
- False Negatives - These are the wrongly predicted negative values which means that the actual class (ground truth) is ‘scoreboard’ but predicted class in ‘not scoreboard’.

Class	Predicted: ’scoreboard’	Predicted: ’not scoreboard’
Ground Truth: ’scoreboard’	True Positives	False Negatives
Ground Truth: ’not scoreboard’	False Positives	True Negatives

Table 2: Confusion Matrix

Precision is defined as the ratio of correctly predicted observations to the total predicted positive observations.

$$Precision = \frac{TP}{TP + FP} \tag{5}$$

Recall is defined as the ratio of correctly predicted positive observations to the total observations in the actual class.

$$Recall = \frac{TP}{TP + FN} \tag{6}$$

F1 Score is defined as the weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

$$F1\ Score = \frac{2 * (Precision * Recall)}{Precision + Recall} \tag{7}$$

Since the model analyses 1 frame from 1 second of the video sequence, the detected events are within 1 second of their actual occurrence. Figures 2 through 11 show the working of the model by depicting images at different stages of the algorithm.

Game	Actual Events	Detected Events	TP	FP	FN	Precision	Recall	F1 Score
Kabaddi	75	78	75	3	0	0.961	1	0.980
Soccer	11	11	11	0	0	1	1	1
Cricket	108	105	102	3	6	0.971	0.944	0.957

Table 3: Performance on Video Footage

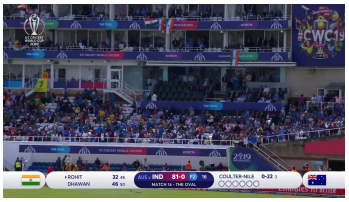

Figure 2: A Frame from the Video

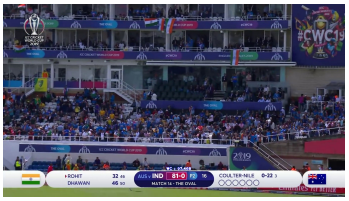

Figure 3: Output of YOLO

Figure 4: Cropped Scoreboard

Figure 5: Input to the OCR

```
0:9 ~ Four
0:13 ~ 3 or 5 runs
0:16 ~ Wicket
0:38 ~ Six
0:45 ~ Four
0:56 ~ Wicket
0:56 ~ Six
1:1 ~ Four
```

Figure 6: Output of the Algorithm

6 Conclusion, Limitations and Future Work

We propose a deep learning model to summarize sports videos and output timestamps of important events. It uses YOLO for the detection of the scoreboard and then some python code for event detection. The model was

Figure 7: A Frame from the Video

Figure 8: Output of YOLO

Figure 9: Cropped Scoreboard

Figure 10: Input to the OCR

```
4:57 ~ Away team scored
6:23 ~ Away team scored
8:13 ~ Away team scored

Goals Scored by the Home Team: 0
Goals Scored by the Away Team: 3
```

Figure 11: Output of the Algorithm

implemented using Python 3.7 and TensorFlow 1.13 on a PC with 1.90GHz, core i5-4460T CPU with 4GB of RAM running Windows 10. Google's open-source ML model training and deployment platform, Colaboratory [20], was used to train the models. Other software used for the model includes OpenCV, Keras API, and ImageAI modules for python. The output had an average (over all classes) F1 Score of 0.979. The model can be trained and deployed for other classes (i.e. can be used to summarize any sports of video). Below we describe the limitations of our current implementation and also layout the scope for future work:

- A need for custom OCR emerged as the existing OCRs could not detect the text or were giving false output. The OCR used in the project works on very specific kinds of texts, we would like to make it more generalized as part of a future project.
- The algorithm currently works for mp4 videos only, we plan to extend the algorithm to work on different video formats.
- Because of limitations of hardware, video clips extraction was not possible and only the timestamps of key events are the output. In the future, we plan to extract the highlight clips.
- We also plan to generalize the scoreboard detection model to detect any kind of scoreboard so as to mitigate the need to train the model on new scoreboard classes. This can be achieved by training the

model on a very large image database.

Here we conclude our paper and express our heartfelt gratitude towards Professor Hansa Shingrakhia, who mentored us throughout our journey.

References

[1] D. Ringis and A. Pooransingh, "Automated highlight generation from cricket broadcasts using orb," in *2015 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*, pp. 58–63, 2015.

[2] P. Shukla, H. Sadana, A. Bansal, D. Verma, C. Elmadjian, B. Raman, and M. Turk, "Automatic cricket highlight generation using event-driven and excitement-based features," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1881–18818, 2018.

[3] Hari R. and Wilscy M., "Event detection in cricket videos using intensity projection profile of umpire gestures," in *2014 Annual IEEE India Conference (INDICON)*, pp. 1–6, 2014.

[4] A. Javed, A. Irtaza, Y. Khaliq, H. Malik, and M. Mahmood, "Replay and key-events detection for sports video summarization using confined elliptical local ternary patterns and extreme learning machine," *Applied Intelligence*, 02 2019.

[5] R. Agyeman, R. Muhammad, and G. S. Choi, "Soccer video summarization using deep learning," in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, pp. 270–273, 2019.

[6] M. Z. Khan, S. Saleem, M. A. Hassan, and M. Usman Ghanni Khan, "Learning deep c3d features for soccer video event detection," in *2018 14th International Conference on Emerging Technologies (ICET)*, pp. 1–6, 2018.

[7] H. Shingrakhia and H. Patel, "Emperor penguin optimized event recognition and summarization for cricket highlight generation," *Multimedia Systems*, vol. 26, no. 6, pp. 745–759, 2020.

[8] H. Shingrakhia and H. Patel, "Sgrnn-am and hrf-dbn: a hybrid machine learning model for cricket video summarization," *The Visual Computer*, 04 2021.

[9] Y. Junqing, L. Aiping, and H. Yangliu, "Soccer video event detection based on deep learning," in *MultiMedia Modeling*, (Cham), pp. 377–389, Springer International Publishing, 2019.

[10] P. Kathirvel, M. Manikandan, and K. Soman, "Automated referee whistle sound detection for extraction of highlights from sports video," *International Journal of Computer Applications*, vol. 12, 12 2011.

[11] A. Tejero-de-Pablos, Y. Nakashima, T. Sato, N. Yokoya, M. Linna, and E. Rahtu, "Summarization of user-generated sports video by using deep action recognition features," *IEEE Transactions on Multimedia*, vol. 20, no. 8, pp. 2000–2011, 2018.

[12] R. B. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," *CoRR*, vol. abs/1311.2524, 2013.

[13] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, pp. 91–99, 2015.

[14] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 1440–1448, 2015.

[15] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv*, 2018.

[16] "How to train a yolo model." https://medium.com/deepquestai/train-object-detection-ai-with-6-lines-of-code-6d087063f6ff.

[17] "Digit recognition using cnn." https://towardsdatascience.com/image-classification-in-10-minutes-with-mnist-dataset-54c35b77a38d.

[18] "Python - regular expressions." https://www.tutorialspoint.com/python/python_reg_expressions.htm.

[19] "Accuracy, precision, recall and f1 score: Interpretation of performance measures." https://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/.

[20] "Google colaboratory." https://colab.research.google.com.