

Cricket Scoreboard Automation using Umpire Gestures

Dhanusha T. John¹, Kavya S. Kumar², Vaishak T. Nair³, P. Visakh⁴, B. R. Poorna⁵

^{1,2,3,4}Student, Department of Computer Science and Engineering, Mar Baselios College of Engineering and Technology, Trivandrum, India

⁵Assistant Professor, Department of Computer Science and Engineering, Mar Baselios College of Engineering and Technology, Trivandrum, India

Abstract: We are living in a modern world where technologies are being invented and the existing technologies are being refined in order to improve the accuracy and also ease the effort by humans. In cricket also different technologies have been already implemented like the Hawk eye technology, the Hotspot technology, the Snicko and so on. But the scoreboard of a cricket match is still being calculated manually. We intend to create a system in which the scoreboard is calculated automatically by recognizing the gestures that are signaled by the umpires. With the advent of cost effective depth sensors and the development of fast human-pose estimation algorithms, interest in action recognition from temporal skeleton sequences has been renewed. In the method we propose, a pretrained model called Caffe will be used to extract skeleton points from the images provided. These skeleton points are used to build skeleton images which are later trained using Deep Neural Networks(DNNs). The Random Forest Classifier is used to classify the test images provided into eight different output classes. These three main stages of the proposed model helps to detect the action of the umpire from any angle and predict it causing the updation of the score board.

Keywords: Machine Learning, Deep Neural Network, Random Forest, Caffe Model.

1. Introduction

Cricket is one of the most popular games not just in India but across the world. It is played between two teams consisting of a total of eleven players each. Cricket umpires are responsible for making decisions and calls in a game of cricket. They also ensure that players and matches follow the rules of the game, taking a similar role to referees in other sports. Most umpires, at least at a professional level, are male, even in women's cricket, and there was only one female umpire at the Women's Cricket World Cup in 2013. The number of umpires on the field varies according to the type of match played. Before the cricket match starts, umpires inspect the pitch area, make sure that all equipment is set out, and check that the boundary is correctly marked before the play begins. They then meet with team captains to check team details and to discuss hours of play, the timing of food and drink breaks, and rules of play, before the coin toss to decide which team bats or bowls first. Most cricket matches have two umpires on the field. The first stands at the stumps behind the bowler, and the second at a position known

as "square leg" to the left of the batsman. There are three umpires in an international match. The third official gives a final decision from a TV monitor if the other umpires feel that a decision is unclear and they need to see a replay. At international "Test" level, players can also appeal against an on-field umpire decision and use the third umpire to challenge the call. Umpires monitor fair play and make decisions on game play. They also monitor overs – sets of six balls bowled at one end of the pitch – and tell teams when to change ends. It is the umpire's job to rule on whether a batsman is out, whether a bowler has bowled illegally and to make a call on how many runs a batsman hits. The position on the pitch dictates the decisions each umpire makes. For example, the bowler-end umpire looks for mistakes such as illegal bowls (no-balls), wide/high balls (wides) and balls that would have hit the wicket, but that were diverted by the batsman's leg (LBW). The other umpire looks for things like the ball hitting the wicket before the batsman gets back to it from a run (run-outs and stumpings). The two switch positions after each over. An effective cricket umpire has an in-depth understanding of the game and its rules. Umpires have good communication and management skills, must stay calm under pressure and have the ability to make quick and accurate decisions. Amateur umpires may only need a short training course; professional umpires have more intensive programs. Accreditation as a fully certified umpire through the United States of America Cricket Association, for example, takes four years.

When an umpire makes a decision, he may shout out a call and make a hand signal if the judgment needs to be communicated. Players on a cricket pitch are spread over a wide area, and umpires need to be loud and to gesticulate clearly so that both teams understand what has happened. This also helps off-field scorers who may not be able to see the finer details of play as it happens. As soon as the gesture is signaled, the scoreboard is updated manually by a person. This updation process can be error prone and at times can take more time. In important matches this increases the tension of the viewers as well as the players and to avoid this unnecessary tension, our system is a solution.

2. Proposed system

The goal is trying to recognize the gestures an umpire performs in a match and update the scoreboard for the corresponding gesture accurately. We have made use of image processing techniques, Deep Neural Network and Machine Learning algorithm to implement the proposed system. In order to improve the accuracy levels, a dataset of images having umpire gestures was created. The high levels of accuracy that were obtained at the end indicates that this can be a better solution. The proposed method will be able to recognize the gestures of the officials in real time. With the flourishing of cricket, various aspects of the game are being automated with the advent of technology. Use of computer vision in assisting third umpire decision is indubitably the well liked one. Unlike other sports, Cricket does not have enough technological application to make it more fair and accurate. The traditional approaches that have been considered so far involve wearing a specialized hand glove, which collides with the beauty of the originality in the field.

- With the use of machine learning and Computer vision, a fully automated system can be obtained for the updation of scoreboard.
- The project would be useful in reducing the duration of a cricket match by saving time for the updation of scoreboard.
- The tedious task of scorekeeper is eliminated. It aims at easing human effort and will also be beneficial to the players.

3. Caffe model

Caffe is a deep learning framework. Deep networks are compositional models that are naturally represented as a collection of inter-connected layers that work on chunks of data as in fig.4.1. Caffe defines a net layer-by-layer in its own model schema. The network defines the entire model bottom-to-top from input data to loss. As data and derivatives flow through the network in the forward and backward passes, Caffe stores, communicates, and manipulates the information as blobs: the blob is the standard array and unified memory interface for the framework. The layer comes next as the foundation of both model and computation. The net follows as the collection and connection of layers. The details of blob describe how information is stored and communicated in and across layers and nets.

A Blob is a wrapper over the actual data being processed and passed along by Caffe, and also under the hood provides synchronization capability between the CPU and the GPU. Mathematically, a blob is an N-dimensional array stored in a contiguous fashion. Caffe stores and communicates data using blobs. Blobs provide a unified memory interface holding data; e.g., batches of images, model parameters, and derivatives for optimization. Blobs conceal the computational and mental overhead of mixed CPU/GPU operation by synchronizing from the CPU host to the GPU device as needed. Memory on the host

and device is allocated on demand (lazily) for efficient memory usage.

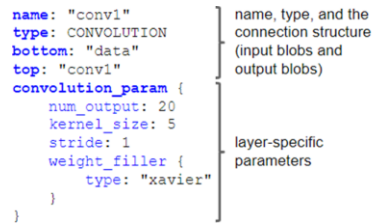


Fig. 1. Caffe layer

4. CV2 DNN

Running deep learning models is computationally expensive. And when it comes to image processing with computer vision, the first thing that comes to mind is high- end GPUs. For this task, it's almost compulsory to add OpenCV to help preprocess data. And the good news is that OpenCV itself includes a deep neural network module, known as OpenCV DNN. It runs much faster than other libraries, and conveniently, it only needs OpenCV in the environment. As a result, OpenCV DNN can run on a CPU's computational power with great speed. OpenCV DNN supports models trained from various networks architectures based on YOLO, MobileNet-SSD, Inception-SSD, Faster-RCNN Inception, Faster-RCNN ResNet, and Mask-RCNN Inception. Neural network is presented as directed acyclic graph (DAG), where vertices are Layer instances, and edges specify relationships between layers inputs and outputs. Each network layer has unique integer id and unique string name inside its network. LayerId can store either layer name or layer id. Let's start with examining the cv2.dnn.blobFromImage function:

1. Blob = cv2.dnn.blobFromImage(image, scale factor=1.0, size, mean, swapRB=True)
2. Image: This is the input image we want to preprocess before passing it through our deep neural network for classification.
3. Scale factor: After we perform mean subtraction we can optionally scale our images by some factor. This value defaults to 1.0 (i.e., no scaling) but we can supply another value as well. It's also important to note that scalefactor should be 1 / sigma as we're actually multiplying the input channels (after mean subtraction) by scale factor.
4. Size: Here we supply the spatial size that the Convolutional Neural Network expects. For most current state-of-the-art neural networks this is either 224,224,227,227, or 299,299.
5. Mean: These are our mean subtraction values. They can be a 3-tuple of the RGB means or they can be a single value in which case the supplied value is subtracted from every channel of the image. If you're performing mean subtraction, ensure you supply the 3-tuple in (R, G, B) order, especially when utilizing the default behavior of swapRB=True.

6. SwapRB: OpenCV assumes images are in BGR channel order; however, the mean value assumes we are using RGB order. To resolve this discrepancy, we can swap the R and B channels in image by setting this value to True. By default, OpenCV performs this channel swapping for us.

The cv2.dnn.blobFromImage function returns a blob which is our input image after mean subtraction, normalizing, and channel swapping.

5. Random forest

Random Forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because it's simplicity and the fact that it can be used for both classification and regression tasks. In this post, you are going to learn, how the random forest algorithm works and several other important things about it. Random Forest is a supervised learning algorithm. The 'forest' it builds, is an ensemble of Decision Trees, most of the time trained with the "bagging" method. The general idea of the bagging method is that a combination of learning models increases the overall result. One big advantage of random forest is, that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Random Forest has nearly the same hyperparameters as a decision tree or a bagging classifier. Fortunately, you don't have to combine a decision tree with a bagging classifier and can just easily use the classifier-class of Random Forest. Random Forest adds additional randomness to the model, while growing the trees. Instead of searching for the most important feature while splitting a node, it searches for the best feature among a random subset of features. This results in a wide diversity that generally results in a better model. Therefore, in Random Forest, only a random subset of the features is taken into consideration by the algorithm for splitting a node.

6. Dataset used

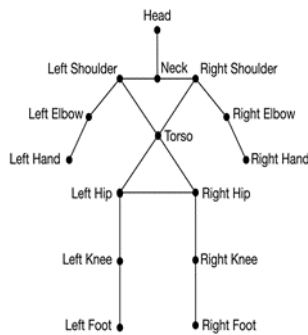


Fig. 2. Skeletal points

Skeleton pose datasets are used for training the network. In the skeleton images, the body is represented by a set of points. Each point has a (x,y) coordinates. The distance between the coordinates and the angle between the line joining the

coordinates are taken into account for the purpose. The attributes are used for predicting the corresponding score.

7. Flowchart

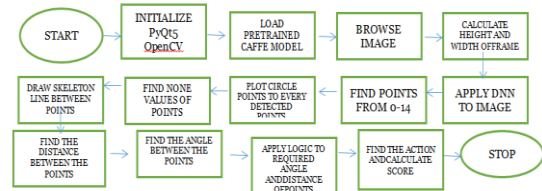


Fig. 3. Flowchart

8. Design

The basic design of the model consists of three stages:

- Finding the skeleton points using Caffe Model.
- Training the dataset using DNN.
- Testing using Random Forest Classifier.

9. Implementation

One of the very first problems that we had to encounter was the availability of a dataset. With further studies, we concluded that the available dataset images will not give the high levels of accuracy that was needed. Thus a new dataset consisting of images of students who were signaling umpire gestures were taken. More than 130 images of the same gesture were taken and trained. After our implementation, we achieved an accuracy level of over 85 percent.

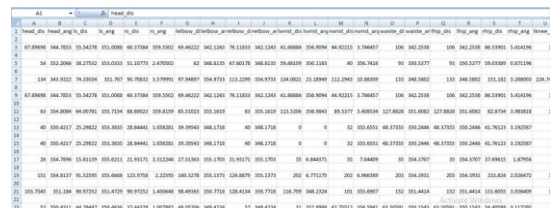


Fig. 4. Attribute values in the dataset

10. Algorithm

We are developing an optimized cricket score prediction system that detects the gestures of the umpires and updates the scoreboard efficiently. The algorithm part can be divided into two phases-the Training part and the Testing part.

A. Algorithm - training

1. **START**
2. Open the graphical user interface. Repeat steps 3-7 for every image in the training phase
3. Browse for the image of the gesture to be detected.
4. After selecting the image process the image. The 14 skeleton points are obtained by making use of the caffe model.
5. The distance and angle between the points that will add to 36 attributes are calculated.

6. The correct action of the corresponding gesture is specified.
7. The values of these 36 attributes along with the value indicating the correct gesture, thus adding up to 37 attribute values are fed as input to the deep neural network.
8. STOP

B. Algorithm - testing

1. START
2. Open the graphical user interface
3. Browse for the image that is to be given as the input for testing
4. Process the image after selecting it. The 14 skeleton points are obtained by making use of the caffe model
5. Click on the predict button. The result of the gesture will be displayed and score will be updated by comparing the image with the already trained images.
6. STOP

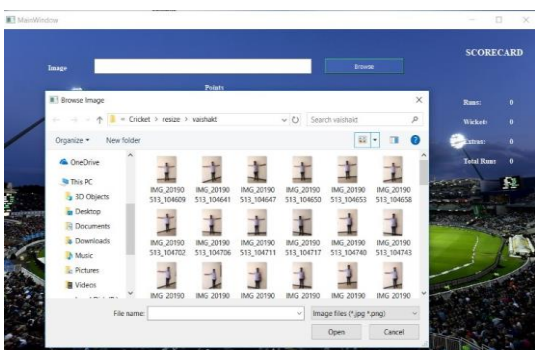


Fig. 5. Selecting the images

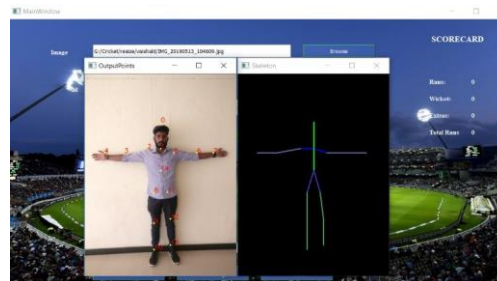


Fig. 7. Skeletal points and Skeleton image of Wide



Fig. 8. Attribute values and Predicted Result



Fig. 9. Processing next image

11. Results

We were able to obtain high levels of accuracy by training our neural network with our dataset. We obtained an accurate score of 45 after training 590 images of various umpire gestures. Thus the overall success percentage can be calculated as $45/590 * 100 = 77$ percentage. We also observed that the accuracy score and percentage increases as the number of input images increases. The accuracy score when we had trained 360 images were 26 and the percentage was 72. The sequence of actions and our corresponding output are given in the following images:



Fig. 6. Processing

12. Conclusion and future work

The main objective of our system is to reduce the time and human effort and improve the speed in the cricket scoreboard calculation. For this purpose, we created a dataset and trained a neural network with the above mentioned dataset and it also resulted in high levels of accuracy. For getting the high levels of accuracy we used a pretrained caffe model. The skeleton points of an input image were obtained by making use of this caffe model. The system or the technology used behind this system can be used in different areas of gesture recognition.

The applications are,

- Sports
- The gestures of dumb people can be recognize,

There are certain advantages of using this system The accuracy levels of our system have been tested and it has been calculated to be more than 85 percent. If further improvements are made to this system, the levels of accuracy can be improved further. The system will reduce the time taken for calculation and the human effort and also improve the speed with which the scores are displayed. However, there are certain drawbacks as well. At present the system focuses only on the gestures of the umpire and the supreme authority is given to the umpire.

Because of this, we had to suggest new gestures for certain events such as one, two and three runs.

Improvements that can be made in the future are:

- At present our system is only able to predict accurately the gestures which are fed at input as images. With the evolution and improvement of this idea, the system can be improved to be able to train and predict video as input. This will further improve the speed and also improve the effectiveness of the system.
- When the system is improved, measures can be taken to incorporate the feature of detection of runs that are obtained as a result of running between the wickets.

References

- [1] Md. Asif Shahjalal, Zubaer Ahmad, "An Approach to Automate the Scorecard in Cricket with Computer Vision and Machine Learning", 2017 3rd International Conference on electrical information and communication technology.
- [2] M. Komar, P. Yakobchuk, V. Golovko, V. Dorosh and A. Sachenko, "Deep Neural Network for Image Recognition Based on the Caffe Framework," 2018 *IEEE Second International Conference on Data Stream Mining & Processing (DSMP)*, Lviv, 2018, pp. 102-106.
- [3] Jürgen Schmidhuber, "Deep learning in neural networks: An overview," *Neural Networks*, vol. 61, pp. 85-117, January 2015.
- [4] S. Mitra and T. Acharya, "Gesture recognition: A survey," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 3, pp. 311-324, 2007.
- [5] "Top 10 list of the internet world's most popular sports," <http://www.topendsports.com/world/lists/popular-sport/fans.htm>.
- [6] <https://www.python.org/>