**Automated Ball by Ball clipping and Event Classification in Cricket: A Deep Learning Approach.**

**Manmeet Patel Shubham Prajapati**

**[Birla Vishvakarma Mahavidyalaya Birla Vishvakarma Mahavidyalaya](https://bvmengineering.ac.in/)**

**[Engineering College Engineering College](https://bvmengineering.ac.in/)**

[21it456@bvmengineering.ac.in](https://bvmengineering.ac.in/) [21it445@bvmengineering.ac.in](https://bvmengineering.ac.in/)

**Dev Lad**

**Birla Vishvakarma Mahavidyalaya**

**Engineering College**

21it436@bvmengineering.ac.in

**Abstract**

*This paper proposes an automated solution utilizing deep learning techniques to clip and classify ball by ball events. The project aims to enhance the viewing experience for cricket enthusiasts by providing real-time event classification. Through a comprehensive analysis of existing applications, feasibility studies, implementation strategies, and testing procedures, this paper outlines the development of the proposed system. Though paper focuses on the game of cricket, the approach can easily be extended to other games or other real world problems with careful tweaking.*

**Keywords:** Cricket, Deep Learning, Event Classification, Event based clipping, LRCN, sampling.

1. **Introduction**

Cricket, a beloved sport with a global following, has seen a surge in streaming rights and viewership. However, the manual classification of events and generation of highlights remain labor-intensive processes. Also certain amount of latency is stuck here, so generating and posting interesting clips and highlights takes time. If the live commentary says “A stunning pull by Rohit”, a fan has to wait to watch that clip, until it is on editing table.

In this project we address this problem by clipping ball by ball deliveries and classifying each ball in particular event. Here, the event is classified in four classes, namely **runs**, **no run**, **boundary** and **wicket.** Effectively two deep learning models works in tandem. First model takes cricket broadcast video as input and outputs each ball delivery clip. This clips serves as input to second model which classify clips into particular event. This labeled clips also enables to generate preferenced highlights.

1. **Related work**

While there is limited work specifically on cricket event classification, our approach draws inspiration from research in related fields:

Karpathy et al. [1] explored various CNN architectures for large-scale video classification, demonstrating the effectiveness of deep learning in understanding video content. Their work on fusion architectures, particularly late fusion, informed our approach to processing cricket video frames.

Donahue et al. [2] introduced Long-term Recurrent Convolutional Networks (LRCNs) for visual recognition and description. This architecture's ability to model temporal dynamics in videos influenced our consideration of temporal information in cricket event classification.

In the sports domain, Dixit and Balakrishnan [3] applied CNNs for ball-by-ball outcome classification in cricket. Their work provided valuable insights into the challenges and potential of deep learning in cricket analytics.

For ball starting detection, Kumail Abbas and Muhammad Saeed [4] applied variety of techniques. Their work provided great insight on the challenge.

1. **Dataset:**
   1. **Creation and labeling event classifier model dataset**

Our dataset comprises of individual ball clips. These clips are labeled in one of four labels, other possible complex scenario were ignored. Full length broadcast videos of two T20I matches were downloaded from Youtube. To construct dataset this matches are clipped ball by ball manually. So around 480 video clips were generated.

Labeling was though to be done with web scraping from ball by ball text commentary, but these sites like, cricbuzz[6] are dynamically spread content with JavaScript. So, was done manually too. The quantity of the video clips could easily be aped-up for better results.

**3.2 Inbalance in dataset**

As cricket is skewed game, so there are more number of runs and no runs than boundary and wicket. After facing bias toward maximum labeled clip in dataset. Results were unsurprisingly skewed towards runs and no runs. The ratio of the number of clips per label is managed and tried with different ratios. So reduction in the number of runs and no runs clips was accepted. Also data augmentation was used for reducing bias in the dataset among categories.

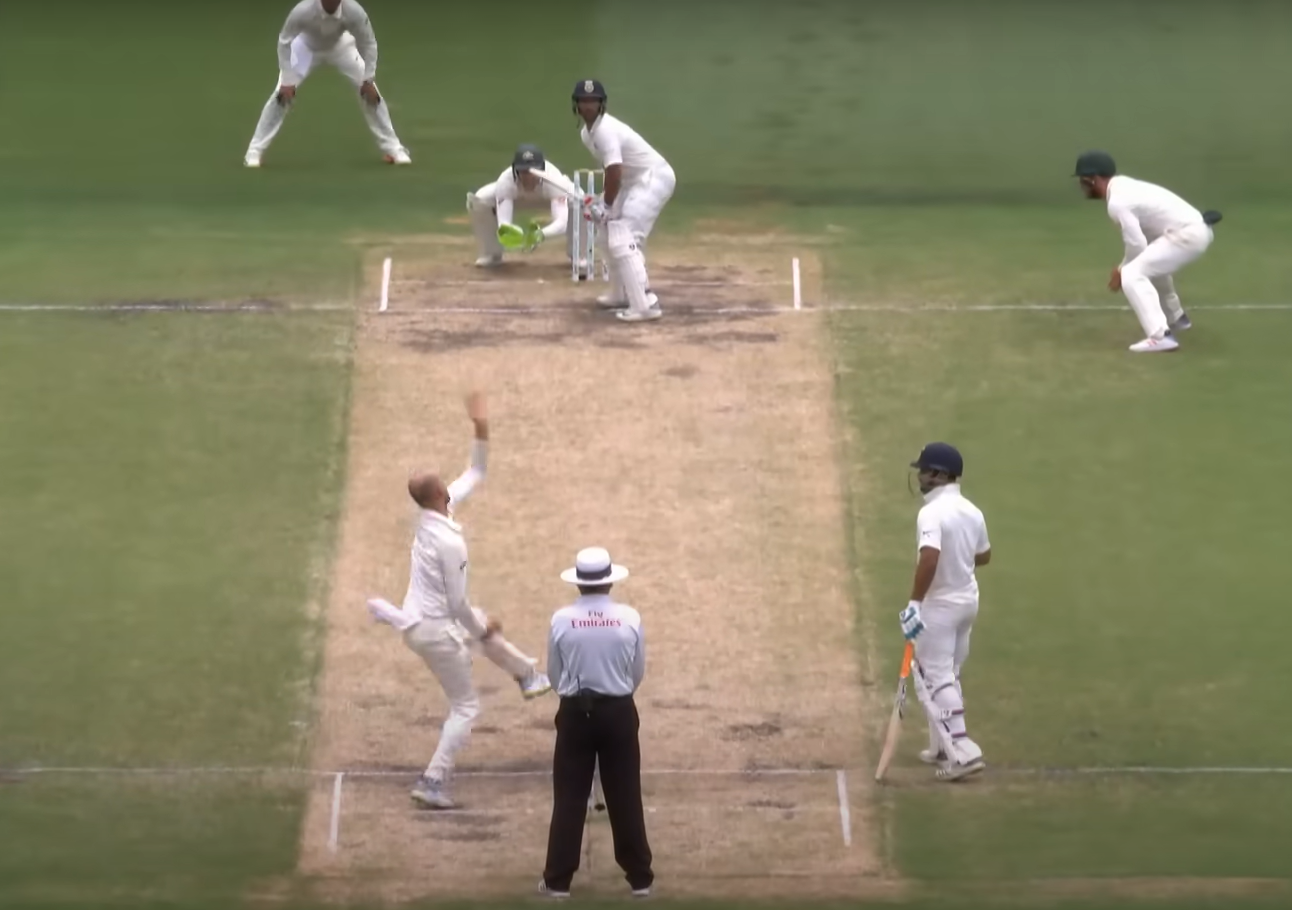
Also scorecard column on the bottom was removed, as model could learn from that quite easily, which is not idle.

**Figure1:** First image shows broadcast video as it is while the scoreboard is cropped in second one, for correct learning of model.

* 1. **Creation and labeling image classifier model dataset**

This is a binary Classification CNN model that classified frames into front view and not front view. The purpose of this model is to automatically clip per ball delivery from the long broadcast video. Dataset was created manually by taking snapshot of respective views from broadcast video and labeling them. We made sure to prevent any kind of bias towards one class in dataset. There were around ~500 images in each class.



**Figure 2(a):** Samples of frames labeled as front-view.



**Figure 2(b):** Samples of frames labeled as not front-view.

1. **Methods**
   1. **Creating clips from broadcast video:**

Model architecture looks like:

**First Convolutional Block**:

A convolutional layer with 16 filters of size 3x3, ReLU activation, and same padding.

A max-pooling layer with a pool size of 2x2.

**Second Convolutional Block**:

A convolutional layer with 32 filters of size 3x3, ReLU activation, and same padding.

A max-pooling layer with a pool size of 2x2.

**Third Convolutional Block**:

A convolutional layer with 64 filters of size 3x3, ReLU activation, and same padding.

A max-pooling layer with a pool size of 2x2

**Fully Connected Layers**:

A flatten layer to convert the 3D outputs of the convolutional layers into 1D.

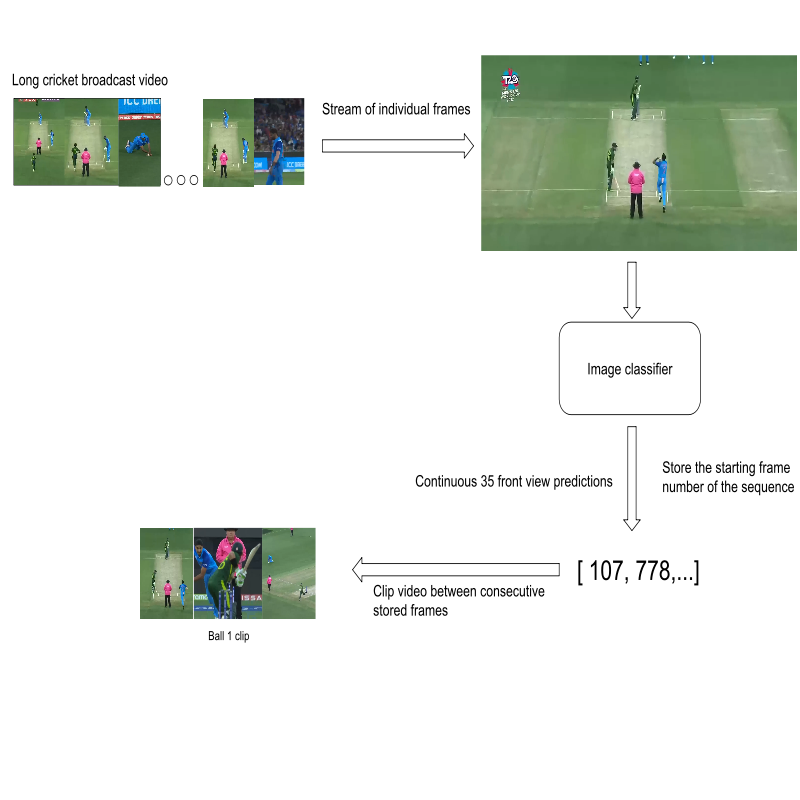
A dense (fully connected) layer with 128 units and ReLU activation.

An output dense layer with 1 unit and sigmoid activation.

When this model is applied to a stream of video frames, it analyzes each frame individually to identify specific patterns or features. The model continuously monitors the input frames, checking for a specific frame that signals the start of a ball delivery. It achieves this by continuously classifying the frames into front-view or not. The continuous stream of n=35 front view is considered to be the start of the ball. Here n is hyper parameter, that could be tweaked for variation of result. The inference loop saves the starting frame number. As the video stream progresses, the model looks for the recurrence of the same specific sequence, which it interprets as marking the end of the first ball and the beginning of the second ball. From this list of frame numbers of ball starting, inference loop clips each ball delivery as it occurs. This approach enables model to effectively segment and analyze distinct events within the continuous video stream.

To further reduce latency for whole process frame skipping was used. Considering every other frame for inspection worked quite well.

There exist a replay issue. As we are only interested in real-time action, the replays in broadcast video could interrupt the clipping task, as model will consider the bunch of front-view in replay as the ball starting event, which is undesired. To over come this we can use two facts about replays: i) It do not contain scoreboard. ii) It has slower frame rate than real-time match. So scoreboard detection and frame-rate detection comes handy for the application with the trade-off of infusing latency into the system.



**Figure-3:** Flowchart of the model for clipping individual ball clips from a broadcast video.

* 1. **Classification of clips:**

The Long-term Recurrent Convolutional Network (LRCN) combines convolutional neural networks (CNNs) for spatial feature extraction with recurrent neural networks (RNNs) for temporal sequence processing, enabling the classification of video sequences into different categories. Given your specified categories: Wicket, Run, No run, and Boundary, here’s how the model can be expanded:

* **TimeDistributed Conv2D Layers:**

The model begins with a TimeDistributed wrapper applied to a series of 2D convolutional layers. This wrapper enables the convolutional operations to be applied independently to each frame in the input sequence.

The first convolutional layer uses 16 filters of size 3x3 with 'same' padding and ReLU activation. This is followed by a TimeDistributed MaxPooling2D layer with a pooling size of 4x4 and a TimeDistributed Dropout layer with a dropout rate of 0.25.

Subsequent convolutional layers have 32 and 64 filters respectively, with similar configurations. The pooling operations reduce the spatial dimensions of the feature maps, and dropout is used to prevent overfitting.

* **Flattening and LSTM Layer:**

After the convolutional and pooling operations, the output is flattened using the TimeDistributed Flatten layer to prepare it for the recurrent layers.

An LSTM layer with 32 units is employed to capture temporal dependencies and sequences within the flattened feature vectors.

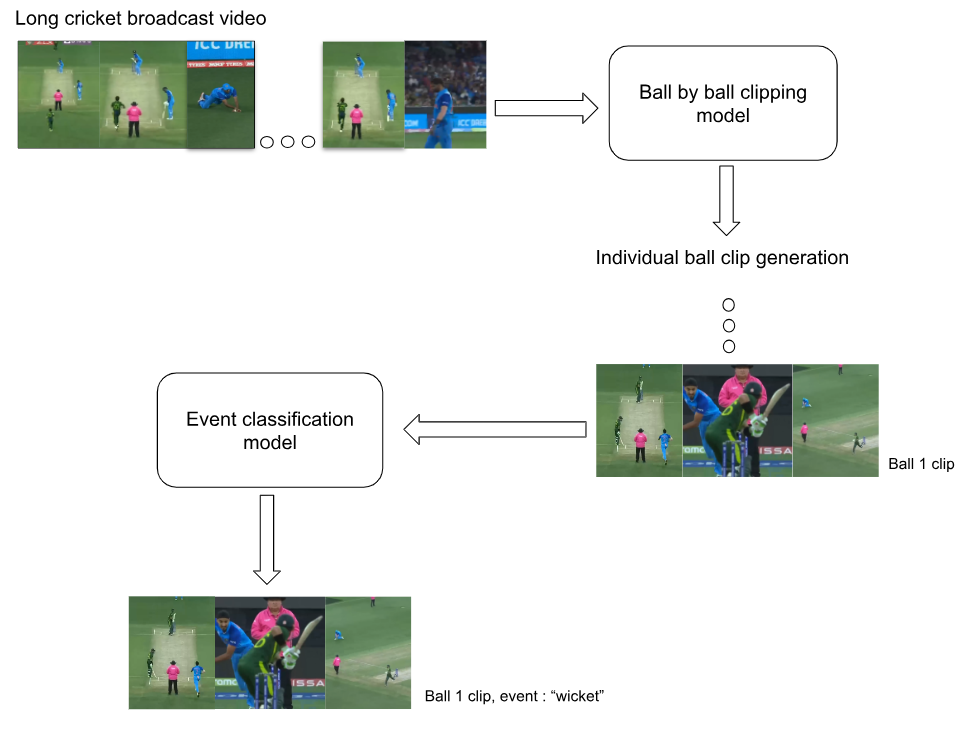
* **Dense Output Layer:**

The final layer is a Dense layer with a softmax activation function, which outputs probabilities for each class. The number of units in this layer corresponds to the number of classes in the classification task.

So in nutshell, the approach looks like, TimeDistributed CNN for Spatial Feature Extraction: The CNN layers are applied to each frame to extract spatial features.

Recurrent Layers for Temporal Processing: LSTM or GRU layers are used to process the sequence of features extracted from the frames to capture the temporal dynamics.

Dense Layers for Classification: Fully connected layers are used to map the temporal features to the final classification categories. This approach is inspired by “Long-term recurrent convolutional networks for visual recognition and description”. [2]



**Figure-4:** Flowchart of the model for classifying individual ball clips into particular class.

* 1. **Training and Optimization**

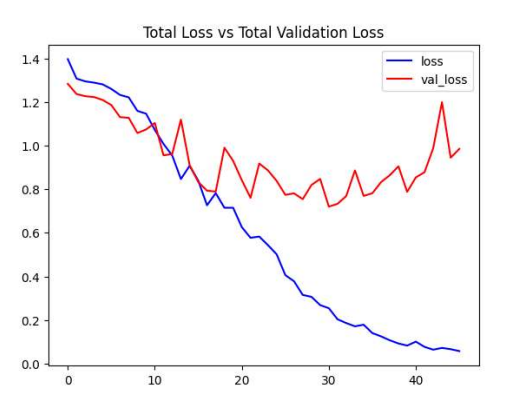
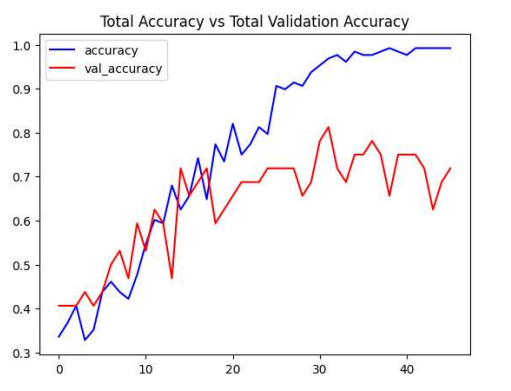
We trained our model using the Adam optimizer with low learning rate. The model was trained on a dataset of approximately 600 ball delivery clips, with 20% reserved for validation. We employed data augmentation techniques, including jittered sampling, to enhance the robustness of our video classification model.

* 1. **Real-Time Classification**

For live event classification, we implemented a sliding window approach that processes incoming video streams in real-time. This allows for near-instantaneous event labeling as the match progresses. There seems to exist a trade-off between accuracy and latency in video classification model which could be resolved with high-end computation.

1. **Results and Discussion**

Our best-performing classification model achieved an accuracy of approximately 56% on the validation set for ball-by-ball event classification.



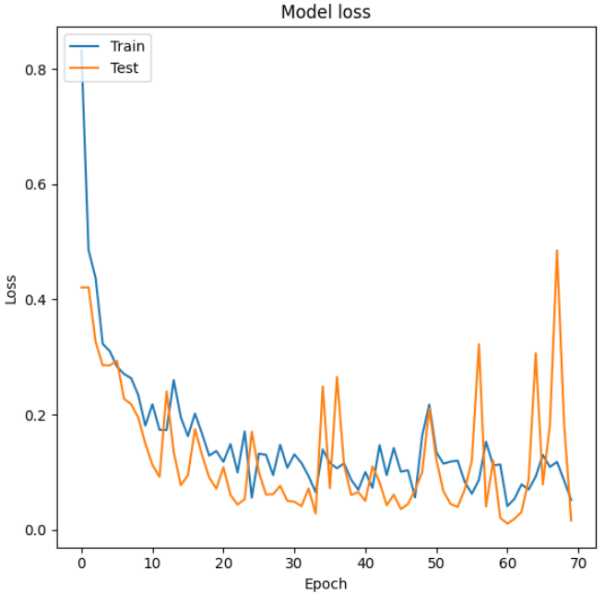
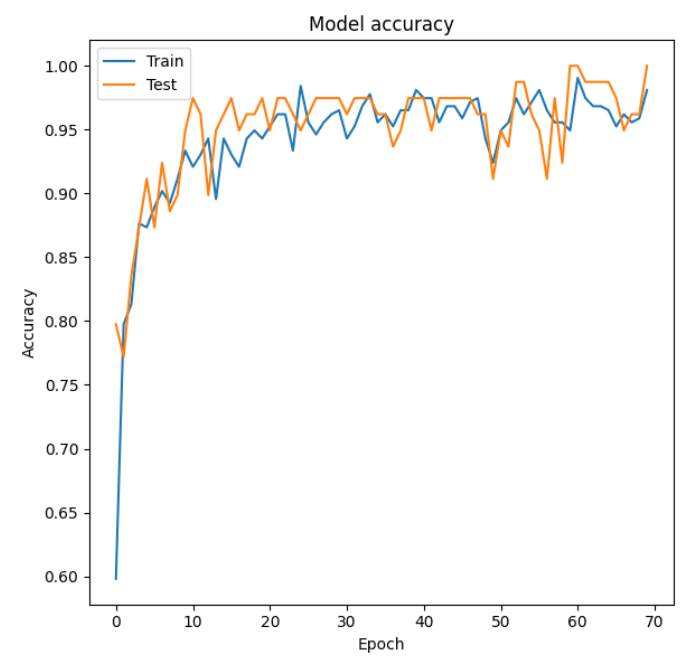
**Figure-5(a) Figure-5(b)**

The image presents two line graphs comparing training accuracy and loss against their corresponding validation metrics over multiple epochs during the classification model training process.

The reason to this relatively low accuracy is very limited training data. As the dataset is created manually it is tough to exponentiate the quantity. To automate the creation of the dataset is itself a topic of research.

The model performed well in distinguishing between most event types. The "wicket" class showed lower accuracy, likely due to the relative scarcity of wicket events in the training data and the diversity of wicket types.

The clipping model had validation accuracy of 92%, it worked as desired with little bit of latency infused.



**Figure-6(a)** **Figure-6(b)**

The image presents two line graphs comparing training accuracy and loss against their corresponding validation metrics over multiple epochs during the clipping model training process.

1. **Conclusion**

Our deep learning approach to cricket event classification and highlight generation demonstrates the potential to transform the cricket viewing experience. By automating event classification, we address key limitations of traditional manual methods.

Improvement can focus on:

1. Expanding the dataset to improve classification accuracy, particularly for rare events like wickets. Also approach to automate the dataset creation process.
2. Exploring more advanced architectures, such as 3D CNNs or transformer models, to better capture temporal dynamics
3. Integrating additional features such as player tracking and shot type classification

By continuing to refine and expand this system, this approach can be extended to other similar real world task.

**References**

[1] Karpathy, A., Toderici, G., Shetty, S., Leung, T., Sukthankar, R., & Fei-Fei, L. (2014). Large-scale video classification with convolutional neural networks. In CVPR.

[2] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., & Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In CVPR.

[3] Dixit, K., & Balakrishnan, A. (2016). Deep Learning using CNNs for Ball-by-Ball Outcome Classification in Sports.

[4] Kumail Abbas, Muhammad Saeed. Deep-Learning-Based Computer Vision Approach For TheSegmentation Of Ball Deliveries And Tracking In Cricket.

[5] Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[6] Cricbuzz website for labeling the dataset, https://www.cricbuzz.com