National College of Ireland

Project Submission Sheet – 2020/2021

Student Name:	Omkar Ratnoji Tawade		
Student ID:	19232136		
Programme:	MSc. Data Analytics (MSCDAD_A)	**Year:**	2020-21
Module:	Data Mining and Machine Learning 2		
Lecturer:	Michael Bradford		
Submission Due Date:	13-05-2021		
Project Title:	Umpire Sign Detection using Deep Learning		
Word Count:	9300 words		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature: Omkar Ratnoji Tawade

Date: 13-05-2021

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

National College of Ireland

Project Submission Sheet – 2020/2021

Student Name:	Vivek Kumar		
Student ID:	19201885		
Programme:	MSc. Data Analytics (MSCDAD_A)	**Year:**	2020-21
Module:	Data Mining and Machine Learning 2		
Lecturer:	Michael Bradford		
Submission Due Date:	13-05-2021		
Project Title:	Umpire Sign Detection using Deep Learning		
Word Count:	9300 words		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature:	Vivek Kumar
Date:	13-05-2021

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

National College of Ireland

National College of Ireland

Project Submission Sheet – 2020/2021

Student Name:	Michael Dunne		
Student ID:	15420892		
Programme:	MSc. Data Analytics (MSCDAD_A)	**Year:**	2020-21
Module:	Data Mining and Machine Learning 2		
Lecturer:	Michael Bradford		
Submission Due Date:	13-05-2021		
Project Title:	Umpire Sign Detection using Deep Learning		
Word Count:	9300 words		

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the references section. Students are encouraged to use the Harvard Referencing Standard supplied by the Library. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action. Students may be required to undergo a viva (oral examination) if there is suspicion about the validity of their submitted work.

Signature: Michael Dunne

Date: 13-05-2021

PLEASE READ THE FOLLOWING INSTRUCTIONS:

1. Please attach a completed copy of this sheet to each project (including multiple copies).
2. Projects should be submitted to your Programme Coordinator.
3. **You must ensure that you retain a HARD COPY of ALL projects**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. Please do not bind projects or place in covers unless specifically requested.
4. You must ensure that all projects are submitted to your Programme Coordinator on or before the required submission date. **Late submissions will incur penalties.**
5. All projects must be submitted and passed in order to successfully complete the year. **Any project/assignment not submitted will be marked as a fail.**

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Umpire Pose Detection using Deep Learning

Group Project (Team O) – CA - 50%

Michael Dunne
Data Analytics
National College of Ireland
Dublin, Ireland
x15420892@student.ncirl.ie

Omkar Tawade
Data Analytics
National College of Ireland
Dublin, Ireland
x19232136@student.ncirl.ie

Vivek Kumar
Data Analytics
National College of Ireland
Dublin, Ireland
x19201885@student.ncirl.ie

***Abstract*—The goal of this project is to identify and detect Umpire hand movement in the sport of cricket using Deep Learning. Cricket is one of the most played sports in the world and in certain countries considered the national sport. The Umpire who oversees the game must make calls on the plays that have been made by the two teams, decision making must be instant or justified if a moment is needed, although because the plays are so fast, a decision must be made based on human decision which can sometimes be a mistake which is unfortunate due to the high stakes involved in big matches. In this document we look to identify the Umpires decision based off the hand signal detection that the umpire uses to represent the call he/she has made. Using this detection, we can identify what call is being made for the play just made based off imaging video data we have collected and analyzed, this will be test and molded into our network once refined into a functioning model. For the signals used by the Umpire that ultimately decide the game we need to identify gestures or signals such as No Ball and Out. We will be using convolution neural network to classify umpire and non-umpire along with this we will train a model using LSTM and Google Video Intelligence API on video data to detect the signals shown by the umpire.**

Keywords—Umpire, Detection, Cricket, Pose, Convolutional Neural Network, LSTM, Google Video Intelligence

I. INTRODUCTION

The game of cricket was been a staple of sport in many countries for many of years, first compiling the rules of the game in 1744 and overtime gradually new rules have been added and some rules changed or altered in such a way to overall improve he sport for player safety and also for entertainment purposes. It is important we understand the rules of the game and how it is played. The game of cricket is comprised of two teams each of eleven players at a time. One team will take the turn of batting and the other will then proceed to bowl and field the pitch. The overall goal of the team who is batting is to hit the ball when thrown by the opposition and begin to run between the wickets, once this is done it is called a run. If the batsman hits the ball so good it goes out of the area(boundary), this will equal 6 runs. Hitting the boundary along the ground equals 4 runs. The bowling team need to get the batsmen out by doing the following, hitting the wickets with the ball, or hitting the batsman's leg in front of the wicket or catching the ball when hit by the batsman. 10 batsmen must be eliminated before a change of turn in objectives. The team must score as many runs as possible before 10 wickets are made, then the winning team is the team with most runs.

Now that the basic rules have been outlined, we can go over the job the umpire has in this game. The Umpire has several responsibilities including calling a NO BALL which is the heel of the bowler's front foot lands on or in front of the line (batting crease). The Umpire also can signal a WIDE which occurs when the ball is unable to be hit due to being thrown to wide of the batsmen, there is a lot to take in here as it's up to the umpire if the batsmen could in fact reach and get the ball from their position. It also must be noted that Umpires can in fact be challenged on their calls made by the players if they feel a mistake is made.

A six as we discussed earlier, is when the ball does not bounce before reaching the outer are of the boundary of the cricket field. This is the equivalent of six runs. The Umpire will signal this. Finally, an Umpire will call an out or dismissal which is when the batsmen has finally been eliminated by the other team. Below we can see a list of the signals that our used by the Umpires to identify the call they have made. We can see all the ones we have described above.

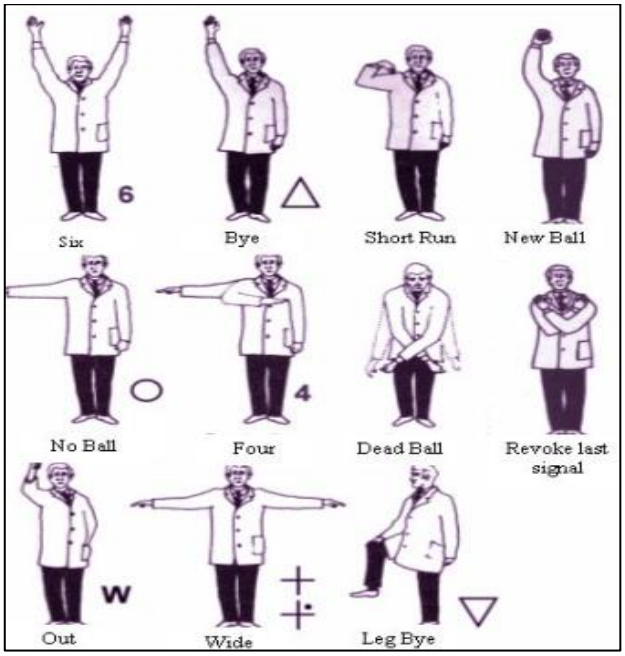

Figure 1. Scoring Signals by Umpire

Over the course of this paper, we will discuss related work and take from this what can be improved on and what realistically what we can achieve. We will also look at what methodology approach we made to achieve our goals as best as possible. We will then evaluate our results and draw up some conclusions with relation to future work that can be carried out from this research. This project should be a great benefit to match highlight analysis as well as match predictions in the game of cricket.

II. Related work

In their research, Aravind Ravi et al. [1] presented a study in which they first created a new dataset which they named SNOW which they used for umpire sign detection in cricket. They used this dataset to create a system that was able to generate the highlights of a full cricket match. In a cricket match, umpire is responsible for making the decisions about what is going on the field and for signalling those important decisions that umpire make the umpire used his hands. Researchers of this research have used four events for classification which are six, no ball, Out and wide and for classifying those events the researcher tried to detect the pose of umpire from cricket videos. For feature extraction researchers have used V3 and VGG19 networks. Researchers collected several images of umpires in which umpires are making the decision actions for events. They have downloaded the images from internet. The researchers created five classes in their research where each class belongs to a separate type of umpire signalling and the last class belongs to those images in which umpires was not performing any actions. In total there were 390 images. The researchers proposed their research in two phases. In the first phase the researchers design those classifiers which were able to differentiate images of umpires and non-umpires. After that using the pre trained features the researchers trained a linear SVM classifier which was able to detect the pose of the umpire. In the second phase, the researchers tried to detect the signalling of the umpire and for that the researcher have used the saved classifier models and generating the summary of the videos. In this research, the feature extraction methods were able to achieve good level of accuracy. Classifier 1 with Inception V3 was able to achieve the 10-fold accuracy of 96.97% and jack-knife accuracy of 97.76% and test accuracy of 94.23%. Classifier 1 with VGG19 – FC1 Layer was able to achieve the 10-fold accuracy of 97.75% and jack-knife accuracy of 97.59% and test accuracy of 96.15%. Classifier 1 with VGG19 – FC2 layer was able to achieve the 10-fold accuracy of 96.47% and jack-knife accuracy of 97.79% and test accuracy of 94.87%. Classifier 2 with Inception V3 was able to achieve the 10-fold accuracy of 77.71% and jack-knife accuracy of 77.56% and test accuracy of 85.99%0. Classifier 2 with VGG19 – FC1 Layer was able to achieve the 10-fold accuracy of 82.43% and jack-knife accuracy of 81.09% and test accuracy of 83.33%. Classifier 2 with VGG19 – FC2 layer was able to achieve the 10-fold accuracy of 78.14% and jack-knife accuracy of 81.09% and test accuracy of 78.21%.

Hari, Wilscy [2] presented a research in which they have also tried to detect the gestures of umpire. For this research, the researchers have used the twenty – twenty cricket match videos. For scene segmentation the researcher had detect the bowling events in cricket match after that the researchers have identified the frame with umpires for analyzing their gestures and for analysis the researchers have used vertical and horizontal intensity projection profiles. The researchers have used the gestures of umpires as a feature for training their Random Forest classifier for the extraction of all the events. The first thing that researchers have done in this project is that they have cut down the entire video in shots like bowling shots after that they checked that cut shot video for pitch frame. In the video if they found out that the area of pitch is increasing in initial shots and it is decreasing after that, then they call it a pitch shot and all the shots in between two pitch shots create a scene. After the scene creation they separate the shot and called the phase as shot separation. After the shot separation phase, the next phase was pitch identification. After that they detect the umpire frames and after the detection of umpires, they extract the gestures of umpires and train their classifier with extracted features. The researchers random forest model was able to get the precision level of 85% in 1st game, In the second game the researcher's model was able to achieve the precision level of 81% and in this last game, the researcher's model was also able to achieve the accuracy level of 81%. In the first game, their model was able to detect 11 fours out of 13, 2 out of 2 sixes, 5 out of 10 outs, 3 out of 4 wide's and surprisingly it caught 1 no ball when there was not a single no ball in the whole game. In the second game, their model was able to detect 7 fours out of 10, 6 out of 8 sixes, 2 out of 8 outs, 0 out of 0 wide's and surprisingly it caught 2 no ball when there was not a single no ball in the whole game. In the third game, their model was able to detect 16 fours out of 20, 6 out of 7 sixes, 1 out of 5 outs, 0 out of 0 wide's and 1 no ball out of 1 no ball in the whole game.

Kowsher et al. [3] presented a research in which they tried a classification method with the help of Convolutional Neural Network (CNN) with inception V3 so that they can automate the decisions of third umpire, and they have also tried to automate the scoring system for which they have tried to detect the signals of umpires. They have also proposed a way for increasing the performance of CNN for which they have used deep CNN. For this research, the researchers have used the 6 datasets which contains images of umpires giving signals for No Ball, Bounce Ball, LBW, Run Out, Stumping. There were total of 633 images for foot cross no ball, 617 for waist height no ball, 646 for bounce ball, 719 for LBW, 367 for run out and 672 for stumping. For detection of umpire signals they have gathered 2479 images in which 229 images were for Out, 221 were for Four, 214 were for six, 203 were for bye, 208 for leg bye, 230 for one bounce, 198 for wide, 205 for no ball, 187 for dead ball, 224 for cancel call, 175 for new ball and 185 for penalty run. Their CNN model was able to achieve the 74% level of accuracy for Foot cross No ball, 77% for waist height No ball, 68% for Bounce ball, 56% for LBW, 53% for Run out and 54% for Stumping and their Inception V3 model was able to achieve the 84% accuracy for Foot cross No ball, 85% for waist height No ball, 81% for Bounce ball, 63% for LBW, 69% for Run out and 70% for Stumping. In their research they were able to achieve the accuracy of 83% in CNN for classification of umpire signals.

In another research, Rock et al. [4] tried to analyze the use of wavelets for creating an edge-detection system for cricket matches. For automate this edge detection process, the researchers have used Neural Networks. First the researchers record the audio samples of ball-on-bat and ball-on-pad events and then they perform DSP analysis after that they perform feature extraction and then they implemented neural network classification on these audio samples. In their research, they were able to classify the 97.5% data correctly using their neural network model.

Sanjeev and Suvarna [5] presented a research in which they also tried to recognize the events of umpire giving signals by hands and for recognizing the signals the researchers have used Random Forest classifier in this research. In this research, the researchers tried to recognize the events of Four, Six, No ball, out and wide in cricket. The researchers first did the shot separation, after the separation of shots the

researchers detect the frames of umpires and then from these umpire frames, the researchers extract the gesture of umpires then they implement random forest classifier on these extracted gestures. They follow this process for 3 games. In game 1 there were 13 events of four in which their model was able to correctly detect 11 events, for total of 2 sixes, their model detected both six, out of 10 outs the model detected only 5, out of 4 wide their model detected 3 and surprisingly there were no no-ball, but their model detected 1. In game 2 there were 10 events of four in which their model was able to correctly detect 7 events, for total of 8 sixes, their model detected 6 sixes, out of 8 outs the model detected only 2, out of 0 wide their model detected 0 and surprisingly there were no no-ball, but their model detected 2. In game 3 there were 20 events of four in which their model was able to correctly detect 16 events, for total of 7 sixes, their model detected 6 sixes, out of 5 outs the model detected only 1, out of 0 wide their model detected 0 and 1 no-ball and their model detected 1 no-ball correctly.

Vaishnavi et al. [6] account the study in which the researchers tried to automate the scoreboard in a match of cricket by recognizing the gestures of umpires. In this research, the researchers have combined two SVM classifiers. The first classifier differentiates the umpire with non-umpire images and then the second classifier is used for recognizing the gestures of umpires. The four gestures used for identification in this research are Six, Wide, No-ball and Out and after recognizing the gestures the researchers updated the scoreboard. There were total of 5 classes present in this research, 4 classes contain umpire gestures and last class is 'No Action' which contains umpires with no signaling. There were two phases in this research, first phase contains the designing of the classifier and second phase contain updating of the scoreboard. Researchers of this research were able to correctly classify 378 Umpires and 391 Non-umpires out of 400. Their model was able to correctly classify 97 outs out of 100, 93 wides out of 100, 88 no ball out of 100, 95 six out of 100 and 94 out of 100. In their research they could add roll back it will be helpful if any video got misclassified.

Mahmood et al. [7] presented a research in which they proposed a new method called A-Eye which was able to automate the role of third umpire. In their research, they apply their proposed method to a cricket video in which the player is getting run-out, and the results of their method was very accurate and very efficient. The A-Eye is basically a desktop application which was created by researchers in C-Sharp programming language. First, they implemented a video player in GUI then they split the videos into frames, then they try to detect the crease and the wicket after that the researchers tried to detect the run-out and for that they tried to detect the motion of the ball as it approaches the wicket and lastly, they measure the performance of MDA. For sample 1 their model took 0.45 sec, and third umpire took 45.4 seconds which is considerably a great performance and for video sample 2 their model took 0.53 second whereas third umpire took 35.9 seconds. The only limitation with their model is that it worked only in 2d environment.

In another research Aftab et al. [8] created a method which was able to detect the nick generated by ball connecting with bat or gloves or pad or a combination of bat or pad. First, they removed the noises from audio files by using the audacity program then train neural network after that they trained the system and then test it. There were total of 132 signals and researcher's system was able to achieve the accuracy level of 98.3% for collected videos and 85.7% for real cricket scenarios. They can implement audio sensors so that their system can work faster and efficiently.

Simon and Chilukuri [9] present a research in which they used snicko-analysis and a pattern-based recognition technique so that a cricket decision can be made intelligently. For catching and analyzing the audio coming from the contact of ball and bat the researchers have used snicko-meter. Their system was found out satisfactory and suitable for making decisions using snicko-analysis. Their system was able to get the connection of ball-bat 84% accurately, connection of ball-pad 80% accurately, connection ball-glove of 88% accurately and noise 92% accurately.

In another research, Asif et al. [10] presented a prototype for automating the umpire decisions by noticing the hands gestures of the umpires. First for selecting the hand of the umpire the researchers have used Haar-cascade-classifier and then for interpreting the gesture of the umpire the researchers have used logistic regression in their research. For Outs, their system got 88% accuracy, for No-ball, their system got 90% accuracy, for wide-ball, their system got 86% accuracy, for fours, their system got 83.33% accuracy and for six their system got 93.33% accuracy.

Julius et al. [11] presented a research in which they tried to recognize the gestures of referee from basketball game videos. They have used Support Vector Machines in this research. First, they got the images from videos and to detect the edges the researchers have used sobel edge detection method. They got the overall accuracy of 97% and F-score of 94%. In their research, the researchers should have used large dataset, for such a big task it was a small dataset.

In another research, Julius et al. [12] proposed a research in which they have used image segmentation technique based on histogram of oriented gradients and local binary pattern features. The researchers have used McDonald dataset for their research so that they can identify the basketball referee signals. There were total of 100 images with three types of hand signals which were player foul, stop clock and three points. After finalizing the dataset, the researchers used labeler application for preparing the data. Labeler application was able to highlight the area of interest which was used as classification data. The researchers trained their model so that it was able to recognize six classes (3 gestures): three classes from front and three classes from back. The researchers were able to get the overall accuracy level of 95.6% using LBP features and SVM classification. When researcher's system was used to recognize the stop time signal (from behind) it achieves the accuracy of 94%, when system was used for recognizing the stop time from front, it was able to achieve the accuracy of 94% as well. When it was used for recognizing the player foul from front it was able to achieve the accuracy level of 94% as well and when checked from front it got 95% accuracy level. When it was checked for 3 points from behind, it managed to get the accuracy level of 96% and when it was checked for 3 points from front, it was able to get the accuracy level of approx. 98%. When they use SVM with HOG feature the training time was 2 minute 25 second, when they use SVM with LBP feature the training time was 3 minute 10 second, when they use RF with HOG feature the training time was 2 minute 1

second and When they use RF with LBP feature the training time was 2 minute 40 second. SVM with HOG took 37 seconds for classification. SVM with LBP took 45 seconds for classification, RF with HOG took 39 seconds and RF with LBP took 30 seconds for classification. They have only analyzed isolated referee gestures which is a limitation.

III. METHODLOGY

In their project, we have implemented CRISP-DM to achieve our research goal.

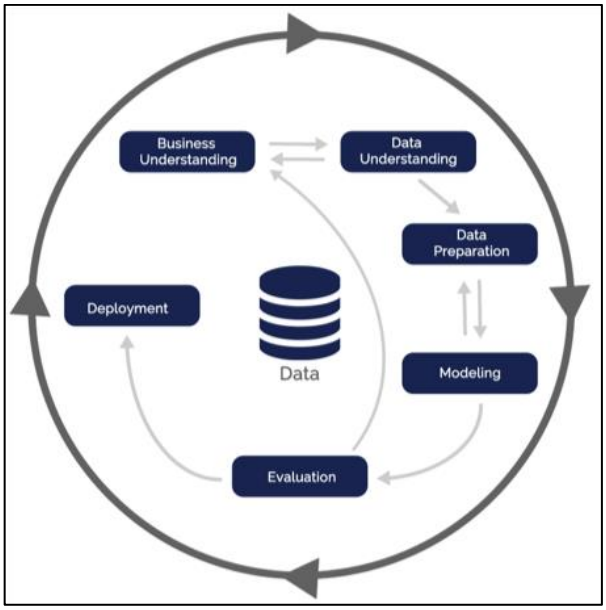

Figure 2. CRISP-DM

CRISP-DM stands for Cross-industry standard process for data mining. It consists six major steps which are:

i.) Business Understanding

ii.) Data Understanding

iii.) Data Preparation

iv.) Modelling

v.) Evaluation

vi.) Deployment

The first phase of CRISP-DM is Business Understanding. In this phase we try to understand what are the overall objectives and requirements from the view point of business and after understanding the objectives we try to convert this into data mining problem and prepare a plan for tackle that. For our project, our main goal was to detect the umpire gestures so that we can automate the scoreboards of cricket matches, because this automated process is less time consuming as well as it is very effective and efficient.

The second phase of CRISP-DM is Data Understanding, in this phase we start with collecting the data initially and try to get familiar with it so that we can understand the hidden information from the data and discover what is behind the top layer of the data. In our project, we created our own video dataset using our computer's camera and use google intelligence API. First, for creating this dataset, we have recorded 50 videos for no-ball signals, and 50 videos for out signals. After recording the data, we saved all the videos in a folder and upload that folder to google drive and from that drive we used those videos for performing data analysis. We will describe the whole dataset in further section.

Third phase of CRISP-DM is Data Preparation. In this phase we use all the knowledge we get from data understanding phase and construct the final dataset using that knowledge. In our project, we used two datasets. One dataset has been used for differentiate between umpire and non-umpire and the second dataset has been used for detecting the signals of the umpire. We have downloaded the first dataset from internet, we will explain about it in upcoming section and the second dataset that we have used for detecting the signals of the umpire, that dataset is created by us. After recording all the videos and uploading it onto the google drive our work for this phase had been done.

Fourth phase of CRISP-DM is Modelling. In this phase we select and implement modelling techniques. Since some techniques required us to prepare the data according to their requirements, therefore we can go to data preparation phase again from this phase. For our project we have used Transfer learning with Convolutional neural network (CNN) and Long short-term memory (LSTM). Transfer learning is the process of creating new models by upgrading the previously already trained neural networks. They don't train their neural network from scratch instead we can download pretrained model and made some changes to it according to the need and use it and these models are also open source. By making some changes, we want to say we can use feature extraction or we can train some layers while freeze other which mean training model partially. In our project we have used it with CNN. The second model that we have used in our project is Long short-term memory (LSTM). It is a type of repetitive neural network which is basically a multiple copy of same network in which each network passes a message to a successor. The recurrent neural networks are intimately related to sequences and lists. LSTMs are a special kind of RNN, capable of learning long-term dependencies. LSTMs are designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behaviour, not something they struggle to learn. LSTMs also have chain like structure, but repeating module has a different structure. There are four neural network layers in a LSTM.

Fifth phase of CRISP-DM is Evaluation. In this phase we check how our model or models have performed and on the basis of performance of model, we select the best model that meets our main business objectives and goals. In our project we are going to plot and check the graphs so that we can know if our model is under fit, over fit or good fit. If our model goes under fit or over fit, we will tune our parameters and plot the graphs again. We will plot the graphs again and again till when we got our model in good fit parameters.

Sixth and last phase of CRISP-DM is Deployment. In this phase we generally prepare a report of how we choose our project goal and how we tackle it using data mining technique, so that people who don't have much knowledge of data mining or people of higher authorities can understand it easily. In this phase we basically plan the deployment. What deployment means is we prepare a strategy by which we can use our model in working in our business. In this phase, we also plan how we will monitor and maintain our project after the deployment, report the finals results of our model which means creating the final report which summarizes the whole project and its results and lastly in this phase, we also review the final results.

A. Convolutional Neural Network (CNN):

For the techniques we have used, let's start with theory of these techniques beginning with the CNN theory. CNN stands for Convolutional neural network is a combination of artificial neural networks and a set of operations that are called convolutions. CNN is regarded as one of the main operations to use with regards to image recognition and image classification making it the best fit for our objective. For image classification the CNN models will input the image through a multiple array of layers called convolutional layers. These layers also have filters called kernels. The CNN algorithm when inputting an image will be preassigned importance based off the input model, these are the weights and biases.

The reason we have chosen this as our algorithm is because pre-processing and cleaning of the data is not as intense as other algorithms. The reason for CNNs being popular in image classification is the structure of the model executes a more accurate fitting to the image dataset which in our case with umpire recognition is needed. This accuracy is due to weights being reusable that we discussed earlier. Below we can see an outline image of the structure of a CNN.

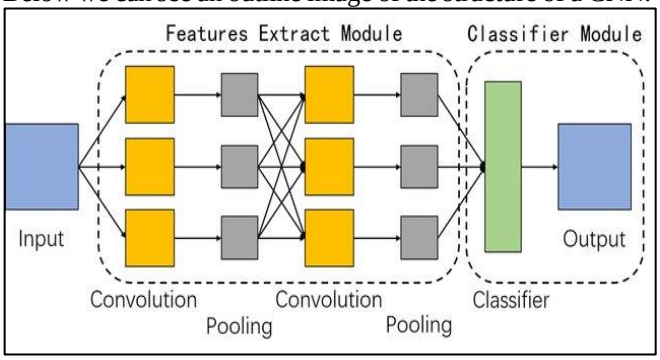

Figure 3. Structure of CNN

The figure above shows the image being inputted, after this the filters are then applied to that image which then we class this stage as the convolutional layer as seen in the Figure *3*. The linearity is then broken up of the image, this process involves the rectifier function.

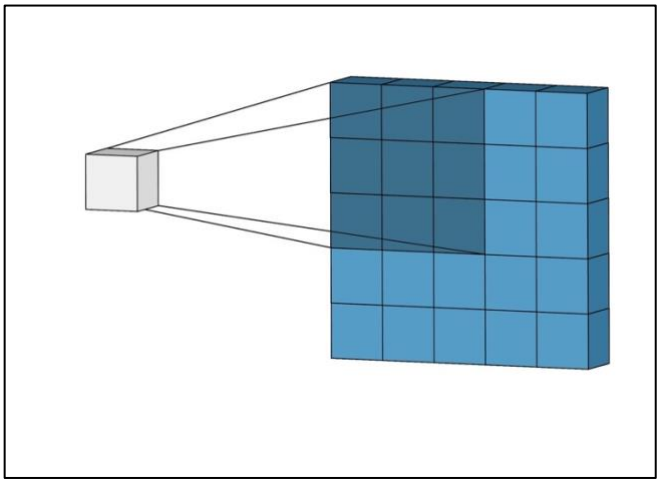

Figure 4. Convolutional Operation

Once that function is applied we can then turn to the next stage known as the pooling layer, the pooling layer offers a filter that is used on the image maps, the operation of pooling is based on the image map in terms of what size is the image map that determines the scale of operation. The pooling layer involves slashing each dimension in half so whatever the pixels dimesnion that will be halved. Average pooling and maximum pooling are the most used operations. Average calculates the average value for each pixel of the image with max pooling calculates the max for each pixel.

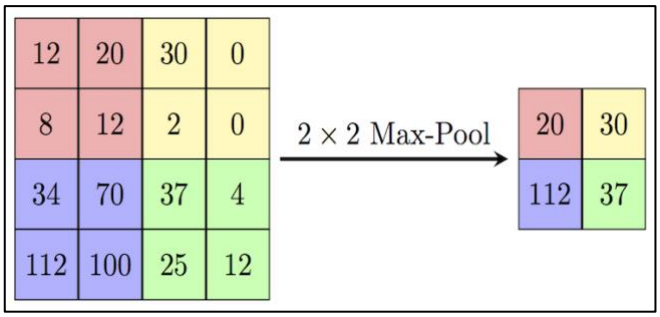

Figure 5. Max Pooling

With the spatial invariance in the CNN, the CNN has captured the spatial and temporal dependencies in the image, a pooled image map is then flattened and then can begin enterring the neural network once we have a fully connected layer.

B. Long Short Term Memory (LSTM):

Recurrent Neural Networks have several problems that affect them, one of which is short term memory, RNNs find it difficult to transfer information from the first stage to the last stage, because of this valuable knowledge is lost from the beginning of the model. This is the reason LSTM was created. LSTM which stands for Long Short-Term Memory are a detailed version of RNN, these networks can learn long term dependencies. LSTM are widely used to solve several problems are a big in computer vision and image description mainly. LSTM is also used for CNNs as well, image description being at the heart of this. See Figure *6* showing the architecture of where these two models fit.

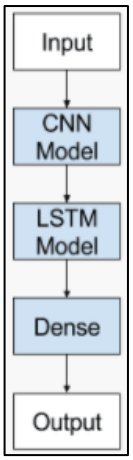

Figure 6. Convolutional Neural Network LSTM architecture

The way LSTM keeps infromation in early steps fresh is due to using memory blocks, this network is formed

by three gates, the input gates, the forget gate and the output gate, once the input gate goes through a simgoid layer on deciding what information needs to be added, the forget gate of LSTM uses a sigmoid layer and dot product to bring information thst is selected and then decides wheter to forget related information from a prevoius inputted cell by the matter of probability, so basically, the forget gate objective is to remove this information from the cell state that is of less relevance. The output gate is then gathered and multiplied by the vectors that passed the information through to a new layer (tanh layer). The tanh layer is basically an activation function used to allow the values to flow through the network. This is to diminish values that lie between -1 and 1 (nulls) as these vectors that flow through the netweok transofrm regularly due to the functions. The given inputs are multiplied by the weight matrices and a bias is added. This is where the sigmoid function is added and ouputs the vector as discussed and multiplied in the cell state. Although the output gate is basically deciding the next cell state, the hidden state which contains the first inputs is used for making the predictions.

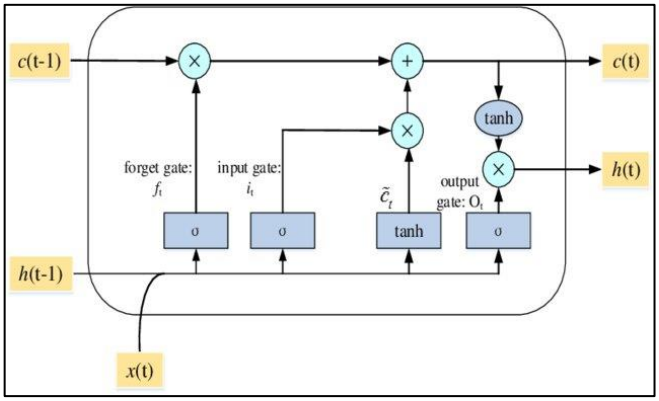

Figure 7. Inside structure of LSTM.

LSTM has been used to great success in image classification models upon research, which is why we decided to implement it with our CNN model. In [1] an image classification based on Fashion-MNIST Dataset. The model achieved an accuracy level of 89% with great time consumption. LSTM will be a great way to predict UMPIRE hand singals and a crucial part of the model we are creating.

C. Google Video Intelligence

Google video intelligence is a sophisticated API that comes with ready installed ML models that can detect actions amongst others in steams and videos presented to them. Part of the google cloud platform the video intelligence API allows users to conduct video analysis. The REST API allows for detailed analysis with information raging from the entire video/stream or per frame which in our case, with hand signals per frame is the one we will be analyzing.

Several steps must be followed to use the video intelligence API, like for example, cloud shell which is a command line environment for running applications in google cloud. Once this is done and the API is enabled, users can create a service account for the project they are creating, as service account can just be a Gmail or email account. Then just the basics of naming a working directory and then you can begin using Video Intelligence API to interpret videos or images that have been stored in the google cloud of your account or can be implemented using data bytes. Video Intelligence API can even detect content that as seen as explicit or over 18s, this is done as the same way we are looking to incorporate with our Umpire hand detection signals by using frame per frame visual signals, this is used without the analyses of audio.

If we look closer to our objectives for the project in terms of video intelligence API, we need to look at the detection of people, to do this we needed to send a video annotation request, this request will reveal all the details about the people detected in a video/image, this means the location of the people as well, this can be formatted in the shape of boundary boxes around the people upon detection. This is done by the PERSON_DETECTION flag.

For our project we need to do one better and not just detect a person but detect a body part, the hand of the umpire. The location of this body part can be also done with providing the values such as ‘right wrist or left wrist’ to detect the hand signals, this landmark can be done on a numerous number of values of the body parts. To identify in our case, a sports event, typically cricket, the Umpire will be in different attire than the players so clothing can also be used as a characteristic to detect the person in the video/image.

Characteristic or landmark	Possible values
Body parts	nose, left_eye, right_eye, left_ear, right shoulder, right_shoulder, left_elbow, rig wrist, right_wrist, left_hip, right_hip, l knee, left_ankle, right_ankle
Upper clothing color	UpperCloth:Black, UpperCloth:Blue, Upper UpperCloth:Gray, UpperCloth:Green, UpperCloth:MultiColor, UpperCloth:Oran UpperCloth:Purple, UpperCloth:Red, Upper UpperCloth:Yellow
Upper clothing type	UpperCloth:Coat, UpperCloth:Dress, Upper UpperCloth:Shirt, UpperCloth:Suit, Upper UpperCloth:T-Shirt, UpperCloth:TankTop

Figure 8. The characteristics library of detection by values.

With this, we can see in the Figure 8, what the output would look like in google cloud.

Figure 9. Google cloud output API

We can see the image is clearly detecting every object in the video at present time. This is the main reason we are using this

approach along with the architecture built of the CNN model and the LSTM theory we have talked about in the last section.

IV. DATASET

For this project, we have used two datasets. One dataset that we have used in our project is downloaded from GitHub website and that dataset is known as SNOW dataset and the second dataset that we have used in our project, we created it ourselves. In the second dataset we recorded 50 videos for each 2 actions of umpires. In total we record 100 videos of umpire actions. The two signal actions that we have recorded are No ball and Out.

SNOW dataset was used for detecting the umpire and non-umpire in our project. There are total of 780 images in this dataset in which 390 images are of umpire and 390 images are of non umpires. Figure 10 and Figure 11 are an example for telling the difference between these two types of images. Figure 10 is an example of non- umpire and Figure 11 is an example of umpire. The SNOW dataset is used for testing and training our models so that it can recognizing the umpire and non-umpire and we can get the accurate and better results.

Figure 10. Example of non-umpire

Figure 11. Example of umpire

The second dataset that we have used in our project is created by ourselves. We have used the Apple MacBook Air's 720p Facetime HD camera. We recorded the actions of umpires using this MacBook's camera. In total we recorded 100 videos for 2 actions of umpires. We recorded 50 videos for no ball signals and 50 videos for out signal. Figure 12 is an example of signal for No Ball and Figure 13 is an example for signal of out.

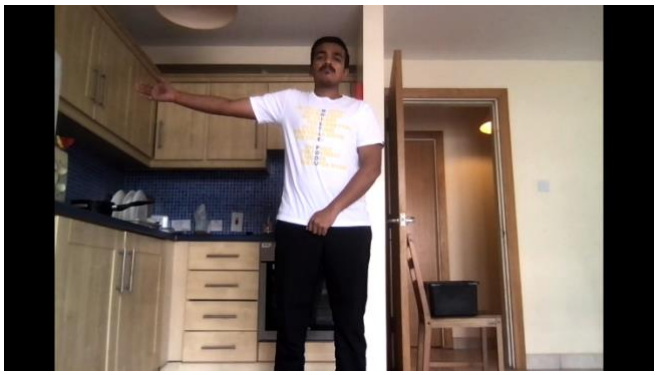

Figure 12. Signal of No ball

Figure 13. Signal of Out

We first store all the videos with all the actions in separate folder and then store that folder onto the google drive and from that google drive, we use it for our modelling. Since we are going to use the google intelligence API, we found google drive as a better option to store our dataset images and videos and use it later from there. We will explain all the process of modelling and how we have used these datasets for modelling in upcoming section.

V. EVALUATION AND RESULTS

A. Umpire Classfication:

We have divided our project into two stages. In first stage we will need to classify the umpires. We will be using SNOW dataset which consists two type of data. First dataset contains umpire and non-umpire images. Second dataset contains umpire showing different signals such as no ball and out, but we will be using different dataset for umpire sing detection which we will discuss further in details. Our first model will be trained to classify umpires on the ground. Umpire dataset folder contains the images of various umpires whereas in non-umpire dataset contains the images players playing on the ground. We will be using the transfer learning approach in our study. It is one of the best approaches if we want to build a good model using less data. We have already discussed about the transfer learning in the methodology section. There are various models available on Keras like VGG, Inception, Resnet, Mobilenet which can be used for the transfer learning, but transfer learning has a huge advantage because we can simply use the same template on various models and can decide which model is best suited for our study. In our study, we will be using VGG16 model. The architecture of VGG 16 model is simple to understand.

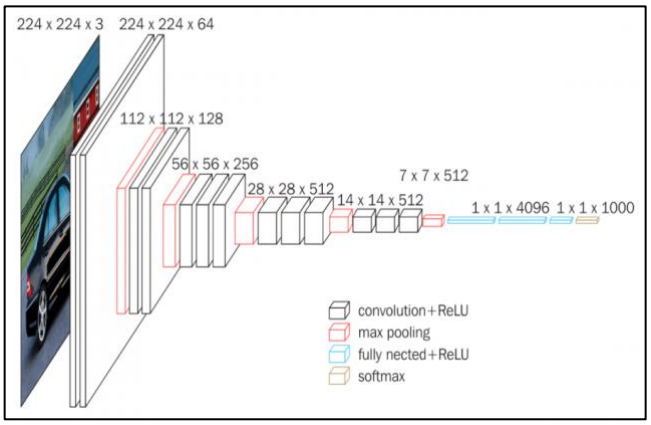

Figure 14. VGG 16 Architecture

As we can see in the Figure 14, VGG 16 model takes an input image of size 224*224. The image size is appended with encoding of RGB intensity. The number 3 denotes that image is a color image. The first two layer used in VGG 16 are convolutional layer with kernel size of 64. Further they have used one max pooling layer along with two convolutional layers. Again, they have used one max pooling layer of reduced size of 56*56 along with three convolutional layers and they repeated same order of max pooling and convolutional layers for two time and at each time input size of layers is decreasing by 50% of previous layer. This decrease in the size of filters is due to the size of stride of max pooling layer which is almost twice the convolution layer. So, whenever the max pooling layers is added the filter size has been reduced to half. At the end of the model, they appended three full connected layers and output of the VGG16 model is 1000 because this model was trained on ImageNet dataset which consisted of 1000 categories. We will now understand the implementation of transfer learning in our first model using VGG model. We had first implemented the model without tuning and, we did not used data augmentation technique and found that our model performed very poor, and we will need to use few techniques which will help model to learn better. At first, we have imported all libraries which were required for transfer learning which can be found on keras portal. We have initialized the image size (224*224) as VGG16 input will be expecting the image of this size. It is not mandatory to use this size specifically, but we are using it for a better practice in our study. As we imported VGG16 library no we can use the VGG16 function and will pass the necessary parameters such as image size along with the RGB channel value, ImageNet weights are used, and last parameter suggest that remove the first and last layer of the VGG16 model which is obvious that first layer will take the input image size, and in our case our last layer should be for two categories. We will add tow dense layers at the output of the model. Since we imported the model, we must initialize that apart from last layers which we will add later all other layers should not be trained. We are using the existing weights of ImageNet model so it very important to initialize the layers as not trainable. Now we can add our layers at the output, so first we will flatten the last layer of VGG16. After flattening the last layer, we also add a dropout layer. Dropout layer helps to avoid the overfitting of the data. We have passed the dropout value equals to 0.2 which suggest that at each layer 20 percent of neuron will be activated and rest of the neurons will be deactivated. Further we calculated the number of classes for prediction based on number of folders in the directory.

```
Model: "model"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_1 (InputLayer)         [(None, 224, 224, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 56, 56, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 28, 28, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 28, 28, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 28, 28, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 14, 14, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 14, 14, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 7, 7, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dropout (Dropout)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 2)                 50178
=================================================================
Total params: 14,764,866
Trainable params: 50,178
Non-trainable params: 14,714,688
_________________________________________________________________
```
Figure 15. Model Architecture

We have used Adam optimizer with learning rate of 0.001 during our model compilation which is not that low that can overfit the data. We have used the data augmentation technique in our study because our training dataset was quite low. This technique was the most important step of our project because it increased the performance of our model greatly. Data augmentation helps to generate multiple copy of image but with a change like an image being rotated at 40-degree, image being shifted to left to 20%, image being shifted to up to 20%, image being stretch by 20% at one axis, image being zoomed to 20%, image being horizontally flipped, and empty pixel is substituted with the closest pixel value. We looped all our initial training data through this augmentation function and stored augmented data of each image in our new training folder. We have used this new training data to produce our model. Further, we trained our model.

```
Epoch 1/5
108/108 [==============================] - 1363s 13s/step - loss: 0.1176 - accuracy: 0.9659 - val_loss: 0.1640 - val_accuracy: 0.9412
Epoch 2/5
108/108 [==============================] - 1357s 13s/step - loss: 0.1128 - accuracy: 0.9691 - val_loss: 0.1643 - val_accuracy: 0.9338
Epoch 3/5
108/108 [==============================] - 1352s 13s/step - loss: 0.1046 - accuracy: 0.9703 - val_loss: 0.1585 - val_accuracy: 0.9338
Epoch 4/5
108/108 [==============================] - 1358s 13s/step - loss: 0.0967 - accuracy: 0.9741 - val_loss: 0.1599 - val_accuracy: 0.9265
Epoch 5/5
108/108 [==============================] - 1363s 13s/step - loss: 0.0928 - accuracy: 0.9717 - val_loss: 0.1585 - val_accuracy: 0.9338
```
Figure 16. Output of model_fit

Figure 16 is the output we got after training our model, we have used total 5 epoch for our training. Steps per epoch is equal to the length of training data and validation steps is equal to the length of test set. We can observe that loss is decreasing for both training and validation data also accuracy of training data shows and upward trend whereas the accuracy is fluctuated but percentage of fluctuation is limited to less than 1%. Model is good fit when the loss value of the training data and validation data is almost equal and obviously both values should be less.

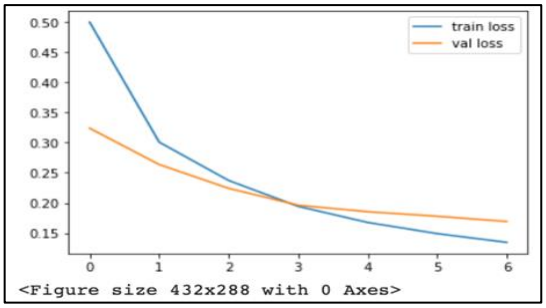

Figure 17. Plot of loss

As we can see in the Figure 17, the loss is decreased for both training and validation data and at epoch the loss of both training and validation data is decreased and almost becomes an equal which is shows that at this point our model is fit good and we can stop the training. Our model is now good to classify the umpires on the ground.

B. Umpire Pose Classification:

We will arrange these two models serially so that frames at which umpires are classified by our first model will be given to our second model. For producing the second model we have used the video data. We have performed our code on google colab and we had imported our drive directory which contains videos of umpire giving signs. It consists of 2 classes each class represent one signal shown by the umpire. We have used Google Video Intelligence API to detect the person in our video data and output of this API call will give a data frame which consists records of body part coordinates of person detected in the video at every 10 seconds. Our first step was to load the drive directory which consist our video data of the umpire signal. Further, we had to set up few things at Google Cloud services to call API.

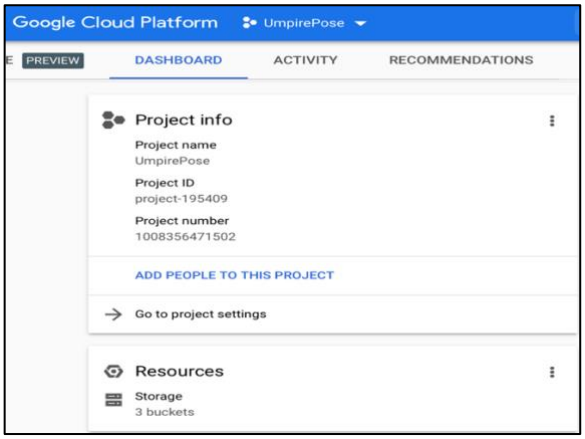

Figure 18. Google Cloud Setup.

We have created a project in the google cloud platform and system itself generate its project id and project number. Project id will be required while calling and API. After project info we can see the resources tab which consists of our training and test. We need to upload our train data in training bucket and our test data in test bucket. After uploading all files in the respective buckets, we will initialize few parameters such as project_id, bucket, service_account_name to make an API. Once we initialized all parameters, we can run the API call. Now we can use the Google Video Intelligence feature to detect person in the videos. We have implemented separate code for each signal to generate the body parts coordinate data. We got the function of detect_person on the google cloud video intelligence. This function uses the auto machine learning feature of google cloud to detect person in the video and it gives body part coordinate of that person at every 10 sec till the video is finished. We have the option of storing this data in json so all the json files will be stored in the buckets. This json file will contains other attributes too like color of shirt or t-shirt of detected person in the video, trousers of detected person, etc. So, we will need to select only those attributed which are related to body parts.


```
"landmarks": [ {
  "name": "nose",
  "point": {
    "x": 0.46820417,
    "y": 0.15418161
  },
  "confidence": 0.99137962
}, {
  "name": "left_eye",
  "point": {
    "x": 0.48312333,
    "y": 0.14076057
  },
  "confidence": 0.9992919
}, {
  "name": "right_eye",
  "point": {
    "x": 0.45626882,
    "y": 0.13628687
  },
  "confidence": 0.9984774
}, {
  "name": "left_ear",
  "point": {
    "x": 0.5040102,
    "y": 0.16760263
  },
  "confidence": 0.96607745
}, {
  "name": "right_ear",
  "point": {
    "x": 0.43836579,
    "y": 0.15865526
  },
```

Figure 19. Body Part Coordinate JSON Data

Figure 19 shows the result of person detection function in the google video intelligence. We stored all json files on our drive. Json file for each video has been generated and the file contains the result of person detection for every 10 second of the video, we will get the timestamps form this json file. We have created an anaylzePerson function which is used during json parsing. This function is a helper function to arrange the data according to time which will help us to do our task easily. After parsing the data, we have checked if there are any NA values in the data. There are almost 35 columns in the data after parsing the json file. These columns hold the value of x and y coordinates of body parts such as nose, eye, ear, shoulder, elbow, wrist, hip, knee, and ankle. We have observed the NA's mostly in knee and ankle because of the way we had recorded the video in which knee and ankle are not visible and these body parts are not used for giving any

signal. The main body part in our data is wrist and our model can learn with the help of wrist data.

	timestamp	nose_x	nose_y	...	left_hip_y	right_hip_x	right_hip_y
0	0.0	0.468204	0.845818	...	0.389503	0.426430	0.393977
1	0.1	0.469055	0.852183	...	0.386314	0.427625	0.386314
2	0.2	0.466370	0.852608	...	0.392038	0.428696	0.392038
3	0.3	0.466311	0.850931	...	0.387964	0.427712	0.383512
4	0.4	0.464404	0.852926	...	0.391466	0.426298	0.387071
...	...	...	...	...	...	...	...
1396	2.6	0.486819	0.668361	...	0.222033	0.429662	0.229174
1397	2.7	0.484276	0.666302	...	0.222535	0.429376	0.226113
1398	2.8	0.484198	0.665778	...	0.218785	0.429342	0.225937
1399	2.9	0.484944	0.665427	...	0.218670	0.430117	0.225818
1400	3.0	0.481619	0.665694	...	0.222333	0.429154	0.229484

[1401 rows x 27 columns]

	timestamp	nose_x	nose_y	...	left_hip_y	right_hip_x	right_hip_y
0	0.0	0.469584	0.677368	...	0.254357	0.424912	0.254357
1	0.1	0.469455	0.677639	...	0.254535	0.424773	0.258061
2	0.2	0.469671	0.677104	...	0.248575	0.424791	0.255658
3	0.3	0.470505	0.671053	...	0.255565	0.426259	0.255565
4	0.4	0.474493	0.668530	...	0.259431	0.427850	0.259431
...	...	...	...	...	...	...	...
1572	3.1	0.499710	0.707671	...	0.269655	0.451019	0.269655
1573	3.2	0.500570	0.706815	...	0.269074	0.449477	0.269074
1574	3.3	0.500874	0.707942	...	0.267449	0.449459	0.267449
1575	3.4	0.500768	0.706398	...	0.269732	0.451819	0.269732
1576	3.5	0.501760	0.707995	...	0.268895	0.450508	0.268895

[1577 rows x 27 columns]

Figure 20. Data-frame of no ball and out signals

We will need to create a sequence data for training a LSTM model We have created the sequence of the data depending on the time stamps as time stamp of each vide start from zero, so it was easy to differentiate the videos based on timestamps. For the first video of no ball dataset, Google video intelligence API has provided us the data of person for 34 frames. Similarly, the next video start from the 35th row and ended at 63rd row. Further we had implemented the data preparation steps like making position at each timestamp relative to beginning position. This step made the first frame data of training videos to 0. So all body coordinates will be initialised to zero at first frame of every videos. We have converted this data frame in to a dictionary and further with the help of key and value function of the dictionary, we were able to store the data in a NumPy array which consist only values. We have plotted the frequency plot of length of sequences for no ball as well as out.

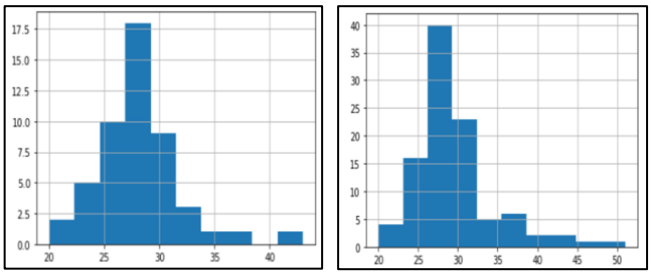

Figure 21. Frequency plot of length of sequences of no ball and out

Figure 21 suggest that we will need to pad the sequence because of varied length of sequence. In our study, we have decided to pad the sequence with the values in last row to maximum length. In no ball case, we can see the maximum length of sequence is 44. We will need to calculate the difference of length of each sequence and this maximum length sequence, and this output will suggest us to add how many rows after the current length of sequence. If length of sequence is 36 and maximum length of sequence is 44 then the difference of length of sequence is 8. So, we will have to add 8 rows more in the sequences and in this case, we are adding the 8 rows consisting of zeros. Each sequence will go through this step and updated sequences are stacked one over another after padding. After padding we will need to truncate the sequence to a length because LSTM uses input sequences of same length. In our study we are truncating the length of all sequences to 24. We have used pad_sequences function in which we passed the stacked sequence and initialized the maxlen equals to 24 also we have initialized the post padding which will pad after each sequence and we have initialized the truncating to pre which will truncate or remove the values from sequence larger than maxlen at the beginning of the sequence. After executing all these steps, we can now proceed to the splitting of the data. The shape of final sequence of both no ball and out data is (50,24,26) where 50 is number of videos of each category and (24,26) is the input shape which is now constant for each sequence. We had stacked the no ball and out sequence also we have created the labels for no ball and out. We had used splitting function to split the train and validation data. We used 75% percent of the data for training and 25% of the data for validation. Since our data is ready for training, we had initialized our LSTM model. We have first initialized the model to be a sequential model. Further, we added our first layer of LSTM in which we passed the number of neurons at input equal to 32 which is also known as the batch size. Next parameter is an activation function which is tanh in default this is because LSTM were designed to mitigate the vanishing and exploding gradient problem. In LSTM, apart from the hidden state vector each LSTM cell maintains a cell state vector and at each time step the next LSTM can opt to read from it, write from it or to reset the cell using an explicit gating mechanism. Each unit has three gates of the same shape. Each gate is a binary gate. Input gate controls whether memory cell is updated. Forget gate controls whether memory gate reset to zero and Output gate control whether information of current cell state is made visible or not. These gates have sigmoid activation to constitute a smooth curve from 0 to 1 and which helps model to remain differentiable. Apart from these gates, there is vector which modifies the cell state, and this vector has tanh activation because with a zero centered range a long sum operation will distribute the gradients well. This allows the cell state information to flow longer without vanishing or exploding. Next parameter is input shape which indicates the timesteps and number of features. Time steps suggest how long in time each of your sample is. In our study the input shape is (24, 26) which is suggested that there are 26 number of features present in the data and timestep is equal to 24. Last parameter of our model is return_sequences which returns the final sequence. We have added the dropout layer which means the data on few input connections to each LSTM block will be excluded. In our study we have initialized the dropout to 0.3 which suggests that 30% of input connection will be deactivated. This is applied to avoid the overfitting and underfitting problem. After dropout layer we have again added the LSTM layer with batch size equals to 32 and last layer is a dense layer which have activation function of softmax because we have two categories in the dataset. Further we have initialized the loss which is sparse categorical entropy and therefore we have not implemented the one hot encoding of the label and converted them to integer only. This loss takes low computation memory compared to other loss. We have used the adam optimizer which is used to update the weights. We have completely initialized the model and now we can now start training our model. We have used 5 epoch and 25% of data is considered

as validation data. We will now interpret the results of our model.

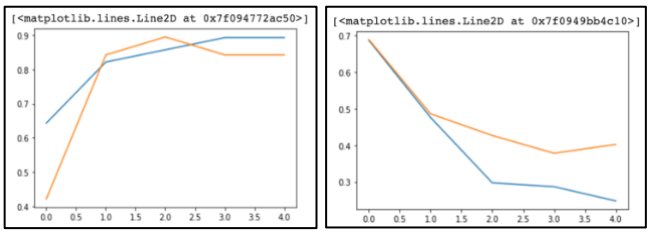

Figure 22. Plot of accuracy and loss of training and validation

Figure 22 suggest that the training accuracy is growing smoothly compared to the validation accuracy which suggest that model is trying to overfit. We can also observe the loss graph where loss of validation was decreasing and again it got increased which suggest that model is overfitting. We tried to change the parameters to make model perform better by changing the batch size, increasing, or decreasing the dropout layer, increasing, and decreasing number of epochs but we were not able to produce a model which fits well.

VI. Conclusion and Future Work

We have implemented the two model in which first model was produced with the help of transfer leaning to classify the umpire and players on the ground and based on this result we can pass the umpire classified frame to our next model to predict the signal shown by the umpire. Our first model which is used for umpire classification has a training accuracy of 97.17% and validation accuracy of 93.38%. Whereas our second which is used for umpire signal classification has a training accuracy of 83.15% and validation accuracy of 84.21%. We have used our own generated data for second model so in future work we can generate the data directly from umpire and record the dataset of that umpire showing different signals as well as we can use different umpires showing different signals to make our model perform better. We can increase the size of dataset to make our model more generalize for predictions.

References

[1] Aravind Ravi, Harshwin Venugopal, Sruthy Paul, Hamid R. Tizhoosh, A Dataset and Preliminary Results for Umpire Pose Detection Using SVM Classification of Deep Features, 2018 IEEE Symposium Series on C omputa tional Intelligence, November 2018.

[2] Hari R., Wilscy M., Event Detection in Cricket Videos Using Intensity Projection Profile of Umpire Gestures, Annual IEEE India Conference, 2014.

[3] Md. Kowsher, M Ashraful Alam, Md. Jashim Uddin, Faisal Ahmed, Md Wali Ullah, Md. Rafiqul Islam, Detecting Third Umpire Decisions & Automated Scoring System of Cricket, International Conference on Computer, Communication, Chemical, Materials and Electronic Engineering, 2019

[4] R, Rock, A. Als, P. Gibbs, C. Hunte, The %th Umpire:Automating Cricket's Edge Detection System, Systemics, Cybernetics and Informatics, 2014.

[5] Sanjeev Kumar Huggi, Dr. Suvarna Nandyal, Detecting Events in cricket videos using RF Classifier, International Journal of Advanced Research Foundation, 2016.

[6] Vaishnavi K. Nair, Raakhi Rachel Jose, Parvathy B. Anil, Minnu Tom, Lekshmy P.L., Automation of Cricket Scoreboard by Recognizing Umpire Gestures, IJISME, 2020.

[7] Tariq Mahmood, Syed Obaid Ahmed, Syed Hassan Nayyar, Nuhammad Hadi Swaleh, A-Eye: Automating the role of the third umpire in the game of cricket, Expert Systems with Applications, 2012.

[8] Aftab Khan, Syed Qadir Hussain, Muhammad Waleed, Ashfaq Khan, Umair Khan, An automated snick detection and classification scheme as a cricket decision review system, Turkish Journal of Electrical Engineering and computer Sciences, 2019.

[9] Simon Ting, M. V. Chilukuri, Novel Pattern Recognition Technique for an Intelligent cricket decision making system, I2MTC, 2009.

[10] Md. Asif Shahjalal, Zubaer Ahmad, Rushrukh Rayan, Lamia Alam, An approach to automate the scorecard in cricket with computer vision and machine learning, EICT, 2017.

[11] Julius Zemgulys, Vidas Raudonis, Rytis Maskeliunas, Robertas Damasevicius, Recognition of basketball refree signal from videos using histogram of oriented gradients and support vector machines, ANT, 2018.

[12] Julius Zemgulys, Vidas Raudonis, Rytis Maskeliunas, Robertas Damasevicius, Recognition of basketball referee signals from real-time videos, Journal of Ambient Intelligence and Humanized Computing, 2019.