AI CRICKET SCORE BOARD

P Radha Rai[1] ,Students, Dept. of CSE, Vemana Institute of Technology, Bengalore
Prakhar Kumar Chandraker[2] ,Students, Dept. of CSE, Vemana Institute of Technology, Bengalore
Shaik Nowsheen[3] ,Students, Dept. of CSE, Vemana Institute of Technology, Bengalore
Shantanu Kumar Singh[4] ,Students, Dept. of CSE, Vemana Institute of Technology, Bengalore
Ms. Veena G[5], Assistant Professor, Dept. of CSE, Vemana Institute of Technology, Bengalore

***Abstract*— Gesture Recognition pertains to recognizing meaningful expressions of motion by a human, involving the hands, arms, face, head, and/or body. It is of utmost important in designing an intelligent and efficient human–computer interface. The applications of gesture recognition are manifold, ranging from sign language through medical rehabilitation, monitoring patients or elder people, surveillance systems, sports gesture analysis, human behavior analysis etc., to virtual reality. In recent years, there has been increased interest in video summarization and automatic sports highlights generation in the game of Cricket. In Cricket, the Umpire has the authority to make important decisions about events on the field. The Umpire signals important events using unique hand on signals and gestures. The primary intention of our work is to design and develop a new robust method for Umpire Action and Non-Action Gesture Identification and Recognition based on the Umpire Segmentation and the proposed Histogram Oriented Gradient (HOG) feature Extraction oriented of Deep Features. Primarily the 80% of Umpire action and non-action images in a cricket match, about 1, 93,000 frames, the Histogram of Oriented Gradient Deep Features are calculated and trained the system having six gestures of Umpire pose using Densenet121.**

***Keywords*— Python, Artificial Intelligence, Jupyter, Cricket Score, Sign Language Recognition, Umpire, Histogram of Oriented Gradients, DenseNet121,CNN.**

I. INTRODUCTION

Gestures are known as expressive, meaningful body movement which involves physical motion of the fingers, hands, arms, head, face, or body with the intent of assigning meaningful information or interacting with the environment. In recent years' gestures are widely used by humans to interact with computers and machines. Many common everyday equipment like TV, Smartphone, Car dashboard etc. can now be controlled by simple hand gestures. Gestures are also applied in many fields like developing aids for the hearing impaired, enabling very young children to interact with computers, recognizing sign language, medically monitoring patients' emotional states or stress levels, lie detection, monitoring automobile driver's alertness/ drowsiness levels, etc. Involvement of gesture recognition technology in sports will make the gameplay fairer and proficient.

The gestures performed by the sports officials which indicate what is going on in the game. Which also can provide something meaningful about a player or the entire game. If the gestures of these officials are able to be recognized, meaningful information can be derived. We refer to a gesture as an intentional action by a person whereby part of the body is moved in a predefined way to indicate a specific event. Detecting these events enables automatic generation of highlights, contextual labeling of video and more importantly helps in decision making and automatic score update.

Cricket is the most popular sport in India. It is the 2nd most famous sports in Asia, 4th in Europe and also 2nd most famous sport in the whole world. But from the 19th century the same old manual method is being use to update the scoreboard, which is a great burden for the scorekeeper. The way of viewing the score has changed a lot in time, but the basic score updating process is still the same and performed by a person. So, in the 21st century an automatic system is very much needed at this sector. Automatic systems are taking places of many boring manual task which was performed by humans 5 or 10 years ago. So, developing a fully operational automated system like this is in dire need. Using modern equipment's and machine learning algorithms we can increase the accuracy of the decision and provide flawless result what the naked eye misses.

On the other hand, in sensor-based system the person who perform the gesture wears sensor, and when the gesture is performed the sensor data taken and used to identify the gesture. In the area of sports and other sector, several attempts have been made for gesture recognition using both sensor-based and vision-based technique. [10]

II. LITERATURE SURVEY

2.1 Title: Low cost approach for Real Time Sign Language Recognition

Identify the signs and convert them into text and speech using appearance based approach with a low cost web camera. A series of image processing techniques with Hub-moment classification was identified as the best approach. Uses fast Convex Hull Algorithm for Binary image for pattern recognition, Histogram based classification. During the research available human computer interaction approaches in posture recognition were tested and evaluated. A series of image processing techniques with Hu-moment classification was identified as the best approach. The system is able to recognize selected Sign Language signs with the accuracy of 76% without a controlled background with small light adjustments.

The advantage is it helps in identifying a low cost, affordable method that can facilitate hearing and speech impaired people to communicate with the world in more comfortable way where they can easily get what they need from the society and also can contribute to the well-being of the society.

The disadvantage of this project only looks at the hand postures not on hand gestures.

Title: Static Hand Gesture Recognition Based on Convolutional Neural Networks

Proposes a gesture recognition method using convolutional neural networks. The procedure involves the application of morphological filters, contour generation, polygonal approximation, and segmentation during preprocessing, in which they contribute to a better feature extraction. Segmentation algorithms can be implemented to separate, colors, textures, points, lines, discontinuities, borders, among others. Training and testing are performed with different convolutional neural networks, compared with architectures known in the literature and with other known methodologies. All calculated metrics and convergence graphs obtained during training are analyzed and discussed to validate the robustness of the proposed method. The images are obtained from the database.

The advantage of this paper is proposed methodology are much simpler and have a lower computational cost.

The disadvantage of this paper is proposed methodology approaches only cases of gestures present in static images.

2.3 Title: Hand Gesture Recognition Systems with the Wearable Myo Armband

The hand gesture recognition systems deal with identifying a given gesture performed by the hand. Utilized machine learning techniques to recognize the hand gestures. Seven different time domain features are extracted from the raw EMG signals using sliding window approach to get distinctive information. The performance of KNN, SVM and ANN algorithm will be compared. Then, the dimension of the feature matrix is reduced by using the principal component analysis to reduce the complexity of the deployed machine learning methods.

The advantage is to achieve good accuracy.

The disadvantage is need to testing the proposed method with recording signals from different people.

2.4 Title: Gesture Recognition to Make Umpire Decisions

The Umpire Gesture Recognition System aims squarely to introduce a more robust technology to show Umpire choices with the assistance of Gesture Recognition and trailing of hand movement of the Umpire. This technology helps to alleviate the burden of the scorekeepers. Authors tested the subsequent six gestures particularly OUT, SIX, NEWBALL, NO-BALL, DEAD_BALL, FOUR. Edge detection algorithms which emphasize edges and transitions. The Umpire Gesture Recognition System aims squarely to introduce a more robust technology to show Umpire choices with the assistance of Gesture Recognition and trailing of hand movement of the Umpire.

The advantage of this is capable of recognizing a group of six umpire gestures from the game of cricket.

The disadvantage of this is no performance of segmenting gestures.

2.5 Title: Automatic Labeling of Sports Video Using Umpire Gesture Recognition

A hierarchical hidden Markov model to solve the problem of automatic segmentation and robust gesture classification. The algorithm proceeds as follows: for each period of movement ahead in time (up to 10sec) of the start of a candidate gesture, calculate the likelihood of each model for that region. Gestures can be considered to exist at multiple levels in a hierarchy, where simple movements are grouped into more complex movements and complex movements are grouped into ordered sequences. The advantage of hierarchical modelling is this temporal decomposition of gestures.

The advantage is system performs well overall with the exception of handling unknown movements which have similarities to known movements.

The disadvantage is filler ratio requires further investigation for deciding when a known gesture occurs.

III. EXISTING SYSTEM

The existing system of the project is to detect the umpire gesture and update it onto the final score board. in real time ICC matches the umpire has a counter device which contains buttons on it such as wicket's, balls, overs, through which umpire sends the decision to the score board updater where he analyzes and then display the output onto the board. everything happening in real time is manual in consideration with the existing project's the updation in this is that it does not consider the crowd and unwanted noise in the stadium it focuses only on the umpire's signal, its feature. whereas existing projects contains detection of both umpire and players signal. The system relies heavily on the quality of the input data, including the video feed and the training dataset. If the video feed is of poor quality or the training dataset does not cover a diverse range of gestures and actions, the system's accuracy will be compromised. Additionally, the system is not foolproof and can make errors in recognizing gestures, especially if the umpire's gestures are not distinct or if there is interference from other factors, such as noise or movement in the background. Another limitation of the system is that it may not be able to recognize new gestures that are not part of the training dataset, which could limit its overall effectiveness. Finally, the system requires specialized equipment and software, which could make it expensive and challenging to implement in all cricket stadiums. Therefore, while the gesture recognition system has the potential to improve the accuracy and efficiency of updating the scoreboard, it is important to consider its limitations and drawbacks before implementing it in real-world scenarios

IV. PROPOSED SYSTEM

In this project, an umpire action and non-action and gesture detection and classification is developed based on Histogram of Oriented Gradients (HOG) and CNN. The general methodology of the proposed technique includes division, highlight extraction, and umpire activity non-activity and gesture outlines arrangement.

At first, the Umpire database have been created using OpenCV and the 80% of Umpire Action, Non-action and gesture frames are selected manually and feature extraction is performed based on HOG which is trained using DenseNet121 model. After that, the remaining 20% of the frames, feature extraction is performed using HOG and tested using trained DenseNet121 model, which categories the Umpire Action , Non-Action and gesture Frames.

V. DESIGN METHODOLOGY

A. System Architecture:

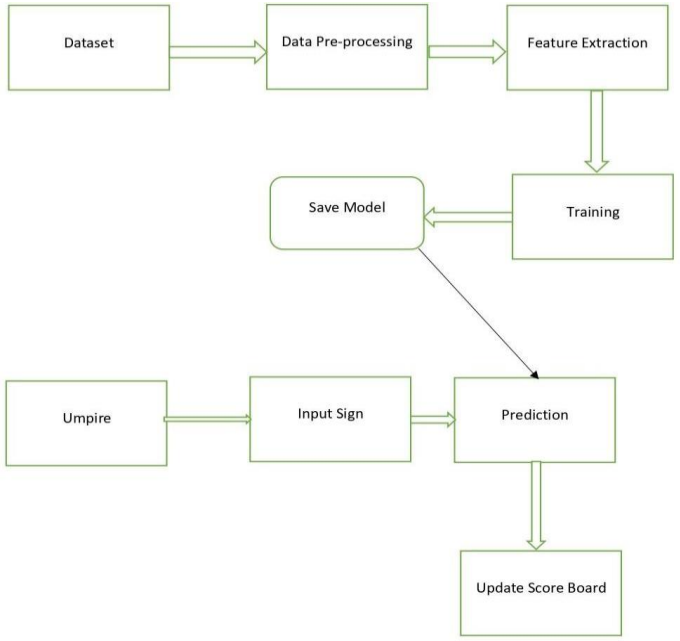


Fig. 1. System Architecture

The above figure depicts the system architecture of AI cricket score board.

B. Module Description:

A module description provides detailed information about the modules.

- Create Dataset of Umpire and Gestures
- Preprocessing Gesture dataset
- Train and test gesture dataset
- Preprocessing Umpire dataset
- Train and test umpire dataset
- Capturing and detecting Umpire
- Detecting Hand Gesture of Umpire

1. Create Dataset of Umpire and Gestures:

This Python module is designed to continuously capture images from a camera until a specific number of images is reached. It utilizes the OpenCV (cv2) module for capturing and displaying images from the camera and the numpy module for manipulating image data.

2. Preprocessing Gesture dataset

This process involves applying various techniques to a collection of hand gesture images to enhance their quality and prepare them for further analysis or use. First, it imports the necessary modules such as os, time, cv2, numpy, and matplotlib.pyplot. Then, it defines the folder path where the dataset is located and the folder path where the processed images will be saved. Next, the script reads the list of gestures from the dataset folder using the os.listdir() function and prints them. It then enters a for loop that iterates over each gesture folder in the dataset, creates a new folder with the same name in the processed image folder using the os.mkdir() function, reads the list of images in the gesture folder, and enters another for loop that iterates over each image in the folder.

3. Train and test gesture dataset

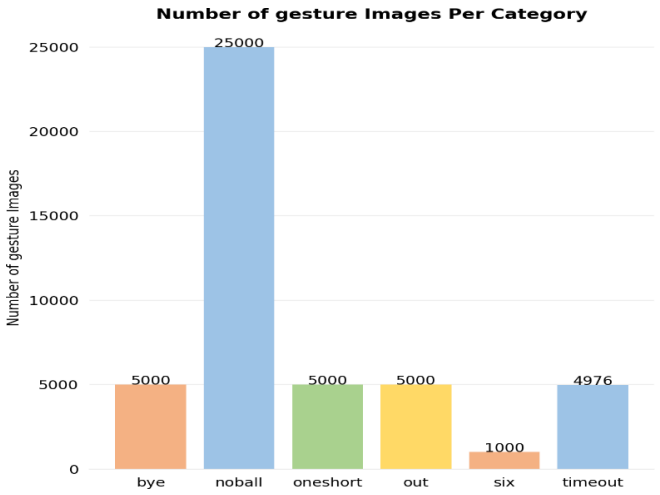
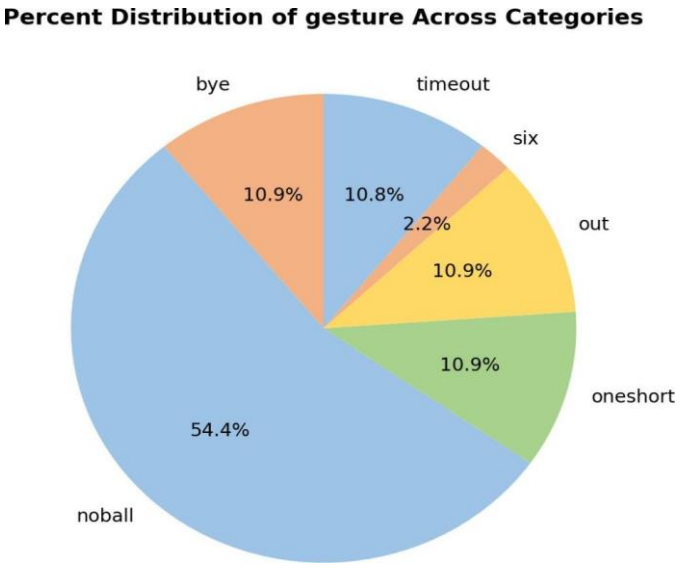


Fig. 2. Data Splitting phase

A bar chart is generated to visualize the distribution of images among subdirectories. It begins by setting up the necessary variables, including x-coordinates and colors for the bars. The code then creates a figure and axes with specific formatting options. The bar chart is plotted using the x-coordinates and the corresponding number of images. The appearance of the plot is adjusted by removing spines, adding grid lines, and customizing the axes. Subdirectory names are displayed as x-axis labels. Text labels representing the number of images are placed on top of each bar. The plot is then adjusted for a tight layout and saved as an image file. Finally, the plot is displayed.



pie chart is generated to visualize the percentage distribution of images across subdirectories. It uses plt.pie() with num_of_images as values, dir_list as labels, colors for slice colors, autopct='%.1f%%' to display percentages, and startangle=90 to set the initial angle. The chart is given a title using plt.title() and saved as "pie_chart.png" with plt.savefig(). Finally, plt.show() displays the chart. This pie chart is representing the distribution of images across subdirectories by percentage.

Fig 5.4: Random images

The random images are displayed, with one image representing each category. It begins by creating a figure with a white background and dimensions of 12x3. The code then iterates through a loop six times to create subplots for each category. Within each iteration, an axes object is created for the current subplot. Randomly, an image path is selected from the corresponding category using the random.choice() function in combination with glob(). The selected image is opened using Image.open() and displayed in grayscale using plt.imshow().
The title of the subplot is set to the category name, and axis ticks and labels are turned off. The resulting figure is saved as an image file named "random_images.png" with a resolution of 200 dpi using plt.savefig(). Finally, the figure is displayed using plt.show().
Next, a random image path is selected from the corresponding category. The random.choice() function is used in combination with glob() to randomly choose an image path from the available images in the category. The glob() function returns a list of file paths that match a specified pattern.

Once the image path is selected, the code uses the Image.open() function to open the image. The image is then displayed in grayscale using plt.imshow(), which is a function from the Matplotlib library that is used to display images.

The title of the subplot is set to the category name using plt.title(). This adds a title above each subplot, indicating the category that the image represents.

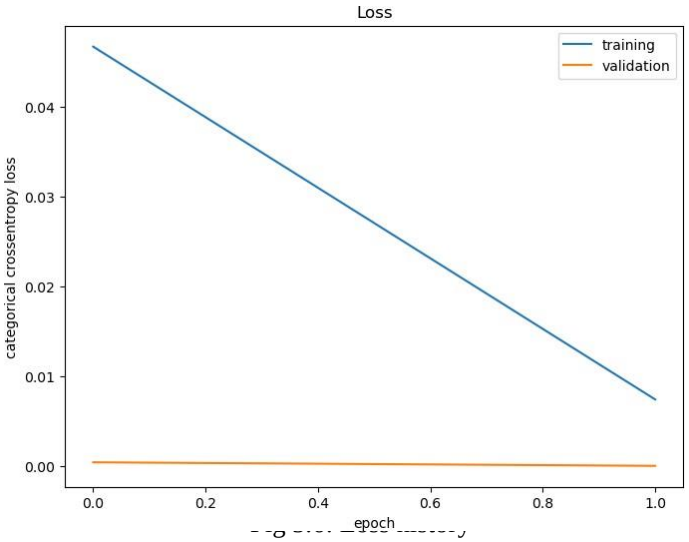
To ensure a clean and minimalistic appearance, the axis ticks and labels are turned off using plt.xticks([]) and plt.yticks([]).

ML Models:

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 densenet121 (Functional)    (None, 3, 3, 1024)        7037504

 flatten (Flatten)           (None, 9216)              0

 batch_normalization (BatchN  (None, 9216)             36864
 ormalization)

 dense (Dense)               (None, 1024)              9438208

 dropout (Dropout)           (None, 1024)              0

 batch_normalization_1 (Batc  (None, 1024)             4096
 hNormalization)

 dense_1 (Dense)             (None, 512)               524800

 dropout_1 (Dropout)         (None, 512)               0

 batch_normalization_2 (Batc  (None, 512)              2048
 hNormalization)

...
Total params: 17,176,390
Trainable params: 10,117,382
Non-trainable params: 7,059,008
_________________________________________________________________
```

Fig 5.5: DenseNet121 model

A modified DenseNet121 model is created using a sequential approach. It adds layers to process the input data, including batch normalization and dropout layers for improved performance and prevention of overfitting. Fully connected layers with non-linear activations are used to learn high-level representations and make predictions. The model incorporates a pre-trained base model, DenseNet121, and the final layer outputs class probabilities. Overall, the modified model benefits from the pre-trained base model and employs various techniques to enhance its performance in image classification tasks. The final layer of the model is a dense layer with a softmax activation function. This layer outputs probabilities for each class, allowing the model to make predictions on the input images. The number of units in this layer corresponds to the number of classes in the classification task.



The graph visualizes the loss history of a model during training. It generates a plot that shows the training loss and validation loss over epochs. The plot helps monitor the model's convergence and generalization performance, providing insights into its learning progress.

By comparing the training and validation loss trends, potential issues like overfitting or underfitting can be identified. Overall, the plot serves as a valuable tool for assessing and optimizing the model's training process.

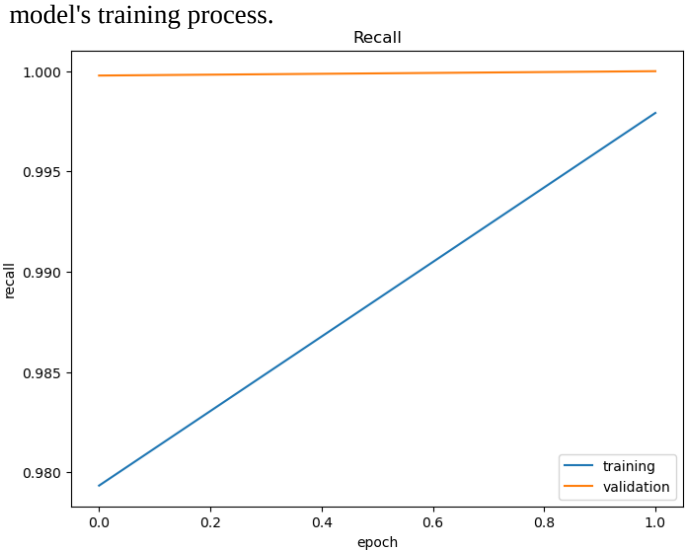


Fig 5.9: Visualizing Recall history

The graph generates a plot visualizing the recall history of a model during training. It shows the training and validation recall over epochs, indicating the model's ability to correctly identify positive samples. The plot helps assess the model's performance in tasks requiring high recall, such as anomaly detection or medical diagnosis.

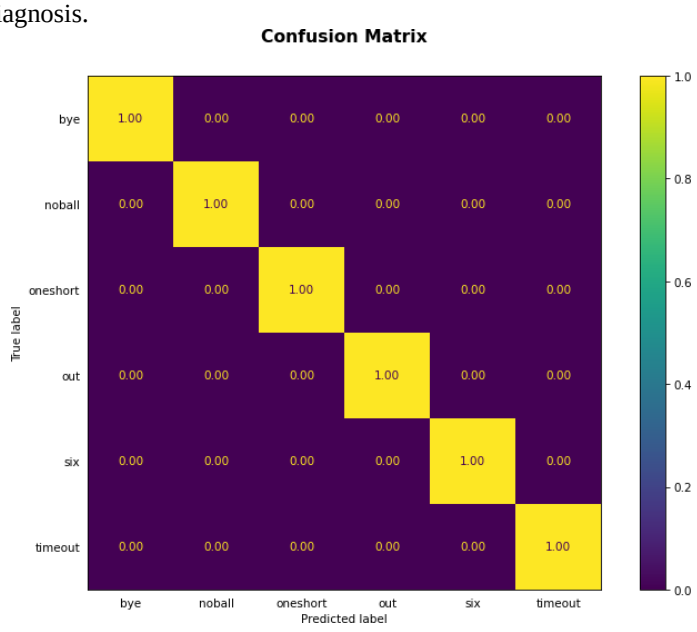


Fig 5.10: Confusion Matrix

The confusion matrix is generated based on true and predicted labels using the ConfusionMatrixDisplay class. The plot displays labels, includes numerical values, and applies normalization. It uses a specified colormap and customizes tick parameters and title. The plot is saved as an image and displayed.

4. Preprocessing Umpire dataset

This module provides functionality for preprocessing a dataset of hand gesture images. It includes functions for reading, converting to grayscale, thresholding, resizing, and saving images. It takes the input and output directory paths as arguments. The module retrieves a list of all thesubdirectories each representing a hand gesture class, from the input directory. It then creates a new directory with the same name in the output directory to store the preprocessed images.

5. Train and test umpire dataset

This module performs various tasks related to image classification. It imports necessary libraries such as os, random, glob, matplotlib.pyplot, preprocessing Image. They also define various variables such as data_dir, train_dir, valid_dir, and test_dir that store the paths to the directories containing the image data.

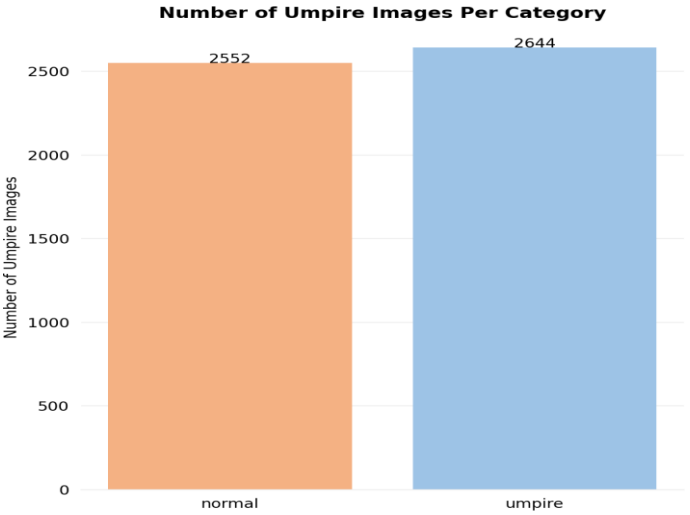


Fig 5.11: Number of umpire images per category

This bar graph represents the distribution of images across two categories: "normal" and "umpire". The category "umpire" is associated with images that depict umpires, while the category "normal" includes all other types of images. The height of each bar in the graph corresponds to the number of images in the respective category.

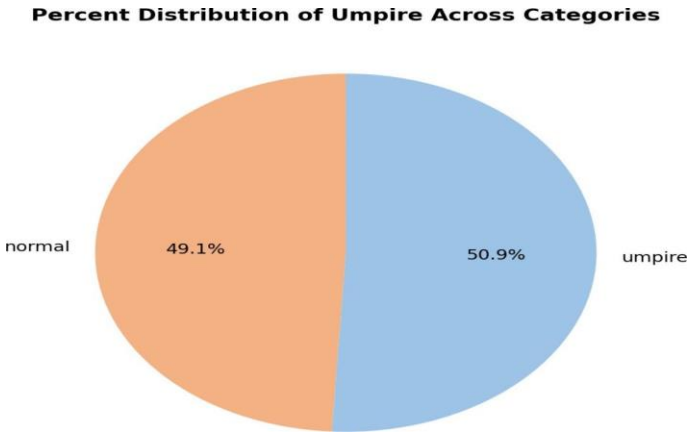


Fig 5.12: Percent Distribution of umpire across categories

The dataset contains two categories of individuals, namely "Normal" and "Umpire". Upon analysis, it was found that 50.9% of the individuals in the dataset were umpires, while the remaining 49.1% were normal individuals such as spectators or coordinators. This distribution can be visually represented through a pie chart, where the umpires' percentage is depicted by one section and the normal individuals' percentage by the other section.

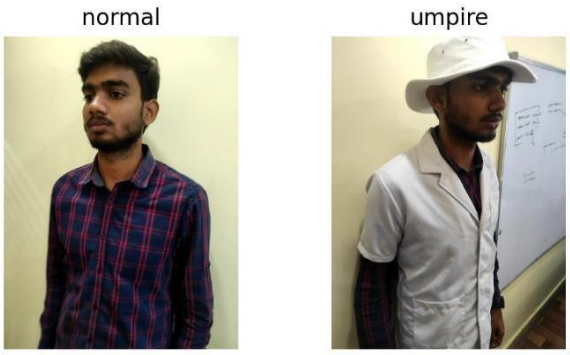


Fig 5.13: Normal and Umpire random image

This figure displays random images from different categories in a given dataset. The figure has a size of 12x3 and a white background. Two directories are selected, and a random image is chosen from each directory using the 'random.choice' and 'glob' functions. Each image is displayed in a subplot with a title that corresponds to the name of the directory, and the axis labels are turned off. The 'imshow' function is used to display the image.The resulting figure provides a sample of images from different categories in the dataset and allows the user to visually inspect them.

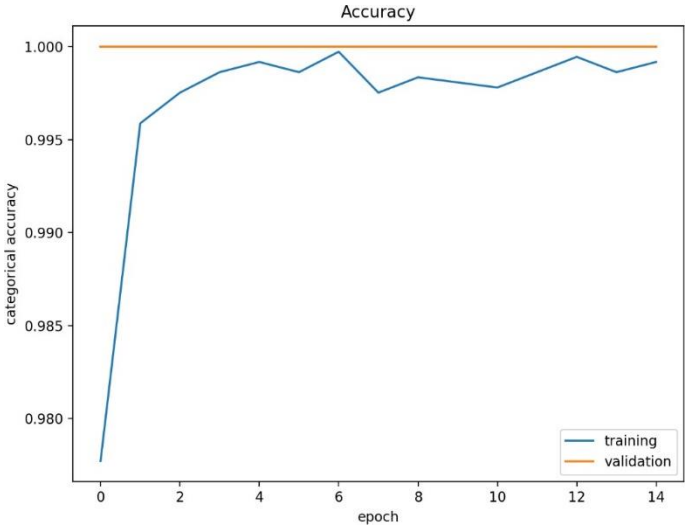


Fig 5.14: Categorical Accuracy

The code snippet generates a line plot that visualizes the accuracy history of a neural network model during training. The plot shows how the training accuracy and validation accuracy change as the number of training epochs increases. The blue line represents the training accuracy, which indicates how well the model performs on the training data. The orange line represents the validation accuracy, which measures the model's performance on a separate validation dataset. . By observing the trends and patterns in the plot, one can gain insights into the model's learning progress and its ability to generalize to unseen data. Deviations or discrepancies between the training and validation accuracy can indicate issues such as overfitting (when the training accuracy is significantly higher than the validation accuracy) or underfitting (when both accuracies are low). Analyzing the accuracy history plot can help in making informed decisions about model training, tuning hyperparameters, or adjusting the training strategy to improve the model's overall performance.

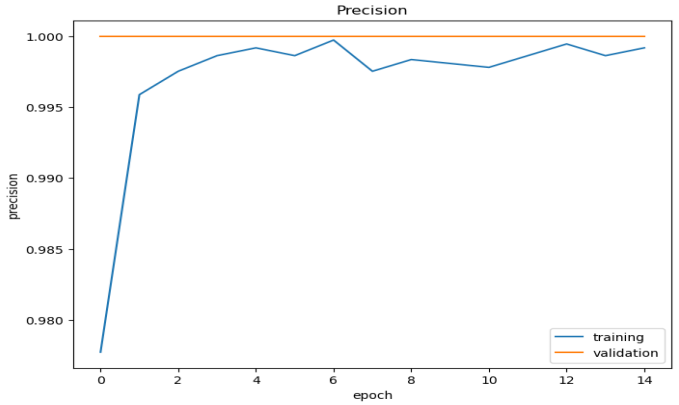
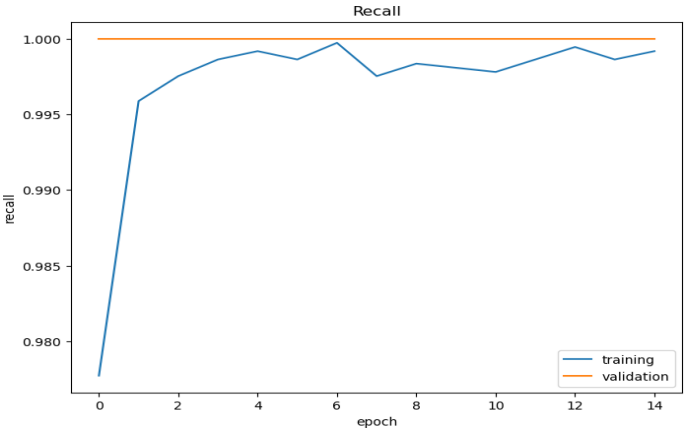
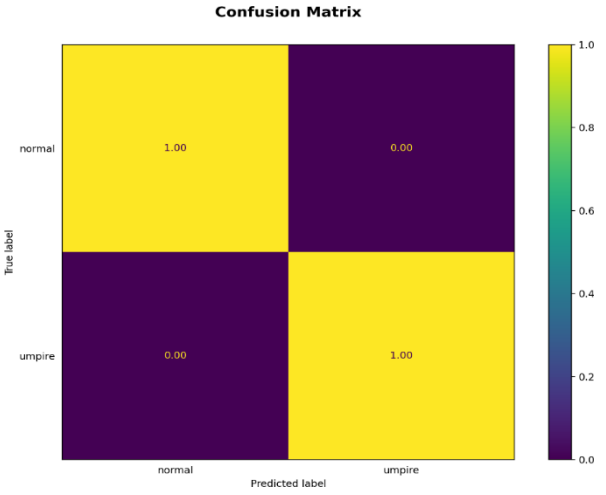


Fig 5.15: Precision

The plotted graph displays the precision history for a neural network model during training. It plots the training precision and validation precision against the number of training epochs. The resulting graph can be used to evaluate the model's performance in terms of precision over time and to compare the performance of different models or hyper parameters.



The plotted graph displays the recall history of a neural network model during training. The resulting plot displays the training and validation recall as a function of the number of training epochs, with the training recall plotted in blue and the validation recall plotted in orange. The plot allows the user to easily visualize the performance of the model over time with respect to recall and identify any issues. This can be useful for comparing the performance of different models or hyper parameters.



This generates a confusion matrix plot to evaluate the performance of a classification model using the true and predicted labels. The resulting grid of squares represents the number of samples that fall into each category. The diagonal of the matrix represents the correct predictions, while off-diagonal entries represent misclassifications. The plot allows the user to visualize the performance of the model and identify areas for improvement.

6. Capturing and detecting Umpire

The image classification module uses a pre-trained DenseNet121 model from the Keras library to predict the class of an input image. The module provides a function called "predict" that takes an image path as input and returns the predicted class label of the image. The function first loads the input image using the Keras utility function, resizes it to (100, 100) pixels, and then converts it to a NumPy array. The array is then normalized by dividing it by 255. The pre- trained model is used to predict the class label of the image, which is returned by the function.

7. Detecting Hand Gesture of Umpire

The provided code implements a live camera application that performs gesture recognition for Indian sign language. The code utilizes the OpenCV library to capture frames from the camera and process them for gesture recognition. The application uses a pre-trained model, specifically a DenseNet121 model, which is loaded from the file 'densenet121'.

The main loop of the application continuously captures frames from the camera and performs the following steps for each frame:

- Preprocessing
- Gesture Prediction
- Score and Wicket Tracking
- Display

8. Data Flow Diagram

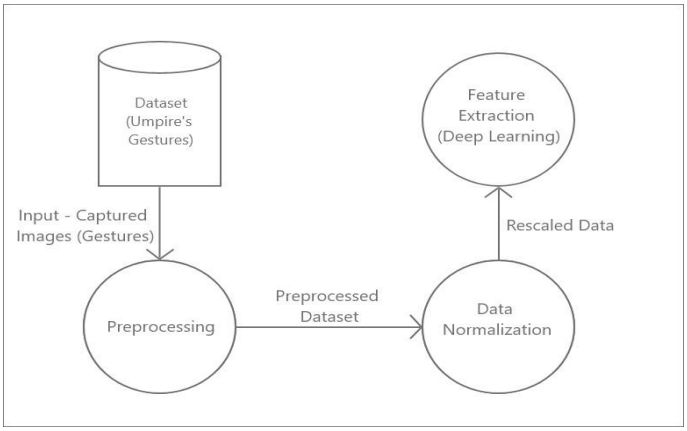
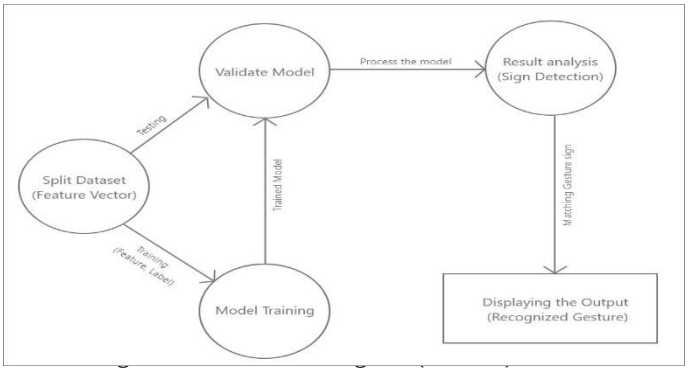


Fig 5.18: Data Flow Diagram (Level 1)

In this fig, A Dataset which has images of the umpire hand signal are Preprocessed using OpenCV and after that data normalization is applied. Once the data is normalized feature extraction technique is used.



In this fig, the dataset is spitted into training and testing. The training dataset is passed to the deep learning algorithm for training. Then the trained model goes for validation where testing data is also sent. Once the model is Validated, it is processed for Detecting the signs. The matched gesture sign which are recognized is then displayed.

9. Use case Diagram

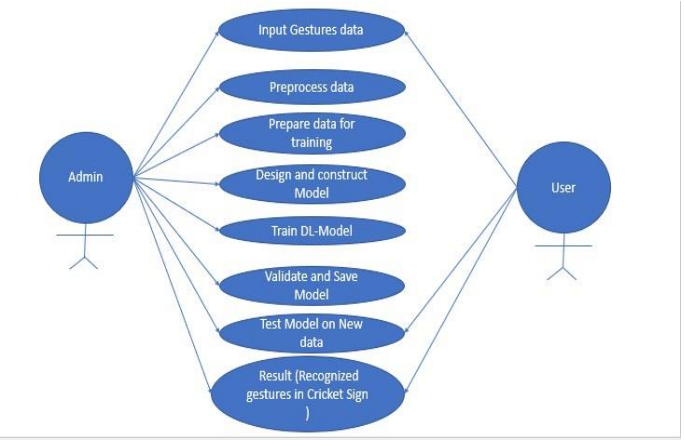


Fig 5.20: Use Case Diagram

This use case diagram illustrates the various tasks that an admin can perform in order to develop and deploy a gesture detection system. The first step is to preprocess the input data using OpenCV and then normalize it for further analysis. The preprocessed data is then subjected to feature extraction techniques in order to obtain relevant information for training the model. Once the data is ready, the admin can design and construct a suitable neural network model and train it using the prepared data. The model is validated using test data to ensure its accuracy and avoid overfitting. If the validation is successful, the admin saves the final model for future use. The system is capable of detecting umpire gestures, and when a user inputs a gesture, the system matches it with the output from the trained model and displays the matched gesture on the screen. This process allows the user to easily identify the detected gesture and take the appropriate action.

10. Sequence Diagram

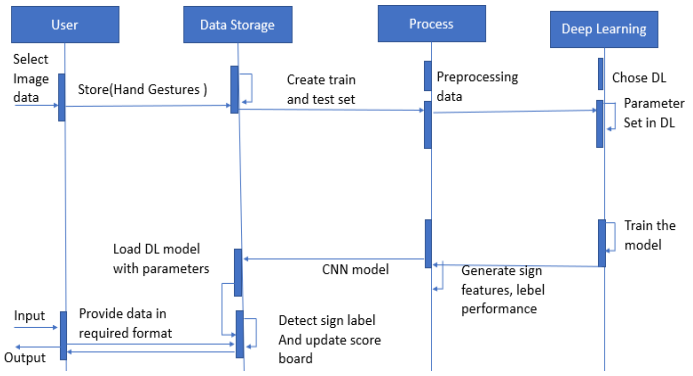


Fig 5.21: Sequence Diagram

Here in this fig, First the user selects the image data for the gesture to be detected. This data is then stored and sent for both training and testing.

Once the data is prepared, preprocessing occurs before deep learning is applied. The admin chooses a deep learning method and sets the required parameters. The model is then trained using the training data. A convolutional neural network (CNN) model is used to detect the sign label, and the score board is updated accordingly.

VI.RESULT

	precision	recall	f1-score	support
0	1.00	1.00	1.00	1000
1	1.00	1.00	1.00	5000
2	1.00	1.00	1.00	1000
3	1.00	1.00	1.00	1000
4	1.00	1.00	1.00	200
5	1.00	1.00	1.00	996

Fig 5.22: Accuracy table

Using the trained model for Gestures and Umpire detection we have got 100% accuracy.

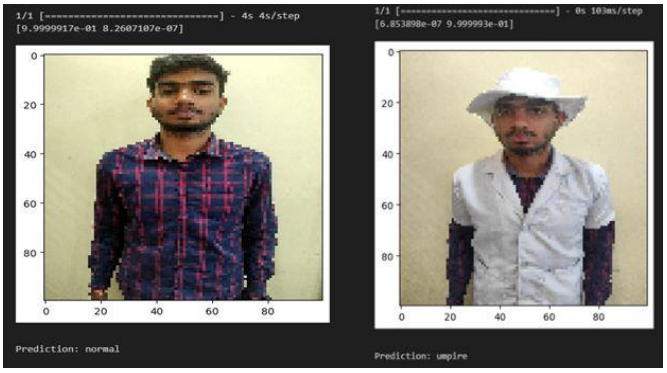


Fig 5.23: Detection of umpire and non-umpire

The above image shows an example of umpire detection and non-umpire detection.

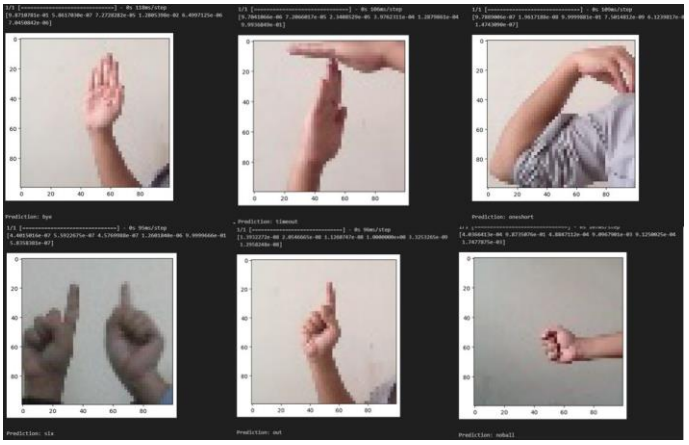


Fig 5.24: Umpire Gestures

the above gestures including Bye, Timeout, Oneshort, Six, Out,No ball have been correctly Detected using our trained model.

VII. CONCLUSION

We surveyed total of 10 Research Papers of different authors that have done projects in this field. We found that we can create dataset ourself using webcam and save in the respective directory of the sign and umpire. The images will then undergo for pre-processing techniques so that better quality of the images can be used for feature extraction. We also found about DenseNet121 Algorithm that can been implemented on the saved model for prediction. Our work will also include data gathering using a web camera to increase the dataset size to more than 50000 RGB images making our prediction more robust. Process of real-time prediction using image frames from a web camera with rates of 50 to 100 Hz is used. Currently the aim of this project is limited to detecting Umpire's gesture. Future Enhancement may include more cameras for detecting the umpire in 360 degrees to improve gesture recognition and unique gestures for actions that do not have gestures can be added to reduce the workload on the review team. With enough funds this project can overtake and detect even the non-gesture based actions like boundary detection for run out, four and wide.

ACKNOWLEDGMENT

We are very grateful to experts for their valuable suggestions to improve this paper.

REFERENCES

[1] M. Fernando and J. Wijayanayaka, "Low cost approach for real time sign language recognition," 2013 IEEE 8th International Conference on Industrial and Information Systems, 2013, pp. 637-642, doi: 10.1109/ICIInfS.2013.6732059.

[2] M. Z. Islam, M. S. Hossain, R. ul Islam and K. Andersson, "Static Hand Gesture Recognition using Convolutional Neural Network with Data Augmentation," 2019 Joint 8th International Conference on Informatics, Electronics & Vision (ICIEV) and 2019 3rd International Conference on Imaging, Vision & Pattern Recognition (icIVPR), 2019, pp. 324-329, doi: 10.1109/ ICIEV .2019.8858563.

[3] E. Kaya and T. Kumbasar, "Hand Gesture Recognition Systems with the Wearable Myo Armband," 2018 6th International Conference on Control Engineering & Information Technology (CEIT), 2018, pp. 1-6, doi: 10.1109/CEIT.2018.8751927.

[4] Lesha Bhansali and Meera Narvekar. Gesture Recognition to Make Umpire Decisions. International Journal of Computer Applications 148(14):26-29, August 2016.

[5] Suvarna Nandyal and Suvarna Laxmikant Kattimani

2021 J. Phys.: Conf. Ser. 2070 012148
[6] Y. Madhuri, G. Anitha. and M. Anburajan., "Vision-based sign language translation device," 2013 International Conference on Information Communication and Embedded Systems (ICICES), 2013, pp. 565-568, doi: 10.1109/ICICES.2013.6508395.
[7] Nusirwan Anwar bin Abdul Rahman, Kit Chong Wei and John See Faculty of Information Technology, Multimedia University.
[8] Moin, A., Zhou, A., Rahimi, A. et al. A wearable biosensing system with in-sensor adaptive machine learning for hand gesture recognition. Nat Electron 4, 54–63 (2021). https://doi.org/10.1038/s41928-020-00510-8
[9] Ravi, H. Venugopal, S. Paul and H. R. Tizhoosh, "A Dataset and Preliminary Results for Umpire Pose Detection Using SVM Classification of Deep Features," 2018 IEEE SymposiumSeries on Computational Intelligence (SSCI), 2018, pp. 1396-1402, doi: 10.1109/SSCI.2018.8628877.
[10] M. A. Shahjalal, Z. Ahmad, R. Rayan and L. Alam, "An approach to automate the scorecard in cricket with computer vision and machine learning," 2017 3rd International Conference on Electrical Information and Communication Technology (EICT), 2017, pp. 1-6, doi: 10.1109/EICT.2017.8275204.