Real-Time Foul Detection in Football Matches Using Machine Learning Techniques

by

Bishal Sadi Siddiqui
18201096
Zeeshan Ahmed Mridul
19101329
Zaki Habib
19201073
Ibrahim Sakib
19201083
Md. Ahmarul Islam Chowdhury
21201829

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science and Engineering

Department of Computer Science and Engineering
School of Data and Sciences
Brac University
January 2024

Declaration

It is hereby declared that

1. The thesis submitted is our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Bishal Sadi Siddiqui

18201096

Zeeshan Ahmed Mridul

19101329

Zaki Habib

19201073

Ibrahim Sakib

19201083

Md. Ahmarul Islam Chowdhury

21201829

Approval

The thesis titled “Real-Time Foul Detection in Football Matches Using Machine Learning Techniques” submitted by

1. Bishal Sadi Siddiqui (18201096)
2. Zeeshan Ahmed Mridul (19101329)
3. Zaki Habib (19201073)
4. Ibrahim Sakib (19201083)
5. Md. Ahmarul Islam Chowdhury (21201829)

Of Fall, 2023 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science and Engineering on January 18, 2024.

Examining Committee:

Supervisor:
(Member)

Amitabha Chakrabarty, PhD

Associate Professor
Department of Computer Science and Engineering
Brac University

Co-supervisor:
(Member)

Sifat Tanvir

Lecturer
Department of Computer Science and Engineering
Brac University

Program Coordinator:
(Member)

Md. Golam Rabiul Alam, PhD

Professor
Department of Computer Science and Engineering
Brac University

Head of Department:
(Chair)

Sadia Hamid Kazi, PhD

Chairperson and Associate Professor
Department of Computer Science and Engineering
Brac University

Abstract

As a widely popular sport worldwide, football necessitates precise and consistent decision-making to uphold fair game-play. It has become essential to automate and optimize certain aspects of the game for fairness and efficiency. Foul detection stands as one of the most challenging and contentious areas where this could be applied. This paper presents an approach for real-time foul detection in football matches using advanced machine-learning techniques. Our research focuses on developing and validating a machine learning-based model that uses video feed data, position coordinates and historical match data to detect fouls in real-time. Faster R-CNN, YOLOv5, YOLOv8 and YOLO-NAS like SOTA machine learning models have been used for this research due to their higher processing speed and accuracy at real time object detection workings. For the detection of foul, machine learning models YOLOv5, YOLOv8, YOLO-NAS and Fast R-CNN have shown an accuracy of about 96%, 97%, 94% and 90% respectively. The potential impact of this system extends beyond football, offering a framework that could be adapted to automate decision-making in various sports, thereby ushering in a new era in sports technology.

Keywords: Real-time Foul Detection, Machine Learning, Football Refereeing, Fair game-play

Acknowledgement

Firstly, we want to start by expressing our deepest gratitude to Allah, the Creator, whose guidance and blessings have helped us on this academic journey.

Secondly, we want to thank our respected supervisor, Dr. Amitabha Chakrabarty Sir, from the bottom of our hearts for all the help, advice, and academic insights he has given us. We'd like to thank our co-supervisor, Sifat Tanvir Sir, for working with us, supporting us, and giving us helpful feedback throughout the study process. We'd also like to thank our research assistant, MD. Fahim-Ul-Islam Sir, for all of his hard work, dedication, and helpful suggestions that made this thesis possible.

Lastly, we want to thank our parents from the bottom of our hearts for their unending love, support, and efforts which have been the foundation of our academic journey.

Table of Contents

Declaration **i**

Approval **ii**

Abstract **iv**

Acknowledgment **v**

Table of Contents **vi**

List of Figures **viii**

List of Tables **x**

1 Introduction **1**
- 1.1 Research Problem 2
- 1.2 Research Objectives 3

2 Related Work **4**
- 2.1 Related Works 4
- 2.2 Summary Table 7

3 Dataset **11**
- 3.1 Dataset Creation 11
- 3.2 Dataset Preparation 11
- 3.3 Dataset Splitting and Formation 14

4 Methodology **17**
- 4.1 Working Plan 17
- 4.2 Model Explanation 18
  - 4.2.1 YOLOv5 19
  - 4.2.2 YOLOv8 20
  - 4.2.3 YOLO-NAS 22
  - 4.2.4 Faster R-CNN 25
  - 4.2.5 ByteTrack 27
  - 4.2.6 BoT-SORT 29
  - 4.2.7 Dark Channel Prior 30

5 Implementation and Results **32**
5.1 Implementation . . . 32
5.2 Results . . . 33
5.3 YOLOv5 Model Results . . . 33
5.4 YOLOv8 Model Results . . . 38
5.5 YOLO-NAS Model Results . . . 42
5.6 Faster R-CNN Model Results . . . 43
5.7 Analysis . . . 46

6 Web Application Deployment **48**
6.1 Web App Work Process . . . 48
6.2 Limitations . . . 50

7 Conclusion and Future Work **52**
7.1 Conclusion . . . 52
7.2 Future Work . . . 52

Bibliography **56**

List of Figures

3.1 Horizontal Augmentation . . . 12
3.2 Shear Augmentation . . . 12
3.3 Brightness Augmentation . . . 13
3.4 Blur Augmentation . . . 13
3.5 Noise Augmentation . . . 14
3.6 Working Dataset after Augmentation . . . 15
3.7 Example of Annotations . . . 16

4.1 Creation Process of Datasets . . . 17
4.2 Flowchart for Basic Working Plan of Real-Time Foul Detection . . . 18
4.3 Network Architecture for YOLOv5 . . . 20
4.4 Architecture of YOLOv8 . . . 22
4.5 Architecture of YOLO-NAS . . . 24
4.6 Architecture of Faster R-CNN . . . 26
4.7 Architecture of ByteTrack Algorithm . . . 28
4.8 Architecture of BoT-SORT Algorithm . . . 30

5.1 mAP_0.5 Graph . . . 34
5.2 Precision Graph . . . 34
5.3 Precision-Confidence Curve . . . 35
5.4 Train Class Loss Graph . . . 35
5.5 Validation Class Loss Curve . . . 36
5.6 Precision-Recall Curve . . . 36
5.7 F1-Confidence Curve . . . 37
5.8 Confusion Matrix . . . 37
5.9 mAP_0.5 Graph . . . 38
5.10 Precision Graph . . . 38
5.11 Precision-Confidence Curve . . . 39
5.12 Train Class Loss Graph . . . 39
5.13 Validation Class Loss Curve . . . 40
5.14 Precision-Recall Curve . . . 40
5.15 F1-Confidence Curve . . . 41
5.16 Confusion Matrix . . . 41
5.17 mAP 50 Graph . . . 42
5.18 Validation Class Loss Graph . . . 42
5.19 Confusion Matrix . . . 43
5.20 AP50 Graph . . . 43
5.21 False Negative Graph . . . 44
5.22 Foreground Class Accuracy Graph . . . 44

5.23 Classification Accuracy Graph 45
5.24 Classification Loss Graph 45
5.25 Total Loss Graph 46
5.26 Examples of Detected Fouls 47

6.1 User Interface of Web Application 49
6.2 Web Application Workflow 49
6.3 Example of Blurred Images 50
6.4 Example of Overlapping Image 50

List of Tables

3.1 Train,Valid and Test set 14
3.2 YOLO format dataset 15
3.3 COCO format dataset 15

5.1 Models Fine Tune 33
5.2 Hardware Specification 33
5.3 Analysis Table 46

Chapter 1

Introduction

As football's global popularity and complexity increases, the need for accurate and consistent decision-making within the sport grows proportionately [7]. The delineation of fair play and foul conduct has long been a contentious area in football, with referees' judgment calls often in the eye of the storm [3]. To alleviate these controversies and enhance the game's integrity, recent research has leveraged machine learning algorithms for real-time foul detection, a domain which this thesis further explores and innovates.

The incorporation of machine learning techniques into sports analytics has gained significant traction in recent years. Predominantly, convolutional neural networks (CNNs) have been employed for image processing in sports, showcasing promising results [8]. Recurrent neural networks (RNNs), particularly long short-term memory (LSTM) units, have demonstrated their capability in processing sequential data, especially in sports where position and time sequence is crucial [29]. This thesis aims to develop a machine learning-based model for realtime foul detection in football matches.

Training and validating this machine learning model was carried out using a custom dataset derived from several football leagues worldwide of all age ranges using their videos converting them into frames and also collecting images from various online image stocks. After creating the dataset we did various preprocessing and augmentation steps using Roboflow. This vast data availability has not only facilitated in-depth model training but has also ensured the generaliz-ability and robustness of the foul detection system. Moreover, YOLOv5, YOLOv8, YOLO-NAS and Faster R-CNN machine learning models primarily have been used for this research due to their higher processing speed and accuracy in the real time object detection sector. After training the models using our custom dataset we found that YOLOv5 and YOLOv8 yielded best accuracies which were 96.6% and 97.2% respectively. After finding the best models we used these models trained weights for our custom dataset which were later used at web application deployment. This web application can be used for real-time foul detection in football matches, as this has various sources available to input data like images, videos, Real Time Stream Protocol(RTSP) and YouTube links. Also multi-object trackers like ByteTrack and BotSORT are used as well to track involved players and fouls in multiple frames. We also implemented the Dark Channel Prior algorithm for Dehazing images or videos as because of weather conditions it can be hard for models to predict foul at poor visibility. Thus our web application can be real handy at any weather conditions.

Our research contributes to the academic discourse on the application of machine learning in sports analytics and the broader scope of artificial intelligence in decision-making automation. Further, the proposed model's real-world implications could be profound, potentially revolutionizing football refereeing, and reducing human bias and error.

Finally, the thesis addresses some of the ethical and societal implications of the proposed technology. As automation continues to pervade all sectors of society, it is crucial to evaluate and regulate its implications to ensure fairness and mitigate unintended consequences.

1.1 Research Problem

The sport of football, due to its dynamic and unpredictable nature, brings forth an intricate blend of strategies, skills, and actions [19]. One of the enduring challenges in managing football games lies in identifying and adjudicating fouls. Despite the professional acumen of referees, the fast-paced nature of football and the increasing complexity of player interactions often makes the detection of fouls an arduous task. Consequently, the inconsistent application of rules, stemming from potential human bias or error, could affect the game's fairness and outcomes [9].

Contemporary football leagues have partially mitigated this issue by implementing technologies such as the Video Assistant Referee (VAR) system. However, VAR has been critiqued for its inability to provide real-time decisions, thus causing significant game interruptions, and detracting from the natural flow of the match [24]. This identifies the exigency for an efficient, real-time system capable of detecting fouls with high accuracy, thereby reducing reliance on subjective human judgment.

Machine learning, especially deep learning techniques like Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) with Long Short-Term Memory (LSTM) units, have made significant inroads into sports analytics, particularly in tasks such as player tracking, game pattern recognition, and injury prediction [18]. However, their utilization for real-time foul detection in football is still in the nascent stages, and the existing models demonstrate limitations.

This research aims to address these challenges by developing a more comprehensive and robust machine learning model for real-time foul detection in football. The model intends to leverage the strengths of CNNs for video data analysis, thereby creating a synergy of these advanced machine learning techniques. Furthermore, the model will be trained on a diverse custom dataset spanning multiple leagues worldwide, enhancing its gener-alizability and real-world applicability.

The problem, therefore, lies in the development of this efficient, real-time machine learning model for foul detection in football matches, capable of overcoming the limitations of current approaches, thereby potentially enhancing the game's fairness, reducing controversies, and improving the viewing experience.

1.2 Research Objectives

The research objective of this study is to provide accurate and instant information about the fouls occurring on the football field. The images taken from the camera will be analyzed with the already existing datasets made by us to see if a foul has been committed and will notify the authority about the following results. There are several objectives which are given below.

1. To analyze the potential of machine learning in case of accuracy.
2. To collect and create a comprehensive dataset of football fouls.
3. To develop a model which will work better with custom dataset in case of foul detection programs.
4. To experiment how fast the research process can comply in the case of delivering results for foul detection.
5. To analyze the already made models and how to further improve those to satisfy the current needs.
6. To examine the whole working flow for our current models.
7. To offer further suggestions and recommendations on the following research.

Chapter 2

Related Work

The integration of artificial intelligence (AI) in football foul detection has the potential to bring forth numerous benefits and can hold great significance for the sport. AI technologies, such as machine learning and computer vision, offer the potential to revolutionize the accuracy, consistency, and fairness of foul detection in football matches. By automating the process and reducing reliance on subjective human judgment, AI-based systems can provide objective and real-time decision-making capabilities.

There are similarities in between the detection of abnormalities in surveillance camera footage and foul detection in the football match. For example, Computer Vision Techniques, Object Detection and Tracking, Anomaly Detection, Real-Time Processing and Training Data and Model Development. The similarities have the potential for the further research development.

2.1 Related Works

In this part, previous relevant work has been taken into consideration for the analysis of real time foul detection in the football field which focuses mainly on the usage of camera, frame detection and comparison to identify any abnormalities. As the rate of occurrence of abnormality is low so the collection of datasets is one of the challenging tasks. Several algorithms from different research papers seems to be relevant for the analysis and detection of foul in real time match.

Principal Component Analysis (PCA) and Support Vector Machine (SVM) are used to classify significant behaviors. The research paper [1], shows real-time human abnormal behavior detection. It uses the border information of consecutive blobs to select features and then applies PCA to extract relevant information and SVM to classify normal or abnormal behavior. Consequently, optical flow is used to determine velocity of pixels and detect abnormal behavior based on the obtained information.

In the research paper [17], a feature descriptor called Histograms of Optical Flow Orientation (HOFO) has been introduced, to encode the movement information of the region of interest in a video frame. The HOFO feature is computed using the Horn-Schunck algorithm to extract optical flow from consecutive frames. The Hidden Markov Model (HMM) based classifier is used to distinguish between normal events and abnormal events.

In this paper [5], real-time hand gesture recognition is covered where their main components are hand detection, hand gesture tracking and hand gesture recognition. The hand detection part system identifies skin pixels based on color tone information and then detects moving objects to accurately detect hand regions. Then, at hand gesture tracking it tracks the trajectory of the hand by analyzing the positions of the hand centers overtime and analyzing the positions of the last 10 frames by storing positions. Finally, at hand gesture recognition, the system obtains information from hand gesture tracking and compares those with pre-defined hand gestures.

In this paper [11], suspicious behavior detection of people by monitoring cameras is represented. They analyze the trajectory of moving objects based on their motion vectors. By calculating the motion vector, they determine the speed and direction of movement to detect suspicious behavior. Once suspicious behavior is detected the system segments the object of interest from the background and tracks it within the camera's field of view to detect suspicious behavior more accurately.

In this research paper [28], there have been use of target detection and tracking methods. First, comparison between given datasets and the target is identified. With the help of target detection, advanced video analysis technology is applied. Then different block background modeling such as random block background modeling (RBBM) or mixed block background modeling (MBBM) is used with the combination of adaptive block propagation background subtraction method (ABPBSM) for foreground detection. Here evaluation matrices are also being used to assess the performance of the ensemble model.

In the research paper [33], for object detection specially the players, a tool of object detector called YoloV3 is used. Accuracy has been on focus for the detection which sacrificed some speed. For tracking all the players on the field, a positional based tracker is used. The researcher also trained the CNN to track and detect Numbers on the jersey for accuracy. Also an outlier detection system is used to detect and make the referee unique from the other teams using DBSCAN.

In this paper [10], a convolution neural network (CNN) method to recognize hand gestures of human task activities from a camera image has been proposed. In order to deal with light related complexities, a Gaussian Mixture model (GMM) has been adopted. The task of such a type of model is to train the skin model which is used to filter out non-skin colors of an image. The proposed system also had the satisfactory results on the transitive gestures in a continuous motion using the proposed rules.

The idea of handling the problem based on correlation analysis of the optical flow has been proposed in this paper [21] for accurate and quick identification of abnormal behavior of a crowd. The related events are divided into categories such as accidents, crowd density, crowd egress behavior and others. The modeling of crowd movements has been done using the optical overflow technique. The hierarchical agglomerative clustering algorithm, is used for understanding the motion pattern of crowded circumstances. For the analysis of human crowd behavior, graph modeling and matching based on Delaunay triangulation have been proposed. The procedures employed in this case involve gathering video from cctv cameras and determining if the anomalous criteria have been met or not. If the answer is affirmative, the event is either detected or it is not.

Data collection from raw films or events is quite difficult because the frequency of odd happenings is relatively low. In this paper [2], unusual event detection has been predicted. Using Bayesian adaptation models, unexpected models are derived from the common models and then compared for detection. Because they perform well in unsupervised learning, HMMs are used for temporal dependencies. The results were satisfactorily produced using audiovisual features and comparisons to supervised and unsupervised baseline systems.

The research paper [4], focuses on the automatic detection of suspicious and anomalous behavior by surveillance systems. Abnormal events are unpredictable due to their timing, location, and circumstances of occurrence. Firstly, normal model has been introduced, then comparison with the suspicious or changed ones has been compared with normal ones. Thus, learning model like FFNN (Feed Forward Neural Network) is used. The hardware components that are employed are the Xetal IC30 processor, the 8051 microcontrollers (for external connectivity and high-level image processing), DPRAM and Wi-ca Smart camera (real-time behavior analysis of high-level moving objects). Due to the limitation of IC3D memory capabilities, the system can only follow one route at a time and be used on one object, but it is still capable of detecting irregularities.

Utilization of a completely data-driven approach using deep learning for the prediction of foul situations in real-time has been discussed in this paper [12]. Despite achieving significant success, their model faced limitations in terms of scalability and real-world applicability, as it relied on detailed player biometric data that may not always be available or ethically permissible to use in real scenarios.

In this paper [20], the "Pose-guided RCNN" framework is created by replacing a standard Faster R-CNN with a 3-class RPN, extending it with additional key-point branches, and adding human pose supervision. The dataset is taken from football matches taken by camera. The following three insights are applied to this performance: The implementation of pose-guided localization network, which can enforce proposal refinement for jersey number location by human stance, the redesign of the three-class RPN for anchor association, and the universality of the region-based CNN model are the first two. The proposed strategy can be simply applied to different sports and is end-to-end trainable by combining the three elements.

In the research paper [27], YOLOv4 and deep sort is used to identify the object tracking in case of camera movement. The dataset used is INRIA person dataset for training YOLOv4. Market 1501 and Mars dataset used to train deepsort. It has been seen that the object tracking problem is solved well by the model. Deepsort and YOLOv4 both models have successfully identified the human body.

To sum up, it can be said that all the procedures and techniques used in the above mentioned articles and paper are relevant for the practical impliacations. So that, the work for the implimentations can be enhanced. the real-time foul detection can be managed more swiftly and smoothly.

2.2 Summary Table

Sl. No	**Name of the Work**	**Algorithms/Models Used**	**Dataset**	**Accuracy**
1	A Detection System for Human Abnormal Behavior [1]	Principal component analysis, Support vector machine, Optical flow	Not mentioned	Not mentioned
2	Abnormal event detection based on analysis of movement information of video sequence [17]	Histograms of optical flow orientation, Horn-schunck, Hidden markov model	UMN, PETS	91.39% to 97.24% to 100%
3	Hand Gesture Recognition System [5]	Hand gesture tracking using positions of the hand center	Not mentioned	94%
4	Feature Extraction of Foul Action of Football Players Based on Machine Vision [32]	Threshold recognition algorithm using Harris 3D operator, AdaBoost, Clustering and Fusion	Not mentioned	Higher (Percentage not mentioned)
5	Suspicious Behavior Detection of People by Monitoring Camera [11]	Analyzing trajectory of moving objects based on the motion vector	Not mentioned (CAVIAR /Real image sequence used)	Percentage not mentioned
6	Development of an algorithm for abnormal human behavior detection in intelligent video surveillance system [30]	Convolutional neural networks, Data processing	Not mentioned (Can be static and dynamic hand gestures dataset)	91.31%

Continued on next page

(Continued)

7	Design and Research on the Foul Detector of a Long Jump Jumping Line Based on a Vision Sensor [31]	Transmitter Principle, Receiver Principle, Voltage Regulator Circuit, Sound and Light Alarm Circuit	Not mentioned	Not mentioned
8	Football Player Posture Detection Method Combining Foreground Detection and Neural Networks [28]	Pixel Classification, DetectNET	MPII, MSCOCO, LSP, DPoseNet	Not mentioned
9	Fish Detection Using Deep Learning [22]	CNN Architecture, Dropout Algorithm, YOLO	ImageNET	Accurate for application(Not mentioned directly)
10	Football Games Analysis from video stream with Machine Learning [33]	YOLO, CNN, DBSCAN	COCO, Street View House-Numbers (SVHN)	Not Fully Covered, still have flaws
11	Hand Gesture Recognition using Image Processing and Feature Extraction Techniques [25]	Support Vector Machine, K-Nearest Neighbors, Random Forest	ASL[1], ASL + Digits [18], Mobile-ASL [25], ASL (Proposed Approach)	92.25% to 95.81% to 96.96%
12	Real-Time Hand Gesture Detection and Recognition Using Bag-of-Features and Support Vector Machine Techniques [6]	Support Vector Machine	Not available publicly	96.23%

Continued on next page

(Continued)

13	Hand Gesture Detection with Convolutional Neural Networks [14]	Convolutional neural network	Manually collected and not available publicly	in localization (96%–100%) and in recognition (99.45%)
14	Abnormal Behavior Recognition Based on Key Points of Human Skeleton [23]	Single-person human pose estimation (HPE), Convolutional neural network	Manually created and not available publicly	89%
15	Human Hand Gesture Recognition Using a Convolution Neural Network [10]	Convolution neural network (CNN), Gaussian Mixture model (GMM)	Manually created and not available publicly	95.96%
16	Detecting anomalous crowd behavior using correlation analysis of optical flow [21]	Pure optical flow, Social force model, Graph modeling and matching, Correlation analysis of optical flow	UMN, PETS	78% to 85% to 91% to 97.05%
17	Semi-supervised Adapted HMMs for Unusual Event Detection [2]	Semi Supervised adapted Hidden Markov Model (HMM)	Unusual datasets created using the model then compared	Not mentioned
18	Abnormal Motion Detection in a Real-Time Smart Camera System [4]	Feed Forward Neural Network (FFNN)	Not mentioned	After 500 iterations error of 3.34%
19	Wide Open Spaces: A statistical technique for measuring space creation in professional soccer [15]	The Voronoi tessellation	Not mentioned	Not mentioned
20	Big Data, Artificial Intelligence and Quantum Computing in Sports [26]	miCoach System, Neural network modeling	Not mentioned	Not mentioned

Continued on next page

(Continued)

21	Big data and tactical analysis in elite soccer: future challenges and opportunities for sports science [12]	Neural network modeling, the team centroid method, expectation maximization algorithm with position data	Not mentioned	Not mentioned
22	Exploring Generalization in Deep Learning [13]	Lipschitz Continuity and Robustness, PACBayesian Analysis	CIFAR 10 dataset	Obtains close to Zero Training Error
23	Small sided games in soccer-a systematic review [16]	SSG model	Author used two group of people to test the result	93.12%
24	Pose-Guided R-CNN for Jersey Number Recognition in Sports [20]	R-CNN	Custom dataset	94.09%
25	A Sport Athlete Object Tracking Based on Deep Sort and Yolo V4 in Case of Camera Movement [27]	YOLOv4, Deep Sort	INRIA Person, Market1501 and MARS	97.67%

Chapter 3

Dataset

3.1 Dataset Creation

As Real-time Football Foul Detection is a totally rare concept and we have to detect abnormal or suspicious behaviors of players so that we can define foul type, that's why we had to make our dataset from scratch. Firstly, we have collected some videos of different types of foul from the internet. Then, we collected frames from those videos using opencv python, and we also collected images of different types of foul or unfair tackle images from various online image stock resources. After that, using those images or frames we have done annotation and labeling of different types of fouls like tripping, sliding tackle, pushing etc, and players who were involved in that foul using the bounding box with Computer Vision Annotation Tool(CVAT). Then using Roboflow we generalized all foul types into foul only, thus we have mainly 3 classes which are foul, fouling_player and victim.

3.2 Dataset Preparation

To prepare our dataset for our model we utilized Roboflow's built in pre-processing and augmentation technology. Firstly, we pre-processed our frames using auto-orientation and resizing into the same size at 640 resolution so that no frames were exported into different angles or sizes which might affect the accuracy of the model. After preprocessing our scratch made custom dataset, we did data-augmentation to get different perspectives of the same frames which will help our models to prevent any kind of overfitting or under-fitting while detecting foul using the dataset. We did augmentation at random images to help the model generalize for unseen images. To augment our custom dataset, firstly, we flipped images from our custom dataset for a different perspective view.

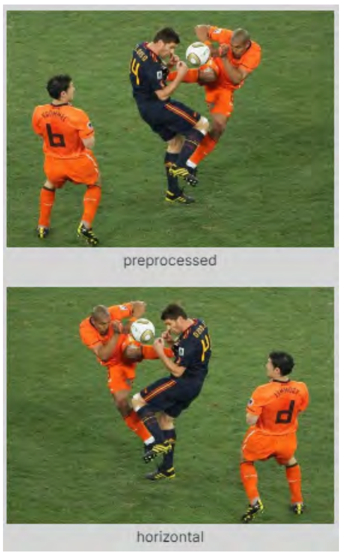


Figure 3.1: Horizontal Augmentation

Then we did shear augmentation by tilting the images ±8°Horizontal and ±8°Vertical to simulate views in different angles.

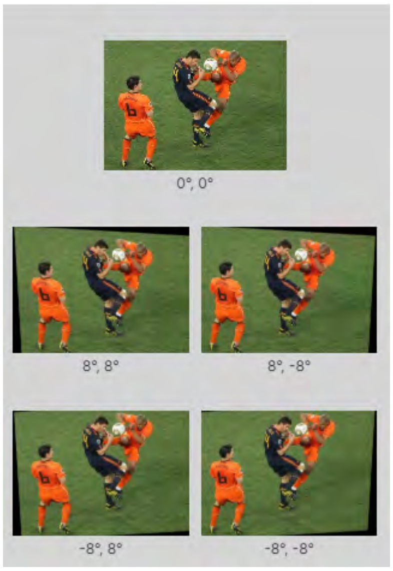


Figure 3.2: Shear Augmentation

Then our next augmentation step was changing brightness to -10% Dark and +10% Bright to help in different lighting conditions.

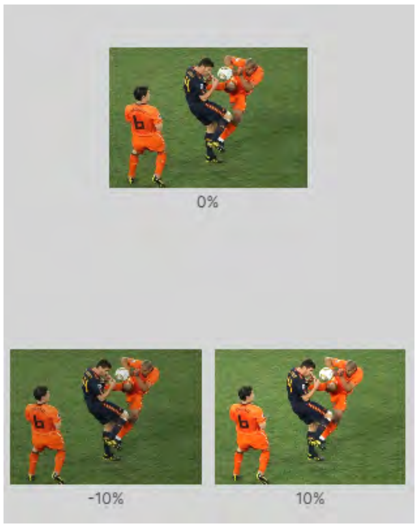


Figure 3.3: Brightness Augmentation

After that we blurred our dataset images up to 0.5px to help model when predicting at blurry images.

Figure 3.4: Blur Augmentation

Then finally we added noises into our custom dataset images up to 0.5% of pixels. We did blur and noise augmentation a small percentage because after testing our models we saw that blurring or adding noises more than that was affecting model precision and accuracy of prediction, so using blur up to 0.5px and noise up to 0.5%px was good enough to help model predicting foul in different low resolution or blurry images or videos scenario and affecting at accuracy was negligible as well.

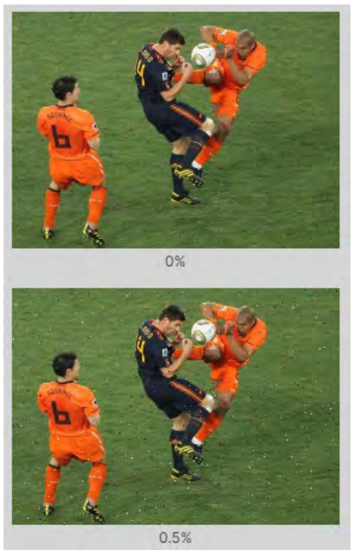


Figure 3.5: Noise Augmentation

Examples of Horizontal, Shear, Brightness, Blur and Noise Augmentations in our custom dataset are shown at Figure 3.1, 3.2, 3.3, 3.4 and 3.5 respectively. This augmentation was applied into the training set only as underfitting or overfitting happens while training only so the model would need different perspectives at training only.

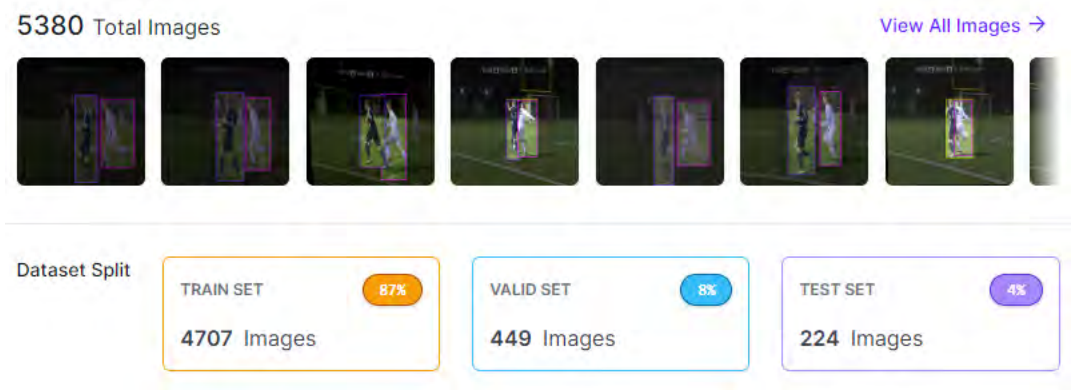
3.3 Dataset Splitting and Formation

Initially we gathered and annotated a total of 2497 frames. Then after preprocessing we had to delete some of the annotations which were not good enough or blur and affecting accuracy, thus our number of annotations which were working perfectly turned into 2242 frames.Then, we splitted our working dataset randomly into 3 parts: Training, Validation and Testing set. Dataset split percentage and number of frames are given at below Table 3.1.

Subset Name	**Percentage**	**Number of Frames**
Training set	70%	1569
Validation set	20%	449
Testing set	10%	224

Table 3.1: Train,Valid and Test set

After data augmentation in the training set only, it turned into a total of 5380 images for our whole working dataset. At Figure 3.6, we can visualize the snippet of our custom dataset from Roboflow after augmentation.


5380 Total Images

View All Images →

Dataset Split

TRAIN SET 87%
4707 Images

VALID SET 8%
449 Images

TEST SET 4%
224 Images

Figure 3.6: Working Dataset after Augmentation

We exported our dataset into two formats which are YOLO and COCO, as for the YOLOv5, YOLOv8 and YOLO-NAS model we used the YOLO format dataset and for the Fast R-CNN we used COCO format dataset. At YOLO format every frame has their corresponding text folder which contains coordinates of bounding boxes. Example has been given at below Table 3.2.

Class Index	**Center X**	**Center Y**	**Width**	**Height**
0	0.65125	0.66666	0.0425	0.11666
1	0.68	0.66888	0.03437	0.12444
2	0.66375	0.66888	0.06687	0.12444

Table 3.2: YOLO format dataset

On the other hand in COCO format per train, valid and test sets have only 1 json file which contains annotation coordinates of whole sets frames. Examples related to this have been given at below Table 3.3.

Image ID	**Class Index**	**X**	**Y**	**Width**	**Height**	**Area**
1	1	823	14	885.71	1001.43	886976.565
2	1	563	74	307.14	752.86	231233.42
2	2	763	83	654.29	651.43	426224.135

Table 3.3: COCO format dataset

At total we have annotated around 2500 images with more than 6500 annotations using 3 classes or labels foul, fouling player and victim using bounding box to train model in foul features and involved players behaviors in the scenario. Some examples of our annotated images are shown below at Figure 3.7 from our workings.

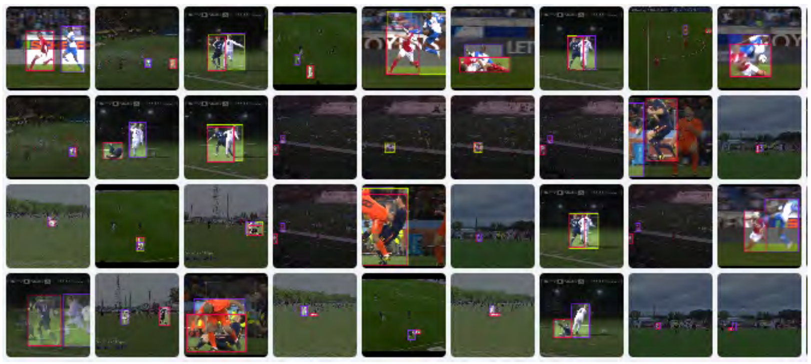


Figure 3.7: Example of Annotations

Thus, using our custom dataset with the help of proper models and algorithms system will be able to detect involved players, and their locations on the field, the system will analyze both fouling and victim players body gestures, compare them with pre-defined gestures, and determine whether they indicate a foul or normal behavior while in collision.

Chapter 4

Methodology

4.1 Working Plan

In our real-time football foul detection , we have first collected different video footage of fouls which were converted into frames and images from various online image stock resources. Then labeled and annotated those frames and images into foul and players who were involved into it. And by doing this after some pre-processing and augmentation we got the needed dataset to train our foul detection model.

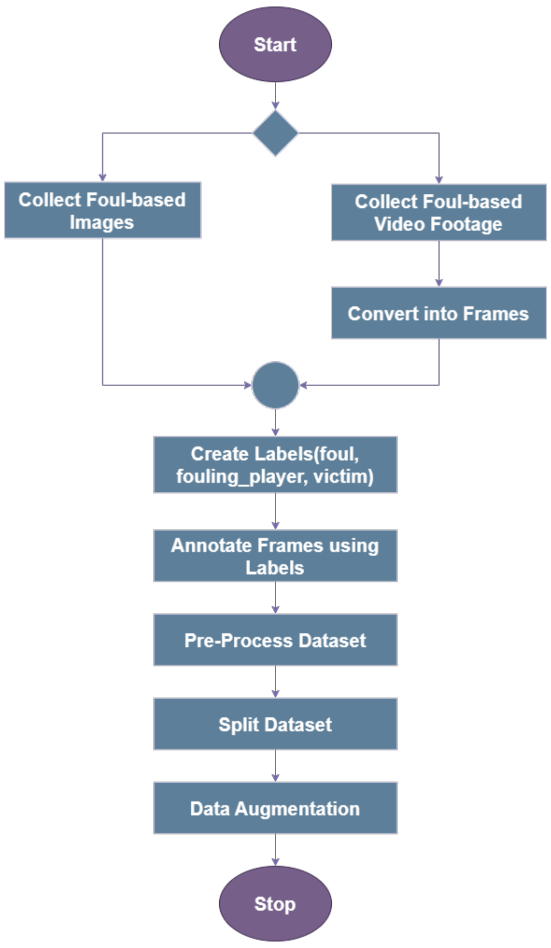

Start
Collect Foul-based Images
Collect Foul-based Video Footage
Convert into Frames
Create Labels(foul, fouling_player, victim)
Annotate Frames using Labels
Pre-Process Dataset
Split Dataset
Data Augmentation
Stop

Figure 4.1: Creation Process of Datasets

With the augmented train set, the model will be trained. For the model we selected Fast R-CNN, YOLOv5, YOLOv8 and YOLO-NAS, so we exported their own supported dataset format to train. After training, we have used our testing set to test that our model is actually detecting foul or not using our custom dataset.

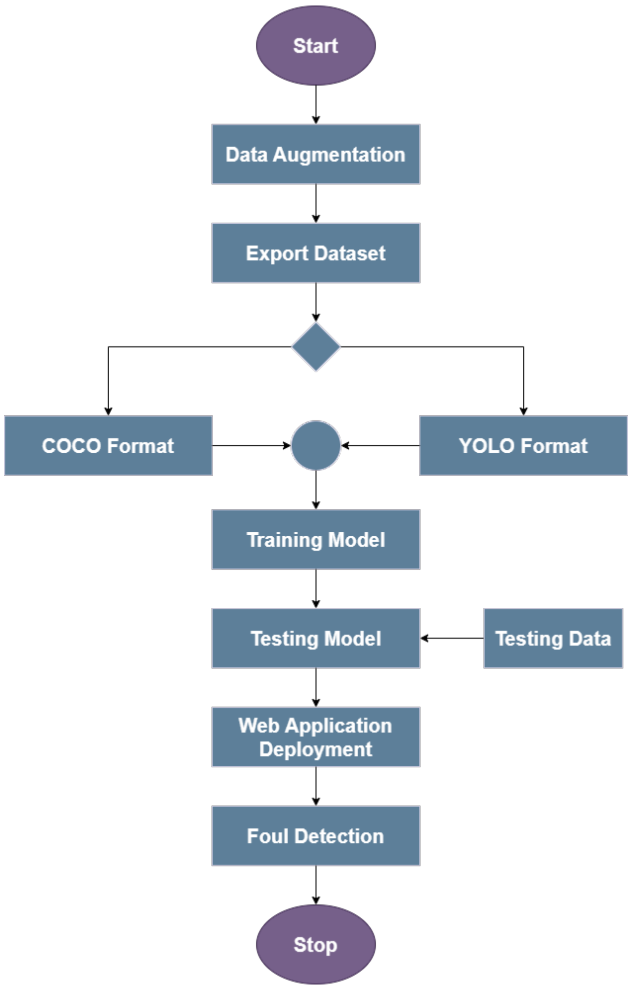


Figure 4.2: Flowchart for Basic Working Plan of Real-Time Foul Detection

After testing our model with proper tuning we deployed our model as a web application which has various functionalities when detecting foul from football matches. This whole working flow can be seen at Figure 4.1 and 4.2.

4.2 Model Explanation

YOLOv5, Fast R-CNN, YOLOv8, and YOLONas are the models that have been used to detect foul and analyze the custom dataset for better precision and accuracy, including data training.

4.2.1 YOLOv5

YOLOv5 is one of the most used models for object detection from the computer vision models available. It was introduced in four versions where each offers different accuracy which are mostly high. These versions take different amounts of time to get trained. The excellence of YOLOv5 is found in its capacity to strike a balance between speed and precision, which makes it a flexible option for a range of applications. Moreover, YOLOv5 keeps its real-time capabilities, which makes it appropriate for applications where minimal latency is essential. Because of its effective implementation, it can operate on a variety of hardware platforms, such as GPUs and TPUs, which makes it accessible and flexible in a range of computing settings. YOLOv5 is more advanced than YOLO because it generates anchor boxes which are more dynamic. Allowing the bounding boxes to be more closely aligned with the objects. The spatial pyramid pooling(SPP) is a type of pooling layer which is used to reduce the spatial radiation of the feature maps. SPP performs better in identifying small objects since it allows to view the objects in various different scales. The previous version YOLOv4 also used the SPP but the YOLOv5 includes several improvements so that the object detection of the small objects have become more advanced to achieve better results. This model uses labeled images as a dataset with different classes. Then will use these datasets to train the model for which google colab is used. In this case we collected the frames from the selected videos for each fouls with labeling for the dataset. The free graphic process unit and tensor process unit is supported by google for research and learning about artificial intelligence. The YOLOv5 has increased its speed and accuracy compared to the YOLOv4. With the built-in pytorch framework that is user friendly and has a larger community than the darknet framework. Thus, YOLOv5 is able to attain competitive outcomes on benchmark datasets because of the enhanced detection accuracy brought about by the architecture optimizations and increased model complexity. In general, YOLOv5 is a noteworthy progression in real-time object identification, providing an attractive blend of precision, velocity, and adaptability.

Architecture of Yolov5:

YOLOv5 architecture is made of three main parts which are Backbone, Neck and Head. At Figure 4.4 we can see that CSP-Darknet53 is used as the backbone for YOLOv5. CSP-Darknet53 is used as the backbone for YOLOv5. CSP-Darknet53 is just a convolutional neural network and works as backbone for object detection which uses Darknet53 architecture. This Darknet53 architecture was used as the backbone for previous versions as well. Main structure backbone is multiple CBS modules stacking which stands for Convolution Layer, Batch Normalization and Sigmoid Weighted Liner Unit, and then at end one SPPF module. Here CBS module is used to assist in feature extraction for C3 module and SPPF module just enhances feature extraction ability. PANet or Path Aggregation Network is used as the neck to get the feature pyramid and upsample it which helps to identify the same target in different sizes and scales. For head it uses the same as YOLOv3 Head which works to generate the final output. It is composed from three convolution layers which predicts location of bounding boxes, scores and object classes. Equations to compute coordinates for the target bounding boxes are given below.

$$b_x = (2 \cdot a(t_x) - 0.5) + c_x$$
$$b_y = (2 \cdot a(t_y) - 0.5) + c_y$$
$$b_w = p_w(2 \cdot a(t_w))^2$$
$$b_h = p_h(2 \cdot a(t_h))^2$$

Here c_x and c_y are coordinates of unadjusted predicted center point, b_x, b_y, b_w and b_h are coordinates of adjusted prediction box, p_w and p_h are information of the prior anchor, t_x and t_y are to represent offsets calculated by the training model.

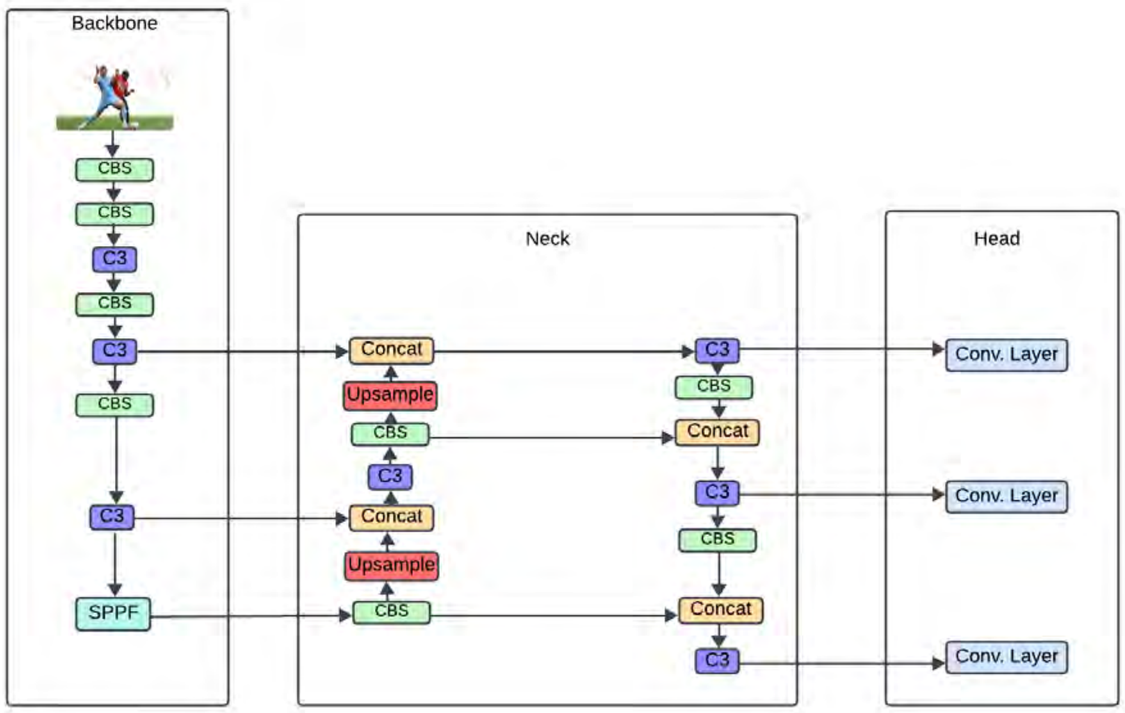


Figure 4.3: Network Architecture for YOLOv5

4.2.2 YOLOv8

The most recent model in the Computer Vision discipline to be offered in the YOLO algorithm series is YOLOv8. With an easy-to-implement architecture, it offers cutting-edge results for image or video analytics. YOLO v8 is simple to use and create because of its CLI-based implementation and Python package. It offers a unified framework for training models to conduct object identification, instance segmentation, and image classification; it is faster and more accurate than YOLOv5 and other YOLO models. In order to increase performance and versatility, YOLOv8 adds new features and enhancements to build on the success of its predecessors. To improve its object identification skills, YOLOv8 adds new features. The model can dynamically focus on different areas of a picture by incorporating a self-attention mechanism into the network's head. This allows the model to modify the significance of characteristics according to how relevant they are to the job at hand. The model's capacity to extract context and minute features from the input data is improved by this flexibility. Furthermore, by using a feature pyramid network, YOLOv8 achieves superior results in multi-scaled object detection. Built on several layers, this network recognizes objects in an image that vary in size and scale with ease, guaranteeing thorough coverage. By utilizing these sophisticated features, YOLOv8 exhibits an innovative method of object recognition, maximizing scalability and attentional focus for enhanced performance in a variety of circumstances. YOLOv8 is likely to train more quickly than the other two-stage object detection models.

Architecture of YOLOv8:

YOLOv8 architecture consists of two main parts which are backbone and head. YOLOv8's backbone essentially contains the C2f module instead of C3 module of YOLOv5, and it is inspired by the ELAN module. Here C2 is CSP Bottleneck with 2 convolutions and C2f is the faster version of C2, C2f is designed to improve performance by increasing the number of skip connections and adding an extra split operation and that is why C2f was used here in this architecture of YOLOv8. To form the backbone of YOLOv8 here also CSPDarknet53 architecture is used but it is the modified version. With its 53 convolutional layers and a cross-stage partial connection, this design essentially aids the model in enhancing information flow between various architectural layers. At the head part of YOLOv8 architecture it contains multiple convolution layers, C2f module, Upsampling and Concatenation layers. At the head part of YOLOv8 architecture it contains multiple convolution layers, C2f module, Upsampling and Concatenation layers which is visible at Figure 4.5. These layers are responsible for predicting bounding boxes, scores and classes. Also, here all convolutional layers are 3x3 whereas for YOLOv5 convolutional layers were 1x1.

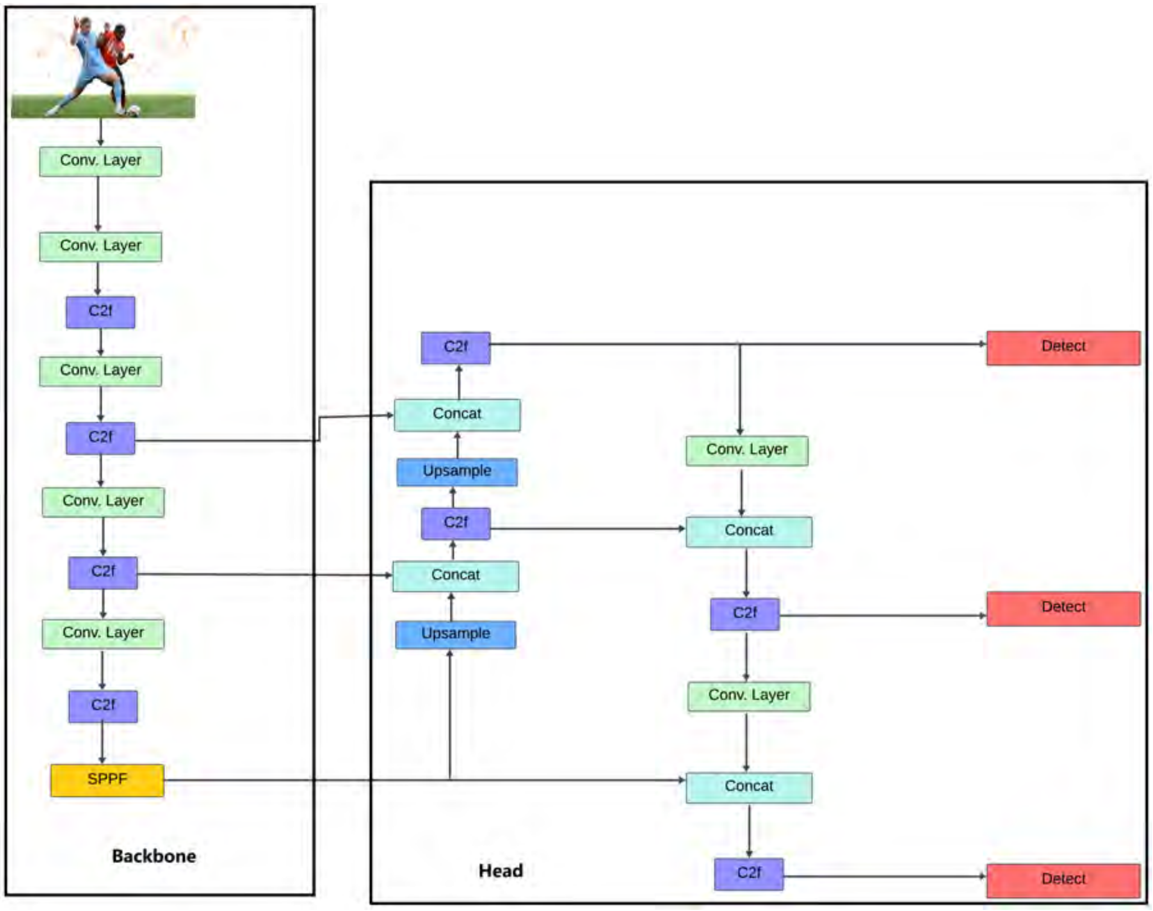


Figure 4.4: Architecture of YOLOv8

4.2.3 YOLO-NAS

With improved real-time object detection capabilities and production-ready performance, YOLO-NAS is a foundation object detection model. The model's key strength is greater real time object detection capability. It exceeds previous versions by providing cutting-edge performance with unmatched accuracy-speed performance. RepVGG is the foundation of YOLO-NAS, providing opportunities for post-training optimization through methods such as Re-parameterization and Post-training Quantization. This particular VGG-based neural network architecture incorporates regularization techniques to improve the generalization performance of deep learning models. Its architecture stands out for having more memory and speed. RepVGG is first trained with a multi-branch configuration, then re-parameterized to operate as a single branch in order to improve inference time. This feature is fully utilized by YOLO-NAS, which is highly beneficial for efficient production deployment. With this method, training and optimization may be precisely adapted for maximum inference speed and memory efficiency. By combining the re-parameterization of 8-bit quantization, the model employs QSP and QCI which are called quantization-aware modules to lessen the accuracy loss during post-training quantization. YOLO-NAS outperforms the previous YOLO versions by 10–20% in speed. It employs a superior architecture, AutoNAC, and is more accurate than the previous YOLO models. This offers the best accuracy and latency tradeoff performance, setting a new benchmark in object detection smooth support for NVIDIA and other inference engines. It is a production-ready model because of this functionality. It offers faster inference rates and improved memory efficiency. Compared to the previous YOLO models, YOLO-NAS is 10–20% faster and is more accurate. It also makes use of the superior AutoNAC architecture. With the best accuracy and latency tradeoff performance and seamless support for inference engines like NVIDIA, this breaks the previous record in object detection. YOLO-NAS has set new standards for accuracy and speed with a phenomenal latency of less than 5 milliseconds. This accomplishment represents a significant advancement in object detection models, especially in the context of the YOLO framework. It represents a significant breakthrough in computer vision as it pushes the envelope by combining neural architecture search methodologies, providing both high precision and quick processing. YOLO-NAS offers a customizable solution that allows users to customize and fine-tune their experience. By using fine-tuning recipes that are customized for the Roboflow-100 datasets and come with pre-trained weights, users may precisely customize the model to meet their needs. Users also gain from the features of Deci's SuperGradients, an open-source computer vision training library. Users are able to train new models and improve ones that already exist with the help of this library. Adding sophisticated approaches like knowledge distillation makes training more subtle and effective by adding another level of complexity. Combining these characteristics gives YOLO-NAS a more comprehensive approach and offers a strong toolkit for those looking for computer vision applications that combine flexibility and advanced capability. The model is ready for manufacturing because of this feature. Its inference speeds are higher and its memory efficiency is better.

Architecture of YOLO-NAS:

YOLO-NAS model developed by DeciAI were found utilizing their own exclusive NAS or Neural Architecture Search algorithm which is AutoNAC or Automated Neural Architecture Construction. AutoNAC was utilized to determine the most suitable architecture that combined the fundamental architectural contributions of other YOLO variants for the given task and thus it automates the design process of optimized training and inference which makes about $10\hat{1}4$ network architectures. Throughout the whole NAS process Quantization-Aware RepVGG or QA-RepVGG blocks are integrated into the model architecture to ensure that the model is compatible with Post-Training Quantization or PTQ. By using quantization-aware "QSP" and "QCI" modules which contain QA-RepVGG blocks that offer the advantages of 8-bit quantization and reparameterization allowing for the least amount of accuracy during PTQ. Here also Spatial Pyramid Pooling or SPP is utilizing the backbone of this architecture.

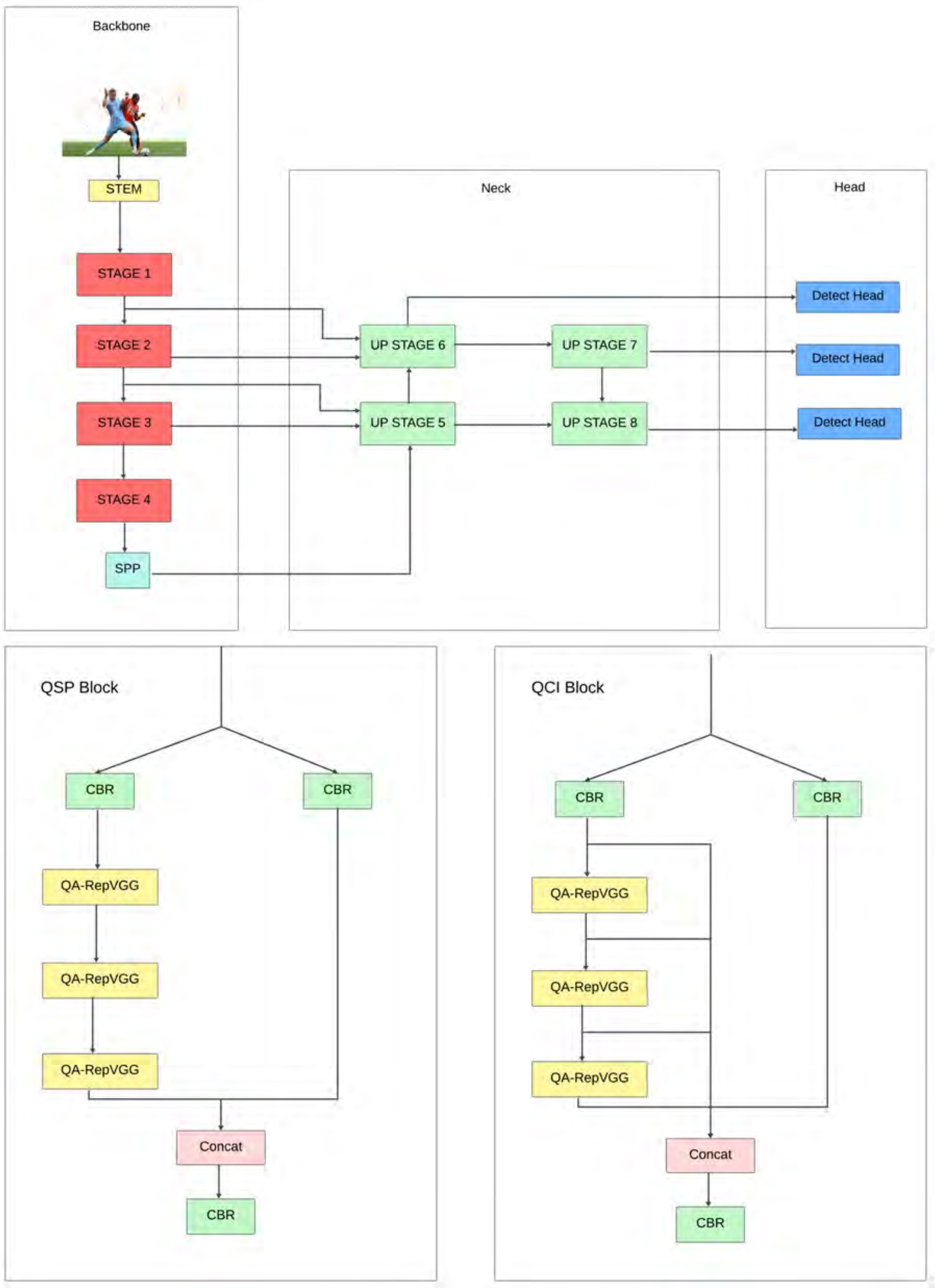


Figure 4.5: Architecture of YOLO-NAS

4.2.4 Faster R-CNN

Faster R-CNN is also a deep learning model that is used for object detection in computer vision. It stands for "Fast region based convolutional neural network". The processing inefficiencies of R-CNN were addressed with the introduction of Fast R-CNN, which made object detection quicker and more useful for real-world applications. The network basically suggests candidate bounding boxes that might contain objects of interest and reduces the number of areas making it faster. After region selection, Faster R-CNN extracts features using the convolutional neural network(CNN). The region classification and bounding box regression refine the bounding box coordi-nates, for a good alignment with the actual objects.The streamlined architecture of Fast R-CNN is the main benefit. Fast R-CNN distributes the convolutional feature computation over the whole image, in contrast to R-CNN, which handles each area proposal separately. This pooled computation expedites the detection process overall and considerably avoids unnecessary calculations. Additionally, the Region of Interest or ROI pooling layer is introduced by Fast R-CNN, facilitating the accurate and efficient extraction of features from region proposals. Compared to its predecessor, this innovation results in faster inference times and better accuracy. It is advantageous because it strikes a good balance between speed and precision, which makes it appropriate for applications where object identification in real-time is required.

Architecture of Faster R-CNN

Faster R-CNN architecture is built on two main components which are Region Proposal Network or RPN and Fast R-CNN detector. At Figure 4.8, which is an architecture diagram for our utilized Faster R-CNN model, we can see that at the start of this architecture Convolution Neural Network layers worked as the backbone. Here it has multiple convolution layers which can be AlexNet, ResNet, VGG etc. Then it goes to feature maps to extract different features or visual information from the images. After that it goes to RPN and Fast R-CNN detector. Previously R-CNN and Fast R-CNN models were utilizing the specific pursuit calculation that produces around 2000 area propositions. These 2000 district propositions are then given to CNN Engineering, which registers CNN highlights. But here in Faster R-CNN architecture it utilizes RPN which shares layers with different detection stages to improve feature representation and reduce proposal time of bounding boxes to focus for each image than its previous ancestor models. Then at the Fast R-CNN detector part it first goes to the Region of Interest or RoI pooling layer which helps the model to transform variable sized region proposals from RPN into fixed size feature maps that are flowed into the next network layers. Then from here it goes to Fully Connected layers which are responsible for classifying objects and bounding box regression. These are utilized by the Fast R-CNN detector which combines classification and regression losses and computes classification loss using a multi-task loss function, which is given below.

$$L(p_i, t_i, v_i) = \frac{1}{N_{\text{cls}}} \sum_i L_{\text{cls}}(p_i, p_i^*) + \lambda \cdot \frac{1}{N_{\text{reg}}} \sum_i p_i^* \cdot L_{\text{reg}}(t_i, v_i)$$

Here N_cls and N_reg are the number RoI's utilized for classification and regression respectively, pi is the predicted probability of classifying the i-th RoI, p_i* is the ground truth indicator for whether the i-th RoI is a foreground or background object, t_i and v_i represents the ground truth bounding box parameter and the predicted bounding box adjustment for the i-th RoI. L_cls and L_reg are the classification and regression loss functions and finally lambda is the balancing parameter that regulates trade offs between two losses.

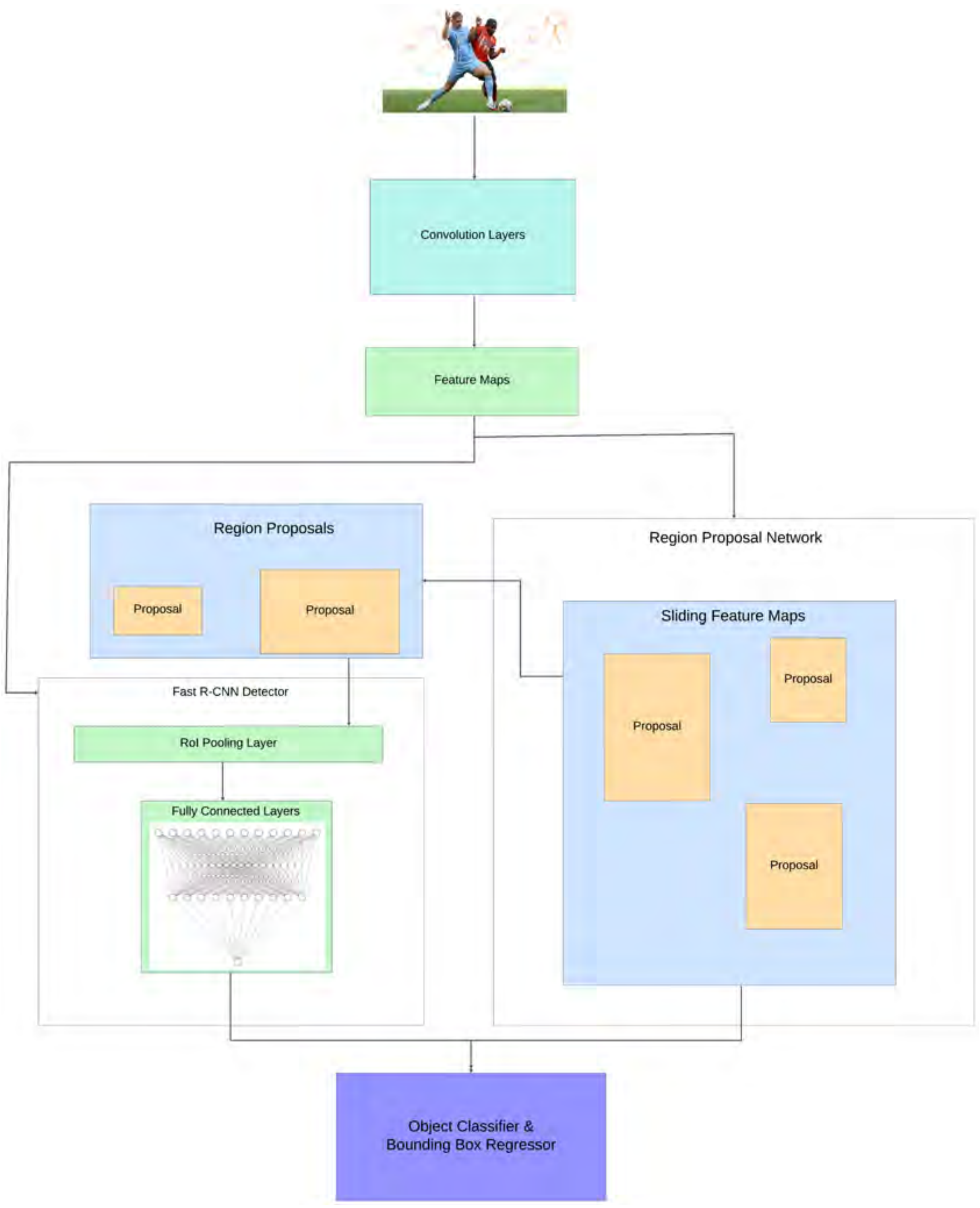


Figure 4.6: Architecture of Faster R-CNN

Moreover, two multi-object tracking algorithms, ByteTrack and BoT-Sort, are also used in the research.

4.2.5 ByteTrack

ByteTrack is a recent object-tracking algorithm that proposes a simple but effective optimization for the data association step. Most methods filter out detections with low confidence scores. This is because low-confidence detections are more likely to be false positives or related to objects not present in the scene. ByteTrack solves this problem using all detections, regardless of their confidence score. The algorithm works in two steps.

1. **High-confidence detections:** High-confidence detections are associated with tracks using IoU or appearance features. Both approaches are evaluated in the results section of the paper.

2. **Low-confidence detections:** Low-confidence detections are associated with tracks using only IoU. This is because low-confidence detections are more likely to be spurious or inaccurate, so it is important to be more conservative when associating them with tracks.

ByteTrack's combination with cutting-edge object detection systems, such as YOLOv5, YOLOv8, YOLO-NAS and Faster R-CNN, upgrades its following capacities. By beginning with high-accuracy object discoveries, ByteTrack establishes a strong starting point for its subsequent interaction, guaranteeing that each ensuing step is based on dependable information. In synopsis, ByteTrack is something other than a calculation; it's an exhaustive answer for genuine difficulties in PC vision. Its capacity to precisely track numerous objects progressively, no matter the climate's intricacies, positions it works as a significant method in the consistently developing scene of computer-based intelligence and innovation.

Architecture of the ByteTrack algorithm

ByteTrack algorithm works in two stages. In stage 1, it works with the high confidence box, and current frames are matched with previous frame tracklets. After that, in the second stage, low-confidence detection boxes are matched with the remaining unmatched predicted boxes from previous frames. These stages are shown at our drawn Figure 4.9 which is a architecture diagram of ByteTrack algorithm.

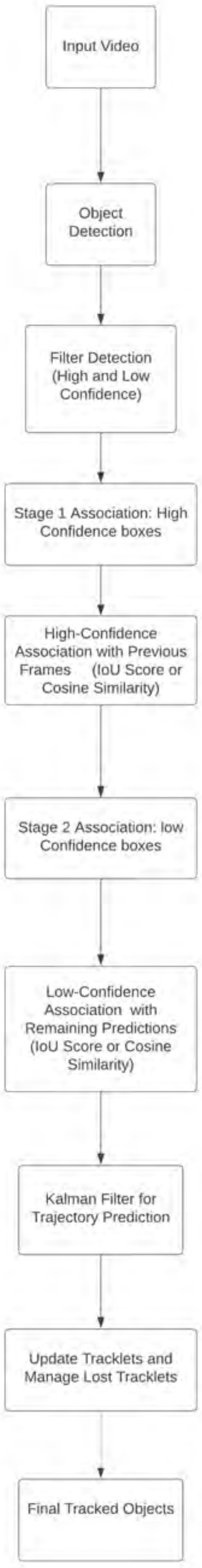

Input Video
Object Detection
Filter Detection (High and Low Confidence)
Stage 1 Association: High Confidence boxes
High-Confidence Association with Previous Frames (IoU Score or Cosine Similarity)
Stage 2 Association: low Confidence boxes
Low-Confidence Association with Remaining Predictions (IoU Score or Cosine Similarity)
Kalman Filter for Trajectory Prediction
Update Tracklets and Manage Lost Tracklets
Final Tracked Objects

Figure 4.7: Architecture of ByteTrack Algorithm

4.2.6 BoT-SORT

Botsort is built based on ByteTrack and consists of three ideas that work very well together. These are given below.

1. **Kalman Filter update:** SORT introduced a way of modeling the track state vector using a seven-tuple. BoT-SORT proposes replacing the bounding box's scale and aspect ratio with the width and height to create an eight-tuple. They also choose matrices from the Kalman filter as functions of the bounding box width and height. Recall that in DeepSORT, only the scale of the bounding box influences the matrices.
2. **Camera Motion Compensation:** In dynamic camera situations, static objects can appear to move, and moving things can appear to be fixed. The Kalman Filter does not consider camera motion for its predictions, so BoT-SORT proposes incorporating this knowledge. They use the global motion compensation technique (GMC) from the OpenCV Video Stabilization module. This technique extracts critical points from consecutive frames and computes the homography matrix between similar pairs. This matrix can then be reused to transform the predicted bounding box from the coordinate system of the structure to the coordinates of the next frame.
3. **IoU - ReID Fusion:** BoT-SORT proposes a new way of solving the association step by combining motion and appearance information. The cost matrix elements are determined, and the appearance distance is recalculated. The datasets are filtered out in pairs with large IoU or significant appearance distance. Then, the cost matrix element is updated as the minimum between the IoU and the new appearance distance.

Architecture of BoT-SORT algorithm

Here, first we input the video footage of the foul section, and then it will detect the objects. The tracking and motion estimation will be detected by using the Kalman Filter, which can be seen at Figure 4.10 of BotSORT tracker architecture. Then association and fusion occur using IoU and ReID. Moreover, the CMC (Camera Motion Compensation) is used to register the images.

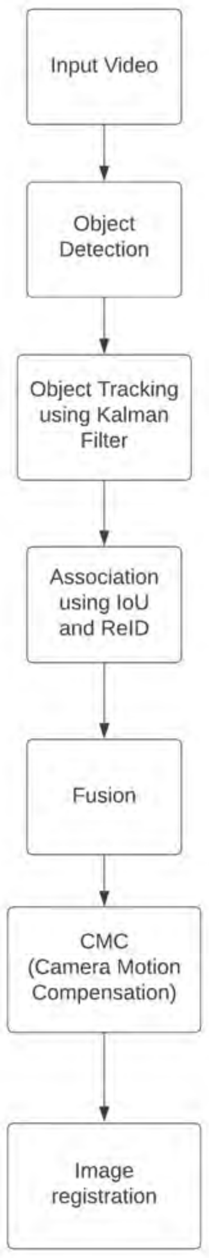

Input Video
Object Detection
Object Tracking using Kalman Filter
Association using IoU and ReID
Fusion
CMC (Camera Motion Compensation)
Image registration

Figure 4.8: Architecture of BoT-SORT Algorithm

4.2.7 Dark Channel Prior

As it is possible to get hazy video footage in a football match due to weather issues. Due to bad weather, we can get frames with poor visibility.That's why we need this Dark Channel Prior or DCP method to dehaze or remove fog from images, so that our model can predict foul properly in bad weathers as well. Mainly, DCP works in 4 steps.

1. **Dark Channel Construction:** In this stage, the color textures of the image are transferred to the dark channel, whereas when a prominent local patch is

utilized the hazy dark channels are obtained. Furthermore, a physically less significant median filter is not effective when constructing dark channels.

2. **Atmospheric light estimation:** In the next stage, the atmospheric light is predicted from the dark channel. It is therefore recommended to use an additional dark channel with a larger local patch size exclusively for atmospheric light estimation if the local patch size used while constructing dark channel is not large enough. Because of the possibility of preventing atmospheric light estimation from bright objects, local entropy also effectively balances the estimation accuracy.

3. **Transmission map estimation:** The traditional gain and offset control techniques are investigated in the third stage, however in order to accurately predict the transmission map, an adaptive correction strategy is required.

4. **Transmission map refinement:** In the last stage of DCP, when a hazy image is utilized here as a guidance image, performance of transmission map refinement improves . Moreover, the best transmission map estimation accuracy can be seen when the soft matting method is used and second best accuracy can be seen by utilizing the guided and cross-bilateral filters. Here, the Gaussian, guided and cross-bilateral filters are used to complete the stage.

After selecting YOLOv5, YOLOv8, YOLO-NAS and Fast R-CNN models we started to train to get trained weights or checkpoints for next stages.

1. **Input data:** We generated frames from the videos and also collected images from various online image stocks, which were annotated using bounding boxes into 3 different classes like foul, fouling-player and victim.

2. **Processing:** In this stage, frames with annotation data were used by the selected models and trained with their supported annotation format like YOLO for YOLOv5, YOLOv8 and YOLONas, and COCO for Fast R-CNN.

3. **Validation:** After training in the given dataset, after every iteration or epoch it predicts from validation set frames and matches with ground truth to give prediction accuracy.

After using these inputs, training has been done. The accuracy has been successfully generated for the separate models used.

Chapter 5

Implementation and Results

In this section, we will be describing the application of our selected models YOLOv5, YOLOv8, YOLO-NAS and Fast R-CNN which were used in the process of detection. In our implementation, we divided this into different stages: data pre-processing and augmentation, model training and accuracy detection.

5.1 Implementation

For Fast R-CNN, YOLOv5, YOLOv8 and YOLO-NAS, basically for all of our models, we used the same dataset. At first, we used our collected frames from videos of various types of football fouls and images from online stock resources to annotate and label them. Using the frames and annotated data, we started to train our models simultaneously. After successful training, we started to test the model for its accuracy in detecting foul and players who were involved in that foul. We have split our whole dataset into 3 parts, 70% for training, 20% for Validation and 10% for Testing. In both models we applied data augmentation by making changes in brightness- dark/bright, shearing- horizontal/vertical, blurring images, adding noise in images and flipped images horizontally to prevent any kind of over-fitting. We also pre-processed with the help of auto-orientation so that our frames do not flip into the wrong side and resize all images to 640 resolution as well.

For YOLOv5, YOLOv8 and YOLO-NAS models we used the YOLO format dataset. In this format, all the images have their own respective text file which contains the specified image's annotation coor-dinates.

For YOLOv5, YOLOv8 and YOLO-NAS model we used their medium variant of them, like for YOLOv5 is YOLOv5m , for YOLOv8 is YOLOv8m and for YOLO-NAS SOTA model we used YOLO_NAS_M pre-trained weights for transfer learning and also did some fine tuning by changing hyperparameters like number of epochs, batch size, learning rate etc to make our model work efficient as much as possible for our custom dataset and hardware as well.

For the Faster R-CNN model we implemented that using Detectron2 architecture. This Detectron2 architecture is a software system which helps to implement state-of-the-art object detection models and it was developed by Facebook AI Research or FAIR. For the Faster R-CNN model, COCO dataset format was utilized. But in case of COCO dataset format, all of the images have only one json file which contains all of the annotation markings all together. For Faster R-CNN we used R101-FPN pre-trained weight and also did some fine-tuning like before so that our model makes efficient usage of hardware resources

At Table 5.1 we can see the used hyperparameters and pre-trained weights for all of the selected models YOLOv5, YOLOv8, YOLO-NAS and Faster R-CNN when training using our custom dataset.

Models	**Variants**	**Number of Epochs**	**Batch Size**	**Learning Rate**
YOLOv5	YOLOv5m	150	8	0.01
YOLOv8	YOLOv8m	100	8	0.01
YOLO-NAS	YOLO_NAS_M	75	8	5e-4
Faster R-CNN	ResNeXt_101_FPN	10000	4	0.001

Table 5.1: Models Fine Tune

All the models were trained and tested into the same hardware specification.

RAM	29 GB
CPU	Intel Xeon 2GHz
GPU	Nvidia Tesla T4 x2

Table 5.2: Hardware Specification

5.2 Results

The YOLOv5, YOLOv8, YOLO-NAS and Faster R-CNN models both displayed different results regarding the detection process. Here, only YOLO-NAS and Faster R-CNN results are processed by using an external Tensorboard.

5.3 YOLOv5 Model Results

The YOLOv5 model presented the following results when trained with our custom dataset.

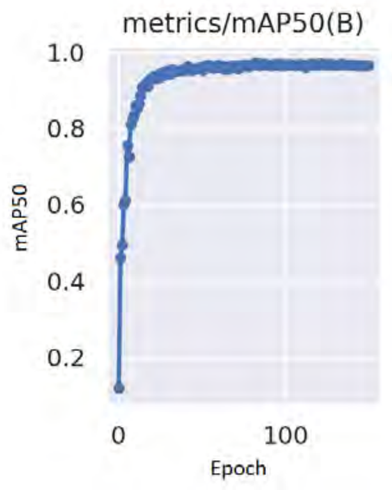


Figure 5.1: mAP_0.5 Graph

Here, in the Figure 5.1 above, mAP 0.5 reached its final point at 0.966. Again, in the figure 5.2 below, the precision graph shows that the precision of our model is around 0.95. It shows that the accuracy and precision of predicting foul increased as the model trained further.

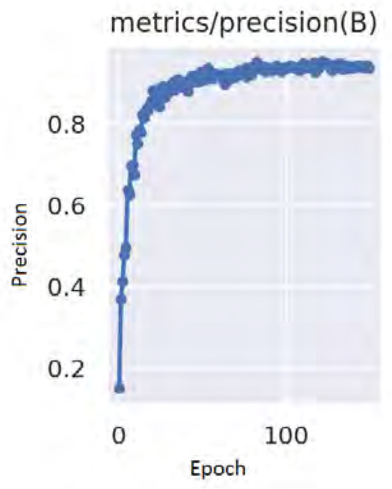


Figure 5.2: Precision Graph

In the graph 5.3 below, a high level of precision in the classification findings is indicated by the precision-confidence curve's produced outcome of 1 for each of our three classes, meaning the model showed a high degree of accuracy in identifying and classifying cases that fall into these three groups, with almost 97% confidence level.

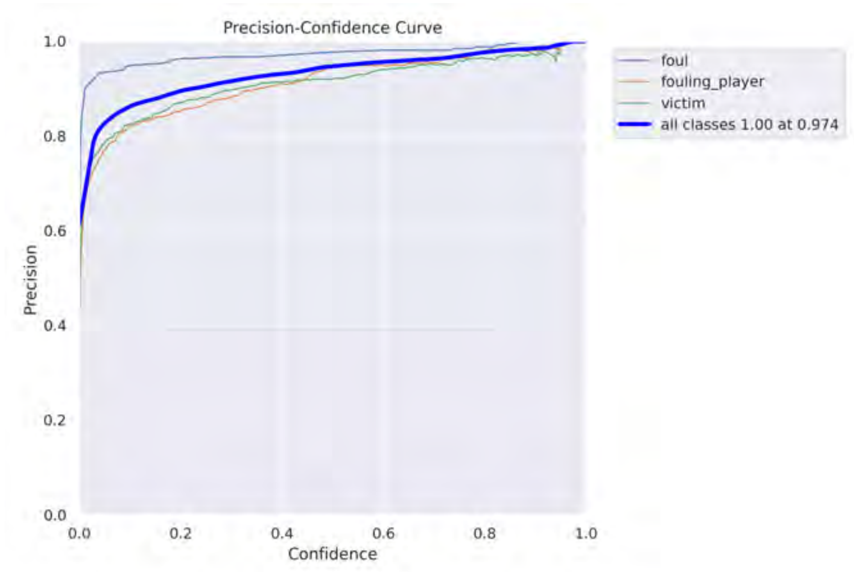


Figure 5.3: Precision-Confidence Curve

In the figure 5.4 below, we can see that as training progresses, the loss in predicting the class probabilities started decreasing and settled below 0.3. This is because from the start the classification loss was very high, but as the training iteration passes, the accuracy of predicting class probabilities of detected objects also in-creases. So, the class loss was gradually reduced.

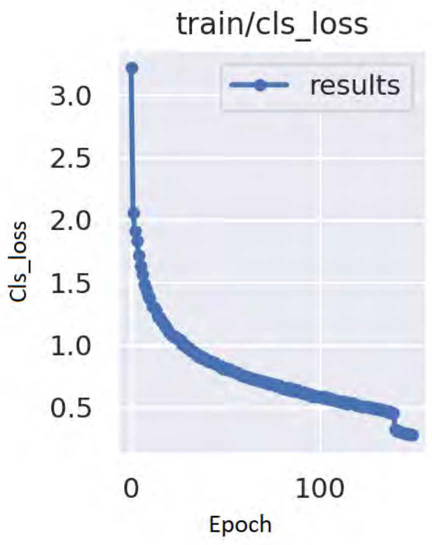


Figure 5.4: Train Class Loss Graph

We analyzed the loss being less than 0.4 based on the validation class loss in the figure 5.5. It implies that the model's performance is maintaining a relatively low level of error. A validation class loss of less than 0.4 suggests that the model is generalizing to new, unobserved data in addition to fitting the training set of data well.

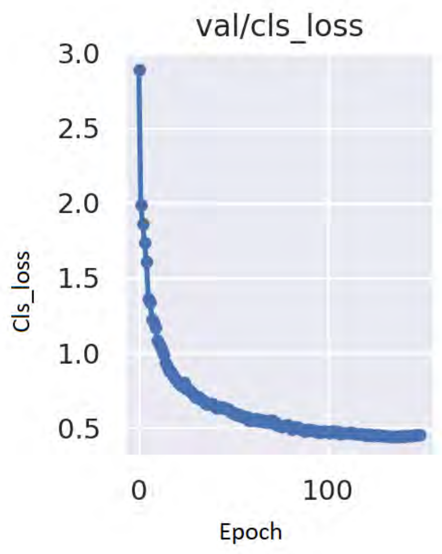


Figure 5.5: Validation Class Loss Curve

Here, in the figure 5.6 below, our Precision started from 1 while Recall was 0. But as the Recall increased, the Precision started to decrease because of the trade off between correct predictability of a positive instance vs identifying most of the positive instances in the given dataset.

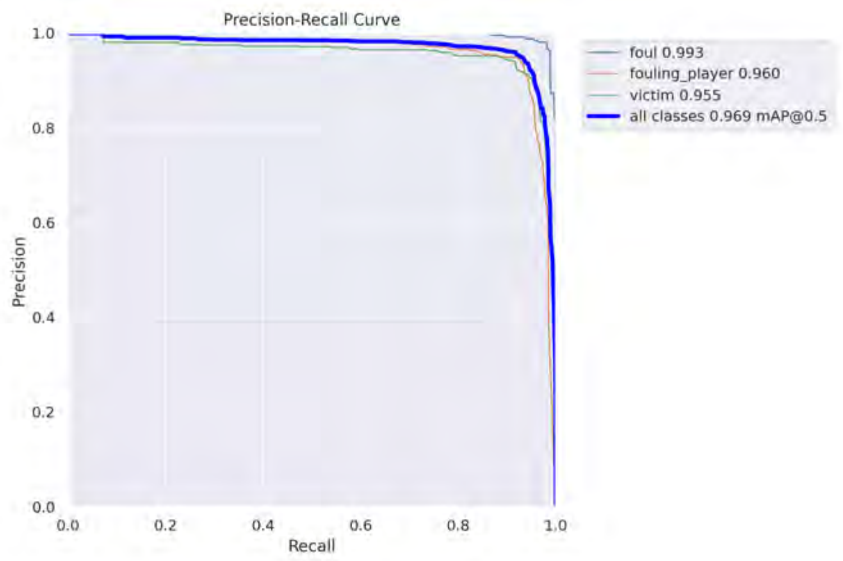


Figure 5.6: Precision-Recall Curve

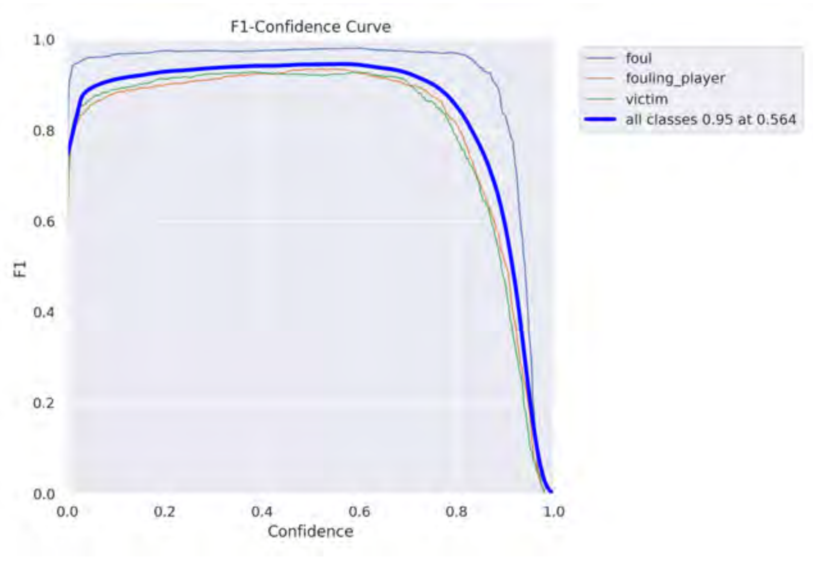


Figure 5.7: F1-Confidence Curve

From the figure 5.7 above, we analyzed that our best F1 score is when the confidence level threshold is between 0.5 and 0.6. In other confidence level thresholds, the F1 score does not wield a good result.

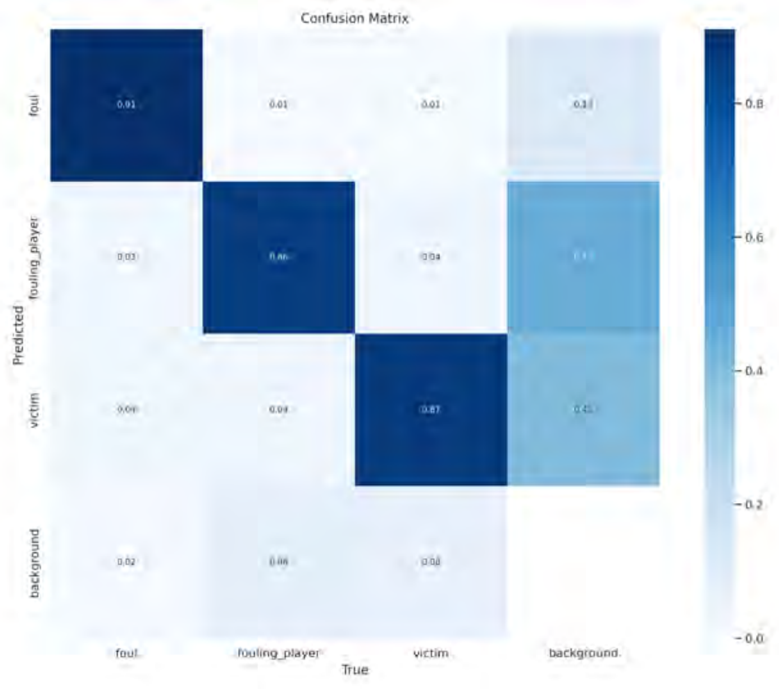


Figure 5.8: Confusion Matrix

From the confusion matrix, figure 5.8 above, we can see that our YOLOv5 model can predict 91% of the time foul, 86% of the time fouling_player and 87% of the time victim correctly.

5.4 YOLOv8 Model Results

After being trained on our own dataset, the YOLOv8 model produced the outcomes listed below.

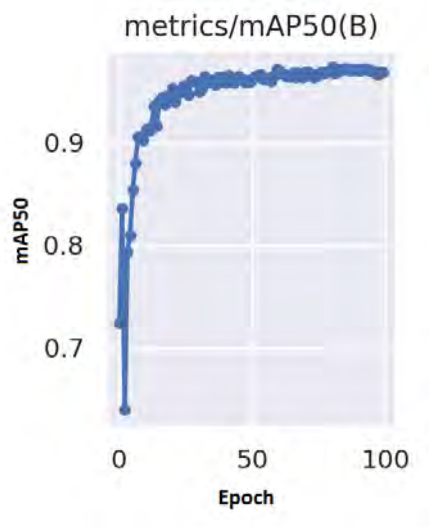


Figure 5.9: mAP_0.5 Graph

Here, in the above shown figure 5.13, mAP 0.5 reached its endpoint at 0.972. Once more, the precision graph in figure 5.2.2 below indicates that our model's precision is approximately 0.96. It demonstrates how, as the model was taught more, accuracy and precision grew.

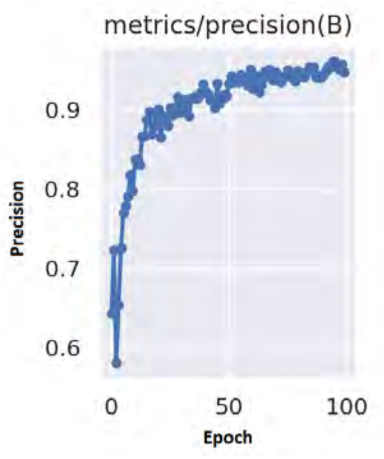


Figure 5.10: Precision Graph

Our model's precision score of 0.97 from the above figure 5.2.2 indicates its ability to distinguish genuine positive instances from all of the predicted positive instances is 97%. With a precision of 0.97, the model appears to be correct for the majority of its positive predictions. This is an important measure, especially in situations where reducing false positives is essential.

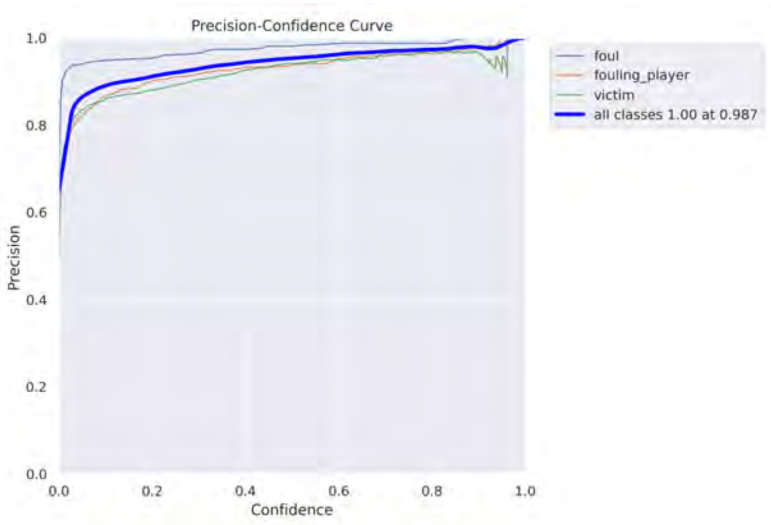


Figure 5.11: Precision-Confidence Curve

The precision-confidence curve's from the Figure 5.15 above produced outcome of 1 for each of our three classes indicates a high degree of precision in the classification findings, indicating that the model achieved a nearly 98.7% confidence level in correctly identifying and classifying cases that fall into these three groups.

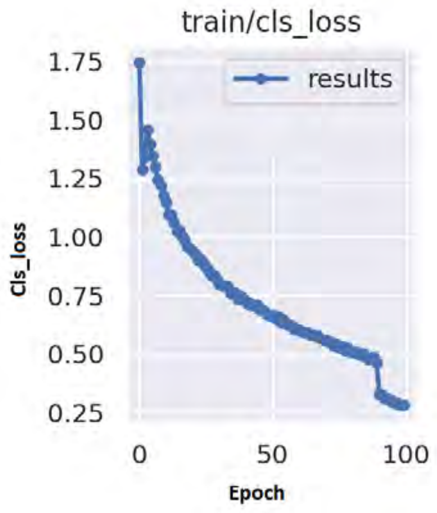


Figure 5.12: Train Class Loss Graph

As training goes on, the loss related to class probability prediction shows an apparent decrease and finally settles around 0.25, as Figure 5.16 shows. The learning process of the model is the cause of this behavior. At first, there was a noticeable difference between the expected and actual class probabilities, as evidenced by the comparatively large classification loss. Nonetheless, as training rounds continued, the model's ability to accurately forecast the class probabilities of recognized items increased. As a result, the class loss steadily dropped.

From Figure 5.17, we can see that the loss in class probability prediction began to decrease as training went on and settled at approximately 0.4. This occurs because,

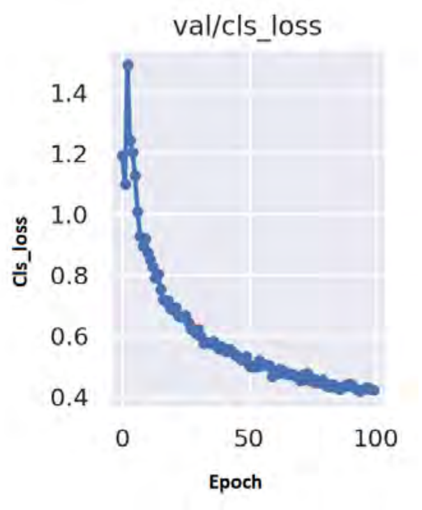


Figure 5.13: Validation Class Loss Curve

although the classification loss was initially quite significant, the accuracy of estimating the class probabilities of objects that are detected also increases with each training repetition. In other words, the class loss decreased with time.

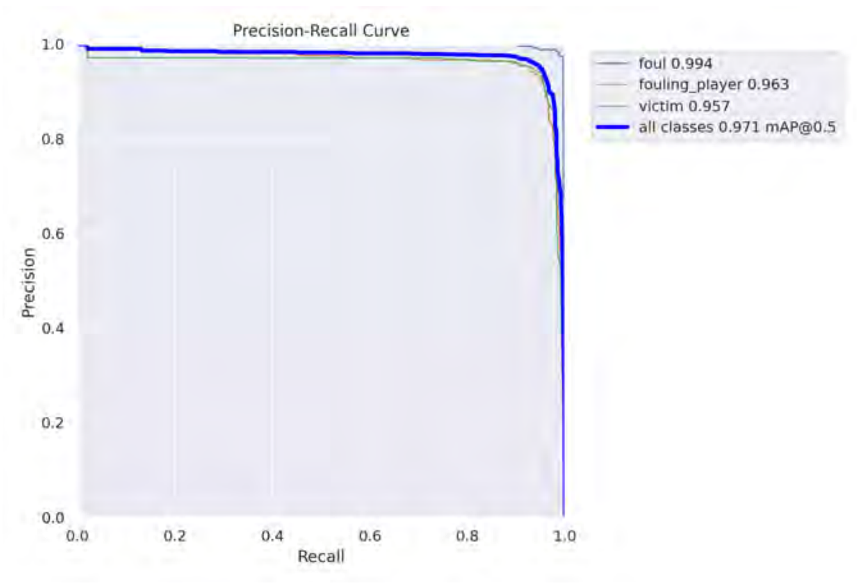


Figure 5.14: Precision-Recall Curve

Here, in figure 5.18 above, our Recall was 0 and our Precision was 1. However, as recall rose, precision began to decline as a result of a trade-off between accurately predicting a positive instance and finding the majority of positive cases in the dataset.

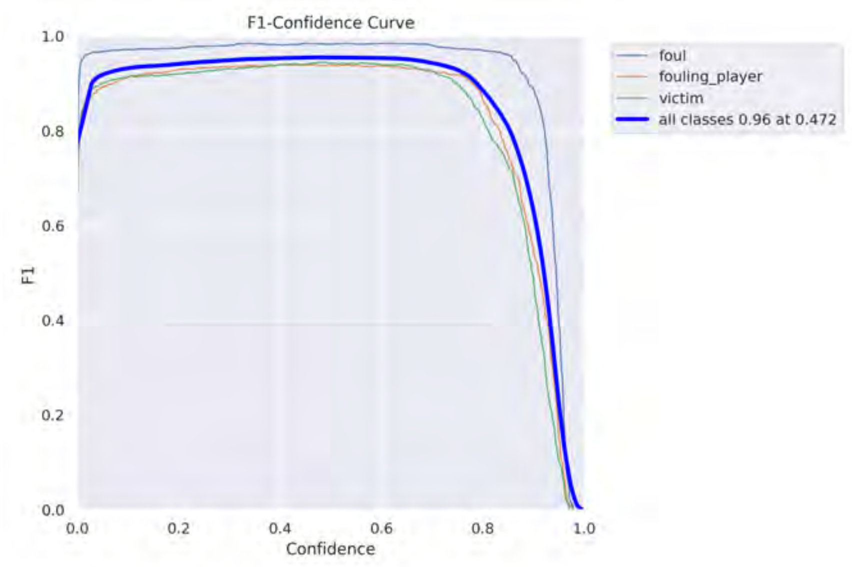


Figure 5.15: F1-Confidence Curve

We can see from the above figure 5.19 that we have the best F1 score around 0.6, or approximately 0.45, when the confidence level barrier is reached. The F1 score does not yield a favorable outcome at other confidence levels.

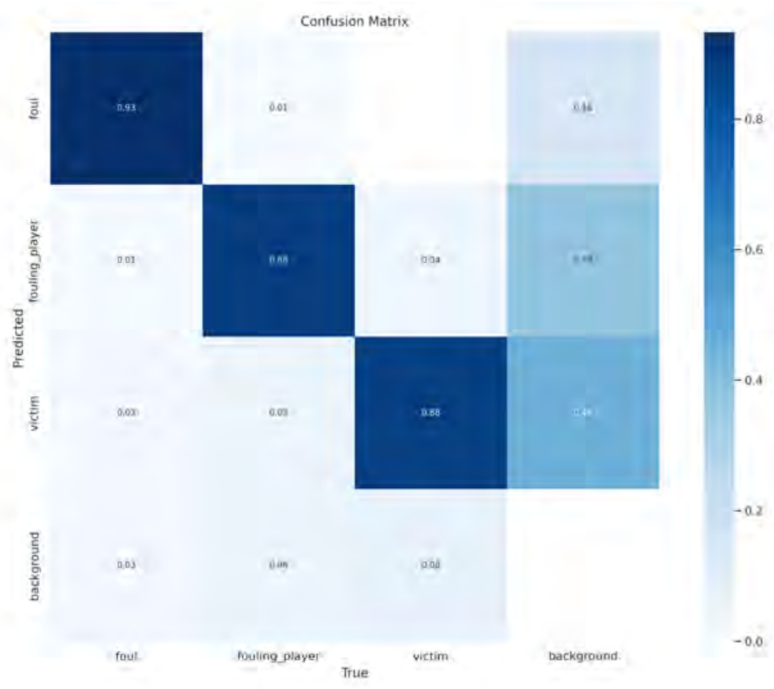


Figure 5.16: Confusion Matrix

From the Figure 5.20 above, the confusion matrix, shows that our YOLOv8 model can accurately anticipate 93% of time fouls and 88% of fouls involving both the fouling player and the victim.

5.5 YOLO-NAS Model Results

Following its training on our custom made dataset, the YOLO-NAS model yielded the following results.

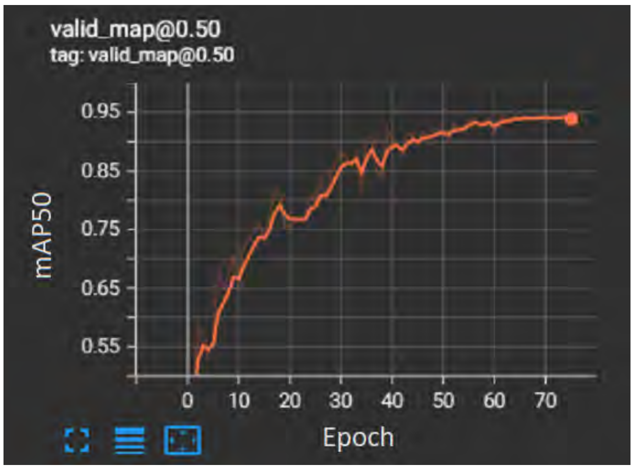


Figure 5.17: mAP 50 Graph

Here, mAP 0.5 reached its terminus at 0.944 in the figure 5.21 above.

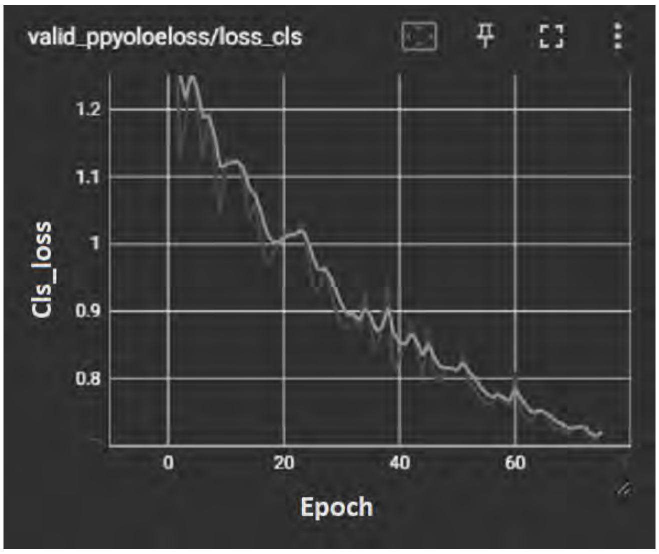


Figure 5.18: Validation Class Loss Graph

Here, in the Figure 5.22 given above, we can see that our loss in class probability prediction began to decrease as training went on and settled at around 0.7.

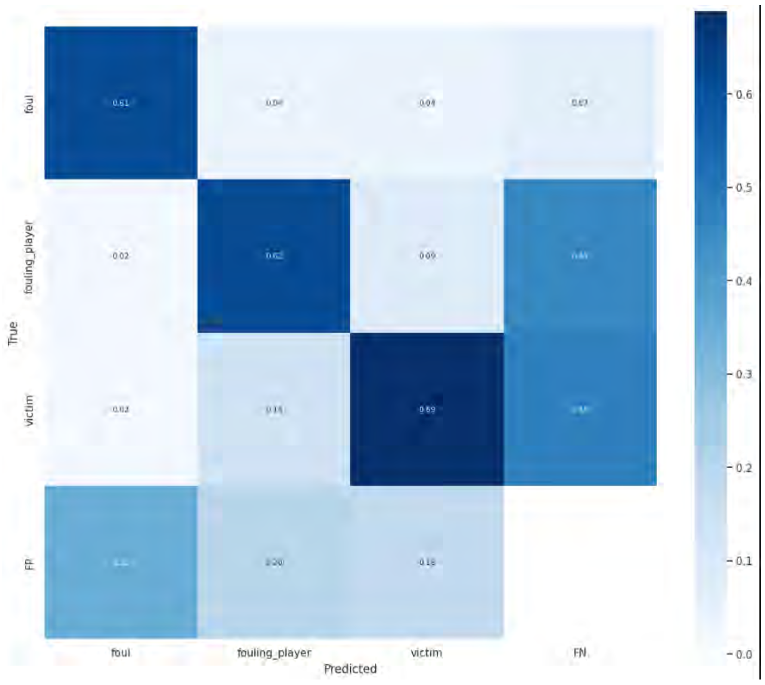


Figure 5.19: Confusion Matrix

By analyzing the confusion matrix, as shown in Figure 5.23, we can see that our YOLO-NAS model showed average predictive capability. It scored 61%, 62%, and 69% for foul, fouling_player and victim, respectively. These percentages represent how well the model detects and categorizes incidents within each relevant category.

5.6 Faster R-CNN Model Results

After being trained on our proprietary dataset, the YOLO-NAS model produced the subsequent outcomes.

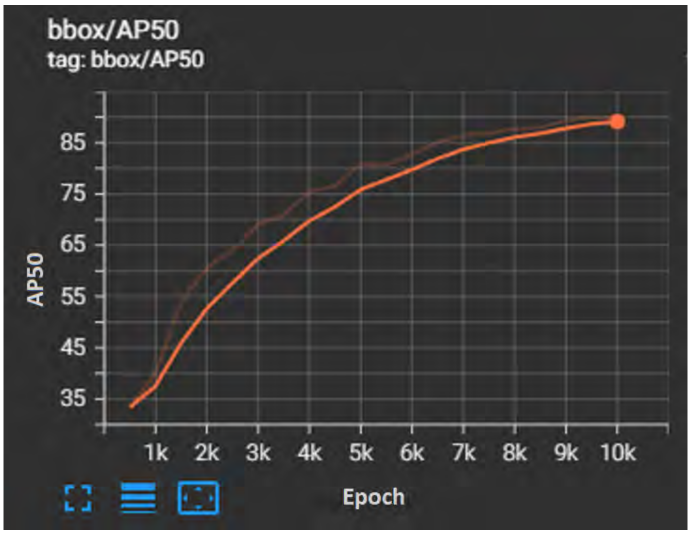


Figure 5.20: AP50 Graph

In the figure 5.24 above, for Faster R-CNN we can see that mAP 50 reached its final point at 89.85. Here, the accuracy of prediction also increased as the model trained further.

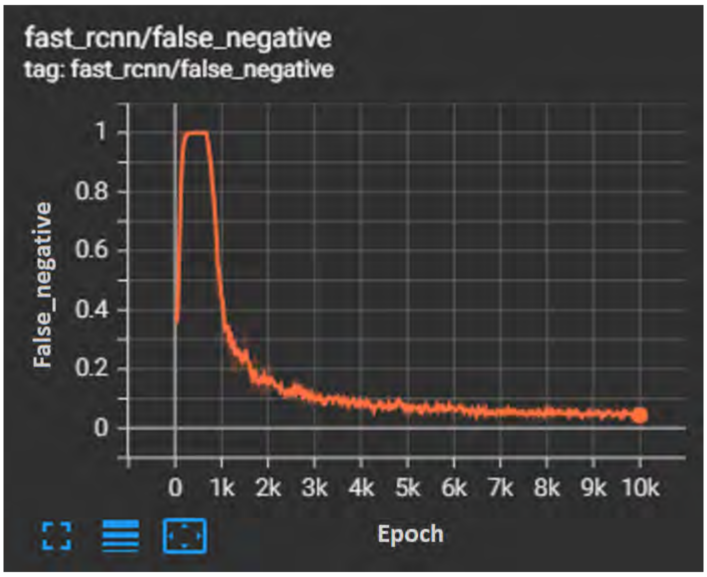


Figure 5.21: False Negative Graph

The model depicted in Figure 5.25, has demonstrated a great ability to minimize the number of times it incorrectly identifies positive cases, as seen by its false negative rate of less than 0.1.

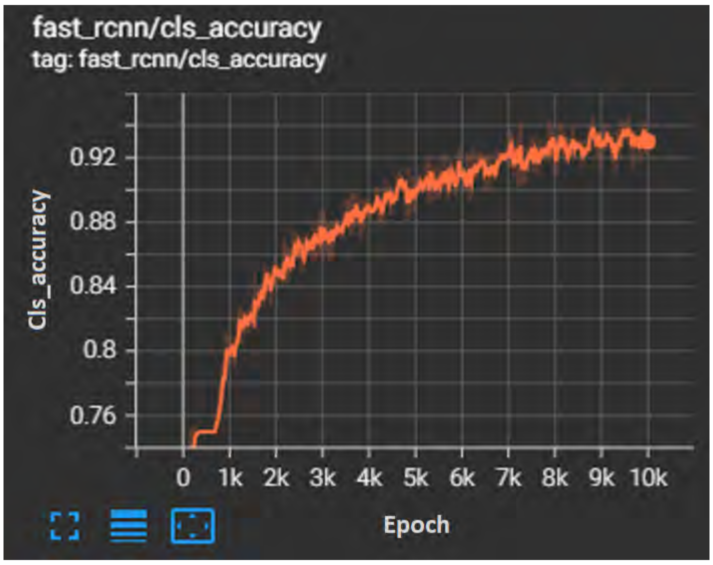


Figure 5.22: Foreground Class Accuracy Graph

From the two graphs presented in the figure 5.26 above and 5.27 below, we can see that in fg_cls_accruacy and cls_accuracy respectively, the rate of accuracy of classifying objects in our custom dataset has increased the further the model is trained, which is about 0.8 and 0.93 respectively.

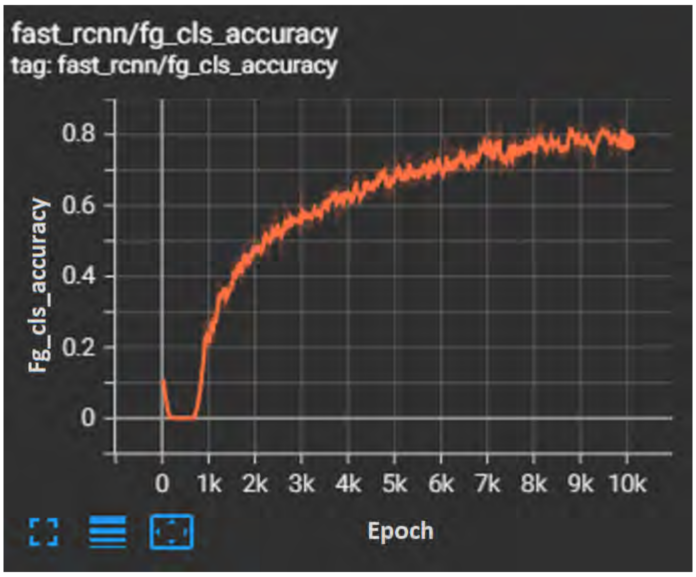


Figure 5.23: Classification Accuracy Graph

As can be seen in Figure 5.28 below, during the training phase, the loss related to class probability prediction seems to have dropped and finally stabilized at 0.1. This suggests that as time went on, the model's capacity to forecast class probabilities increased, eventually resulting in a comparatively low and steady loss amount.

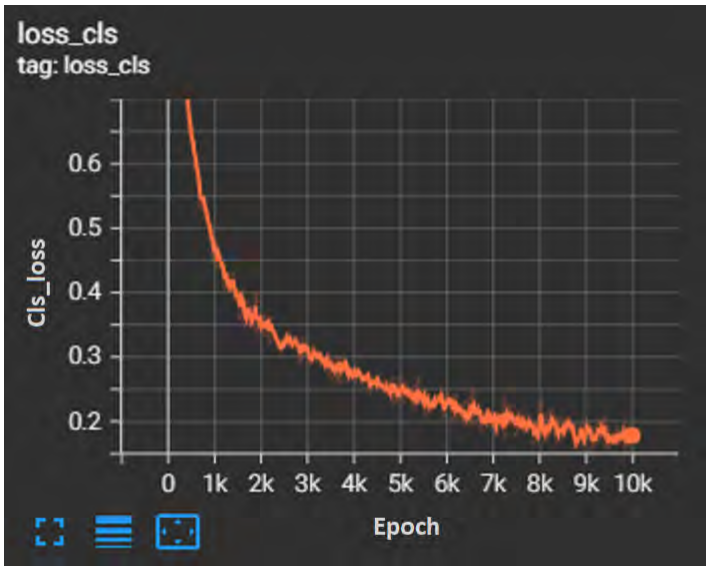


Figure 5.24: Classification Loss Graph

From Figure 5.29 below, we can analyze the fact that our model's total loss has stabilized at around 0.5 which indicates that the model's training procedure resulted in average in reducing the difference between the intended and projected outputs.

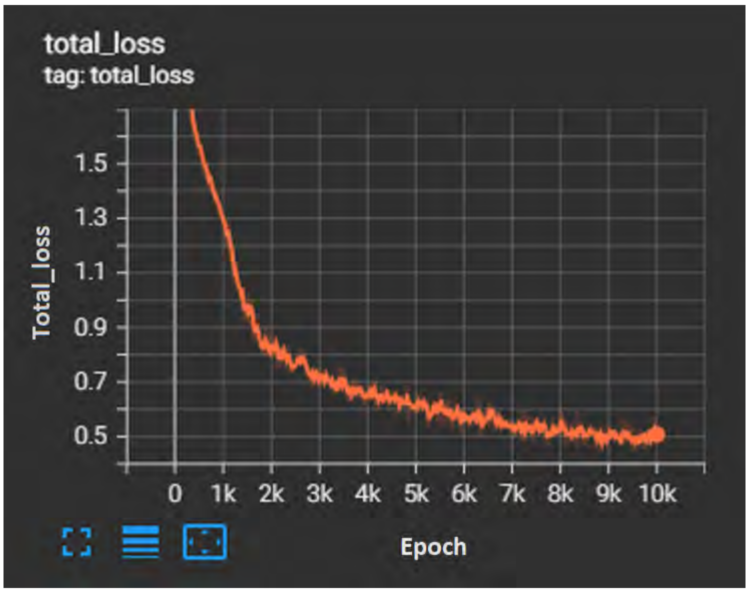


Figure 5.25: Total Loss Graph

5.7 Analysis

Models	Mean Average Precision	Class Loss
YOLOv5	0.966	0.2816
YOLOv8	0.972	0.2521
YOLO-NAS	0.944	0.713
Faster R-CNN	0.8985	0.125

Table 5.3: Analysis Table

From Table 5.3, result analysis table of all our trained model, we can come to say that YOLOv8 models have the highest accuracy detection rate than the other YOLOv5, YOLO-NAS and Fast R-CNN models while classification loss is lowest in Fast R-CNN than all other models. Overall, YOLOv8 yields the best performance among all other models for our custom dataset as it has the highest mean average precision and second lowest class loss comparison to all of our trained models. Here, for all of the cases the average precision was found in 0.5 IoU or Intersection over Union threshold. Examples of the detected Fouls are given below Figure 5.29 for our test sets.

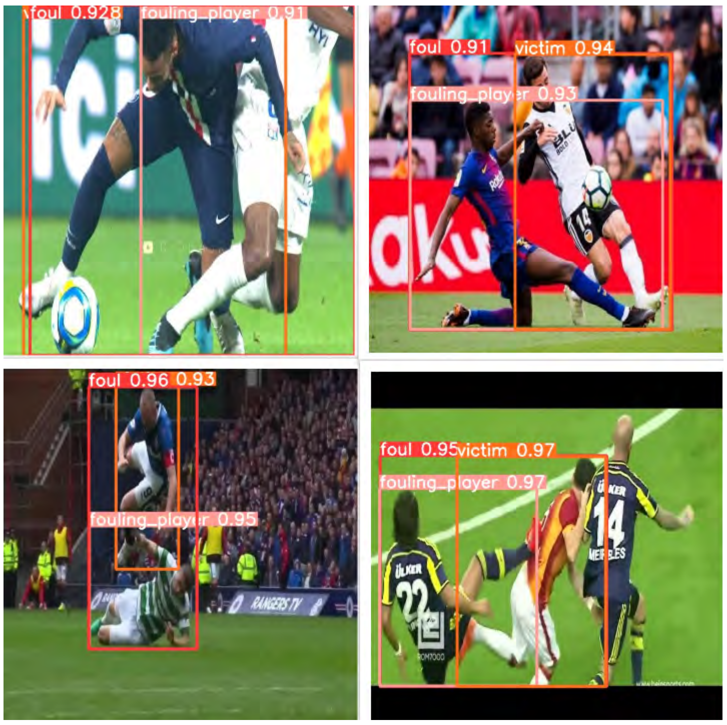


Figure 5.26: Examples of Detected Fouls

In both the images, our detection models have detected the foul and the related players with a very high accuracy ranging from 0.91 to 0.97.

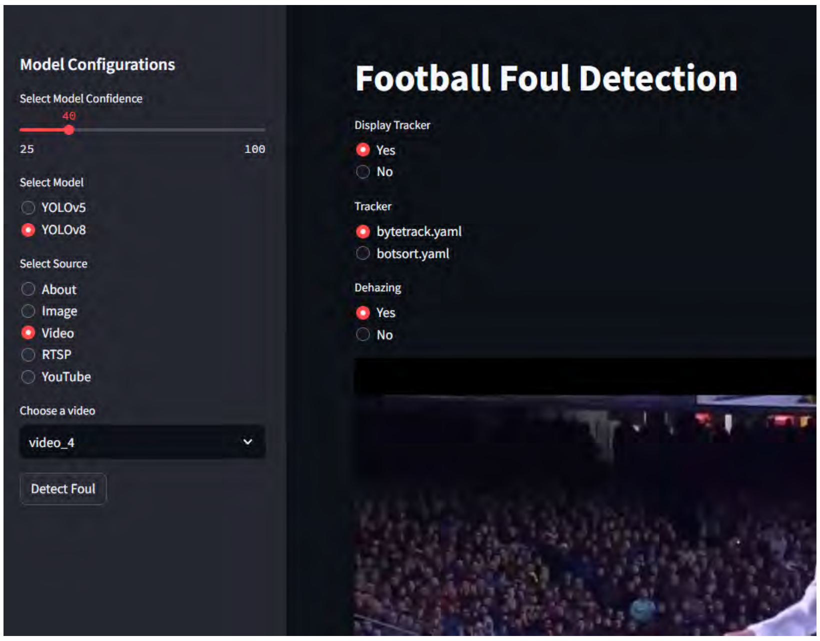
We can observe from the confusion matrices of our trained YOLO models that, while YOLO-NAS prediction of true positive for all classes, it was less than 70%. However, the prediction of true positives for all classes was greater than 85% for YOLOv5 and YOLOv8. This might be a result of the different architecture used by YOLO-NAS. The model uses a new architecture which is different from previous incarnations of the YOLO framework due to its distinct model design. A noteworthy feature is the addition of a brand-new basic block that is optimized for quantization, which has been carefully designed to improve quantization performance. This innovative architectural style deviates from traditional YOLO designs and emphasizes YOLO-NAS's dedication to optimizing the model's adaptability to many operational conditions in addition to speed and accuracy. It's critical to understand that, even though this design is a huge breakthrough, its efficacy may fluctuate depending on the particulars of each dataset. Because of this, we came to the conclusion that our custom dataset which has been tuned for other models did not work well for the YOLO-NAS training. Another reason might be that YOLO-NAS finds it challenging to identify small objects in clusters. As the model gains the ability to predict bounding boxes from the input itself, it finds it difficult to generalize objects in novel or unusual aspect ratios. In our dataset, we have bounding boxes of different sizes. So, it may have affected the training process of the model which resulted in lower true positives in the confusion matrix.

Chapter 6

Web Application Deployment

6.1 Web App Work Process

The structure of our web application is created for optimal functionality and user engagement. The first decision lies in the selection of a foul detection model, with either YOLOv5 or YOLOv8. The models will determine the further detection working process. Secondly, the kinds of sources that the web application will support must also be decided. Users can enter a variety of data forms, including YouTube links, RTSP streaming URLs, images, and videos. Because of its adaptability, the application can meet a wide range of user needs, which makes it a reliable and flexible option for foul detection jobs in a variety of settings. Subsequently, distinct techniques for data feeding need to be chosen for the various sources, including picture uploads, pre-uploaded video selection, RTSP streaming URL input, and YouTube link integration. With these features, the application can accommodate different data sources and preferences which makes the model function well with a variety of input formats. There can be an additional level of customisation added by choosing to integrate dehazing into the Foul detection process. It is very helpful in situations with difficult environmental conditions, such foggy weather, to be able to select whether the program should conduct dehazing. The ability to boost visibility is essential for accurate foul detection in many circumstances. In situations where there is video, RTSP streaming, or YouTube links, there is an option for adding a tracking tool. An ongoing and cohesive analysis of the visual data is provided by the application's ability to track fouls over time, which can be done by selecting from the two tracking algorithms, BotSort or ByteTrack. Lastly, with all the selections, the web app becomes ready to detect any foul in football in the given sources. We can sethe User interface of our web application below in the Figure 6.1 and whole workflow of our web application is drawn at Figure 6.2.


Model Configurations

Select Model Confidence: 40 (25–100)

Select Model: YOLOv5 [✓ YOLOv8]

Select Source: About, Image, [✓ Video], RTSP, YouTube

Choose a video: video_4

Detect Foul

Football Foul Detection

Display Tracker: [✓ Yes] No

Tracker: [✓ bytetrack.yaml] botsort.yaml

Dehazing: [✓ Yes] No

Figure 6.1: User Interface of Web Application

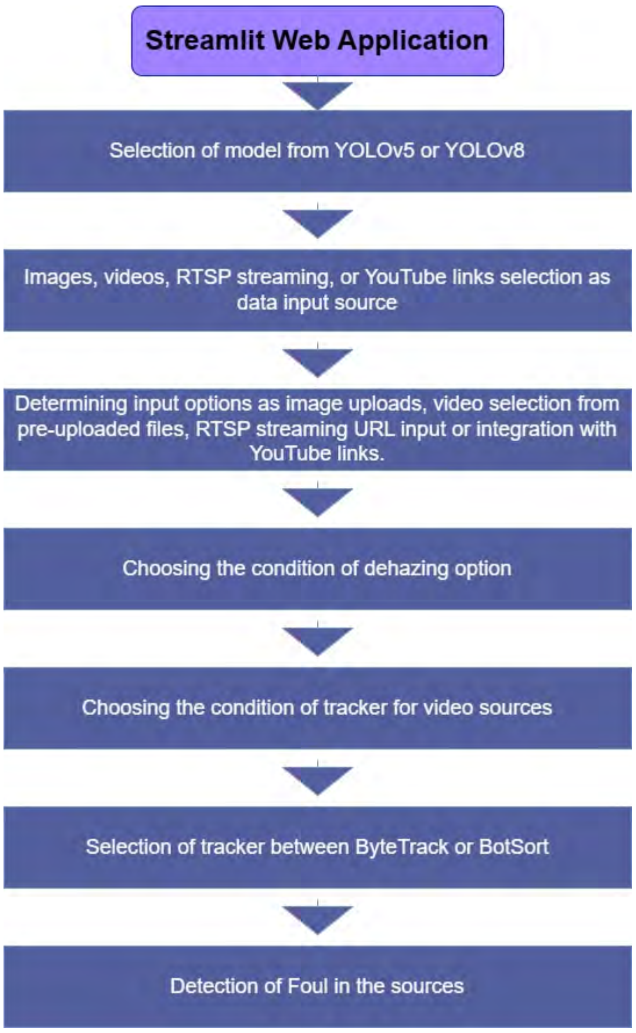

Streamlit Web Application
Selection of model from YOLOv5 or YOLOv8
Images, videos, RTSP streaming, or YouTube links selection as data input source
Determining input options as image uploads, video selection from pre-uploaded files, RTSP streaming URL input or integration with YouTube links.
Choosing the condition of dehazing option
Choosing the condition of tracker for video sources
Selection of tracker between ByteTrack or BotSort
Detection of Foul in the sources

Figure 6.2: Web Application Workflow

6.2 Limitations

The YOLO-NAS library's enforced limitations are one of the challenges faced during the creation of the web application. A dependency error pertaining to the YOLO-NAS library presented a challenge for the project, impeding the smooth integration and operation of the application. As YOLO-NAS is an almost new model, it still has many working problems which still need fixing. This constraint presented a technical hurdle as well as a reminder of how crucial it is to identify and resolve dependencies early in the development process in order to guarantee the web application runs smoothly which can't be done in the model's current form. This restriction made it more difficult to increase the system's overall robustness and reliability.

We encountered the issue of improper datasets when dealing with a dataset. More specifically, there are extremely few databases for photos related to ongoing foul play. Fouls happen quickly, as they usually do, therefore it might be challenging to get clear views at the right moment in photos or frames.This fact causes the majority of the photos to become blurry or to have an improper angle, examples of this shown at Figure 6.3 and 6.4. As a result, it becomes challenging to get the models to effectively extract information from the photos, which eventually reduces reliability as well as accuracy.

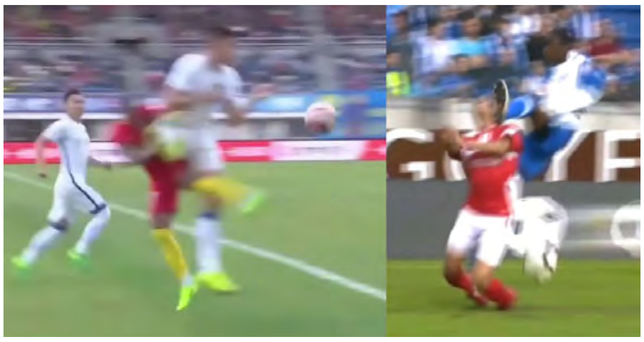


Figure 6.3: Example of Blurred Images

In this case, it is difficult to determine how the foul occurred because both photos appear sufficiently unclear. This makes recognizing the foul more difficult and complex.

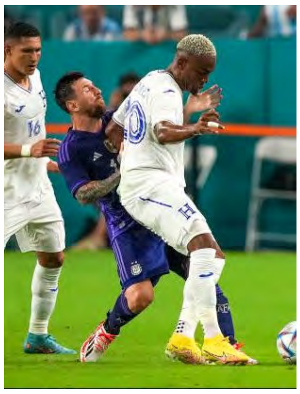


Figure 6.4: Example of Overlapping Image

Because both players are practically in a serial view of the camera in this picture, it becomes impossible to determine who committed the foul and how. This indicates that there is an error in the camera's angle meaning, the camera is not at the right position for a clear shot. These kinds of datasets cause the model to underperform compared to expectations, which leads to limitations in both performance and accuracy.

Lastly, hardware constraints also arose when we tried to implement the models. We were unable to run the four models on our local hardware because they require a lot of resources to run. Consequently, needing the assistance of third-party cloud services like Kaggle and Google Colab. However, these platforms also have their own restrictions, such time limits or restricted access. Again, the average run time for these models was two hours, which increased the time required for adding and applying additional datasets and other modifications.

Chapter 7

Conclusion and Future Work

7.1 Conclusion

Football is a very popular game for many people. As in the world of football, there have been flaws in case of detecting fouls either by referee's mistake or by any other circumstances. From the starting of the game to the present date, these flaws have immensely lowered in each passing time. But many flaws are still presenting different challenges to solve those problems. Our research in Real-Time Foul Detection can help tremendously in case of detecting foul more accurately. This will also ensure to deliver a fair match to both the teams playing and also to the audience. In this research, machine learning models YOLOv5, YOLOv8, YOLO-NAS and Fast R-CNN successfully detected fouls from our custom datasets with 96.6%, 97.2%, 94.4% and 89.85% accuracy respectively. Between these four models, YOLOv8 has the highest accuracy rating. After finding our overall two best performing models we used their trained weight for our custom dataset to deploy as a web application which has multiple functionalities like tracker selection, confidence meter, dehaze, source etc to detect foul from football matches for their own perspective. In future, we plan to research our model further to improve the accuracy as close to perfection, so that the detection rate of objects is more accurate in the live video services for fair football matches.

7.2 Future Work

Going forward, a number of interesting ideas come to the forefront for investigation and improvement. First off, improving the accuracy and consistency of our model will depend much on our search for new precise football foul related datasets. The intricacies of many foul scenarios can be more accurately captured by a large and varied dataset, which enhances the algorithmic comprehension. Also, different types of fouls can be implemented in our detection model. Furthermore, the incorporation of more recent and reliable models, such the YOLO-NAS, has great potential to enhance the complexity and effectiveness of our detection system. Using state-of-the-art models keeps our system on the leading edge of technology and maximizes its capacity to identify fine nuances and minute differences in foul occurrences. In addition, the introduction of high-definition cameras is a significant advancement in the search for clearer and more precise photographs. Changing to a more modern camera raises the accuracy of foul detection overall by improving the quality of the input data and enabling a more complex analysis of player motions and interactions. A revolutionary path for our study, looking farther forward, is to deploy our detection technology right on the football pitch. Complete integration with the live broadcast infrastructure would be necessary for this audacious step to be taken, allowing for real-time analysis and prompt intervention to ensure fair play. A paradigm shift towards proactive action in the area of foul identification during live matches is marked by this method, which also corresponds with the continuously increasing demand for live sports analytics. Essentially, the work that is planned for the future involves not just improving the computational components but also advances further in data quality, model architecture, and real-time application.

Bibliography

[1] X. Wu, Y. Ou, H. Qian, and Y. Xu, “A detection system for human abnormal behavior,” in *2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2005, pp. 1204–1208. DOI: 10.1109/IROS.2005.1545205.

[2] D. Zhang, D. Gatica-perez, S. Bengio, and I. Mccowan, “Semi-supervised adapted hmms for unusual event detection,” Jul. 2005, 611–618 vol. 1, ISBN: 0-7695-2372-2. DOI: 10.1109/CVPR.2005.316.

[3] S. Dobson, P. Dawson, J. Goddard, and J. Wilson, “Are football referees really biased and inconsistent?: Evidence on the incidence of disciplinary sanction in the english premier league,” *Journal of the Royal Statistical Society Series A*, vol. 170, pp. 231–250, Feb. 2007. DOI: 10.1111/j.1467-985X.2006.00451.x.

[4] M. Akbarniai Tehrani, R. Kleihorst, P. Meijer, and L. Spaanenburg, “Abnormal motion detection in a real-time smart camera system,” Oct. 2009, pp. 1–7. DOI: 10.1109/ICDSC.2009.5289359.

[5] M. Alhelou, “Hand gesture recognition system,” Jan. 2011.

[6] N. H. Dardas and N. D. Georganas, “Real-time hand gesture detection and recognition using bag-of-features and support vector machine techniques,” *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 11, pp. 3592–3607, 2011. DOI: 10.1109/TIM.2011.2161140.

[7] I. Mchale, P. Scarf, and D. Folker, “On the development of a soccer player performance rating system for the english premier league,” *Interfaces*, vol. 42, pp. 339–351, Aug. 2012. DOI: 10.2307/23254864.

[8] M. El-Sayed and M. Ahmed, “Automated edge detection using convolutional neural network,” *International Journal of Advanced Computer Science and Applications*, vol. 4, Jan. 2013.

[9] A. Bialkowski, P. Lucey, P. Carr, Y. Yue, S. Sridharan, and I. Matthews, “Large-scale analysis of soccer matches using spatiotemporal tracking data,” Dec. 2014.

[10] H.-I. Lin, M.-H. Hsu, and W.-K. Chen, “Human hand gesture recognition using a convolution neural network,” vol. 2014, Aug. 2014, pp. 1038–1043. DOI: 10.1109/CoASE.2014.6899454.

[11] W. Aitfares, A. Kobbane, and A. Kriouile, “Suspicious behavior detection of people by monitoring camera,” in *2016 5th International Conference on Multimedia Computing and Systems (ICMCS)*, 2016, pp. 113–117. DOI: 10.1109/ICMCS.2016.7905601.

[12] R. Rein and D. Memmert, “Big data and tactical analysis in elite soccer: Future challenges and opportunities for sports science,” *SpringerPlus*, vol. 5, Dec. 2016. DOI: 10.1186/s40064-016-3108-2.

[13] B. Neyshabur, S. Bhojanapalli, D. Mcallester, and N. Srebro, “Exploring generalization in deep learning,” in *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2017/file/10ce03a1ed01077e3e289f3e53c72813-Paper.pdf.

[14] S. Alashhab, A.-J. Gallego, and M. Á. Lozano, “Hand gesture detection with convolutional neural networks,” in *Distributed Computing and Artificial Intelligence, 15th International Conference*, Springer International Publishing, Jul. 2018, pp. 45–52. DOI: 10.1007/978-3-319-94649-8_6. [Online]. Available: https://doi.org/10.1007/978-3-319-94649-8_6.

[15] J. Fernández and L. Bornn, “Wide open spaces: A statistical technique for measuring space creation in professional soccer,” Mar. 2018.

[16] H. Sarmento, F. Clemente, L. Harper, I. Teoldo da Costa, A. Owen, and A. Figueiredo, “Small sided games in soccer-a systematic review,” *International Journal of Performance Analysis in Sport*, vol. ahead-of-print, Nov. 2018. DOI: 10.1080/24748668.2018.1517288.

[17] T. Wang, M. Qiao, Y. Deng, *et al.*, “Abnormal event detection based on analysis of movement information of video sequence,” *Optik*, vol. 152, pp. 50–60, Jan. 2018. DOI: 10.1016/j.ijleo.2017.07.064. [Online]. Available: https://utt.hal.science/hal-03320601.

[18] R. Agyeman, R. Muhammad, and G. S. Choi, “Soccer video summarization using deep learning,” in *2019 IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*, 2019, pp. 270–273. DOI: 10.1109/MIPR.2019.00055.

[19] K. Apostolou and C. Tjortjis, “Sports analytics algorithms for performance prediction,” in *2019 10th International Conference on Information, Intelligence, Systems and Applications (IISA)*, 2019, pp. 1–4. DOI: 10.1109/IISA.2019.8900754.

[20] H. Liu and B. Bhanu, “Pose-guided r-cnn for jersey number recognition in sports,” Jun. 2019, pp. 2457–2466. DOI: 10.1109/CVPRW.2019.00301.

[21] N. Nayan, S. Sahu, and S. Kumar, “Detecting anomalous crowd behavior using correlation analysis of optical flow,” *Signal, Image and Video Processing*, vol. 13, Sep. 2019. DOI: 10.1007/s11760-019-01474-9.

[22] S. Cui, Y. Zhou, Y. Wang, and L. Zhai, “Fish detection using deep learning,” *Applied Computational Intelligence and Soft Computing*, vol. 2020, pp. 1–13, Jan. 2020. DOI: 10.1155/2020/3738108.

[23] Y. Liu, S. Zhang, Z. Li, and Y. Zhang, “Abnormal behavior recognition based on key points of human skeleton,” *IFAC-PapersOnLine*, vol. 53, no. 5, pp. 441–445, 2020. DOI: 10.1016/j.ifacol.2021.04.120. [Online]. Available: https://doi.org/10.1016/j.ifacol.2021.04.120.

[24] I. Lucić, S. Babić, and D. Vučkov, "Perception of using var technology in football after completion of training and education and experiences of croatian video assistant referees (vars) and assistant vars (avars)," in *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, 2020, pp. 905–911. DOI: 10.23919/MIPRO48935.2020.9245111.

[25] A. Sharma, A. Mittal, S. Singh, and V. Awatramani, "Hand gesture recognition using image processing and feature extraction techniques," *Procedia Computer Science*, vol. 173, pp. 181–190, Jan. 2020. DOI: 10.1016/j.procs.2020.06.022.

[26] B. Torgler, "Big data, artificial intelligence, and quantum computing in sports," in Sep. 2020, pp. 153–173, ISBN: 978-3-030-50800-5. DOI: 10.1007/978-3-030-50801-2_9.

[27] Y. Zhang, Z. Chen, and B. Wei, "A sport athlete object tracking based on deep sort and yolo v4 in case of camera movement," in *2020 IEEE 6th International Conference on Computer and Communications (ICCC)*, IEEE, Dec. 2020. DOI: 10.1109/iccc51575.2020.9345010. [Online]. Available: https://doi.org/10.1109/iccc51575.2020.9345010.

[28] X. Hu, "Football player posture detection method combining foreground detection and neural networks," *Scientific Programming*, vol. 2021, pp. 1–11, Jun. 2021. DOI: 10.1155/2021/4102294.

[29] B. Mahaseni, E. R. M. Faizal, and R. G. Raj, "Spotting football events using two-stream convolutional neural network and dilated recurrent neural network," *IEEE Access*, vol. 9, pp. 61 929–61 942, 2021. DOI: 10.1109/access.2021.3074831. [Online]. Available: https://doi.org/10.1109/access.2021.3074831.

[30] D. Satybaldina, N. Glazyrina, K. Kalymova, and V. Stepanov, "Development of an algorithm for abnormal human behavior detection in intelligent video surveillance system," *IOP Conference Series: Materials Science and Engineering*, vol. 1069, p. 012 046, Mar. 2021. DOI: 10.1088/1757-899X/1069/1/012046.

[31] D. Yang and X. Bian, "Design and research on the foul detector of a long jump jumping line based on a vision sensor," *Journal of Sensors*, vol. 2021, pp. 1–17, Nov. 2021. DOI: 10.1155/2021/8364194.

[32] H. Guan and H. Niu, "Feature extraction of foul action of football players based on machine vision," *Mobile Information Systems*, vol. 2022, pp. 1–6, Jan. 2022. DOI: 10.1155/2022/7253159.

[33] N. Lucchesi, *Football Games Analysis from video stream with Machine Learning — towardsdatascience.com*, https://towardsdatascience.com/football-games-analysis-from-video-stream-with-machine-learning-745e62b36295, [Accessed 18-09-2023].