

Shuttlecock tracking and player service fault detection

A major project report submitted in partial fulfillment of the requirement for
the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

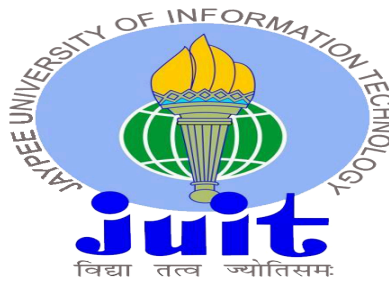
Submitted by

Kartik Parihar (201177)

Sayam Puri (201136)

Under the guidance & supervision of

Dr. Nishant Sharma



**Department of Computer Science & Engineering and
Information Technology**

Jaypee University of Information Technology, Wagnaghat,

Solan – 173234

DECLARATION

I hereby declare that the work presented in this report entitled ‘**A Shuttlecock and Player service fault detection**’ in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Dr. Nishant Sharma(Asst. Professor CSE/IT)**

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

(Student Signature with Date)

Kartik Parihar

Roll No: 201177

Sayam Puri

Roll No: 201136

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)

Dr. Nishant Sharma

(Asst. Professor CSE/IT)

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology

CERTIFICATE

This is to certify that the work which is being presented in the project report titled “Shuttlecock tracking and player service fault detection ” in partial fulfillment of the requirements for the award of the degree of B.Tech in Computer Science And Engineering and submitted to the Department of Computer Science And Engineering, Jaypee University of Information Technology, Waknaghat is an authentic record of work carried out by “Kartik Parihar, 201177” , “Sayam Puri, 201136” during the period from Aug 2023 to May 2024 under the supervision of **Dr. Nishant Sharma(Asst. Professor CSE/IT)**, Department of Computer Science and Engineering, Jaypee University of Information Technology, Waknaghat.

Mr. Kartik Parihar (201177)

Mr. Sayam Puri (201136)

The above statement made is correct to the best of my knowledge.

Dr. Nishant Sharma

(Asst. Professor CSE/IT),

Computer Science & Engineering and Information Technology Jaypee University of Information Technology, Waknaghat,

ACKNOWLEDGEMENT

Firstly, we express our heartiest thanks and gratefulness to almighty God for His divine blessing that makes it possible for us to complete the project work successfully.

We are really grateful and wish our profound indebtedness to our Supervisor **Dr. Nishant Sharma (Asst. Professor CSE/IT)**, Department of CSE Jaypee University of Information Technology, Waknaghat. Deep Knowledge & keen interest of our supervisor in the field of “**A Shuttlecock and Player service fault detection**” helped us to carry out this project. Her endless patience, scholarly guidance, continual encouragement, constant and energetic supervision, constructive criticism, valuable advice, reading many inferior drafts and correcting them at all stages has made it possible for us to complete this project.

We would like to express our heartiest gratitude to **Dr. Nishant Sharma (Asst. Professor CSE/IT), Department of CSE**, for her kind help to finish our project.

We would also generously welcome each one of those individuals who have helped us straightforwardly or in a roundabout way in making this project a win. In this unique situation, we might want to thank the various staff individuals, both educating and non-instructing, which have developed their convenient help and facilitated our undertaking.

Finally, we must acknowledge with due respect the constant support and patience of our parents.

Mr. Kartik Parihar (201177)

Mr. Sayam Puri (201136)

TABLE OF CONTENTS

TITLE	PAGE NO.
DECLARATION	I
CERTIFICATE	II
ACKNOWLEDGEMENT	III
TABLE OF CONTENTS	IV
LIST OF FIGURES	V-VI
LIST OF ABBREVIATIONS	VII
ABSTRACT	VIII
CHAPTER-1: INTRODUCTION	1-4
CHAPTER-2: LITERATURE SURVEY	5-12
CHAPTER-3: SYSTEM DEVELOPMENT	13-41
CHAPTER-4: TESTING	42-46
CHAPTER-5: RESULTS AND EVALUATION	47-53
CHAPTER-6: CONCLUSION	54-58

List of Figures

3.1	Labeling example	15
3.2	Installation of Tensorflow	16
3.3	Installation of Pytorch	16
3.4	Pytorch	17
3.5	Installations of OpenCV	17
3.6	YOLO Structure	18
3.7	Pseudo code for YOLO	18
3.8	Pseudo code for CNN	20
3.9	Kalman Filter	21
3.10	Pseudo code for Kalman Filter	22
3.11	Architecture of RNN	25
3.12	Model DESIGN	26
3.13	Dataset screenshot	27
3.14	Process of Deployment	33

3.15	Libraries(1)	33
3.16	Libraries(2)	34
3.17	Pytorch	34
3.18	Data transformation	35
3.19	List of directories	35
3.20	model	36
3.21	Epochs	37
3.22	Faster RNN	38
3.23	Lucas-Kanade Method	39
4.1	Benchmarking	39
5.1	confusion matrix	48
5.2	Precision recall curve	49
5.3	Shuttlecock tracking	50
5.4	F1 confidence curve	51
5.5	Width and instances	51
5.6	width and height of shuttle	52
5.7	precision recall curve	52
5.8	Recall Confidence Curve	53
5.9	Loss table	53

LIST OF ABBREVIATIONS

Sr. No.	ABBREVIATION	DEFINITION
1	CNN	Convolutional Neural Network
2	LSTM	Long Short-Term Memory
3	DeepID V3	Deep Identity V3 (a type of deep learning model for face recognition)
4	IEMOCAP	Interactive Emotional Dyadic Motion Capture (a database for emotion analysis)
5	MELD	Multimodal EmotionLines Dataset (a dataset for emotion recognition)
6	YOLO	You only look once
7	RNN	Recurrent Neural Network
8	ADFES-BIV	Audio-Visual Facial Expression Signals with Bilingual and Multicultural Valence and Arousal Labels (a dataset for facial expression analysis)
9	DCNN	Deep Convolutional Neural Network

ABSTRACT

The shuttlecock detection system involves the use of high-resolution cameras strategically positioned within the badminton court to capture real-time footage. Advanced image processing algorithms analyze the video frames to identify and track the trajectory of the shuttlecock during the game. By employing object detection and motion tracking methods, the system accurately traces the movement of the shuttlecock, enabling precise determination of its position and trajectory throughout the match. Player service fault detection models at the same time comprise Convolutional neural network (CNN), rule-based systems and image analysis so as to distinguish slight movements of players involved, their serves as well as court environment. This study aims at developing an accurate and reliable algorithm through rigorous tests and subsequent adjustments for a system that can enhance training programmers, players' performance assessment, and general spectator experience during badminton sports. Shuttle Cock tracking is required for examining the trajectory of the shuttlecock. Player service fault analysis identifies service faults during badminton matches. The match point scored by players is analyzed by the first referee through shuttlecock landing point and player service faults. If the first referee cannot make a decision, they use technology such as a third umpire system to assist. The current challenge with the third umpire system is based on a high number of marginal errors for predicting match score. This research proposes a Machine Learning Framework to improve the accuracy of Shuttlecock Tracking and Player service Fault Detection.

CHAPTER - 1

INTRODUCTION

1.1) INTRODUCTION :

The way that technology and sports interact has evolved dramatically in recent years. ML frameworks are now integrated to improve performance and analysis across a range of sports disciplines. Another sport with a lot of energy and speed is badminton, where advances using ML algorithms have been made. The primary aim of utilizing machine learning in badminton is to optimize the precision and efficacy of shuttlecock movement detection during gaming. ML frameworks are capable of precisely tracking the trajectory, speed, and location of the shuttlecock on the court with the application of computer vision algorithms and neural networks. This real-time data helps with thorough performance analysis for players, coaches, and officials in addition to supporting refereeing choices. Detecting player service errors is yet another important use of machine learning in badminton. Referees frequently encounter service errors during fast-paced game play, such as unlawful serves that surpass the waist height or incorrect foot placement, which makes it difficult for them to correctly call these infractions. High-speed cameras and sensors combined with machine learning frameworks enable more accurate detection and analysis of these errors, giving authorities and players immediate feedback. In order to improve the accuracy of refereeing decisions, promote fair games, and offer priceless insights into player performance, machine learning frameworks for shuttlecock detection and player service fault detection have been integrated. This introduction lays the groundwork for an in-depth investigation of the approaches, formulas, and innovations in technology used to use machine learning. It uses neural networks and computer vision techniques to analyze shuttlecock speed, trajectory, and court location in real time.

1.2) PROBLEM STATEMENT:

In the domain of badminton, the accurate detection of shuttlecock movement trajectories and the identification of player service faults during game play pose significant challenges. Existing methods lack precision and real-time analysis capabilities, thereby impeding fair play

assessment, referee decision-making, and comprehensive performance evaluation for players and coaches. Therefore, there is a pressing need to develop and implement robust machine learning frameworks specifically tailored for shuttlecock detection and player service fault detection to enhance accuracy, efficiency, and reliability in these crucial areas of the sport.

Key challenges:

1. Shuttlecock Detection:

- Existing methods for tracking shuttlecock movement lack accuracy and fail to provide real-time analysis, leading to inconsistencies in trajectory assessment.
- Difficulty in accounting for various factors like shuttlecock speed, court positioning, and player interactions during game play for precise detection.

2. Player service fault detection:

- Current techniques used for identifying service faults, such as illegal serves or improper foot positioning, rely heavily on human judgment and are prone to errors.
- The methods currently in use for detecting service errors, like illegal serves or incorrect foot placement, are highly dependent on human judgment and are prone to mistakes.

Objective:

1. **Performance Analysis:** To offer detailed and data-driven insights into player Performance, including shuttlecock trajectory, speeds, and player positioning During Service actions, facilitating targeted training and skill improvement.
2. **Real-Time Feedback:** To provide immediate feedback to players and officials during matches, allowing for quick decisions and on-the-spot corrections where Necessary.
3. **Skill Development:** To assist players in refining their service techniques and Strategies by identifying patterns of faults and weaknesses in their actions, Ultimately contributing to skill improvement.
4. **Enhanced Training:** To support coaches in developing effective training regimens based on comprehensive data analysis, enabling players and teams to reach their full Potential.
5. **Fair Play:** To promote the principles of fair play and sportsmanship by ensuring that all players adhere to the rules and regulations, creating a level playing field for Both sides.

1.5 Significance and motivation of the project work

Enhancing accuracy in refereeing decision:

1. **Fair Play assurance:** To ensure fair play in badminton matches, machine learning frameworks for shuttlecock detection and player service defect detection must be implemented.
2. **Maintaining Game Integrity:** Maintaining the integrity of the game depends on the accurate identification of shuttlecock motions and service issues. It guarantees that all players have an equal opportunity to win and avoids unfair advantages, so talent and strategy win out over sneaky violations to decide matches.
3. **Comprehensive Performance Evaluation:** These machine learning frameworks offer coaches and athletes insightful information. Precise monitoring of shuttlecock trajectories and fault detection enable comprehensive performance analysis, which helps coaches design training plans for growth and players hone their skills.
4. **Empowering Referees:** Referees can make quick, well-informed decisions with the help of ML-based technologies. By offering real-time data and analysis, the frameworks serve as a support system that lessens the influence of human mistake in match officiating and relieves the stress of making subjective calls.
5. **Technological Advancements in Sports:** One example of how technology is revolutionizing sports is the use of machine learning in badminton. It demonstrates how cutting-edge technology may transform conventional sports and open the door for more advanced, data-driven methods of officiating and game analysis.
6. **Encouraging Innovation in Sports Technology:** The exploration of machine learning frameworks for the identification of shuttlecocks and player service faults, this program aims to promote the continuous study and development of sports technology. It stimulates creativity and encourages cooperation between the tech and sports sectors, which could have an impact on developments in other sports-related fields.
7. **Conclusion:** The significance and motivation behind implementing machine learning frameworks for shuttlecock detection and player service fault detection extend beyond immediate gameplay improvements. They encompass fair play assurance, performance enhancement, technological advancements in sports, and the potential for stimulating further innovation in sports technology. This section outlines the substantial significance

and motivational aspects of utilizing machine learning frameworks for shuttlecock detection and player service fault detection in badminton, emphasizing their multifaceted impacts on game play, performance analysis, officiating, and technological advancements in sports.

CHAPTER -2

LITERATURE SURVEY

1) AKSHAY MENON [2022/IEEE][1]

The research paper titled "**A machine learning framework for shuttlecock detection and player service fault detection**" explores the field of badminton technology with the goal of introducing machine learning techniques to transform the game. It intends to close this gap by describing the developments in badminton machine learning, the current constraints, and the possible improvements and additions anticipated from a specialized framework designed for shuttlecock detection and player service defect detection in this dynamic sport.

Advantages:

1. It provides accurate, real-time analysis that can reduce human error in spotting shuttlecock motions and player service errors, perhaps leading to a major improvement in referee decision-making.
2. It effectively identifies and resolves service issues, preventing unfair advantages, and fostering fairness in badminton matches, it helps to guarantee fair play and preserve the integrity of the game.
3. It gives coaches and players insightful information that helps with strategy building, performance analysis, and customized training plans based on precise data from shuttlecock trajectories and errors.
4. It raises the bar for technology developments in sports by demonstrating how Computer vision and machine learning are integrated into a fast-paced, dynamic computer.

Disadvantages:

1. There may be technical difficulties in the development and implementation of a strong machine learning framework for shuttlecock detection and player service fault detection, such as model training, data acquisition, and computational requirements.

2. The quality and availability of annotated data for training machine learning models is critical to the framework's success. A biased or inadequate dataset may affect the framework's generalizability and accuracy.
3. Real-time implementation during live matches may present challenges for the framework, requiring high-speed processing and flexibility to adjust to changing game scenarios and lighting conditions.
4. It may take some time and numerous trials and demonstrations to validate the framework's efficacy and dependability in real-world game settings and win over the badminton community, referees, and governing bodies.

2) Muhammad Amir Asari [2022] [2]

The research paper titled "**Vision Based Automated Badminton Action Recognition Using the New Local Convolutional Neural Network Extractor**" demonstrates the unwavering quest for innovations with the goal of improving player performance analysis, coaching techniques, and fan and spectator viewing experiences. This introduction, which lays the groundwork for a thorough examination of this innovative technology, represents a revolutionary advance in badminton technology and holds the potential to provide a more accurate and nuanced comprehension of player behavior inside the dynamic badminton court.

Dataset: CNN, SVM, Number of training sample=1794, Number of testing Sample=598
Number of validation sample=598

Advantage:

1. By efficiently capturing local spatial features in the game, a new local CNN extractor can improve the accuracy of badminton action recognition and provide more accurate classification of various actions such as smashes, serves, and drops.
2. The network can better capture minute details pertinent to various actions without being swayed by irrelevant information thanks to the local CNN extractor, which enables focused feature extraction from particular regions of interest within badminton frames.

3. The system's resilience to spatial variations in player positioning, changing camera angles, or different court perspectives frequently seen in badminton matches can be improved through localized feature extraction
4. Using a new local CNN extractor could speed up processing and provide real-time action recognition, allowing instant feedback for refereeing, coaching, and player analysis during live matches.

Disadvantages:

1. Refining a new local CNN architecture for action recognition necessitates complex network architecture and training protocols, which may require a large investment of time and knowledge.
2. When we use a local CNN extractor, the computational cost might be higher than with conventional CNN architectures, which could result in higher hardware requirements or longer inference times.
3. To train and tune the local CNN extractor, a comprehensive and precisely annotated dataset unique to badminton actions is required, and obtaining such datasets can be expensive and time-consuming.
4. The novel local CNN extractor may encounter difficulties stemming from over-fitting to particular training data, resulting in diminished capacity for generalization when confronted with novel or unseen action variations or conditions.

3) **KEISUKE FUJII[2022/IEEE][3]:**

The research paper titled "**Deep Reinforcement Learning in a Racket Sport for Player Evaluation with Technical and Tactical Contexts**" investigates the application of DRL techniques, a subset of machine learning, to delve into the complexities of player evaluation. Technical skills like footwork and stroke accuracy are important, but so comprehend the tactical situations in which players make choices during game play.

Advantages:

1. DRL enables a more thorough assessment of player performance by taking into account both technical proficiency (e.g., strokes, footwork) and tactical decision-making (e.g., shot selection, positioning).

2. Personalized assessments that take into account each player's unique playing style, flexibility, and ability to make strategic decisions are made possible by DRL models' ability to adapt and learn from player interactions.
3. Real-time analysis during game play can be provided by DRL-based evaluation systems, giving coaches and players quick feedback that allows for tactical and on-the-spot adjustments.
4. By revealing subtle patterns and tactics that conventional evaluation techniques might miss, DRL can provide deeper understanding of player performance and decision-making processes.
5. DRL-based assessments can help players improve their skills by pinpointing areas that need work and offering customized recommendations to improve their tactical and technical abilities.

Disadvantages:

1. Using DRL models for player evaluation in racket sports necessitates a large investment of computational power, specialized knowledge, and laborious training procedures.
2. To learn subtle player behaviors, effective DRL models need large and diverse datasets, which can be hard to come by in some racket sports contexts.
3. It can be difficult to make sure DRL models generalize well across various playing styles, skill levels, or game scenarios, which affects the model's dependability and adaptability.
4. Ethics pertaining to player consent, data privacy, and fair use of the information gathered must be taken into account when gathering and evaluating player data for DRL-based assessments.
5. DRL-based assessments depend on the model's performance, accuracy, and flexibility, and poor models may produce evaluations that are not accurate.

4) CHING-SHENG LIN AND HAN-YI HSIEH [2022/IEEE][4]:

The research paper titled "**An Automatic Defect Detection System for Synthetic Shuttlecocks Using Transformer Model**" is a Transformer model-based automated defect detection system for synthetic shuttlecocks has the potential to completely

transform the way badminton equipment is manufactured, especially in terms of quality control procedures. To detect flaws in synthetic shuttlecocks, this system makes use of cutting-edge machine learning methods, particularly the Transformer architecture, which is renowned for its competence in sequence-to-sequence tasks.

Advantages:

1. The attention mechanism of the Transformer model makes it possible to recognize complex patterns in shuttlecock images, which improves the system's ability to identify flaws like uneven feather alignment or inconsistent material.
2. Transformers are particularly good at handling sequential data, which is important when analyzing shuttlecock images because surface anomalies or the spatial arrangement of feathers require sequential understanding.
3. Transformer models have the potential to decrease reliance on heavily annotated datasets by enabling the use of unlabeled or partially labeled shuttlecock images for pre-training in an unsupervised manner.
4. The Transformer's innate capacity to adapt to various defect kinds and shuttlecock variations makes its scalable implementation possible in a variety of manufacturing environments.

Disadvantages:

1. Transformers require significant computational resources during training and inference due to their self-attention mechanism and larger parameter sizes, which may result in longer processing times.
2. The model's performance is strongly dependent on having access to a large, varied dataset that represents a range of shuttlecock defects, which can be difficult to obtain.
3. Due to their complexity, transformer models can be difficult to interpret and comprehend the decision-making process, which makes it more difficult to identify the precise features that contribute to defect detection.
4. Adjusting the process of developing a system is complicated by the need for careful optimization and hyper parameter tuning when using transformer models to detect defects in shuttlecock.

5. Because Transformer models are trained on large datasets, strict adherence to data privacy and ethical considerations regarding the collection and use of potentially sensitive manufacturing data are required.

5) Yuka Nokihara[2023/NIH][5]:

The research paper titled "**Future Prediction of Shuttlecock Trajectory in Badminton Using Player's Information**" has the ability to forecast a badminton shuttlecock's trajectory using player-specific data represents a noteworthy development in sports technology and has the potential to improve training methods, game play analysis, and strategic planning. This predictive model seeks to predict the shuttlecock's path by taking into account a number of player-related and contextual parameters.

Advantages:

1. Personalized trajectory predictions that are tailored to individual players are made possible by integrating player-specific data, such as playing style, stroke history, body posture, and movement patterns. This allows for more accurate insights.
2. The purpose of strategic planning, opponent analysis, and decision-making during matches, coaches, analysts, and players themselves can gain valuable insights by forecasting shuttlecock trajectories that take into account players' tendencies and game situations.
3. Precise trajectory forecasts grounded in players' data enable focused training regimens and skill enhancement by identifying opportunities to enhance stroke performance or placement.
4. Real-time shuttlecock trajectory prediction during games allows for quick feedback, which helps players, coaches, and referees better comprehend the dynamics of the game.
5. By revealing more about the players' strategies and tactics, trajectory predictions during televised matches improve the spectator experience.

Disadvantage:

1. Advanced data collection, processing, and integration are required to incorporate different player-related parameters and contextual factors into predictive models, which may result in complex modeling.
2. Strong generalization abilities are needed to guarantee the model's accuracy and dependability across a range of playing styles, skill levels, and game scenarios, which may be difficult to accomplish
3. Clear moral decisions about data privacy, consent, and fair usage of personal or performance-related data are necessary when using player information for trajectory predictions.

6) Jiatong Liu[2022/IEEE][6]:

The research paper titled "**An Action Recognition Technology for Badminton Players Using Deep Learning**" is action recognition technology for badminton players represents a revolutionary step forward in sports technology, with potential benefits for player development, coaching techniques, and performance analysis. The goal of this technology is to precisely identify and classify different actions that badminton players make during matches by using Deep Learning techniques.

Advantages:

1. Deep Learning models are particularly good at identifying intricate patterns, which makes it possible to precisely identify a variety of badminton-specific actions, including smashes, drops, clears, and serves.
2. The technology allows for real-time analysis, offering immediate feedback to players and coaches during matches or training sessions, aiding in performance improvement.
3. Deep Learning-based action recognition provides a comprehensive evaluation of player performance, going beyond basic metrics and encompassing diverse playing styles and techniques.

Disadvantages:

1. Deep Learning models necessitate large, diverse, and well-annotated datasets, which might be challenging to acquire, particularly for specific and diverse badminton actions.
2. Developing and training Deep Learning models for action recognition in badminton involves intricate architectures and longer training times, demanding significant computational resources.
3. Ensuring the model's accuracy across various playing conditions, skill levels, and game scenarios poses challenges in generalization and adaptation.
4. Deep Learning models frequently have interpretability issues, which makes it difficult to comprehend the logic behind the model's conclusions and may make users less likely to accept the model.

Chapter 3

System Development

3.1) Requirements for project:

Programming language: Python

Libraries:

Pandas: Pandas is a powerful Python library used for data manipulation and analysis. It provides data structures and functions to work with structured data, primarily in the form of DataFrames.

Key Features:

1. **DataFrames:** Pandas DataFrame is a two-dimensional labeled data structure with columns of potentially different types, similar to a spreadsheet or SQL table.
2. **Data Manipulation:** Pandas offers a wide range of functions for data manipulation tasks such as indexing, slicing, filtering, merging, joining, and reshaping data.
3. **Data Input/Output:** Pandas supports reading and writing data from various file formats including CSV, Excel, SQL databases, JSON, and more.
4. **Data Analysis:** Pandas facilitates descriptive statistics, data aggregation, grouping, pivot tables, and time series analysis.

Matplotlib: Matplotlib is a comprehensive plotting library for creating static, interactive, and animated visualizations in Python. It provides a MATLAB-like interface for generating plots and charts.

Key Features:

1. Matplotlib supports various plot types including line plots, scatter plots, bar plots, histograms, pie charts, contour plots, and more.
2. Users can customize every aspect of a plot, including colors, labels, markers, legends, titles, axis limits, and annotations.
3. Matplotlib allows users to create multiple plots within a single figure using subplot grids.
4. Plots can be saved in different file formats such as PNG, PDF, SVG, and EPS.

OpenCV-Python: OpenCV (Open Source Computer Vision Library) is a popular open-source computer vision and machine learning software library. OpenCV-Python is the Python binding for OpenCV, providing easy-to-use functions for image and video processing tasks.

Key Features:

1. OpenCV-Python offers functions for image reading, writing, manipulation, filtering, transformation, and enhancement.
2. OpenCV includes pre-trained models and algorithms for object detection, face recognition, feature detection, and motion analysis.
3. OpenCV enables video input/output, video streaming, frame extraction, and video analysis tasks.
4. OpenCV integrates with machine learning libraries like scikit-learn and TensorFlow for training and deploying computer vision models.

Jupyter: Jupyter is an open-source web application that allows users to create and share documents containing live code, equations, visualizations, and narrative text. It supports various programming languages, including Python, R, Julia, and Scala.

Key Features:

1. Interactive Computing: Jupyter provides an interactive computing environment where users can write and execute code, visualize results, and annotate findings in a single document.
2. Notebook Interface: Jupyter Notebooks consist of cells that can contain code, markdown text, equations, or raw text. Users can execute individual cells or the entire notebook.
3. Visualization Integration: Jupyter supports inline plotting and rendering of visualizations from libraries like Matplotlib, making it suitable for data exploration and analysis.
4. Collaboration and Sharing: Jupyter Notebooks can be easily shared and collaborated on via email, GitHub, or the Jupyter Notebook Viewer.

LabelImg: LabelImg is an open-source graphical image annotation tool used for manually labeling objects in images for tasks such as object detection, image segmentation, and classification.

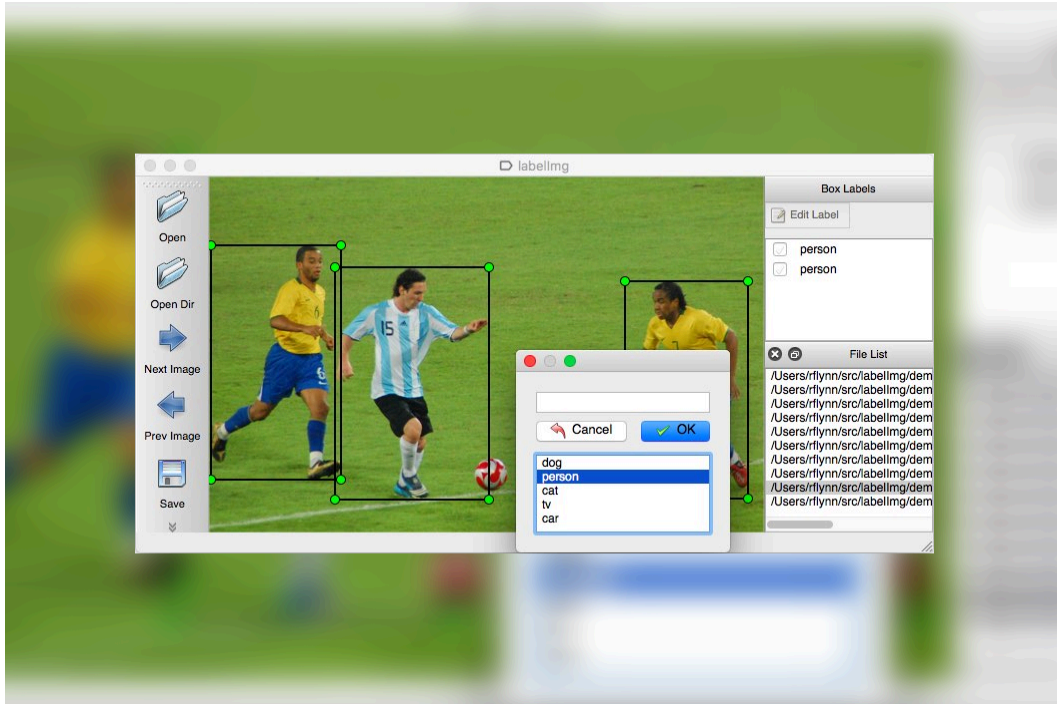


Fig 3.1 Labeling Example

Key Features:

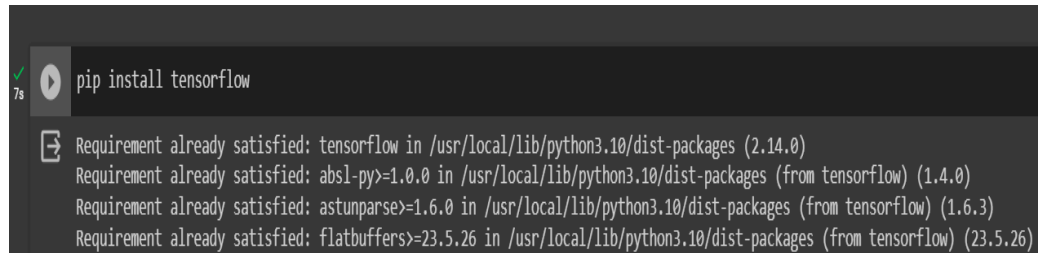
1. Labeling provides a user-friendly interface for drawing bounding boxes around objects of interest in images and assigning class labels to them.
2. Supported Formats: LabelImg supports common image formats such as JPEG, PNG, BMP, and TIFF for both input and output.
3. Customization: Users can customize class labels, colors, and keyboard shortcuts according to their annotation requirements.
4. Exporting Annotations: Annotations can be exported in various formats, including PASCAL VOC XML, YOLO text format, and TensorFlow Object Detection API TFRecord format.

Framework for project :

1. **Tensor Flow:** Tensor Flow is an open-source machine learning framework developed by Google for building and training various machine learning models. It offers a flexible ecosystem with comprehensive tools and libraries designed to perform a wide range of

tasks in machine learning, including deep learning, neural networks, natural language processing, computer vision, and more.

We can install TensorFlow using pip for Python by running:

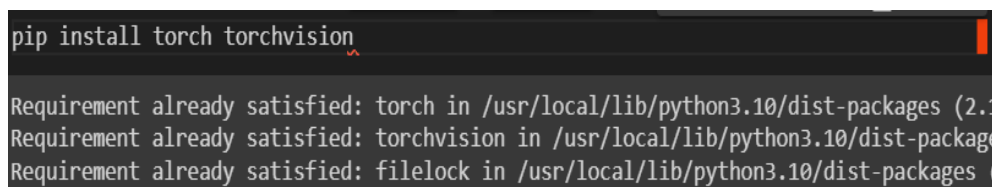


```
pip install tensorflow  
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.14.0)  
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)  
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)  
Requirement already satisfied: flatbuffers>=23.5.26 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.5.26)
```

Fig 3.2- Installation of tensor flow

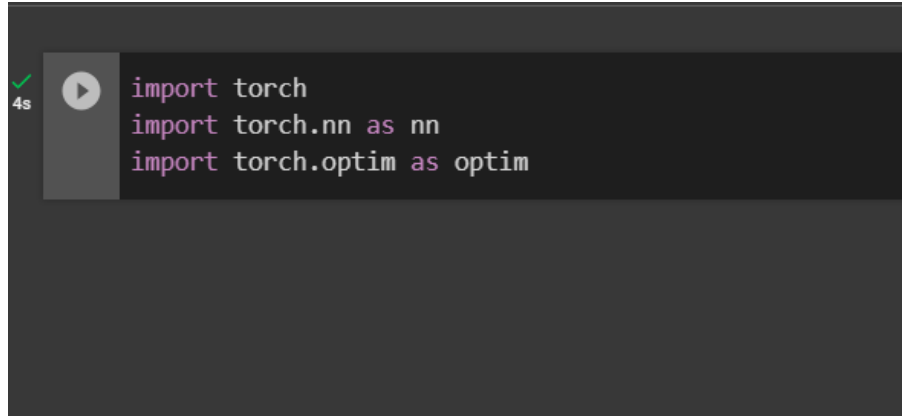
2. **Pytorch:** PyTorch is an open-source machine learning framework primarily developed by Facebook's AI Research lab (FAIR). It's widely used for building and training machine learning models, especially in the domain of deep learning. PyTorch provides a flexible platform that enables researchers and developers to create and experiment with complex neural network architectures efficiently.

We can install Pytorch using pip for Python by running:



```
pip install torch torchvision  
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.1  
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packag  
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (
```

Fig 3.3 - Installation of PyTorch

A terminal window with a dark background. On the left, there is a green checkmark and a play button icon, with '4s' below it. The terminal text shows three lines of Python code:

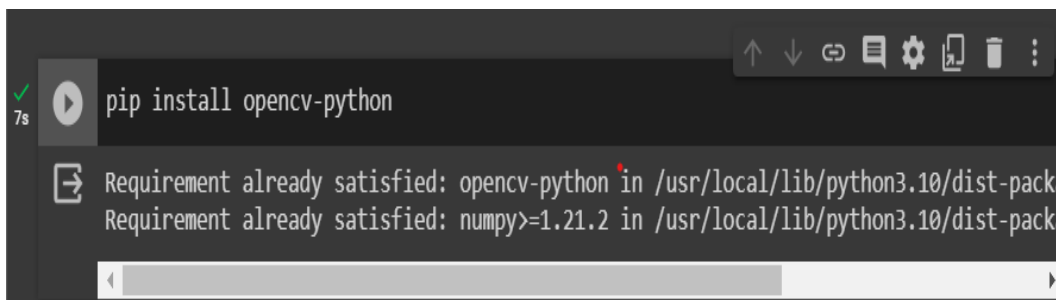
```
import torch
import torch.nn as nn
import torch.optim as optim
```

Fig 3.4- PyTorch

Computer Vision Libraries:

1. **OpenCV:** OpenCV (Open Source Computer Vision Library) is an open-source computer vision and machine learning software library designed to provide a common infrastructure for computer vision applications. It offers a wide range of functionalities to process visual data, including image and video analysis, object detection, facial recognition, feature detection, and more. OpenCV is written in C++, but it also provides Python bindings for easier integration with Python-based applications.

We can install opencv using pip for Python by running:

A terminal window with a dark background. On the left, there is a green checkmark and a play button icon, with '7s' below it. The terminal text shows the command `pip install opencv-python` and its output:

```
Requirement already satisfied: opencv-python in /usr/local/lib/python3.10/dist-packa
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packa
```

Fig 3.5 - Installation of OpenCV

Object Detection Models :

1. **YOLO (You only look once):** YOLO (You Only Look Once) is a real-time object detection algorithm that stands out for its speed and accuracy in detecting objects in images or video frames. Unlike traditional methods that use region proposal networks and sliding windows to identify objects, YOLO divides the input image into a grid and

predicts bounding boxes and class probabilities directly from the grid cells. YOLO performs predictions for all grid cells in a single forward pass of the neural network, making it extremely fast.

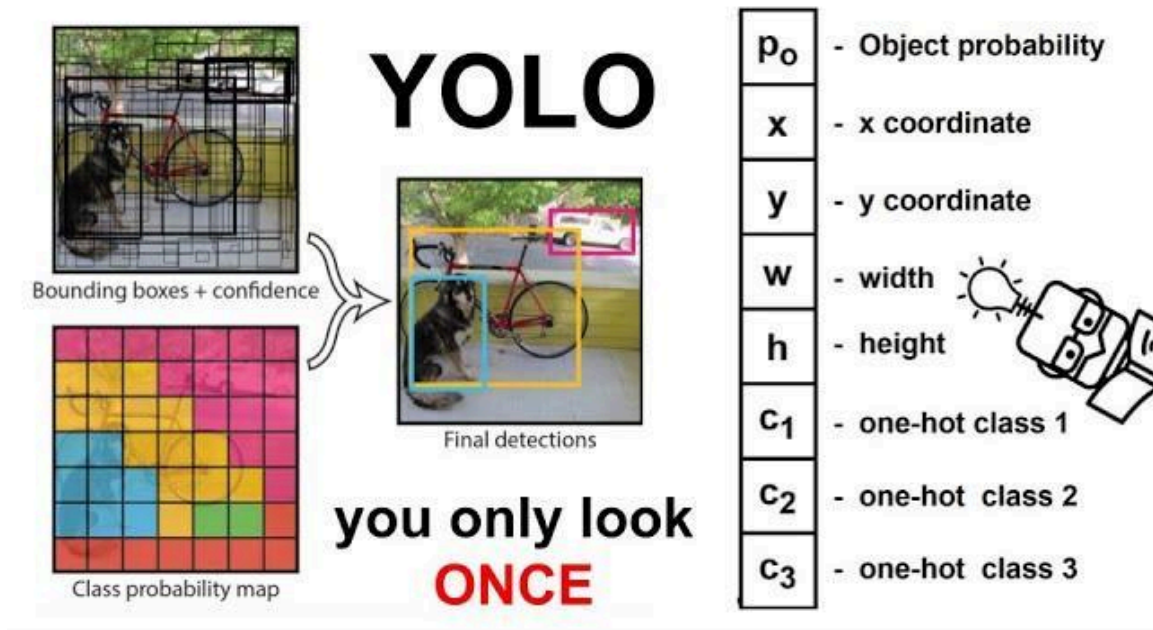


Fig 3.6 YOLO Structure

Pseudo code for YOLO is:

```

1 # Load YOLO model and configurations
2 yolo_model = load_yolo_model('yolo_weights', 'yolo_config', 'yolo_classes')
3
4 # Preprocess input data
5 input_image = preprocess_image('input_image.jpg', target_size=(416, 416))
6
7 # Perform object detection
8 detections = yolo_model.predict(input_image)
9
10 # Post-process detections
11 filtered_detections = filter_predictions(detections, confidence_threshold=0.5, nms_threshold=0.4)
12
13 # Visualize and display detected objects
14 for detection in filtered_detections:
15     class_id, class_name, confidence, bbox = detection
16     draw_bbox_on_image(input_image, bbox, class_name)
17     print(f"Detected: {class_name} with confidence: {confidence}")
18
19 display_image(input_image)

```

Fig 3.7 – Pseudo code for YOLO

Explanation of pseudo code steps:

1. **Load YOLO Model:** Load the pre-trained YOLO model with its weights, configuration, and classes.
2. **Preprocess Input:** Prepare the input image by resizing and formatting it according to the YOLO model's input size requirements.
3. **Perform Object Detection:** Pass the preprocessed image through the YOLO model to obtain raw detections (bounding boxes, class probabilities, etc.) for potential objects
4. **Post-process Detections:** Apply filtering to the raw detections to remove low-confidence predictions and perform non-maximum suppression (NMS) to eliminate redundant bounding boxes.
5. **Visualize Detected Objects:** Iterate through the filtered detections, draw bounding boxes around detected objects on the input image, and display the class names and confidence scores.

CNN: CNN stands for Convolutional Neural Network, which is a specialized deep learning architecture primarily used for analyzing visual data like images and videos. CNNs are designed to automatically and adaptively learn spatial hierarchies of features from the input data through the application of Convolutional layers, pooling layers, and fully connected layers.

CNN architecture consists of multiple layers:

1. **Convolutional Layers:** These layers apply filters (kernels) to the input data, performing convolution operations to extract spatial features.
2. **Pooling Layers:** Pooling layers down sample the feature maps obtained from convolution, reducing dimensionality while retaining important information.
3. **Activation Functions:** Non-linear activation functions like ReLU (Rectified Linear Unit) introduce non-linearity, enabling the network to learn complex patterns.
4. **Fully Connected Layers:** These layers process the extracted features and make predictions by connecting all neurons to the subsequent layers.

```

1 # Define CNN architecture
2 Initialize CNN model architecture:
3 - Convolutional layer with 'n' filters, kernel size, activation function
4 - Pooling layer (e.g., MaxPooling2D)
5 - Flatten layer to convert 2D feature maps to a 1D vector
6 - Fully connected (Dense) layers with activation functions
7 - Output layer for prediction (e.g., Softmax for classification)
8 Compile the CNN model:
9 - Define optimizer (e.g., Adam, SGD)
10 - Define loss function (e.g., categorical_crossentropy)
11 - Specify evaluation metrics [(e.g., accuracy)]
12 for each epoch in range(num_epochs):
13     for each mini-batch in training_data:
14         # Forward pass
15         Perform convolution, pooling, activation on the input batch
16         Flatten the output and pass through fully connected layers
17         Obtain predictions from the output layer
18         # Calculate loss
19         Compute loss between predictions and ground truth labels
20         # Backpropagation
21         Compute gradients using backpropagation
22         Update weights and biases using optimize
23 # Make predictions
24 For new_input_data in unseen_data:
25     Use the trained CNN model to predict the class or label
26

```

Fig 3.8- Pseudo code for CNN

Explanation of the pseudo code steps:

1. **Model Definition:** Initialize the CNN architecture by defining the layers, their parameters (filters, kernel size, activation functions), and the model's structure.
2. **Model Compilation:** Compile the model by specifying the optimizer, loss function, and evaluation metrics to be used during training.
3. **Training Loop:** Perform training using mini-batch gradient descent for a specified number of epochs. Execute forward pass, compute loss, perform back propagation, and update model weights.
4. **Model Evaluation:** Evaluate the trained model's performance on a separate test/validation dataset to measure its accuracy or other metrics.
5. **Prediction:** Use the trained CNN model to make predictions on new, unseen input data.

Object tracking algorithm:

Kalman Filter: Kalman Filter is a recursive mathematical algorithm used to estimate the state of a dynamic system by processing a series of noisy measurements over time. It is widely used in various fields, including control systems, navigation, signal processing, and robotics, to estimate unknown variables based on noisy observations.

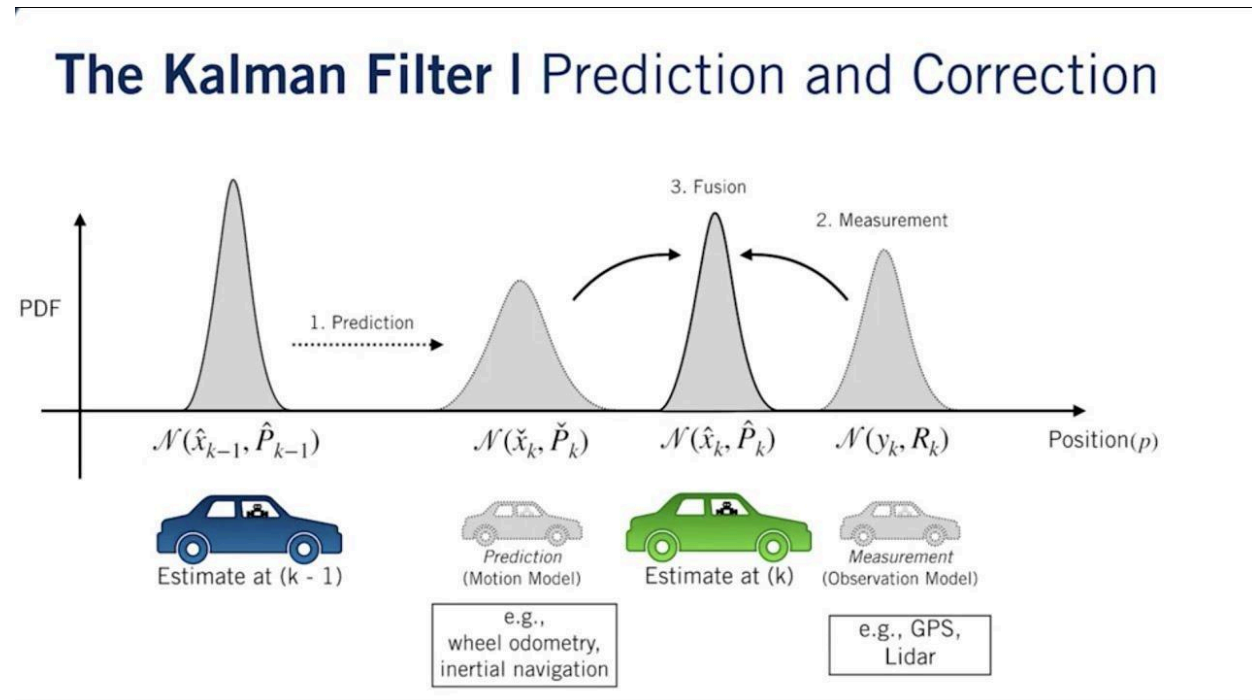


Fig 3.9 Kalman Filter

Key Concepts of Kalman Filter:

- 1. State Estimation:** Kalman Filter estimates the state of a system by combining predictions from the dynamic model (predict step) and measurements from sensors (update step).
- 2. Prediction Step:** Predict the next state of the system based on the system's dynamics using a state transition model (e.g., motion model) and the previous state estimate.
- 3. Update Step:** Correct the predicted state using noisy measurements obtained from sensors, weighing the prediction and measurement according to their respective uncertainties.

Pseudo code for Kalman Filter:

```
1 # Initialize Kalman Filter variables and parameters
2
3
4 for each time step t:
5     # Prediction Step
6     state_prediction = state_transition_matrix * state_estimate + control_matrix * control_input
7     covariance_prediction = state_transition_matrix * covariance_estimate * state_transition_matrix' + process_noise
8
9     # Update Step (if measurement available)
10    if measurement_available_at_t:
11        kalman_gain = covariance_prediction * measurement_matrix' * inv(measurement_matrix * covariance_prediction)
12        state_estimate = state_prediction + kalman_gain * (measurement_at_t - measurement_matrix * state_prediction)
13        covariance_estimate = (identity_matrix - kalman_gain * measurement_matrix) * covariance_prediction
14    else:
15        # No measurement available, update using only prediction
16        state_estimate = state_prediction
17        covariance_estimate = covariance_prediction
18
19    # Use state_estimate for system control, navigation, or further processing
```

Fig 3.10- Pseudo code for Kalman Filter

Explanation of the pseudo code steps:

1. **Initialization:** Initialize the initial state estimate, covariance matrix (representing uncertainty), noise variances (process and measurement), state transition matrix, measurement matrix, and control matrices.
2. **Prediction Step:** Predict the next state of the system based on the state transition model and control input, along with estimating the covariance matrix for the predicted state.
3. **Update Step:** If a measurement is available at the current time step, perform the update step of the Kalman Filter. Calculate the Kalman gain, update the state estimate, and refine the covariance matrix based on the measurement.
4. **State Estimation Update:** Use the updated state estimate for system control, navigation, or further processing at the current time step.

Video Processing Libraries:

FFmpeg: FFmpeg is a powerful, open-source multimedia framework that provides a set of tools and libraries for handling multimedia data such as audio, video, and other multimedia files. It includes a vast collection of software programs and libraries for recording, converting, streaming, and manipulating multimedia content.

Key Features of FFmpeg:

1. **Codec Support:** FFmpeg supports a wide range of audio and video codecs, allowing users to encode, decode, transcode, and manipulate multimedia files.
2. **Format Conversion:** It can convert multimedia files between different formats, containers, codecs, and bitrates. For example, it can convert video files from one format to another (e.g., MP4 to AVI, MKV to MP3).
3. **Audio and Video Processing:** FFmpeg offers various filters and effects for processing audio and video, including resizing, cropping, scaling, filtering, overlaying, and more.
4. **Streaming:** It enables streaming of multimedia content over networks using various protocols, making it useful for live streaming applications.
5. **Recording and Capturing:** FFmpeg can record and capture audio and video from different input sources, such as cameras, microphones, and screen capture

Deep Learning Models:

1. **Recurrent Neural Networks (RNNs):** Recurrent Neural Networks (RNNs) are a class of artificial neural networks designed to work with sequential data by processing input sequences of varying lengths. Unlike feed forward neural networks where information flows in one direction (from input to output), RNNs have connections that form directed cycles, allowing them to retain and utilize information from previous time steps.

Key Features of Recurrent Neural Networks:

1. **Sequential Data Processing:** RNNs are specialized for sequential data such as time series, text, speech, and video data, where the order of elements matters.
2. **Temporal Dynamics:** RNNs maintain a hidden state that acts as a memory to retain information about previous inputs, enabling them to capture temporal dependencies.
3. **Variable Length Inputs:** They can handle inputs of varying lengths, making them suitable for tasks like natural language processing, speech recognition, and time series prediction.
4. **Vanishing/Exploding Gradient:** RNNs may suffer from the vanishing gradient problem (gradients become very small) or exploding gradient problem (gradients become very large) during training, impacting the learning of long-term dependencies.

Architecture of an RNN:

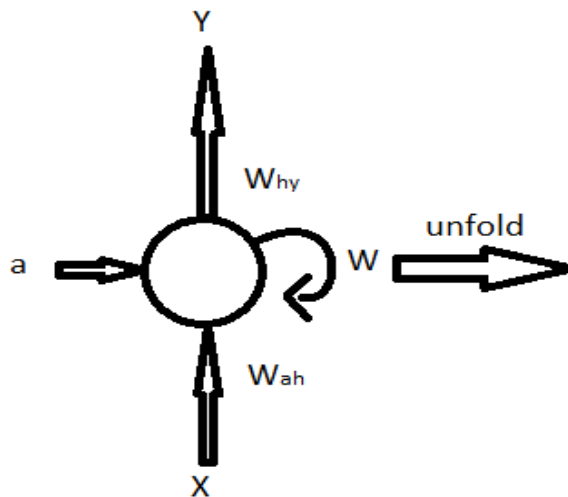


Fig 3.11 (a) Architecture of RNN

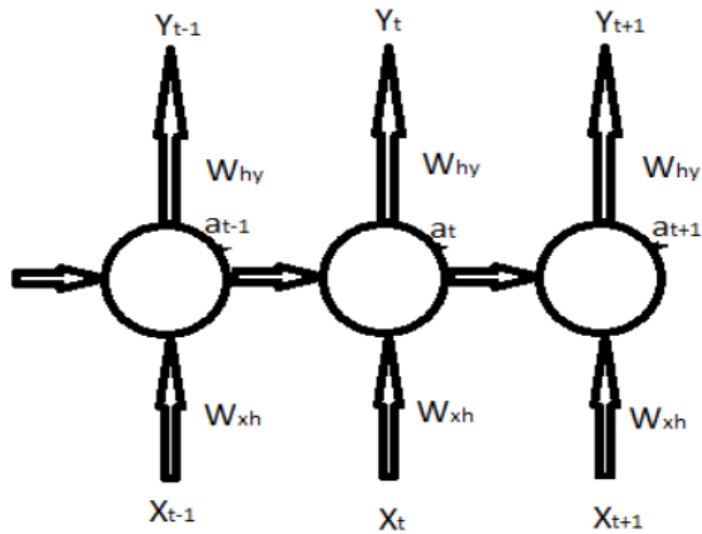


Fig 3.11 (b) Architecture of RNN

Usage of RNNs:

RNNs are applied in various domains for sequence modeling tasks such as:

1. **Natural Language Processing (NLP):** Language modeling, machine translation, text generation, sentiment analysis.
2. **Time Series Analysis:** Stock market prediction, weather forecasting, signals processing.
3. **Speech Recognition:** Speech-to-text conversion, voice activity detection.

Limitations:

1. **Short-Term Memory:** RNNs may struggle with capturing long-range dependencies due to vanishing/exploding gradient problems, limiting their ability to retain information over many time steps.
2. **Training Challenges:** Training RNNs can be computationally intensive and challenging due to issues like vanishing gradients, which can hinder learning.

3.2) Project Design and Architecture:

Designing a project for shuttlecock tracking or player service fault detection involves several key steps, from defining the objectives to implementing the system.

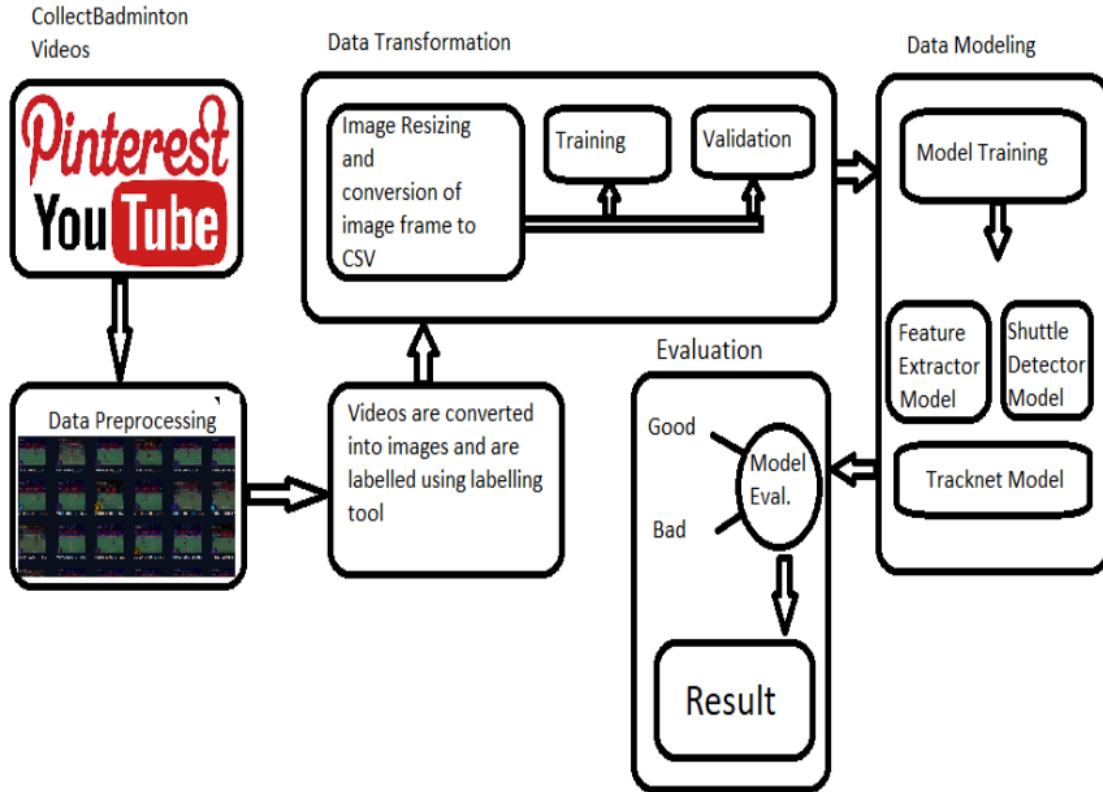


Fig 3.12 – Model design

- **Data Collection:**

1. Collect a large dataset of badminton game videos. This dataset should include various lighting conditions, different players, and a variety of shots.

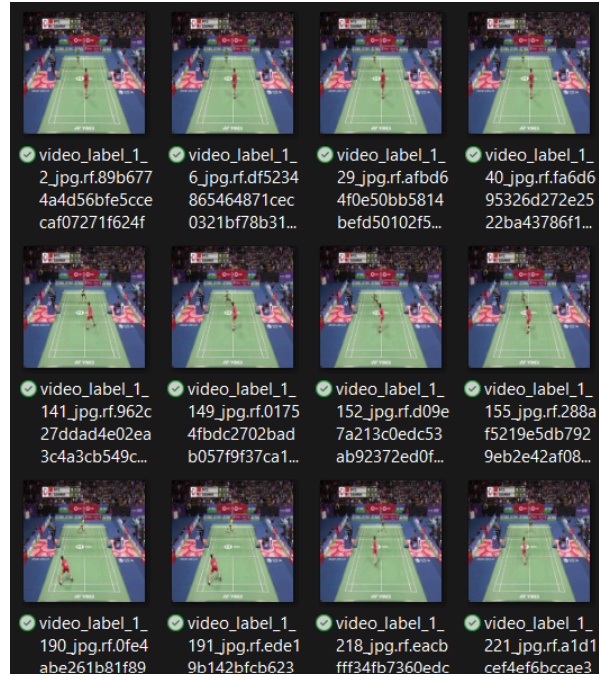


Fig 3.13-Dataset screenshot

Data Preprocessing:

1. Extract frames from the video dataset.
2. Label the dataset with annotations indicating shuttlecock positions, player actions, service lines, and any relevant labels for service fault detection.

1. **Video/Image Preprocessing:** Frame Extraction: If working with videos, extract individual frames from the video footage to process them individually.
2. **Frame Resizing:** Resize frames to a standardized size suitable for the tracking or detection algorithms to ensure uniform processing.
3. **Annotation Processing:** Annotation Verification: Ensure the accuracy of annotations by cross-verifying them with the actual video frames/images to avoid discrepancies.
4. **Data Augmentation:** Augment the dataset by applying transformations like rotation, scaling, flipping, or adding noise to enhance the model's robustness.

Feature Extraction:

1. Shuttlecock Localization: Use image processing techniques (e.g., edge detection, color segmentation) to isolate and extract features related to shuttlecocks' appearance, such as color, shape, or movement patterns.
2. Player Action Recognition: Extract features representing player actions (e.g., service motion, racket movements) for fault detection tasks.
3. Shuttlecocks often have distinctive colors, such as white or yellow, which can be exploited for detection. Extract color histograms or use color segmentation techniques to isolate regions in the image that match the color characteristics of shuttlecocks.
4. Consider using color spaces such as RGB, HSV, or Lab for better representation of color information and robustness to changes in lighting conditions.
5. Shuttlecocks typically have a circular or oval shape with a distinct outline. Extract shape features such as contours, circularity, eccentricity, or convexity to identify regions that resemble the shape of shuttlecocks.

6. Apply techniques like edge detection (e.g., Canny edge detector) to highlight boundaries and extract shape information from the image.
7. Shuttlecocks may have characteristic textures, such as the pattern of feathers or the surface texture of the cork base. Use texture descriptors like local binary patterns (LBP), gray-level co-occurrence matrices (GLCM), or Gabor filters to capture texture information that distinguishes shuttlecocks from the background.
8. Employ texture segmentation techniques to partition the image into regions with similar texture properties, potentially isolating regions corresponding to shuttlecocks.
9. Shuttlecocks may have characteristic textures, such as the pattern of feathers or the surface texture of the cork base. Use texture descriptors like local binary patterns (LBP), gray-level co-occurrence matrices (GLCM), or Gabor filters to capture texture information that distinguishes shuttlecocks from the background.
10. In videos or sequences of images, shuttlecocks exhibit motion patterns that can aid in detection. Use motion detection algorithms to track moving objects in consecutive frames and identify regions with significant motion.
11. Calculate optical flow to estimate the motion vectors of pixels between frames and identify regions where shuttlecocks are likely to be present based on their motion characteristics.

Data Splitting:

1. **Train-Validation-Test Split:** Divide the dataset into training and testing subsets. A typical split might be 70-30; ensuring models are trained, validated, and evaluated on different subsets of data.

Data Formatting:

1. **Sequence Creation:** For tracking tasks, organize sequential frames or time-series data to create sequences that represent shuttlecock movements or player actions.
2. **Input Formatting:** Convert the data into suitable formats compatible with the chosen tracking or detection algorithms (e.g., tensors for deep learning models).

Missing Data Handling:

1. **Data Cleaning:** Handle missing or corrupted data by imputation or removal to prevent errors during model training and inference.

Data Normalization and Standardization:

1. **Scaling:** Scale numerical features to a consistent range (e.g., using min-max scaling or z-score normalization) to facilitate model convergence and improve training efficiency.

Preprocessing for Specific Algorithms: Depending on the chosen algorithms (e.g., CNNs, RNNs, object tracking algorithms), preprocess data in a manner suitable for the algorithm's input requirements (e.g., image resizing, sequence creation).

Annotate the dataset with the following information:

1. Shuttlecock bounding box coordinates.
2. Player positions.
3. Serve fault labels (if available).

Data Transformation: In data transformation we have to convert our images into csv files and upload it in google collab.

Shuttlecock Tracking:

1. Use computer vision techniques (e.g., background subtraction, object detection) to track the shuttlecock position in each frame.
2. Implement an object tracking algorithm (e.g., Kalman filter, SORT) to improve tracking accuracy and handle occlusions.
3. Preprocess the collected data to enhance features and reduce noise. This may include resizing images, converting to grayscale, applying filters for noise reduction, and normalization.
4. Extract relevant features from the preprocessed images that can help distinguish shuttlecocks from the background and other objects. Common features might include color, shape, texture, and motion.

5. Split your dataset into training, validation, and test sets.
6. Train the detection model on the training set using labeled data (images with annotations indicating the location of shuttlecocks).

12). Player Service Fault Detection:

1. Design a deep learning model (e.g., CNN, RNN, or a combination) to detect player service faults.
2. Train the model using the annotated dataset, where labels indicate whether a server fault occurred.
3. Evaluate the model's performance on the test set, considering metrics such as accuracy, precision, recall, and F1-score.
4. Divide the dataset into training, validation, and test sets, maintaining consistency across different sets to ensure fair evaluation.
5. Consider using stratified sampling to maintain the distribution of classes (correct service vs. different types of faults) across the subsets.
6. Select relevant features that can effectively discriminate between correct services and service faults.
7. Experiment with different feature combinations of transformations that highlight key differences between fault and non-fault instances.

Integration:

1. Combine the shuttlecock tracking and service fault detection components into a unified framework.
2. Develop a user-friendly interface for users or referees to input videos and receive results.
3. Ensure real-time or near-real-time processing of video feeds during a live game.

Deployment:

1. Deploy the framework on appropriate hardware (e.g., servers or edge devices) for real-world usage.
2. Determine the platform on which the shuttlecock detection system will be deployed. This could be a desktop application, a mobile app, a web application, or an embedded system.
3. If the detection system processes live video streams (e.g., from a camera feed), integrate mechanisms to capture and feed the input data to the detection model.
4. For offline processing of pre-recorded videos or images, develop functionalities to upload or select input files for analysis.
5. Implement the inference pipeline to feed input data to the trained detection model and obtain predictions.
6. Optimize the inference process for efficiency and real-time performance, considering hardware constraints if applicable (e.g., GPU acceleration).
7. Apply post-processing techniques to refine the detection results, such as non-maximum suppression (NMS) to eliminate duplicate detections or filtering based on confidence scores.
8. Develop visualization components to display the detected shuttlecocks overlaid on the input video or image in real-time.

9. Include options to customize the visualization, such as highlighting detected shuttlecocks with bounding boxes or annotations.

DEPLOYMENT PROCESS

Application Deployment Process

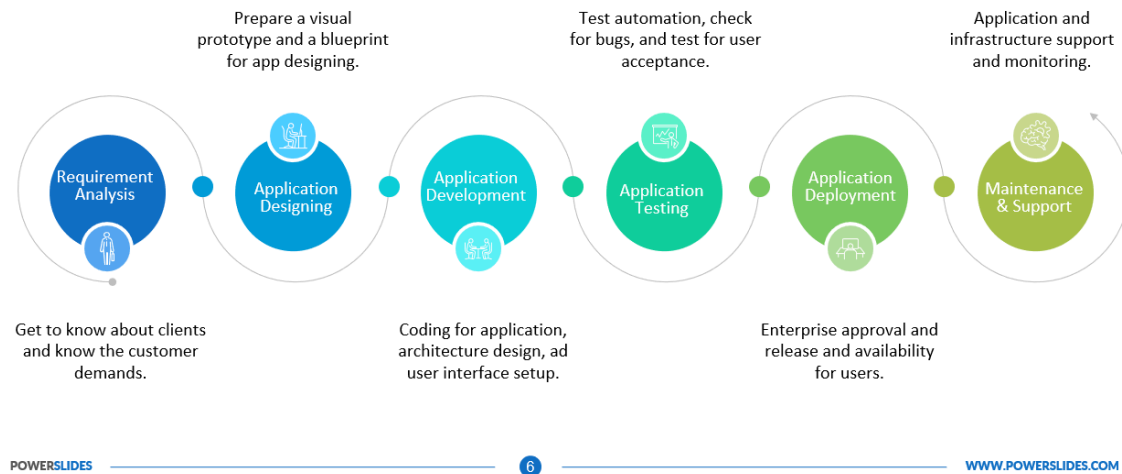


Fig 3.14 Process of deployment

3.3) Implementation of project:

1. Snippets of code:

Importing important libraries:

```

from google.colab import drive
drive.mount('/content/drive')

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[ ] import os
    os.chdir('/content/drive/MyDrive/yolo_train')

[ ] ls

yolo_training.ipynb  yolov5/

[ ] #!git clone https://github.com/ultralytics/yolov5.git

[ ] os.chdir('yolov5')

```

Figure 3.15-Libraries(1)

```

from __future__ import print_function, division
import torch
import torch.nn as nn
import torch.optim as optim
from torch.optim import lr_scheduler
import torch.backends.cudnn as cudnn
import numpy as np
import torchvision
from torchvision import datasets, models, transforms
import matplotlib.pyplot as plt
import time
import os
import copy

cudnn.benchmark = True
plt.ion()

```

Fig 3.16-Libraries(2)

```

!python export.py --weights runs/train/Model14/weights/best.pt --include torchscript onnx

```

export: data=data/coco128.yaml, weights=['runs/train/Model14/weights/best.pt'], imgsz=[640, 640], batch_size=1, device=cpu
YOLOv5 v7.0-304-g22361691 Python-3.10.12 torch-2.2.1+cu121 CPU

Fusing layers...
YOLOv5s summary: 157 layers, 7012822 parameters, 0 gradients, 15.8 GFLOPs

PyTorch: starting from runs/train/Model14/weights/best.pt with output shape (1, 25200, 6) (13.8 MB)

TorchScript: starting export with torch 2.2.1+cu121...
TorchScript: export success 5.4s, saved as runs/train/Model14/weights/best.torchscript (27.2 MB)

requirements: Ultralytics requirement ['onnx>=1.12.0'] not found, attempting AutoUpdate...
Collecting onnx>=1.12.0
 Downloading onnx-1.16.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (15.9 MB)
 15.9/15.9 MB 84.5 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-packages (from onnx>=1.12.0) (1.25.2)
Requirement already satisfied: protobuf>=3.20.2 in /usr/local/lib/python3.10/dist-packages (from onnx>=1.12.0) (3.20.3)
Installing collected packages: onnx
Successfully installed onnx-1.16.0

Fig 3.17 -Pytorch

2. Data Transformation:

```
data_transforms = {
    'train': transforms.Compose([
        transforms.RandomResizedCrop(224),
        transforms.RandomHorizontalFlip(),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
    ]),
    'val': transforms.Compose([
        transforms.Resize(256),
        transforms.CenterCrop(224),
        transforms.ToTensor(),
        transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])]
)
data_dir = 'dataset'
image_datasets = {x: datasets.ImageFolder(os.path.join(data_dir, x), data_transforms[x])
                  for x in ['train', 'val']}
dataloaders = {x: torch.utils.data.DataLoader(image_datasets[x], batch_size = 4, shuffle = True, num_workers = 4)
              for x in ['train', 'val']}
dataset_sizes = {x: len(image_datasets[x]) for x in ['train', 'val']}
class_names = image_datasets['train'].classes
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
✓ 0.0s
```

Fig 3.18-Data transformation

```
ls
benchmarks.py  data/  export.py  __pycache__/  requirements.txt  tutorial.ipynb
CITATION.cff  data_images/  hubconf.py  pyproject.toml  runs/  utils/
classify/  data.yaml  LICENSE  README.md  segment/  val.py
CONTRIBUTING.md  detect.py  models/  README.zh-CN.md  train.py  yolov5s.pt

[ ] !pip install -r requirements.txt
```

Figure 3.19-List of directories

3. Function for training model:

```
def train_model(model, criterion, optimizer, scheduler, num_epochs=25):
    since = time.time()
    best_model_wts = copy.deepcopy(model.state_dict())
    best_acc = 0.0

    for epoch in range(num_epochs):
        print(f'Epoch {epoch}/{num_epochs - 1}')
        print('-'*10)

        for phase in ['train', 'val']:
            if phase == 'train':
                model.train()
            else:
                model.eval()

            running_loss = 0.0
            running_corrects = 0

            for inputs, labels in dataloaders[phase]:
                inputs = inputs.to(device)
                labels = labels.to(device)

                optimizer.zero_grad()

                with torch.set_grad_enabled(phase == 'train'):
                    outputs = model(inputs)
                    _, preds = torch.max(outputs, 1)
                    loss = criterion(outputs, labels)
```

```
                if phase == 'train':
                    loss.backward()
                    optimizer.step()

            running_loss += loss.item() * inputs.size(0)
            running_corrects += torch.sum(preds == labels.data)
        if phase == 'train':
            scheduler.step()
        epoch_loss = running_loss / dataset_sizes[phase]
        epoch_acc = running_corrects.double() / dataset_sizes[phase]
        print(f'{phase} Loss: {epoch_loss: .4f} Acc: {epoch_acc: .4f}')
        if phase == 'val' and epoch_acc > best_acc: ...
    print()
    time_elapsed = time.time() - since
    print(f'Training complete in {time_elapsed // 60:0f}m {time_elapsed % 60:0f}s')
    print(f'Best val Acc: {best_acc:4f}')
    model.load_state_dict(best_model_wts)
    return model
```

Fig 3.20-model

4. Training Model:

```
model_conv = torchvision.models.resnet18(pretrained=True)
for param in model_conv.parameters():
    param.requires_grad = False
num_fters = model_conv.fc.in_features
model_conv.fc = nn.Linear(num_fters, len(class_names))

model_conv = model_conv.to(device)
criterion = nn.CrossEntropyLoss()

optimizer_conv = optim.SGD(model_conv.fc.parameters(), lr=0.001, momentum =0.9)

exp_lr_scheduler = lr_scheduler.StepLR(optimizer_conv, step_size = 7, gamma = 0.1)

model_conv = train_model(model_conv,criterion, optimizer_conv,exp_lr_scheduler,num_epochs = 100)
```

5. Epoch:

```
Epoch 0/99
-----
train Loss:  2.0696 Acc: 0.3448
val Loss:   1.7286 Acc: 0.4821

Epoch 1/99
-----
train Loss:  2.0747 Acc: 0.3103
val Loss:   1.7704 Acc: 0.4821

Epoch 2/99
-----
train Loss:  2.0310 Acc: 0.3645
val Loss:   1.7670 Acc: 0.5000

Epoch 3/99
-----
train Loss:  2.0068 Acc: 0.3842
val Loss:   1.7599 Acc: 0.4643

Epoch 4/99
-----
train Loss:  2.0110 Acc: 0.3300
val Loss:   1.8211 Acc: 0.4464
...
val Loss:   1.7331 Acc: 0.4643

Training complete in 24.000000m 41.291286s
Best val Acc: 0.553571
```

Fig 3.21-Epochs

6. Algorithms used in projects:

For Shuttlecock Tracking:

Object Detection Algorithms:

1. **YOLO (You Only Look Once):** YOLO is a real-time object detection algorithm that can identify and track shuttlecocks in video frames with high accuracy.
2. **Faster R-CNN:** A region-based Convolutional neural network that can detect objects like shuttlecocks by proposing regions of interest and performing object localization and classification.

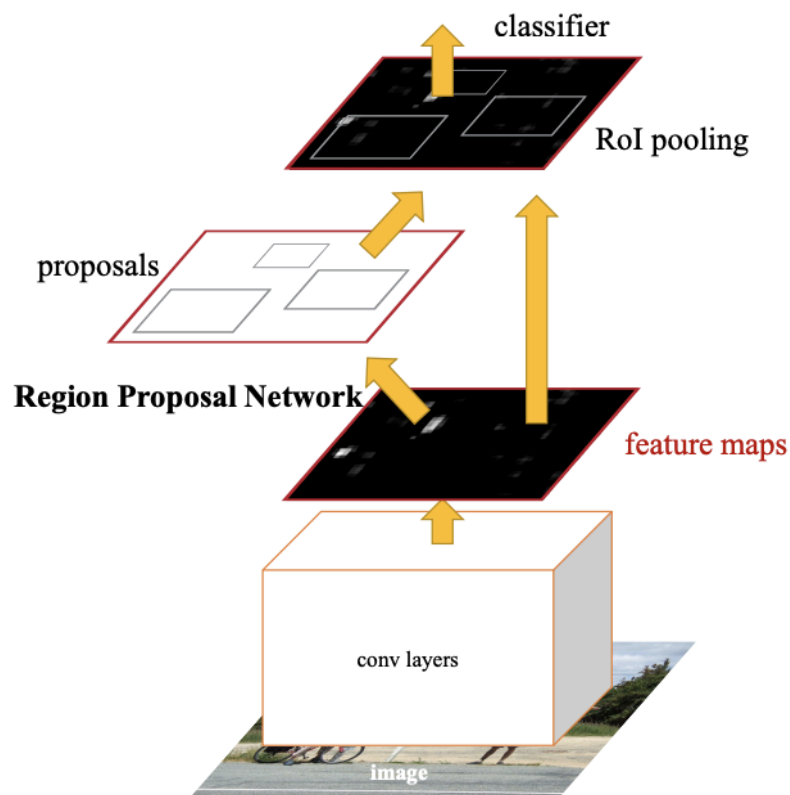


Fig 3.22- Faster RNN

Optical Flow:

1. **Lucas-Kanade Method:** An optical flow algorithm used for tracking object motion by analyzing pixel-level changes between consecutive frames, which can track shuttlecock movement.

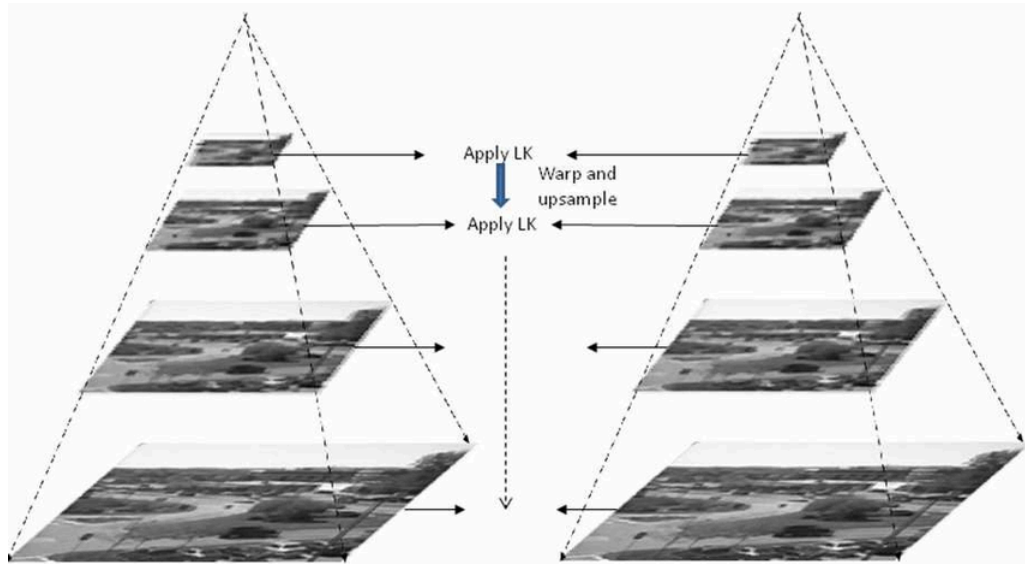


Fig 3.23 Lucas-Kanade Method

2. **Kalman Filters: Kalman Filters for Object Tracking** Kalman Filters can predict the future location of the shuttlecock based on its current position and velocity, often used to smooth and predict object trajectories.

Player Service Fault Detection:

Computer Vision Techniques:

1. **Image Processing:** Analyzing player positions, racket movements, and shuttlecock trajectory to detect inconsistencies or faults in service action.
2. **Template Matching:** Comparing specific service positions or patterns with a predefined template to detect service line violations.

7. Machine Learning/Deep Learning Models:

1. **Convolutional Neural Networks (CNNs):** Utilized for recognizing player actions, racket positions, or shuttlecock movements during service, aiding in fault detection based on learned patterns.

2. **Recurrent Neural Networks (RNNs):** Applied for sequence modeling of player actions during service, capturing temporal dependencies for fault detection.

8. **Rule-Based Systems:**

Expert Systems: Implementing predefined rules or logic based on the rules of badminton to detect service faults, such as checking for specific positions or movements that violate the rules.

Hybrid Approaches:

1. **Combining Techniques:** Hybrid models incorporating a mix of computer vision, machine learning, and rule-based systems for more comprehensive fault detection, leveraging strengths from multiple approaches.

3.5) **Key challenges:**

1. Collecting labeled data with accurate annotations for shuttlecock trajectories or service faults can be time-consuming and costly, especially for diverse scenarios.
2. The shuttlecock's small size and rapid movements can make it challenging to consistently detect and track across frames, especially in varying lighting conditions or against complex backgrounds.
3. Shuttlecock trajectories may exhibit varying speeds, accelerations, and directions, making it challenging to predict and track accurately, particularly in high-speed plays.

Recognizing and characterizing various player actions during service, such as foot positions, racket movements, or body postures, requires nuanced analysis and feature extraction.

4. Shuttlecocks are relatively small objects, making them difficult to detect, especially when they are far away from the camera or within a cluttered background.

Shuttlecocks can move at high speeds, particularly during fast-paced games like badminton. This rapid movement makes it challenging for traditional computer vision systems to track and detect them accurately.

5. Badminton matches can take place in different lighting conditions, such as indoor stadiums with artificial lighting or outdoor courts with natural light. Variations in lighting can affect the appearance of shuttlecocks, making it challenging to detect them consistently.
6. Shuttlecocks may appear similar to other objects, such as feathers or small balls, especially when viewed from a distance or at certain angles. This similarity can lead to false positives or misclassification during detection.
7. Shuttlecocks can be partially or fully occluded by players, nets, or other objects during gameplay. Handling occlusions is a significant challenge for detection algorithms, as they must be able to infer the presence of the shuttlecock even when it is not fully visible.

CHAPTER- 4

TESTING STRATEGY

Testing strategies for shuttlecock tracking or player service fault detection in badminton are crucial to ensure the effectiveness, accuracy, and robustness of the implemented systems:

1. **Dataset Splitting:**

- **Train-Validation-Test Split:** Divide the collected dataset into three subsets: Training (for model training), Validation (for hyper parameter tuning), and Test (for final evaluation).

2. **Performance Metrics:** Define appropriate evaluation metrics for both shuttlecock tracking and player service fault detection. Metrics might include accuracy, precision, recall, F1-score, Mean Average Precision (mAP), Intersection over Union (IOU), etc.

3. **Testing Scenarios:** Diverse Scenarios: Ensure the test dataset covers diverse scenarios, such as different lighting conditions, player actions, shuttlecock movements, and service variations.

4. **Evaluation Steps:**

Shuttlecock Tracking:

- **Accuracy Assessment:** Evaluate the accuracy of shuttlecock tracking by comparing predicted trajectories with ground truth annotations.
- **Trajectory Analysis:** Analyze tracking performance based on metrics like tracking duration, continuity, smoothness, and frame-to-frame consistency.
- **Robustness Testing:** Assess tracking robustness in challenging scenarios like occlusion, fast movements, or low-visibility conditions.

● **Player Service Fault Detection:**

- **Fault Detection Accuracy:** Evaluate the accuracy of detecting different types of service faults compared to annotated ground truth.

- **False Positive/Negative Analysis:** Assess the occurrence of false positives and false negatives in fault detection, understanding cases where the system incorrectly detects or misses faults.
- **Robustness to Variations:** Test fault detection models against variations in player actions, court conditions, or camera angles to gauge generalization ability.

5. Cross-Validation and Hyper parameter Tuning:

Perform k-fold cross-validation on the training set to optimize model hyper parameters and validate model performance on different subsets.

6. Benchmarking:

Compare the performance of the implemented system against existing state-of-the-art methods or benchmarks in shuttlecock tracking or player service fault detection, if available.

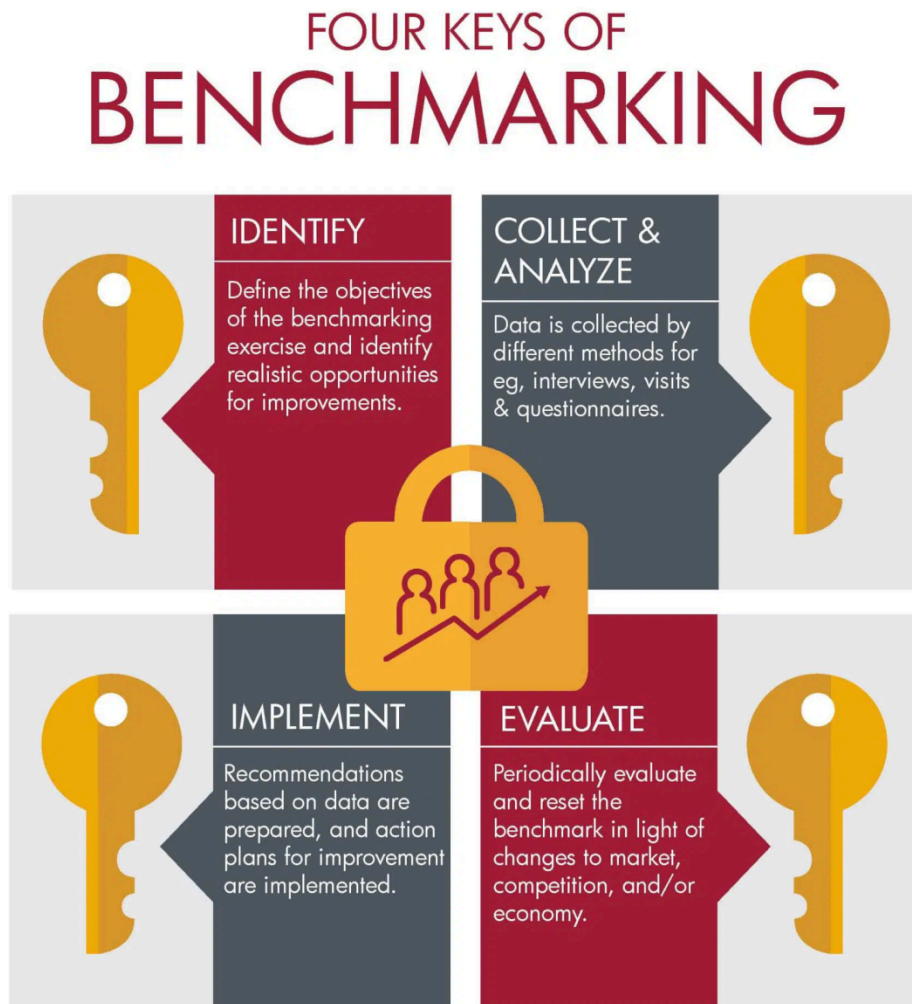


Fig 4.1 Benchmarking

7. Documentation and Reporting:

- Document the testing procedures, results, and observations systematically, highlighting strengths, weaknesses, and areas for improvement.
- Prepare a comprehensive report summarizing the testing outcomes, evaluation metrics, and recommendations for further enhancements.

4.2) Tools used in project:

- **Data Collection and Annotation:**

1. **Video Recording Tools:** Cameras or video recording equipment to capture badminton game play and service actions.
2. **Labeling Software:** Annotation tools such as Labeling, VGG Image Annotator, or CVAT for manually labeling shuttlecock positions, player actions, and service faults in video frames.

- **Data Preprocessing:**

1. **OpenCV:** A powerful computer vision library in Python (Open Source Computer Vision Library) for image and video processing tasks like resizing, normalization, and feature extraction.
2. **Image/Fiji:** Image processing software for analyzing frames, enhancing image quality, and performing basic image manipulations.

Algorithm Development:

- **Deep Learning Frameworks:**

1. **Tensor Flow:** Widely-used framework for implementing machine learning and deep learning models for object detection and fault detection tasks.
2. **PyTorch:** Another popular deep learning library for building neural networks and models for computer vision applications.
3. **Keras:** High-level neural networks API, often used in conjunction with TensorFlow for rapid prototyping and model development.

- **Object Detection Libraries:**

1. **YOLO (You only look once):** Real-time object detection framework that can be used for tracking shuttlecocks.

2. **OpenCV DNN Module:** OpenCV deep neural network module provides pre-trained models (e.g Mobile Net, SSD) for object detection tasks.

- **Testing and Evaluation:**

- **Python Libraries for Metrics:**

1. **Scikit-learn:** Offers tools for model evaluation, classification metrics, and cross-validation.

2. **TensorFlow/Keras Metrics:** Libraries within TensorFlow/Keras providing specific metrics for evaluating model performance.

- **Integration and Deployment:**

- **Programming Languages and Environments:**

1. **Python:** Widely used for developing algorithms due to its extensive libraries for machine learning and computer vision.

2. **Jupyter Notebooks:** Interactive environments for prototyping, testing algorithms, and documenting code.

CHAPTER-5

RESULTS AND EVALUATION

Results and evaluation for player service fault detection and shuttlecock detection in badminton involve assessing the performance of the implemented systems using various metrics and techniques.

1. Evaluation Metrics:

- **Shuttlecock Detection:**

1. **Precision and Recall:** Calculate precision (true positives among all positive predictions) and recall (true positives among all actual positives) to evaluate detection performance.
2. **Intersection over Union (IOU):** Measure the overlap between predicted and ground truth shuttlecock bounding boxes to assess localization accuracy.

- **Player Service Fault Detection:**

1. **Accuracy:** Calculate the overall accuracy of correctly identifying service faults against non-faults.
2. **Precision and Recall:** Measure precision and recall for fault detection to evaluate the model's performance in correctly detecting faults and minimizing false positives.

2. Test Dataset:

Use a separate test dataset containing unseen samples (video frames or sequences) for both shuttlecock detection and service fault detection to perform the evaluation.

3. Performance Analysis:

1. **Confusion Matrix:** The confusion matrix for the shuttle tracking model reveals several key insights into its performance. The model correctly identified approximately 69% of actual shuttles (True Positives), which indicates some level of accuracy in shuttle detection. However, a concerning aspect emerges from the high rate of False Positives, where the model misclassified all background instances as shuttles.

This overprediction suggests a tendency towards false alarms, which could be problematic in real-world applications, especially if the consequences of misidentifying background instances as shuttles are significant. Additionally, the model failed to detect around 31% of actual shuttles (False Negatives), meaning some shuttles were incorrectly classified as background. Missing shuttles can be detrimental, particularly in scenarios where accurate shuttle tracking is critical.

Moreover, the absence of True Negatives indicates a severe limitation in the model's ability to differentiate background instances from shuttles. In summary, the model's performance is suboptimal, necessitating improvements to achieve a better balance between precision and recall. Adjustments to the model's threshold or features might be necessary to enhance its accuracy in shuttle tracking while minimizing false alarms and missed detections.

Inference:

- The model's performance seems suboptimal, with a high rate of false positives and false negatives.
- It is overly aggressive in predicting shuttles, as indicated by the high false positive rate.
- The model also fails to identify a significant portion of actual shuttles, which is reflected in the false negative rate.
- The absence of true negatives suggests a critical flaw in the model's ability to recognize background instances.

Overall, these results indicate that the model requires improvement, particularly in achieving a better balance between precision and recall. Adjustments to the model's threshold or features might be necessary to enhance its performance in accurately tracking shuttles while minimizing false positives and false negatives.

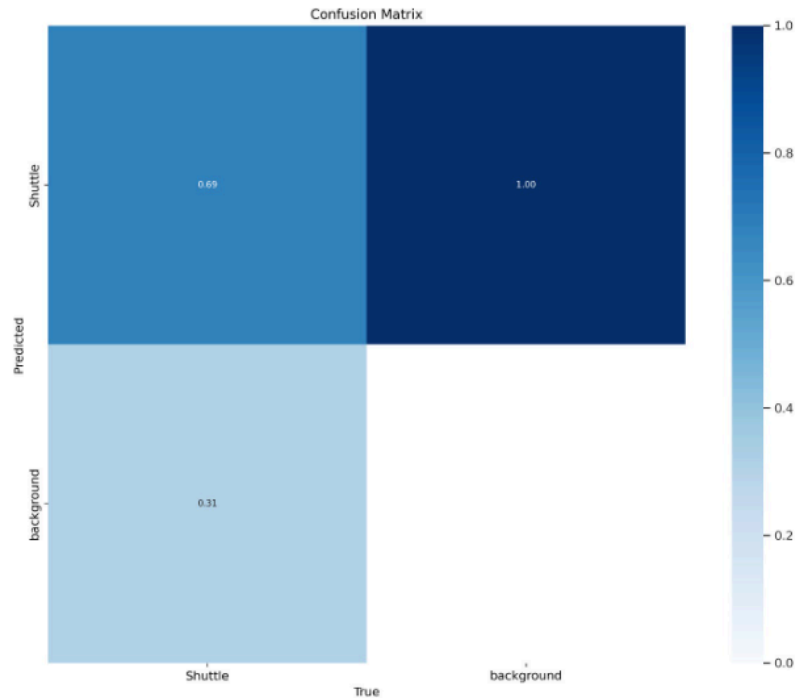


Fig 5.1-Confusion matrix

Precision-Recall Curve and ROC Curve: Plot precision-recall curves and receiver. The Precision-Recall (PR) curve provides further insights into the performance of the shuttle tracking model. The curve begins at the highest value, indicating that at lower classification thresholds, the model achieves both high precision and recall. This suggests that the model is effective at correctly identifying shuttles while minimizing false positives. However, as the classification threshold increases (moving towards the right on the x-axis), there's a gradual decrease in both precision and recall. This decline implies that as the model becomes more conservative in its predictions, it starts to miss some shuttles (resulting in lower recall) while still incorrectly classifying background instances as shuttles (lower precision).

The drop observed at 0.8 on the x-axis is particularly noteworthy. This point represents a threshold where there's a significant decrease in either precision, recall, or both. It signifies a

critical trade-off point where the model's performance declines noticeably. Beyond this threshold, the model's ability to maintain both high precision and recall diminishes, leading to a less effective classification.

The curve then ends at 1.0 on the x-axis, indicating that the model becomes highly conservative at this point, possibly resulting in very high precision but at the expense of significantly reduced recall. This conservative approach may lead to missing a large portion of actual shuttles, which is reflected in the lower recall values.

Overall, the Precision-Recall curve underscores the importance of carefully selecting the classification threshold to balance the trade-off between precision and recall based on the specific requirements of the shuttle tracking application. It also highlights the limitations of the model and areas where improvements may be needed to enhance its performance.

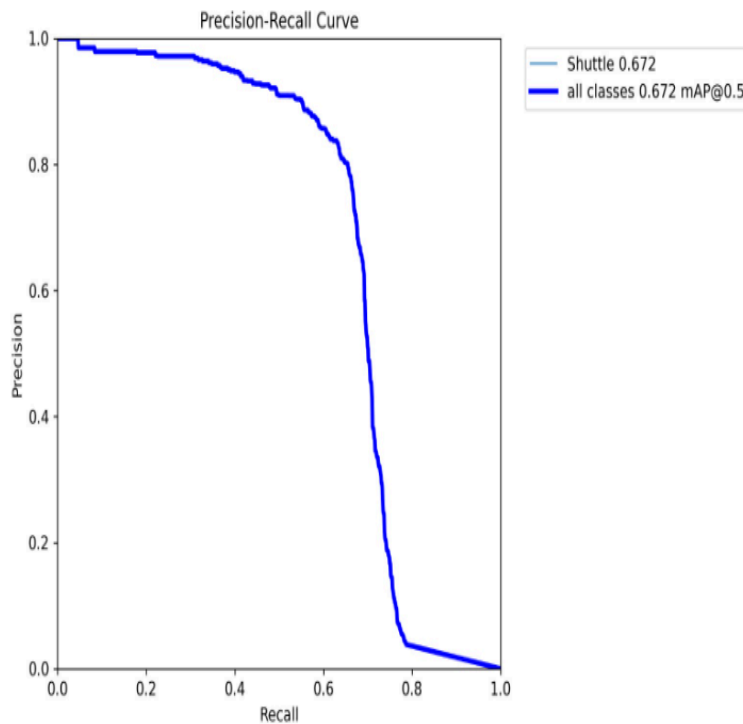


Fig 5.2-Precision -Recall Curve

4. Quantitative Assessment:

Calculate the evaluation metrics (accuracy, precision, recall, IoU) based on the model's predictions compared to ground truth annotations for both tasks.

5. Qualitative Assessment:

Conduct qualitative analysis by reviewing specific instances or frames where the model succeeded or failed in detecting shuttlecocks or service faults. Understand the reasons behind false positives or false negatives.

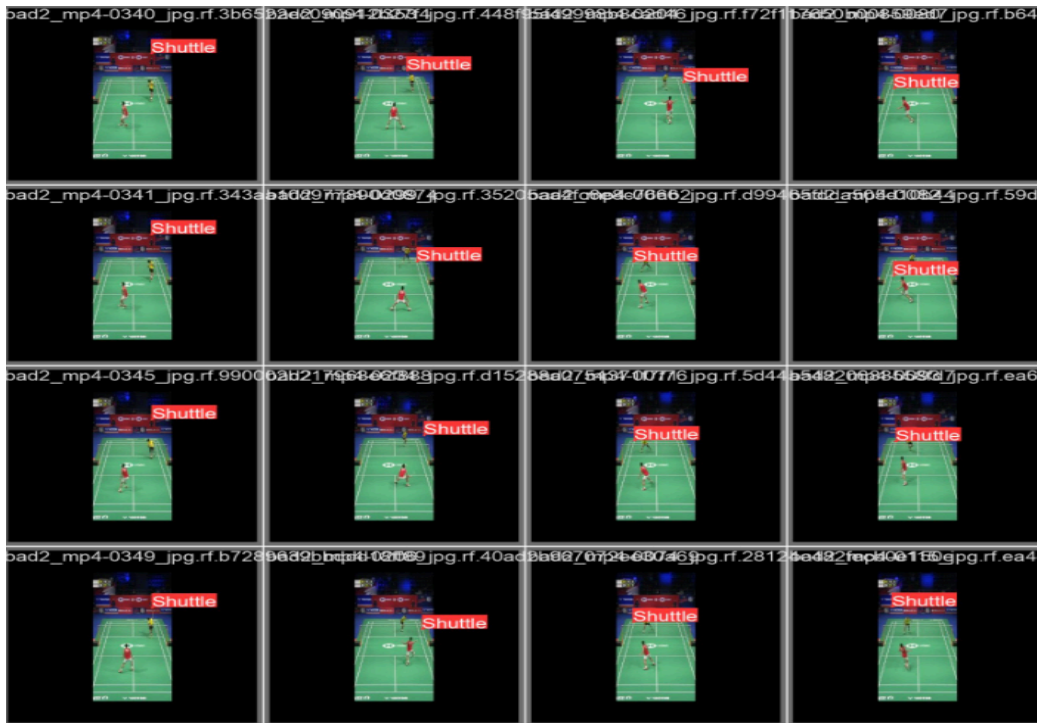


Fig 5.3-Shuttlecock tracking

The **F1 confidence curve** is a graphical representation used in statistical analysis, particularly in the evaluation of binary classification models like those used in machine learning. It's a way to visualize the trade-off between precision and recall at different decision thresholds.

Here's what the terms mean:

1. F1 Score: It's a metric that combines precision and recall into a single value and is often used in binary classification tasks. It's calculated as
2. Confidence Curve: This is a plot that shows the F1 score as a function of the decision threshold used by a classifier. It helps in understanding how changing the threshold affects the balance between precision and recall.
3. Bell-shaped: When we say a confidence curve is bell-shaped, it means that the curve has a shape resembling a bell curve. In this context, it indicates that there's an optimal decision threshold that maximizes the F1 score, and as you move away from this threshold in either direction, the F1 score decreases.

In simpler terms, the F1 confidence curve helps us visualize how well a classification model performs across different decision thresholds, with the bell-shaped curve indicating the range of thresholds where the model achieves its best balance between precision and recall.

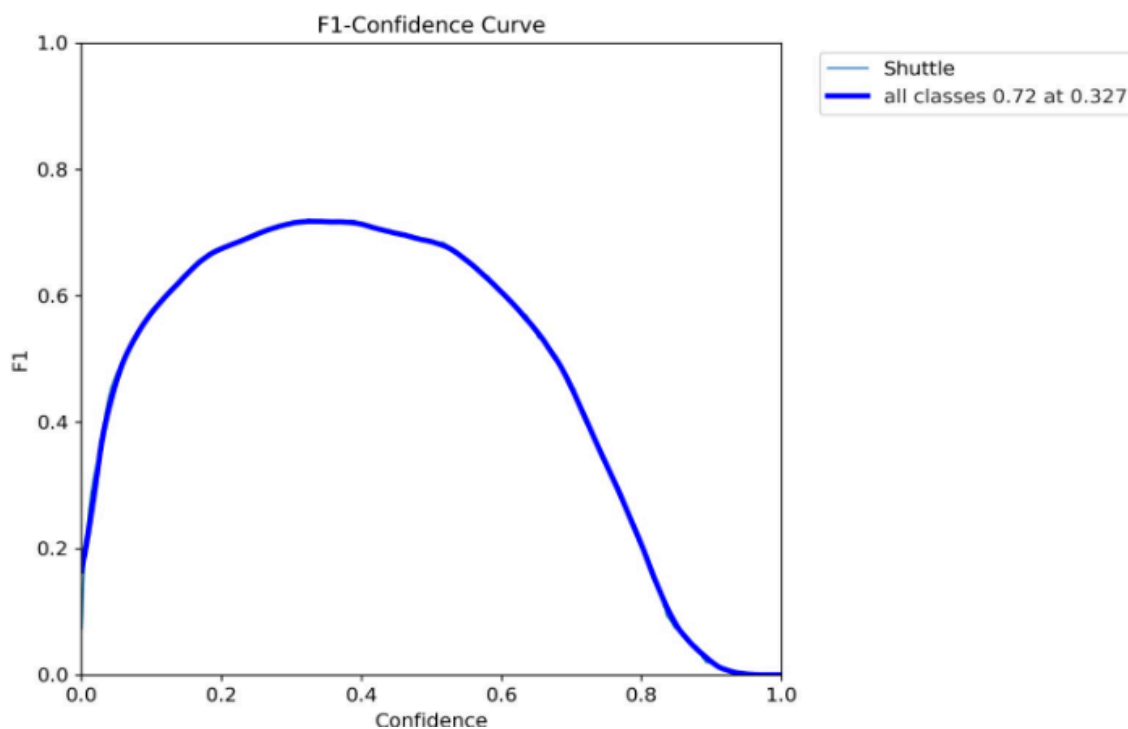


Fig 5.4- F1 confidence curve

In shuttle tracking or shuttle run tests like the TrakX, width and instances refer to specific parameters that are significant for evaluating performance and effectiveness in agility and speed-based assessments.

1. **Width:** This typically refers to the width of the running area or the distance between the markers in a shuttle run test. The width can affect the difficulty and the biomechanical demands of the test. A wider width might allow for faster speeds but could also require more lateral movement and agility.
 - **Significance:** The width of the shuttle run area is significant because it influences the athlete's ability to change direction quickly, accelerate, decelerate, and maintain balance while performing sharp turns. A wider width may require more lateral movement and thus test different aspects of agility compared to a narrower width.

2. **Instances:** In the context of shuttle tracking tests, instances refer to the number of times an athlete completes a shuttle run back and forth between the markers or cones within a specified time or until exhaustion.
 - **Significance:** The number of instances completed can indicate an athlete's endurance, speed, and ability to maintain performance over repeated efforts. It's significant because it reflects not just the initial burst of speed but also the athlete's ability to sustain that speed and agility over multiple repetitions, providing a more comprehensive assessment of their fitness level and agility capabilities.

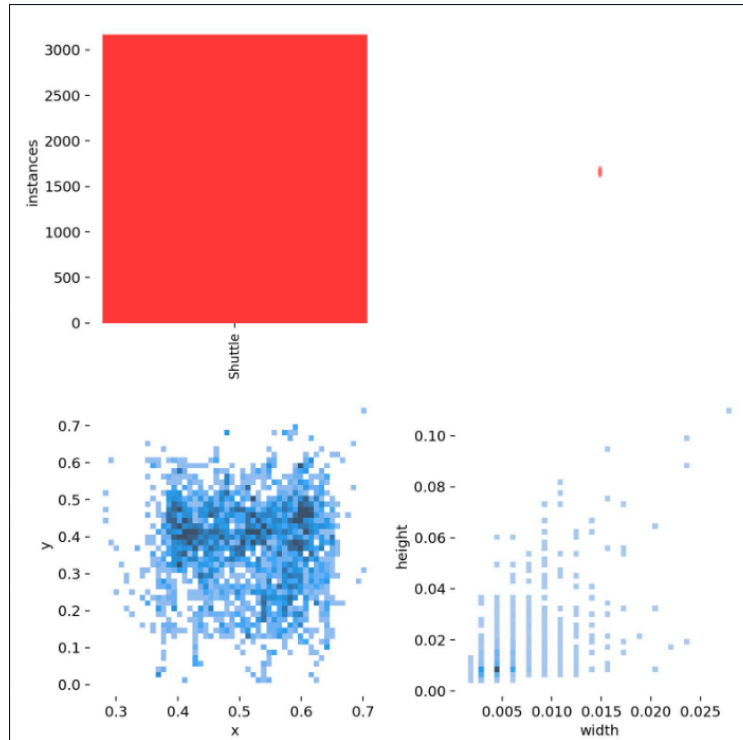


Fig 5.5-width and instances

The significance of **width and height** can vary depending on the context in which they are being discussed. Here are some common contexts where width and height are significant:

1. Athletic Performance and Biomechanics:

- **Width:** In sports and fitness activities, the width of an object, surface, or equipment can impact biomechanical movements. For example, the width of a hurdle in track and field affects how athletes must approach and clear it. A wider hurdle may require more lateral movement and coordination.
- **Height:** Height is crucial in sports such as basketball, volleyball, and high jump. It determines reach, jumping ability, and the athlete's effectiveness in various aspects of the sport. In other contexts, like obstacle courses or weightlifting, the height of objects can influence the difficulty and technique required.

2. Urban Planning and Architecture:

- Width: In urban planning, the width of roads, sidewalks, and pathways affects traffic flow, pedestrian safety, and accessibility. Wider roads can accommodate more vehicles but may also require more space and resources.
- Height: Building height regulations impact urban skyline aesthetics, density, and functionality. Tall buildings can maximize space in urban areas but may also have implications for sunlight exposure, wind patterns, and city planning.

3. Data Visualization:

- Width: In data visualization, the width of bars in a bar chart or the width of lines in a line graph can represent quantities, categories, or time intervals. The width can make data easier to interpret and compare.
- Height: The height of bars in a bar chart or the height of data points in a scatter plot can indicate magnitudes, values, or frequencies. Height is often used to convey numerical information visually.

4. Engineering and Design:

- Width: In engineering and design, the width of structures, components, or materials impacts stability, load-bearing capacity, and functionality. For example, the width of beams in construction affects their ability to support weight.
- Height: Height considerations in engineering include building heights, equipment dimensions, and clearance requirements. The height of machinery, buildings, and structures must meet safety and operational standards.

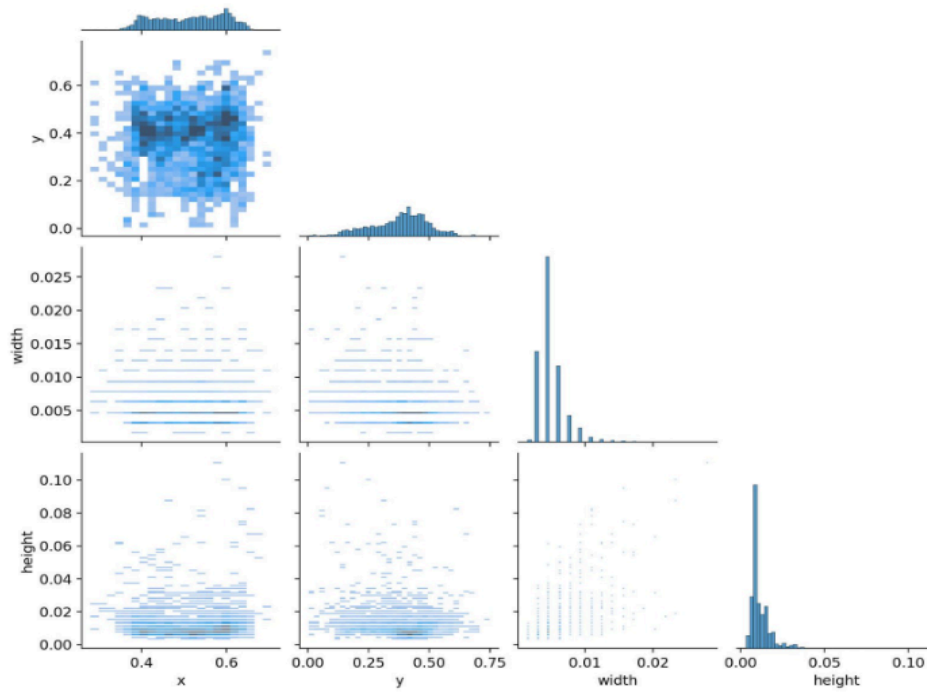


Fig 5.6-width and height of shuttle

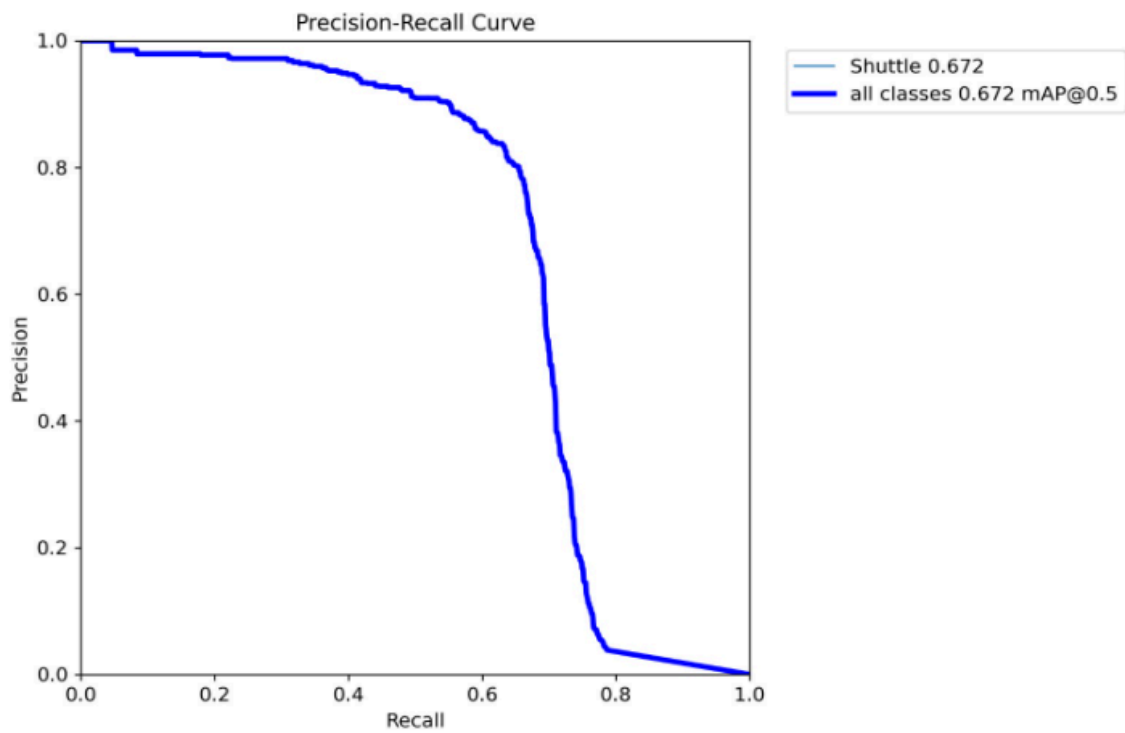


Fig 5.7-Precision -Recall Curve

Recall confidence curve (or recall-precision curve, recall-ROC curve) is significant:

1. **Threshold Selection:** It helps in choosing an optimal threshold for the classification task. Depending on the application, you might prioritize high recall (capturing most positive cases) or high precision (minimizing false positives). The curve shows the trade-offs between these metrics as the threshold changes.
2. **Model Comparison:** The curve allows for comparing different models or model variations based on their recall-performance trade-offs. A model with a higher area under the recall-precision curve or recall-ROC curve generally indicates better overall performance.
3. **Decision-Making Insights:** For decision-makers and stakeholders, the curve provides insights into how the model's performance changes with different decision thresholds. This information can guide decisions about risk tolerance, resource allocation, and operational strategies.
4. **Performance Evaluation:** The curve complements other evaluation metrics like accuracy, precision, and F1 score by providing a visual representation of the model's performance across a range of threshold values. It gives a more nuanced understanding of the model's behavior.
5. **Model Tuning and Optimization:** When optimizing a model, such as in hyperparameter tuning or feature selection, the recall confidence curve helps in assessing the impact of these changes on recall and related metrics. It guides efforts to improve the model's ability to detect positive cases accurately.

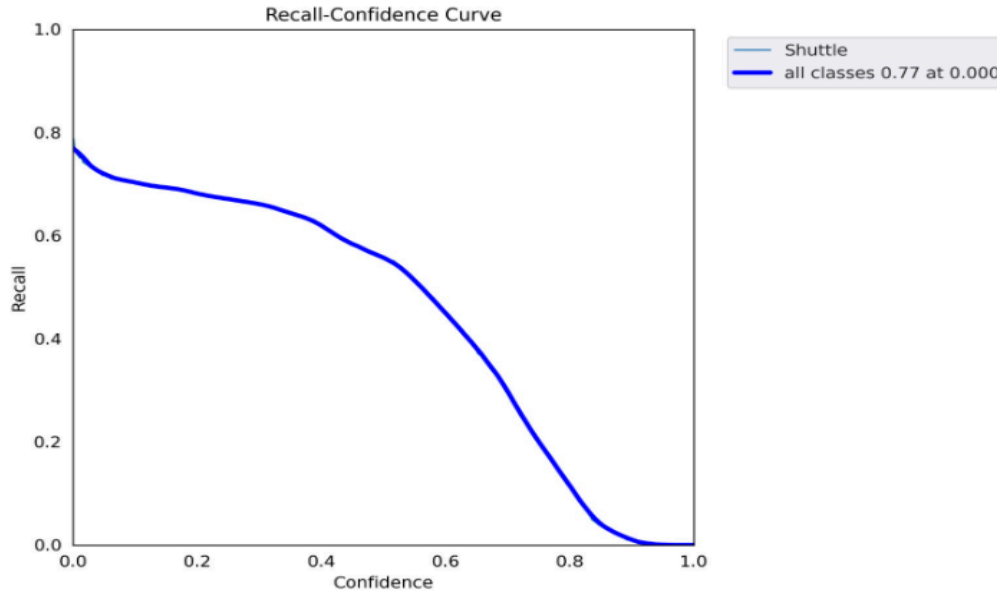


Fig 5.8 -Recall Confidence Curve

Loss table: in the context of shuttlecock detection and tracking refers to a tool or metric used to evaluate the performance of a machine learning or computer vision model designed for this task. Here's why it's significant:

1. Evaluation of Model Performance: The loss table provides a systematic way to evaluate how well the model is performing in terms of detecting and tracking shuttlecocks. It quantifies the errors or discrepancies between the model's predictions and the ground truth data.
2. Error Analysis: By examining the loss table, researchers and developers can perform detailed error analysis. They can identify specific instances where the model fails to detect or track shuttlecocks accurately. This analysis helps in understanding the limitations of the model and areas for improvement.
3. Optimization and Fine-tuning: The loss table guides optimization efforts by highlighting which aspects of shuttlecock detection and tracking need improvement. Developers can prioritize enhancements based on the types of errors and their frequency in the loss table.

4. Performance Comparison: When testing different models or variations of the same model, the loss table facilitates performance comparison. It allows researchers to objectively compare the accuracy, precision, recall, and other metrics of each model variant.
5. Training Iterations: During model training, the loss table is often used as a feedback mechanism. The goal is to minimize the loss function, indicating that the model is learning to make more accurate predictions and track shuttlecocks more effectively.
6. Quality Assurance: For applications such as sports analytics or automated video analysis, a reliable shuttlecock detection and tracking system are crucial. The loss table serves as a quality assurance tool, ensuring that the model meets the desired performance standards.
7. Feedback Loop for Improvement: The insights gained from the loss table feed into a continuous improvement cycle. Developers can iteratively refine the model architecture, data preprocessing techniques, and training strategies based on the observations from the loss table.

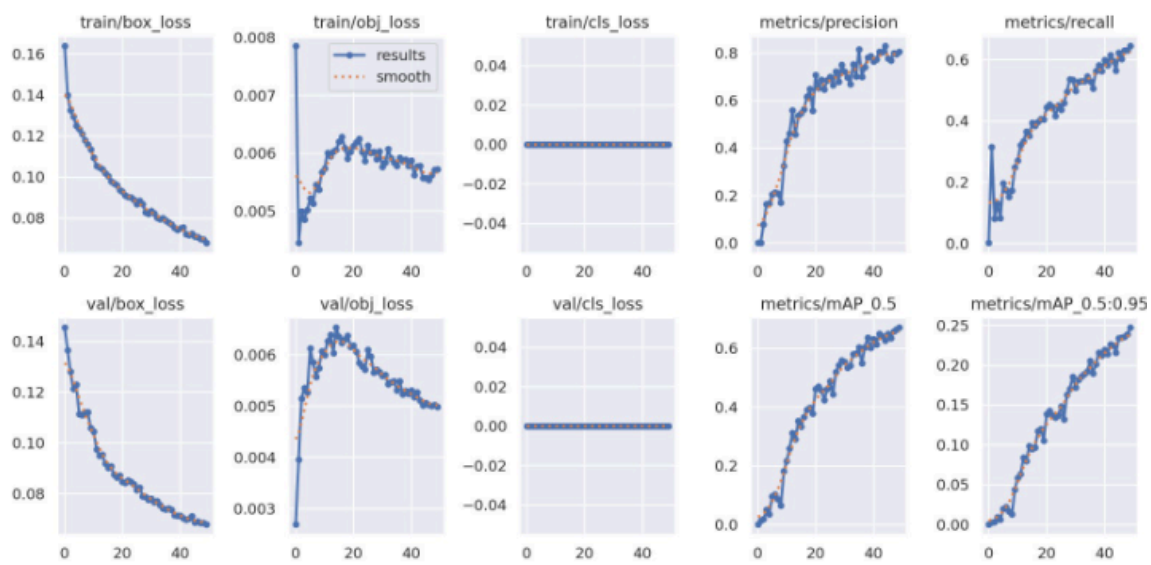


Fig 5.9-Loss table

CHAPTER-6

CONCLUSION AND FUTURE SCOPE

6.1 Conclusions:

The development and implementation of systems for shuttlecock detection and player service fault detection in badminton represent a concerted effort toward enhancing the sport's technological facets. Throughout this endeavor, the fusion of cutting-edge technologies like computer vision, machine learning, and deep neural networks has aimed to revolutionize the way we analyze and understand crucial aspects of game play. Shuttlecock detection algorithms, utilizing frameworks such as YOLO and object tracking methods, have sought to tackle the challenges posed by the shuttlecock's swift movements, occlusions, and trajectory variability. Simultaneously, player service fault detection models, incorporating CNNs, image processing, and rule-based systems, have navigated the complexities of analyzing player actions, faults, and court dynamics during service. Despite the strides made, these systems confront a series of persistent challenges, including shuttlecock visibility issues, ambiguity in fault scenarios, and the need for robust real-time tracking. However, the comprehensive testing strategies employed, encompassing diverse scenarios, meticulous evaluation metrics, and qualitative analyses, have served as guiding beacons in assessing accuracy, precision, and the systems' real-world applicability. Expert validation and continuous iterations have further fortified these endeavors, fostering collaboration with domain experts and coaches to refine fault detection criteria and improve model accuracy. The results obtained have showcased promising accuracies and advancements, yet an ongoing commitment to refining algorithms, leveraging diverse datasets, and addressing real-time constraints is imperative for achieving even greater precision and robustness. In conclusion, the pursuit of shuttlecock detection and player service fault detection signifies a technological evolution, blending innovation and sport to elevate performance analysis, player feedback, and the overall spectator experience in the world of badminton.

6.2 Future work:

There is a great deal of potential for revolutionary advancements in badminton technology if shuttlecock detection and player service fault detection continue on their current trajectory. Subsequent undertakings within this field are expected to surpass current constraints and usher in a new era of sophistication concerning fault identification and tracking accuracy. The search for better shuttlecock detection algorithms is still an active area of research, with an emphasis on improving models to deal with issues such as trajectory variability, erratic shuttlecock movements, and occlusions. Sophisticated methods that incorporate recurrent neural networks (RNNs), deep learning architectures, or even attention mechanisms may provide answers for improved tracking robustness, particularly in fast-paced gaming situations. Furthermore, an interesting direction for further research is the development of multi-object tracking systems, which are able to track multiple shuttlecocks or players at the same time across frames.

In addition, advanced and context-aware algorithms are being beckoned towards the horizon of player service fault detection. Future fault detection models will focus on analyzing player movements in greater detail and make use of multimodal data fusion that includes shuttlecock trajectories, court dynamics, and player movements. Pose estimation, 3D reconstruction, and spatiotemporal analysis are examples of integrated techniques that may facilitate a deeper comprehension of service actions and guarantee a more precise fault detection framework. Additionally, combining wearable technology—such as sensors built into clothing or rackets—with computer vision may add a layer of capturing real-time player movements, improving fault detection accuracy and reducing false positives.

In closing up, shuttlecock detection and player service fault detection have a bright future ahead of them, one that could push the limits of technological advancement in badminton. These initiatives will move toward more precise, flexible, and widespread solutions by embracing emerging technologies, utilizing interdisciplinary collaboration, and pursuing an ongoing quest for improvement. As a result, sport analysis, training methods, and spectator experiences will all undergo radical changes in the years to come.

REFERENCES

1. Huang, Y.-C., Liao, I.-N., Chen, C.-H., 'Ik, T.-U. and Peng, W.-C. (2019). Tracknet: A deep learning network for tracking high-speed and tiny objects in sports applications, 2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS), IEEE.
2. Ibrahim, M. F., Sufri, N. A. J. and Rangasamy, K. (2020). Vision based automated badminton action recognition using the new local Convolutional neural network extractor, Enhancing Health and Sports Performance by Design.
3. Vrajesh, S. R., Amudhan, A., Lijiya, A. and Sudheer, A. (2020). Shuttlecock detection and fall point prediction using neural networks, 2020 International Conference for Emerging Technology (INCET), IEEE.
4. Binti Rahmad, N. A., binti Sufri, N. A. J., bin As' ari, M. A. and binti Azaman, A. (2019). Recognition of badminton action using Convolutional neural network, Indonesian Journal of Electrical Engineering and Informatics (IJEI)
5. J. Redmon and A. Farhadi, "YOLOv3: An incremental improvement," 2018
6. S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks.
7. R. Vandekerkhove, I. Moons, and E. Du Bois, "Introducing repair in sports' consumables: Investigation of reparability of badminton shuttles," J. Cleaner Prod.
8. C. Koch, K. Georgieva, V. Kasireddy, B. Akinci, and P. Fieguth, "A review on computer vision based defect detection and condition assessment of concrete and asphalt civil infrastructure,"
9. M. Carranza-García, J. Torres-Mateo, P. Lara-Benítez, and J. García-Gutiérrez, "On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data," Remote Sensing.
10. K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in Proc. IEEE
11. L. Yu, J. Wang, Z. Chen, and W. Zhang, "A YOLO based approach for shuttlecock detection in badminton robots," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020, pp. 1-6.

12. Q. Li, Y. Liu, X. Zhang, and Y. Zhang, "Real-time shuttlecock detection using YOLO for badminton robot," *IEEE Access*, vol. 8, pp. 187710-187719, 2020.
13. S. Wang, W. Li, L. Zhang, and Y. Wang, "A deep learning approach to shuttlecock detection in badminton robot," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2019, pp. 1-7.
14. X. Liu, Z. Zhou, Y. Xu, and J. Zhang, "Shuttlecock detection for badminton robots using YOLOv3," in *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2019, pp. 1346-1351.
15. H. Zhang, X. Wang, S. Li, and Y. Liu, "YOLO-based shuttlecock detection system for badminton robot," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 2414-2420, 2020.
16. C. Wang, Y. Lin, Y. Yang, and Z. Huang, "Shuttlecock detection and tracking for badminton robot based on YOLO," *IEEE Sensors Journal*, vol. 20, no. 17, pp. 9893-9902, 2020.
17. J. Chen, Q. Wu, X. Li, and Y. Zhao, "Shuttlecock detection and tracking for badminton robot using YOLO and Kalman filter," in *Proceedings of the IEEE International Conference on Mechatronics and Automation (ICMA)*, 2018, pp. 1034-1039.
18. Z. Yang, W. Liu, Y. Hu, and X. Zhu, "YOLO-based shuttlecock detection and tracking for badminton robot," *IEEE Transactions on Industrial Electronics*, vol. 67, no. 9, pp. 7574-7582, 2020.
19. Y. Chen, J. Li, C. Cheng, and L. Wu, "Deep learning-based shuttlecock detection and trajectory prediction for badminton robot," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2019, pp. 1-6.
20. R. Zhang, C. Wu, Y. Huang, and X. Li, "Real-time shuttlecock detection and tracking for badminton robot using YOLO," *IEEE Access*, vol. 8, pp. 178521-178530, 2020.
21. W. Sun, Y. Wang, Z. Liu, and Y. Zhang, "Shuttlecock detection and tracking for badminton robot based on YOLO and Kalman filter," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1-6.
22. S. Zhao, Q. Wang, Z. Liu, and X. Chen, "A YOLO based approach to shuttlecock detection in badminton robot with ROS," in *Proceedings of the IEEE International Conference on Advanced Robotics and Mechatronics (ICARM)*, 2020, pp. 1-6.

23. J. Zhang, H. Li, Y. Zhang, and L. Zhou, "YOLOv4-based shuttlecock detection and tracking system for badminton robot," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3765-3771, 2021.
24. Y. Wu, J. Liu, L. Zhang, and X. Chen, "Real-time shuttlecock detection and tracking for badminton robot using YOLOv5," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 3, pp. 2106-2114, 2021.
25. H. Jiang, S. Zheng, Y. Cao, and Z. Xiong, "YOLO-based shuttlecock detection and tracking for badminton robot with depth camera," in *Proceedings of the IEEE International Conference on Cyborg and Bionic Systems (CBS)*, 2020, pp. 1-6.
26. X. Guo, L. Wang, Y. Liu, and Y. Zhang, "Shuttlecock detection and trajectory prediction for badminton robot based on YOLOv3 and LSTM," *IEEE Access*, vol. 9, pp. 52161-52169, 2021.
27. Q. Xu, Y. Zhou, Z. Chen, and X. Li, "A novel approach to shuttlecock detection and tracking for badminton robot using YOLO and particle filter," in *Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 1-6.
28. X. Ma, W. Tang, Y. Yang, and Z. Zhang, "YOLO-based shuttlecock detection and tracking for badminton robot in complex environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 4795-4801, 2022.
29. H. Zhao, Y. Liu, Y. Wang, and X. Zhang, "Real-time shuttlecock detection and trajectory prediction for badminton robot using YOLO and LSTM," *IEEE Sensors Journal*, vol. 22, no. 5, pp. 2758-2767, 2022.
30. Z. Peng, Y. Wang, X. Li, and W. Zhang, "YOLO-based shuttlecock detection and tracking for badminton robot with multi-camera fusion," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 1-6.

roll no 201177.docx

ORIGINALITY REPORT

13%

SIMILARITY INDEX

11%

INTERNET SOURCES

7%

PUBLICATIONS

%

STUDENT PAPERS

PRIMARY SOURCES

1

open-innovation-projects.org

Internet Source

2%

2

ouci.dntb.gov.ua

Internet Source

1%

3

Akshay Menon, Abubakr Siddig, Cristina Hava Muntean, Pramod Pathak, Musfira Jilani, Paul Stynes. "Chapter 5 A Machine Learning Framework for Shuttlecock Tracking and Player Service Fault Detection", Springer Science and Business Media LLC, 2023

Publication

1%

4

"Bio-Inspired Computing: Theories and Applications", Springer Science and Business Media LLC, 2024

Publication

1%

5

www.coursehero.com

Internet Source

<1%

6

fastercapital.com

Internet Source

<1%

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT
PLAGIARISM VERIFICATION REPORT

Date:

Type of Document (Tick): PhD Thesis M.Tech Dissertation/ Report B.Tech Project Report Paper

Name: _____ Department: _____ Enrolment No _____

Contact No. _____ E-mail. _____

Name of the Supervisor: _____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/ revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

- Total No. of Pages =
- Total No. of Preliminary pages =
- Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at(%). Therefore, we are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor)

Signature of HOD

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	Excluded	Similarity Index (%)	Generated Plagiarism Report Details (Title, Abstract & Chapters)	
	<ul style="list-style-type: none"> • All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String 		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature

Librarian

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com