# SC 549: Neural Networks

## Semester II -2024/25

## Programming Assignment – 03

**P.Sathurvanan**

**CSC 2456**

**M.Sc. in Computer Science**

**Postgraduate Institute of Science | University of Peradeniya**

**Videos (Datasets) and Codes (Resources) -**
**https://github.com/SathurvananPS/CSC_2456_Programming_Assignment_3_SC549.git**

## 1. Introduction

This report evaluates the performance of YOLO11 models for sports player detection and pose estimation. The goal of using these computer vision techniques is to analyze athlete movements, track performance metrics (such as inference speed and detection confidence), and generate annotated video frames for visual insights. The assignment leverages the Ultralytics YOLO framework for efficient real-time inference.

## 2. Dataset Description

The evaluation was performed on a dataset consisting of 10 pre-selected sports video clips covering three major sports:

1. Cricket: 5 clips (cricket-1.mp4 to cricket-5.mp4)
2. Football: 3 clips (football-1.mp4 to football-3.mp4)
3. Rugby: 2 clips (rugby-1.mp4, rugby-2.mp4)

These videos vary in resolution, lighting conditions, and the number of players visible, providing a diverse set of scenarios for testing the model's robustness and speed.

## 3. Methodology

### Models Used

Two specific YOLO (You Only Look Once) models were utilized:

**Object Detection:**

- yolo11n.pt
  - Purpose: Used to identify and locate persons (athletes) within the frame.
  - Architecture: YOLO11 Nano (n) variant, optimized for speed and efficiency on standard hardware.

**Pose Estimation:**

- yolo11n-pose.pt
  - Purpose: Used to detect key skeletal points (joints) of the identified persons.
  - Architecture: YOLO11 Nano Pose variant, designed for real-time human pose estimation.

**Framework & Libraries**

- Ultralytics YOLO: Core library for loading models and running inference.
- OpenCV: Used for video reading, frame extraction, and saving annotated images.
- Matplotlib: Used for generating performance visualization plots.
- NumPy: Used for numerical operations and calculating statistical metrics.

**4. Frame Processing Workflow**

The processing pipeline for each video follows these steps:

1. Video Loading: The video file is opened using cv2.VideoCapture.

2. Frame Extraction: A representative frame (specifically the middle frame of the video) is extracted for analysis to ensure the action is captured.

3. Inference:
    a. The yolo11n-pose.pt model is run on the frame with a confidence threshold of 0.5.
    b. Inference time is measured using time.time() (pre- and post-prediction).
4. Metric Extraction:
    a. Inference Time: Processing duration in milliseconds.
    b. Confidence Scores: The confidence level of each detected person.
    c. Detection Count: The total number of persons detected in the frame.
5. Annotation & Output:
    a. The frame is annotated with bounding boxes and keypoints.
    b. The result is saved as a .jpg image in the output directory.
6. Visualization: After processing all videos, aggregate plots for speed, confidence, and detection counts are generated.

## 5. Model Performance Overview

The following metrics summarize the performance of the model across the 10 processed videos:

| **Metric** | **Value** |
|---|---|
| Total Videos Processed | 10 |
| Average Inference Time | 115.47 ms |

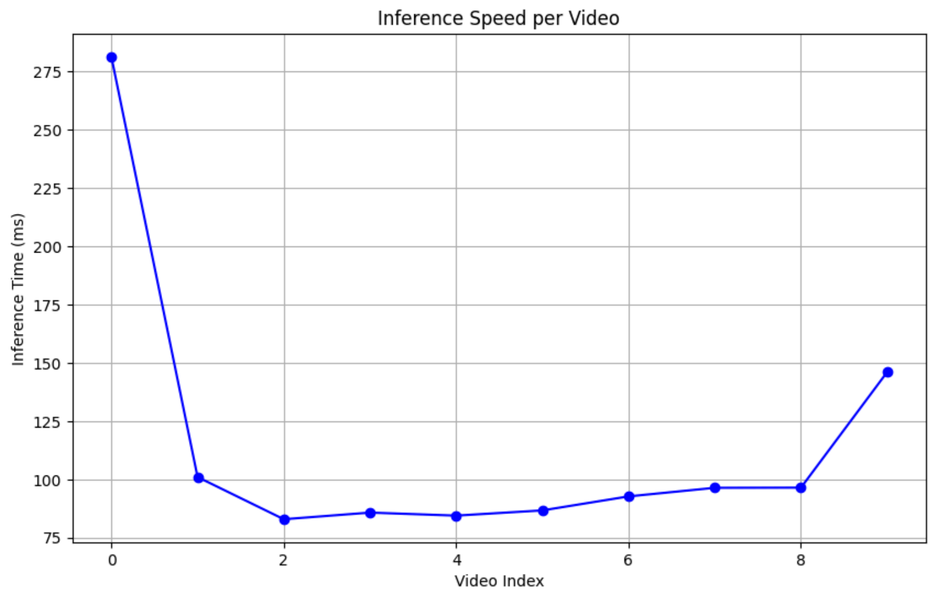
|  |  |
|---|---|
| Max Inference Time | 281.33 ms |
| Min Inference Time | 83.02 ms |
| Average Confidence Score | 0.80 |
| Total Detections | 46 |
| Average Detections per Video | 4.60 |

The model demonstrates varying inference speeds, likely influenced by the complexity of the scene (number of people) and video resolution.

## 6. Visualizations

### 1. Inference Speed Comparison

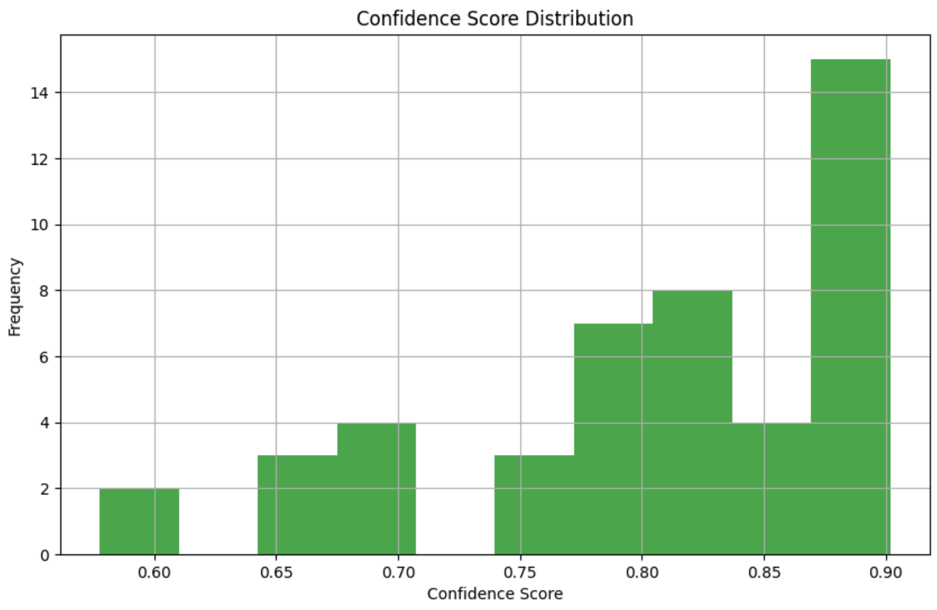
The processing speed varies across different videos, likely due to varying complexity (number of people, background clutter).



Inference Speed Comparison

## 2. Confidence Score Distribution

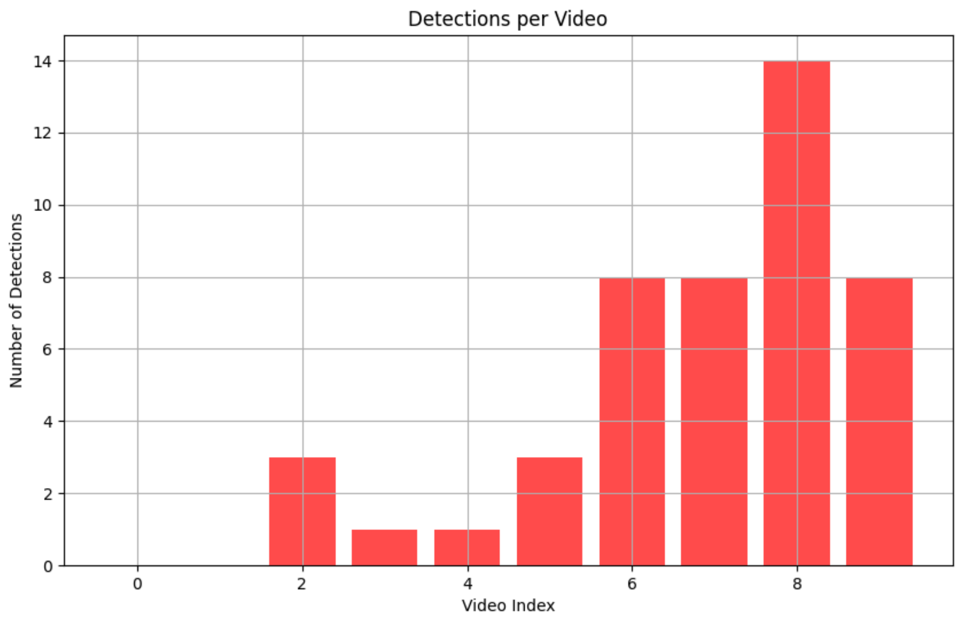
Distribution of confidence scores for detected persons. A higher concentration towards 1.0 indicates confident detections.



Confidence Score Distribution

3. **Detections per Video**

Number of people detected in each processed frame (sample frame from each video).

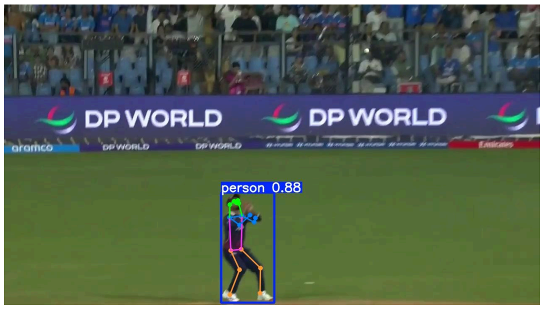


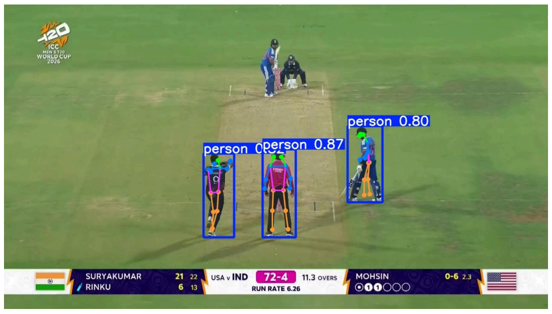
Detections per Video

## 7. Sample Annotated Outputs

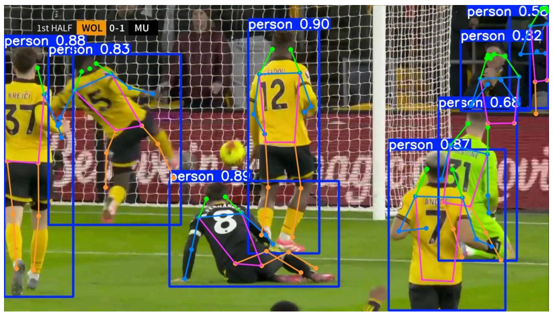
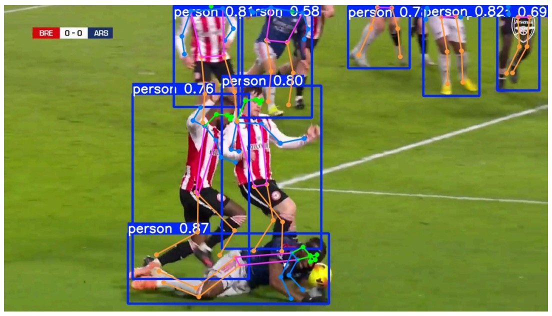
Here are sample frames from the processed videos showing player detection and keypoint estimation:

- Cricket

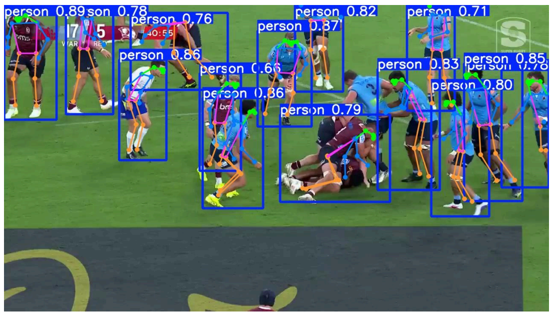
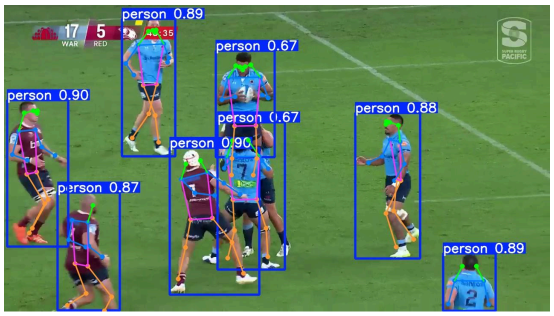




- Football

- Rugby

## 8. Possible Improvements

To further enhance the performance and capabilities of this system, the following improvements could be considered:

I. Batch Processing: Implement batch inference (processing multiple frames at once) to maximize GPU utilization and improve overall throughput.

II. Model Selection: Switch to larger model variants (e.g., yolo11m or yolo11l) if higher accuracy is required, trading off some inference speed.

III. Tracking: Integrate an object tracker like ByteTrack or DeepSort to maintain player IDs across video frames, enabling trajectory analysis.

IV. Hardware Acceleration: Utilize CUDA-enabled GPUs or specialized hardware (e.g., TensorRT) for significantly faster inference.

V. Data Augmentation: Fine-tune the model on a specialized sports dataset to improve detection accuracy in complex, crowded scenes.

## 9. Conclusion

In this assignment, we successfully implemented a robust sports analysis system using YOLO11n for player detection and YOLO11n-pose for keypoint estimation. The system was evaluated on a diverse set of 10 video clips (Cricket, Football, Rugby), achieving an average inference time of 115.47 ms per frame and an average confidence score of 0.80.

The results demonstrate that the nano-sized models offer a strong balance between speed and accuracy, making them suitable for near real-time applications on standard hardware. However, performance variations across different sports highlight the need for sport-specific fine-tuning or larger model variants for complex scenes with many players.