SNCC Crowd Monitoring System

1st Shriram Pragash M
Epoch IIT Hyderabad

2nd Siddharth Pamidi
Epoch IIT Hyderabad

I. Introduction

The project seeks to address the problem faced by students of not knowing whether or not their desired venue/ground in the SNCC is currently being used i.e whether the venue/ground is currently free for them to play/use. In order to combat this problem, this project seeks to serve as a means for students to see the current status of their desired location. In order to accomplish this, it utilises object detection to identify the number of people on the court/in the venue. The object detection will be fine tuned to only look out for people 'using' the court/venue and to ignore persons who are likely just audience members. The system can also be expanded to include an option to 'book' courts following approval by the sports secretary or respective sports coaches, in order to also serve as a means to communicate court reservations to the student body.

II. System Design Outline

The system will be fed real-time footage (CCTV for example) and will be given the task of detecting the crowd present, i.e the number of people actively using the court. This can be achieved using various methods such as counting the number of persons within the perimeter of the court or detecting the number of racket bounding boxes and person bounding boxes within a certain distance (both can be achievable assuming a fixed camera position). It utilises object tracking in order to provide a relatively more accurate and useful metric while counting the number of 'active' users.

III. Background Study

There are generally two types of object detection models, i.e single-shot object detectors and two-stage object detectors. Single-shot object detector models perform object classification and localization in a single pass through the neural network, making them faster and suitable for real-time applications. Two-stage object detector models divide the detection task into two stages: Generate region proposals likely to contain objects. Perform classification and refinement of these proposals. We went through papers from paperswithcode.com in order to better understand both of these types. We found that both are highly precise and have a popular module implementation and hence we made the choice to further look into single-shot detection as it is used popularly for real-time detection due to reasonable tradeoff between accuracy and speed. There is also a third, recently introduced paradigm, that is the use of DeTrs (Detection Transformers) for real-time detection. We further looked into its viability and accuracy in comparision to single-shot detection. Detrs initially had convergence problems and was computationally costly and hence not suitable for real-time detection. But *a recent paper* introduced a newer transformer which eliminated these problems, made detrs a viable option.

A. YOLOV11: AN OVERVIEW OF THE KEY ARCHITECTURAL ENHANCEMENTS

YOLOv11 introduces several architectural enhancements aimed at improving accuracy, efficiency, and versatility across various computer vision (CV) tasks, such as object detection, instance segmentation, pose estimation, and oriented object detection (OBB).

It builds upon its predecessors, offering improvements in mean Average Precision (mAP), computational efficiency, and scalability for different deployment environments.

Achieves higher accuracy and faster inference than previous YOLO models, demonstrating substantial mAP gains on the COCO dataset.

YOLOv11m, a mid-sized variant, achieves superior mAP while using fewer parameters than its counterparts like YOLOv8m.

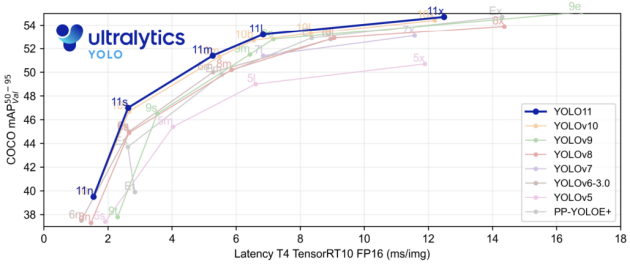

Fig. 1: Benchmarking YOLOv11 Against Previous Versions

B. DETRs Beat YOLOs on Real-time Object Detection

1) Challenges with YOLO Models:
   - YOLO models rely on Non-Maximum Suppression (NMS) for post-processing, which slows inference and introduces instability due to hyperparameters.
   - Selecting appropriate NMS thresholds for different scenarios (emphasizing recall vs. accuracy) is complex and limits the development of real-time detectors.
2) Limitations of Transformer-based DETRs:

- While DETRs eliminate NMS and hand-crafted components, their high computational cost hinders real-time performance.
- Current DETRs fail to exploit the advantages of their NMS-free design in real-time applications.

3) Proposed Solution: Redesigning DETRs for Real-Time Use:
   - Encoder Redesign: Multi-scale features in DETRs improve training convergence but increase sequence length, making the encoder a computational bottleneck. A redesigned encoder is essential for real-time performance.
   - Improved Query Selection: Current query selection methods rely only on classification scores, neglecting the need to model both object category and location. This results in low-quality queries and reduced performance. A focus on better query initialization is identified as a key area for improvement.
   - Flexible Speed Tuning: Allows performance adjustments across scenarios by modifying decoder layers without retraining.

This motivates the development of RT-DETR, a real-time DETR framework addressing these issues to achieve superior speed and accuracy in object detection.

C. Dataset compilation

The datasets were obtained from roboflow universe, in particular the 'badminton Object Detection' and the 'table tennis table Object Detection' dataset. These datasets were augmented using albumentations.

IV. System Architechture

The below figure is not available on official channels, this can be obtained through analysing the source code of the YOLO11 model.

A. Conv layer

Conv layers computes the dot product between the input image and kernel by moving the kernel across the image to detect local features at different positions. Mathematically, we get the large value in the feature maps where the template of the filter is found in the input image. In the figure, k refers to the number of kernels and s refers to the stride.

B. C3K2 layer

YOLOv11 uses C3K2 blocks to handle feature extraction at different stages of the backbone. The smaller 3x3 kernels allow for more efficient computation while retaining the model's ability to capture essential features in the image. At the heart of YOLOv11's backbone is the C3K2 block, which is an evolution of the CSP (Cross Stage Partial) bottleneck introduced in earlier versions. The C3K2 block optimizes the flow of information through the network by splitting the feature map and applying a series of smaller kernel convolutions (3x3), which are faster and computationally cheaper than larger kernel convolutions.

The C2F block (Cross Stage Partial Focus, CSP-Focus),which was used in YOLO v8, is derived from CSP network, specifically focusing on efficiency and feature map preservation. This block contains a Conv Block then splitting the output into two halves (where the channels gets divided), and they are processed through a series of 'n' Bottle Neck layers and lastly concatinates every layer output following with a final Conv block. This helps to enhance feature map connections without redundant information.

The C3K block contains a similar structure to C2F block but no splitting will be done here, the input is passed through a Conv block following with a series of 'n' Bottle Neck layers with concatinations and ends with final Conv Block.

The C3K2 uses C3K block to process the information. It has 2 Conv block at start and end following with a series of C3K block and lastly concatinating the Conv Block output and the last C3K block output and ends with a final Conv Block.This block focuses on maintaining a balance between speed and accuracy, leveraging the CSP structure.

C. SPPF (Spatial Pyramid Pooling Fast)

SPPF pools features using multiple max-pooling operations (with varying kernel sizes) to aggregate multi-scale contextual information. This module ensures that even small objects are recognized by the model, as it effectively combines information across different resolutions. The inclusion of SPPF ensures that YOLOv11 can maintain real-time speed while enhancing its ability to detect objects across multiple scales.

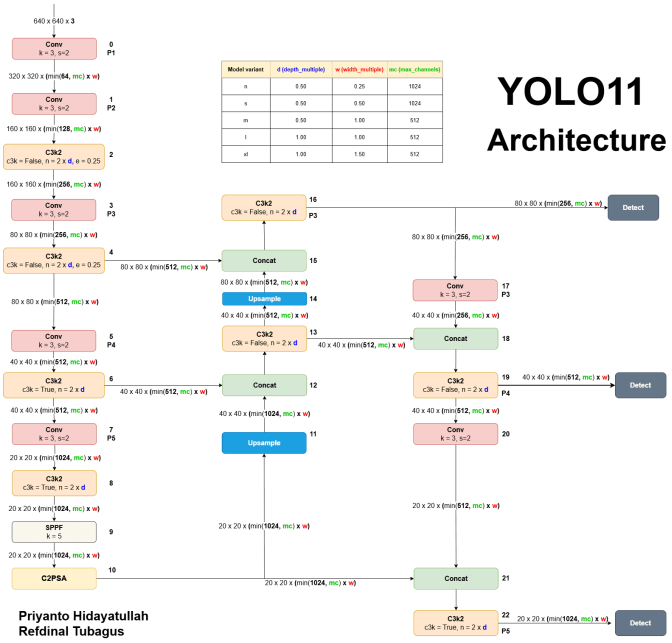

Fig. 2: YOLO11 Architechture

D. C2PSA layer

One of the significant innovations in YOLOv11 is the addition of the C2PSA block (Cross Stage Partial with Spatial Attention). This block introduces attention mechanisms that

improve the model's focus on important regions within an image, such as smaller or partially occluded objects, by emphasizing spatial relevance in the feature maps. The C2PSA block uses two PSA (Partial Spatial Attention) modules, which operate on separate branches of the feature map and are later concatenated, similar to the C2F block structure. This setup ensures the model focuses on spatial information while maintaining a balance between computational cost and detection accuracy. The C2PSA block refines the model's ability to selectively focus on regions of interest by applying spatial attention over the extracted features. This allows YOLOv11 to outperform previous versions like YOLOv8 in scenarios where fine object details are necessary for accurate detection.

E. Upsampling layer

Upsampling, otherwise known as oversampling, is a data processing and optimization technique that addresses class imbalance in a dataset by adding data. Upsampling adds data by using original samples from minority classes until all classes are equal in size.

YOLO uses upsampling layers to merge features from previous layers, providing a richer feature set for detection at different scales. This is especially beneficial for detecting smaller objects.

F. Head

The head of YOLOv11 includes several CBS (Convolution-BatchNorm-Silu) layers after the C3k2 blocks. These layers further refine the feature maps by:

- Extracting relevant features for accurate object detection.
- Stabilizing and normalizing the data flow through batch normalization.
- Utilizing the Sigmoid Linear Unit (SiLU) activation function for non-linearity, which improves model performance.

CBS blocks serve as foundational components in both feature extraction and the detection process, ensuring that the refined feature maps are passed to the subsequent layers for bounding box and classification predictions.

Each detection branch ends with a set of Conv2D layers, which reduce the features to the required number of outputs for bounding box coordinates and class predictions. The final Detect layer consolidates these predictions, which include:

- Bounding box coordinates for localizing objects in the image.
- Objectness scores that indicate the presence of objects.
- Class scores for determining the class of the detected object.

V. Implementation

The first and primary objective was to work on the detection and training models. Both created models are based on YOLOv11, and have been trained manually on databases curated by ourselves. One of the models is trained on badminton racket detection alone, with an inclusion of object tracking. The other model is trained on detecting all classes of objects around a table tennis court, including the ball, the players, etc. We have also included the option to select the YOLOv11 model itself if that is so preferred. This allows an expandibility of operations to other more general crowd detection situations.

Our GUI is built on Streamlit, we were choosing between PyQT and Streamlit. We finally settled on Streamlit due to the higher convenience and availability of several pre-built functions like file_uploader were useful. The app allows users to select the model confidence as well as the models. While in the video and youtube url options, there is also an option to enable and choose an object tracker from YOLOv11's two options, ByteTrack and BotSort.

The function _display_detected_frames detects objects on the passed frames, draws bounding boxes and labels them with their detected classes. We also incorporated a text display on the frames, that will show the number of detected objects of each class in each frame. The trackers also utilise the persist option available in YOLOv11 models, which prevents the detection of still objects in video frames, which itself solves the problem of detecting 'active' players.

https://github.com/SidPam/EpochSubmission

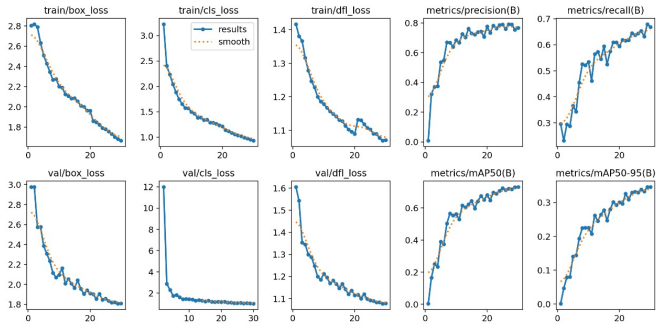

Fig. 3: Results on training for 30 epochs

VI. Issues Faced/Addressed

- Training data is obtained from matches with fixed camera angle and only single court in focus. Since we are planning to integrate the model with CCTV cameras, testing is yet to be performed on the same. Although a wide variety of available videos were used for training.
- Current UI only supports video footage, real-time footage is yet to be supported. A web-cam option has been enabled which works accurately, due to a lack of real-time connection options, no option has been enabled yet.
- The current testing data has not been tested on crowded courts. This was well resolved through the object tracker option, which utilises persist to prevent detecting crowd options.

VII. Results

- mAP50 is 0.73
- mAP50-95 is 0.347

The above results are for the badminton racket model.

- mAP50 is 0.952

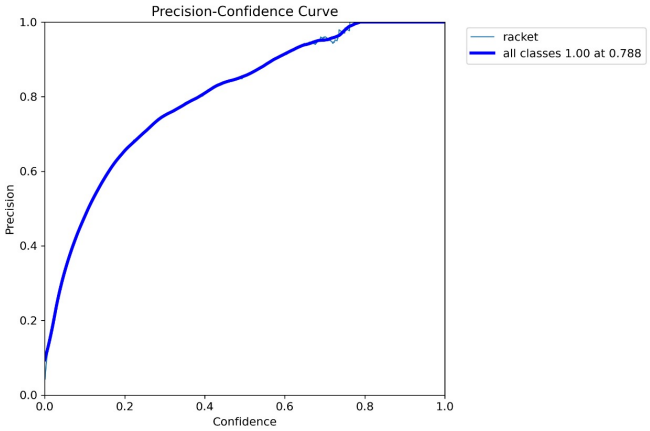

Fig. 4: Precision curve (badminton racket)

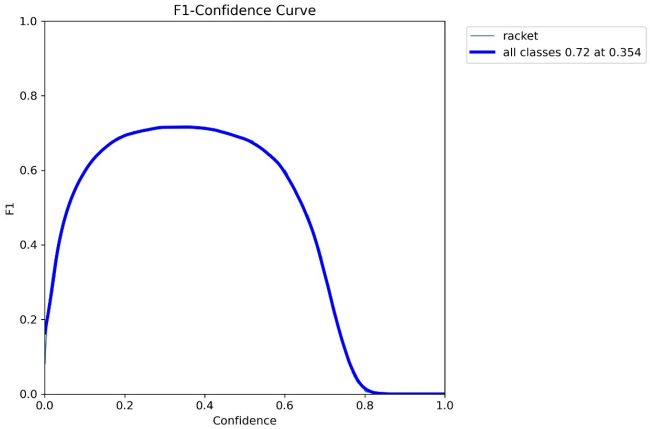

Fig. 5: F1 curve (badminton racket)

- mAP50-95 is 0.818 (we believe this is due to less training data)

The above results are for the table tennis model.

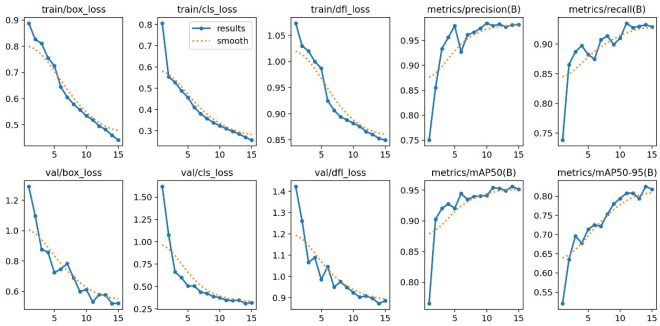

Fig. 6: Table Tennis model results

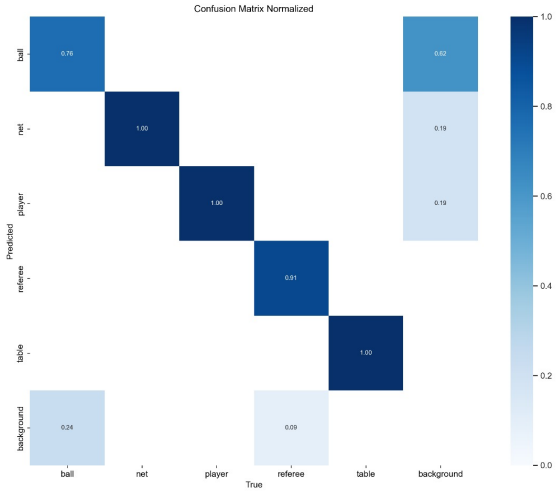

Fig. 7: Confusion Matrix for Table Tennis model

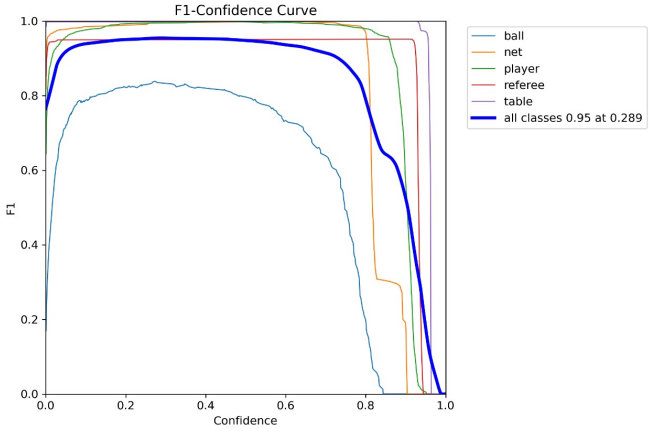

Fig. 8: F1 curve (Table Tennis Model)

REFERENCES

[1] Jocher, G., & Qiu, J. (2024). Ultralytics YOLO11 [Software]. Retrieved from https://github.com/ultralytics/ultralytics (Version 11.0.0) [AGPL-3.0 License]. https://github.com/ultralytics/ultralytics 11.0.0 AGPL-3.0

[2] Rahima Khanam and Muhammad Hussain, "YOLOv11: An Overview of the Key Architectural Enhancements," in *Proceedings of the [Conference Name]*, 2024, arXiv:2410.17725 [cs.CV]. Available: https://arxiv.org/abs/2410.17725.

[3] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen, "DETRs Beat YOLOs on Real-time Object Detection," arXiv:2304.08069 [cs.CV], 2024. Available: https://arxiv.org/abs/2304.08069.

[4] *YOLO11 Architechture - Detailed Explanation*, YouTube, 2024. [Online]. Available: https://www.youtube.com/watch?v=L9Va7Y9UT8E. [Accessed: Jan. 5, 2025].

[5] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush, "Transformers: State-of-the-Art Natural Language Processing," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online, 2020, pp. 38–45. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[6] https://rs-punia.medium.com/