*Advances in Technology Innovation, vol. 9, no. 3, 2024, pp. 157-171*

# An Efficient Application of Modified YOLOv5 in Basketball Player Detection and Analysis

Jia-Shing Sheu*, Sheng-Ju Lin

Department of Computer Science, National Taipei University of Education, Taipei, Taiwan, ROC

Received 09 May 2024; received in revised form 14 July 2024; accepted 15 July 2024

DOI: https://doi.org/10.46604/aiti.2024.13702

## Abstract

Effective analysis helps players evaluate their performance, make necessary adjustments, and develop diverse game strategies. Moreover, the analysis provides viewers with different perspectives, enhancing their understanding of the game. This study aims to develop a basketball player detection and analysis system to assist in analyzing on-court situations. The system uses perspective transformation to obtain player tracking information on the top view image in basketball games. The system uses a modified you only look once (YOLO) v5 model that replaces the backbone of YOLOv5s with the MobileNetv3-small architecture for player detection. Compared to the original YOLOv5, the modified YOLOv5 reduces parameters from $7.02 \times 10^6$ to $3.5 \times 10^6$, a decrease of 49.8%. The number of frames obtainable per second increases from 12.4 to 17.5, an improvement of about 41.1%. Finally, the system performs perspective transformation and tracks the detected player positions onto the top-view court image using the YOLOv5 model.

**Keywords:** player detection, player tracking, basketball game, YOLOv5

## 1. Introduction

Image processing is a major research area in computer science that involves techniques such as image analysis, enhancement, and reconstruction. Common image processing techniques include edge detection, color model conversion, and image filtering. These techniques have applications in various fields, including artificial intelligence, autonomous driving [1-3], virtual reality, defect detection [4-5], and medicine [6-7].

Object detection, a crucial research area in computer vision, involves accurately identifying and locating specific objects in images. This technology has been extensively used in fields such as autonomous driving, medical imaging, and security surveillance. Deep learning significantly improves object detection performance. Traditional methodologies for object detection include histograms of oriented gradients (HOG) and AdaBoost [8]. Deep learning approaches that have been employed to enhance object detection include convolutional neural networks (CNNs), you only look once (YOLO), single shot multibox detector (SSD) [9], and MobileNet.

This study focuses on the following questions: how to determine the court boundaries to facilitate the accurate projection of player positions onto the court view, what methods to use for player detection, how to confirm player positions and project them onto the court view, and how to improve the model to increase efficiency and accuracy. Therefore, this study proposes a basketball player detection system based on YOLOv5. It employs techniques such as the hue saturation value (HSV) model, Canny edge detection, and Hough transform to confirm basketball court boundaries. The study conducts player detection by

* Corresponding author. E-mail address: jiashing@tea.ntue.edu.tw

English language proofreader: Yen-Chun Hsieh

using a modified YOLOv5 model and projects player position information onto a top-view basketball court image through perspective transformation to obtain the desired planar information. The modified YOLOv5 improves frames per second (FPS) by approximately 41.1%.

The remainder of this article is organized as follows: Section 2 describes the relevant literature, Section 3 describes the system architecture, Section 4 presents a description of the experiments and the experiment results, and Section 5 presents the conclusions.

## 2. Literature Review

In recent years, many object detection methods have been continuously introduced, and due to the widespread use of mobile devices and diverse application scenarios, how to lightweight models for deployment on mobile platforms has become a highly researched topic. The popular object detection methods in recent years are summarized in Table 1. Dalal and Triggs [10] introduced HOG, a widely utilized feature descriptor method in image processing and computer vision in 2005. HOG divides an image into small blocks and computes the gradient magnitude based on the pixel gradient directions within each block, with this information considered to be the block's feature. It addresses the feature extraction problem in pedestrian detection. The HOG feature vector is frequently combined with a support vector machine (SVM) for training to perform object detection tasks.

Table 1 Methodologies of object detection

| | | |
|---|---|---|
| Object detection | Traditional methods | HOG |
| | | Adaboost |
| | Deep learning | CNN |
| | | YOLO |
| | | SSD |
| | | MobileNet |

The YOLO algorithm [11] is a popular real-time object detection algorithm that employs one-stage detection. In such detection, an image is examined only once to recognize and locate objects, and therefore, YOLO offers efficiency advantages in object detection. YOLO divides each image into a fixed-size grid and analyzes each grid for potential objects and their positions. Each grid prediction comprises the probability of belonging to a particular object, the location of the bounding box, and the object's class.

MobileNet [12], an optimized network architecture specifically designed for mobile devices, is lightweight and efficient, therefore, it is ideal for deep learning on mobile or embedded devices. The core innovation of this architecture is the use of depthwise separable convolution, which decomposes traditional convolution into depthwise and pointwise convolution steps, thereby reducing computation and parameters and resulting in a lighter model. MobileNetv2 [13] is an improvement on MobileNet because it includes Inverted Residual blocks and Linear Bottlenecks. MobileNetv3 [14] builds on MobileNetv2 because it employs a squeeze and excitation structure [15]; through global average pooling, it calculates each feature map's weights to improve the model's focus on crucial features and thereby improve its recognition performance.

In 2019, Google introduced EfficientNet [16], which has the core concept of compound model scaling. EfficientNet is based on compound coefficients, which enable uniform scaling of the network's depth, width, and resolution in its architecture. Although individually increasing depth, width, or resolution can improve model accuracy, saturation tends to occur once accuracy reaches approximately 80%, which can make further improvement challenging. To enhance accuracy, compound model scaling balances the network's depth, width, and resolution by adjusting them simultaneously based on compound coefficients, which improves the overall model performance. EfficientNetv2 [17], an advancement over EfficientNet, addresses problems such as slow training speeds associated with large input sizes through the introduction of the Fused-MBConv structure in certain modules.

In contemporary sports, performance analysis through computer vision has emerged as a critical area of research. Player detection in sports scenes has wide applicability, including in sports such as ice hockey and basketball. Researchers proposed a two-stage CNN model for detecting players in ice hockey [18]. In addition, for basketball, researchers used YOLOv2 for player position tracking [19] and trained a two-layer long short-term memory model [20] for player action recognition. In addition, researchers used combinations of HOG and SVM for player detection [21], followed by the transformation of player positions to a top view for analysis [22]. In addition, AdaBoost was proposed for player detection [23]; however, experimental results indicated that its effectiveness was below expectations and that it was unsuitable for object detection in sports events.

## 3. System Structure and Methodology

This study structures its system by using the integrated definition for function modeling (IDEF0) modeling method, which is based on the structured analysis and design technique, to describe system functionality. IDEF0 can help organizations analyze and understand their processes or systems, providing a clearer understanding of how a system works. As illustrated in Fig. 1, the system (A0) comprises three submodules designed to achieve player detection and tracking: court detection, YOLOv5 player detection, and player position localization. The purpose of the court detection module is to define the court boundaries using various image processing methods. Then the YOLOv5 player detection module identifies the player positions. Finally, the player position localization module projects the player positions onto the top-view image of the court to obtain the required planar information.

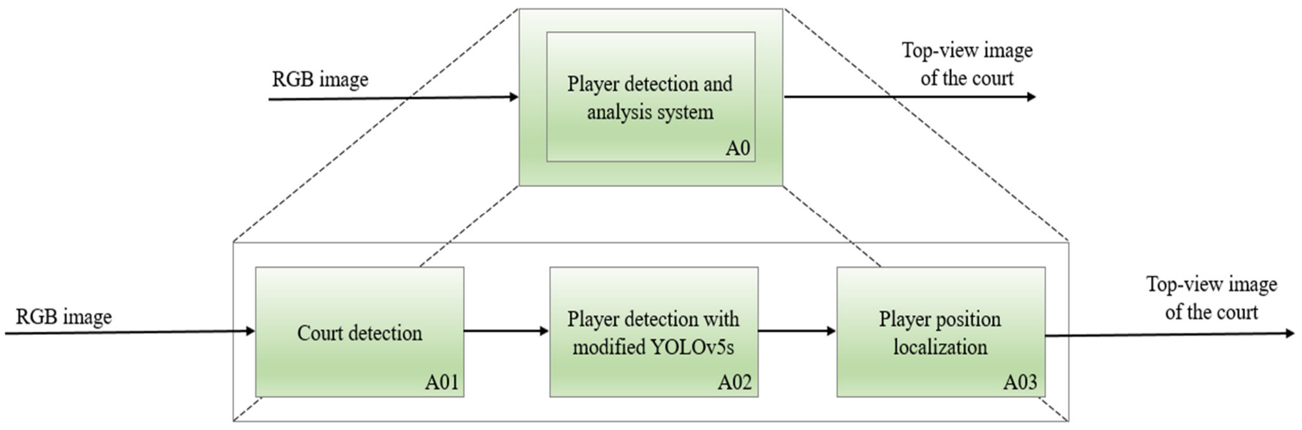
RGB image
Player detection and analysis system
A0
Top-view image of the court
RGB image
Court detection
A01
Player detection with modified YOLOv5s
A02
Player position localization
A03
Top-view image of the court
Fig. 1 IDEF0 for player detection and analysis system

Fig. 2 displays the court detection module (A01), which takes red, green, and blue (RGB) images from basketball videos as input. It converts the RGB color model to the HSV color model, uses the Canny edge detection algorithm to identify basketball court edges, and uses the Hough transform to define court boundaries. After summarizing these steps, the module obtains an image confirming the basketball court's extent, which is used for subsequent perspective transformation.

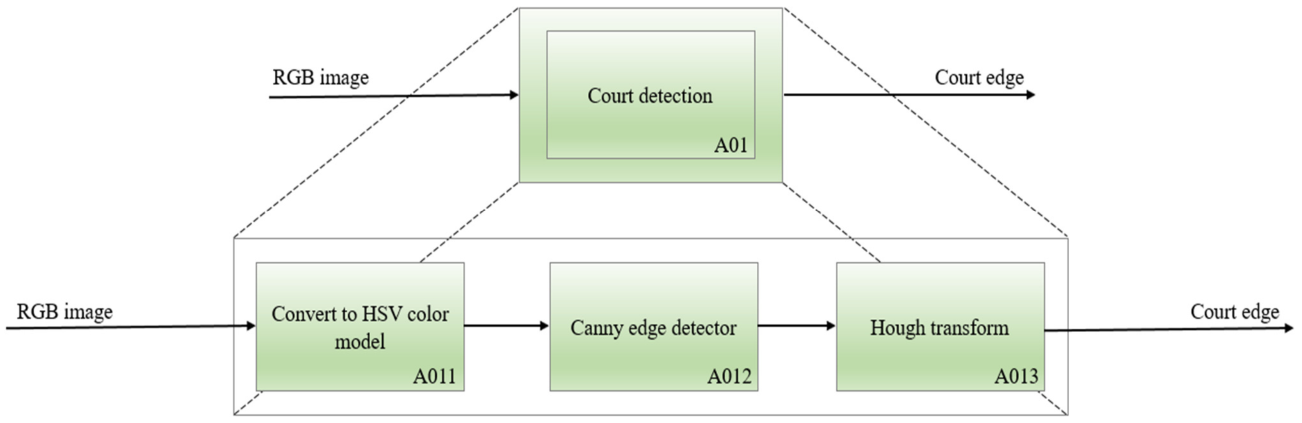

Fig. 2 IDEF0 for court detection

The study converts images from the RGB to the HSV color model because of the ease of extracting information with specific color ranges in the HSV color model (A011). In the HSV color model, ($R'$, $G'$, $B'$) represent the coordinates of RGB. They are normalized by dividing each value by 255, which ensures that values range from 0 to 1, which accurately describes color information in the HSV color model. The conversion formulas are displayed below.

$$(R', G', B') = \left(\frac{R}{255}, \frac{G}{255}, \frac{B}{255}\right) \tag{1}$$

$V$ represents value, which indicates the intensity of light reflected from the object.

$$V = \max(R', G', B') \tag{2}$$

$S$ represents saturation, which indicates the color's purity.

$$S = \frac{\left[V - \min(R', G', B')\right]}{V} \tag{3}$$

$H$ represents hue, which indicates the color's basic characteristic.

$$H = \begin{cases} 60^\circ \times \left[\dfrac{(G' - B')}{S} + 0^\circ\right], & if\ V = R' \\ 60^\circ \times \left[\dfrac{(B' - R')}{S} + 120^\circ\right], & if\ V = G' \\ 60^\circ \times \left[\dfrac{(R' - G')}{S} + 240^\circ\right], & if\ V = B' \end{cases} \tag{4}$$

This study uses the Canny edge detector (A012) to detect basketball court boundaries. The Canny edge detection algorithm, which is a classic edge detection technique known for its ability to detect edges in images while resisting noise interference, is widely used in various image processing tasks, including object detection, feature extraction, and image segmentation. The algorithm involves four steps: Gaussian smoothing, gradient calculation, nonmaximum suppression, and hysteresis thresholding.

First, before detecting edges in the image, Gaussian blurring is applied to reduce noise, typically using a Gaussian smoothing filter. Second, gradients are computed using Sobel filters to compute the gradient magnitude and direction for each pixel. Third, non-maximum suppression is applied to filter out pixels by retaining only those where the gradient has the maximum magnitude in its direction, preserving finer edges. Finally, double thresholding is applied using two thresholds: a low threshold and a high threshold. These thresholds are used to select edges by either preserving or discarding pixels based on their edge strength.

Hough transform (A013), a technique used to detect geometric shapes such as lines and circles in images, uses a voting mechanism to confirm lines, with line equations typically represented using the slope-intercept form. When a line is perpendicular to the x-axis, the slope becomes infinite, which can lead to computational difficulties. Therefore, the Hough transform uses normal parametrization to represent lines in such cases. A point can be the intersection of infinitely many lines. When these lines are transformed into polar coordinates through plane coordinate transformation, if the curves formed by two points intersect in polar coordinates, it implies that the two points are on the same straight line.

In the player detection component of this study, a modified YOLOv5s model (A02) is applied. The main modification of the model involves changing the backbone architecture of YOLOv5s to those of other models to reduce computation and increase processing speed.

**A. YOLOv5s (A02)**

For the YOLOv5s input, the mosaic data augmentation method is applied, as it is for YOLOv4, and adaptive computation and scaling of anchor boxes are incorporated. The mosaic data augmentation technique combines multiple images by randomly cropping, scaling, and arranging them to increase the diversity of training data. This approach can improve a model's sensitivity to object detection in different scenes and at different angles.

The backbone of the YOLOv5s model mainly extracts image features by converting an input image into multilayer feature maps, which are used for subsequent object detection tasks. Its structure mainly consists of the focus module, the Conv module, the C3 module, and the spatial pyramid pooling-fast (SPPF) module. Fig. 3 illustrates the architecture of the YOLOv5s model.

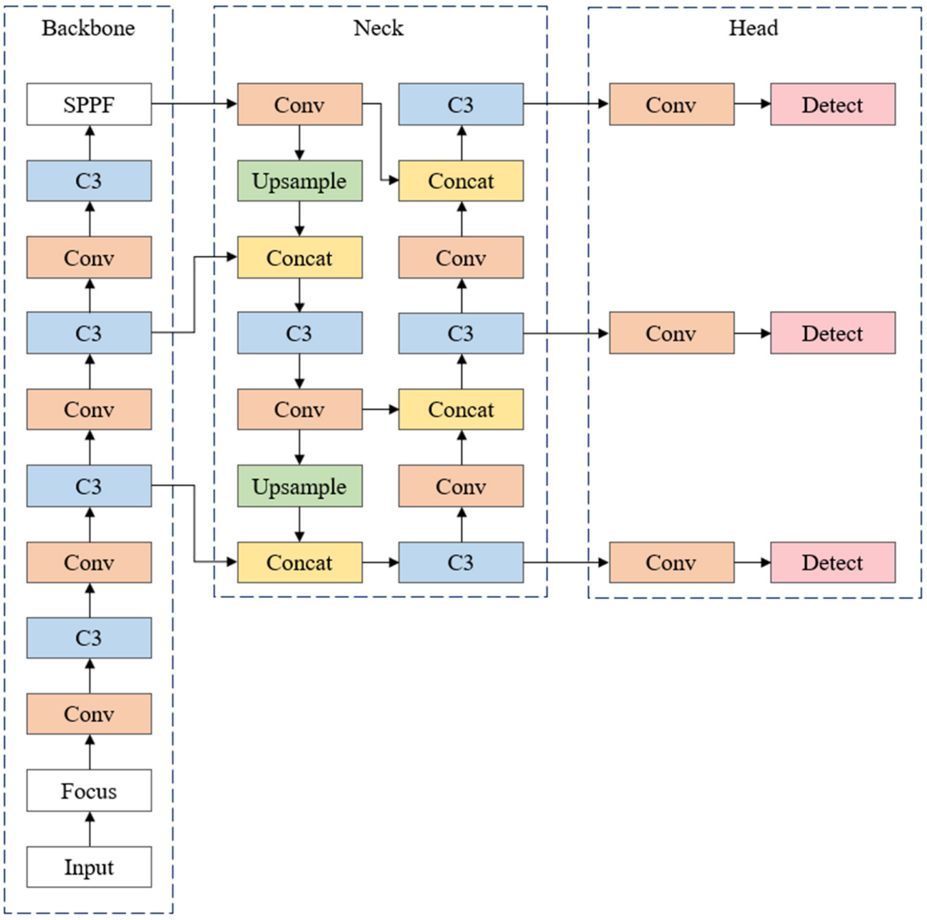

Fig. 3 Architecture of YOLOv5s model

The focus module in YOLOv5s is used for slice operation. It is designed to process input images efficiently by reducing computation while preserving relevant features. For example, after receiving an input image of the size 608 × 608 × 3, the module performs slice operations to transform it into a feature map of the size 304 × 304 × 12. It then performs a convolution operation, which results in a feature map of the size 304 × 304 × 32.

The C3 module comprises three conv modules and several bottleneck modules. Each bottleneck module has two convolutional layers. The C3 module enhances the feature extraction capability by increasing the network's depth. Fig. 4 illustrates the architecture of the C3 module.

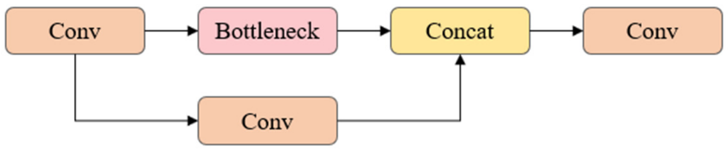

Fig. 4 Architecture of C3 module

SPPF is an improved version based on spatial pyramid pooling (SPP). It processes input through multiple MaxPool layers of different sizes to obtain feature information at different scales. These features are then merged. This is completed to address the multiscale problem in object recognition tasks. Fig. 5 illustrates the architecture.

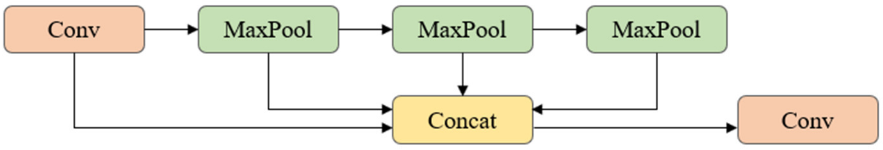

Fig. 5 Architecture of the SPPF module

The neck of the YOLOv5s model is used for feature fusion and enhancement. It uses the feature pyramid network (FPN) and path aggregation network (PAN) structure to combine image features and pass them to the prediction layer. In the FPN structure, information flows from top to bottom, with features from higher layers merged with those from lower layers through upsampling. PAN is a feature fusion structure that adds a layer of bottom–top feature fusion following the FPN process. Fig. 6 illustrates the architectures.

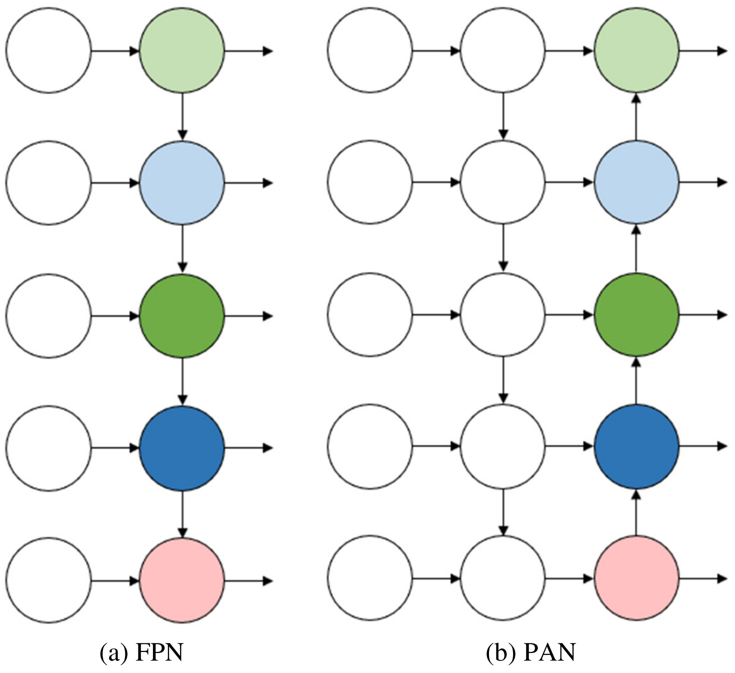

Fig. 6 Architectures of FPN and PAN

The head of YOLOv5s is responsible for predicting image features and generating bounding boxes to predict object classes. The detection layer comprises several components, including anchor boxes, convolutional layers, prediction layers, and nonmaximum suppression.

## B. MobileNetv3

MobileNetv3 is a lightweight CNN that mainly uses depthwise separable convolution to reduce the number of model parameters while maintaining accuracy. In this study, the backbone of YOLOv5s was replaced with the MobileNetv3-small architecture, named Mob_YOLOv5s, to reduce the number of parameters and achieve more favorable computational speed. Fig. 7 illustrates the MobileNetv3-small architecture, and Fig. 8 displays the Mob_YOLOv5s architecture.

Depthwise separable convolution can be divided into two components: depthwise convolution and pointwise convolution. Depthwise convolution performs separate convolutions on each channel individually. As displayed in Fig. 9, each channel is convolved with a separate kernel. Pointwise convolution applies a $1 \times 1$ convolution kernel to the feature map obtained from depthwise convolution. A flow diagram of this process is presented in Fig. 10.

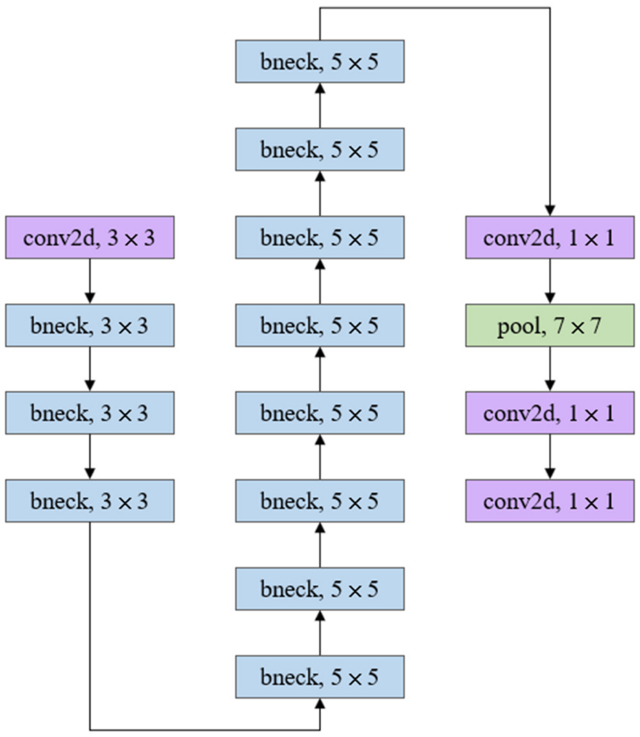

Fig. 7 Architecture of MobileNetv3-small

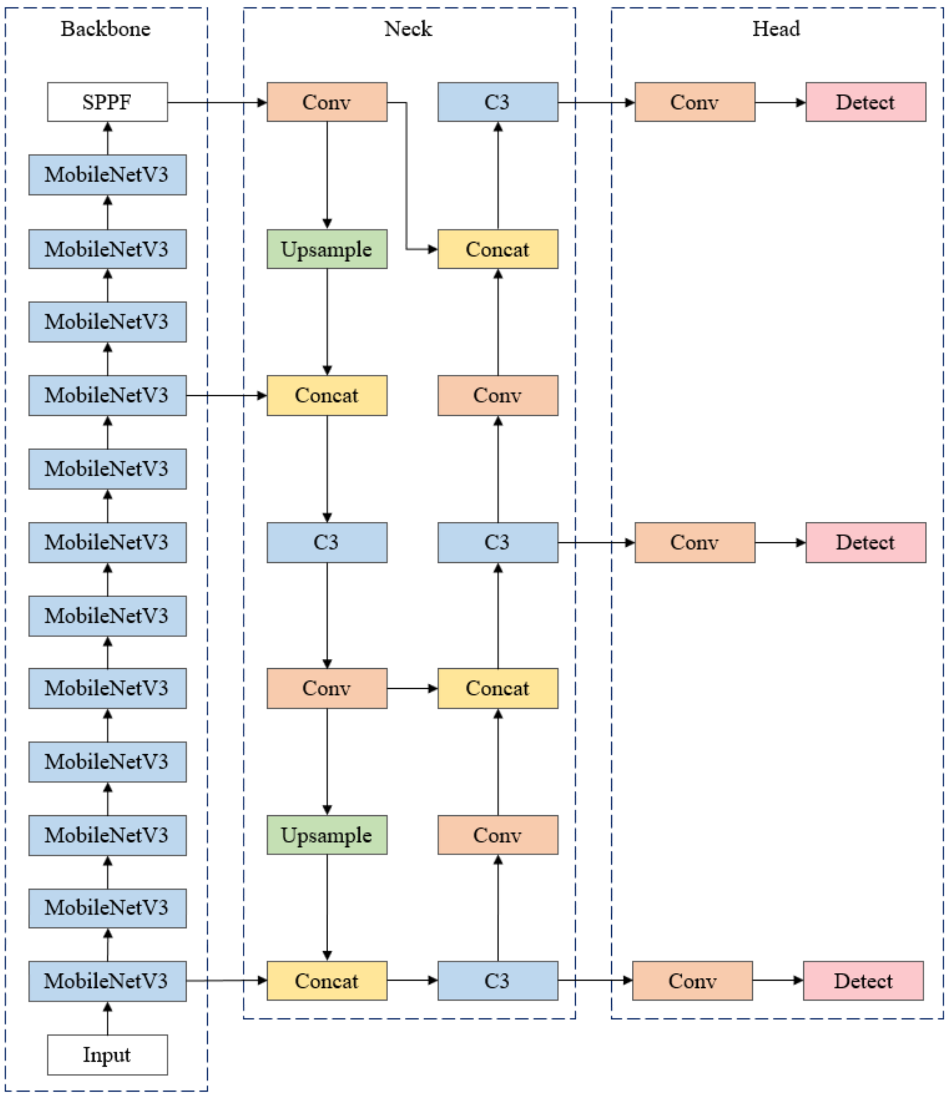

Fig. 8 Architecture of Mob_YOLOv5s

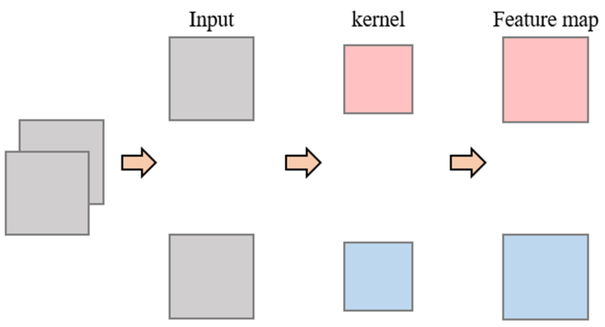

Fig. 9 Flow diagram of depthwise convolution

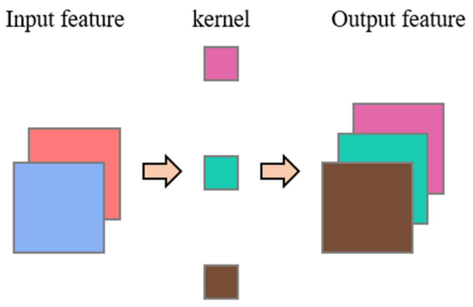

Fig. 10 Flow diagram of pointwise convolution

As described in the following formula, $S$ represents the computational cost of the standard convolution, $D_K$ represents the kernel size, $D_F$ represents the feature map size, $M$ represents the number of input channels, and $N$ represents the number of output channels.

$$S = D_K \times D_K \times M \times N \times D_F \times D_F \tag{5}$$

The computational cost of depthwise convolution is denoted as $M_d$, and that of pointwise convolution is denoted as $M_p$, as presented below:

$$M_d = D_K \times D_K \times M \times D_F \times D_F \tag{6}$$

$$M_p = M \times N \times D_F \times D_F \tag{7}$$

Relative to standard convolution, depthwise separable convolution reduces the computational cost by $1/N + 1/D_k^2$, as illustrated in:

$$\frac{M_d + M_p}{S} = \frac{D_K \times D_K \times M \times D_F + M \times N \times D_F \times D_F}{D_K \times D_K \times M \times N \times D_F \times D_F} = \frac{1}{N} + \frac{1}{D_k^2} \tag{8}$$

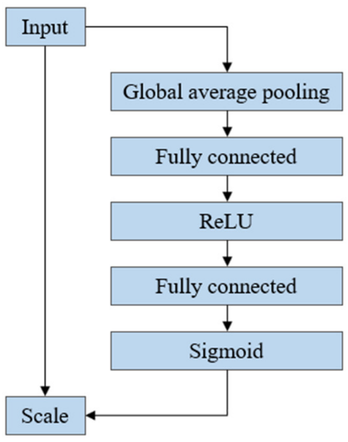

Fig. 11 Architecture of SE module

The introduction of the squeeze-and-excitation (SE) module in MobileNetv3 is another improvement. The SE module is an attention mechanism designed to improve the performance of deep neural networks. Its core concept involves squeeze and excitation, enhancing crucial features while ignoring irrelevant ones, thereby improving the feature extraction ability and overall performance of a model. Fig. 11 illustrates the SE architecture.

During the compression phase, the SE module aggregates information from each channel of the input feature map through global average pooling to obtain global statistical information for each channel. This step helps with evaluating the importance of each channel. In the excitation phase, the focus is further extracting the features obtained in the compression phase. The SE module uses two fully connected layers to reduce the number of channels and parameters. Rectified linear unit and sigmoid activation functions are applied to the output of the fully connected layers. The formulas are presented below.

$$ReLU\left(x\right) = \max\left(0, x\right) \tag{9}$$

$$Sigmoid\left(x\right) = \frac{1}{1+e^{-x}} \tag{10}$$

## C. EfficientNetv2

Proposed in 2021, EfficientNetv2 is an iteration of EfficientNet that addresses the slow training speeds associated with training using large images in EfficientNet. EfficientNetv2 mainly differs from EfficientNet in the following four aspects:

(1) Module usage: In addition to using the MBConv module, EfficientNetv2 uses the Fused-MBConv module. The MBConv module uses depthwise separable convolution, which has fewer parameters and computations and is suitable for mobile platforms. However, it cannot fully leverage modern accelerators. Therefore, some modules are replaced with Fused-MBConv to improve computational performance.

(2) Expansion ratio: EfficientNetv2 uses smaller expansion ratios in the MBConv module because smaller expansion ratios can reduce memory consumption.

(3) Convolution kernel size: EfficientNet uses many 5 × 5 convolution kernels, whereas EfficientNetv2 uses 3 × 3 convolution kernels.

(4) Layer removal: EfficientNetv2 removes the last MBConv module of EfficientNet to reduce parameter and memory consumption.

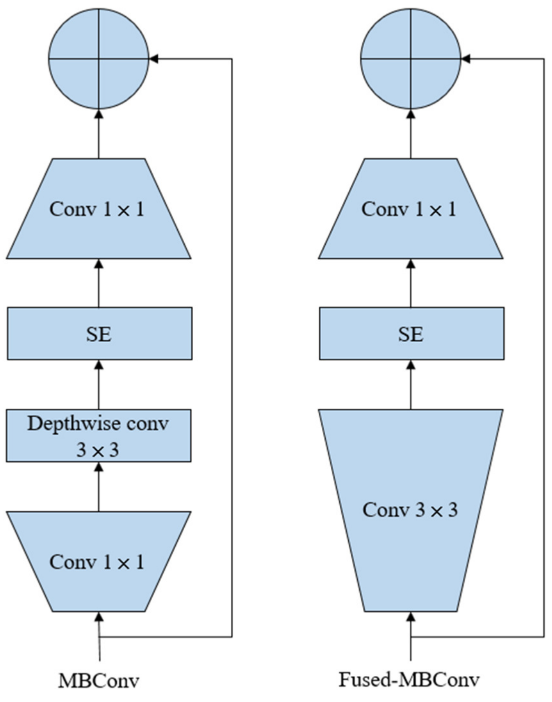

Fig. 12 Architectures of MBConv and Fused-MBConv

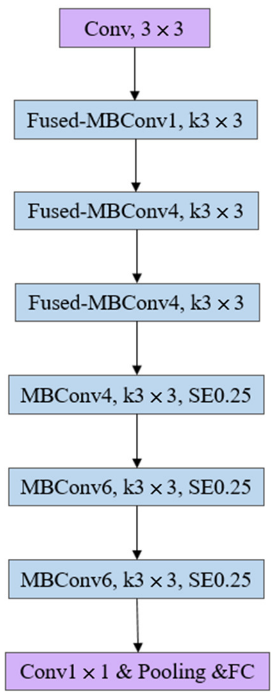

Fig. 13 Architecture of EfficientNetv2 model

Fused-MBConv, an enhanced version of MBConv, further improves the computational efficiency and performance of a model. It merges the depth-separable convolution in MBConv into a single standard convolution (Conv 3 × 3). Fig. 12 illustrates the architectures of MBConv and Fused-MBConv, and Fig. 13 illustrates the architecture of EfficientNetv2.

In this study, the original YOLOv5s backbone was replaced with the EfficientNetv2 architecture to create Eff_YOLOv5s to integrate the computational efficiency of EfficientNetv2 with YOLOv5s. The goal is to optimize YOLOv5s by utilizing the computational efficiency of EfficientNetv2 to achieve improved computational speed and performance. Additionally, the study compares the effects of different backbones on computational efficiency. Fig. 14 illustrates the architecture of Eff_YOLOv5s.

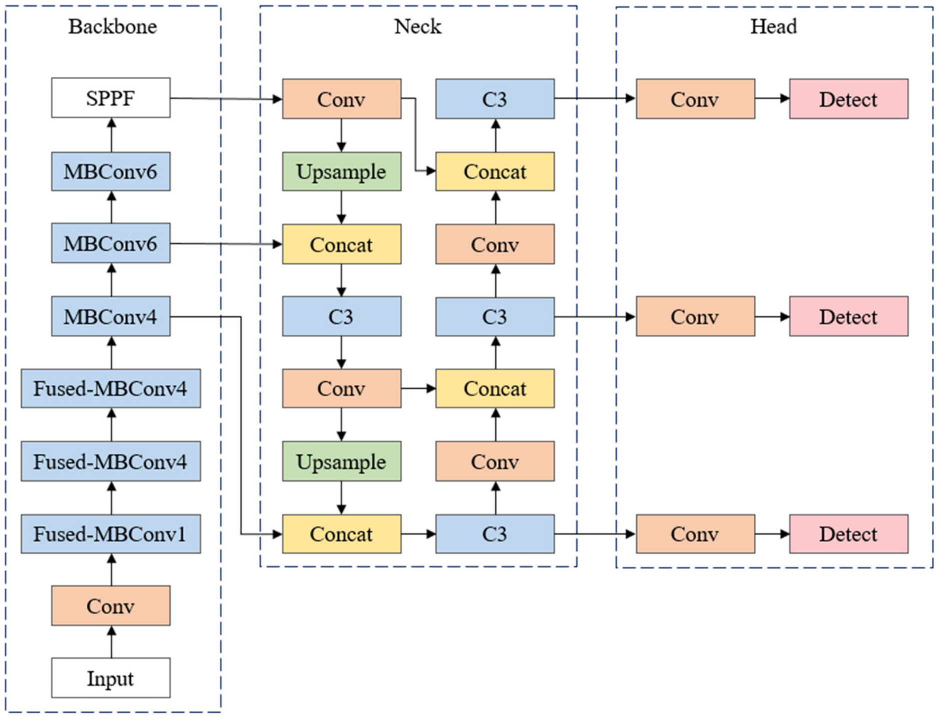

Fig. 14 Architecture of Eff_YOLOv5s

## D. Locate player locations (A03)

This study uses a perspective transformation to track and record the positions of players detected by YOLOv5s. Perspective transformation refers to the process of projecting a point from a three-dimensional real-world space onto a two-dimensional plane. Given that a basketball court is effectively a plane, the transformation of real-world coordinates to image plane coordinates represents a projection from one plane to another. The formula for the perspective transformation is presented in:

$$p = Hp' \tag{11}$$

where $H$ is the perspective transformation matrix that projects the position coordinates $p'$ to the desired top view coordinates $p$. This formula can be expressed as follows:

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{pmatrix} h_{00} & h_{01} & h_{02} \\ h_{10} & h_{11} & h_{12} \\ h_{20} & h_{21} & h_{22} \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ w' \end{pmatrix} \tag{12}$$

Because the basketball court typically lacks distinct features, this study uses the four corners of the court that are visible to the camera as reference points. Fig. 15 illustrates the top view of the field, with the red dots representing $p$, and Fig. 16 presents the real-world view of the court, with the green dots representing $p'$. The $H$ matrix is derived from coordinates $p$ and $p'$, the player coordinates in the image are multiplied by this matrix to yield their corresponding top-view positions.

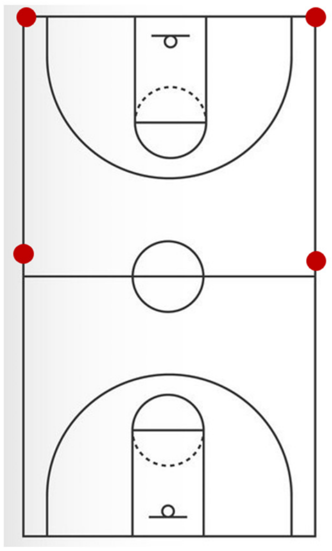
Fig. 15 Top view image

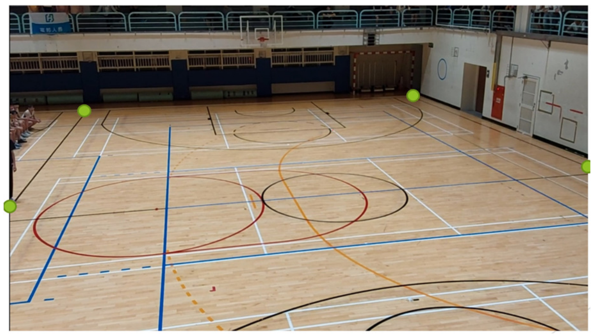
Fig. 16 Real-world view of the court

## 4. Experimental Results

This section presents the results to evaluate the proposed approach. The experiments were performed using an NVIDIA GeForce RTX 3070 GPU, significantly accelerating the training process. Detailed software and hardware system configurations used in the experiments are provided in Table 2.

Table 2 Hardware equipment

| Item | Specification |
|---|---|
| CPU | AMD Ryzen5 5600x |
| RAM | 16GB DDR4-3200 |
| GPU | NVIDIA GeForce RTX 3070 |
| Operating system | Windows 10 |

The data set used in this study primarily comprises images taken from basketball games played on campus, as illustrated in Fig. 17. The dataset was generated by recording matches and capturing an image every 2 seconds. A total of 2 different match videos were collected. The data set comprises 1270 images, with 960 used for training, 200 used for validation, and 100 used for testing, accounting for approximately 76%, 16%, and 8% of the total images, respectively. The epoch is set to 300, and the batch size is set to 4.

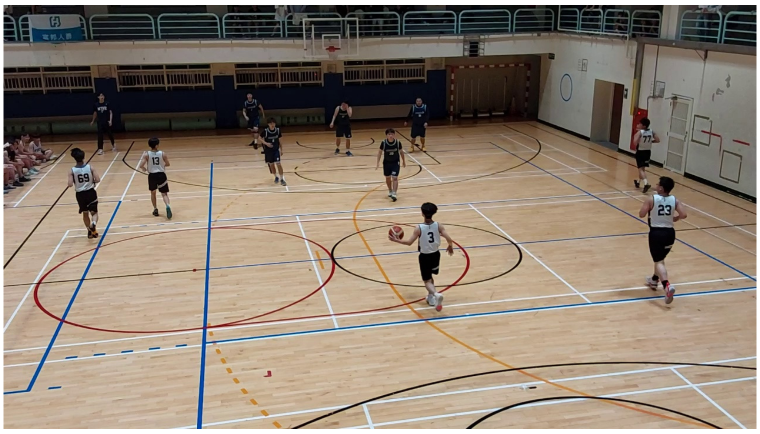
Fig. 17 Image from the data set

Table 3 Confusion matrix

| | Predicted positive | Predicted negative |
|---|---|---|
| Actual positive | TP | FN |
| Actual negative | FP | TN |

A confusion matrix was mainly used to evaluate the training results in this study (Table 3). The confusion matrix is commonly used in deep learning to evaluate a model's performance based on precision and recall, as illustrated in:

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

Fig. 18 illustrates the line chart of the precision of each model during training, gradually converging after about 40 epochs. YOLOv5s shows slightly higher precision after convergence, about 1% higher than the other two models. Fig. 19 illustrates the line chart of recall of each model during training, where YOLOv5s also shows a slight lead of 1 to 2% over the other two models in terms of recall. In the current study, the mean average precision (mAP) was considered to be the primary indicator of the proposed model's performance. The accuracy of object detection was measured using mAP@0.5, which evaluates the mAP at an intersection over union (IoU) threshold of 0.5. mAP@0.5:0.95 measures the mAP over a range of IoU thresholds from 0.5 to 0.95. A lower IoU threshold indicates less overlap between the detection and target boxes, whereas a higher IoU threshold indicates greater overlap.

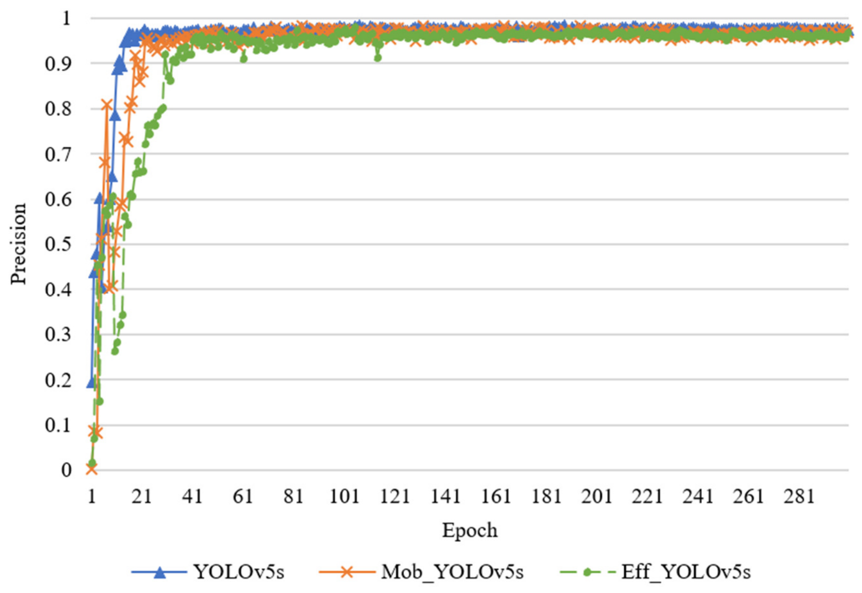

Fig. 18 Line chart of precision of each model

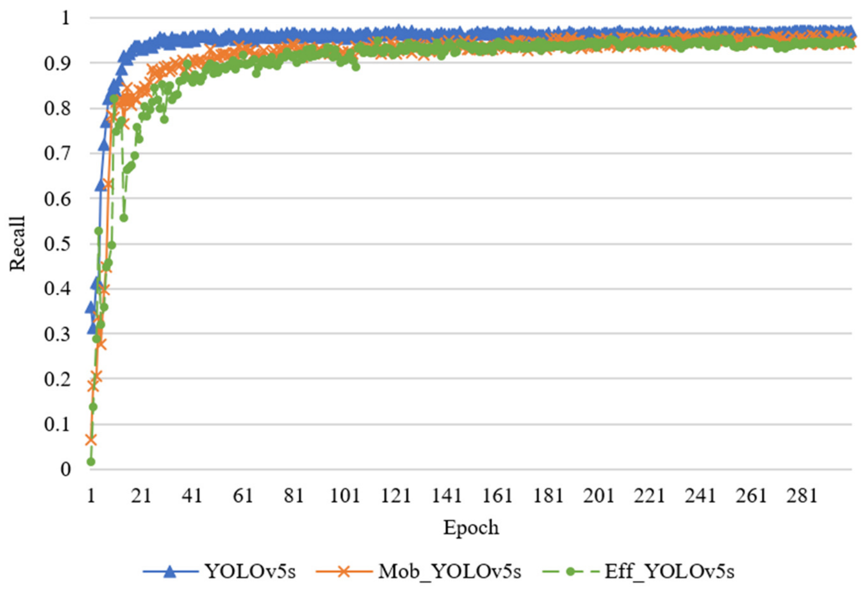

Fig. 19 Line chart of recall of each model

The study trained three models, that is, YOLOv5s, Mob_YOLOv5s, and Eff_YOLOv5s, on the data set. The main evaluation metrics that were used were precision, recall, mAP@0.5, and mAP@0.5:0.95. Table 4 compares the training results for each model. Table 5 presents a comparative analysis of the performance of each model, focusing mainly on differences in the parameters, model size, giga floating point operations per second (GFLOPs), and FPS.

Table 4 Comparison table of training results for each model

| Model / Date | YOLOv5s | Mob_YOLOv5s | Eff_YOLOv5s |
|---|---|---|---|
| Precision | 97.5% | 97.4% | 95.8% |
| Recall | 96.9% | 94.4% | 94.7% |
| mAP@0.5 | 98.3% | 98.2% | 97.4% |
| mAP@0.5:0.95 | 80.9% | 72.3% | 74.5% |

Table 5 Performance comparison chart of each model

| Model / Date | YOLOv5s | Mob_YOLOv5s | Eff_YOLOv5s |
|---|---|---|---|
| Parameters | $7.02 \times 10^6$ | $3.5 \times 10^6$ | $5.4 \times 10^6$ |
| Model size (MB) | 13.7 | 7.08 | 10.6 |
| GFLOPs | 15.8 | 6.3 | 6.9 |
| FPS (CPU) | 12.4 | 17.5 | 13.7 |

After conducting an extensive comparison, it is evident that the performance of the original YOLOv5s and Mob_YOLOv5s is similar, with only a slight difference in mAP@0.5:0.95. Eff_YOLOv5s is slightly behind the other two models. Mob_YOLOv5s has the fewest parameters, reducing them by approximately 49.8%. The FPS increased from the original 12.4 to 17.5, showing an improvement of approximately 41.1%. The parameters of EFF_YOLOv5s are reduced by approximately 23.08% compared to YOLOv5s, and the FPS is increased by 10.48%. When all factors are considered, Mob_YOLOv5s appears to be the most suitable model for player detection because of its higher precision and recall rates, consistent prediction rates at mAP@0.5, reduced model parameters, and greater computational efficiency. Therefore, the current study uses Mob_YOLOv5s for subsequent player detection tasks. However, the final selection of a model should also be based on a comprehensive evaluation of application scenarios and hardware resources.

This study uses a test video segment of approximately 15 seconds for player detection using Mob_YOLOv5s, as illustrated in Fig. 20, with player positions subsequently transformed to a top view through perspective transformation. Perspective transformation can be used to project the original view of the court onto the top view image, which can facilitate tracking of players. Subsequently, the study uses perspective transformation to obtain player tracking information on the top view image, with players from different teams marked with green and red dots, as depicted in Fig. 21.

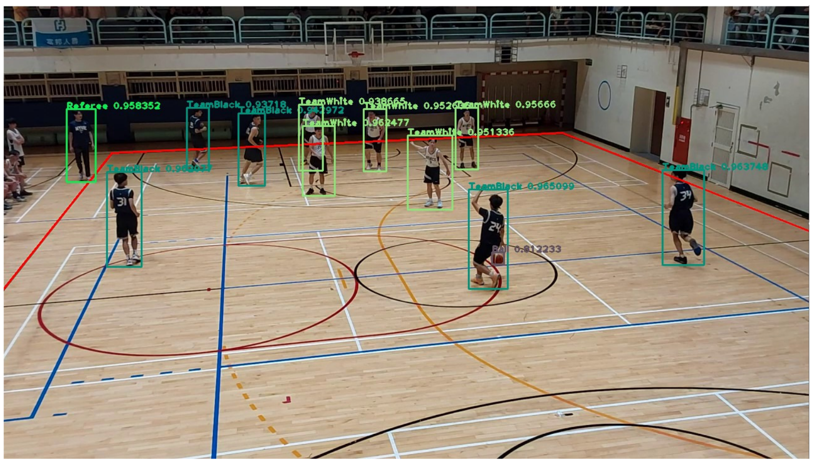

Fig. 20 Mob_YOLOv5s player detection scene

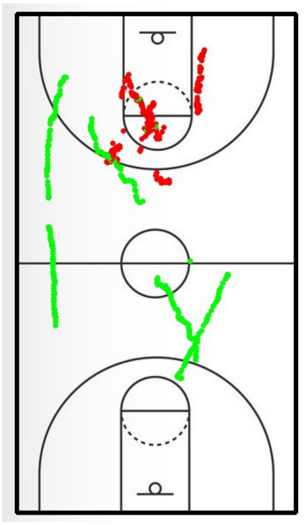
Fig. 21 Top view image of the player position tracking information

## 5. Conclusions

This study proposes a basketball player detection and analysis system based on YOLOv5 that enables the detection and analysis of player positions during basketball games. With the introduction of the improved YOLOv5 model, the system achieves more efficient detection of player positions than the original version does. The improved model achieves improved performance, uses fewer parameters, and achieves more FPS. The incorporation of MobileNetv3-small significantly improves the system's performance because it enables the number of parameters to be reduced by approximately 49.8% and increases the number of FPS by approximately 41.1%.

Further improvement is required to ensure the current system can be adapted to different venues and game conditions. Currently, the system can primarily be used for school venues. Given the diversity of basketball venues, modifications to the court detection module are necessary to adapt to different court environments. This ensures that the court drawing is specific to each court where it is used. In addition, in the future, the performance of the model can be further optimized to enable it to be applied to mobile platforms or embedded devices, which can increase the system's usability on the court.

## Conflicts of Interest

The authors declare no conflict of interest.

## References

[1] J. S. Sheu, C. K. Tsai, and P. T. Wang, "Driving Assistance System With Lane Change Detection," Advances in Technology Innovation, vol. 6, no. 3, pp. 137-145, July 2021.

[2] P. T. Wang, S. Y. Lin, and J. S. Sheu, "Vehicle Path Planning With Multicloud Computation Services," Advances in Technology Innovation, vol. 6, no. 4, pp. 213-221, October 2021.

[3] W. Song and S. A. Suandi, "Sign-YOLO: A Novel Lightweight Detection Model for Chinese Traffic Sign," IEEE Access, vol. 11, pp. 113941-113951, 2023.

[4] S. Zhang, Y. Chang, S. Wang, Y. Li, and T. Gu, "An Improved Lightweight YOLOv5 Algorithm for Detecting Railway Catenary Hanging String," IEEE Access, vol. 11, pp. 114061-114070, 2023.

[5] S. Han, X. Jiang, and Z. Wu, "An Improved YOLOv5 Algorithm for Wood Defect Detection Based on Attention," IEEE Access, vol. 11, pp. 71800-71810, 2023.

[6] Y. Guo and M. Zhang, "Blood Cell Detection Method Based on Improved YOLOv5," IEEE Access, vol. 11, pp. 67987-67995, 2023.

[7] Y. He, "Automatic Blood Cell Detection Based on Advanced YOLOv5s Network," IEEE Access, vol. 12, pp. 17639-17650, 2024.

[8] Y. Freund and R. E. Schapire, "A Decision-Theoretic Generalization of On-Line Learning and an Application to Boosting," Journal of Computer and System Sciences, vol. 55, no. 1, pp. 119-139, August 1997.

[9] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Y. Fu, et al., "SSD: Single Shot MultiBox Detector," Computer Vision – ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, pp. 21-37, October 2016.

[10] N. Dalal and B. Triggs, "Histograms of Oriented Gradients for Human Detection," IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pp. 886-893, June 2005.

[11] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," IEEE Conference on Computer Vision and Pattern Recognition, pp. 779-788, June 2016.

[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, et al., "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," https://doi.org/10.48550/arXiv.1704.04861, April 17, 2017.

[13] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 4510-4520, June 2018

[14] A. Howard, M. Sandler, B. Chen, W. Wang, L. C. Chen, M. Tan, et al., "Searching for MobileNetV3," IEEE/CVF International Conference on Computer Vision, pp. 1314-1324, October-November 2019.

[15] J. Hu, L. Shen, and G. Sun, "Squeeze-and-Excitation Networks," IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 7132-7141, June 2018.

[16] M. Tan and Q. Le, “EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks,” Proceedings of the 36th International Conference on Machine Learning, vol. 97, pp. 6105-6114, June 2019.

[17] M. Tan and Q. Le, “EfficientNetV2: Smaller Models and Faster Training,” Proceedings of the 38th International Conference on Machine Learning, vol. 139, pp. 10096-10106, July 2021.

[18] T. Guo, K. Tao, Q. Hu, and Y. Shen, “Detection of Ice Hockey Players and Teams via a Two-Phase Cascaded CNN Model,” IEEE Access, vol. 8, pp. 195062-195073, 2020.

[19] D. Acuna, “Towards Real-Time Detection and Tracking of Basketball Players Using Deep Neural Networks,” Proceedings of the 31st Conference on Neural Information Processing Systems, pp. 4-9, December 2017.

[20] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” Neural Computation, vol. 9, no. 8, pp. 1735-1780, November 1997.

[21] Z. Ivankovic, M. Rackovic, and M. Ivkovic, “Automatic Player Position Detection in Basketball Games,” Multimedia Tools and Applications, vol. 72, no. 3, pp. 2741-2767, October 2014.

[22] P. K. Santhosh and B. Kaarthick, “An Automated Player Detection and Tracking in Basketball Game,” Computers, Materials & Continua, vol. 58, no. 3, pp. 625-639, 2019.

[23] B. Markoski, Z. Ivanković, L. Ratgeber, P. Pecev, and D. Glušac, “Application of AdaBoost Algorithm in Basketball Player Detection,” Acta Polytechnica Hungarica, vol. 12, no. 1, pp. 189-207, 2015.