Journal of Applied Informatics and Computing (JAIC)
Vol.10, No.3, June 2026, pp. 2190~2197
e-ISSN: 2548-6861

Comparison of the Application of YOLOv5, YOLOv8, and YOLOv11 for Training Chinese Chess Objects

Ahmad Mufid Panisti [1]*, Ryan Satria Wijaya[2]*, Eko Rudiawan Jamzuri [3]*, Anugerah Wibisana[4]*
*Jurusan Teknik Elektro, Prodi Teknik Rekayasa Robotika, Politeknik Negeri Batam
ryan@polibatam.ac.id [2]

Article Info

Article history:

Received 2026-01-13
Revised 2026-04-29
Accepted 2026-04-30

Keyword:

Yolo,
Xiangqi,
Detection Objects,
Vision,
Camera.

ABSTRACT

The use of YOLO (You Only Look Once)-based object detection algorithms has become one of the main approaches in visual object recognition and training. This study aims to compare the performance of three versions of YOLO, namely YOLOv5, YOLOv8, and YOLOv11, in training models to detect objects in Chinese chess images. The dataset used consists of images of Chinese chess boards and pieces in various positions and lighting variations. The training process was carried out using uniform parameters to ensure fair evaluation, including batch size, number of epochs, and image resolution. The performance of each model was evaluated based on detection accuracy, inference speed, and computational efficiency metrics. The results of the study show that each version of YOLO has specific advantages in certain aspects, such as training speed or detection precision. From the 7224 images used as the dataset, several results were obtained that were necessary in helping to compile this journal. These included Precision (YOLOv5: 0.94, YOLOv8: 0.96, YOLOv11: 0.98), Recall (YOLOv5: 0.93, YOLOv8: 0.98, YOLOv11: 0.96), and mAP (YOLOv5: 0.96, YOLOv8: 0.98, YOLOv11: 0.99). This study provides important insights into the advantages and disadvantages of each version of YOLO in the specific application of Chinese chess object recognition, as well as providing guidance for developers in choosing the model that suits their project needs. This study provides insights into the strengths and limitations of each YOLO version, offering guidance for selecting appropriate models in real-time Chinese chess object detection applications.

I. INTRODUCTION

Xiangqi is a traditional brain-teasing game played by two people using specific strategies, and has developed into a sport originating in China[1]. The board of Xiangqi is a gridding with 8 horizontal lines and 9 vertical lines crossing together and forming 72 crossing points. Each of the 32 pieces is put on the crossing points. Each side (red and black) has 16 pieces with 2 rooks (車), two knights (馬), two bishops (象), two cannons (炮/砲), five pawns (兵), two safeguards (仕) and one commander (將 or 帥). The above figure shows the positions of the 32 chessmen[2]. Unlike the Latin alphabet, Chinese chess or xiangqi uses Chinese characters on its pieces, which are more complex[3].

In order for players to know the names of each Chinese chess piece or Xiangqi, AI technology is needed to identify these pieces. Currently, AI has penetrated various fields, ranging from general areas such as learning and perception to specialized fields such as human object detection, disease diagnosis, and others[4]. Computer vision, with its ability to perform tasks such as object recognition, visual tracking, semantic segmentation, and image restoration, is the solution[5][6][7][8].

In order to detect objects, the camera must apply several image processing techniques. Some image processing algorithms for detecting objects are Deep Learning[9][10][11]. Deep learning is divided into two types: region-based methods and end-to-end methods. In the region-based method, the approach is to form a bounding box using

various methods such as sliding window[12], selective search, and the features of each area object will be extracted with a Convolutional Neural Network (CNN)[13][14][15]. The second approach to the end-to-end method in object detection is You Only Look Once (YOLO)[16][17]. The YOLO (You Only Look Once) algorithm is a very fast and accurate real-time object detection method that uses a single convolutional neural network (CNN) to predict bounding boxes and object class probabilities simultaneously, processing the entire image at once, making it efficient for applications such as surveillance and autonomous vehicles. by dividing the image into a grid and having each cell predict objects.

The use of YOLO in this era has developed significantly. In the context of heavy traffic, Rahman et al. used YOLOv5 to detect vehicles in urban areas and achieved high accuracy even under complex visual conditions[18][19]. In the industrial sector, Simeth et al. applied YOLOv5 in manufacturing production lines and obtained consistent and reliable detection results[20]. In addition, Muhammad Harun Ashar and Dedi Suarna have proven the accuracy of YOLOv5 in implementing object detection cameras to distinguish between people wearing masks and those not wearing masks at the West Sulawesi Governor's General Affairs Office[21]. This is also proven by Ryan Satria Wijaya and colleagues in comparing Deep Learning Algorithms Between YOLOv5 and Mobilenet-SSDv2 as Fast and Robust Outdoor Object Detection Solutions[22].

Initially, the application of YOLO in detecting objects was limited to learning purposes, but as time progressed, its application evolved to facilitate human work. Such as using the You Only Look Once version 8 (YOLOv8) algorithm for detecting congregation members' faces and Convolutional Neural Network (CNN) in the process of recognizing congregation members' faces from images or videos[23], applying the YOLOv8 method for human object detection using datasets from Google Image/OpenImage[24].

The objective of this study is to identify the differences in object detection performance among three YOLO versions: YOLOv5, YOLOv8, and YOLOv11. And unlike in previous cases, which used chess as the subject in Hu Moments-Based Position Determination or the Implementation of Magnetic Sensors for Chess Piece Position[25][26], I have replaced it with Chinese chess, or Xiangqi, as the main subject. This comparative case study aims to provide a clear understanding of the evolution of these models, including differences in detection accuracy, performance metrics, and other relevant aspects. Its main contribution is a systematic benchmark of YOLOv5, YOLOv8, and YOLOv11 on a challenging, domain-specific dataset of Chinese chess (Xiangqi) pieces, offering practical guidance for real-time object detection applications in educational robotics and cultural heritage preservation.

Based on the previous test, which used the Jatson Nano as the primary device,[27] In conducting this research, a device in the form of a laptop with an NVIDIA GeForce GTX1650 GPU (CUDA enabled, 4 GB VRAM GDDR5) is required. This device was chosen because the training process was carried out on a laptop with an NVIDIA GeForce GTX 1650 GPU (CUDA enabled, 4 GB VRAM GDDR5) to accelerate training and benchmarking [28]. Such capability is essential for real-time applications, including the detection of Xiangqi pieces, where each frame captured by a camera must be processed quickly and accurately to provide immediate feedback to users.

II. METHOD

In this study, we adopted two object detection methods that have been proven effective, namely YOLO5, YOLOv8, and YOLOv11. These methods are commonly used in various fields and have their own advantages in object detection. Before discussing which method to use, let's talk about progress from the beginning.

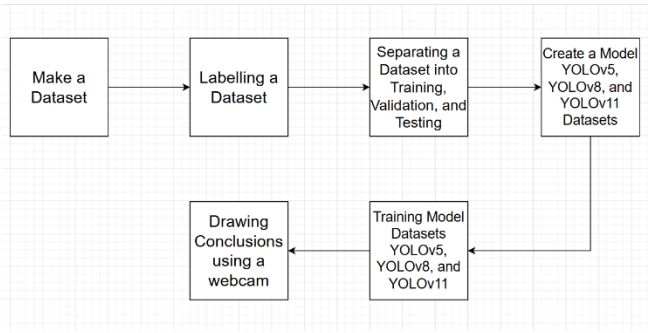

Fig. 1 Diagram Progressing

A. Dataset

The dataset used in this study was designed to evaluate and compare the performance of YOLOv5, YOLOv8, and YOLOv11 for Chinese chess object detection. The dataset consists of 7,224 images of Chinese chess pieces, each accompanied by corresponding object labels for training and evaluation purposes. This dataset plays a crucial role in ensuring the reliability and validity of the performance comparison among the three YOLO models.

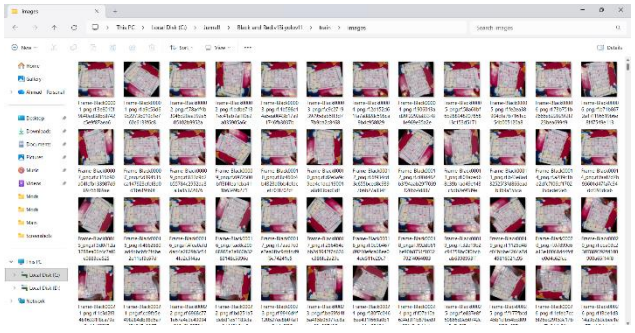

Fig. 2. Collecting Data

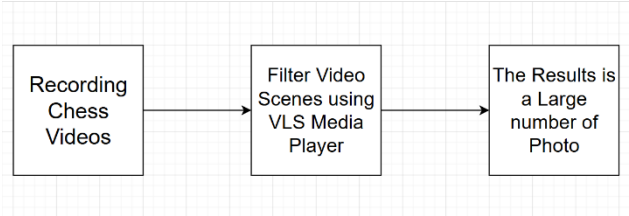

Fig. 3. Diagram make a Dataset

Data was collected independently by recording videos of Chinese chess pieces using a webcam mounted on a laptop. Video recordings were made from a top-down view, with the camera positioned approximately 30–50 cm above the chessboard. The main recordings were conducted under bright lighting to ensure clear object visibility. Additionally, variations in lighting, camera distance, and viewing angles were intentionally introduced to simulate real-world conditions. No additional data augmentation was applied beyond YOLO's default transformations (such as mosaic, flip, and color jitter).

TABLE 1.
ASPECT DATASET

No.	Aspect Dataset	Description
1	Total Images	7,224 images
2	Original Image Resolution	1280 x 720 pixels
3	Number of Classes	14 classes
4	Class Composition	7 types of Xiangqi pieces x 2 colors (Red and Black)
5	Piece Types	Advisor, Canon, Chariot, Elephant, General, Horse, Soldier
6	Lighting Variations (Illuminance)	Bright (300 - 500 lx), Dim (50 - 150 lx), Dark (< 30 lx)
7	Camera Distance	30 - 50 cm (top-down view)
8	Other Variations	Different viewing angles and partial occlusion simulation
9	Annotation Format	YOLO format (class_id, x_center, y_center, width, height - normalized)
10	Data Collection Method	Self-collected using webcam and VLC scene video filter

The recorded video was then converted into individual images using the video scene filter feature in VLC Media Player, allowing for the automatic extraction of multiple images from each video sequence. A custom dataset consisting of 7,224 images was specifically collected for this study. All images were captured at an original resolution of 1280 × 720 pixels using a webcam in a top-down configuration. The dataset covers 14 classes, representing 7 types of Chinese chess (Xiangqi) pieces in both red and black colors, namely: Advisor, Canon, Chariot, Elephant, General, Horse, and Soldier.

The dataset incorporates various real-world conditions, including different lighting intensities measured in Lux (bright: 300–500 lx, dim: 50–150 lx, dark: <30 lx), camera distances ranging from 30 to 50 cm, varying viewing angles, and partial occlusion scenarios to enhance model robustness

B. Labelling or Anotation Data

After collecting the dataset, the next step is to prepare the data annotation. Annotation refers to the process of labeling or categorizing data in a dataset, which aims to facilitate the training and testing of deep learning models. Based on images 4 and 5, All images in the dataset were manually annotated using the Roboflow annotation platform. The annotation process followed the YOLO bounding box format, where each object is represented by a text file containing the class index and normalized bounding box coordinates in the format:

$$(class\_id, x\_center, y\_center, width, height).$$

The coordinate values were normalized relative to the image width and height to ensure compatibility with YOLO-based detection models.

Fig. 4. Labelling Data

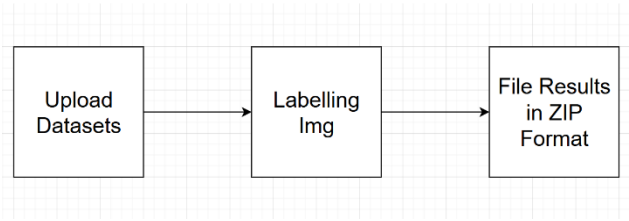

Fig. 5. Diagram Progres Labelling Img

Each chess piece is annotated using a rectangular bounding box that tightly encloses the visible area of the object. The bounding box is positioned to encompass the entire piece while minimizing unnecessary background area. For partially occluded pieces, the bounding box is adjusted to cover only the visible portion. Annotation quality was ensured through manual review by multiple annotators and random sampling verification to minimize labeling errors.

The dataset, consisting of 7,224 images, is divided into 14 classes corresponding to the names of the chess pieces (Advisor, Canon, Chariot, Elephant, General, Horse, and Soldier, each for black and red). Since each image typically contains two pieces, each class has approximately 1,032

annotations. The dataset was then split into three parts: 10% for the test set (704 images), 10% for the validation set (702 images), and the remaining 80% for the training set (5,626 images). The annotated dataset was exported in YOLO format and used directly to train and validate the YOLOv5, YOLOv8, and YOLOv11 models.

C. Model Training

The YOLOv5n, YOLOv8n, and YOLOv11n models were trained using identical hyperparameters to ensure a fair comparison. All models were initialized with pretrained weights from the COCO dataset, specifically YOLOv5n.pt, YOLOv8n.pt, and YOLOv11n.pt.

TABLE 2.
TRAINING CONFIGURATIONS

No.	Hyperparameter	Value / Description
1	Pretrained Weights	YOLOv5n.pt, YOLOv8n.pt, YOLOv11n.pt (COCO dataset)
2	Batch Size	32
3	Maximum Epochs	50
4	Early Stopping Patience	10 epochs
5	Initial Learning Rate	0.001
6	Optimizer	Adam
7	Input Image Size	640 Ã— 640 pixels
8	Data Augmentation	Default Ultralytics augmentations (mosaic, horizontal flip, color jitter)
9	Loss Functions	CIoU and Distribution Focal Loss (DFL) for bounding box regression, Classification loss for class prediction

Due to hardware limitations, training was performed on a laptop equipped with an NVIDIA GeForce GTX 1650 GPU (4 GB VRAM), Intel Core i7 processor, 16 GB RAM, and SSD storage. The training time per model ranged from 4 to 7 hours, with YOLOv11 requiring the longest duration due to its higher architectural complexity.

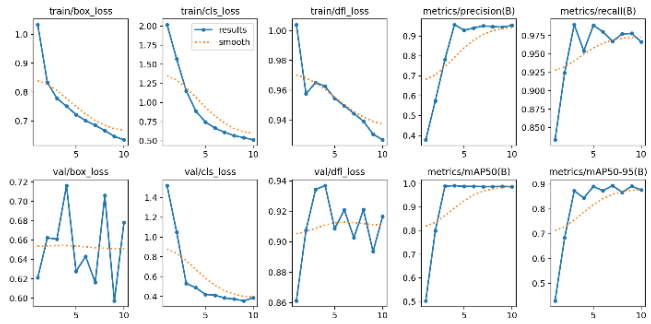

Fig. 6. Example of result train from Yolov5

Based on the image above, there is a decrease in the box and class when training reaches 32 epochs, while precision and recall increase to almost reach 1. This shows that the model can accurately identify target images.

D. Evaluation Metrics and FPS

To determine the differences between Yolov5, Yolov8, and Yolov11 in training datasets, evaluation data is required in the form of model performance results in training data. Model performance was evaluated using several standard metrics, including the Precision–Confidence Curve, Recall–Confidence Curve, Precision–Recall Curve, Confusion Matrix, and Frames Per Second (FPS).

Evaluation was conducted at an IoU threshold of 0.5 for calculating precision, recall, and mAP@0.5, with additional reporting of mAP@0.5:0.95. A confidence threshold of 0.25 was applied during inference. Precision and recall were calculated using standard definitions based on true positives (TP), false positives (FP), and false negatives (FN). mAP@0.5 represents the mean Average Precision at a fixed IoU threshold of 0.5, while mAP@0.5:0.95 is the average of mAP values computed across IoU thresholds ranging from 0.5 to 0.95 in steps of 0.05.

The Precision-Confidence Curve seeks to determine how accurate the predictions generated by the model are. To find this, the following formula 1 is used.

$$Precision(t) = \frac{TP(t)}{TP(t) + FP(t)} \quad (1)$$

For the Recall-Confidence Curve, the goal is to determine how completely the model finds objects. To find this, the following formula 2 is used:

$$Recall(t) = \frac{TP(t)}{TP(t) + FN(t)} \quad (2)$$

Not only that, this data collection also includes FPS. Frames Per Second (FPS) represents the processing speed of the object detection model. A higher FPS indicates faster inference time, which improves real-time performance but does not directly affect detection accuracy. In this study, the variation in FPS among YOLOv5, YOLOv8, and YOLOv11 shows a trade-off between detection speed and model complexity.

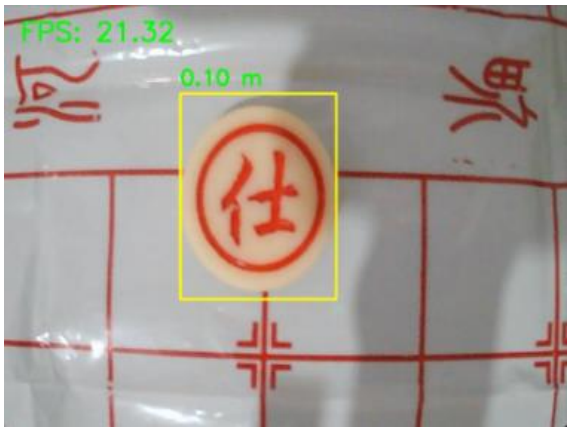

Fig. 7. FPS Object detection model

III. Results And Discussion

In this chapter, we present the results of our analysis. These results can be presented in the form of images and tables that are easy for readers to understand. In the discussion section, there are several important points that need to be identified for more accurate object detection. These points may include important variables such as object parameters, brightness levels, and observation distance. This allows readers to better understand the analysis results and the influence of important factors in object detection.

A. Detection Result

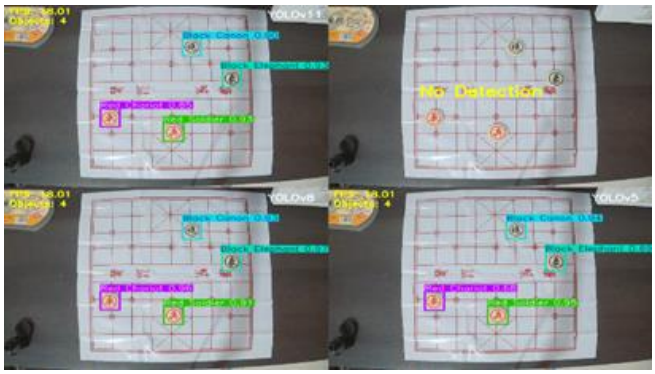

Fig. 8. Dataset YoloV5, YoloV8, and YoloV11

We can see the object detection results from the YOLOv5, YOLOv8, and YOLOv11 algorithms, as shown. The image can be detected thanks to the previous steps, namely preparing the dataset, labeling the dataset, and then training the model.

B. Confusion Matrix.

The Confusion Matrix is a method used to evaluate the performance of algorithms or classifications, including deep learning algorithms such as YOLOv5, YOLOv8, and YOLOv11, which use matrices to assess model performance in object detection tasks

To measure object detection performance, there are three results, namely.

1.) Precision - Confidence Curve.

Table 1.
Precision Confidence Curve

Precision - Confidence Curve	
Version	Precision
YoloV5	0.94608
YoloV8	0.91852
YoloV11	0.96017

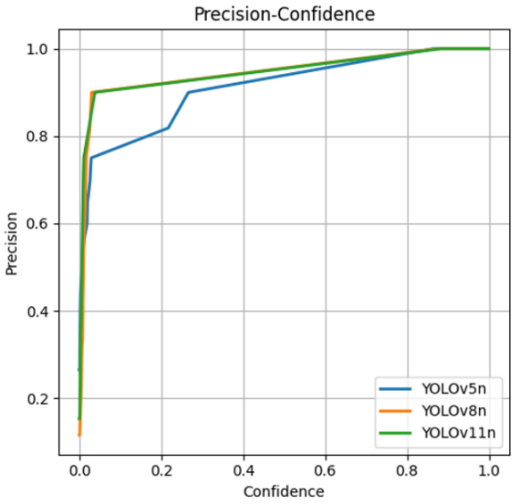

Fig. 9. Precision Curve

The following table shows the Precision-Confidence Curve for YOLOv5, YoloV8, and YoloV11. As can be seen in the table, YOLOv5 has a Precision rate of around 0.94, YOLOv8 has a Precision rate of around 0.91, and YOLOv11 has a Precision rate of around 0.96.

The Precision-Confidence curve shows that YOLOv11n achieved the highest precision across most confidence thresholds, rising rapidly to above 0.9 at a very low confidence level and maintaining the leading position. YOLOv5n exhibited a more gradual increase, while YOLOv8n showed an initial sharp rise followed by steady improvement. Overall, YOLOv11n displayed better confidence calibration, reaching near-perfect precision at lower confidence thresholds than the other two models.

2.) Precision-Recall Curve.

Tabel 2.
PR Curve Table

Precision-Recall Curve	
Version	PR
YoloV5	mAP@0.5 = 0.981
YoloV8	mAP@0.5 = 0.940
YoloV11	mAP@0.5 = 0.983

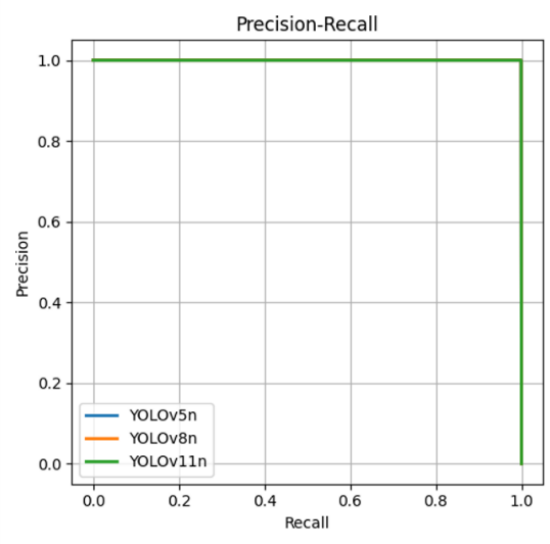

Fig. 10. Precision-Recall Curve

The Precision–Recall curve provides a comprehensive evaluation of detection performance by illustrating the balance between precision and recall across all thresholds. Detection performance was primarily evaluated using mean Average Precision at IoU threshold 0.5 (mAP@0.5), which is the standard metric in YOLO benchmarks. Additional metrics include mAP@0.5:0.95, Precision, Recall, and F1-score. IoU threshold was set to 0.5 and confidence threshold to 0.25 during evaluation

As illustrated in the Precision-Recall curve, all three models exhibited outstanding performance, maintaining a precision value of 1.0 across nearly the entire range of recall values (from 0.0 to 1.0). The curves remain flat at the maximum precision level before dropping sharply at recall = 1.0. This indicates excellent discriminative capability and a minimal trade-off between precision and recall for all models. YOLOv11n demonstrated marginally superior stability at high recall values compared to the earlier versions.

3.) Recall-Confidence Curve.

TABEL 3.
RECALL CONFIDENCE CURVE

Recall - Confidence Curve	
Version	Recall
YoloV5	0.96489
YoloV8	0.68207
YoloV11	0.94924

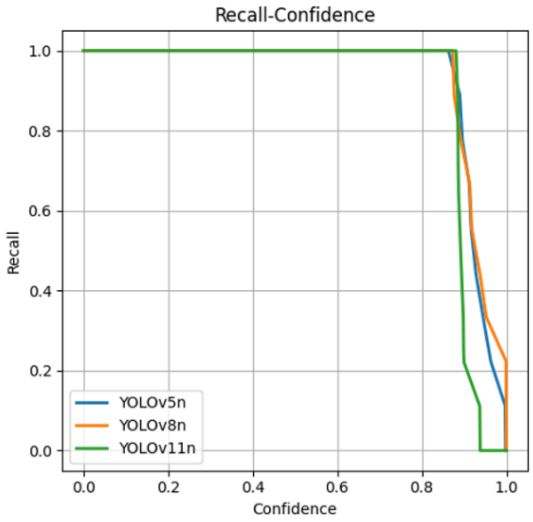

Fig. 11. Recall Curve

The following table shows the Recall-Confidence Curve for YOLOv5, YoloV8, and YoloV11. As can be seen in the table, YOLOv5 has a Precision rate of around 0.91, YOLOv8 has a Precision rate of around 0.68, and YOLOv11 has a Precision rate of around 0.94.

In the Recall-Confidence curve (right), all models maintained near-perfect recall (approximately 1.0) up to a confidence threshold of around 0.8. Beyond this point, recall declined sharply. YOLOv5n was able to sustain high recall until a slightly higher confidence threshold compared to YOLOv8n and YOLOv11n, suggesting that the older architecture is more lenient in retaining detections at higher confidence levels. YOLOv11n experienced the earliest drop in recall, indicating a more conservative detection behavior that prioritizes precision.

C. Object Detection Results in various conditions.

TABEL 4.
DETECTION CONDITIONS YOLOV5

Comparison Of Yolo Version Condition			
Condition	Bright	Dim	Dark
YoloV5	150 - 100 lx	100 - 50 lx	50 - 30 lx
YoloV8	150 - 100 lx	100 - 50 lx	50 - 45 lx
YoloV11	150 - 100 lx	100 - 50 lx	50 - 30 lx

Comparison of bright, dim, and dark conditions using the YOLOv5, YOLOv8, and YOLOv11 algorithm. The comparison above shows that in bright and dim conditions, the YOLOv5 algorithm successfully detects objects perfectly. Meanwhile, in dark conditions, the YOLOv5 algorithm is less able to detect objects, even though there are signs that objects are detected for a few second. The comparison above shows that in bright and dim conditions, the YOLOv8 algorithm successfully detects objects perfectly. Meanwhile, in dark conditions, the YOLOv8 algorithm can still detect objects, although it is not accurate and sometimes misses detections. The comparison above shows that in bright, theIn dim

conditions, the object can still be detected, but sometimes there is a blink in detecting objects and errors occur. YOLOv11 algorithm successfully detects objects perfectly. Meanwhile, in dark conditions, the YOLOv11 algorithm fails to detect objects.

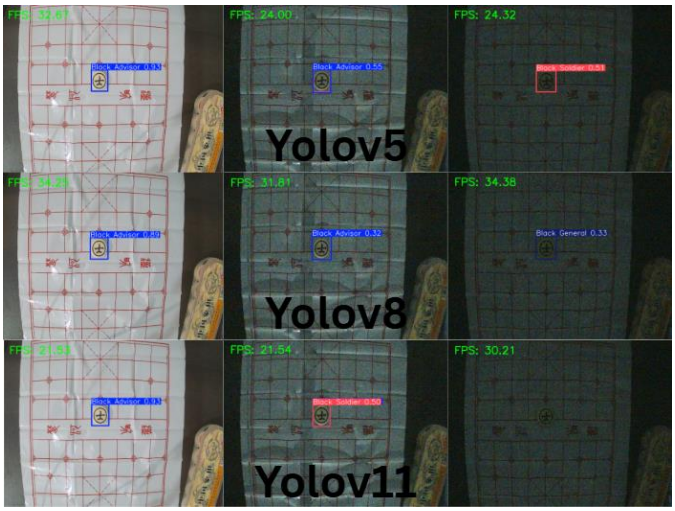

Fig. 12. Condition Detection Comparision result YoloV5, YoloV8, and YoloV11

D. FPS (Frames Per Second)

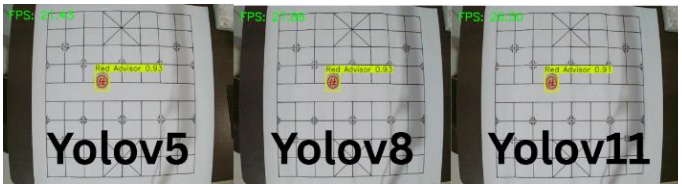

Fig. 13. FPS (Frames Per Second) Yolov5, Yolov8, and Yolov11

Frames Per Second (FPS) represents the number of video frames processed per second during object detection. Each YOLO version processes the same input frames through identical preprocessing, inference, and post-processing stages. Differences in FPS among YOLOv5, YOLOv8, and YOLOv11 are primarily influenced by architectural optimizations and model complexity. FPS was measured at input resolution 640×640 on GTX 1650 GPU. FPS includes full pipeline (preprocessing, inference, and NMS post-processing).

TABLE 5.
COMPARISON OF YOLO VERSION FPS

Model	FPS	Inference Time (ms)	Persentase FPS (%)
YOLOv8n	10.25	97.57	100
YOLOv11n	9.26	108.02	90.34
YOLOv5n	9.13	109.54	89.07

E. Final Results Of The Experiment

TABLE 6.
FINAL RESULT

Final Result on Validation				
Model	Precision	Recall	PR Curve	FPS
YoloV5	0.94608	0.96489	mAP@0.5 = 0.981	10.25
YoloV8	0.91852	0.68207	mAP@0.5 = 0.940	9.26
YoloV11	0.96017	0.94924	mAP@0.5 = 0.983	9.13

Table 4 presents the final experimental results of the three YOLO models. YOLOv11 achieved the highest precision and mAP values, indicating superior detection accuracy. YOLOv8 outperformed other models in recall and inference speed (FPS), making it suitable for real-time object detection. YOLOv5 demonstrated stable performance with relatively lower computational complexity.

IV. CONCLUSION.

From the results of this study, it can be concluded that YOLOv5, YOLOv8, and YOLOv11 are deep learning algorithms that are very useful for implementing real-time computer vision applications. This study demonstrates that YOLOv5, YOLOv8, and YOLOv11 are effective for real-time Chinese chess object detection. YOLOv11 achieves the highest precision and mAP values, making it suitable for applications requiring high detection accuracy. YOLOv8 excels in recall, offering better object coverage, especially under challenging conditions. YOLOv5, while less accurate, provides faster inference and lower computational cost. The limitations of this study include a limited dataset and test conditions that may not cover all possible scenarios in the real world. In addition, external factors such as lighting and weather conditions can affect the performance of object detection algorithms. The results of this study provide valuable insights into the performance comparison between YOLOv5, YOLOv8, and YOLOv11 in object detection. The main contribution of this study is to provide a better understanding of the advantages and disadvantages of each algorithm, so that researchers and practitioners can choose the most suitable algorithm for their applications. For future research, it is recommended to expand the test dataset to include a wider range of conditions and objects. In addition, further research can be conducted to improve the accuracy and speed of object detection by integrating new technologies or adjusting existing algorithm parameters.

REFERENCES

[1] T. Kasa, R. Adha, and S. Haraha, "Fungsi Xiangqi Bagi Masyarakat Tionghoa Di Kota Medan," *Jurnal Cakrawala Mandarin Asosiasi Program Studi Mandairn Indonesia*, vol. 1, no. 2, pp. 49–56, 2017.

[2] L. Ma, "Xiangqi vs Chess—The Cultural Differences Reflected in Chinese and Western Games," *Open J. Soc. Sci.*, vol. 08, no. 03, pp. 52–61, 2020, doi: 10.4236/jss.2020.83006.

[3] C. N. Sari, M. Istoningtyas, and M. Rosario, "Jurnal Politeknik Caltex Riau," 2019. [Online]. Available: https://jurnal.pcr.ac.id/index.php/jkt/

[4] H. Herdianto, D. Nasution, N. S. Atmaja, and S. Ramadhan, "Penerapan Deep Learning Yolo Untuk Pengukuran Jarak Objek Menggunakan Mono Kamera," *METHOMIKA Jurnal Manajemen Informatika dan Komputerisasi Akuntansi*, vol. 8, no. 1, pp. 51–56, Apr. 2024, doi: 10.46880/jmika.Vol8No1.pp51-56.

[5] A. Purno and W. Wibowo, "Implementasi Teknik Computer Vision Dengan Metode Colored Markers Trajectory Secara Real Time," 2016.

[6] J. Subur *et al.*, "Cyclotron : Jurnal Teknik Elektro Pemanfaatan Teknologi Computer Vision untuk Deteksi Ukuran Ikan Bandeng dalam Membantu Proses Sortir Ikan".

[7] R. S. Wijaya, W. Saputra, S. Prayoga, and E. R. Jamzuri, "Application of Object Detection and Face Recognition with Customize Dataset on Service Robot," 2024, pp. 51–65. doi: 10.2991/978-94-6463-620-8_5.

[8] Ryan Satria Wijaya, Rifqi Amalya Fatekha, Senanjung Prayoga, Dzaky Andrawan, Naurah Nazhifah, Mochamad Ari Bagus Nugroho, "Penerapan Visual servoing Robot Lengan dengan Metode Color Recognition sebagai Pemindah Objek Dua Warna Berbeda" Journal Of Applied Electrical Engineering (E-ISSN: 2548-9682), VOL. 9, NO. 1, JUNE 2025

[9] J. Zophie, H. Himawan Triharminto, D. Elekronika, and A. Angkatan Udara, "Implemetasi Algoritma You Only Look Once (YOLO) menggunakan Web Camera untuk Mendeteksi Objek Statis dan Dinamis Implementation of You Only Look Once (YOLO) Algorithm using Web Camera for Static dan Dinamic Object Detection," vol. 1, no. 1, 2022.

[10] F. Nobis, M. Geisslinger, M. Weber, J. Betz, and M. Lienkamp, "A Deep Learning-based Radar and Camera Sensor Fusion Architecture for Object Detection," May 2020, [Online]. Available: http://arxiv.org/abs/2005.07431

[11] S. Schneider, G. W. Taylor, and S. C. Kremer, "Deep Learning Object Detection Methods for Ecological Camera Trap Data," Mar. 2018, [Online]. Available: http://arxiv.org/abs/1803.10842

[12] Z. Tian, C. Shen, H. Chen, and T. He, "FCOS: Fully Convolutional One-Stage Object Detection," Aug. 2019, [Online]. Available: http://arxiv.org/abs/1904.01355

[13] J. Pang, K. Chen, J. Shi, H. Feng, W. Ouyang, and D. Lin, "Libra R-CNN: Towards Balanced Learning for Object Detection," Apr. 2019, [Online]. Available: http://arxiv.org/abs/1904.02701

[14] X. Xie, G. Cheng, J. Wang, X. Yao, and J. Han, "Oriented R-CNN for Object Detection." [Online]. Available: https://github.com/jbwang1997/

[15] H. Jiang and E. Learned-Miller, "Face Detection with the Faster R-CNN," Jun. 2016, [Online]. Available: http://arxiv.org/abs/1606.03473

[16] J. Pedoeem and R. Huang, "Yolo-Lite: A Real-Time Object Detection Algorithm Optimized for Non-GPU Computers," Nov. 2018, [Online]. Available: http://arxiv.org/abs/1811.05588

[17] W. Fang, L. Wang, and P. Ren, "Tinier-YOLO: A Real-Time Object Detection Method for Constrained Environments," *IEEE Access*, vol. 8, pp. 1935–1944, 2020, doi: 10.1109/ACCESS.2019.2961959.

[18] Y. Apridiansyah, Z. Padli, Y. Reswan, and H. Witriyono, "Penerapan Metode YOLOv5 untuk Klasifikasi dan Deteksi Objek Menggunakan Video Non-Real-Time," *Jurnal PROCESSOR*, vol. 20, no. 2, Oct. 2025, doi: 10.33998/processor.2025.20.2.2508.

[19] R. Rahman, Z. Bin Azad, and Md. B. Hasan, "Densely-Populated Traffic Detection using YOLOv5 and Non-Maximum Suppression Ensembling," Aug. 2021, doi: 10.1007/978-981-16-6636-0_43.

[20] A. Simeth, A. A. Kumar, and P. Plapper, "Flexible and robust detection for assembly automation with YOLOv5: a case study on HMLV manufacturing line," *J. Intell. Manuf.*, vol. 36, no. 5, pp. 3447–3463, Jun. 2025, doi: 10.1007/s10845-024-02411-5.

[21] M. H. Ashar and D. Suarna, "KLIK: Kajian Ilmiah Informatika dan Komputer Implementasi Algoritma YOLOv5 dalam Mendeteksi Penggunaan Masker Pada Kantor Biro Umum Gubernur Sulawesi Barat," *Media Online*, vol. 3, no. 3, pp. 298–302, 2022, [Online]. Available: https://djournals.com/klik

[22] R. S. Wijaya, S. Hasibuan, A. Wibisana, E. R. Jamzuri, and M. A. B. Nugroho, "Comparative Study of Deep Learning Algorithms Between YOLOv5 and Mobilenet-SSDv2 As Fast and Robust Outdoor Object Detection Solutions," 2024, pp. 94–106. doi: 10.2991/978-94-6463-620-8_8.

[23] "9757-Article Text-29352-1-10-20250310".

[24] M. Yusup Efendi, R. Wulanningrum, A. Bagus Setiawan, and U. Nusantara PGRI Kediri, "Rancang Bangun Sistem Deteksi Manusia dengan YOLO pada video CCTV," Online, 2024.

[25] D. Pang and G. Mangindaan, "Techno Science Journal Penentuan Posisi Buah Catur Berbasis Hu Moments yang Dimodifikasi untuk Robot Pemain Catur dengan Sistem Tersemat Identifying the Chess Pieces Configuration Based on Modified Hu Moments for Chess Playing Robot with Embedded System".

[26] N. Khamdi, dan Riki Putra, and P. Caltex Riau, "Cyclotron : Jurnal Teknik Elektro Implementasi Sensor Magnet untuk Posisi Bidak Catur pada Robot Catur," vol. 6.

[27] R. Satria Wijaya, A. Yunisa Anadia, R. Amalya Fatekha, and S. Prayoga, "Real-Time Chinese Chess Piece Character Recognition using Edge AI," 2025. [Online]. Available: http://jurnal.polibatam.ac.id/index.php/JAIC

[28] D. Ramadhany and B. Henryranu Prasetio, "Sistem Deteksi Kebosanan dan Kantuk Mahasiswa Pada Proses Pembelajaran Berbasis YOLOv11 yang diimplementasikan dengan NCNN di Raspberry Pi," 2025. [Online]. Available: http://j-ptiik.ub.ac.id