Journal of Applied Informatics and Computing (JAIC)
Vol.9, No.4, August 2025, pp. 1984~1990
e-ISSN: 2548-6861

Real-Time Chinese Chess Piece Character Recognition using Edge AI

Ryan Satria Wijaya [1]*, Atika Yunisa Anadia [2]*, Rifqi Amalya Fatekha [3]*, Senanjung Prayoga [4]*
* Teknik Elektro, Politeknik Negeri Batam
ryan@polibatam.ac.id [1], atikayunisa@gmail.com [2], rifqi@polibatam.ac.id [3], senanjung@polibatam.ac.id [4]

Article Info

Article history:

Received 2025-07-03
Revised 2025-08-04
Accepted 2025-08-22

Keyword:

Chinese Chess,
Computer Vision,
Deep Learning,
Jetson Nano.

ABSTRACT

This research focuses on developing a character analysis system on Chinese chess pieces (xiangqi) using computer vision technology with the deep learning framework PyTorch. The system is designed to detect and interpret text written on chess pieces in real time, making it easier for players to identify the function of each piece. The implementation is done using a web camera and can be applied to embedded devices such as Jetson Nano. This research aims to develop an automatic recognition system that can help players better understand the game of xiangqi by identifying characters on pieces in real time. The test results show that the system successfully recognized 14 pieces correctly. The system developed using Jetson Nano can directly process image data with a processing time of 0.0222 seconds. This data is obtained from the average of each FPS image from the web camera.

I. INTRODUCTION

Xiangqi, a traditional Chinese game, is an interesting aspect of Chinese culture to study. It is a traditional brain teaser game played by two people who rely on certain strategies. It has developed into a sport originating from China[1]. For Chinese people, xiangqi serves five functions: as a means of communication, to improve concentration, to practice patience, and for entertainment[2].

Chinese chess, also known as xiangqi, is a variation of chess that originated in China thousands of years ago. It offers a unique strategic approach compared to Western chess. The first recorded instance of Chinese chess in Chinese history was around the third century[3]. The game involves 32 pieces that are divided into two groups by color: red and black. Each color group consists of 16 pieces, which are categorized into seven types. Unlike the Latin alphabet, Chinese chess or xiangqi uses Chinese characters on its pieces, which are more complex[4].

The Chinese chess, or xiangqi, character analysis system aims to interpret the written text so players can identify each piece. This requires a real-time artificial intelligence system. 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]. Recognition detection in Chinese chess, also known as xiangqi, is built using deep learning techniques. Deep learning is one branch of machine learning that focuses on developing algorithms to model complex patterns using artificial neural networks. These algorithms are inspired by the structure and function of human neural networks[9][10][11]. Deep learning technology is widely used in image processing, speech recognition, and text analysis to predeict, classify, and extract features from highly complex data[12][13][14]. This research involves using deep learning training techniques with the SSD-MobileNet architecture model to detect Chinese chess pieces and recognize the xiangqi characters on them.

Compared to methods such as You Only Look Once (YOLO) and Region-Based Convolutional Neural Networks (RCNN), the Single Shot Detector (SSD) method more accurately identifies objects in images or videos and processes images from cameras faster[15][16]. It can produce higher accuracy than YOLO and faster processing than RCNN. However, it is considered less efficient at detecting small objects[17].

Study [18] compared six object detection models, including YOLO, SSD, and Faster-RCNN. The results showed that YOLO had the fastest computation time, while SSD had the highest accuracy[19][20]. The SSD model used in this study has the advantage of an optimal balance between computational speed and accuracy. In contrast, the Faster-RCNN model has the highest accuracy but the slowest

computational speed[21]. Compared to YOLO and Faster RCNN, SSD can operate on computing systems with the lowest hardware requirements. Other types of deep neural network architectures include artificial neural networks (ANNs), recurrent neural network (RNNs), long short-term memory networks (LSTMs), and self-organizing maps (SOMs). These architectures present several challenges, including the imprecision of neural networks, ensuring data quality, ensuring data security, and producing artificial intelligence. Each architecture has its own advantages and disadvantages. However, convolutional neural networks (CNN) are suitable for image processing in computer vision because they can classify the smallest interconnected nodes [22][23].

The goal of this research is to design a China chess detection system using the Jetson Nano, an alternative mini PC based on an internal GPU[24][25]. This device was chosen for the study because it is an edge computing device that can process data locally, reducing dependency on internet connections and increasing system response speed. Unlike cloud computing, which requires sending data to an external server, edge computing provides much lower latency and more efficient power consumption, especially on embedded devices such as the Jetson Nano. This is important for real-time applications, such as character detection on xiangqi pieces, where every camera frame needs to be processed in real time to provide users with fast, accurate feedback. This compact device can record real-time video due to its small size, and it is equipped with an integrated GPU for processing digital image data[26].

Several libraries are needed to run SSD-MobileNet on Jetson Nano and maximize deep learning performance. These include frameworks developed by NVIDIA to run inference using various deep learning models, such as SSD-MobileNet. TensorRT then aims to accelerate these models on the GPU. TensorRT can optimize SSD-MobileNet models to run inference more efficiently on the Jetson Nano. SSD-MobileNet uses CUDA to speed up the inference process. cuDNN accelerates convolutions, pooling, and other important neural network layer operations for models like SSD-MobileNet. The NVIDIA Jetson Nano supports PyTorch. The Jetson Inference Framework allows the conversion of PyTorch-trained models to other formats, such as ONNX and TensorRT. OpenCV processes images and videos that will be provided as input to SSD-MobileNet. OpneCV enables real-time image processing. Matplotlib and Numpy are often used for visualizing object detection results and model evaluation during training and evaluation. This implementation was aided by a Taffware US829 camera as a source of high-quality image data.

II. METHOD

A Convolutional Neural Network (CNN) is a type of artificial neural network designed to process and recognize patterns in images. Similar to the visual cortex of the human brain, CNNs automatically extract important features, eliminating the need for manual extraction, as is done in traditional computer vision methods.

A. System Design

This research focuses on designing a Chinese chess or xiangqi detection system with deep learning for text interpretation, consisting of hardware and software designs. For the hardware, a webcam captures the object to be detected[27]. Figure 1 shows the hardware design. The object used is a Chinese chess pieces, or xiangqi piece. This study uses the SSD-MobileNet architecture model for deep learning training techniques. This model is applied to a Jetson Nano device via a webcam, enabling real-time detection of objects in images.

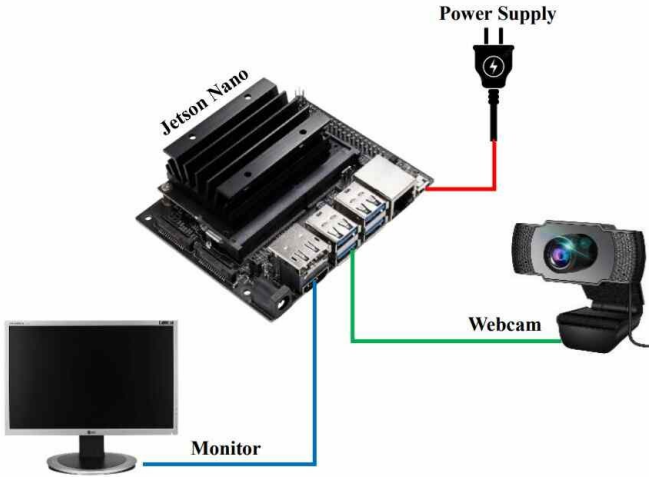


Figure 1. Hardware Design

The Jetson Nano webcam uses libraries such OpenCV to capture images[28], while the Jetson Inference engine performs inference using the SSD-MobileNet model. The developed model begins by loading the pawn detection dataset. The following Python deep learning libraries are used for data preparation: OpenCV, imutils, matplotlib, torchvision, torch, boto3, and urlib3.

This library is used to train SSD-MobileNet models for use on Jetson Nano device, as shown in Figure 2. To build and train these models, we recommend using the Ubuntu operating system, version 18.04 or later. This research uses Ubuntu 18.04 because it supports various frameworks, such as TensorFlow, PyTorch, and OpenCV. If a newer version of Ubuntu is used, its compatibility with the Jetpack used on the Jetson Nano must be checked. Model recognition in the software consists of two stages: training and testing. Training uses the PyTorch library, and testing uses the PyTorch library with support from Torchvision, OpenCV, and Matplotlib. The first step of the system is to acquire a dataset in image form. The first stage is to retrieve the dataset as an image. This system has 14 classes that need to be detected: Black General, Black Chariot, Black Advisor, Black Cannon, Black Elephant, Black Soldier, Black Horse, Red General, Red Chariot, Red Advisor, Red Cannon, Red Elephant, Red Soldier, and Red Horse. The training process aims to teach the

computer to process previously prepared images and object labels, forming patterns based on the characteristics of each class[29].

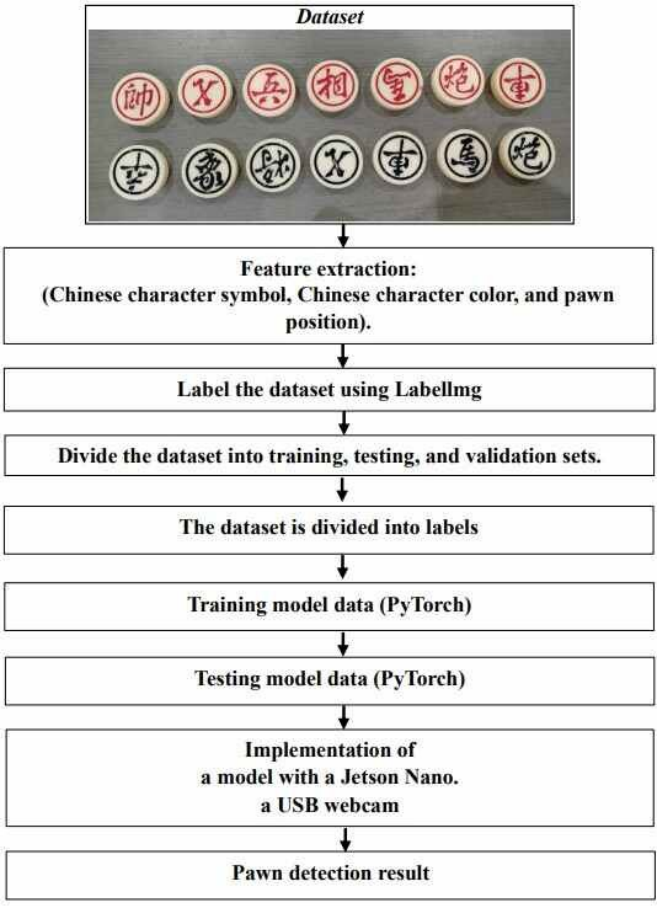


Figure 2. Flow Diagram

Before training begins, all datasets must be extracted based on complex symbols such as Chinese characters, visual variations such as color and lighting, and the relative position of pieces, as these factors can affect the appearance of dataset. Understanding and handling these variations is important so that the model can recognize objects consistenly under various conditions. After training, a training graph is typically generated to illustrate the model's loss function and accuracy during training. The best model is the one with the lowest loss value and the highest accuracy on the validation or test data. Then, the trained model is converted to ONNX (Open Neural Network Exchange) format, which is used to transfer the model from the training environment to the Jetson Nano device. There, it can be optimized using TensorRT for fast inference. Once the model is converted to ONNX, the ONNX file is converted into a TensorRT engine optimized for the Jetson Nano.

The model is implemented on the Jetson Nano device. A webcam detects objects in real-time images and converts them to ONNX (Open Neural Network Exchange) format. This format is used to transfer the model from the training environment to the Jetson Nano device. There, the model is optimized using TensorRT for fast inference. After converting the model to ONNX, the ONNX file is converted to a TensorRT engine optimized for the Jetson Nano. The model is then implemented on the Jetson Nano device, which uses a webcam to detect objects in real-time images.

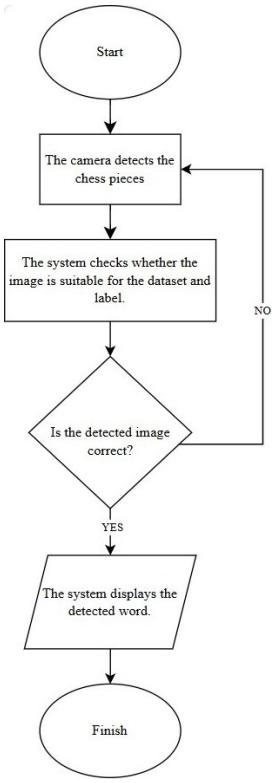


Figure 3. Flowchart

Figure 3. Shows a flowchart illustrating how the piece detection system works. A camera is placed in front of the piece to determine if it matches the pre-trained type in the data classification.

B. Image Dataset

The detection system has several stages. The first stage is collecting datasets in the form of pawn images. This research uses a dataset of 1.500 images divided into 15 classes, including a background class. Each class consists of 100 images. 20% of the total data is tested at this stage, while the remaining 80% is used for training. Each piece was photographed manually, and then each object in the image was annotated using LabelImg software in the form of a bounding box to mark its location and class. This stage is crucial to ensuring the quality of the annotation data needed to optimally train the object detection model.

C. Model Training

Model training ensures an optimal file. This file serves as the piece detection model, which identifies the piece image captured from the webcam. During the 150 epoch training process, the training loss value consistently decreases during

the initial epochs but then stabilizes near 1.0 at the end. This indicates that the model can accurately identify the target image.

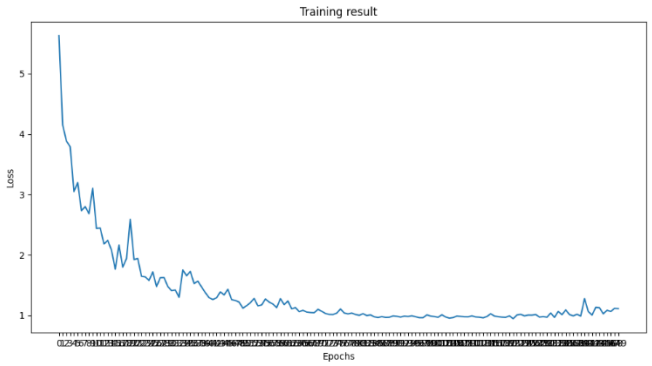


Figure 4. Appearance of the training loss graph

A confusion matrix is used to test the system and classify the model results into True Positive (TP), True Negative (TN), and False Positive (FP).

Precision describes the number of expected or factually correct positive values. For example, if TP equals 1 and FP equals 1, the actual value obtained is 0,5.

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

To make it easier to remember the correctly identified proportion, recall statistics are used a quantitative measure of the algorithm's ability to classify all positive cases. Recall is mathematically represented by the equation (2).

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

FN stands for false negative.

$$F1\ score = 2 \times \frac{[Precision \times Recall]}{[Precision+Recall]} \quad (3)$$

Then, the accuracy of the F1 score test is quantified. For a balance dataset, this evaluation step yields the most accurate results. True positive (TP) referes to the number of times the model correctly predicts a positive class. True negative (TN) refers to the number of times the model correctly predicts a negative class. A False positive (FP) occurs when the model incorrectly predicts a positive class when the true value is negative. A False negative (FN) occurs when the model incorrectly predicts a negative class when the true value is positive. This stage verifies accurate predictions.

D. Model Implementation

This model was implemented on a Jetson Nano device. This portable mini computer is fairly small. NVIDIA designed the Jetson Nano to develop compact, efficient embedded systems that can handle various neural network applications, including image classification, object detection, segmentation, and speech processing.

The Jetson Nano specification used is the Jetson Nano Developer Kit with 4GB of LPDDR4 memory. It has an NVIDIA Maxwell GPU core and can encode and decode up to 1080p video (F28 CUDA cores, CPU Quad-core ARM Cortex-A57, with HD).

Once the webcam detects the piece, the model is implemented on the piece detection system's device. When the object is detected, the image is reprocessed by resizing the pixels. Then, it is converted into an array matrix. Finally, SSD-MobileNet is used to predict the comparison data with the pre-trained model.

III. RESULTS AND DISCUSSION

A. Testing Pawn Detection Against the Model

Testing the model on the dataset is essential to validate the accuracy of the processed predictions. Figure 5 shows that the system identifies the characters on each piece as hanzi characters (Black General, Black Chariot, Black Advisor, Black Cannon, Black Elephant, Black Soldier, Black Horse, Red General, Red Chariot, Red Advisor, Red Cannon, Red Elephant, Red Soldier, and Red Horse) when compared to the previous model data. This is in accordance with the applied method.

Figure 5. The results of the chess piece detection test are as follows, (a) red and black advisor piece, (b) red and black cannon piece, (c) red and black chariot piece, (d) red and black elephant piece, (e) red and black general piece, (f) red and black horse piece, and (g) red and black soldier piece.

Figure 5(a) shows that the system successfully detects the pawn image, specifically the red and black advisor model, which is marked with green and yellow boxes labeled “Black Advisor” and “Red Advisor”, respectively. The test was conducted 150 times, resulting in an accuracy of 94.4% and 93.7% for the black and Red Advisor models, respectively.

Figure 5(b) shows that the system successfully detects the pawn image, specifically the red and black cannon model, which is marked with red and green boxes labeled “Black Cannon” and “Red Cannon”, respectively. The test was conducted 150 times, resulting in an accuracy of 95.8% and 79.2% for the black and Red Cannon models, respectively.

Figure 5(c) shows that the system successfully detects the pawn image, specifically the red and black chariot model, which is marked with red and green boxes labeled “Black Chariot” and “Red Chariot”, respectively. The test was conducted 150 times, resulting in an accuracy of 97.8% and 99.0% for the black and Red Chariot models, respectively.

Figure 5(d) shows that the system successfully detects the pawn image, specifically the red and black elephant model, which is marked with brown and blue boxes labeled “Black Elephant” and “Red Elephant”, respectively. The test was conducted 150 times, resulting in an accuracy of 98.8% and 98.0% for the black and Red Elephant models, respectively.

Figure 5(e) shows that the system successfully detects the pawn image, specifically the red and black general model, which is marked with green and white boxes labeled “Black General” and “Red General”, respectively. The test was conducted 150 times, resulting in an accuracy of 99.7% and 99.9% for the black and Red General models, respectively.

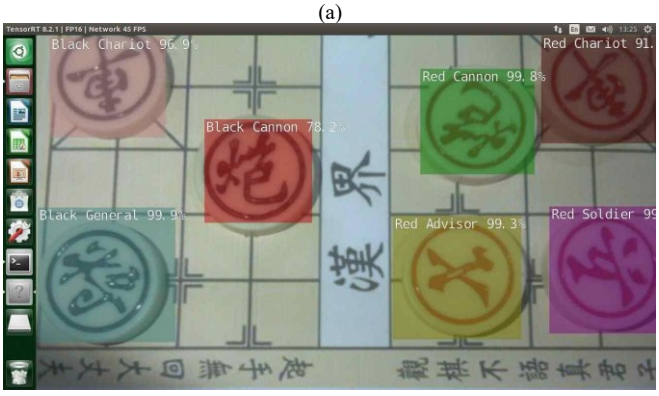
Figure 5(f) shows that the system successfully detects the pawn image, specifically the red and black horse model, which is marked with purple and green boxes labeled “Black Horse” and “Red Horse”, respectively. The test was conducted 150 times, resulting in an accuracy of 97.4% and 99.0% for the black and Red Horse models, respectively.

Figure 5(g) shows that the system successfully detects the pawn image, specifically the red and black soldier model, which is marked with blue and purple boxes labeled “Black Soldier” and “Red Soldier”, respectively. The test was conducted 150 times, resulting in an accuracy of 99.6% and 83.6% for the black and Red Soldier models, respectively.

B. Pawn Detection Testing using a Live Camera

This test used models that had not previously been used as training datasets. Tests that detect pawns throught live cameras are classified based on the following fourteen criteria: The first criterion is Black General; The second, Black Chariot; the third, Black Advisor; the fourth, Black Cannon; the fifth, Black Elephant; the sixth, Black Soldier; the seventh, Black Horse; the eighth, Red General; the ninth, Red Chariot; the tenth, Red Advisor, the eleventh, Red Cannon; the twelfth, Red Elephant; the thirteenth, Red Soldier; and the fourteenth, Red Horse.

(a)



(b)

Figure 6. Test results for live piece detection on the chessboard, (a) red piece and, (b) black piece

Figure 6 shows that classification of pieces placed on the chessboard and captured directly throught the camera is based on two main criteria.The first criterion includes seven types of black pieces, and the second includes seven types of red pieces. The camera can detect seven types of red pieces with an average accuracy of 99.2%, and seven types of black pieces with an average accuracy of 97.2%. under these conditions, all tested models were found to accurately reflect the actual situation. The test data are presented in TABLE I and TABLE II.

TABLE I
RED PAWN DETECTION PREDICTION TESTING ON THE CHESSBOARD

No.	Type Pawn	Camera Live Detection Testing	
		Prediction Detection	Accuracy
1	Red General	As per	99.9%
2	Red Chariot	As per	99.7%
3	Red Advisor	As per	97.1%
4	Red Cannon	As per	99.8%
5	Red Elephant	As per	99.2%
6	Red Soldier	As per	99.4%
7	Red Horse	As per	99.3%
Average accuracy			99.2%

TABLE II
BLACK PAWN DETECTION PREDICTION TESTING ON THE CHESSBOARD

No.	Type Pawn	Camera Live Detection Testing	
		Prediction Detection	Accuracy
1	Black Soldier	As per	98.7%
2	Black General	As per	99.9%
3	Black Chariot	As per	99.6%
4	Black Cannon	As per	99.2%
5	Black Elephant	As per	91.5%
6	Black Horse	As per	95.5%
7	Black Advisor	As per	96.2%
Average accuracy			97.2%

The system developed with Jetson Nano could process image data in just 0.0222 seconds. This data was obtained by averaging each frame per seconds (FPS) of the image from the webcam. This processing speed is much faster than that of other mini computer devices. For example, Muhammad Luqman Bukhori and Erwan Eko Prasetiyo conducted research using a Raspberry Pi 4 mini computer, which was published in the National Journal of Electrical and Information Engineering. Therefore, the Jetson Nano device outperforms others in processing and direct object identification.

C. Confusion Matrix

A confusion matrix is a table that evaluates the performance of a classification model by displaying the correct and incorrect predictions for each class. Figure 7 shows the classification for 15 classes, the background class and 14 chess piece classes.

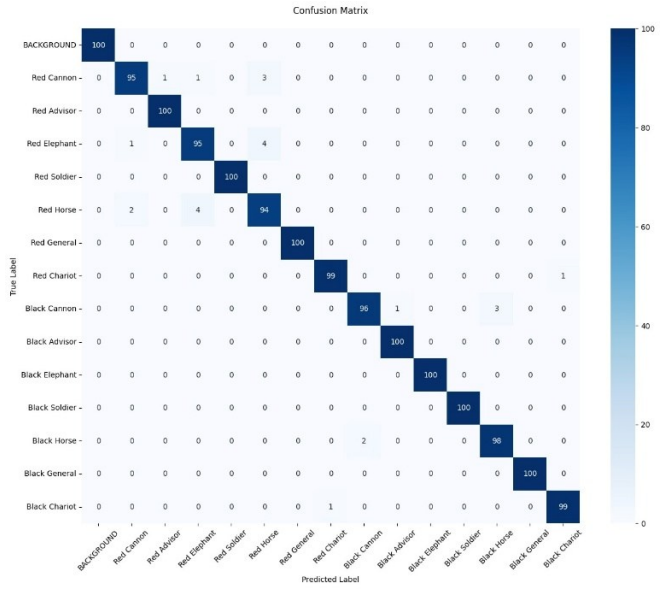


Figure 7. Confusion Matrix

Overall, the model performed very well, achieving an average accuracy of 97.47% (1.462 correct predictions out of 1.500 total samples). Eight of the 15 classes achieved perfect accuracy, while the remaining seven classes showed accuracies above 94%. There were only 38 misclassifications out of 1.500 predictions, showing that the model can recognize and classify xiangqi chess pieces well and distinguish between pieces and background perfectly.

TABLE III
MODEL NETWORK EVALUATION

Class	Precision (%)	Recall (%)	F1-score (%)
Background	100.00	100.00	100.00
Red Cannon	96.94	95.00	95.96
Red Advisor	99.01	100.00	99.50
Red Elephant	95.00	95.00	95.00
Red Soldier	100.00	100.00	100.00
Red Horse	93.07	94.00	93.53
Red General	100.00	100.00	100.00
Red Chariot	99.00	99.00	99.00
Black Cannon	97.96	96.00	96.97
Black Advisor	99.01	100.00	99.50
Black Elephant	100.00	100.00	100.00
Black Soldier	100.00	100.00	100.00
Black Horse	97.03	98.00	97.51
Black General	100.00	100.00	100.00
Black Chariot	99.00	99.00	99.00

Table III shows that the xiangqi chess piece classification model performs very satisfactorily, with an overall accuracy of 97.47%. The relatively even distribution of errors across classes suggests that the model is well-trained and ready for implementation in an automatic xiangqi chess piece recognition system.

IV. CONCLUSIONS

A device has been designed and built using Jetson Nano to detect pawn objects. Test results show that a model consisting of 14 pieces, Black General, Black Chariot, Black Advisor, Black Cannon, Black Elephant, Black Soldier, Black Horse, Red General, Red Chariot, Red Advisor, Red Cannon, Red Elephant, Red Soldier, and Red Horse, was successfully detected using the PyTorch deep learning framework. Using a PyTorch based model to detect chess pieces yields highly accurate results. Tests on pieces placed on a chessboard demonstrate that the camera can directly detect seven types of red pieces with an average accuracy of 99.2%. Under the same criteria, seven types of black pieces were successfully detected with an average accuracy of 97.2%. In summary, the PyTorch based detection system can accurately detect all types of pieces, both red and black.

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] A. lestari purba, "Designing a Chess Game Application Using the Android-Based Breadth First Search Method," Journal Of Computer Science And Informatics Engineering (CoSIE), vol. 03, no. 3, pp. 123–130, 2024, [Online]. Available: http://creativecommons.org/licenses/by-sa/4.0/

[3] M. H. Kurniawan and D. Udjulawa, "Perbandingan Performa Algoritma Minimax Dan Alpha-Beta Pruning Pada Game Catur Cina," 2020.

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

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

[6] J. Subur et al., "Pemanfaatan Teknologi Computer Vision untuk Deteksi Ukuran Ikan Bandeng dalam Membantu Proses Sortir Ikan," vol. 7, Jan. 2024.

[7] R. S. Wijaya, W. Saputra, S. Prayoga, and E. R. Jamzuri, "Application of Object Detection and Face Recognition with Customize Dataset on Service ROobot," in 7th International Conference on Applied Engineering (ICAE 2024), Dec. 2024, pp. 51-65.

[8] R. S. Wijaya, R. A. Fatekha, S. Prayoga, D. Andrawan, N. Nazhifah, and M. A. B. Nugroho, "Penerapan Visual Servoing Robot Lengan dengan Metode Color Recognition sebagai Pemindah Objek Dua Warna Berbeda," Journal of Applied Electrical Engineering, vol. 9, no. 1, pp. 51-57, 2025.

[9] S. Lu, "Deep learning for object detection in video," in Journal of Physics: Conference Series, Institute of Physics Publishing, Mar. 2019. doi: 10.1088/1742-6596/1176/4/042080.

[10] S. Ilahiyah and A. Nilogiri, "Implementasi Deep Learning Pada Identifikasi Jenis Tumbuhan Berdasarkan Citra Daun Menggunakan Convolutional Neural Network," JUSTINDO (Jurnal Sistem & Teknologi Informasi Indonesia), vol. 3, 2018.

[11] M. Haris, T. Pustaka, M. H. Diponegoro, S. Kusumawardani, and I. Hidayah, "Tinjauan Pustaka Sistematis: Implementasi Metode Deep Learning pada Prediksi Kinerja Murid (Implementation of Deep Learning Methods in Predicting Student Performance: A Systematic Literature Review)," 2021.

[12] A. Santoso and G. Ariyanto, "Implementasi Deep Learning Berbasis Keras Untuk Pengenalan Wajah," Jurnal Teknik Elektro, vol. 18, no. 01, 2018, [Online]. Available: https://www.mathworks.com/discovery/convol

[13] I. Virgiawan, F. Maulana, M. A. Putra, D. D. Kurnia, and E. Sinduningrum, "Deteksi dan tracking objek secara real-time berbasis computer vision menggunakan metode YOLO V3," Jurnal Ilmiah Multidisiplin Indonesia, vol. 3, no. 3, 2024, [Online]. Available: https://journal.ikopin.ac.id/index.php/humantech

[14] R. S. Wijaya, S. Hasibuan, A. Wibisana, E. Rudiawan, and M. A. B. N. Jamzuri, "Comparative Study of Deep Learning Algorithms Between YOLOv5 and Mobilenet-SSDv2 As Fast and Robust Outdoor Object Detection Solutions," in Proceedings of the 7th International Conference on Applied Engineering (ICAE 2024), Dec. 2024, p. 94.

[15] C.-Y. Fu, W. Liu, A. Ranga, A. Tyagi, and A. C. Berg, "DSSD : Deconvolutional Single Shot Detector," Jan. 2017,[Online].Available:http://arxiv.org/abs/1701.06659

[16] W. Liu et al., "SSD: Single Shot MultiBox Detector," Dec. 2015, doi: 10.1007/978-3-319-46448-0_2.

[17] K. R. D. B. A. A. E. J.-Y. Chen Zhihao, Real Time Object Detection, Tracking, and Distance and Motion Estimation based on Deep Learning: Application to Smart Mobility. IEEE, 2019.

[18] S. A. Sanchez, H. J. Romero, and A. D. Morales, "A review: Comparison of performance metrics of pretrained models for object detection using the TensorFlow framework," in IOP Conference Series: Materials Science and Engineering, Institute of Physics Publishing, Jun. 2020. doi: 10.1088/1757-899X/844/1/012024.

[19] Sukusvieri Andrianto, "Implementasi Metode Single Shot Detector (SSD)," 2020.

[20] R. S. Wijaya, S. Santonius, A. Wibisana, E. R. Jamzuri, and M. A. B. Nugroho, "Comparative Study of YOLOv5, YOLOv7, and YOLOv8 for Robust Outdoor Detection," Journal of Applied Electrical Engineering, vol. 8, no. 1, pp. 37-43, 2024.

[21] S. Zhang, L. Wen, X. Bian, Z. Lei, and S. Z. Li, "Single-Shot Refinement Neural Network for Object Detection," Dec. 2018.

[22] Hidayatulloh Syarif Muhammad, "Sistem Pengenalan Wajah Menggunakan Metode Yolo (You Only Look Once)," 2021.

[23] Arganata Rifai Ahmad, "Analisis Perhitungan Bibit Ikan Gurame Menggunakan Webcam Dengan Metode Yolo (You Only Look Once)," 2020.

[24] A. C. S. S. A. Supriyanto Hadi, "Deteksi Helm Keselamatan Menggunakan Jetson Nano dan YOLOv7," Journal Of Applied Computer Science And Technology (JACOST), vol. 5, 2024, doi: 10.52158/jacost.637.

[25] N. L. A. F. Ilman Zidni Mukhamad, "Perbandingan Performa Jetson Nano, Jetson Xavier NX dan Lenovo Legion 5 terhadap Penggunaan YOLOv7," Apr. 2024.

[26] M. Luqman Bukhori and E. E. Prasetiyo, "Sistem Deteksi Masker Berbasis Jetson Nano dengan Deep Learning Framework TensorFlow," Jurnal Nasional Teknik Elektro Dan Teknologi Informasi, Sep. 2022.

[27] M. M. H. T. H. W. W. J. M. G. I. Aryeni Illa, "Application of Computer Vision for Real-Time Detection of Fruit Color and Size in Fruit Sorter," Journal Of Applied Electrical Engineering, vol. 7, pp. 1–6, Dec. 2023.

[28] Prisunia Fitriyati Sofianisa, "Pemanfaatan Jetson Nano Nvidia Untuk Mendeteksi Penggunaan Masker Secara Real-Time Menggunakan Opencv Python," 2023.

[29] A. Faqih, K. Mutmainnah, and M. R. Afifah, "Seperation Deteksi Kendaraan Pada Citra Digital Dengan Menggunakan Algoritma YOLO (You Only Look Once)," Jurnal Teknik Informatika dan Elektro, vol. 3, no. 2, 2021, [Online]. Available: https://jurnal.ugp.ac.id