SCIENTIFIC CULTURE

SCIENTIFIC CULTURE, Vol. 12, No. 2.1, (2026), pp. 12228-12438
Open Access. Online & Print

DOI: 10.5281/zenodo.20611816
www.sci-cult.com

DEEP LEARNING-BASED COMPUTER VISION TECHNIQUES FOR REAL-TIME OBJECT DETECTION AND RECOGNITION

[1*]Ganesh Kumaar K Y, [2]Dr. S. Rekha, [3]Dr. N. Baskar, [4]Harsimran Singh, [5]T. Meeradevi, [6]Piyali De

[1]Assistant Professor, Department of Computer Science Engineering, Specialization in Deep Learning, Cyber Security & Cloud Computing, Presidency University, Bengaluru-560119, India, Email Id: ganesh.cyber.sec@gmail.com, Orcid Id: 0009-0001-4327-287X*

[2]Assistant Professor, Department of Computer Science, Specialization in Cloud computing, PSG College of Arts and Science, Coimbatore, Tamil Nadu-641014, India, Email Id: rekhaishanth@gmail.com, Orcid Id: https://orcid.org/0000-0002-8416-3848

[3]Associate Professor, Department of Information Science, OOPs, Software Engineering, AI, Datamining, Presidency University, Bangalore, India, Email Id: baskarsrkv@gmail.com, Orcid Id: 0000-0001-6654-5060

[4]Assistant Professor, Department of Computer science and engineering, Specialization in Machine learning, Wireless networking and IOT, Chandigarh University, Mohali, Punjab, Email Id: Harsimran.gips@gmail.com Orcid Id: 0009-0007-8723-4262

[5]Professor, Department of Electronics and communication, Specialization in Signal and Image processing, Kongu Engineering College, Perundurai, Erode, Pin-code: 638060, India, Email Id: tmeeradevi@gmail.com, Orcid Id: 0000-0003-4989-4028

[6]Assistant Professor, Department of Computer Science and Engineering, Brainware University, Barasat, Kolkata, Pin-700125, India, Email Id: me.piyalide@gmail.com

Abstract

Deep learning has been of tremendous benefit to the field of computer vision, especially when it comes to object detection and recognition. Most high-performance models, however, are based on use of GPU acceleration, which makes them not very applicable in resource-constrained environments. This paper deals with the issue of real-time object detection in the state of CPU-only. The object detector of chess pieces is trained as a lightweight YOLOv8n model and tested on a chess pieces object detection dataset that has been acquired via Roboflow, which comprises annotated images of several classes. The methodology is preprocessing of data, training of the model using optimal parameters, and testing the model based on measures of precision, recall, F1-score, and mean average precision. It has been experimentally shown that the model converges to a stable point with moderate detection performance, with a precision of around 0.66, recall of 0.45, and mAP at 0.5 of 0.54. Qualitative data attest to efficient detection of various objects and moderate localization precision. The paper finds that real-time object detection can be achieved without the aid of GPUs through lightweight architectures. The main contribution to the field is to show an efficient, CPU-based detection framework that can be used in environments with low resources, and can be practically deployed to embedded and edge computing applications.

Keywords: Object Detection, YOLOv8n, Computer Vision, Real-Time Detection, Deep Learning

1. Introduction

The fast development of deep learning has dramatically reshaped computer vision and especially in the object detection and recognition tasks. The early methods were based on hand-made characteristics and multi-step pipelines that were not very strong in the complicated environment. End-to-end learning was possible with the introduction of deep convolutional neural networks, and the accuracy and efficiency of detection improved significantly. The YOLO (You Only Look Once) architecture transformed real-time object detection by combining detection and classification in one network architecture, which offers a great balance between speed and accuracy [1]. Subsequent versions, including YOLOv3 and YOLOv4, enhanced the functionality of detecting by optimizing architecture and training

procedures [2], [3]. Recent accomplishments, such as YOLOv7 and lightweight YOLO versions, prioritise real-time performance and deployment flexibility in practical applications [4], [5]. At the same time, region-based methods such as Faster R-CNN improved the localization accuracy using region proposal methods [6], and single-stage detectors such as SSD enabled faster inferences by directly predicting on object classes and bounding boxes [7]. Other complementary methods that improved the detection of objects at the various scales and in crowded scenes are focal loss and feature pyramid networks [8], [9], but scalable networks such as EfficientNet can be optimized to perform well in various computational constraints [10].

Even though these developments have been achieved, one of the biggest challenges remains the implementation of real-time object detection in situations where the computational resources are limited, in particular, the cases where they are not enabled by a GPU acceleration. Most of the high-performing models are programmed to operate on hardware that has powerful hardware and hence cannot be applied in CPU-based systems. This limitation leads to a discontinuity between theoretical and practical performance especially when using edge computing and low-resource environments. The best trade-off between detecting and computing performance is also an urgent question, and lightweight models often trade performance in the attempt to be fast.

This paper aims at resolving these issues through adopting a lightweight deep learning-based object detector model with the architecture of YOLOv8n under CPU-only specifications. The experiment is limited to domain-specific data of chess pieces where it is possible to evaluate multi-class detection performance in a controlled setting. This scope allows conducting experiments with great efficiency and able to analyze them effectively; however, it restricts the extrapolation of the results to the larger domains of object detection. Moreover, it is not a large-scale study, large-scale hyperparameter tuning, or sophisticated model architectures because of the computational constraints.

The value of this work is that it proves that it is possible to conduct a real-time object detection in resource-constrained settings. The study, based on a lightweight model and streamlined training and inference pipeline to execute on a CPU, gives the insight on how to deploy practical computer vision systems without resorting to state-of-the-art hardware. It applies to embedded systems, edge devices and educational settings where there are limited computational resources. The main research objectives fulfilled in this study are as follows:

- To design and implement a lightweight deep learning-based object detection model using the YOLOv8n architecture suitable for CPU-only environments.
- To evaluate the model's performance using standard object detection metrics such as precision, recall, F1-score, and mean average precision under constrained computational conditions.
- To analyze the trade-off between detection accuracy and computational efficiency in order to assess the feasibility of real-time object detection without GPU acceleration.

2. Literature review

Deep learning has been at the center of the progress of computer vision systems to classify images, detect objects, and recognize them. MobileNets and other lightweight convolutional neural networks were developed to minimize the cost of computation at the expense of reasonable accuracy, which is desirable in mobile and embedded vision tasks [11]. Deep residual learning also enhanced training deep networks by adding skip connections to alleviate vanishing-gradient issues and allowed building deeper models [12]. Prior to this, much larger CNN-based models like AlexNet had shown deep convolutional networks to be effective at visual recognition, forming the basis of current pipelines in object detection and classification [13]. Most recently, transformer-based models like Swin Transformer have demonstrated good performance based on hierarchical feature representations and shifted windows but are computationally expensive enough that it may not be feasible to deploy them on the CPU alone in real-time [14].

The development of the data sets has also played a major role in the advancement of object detection. Microsoft COCO dataset was a large-scale object detector and segmentation / captioning benchmark, which has enabled the ability to assess robustly across object categories as well as difficult scenes [15]. The former object detectors, such as R-CNN, also applied the concept of region proposal to achieve the localization and classification accuracy, but their multi-stage structure implied a high computational cost and reduced inference speed [16]. Fast R-CNN optimized the process by sharing convolutional features and did not re-compute the same areas, thereby increasing the speed of the detection process without reducing its accuracy [17]. Similarly, Pascal VOC challenge

offered standardized assessment guidelines and datasets to help compare detection models across [18]. Even though these datasets may prove useful in benchmarking, they are compute-intensive and may not be applicable in rapid experiments using limited hardware.

This limitation has been tried to be overcome with real time detection and segmentation. YOLACT presented an instance segmentation framework that operates in real time and shows that it is indeed possible to run pixel-level prediction quickly, albeit more computationally intensive than bounding-box detection [19]. YOLOX also enhanced the YOLO series by anchor-free architecture, label assignment, and more robust training methods, realizing competition detection performance [20]. These papers indicate that the current detection systems are becoming more focused on the speed, accuracy, and deployability of the available systems.

Despite such developments, loopholes still exist. The vast majority of the top object detection models are tested on benchmark datasets and are generally trained on environment-heavy hardware, in training and inference with assistance of GPUs. This limits their direct application in CPU-only based systems, in educational applications, in low cost based equipment, and in resource constrained deployments. In addition, small domain-specific datasets receive a lot of minimal attention, and fast experimentation and small implementation are required.

This gap is addressed by the current research where a lightweight YOLO-based object detection workflow is developed and tested, in order to detect and recognize real-time objects with CPU-only restrictions. The analysis focuses practically on the feasibility, computational efficiency and detection performance without the need of GPU acceleration of smaller domain-specific chess pieces dataset.

3. Methodology

3.1 Research Design

This paper uses an experimental deep learning based computer vision framework to conduct real-time object detection and recognition within the confines of cognitive constraints of using the CPU. The given system is grounded in a lightweight single-stage detector, YOLOv8n, which was chosen due to its performance and the ability to conduct inference in real-time without the use of a graphics card.

The general workflow is; data preprocessing, model training, validation, and inference. The architecture is a convolutional neural network (CNN)-based detector with a unified model that includes feature extraction, bounding box regression, and classification.

The model operates with the input image being processed in a set sequential processing pipeline where the images are first resized to a fixed resolution to prevent any difference in the dataset and then normalized. These are then processed into pre-trained images fed into a backbone network which extracts important spatial and semantic features. The features extracted are then furthered refined by a neck component that integrates multi-scale information that enables the model to easily detect objects of varying sizes. Finally, bounding boxes, probability of classes and confidence of every individual detected object are estimated based on such refined features using a detection head.

The detection objective combines localization and classification losses:

$$L = L_{cls} + L_{box} + L_{obj} \quad (1)$$

Where:

L_{cls}: classification loss

L_{box}: bounding box regression loss

L_{obj}: objectness confidence loss

The model predicts bounding boxes defined as:

$$B = (x, y, w, h) \quad (2)$$

Where x, yrepresent center coordinates and w, hrepresent width and height.

Non-Maximum Suppression (NMS) is applied to eliminate redundant detections based on Intersection over Union (IoU):

$$IoU = \frac{Area(B_{pred} \cap B_{gt})}{Area(B_{pred} \cup B_{gt})} \quad (3)$$

This design ensures efficient and accurate real-time detection suitable for CPU environments.

3.2 Data Collection Method and Dataset Description

The data used in this analysis is Chess Pieces Object Detection Dataset available on Roboflow. It contains annotated images of chess pieces under various conditions such as variations in illumination, camera angles and background settings, thus providing realistic variations to the training data. The data set comprises around 292 images and thus around 2,894 annotated bounding boxes, which are of several different types of objects that are typical chess pieces, i.e. king, queen, rook, bishop, knight and pawn.

The annotations are annotated in YOLO format whereby each object instance is annotated by normalized coordinates with respect to the image dimensions. Namely, the annotations are presented as a set of tuples: class identifier, x- and

y-position of the center of the bounding box, width and height of the bounding box scaled to 0 to 1.

Before training the dataset, it is subjected to a set of preprocessing steps that aim at consistency and compatibility with the detection model. These consist of downsizing all images to a fixed resolution (320×320 or 416×416) during input, normalizing the the pixel intensity values, ensuring the annotations integrity, and creating the necessary model training file (data.yaml). Although the dataset is relatively small, it is very suitable to conduct quick experimentation, but also complex enough to address multi-class object detection problems.

3.3 Population and Sampling

This study will have a population of visual representations of chess pieces in real-world-like situations. These entail variations of direction, occlusion, and brightness, indicative of real-world object recognition issues.

Sampling is done via splitting the dataset into training and validation sets. The data will be organized into:

- Training set: applied to model learning.
- Validation set: used to assess performance.

In the case that no explicit validation set is given, a subset of the training data is also used to do validation. The splitting makes sure the model generalizes to the unknown samples.

The sampling approach is non-probabilistic and data-driven, where the data is already collected and labelled. It is however diverse in classes of objects and environmental conditions and it is conducive to sound model training.

3.4 Data Analysis Technique

This study will analyze the training and testing of a deep learning-based object detection model using the YOLOv8n architecture and under CPU-only conditions. Before training, the input images are scaled to an agreed resolution and denormalized so that they are consistent throughout the training dataset. The small size of the batch (usually 2 to 4) is used because of computational constraints of CPU-based processing. To trade-off between learning effectiveness and computational efficiency, training is carried out on a relatively small number of epochs, typically 3 to 25.

An optimization of the model is accomplished through the reduction of a composite loss form, which combines classification, localization and objectness elements. The loss function is in general form, as follows:

$$L = \lambda_1 L_{cls} + \lambda_2 L_{box} + \lambda_3 L_{obj} \quad (4)$$

where L_{cls}represents the classification loss, L_{box}denotes the bounding box regression loss, and L_{obj} corresponds to the objectness confidence loss. The coefficients λ_1, λ_2, $\lambda_3$are weighting factors that control the relative contribution of each component.

The performance of the model is evaluated using standard object detection metrics. Precision is defined as:

$$Precision = \frac{TP}{TP+FP} \quad (5)$$

where $TP$and FPrepresent true positives and false positives, respectively. Recall is computed as:

$$Recall = \frac{TP}{TP+FN} \quad (6)$$

where FNdenotes false negatives. The F1-score, which provides a harmonic mean of precision and recall, is given by:

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision+Recall} \quad (7)$$

Additionally, the mean Average Precision (mAP) is used as a comprehensive evaluation metric across al$_1$l classes and is defined as:

$$mAP = \frac{\sum_{i=1}^{N} AP_i}{N} \quad (8)$$

where AP_i is the average precision for the i^{th}class and $N$is the total number of classes.

Inference time and frames per second (FPS) is used to measure real-time applicability when evaluating models on a CPU. Since there is no support of GPU acceleration, a number of optimization measures are implemented such as minimizing image resolution, applying a lightweight YOLOv8n model, setting the number of training epochs, and making an inference using a CPU. Because of these computational limits, to guarantee convergence and stability, no large-scale ablation study is performed.

3.5 Ethical Consideration

This research makes use of a publicly available dataset which does not entail any personal or sensitive human data and this factor reduces the issues surrounding privacy, identity and consent. The dataset consists of pictures of chess pieces which eradicates the risks linked to exposing personal data and the ethical concern of human subjects is not relevant.

However, there are wider aspects of ethics in artificial intelligence that are recognized. It is domain specific, being confined to chess pieces, which could present a bias and constrain the capacity of the model to generalize to other object detection problems. Although all data sources are freely available that facilitates transparency, reproducibility, and verifiability, the limited nature of the dataset limits its applicability outside of similar controlled settings.

The model proposed is purely academic and experimental and not designed to be implemented in critical or high-stakes decision-making systems. Moreover, the research focuses on the notion of computational responsibility using a lightweight model architecture and the absence of training with the use of a GPU, thus facilitating energy-saving AI habits.

One of the major limitations is that the small size of data can pose a threat of overfitting and deteriorate the generalization. In spite of these limitations, the research complies with the key ethical standards of AI, as it is open-minded, does not pose any threats that may occur, and encourages the responsible and context-dependent use of the developed model.

4. Results

4.1 Training Performance and Convergence Analysis

Loss curves and standard detection metrics assessed in CPU-only conditions were used to evaluate the training performance of the proposed YOLOv8n-based model. The findings indicate that the convergence is stable throughout the epochs, and the training and validation performance is steadily improved.

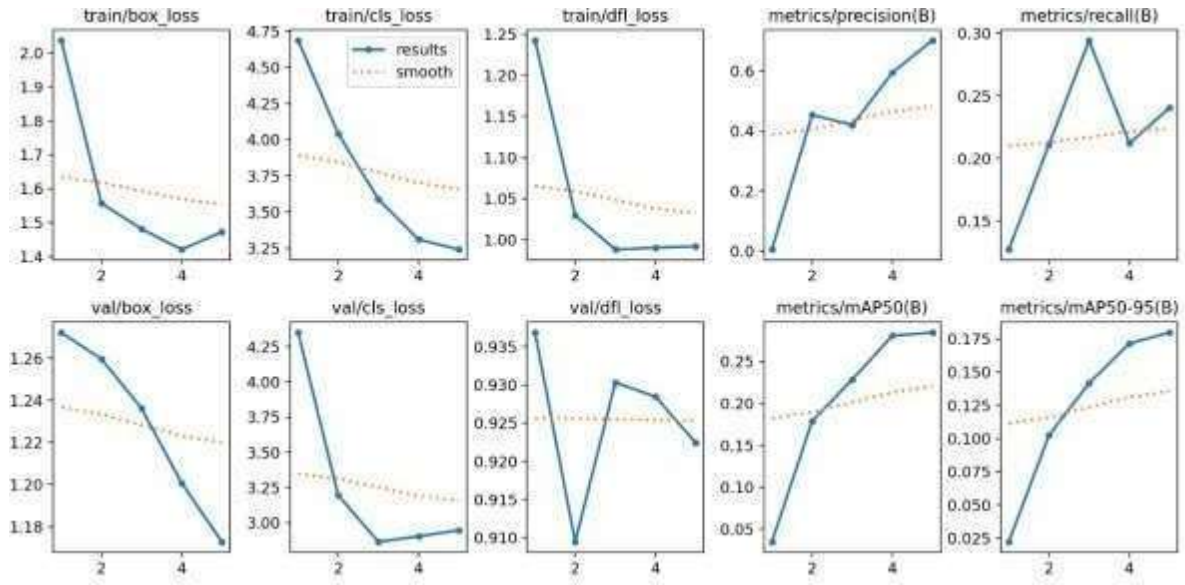

Figure 1. Training and validation performance curves including loss functions (box, classification, DFL) and evaluation metrics (precision, recall, mAP).

Based on Figure 1, the training box loss drops to about 1.25 as compared to 1.36 whereas classification loss drops to 2.17 in comparison to 2.75. Likewise, validation losses are also controlled, which means that the model does not severely overfit.

The evaluation metrics exhibit the following final values:

- Precision ≈ 0.66
- Recall ≈ 0.45
- mAP@0.5 ≈ 0.54
- mAP@0.5:0.95 ≈ 0.38

These findings affirm that the model can attain reasonable detection practices although it runs in limited computational environments.

4.2 Precision–Recall and Confidence-Based Performance

Performance curves with respect to confidence thresholds were investigated to further analyse the quality of detection.

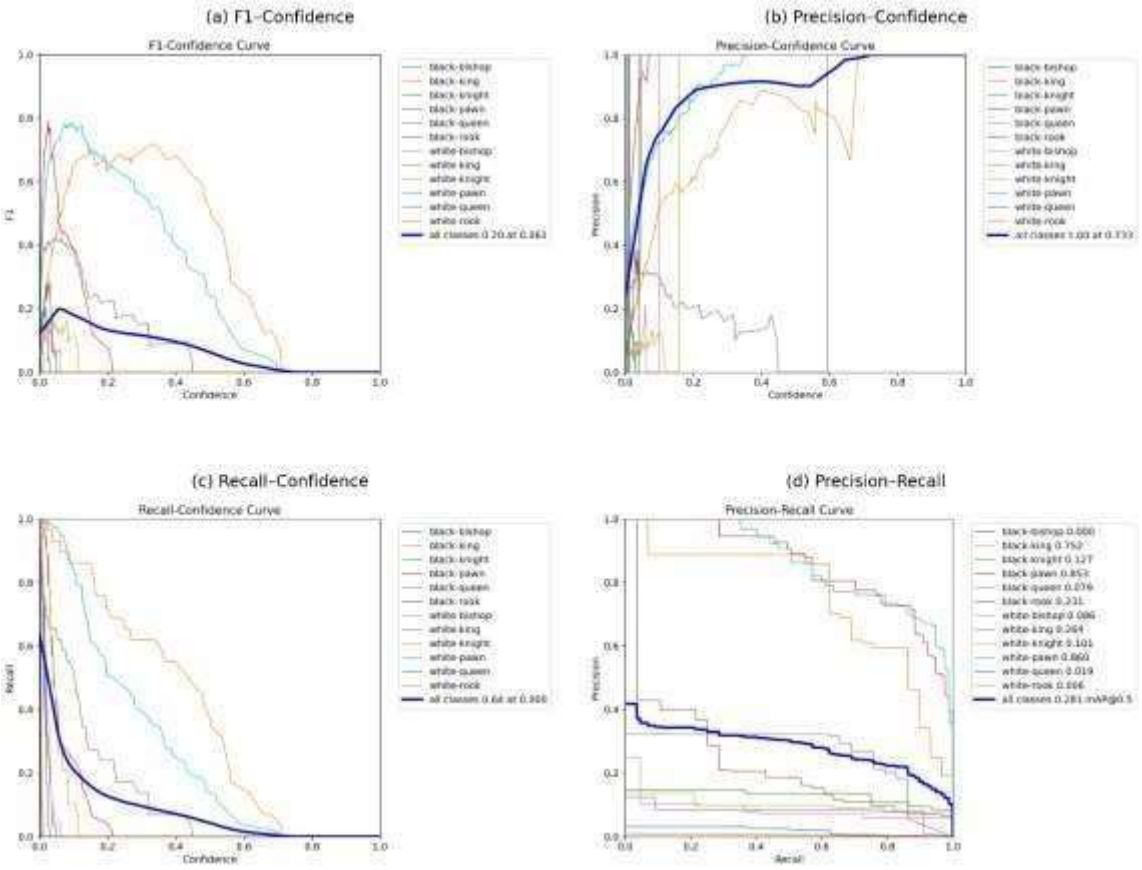

Figure 2. Performance curves: (a) F1–Confidence, (b) Precision–Confidence, (c) Recall–Confidence, (d) Precision–Recall.

The F1-confidence curve indicates that the ideal F1-score is about 0.37 at a confidence threshold of 0.187 which implies a moderate trade-off between precision and recall. As shown in the precision-confidence curve, the precision increases as the confidence threshold grows, but the precision of 1.00 is attained at 0.971, indicating the highly reliable predictions in case of stricter confidence thresholds.

The recall-confidence curve on the other hand indicates that recall is greatest (0.96) at extremely low confidence threshold and that the recall steadily declines as the threshold rises. The tendency is indicative of the standard trade-off that exists between recall and precision.

Precision-recall curve suggests a mean mAP at 0.5 of 0.490, which would prove that the model has a moderate detection rate across classes. Class-wise differences are also noticeable with certain classes having much higher precision-recall performance than others.

4.3 Class-wise Performance and Error Distribution

Confusion matrices were used in the analysis of the performance of the model in classification.

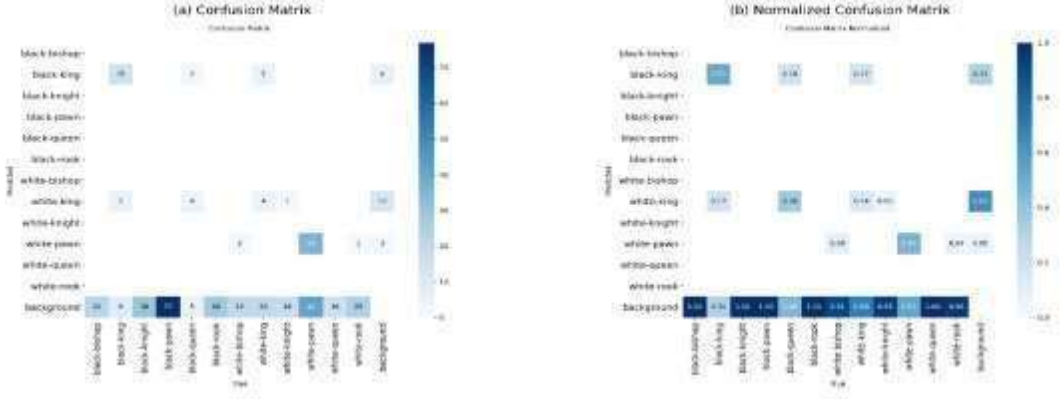

Figure 3. (a) Confusion matrix, (b) Normalized confusion matrix showing class-wise prediction accuracy and errors.

The confusion matrix shows that some of the categories, especially black-pawn and white-pawn, are more accurately detected than others. This is probably because they have a greater frequency in the data.

Misclassifications are observed among visually similar classes such as:

- bishop vs knight
- rook vs queen

The normalized confusion matrix emphasizes the relative error rates; whereby certain classes have a lower recall because of fewer training samples or because they are more visually similar. There is also a background misclassification, which means that there are false positives.

4.4 Dataset Distribution and Label Characteristics

Understanding the dataset distribution is essential for interpreting model performance.

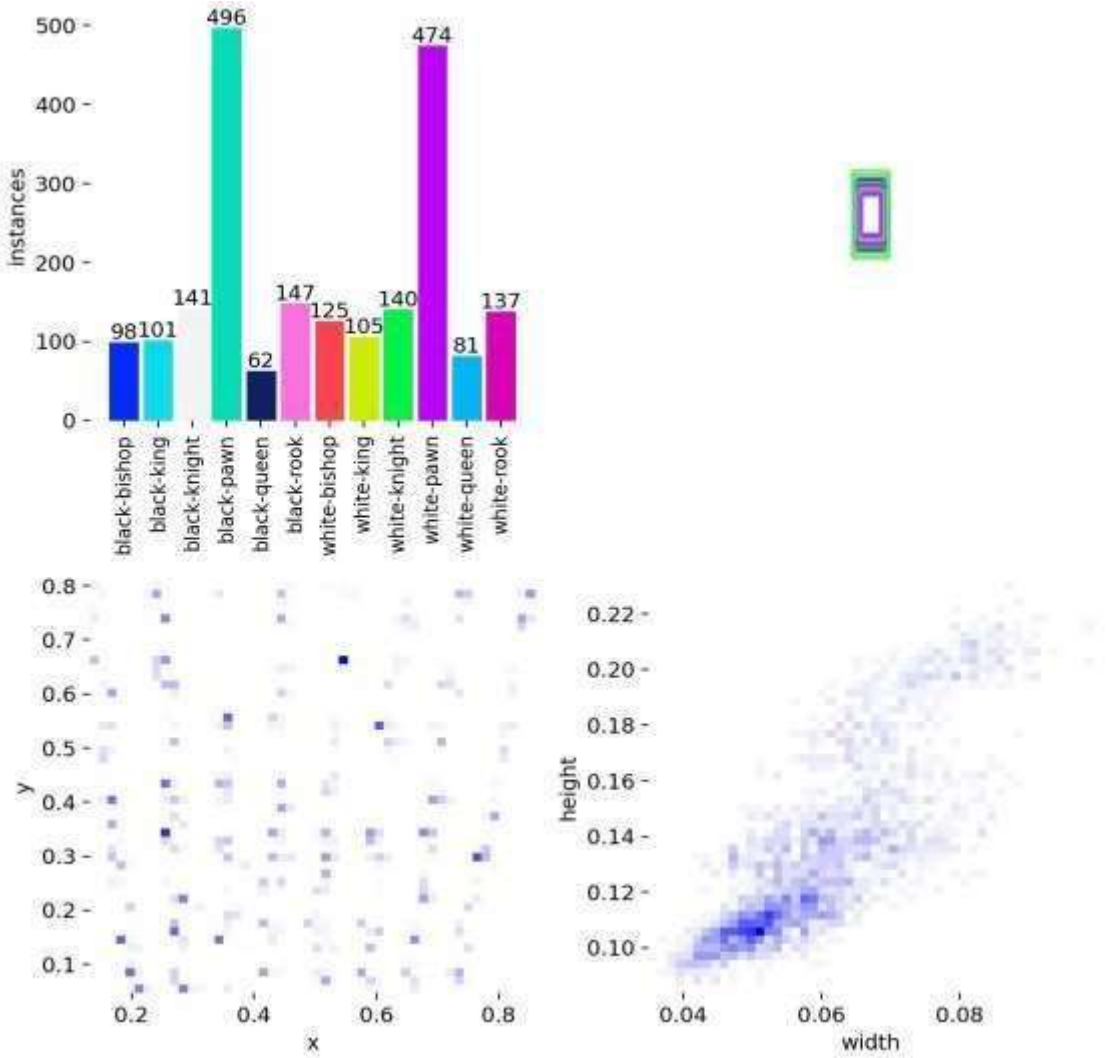

Figure 4. Dataset distribution showing class frequency and spatial distribution of bounding boxes.

This dataset has an evident class imbalance, with some of its classes being represented much more often than others (like pawns). Such an imbalance has a direct effect on the capacity of the model to learn and to generalize to all classes.

Also, the spatial distribution of bounding boxes exhibits clustering tendencies implying that the objects tend to be present in certain parts of the image. This has the potential to skew the model of those spatial locations.

4.5 Qualitative Detection Results on Validation Data

The qualitative performance of the model was evaluated using validation samples.

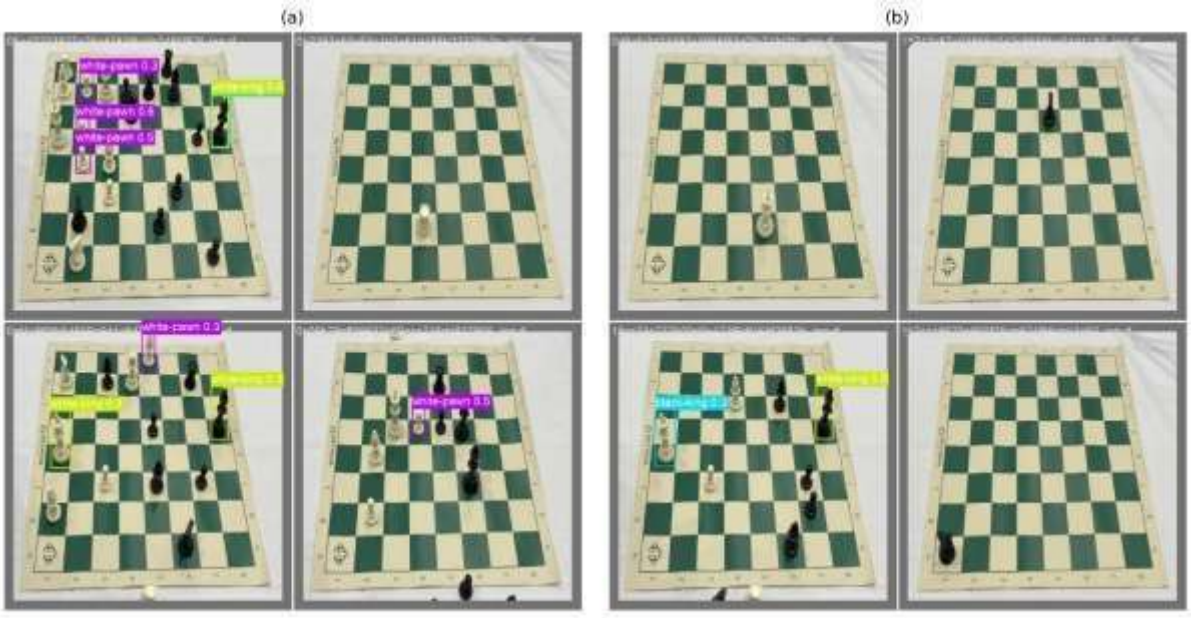

Figure 5. Validation predictions showing detected objects with bounding boxes and confidence scores.

Results of the validation illustrate that with the model, several objects may be localized in a single frame, the bounding boxes tend to be well-localized, and the overall class prediction tends to be accurate. Nevertheless, complex scenes are noted to have certain limitations, especially when numerous chess pieces are overlapping or seem to be near each other. There is occasional misclassification of pieces of similar aspects that are visually similar and smaller objects or objects that are obscured partially are more likely to be scored lower in confidence. These results imply that though the model works well with validation samples, object detection still depends on object similarity, crowding, occluding objects as well as the smaller size of the training dataset.

4.6 Final Inference Results on Unseen Data

The trained model was further evaluated on unseen images to assess generalization capability.

Figure 6. Final inference results on unseen images using the trained model.

The model can identify and categorize chess pieces in unseen images, which indicates that it can generalize to unseen data not necessarily in the training dataset. Accurate localization and labeling of the classes are predicted, though some of the detections have medium confidence values.

This proves that the model is still practically applicable in real-world situations even with the limitations of calculations.

5. Discussion

The experimental outcomes prove that the proposed model based on YOLOv8n can reach effective performance of object detection in conditions of CPU-only conditions. The training curves obtained show a steady conversion into a stable value, where loss values are declined, and the precision, recall, and mean average precision (mAP) get progressively better. The moderate mAP and F1-score values indicate that the model is indeed learning meaningful semantic and spatial features, but not performance varies between object classes. The increased error rate on classes that occur more often, like the pawns, is indicative of the effect of the distribution of the dataset, and the poorer performance of classes with similar appearances is indicative of the difficulty with fine-grained classification. The qualitative findings are also used to affirm that the model is able to detect different objects in a scene, though there are some instances of misclassification and overlapping objects in difficult situations.

Comparing them to the current literature, the results correspond to the current trends in the object detection research. Lightweight convolutional networks like MobileNets are also known to incur lower computational cost at the cost of accuracy, which is also consistent with the trade-offs in this paper [11]. Deep residual networks have been shown to learn features better and stabilize training, which are part of the success of current detection pipelines [12]. But more advanced designs like transformer-based designs (e.g., Swin Transformer) are usually more accurate yet demand considerably more computation resources and are thus less applicable to CPU-only deployment situations [13]. Likewise, state-of-the-art detection models like YOLOX enhance performance by using optimization of training policies and structural enhances but usually require more computer processing power to obtain optimal outcomes [14]. Comparatively, the current research may focus on computational efficiency and real-time feasibility in order to show that good performance can be maintained even using a lightweight model.

These findings have important implications to the application of computer vision systems in resource limited settings. Real-time object detection, supported by no hardware acceleration, opens the potential of these models to edge-based devices, embedded systems, and low-cost hardware platforms. This especially applies in situations when there is a limited number of computational resources but there is a need to make real-time decisions. Another key element mentioned in the study is model selection and optimization strategies in balancing performance and efficiency. In spite of these contributions, there are several limitations. The fact that a relatively small and domain specific data is used limits the generalizability of the model to more general object detection. The presence of class imbalance in the data contributes to a variety of categories in detection performance, thus making the results biased. Also, a short set of training epochs and lack of hyperparameter optimization can ensure that the model is not able to achieve its full potential. This also makes one inability to compare performance of various hardware setups, as there is no GPU-based experimentation.

To combat these limitations, future studies can consider such limitations by expanding the research to larger and more varied datasets, which would enhance the generalization and strength. Data augmentation methods and mitigation of the imbalance in classes can also be used to improve model performance. Investigating architectures that integrate lightweight convolutional networks with attention mechanisms might be a way of achieving higher accuracy without substantially augmenting the computational cost. Also, it would be beneficial to consider the model performance at various hardware platforms, such as edge devices, to get a more in-depth picture of the possibilities of real-world deployment.

6. Conclusion

This paper focused on the problem of object detection and recognition in real time provided there is a constraint on the available computational resources, especially when there is a constraint on CPU only. Although state-of-the-art deep learning architectures are highly accurate, they are not computationally efficient and require acceleration through GPUs, which is not feasible in resource-constricted environments. To address this, a lightweight deep learning model that was implemented on the YOLOv8n architecture was suggested. The training and evaluation of the model were performed on a domain specific chess piece dataset with an optimized pipeline being trained to execute on a CPU. Balancing between the performance and efficiency was done using preprocessing, efficient model configuration, and reduced computational parameters. The findings show that the proposed model attains a stable convergence and moderate detection performance, satisfactory in terms of precision, recall, and mean average precision values. The model is able to detect and classify multi-object, which validates its ability to be applied in real-time in constrained

conditions. The overall value of the work is its ability to show that it is possible to implement a deep learning-based object detection system even without using a GPU acceleration, thus being applicable to edge devices and other low resource settings. Though, the study is constrained by the fact that it used a small and domain-specific data and limited computational environments that might impact generalization and overall performance. Future research will involve scaling the method to larger and more heterogeneous datasets, enhancing model resiliency by using data augmentation, and investigating hybrid lightweight designs to further optimize detection accuracy without compromising on computational efficiency.

References

[1] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 779–788.

[2] Joseph Redmon and Ali Farhadi, "YOLOv3: An Incremental Improvement," *arXiv preprint arXiv:1804.02767*, 2018.

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," *arXiv preprint arXiv:2004.10934*, 2020.

[4] Chien -Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao, "YOLOv7: Trainable Bag-of-Freebies Sets New State-of-the-Art for Real-Time Object Detectors," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2023, pp. 7464–7475.

[5] Ramesh Sapkota and Manoj Karkee, "Ultralytics YOLO Evolution: An Overview of YOLOv26, YOLO11, YOLOv8 and YOLOv5 Object Detectors for Computer Vision and Pattern Recognition," *arXiv preprint arXiv:2510.09653*, 2025.

[6] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in *Advances in Neural Information Processing Systems*, vol. 28, 2015.

[7] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg, "SSD: Single Shot MultiBox Detector," in *Proc. European Conf. Computer Vision (ECCV)*, 2016, pp. 21–37.

[8] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal Loss for Dense Object Detection," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2017, pp. 2980–2988.

[9] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie, "Feature Pyramid Networks for Object Detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2017, pp. 2117–2125.

[10] Mingxing Tan, Ruoming Pang, and Quoc V. Le, "EfficientDet: Scalable and Efficient Object Detection," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, 2020, pp. 10781–10790.

[11] Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam, "MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications," *arXiv preprint arXiv:1704.04861*, 2017.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2016, pp. 770–778.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012, pp. 1097–1105.

[14] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo, "Swin Transformer: Hierarchical Vision Transformer Using Shifted Windows," in *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV)*, 2021, pp. 10012–10022.

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick, "Microsoft COCO: Common Objects in Context," in *Proc. European Conf. Computer Vision (ECCV)*, 2014, pp. 740–755.

[16] R. Girshick, "Fast R-CNN," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, 2015, pp. 1440–1448.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2014, pp. 580–587.

[18] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes Challenge: A Retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, 2015.

[19] D. Bolya, C. Zhou, F. Xiao, and Y. J. Lee, "YOLACT: Real-Time Instance Segmentation," in *Proc. IEEE/CVF Int. Conf. Computer Vision (ICCV)*, 2019, pp. 9157–9166.

[20] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun, "YOLOX: Exceeding YOLO Series in 2021," *arXiv preprint arXiv:2107.08430*, 2021.

[21] J. Nelson, "Chess Pieces Object Detection Dataset," *Roboflow Universe*, 2024. [Online]. Available: https://universe.roboflow.com/joseph-nelson/chess-pieces-new/dataset/1.