World Journal of Information Technology
ISSN: 2959-9911
DOI: 10.61784/wjit3002

YOLO AND COCO DATASET DETECTIVE PORFERMANCE

MingYu Gao
School of Mathematics and Physics, Xi'an Jiaotong-Liverpool University, Suzhou 215123, Jiangsu, China.
Corresponding Email: Mingyu.Gao2302@student.xjtlu.edu.cn

Abstract: As convolutional neural networks continue to advance, more excellent models for image recognition have emerged in the field of computer vision. These models can help doctors identify causes of diseases in the medical field, reduce accidents in the transportation field, and collect facial recognition information in the security field. This study mainly focuses on the improvements of YOLOv8 compared to previous versions and its performance after training on the COCO dataset. It also briefly discusses the comparative results of YOLO with RCNN and SSD. Additionally, the development history of YOLO is introduced, with an emphasis on the performance of YOLOv8 after training and analysis the data. In the end, we see the future prospects of YOLO algorithm.
Keywords: YOLO; COCO dataset; Images

1 INTRODUCTION

With the improvement of living standards, speed and convenience has become more and more important. There are more and more unmanned supermarkets, unmanned driving, automated farming, drone object recognition and so on. People want to find what they need quickly, and out of the need for speed and accuracy, the YOLO（you only look once） algorithm was born. In any case, this algorithm facilitates people's lives and changes the way machines behave.
Compare YOLO and Randone of the most significant advantages of YOLO over RCNN is speed. YOLO processes the entire image in a single forward pass of the network, making it extremely fast and suitable for real-time applications. In contrast, RCNN involves a multi-step process: generating region proposals, running a CNN on each region, and then classifying these regions, which makes it much slower [1]. YOLO's architecture is simpler and more straightforward. It eliminates the need for the region proposal stage, which is a critical and time-consuming component of RCNN. By doing everything in one network, YOLO simplifies both the training and inference processes, leading to a more streamlined pipeline [1]. YOLO allows for end-to-end training, optimizing the entire model in one go. This contrasts with RCNN, which often requires separate training stages for different components. End-to-end training can result in better performance and easier implementation[1]. YOLO considers the entire image during training and testing, which helps it capture contextual information better than RCNN, which only focuses on the regions of interest. This holistic approach can reduce false positives and improve overall detection accuracy for larger objects [1].
SSD is also a one stage object detection algorithm like YOLO. Yolo maintains its advantage in speed even over SSD, which is also designed for real-time object detection. while both yolo and SSD are fast, YOLO's single forward pass through the network is often more efficient than SSD's approach, which uses multiple convolutional layers for predictions at different scales. this makes yolo particularly suitable for applications that require ultra-low latency [2]. YOLO's architecture is inherently simpler than SSD's. SSD involves multiple layers of feature maps and different scales for detecting objects, adding complexity to the model. YOLO, on the other hand, uses a straightforward grid system for prediction, making it easier to implement and understand [2]. YOLO's architecture is inherently simpler than SSD's. SSD involves multiple layers of feature maps and different scales for detecting objects, adding complexity to the model. YOLO, on the other hand, uses a straightforward grid system for prediction, making it easier to implement and understand. YOLO treats object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities. This unified approach contrasts with SSD's multiple stages of feature extraction and prediction, potentially leading to more consistent and coherent training and inference [2]. Similar to its advantage over RCNN, YOLO benefits from processing the entire image at once, utilizing global context more effectively than SSD. This can be particularly beneficial in scenarios where objects are larger or where understanding the broader scene is crucial for accurate detection [2].
The primary advantages of YOLO over RCNN and SSD are its simplicity, speed, and efficient use of global context. Because of these features, YOLO is a great fit for real-time applications and situations where a simplified, effective pipeline is necessary.

2 EVOLUTION OF YOLO

The YOLO (you only look once) algorithm, a new object detection algorithm, was proposed in 2015 by a team led by Joseph Redmon. As the name "You Only Look Once" suggests, it differs from the previous two-stage algorithms like RCNN; it only needs to look once. It processes the input image through a single neural network to directly obtain the segmented grid. Because the entire detection process involves just one network, it allows for end-to-end optimization directly [4].
Equations

2.1 YOLO Model

The core idea of YOLO is to divide the input image into an S x S grid, with each grid cell responsible for detecting objects within it. For each grid cell, YOLO predicts B bounding boxes and a confidence score for each box, along with the probability distribution for C classes [4].

$$S \times S \times (B * 5 + C) \tag{1}$$

where S is the grid size, B is the number of bounding boxes predicted per grid cell, and C is the number of classes. Specifically, each bounding box is described by five values: the center coordinates (x, y), width w, height h, and confidence score c. The confidence score reflects the probability that the bounding box contains an object and how well it overlaps with the actual object.

For each square the output is

$$\hat{y}_{i,j} = (\hat{x},\hat{y},\hat{\omega},\hat{h},\hat{c},\hat{p}_1,\hat{p}_2,\hat{p}_C) \tag{2}$$

$$\hat{c} = P(Object) \times IOU_{pred}^{truth}\# \tag{3}$$

Loss Function

The YOLO loss function consists of three parts: localization error, confidence error, and classification error. The overall loss function can be expressed as:

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2]$$

$$+ \lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left[\left(\sqrt{\omega_i} - \sqrt{\hat{\omega}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{obj} \left(C_i - \hat{C}_i \right)^2$$

$$+$$

$$\lambda_{coord} \sum_{i=0}^{s^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{noobj} \left(C_i - \hat{C}_i \right)^2$$

$$+ \sum_{i=0}^{s^2} \mathbb{1}_{i}^{obj} \sum_{c \in classes} (p_i(c) - \hat{p}_i(c))^2 \tag{4}$$

In this loss function, it means grid cell and j means bounding box. Since each grid cell can only predict two boxes and one class, YOLO imposes strong spatial constraints on the prediction of bounding boxes. This spatial constraint limits the number of nearby objects our model can predict. As a result, our model struggles to predict small objects that appear in groups (such as flocks of birds).

2.2 YOLOv2 (YOLO9000)

YOLOv2, also known as YOLO9000, improves recall and localization accuracy compared to the original version. It adds Batch Normalization to all convolutional layers in the YOLO model

$$BN(X_{test}) = \gamma \cdot \frac{X_{test} - \mu_{test}}{\sqrt{\sigma_{test}^2 + \epsilon}} + \beta \tag{5}$$

$$\mu_{test} = \mathbb{E}(\mu_{batch}) \tag{6}$$

$$\sigma_{test}^2 = \frac{m}{m-1} \mathbb{E}\left(\sigma_{batch}^2\right) \tag{7}$$

YOLO9000 removes the fully connected layers and the last pooling layer, allowing the final convolutional layers to have higher resolution features. It also reduces the input image size to 416×416, positioning objects more often at the center of the image. Therefore, it is preferable to predict these objects with a dedicated location at the center rather than at the four surrounding positions. And use the Anchor Boxes. Each grid cell generates 5 anchor boxes. By calculating the Intersection over Union (IOU), one anchor is selected to produce the prediction box that best fits the ground truth box [6].

2.3 YOLOv3

Move to YOLOv3 the backbone network has been improved in YOLOv3, which uses multi-scale feature maps for detection and replaces the SoftMax with multiple independent logistic regression classifiers for class prediction. YOLOv3 advances from YOLOv2 by producing predictions at three scales: 13×13, 26×26, and 52×52, corresponding to the number of grid cells.

$$N \times N \times [3 * (4 + 1 + 80)] \tag{8}$$

Each grid cell generates 3 anchor boxes, and an anchor is selected based on the IOU with the ground truth to produce the prediction box that best fits the true box.
YOLOv3 uses a new network for feature extraction, which is a hybrid approach between the network used in YOLOv2, Darknet-19, and the newer Darknet network. This network employs a combination of 3×3 and 1×1 convolutional layers with additional shortcut connections and is significantly larger. It is referred to as Darknet-53. Darknet-53's performance is comparable to state-of-the-art classifiers but requires fewer floating-point operations, making it faster. Additionally, Darknet-53 achieves the highest floating-point operations per second. This means that the network structure utilizes the GPU more effectively, leading to more efficient evaluations and faster processing [6].

2.4 YOLOv4

YOLOv4 can be seen as an ensemble, with its network primarily based on Darknet-53, but it enhances feature representation capabilities through the use of the CSP (Cross Stage Partial) module, resulting in the new backbone network structure known as CSPDarknet-53. By introducing the SAM (Spatial Attention Module) module, YOLOv4 can adaptively adjust the channel attention weights of the feature maps, enhancing its ability to perceive objects. It also introduces a new distance metric called CIOU (Complete Intersection over Union). CIOU is an improved loss function for object detection that measures the distance between the predicted box and the ground truth box. CIOU is an extension of DIoU (Distance Intersection over Union) and, besides considering the distance between the positions and shapes of the boxes, it also incorporates an additional parameter to measure the consistency of the aspect ratio of the boxes [7].

$$CIOU = IoU - \left(\frac{d^2}{c^2} + \alpha v\right) + v \tag{9}$$

In CIOU, d represents the Euclidean distance between the center points of the predicted box and the ground truth box, and c represents the diagonal distance of the smallest enclosing box covering the two boxes. In CIOU, is a parameter used to balance the consistency of the aspect ratio of the boxes and the distance between the box positions. v is an auxiliary term used to penalize the difference in aspect ratios between the predicted box and the ground truth box [7].

2.5 YOLOv5

YOLOv5 introduced the Focus structure, an important component for extracting high-resolution features. It employs a lightweight convolution operation that helps the model maintain a high receptive field while reducing computational burden. The Focus structure slices the input feature map by channels and spatially, transforming the original feature map into a smaller-sized feature map while retaining important information. This approach enhances the model's perceptive ability and improves the accuracy of detecting small-sized objects [8, 10].

2.6 YOLOv8

YOLOv8 reintroduced the Darknet53 structure and replaced the C3 module with the C2F module. For the loss function calculation, it adopted the Task Aligned Assigner strategy for positive sample assignment. This approach combines three loss functions: classification loss (Varifocal Loss, VFL) and regression loss (Complete Intersection over Union, CIOU) with Deep Feature Loss (DFL), which is new weighted together. Additionally, YOLOv8 moved to an anchor-free approach, eliminating the use of anchor boxes.

3 COCO DATASET

The COCO dataset, which was first presented by Lin et al. [9] in 2014, is a varied collection of photos showing common things in a range of settings. It is an important tool for developing and evaluating object identification algorithms. An overview of the COCO dataset's structure, methods for gathering and annotating data, and evaluation criteria are given in this section.
The COCO dataset is a large and rich dataset for object detection, segmentation, and captioning. Aimed at scene understanding, this dataset primarily comprises images taken from complex everyday scenes, with objects precisely located through segmentation. The images include 91 object categories, 328,000 images, and 2,500,000 labels. It is currently the largest dataset for semantic segmentation, offering 80 categories, with over 330,000 images, of which 200,000 are annotated. The total number of instances in the dataset exceeds 1.5 million.
After the ImageNet competition was discontinued, the COCO competition became the most authoritative and important benchmark in the fields of object recognition and detection. It is currently the only international competition in this field that brings together top institutions like Google, Microsoft, Facebook, and many leading universities and innovative enterprises worldwide [18].
The COCO dataset addresses three main problems: object detection, the contextual relationships between objects, and the precise 2D localization of objects. The COCO dataset includes 91 categories, which is fewer than ImageNet and SUN, but it has more images per category. This helps in gaining better capability for recognizing objects in specific scenes. Compared to PASCAL VOC, COCO offers more categories and images [9].

4 YOLOv8

4.1 Theoretical Foundation of YOLOv8

The fundamental idea of the YOLO series is carried over into YOLOv8, which is to identify objects directly in a single neural network instead of using a region proposal network (RPN) and then classifying them. With the addition of more effective feature extraction networks and enhanced loss functions, YOLOv8 further optimizes the model architecture. With these improvements, YOLOv8 can continue to achieve high-speed detection while greatly increasing detection accuracy.

4.2 Improvement Methods

A number of studies have looked into particular enhancements for YOLOv8. For instance, in order to improve the model's capacity to identify objects at various scales, several research have added new convolutional modules and attention mechanisms. Previous research has concentrated on refining data augmentation techniques and training methodologies to increase the model's resilience and capacity for generalization. With these enhancements, YOLOv8 has shown notable advances in performance across a number of benchmark datasets [10-12].

4.3 Application Areas

YOLOv8 has proven to be highly applicable in a variety of sectors. It is extensively utilized in medical picture analysis, intelligent surveillance, and autonomous driving. To improve driving safety, YOLOv8 is utilized in autonomous driving to instantly identify and classify pedestrians, cars, and traffic signs. YOLOv8 can follow and identify several targets in real-time using intelligent surveillance, increasing the effectiveness of security systems [13-15].

4.4 Performance Evaluation

The exceptional performance of YOLOv8 has been empirically proven in numerous investigations. On a number of datasets, YOLOv8 outperforms YOLOv7 in terms of accuracy and detection speed. Additionally, YOLOv8 has been compared in some studies to other popular detection models, like SSD and Faster R-CNN, showing that YOLOv8 strikes the optimum balance between speed and accuracy. YOLOv8 can yet be improved upon, despite its considerable advancement. Prospective avenues for investigation encompass investigating more effective network configurations, integrating additional contextual data, and refining multi-object tracking algorithms. Furthermore, attaining effective detection in low-computation settings continues to be a crucial area of study [16-17].

5 YOLOV8 AND COCO DATASET

Because of its richness and diversity, the COCO dataset is frequently used to assess object identification algorithms, including YOLOv8. This section highlights the benefits and drawbacks of YOLOv8 by contrasting its performance on the COCO dataset with that of other state-of-the-art algorithms [18].
Performance
YOLOv8 demonstrated competitive performance on the COCO dataset, with high mean average accuracy (mAP) scores across multiple intersection over union (IoU) criteria. The model's performance on the COCO dataset has been improved by the Yolov8 changes, which include a redesigned backbone, improved anchor boxes, and improved feature extraction. This has allowed the model to detect objects of various sizes and under challenging circumstances.
Step 1: Convert the COCO dataset to YOLO format
1. Read the COCO JSON file:
Parse the image, annotation, and category information from the JSON file.
2. Create save paths:
Ensure the paths for saving annotation files are created.
3. Create category ID mapping:
Map the COCO category IDs to new indices.
4. Initialize annotation lists:
Find the maximum image ID and initialize the annotation list.
5. Create image annotation dictionary:
Group each annotation by image ID.
6. Process each image:
Get image information and create corresponding annotation text files.
7. Convert annotations to YOLO format:
Write the annotations in YOLO format to the text files.
Step 2: YOLOv8 Model Training and Evaluation
1. Model Training:
Use YOLOv5 for model training (assuming YOLOv8 follows a similar process).
Set input image size, batch size, training epochs, data configuration file, pre-trained weights, and data caching options through command-line arguments.
2. Inference with the trained model:

Load the trained model for inference.
Set the weight file, input image size, confidence threshold, and data source path to perform the inference task.
3. Evaluate the trained model:
Set the weight file, data configuration file, and input image size to perform the evaluation task and generate performance metrics.
4. Generate and display confusion matrix:
Visualize the model's classification performance across different categories.
Step 3: Use the Trained Model for Inference and Display Results
1. Load the trained model for inference:
Perform the inference using the trained model.
2. Process and display inference results:
Visualize the results of the inference.
Step 4: Evaluate Model Performance and Generate Confusion Matrix
1. Evaluate model performance:
Use the val.py script to evaluate the model's performance.
2. Load the confusion matrix file generated during evaluation:
Visualize the confusion matrix to assess the model's performance across different categories.
By following these steps, you can successfully convert the COCO dataset to YOLO format, train and evaluate the YOLOv8 model, and visualize its performance using a confusion matrix.

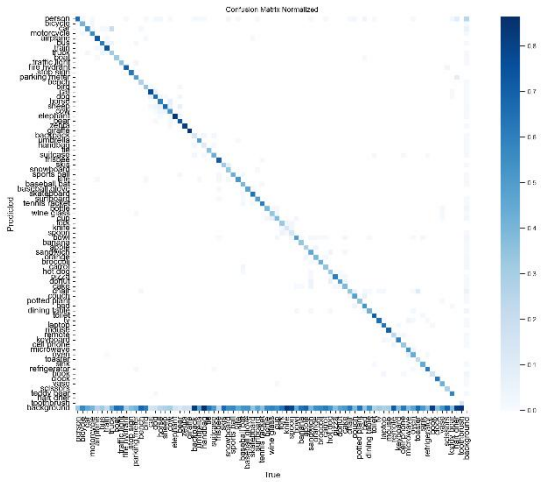

Figure 1：Confusion Matrix Normalization

Figure 1 this confusion matrix illustrates the model's prediction performance across different categories. The horizontal axis represents the true categories, and the vertical axis represents the predicted categories. The depth of color indicates the degree of confusion between categories.
The shade of the color indicates the number of times. On the diagonal, deeper color means more right times. Off the diagonal, deeper color means more false times.
As can be seen in the figure, many of the categories are clustered on the diagonal, indicating that these categories are correctly categorized by the model. In addition, there is some misclassification of some categories, and these off-diagonal squares show misclassification by the model

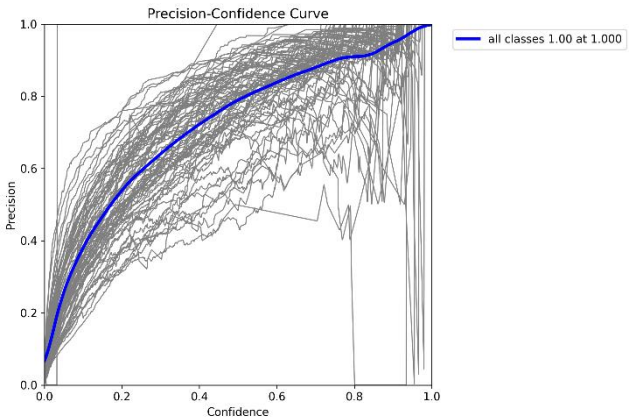

Figure 2：Precision-Confidence Curve

Figure 2 this accuracy-confidence curves show how the accuracy of the model changes for different confidence thresholds.
The confidence threshold is represented by the horizontal axis (Confidence), which has a range of 0 to 1. The more confident the model is in the accuracy of its prediction, the higher the confidence threshold.
The precision of the model is indicated by the vertical axis (Precision), which shows the percentage of positive samples that the model really predicts are positive.
Blue curve：Represents the average precision curve for all categories.
Gray curves: Show the precision curves for each category separately.
As can be seen from the figure, the accuracy gradually increases as the confidence level increases. This indicates that the model predicts more correctly at higher confidence levels. However, when the confidence level is very high, the curve may become unstable, which is due to the small number of samples at high confidence levels, resulting in large statistical fluctuations.
For example:

Figure 3: Detected Result Put on Labels

This is detected and put the labels on the Figure 3.

Figure 4: Result Put on Confidences

Just like Figure 4, in this example, put the confidences on it.
There are some accurate data:5000 images in total.Result of YOLOv8 on COCO dataset can be seen in Table 1.

Table 1: Result of YOLOv8 on COCO Dataset

Class	Images	Instance	Box (P	R	mAP50	Map50-95)
person	2693	11004	0.744	0.657	0.729	0.498
bicycle	149	316	0.63	0.402	0.45	0.254
car	535	1932	0.63	0.52	0.544	0.355
motorcycle	159	371	0.713	0.563	0.639	0.401
Airplane	97	143	0.707	0.727	0.788	0.602

Box(P): the precision of the detection box (Precision). The higher the precision, the higher the percentage of correct prediction of the detection box.
R: Recall. The higher the recall, the more positive samples are detected by the model.
mAP50: Mean Average Precision (Mean Average Precision) at an IoU threshold of 0.5.
mAP50-95: Mean Average Precision at IoU thresholds from 0.5 to 0.95 (in steps of 0.05).
The Person category has the best detection with high precision, recall and average precision.
The Airplane category is also fairly well detected, with excellent performance on mAP0.5 and mAP0.5-0.95.
The Bicycle category is detected poorly, with low recall and average precision, indicating the model's lack of ability to recognize this category.
The Car and Motorcycle categories have a moderate detection effect and a more balanced performance on all the metrics.

Overall, the model's detection performance on different categories varies, which may be related to the category distribution of the dataset and the complexity of the categories. Based on the evaluation results, further model optimization and data enhancement can be carried out for the poorly performing categories.

6 CONCLUSION

This paper focuses on observing the development history of the YOLO algorithm, the COCO dataset was introduced and examines the results of the YOLOv8 algorithm on the COCO dataset. YOLOv8 performs well on the COCO dataset but has its shortcomings. It also looks forward to the future development trends and improvement directions of the YOLO algorithm, where streamlining becomes increasingly important. It is hoped that the YOLO algorithm will continue to make breakthroughs and be increasingly applied in everyday life.

7 FUTURE EXPECTATIONS

Streamlining is one important thing. Future trends including life-wise are going to be faster, cleaner and more efficient. Maybe people will increase robustness and adaptability. And focus on enhancing detection accuracy.

COMPETING INTERESTS

The authors have no relevant financial or non-financial interests to disclose.

REFERENCES

[1] Girshick R. Fast R-CNN. Proceedings of the IEEE International Conference on Computer Vision, 2015, 1440-1448.
[2] Liu W,Anguelov D, Erhan D, et al. SSD: Single Shot MultiBox Detector. Proceedings of the European Conference on Computer Vision, 2016, 21-37.
[3] Redmon J, Divvala S, Girshick R, et al. You Only Look Once: Unified, Real-Time Object Detection. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, 779-788.
[4] Redmon J, Divvala S, Girshick R, et al. You only look once: Unified, real-time object detection. arXiv preprint arXiv:1506.02640, 2015.
[5] Redmon J, Farhadi A, YOLO9000: Better, Faster, Stronger. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 7263-7271.
[6] Redmon J, Farhadi A. YOLOv3: An Incremental Improvement. arXiv preprint arXiv:1804.02767, 2018.
[7] Bochkovskiy A, Wang CY, Liao HYM. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934, 2020.
[8] Jocher G, et al. YOLOv5: Open-Source Object Detection.GitHub repository. The open-source implementation of YOLOv5 provided a foundation for the development of YOLOv8, 2021.
[9] Lin TY, Maire M, Belongie S, et al. Microsoft coco: Common objects in context. in Computer Vision–ECCV 2014: 13th European Conference, 2014, 740–755.
[10] Wang CY, Bochkovskiy A, Liao HYM. Scaled-YOLOv4: Scaling Cross Stage Partial Network. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2021, 13029-13038.
[11] Jocher G, et al.YOLOv5: Open-Source Object Detection. GitHub repository.
[12] Ge Z, Liu S, Wang F, et al. YOLOX: Exceeding YOLO Series in 2021. arXiv preprint arXiv:2107.08430, 2021.
[13] Chen K, et al. Hybrid Task Cascade for Instance Segmentation. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2019, 2961-2970.
[14] Liu W, et al. SSD: Single Shot MultiBox Detector. European Conference on Computer Vision (ECCV), 2016, 21-37.
[15] Lin TY, et al. Focal Loss for Dense Object Detection. Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017, 2980-2988.
[16] Zoph B, et al. Learning Transferable Architectures for Scalable Image Recognition. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, 8697-8710.
[17] Ren S, He K, Girshick R, et al. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Advances in Neural Information Processing Systems (NeurIPS), 2015, 28, 91-99.
[18] Kumar P, kumar V. Exploring the Frontier of Object Detection: A Deep Dive into YOLOv8 and the COCO Dataset. in Proceedings of the IEEE conference on computer vision and Machine Intelligence (CVMI), 2023.