**Science on Natural Resources and Environment**
Journal homepage: tapchikhtnmt.hunre.edu.vn

# APPLICATION DEEP LEARNING YOLOv8 MODEL FOR OBJECT DETECTION

**Thi Thanh Thuy Pham*, Thi Thu Ha Le**
Hanoi University of Natural Resource and Environment, Vietnam

Received 02 November 2023; Accepted 20 December 2023

**Abstract**

*Detection of objects is a crucial aspect within the realm of computer vision, encompassing the identification and precise localization of objects within images or videos. This task holds significant importance in various applications, including but not limited to self-driving cars, robotics, and video surveillance systems. Throughout the years, numerous techniques and algorithms have been devised to detect objects within images and determine their spatial positions. The optimal performance in executing these tasks is achieved through the utilization of convolutional neural networks. Among the prominent neural networks designed for this purpose, YOLO stands out. Introduced in 2015 by Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi in their renowned research paper "You Only Look Once: Unified, Real-Time Object Detection" YOLO has become widely adopted. Since its inception, YOLO has undergone several iterations, with newer versions extending their capabilities beyond object detection. The latest release in this series is YOLOv8. In this article, the authors introduce a Deep Learning approach for object detection employing the YOLOv8 model. The testing outcomes reveal that the model attains a peak accuracy of 95 % for large and well-defined objects, whereas objects that are obscured and small in size exhibit an accuracy of 27.5 %.*

**Keywords:** Foreign Object Detection; Deep Learning; YOLOv8.

***Corresponding author. Email:** pttthuy.tdbd@hunre.edu.vn

## 1. Introduction

With the development of computer technology and artificial intelligence, image recognition technology is increasingly developed and applied. With the continuous advancement of computer technology and artificial intelligence, there has been notable progress in the development and application of image recognition technology. Recent case studies, both international and domestic, highlight this trend.

For instance, research focused on vehicle classification algorithms using computer vision underscores the importance of identifying and categorizing vehicles, particularly in urban planning and traffic management, a topic gaining increasing global attention. This article delves into the research and development of a novel algorithm designed to identify and classify vehicles in real-time video streams. The proposed algorithm harnesses a neural

network implemented with the YOLO and Sort algorithms, specifically tailored for object-tracking applications. The research involved testing live video feeds from surveillance cameras. The results of the tests showcase the proposed algorithm's stability and effectiveness, achieving a high accuracy rate of approximately 95 % for cars and over 80 % for motorcycles. These outcomes are remarkably positive when compared to alternative methods, emphasizing the algorithm's potential for practical applications in diverse scenarios [1].

In the realm of medicine, the detection of diseases through image diagnosis holds paramount significance. Therefore, the application and advancement of an artificial intelligence algorithm for detecting reflux esophagitis on endoscopic image sets play a crucial role in assessing the accuracy of AI algorithms in identifying and evaluating the severity of these lesions. The research methodology employed for this purpose is cross-sectional description. The algorithm was rigorously tested on a set of 1,000 still images with varying lighting conditions, and its performance was assessed by comparing the standard segmentation provided by experts. The evaluation of accuracy involved metrics such as sensitivity, specificity, positive predictive value, and negative predictive value. Additionally, the comparison of proportions and examination of associations were employed to scrutinize factors contributing to omission and misidentification rates. The obtained results revealed an accuracy rate of 81.7 %. Notably, the study found a correlation between the number and size of lesions and the omission rate, while the presence of accompanying damage and cleanliness were linked to the rate of misidentification. In conclusion, the YOLOv8 algorithm demonstrated commendable accuracy, indicating its potential for further development in areas such as co-checking with endoscopists during endoscopy, post-checking after endoscopy, and participation in medical training leveraging big data [2].

In the aviation industry, a deep learning solution is applied to detect and provide early warnings for Foreign Object Debris (FOD) on runways, aiming to minimize risks during aircraft takeoff and landing. The proposed solution by the authors utilizes PTZ cameras to capture wide-angle images, preprocesses the images, and then employs deep learning processing through the YOLOv4 algorithm. The test results demonstrate that the model can operate at a speed of 45 frames per second, achieving an accuracy of 98 % with an Intel Core i7 configuration and RTX 2080 GPU [3].

Some studies around the world include: Performance analysis of deep learning YOLO models for South Asian regional vehicle recognition [4], Real-time factory smoke detection based on two-stage relation-guided algorithm [5], Wildfire and smoke detection using staged YOLO model and ensemble CNN [6].

Utilizing both international and domestic research, the YOLO model facilitates rapid detection and timely response to objects within images or videos. The most recent iteration, YOLOv8, was unveiled in January 2023. Consequently, this study employs YOLOv8 to identify objects in images and assess the model's accuracy.

## 2. Description of data

### *2.1. Dataset acquisition*

The accuracy of the deep learning model was predominantly influenced by the dataset utilized during the training and validation phases. We compiled a collection of images from diverse open-access repositories, including "Open Images Dataset v7 and Extensions" [7] and Roboflow [8]. These sources provided images portraying various conditions such as different shapes, colors, sizes, and environments (both indoor and outdoor).

For real-world challenges, the database needs to be considerably larger. To effectively train a robust model, it is essential to have hundreds or even thousands of annotated images. The image dataset is illustrated in Table 1, comprising a total of 7,198 images.

***Table 1. Image sources for object detection dataset***

| Dataset | Image Size | Number of images |
|---|---|---|
| Open Images Dataset V7 and Extensions | Many different sizes 448 × 640; 660 × 371; 1024 × 576; 600 × 400 | 6,702 |
| Roboflow | 640 × 640 | 496 |
| **Total Images** | | **7,198** |

### *2.2. Data preprocessing*

On Google Drive, establish a "data" folder and generate subfolders named "train" and "validation" subsequently, distribute the collected images into these two subfolders, allocating 80 % to the "train" subfolder and 20 % to the "val" subfolder as shown in Table 2.

***Table 2. Distribution of training and validation images in fire dataset***

| Dataset name | Size | Number of images | Training set | Validation set |
|---|---|---|---|---|
| Data | 640 × 640 | 7,198 | 5,759 | 1,439 |

Then we export the dataset as YOLOv8 Pytorch format and it generates an API link which we used in our model directly.

Next, the research team enhances the image quality and utilizes Make-Sense to annotate objects in the images [9]. After adding and annotating all images, the dataset is ready. The final folder structure can look like Figure 1.

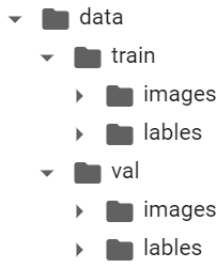


***Figure 1: Dataset structure***

## 3. YOLO models

You Only Look Once (YOLO) proposes using an end-to-end neural network that makes predictions of bounding boxes and class probabilities all at once. It differs from the approach taken by previous object detection algorithms, which repurposed classifiers to perform detection. Numerous iterations of the original YOLO model have been introduced since its initial release in 2015, each iteration building upon and enhancing its predecessor. The following timeline highlights the evolution of YOLO over the past few years.

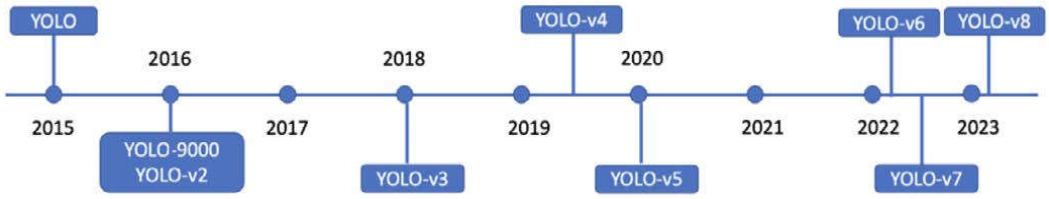


*Figure 2: YOLO timeline [10]*

The YOLO algorithm processes an input image by employing a straightforward deep convolutional neural network to identify objects within the image. The diagram below illustrates the structure of the convolutional neural network model that serves as the foundation for YOLO.

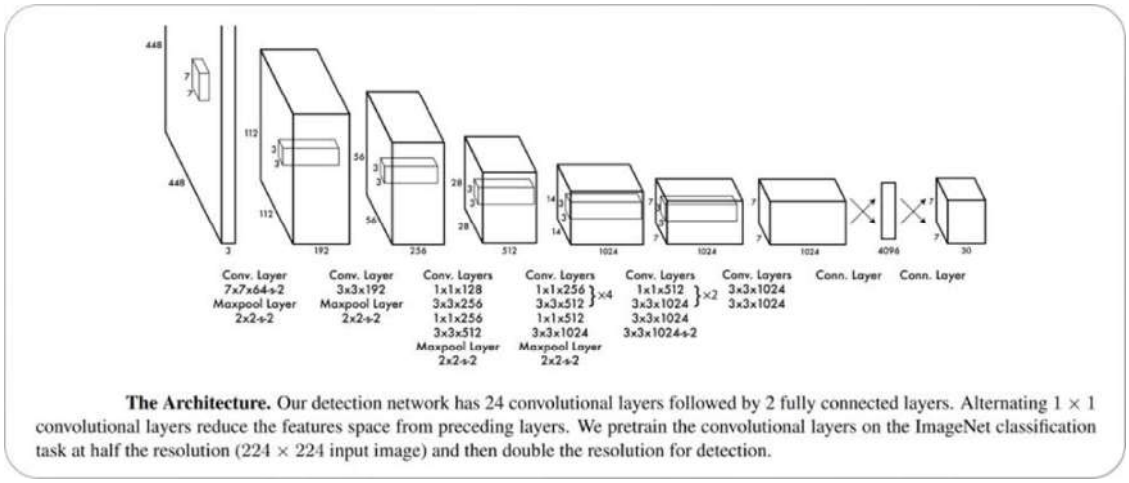


*Figure 3: CNN Network Architecture in YOLO [11]*

Ultralytics has officially announced the upcoming YOLOv8, which is anticipated to introduce enhanced features and performance compared to its earlier versions. YOLOv8 introduces a novel API designed to streamline both training and inference processes on both CPU and GPU devices, while also ensuring compatibility with previous YOLO versions. Although a comprehensive scientific paper detailing the model's architecture and performance is still in development, the release is eagerly awaited by developers.

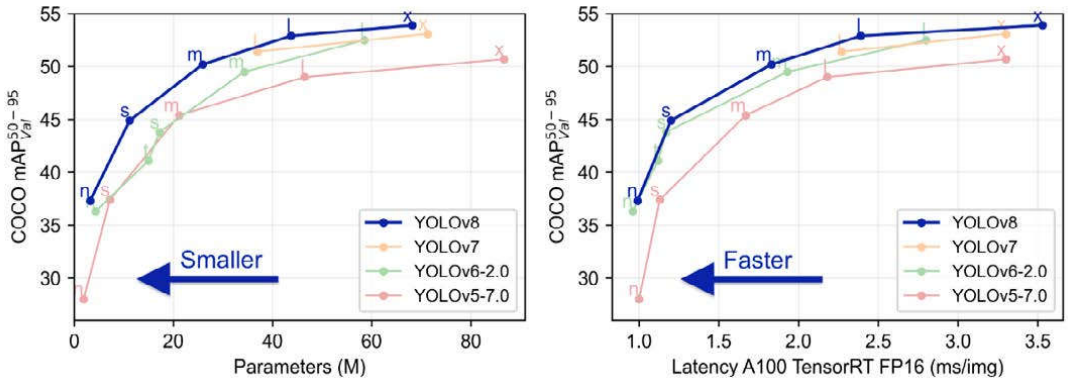


*Figure 4: Compare different YOLO versions [12]*

As we can observe from the chart, YOLOv8 has more parameters than its predecessors like YOLOv5 but fewer parameters than YOLOv6. It provides approximately a 33 % increase in mean average precision (mAP) for models of size "n" and generally higher mAP.

From the second chart, it is evident that YOLOv8 has faster inference times compared to all other YOLO versions. In YOLOv8, we encounter various model sizes such as yolov8-nano, -small, -medium, -large, and -extra-large.

| Model | size (pixels) | $mAP^{val}$ 50-95 | Speed CPU ONNX (ms) | Speed A100 TensorRT (ms) | params (M) | FLOPs (B) |
|---|---|---|---|---|---|---|
| YOLOv8n | 640 | 37.3 | 80.4 | 0.99 | 3.2 | 8.7 |
| YOLOv8s | 640 | 44.9 | 128.4 | 1.20 | 11.2 | 28.6 |
| YOLOv8m | 640 | 50.2 | 234.7 | 1.83 | 25.9 | 78.9 |
| YOLOv8l | 640 | 52.9 | 375.2 | 2.39 | 43.7 | 165.2 |
| YOLOv8x | 640 | 53.9 | 479.1 | 3.53 | 68.2 | 257.8 |

***Figure 5: Size comparison of YOLO models [12]***

The model size correlates linearly with mean average precision (mAP) and inversely with inference time. Larger models take more time for accurate object detection with higher mAP. Smaller models have faster inference times but relatively lower mAP. Larger models perform better when there is limited data. Smaller models are more efficient in scenarios with limited space (edge situations).

This research aims to enhance the object detection capabilities of images using the YOLOv8 architecture. Our system, illustrated in Figure 6, is meticulously crafted from data collection to model construction, as explained in detail below. Following data collection, we conducted preprocessing, dividing the data into training (80 %) and validation (20 %) sets. These images were then fed into the YOLOv8 model for simultaneous training and validation, utilizing the YOLOv8 model variant.

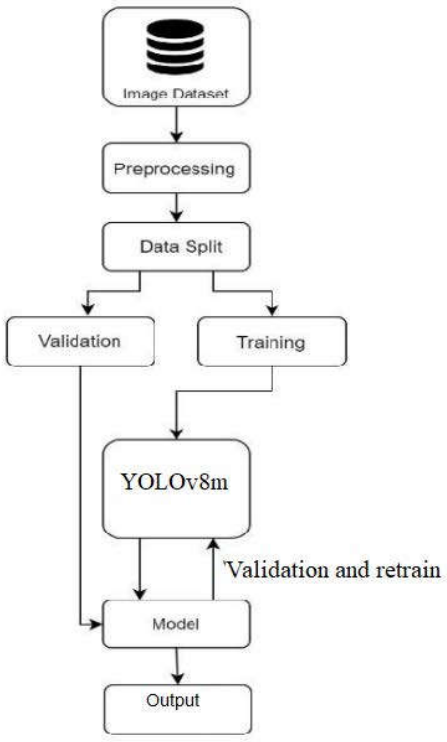


***Figure 6: Implementation process***

Each training cycle consists of two phases: a training phase and a validation phase:

* Trainning phase

During the training phase, the training method does the following:

+ Extracts the random batch of images from the training dataset (the number of images in the batch can be specified using the batch option).

+ Passes these images through the model and receives the resulting bounding boxes of all detected objects and their classes.

+ Passes the result to the loss function that's used to compare the received output with correct result from annotation files for these images. The loss function calculates the amount of error.

+ The result of the loss function is passed to the optimizer to adjust the model weights based on the amount of error in the correct direction. This reduces the errors in the next cycle. By default, the SGD (Stochastic Gradient Descent) optimizer is used.

* Validation phase

During the validation phase, train does the following:

+ Extracts the images from the validation dataset.

+ Passes them through the model and receives the detected bounding boxes for these images.

+ Compares the received result with true values for these images from annotation text files.

+ Calculates the precision of the model based on the difference between actual and expected results.

**4. Results**

The progress and results of each phase for each epoch are displayed on the screen. This way you can see how the model learns and improves from epoch to epoch. When you run the train code, you will see a similar output to the following during the training loop:

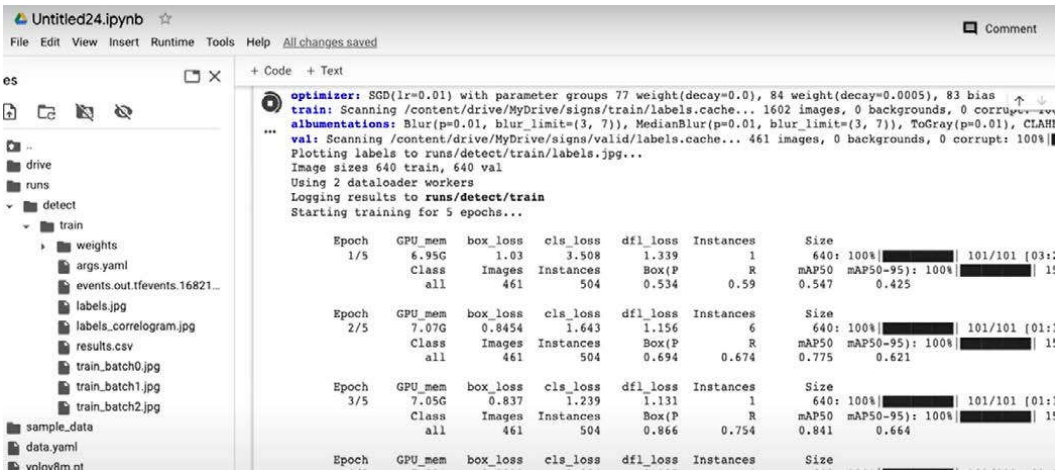


***Figure 7: Train the YOLOv8 model***

After training, the training results will be saved in the runs/detect/train folder. The results of each epoch's training are saved in the results.csv file and the weights of the Yolov8 model will be saved in the weights folder.

The number of objects detected in an image includes: 6 persons, 3 cars, 5 motorcycles, 1 backpack.

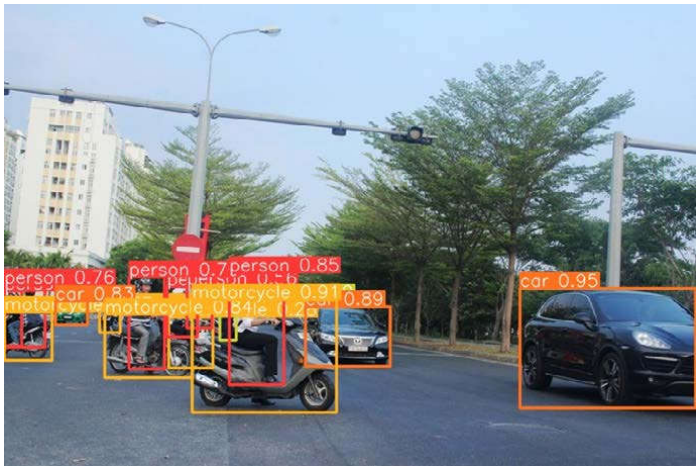


*Figure 8: Detected objects in the image*

```
Object type: car                    Object type: car                    Object type: person
Coordinate: [596, 328, 799, 466]    Coordinate: [59, 343, 96, 369]      Coordinate: [188, 326, 223, 383]
Probability: 0.94977343082428       Probability: 0.828676044940948      Probability: 0.534940421581268
--                                  --                                  --
Object type: motorcycle             Object type: motorcycle             Object type: backpack
Coordinate: [217, 341, 384, 470]    Coordinate: [0, 354, 56, 410]       Coordinate: [249, 340, 269, 388]
Probability: 0.909209012985229      Probability: 0.774635314941406      Probability: 0.485219240188599
--                                  --                                  --
Object type: car                    Object type: person                 Object type: motorcycle
Coordinate: [348, 349, 446, 417]    Coordinate: [0, 324, 48, 395]       Coordinate: [111, 351, 135, 378]
Probability: 0.893734991550446      Probability: 0.759477913379669      Probability: 0.366378962993622
--                                  --                                  --
Object type: person                 Object type: person                 Object type: person
Coordinate: [261, 309, 324, 438]    Coordinate: [145, 314, 185, 421]    Coordinate: [0, 331, 20, 395]
Probability: 0.854530692100525      Probability: 0.704396784305573      Probability: 0.353683233261108
--                                  --                                  --
Object type: motorcycle             Object type: person                 Object type: motorcycle
Coordinate: [116, 358, 218, 430]    Coordinate: [211, 324, 240, 373]    Coordinate: [191, 360, 241, 420]
Probability: 0.837235391139984      Probability: 0.564104795455933      Probability: 0.275619566440582
--                                  --                                  --
```

*Figure 9: Information of the detected object*


*Figure 10: Detection results for different types of objects*

## 5. Conclusion

Effective implementation of convolutional neural networks can significantly improve object detection performance. The YOLOv8 model is capable of detecting many different objects in a quick time. However, achieving high-accuracy results requires the labeling of objects to be accurate and complete. Besides, the size and clarity of the object in the image also greatly affect the results of object detection and classification. In the field of resource and environmental management, YOLOv8 can be employed to detect phenomena such as forest fires, oil spills in the sea, or landslides. This issue will be addressed in upcoming research conducted by the study group.

## REFERENCES


[1]. Nguyen Manh Cuong, Vu Van Ruc, Doan Ngoc Au, Do Cong Danh, Hoang Anh Sang, Vu Tam Long (2021). *Research on vehicle classification algorithms based on computer vision.* Research Gate.

[2]. Bui Tri Thuc, Lam Ngoc Hoa, Vu Thi Ly, Dao Viet Hang (2023). *Application of artificial intelligence in detecting reflux esophagitis on endoscopy images.* Journal of Medical Research, Hanoi Medical University 170 (9), 114-151.

[3]. Minar Mahmud Rafi, Siddharth Chakma, Asif Mahmud, Raj Xavier Rozario, Rukon Uddin Munna, Md. Abrar Abedin Wohra, Rakibul Haque Joy, Khan Raqib Mahmud, Bijan Paul (2022). *Performance analysis of deep learning YOLO models for South Asian regional vehicle recognition.* (IJACSA) International Journal of Advanced Computer Science and Applications, Vol. 13, No. 9, 864-873.

[5]. Zhenyu Wang, Duokun Yin, and Senrong Ji (2022). *Real-time factory smoke detection based on two-stage relation-guided algorithm*. Scientific Reports 12:1786. Nature portolio. https://doi.org/10.1038/s41598-022-05523-1.

[6]. Chayma Bahhar, Amel Ksibi, Manel Ayadi, Mona M. Jamjoom, Zahid Ullah, Ben Othman Soufiene and Hedi Sakli (2023). *Wildfire and smoke detection using staged YOLO model and ensemble CNN*. Electronics 2023, 12, 228. https://doi.org/10.3390/electronics12010228.

[7]. Open Images Dataset v7 and Extensions (2023). [online] Available at: [Accessed 1 December 2023]. Available at: https://storage.googleapis.com/openimages/web/index.html.

[8]. Roboflow (2023). [online] [Accessed 1 December 2023]. Available at: https://universe.roboflow.com/search?q=model%253Ayolov8&p=3.

[9]. Makesence (2023). [online] Available at: https://www.makesense.ai (Accessed 1 December 2023).

[10]. Research Gate (2023). *Timeline of You Only Look Once (YOLO) variants*.[online]. Available at: https://www.researchgate.net/figure/Timeline-of-You-Only-Look-Once-YOLO-variants_fig1_369379818.

[11]. Tra Van Dong, Nguyen Thu Nguyet Minh and Huynh Chi Nhan (2020). *Compare SSD algorithm and YOLO in object detection*. Van Lang University Journal of Scientific. Code TCKH22-10-2020.

[12]. Ultralytics YOLOv8 (2023). [online] [Accessed 2 December 2023]. Available at: https://github.com/ultralytics/ultralytics.