UDC: 004.932 https://doi.org/10.15407/jai2026.01.049

B. Podoliak[1], T. Filimonova[2]
[1, 2] State University of Trade and Economics, Ukraine
19, Kyoto st., Kyiv, 02156
[1]b.podoliak_fit_4m_24_m_z@knute.edu.ua
[2]t.filimonova@knute.edu.ua
[1]https://orcid.org/0009-0002-6667-3278
[2]https://orcid.org/0000-0001-9467-0141

RESEARCH AND OPTIMIZATION OF METHODS FOR DETECTING OBJECTS IN IMAGES

Abstract. This work has conducted research and optimization of object detection methods in images using modern deep learning approaches. The work has conducted a theoretical analysis of the object detection problem, considered the role of computer vision in the modern information environment, and analyzed domestic and foreign scientific and technical sources. An analysis of existing neural network architectures, in particular YOLOv8, Faster R-CNN, and DETR, was conducted, with the determination of their advantages, disadvantages, and areas of effective application. The selected models were optimized by selecting hyperparameters, improving learning processes, and increasing the balance between accuracy and speed. Experimental implementation and comparison of models were conducted, which allowed assessing the impact of the applied optimization methods on the efficiency of detection systems.

Keywords: object detection methods, neural networks, computer vision, optimization, YOLOv8, Faster R-CNN, DETR.

Introduction

The ability to automatically, quickly and efficiently detect objects in images is one of the key tasks of modern computer vision and artificial intelligence. As the information environment develops rapidly, more and more companies and organizations in various fields - from security systems and autonomous transport to medicine, construction and industry - are implementing intelligent image analysis technologies. In this context, object detection has become particularly relevant, as it allows you to automate the processes of detection, localization and classification of objects, increasing the accuracy of systems and reducing time and resource consumption. The main tool for solving object detection problems is deep neural networks, which are represented by different types of architectures. In modern research, three main approaches to building detection models are distinguished: two-stage, one-stage and transformer. Two-stage models, such as Faster R-CNN, first generate regions of interest, and then perform their refinement and classification, which provides high accuracy, but requires significant computing resources. Single-stage models, in particular the YOLO family, perform detection in a single pass, which makes them much faster and suitable for real-time operation, although sometimes this comes at the expense of accuracy. A separate class is formed by transformer models, such as DETR, which combine convolutional networks for feature extraction with a self-attention mechanism for global analysis of relationships between objects. Despite the high efficiency of modern models, they have a number of limitations, including significant computational complexity, the need for large amounts of memory, and high power consumption, which is especially critical for real-time applications or on mobile and embedded devices. This necessitates not only the development of new architectures, but also the constant optimization of existing models. In this regard, the paper considers various methods for increasing efficiency, including improving learning processes, transfer learning, knowledge distillation, pruning, quantization, convolution factorization, and inference optimization. Such approaches allow achieving a better balance between the accuracy and performance of object detection models.

Analysis of recent research and publications

Deep learning models for object detection are one of the most actively researched areas in computer vision. They are widely used in various video surveillance

systems, autonomous transport, medical diagnostics, industrial automation and other areas. Recent research focuses not only on increasing the accuracy of object detection, but also on reducing the computational complexity of models, optimizing them for real-time operation and adapting to various operating conditions. Below is an overview of key works in this field. In [1], the general principles of building object detection systems based on deep neural networks are presented, their main components, such as feature extractors, region suggestion mechanisms and classification heads, are analyzed. The authors also consider the evolution of detection approaches - from classical computer vision methods to modern CNN and transformer models. A detailed analysis of two-stage models, in particular Faster R-CNN, is given in [2], where their advantages in terms of object localization accuracy are shown, as well as the limitations associated with high computational costs. The study demonstrates that such models are effective for tasks with high accuracy requirements, but are less suitable for real-time. In the works devoted to one-stage detectors, special attention is paid to the YOLO family of models. In [3], the basic principles of the YOLO architecture and its performance in object detection tasks are considered. Further development of this approach is presented in [4] and [5], where YOLOv8, its new architecture, improved anchor system and optimized training process are described, which allowed to achieve a better balance between speed and accuracy. A separate direction of research is the application of transformer models for object detection. In [6], the DETR architecture is considered, which replaces traditional methods of generating regions of interest with a self-attention mechanism. The authors show that this approach simplifies the model structure, but requires longer training and significant computational resources. Considerable attention in modern research is paid to methods for optimizing detection models. In [7] and [8], quantization and pruning techniques are analyzed, which allow reducing the size of models and speeding up their operation without significant loss of accuracy. Other works, in particular [9], focus on optimizing the inference process and using graph optimizers (e.g., ONNX Runtime) to accelerate the operation of models in real time.

Main Material

As a result of the implementation and training of the object detection models YOLOv8, Faster R-CNN and DETR based on modern deep neural architectures, satisfactory results of object detection in images were obtained. At the same time, in order to improve the quality and speed of the detection systems, their optimization was carried out. In order to determine which model is best suited for solving the task, three different approaches were implemented and studied: a one-stage model, a two-stage model and a transformer model. The training dataset used contained images with marked objects, which allowed the application of supervised learning methods. Information about the classes and coordinates of the bounding boxes was transferred to the model during training, which ensured the correct formation of predictions. The first stage of model implementation was the connection of the necessary libraries, in particular PyTorch and Torchvision, as well as the normalization of input data. For the Faster R-CNN model, a pre-trained ResNet-50 architecture using the Feature Pyramid Network (FPN) was used, which allows for effective work with multi-level image features. After the image passes through the convolutional network, a feature map is formed, on the basis of which the region of interest generation module forms candidate regions, which are further refined and classified. Such an architecture provides high accuracy of object localization, although it requires significant computational resources (Fig. 1).

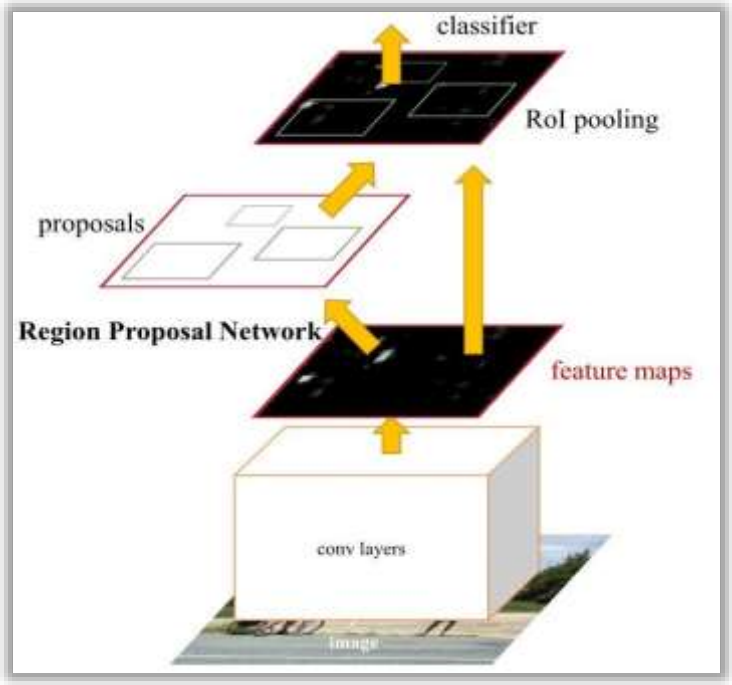

Fig. 1. Architecture of the Faster R-CNN model

The YOLOv8 model uses a modern single-stage architecture that performs object detection in a single pass of the neural network. The input image passes through a sequence of convolutional layers, where a multi-level representation of features is formed, after which the coordinates of the bounding boxes and class probabilities are directly predicted. During the training process, hyperparameters were adapted, which allowed achieving a better balance between the speed and accuracy of the model (Fig. 2).

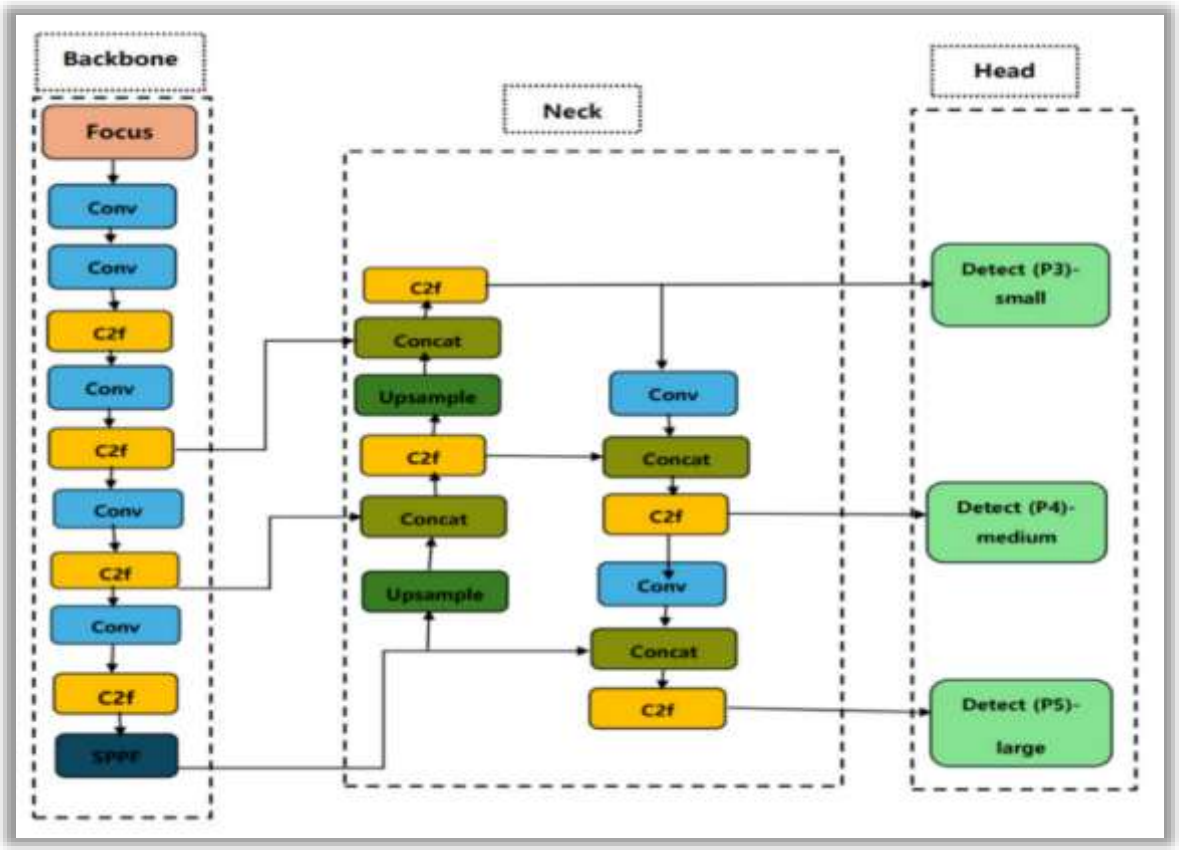

Fig. 2. Architecture of the YOLOv8 model

The DETR model, based on the transformer self-attention mechanism, was implemented to explore an alternative approach to detection without using traditional regions of interest. After feature extraction using the convolutional part, they are transformed into a sequence of vectors that are fed to the input of the transformer encoder-decoder. This approach allows for global dependencies between objects in the image, but requires a larger number of training epochs and significant computational resources (Fig. 3).

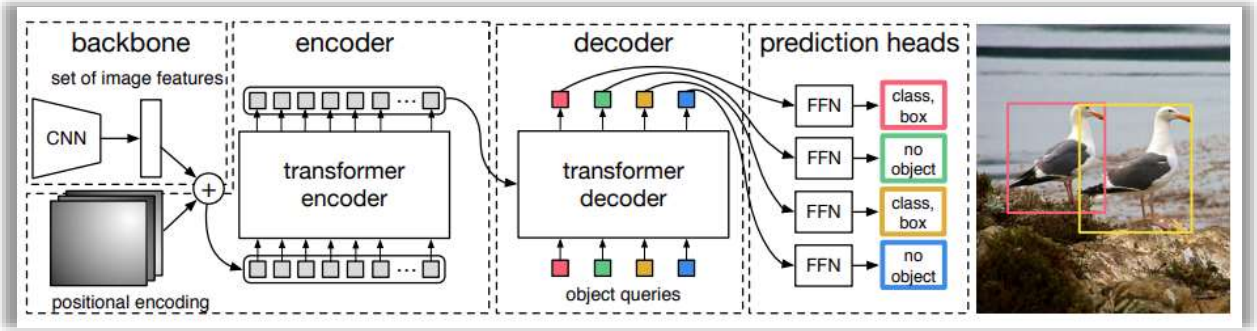

Fig. 3. Architecture of the DETR model

After the models were implemented, the size of the training and validation samples was determined to evaluate their overall performance. During the training process, the loss function and accuracy indicators were monitored. Finally, the detection results were visualized on the test images (Fig. 4).

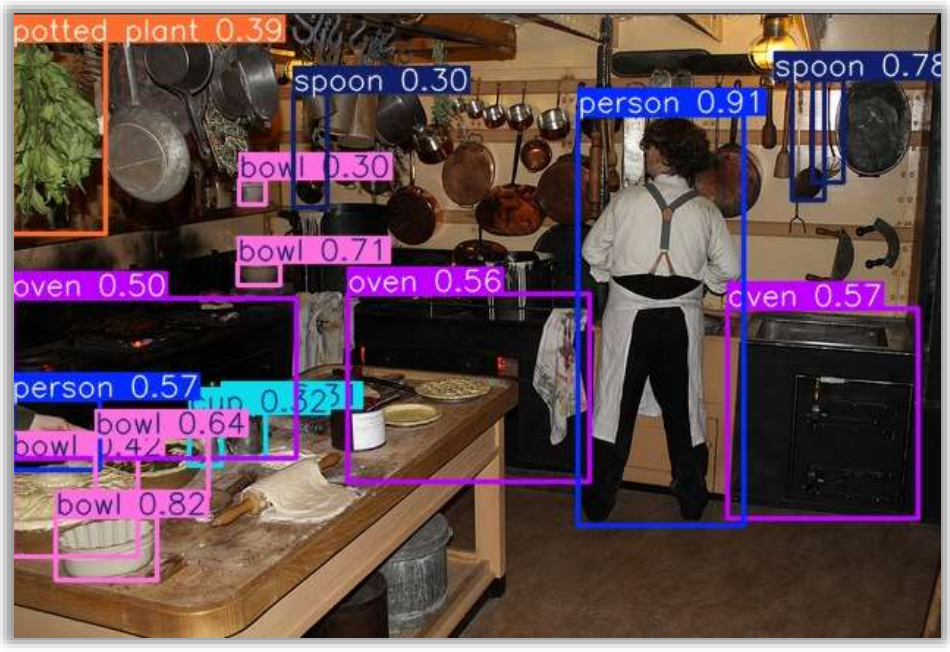

Fig. 4. YOLOv8 model detection results

The optimization process of object detection models was aimed not only at increasing the recognition accuracy, but also at improving the efficiency of using computing resources during training and inference. In practical applications, the balance between speed, model size, and detection quality is of particular importance. Therefore, optimization was considered as a complex process that includes tuning training parameters, improving convergence stability, and adapting models to the characteristics of the dataset. The first

model to be optimized was the YOLOv8s model, which is one of the most compact and fast in its line. Despite the initially high performance, additional tuning of the training process allowed us to improve convergence stability and detection quality (Fig. 5). During the optimization, the training parameters were adjusted so that the model received more stable weight updates and better generalized information from the training data. As a result, it was possible to achieve a noticeable improvement in quality metrics, reduced loss values, and more confident predictions on test images. The optimized model began to more accurately determine the boundaries of objects and reduced the number of false positives, which is clearly confirmed by the detection results (Fig. 6).

```
results = model.train(
    data='coco.yaml',
    epochs=20,
    imgsz=640,
    batch=32,
    lr0=0.002,
    optimizer='SGD',
    momentum=0.937,
    weight_decay=0.0005,
    name='yolov8s_coco_optimized'
)
```

Fig. 5. Optimization of the Yolov8 model

Fig. 6. Detection results of the Yolov8 model after optimization

The next stage was the optimization of the Faster R-CNN model, which has a more complex two-stage architecture. For this model, the balance of the learning process between different components, such as the generation of regions of interest and subsequent object classification, is particularly important. The optimization was aimed at increasing the stability of convergence and improving the accuracy of object localization, especially in complex scenes (Fig. 7). As a result, it was possible to achieve a more coordinated operation of the various components of the model, which was

manifested in the reduction of losses and the improvement of the final quality indicators. Comparison of the results before and after optimization showed a clearer separation of closely located objects and a reduction in the number of redundant frames (Fig. 8).

```
optimizer = torch.optim.SGD(
    params, lr=0.0001,
    momentum=0.95,
    weight_decay=1e-5
)
```

Fig. 7. Changing the optimizer hyperparameters

Fig. 8. Detection results of the Faster R-CNN model after optimization

After that, the DETR model was optimized, which combines a convolutional network for feature extraction and a transformer mechanism for global scene analysis. Due to its more complex architecture, this model demonstrates slower and less stable convergence, which requires more careful tuning of the training process. The improvements were aimed at improving stability, accelerating convergence, and reducing fluctuations in the loss function values (Fig. 9). Additional training stabilization measures made it possible to make the weight update process more controllable and reliable. As a result, the optimized model demonstrated a smoother and more consistent decrease in all components of the loss function, which indicates better consistency between object classification and regression of their boundaries. The detection quality on test images also confirmed the feasibility of the improvements (Fig. 10).

```
param_dicts = [
    {"params": [p for n, p in model.named_parameters() if "backbone" not in n and p.requires_grad]},
    {"params": [p for n, p in model.named_parameters() if "backbone" in n and p.requires_grad], "lr": 1e-5},
]

optimizer = torch.optim.AdamW(param_dicts, lr=1e-4, weight_decay=1e-4)
scheduler = torch.optim.lr_scheduler.CosineAnnealingLR(optimizer, T_max=15, eta_min=1e-6)
scaler = torch.cuda.amp.GradScaler()
```

Fig. 9. DETR model optimizer settings

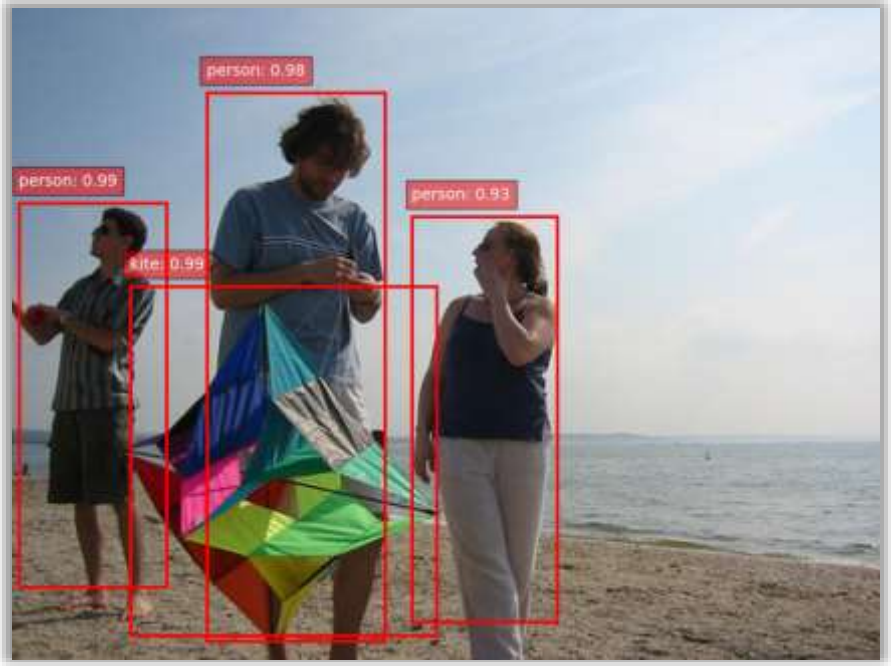

Fig. 10. Detection results of the DETR model after optimization

For a generalized assessment of the learning process and the stability of convergence of the implemented models, graphs of the change in the value of the total loss function depending on the number of training epochs were constructed (Fig. 11). The graph presents the dynamics of loss reduction for the YOLOv8s, Faster R-CNN, and DETR models. From the figure, it can be seen that all three models demonstrate a stable tendency to reduce the value of the loss function during the learning process, which indicates the correct operation of the optimization algorithms and a gradual improvement in the quality of approximation of the training data. The Faster R-CNN model is characterized by the lowest loss values throughout the entire learning process, which is explained by its two-stage architecture and a more accurate object localization procedure. At the same time, the YOLOv8s and DETR models have higher initial loss values, but demonstrate stable and monotonic convergence. In particular, for the YOLOv8s model, a rapid decrease in the value of the loss function is observed in the first epochs of training with a subsequent gradual slowdown in the convergence rate, which is typical for single-stage detectors. The DETR model learns more slowly, but its loss curve is smooth and stable, which confirms the correctness of the tuning of the transformer architecture optimization process. In general, the obtained learning curves confirm that all three models reach a stable state and do not show signs of drift or overtraining, and the applied optimization methods provide reliable and predictable convergence of the learning process.

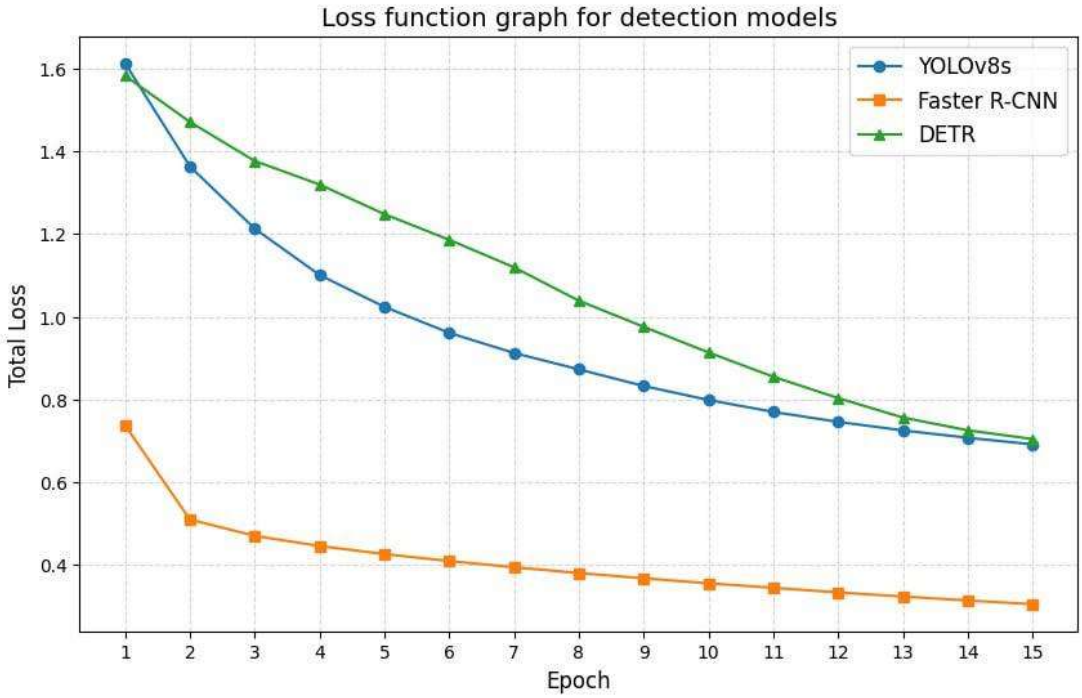

Fig. 11. Loss function graph for detection models

Conclusion

As a result of the research, modern object detection models YOLOv8, Faster R-CNN and DETR were implemented and analyzed, representing different approaches to building computer vision systems. For each of the models, training was performed on a labeled dataset, the convergence process was investigated, and detection results were obtained on test images. Comparative analysis showed that Faster R-CNN provides the highest accuracy of object localization, but requires significant computing resources, YOLOv8 demonstrates the best ratio between speed and detection quality, which makes it most suitable for real-time application, and DETR is distinguished by the stability of results and the ability to take into account the global context of the scene, although it requires more training time. An important part of the research was the optimization of each of the models, taking into account their architectural features. As a result of tuning the training process and applying optimization methods, it was possible to increase the stability of convergence, reduce computational costs, and improve the final detection quality indicators. Optimized versions of the models demonstrated more accurate object boundary detection, reduced false positives, and better generalization. The results obtained confirm that the use of modern optimization methods is a necessary step in the practical implementation of object detection systems. The combination of high accuracy and efficient speed allows such models to be used in a wide range of applications, including video surveillance systems, autonomous transport, industrial automation, and visual information analysis. Thus, the study confirms the feasibility of using different object detection architectures depending on the performance and accuracy requirements, as well as the effectiveness of the applied approaches to their optimization.

References

1. Singh, P. (n.d.). Evolution of Object Detection: RCNN, Fast RCNN, and Faster RCNN. Medium. https://medium.com/@2003priyanshusingh/evolution-of-object-detection-rcnn-fast-rcnn-and-faster-rcnn-90cc872e6dae

2. S. Ren, K. He, R. Girshick and J. Sun, "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 6, pp. 1137-1149, 1 June 2017, doi: 10.1109/TPAMI.2016.2577031.

3. J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer

Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 2016, pp. 779-788,
doi: 10.1109/CVPR.2016.91.

4. Yaseen, M. (2024). What is YOLOv9: An in-depth exploration of the internal features of the next-generation object detector. arXiv preprint arXiv:2409.07813.

5. Ultralytics. (n.d.). Explore Ultralytics YOLOv8. Ultralytics YOLO Docs.
https://docs.ultralytics.com/models/yolov8/

6. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-end object detection with transformers. In European Conference on Computer Vision (ECCV) (pp. 213–229). Cham: Springer.
https://doi.org/10.1007/978-3-030-58452-8_13

7. Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and Training of Neural Networks for Efficient Integer-Arithmetic-Only Inference. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (pp. 2704–2713). https://doi.org/10.1109/CVPR.2018.00286

8. Han, S., Mao, H., & Dally, W. J. (2015). Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. arXiv preprint arXiv:1510.00149.

9. Microsoft. (n.d.). Graph Optimizations in ONNX Runtime. ONNX Runtime Documentation.
https://onnxruntime.ai/docs/performance/model-optimizations/graph-optimizations.html

The article has been sent to the editors 27.01.26.
After processing 14.02.26.
Submitted for printing 30.03.26.