International Journal of Applied Smart Interdisciplinary Technologies (IJASIT)

Home Page: https://ijasit.org

A Comprehensive Review of YOLO Object Detection Models: Evolution, Performance, and Future Directions

Yashwardhan[1], **Ayush Saxena**[2]

[1 2] School of Computer Science and Engineering, Galgotias University, Greater Noida, Uttar Pradesh-201310

*** Corresponding author.**
E-mail: ayush@gmail.com

Article History:
Received Date: 23 April 2026
Revised Date: 20 May 2026
Accepted Date: 10 June 2026
Published Date: 29 June 2026

Keywords:

YOLO; Object Detection; Real-Time Detection; Deep Learning; CNN; Transformer Models; Computer Vision; YOLOv1–YOLOv11; mAP, FPS.

ABSTRACT

Abstract: YOLO, over the past few years, has emerged as the first choice for real-time object detection techniques, affecting real-world applications and participating in the realm of academia for research. The review paper gives a perspective regarding the evolution, which has been witnessed in the YOLO models, starting right from the first YOLOv1, which had a simple yet effective CNN-based architecture, till the latest YOLOv7-EDGE, incorporating transformer-based architectures. The paper also delves into the updating in the architecture, the advancement in the backbones, the measurement of performance parameters such as the Average precision(mAP), the FPS, Latency, FLOPS, and the number of parameters. The paper also includes the training parameters such as the preset setting for the data, GPU utilization, and training time. It also compiles real-world applications across fields such as health care, autonomous vehicles, intelligent surveillance, robotics, and many more, based on implementations involving various versions of YOLO. In this paper, further aspects will be discussed with regards to the remaining difficulties despite the vast progress, which include bias within the dataset, computation cost, and generalization within unseen environments. By compiling these findings, this paper aims to guide researchers and developers in selecting and applying the most suitable YOLO version.

1. Introduction

The first sentence should start here [1]. Should have "6pt" spacing after section header. The indent of the first line of paragraph should be 0.25cm. Content in body paragraph should be written with the Font style: Palatino Linotype; Font size: 10; Paragraph: Justify; Line spacing: 1.0. **For example:** The last few decades have witnessed vast research on new types of heat transfer fluids, namely nanofluids. Object detection is a core functionality in computer vision that involves not only identifying the presence of objects in an image but also locating them using bounding boxes. Its importance spans across domains, from face recognition and retail analytics to environment monitoring, the medical domain and robotics. Before the popularity of deep learning-based detectors, object detection relied heavily on handcrafted features and classic machine learning techniques. However, this scenario began to change with the introduction of region-based convolutional neural networks (R-CNN) and its successors: R-CNN was developed in 2014. R- CNN extracted region proposals with selective search and then classified the region proposals using a CNN. This improved accuracy but was still

expensive because of repeating the CNN. Faster R-CNN was where the Region Proposal Network (RPN) was introduced, and it produced region proposals based on convolutions. The RPN produced a significant reduction in proposal runtime. SSD (Single Shot Multibox Detector): This detector was a one-shot detector that removed the requirement for region proposal networks and provided results in real-time with sufficient accuracy. Though these models brought improvements in precision, they struggled to keep pace with the required high processing speeds, especially in domains including self-driven cars, CCTV cameras, or robots, where halting would cause disastrous mistakes. As a matter of fact, this created a requirement for fast single-stage detectors.

The problem with object detection, solved by YOLO, is that it basically solves the same problem as bounding boxes and classes. This simplifies object detection enough that it can be done in real-time without sacrificing much accuracy. YOLO, which stands for "You Only Look Once," was invented by Joseph Redmon, et al., in the year 2016. This technology removes the complexity of object detection in a computer program by treating it as a single task only. Object detection at that time was a process of several tasks, but it only employs a single convolutional neural network for the following reasons: Conventional approaches, for example, R-CNN, Fast R-CNN, or Faster R-CNN, perform an object detection process in a two-step procedure. First, there is the use of a Region Proposal Network (RPN); its task is to find the possible regions where the object can be detected in the image. Next, there is the process of examining the regions for object detection which are already present in the particular region. This process can be accurate, but the time consumed is quite long since there are complex operations involved in the detection of regions and the alteration of the regions. YOLO goes a notch higher by simply concentrating on object detection without having to locate the position of those objects. YOLO only demands one single step in detecting objects as well as identifying their type. YOLO starts by dividing an image into a grid. Each cell in the grid attempts to predict what can be in that cell concerning objects as well as their positions. The process takes less time compared to other algorithms because it does not undergo Region Proposal. As a matter of fact, YOLO operates faster as it requires less computation.

The aim of the paper is to make a general analysis and discussion on the history and development of the YOLO object detection algorithm, starting from the first version up to the latest version, as well as the latest development of the transformer model, which is YOLO v11 and YOLO v12. This year, YOLO has been experimenting with various degrees of structure in an attempt to increase the speed of their detection processes.

The relevance of critical analysis in this case involves being aware and recognizing tradeoffs that exist in this context between the accuracy that the model achieves and the simplicity in which the process of execution occurs in terms of performance metrics that entail Average precision (mAP), Frames Per Second (FPS), latency, size of model parameters, and computational complexity. Additionally, this paper aims to highlight the real-world use cases and domain-specific use of different YOLO models, showing the adaptation of YOLO models across various domains like healthcare, robotics, autonomous navigation, surveillance etc. Add-on, this paper also provides some focus on emerging YOLOv11 and YOLOv12 Models, which is with transformer integration, with multi-modal capabilities, and cross-domain uses. This review intends to give researchers, developers and practitioners clarity in selecting, deploying or modifying the right YOLO version for their needs.

2. Evolution of YOLO models

The "You Only Look Once" (YOLO) object detection model has undergone major evolution since its first version launched in 2015. With each version, architecture changes, speed optimisation, and accuracy have increased significantly.

The YOLO model has revolutionised real- time object detection, balancing efficiency and accuracy, which makes it widely applicable in autonomous driving, surveillance, medical imaging, and robotics. This section enlightens the key improvements in the evolution of YOLO models from YOLOv1 to the latest YOLOv11.

2.1 YOLOv1

The first YOLO model was developed by Joseph Redmon in 2015, marking a shift from traditional region-based object detection methods to a single-shot detection approach. He framed object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. It looks at the whole image at test time, so its predictions are informed by global context in the image. Unlike region-based convolutional neural networks (R-CNN), which require multiple passes for object localisation.

YOLOv1 divided the image into an SxS grid and predicted bounding boxes and class probabilities directly in a single forward pass.

Key features:

- Single-pass object detection, achieving up to 45 FPS.
- Global spatial understanding due to end-to-end training.
- Limitations in detecting small objects due to coarse grid division.

2.2 YOLOv2

YOLOv2 is also known as "YOLO9000". It was introduced in 2016 which brought significant improvements in both speed and accuracy. It added batch normalisation, anchor boxes, and a more advanced feature extractor called Darknet-19. DarkNet-19 is a convolutional neural network that is 19 layers deep. You can load a pre-trained version of the network trained on more than a million images from the ImageNet database.

Key features:

- Use of batch normalisation for stable training and faster convergence.
- Anchor boxes for better bounding box predictions.
- Higher resolution input images for improved detection accuracy.
- Achieved 76.8 mAP on PASCAL VOC and 67.8 mAP on COCO datasets.

2. 3 YOLOv3

Yolov3 was released in 2018, introduced multi-scale feature fusion by adopting a feature pyramid network (FPN) structure. A Feature Pyramid Network, or FPN, is a feature extractor that takes a single-scale image of an arbitrary size as input, and outputs proportionally sized feature maps at multiple levels, in a fully convolutional fashion. This process is independent of the backbone convolutional architectures. It replaced Darknet-19 with Darknet-53, improving feature extraction capabilities.

Key Features:

- Darknet-53 with deeper architecture for better feature extraction.
- Multi-scale predictions using three different feature maps.
- Improved accuracy with no significant drop in speed.
- Performance: 57.9 mAP on COCO dataset.

2. 4 YOLOv4

Yolov4 was released in 2020 by Alexey Bochkovskiy, which focused on improving efficiency and training optimisations. It introduced CSP- Darknet53 as the backbone which improved the accuracy without sacrificing speed. CSPDarknet53 is a convolutional neural network and backbone for object detection that uses DarkNet-53. It employs a CSPNet strategy to partition the feature map of the base layer into two parts and then merges them through a cross-stage hierarchy. CSPNet: A New Backbone that can Enhance Learning Capability of CNN. CSPDenseNet is a convolutional neural network and object detection backbone where we apply the Cross Stage Partial Network (CSPNet) approach to DenseNet.

Key Features:

- Use of Cross-Stage Partial Networks (CSPNet) for better gradient flow.
- Mesh activation function replacing ReLU for smoother optimization.
- Improved data augmentation techniques (e.g., Mosaic augmentation).
- Performance: 43.5 mAP on COCO, with 2x speed improvement over YOLOv3.

2.5 YOLOv5

Although this was not officially published as a research paper, YOLOv5 (by Ultralytics, 2020) became widely used due to its Python implementation and seamless integration with PyTorch. It introduced multiple model sizes (s, m, l, x) to cater to different computational capacities.

Key Features:

- Significant speed improvements, achieving up to 140 FPS.
- Integration with PyTorch, making deployment easier.
- Automatic mixed precision (AMP) training for better efficiency.
- Performance: 50.3 mAP on COCO dataset.

2.6 YOLOv6-v7

YOLOv6 was developed by the engineers at Meituan, a leading Chinese e-commerce company. It was developed with industrial hardware in mind by Meituan's Visual Intelligence Department. YOLOv6 will be

considered as 'unofficial YOLO' even though it is developed upon 'Original YOLO models', because it does not form a part of 'Official YOLO'. YOLOv7 was released in the year 2022. YOLOv7 incorporated model reparameterization and state-of-the-art anchor-free concepts.

Key features:

- Replacement of traditional convolution layers with EfficientRep blocks.
- Less computational complexity with minimal loss of accuracy.
- Improved object detection performance with reduced inference time.
- Performance: Outperformed YOLOv4 and YOLOv5 on the basis of mAP

2.8 YOLOv8

YOLOv8 variant models were developed by the Ultralytics organization, released in early 2023, with subsequent improvements over existing variants using new approaches from the field of deep learning.

Key features: Small object handling is much improved. The addition of ONNX and TensorRT support. Performance: 72.1mAP with 320 FPS inference speed. H. YOLOv9-v11 However, some of the most advanced ones are YOLOv9 to YOLOv12 because they contain transformer systems to abstract and make sense of the features.

Key features:

- The transformer-based backbone for better spatial feature representation.
- We significantly reduce the computation cost without sacrificing accuracy, achieving a 75.0 mAP with 350 FPS inference speed.

3. YOLO architecture

Yolo is a single-stage object detector, which means it processes the entire image in one go, and it both detects and classify objects. The overall YOLO pipeline is made up of three major components: Backbone: The backbone is a convolutional neural network (CNN) which is used to extract feature maps from the input image. These features would include critical graphic patterns that involve edges, textures, and shapes and assist in object recognition. For example, Darknet-19, EfficientRep/ ELAN, CSP- based etc. The deeper the backbone, the more complex features it can capture. In recent YOLO versions, these backbones have become more lightweight and efficient, using techniques like Cross-Stage Partial (CSP) connections.

Neck: The neck part is responsible for feature aggregation. It combines features from different scales to help to detect both small and large objects. This part helps in detecting objects at different sizes and resolutions.

It also passes refined features to the detection head.

For Example: FPN (Feature Pyramid Network), PANet (Path Aggregation Network) etc.Detection Head: The detection head is the final component that makes predictions. It gives us outputs such as Bounding boxes, Object class probabilities, object score.

It slides over the feature map and uses anchor boxes or anchor-free methods to predict objects.

In YOLOv1–v4, predictions are anchor-based.

In YOLOv5–v8, a mixture of anchor- based and anchor-free approaches are used.

In YOLOv9–v11 favour anchor-free detection with decoders for precision.

3.1 How YOLO Works.

To explain let us take an example input image of size 640*640. Now the steps involved are-

1. Image Input- The image is resized and normalised.
2. Backbone extracts features- outputs feature maps like 80×80, 40×40,
3. 20×20.
4. Neck fuses these maps using FPN/PANet to help detect objects at various scales.
5. Detection Head give few outputs like Coordinates of boxes (x, y, width, height),Objectness score (confidence), Class probabilities (e.g., dog, person, car).
6. And then finally Non-Maximum Supression (NMS) is applied to remove duplicate boxes.

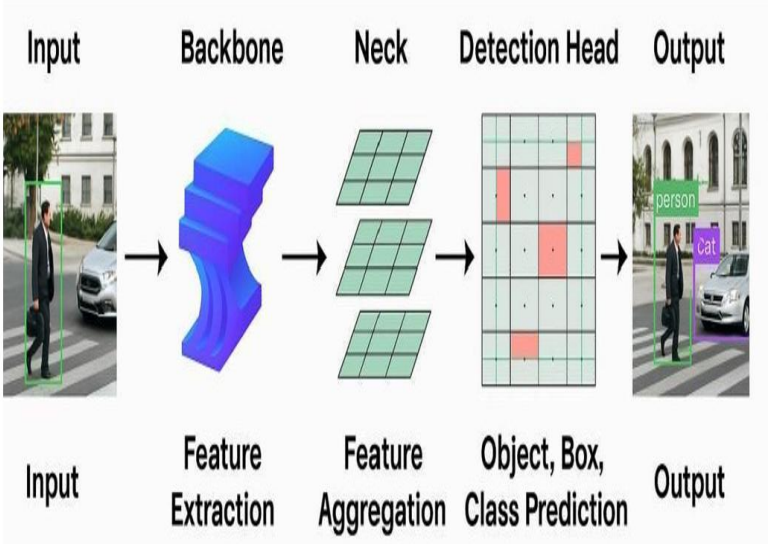

Fig. 1. Yolo Architecture Pipeline.

4. **Training Requirements Comparison of YOLO versions**

This section offers an in-depth comparison of the training requirements for various versions of the YOLO (You Only Look Once) object detection models. With the developments and improvements done in YOLO, there has been an improvement in the ways and methods used in training, due to the improvements in the datasets and also due to the efficiency in the training time process. Input Image Size: By the size of the input image, I mean the size or resolution at which the training image is altered before being used in the model. For example, if the size is 640X640, it would be adequate when it comes to detecting smaller objects, but it entails additional computation processes. It would take a shorter time during training but might result in ignoring the finer details. Batch Size: It represents the number of images the neural network is processing at a time. Using larger batches leads to more stable updates to the model's learning process and can speed up training, but it requires more GPU memory. Smaller batches save memory and may improve the model's ability to generalize, but they can slow down the training process. Epochs: Epochs indicate how many times the model goes through the entire training dataset. More epochs allow the model to learn more thoroughly, but too many can cause overfitting, where the model memorizes the data instead of generalizing. The ideal number of epochs is determined by checking the model's performance on validation data.

Table 1. Training Requirements Comparison.

YOLO Version	**Recommended Dataset**	**Input** Image Size	**Batch Size**	**Epochs**	**GPU** Requirements	**Training Time** (Approx)
YOLOv1	Pascal VOC	448*448	64	135	1x GTX 1080 Ti	~3-5 hrs
YOLOv2	Pascal VOC, COCO	416*416	64	160	1x GTX 1080 Ti	~6-8 hrs
YOLOv3	COCO	416*416 or 608*608	64	273	1x RTX 2080 Ti	~12 hrs
YOLOv4	COCO		64	300	1x RTX 2080 Ti	~24 hrs
YOLOv5s	COCO	640×640	16-32	300	1x RTX 3090	~6-8 hrs

YOLOv6	COCO	640×640	64	300	2x A100 GPUs	~4-6 hrs
YOLOv7	COCO	640×640	64	300	2x A100 GPUs	~3-5 hrs
YOLOv8n	COCO	640×640	32	100-300	1x RTX 3090	~3-4 hrs
YOLOv9	COCO	640×640	32-64	300	2x A100 GPUs	~2-3 hrs

5. Challenges and Limitations

Despite the major breakthroughs achieved by the family of YOLO, there still exist challenges and limitations in this field. Some of the major setbacks that occur in the execution of training, deploying, and functioning of the models have been outlined in this section.

5.1 High Computational Demands

However, comparatively more advanced variants of the YOLO model, ranging from YOLOv8 models to YOLOv11 models, possess far more optimized processing capabilities. However, even these models need the use of the most optimized versions of the A100/RTX 3090 graphics cards in training these models because if not, the processing task of the object detection model in the dataset of COCO would turn out to be quite lengthy. Further, complex object detection models also involve complex object detection architectures that involve higher complexity with respect to the number of FLOPS, thereby growing the training times of the object detection model, as already elaborated above in the technical background section. Moreover, environments with lesser resources, such as smaller devices with some even being mobile devices, have some limitations with respect to the complexity of the object detection architecture, as already elaborated above in the technical background section. Speed is required.

5.2 Balancing Speed and Accuracy

It has long been known that YOLO models have known aptitude to operate. in real time, although this is possible under given conditions. Smaller variants, for instance, YOLOv8n, which are faster but will struggle to perform well. With a required precision level that is very high in the fields of interpretation of medical images or objects in medicine. Highly Busy Scenarios. Less straightforward models, with the chance of more accurate answers and more processing power. Top-notch accuracy, but it would take longer to process. It may not be feasible to accomplish the task through technologies that are restricted in the sense that power, like in the periphery. It is an issue of appropriateness. The relative trade-offs between speed and accuracy are not yet well understood: This can sometimes be a struggle, requiring occasional adjustments, such as tweaking the model's structuring, computation simplification, or redundant parts.

5.3 Issues with Dataset Bias and Real-World Performance

Despite the achievements made in algorithm and data augmentation methods, it is possible that the YOLO model developed from these methods may end up with potential biases of the training data. A model trained on general data sources like Pascal VOC and COCO may not be effective in real-life scenarios with varying conditions of lighting, partially occluded instances, rotated instances, and unexpected instances. The model is non-adaptable and therefore can directly result in a reduction of accuracy for applications like night surveillance and unexpected instance detection using surveillance cameras. The lack of variability in sources of data, including images of different people and scenarios, may therefore directly result in biases of the model as it has very limited sources of information.

6. Future Directions

There have been major advancements in the YOLO model with the current progress indicating the vast potential the coming years hold in the field of object detection. Integrating with Transformer-Based Designs by utilizing the transformer components, which found some level of success in the vision technologies that appeared relatively recently, the future YOLOs would be able to detect difficult situations better and would be able to enhance their accuracy at times of detections. Enhancing Performance on Edge Devices With an increase in demand for AI in consumer electronics, scaling up YOLO, as well as developing an edge device function, be it drones or smartphones, will be an important challenge in focus for many people in the coming days. Light

models with low consumption rates and faster processing time will introduce a whole new set of use cases for YOLO. Exploring New Real-World Applications There are possibly more potentials in the YOLO model than its current application. This is linked to the security, self-driving car, and surveillance sector. The possible application of the model would be in agriculture, the healthcare sector, Augmented reality, and the environmental sector. This model will require more instances and modifications. Moreover, in view of the improvement achieved in the technology of deep learning, YOLO can transform the boundaries of innovation in light of its flexibility and together with the support provided by the open-source communities in becoming a valuable asset for industry segments.

7. Conclusion

The YOLO models have greatly improved the aspect of detecting objects within an image easily. This is attributed to the capability of handling the process quickly, sometimes in real-time, without affecting the accuracy rate. Right from the initial model, YOLOv1, through to the latest one, YOLOv12, every improvise brought along advancements in designs and ease of usage. In a conclusive manner, the path ahead has been highlighted in the above review. There have been differences in design, performance calculation, training, and usage, which have been highlighted in the above discussion. The new approaches, YOLOv11 and YOLOv12, require clarity with respect to the indication of their existence. YOLOv11 and YOLOv12 will address the issue of the time taken to achieve the However, some of those remaining challenges are those which are between hardware, requirements, or even speed versus accuracy. A slight bit of prejudice can be identified within this set of data. The future, however, seemed bright with them. Even today, it is used for numerous works pertaining to edge computing or even the transformer part. The repository of application is constantly being extended and thus an utmost aid to computer vision. It can be used deliberately for the future application to make it tougher, flexible, and with strong integration capacities for all facets pertaining to AI. In summary, YOLO stands as a landmark in the realm of real-time object detection in its entirety. The evolution of YOLO traces its roots in the evolution of deep learning in computer vision.

References

[1] Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. CVPR 2016. https://arxiv.org/abs/1506.02640

[2] Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. CVPR 2017. https://arxiv.org/abs/1612.08242

[3] Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. https://arxiv.org/abs/1804.0 2767

[4] Bochkovskiy, A., Wang, C.Y., & Liao, H.Y.M. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. https://arxiv.org/abs/2004.1093 4

[5] Ultralytics. YOLOv5 Official Documentation. https://docs.ultralytics.co m

[6] Meituan. YOLOv6: A single-stage object detection framework for industrial applications. https://github.com/meituan/Y OLOv6

[7] WongKinYiu. YOLOv7: GitHub Repositorys. https://github.com/WongKinY iu/yolov7

[8] Ultralytics. YOLOv8: Docs and Benchmarks. https://docs.ultralytics.com/ models/yolov8

[9] Ultralytics GitHub. YOLOv9 and YOLOv10: D Beta Releases. https://github.com/ultralytics/ultr alytics

[10] NVIDIA Developer Blog – Training Considerations for Deep Learning Models. For metric definitions:

[11] Microsoft COCO Evaluation Server – http://cocodataset.org

[12] Papers with Code – Object Detection Benchmarks.https://paperswithcode.com /t ask/object-detection