# Evolution of YOLO

## From v1 to v8

ChunWoong Park (202545335)

Fundamental Advancement in AI

# Agenda

1. Motivation
2. YOLOv1–v3: The Early Stage
3. YOLOv4–v8: Evolution & Results
4. Implementation Highlights
5. Discussion & Future Trends

# Motivation – Why YOLO?

“Real-time object detection requires a unified design.”

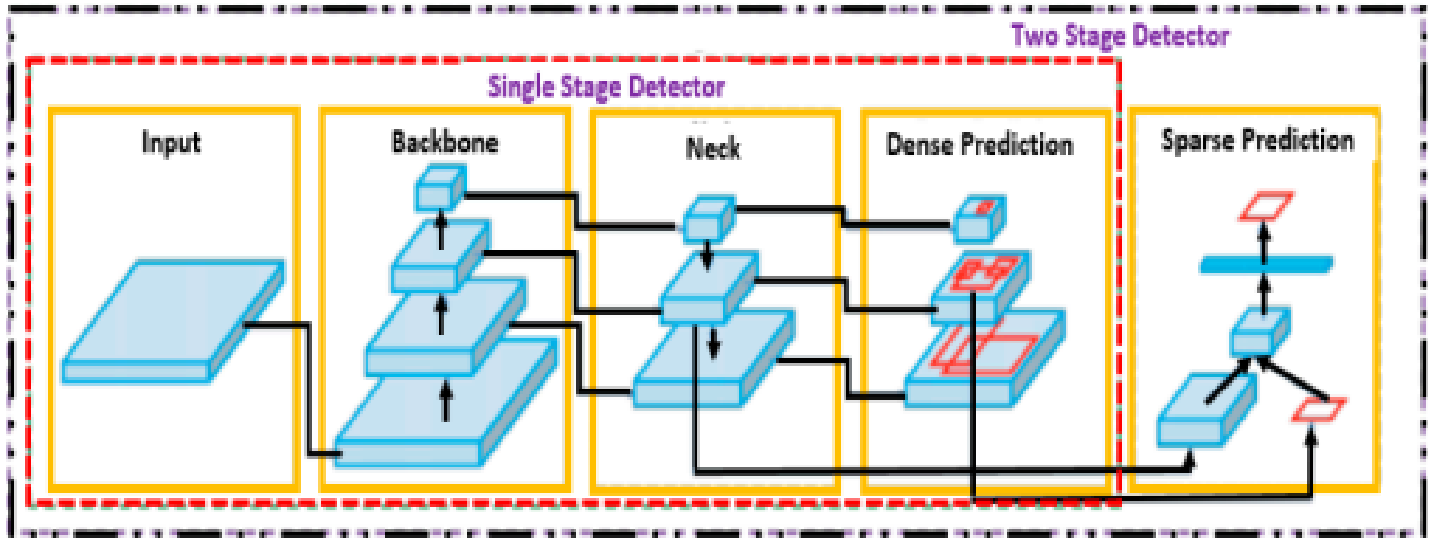



**Figure 1.** Object detector anatomy.

Source: Machines (2023), Figure 1

**Two-Stage Detectors**

Slow but highly accurate.

**Single-Stage Detectors (YOLO)**

Fast and unified for real-time.

# Evolution Timeline

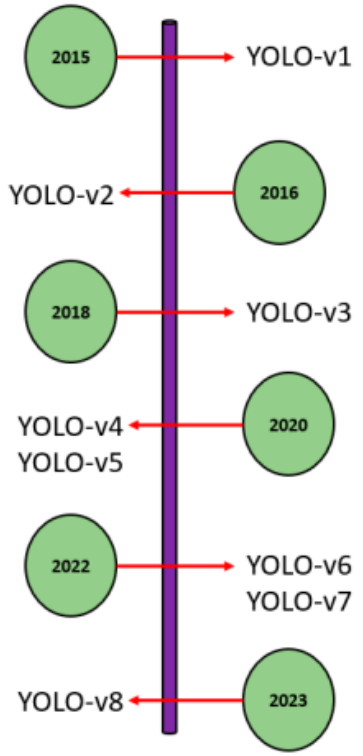



**Figure 2.** YOLO evolution timeline.

Source: Machines (2023), Figure 2

## 2016: The Beginning

v1 & v2 establish the core concept.

## 2018: Refinement

v3 improves small object detection.

## 2020: Optimization Era

v4 & v5 focus on engineering and usability.

## 2022-23: Modernization

v6, v7, & v8 introduce scaling and anchor-free designs.

# YOLOv1–v3: The Foundational Stage

## YOLOv1 (2016)

### Unified Detection

Introduced the core concept of treating detection as a single regression problem using a grid system.

**Weakness: Poor localization.**

## YOLOv2 (2017)

### Better, Faster, Stronger

Added Batch Normalization and Anchor Boxes to improve recall and stabilize training.

**Contribution: Improved accuracy.**

## YOLOv3 (2018)

### Multi-Scale Detection

Used a Feature Pyramid Network (FPN) to make predictions at three different scales, improving small object detection.

**Contribution: Small object focus.**

**These foundational versions established the core principles and tackled early challenges, paving the way for modern detectors.**

# YOLOv4 – Backbone & Neck Architecture

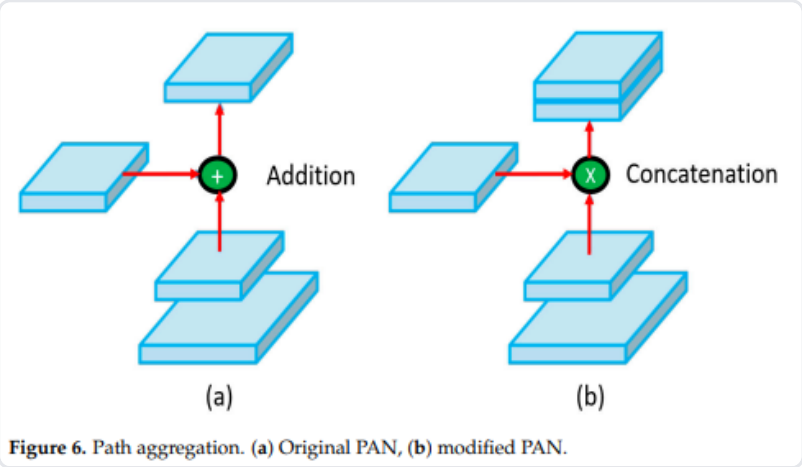



**Figure 6.** Path aggregation. **(a)** Original PAN, **(b)** modified PAN.

Source: Machines (2023), Figure 6

**Introduced CSPDarknet53 backbone and Path Aggregation Network (PANet) neck.**

This combination enabled robust multi-scale feature fusion, significantly improving performance.

# YOLOv4 – "Bag of Freebies & Specials"

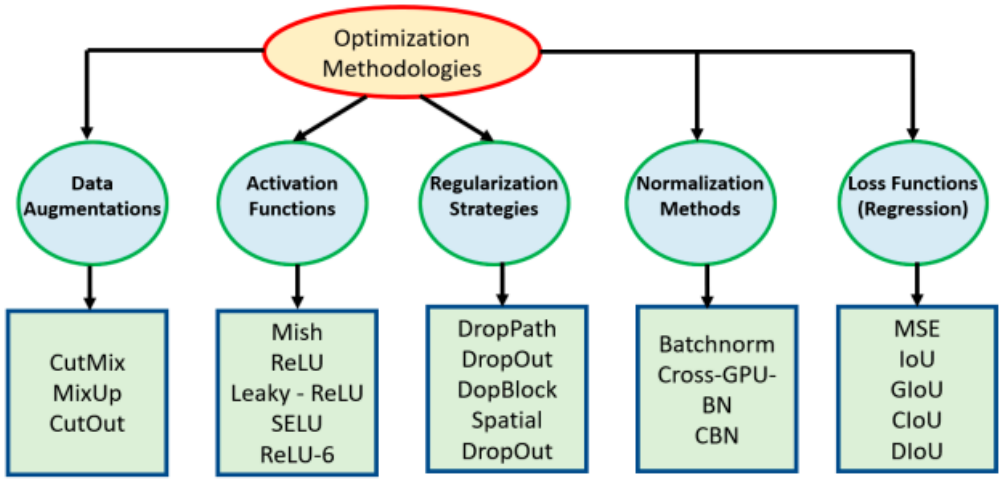



**Figure 7.** State-of-the-art optimization methodologies experimented in YOLO-v4 via bag-of-specials.

Source: Machines (2023), Figure 7

## Augmentation

- Mosaic / CutMix
- DropBlock Regularization

## Activation & Normalization

- Mish Activation
- Cross-GPU Batch Norm

## Loss Function

- CIoU / DIoU / GIoU Loss

# YOLOv5 – Practical Engineering

**PyTorch-Native Framework**
Shifted from Darknet to PyTorch, dramatically improving accessibility and ease of use.

**AutoAnchor & Data Pipeline**
Automated anchor optimization and streamlined data loading for faster training.

**Model Scalability**
Introduced various model sizes (n, s, m, l, x) for different speed-accuracy trade-offs.

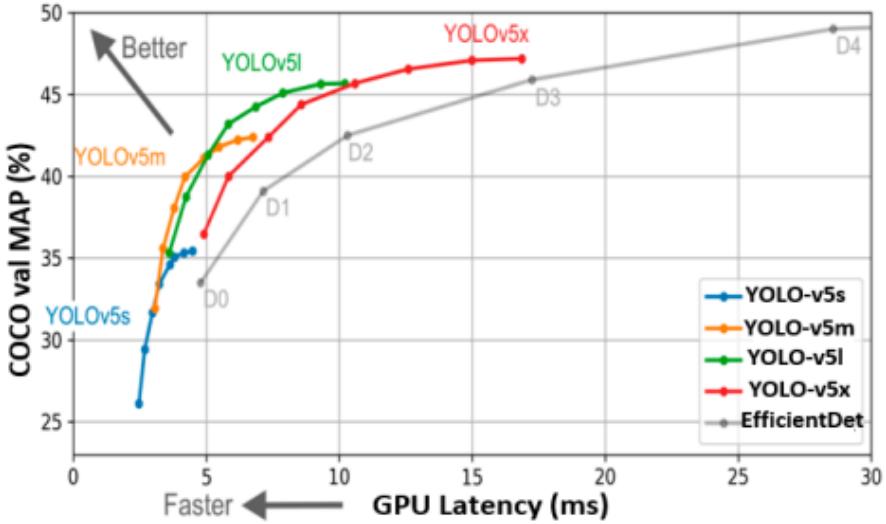



**Figure 8.** YOLO-v5 variant comparison vs. EfficientDet [61].

Source: Machines (2023), Figure 8

**Table 1.** YOLO-v5 internal variant comparison.

| Model | Average Precision (@50) | Parameters | FLOPs |
|---|---|---|---|
| YOLO-v5s | 55.8% | 7.5 M | 13.2B |
| YOLO-v5m | 62.4% | 21.8 M | 39.4B |
| YOLO-v5l | 65.4% | 47.8 M | 88.1B |
| YOLO-v5x | 66.9% | 86.7 M | 205.7B |

Source: Machines (2023), Table 1

# YOLOv6 – Industrial Focus

### EfficientRep Backbone
Designed a hardware-efficient backbone for faster inference on various devices.

### Decoupled Head
Separated classification and regression heads, improving accuracy by resolving task conflicts.

### Anchor-Free & Edge Optimization
Adopted an anchor-free approach, simplifying the pipeline for edge deployment.


**Figure 10.** Relative evaluation of YOLO-v6 vs. YOLO-v5 [71].

Source: Machines (2023), Figure 10

**Table 2.** YOLO-v6 variant comparison.

| Variant | mAP 0.5:0.95 (COCO-val) | FPS Tesla T4 | Parameters (Million) |
|---|---|---|---|
| YOLO-v6-N | 35.9 (300 epochs) | 802 | 4.3 |
| YOLO-v6-T | 40.3 (300 epochs) | 449 | 15.0 |
| YOLO-v6-RepOpt | 43.3 (300 epochs) | 596 | 17.2 |
| YOLO-v6-S | 43.5 (300 epochs) | 495 | 17.2 |
| YOLO-v6-M | 49.7 | 233 | 34.3 |
| YOLO-v6-L-ReLU | 51.7 | 149 | 58.5 |

Source: Machines (2023), Table 2

# YOLOv7 – Scaling Up

### E-ELAN and Compound Scaling

Introduced Extended-ELAN and a compound scaling method to efficiently scale model depth and width together.

### Trainable Bag-of-Freebies

Continued the "Bag of Freebies" concept with more advanced, trainable modules to boost accuracy without inference cost.

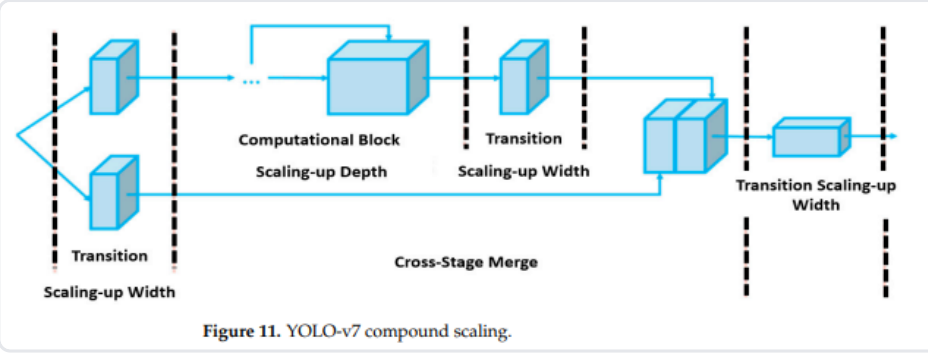



**Figure 11.** YOLO-v7 compound scaling.

Source: Machines (2023), Figure 11

**Table 3.** Variant comparison of YOLO-v7.

| Model | Size (Pixels) | mAP (@50) | Parameters | FLOPs |
|---|---|---|---|---|
| YOLO-v7-tiny | 640 | 52.8% | 6.2 M | 5.8G |
| YOLO-v7 | 640 | 69.7% | 36.9 M | 104.7G |
| YOLO-v7-X | 640 | 71.1% | 71.3 M | 189.9G |
| YOLO-v7-E6 | 1280 | 73.5% | 97.2 M | 515.2G |
| YOLO-v7-D6 | 1280 | 73.8% | 154.7 M | 806.8G |

Source: Machines (2023), Table 3

# YOLOv7 – Performance Gain

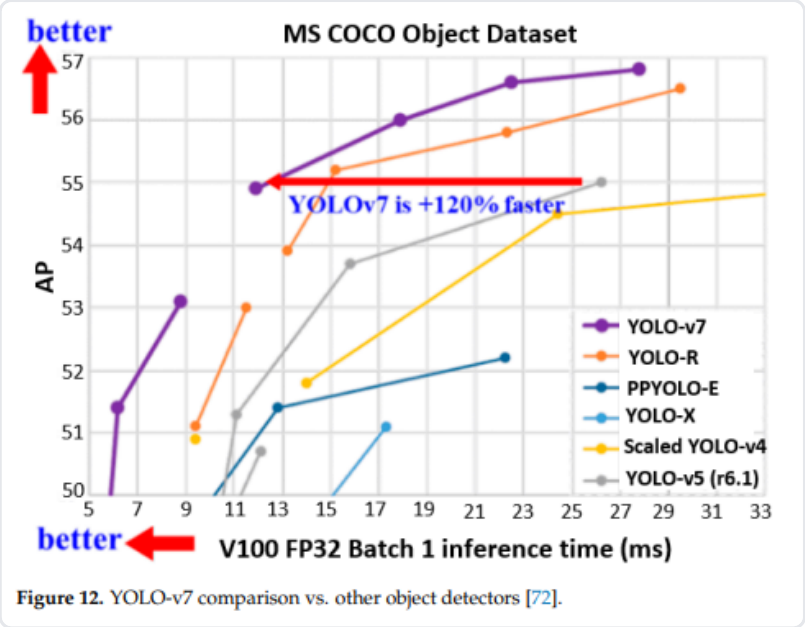



**Figure 12.** YOLO-v7 comparison vs. other object detectors [72].

Source: Machines (2023), Figure 12

### +120% Faster

Significantly faster than YOLOv5 at similar accuracy levels.

### New SOTA mAP

Achieved a new state-of-the-art mAP of ~57% on COCO.

### Efficient Training

Auxiliary head and other techniques led to a more efficient training pipeline.

# YOLOv8 – The Next Step

### Fully Anchor-Free
Completely removes the need for pre-defined anchors, simplifying the model and improving generalization.

### Decoupled Detection Head
Separates classification and regression, a proven technique for better performance.

### Multi-Size Models (n/s/m/l/x)
Continues the successful strategy of providing scalable models for diverse hardware needs.

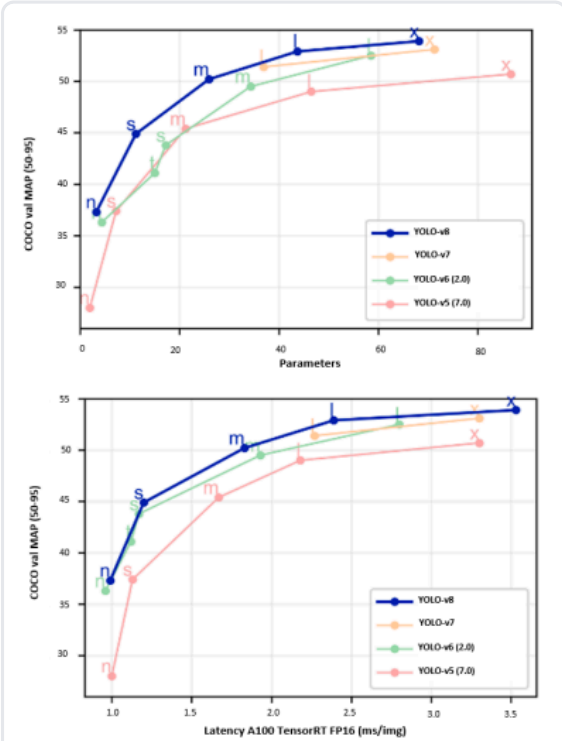



**Figure 13.** YOLO-v8 comparison with predecessors [80].

Source: Machines (2023), Figure 13

**"A new state-of-the-art balance between speed and precision is achieved."**

# Evolution Summary Table

**Table 5.** Abstract variant comparison.

| Variant | Framework | Backbone | AP (%) | Comments |
|---|---|---|---|---|
| V1 | Darknet | Darknet-24 | 63.4 | Only detect a maximum of two objects in the same grid. |
| V2 | Darknet | Darknet-24 | 63.4 | Introduced batch norm, k-means clustering for anchor boxes. Capable of detecting > 9000 categories. |
| V3 | Darknet | Darknet-53 | 36.2 | Utilized multi-scale predictions and spatial pyramid pooling leading to larger receptive field. |
| V4 | Darknet | CSPDarknet-53 | 43.5 | Presented bag-of-freebies including the use of CIoU loss. |
| V5 | PyTorch | Modified CSPv7 | 55.8 | First variant based in PyTorch, making it available to a wider audience. Incorporated the anchor selection processes into the YOLO-v5 pipeline. |
| V6 | PyTorch | EfficientRep | 52.5 | Focused on industrial settings, presented an anchor-free pipeline. Presented new loss determination mechanisms (VFL, DFL, and SIoU/GIoU). |
| V7 | PyTorch | RepConvN | 56.8 | Architectural introductions included E-ELAN for faster convergence along with a bag-of-freebies including RepConvN and reparameterization-planning. |
| V8 | PyTorch | YOLO-v8 | 53.9 | Anchor-free reducing the number of prediction boxes whilst speeding up non-maximum suppression. Pending paper for further architectural insights. |

Source: Machines (2023), Table 5

### Framework & Backbone

Shift from Darknet to PyTorch, with increasingly efficient backbones.

### Core Contributions

Each version introduced key innovations from anchor boxes to anchor-free designs.

### Focus Shift

The focus evolved from raw accuracy to engineering, usability, and modularity.

# Overall Performance Comparison

## YOLOv5 vs. EfficientDet

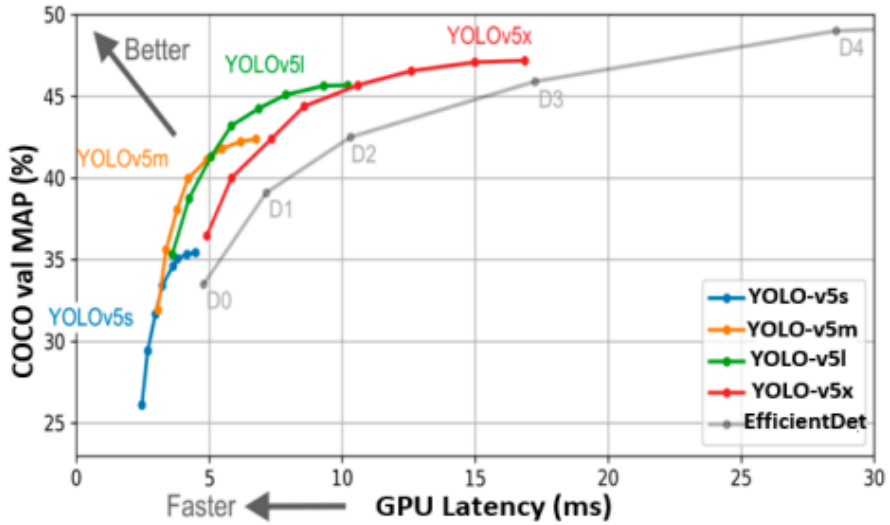



**Figure 8.** YOLO-v5 variant comparison vs. EfficientDet [61].

## YOLOv7 vs. Others

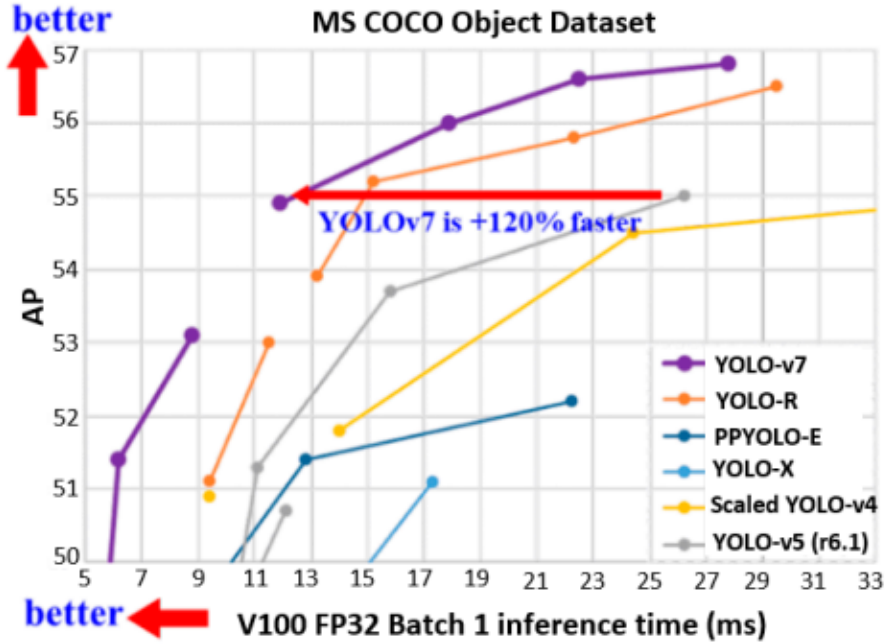



**Figure 12.** YOLO-v7 comparison vs. other object detectors [72].

**YOLOv8 continues this trend, maintaining leading real-time accuracy under low latency conditions.**

# Implementation Highlights

Modern YOLO frameworks like Ultralytics have made training and deployment incredibly simple.

```
# Train a YOLOv8n model on the COCO8 dataset for 100 epochs
yolo detect train data=coco8.yaml model=yolov8n.pt imgsz=640 epochs=100
```

**YAML Configuration**
Dataset paths and class names are defined in a simple YAML file.

**CLI Interface**
Training, validation, and export are handled with a single command.

**Easy Export**
Models can be exported to ONNX, TensorRT, and other formats with one command.

# Future Trends

**Transformer & Attention Fusion**
Integrating attention mechanisms and Transformer blocks (like in DETR or the upcoming YOLOv9) to improve feature extraction.

**Unified Multi-Task Models**
A single model for detection, segmentation, and pose estimation is becoming the standard, as seen in YOLOv8.

**Edge Deployment & Quantization**
Continued focus on model compression, quantization, and hardware-specific optimization for efficient edge AI.

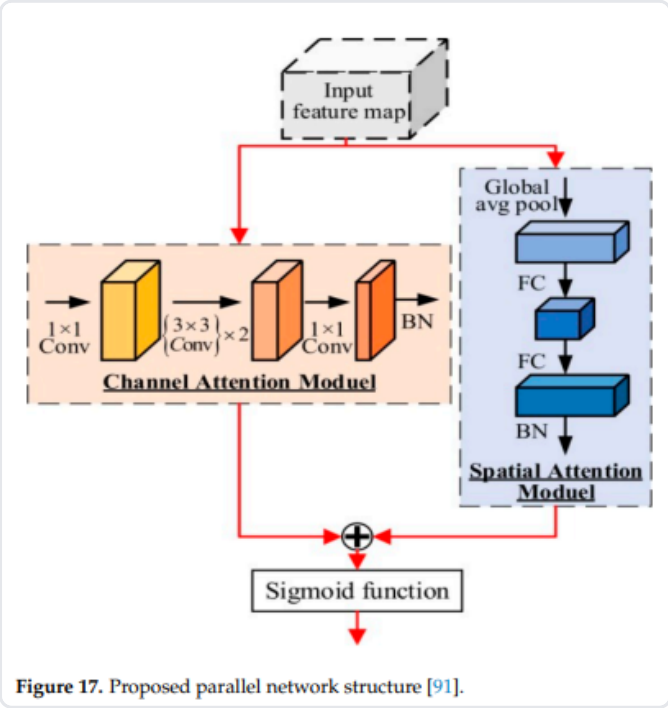



**Figure 17.** Proposed parallel network structure [91].

Source: Machines (2023), Figure 17

# Discussion / Q&A

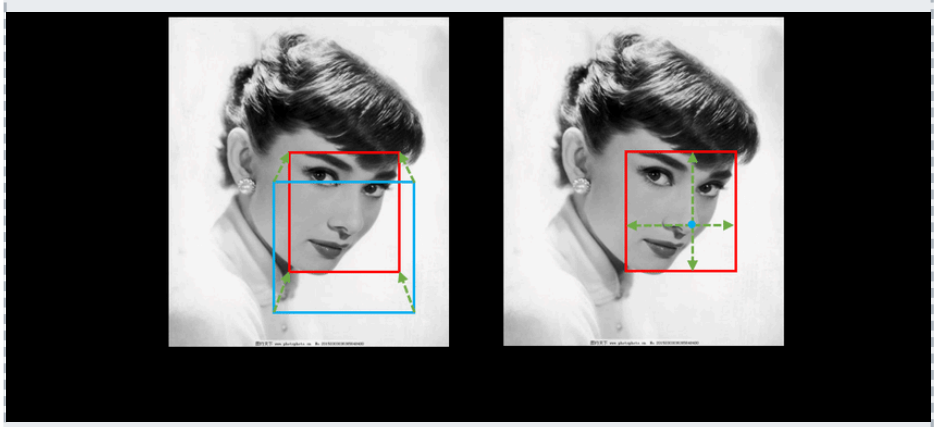
- Is anchor-free always superior to anchor-based designs?
- What are the fundamental limits of YOLO on small object detection?
- Could a hybrid Transformer-YOLO architecture surpass DETR in performance?

# Q1: Is anchor-free always superior?

**Usually, for Generalization**
It simplifies the pipeline by removing anchor-related hyperparameters and often generalizes better to objects with diverse shapes and sizes.

**Not Always**
Anchor-based models can perform better if objects in a dataset have very uniform and specific aspect ratios, where hand-picked anchors provide a strong prior.

**Conclusion: A Favorable Trend**
The industry trend is towards anchor-free for its simplicity and robust performance, but the best choice can still be task-dependent.



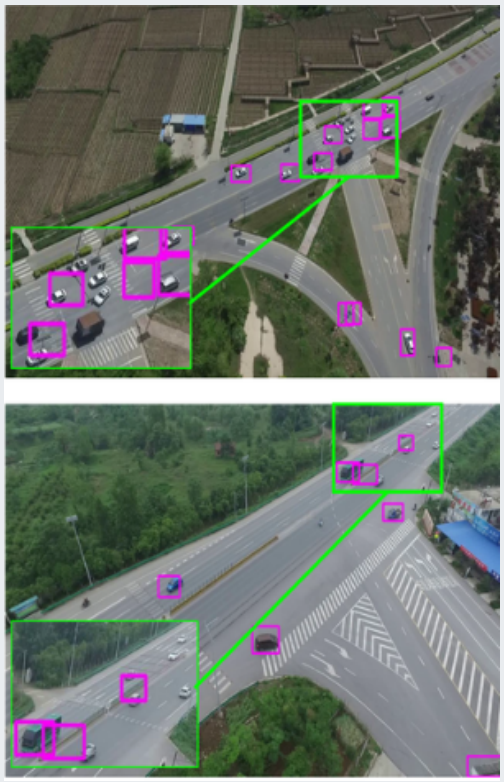
Anchor-based vs. Anchor-free Diagram

# Q2: What limits YOLO on small objects?

**Loss of Spatial Information**
Repeated downsampling in the backbone layers causes fine-grained features of small objects to vanish.

**Insufficient Receptive Field**
The features from the deepest layers may not have enough resolution to distinguish very small objects from the background.

**Solutions**
FPN (in v3+), higher resolution training, and specialized data augmentation (e.g., Mosaic) help mitigate this, but it remains a challenge.



Zoomed-in image showing a missed small object.

# Q3: Could a hybrid Transformer-YOLO surpass DETR?

### DETR's Challenge

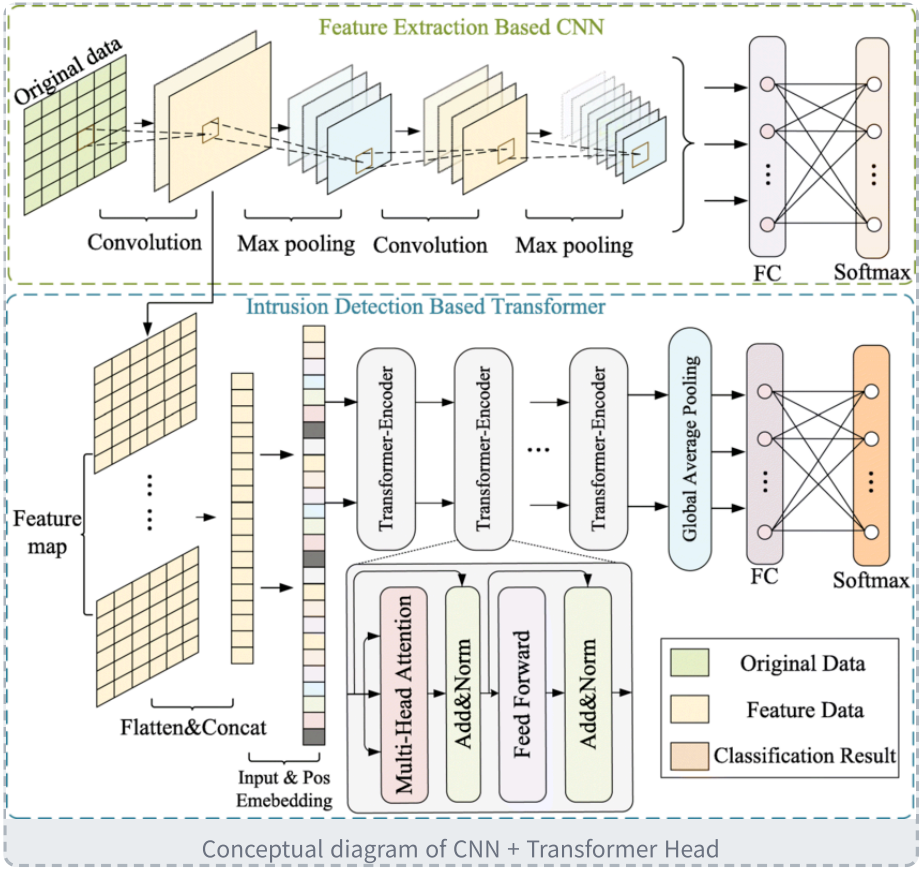
DETR (DEtection TRansformer) is powerful but known for its slow training convergence and high computational cost.

### YOLO's Strength

YOLO's highly optimized CNN backbones and training pipeline are extremely fast and efficient.

### The Hybrid Promise

Yes, it's highly likely. A hybrid model that combines YOLO's efficient CNN backbone with a lightweight Transformer neck/head could achieve DETR-like global context understanding with YOLO's speed.




Conceptual diagram of CNN + Transformer Head

# Conclusion

**“YOLO has evolved from a monolithic CNN to a modular, scalable, and highly engineered architecture for real-time object detection.”**

### v1–v3: Foundational Stage

Established the single-stage paradigm and introduced core concepts like multi-scale features.

### v4–v6: Optimization Stage

Focused on integrating the best "Bag of Tricks" for optimal speed/accuracy and industrial usability.

### v7–v8: Modern Stage

Pushed towards efficient scaling, anchor-free designs, and modular, multi-task frameworks.

# References


Hussain, M. (2023). YOLO-v1 to YOLO-v8, the Rise of YOLO and Its Complementary Nature toward Digital Manufacturing and Industrial Defect Detection. *Machines, 11*(7), 677.

Redmon, J., et al. (2016). You Only Look Once: Unified, Real-Time Object Detection. arXiv:1506.02640.

Redmon, J., & Farhadi, A. (2017). YOLO9000: Better, Faster, Stronger. arXiv:1612.08242.

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. arXiv:1804.02767.

Bochkovskiy, A., et al. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv:2004.10934.

Li, C., et al. (2022). YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. arXiv:2209.02976.

Wang, C. Y., et al. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. arXiv:2207.02696.

Ultralytics YOLOv5 & YOLOv8 Repositories (GitHub).


# Thank You

Q & A