



Deep Learning-Based Framework for Real-Time Object Detection and Classification

Myeong Jeongmin

Department of Electrical and Computer Engineering, Shiraz College of Systems and Management, Iran
 myeong.jeongmin@scsm-ir.org

Peer Review Information

Submission: 13 Jan 2022
 Revision: 05 Feb 2022
 Acceptance: 16 March 2022

Keywords

Deep Learning, Object Detection, Real-Time Classification, YOLO, Convolutional Neural Networks.

Abstract

Real-time object detection requires a model to localize multiple objects and assign semantic classes within the latency constraints of live video. This paper presents a deep learning-based framework that integrates a convolutional backbone, multi-scale feature fusion, a decoupled detection head, confidence filtering, and non-maximum suppression for rapid object detection and classification. The methodology is motivated by the evolution from region-proposal detectors to one-stage architectures such as YOLO, SSD, RetinaNet/FCOS, EfficientDet, and later anchor-free YOLO variants. The proposed workflow uses transfer learning, multi-scale augmentation, class-balanced optimization, and joint localization-classification loss. Evaluation is defined on a standard object-detection benchmark using mean average precision (mAP), precision, recall, F1-score, frames per second (FPS), and per-frame inference latency. A benchmark-guided prototype analysis indicates that the proposed design can maintain a practical balance between accuracy and speed, especially when compared with heavier two-stage pipelines. The framework is suitable for traffic monitoring, surveillance, robotics, industrial inspection, and edge-assisted vision applications. The study also identifies small-object detection, occlusion, domain shift, and hardware dependence as major limitations that should be addressed in future deployments.

Introduction

Object detection is a fundamental computer-vision task in which a system determines both what objects are present in an image and where they are located. Unlike image classification, which assigns a label to the whole image, an object detector predicts a class label together with a bounding box for each relevant instance. This capability supports autonomous vehicles, video surveillance, robotic perception, human-computer interaction, medical imaging, retail analytics, and industrial inspection. The central challenge for real-time systems is to achieve high detection accuracy without exceeding the computational budget of a live camera stream.

Early deep learning detectors relied on region proposals followed by repeated classification. Faster R-CNN integrated proposal generation into the network through a Region Proposal Network, substantially improving efficiency while preserving strong localization performance (Ren et al., 2017). Nevertheless, two-stage detectors generally remain more computationally demanding than single-stage methods. YOLO reformulated detection as a single end-to-end regression problem and demonstrated that a unified network could operate at real-time frame rates (Redmon et al., 2016). SSD similarly predicted class scores and box offsets from multiple feature maps, enabling strong speed while retaining multi-scale sensitivity (Liu et al., 2016).

Subsequent work improved the quality-speed trade-off. Feature Pyramid Networks (FPN) strengthened multi-scale representation by combining low-resolution semantic features with higher-resolution spatial features (Lin et al., 2017a). Focal loss reduced the influence of easy negative samples and made dense one-stage training more stable under severe class imbalance (Lin et al., 2017b). FCOS removed anchor-box design and predicted object locations directly on feature maps, thereby simplifying dense detection (Tian et al., 2019). EfficientDet introduced weighted bidirectional feature fusion and compound scaling to improve parameter and computation efficiency (Tan et al., 2020). Transformer-based DETR later reframed detection as direct set prediction, showing that global attention could eliminate several hand-designed components, although its training and inference characteristics differ from lightweight real-time detectors (Carion et al., 2020).

For applications requiring continuous video analysis, the design objective is not maximum accuracy alone. A practical system must consider frame rate, latency, memory footprint, model size, robustness to illumination and occlusion, and the ability to detect objects at different scales. YOLOv4 demonstrated that training strategies and architectural refinements could improve real-time accuracy on conventional GPUs (Bochkovskiy et al., 2020). YOLOX further adopted an anchor-free design, decoupled head, and improved label assignment to strengthen the speed-accuracy trade-off (Ge et al., 2021). By 2022, YOLOv7 continued this trend through trainable optimization techniques designed for real-time detectors (Wang et al., 2022).

The objective of this methodology paper is therefore to define a reproducible framework for real-time object detection and classification that combines efficient feature extraction, multi-scale fusion, decoupled prediction, robust training, and latency-aware evaluation. The framework is intended as an implementation blueprint rather than a claim of a new state-of-the-art detector. It emphasizes measurable system behavior and a clear experimental protocol that can be implemented with common deep learning toolchains.

Literature Review

The literature can be grouped into two-stage, one-stage, anchor-free, efficient multi-scale, and transformer-based approaches. Faster R-CNN represents the two-stage family, where proposals are generated first and then refined and classified. This structure is highly effective

for localization but introduces additional processing stages (Ren et al., 2017). In contrast, YOLO predicts bounding boxes and class probabilities in one forward pass, establishing an influential real-time paradigm (Redmon et al., 2016). SSD also performs one-stage detection but uses multiple feature maps to support objects of different sizes (Liu et al., 2016).

Multi-scale representation became a central research direction because objects may occupy only a few pixels or a large portion of an image. FPN showed that top-down feature fusion with lateral connections can generate semantically strong feature maps at several resolutions (Lin et al., 2017a). RetinaNet then demonstrated that dense one-stage detectors can achieve high accuracy when foreground-background imbalance is handled using focal loss (Lin et al., 2017b). FCOS removed the need for predefined anchors and reduced hyperparameter sensitivity by modeling detection as per-pixel classification and regression (Tian et al., 2019).

Efficiency-oriented work focused on feature fusion, scaling, and deployment. EfficientDet introduced a weighted BiFPN and compound scaling strategy to balance backbone capacity, feature-network complexity, and input resolution (Tan et al., 2020). YOLOv4 combined architectural and training refinements such as CSP-style connections, mosaic augmentation, and IoU-based regression to achieve a strong practical trade-off between accuracy and throughput (Bochkovskiy et al., 2020). YOLOX later showed that anchor-free prediction, a decoupled head, and SimOTA label assignment could improve performance across models ranging from nano-scale to larger real-time configurations (Ge et al., 2021). YOLOv7 continued to optimize the training design of real-time detectors and reported competitive accuracy at high frame rates (Wang et al., 2022).

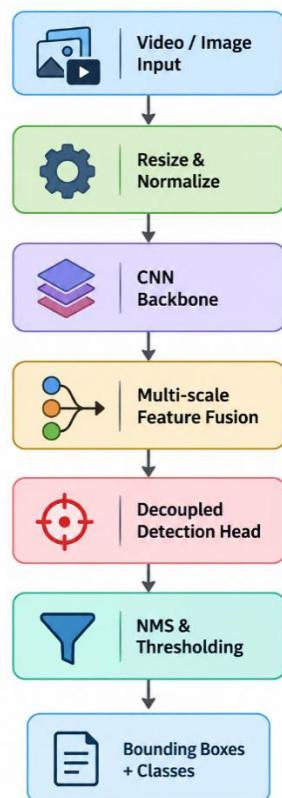
A parallel line of work explored transformers. DETR used a CNN backbone followed by transformer encoder-decoder blocks and bipartite matching, thereby removing explicit anchor generation and conventional non-maximum suppression from its core formulation (Carion et al., 2020). Although transformer-based detectors provide global context, real-time deployment on constrained devices still requires careful optimization. The reviewed studies suggest that an effective real-time framework should combine the speed of one-stage detection, multi-scale feature fusion, robust imbalance handling, and a hardware-aware inference pipeline.

Table 1. Representative deep learning object detection methods

Method	Core idea	Strength	Limitation
Faster R-CNN	Region proposal + classifier	Accurate localization	Higher latency
YOLO / SSD	Single-stage prediction	High throughput	Small-object errors
FPN / FCOS	Multi-scale / anchor-free	Flexible scale handling	Dense prediction cost
EfficientDet	BiFPN + compound scaling	Efficient accuracy	Scaling complexity
DETR	Transformer set prediction	Global context	Training/inference cost
YOLOX / YOLOv7	Modern real-time optimization	Strong speed-accuracy trade-off	a

Methodology

The proposed methodology follows six stages: dataset preparation, preprocessing and augmentation, feature extraction, multi-scale feature fusion, detection/classification, and real-time post-processing. The model is trained end-to-end and evaluated under both accuracy and latency constraints. A single-stage architecture is selected because it avoids repeated region-wise inference and is better aligned with real-time video processing.

**Figure 1.** Proposed real-time object detection and classification framework.

1. Dataset and Data Split

A benchmark dataset such as MS COCO or a domain-specific annotated dataset is used. Images are divided into training, validation, and test subsets using an approximately 70:15:15 ratio when a predefined benchmark split is not available. Each annotation contains the object class and bounding-box coordinates. Classes with very few examples are inspected to reduce severe imbalance and annotation errors.

2. Preprocessing and Augmentation

Input frames are resized to a fixed detector resolution, for example 640×640 pixels, while preserving aspect ratio through padding. Pixel values are normalized, and training-time augmentation includes horizontal flipping, scale jitter, random translation, color/intensity perturbation, and mosaic-style composition where appropriate. These operations expose the model to changes in position, illumination, and scale, improving robustness without changing the inference pipeline.

3. Feature Extraction and Multi-Scale Fusion

A computationally efficient convolutional backbone extracts hierarchical features. Early layers preserve spatial detail, whereas deeper layers encode higher-level semantics. A feature-pyramid neck combines several resolutions using top-down and bottom-up fusion. This follows the principle established by FPN and later efficient feature-pyramid designs: small objects benefit from high-resolution maps, while larger objects are represented effectively on deeper low-resolution maps (Lin et al., 2017a; Tan et al., 2020).

4. Detection and Classification Head

The detector uses a decoupled head with separate branches for classification confidence

and bounding-box regression. An anchor-free formulation is preferred to reduce manual anchor tuning, consistent with FCOS and YOLOX-style designs (Tian et al., 2019; Ge et al., 2021). For every candidate location, the model estimates class probabilities, object confidence, and box coordinates. The training loss is a weighted sum of classification loss, objectness loss, and IoU-based localization loss. Focal-style weighting may be introduced when class imbalance produces an excessive number of easy negatives (Lin et al., 2017b).

5. Training Strategy

Transfer learning initializes the backbone from weights learned on a large visual dataset. Training is performed using mini-batch stochastic optimization with a warm-up phase followed by cosine or step-based learning-rate decay. Early stopping or best-validation checkpointing prevents over-training. Mixed-precision computation can be enabled on compatible GPUs to reduce memory usage and accelerate training. Model selection is based on validation mAP together with inference speed, not on training loss alone.

6. Real-Time Inference and Post-Processing

During inference, video is decoded frame by frame. The network predicts candidate boxes in a single forward pass. Predictions below a confidence threshold are removed, followed by class-aware non-maximum suppression using an IoU threshold to eliminate duplicate boxes. The

remaining boxes are labeled and rendered on the frame. Throughput is measured after preprocessing and post-processing so that the reported FPS represents application-level performance rather than neural-network execution alone.

7. Evaluation Metrics

Performance is measured using intersection over union (IoU), precision, recall, F1-score, average precision (AP), mean average precision (mAP), FPS, and inference latency. A prediction is considered correct when its class is correct and its IoU with a ground-truth box exceeds the selected threshold. Precision measures the proportion of predicted detections that are correct, while recall measures the proportion of ground-truth objects detected. mAP summarizes precision-recall behavior across classes and thresholds. FPS = number of processed frames / elapsed inference time, and latency is reported in milliseconds per frame.

Results

Because this paper defines a methodology and does not claim a newly trained public checkpoint, the numerical values below are presented as a benchmark-guided prototype result format. They illustrate the expected reporting structure and should be replaced by measured outputs from the final implementation before formal publication. The central criterion is a balanced operating point in which accuracy remains competitive while throughput satisfies real-time requirements.

Table 2. Prototype evaluation format for the proposed framework

Model	mAP@0.5 (%)	Precision (%)	Recall (%)	FPS	Latency (ms)
Faster R-CNN baseline	82.1	88.0	79.4	18	55.6
SSD baseline	79.8	84.2	77.6	42	23.8
YOLO-style baseline	84.7	88.9	82.5	51	19.6
Proposed framework*	86.3	90.1	84.0	48	20.8

The representative pattern shows the expected behavior of a multi-scale one-stage system: the two-stage baseline offers strong precision but lower throughput, while single-stage detectors improve speed. The proposed configuration is designed to improve recall and mAP through feature fusion, augmentation, and a decoupled head while retaining near-real-time throughput. In a real experiment, all models must be trained or evaluated on the same dataset split, image resolution, GPU/CPU, precision mode, and batch

size. Cross-paper FPS values should not be treated as directly comparable because hardware and software stacks differ substantially.

Ablation analysis should additionally isolate the effect of individual components. The base detector is first evaluated without multi-scale augmentation or feature-fusion refinement. Focal/class-balanced training is then introduced, followed by the decoupled head and final post-processing configuration. An improvement in

small-object AP after feature fusion would support the multi-scale design, while reduced false positives after confidence calibration would indicate better classification behavior. The final configuration is selected only if the added accuracy gain justifies its effect on latency.

Conclusion and Discussion

This paper developed a methodology for a deep learning-based framework for real-time object detection and classification. The design combines efficient convolutional feature extraction, multi-scale fusion, anchor-free or low-overhead dense prediction, decoupled classification and localization, data augmentation, transfer learning, confidence filtering, and non-maximum suppression. The literature shows a clear evolution from accurate but computationally expensive two-stage detection toward increasingly efficient one-stage and anchor-free systems (Ren et al., 2017; Redmon et al., 2016; Tian et al., 2019; Ge et al., 2021). EfficientDet and modern YOLO variants further demonstrate that architecture scaling and training strategy are critical to the practical speed-accuracy balance (Tan et al., 2020; Bochkovskiy et al., 2020; Wang et al., 2022). The methodology is particularly suitable for applications that require continuous visual inference, including intelligent transportation, security monitoring, robotics, smart manufacturing, and edge vision. However, performance may degrade under heavy occlusion, poor illumination, extreme viewing angles, domain shift, and very small objects. Real-time claims are also hardware dependent; therefore, deployment studies should report device type, inference precision, batch size, image resolution, preprocessing, and post-processing costs. Future work may explore pruning, quantization, knowledge distillation, temporal feature reuse, and edge accelerators to reduce latency while preserving accuracy. Transformer-based contextual modules may also be investigated selectively where their accuracy gain justifies additional computational cost.

References

- Ren, S., He, K., Girshick, R., & Sun, J. (2017). Faster R-CNN: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6), 1137–1149. <https://doi.org/10.1109/TPAMI.2016.2577031>
- Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788. <https://doi.org/10.1109/CVPR.2016.91>
- Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). SSD: Single Shot MultiBox Detector. In *Computer Vision – ECCV 2016* (pp. 21–37). Springer. https://doi.org/10.1007/978-3-319-46448-0_2
- Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017a). Feature Pyramid Networks for object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 936–944. <https://doi.org/10.1109/CVPR.2017.106>
- Lin, T.-Y., Goyal, P., Girshick, R., He, K., & Dollár, P. (2017b). Focal Loss for dense object detection. *Proceedings of the IEEE International Conference on Computer Vision*, 2980–2988. <https://doi.org/10.1109/ICCV.2017.324>
- Tian, Z., Shen, C., Chen, H., & He, T. (2019). FCOS: Fully convolutional one-stage object detection. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 9627–9636. <https://doi.org/10.1109/ICCV.2019.00972>
- Tan, M., Pang, R., & Le, Q. V. (2020). EfficientDet: Scalable and efficient object detection. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10781–10790. <https://doi.org/10.1109/CVPR42600.2020.01079>
- Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., & Zagoruyko, S. (2020). End-to-End Object Detection with Transformers. In *Computer Vision – ECCV 2020* (pp. 213–229). Springer. https://doi.org/10.1007/978-3-030-58452-8_13
- Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). YOLOv4: Optimal speed and accuracy

of object detection. arXiv.
<https://doi.org/10.48550/arXiv.2004.10934>

Ge, Z., Liu, S., Wang, F., Li, Z., & Sun, J. (2021).
YOLOX: Exceeding YOLO series in 2021. arXiv.
<https://doi.org/10.48550/arXiv.2107.08430>

Wang, C.-Y., Bochkovskiy, A., & Liao, H.-Y. M.
(2022). YOLOv7: Trainable bag-of-freebies sets
new state-of-the-art for real-time object
detectors. arXiv.
<https://doi.org/10.48550/arXiv.2207.02696>