# ANALYSING THE IMPACT OF KERNEL SIZE AND PADDING OF LARGE SEPARABLE CONVOLUTION ON TRAINING TIME AND ACCURACY IN YOLOV12

**Thach Minh Hoang**
University of information technology
Thu Duc, Ho Chi Minh City
22520477@gm.uit.edu.vn

**ABSTRACT**

Yolov12 is a newly introduced model that incorporates attention mechanisms into its architecture, leveraging their strengths to achieve high processing speed and accuracy. While the model demonstrates impressive performance on datasets such as MS COCO 2017—which features multiple classes and typically large-sized target objects—the task addressed in this study presents a contrasting scenario: detecting human heads with only a single class. This task involves detecting small, densely packed human heads under heavy occlusion, which poses significant challenges for spatial localization and model robustness. Given the stark differences in data characteristics, this research focuses on analyzing the attention mechanisms within Yolov12, particularly examining how the kernel size and padding of the convolutional block used in position perciever affects model performance when applied to the crowd detection task on the CrowdHuman dataset. We explore the trade-offs between spatial precision and computational overhead introduced by different configurations of the position perciever module. The results indicate that the modified version of the model achieves faster training times compared to the original Yolov12, while still maintaining comparable accuracy across evaluation metrics such as AP (Average Precision), AR (Average Recall) and mAP (mean Average Precision). Furthermore, qualitative results suggest similar localization performance in heavily occluded scenarios, highlighting the benefits of tailoring architectural components to the nature of the detection task. This study contributes valuable insights into how Yolov12 can be effectively optimized for specialized object detection applications beyond general-purpose datasets. Our findings encourage further research into attention-driven design adaptations for detecting small-scale and densely located targets, offering practical implications for surveillance, crowd analysis, and other real-world applications where accuracy and efficiency are equally critical.

***K*eywords** Yolo · Object detection · Kernel size · padding

## 1 Introduction

The evolution of object detection has been driven by the need for higher accuracy, greater speed, and robustness in real-world settings. Early methods—such as sliding-window detectors (Lampert et al., 2008) and the Viola–Jones framework (Castrillón et al., 2011)—relied on handcrafted features like Haar-like descriptors, HOG and LBP, but suffered from heavy computational costs and poor generalization. The advent of deep learning, notably convolutional neural networks, enabled automatic hierarchical feature learning (Khan et al., 2020). Region-based models (R-CNN, Fast R-CNN, Faster R-CNN) greatly improved accuracy, while one-stage detectors such as SSD (Liu et al., 2016) fused proposal and classification for real-time performance. Finally, the YOLO series unified detection and classification in a single end-to-end network, dramatically reducing inference time without sacrificing competitive precision (Redmon et al., 2016). Among state-of-the-art approaches, the YOLO (You Only Look Once) family strikes a compelling balance between real-time throughput and detection accuracy.

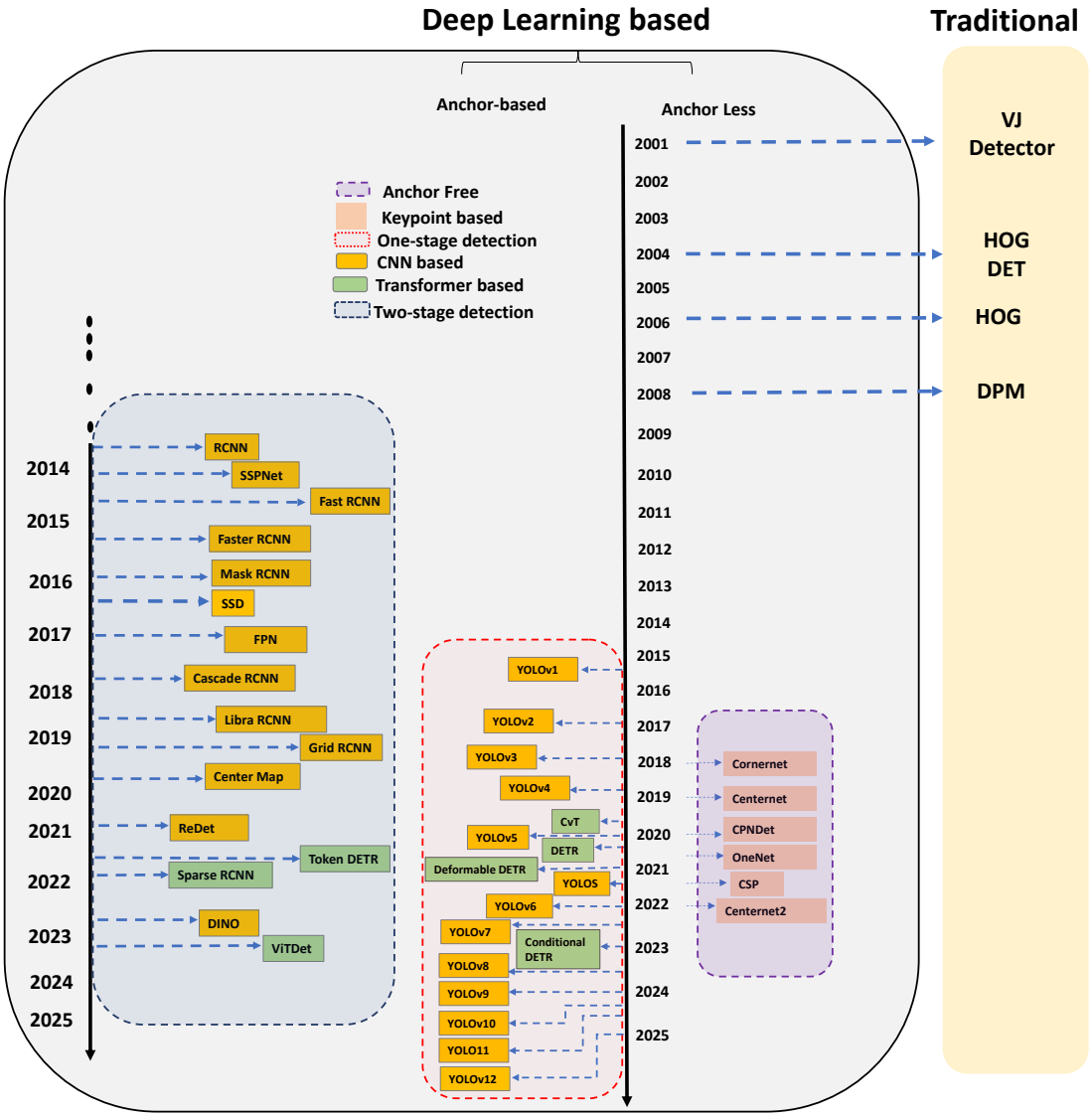


Figure 1: Timeline of the evolution of object detection models

The YOLO series set themselves apart from traditional multi-stage detectors by integrating all detection steps into a single network. **YOLOv1**[1] introduced a grid-based design that simultaneously predicts bounding boxes and class probabilities, pioneering real-time object detection. Building on this, **YOLOv2** or **YOLO9000**[2] raised input resolution and widened the range of detectable classes to boost accuracy and flexibility. **YOLOv3**[3] added multi-scale predictions and a deeper backbone to better handle small objects and complex features. **YOLOv4**[4] and **YOLOv5**[5] further enhanced feature representation and inference speed through CSP architectures and advanced data-augmentation techniques. Subsequent releases like **YOLOv6**[6] and **YOLOv7**[7] optimized the trade-off between precision and computational cost with innovations such as dynamic label assignment and refined architectural blocks. Collectively, these YOLO variants established a high bar for fast, accurate, end-to-end object detection.

**YOLOv8**[8] builds on its predecessors with an efficient, anchor-free design. Its backbone uses a refined CSP-based module for hierarchical feature extraction, while a streamlined Feature Pyramid Network in the neck enhances multi-scale feature representation. A decoupled head separates objectness, classification, and regression tasks, boosting accuracy without significant computational overhead. This simplified detection pipeline flexibly handles varying object sizes and aspect ratios, striking an optimal balance between speed and precision for real-time applications.

**YOLOv9**[9] incorporates two key innovations—Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN). PGI ensures robust, reliable gradient flow throughout the network, improving weight update efficiency and convergence. GELAN offers flexible, efficient multi-scale feature aggregation, mitigating information loss in deep layers and boosting detection accuracy, especially for small objects.

**YOLOv10**[10] builds on **YOLOv9**'s advances by eliminating the need for Non-Maximum Suppression (NMS) through a novel dual-assignment strategy, combining one-to-many and one-to-one label assignments during training to cut inference time. It introduces lightweight classification heads and spatial-channel–decoupled downsampling to minimize feature-extraction losses, while a rank-guided block design optimizes parameter utilization, keeping the network efficient for real-time detection

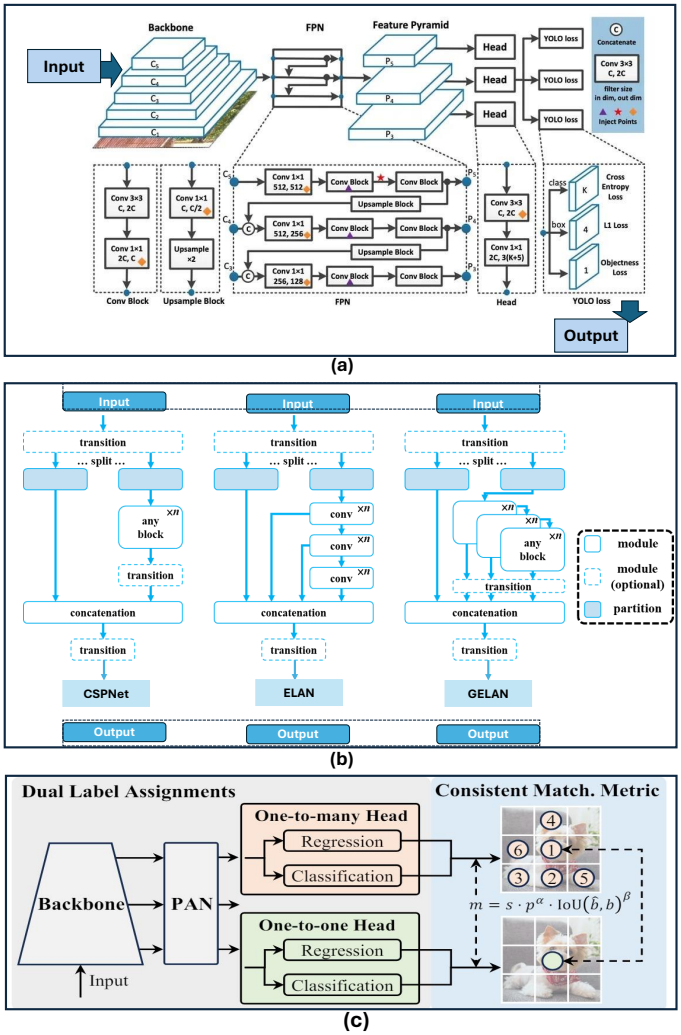


Figure 2: **YOLOv8, YOLOv9 and YOLOv10 Architecture Diagram** (a) YOLOv8 employs a CSP-based backbone, anchor-free decoupled head, and streamlined FPN for efficient multi-scale feature extraction. (b) YOLOv9 integrates programmable gradient information with GELAN for robust feature aggregation. (c) YOLOv10 adopts a dual assignment strategy,lightweight heads, and spatial-channel decoupled downsampling to further boost overall speed and accuracy significantly

**YOLOv11**[11] further refines the detection pipeline by emphasizing enhanced feature extraction and robust detection. It introduces the C3k2 block in place of the previous C2f, markedly improving gradient flow and computational efficiency. A Spatial Pyramid Pooling-Fast (SPPF) module captures richer multi-scale contextual information, while a Convolutional block with Parallel Spatial Attention (C2PSA) sharpens spatial feature representation. Together, these innovations bolster accuracy—especially under occlusion and in complex backgrounds.

The recently proposed **Yolov12**[12] further enhances this paradigm by embedding modern attention mechanisms—most notably its position perceiver modules—to better model spatial dependencies across feature maps.

While Yolov12 achieves top performance on general-purpose benchmarks like MS COCO 2017[13], which contains multiple object categories and relatively large targets, its off-the-shelf configuration may not be ideal for specialized scenarios. In densely crowded scenes, human heads are small, tightly clustered, and highly occluded, creating unique challenges for localization and false-positive suppression. Although attention-driven detectors have shown promise in small-object tasks, the specific role of kernel size and padding in the position perceiver remains unexplored.

This report conduct a focused study on how varying the kernel and padding sizes in the position perceiver affects performance on the CrowdHuman dataset[14]. This report experiments compare a 7×7 kernel to a 5×5 kernel, yielding a training-time reduction of approximately 0.7% without any drop in AP, AR, or mAP. Qualitative results also confirm robust localization under heavy occlusion.

The rest of this paper is structured as follows: Section 2 will mention about related work; Section 3 reviews the CrowdHuman dataset; Section 4 details the Yolov12 architecture and our position-perceiver modifications; Section 5 describes our experimental setup and evaluation metrics; Section 6 presents quantitative and qualitative results, and Section 7 concludes.

## 2 Related Work

Object detection represents a fundamental capability in computer vision systems, enabling automated identification and localization of objects within multispectral imagery. Recent advancements in deep learning, particularly convolutional neural network (CNN) architectures, have significantly enhanced detection performance while reducing inference latency through optimized model structures and loss functions. Prominent CNN-based frameworks include the YOLO series[1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12], RetinaNet[15], Faster R-CNN[16], and SSD[17], which serve as versatile foundations for computer vision applications. These models can be deployed as pre-trained solutions, fine-tuned for domain-specific datasets, or architecturally modified to address specialized detection challenges. The recently introduced YOLOv12 framework[12] has demonstrated state-of-the-art performance across multiple benchmarks, subsequently inspiring numerous architectural investigations and extensions [18, 19] that explore its novel design paradigms.

The CrowdHuman dataset has emerged as a foundational benchmark for advancing human head detection research, providing critical large-scale annotations specifically designed for crowded scenarios. This domain presents distinct challenges characterized by extreme density variations, frequent partial occlusions, and significant scale diversity. Substantial research efforts have consequently focused on developing robust detection methods for these demanding conditions, with recent approaches achieving impressive performance metrics as documented in [20, 21].

## 3 Dataset Overview: CrowdHuman

In this study, we utilize the CrowdHuman dataset[14], a large-scale benchmark specifically designed for human detection in highly crowded scenes. Unlike traditional datasets, CrowdHuman features dense pedestrian populations with extensive occlusion, making it ideal for training and evaluating models that must perform robustly in complex visual environments. The dataset includes rich annotations for each person: full body bounding boxes, visible-region bounding boxes, and head bounding boxes. Leverage multi-level ananlysis of detecting performance. It comprehensive scope and challenging scenarios make it a valuable resource for advancing research in crowded scene detection.

The images in datasets were collected from the Internet using approximately 150 different keywords to ensure diversity in scenes, activities, and viewpoints. The dataset is divided into three subsets: 15,000 – 4,370 – 5,000 for training, validation, and test set respectively, with averaging about 22.6 persons per image. The original CrowdHuman dataset does not include any preprocessing steps or data augmentations. The original dataset annotation file is in odgt format (with JSON lines).

In this research, we focus exclusively on head bounding boxes to evaluate model performance on small, dense, and occluded objects. The original dataset in COCO format was converted to YOLO format, followed by selective extraction of head annotations. We implemented rigorous quality filtering based on these annotation attributes:

- **Occlusion (binary)**: 1 indicates partial visibility, 0 indicates fully visible heads
- **Unsure (binary)**: 1 flags questionable annotations (annotator unsure about head presence), 0 indicates confident identification
- **Ignore (binary)**: 1 marks invalid boxes (excluded from training/testing), 0 indicates valid annotations

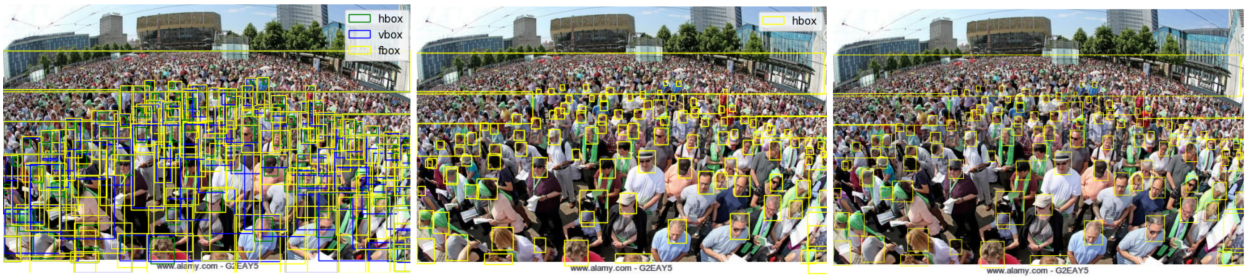


Figure 3: *Illustration of data processing: (a) original images, (b) headbox annotations before filtering, and (c) retained samples after applying annotation quality constraints (Ignore = 0, Unsure = 0).*

Our data selection protocol retains only high-confidence annotations meeting the criteria:

$$\text{Retained Samples} = \{\text{bbox} \mid \text{Unsure} = 0 \wedge \text{Ignore} = 0\}$$

This approach deliberately preserves occluded heads (Occlusion $= 1$) as valid detection targets, since they represent realistic partial visibility scenarios, while eliminating ambiguous or low-quality annotations that could compromise training integrity.

Furthermore, when converting bounding boxes to the YOLO format, instances where an object extends partially beyond the image boundary can result in normalized coordinates outside the valid range [0, 1]. These invalid coordinates (negative or exceeding 1.0) are subsequently discarded by the YOLO model during training, leading to acceptable data loss. To preserve data richness, a common solution involves clamping these coordinates to the [0, 1] range prior to training.

## 4 Yolov12's Architecture

The design of Yolov12 can be divided into three main components: **the backbone**, which extracts and processes multi-scale features; **the neck**, which aggregates and refines those features; and the **head**, which generates the final predictions.

### 4.1 Backbone

The backbone plays a critical role in transforming raw images into multiscale feature maps, providing foundational representations for downstream detection tasks. A key innovation in the backbone is the Residual Efficient Layer Aggregation Network (R-ELAN), which integrates deeper convolutional layers with strategically placed residual connections. This design mitigates gradient bottlenecks and promotes feature reuse, enhancing the model's ability to capture fine-grained object details across diverse sizes and shapes.

#### 4.1.1 Advanced Convolutional Blocks

Yolov12 employs a new convolutional block class emphasising lightweight operations and higher parallelisation. These blocks utilise a series of smaller kernels, represented generically as:

$$F_{\text{out}} = \sum_{i=1}^{n} W_i * F_{\text{in}} + b_i, \tag{1}$$

where $F_{\text{out}}$ is the output feature map, $W_i$ are the convolutional filters, $F_{\text{in}}$ is the input feature map, and $b_i$ is the bias term. By distributing the computation across multiple small convolutions instead of fewer large ones, Yolov12 achieves faster processing without compromising feature extraction quality.

#### 4.1.2 Residual Efficient Layer Aggregation Networks

In previous version, efficient layer aggregation networks (ELAN)[7] are designed to improve feature aggregation. As show in Figure 4a splits the output of a transition layer (a $1 \times 1$ convolution), processes one split through multiple

modules, then concatenates the all the outputs and applies another transition layer (a 1 × 1 convolution) to align dimension. However this architecture can introduce instability. Its design causes gradient blocking and lacks residual connections from input to output.

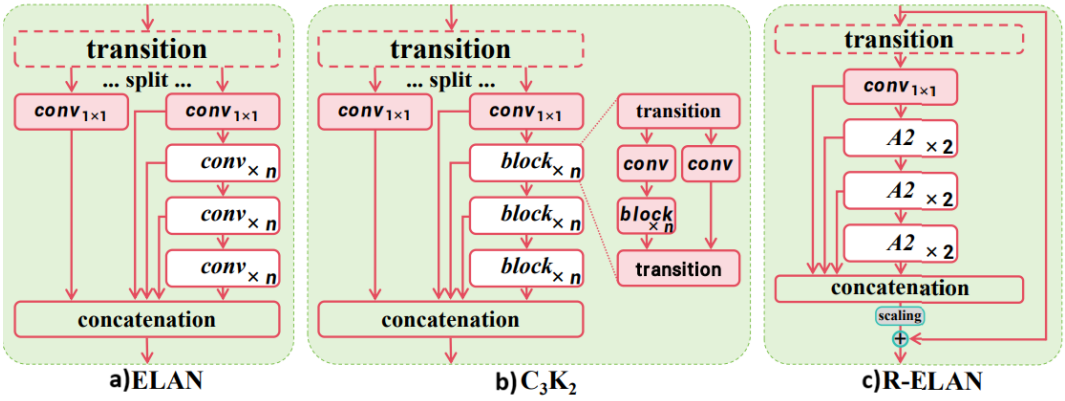


Figure 4: The architecture comparison with popular modules

The authors propose a new aggregation approach[12] as shown in Figure 4c. The proposed design applies a transition layer to adjust the channel dimensions and produces a single feature map. This feature map is then processed through subsequent blocks followed by the concatenation, forming a bottleneck structure. This approach not only preserves the original feature integration capability, but also reduces both computational cost and parameter / memory usage.

#### 4.1.3 Backbone Modifications

As shown is Figure 5a, the Yolo11 has the backbone with convolutional blocks and C3K2 as in Figure 4b for extracting features from image. While in Yolov12, as show in figure 5b, the authors remove the design of stacking three blocks in the last stage of the backbone, which is present in recent versions[8, 9, 10, 11]. Instead, they retained only a single R-ELAN block, reducing the total number of blocks and contributing to optimization, inherit the first two stages of the backbone from YOLOv11 and do not use the proposed R-ELAN. Additionally, the authors proposed to modify several default configurations in the vanilla attention mechanism to better suit the YOLO system. These modifications include adjusting the MLP ratio from 4 to 1.2 (or 2 for the N- / S- / M-scale models) is used to better allocate computational resources for better performance, adopting `nn.Conv2d+BN` instead of `nn.Linear+LN` to fully exploit the efficiency of convolution operators, removing positional encoding, and introduce a large separable convolution (7 × 7) (namely position perceiver) to help the area attention perceive position information. [22, 23]

### 4.2 Neck

Functioning as a conduit between the backbone and head, the neck in YOLOv12 aggregates and refines multi-scale features. One of its key innovations is an area attention mechanism accelerated by FlashAttention, which enhances the model's focus on critical regions in cluttered scenes. Mathematically, this can be interpreted as a segmented attention operation:

$$\text{Attention}(Q, K, V) = \text{softmax}\Big(\frac{QK^T}{\sqrt{d_k}}\Big)\, V, \tag{2}$$

where $Q, K, V$ are query, key, and value matrices, and $d_k$ is the dimensionality of the key. By segmenting feature maps into areas and applying fast attention routines, YOLOv12 reduces memory transfers and computational overhead, enabling real-time inference even at higher input resolutions.

As shown in figure 6, Yolov12 further enhances its architecture by replacing all C3k2 blocks in the neck with a Residual Efficient Layer Aggregation Network (R-ELAN) modules and Area Attention mechanism, which is being accelerated by FlashAttention, both of which are optimized for performance and accuracy.

### 4.3 Head

The head of YOLOv12 transforms the refined feature maps from the neck into final predictions, generating bounding box coordinates and classification scores. Key improvements include streamlined multi-scale detection pathways,

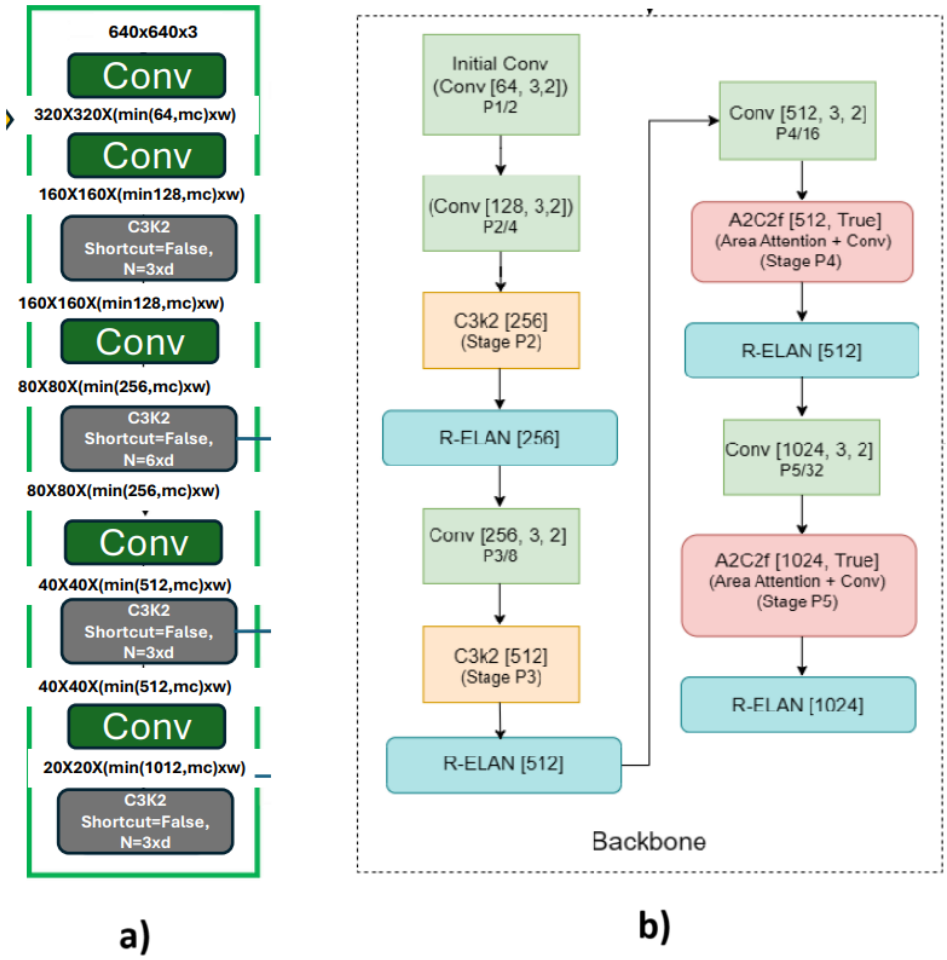


Figure 5: *a) Yolo11's Backbone b) Yolov12's Backbone*

and specialised loss functions that better balance localisation and classification objectives. For example, a typical YOLO-style loss might be extended to incorporate new attention or confidence terms:

$$\mathcal{L} = \lambda_{\text{coord}} \sum (\hat{x} - x)^2 + (\hat{y} - y)^2 \; + \; \lambda_{\text{obj}} \sum (\hat{C} - C)^2 \; + \; \ldots \tag{3}$$

where $\hat{x}, \hat{y}, \hat{C}$ denote predicted bounding box coordinates and confidence, respectively. Such refinements further enhance YOLOv12's performance in real-time applications.

YOLOv12 achieves a significant architectural evolution, blending innovative backbone elements, advanced attention mechanisms, and refined prediction modules. Together, these components set new standards for speed and accuracy in object detection while seamlessly extending to more specialised tasks such as instance segmentation[24]

## 5 Experiment

### 5.1 Hypothesis and Expectation

As referenced in Section 4.1.3, Yolov12 introduces a large separable convolution with a kernel size of 7×7 and padding size of 3×3, termed the position perceiver. This module enhances area attention mechanisms by providing spatial feature information to the attention module, enabling the model to distinguish features at different spatial locations while simultaneously reducing computational costs.

Another hypothesis posits that for convolutional blocks maintaining identical input channel dimensions, output channel dimensions, and stride values, those employing larger kernel sizes with correspondingly increased padding

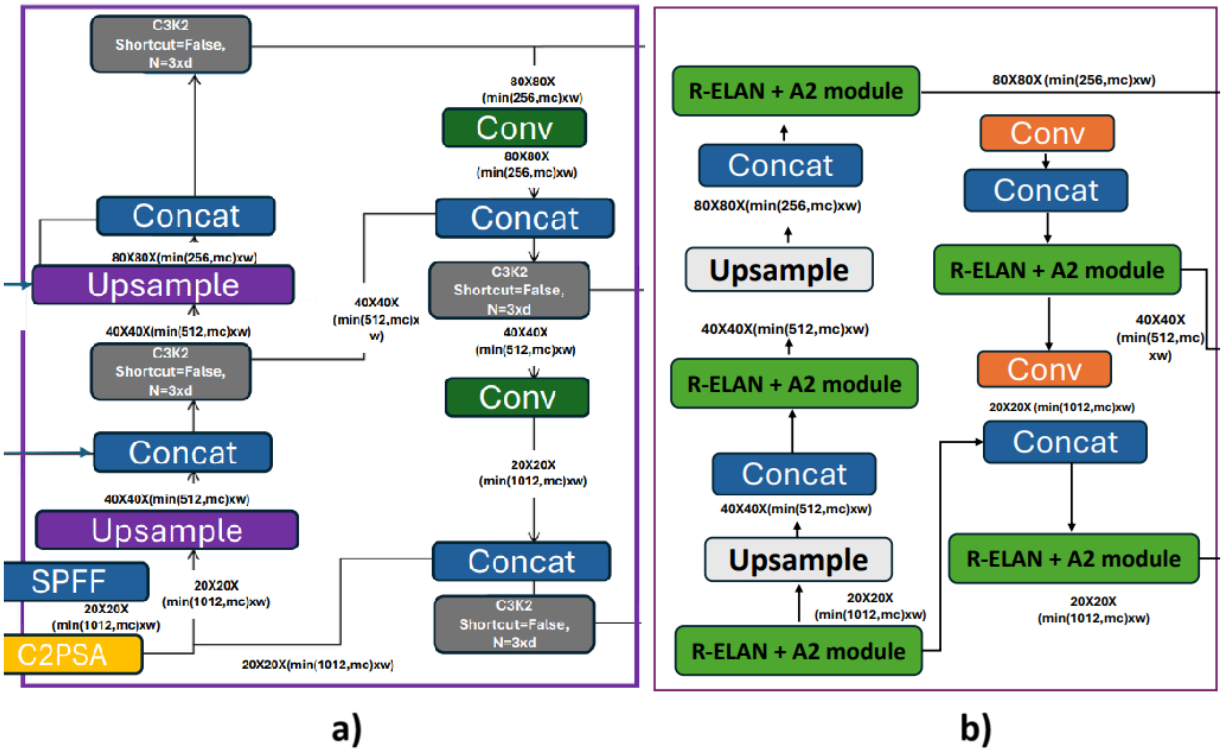


Figure 6: *a) Yolo11's Neck b) Yolov12's Neck*

configurations will exhibit significantly higher computational complexity and longer execution times. This increased computational burden stems fundamentally from the quadratic relationship between kernel size and the number of operations required.

**Theoretical Verification:** This hypothesis is theoretically valid based on convolution operation mathematics:

- Floating Point Operations (FLOPs) for convolution are calculated as:

$$\text{FLOPs} = H_{\text{out}} \times W_{\text{out}} \times C_{\text{in}} \times C_{\text{out}} \times K_h \times K_w \tag{4}$$

  **Where:** $H_{\text{out}}, W_{\text{out}}$: Output spatial dimensions; $C_{\text{in}}, C_{\text{out}}$: Input/output channels; $K_h, K_w$: Kernel dimensions

- When kernel size increases (e.g., from $3 \times 3$ to $7 \times 7$), the $K_{\text{h}} \times K_{\text{w}}$ term grows quadratically:
  - $3 \times 3 \rightarrow 9$ multiplication-addition operations per output pixel
  - $7 \times 7 \rightarrow 49$ operations per output pixel (5.4× increase)
- Padding Relationship: Larger kernels require proportionally increased padding (e.g., padding = (k-1)/2) to maintain output dimensions, but padding itself doesn't directly increase computations. However, it enables the larger kernel to operate across more input pixels, indirectly contributing to the computational load.

Building upon these theoretical foundations, this work empirically examines the position perceiver module with varying kernel and padding configurations in large separable convolutions across multiple Yolov12 model scales. Given the characteristics of our target dataset—which features small-scale, densely clustered, and partially occluded human heads—we conduct comprehensive experiments with different kernel sizes to evaluate critical performance metrics: training time, inference time, and detection accuracy measured through Recall, Precision, and mean Average Precision (mAP). We hypothesize that configurations with smaller kernel sizes will achieve significantly faster training and inference times, though potentially at the cost of marginal reductions in Recall, Precision, and mAP performance due to the spatial information trade-off.

### 5.2 Experimental Setup

We validate our proposed method on the custom CrowdHuman dataset described in Section 3. While the YOLOv12 family comprises five variants (N, S, M, L, X), this study focuses specifically on YOLOv12-N, YOLOv12-S and YOLOv12-L architectures. All models undergo initial training for 50 epochs using SGD optimization with a learning rate of 0.01, followed by extended training of 100 additional epochs for the YOLOv12-S variant. Experiments are conducted on Kaggle using dual T4 GPUs for consistent training, testing, and evaluation.

Beyond benchmarking the baseline YOLOv12 implementation featuring the original position perceiver configuration (7×7 kernel with 3×3 padding), we introduce and evaluate two modified versions that maintain identical stride and channel dimensions while varying spatial parameters as detailed in Section 5.1:

- **Baseline:** 7×7 kernel with 3×3 padding (original configuration)
- **Variant A:** 9x9 kernel with 4x4 padding
- **Variant B:** 5×5 kernel with 2×2 padding
- **Variant C:** 3×3 kernel with 1×1 padding

*Note: Architectural variations affect implementation scale—modifications apply to 8 convolutional blocks in -N, -S, and -M variants, and 16 blocks in -L and -X variants.* This controlled experimental design enables systematic comparison of computational efficiency and detection performance across spatial perception scales.

### 5.3 Result

| Model | Kernel | Padding | TrainingTime | InferenceTime | PostProcessing | Precision | Recall | mAP |
|---|---|---|---|---|---|---|---|---|
| Yolov12-N | 9x9 | 4x4 | 4.509(h) | **2.3ms** | 1.4ms | 0.814 | 0.495 | 0.575 |
| Yolov12-N | 7x7 | 3x3 | 4.533(h) | 2.6ms | 1.4ms | 0.814 | 0.495 | 0.575 |
| Yolov12-N | 5x5 | 2x2 | **4.430(h)** | 2.5ms | **1.3ms** | 0.814 | 0.495 | 0.575 |
| Yolov12-N | 3x3 | 1x1 | 4.535(h) | 2.6ms | 1.4ms | 0.814 | 0.495 | 0.575 |
| Yolov12-S | 9x9 | 4x4 | 4.810(h) | **4.4ms** | 1.3ms | 0.826 | 0.534 | 0.619 |
| Yolov12-S | 7x7 | 3x3 | 4.777(h) | 4.6ms | **1.3ms** | 0.826 | 0.534 | 0.619 |
| Yolov12-S | 5x5 | 2x2 | **4.721(h)** | 4.5ms | 1.5ms | 0.826 | 0.534 | 0.619 |
| Yolov12-S | 3x3 | 1x1 | 4.820(h) | 4.6ms | 1.4ms | 0.826 | 0.534 | 0.619 |
| Yolov12-L | 9x9 | 4x4 | 11.374(h) | 16.9ms | 1.5ms | 0.835 | 0.56 | 0.65 |
| Yolov12-L | 7x7 | 3x3 | **10.619(h)** | **15.2ms** | **1.4ms** | 0.835 | 0.56 | 0.65 |
| Yolov12-L | 5x5 | 2x2 | 11.103(h) | 16.4ms | 1.6ms | 0.835 | 0.56 | 0.65 |
| Yolov12-L | 3x3 | 1x1 | 11.669(h) | 17.0ms | 1.7ms | 0.835 | 0.56 | 0.65 |

Table 1: Training Yolo Variant with different modification in 50 epochs

| Model | Kernel | Padding | TrainingTime | InferenceTime | PostProcessing | Precision | Recall | mAP |
|---|---|---|---|---|---|---|---|---|
| Yolov12-S | 7x7 | 3x3 | 4.777+9.515(h) | 4.5ms | 1.2ms | 0.823 | 0.544 | 0.627 |
| Yolov12-S | 5x5 | 2x2 | **4.721+9.473(h)** | 4.3ms | 1.2ms | 0.823 | 0.544 | 0.627 |
| Yolov12-S | 3x3 | 1x1 | 4.820+9.549(h) | 4.5ms | 1.2ms | 0.823 | 0.544 | 0.627 |

Table 2: Training Yolo12-S Variant with different modification in 150 epochs

The experimental results presented in Table 5.3 did not fully align with our initial predictions in Section 5.1. Specifically, for the -S and -N variants, the performance of configurations such as the 3×3 kernel with 1×1 padding and the 9×9 kernel with 4×4 padding contradicted expectations. Contrary to theoretical assumptions about computational complexity in Section 5.1, the 9×9 kernel unexpectedly demonstrated shorter training times compared to both the 3×3 kernel and 7×7 baseline (e.g., 4.509h vs. 4.535h for -N), while simultaneously achieving the fastest inference times (2.3ms for -N, 4.4ms for -S) without compromising accuracy relative to other configurations.

On a positive note, certain results validated our hypothesis. The 5×5 kernel with 2×2 padding exhibited reduced training times across variants, with a modest improvement for -N (4.430h vs. baseline 4.533h) and a more pronounced reduction for -S (4.721h vs. baseline 4.777h). However, this trend reversed in the -L variant, where the 9×9 kernel required 11.374

hours of training (longer than the 7×7 baseline's 10.619 hours) with slower inference (16.9ms vs. 15.2ms), and both 5×5 and 3×3 kernels incurred longer training durations (11.103h and 11.669h, respectively).

## 6 Conclusion

This study provides deep insights into how large separable convolutions impact training time, inference latency, post-processing efficiency, and overall accuracy across multiple YOLOv12 variants. Although the results deviated from initial expectations, our work delivers valuable empirical evidence on architectural parameter adjustments, laying the groundwork for deeper investigations into the YOLOv12 architecture. We recommend three key research directions: (1) validating these modifications on multi-class datasets, (2) exploring optimal MLP ratios, and (3) replacing attention mechanisms to simultaneously enhance accuracy and minimize latency.

## Acknowledgments

We acknowledge the following contributions to this research: Ultralytics for their YOLO model implementation [URL: https://docs.ultralytics.com/vi/models], arXiv for providing open-access scholarly resources, and OpenAI (ChatGPT) along with DeepSeek for their AI-assisted implementation support. We also express special gratitude to Kaggle for providing complimentary dual T4 GPU resources that enabled efficient model training and evaluation.

## References


[1] J Redmon. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[2] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[3] Joseph Redmon. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[4] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[5] Glenn Jocher, K Nishimura, T Mineeva, and RJAM Vilariño. yolov5. *https://github.com/ultralytics/yolov5/tree*, 2, 2020.

[6] Chuyi Li, Lulu Li, Yifei Geng, Hongliang Jiang, Meng Cheng, Bo Zhang, Zaidan Ke, Xiaoming Xu, and Xiangxiang Chu. Yolov6 v3. 0: A full-scale reloading. *arXiv preprint arXiv:2301.05586*, 2023.

[7] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7464–7475, 2023.

[8] Jocher Glenn. Yolov8. *https://github.com/ultralytics/ultralytics/tree/main*, 2023.

[9] Chien-Yao Wang, I-Hau Yeh, and Hong-Yuan Mark Liao. Yolov9: Learning what you want to learn using programmable gradient information. *arXiv preprint arXiv:2402.13616*, 2024.

[10] Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, and Guiguang Ding. Yolov10: Real-time end-to-end object detection. *arXiv preprint arXiv:2405.14458*, 2024.

[11] Glenn Jocher. yolov11. *https://github.com/ultralytics*, 2024.

[12] Yunjie Tian, Qixiang Ye, and David Doermann. Yolov12: Attention-centric real-time object detectors, 2025.

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[14] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. Crowdhuman: A benchmark for detecting human in a crowd, 2018.

[15] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection, 2018.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[17] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector*, page 21–37. Springer International Publishing, 2016.

[18] Yu-Hsi Chen. Strong baseline: Multi-uav tracking via yolov12 with bot-sort-reid, 2025.

[19] Ranjan Sapkota and Manoj Karkee. Improved yolov12 with llm-generated synthetic data for enhanced apple detection and benchmarking against yolov11 and yolov10, 2025.

[20] Chenyang Zhao, Jia Wan, and Antoni B. Chan. Density-based object detection in crowded scenes, 2025.

[21] Yueming Huang, Chenrui Ma, Hao Zhou, Hao Wu, and Guowu Yuan. Dense object detection based on de-homogenized queries. *Electronics*, 13(12):2312, June 2024.

[22] Nidhal Jegham, Chan Young Koh, Marwan Abdelatti, and Abdeltawab Hendawi. Yolo evolution: A comprehensive benchmark and architectural review of yolov12, yolo11, and their previous versions, 2025.

[23] Ranjan Sapkota, Zhichao Meng, Martin Churuvija, Xiaoqiang Du, Zenghong Ma, and Manoj Karkee. Comprehensive performance evaluation of yolov12, yolo11, yolov10, yolov9 and yolov8 on detecting and counting fruitlet in complex orchard environments, 2025.

[24] Mujadded Al Rabbani Alif and Muhammad Hussain. Yolov12: A breakdown of the key architectural features, 2025.