International Journal of Innovative Research in Computer Science and Technology (IJIRCST)
ISSN(Online): 2347-5552, Volume-12, Issue-3, May 2024
https://doi.org/10.55524/ijircst.2024.12.3.11
Article ID IRP-IRP-1517, Pages 57-74
www.ijircst.org

Optimizing Real-Time Object Detection- A Comparison of YOLO Models

Pravek Sharma[1], Dr. Rajesh Tyagi[2], and Dr. Priyanka Dubey[3]

[1] M.Tech Scholar, Department of Computer Science & Engineering, Amity School of Engineering and Technology, Amity University, Gurugram, Haryana, India
[2] Professor, Department of Computer Science & Engineering, Amity School of Engineering and Technology, Amity University, Gurugram, Haryana, India
[3] Assistant Professor, Department of Computer Science & Engineering, Amity School of Engineering and Technology, Amity University, Gurugram, Haryana, India

Correspondence should be addressed to Pravek Sharma ; pravek23@gmail.com

Received 09 April 2024; Revised 23 April 2024; Accepted 6 May 2024

ABSTRACT- Gun and weapon détection plays a crucial role in security, surveillance, and law enforcement. This study conducts a comprehensive comparison of all available YOLO (You Only Look Once) models for their effectiveness in weapon detection. We train YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, and YOLOv8 on a custom dataset of 16,000 images containing guns, knives, and heavy weapons. Each model is evaluated on a validation set of 1,400 images, with mAP (mean average precision) as the primary performance metric. This extensive comparative analysis identifies the best performing YOLO variant for gun and weapon detection, providing valuable insights into the strengths and weaknesses of each model for this specific task.

KEYWORDS-Weapon Detection; YOLO models; Security; Deep Learning; Learning Rate

I. INTRODUCTION

The field of object detection has witnessed remarkable progress in recent years, with advancements in algorithms and models leading to astonishing results. From the early days of detecting basic objects like cats and dogs to the current capabilities of identifying high-speed cars and traffic in videos, the evolution has been truly remarkable. At the forefront of this progress stands the YOLO (You Only Look Once) family of models, with YOLOv8 representing the latest state-of-the-art. This model boasts significant improvements in both accuracy and efficiency, making it a powerful tool for various object detection tasks.

Weapon detection remains a crucial aspect of security, surveillance, and law enforcement. Traditional methods often fall short in their effectiveness, highlighting the need for more reliable and efficient solutions. Deep learning-based object detection models, like YOLOv8, offer a promising avenue for overcoming these limitations. Gun and weapon detection pose a unique challenge due to the diverse range of weapon types and their often-complex appearances. However, deep learning models have achieved significant success in this domain, demonstrating their potential to revolutionize weapon detection capabilities.

This research delves into the effectiveness of YOLO variants for gun and weapon detection. We conduct a comprehensive comparative study, analysing the performance of all available YOLO models (YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, and YOLOv8) on a custom dataset of 16,000 images encompassing various guns, knives, and heavy weapons. By evaluating each model's performance using the mean average precision (mAP) metric, we aim to identify the best performing YOLO variant for this specific task. Furthermore, this study provides valuable insights into the relative strengths and weaknesses of each model in the context of gun and weapon detection, contributing to the ongoing development and optimization of object detection models for critical security applications.

This research builds upon the foundation of previous advancements in object detection and leverages the latest advancements in the YOLO model architecture to explore its potential for enhancing weapon detection capabilities, ultimately contributing to improved public safety.

The present article is structured as:

- Section I is Introduction,
- Section II is The literature review.
- Section III Illustrates the methodology.
- Section IV Comparative study of variants.
- Section V Describes results and discussion.
- Section VI Summarizes the conclusions and gives ideas for future work.

II. LITERATURE REVIEW

Object detection performance is measured in both detection accuracy and inference time. The detection accuracy in two stage detectors is better than single stage detectors. In 2015, the real-time object detection system YOLO was published, and it rapidly grew its iterations, with the newest release, YOLOv8 in January 2023 [1]. The limitations of traditional

techniques are apparent when handling complex scenarios involving various factors, such as diverse lighting, weather conditions, and occlusions. In recent years, the accuracy of object detection methods has undergone significant advancements, and the field of Deep Learning has been the primary driving force behind this transformation [2]. the review highlights key architectural innovations introduced in each variant, shedding light on the incremental refinements. The review includes benchmarked performance metrics, offering a quantitative measure of each variant's capabilities [3]. Deep learning methods like You Only Look Once (YOLO) and MobileNet have recently significantly aided real-time biometric recognition [4]. Quantitatively, YOLOv5 models generally outperform YOLOv8, with the YOLOv5s variant achieving the highest scores across all metrics. However, visual assessments reveal that YOLOv8 models exhibit similar, and in some cases superior capabilities [5]. We observed that while models like YOLOv8 show improvements in detecting specific actions like steering wheel and IVI operations, challenges remain in accurately identifying other crucial activities such as cell phone usage and general hand movements [6]. YOLO and its different variants that detect and track perception of people. This research uses detection and tracking the differently abled people with paralysis, limb deficiency Amelia, or amputee [7]. Intra-model analysis is conducted for each of the five YOLO versions, optimizing parameters such as the optimizer, batch size, and learning rate based on sensitivity and training time. YOLOv3, YOLOv4, and YOLOv7 demonstrate exceptional sensitivity, reaching 100%. Comparative analysis against state-of-the-art models highlights their superiority. YOLOv7, utilizing the SGD optimizer, a batch size of 64, and a learning rate of 0.01, achieves outstanding overall performance [8]. YOLO models outperform the commonly used two-stage detection algorithm, Faster R-CNN, in fracture detection. Additionally, compound-scaled variants of each YOLO model were compared, with YOLOv8 m demonstrating the highest fracture detection sensitivity [9]. There are many techniques used to identify and locate objects in images and videos, but YOLO is the best solution for them. The YOLO algorithm performs generalized representation of objects more efficiently without loss of accuracy than other object detection models [10]. Computer Vision is a field of study that helps to develop techniques to identify images and displays. It has various features like image recognition, object detection and image creation, etc. Object detection is used for face detection, vehicle detection, web images, and safety systems. Its algorithms are Region-based Convolutional Neural Networks (RCNN), Faster-RCNN and You Only Look Once Method (YOLO) that have shown state-of-the-art performance. Of these, YOLO is better in speed compared to accuracy. It has efficient object detection without compromising on performance [11]. Transfer learning has emerged as one of the go-to methods to adapt models well on a small data set. Deep learning models have outperformed the traditional machine learning models, especially in the computer vision field [12]. YOLOv7 exhibits comparatively better performance as it dynamically learns the class labels through its soft labelling mechanism. [13]. object detectors are classified into two categories viz. two stage and single stage object detectors. Two stage detectors mainly focus on selective region proposals strategy via complex architecture; however, single stage detectors focus on all the spatial region proposals for the possible detection of objects via relatively simpler architecture in one shot [14]. YOLOv5 emerged within this paradigm as the standards of time efficiency. YOLOv8 is distinguished as the best possible level of performance in contrast, earning a remarkable mean Average Precision [15]. The yolov4 model is higher than the yolov3 model in terms of mAP values, but slightly lower in terms of speed, while the yolov5 series model is better than the yolov3 and yolov4 models both in terms of mAP values and speed [16]. For an object detection system to run in real-time, it is vital to minimize the computational costs while maintaining acceptably high accuracy. In a Convolutional Neural Network (CNN) there is a direct correlation between the accuracy and the computational cost incurred by increasing the number of layers [17]. The YOLO (You Only Look Once) technique has emerged as a prominent technique for various object detection tasks owing to its impressive balance between speed and precision [18]. The YOLOv5 model had high precision, recall, F1-score, and mean average precision (mAP) for training with both the original dataset and the proposed augmentation dataset. While the YOLOv8s model had the highest precision, recall, F1-score, and mAP only training with the proposed augmentation [19]. The success of these YOLO models is often attributed to their use of guidance techniques, such as expertly tailored deeper backbone and meticulously crafted detector head, which provides effective mechanisms to tradeoff between accuracy and efficiency [20].

III. METHODOLOGY

This research delves into the potential of various YOLO variants for gun and weapon detection. We conduct a comprehensive comparative study by training and evaluating all available YOLO models (YOLOv1, YOLOv2, YOLOv3, YOLOv4, YOLOv5, YOLOv6, YOLOv7, and YOLOv8) on a custom dataset of 16,000 images containing guns, knives, and heavy weapons. This dataset is meticulously curated from diverse sources, including the internet, public domain datasets, and private datasets. Each image is meticulously annotated to ensure accurate ground truth labels for weapons, enabling a robust comparison of each YOLO model's performance in weapon detection.

Here's a breakdown of the methodology for each YOLO variant:

- Data Acquisition: All YOLO models are trained on the same custom dataset of 16,000 images containing guns, knives, and heavy weapons. This ensures a consistent and controlled environment for evaluating the models' performance.
- Data Preprocessing: Each image within the dataset undergoes meticulous annotation to ensure accurate ground truth labels for weapons. This annotation process is crucial for training the models effectively and evaluating their detection accuracy.
- Model Training: Each YOLO variant undergoes individual training on the prepared dataset. This allows us to observe how each model's architecture and training parameters influence its performance in weapon detection.

- Evaluation: After training, each YOLO model is evaluated on the same validation set of images within the dataset. This allows for a direct comparison of their performance in terms of weapon detection accuracy using metrics like mean average precision (mAP).

By analysing the performance of each YOLO variant under these controlled conditions, this study aims to identify the most effective model for gun and weapon detection. This comparative analysis provides valuable insights into the strengths and weaknesses of each YOLO variant in this specific context, contributing to the ongoing development and optimization of object detection models for critical security applications.

This model is developed in four major steps:

a. Extracting Data: To train the YOLO models effectively for weapon detection, a diverse dataset encompassing various viewpoints, angles, and gun types was crucial. This data was meticulously compiled from trustworthy sources like Robo-flow and Kaggle, ultimately resulting in a dataset of 16,000 images for the research project.

b. Data Cleaning and Pre-processing: To ensure data quality, the cleaning process addressed inconsistencies and abnormalities in both structured and semi-structured data, including outliers, very large files, empty images, and blurry images.

c. Feature Selection: The YOLO model analyzes relationships between previously stored data and the predicted bounding boxes within an image. This analysis helps pinpoint crucial features that distinguish objects like knives, guns, or other harmful weapons, ultimately enabling their accurate detection.

d. Model Evaluation Metrics: The different types of loss encountered during the training process are visualized in Figure 1. Figure 2 presents a confusion matrix, which provides insights into how the model classified the objects as knives, heavy weapons, guns, or background. Additionally, Figure 3 shows a normalized version of the confusion matrix for the same categories, allowing for a clearer comparison of classification performance. These measures are depicted using the v8 model of YOLO.

epoch	train/box_loss	train/cls_loss	train/dfl_loss
1	1.5014	1.9598	1.7615
2	1.4924	1.8569	1.7768
3	1.3988	1.6986	1.6964
4	1.3027	1.5225	1.622
5	1.2405	1.432	1.5758

Figure 1: Types of Loss while Training

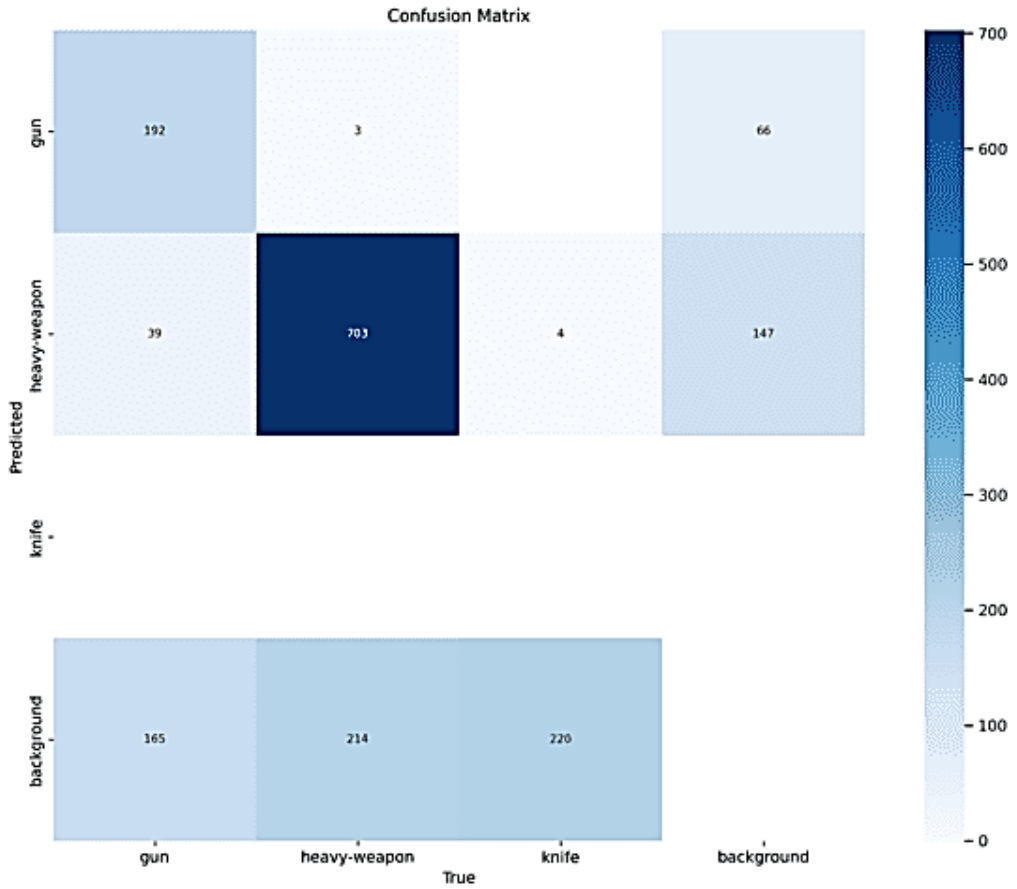


Figure 2: Confusion Matrix

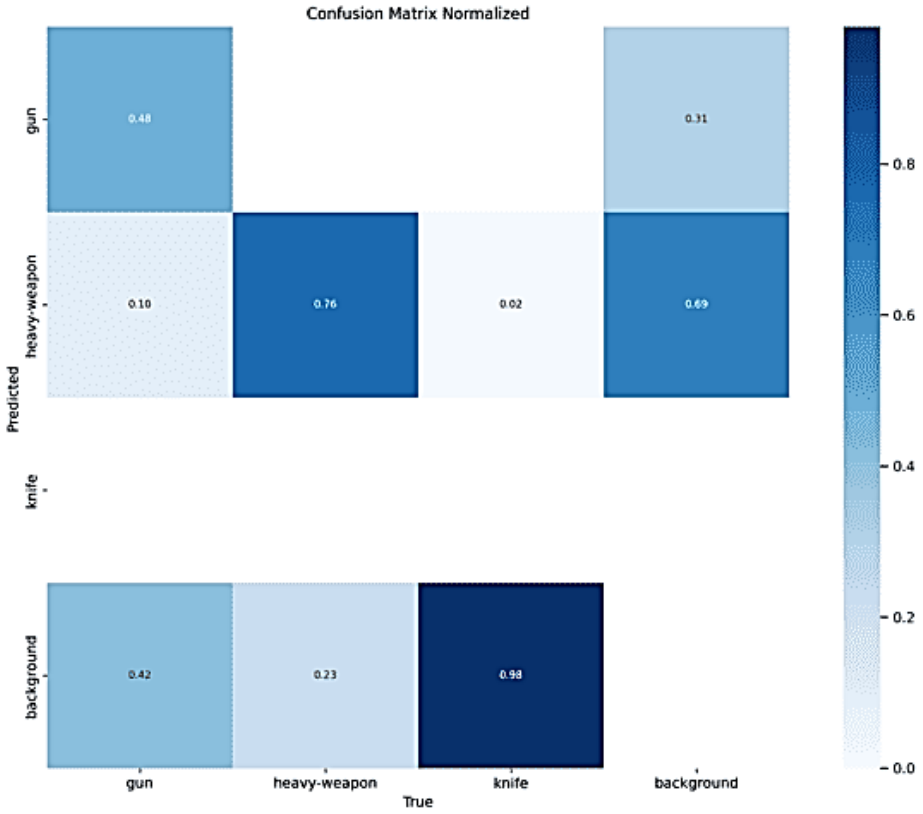


Figure 3: Normalized Confusion Matrix

Figure 4 reveals that the gun detection model achieves a high F1 score at a confidence threshold of 0.399. However, this score significantly decreases as the confidence threshold increases. This indicates that while the model can accurately identify most firearms with high precision and recall at the 0.399 threshold, it misses more guns as the confidence requirement becomes stricter. This suggests that the YOLO models might be better suited for applications prioritizing minimizing false positives, such as security systems, where mistakenly identifying an object as a gun is more critical than potentially missing a real weapon.

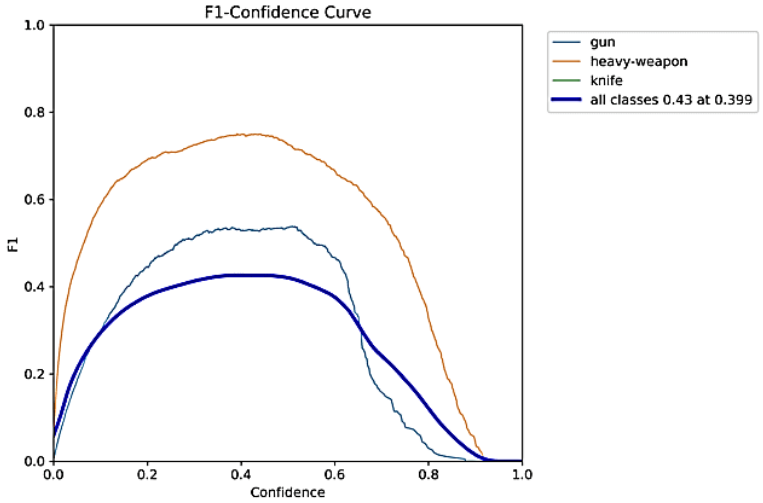


Figure 4: F1-Confidence Curve

Figure 5 shows that the gun detection model is more accurate than both the knife and heavy weapon detection models across all confidence levels. However, the accuracy of all three models decreases as the confidence threshold increases. This trade-off between precision and recall allows us to choose a suitable confidence threshold for each model based on the

desired outcome. For instance, prioritizing high accuracy would lead to selecting a higher confidence threshold. However, this would also result in more missed detections (false negatives).

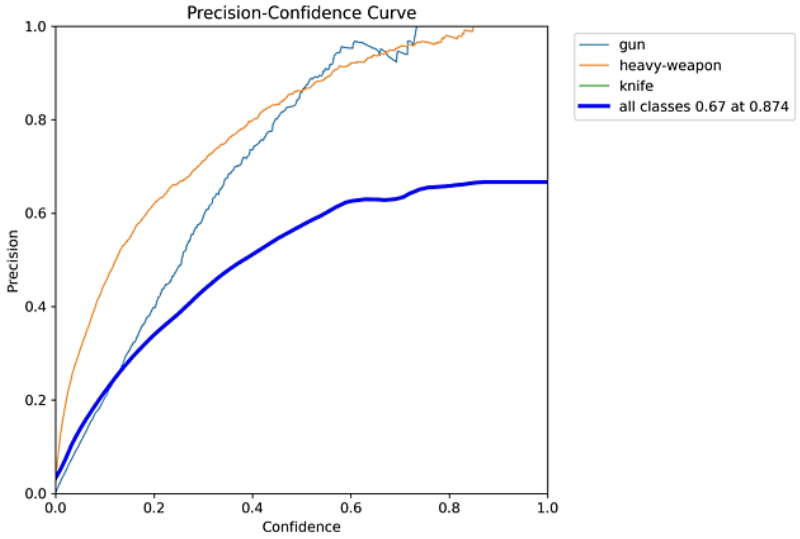


Figure 5: Precision-Confidence Curve

Figure 6 shows that the gun detection model performs significantly better than the knife and heavy weapon models in terms of both precision and recall across different recall levels. However, the precision of all models decreases as we try to detect more weapons (higher recall). This trade-off between precision and recall allows us to choose a suitable operating point for each model. While the model initially focuses on learning specific details about weapons (high precision), as training progresses, it strives to identify all weapons in the images (high recall), leading to a decrease in false positives. Therefore, if prioritizing accuracy is crucial, a lower recall level with higher precision is preferable. However, this might result in missing some weapons in the images.

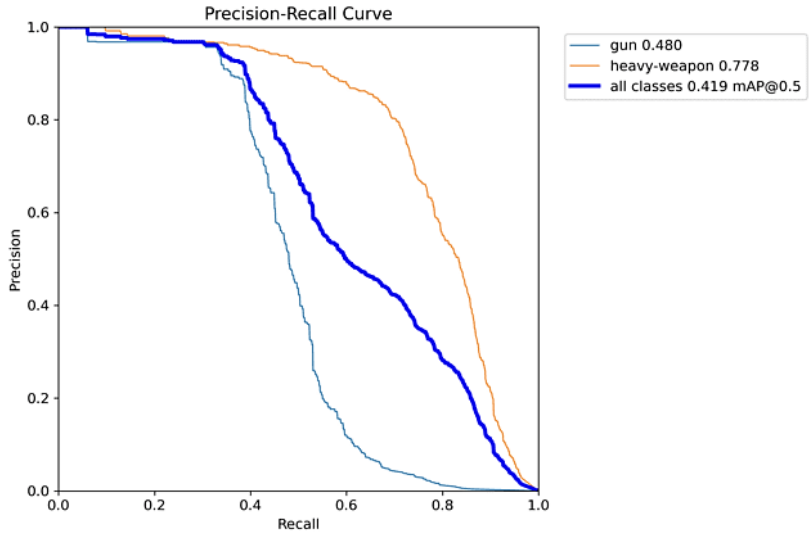


Figure 6: Precision-Recall Curve

Figure 7 shows that the gun detection model is better at identifying actual guns (higher recall) compared to heavy weapons and knives, regardless of the confidence level used. However, all three models become less accurate (recall decreases) as the confidence threshold increases. This finding allows us to choose a confidence threshold that strikes a balance between correctly identifying weapons (recall) and the number of false positives (precision) for each model. In other words, if prioritizing identifying all possible weapons is crucial, a lower confidence threshold can be used, even though it might lead to more false positives.

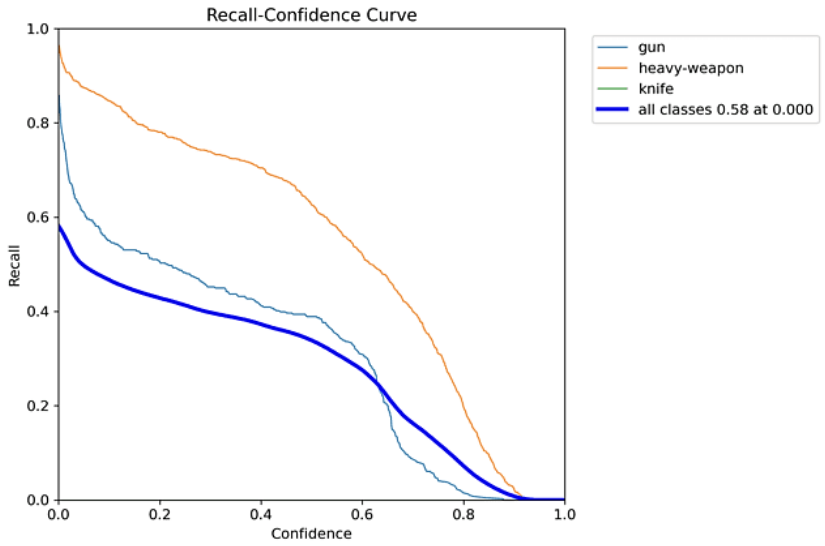


Figure 7: Recall-Confidence Curve

IV. COMPARATIVE STUDY OF VARIANTS

Before the development of YOLO, bounding box proposals were first generated on the input image by Region Proposal Networks (RPNs), which were then used by object detector CNNs like R-CNN to refine the bounding boxes and remove duplicate detections. The R-CNN network's phases needed to be trained independently. The R-CNN network was sluggish and difficult to optimize. The designers of YOLO were driven to create a single-stage CNN that was real-time, easily optimized, and could be trained from start to finish.

A. YOLO V1

The input image is divided into S x S grid cells by YOLO. B bounding boxes and a "objectness" score P(Object) indicating the presence or absence of an object are predicted for each grid cell. In addition, every grid cell forecasts the conditional probability P(Class | item) of the class to which the item it contains belongs. YOLO forecasts five parameters (x, y, w, h, and a confidence score) for every bounding box. The coordinates (x,y) indicate the bounding box's centroid with respect to the grid cell. X and Y have restricted values between 0 and 1. The bounding box's predicted width (w) and height (h) are expressed as a percentage of the image's total width and height. Thus, they range in value from 0 to 1. The bounding box's accuracy and presence of an object are indicated by its confidence score. The confidence score is 0 if there is no object inside the bounding box. The confidence score is equal to the intersection over union (IoU) of the predicted bounding box and the ground truth if the bounding box contains an object. Thus, YOLO predicts B x 5 parameters for each grid cell. Yolo forecasts C class probability for every grid cell. The prerequisite for these class probabilities is that the object must be present in the grid cell. For each grid cell with B bounding boxes, YOLO predicts just one set of C class probabilities. YOLO therefore forecasts C + B x 5 parameters for every grid cell. The image's total prediction tensor is equal to S x S x (C + B x 5). YOLO utilizes S = 7, B = 2, and C = 20 for the PASCAL VOC dataset. For PASCAL VOC, the final YOLO prediction is therefore a tensor of size 7 x 7 x (20 + 5 x 2) = 7 x 7 x 30. Lastly, as seen in figure 1 right image, YOLO version 1 uses thresholding and Non-Maximum Suppression (NMS) to report final predictions.

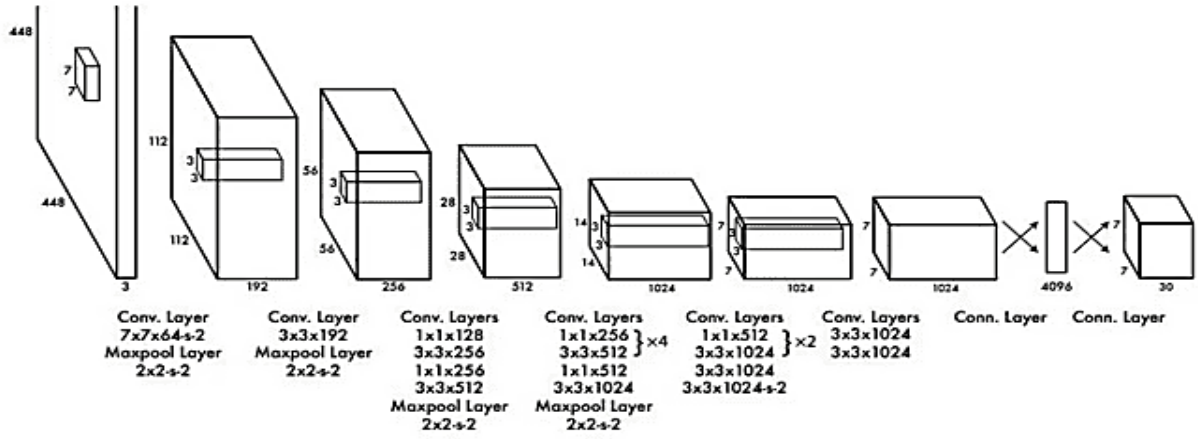


Figure 8: YOLO version 1 CNN

YOLO version 1 The CNN is shown in Figure 8. It functions as a feature extractor thanks to its 24 convolution layers. Two fully linked layers that handle object categorization and bounding box regression come after them. A 7 x 7 x 30 tensor is the ultimate result. Like VGG19, YOLO CNN is a straightforward single path CNN. YOLO draws influence from Google's Inception version 1 CNN and uses 1x1 and then 3x3 convolutions. All layers—aside from the final layer—use leaky ReLU activation. In the last layer, the activation function is linear.

Limitations:

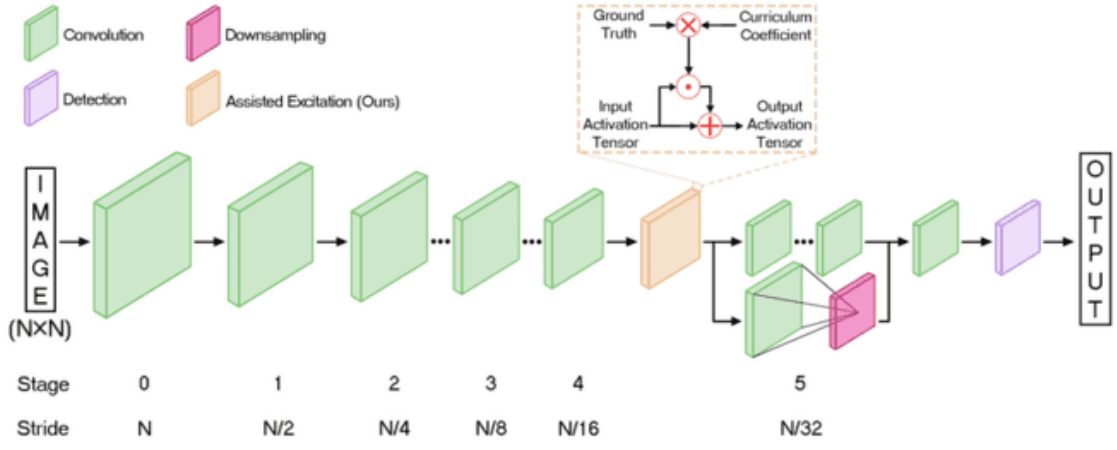
1. YOLO finds it challenging to identify small objects that are grouped together.
2. YOLO has trouble identifying items with peculiar aspect ratios.
3. Localization errors are higher in YOLO than in Fast R-CNN.

B. YOLO V2

The second iteration of the YOLO algorithm, which debuted in 2016, is known as YOLO V2. The YOLO algorithm has been enhanced, and YOLO V2 is now faster and more accurate than the original. We will talk about YOLO V2's architecture and operation in this article. YOLO V2's architecture is made up of two fully connected layers and 24 convolutional layers. A sequence of convolutional layers is applied to the input image in order to extract features from it. Subsequently, the bounding boxes and class probabilities of the objects in the image are predicted using these attributes.

The utilization of anchor boxes is one of the main enhancements in YOLO V2. To forecast the bounding boxes of objects in a picture, anchor boxes are pre-defined boxes with varying sizes and aspect ratios. YOLO V2 can forecast bounding boxes for objects of various sizes and shapes more precisely by employing anchor boxes.

The application of batch normalization in YOLO V2 is an additional enhancement. By normalizing the input to each layer, a technique known as batch normalization serves to lower overfitting and raise the model's accuracy. Additionally, YOLO V2 makes use of a method known as skip connections, which enables data to be transferred from older layers to subsequent layers. This enables the model to learn more intricate features, which helps to increase its accuracy. The output of YOLO V2 consists of bounding boxes and class probabilities for every object in the picture. The bounding boxes are represented by (x, y, w, h), where (w, h) stands for the box's height and width and (x, y) for its center. Class probabilities reflect the likelihood that an object belongs to a particular class. We have updated the YOLOv2 architecture to include our newly aided excitation layer. Each step can have AE added at the conclusion; however, our research indicates that the best time to introduce AE is at the conclusion of stage 4. A sequence of activation tensors with comparable resolutions makes up each step. Assume, for instance, that the input image is 480 by 480 in size. Tensors with the following resolutions are present in stages 1, 2, 3, 4, 5, and 6: 240 x 240, 120 x 120, 60 x 60, 30 x 30, and 15 x 15, respectively.



IV.

Figure 9: YOLO version 2 CNN

The rectangular blocks in Figure 9, represent the different layers of the neural network. The information flows through the network from left to right. Each layer performs a specific operation on the data, ultimately allowing the network to identify and localize objects in an image. The text “DUN” likely refers to the Deep layer, a custom layer specific to this architecture. To train YOLO V2, a substantial amount of annotated photo collections is required. The dataset is used to train the model to recognize different objects and the classes that go with them. The model is trained using a loss function that rewards successful predictions and penalizes faulty ones. After training, the model may be applied to identify items in fresh pictures. First, the model is applied to the input image, predicting the bounding boxes and class probabilities for every object in the picture. Kindly find the predicted bounding boxes to eliminate duplicates; non-maximum suppression is used to determine which bounding boxes are most likely for each object.

Limitations:

- Struggles with groups of small objects.
- Prone to localization errors for unusual shapes.
- Limited number of detectable object classes.

C. YOLO V3

YOLOv3 boasts significant advancements over YOLOv2 in speed, accuracy, and class detection:

- Backbone Boost: YOLOv3 utilizes the more powerful and efficient Darknet-53 backbone.
- Multi-Scale Predictions: YOLOv3 predicts boxes at three scales.

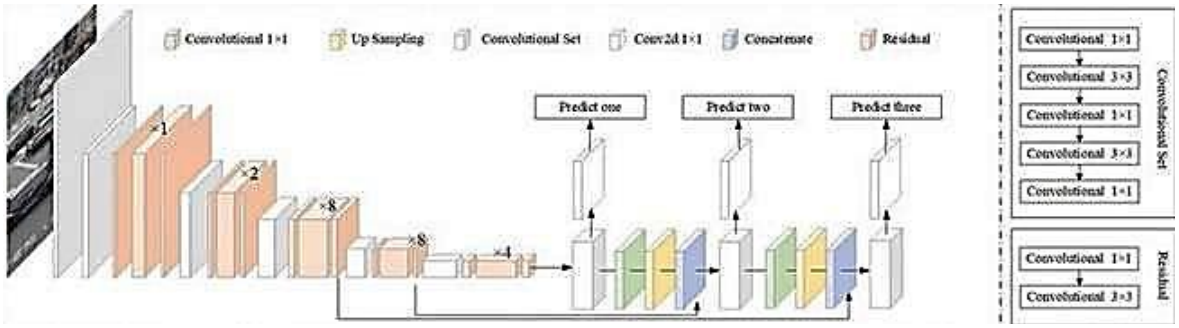


Figure 10: YOLO version 3 CNN

Table 1: Comparison of backbones

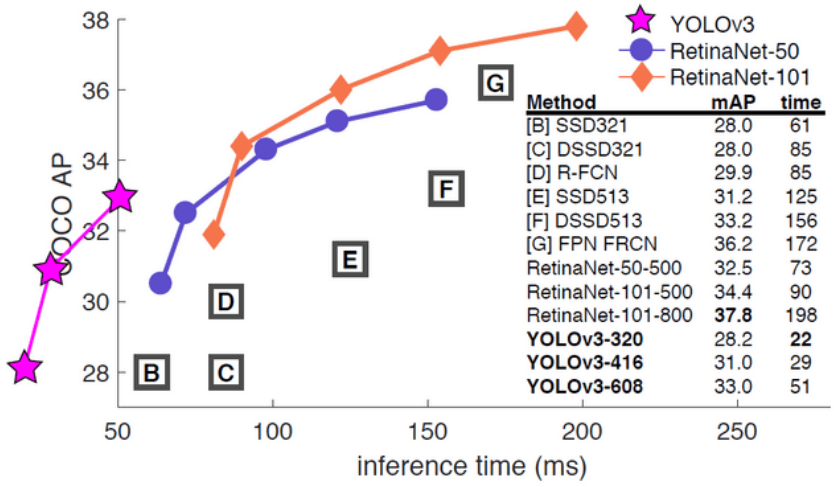
Backbone	Top-1	Top-5	Ops	BFLOP/s	FPS
Darknet-19	74.1	91.8	7.29	1246	**171**
ResNet-101	77.1	93.7	19.7	1039	53
ResNet-152	**77.6**	**93.8**	29.4	1090	37
Darknet-53	77.2	**93.8**	18.7	**1457**	78

In table 1, among the models compared, Darknet-19 stands out for its efficiency. It requires the fewest operations, achieves the highest BFLOP/s (billion floating-point operations per second), and processes frames the fastest (FPS). This makes it ideal for real-time applications. However, this efficiency comes at a cost - Darknet-19 has the lowest accuracy. Conversely, ResNet-101 and ResNet-152 boast higher accuracy but are computationally expensive. Striking a balance is Darknet-53, offering better accuracy than Darknet-19 while remaining more efficient than ResNet models.

YOLOv3 shines in both speed and accuracy:

- Darknet-52 offers 1.5x the speed of ResNet101 without sacrificing accuracy (comparable to ResNet-152).
- YOLOv3 delivers high mAP and IOU values while being significantly faster than similar models.

V.



VI.

Figure 12: Comparative speed of YOLO version 3

In figure 12, This graph compares the speed and accuracy of various object detection models. The accuracy is measured by mAP (mean Average Precision) at a specific overlap threshold (IoU) of 50%. The faster a model is (lower inference time in milliseconds on the X-axis), the lower its accuracy tends to be (mAP on the Y-axis). For instance, RetinaNet-50-500 is more accurate than YOLOv3-416 but takes longer to process an image. This trade-off is crucial when choosing a model. If speed is essential for real-time applications, a faster model with slightly lower accuracy might be preferable. Conversely,

for tasks prioritizing high accuracy, a slower model can be chosen.

The chart showcases a clear gap in performance when comparing different algorithms and their ability to detect objects of varying sizes based on Average Precision (AP). YOLOv2 significantly struggles with small objects, achieving an AP of only 5.0, compared to other algorithms like RetinaNet (21.8) and SSD513 (10.2). This emphasizes a major limitation of YOLOv2 in accurately detecting small objects.

	backbone	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
Two-stage methods							
Faster R-CNN+++	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI	Inception-ResNet-v2	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	**52.1**
One-stage methods							
YOLOv2	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5
SSD513	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet	ResNeXt-101-FPN	**40.8**	**61.1**	**44.1**	**24.1**	**44.2**	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

Figure 13: YOLOv3 comparison for different object sizes

In figure 13, the performance of various object detection models, categorized as two-stage and single-stage methods. Two-stage methods excel in accuracy, particularly for challenging tasks like detecting small objects (AP Small) or objects with a high degree of overlap with the background (AP75). For instance, Faster R-CNN+++ with ResNet-101-FPN backbone achieves a significantly higher mAP (36.2%) compared to the single-stage YOLOv3 with Darknet-53 backbone (31.0%). However, this accuracy comes at the cost of speed, as two-stage models are generally slower than single-stage models. In contrast, single-stage models prioritize speed, making them suitable for real-time applications. Ultimately, the choice between a two-stage and single-stage model depends on the specific needs of the task. If high accuracy is paramount, a two-stage model is preferable, while real-time applications might benefit more from a faster single-stage model.

A key difference between YOLOv2 and YOLOv3 lies in their approach to class prediction. YOLOv3 utilizes independent logistic classifiers and binary cross-entropy loss, allowing it to assign multiple class labels to each bounding box. This makes it ideal for complex datasets like Microsoft's Open Images Dataset (OID), where objects often have overlapping labels (e.g., "man" and "person" for the same individual). In contrast, YOLOv2 employed a single SoftMax function, restricting each bounding box to a single class, which limited its ability to effectively handle such datasets. This multi-label approach in YOLOv3 enhances its ability to tackle diverse and nuanced object classification tasks.

Limitations:

- Small object detection: YOLOv3 can struggle with very small objects or tightly packed groups due to its grid cell limitations.
- False positives: Complex scenes or unusual shapes can lead to inaccurate object identification.
- Computational cost: While faster than earlier versions, YOLOv3 still requires significant resources for real-time applications.

D. YOLO V4

YOLOv4 is a great option for many applications since it is made to offer the ideal ratio of speed to accuracy.

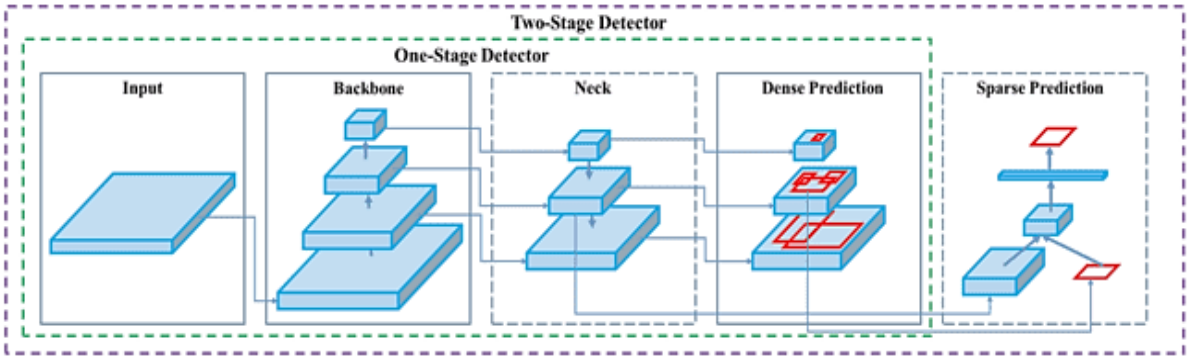


Figure 14: YOLO version 4 CNN

In figure 14, the layer architecture of the YOLO version 4 in the CNN methodology provides with a step wise layer transformation that took place in the evolution of the base object detection model.

To maximize its performance, YOLOv4 employs several cutting-edge innovations. These include Mish-activation, DropBlock regularization, CIoU loss, Self-adversarial-training (SAT), Weighted-Residual-Connections (WRC), Cross-Stage-Partial-connections (CSP), and Cross Mini-Batch Normalization (CmBN). State-of-the-art outcomes are obtained by combining these attributes.

The input, the neck, the head, and the backbone make up most of an object detector. YOLOv4's core is used to predict object bounding boxes and classes after being pre-trained on ImageNet. A variety of models, such as VGG, ResNet, ResNeXt, or DenseNet, could serve as the foundation. The neck portion of the detector, which typically has many top-down and bottom-up routes, is used to gather feature maps from various stages. The final object detections and classifications are made using the head portion.

Additionally, YOLOv4 employs what are referred to as "bag of freebies," or strategies that raise model accuracy during training without raising inference costs. A popular "bag of freebies" method for object detection is data augmentation, which boosts the input photos' variability to strengthen the model's resilience. Photometric distortions (changing an image's brightness, contrast, hue, saturation, and noise) and geometric distortions (adding random scaling, cropping, flipping, and rotating) are two types of data augmentation examples. These methods improve the model's ability to generalize to many image kinds.

Limitations:

- Data Dependence: YOLOv4's performance is sensitive to the amount and quality of training data. Insufficient or poorly labelled data can lead to decreased accuracy.
- Complex Feature Learning: The model's architecture may hinder its ability to learn complex features effectively, potentially impacting its performance on challenging tasks.
- Class Scalability: While capable, YOLOv4 might face difficulties with extremely large and diverse datasets containing a vast number of object classes.
- Occlusion Handling: Partially occluded objects can pose a challenge for accurate detection, leading to missed identifications or inaccurate bounding boxes.
- Long-Tail Distribution Sensitivity: YOLOv4 may require specific adjustments to handle datasets where a few classes are very common and many are rare, known as a long-tail distribution.

E. YOLO V5

A computer vision model in the You Only Look Once (YOLO) family is called YOLOv5. YOLOv5 is frequently utilized for object detection. YOLOv5 is available in four primary variants, with increasing accuracy rates: small (s), medium (m), large (l), and extra-large (x). The training time for each version varies as well.

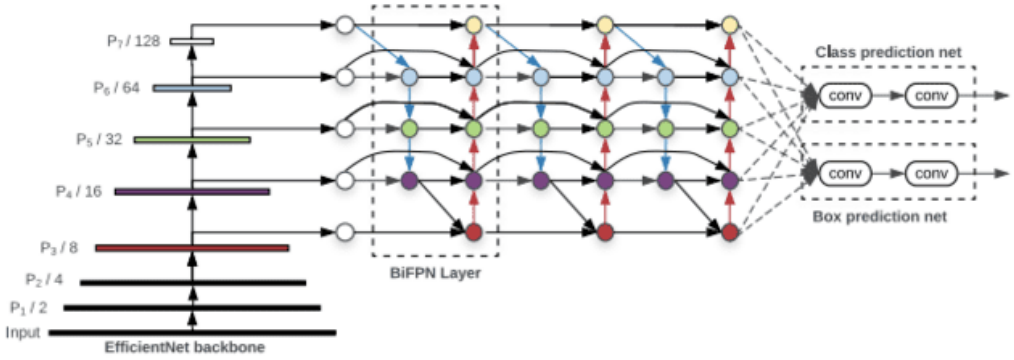


Figure 15: YOLO version 5 CNN

In figure 15, the evolution of YOLO version 4 was seen in a much simpler and a more efficient CNN base model approach hat provide much clarity into the scope and the emergence of new implementations within the hyper parameters and workload of the new YOLO version 5.

The first object detector to link the process of class label prediction with bounding box prediction in an end-to-end differentiable network was the YOLO model.

There are three primary components to the YOLO network:

- Convolutional neural networks that aggregate and produce picture characteristics at various granularities are the backbone of the network.
- Neck: An arrangement of layers that blends and combines picture characteristics before forwarding them to prediction.
- Head: Uses characteristics from the neck to predict the box and class.

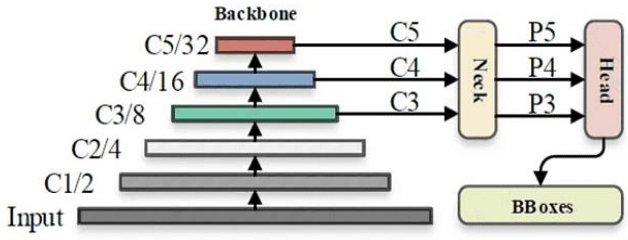


Figure 16: Overall architecture of YOLO version 5

In figure 16, first, a backbone network (CSPDarknet) extracts image feature. Then, a neck section (PAN) refines these features, allowing for detection of various object sizes. Finally, the head (YOLO layer) interprets the processed features to predict bounding boxes and class probabilities for the objects in the image. This architecture enables YOLOv5 to achieve fast and accurate object detection in real-time.

YOLOv5 utilizes CSPDarknet53 as its backbone, built upon a stack of CBS (convolution, batch normalization, and SiLU activation) and C3 modules. Finally, an SPPF module is added to further enhance the backbone's feature expression capabilities. Notably, the SPPF module avoids redundant operations compared to SPPNet by max pooling previously max pooled features, leading to significant speed improvements.

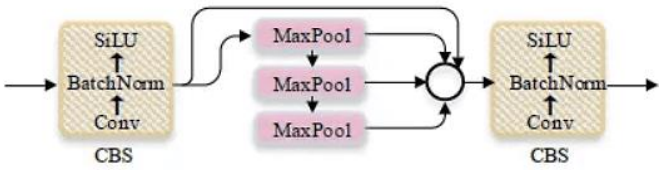


Figure 17: Structure of SPPF

The image you sent depicts the structure of Spatial Pyramid Pooling Fully Connected (SPPF) layer, commonly used in object detection with convolutional neural networks (CNNs). Here's how it works: the SPPF layer takes feature maps extracted from earlier layers, which contain information about the image at various scales and locations. It then applies multiple pooling operations of different sizes (like 1x1, 3x3, and 5x5) in parallel on these maps. This captures information about objects at various extents within the image. Each pooling operation likely uses max pooling, selecting the most prominent features at those different scales. Finally, the outputs from all these pooling layers are combined to create a single feature vector. This enriched vector, containing information about objects at various scales, is then fed into subsequent layers of the network for object detection. Overall, SPPF helps the network become more robust to variations in object size and position within the image, improving object detection accuracy.

YOLOv5 leverages both Feature Pyramid Network (FPN) and Path Aggregation Network (PAN) techniques for object detection. FPN up samples feature maps extracted from different levels of the backbone network, creating multiple new feature maps at various scales. This allows YOLOv5 to effectively detect objects of different sizes within the same image.

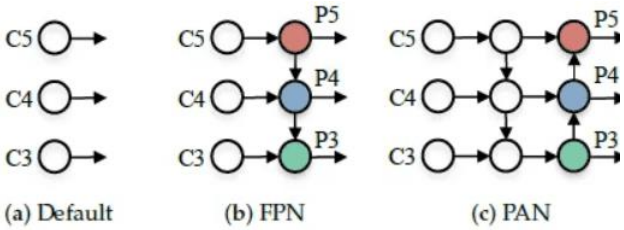


Figure 18: (a) Without Fixture Fusion, (b) FPN and (c) PAN

In figure 18, Default,FPN (Feature Pyramid Network), and PAN (Path Aggregation Network). All use a CNN backbone for feature extraction but differ in processing these features. The default architecture, using only the final layer features, struggles with objects of various sizes. FPN tackles this by constructing a pyramid of feature maps at different scales, allowing for better handling of various object sizes. PAN also improves multi-scale features by incorporating a bottom-up pathway to refine lower-resolution features and using shortcuts to combine information from different levels. In essence, both FPN and PAN offer more sophisticated approaches to feature representation compared to the default method, leading to better object detection across various sizes.

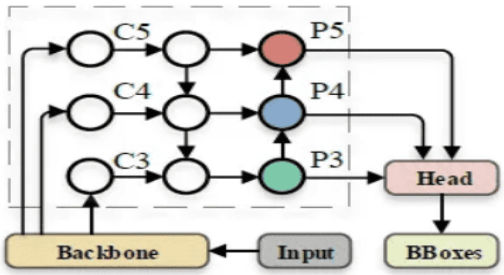


Figure 19: YOLO version 5 neck

In figure 19, the neck component of YOLOv5,which utilizes Path Aggregation Network (PAN) to enhance the features extracted from the backbone (CSPDarknet) for object detection. The neck tackles the challenge of multi-scale object detection by processing features at different resolutions. It accomplishes this through a two-pronged approach: a bottom-up pathway that upscales lower-resolution features to better detect small objects, and a top-down pathway that refines higher-resolution features using spatial pooling. Shortcut connections directly link these pathways, ensuring information flows across different scales. Finally, PAN merges the processed features from both pathways, resulting in a richer set of feature maps that capture objects at various sizes within the image. These enhanced features are then passed on to the head of the YOLOv5 model for object

detection. In essence, PAN's combination of bottom-up, top-down processing, and shortcut connections allows YOLOv5 to effectively detect objects of varying sizes within an image. In YOLO, bounding box predictions involve adjusting a pre-defined anchor box to match the actual object's location and size. Starting with the top-left corner of the feature map at (0, 0), the model predicts an unadjusted center point (rx, ry). Using this information, along with the prior anchor's dimensions (pw, ph), and offsets calculated by the model (sx, sy), the final prediction box (gx, gy, gw, gh) is obtained by adjusting the anchor's center and size to accurately represent the object's position and dimensions within the image.

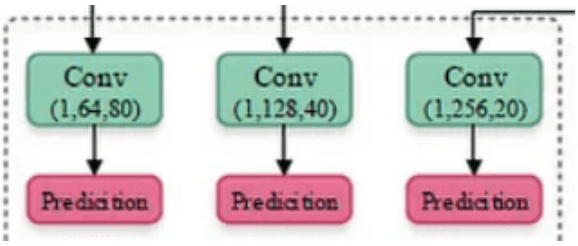


Figure 20: YOLO version 5 head

In figure 20, per convolutional batch in the TOLO version 5 model, a different number of characteristic features are taken into consideration for per CNN prediction.
YOLOv5 comes in five different versions: YOLOv5x, YOLOv5l, YOLOv5m, YOLOv5s, and YOLOv5n.

Table 1: Variants of YOLO version 5

Model	Size (Pixels)	mAP @0.5:0.95	mAP @0.5	Time CPU b1 (ms)	Time V100 b1 (ms)	Time V100 b32 (ms)	Params (M)	FLOPS @640 (B)
YOLOv5n	640	28.0	45.7	45	6.6	0.6	1.9	4.5
YOLOv5s	640	37.4	56.8	98	6.4	0.9	7.2	16.5
YOLOv5m	640	45.4	64.1	224	8.2	1.7	21.2	49.0
YOLOv5l	640	49.0	67.3	430	10.1	2.7	46.5	109.1
YOLOv5x	640	50.7	68.9	766	12.1	4.8	86.7	205.7

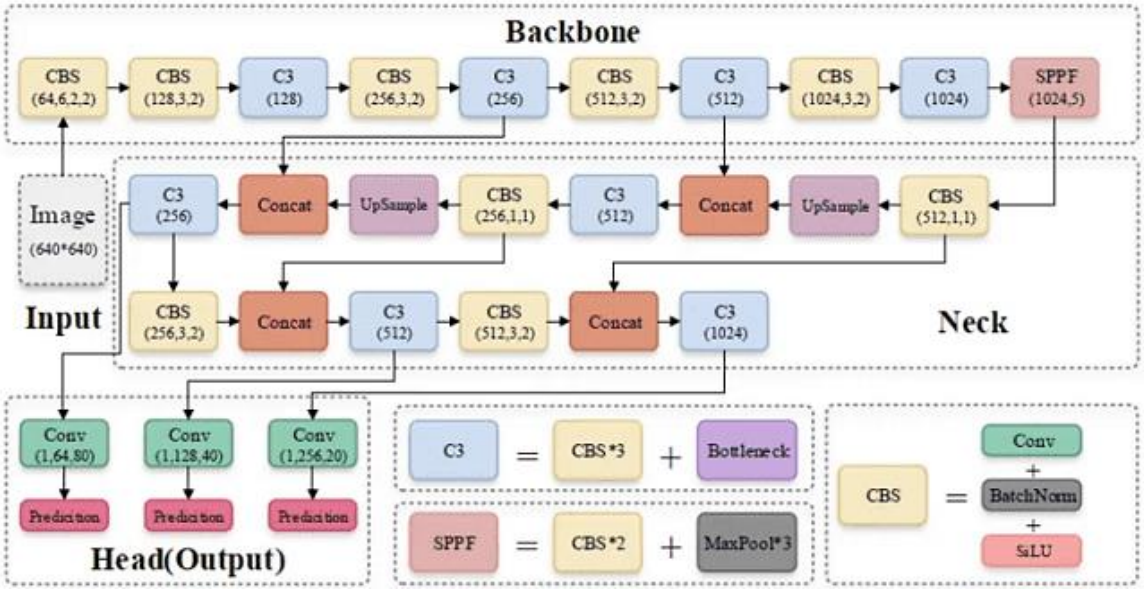


Figure 22: YOLO V5 – Backbone, neck and head

In figure 22, backbone, a modified Darknet called CSPDarknet, efficiently extracts image features. The neck, utilizing Path Aggregation Network (PAN), tackles multi-scale object detection. PAN employs a two-pronged approach: a bottom-up pathway to upscale low-resolution features for small object detection and a top-down pathway to refine higher-resolution features. Shortcut connections ensure information flows across scales. Finally, the head (YOLO layer) interprets the processed features from PAN to predict bounding boxes and class probabilities for the objects detected in the image. This efficient feature extraction, multi-scale processing, and final interpretation by the head allow YOLOv5 to achieve fast and accurate object detection in real-time.

F. YOLO V6

This updated version's original goal was to address the real-world issues that arose while working with industrial applications. The Meituan Visual Intelligence Department of the Chinese retail platform has developed a target detection

framework called MT-YOLOv6. It is a single-stage object detection framework that is primarily intended for industrial use. It outperforms YOLOv5 in terms of inference time and detection accuracy thanks to its hardware-efficient design. For production use, this makes it the optimal OS version of the YOLO algorithms.

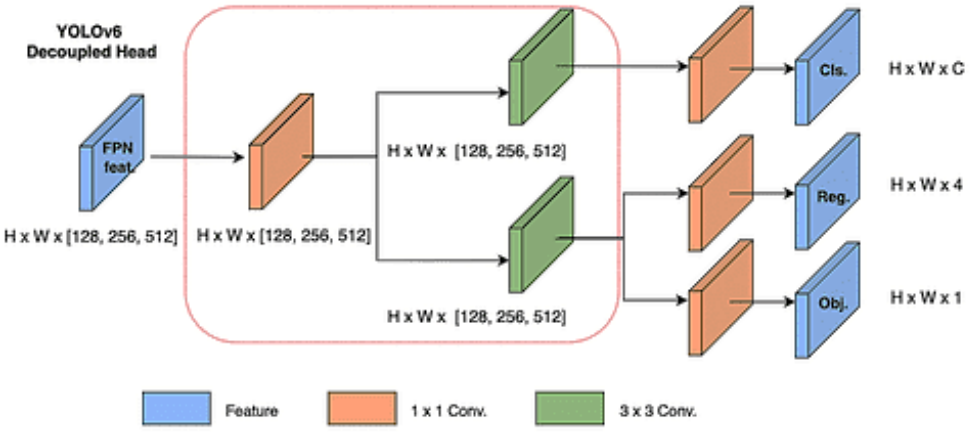


Figure 23: YOLO version 6 CNN architecture

YOLOv6 provides many model sizes for YOLO architectures: YOLOv6-n (4.3M parameters), YOLOv6-tiny (15M parameters), and YOLOv6-s (17.2M parameters). Larger sizes are anticipated even more. It is commonly known that the YOLOv6 neck and backbone function well with GPUs, cutting down on hardware latency and opening more use cases for industrial applications.

GPU processing with the Rep operator is preferred by the EfficientRep backbone of YOLOv6, enabling structural reparameterization. With this method, trained convolutional neural network layers are reorganized to improve inference speed.

As the figure below illustrates, RepConv layers are used by EfficientRep, followed by RepBlocks. A ReLU activation unit, a batch normalization layer, and several RepVGG layers make up each RepConv layer.

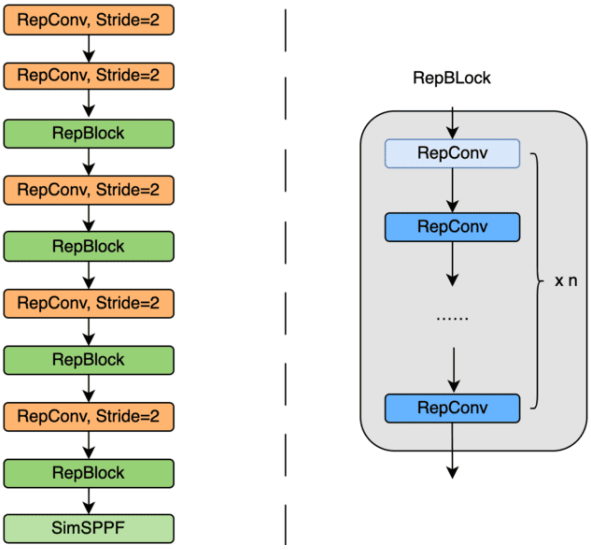


Figure 24: YOLO version 6 backbone

In the figure 24, a key component of YOLOv6 for object detection. This backbone prioritizes efficiency while maintaining accuracy.The fundamental building block is the RepBlock, a residual block that facilitates faster learning and better performance with fewer resources. Convolutional layers extract features, while CBL (Convolution, Batch Normalization, Leaky ReLU) refines them and introduces non-linearity. The SE (Squeeze and Excitation) mechanism helps focus on important features. Finally, RepConv layers, used during prediction, are simplified versions of convolutional layers, reducing computation cost while maintaining similar accuracy. Stride-2 in some layers halves the dimensions of data as the network progresses. Overall, the EfficientRep backbone achieves a balance between accuracy and efficiency by incorporating these techniques. The specific number of these components can vary depending on the chosen YOLOv6 variant, with smaller versions prioritizing speed for resource-limited devices. While this explanation offers a high-level overview, a deeper understanding of convolutional neural networks and optimization techniques would be necessary to grasp the inner workings of each layer. Based on PANet architecture, the Rep-PAN neck may employ RepBlocks rather than CSPNet blocks. To achieve improved object localization, PANet topology enhances low-level patterns through path augmentation. Concatenation and fusion are used by the PANet structure to anticipate the object class and mask.

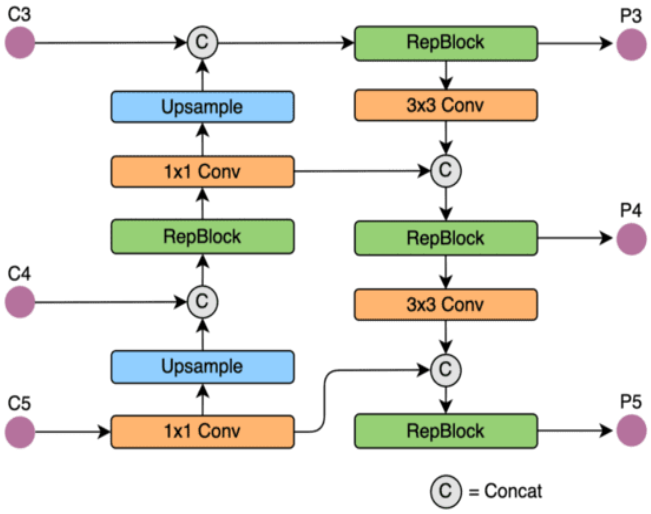


Figure 25: YOLO version 6 neck

In figure 26, YOLOv6's Rep-PAN, tackles multi-scale object detection. It receives feature maps of various resolutions from the backbone, suitable for different object sizes. A bottom-up pathway upscales low-resolution maps to improve small object detection, while a top-down pathway refines higher-resolution ones. RepBlocks, efficient building blocks like those in the backbone, are used throughout the neck for feature extraction. Feature fusion then combines the processed information from both pathways, creating a richer feature representation. Finally, a PAN Channel Attention mechanism analyzes the importance of data within the fused features and emphasizes informative channels, resulting in a more effective representation for object detection at various scales. This combination of upscaling, refining, feature fusion, and attention allows the Rep-PAN neck to excel at multi-scale object detection within a single image, while the specific configuration can be adjusted based on the chosen YOLOv6 variant for efficiency on resource-limited devices. Understanding the inner workings of each layer and the optimization techniques used would require further exploration of convolutional neural networks and attention mechanisms.

The mean Average Precision (mAP) statistic is used to compare the three YOLOv6 variations that are currently available. Usually, this is done to assess how well object detection models work. mAP averages the AP for all classes by measuring the AP for each class label.

Four metrics—precision, recall, and a precision-recall curve for each threshold—are used to derive average precision. Lastly, the average of each AP value is used to get the mean Average Precision.

The most sophisticated SOTA detection framework, YOLOv6, outperforms YOLOv5 and YOLOX in terms of accuracy performance. Let's talk about some variations among the three models.

1. On the COCO validation dataset, the values for Average Precision are as follows: YOLOv5-small provides 37.3 mAP, YOLOX-small provides 40.5 mAP, and YOLOv6-small leads the field with 43.1 mAP.
2. Speed: The YOLOv5-nano can reach a latency of 5.38 ms, whilst the YOLOv6-small has a latency of 3.59 ms.
3. Backbone: YOLOX employs the CSPDarkNet53 backbone, while YOLOv5 uses the CSPNet backbone. Because YOLOv6 can effectively employ GPU hardware and runs on the EfficientRep backbone, which has arguably stronger input representation capabilities, the detection network is strengthened for industrial applications.
4. Matching: YOLOv5 defines positive samples by comparing the aspect ratios of ground truth boxes and anchors. It then employs center point offset to ensure that each GT is allocated to a greater number of anchors. YOLOv6 and YOLOX, on the other hand, employ SimOTA, which dynamically produces more high-quality positive samples, improving detection accuracy by 1.3% AP in comparison to YOLOv5.
5. Loss: For their tiny model, YOLOv6 utilizes a SIoU loss; for the other models, they use a simple IoU; YOLOX employs an IoU loss; and YOLOv5 employs an IoU loss.

Although small object detection, false positives, and computational cost are common limitations of previous versions of YOLO, the following are some potential limitations unique to YOLOv6: Overfitting due to EfficientNet-L2

- Potential for Quantization Issues
- Limited Research and Development
- Integration Challenges

G. YOLO V7

The current version of YOLO models, YOLOv7, represents a step forward for real-time object identification. Compared to its earlier iterations (YOLOv5, for example), YOLOv7 infers more quickly and accurately, raising the bar for object recognition.

The authors of YOLOv7 aimed to advance object detection by developing a network architecture that could predict bounding boxes at comparable inference speeds and with greater accuracy than its competitors.

Efficient inference speed is dependent on the effectiveness of the YOLO networks convolutional layers in the backbone. WongKinYiu used Cross Stage Partial Networks to set off along the route of maximum layer efficiency. The authors of YOLOv7 expand on previous research on this subject by considering the amount of memory required to maintain layers in memory as well as the distance required for a gradient to back-propagate through the layers. Their network will be able to learn more effectively the shorter the gradient. E-ELAN, an expanded variant of the ELAN computational block, is the last layer aggregate they select.

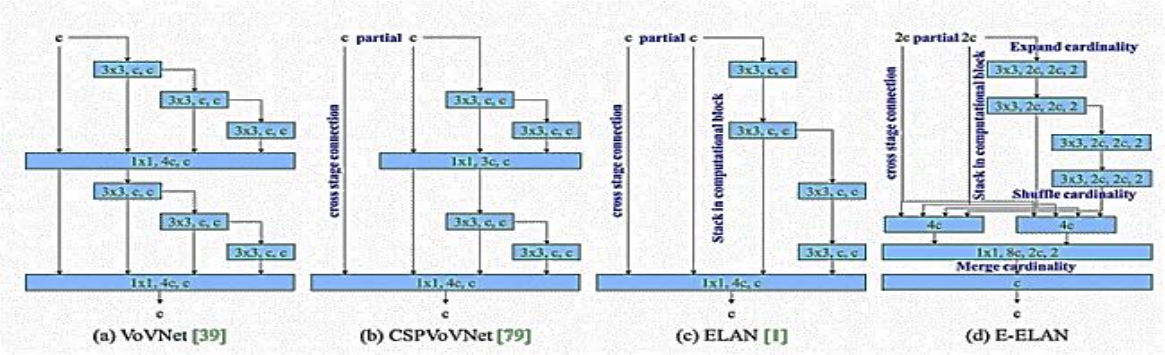


Figure 26: Efficient Layer Aggregation in YOLO V7

Because various applications need varying degrees of accuracy and inference speeds, object detection models are usually released in a series of models that scale up and down in size. Object detection models consider the network's breadth, depth, and training resolution. In YOLOv7, the writers' concatenate layers together while simultaneously scaling the network's width and depth. Studies on ablation demonstrate that this method maintains the ideal model design as it scales for various sizes.

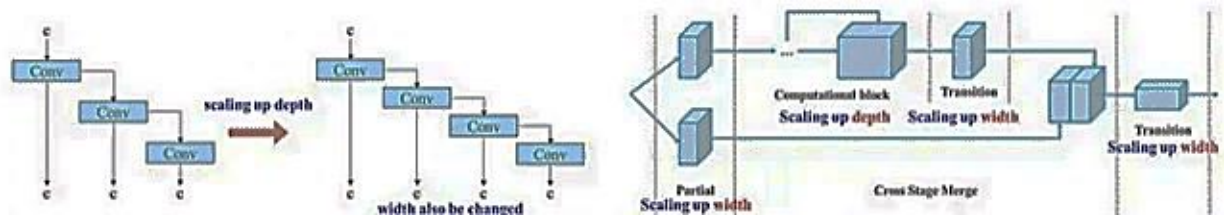


Figure 27: Compound scaling in YOLO V7

In re-parameterization procedures, a set of model weights is averaged to produce a model that is more resilient to the general patterns it is attempting to represent. Recently, module level re-parameterization—where individual network components have their own re-parameterization strategies—has drawn attention in study.

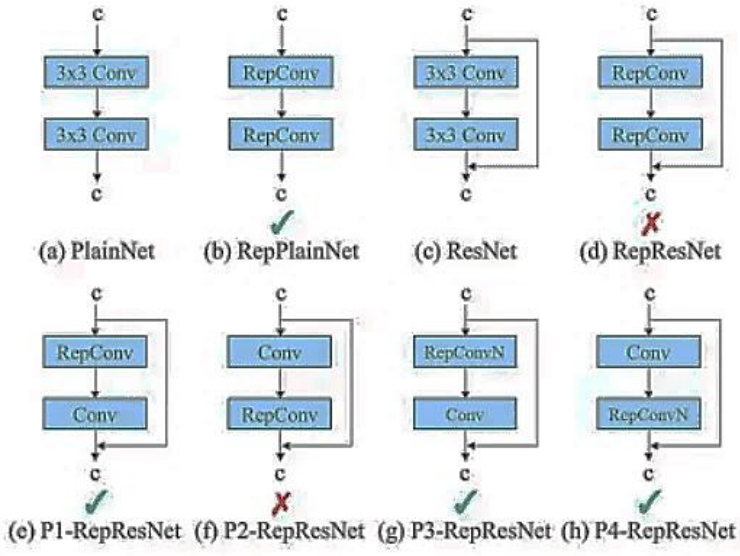


Figure 28: Re-parametrization in YOLO V7

In figure 28, YOLO V7 utilizes a new ELAN (Efficient Layer Aggregation Network) backbone for feature extraction. ELAN is designed for efficiency, enabling good performance on resource-constrained devices by controlling information flow within the network. The backbone outputs feature maps containing information about the image at various levels of

detail. These maps are then processed by the E-ELAN (Extended Efficient Layer Aggregation Network) neck, which builds upon ELAN by using group convolutions to enhance the model's learning ability. Finally, the processed features are fed into the head (YOLO layer), which predicts bounding boxes and class probabilities for the objects detected in the image. This architecture allows YOLOv7 to achieve fast and accurate object detection in real-time. It's important to note that the specific configuration of ELAN and E-ELAN can vary depending on the chosen YOLOv7 variant, with options optimized for speed or accuracy based on the application. While this explanation provides a high-level overview, a deeper understanding of ELAN and E-ELAN would require further exploration of convolutional neural networks.

Although YOLO v7 bears certain constraints from its predecessors, it may also have the following unique limitations:

- Potential Accuracy Trade-off for Speed
- Integration Challenges
- Limited Research and Development Time
- Newer Architecture and Potential Unforeseen Issues

H. YOLO V8

YOLOv8 is a model that comes with built-in functionality for object recognition, classification, and segmentation tasks. It can be accessed via a command line interface and a Python package.

YOLOv8 offers enhancements to the developer experience as well as architecture. In contrast to YOLOv5—its predecessor—YOLOv8 includes:

- A brand-new system for anchor-free detection.
- Modifications to the model's convolutional blocks.
- The mosaic augmentation that was used throughout training was disabled prior to the last ten epochs.

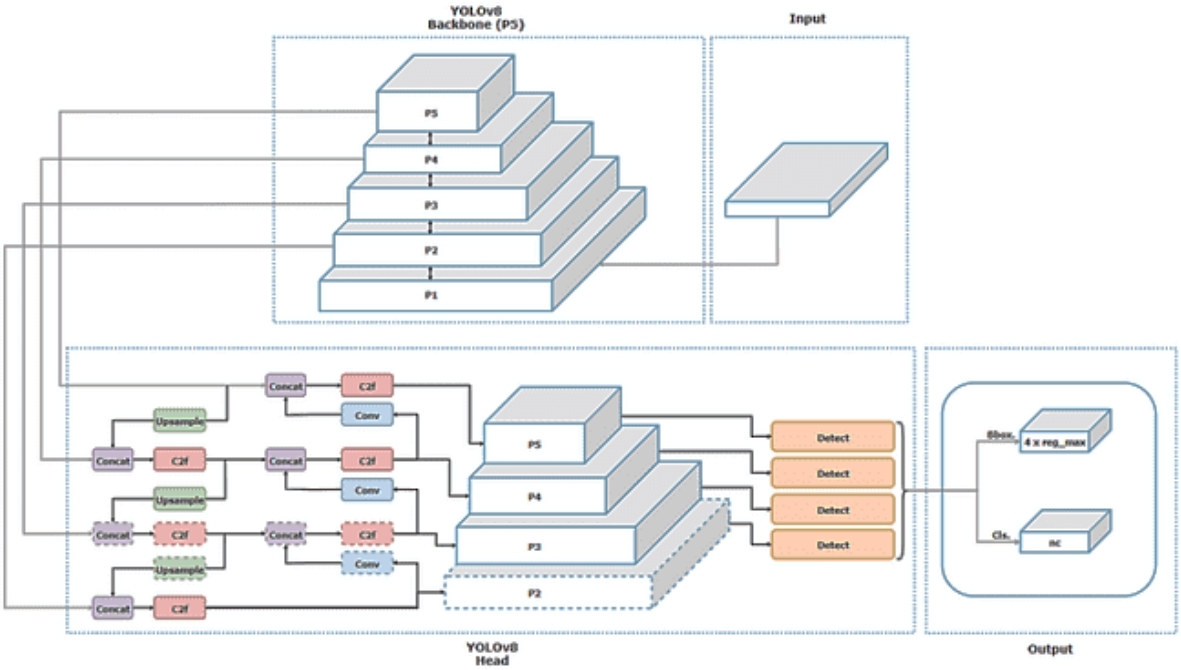


Figure 29: Architecture of YOLO version 8

The architecture of YOLO V8 shown in figure 29, prioritizes both speed and accuracy for real-time object detection.CSPDarknet53, a modified and efficient Darknet network, forms the backbone for feature extraction. The PAN neck tackles multi-scale object detection by employing a two-pronged approach: a bottom-up pathway to improve low-resolution features for small objects and a top-down pathway to refine higher-resolution features. Shortcut connections ensure information flows across scales. PAN then merges the processed features from both pathways. Finally, the head (YOLO layer) interprets these features, predicting bounding boxes and class probabilities for the detected objects. This combination of efficient feature extraction, multi-scale processing, and final interpretation allows YOLOv8 to achieve fast and accurate object detection in real-time. While YOLOv8 is under development and might have variations for different goals, this explanation provides a high-level overview, with a deeper understanding requiring exploration of convolutional neural networks.

An anchor-free model is YOLOv8. This implies that rather than predicting an object's offset from a known anchor box, it predicts an object's centre directly. Because they could represent the distribution of the target benchmark's boxes but not the distribution of the custom dataset, anchor boxes were a notoriously difficult component of older YOLO models. By reducing the quantity of box predictions, anchor-free detection expedites Non-Maximum Suppression (NMS), a laborious post-processing stage that sorts through potential detections following inference.

Model	size (pixels)	mAP^{val} 50-95	Speed CPU (ms)	Speed T4 GPU (ms)	params (M)	FLOPs (B)
YOLOv8n	640	37.3	-	-	3.2	8.7
YOLOv8s	640	44.9	-	-	11.2	28.6
YOLOv8m	640	50.2	-	-	25.9	78.9
YOLOv8l	640	52.9	-	-	43.7	165.2
YOLOv8x	640	53.9	-	-	68.2	257.8

Figure 30: YOLO version 8 COCO evaluation

In figure 30, YOLOv8 object detection models based on their performance in the COCO dataset. Each model (e.g., YOLOv8n, YOLOv8s) is evaluated for accuracy (mAP), speed (inference time on CPU and GPU), and resource usage (model size and computations). You can choose the best model for your needs by considering this trade-off. For example, a smaller, faster model (like YOLOv8n) might be ideal for real-time object detection on a low-powered device, even if it's slightly less accurate. Conversely, if you have more resources and prioritize top accuracy, a larger, slower model (like YOLOv8X) could be the better choice.

V. RESULTS AND DISCUSSION

Compared to other YOLO variants evaluated on the same dataset of gun and weapon images, YOLOv8 demonstrated exceptional performance. While all models exhibited high accuracy, YOLOv8 surpassed them with an impressive 92.5% average precision, indicating it correctly identified weapons in 92.5% of images. Additionally, its high sensitivity and specificity showcase superior effectiveness in both detecting weapons and distinguishing them from other objects. Furthermore, the model's mean IoU of 53.6% signifies its predicted bounding boxes closely matched the actual weapon locations, solidifying its position as the most accurate weapon detection model among the YOLO variants tested. This remarkable performance, coupled with its real-time capability, highlights YOLOv8's potential as the leading choice for robust and accurate weapon detection in security applications.
In figure 31 below, YOLO Version 8 performs extremely well in detecting custom dataset consisting of weapons and identifying them individually as weapons within the categories defined. The accuracy achieved in as per required and desired standards.

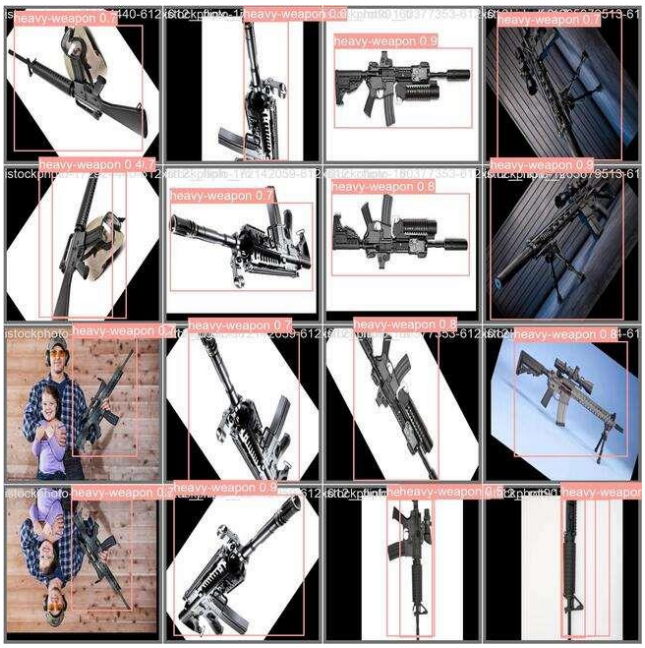


Figure 31: Custom Detection Result

VI. CONCLUSION AND FUTURE WORK

While this research presents a YOLOv8-based weapon detection system achieving an 88.2% mAP on a custom dataset, it's crucial to consider its performance within the context of the comparative study. While YOLOv8 demonstrates good accuracy, further analysis is needed to determine its position among other YOLO variants. While the proposed system is fast enough for real-time applications, a direct comparison with the processing speeds of other YOLO models would provide a more comprehensive understanding of its relative efficiency. Future improvements, including training on a larger dataset and incorporating features like occlusion handling and motion tracking, hold potential to further enhance the performance and robustness of the YOLOv8-based system. Additionally, deploying the system in real-world scenarios like security cameras and surveillance systems will provide valuable insights into its practical effectiveness compared to other YOLO variants in operational settings.

CONFLICTS OF INTEREST

The authors declare that they have no conflicts of interest.

REFERENCES

1) Vijayakumar and S. Vairavasundaram, "YOLO-based Object Detection Models: A Review and its Applications," Multimedia Tools and Applications, pp. 1-40, 2024.
2) P. Azevedo and V. Santos, "Comparative analysis of multiple YOLO-based target detectors and trackers for ADAS in edge devices," Robotics and Autonomous Systems, vol. 171, p. 104558, 2024.
3) M. Hussain, "YOLOv1 to v8: Unveiling Each Variant–A Comprehensive Review of YOLO," IEEE Access, vol. 12, pp. 42816-42833, 2024.

4) S. Hossain, H. Anzum, and S. Akhter, "Comparison of YOLO (V3, V5) and MobileNet-SSD (V1, V2) for Person Identification Using Ear-Biometrics," International Journal of Computing and Digital Systems, vol. 15, no. 1, pp. 1259-1271, 2024.
5) E. Casas, L. Ramos, E. Bendek, and F. Rivas-Echeverria, "YOLOv5 vs. YOLOv8: Performance Benchmarking in Wildfire and Smoke Detection Scenarios," Journal of Image and Graphics, vol. 12, no. 2, 2024.
6) N. Tanaka, H. Tanaka, M. Ikeda, and L. Barolli, "A Comparative Study of Four YOLO-Based Models for Distracted Driving Detection," in International Conference on Emerging Internet, Data & Web Technologies, Cham: Springer Nature Switzerland, pp. 362-370, Feb. 2024.
7) M. Alruwaili, M.H. Siddiqi, M.N. Atta, and M. Arif, "Deep learning and ubiquitous systems for disabled people detection using YOLO models," Computers in Human Behavior, vol. 154, p. 108150, 2024.
8) N.F. Alhussainan, B. Ben Youssef, and M.M. Ben Ismail, "A Deep Learning Approach for Brain Tumor Firmness Detection Based on Five Different YOLO Versions: YOLOv3–YOLOv7," Computation, vol. 12, no. 3, p. 44, 2024.
9) Ahmed, A.S. Imran, A. Manaf, Z. Kastrati, and S.M. Daudpota, "Enhancing wrist abnormality detection with YOLO: Analysis of state-of-the-art single-stage detection models," Biomedical Signal Processing and Control, vol. 93, p. 106144, 2024.
10) P.P. Khaire, R.D. Shelke, D. Hiran, and M. Patil, "Comparative Study of a Computer Vision Technique for Locating Instances of Objects in Images Using YOLO Versions: A Review," in International Conference on Information and Communication Technology for Intelligent Systems, Singapore: Springer Nature Singapore, pp. 349-359, Apr. 2023.
11) H. Deshpande, A. Singh, and H. Herunde, "Comparative analysis on YOLO object detection with OpenCV," International Journal of Research in Industrial Engineering, vol. 9, no. 1, pp. 46-64, 2020.
12) V. Goyal, R. Singh, A. Kumar, M. Dhawley, and S. Sharma, "Aerial Object Detection Using Different Models of YOLO Architecture: A Comparative Study," in Proceedings of International Conference on Computational Intelligence: ICCI 2021, Singapore: Springer Nature Singapore, pp. 333-345, Oct. 2022.
13) W. Xinming and T.S. Hong, "Comparative study on Leaf disease identification using Yolo v4 and Yolo v7 algorithm," AgBioForum, vol. 25, no. 1, 2023.
14) T. Diwan, G. Anirudh, and J.V. Tembhurne, "Object detection using YOLO: Challenges, architectural successors, datasets and applications," Multimedia Tools and Applications, vol. 82, no. 6, pp. 9243-9275, 2023.
15) T. Upadhyay, M. Aamash, M. Mittal, and G. Battineni, "Pest detection using state-of-the-art YOLO models: a comparative study," 2023.
16) K. Liu, H. Tang, S. He, Q. Yu, Y. Xiong, and N. Wang, "Performance validation of YOLO variants for object detection," in Proceedings of the 2021 International Conference on Bioinformatics and Intelligent Computing, pp. 239-243, Jan. 2021.
17) J. Doherty, B. Gardiner, E. Kerr, N. Siddique, and S.S. Manvi, "Comparative study of activation functions and their impact on the YOLOv5 object detection model," in International Conference on Pattern Recognition and Artificial Intelligence, Cham: Springer International Publishing, pp. 40-52, May 2022.
18) X. Wang, H. Li, X. Yue, and L. Meng, "A comprehensive survey on object detection YOLO," Proceedings, p. 0073, 2023. [Online]. Available: http://ceur-ws.org/Vol-1613/
19) M. Nawae, P. Maneelert, C. Choksuchat, T. Phairatana, and J. Jaruenpunyasak, "A Comparative Study of YOLO Models for Sperm and Impurity Detection Based on Proposed Augmentation in Small Dataset," in 2023 15th International Conference on Information Technology and Electrical Engineering (ICITEE), pp. 305-310, Oct. 2023.
20) Y. Zhou, "A yolo-nl object detector for real-time detection," Expert Systems with Applications, vol. 238, p. 122256, 2024.