International Journal of Advances in Scientific Research and Engineering (ijasre)

E-ISSN : 2454-8006

DOI: 10.31695/IJASRE.2023.9.11.7

Volume 9, Issue 11
November - 2023

Yolo Versions Architecture: Review

Rusul Hussein Hasan[1*], Rasha Majid Hassoo[2], and Inaam Salman Aboud[3]

[1,2] University of Baghdad, Baghdad, Iraq

[3]College of Education

Al-Mustansiriya University, Baghdad, Iraq

ABSTRACT

Deep learning techniques are used across a wide range of fields for several applications. In recent years, deep learning-based object detection from aerial or terrestrial photos has gained popularity as a study topic.

The goal of object detection in computer vision is to anticipate the presence of one or more objects, along with their classes and bounding boxes. The YOLO (You Only Look Once) modern object detector can detect things in real-time with accuracy and speed. A neural network from the YOLO family of computer vision models makes one-time predictions about the locations of bounding rectangles and classification probabilities for an image. In layman's terms, it is a technique for instantly identifying and recognizing items in images.

This article, will be focusing on comparing the main differences among the YOLO version's Architecture, and will discuss its evolution from YOLO to YOLOv8, its network architecture, new features, and applications. It starts by looking at the basic ideas and design of the first YOLO model, which laid the groundwork for the following improvements in the YOLO family. In additionally, this article will provide a step-by-step guide on how to use the YOLO version architecture, Understanding the primary drivers, feature development, constraints, and even relationships for the versions is crucial as the YOLO versions advance. Researchers interested in object detection, especially beginning researchers, would find this paper useful and enlightening.

Keywords: Artificial Neural Networks, Deep Learning, YOLO, YOLO Version.

---

1. INTRODUCTION

Artificial neural networks are computational methods that mimic how the brain of a person performs a particular task through massive parallel processing and the use of simple processing units known as neurons or nodes. Neurons have a neurological property in that they store real-world information and experimental data that they can be made available to the user by adjusting the weights [1,2].

The computer vision field has advanced quickly in recent years after making significant strides before. It is a real revolution in the growth of this subject, starting with the analysis of X-ray pictures and patient diagnosis all the way to self-driving automobiles. The main driver behind these innovations and achievements is the creation of better and more user-friendly computer resources [3, 4, 5].

YOLO, which was first created by Joseph Redmon and et al., target to detect objects with high precision and speed and work in real-time. YOLO is one-stage object detection models, which use convolutional neural networks (CNNs) to process the entire images in a single forward-pass [5, 6, 7].

There are two categories for deep learning: the first is referred to as two-stage detection algorithms that make multiple-stage predictions, the networks including Fast-RCNN, RCNN, and etc. while the other category is referred to as one-stage detectors, including Efficient Det, SSD, and YOLO as shown in Fig.1[8, 9, 10,11]

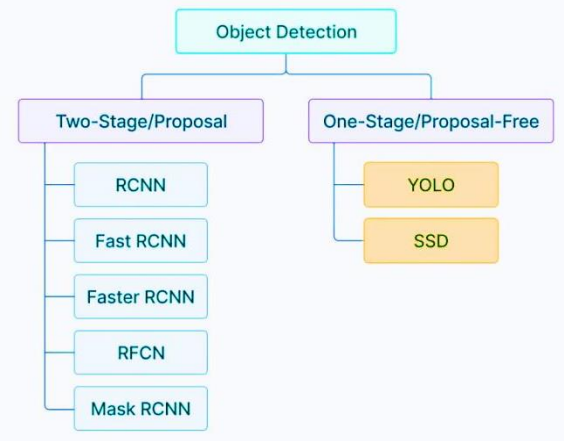


FIGURE 1: Deep Learning Categories

YOU ONLY LOOK ONCE (YOLO)

YOLO (You Only Look Once) is used in computer vision for real-time object recognition and classification[12] With YOLO an object detection network, class label identification and bounding box drawing were merged into one end-to-end differentiable network for the first time [12, 13]

YOLO models are frequently quick and compact, making them quicker to learn and simpler to use, especially on devices with constrained processing resources. If you think of the detection problem as a one-step regression strategy for identifying the bounding box, YOLO models are generally extremely quick and small [14]

The single-stage detection method used by YOLO, which aims to accurately and quickly detect objects, is its key characteristic . In contrast to two-stage detection algorithms like R-CNN, Anchor-free detection is superior to anchor-based detection because it is more flexible and efficient. Because there are no restrictions for manual anchor boxes, this can be difficult, and it might lead to less-than-ideal results in earlier YOLO models, like v1 and v2 [15, 16]

On a Titan X GPU, YOLO system forecasts images at 45 frames per second (FPS). Additionally, the authors created Fast YOLO, a much simpler variation of YOLO that operates at 155 frames per second and uses fewer layers[17,18,19]

As a result, YOLO's average accuracy of 63.4 maps, which is more than other real-time detectors, makes it even more unique. As shown in Fig. 2, YOLO and Fast YOLO both perform noticeably better than the DPM real-time object detector variations in terms of FPS and average accuracy [18, 19, 20].

Real-Time Detectors	Training Data	Evaluation (mAP)	FPS
100Hz Deformable Parts Model	2007	16.0	100
30Hz Deformable Parts Model	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45

FIGURE 2: Real-time Object Detector

YOLO VERSIONS

In Fig. 3 shows the YOLO versions like (YOLOV1, YOLOV2, YOLOV3, YOLOV4, YOLOV5, YOLOV6, YOLOV7 and YOLOV8) [37].

DOI: 10.31695/IJASRE.2023.9.11.7

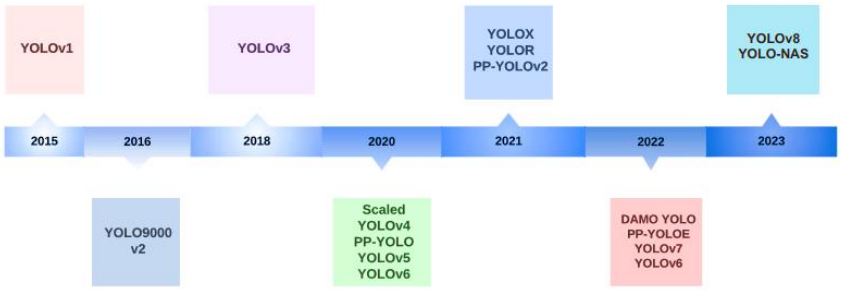


FIGURE 3: YOLO Versions

YOLO APPLICATIONS

For use cases that call for quick inference, like spotting autos, recognizing animals, and keeping an eye out for safety infractions, YOLO offers the speed that is required.

Additional scenarios where YOLO may be helpful include the following:

1. Recognizing trespassers in a factory.
2. Keeping an eye on traffic on a construction site.
3. Recognizing traffic patterns on a road (i.e., determining the busiest and least-busy times of day).
4. Recognizing smoke from wildfires.
5. Checking to make sure employees are wearing the proper PPE in specific situations (for as when using equipment or working with substances that release hazardous gases).

There are many applications for YOLO. It is always worth taking into account if you need to set up a camera to identify objects in real-time [12-21]

YOLO CHALLENGES

1- Overcrowding or Cluttered Scenario: The image in Fig. 4 is overcrowded with things. This poses a number of difficulties for the object detection model, including the possibility of massive occlusions, small objects, and inconsistent scaling.

FIGURE 4: A Large Group of People.

2- Intra-Class Variance: Accurately detecting objects of the same class, which can have substantial variance, is another significant problem for object detection. It can be difficult to identify objects of the same class, for instance, when there are six various breed of dogs, as is seen in Fig. 5. These canines vary in size, color, hair length, ears, and other characteristics.

DOI: 10.31695/IJASRE.2023.9.11.7

FIGURE 5: Six Breeds of Dogs.

3- Class Imbalance: This issue affects practically all modalities, including images, text, and time-series data. More particularly, image classification in the picture domain struggles significantly, and object recognition is no exception. Fig. 6 illustrates what we refer to as a foreground-background class imbalance in object detection.

Consider an image with a small number of primary items to comprehend how class imbalance could complicate object detection. The background takes up the remaining space in the photograph. As a result, the model would examine numerous areas of the image (dataset), the majority of which would be regarded as negative. These drawbacks prevent the model from picking up any valuable information and might make the training process overwhelming.

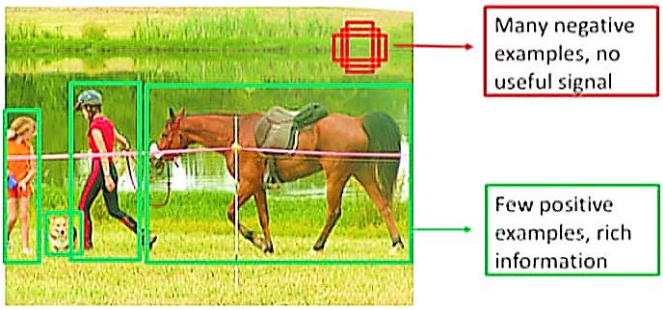


FIGURE 6: Class Imbalance

Occlusion, deformation, viewpoint variation, lighting conditions, and the crucial speed for real-time identification (needed in many industrial applications) are just a few of the difficulties that come with object detection [22-23]

THE YOLO ALGORITHM AND ARCHITECTURE

To recognize objects in the input image, the YOLO algorithm employs a conventional deep convolutional neural network [24 25].

The phrase "You Only Look Once" refers to the YOLO algorithm, which uses only one pass (forward-propagation) through the neural network to identify multiple objects in a single image. As a result, the image is splitting into parts, with the bounding box and probability for each part predicted [26 27 28]

Creating a single neural network from each component of the object detection pipeline, YOLO operates on the single-stage detection principle. To forecast bounding box coordinates and class probabilities, it makes use of the characteristics that are present throughout the entire images [29 30 31].

This method aids models in making generalized inferences about the entire image and its constituent objects. The proposal generator in older two-stage detectors, such as the RCNN, creates rough proposals for the picture, which are subsequently sent to the classification and regression stage in the next step as shown in Fig. 7 [32 33 34 35].

DOI: 10.31695/IJASRE.2023.9.11.7

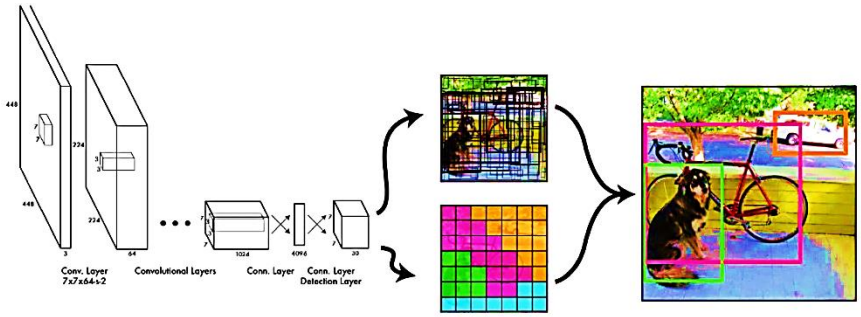


FIGURE 7: Detection of the YOLO System

YOLO V1 WORK AND ARCHITECTURE

Fig 8 demonstrates that there are three steps in the detection process: scaling the input image, using a convolutional single network to process the entire image, and thresholding the resulting detections by the model's confidence to eliminate redundant detections [24 36].

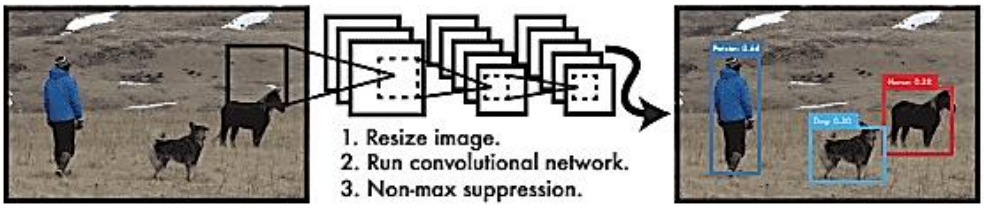


FIGURE 8: YOLOv1 Detection Process.

The YOLO architecture can train more quickly and reach real-time speed during inference thanks to this end-to-end unified detection design, which also ensures high-average precision (comparable to two-stage detectors) [36 38]

As seen in Fig. 9, the YOLO model separates the image into a grid. A cell is in charge of detecting an object if the center of one of the 49 grids contains the object [24 36 38]

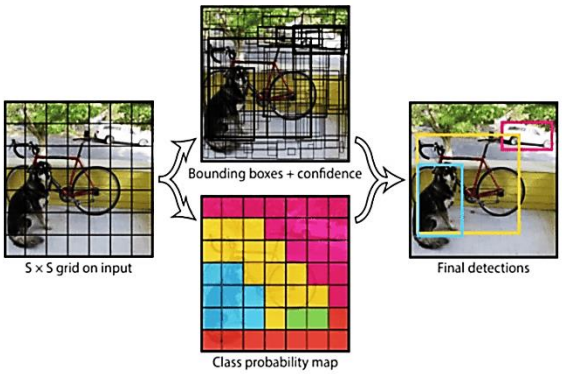


FIGURE 9: YOLO Detection Process

The probabilities and bounding box coordinates are predicted by two fully connected layers after 24 convolutional layers in the YOLOv1 architecture. Except for the final layer, which employed a linear activation function, all layers used leaky rectified linear unit activations. YOLO, which GoogLeNet inspired, uses a 1x1 convolutional layer to maintain a manageable parameters number and minimize the number of feature maps. Table 1 describes the YOLOv1 design as activation layers. In addition, the authors presented the Fast YOLO model, a simpler model with nine convolutional layers [24, 36, 37]

While the network's early convolutional layers extract characteristics from the image, its fully connected layers predict coordinates and output probabilities. YOLO network design was built on the Google Net concept for image classification. This network has 24 convolutional layers come before 2 completely coupled layers. Use 3x3 convolutional layers after 1x1 reduction layers in place of GoogleNet's inception modules [24, 37, 38]

DOI: 10.31695/IJASRE.2023.9.11.7

In Fig. 10, the architecture network is displayed. To test the limits of speedy object identification, a quick iteration of YOLO is being developed. Instead of using a neural network with 24 convolutional layers, Fast YOLO employs one with nine, along with fewer filters in each layer. The network size is the sole variable that differs in the testing and training parameters between Fast YOLO and YOLO [24, 36, 37, 38]

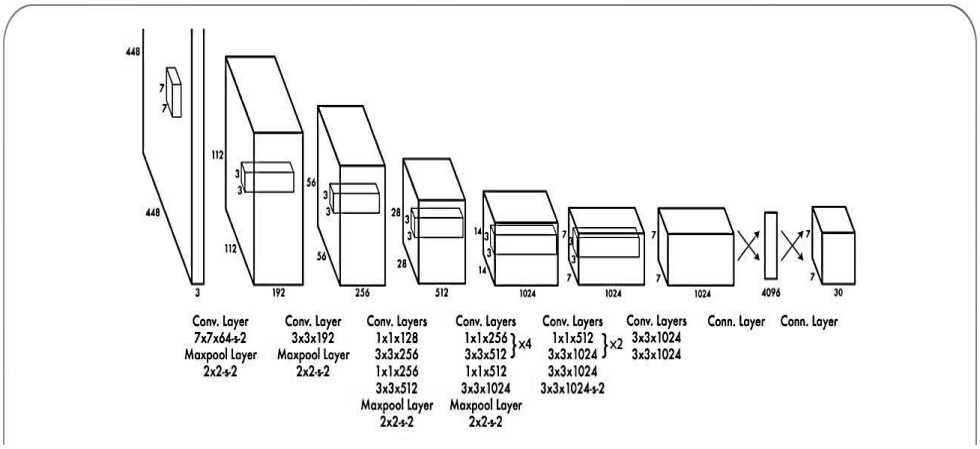


FIGURE 10: The Architecture YOLO V1

YOLO V2 WORK AND ARCHITECTURE

The YOLO9000 algorithm, also known as the YOLO v2 algorithm. It was designed to be quicker and more accurate than YOLO, and to recognize a wider range of item classifications. This version includes makes use of Darknet-19, an alternative CNN backbone constructed utilizing the simple progressive convolution and pooling layers of the VGGNet architecture [35, 38, 39]

One of the main improvements in YOLOV2 is the utilization of anchor boxes. And the most important modifications in YOLOv2 is the use of anchor boxes, a collection of bounding boxes with varied scales and aspect ratios. By merging the anchor boxes and predicted offsets to create the final bounding box, YOLOv2 predicts bounding boxes of anchor boxes. Therefore, a larger range of item sizes and aspect ratios can be handled by the algorithm [35, 37 , 39]

YOLO v2 also features an updated loss function that is more suited for object detection applications. The loss function is built upon the ground truth bounding boxes and class probabilities; it relies on the total of their squared mistakes [35, 38, 39]

By integrating a number of novel methodologies, the improved YOLOv2 surpassed existing systems like SSD and Faster-RCNN in terms of both accuracy and speed. Multi-scale learning, this method enables the network to predict at various input data sizes, offering a trade-off between speed and accuracy.YOLOv2 obtained 67 FPS and 76.8 maps at 416x416 input resolution of data on the VOC 2007 dataset. On the same 544x544 dataset, YOLOv2 obtained 40 FPS and 78.6 maps, as shown in Figure 11 [35, 37, 39]

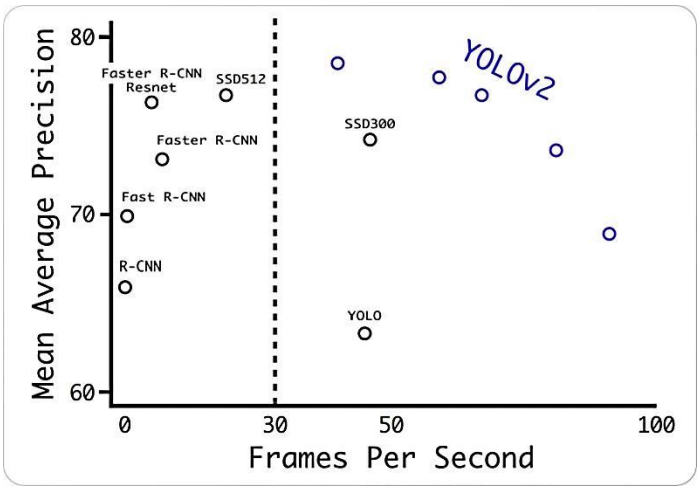


FIGURE 11: Accuracy and Speed YOLO V2

DOI: 10.31695/IJASRE.2023.9.11.7

YOLOv2 Architecture

Darknet-19, YOLOv2's backbone design, contains 5 max-pooling and 19 convolutional layers. Its architecture is influenced by network and employs 1x1 convolutions between the 3x3 to reduce the parameters amount, similar to YOLOv1. Additionally, the employ batch normalization to regularize and aid convergence, as was already mentioned [35 38 39]

The whole Darknet-19 backbone, including the object detection head, is displayed in Table 2. Using the PASCAL VOC dataset, YOLOv2 forecasts 20 classes and 5 bounding boxes with 5 values. The final four convolutional layers are replaced by a single convolutional layer with 1000 filters in the object classification head, which is then followed by a Soft max and a layer of global average pooling. A more powerful YOLOv2 is 5.2 YOLO9000. It used ImageNet classification data to expand the number of categories it can recognize and COCO detection labeled data to learn the coordinates of the bounding box[37 38 39]

The two datasets were combined during training so that when a training image for detection is used, the detection network propagates, and when a training image for classification is utilized, the architecture propagates classification. The outcome is a YOLO system called YOLO9000, which can identify more than 9000 categories [35 39]

Fig. 12 shows the YOLOv2 architecture. The detection head, which is made up of the final 4 convolutional layers and the pass through layer that reorganizes the features of the 17 output of 26x26 x512 into 13x13 x2048 before concatenating with the 25 layer, makes up the Darknet-19 backbone (layers 1 to 23). In order to accommodate 25 predictions (20 classes+5 coordinates) for 5 bounding boxes, while the last convolution creates 125 channels with grid of 13x13[38 39]

Num	Type	Filters	Size/Stride	Output
1	Conv/BN	32	3 × 3 / 1	416 × 416 × 32
2	Max Pool		2 × 2 / 2	208 × 208 × 32
3	Conv/BN	64	3 × 3 / 1	208 × 208 × 64
4	Max Pool		2 × 2 / 2	104 × 104 × 64
5	Conv/BN	128	3 × 3 / 1	104 × 104 × 128
6	Conv/BN	64	1 × 1 / 1	104 × 104 × 64
7	Conv/BN	128	3 × 3 / 1	104 × 104 × 128
8	Max Pool		2 × 2 / 2	52 × 52 × 128
9	Conv/BN	256	3 × 3 / 1	52 × 52 × 256
10	Conv/BN	128	1 × 1 / 1	52 × 52 × 128
11	Conv/BN	256	3 × 3 / 1	52 × 52 × 256
12	Max Pool		2 × 2 / 2	52 × 52 × 256
13	Conv/BN	512	3 × 3 / 1	26 × 26 × 512
14	Conv/BN	256	1 × 1 / 1	26 × 26 × 256
15	Conv/BN	512	3 × 3 / 1	26 × 26 × 512
16	Conv/BN	256	1 × 1 / 1	26 × 26 × 256
17	Conv/BN	512	3 × 3 / 1	26 × 26 × 512
18	Max Pool		2 × 2 / 2	13 × 13 × 512
19	Conv/BN	1024	3 × 3 / 1	13 × 13 × 1024
20	Conv/BN	512	1 × 1 / 1	13 × 13 × 512
21	Conv/BN	1024	3 × 3 / 1	13 × 13 × 1024
22	Conv/BN	512	1 × 1 / 1	13 × 13 × 512
23	Conv/BN	1024	3 × 3 / 1	13 × 13 × 1024
24	Conv/BN	1024	3 × 3 / 1	13 × 13 × 1024
25	Conv/BN	1024	3 × 3 / 1	13 × 13 × 1024
26	Reorg layer 17			13 × 13 × 2048
27	Concat 25 and 26			13 × 13 × 3072
28	Conv/BN	1024	3 × 3 / 1	13 × 13 × 1024
29	Conv	125	1 × 1 / 1	13 × 13 × 125

Figure 12: YOLOv2 Architecture

YOLO v3 Work and Architecture

The YOLO object identification algorithm's third version is known as YOLO v3. It was released in 2018 as an upgrade to YOLO v2 with the goal of enhancing the algorithm's precision and speed [35, 38 40]

One of the most significant breakthroughs in YOLOv3 is the development architecture of CNN known as Darknet-53. It is variation of the architecture of ResNet was designed particularly for tasks of object detection. Its 53 convolutional layers enable it to generate cutting-edge performance on a range of object detection benchmarks [38 40]

Another improvement in YOLO v3 is the addition of anchor boxes with varied scales and aspect ratios. The computer had problems identifying objects of diverse shapes and sizes. YOLO v3 modifies the aspect ratios and scales the anchor boxes to better match the size and shape of the detected objects [35 37 40]

DOI: 10.31695/IJASRE.2023.9.11.7

In additional the YOLO v3 introduces the concept of Feature Pyramid Networks (FPNs), architecture of CNN, which is used to find objects at different area. The build a feature map pyramid and utilize each level to find objects at various scales. YOLO v3 is also capable of handling an aspect ratios and wider range of item sizes. Additionally, compared to earlier YOLO versions, it is more reliable and accurate[35 37 40]

It has been trained on various image dimensions, including 320x320 and 416x416. On a Titan X GPU, YOLOv3 delivers 320x320 resolutions at 28.2 maps at 45 frames per second and has the same accuracy like Single-Shot Detector (SSD321) but is faster for 3 times, as shown in Fig. 13[37 38 40]

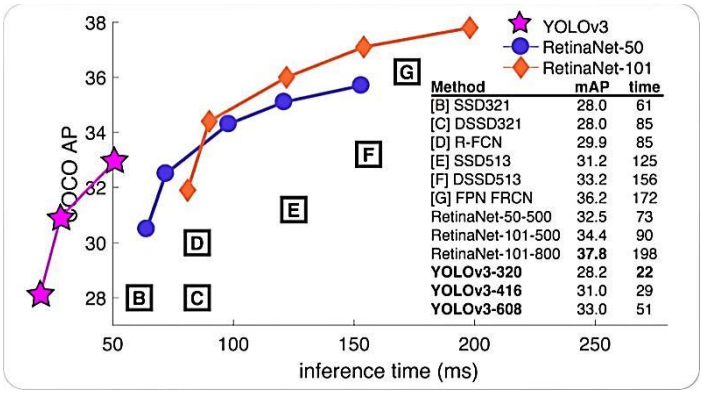


FIGURE 13: YOLO v3 Results Comparison

While being trained and tested on the COCO dataset, the YOLO V3 attempts to address the issues mentioned above and provides a quick and simple way for locating things. While handling smaller objects with ease, this version has trouble producing accurate results for medium- and large-sized objects [37 38 40]

The architecture of Darkent-53 network with 53 convolutional layers that uses 3x3 and 1x1 convolutional filters as well as a few shortcut connections is the foundation for this design. Compared to ReNet-152, it is twice as fast without compromising performance. The general architecture that underpins YOLO V3 is shown in Fig. 14 [35 37 38 40]

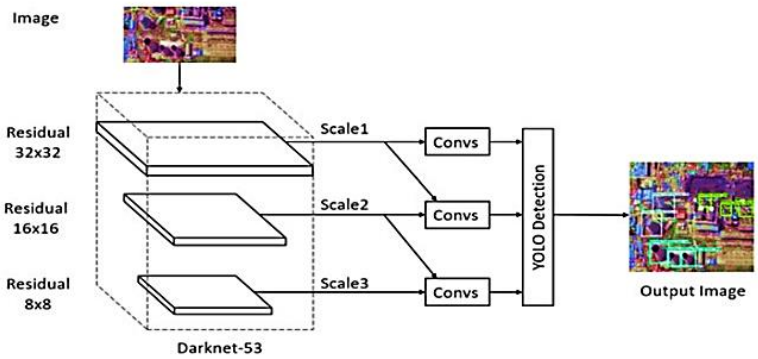


FIGURE 14: Architecture YOLO V3

YOLO V4 WORK AND ARCHITECTURE

Bochkovskiy et al. improved the YOLO object identification algorithm from YOLO v3 to YOLO v4 in 2020.The key improvement from YOLO v3 to YOLO v4 is the implementation of a new CNN architecture called CSPNet as shown in Fig. 15. Specifically designed for object detection applications, the ResNet architecture modification known as "Cross Stage Partial Network. Its structure is fairly simple, with just 54 convolutional layers. On many object detection benchmarks, it can still produce cutting-edge results [37 38 41]

DOI: 10.31695/IJASRE.2023.9.11.7

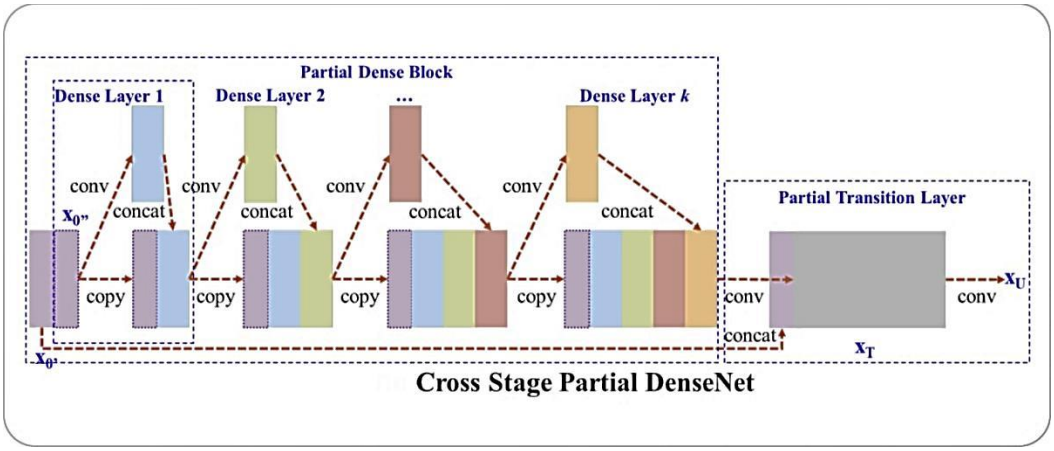


FIGURE 15: CSPNet Architecture

Selecting the right balance between the input network resolution, convolutional layer count overall, parameter count (filter size, channel of filters and groups), and the objective of architectural selection is the number of layer output filters [35 41]

According to numerous tests, the CSPResNext50 categorizes objects on the ILSVRC2012 (ImageNet) dataset considerably better than CSPDarknet53. The CSPDarknet-53 performs better at object detection than CSPResNext50 on the MS COCO dataset. The next stage is to choose more blocks to expand the receptive field and the most effective method for parameter aggregation for different detector levels, including ASFF, FPN, BiFPN, and PAN. For detection, a strong reference model for classification is not always helpful, the following are necessary for the detector as opposed to the classifier:[35 37 41]

- Increased input network size resolution for detecting a lot of little objects.
- More layers to cover a greater receptive area and a larger input network.
- Extra parameters to help a model be better able to differentiate many objects of various sizes in a single image. Theoretically, we can choose a model with a larger receptive field size (with more 3x3 convolutional layers) and more parameters as the basis.

Larger input network resolution for detecting many little things.

- Full object viewing is possible up to object size.
- Up to network size - enables monitoring of the environment in which the object is located.
- As the network size increases, more connections are made between the picture point and the end activation.

The receptive field is enhanced by the SPP block, which is preferred over the CSPDarknet53 block, causes little to no network operational speed decrease and separates out the most critical context elements. Instead of the FPN used in YOLOv3, it is used PANet to aggregate parameters from different backbone levels for different detector levels. Finally, choose the backbone of CSPDarknet53, path aggregation neck of PANet, anchor of YOLOv3 based head for the architecture of YOLOv4 and extra module for SPP as shown in Fig. 16 [35 38 41]

DOI: 10.31695/IJASRE.2023.9.11.7

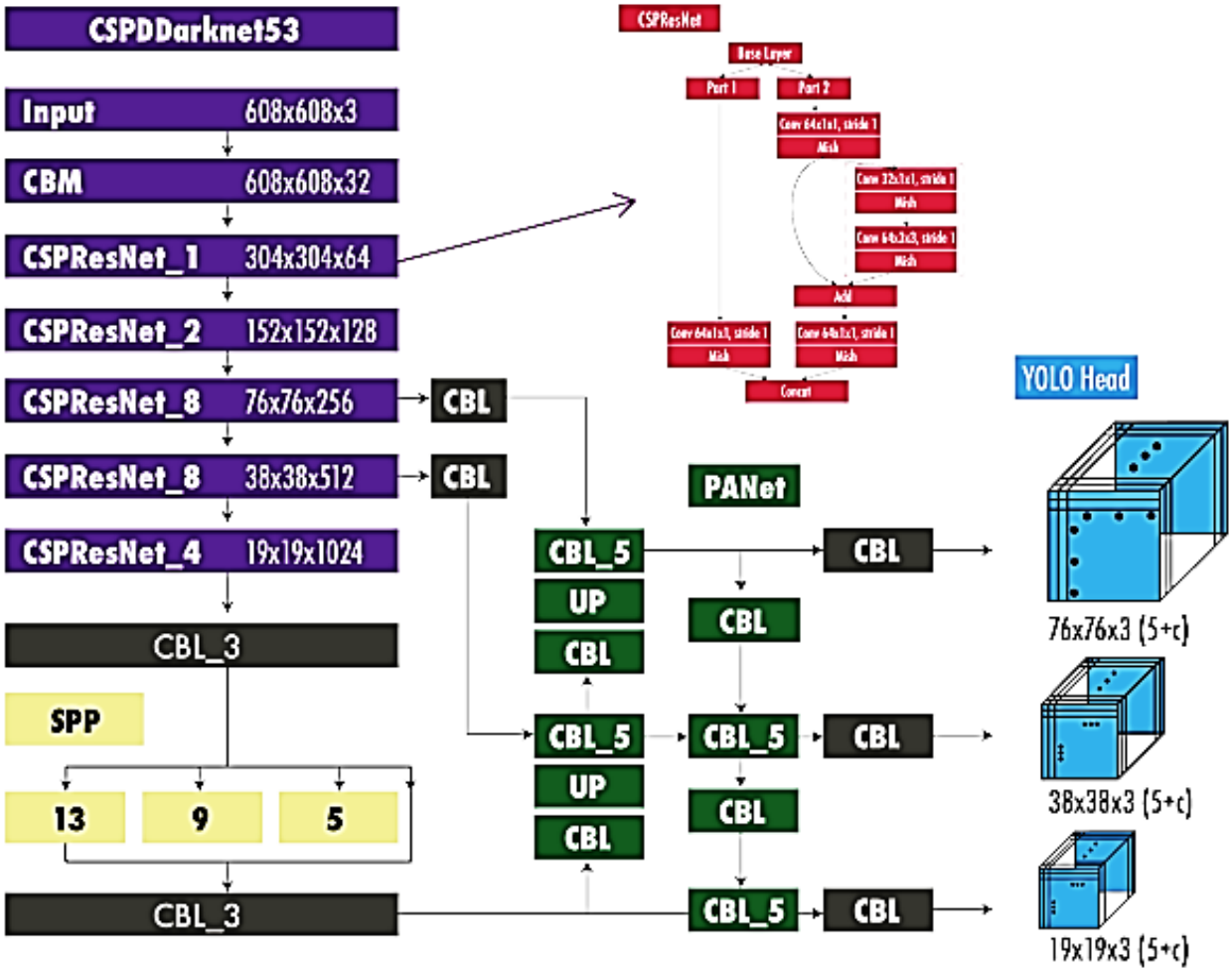


FIGURE 16: YOLOv4 Architecture for Object Detection

YOLO V5 WORK AND ARCHITECTURE

The same group that created the original YOLO algorithm released YOLO v5 as an open-source project in 2020. Through the addition of numerous new features and improvements, YOLO v5 builds on the success of earlier editions[37 38 44 45]

The Efficient Det design, which is based on the Efficient Net network architecture as shown in Fig. 17, is more complex than YOLO and is used in YOLO v5. With YOLO v5, a more complex architecture was used, resulting in improved generalization and accuracy to a wider range of types of object [35 38 44 46]

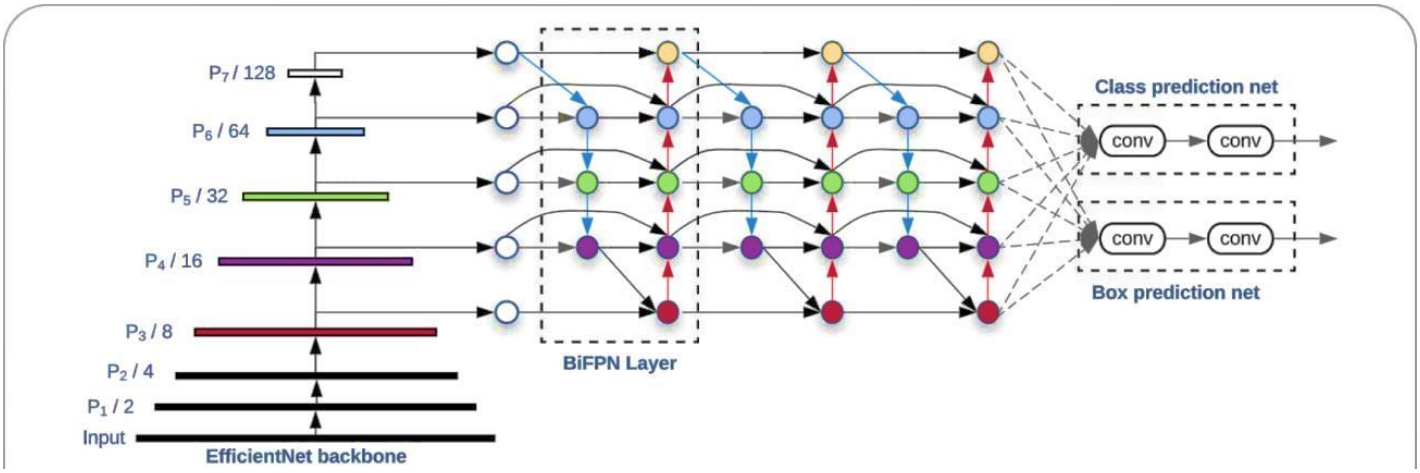


FIGURE17: Architecture of Efficient Det Model.

DOI: 10.31695/IJASRE.2023.9.11.7

The object identification model developed for YOLO and YOLO v5 uses different training data. YOLO was trained using the 20 different object categories of the PASCAL VOC dataset. On the other hand, D5, a bigger and more diversified dataset with a total of 600 item types, was used to train YOLO v5 [44 45 46]

The anchor boxes in YOLO v5 are constructed using a novel technique called dynamic anchor boxes. The centroids of these clusters are employed as anchor boxes once the ground truth bounding boxes have been divided into clusters using a clustering technique. Finally, result, the size and shape of the identified items can be more closely approximated by the anchor boxes.

Additionally, the idea of spatial pyramid pooling (SPP) is introduced in YOLO v5, a kind of pooling layer used to lower the feature maps' spatial resolution. Since SPP enables the model to view things at various scales, it is employed to enhance the detection performance for small items. SPP is used by YOLO v4 as well, although YOLO v5 makes a number of changes to the SPP design to enhance performance [35 37 44]

YOLOv5 ARCHITECTURE

The system's brain is a modified CSPDarknet53 with a Stem at its center, a large-window stride convolution layer to conserve memory and processing resources; then, convolutional layers are applied to the input image to extract relevant properties. The layer known as spatial pyramid pooling fast (SPPF), while processing the data at various scales, the succeeding convolution layers and up-sample layers increase the resolution of the feature maps[44 45 46]

The SPPF layer pools feature from multiple scales into a fixed-size feature map in order to accelerate network computation. Batch normalization (BN) and SiLU activation come after each convolution. The head is similar to YOLOv3, while the neck makes use of SPPF and a modified CSP-PAN. YOLOv5 makes use of a number of augmentations from the augmentations package, including Mosaic, copy-paste, random affine, MixUp, HSV augmentation, random horizontal flip, and others. Additionally, it enhances grid sensitivity, making it more resistant to runaway gradients [35 37 44 45]

In order to accommodate different applications and hardware specifications, YOLOv5 offers five scaled versions: YOLOv5x (extra-large), YOLOv5l (large), YOLOv5m (medium), YOLOv5s (small) and YOLOv5n (nano). These versions differ in the width and depth of the convolution modules. For example, the lightweight variants YOLOv5n and YOLOv5s are designed for low-resource devices, but the high performance and albeit slower, as shown in Fig. 18 [37 35 44 46]

DOI: 10.31695/IJASRE.2023.9.11.7

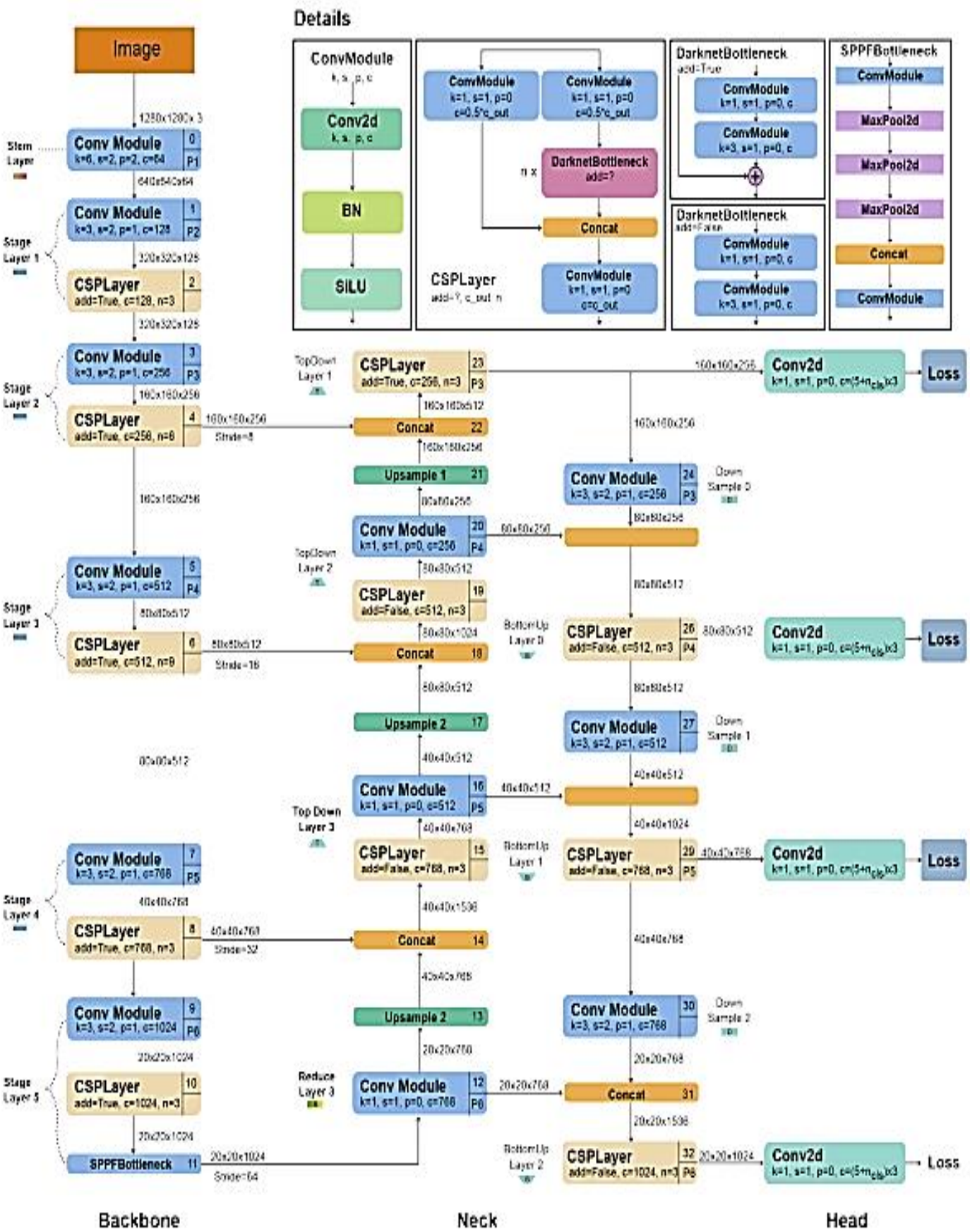


FIGURE 18: YOLOv5 Architecture

DOI: 10.31695/IJASRE.2023.9.11.7

YOLO v6 Work and Architecture

In 2022, Li et al. put forth YOLO v6 as an improvement over past iterations. One of the key differences between YOLOv5 andYOLOv6 is the architecture of CNN are used. A variant of the architecture of EfficientNet utilized by YOLOv6 is EfficientNet-L2. Compared to the EfficientDet architecture used in YOLO v5, it is more efficient computationally and has fewer parameters. On several object detection benchmarks, it can achieve cutting-edge performance. The structure of the YOLO v6 model as shown Fig. 19 [353847]

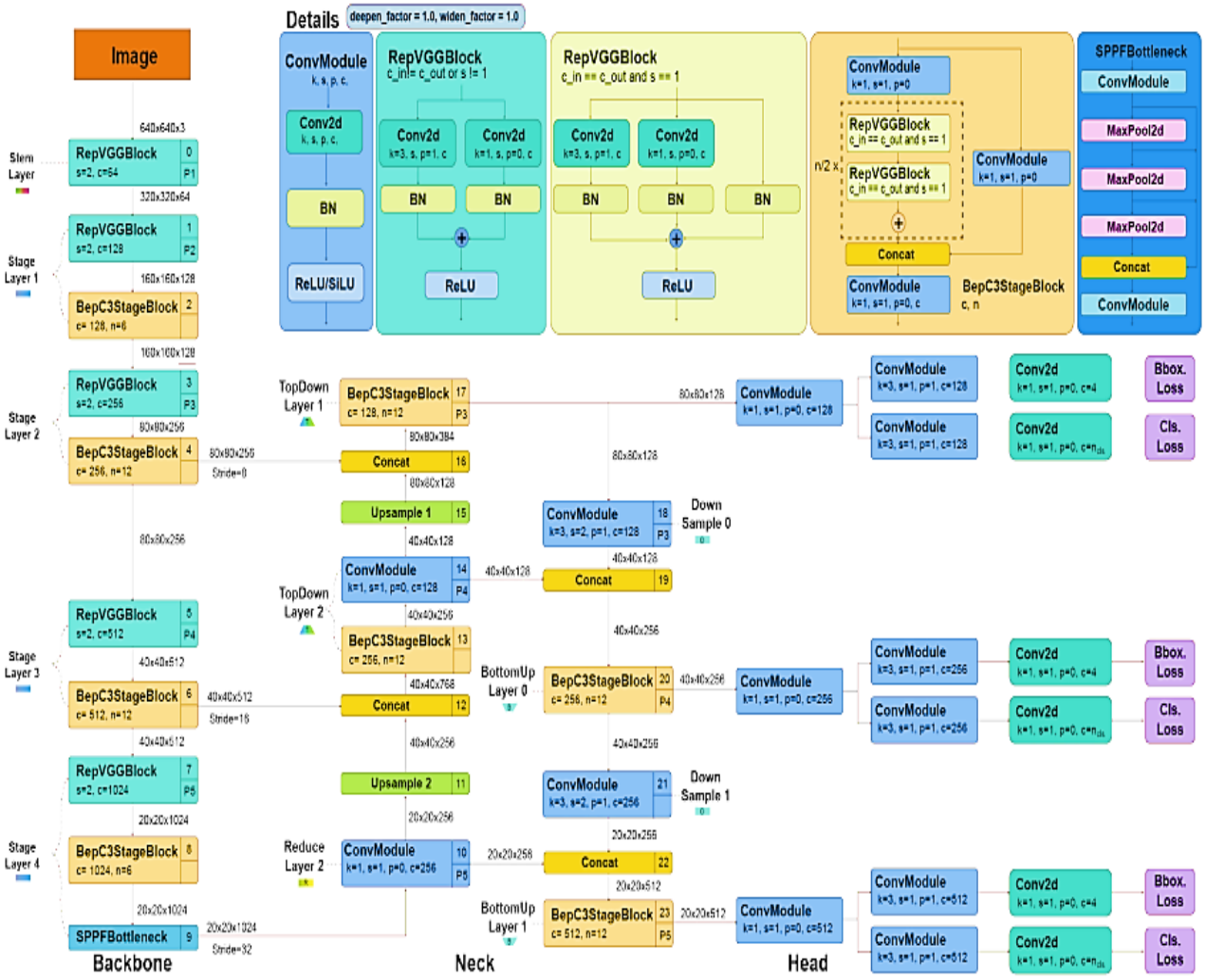


Figure 19: The Structure of the YOLO v6 Model

Method

YOLOv6's redesigned design includes the following elements: network design, label assignment, loss function, data augmentation, industry-friendly changes, deployment and quantization (see Fig. 20): [21 37 38 47]

• **Network Design Backbone**: RepVGG backbones are found to have more feature representation power in small networks compared to other common architectures at comparable inference speeds, but they can hardly be scaled up to obtain larger models due to the computational costs and exponential parameters growth.

• **Label Assignment:** a number of trials were used to assess the most recent developments in label assignment strategies on YOLOv6, and the findings show that TAL is more efficient and user-friendly for training.

• **Loss Function:** Classification loss, box regression loss, and object loss are all components of the loss functions of common anchor-free object detectors. Choose VariFocal Loss as our classification loss and SIoU/GIoU Loss as our regression loss after thoroughly testing each loss with all methods available.

DOI: 10.31695/IJASRE.2023.9.11.7

• **Industry-handy improvements:** Introduce more standard techniques, such as self-distillation, and training epochs to boost performance. Both classification and box regression for self-distillation are, respectively, supervised by the instructor model. Thanks to DFL, box regression can be reduced to its essence. Additionally, the amount of knowledge from the soft and hard labels is dynamically decreased by cosine decay, which aids the student in acquiring knowledge selectively at various stages of training.

• **Quantization and deployment:** To restore the performance of models that rely on quantizing parameters, train YOLOv6 with an optimizer to produce weights that are PTQ-friendly. To achieve extreme performance, combine QAT with channel-wise distillation and graph optimization.

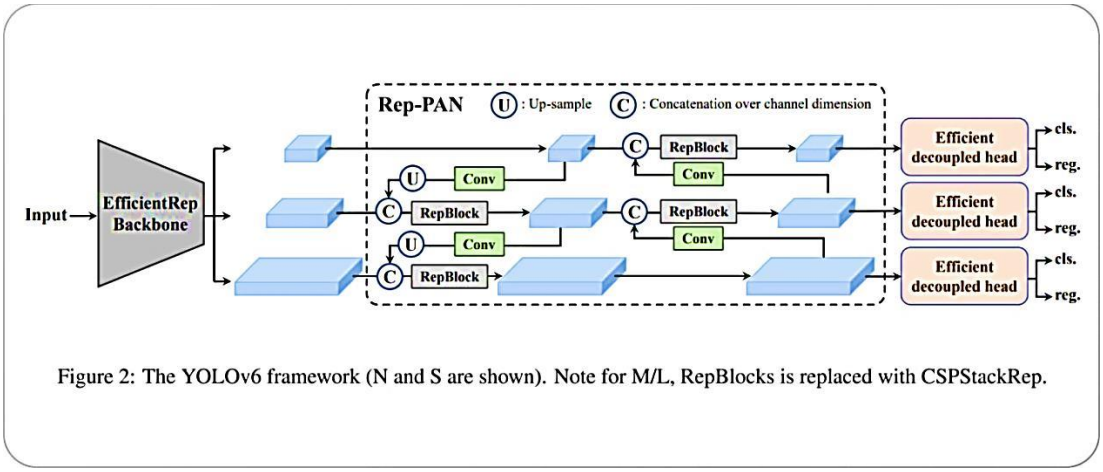
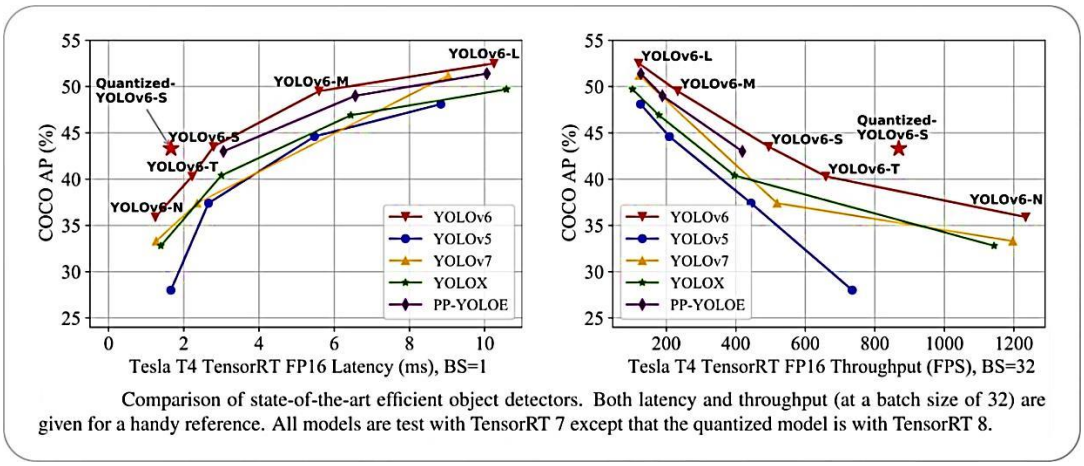


Figure 2: The YOLOv6 framework (N and S are shown). Note for M/L, RepBlocks is replaced with CSPStackRep.

FIGURE 20: YOLOv6 Overview.

Additionally, YOLO v6 introduces thick anchor boxes, a brand-new method for building anchor boxes. The Fig. 21 displays the outcomes of YOLO v6 in comparison to other state-of-the-art methods.



Comparison of state-of-the-art efficient object detectors. Both latency and throughput (at a batch size of 32) are given for a handy reference. All models are test with TensorRT 7 except that the quantized model is with TensorRT 8.

FIGURE 21: Compared Cutting-Edge Techniques and YOLO v6 Results.

YOLO V7 WORK AND ARCHITECTURE

One version of YOLO, referred to as YOLO v7, includes a number of improvements over earlier versions. The major development is the use of anchor boxes. It is goal, which is a collection of preconfigured boxes with many aspect ratios, is to recognize objects of varied shapes [21 35 48]

YOLO V7 WORK

In YOLO v7, a brand-new loss function called focused loss has been included, which is a significant improvement. In early iterations of YOLO, The typical cross-entropy loss function, which is well known to be less successful at identifying small objects, was employed. By de-weighting the loss for well-classified samples and concentrating on the challenging examples and difficult-to-detect objects, focal loss resolves this problem, as shown in Fig. 22 [37 38 48]

DOI: 10.31695/IJASRE.2023.9.11.7

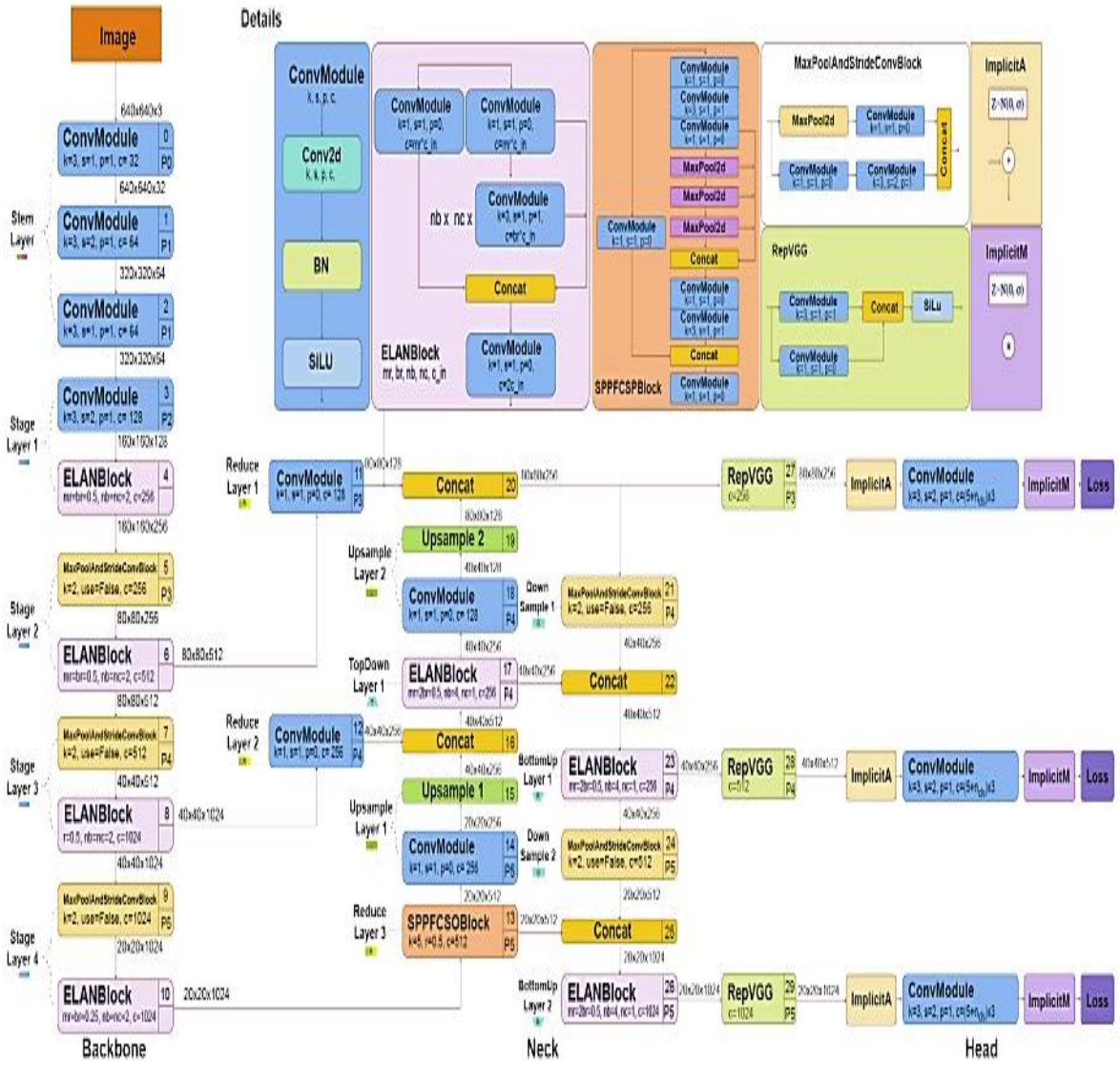


FIGURE 22: YOLO v7 Work

One of the main advantages of YOLO v7 is its quickness. When evaluating photos, 155 frames per second is noticeably faster than other state-of-the-art object detection algorithms. The peak processing speed of even the most basic YOLO model was 45 frames per second. This makes it suitable for delicate real-time applications needing quicker processing speeds, like surveillance and driverless cars, as seen in Fig. 23 [35, 37 ,38, 48]

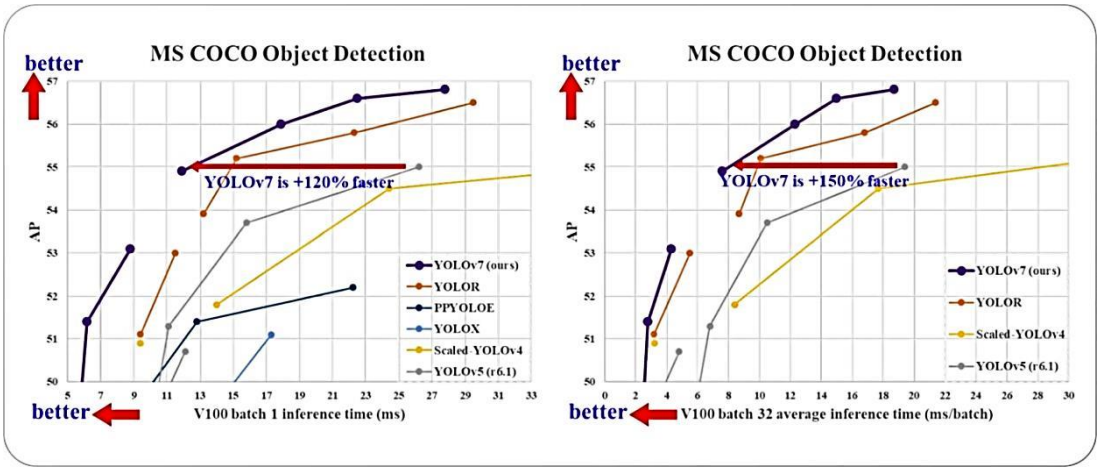


FIGURE 23: YOLO Model had a Peak Processing Speed of 45 Frames Per Second

A comparison of YOLO v7 performance and inference speed with modern, cutting-edge real-time object detectors Average Precision (AP).

DOI: 10.31695/IJASRE.2023.9.11.7

YOLO v7 outperforms other object detection algorithms in terms of accuracy. At an IoU (intersection over union) threshold of 0.5, it obtains an average precision of 37.2% on the well-known COCO dataset, This is equivalent to other cutting-edge object detection methods, Fig. 24 displays the quantitative performance comparison[38 48]

Model	#Param.	FLOPs	Size	AP^{val}	AP^{val}_{50}	AP^{val}_{75}	AP^{val}_{S}	AP^{val}_{M}	AP^{val}_{L}
YOLOv4 [3]	64.4M	142.8G	640	49.7%	68.2%	54.3%	32.9%	54.8%	63.7%
YOLOR-u5 (r6.1) [81]	46.5M	109.1G	640	50.2%	68.7%	54.6%	33.2%	55.5%	63.7%
YOLOv4-CSP [79]	52.9M	120.4G	640	50.3%	68.6%	54.9%	34.2%	55.6%	65.1%
YOLOR-CSP [81]	52.9M	120.4G	640	50.8%	69.5%	55.3%	33.7%	56.0%	65.4%
YOLOv7	36.9M	104.7G	640	**51.2%**	**69.7%**	**55.5%**	**35.2%**	**56.0%**	**66.7%**
improvement	-43%	-15%	-	+0.4	+0.2	+0.2	+1.5	=	+1.3
YOLOR-CSP-X [81]	96.9M	226.8G	640	52.7%	**71.3%**	57.4%	36.3%	57.5%	68.3%
YOLOv7-X	71.3M	189.9G	640	**52.9%**	71.1%	**57.5%**	**36.9%**	**57.7%**	**68.6%**
improvement	-36%	-19%	-	+0.2	-0.2	+0.1	+0.6	+0.2	+0.3
YOLOv4-tiny [79]	6.1	6.9	416	24.9%	42.1%	25.7%	8.7%	28.4%	39.2%
YOLOv7-tiny	6.2	5.8	416	**35.2%**	**52.8%**	**37.3%**	**15.7%**	**38.0%**	**53.4%**
improvement	+2%	-19%	-	+10.3	+10.7	+11.6	+7.0	+9.6	+14.2
YOLOv4-tiny-3l [79]	8.7	5.2	320	30.8%	47.3%	32.2%	**10.9%**	31.9%	51.5%
YOLOv7-tiny	6.2	3.5	320	**30.8%**	**47.3%**	**32.2%**	10.0%	**31.9%**	**52.2%**
improvement	-39%	-49%	-	=	=	=	-0.9	=	+0.7
YOLOR-E6 [81]	115.8M	683.2G	1280	55.7%	73.2%	60.7%	40.1%	**60.4%**	69.2%
YOLOv7-E6	97.2M	515.2G	1280	**55.9%**	**73.5%**	**61.1%**	**40.6%**	60.3%	**70.0%**
improvement	-19%	-33%	-	+0.2	+0.3	+0.4	+0.5	-0.1	+0.8
YOLOR-D6 [81]	151.7M	935.6G	1280	56.1%	73.9%	61.2%	**42.4%**	60.5%	69.9%
YOLOv7-D6	154.7M	806.8G	1280	56.3%	73.8%	61.4%	41.3%	60.6%	70.1%
YOLOv7-E6E	151.7M	843.2G	1280	**56.8%**	**74.4%**	**62.1%**	40.8%	**62.1%**	**70.6%**
improvement	=	-11%	-	+0.7	+0.5	+0.9	-1.6	+1.6	+0.7

FIGURE 24: A Comparison of YOLO v7 Performance and Inference Speed with Modern

Faster R-CNN and Mask R-CNN two-stage detectors, Based on the COCO dataset tend to obtain higher average precision. It should be stressed that are less precise than YOLO v7 and require longer inference durations.

LIMITATIONS OF YOLO V7

Although YOLO v7 is a strong and successful object identification system, it does have several drawbacks.

1. The YOLO v7 struggles to identify small objects, like many object recognition algorithm design. It might not be able to detect things properly in crowded areas or at a distance from the camera.
2. The YOLO v7 has trouble correctly detecting objects of different sizes. As a result, it could be difficult to distinguish objects that are significantly larger or smaller than the other objects in the scene.
3. The YOLO v7 could be challenging to use in practical applications where the lighting conditions could shift It could be delicate to alterations in illumination or other environmental elements.
4. The YOLO v7 may require a lot of computing making real-time operation on devices with low resources difficult, like cell phones or other cutting-edge technology [48]

YOLO V8 WORK AND ARCHITECTURE

Ultralytics has confirmed the existence of YOLO v8, which promises to have more features and run faster than its predecessors, on CPU, and GPU devices, training and inference are made much simpler by the new API in YOLO v8. The developers are currently working on a scholarly publication that will give a thorough description of the model design and performance, even though prior YOLO variations are still supported by the framework [21 37 49]

Because YOLOv8 does not yet allow models trained at 1280-pixel resolution, it is not advisable to use it if you want to execute inference at a high resolution. See Fig. 25 [37 38 49]

DOI: 10.31695/IJASRE.2023.9.11.7

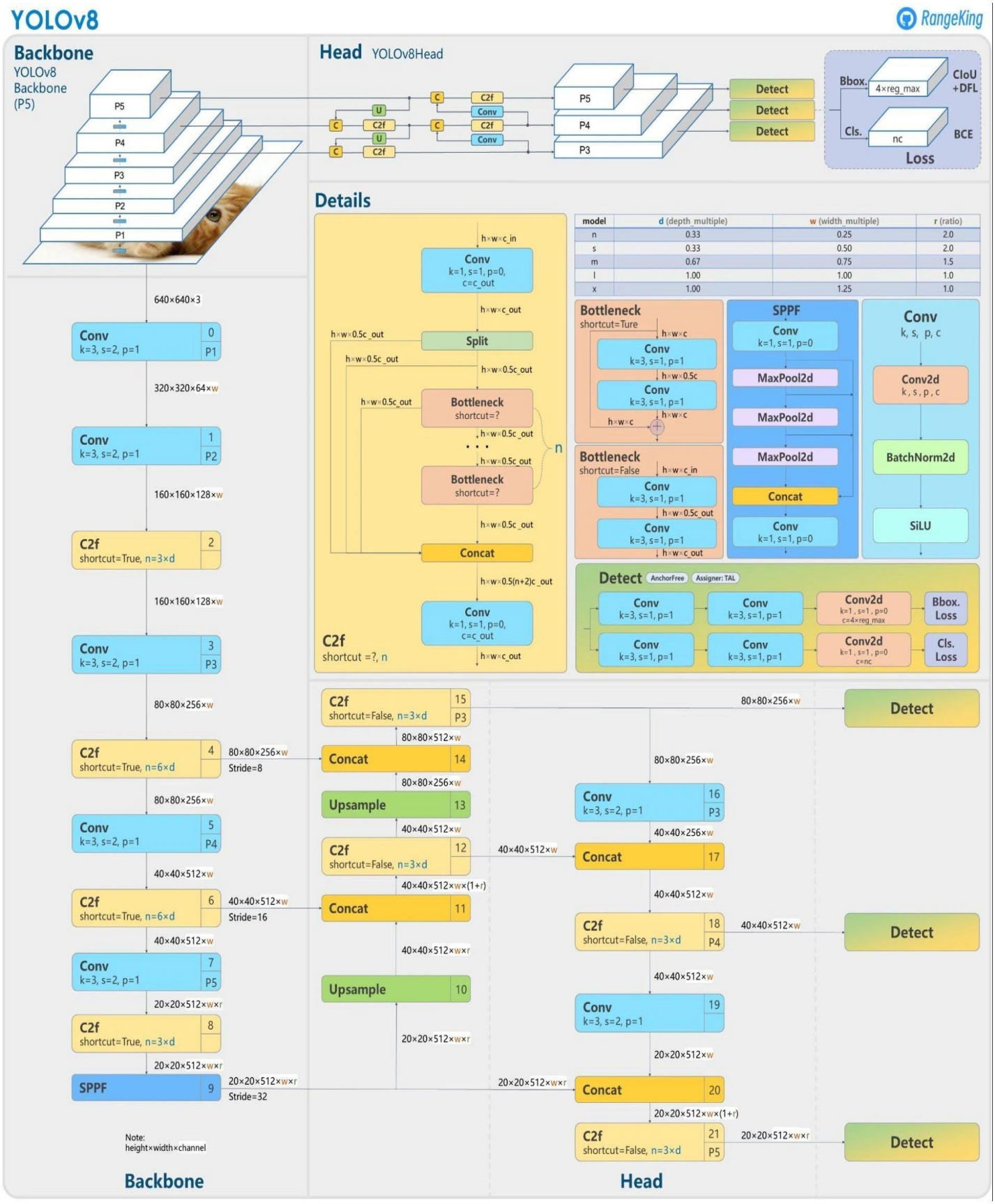


FIGURE 25: YOLOv8 Work and Architecture

YOLOv8 CONVOLUTIONS

According to the inaugural post from Ultralytics, the YOLOv8 architecture has undergone a number of modifications and new convolutions: [35 37 49]

DOI: 10.31695/IJASRE.2023.9.11.7

- With the replacement of C3 with C2f, the system's core experienced alterations. The stem's initial 6x6 convolution was changed to a 3x3 convolution. While just the output from the final Bottleneck was used in C3, C2f combines the outputs from the Bottleneck (which is made up of two 3x3 convolution with residual connections).
- The YOLOv5 configuration's convolutions #10 and #14 were eliminated.
- The Bottleneck in YOLOv8 is identical to that in YOLOv5, with the exception that the size of the first convolution's kernel was increased from 1x1 to 3x3. This modification denotes a move in favor of the ResNet block identified in 2015.
- When an object detection model predicts the center of an object directly, as opposed to the offset from a predetermined anchor box, this is known as "anchor-free detection."
- To locate object classes with the required aspect ratio, anchor boxes, and scale, the use of a predefined set of boxes with fixed heights and widths. Selected based on the size of the objects in the training dataset and tiled across the image during detection.
- The network generates probability and properties for each tiled box, such as background, IoU, and offsets, which are utilized to modify the anchor boxes. Different object sizes can have several anchor boxes established as fixed beginning locations for boundary box hypotheses.

Because it is more adaptable and effective, anchor-free detection has an advantage, due to the lack of manual anchor box standard requirements, This can be challenging to select and can produce less-than-ideal outcomes in earlier YOLO models, such as v1 and v2.

CONCLUSION

This article gave a timeline of YOLO's development, between YOLOv1 and YOLOv8, and talked about its new features, apps, and network architecture. also comparing the main differences among the YOLO version's Architecture and will discuss its evolution from YOLO to YOLOv8, its network architecture, new features, and applications. And starts by looking at the basic ideas and design of the first YOLO model, which laid the groundwork for the following improvements in the YOLO family. In addition, this article also provided a step-by-step guide on how to use the YOLO version architecture, In addition, included in this article was a step-by-step tutorial on how to utilize YOLOv8 for object identification and model-assisted image creation. The release of YOLO v8 has been confirmed by Ultralytics as of the time this article was written, and it offers new features and better performance than its predecessors.

REFERENCE

1. Albawi, S., Mohammed, T. A., Al-Zawi, S. (2017, August). Understanding of a convolutional neural network. In 2017 International Conference on Engineering and Technology (ICET) :1-6
2. Zaytar MA, El Amrani C (2016) Sequence to sequence weather forecasting with long short-term memory recurrent neural networks. Int J Comput Appl 143(11):7–11
3. Ye A, Pang B, Jin Y, Cui J (2020) A YOLO-based neural network with VAE for intelligent garbage detection and classification. In 2020 3rd international conference on algorithms computing and artificial intelligence pp 1-7
4. Khan A, Sohail A, Zahoora U, Qureshi AS (2020) A survey of the recent architectures of deep convolutional neural networks. Artif Intell Rev 53(8):5455–5516. https://doi.org/10.1007/s10462-018-9633-3
5. Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2021). ByteTrack: Multi-Object Tracking by Associating Every Detection Box. *ArXiv*. /abs/2110.06864
6. Zhang, Y., Wang, C., Wang, X., Zeng, W., & Liu, W. (2021). FairMOT: On the Fairness of Detection and Re-Identification in Multiple Object Tracking. *ArXiv*. https://doi.org/10.1007/s11263-021-01513-4
7. Xie, X., Cheng, G., Wang, J., Yao, X., & Han, J. (2021). Oriented R-CNN for Object Detection. *ArXiv*. /abs/2108.05699

DOI: 10.31695/IJASRE.2023.9.11.7

8. Atik, S. O., & Ipbuker, C. (2021). Integrating Convolutional Neural Network and Multiresolution Segmentation for Land Cover and Land Use Mapping Using Satellite Imagery. Applied Sciences, 11(12), 5551. https://doi.org/10.3390/app11125551.
9. Gillani, I.S., Munawar, M.R., Talha, M., Azhar, S., Mashkoor, Y., uddin, M.S., & Zafar, U. (2022). Yolov5, Yolo-x, Yolo-r, Yolov7 Performance Comparison: A Survey. Artificial Intelligence and Fuzzy Logic System.
10. Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. ArXiv. /abs/1506.01497
11. He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN. *ArXiv*. /abs/1703.06870
12. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi. (2016). A. You only look once: Unified, real-time object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30. 10.1109/CVPR.2016.91.
13. Redmon, J.; Farhadi, A. (2017). YOLO9000: Better, faster, stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Honolulu, HI, USA, 21–26. 10.1109/CVPR.2017.690.
14. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *ArXiv*. /abs/1804.02767.
15. W. Lan, J. Dang, Y. Wang and S. Wang, (2018) "Pedestrian Detection Based on YOLO Network Model," 2018 IEEE International Conference on Mechatronics and Automation (ICMA), Changchun, China, pp. 1547-1551, doi: 10.1109/ICMA.2018.8484698.
16. W. -Y. Hsu and W. -Y. Lin, (2021) "Adaptive Fusion of Multi-Scale YOLO for Pedestrian Detection," in IEEE Access, vol. 9, pp. 110063-110073, doi: 10.1109/ACCESS.2021.3102600.
17. Benjumea, A., Teeti, I., Cuzzolin, F., & Bradley, A. (2021). YOLO-Z: Improving small object detection in YOLOv5 for autonomous vehicles. ArXiv. /abs/2112.11798.
18. N. M. A. A. Dazlee, S. A. Khalil, S. Abdul-Rahman, and S. Mutalib, (2021). "Object detection for autonomous vehicles with sensor-based technology using yolo," International Journal of Intelligent Systems and Applications in Engineering, vol. 10, no. 1, pp. 129–134.
19. S. Liang, H. Wu, L. Zhen, Q. Hua, S. Garg, G. Kaddoum, M. M. Hassan, and K. Yu, (2022) "Edge yolo: Real-time intelligent object detection system based on edge-cloud cooperation in autonomous vehicles," IEEE Transactions on Intelligent Transportation Systems, vol. 23, no. 12, pp. 25345–25360.
20. Q. Li, X. Ding, X. Wang, L. Chen, J. Son, and J.-Y. Song, (2021). "Detection and identification of moving objects at busy traffic road based on yolo v4," The Journal of the Institute of Internet, Broadcasting and Communication, vol. 21, no. 1, pp. 141–148.
21. Terven, J. (2023). A Comprehensive Review of YOLO: From YOLOv1 and Beyond. ArXiv. /abs/2304.00501
22. Diwan, T., Anirudh, G. & Tembhurne, J.V., (2023) Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimed Tools Appl* **82**, 9243–9275. https://doi.org/10.1007/s11042-022-13644-y
23. Rastogi A, Ryuh BS (2019) Teat detection algorithm: YOLO vs Haar-cascade. J Mech Sci Technol 33(4):1869–1874
24. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You Only Look Once: Unified, Real-Time Object Detection. ArXiv. /abs/1506.02640
25. Y. Tian, G. Yang, Z. Wang, H. Wang, E. Li, and Z. Liang, (2019), "Apple detection during different growth stages in orchards using the improved yolo-v3 model," Computers and electronics in agriculture, vol. 157, pp. 417–426.
26. D. Wu, S. Lv, M. Jiang, and H. Song, (2020) "Using channel pruning-based yolo v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments," Computers and Electronics in Agriculture, vol. 178, p. 105742.
27. M. Lippi, N. Bonucci, R. F. Carpio, M. Contarini, S. Speranza, and A. Gasparri, (2021)"A yolo-based pest detection system for precision agriculture," in 2021 29th Mediterranean Conference on Control and Automation (MED), pp. 342–347, IEEE.

DOI: 10.31695/IJASRE.2023.9.11.7

28. W. Yang and Z. Jiachun, (2018) "Real-time face detection based on yolo," IEEE international conference on knowledge innovation and invention (ICKII), pp. 221–224, IEEE, 2018.
29. W. Chen, H. Huang, S. Peng, C. Zhou, and C. Zhang, (2021), "Yolo-face: a real-time face detector," The Visual Computer, vol. 37, pp. 805–813.
30. M. A. Al-Masni, M. A. Al-Antari, J.-M. Park, G. Gi, T.-Y. Kim, P. Rivera, E. Valarezo, M.-T. Choi, S.-M. Han, and T.-S. Kim, (2018), "Simultaneous detection and classification of breast masses in digital mammograms via a deep learning yolo-based cad system," Computer methods and programs in biomedicine, vol. 157, pp. 85–94.
31. Y. Nie, P. Sommella, M. O'Nils, C. Liguori, and J. Lundgren, (2019), "Automatic detection of melanoma with yolo deep convolutional neural networks," E-Health and Bioengineering Conference (EHB), pp. 1–4, IEEE,.
32. H. M. Ünver and E. Ayan, (2019) "Skin lesion segmentation in dermoscopic images with combination of yolo and grabcut algorithm," Diagnostics, vol. 9, no. 3, p. 72.
33. L. Tan, T. Huangfu, L. Wu, and W. Chen, (2021), "Comparison of retinanet, ssd, and yolo v3 for real-time pill identification," BMC medical informatics and decision making, vol. 21, pp. 1–11.
34. L. Cheng, J. Li, P. Duan, and M. Wang, (2021) "A small attentional yolo model for landslide detection from satellite remote sensing images," Landslides, vol. 18, no. 8, pp. 2751–2765.
35. R. Girshick, "Fast r-cnn, (2015)," in Proceedings of the IEEE international conference on computer vision, pp. 1440–1448. https://doi.org/10.1016/j.procs.2022.01.135
36. M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. (2015), The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision, 111(1):98–136.
37. Terven, J. (2023). A Comprehensive Review of YOLO: From YOLOv1 and Beyond. *ArXiv*. /abs/2304.00501
38. Diwan, T., Anirudh, G. & Tembhurne, J.V. (2023), Object detection using YOLO: challenges, architectural successors, datasets and applications. *Multimed Tools Appl* **82**, 9243–9275. https://doi.org/10.1007/s11042-022-13644-y
39. Redmon J & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger. ArXiv. /abs/1612.08242,https://doi.org/10.48550/arXiv.1612.08242
40. Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. *ArXiv*. /abs/1804.02767
41. Roy, A. M., Bose, R., & Bhaduri, J. (2021). A fast accurate fine-grain object detection model based on YOLOv4 deep neural network. *ArXiv*. /abs/2111.00298
42. Bochkovskiy, A., Wang, C., & Liao, H. (2020). YOLOv4: Optimal Speed and Accuracy of Object Detection. *ArXiv*. /abs/2004.10934
43. Wang, C., Liao, H., Yeh, I., Wu, Y., Chen, P., & Hsieh, J. (2019). CSPNet: A New Backbone that can Enhance Learning Capability of CNN. *ArXiv*. /abs/1911.11929
44. GitHub - ultralytics/yolov5: YOLOv5 □ in PyTorch > ONNX > CoreML > TFLite
45. Tan, M., Pang, R., & Le, Q. V. (2019). EfficientDet: Scalable and Efficient Object Detection. *ArXiv*. /abs/1911.09070
46. Wang, J. Zhao, N. Ta, X. Zhao, M. Xiao, and H. Wei, (2021) "A real-time deep learning forest fire monitoring algorithm based on an improved pruned kd model," Journal of Real-Time Image Processing, vol. 18, no. 6, pp. 2319–2329.
47. Li, C., Li, L., Jiang, H., Weng, K., Geng, Y., Li, L., Ke, Z., Li, Q., Cheng, M., Nie, W., Li, Y., Zhang, B., Liang, Y., Zhou, L., Xu, X., Chu, X., Wei, X., & Wei, X. (2022). YOLOv6: A Single-Stage Object Detection Framework for Industrial Applications. *ArXiv*. /abs/2209.02976
48. Wang, C., Bochkovskiy, A., & Liao, H. (2022). YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *ArXiv*. /abs/2207.02696
49. G. Jocher, A. Chaurasia, and J. Qiu, (2023), "YOLO by Ultralytics." https://github.com/ultralytics/ ultralytics, 2023.

DOI: 10.31695/IJASRE.2023.9.11.7