Mastering All YOLO Models from YOLOv1 to YOLO-NAS: Papers Explained (2024)

soumyadip

DECEMBER 26, 2023LEAVE A COMMENT

Computer VisionObject DetectionYOLO

What is YOLO? **You Only Look Once (YOLO)**: Unified, Real-Time Object Detection is a single-stage object detection model published at CVPR 2016, by Joseph Redmon, very famous for having very low latency and high accuracy. The entire YOLO series of models is a collection of pioneering concepts that have shaped today's object detection methods. YOLO Models has emerged as an industry de facto, achieving high detection precision with minimal computational demands. Some **YOLO models** are tailored to align with the specific processing capabilities of the device, whether it's a CPU or a GPU. Most YOLO models are designed to cater to different scales, such as small, medium, and large, which can be easily serialized in ONNX, TensorRT, OpenVINO, etc. This gives users the liberty to choose which is best suited for their application.

In this article, we will go through all the different versions of YOLO, from the original YOLO to YOLOv8 and YOLO-NAS, and try to understand their internal workings, architecture, design choices, improvements, and custom training. There are limited resources on the internet that go over all the YOLO models, from their inner workings to how to train every model on the data of your choice end-to-end.

Introduction to Object Detection

Object detection is an important task in computer vision. In layman's terms, Object detection is defined as Object Localization + Object Classification. Object Localization is the method of locating an object in the image using a bounding box and Object Classification is the method that tells what is in that bounding box.

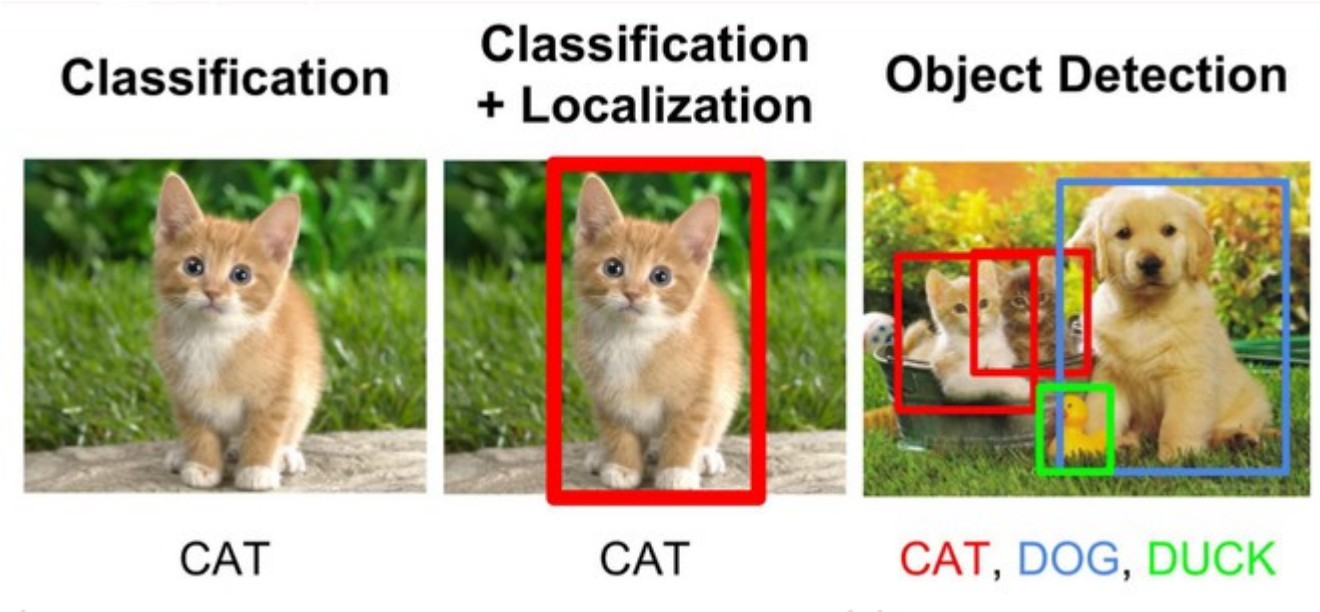


Figure 1: Difference between Object Classification, localization and detection. (Source)

There are lots of real-life applications for object detection, in the field of **Autonomous Vehicles**, it is used for detecting vehicles, pedestrians, road delimiter prediction, **HD-Map** creation, traffic lights, and traffic signs, etc., in **surveillance and monitoring**, it is used in detecting trespassers, vehicle license plates, face mask detection, weapon detection etc. in biometric attendance systems, In **medical imaging**, its used for detecting certain cells, cancer detection, tumor detection, etc. and the list goes on.

Object Detection using Classical Computer Vision and Deep Learning

Earlier, detection algorithms used to be based on classical computer vision techniques such as template matching, Haar cascade, Feature detection and matching using SIFT or SURF, HOG Detector, **Deformable Part-based Model (DPM)**, etc. However, most of these methods are very specific to the use case and don't have generalization capabilities, which redirected the research toward deep learning-based methods. The Overfeat paper was the pioneer in deep learning-based object detection, it was a single network model employed to perform object classification and localization.

The Overfeat architecture closely resembles the **AlexNet** architecture. It does image classification on different scales in a sliding window fashion and carries out bounding box regression on the same CNN layer. Later other models like RCNN, FastRCNN, SPPNet, YOLO etc. emerged in this field.

One-Stage Vs. Two-Stage Detectors

Initial models were two-stage detectors, which means there was a step for region proposals, which means finding out the region of interest and a step for predicting the bounding box coordinates and class using the proposed region. This was very innovative, yet it demanded substantial computational resources and resulted in considerable delays during the inference phase. Later the YOLO authors proposed a single-stage strategy that overlooked the **Region Proposal** step and ran detection directly over the entire image. This helped speed up the training and inference, preserve global context and keep the accuracy on par.

Challenges of Object Detection

1. Occlusion and Small Objects

Small objects are always difficult to detect because models get little information about them, or the dataset might not have many instances. This issue comes under the scope of **shape invariance** problem. Additionally, occlusion and partially visible objects make it hard for the model to detect small objects.

2. Global Context and Local Contexts

Global context is as important as local context for a model. **Global context** means the usual surroundings of the object, e.g. the traffic light is mostly seen on the side of the road, or it might have cars or pedestrians nearby. The meaning of **Local context** is the object's geometrical structure, texture, and colors. For instance, traffic lights typically appear rectangular, containing lights of various colors, red, green, or yellow. For the shape invariance property, a well-trained model doesn't confuse a red light in a car's rear as a red light from a traffic light. However, if the data is not properly curated, this issue might happen.

Introduction to YOLO

Chronology of YOLO Models: Evolution and Milestones

At CVPR 2016, Joseph Redmon, along with researchers from FAIR (Facebook AI Research) and the Allen Institute for AI, published the seminal paper on **YOLO** (You Only Look Once). It was **state-of-the-art** as a single-stage object detector at that time. Using the same concept of single-stage detection along with some significant changes, people published their own models, such as SSD, RetinaNet, etc.

At the 2017 CVPR, Joseph Redmon and Ali Farhadi published the 2nd iteration of the YOLO model as **YOLOv2, built on top of YOLOv1**, integrating some advancements of that time to make it faster and more accurate. Redmon improved the architecture by adding a Neck and using a bigger backbone and published a very casually written paper in ArXiv in 2018.

Two years later, in April 2020, other authors used the YOLO name to publish version 4 of the model, and a lot of significant changes were made. Two months later, Ultralytics open-sourced the YOLOv5 model but didn't publish any paper. In the same year, YOLOv4 authors published another paper named Scaled-YOLOv4 which contained further improvements on YOLOv4. In the following year, 2021, YOLOR and YOLOX were published. Skipping version 6, in 2022, the authors of YOLOv4 published the YOLOv7, which was the state of the art at that time in terms of speed and accuracy. In the same year, researchers from Meituan Vision published the **YOLOv6, which was better than version 7**.

In January 2023, Ultralytics open-sourced the YOLOv8 with semantic segmentation capabilities alongside detection. In May 2023, Deci AI came up with YOLO-NAS, an algorithm-generated architecture that surpassed all the predecessors of YOLO.

YOLO Controversy: Ethical Considerations and the Naming Saga

In February 2020, Redmon tweeted that he stopped researching computer vision because of the concerns that his research is bringing in military applications and privacy. Growing up, he believed that science was independent of political views and the act of conducting research was inherently ethical and beneficial, regardless of the topic. His purpose was to share his point of view about one of the NeurIPS guidelines emphasizing the social benefit of research.

After that, everyone questioned whether YOLO was a good name for a series. Some also called the YOLOv4 “**The last YOLO**.”

Almost two years had passed since YOLO released its last model. It was not until April 2020 that Bochkovskiy et al. published their work as YOLOV4. He was also a DarkNet maintainer. In light of the tweet from Redmon, it appeared that the authors of YOLOv4 did not consult with Redmon or other authors before publishing their work under the name YOLO. Nevertheless, Redmon understood the hard work put into YOLOv4 and did not shy away from acknowledging that. Looking at the satisfactory improvements, the CV community also accepted the work as the official YOLOV4.

In May 2020, Glenn Jocher created a repository named **YOLOV5** under the Ultralytics Organization on GitHub. In June 2020, he pushed his first commit on that repository with the message “YOLOv5 Greetings,” the code was ported to Pytorch from the Darknet framework. YOLOv5 contained a lot of improvements from the YOLOV4 model, no paper was published, and it marked the **first instance of a YOLO model** that did not use the DarkNet framework for its development, which started a debate about whether Glenn was justified in releasing the model under the YOLO name. At that time, Roboflow also published an article comparing YOLOV5 and YOLOV4, which went trending on HackerNews and fueled the debate. Everyone started talking about this on Reddit, Twitter, and even GitHub. On June 14th, 2020, in a GitHub issue thread, Glenn Jocher stated that he would publish a summary of YOLOv5, by the end of 2020, On November 2023, they published that short synopsis.

Later, many folks started borrowing the name YOLO for their model(which was, in a way, an improvement to the original model). On the one hand, allowing researchers to use the name helped in the development of the YOLO model around the globe. Still, it also made it difficult to keep track of the progress because people continued improving on their old contributions. For example, you can’t understand YOLOv7 if you read YOLOv5 or YOLOv6, to understand that, you need to follow the authors and see what they published

before the current paper, which was YOLOv4-Scaled, YOLO-v4, and CSPNet.

YOLOv1

Paper Summary

- Generally, there are two types of Deep Learning object detection models: single-stage and two-stage models. The single-stage model follows a specific design pattern, which is the **Backbone-Neck-Head**. However, in **YOLOv1, there is no concept of a neck**, only a backbone and head. YOLOv1 architecture is inspired by GoogleNet's architecture, it has 24 convolutional layers and two fully-connected layers. In these 24 convolutional layers, the first twenty convolution layers act as a backbone, and 4 convolutional layers lead up to an additional two fully-connected layers, acting as a detection head. Instead of the inception module, they used a 1×1 convolution layer with 3×3 convolutional layers in the backbone. This helped to reduce the number of channels without having to reduce spatial dimensions, and the number of parameters became relatively low.

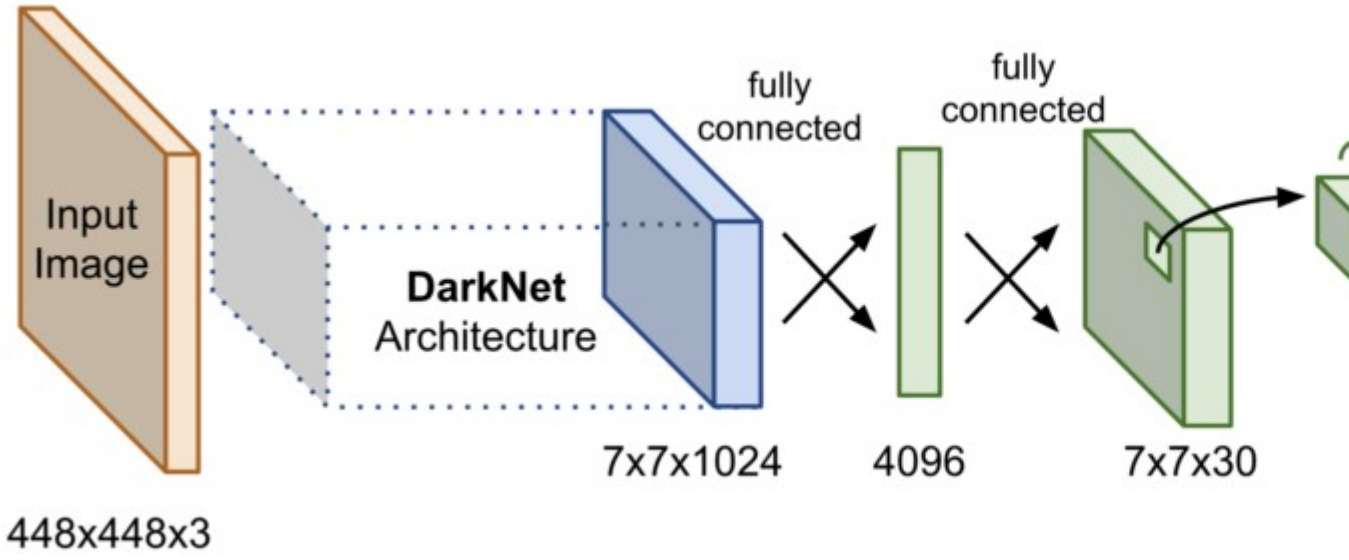


Figure 2: YOLOv1 model architecture (Source)

- The authors of YOLOV4 pre-trained the first 20 layers of YOLO in ImageNet at a resolution of 224×224, and the four remaining layers were finetuned in PASCAL VOC 2012 at a resolution of 448×448. This increased the information for the model to detect small objects more accurately. They trained their model for 135 epochs on the training and validation sets from VOC 2007 and 2012. They used a batch size of 64, momentum of 0.9, and learning rate decay of 0.0005. The learning rate(LR) scheduling is such that in the first few epochs, the LR rate rises from 10^{-3} to 10^{-2}. Till 75 epochs they train the model in 10^{-2} and again with 10^{-3} for 30 epochs, and finally 10^{-4} for 30 epochs.

- To avoid overfitting, the authors used a dropout rate of 0.5 and extensive augmentation. Random scaling and translations of up to 20% of the total dataset were used. They randomly adjusted the exposure and saturation of the image by up to a factor of 1.5 in the HSV color space. A linear activation function was set for the final layer, and the rest of the other layers used the leaky-ReLU activation.

- There is no concept of **anchor-free** boxes. The authors divided the image into an SxS grid, where **each grid predicts B bounding box** coordinates and an objectness score associated

with bounding boxes and C number of class probabilities. The prediction gets encoded in a $\mathrm{SxSx(Bx5+C)}$ tensor. It also

means that One grid cell can only predict one object. As bounding box coordinates, YOLO predicts the center (x,y) of the bounding box and the width(w) and height(h) of the box. **The center is relative to the grid cell**, so it's between 0 and 1, and width and height are relative to the image size, which also comes in the range of 0 and 1. Objectness score refers to the score that tells if a bounding box contains an object or not, which is denoted as $\mathrm{p}(Object) * \mathrm{IOU}_{\mathrm{pred}}^{\mathrm{truth}}$, where $\mathrm{p}(\mathrm{Object})$ is the probability to have an object in the cell and $\mathrm{IOU}_{\mathrm{pred}}^{\mathrm{truth}}$ is the intersection over union between predicted region and the grand truth.

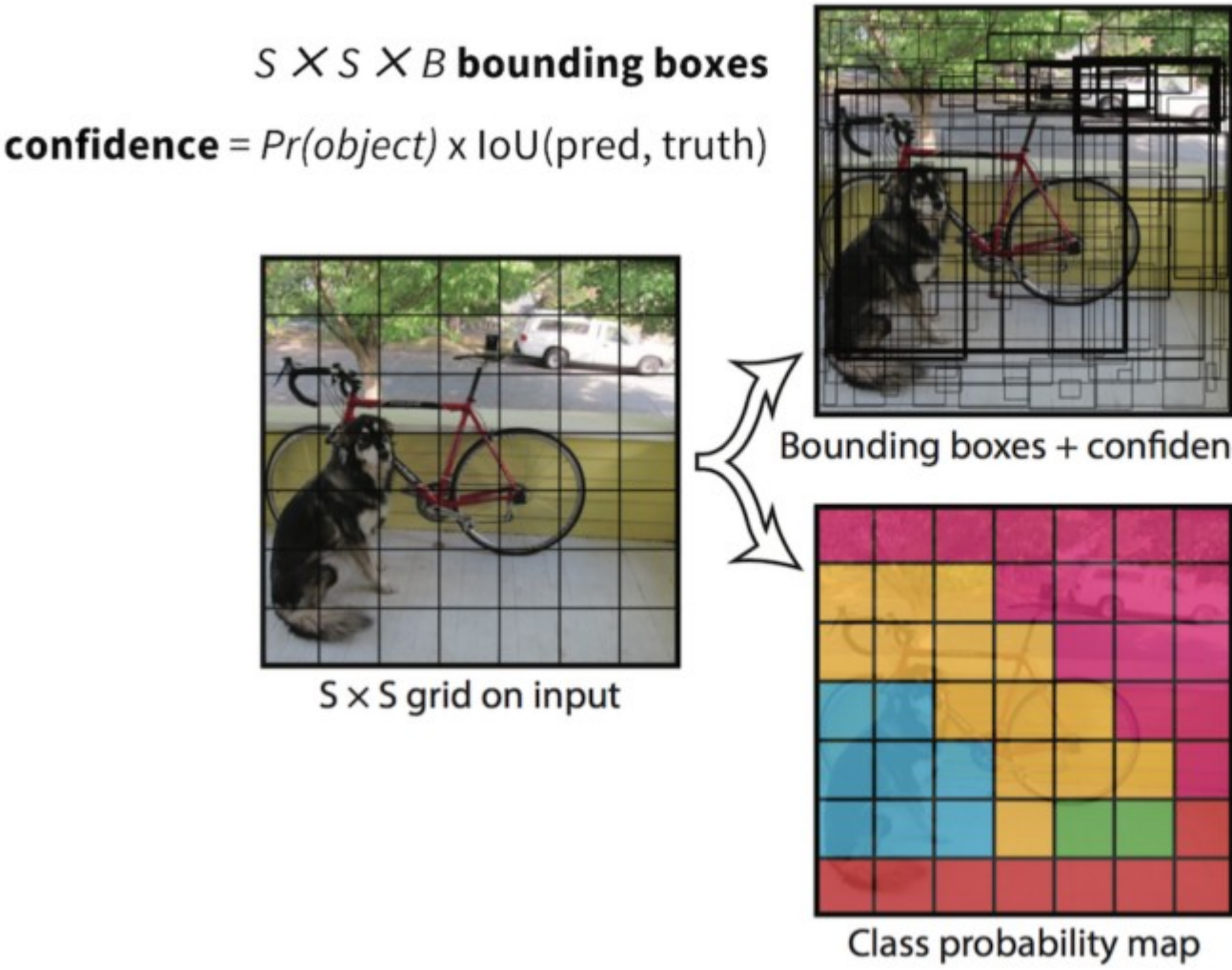


Figure 3: YOLO Bounding anchor-free bounding box prediction (Source)

- As the detection head needs to predict the bounding box coordinates, objectness score, and object class, they have three parts to the loss function: **localization loss**, **confidence loss**, and **classification loss**. As the object detection was depicted as a regression problem, all losses are sum-squared errors. The first two loss terms belong to localization loss, the next two losses belong to the confidence loss, and the last one belongs to the classification loss. Often there are grids that don't predict any bounding box, which pushes the confidence score of those cells toward zero, overpowering the cells that contain an object.

Two parameters $\lambda_{coord} = 5$ and $\lambda_{noobj} = 0.5$ were proposed to resolve this issue.

$$
\begin{aligned}
&\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
&+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
&+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\
&+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2 \qquad (3)
\end{aligned}
$$

Figure 4: YOLOv1 loss function (Source)

YOLOv2

Paper Summary

YOLOv2 was published at CVPR, in 2017, by the same authors as the YOLOv1. They slightly modified the YOLOv1 architecture and improved the training process to make YOLOv2 faster and more accurate. Here are the changes that enhanced YOLOv2.

Using High-Resolution Images:

They trained their classification network at 448×448 in ImageNet for 10 epochs. This helps the network to adjust the filters to better work with high-resolution images. Later the resulting network is trained on

detection tasks, and by doing that, they are promoting joint training that helps to train object detectors in both object detection and classification data. Their objective mainly was improving recall and localization while maintaining classification accuracy. They use 416×416 instead of 448×448 so that feature maps are sized in odd numbers, and the object center should lie at one point instead of multiple locations. YOLO's convolution layers **downsample the image by a factor of 32**, so by using an input image of 416, we get an output feature map of 13 × 13.

Introduction to Anchor Boxes:

Generally, the **box dimensions are hand-picked**, which is not always a good prior. To solve this, the authors proposed the method of using **k-means clustering** on all the bounding boxes of the dataset. This gives them the most dominant sizes of the bounding boxes from the dataset. But, in k-means, using Euclidean distance will produce a higher error(distance) for large boxes and a smaller error(distance) for small boxes. But, as the IOU score is independent of the size of the box, the distance measurements needed to be changed. They proposed,

$$d(\text{box}, \text{centroid}) = 1 - \text{IOU}(\text{box}, \text{centroid})$$

Figure 5:

YOLOv2 K-Means Clustering Distance Metric for bounding box prior (Source)

This part of the equation implies that the distance is inversely proportional to the IOU. Subtracting the IOU from 1 essentially means that the distance decreases as the IOU increases (indicates higher overlap or similarity), and vice versa. We run k-means for various values of k and plot the average IOU with the closest centroid, see Figure 6. We choose k = 5 as a good tradeoff between model complexity and high recall.

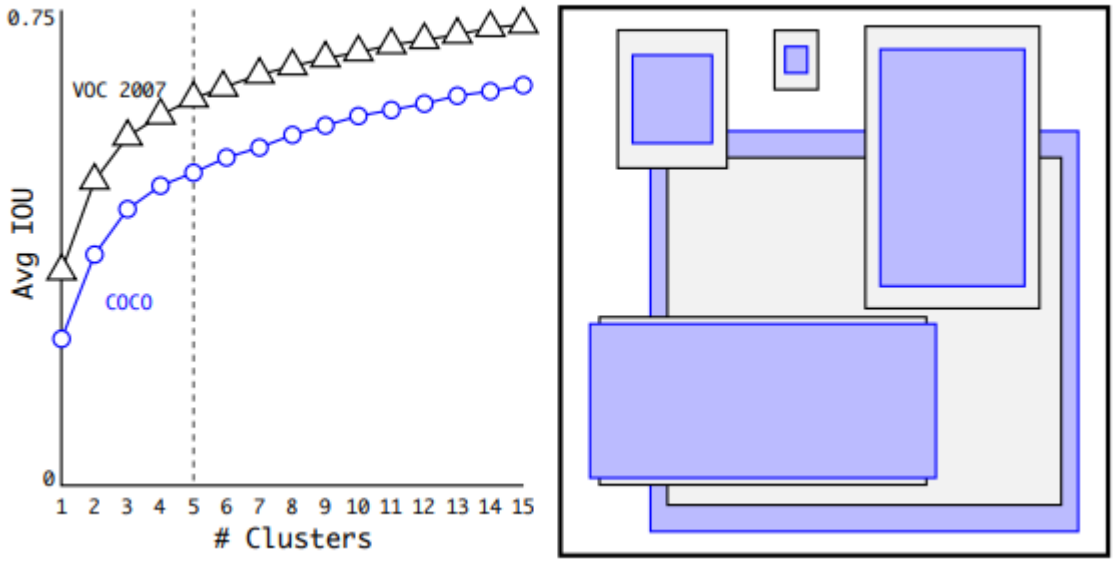


Figure 6: Clustering of box dimensions on VOC and COCO (Source)

Fine-grained features:

Like YOLOV1, YOLOV2 also predicts the bounding box coordinates relative to the grid cell. The modified YOLO predicts a 13×13 feature map, while this helps in detecting large objects, having a fine-grained feature map might help in detecting small objects. Many detection models have different approaches, but in YOLOv2, the authors proposed a **passthrough layer** that concatenates features from a higher resolution layer to a lower resolution layer. When concatenating a higher-resolution image with a lower-resolution image, it internally restructures the higher-resolution image such that the special dimension reduces, but the depth increases. This means that if given a 26x26x512 feature map, it gets resized as 13x13x2048, here, the channel dimension 2048 comes from 512 x 2 x 2 = 2048.

Location Prediction:

The diagram below is explained as,

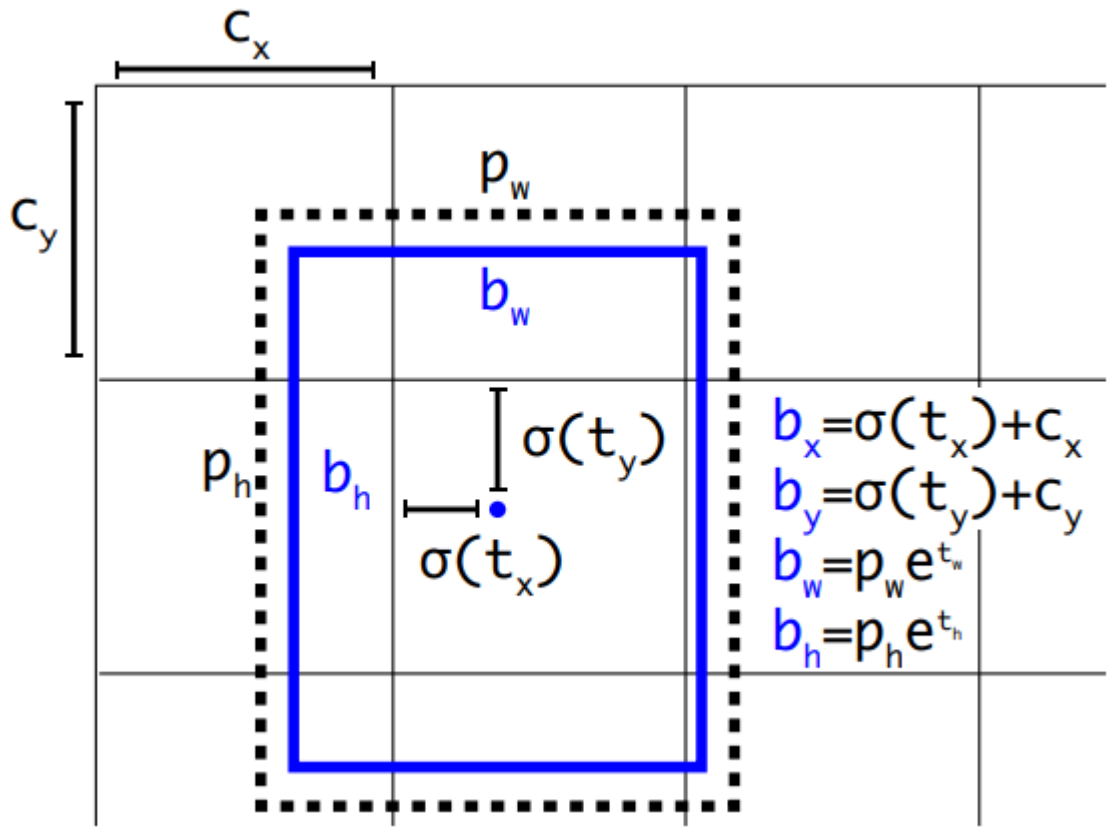


Figure 7: bound box(relative to grid cell) parameter equations (Source)

Here, c_x, c_y is the grid cell center. b_x, b_y are the coordinates of the center of the predicted bound box relative to the grid cell. b_w and b_h are the width and height of the predicted bounding box, which is an offset of the prior anchor box associated with the grid. σ ensures the center point (b_x, by) stays within the grid cell by restricting the values between 0 and 1. e^{t_w} and e^{t_h} represent the exponential transformation of predicted width and height offsets t_w and t_h, ensuring that the width and height stay positive, although using an exponential transform is mathematically unstable.

Multiscale training:

They introduced multiscale training to make the model more robust. After every 10 batches, the model takes a randomly selected image

dimension and continues training. Since their model downsamples by a factor of 32, they pull from the following multiples of 32: {320; 352; …; 608}.

Hierarchical classification:

The model is first trained for classification on ImageNet before even training detection. For that, they hierarchically prepare their data. Hierarchical Classification improves the model accuracy by leveraging the structure of **class label's hierarchically**, the model can better understand the relationships between different classes.

YOLOv3

Paper Summary

- In YOLOv2, the model predicts the object class for each grid. But, in YOLOv3, the model predicts class for each bounding box predicted. The model **predicts 3 bounding boxes for each grid**, objectness score, and class predictions. At the output side the tensor is of $\mathrm{N} \times N \times (3 \times (4+1+80))$. The model outputs bounding boxes at 3 different scales.

- YOLOv3 uses multiple independent logistic classifiers rather than one softmax layer for each class. During training, they use binary cross-entropy loss in a **one vs. all setup**.

- YOLOv3 uses the **DarkNet-53** as a backbone for feature extraction. The architecture has **alternative 1×1 and 3×3 convolution** layers and skip/residual connections inspired by the ResNet model. They also added the idea of **FPN** to leverage the benefit from all the prior computations and fine-grained features early on in the network. Although DarkNet53 is smaller than ResNet101 or RestNet-152, it is faster with equivalent or better performance.

- Similar to YOLOv2, YOLOv3 also uses k-means to find the bound box before the anchors. In this model, they used 3 prior boxes for different scales, unlike YOLOv2.

Training YOLOv3

Ultralytics has a YOLOv3 repository that is implemented in pytorch. They provide a command line interface to train a model swiftly. The official repository for yolov3 is in darknet framework. YOLOv3 custom training is a good resource to understand how scratch training works. We also show the training of YOLOv3 using Opencv python and c++ on the coco dataset.

YOLOv4

Paper Summary

- Alexey Bochkovskiy collaborated with the authors of CSPNet(Nov 2019) Chien-Yao Wang and Hong-Yuan Mark Liao, to develop YOLOv4. The only similarity between YOLOv4 and its predecessors was that it was built using the Darknet framework. They experimented with many new ideas and later published them in a separate paper, Scaled-YOLOv4.

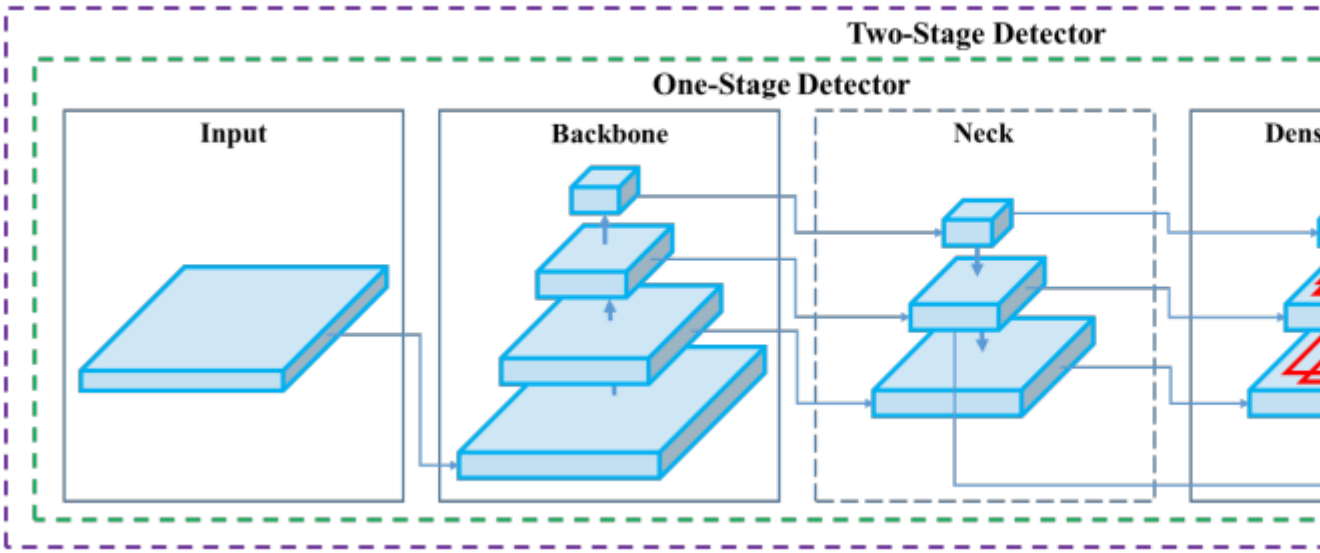


Figure 8: One-stage VS. Two-stage detector architecture comparison (Source)

- Generally, an object detector comprises an ImageNet pre-trained backbone for feature extraction and a Head for doing the bounding box regression and classification. But, from YOLOv3 and YOLOv4, the authors introduced another part of the structure called the **Neck**, placed between the backbone and the head. The use of this block is to collect features of different stages from the backbone and pass them to the head. In YOLOv4, we had **CSPDarkNet53** as the backbone, Spatial Pyramid Pooling (SPP), and Modified PANet as the Neck and the YOLOV3 head. In the paper, they introduced two concepts, **bag of freebies** and **bag of specials**. They imposed methods that only changed the training strategy or increased the training cost without increasing the inference cost as Bag of Freebies. One of the common examples of this would be data augmentation. On the other hand, methods that slightly increase the inference cost but significantly improve the accuracy are called bag of specials. Methods that belong to this category are enlarging the receptive field, combining features, post-processing, etc. **SPP and PANet** were discussed in detail in the YOLOv5 section.

- **Cross Stage Partial Network** is designed for faster inference while keeping the accuracy of the original model. It acts as an additional component that can be integrated into your existing setup. The concept behind CSPNet is that architectures like ResNet and DensNet have many skip connections. A large amount of gradient information is reused during the gradient updation of different layers, which results in different layers repeatedly learning copied gradient information. To stop this duplicate gradient, CSPNet divides the input into two parts, one passes through the usual in-between layer, and the 2nd part gets concatenated after that block for further operations.

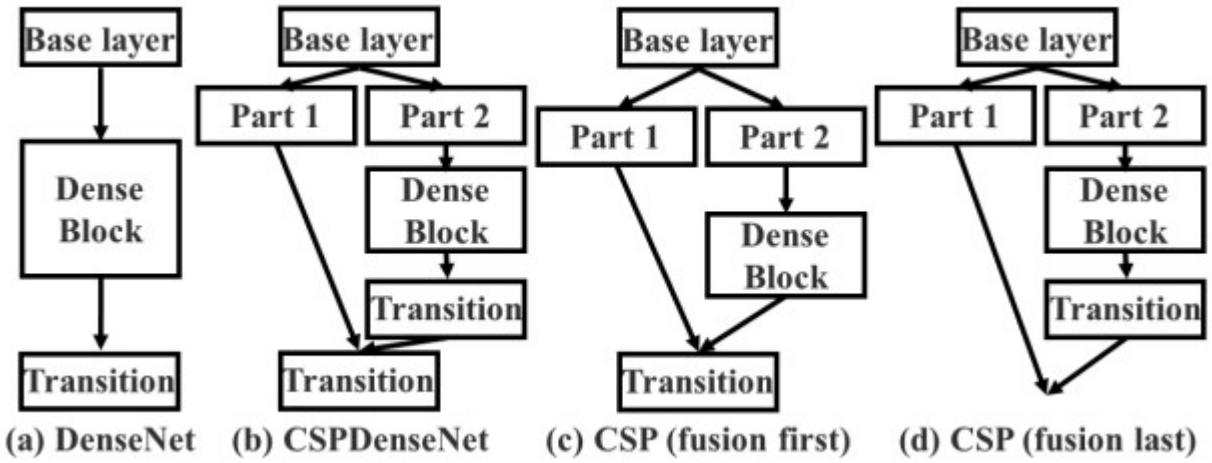


Figure 9: Different kinds of feature fusion strategies in CSPNet (Source)

- The authors experimented with many architectures such as ResNet50, ResNext50, VGG16, DarkNet53, etc. but later, a variant of DarkNet53 with the CSPNet was used for the backbone. They found numerous studies demonstrating that the CSPResNext50 was better than CSPDarkNet53 for object classification. But, through experimentation, they found out that the CSPDarkNet53 is more suited for Object Detection tasks.

- They introduced **Spatial Pyramid Pooling(SPP)** as in YOLOV3-SPP for the multi-scale feature pooling. They use PANet for the feature pyramid instead of FPN from the YOLOV3 model. In the literature, two types of attention modules are

mentioned that are being used in an object detector, channel-wise attention and point-wise attention. **Spatial Attention Module (SAM)** and Squeeze-and-Extraction are examples of that, respectively. SAM is used in YOLOv4 because it improves accuracy and decreases the inference latency. Finally, for the detection head, they use anchors as in YOLOv3. Above mentioned integrations are a part of the bag of specials, where CSPNet cuts the computations cost while maintaining the accuracy, SPP block helps to increase the receptive field, and PANet helps for better feature aggregation.

- As a **bag of freebies** method, the authors introduced the mosaic data augmentation, which combines 4 images in one image. By doing that, they add additional contextual information and decrease the mini-batch size for batch normalization. Except for **mosaic**, they also used **MixUp and CutMix** alongside other geometrical augmentations. They used cIOU loss and cross mini-batch normalization for better detection, replacing normal Dropout with the DropBlock.

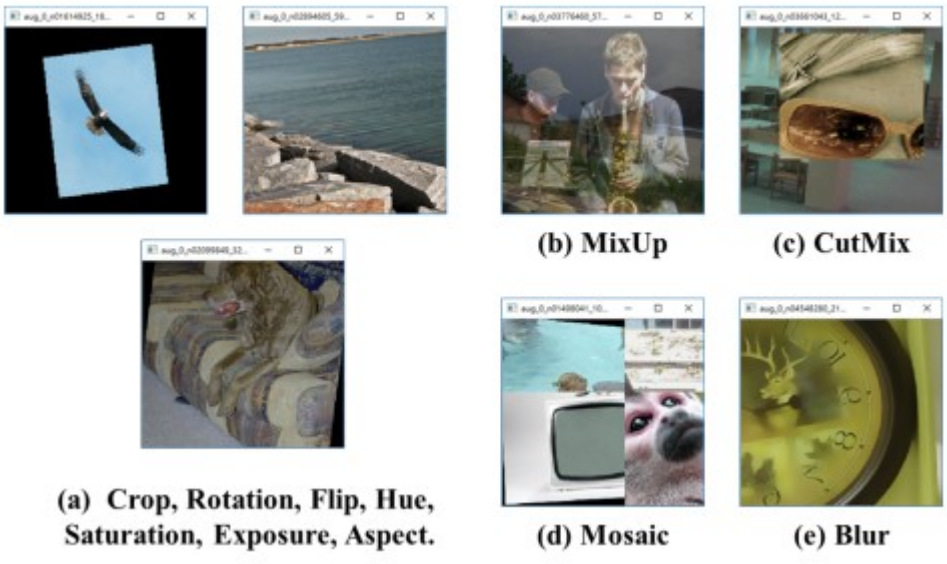


Figure 10: Different Augmentation techniques used in YOLOv4 (Source)

Training YOLOv4

We discuss the custom training of YOLOv4 on pothole detection using YOLOv4 and the darknet framework. The dataset contained 1265 training images, 401 validation images, and 118 validation images.

YOLOv5

Paper Summary

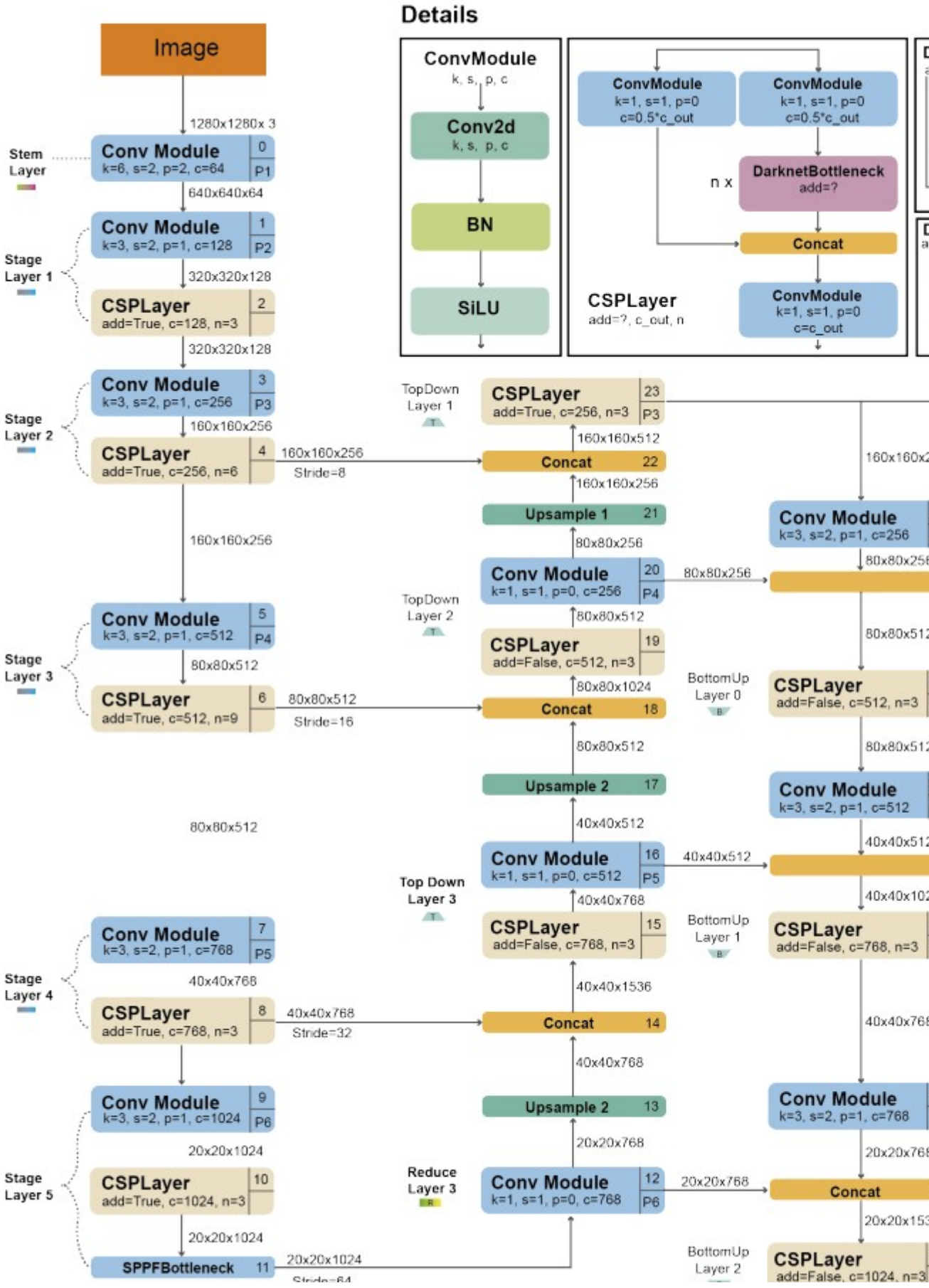

Details
Image
1280x1280x 3
Stem Layer
Conv Module
k=6, s=2, p=2, c=64
0
P1
640x640x64
Stage Layer 1
Conv Module
k=3, s=2, p=1, c=128
1
P2
320x320x128
CSPLayer
add=True, c=128, n=3
2
320x320x128
Stage Layer 2
Conv Module
k=3, s=2, p=1, c=256
3
P3
160x160x256
CSPLayer
add=True, c=256, n=6
4
160x160x256
Stride=8
160x160x256
Stage Layer 3
Conv Module
k=3, s=2, p=1, c=512
5
P4
80x80x512
CSPLayer
add=True, c=512, n=9
6
80x80x512
Stride=16
80x80x512
Stage Layer 4
Conv Module
k=3, s=2, p=1, c=768
7
P5
40x40x768
CSPLayer
add=True, c=768, n=3
8
40x40x768
Stride=32
Conv Module
k=3, s=2, p=1, c=1024
9
P6
20x20x1024
Stage Layer 5
CSPLayer
add=True, c=1024, n=3
10
20x20x1024
SPPFBottleneck
11
20x20x1024
Reduce Layer 3
Conv Module
k=1, s=1, p=0, c=768
12
P6
20x20x768
Upsample 2
13
40x40x768
Concat
14
40x40x1536
CSPLayer
add=False, c=768, n=3
15
40x40x768
Top Down Layer 3
Conv Module
k=1, s=1, p=0, c=512
16
P5
40x40x512
Upsample 2
17
80x80x512
Concat
18
80x80x1024
CSPLayer
add=False, c=512, n=3
19
80x80x512
TopDown Layer 2
Conv Module
k=1, s=1, p=0, c=256
20
P4
80x80x256
Upsample 1
21
160x160x256
Concat
22
160x160x512
CSPLayer
add=True, c=256, n=3
23
P3
TopDown Layer 1
ConvModule
k, s, p, c
Conv2d
k, s, p, c
BN
SiLU
ConvModule
k=1, s=1, p=0
c=0.5*c_out
ConvModule
k=1, s=1, p=0
c=0.5*c_out
n x
DarknetBottleneck
add=?
Concat
CSPLayer
add=?, c_out, n
ConvModule
k=1, s=1, p=0
c=c_out
BottomUp Layer 0
BottomUp Layer 1
BottomUp Layer 2
Conv Module
k=3, s=2, p=1, c=256
CSPLayer
add=False, c=512, n=3
Conv Module
k=3, s=2, p=1, c=512
CSPLayer
add=False, c=768, n=3
Conv Module
k=3, s=2, p=1, c=768
Concat
CSPLayer
add=False, c=1024, n=3

Figure 11: YOLOv5 Model Architecture (Source)

Model Architecture:

The YOLOv5 model is divided into 3 parts, Backbone, Neck, and Head. Each convolution is followed by batch normalization (BN) and SiLU activation. Below is the 3000-foot overview of the YOLOv5 Model Architecture

- Backbone: **CSP-DarkNet53**
- Neck: **SPPF and CSP-PANet**
- Head: **YOLOv3 Head**

Spatial Pyramid Pooling Fast (SPPF):

SPPF is one of the improvements that they introduced in their model. This is nothing but SPP, it's just that instead of passing the input to three different max-pool layers, the output of one max-pool block is being fed to the subsequent max-pool layer, which makes the process much faster. Below is the Pytorch implementation of SPP and SPPF:

Figure 12: SPP and SPPF Pytorch implementation (Source)

PANet (Path Aggregation Network):

PANet was used in YOLOv4. The aim of using PANet was to create a rich multi-stage feature hierarchy for robust object detection. The only modification is that PANet concatenates the feature maps instead of adding them, as mentioned in the original paper. CSP-PANet is the same as PANet. Instead, they add a few **CSP layers in between the PANet**. This change results in a processing speed that is more than twice as fast.

AutoAnchors:

YOLOv5 also incorporated another improvement which is called **AutoAnchor**. In AutoAnchor, they change the shape of the anchors during training. First, the model starts with the prior anchor boxes generated from running k-means on the ground truth bounding boxes. Later these bounding boxes get updated using **genetic evolution (GE) algorithm**. GE algorithm develops these anchors across 1,000 generations, employing CIoU loss and the Best Possible Recall for its fitness evaluation.

Bag of freebies:

Apart from this bag of special methods, they also used a bag of freebies methods and data augmentations such as **Mosaic, copy-paste, random affine, MixUp, HSV** augmentation, etc. I remember from the TensorFlow – Help Protect the Great Barrier Reef Kaggle competition, where people suddenly started training YOLOv5 on high-resolution images, which resulted in higher Leader Board scores. However, it was later tested that using a larger input size of 1536 pixels and test-time augmentation (TTA), YOLOv5 achieves an AP of 55.8%.

Training YOLOv5

YOLOv5 is one of the best working YOLO versions, and one should know how to train that on a custom dataset. Here we showed how to train YOLOv5 on the vehicle OpenImage dataset. As Glenn Jocher ported the model from the darknet framework to Pytorch, it became much easier to train. YOLOv5 – Custom Object Detection Training will help you learn about the custom training in depth. As the models will mostly be deployed in edge devices, it is important to understand how to run YOLOv5 using OpenCV in Python and C++. As a new addition, Ultralytics has enabled the feature of the training instance segmentation model using the YOLOv5 repository. Check out: YOLOv5 Instance Segmentation to understand in depth.

YOLO-R

Paper Summary

- YOLOv4 arrived in April 2020, and after that few other researchers started using the YOLO name and publishing their models. In November 2020, the same authors from the YOLOv4 published the continuation of their previous work as Scaled-YOLOv4. In mid-2021 few authors from the YOLOv4 team published YOLO-R. With this paper, the authors started exploring along the lines of multi-task learning.

- The authors found that the feature extracted for a specific task is not generalized enough to be applied to other tasks. This means one can't directly use the features from a detection model in a segmentation model. To solve this, everything boils down to producing a generalized representation. And this is where **Multi-task Learning(MTL)** comes into the picture. In MLT the model is trained to perform multiple different tasks simultaneously, rather than training task-specific models. By training a model for multiple tasks parallelly, one can improve the model's generalization capabilities. This is because the model leverages the commonalities and differences across the tasks. MLT is also

helpful because it reduces the number of model parameters, resulting in fewer FLOPs and a decreased model latency.

- The concept of **YOLOR** is not like usual MLT models, where you see shared representation components, task task-specific components, rather, the concept of YOLOR comes from implicit knowledge and explicit knowledge. Knowledge is two types: **explicit knowledge and implicit knowledge**. Humans learn by looking, hearing, tactile, etc., which are a type of explicit knowledge, and they also learn from past experiences, which is categorized as implicit knowledge. Implicit knowledge refers to the knowledge learned in a **subconscious state**.

- In terms of neural networks, there will be a network for learning explicit knowledge, and there will be another model jointly trained with the explicit model for learning implicit knowledge. In this diagram, the authors presented the same concept,

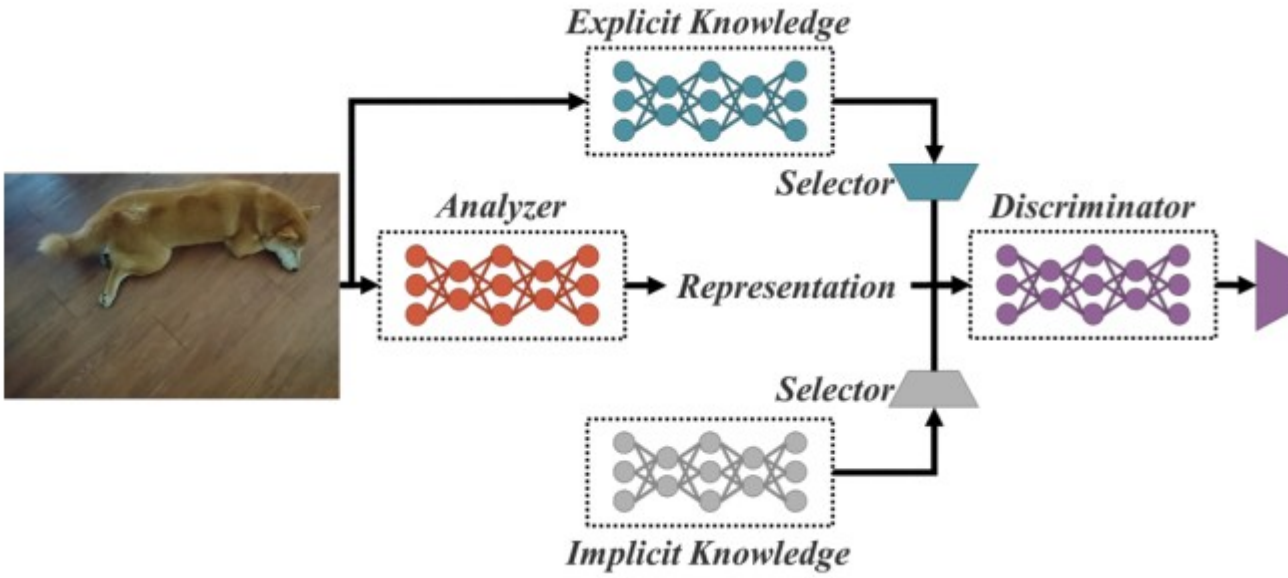


Figure 13: YOLOR workflow diagram (Source)

- For the Explicit knowledge network, the authors used YOLOv4-Scaled with a slight modification of adding a **reOrg Layer** followed by two convolution layers. In the Implicit network, they used a single convolutional layer with two **learnable priors** for kernel space alignment and prediction

refinement. They included object detection, multi-label image classification, and feature embedding for multi-task learning.

- For further understanding YOLO-R comprehensive paper review is a very good resource that goes over Implicit and explicit knowledge in depth, how humans understand and learn, how to model implicit knowledge, how to combine that with explicit knowledge to improve model capabilities, and so on.

YOLOX

Paper Summary

- In July 2021, **YOLOX** was published by Megvii Technology, China. This model was the first-place winner of the Streaming Perception Challenge of the WAD workshop at CVPR 2021. YOLOR was published on the same timeline as YOLO-X. When the authors of YOLOR were trying to explore multi-task learning, the authors of YOLOX focused on integrating the recent advances into the YOLO model. All YOLO models before YOLOX were using Anchor boxes. But, during that period, most state-of-the-art models used Anchor-free detectors. Hence, the authors incorporated this approach into their model, along with the advanced label assignment strategy simOTA, Decoupled Head, and Strong Augmentations.

- The YOLOX model features the DarkNet53 from YOLOv3 with **spatial pyramid pooling(SPP)** as the backbone, FPN as the Neck, and a customized decoupled head. The YOLOX model was trained from scratch, the authors excluded ImageNet Pre-training because using that with strong augmentations was no longer useful. At that time, YOLOv3 was still one of the best detectors to which a large part of the industry could turn because of limited computation. YOLOv5, on the other hand, was optimized, keeping the anchor-based method in mind. Although the Backbone and Neck of the YOLO detectors improved over time, the head part was untouched, and

decoupling the head also resolved the long ongoing conflict of classification, and regression tasks. This helped them in faster convergence.

- It starts with one 1×1 conv layer, two 3×3 conv layers again, and a 1×1 conv layer (for each branch). They had a very good training regimen, 300 epochs of training with SGD, cosine scheduler, multi-scale training, **EMA weight updation**, BCE loss for the classification branch, and IoU loss for the regression branch.

- The authors used heavy augmentations to improve the model performance, they explicitly added the famous **Mosaic and MixUp** in their augmentation pipeline. They found out that for large models, strong augmentation was useful. They used MixUp with scale jittering in their pipeline after exterminating with MixUp and Copypaster.

- Taking inspiration from end-to-end (**NMS-free**) detectors, the authors added two additional convolution layers, one-to-one label assignment, and stop gradient. Still, unfortunately, that slightly decreased the performance and the inference speed, so they dropped it.

Training YOLO-X

YOLOX is a meticulously engineered model specifically crafted for object detection. We have shown the YOLOX custom training on the Drone Detection dataset, which consists of 4014 labeled images. We used the official YOLOX repository for training. We trained the model in different configurations and shared the results there.

YOLOv7

Paper Summary

The YOLOv7 is the continuation after Scaled-YOLOv4. YOLOv4 and YOLOv7 are published by the same authors. The YOLOv7 model introduces several novel modifications such as E-ELAN, Model Scaling, Planned re-parameterized convolution, Coarse for auxiliary, and penalty for lead loss.

Extended-ELAN (E-ELAN):

E-ELAN is an Extended efficient layer aggregation network, which is a **variant of ELAN**. ELAN is inspired by VoVNet and CSPNet. We already know about CSPNet, and VoVNet is nothing but an Object Detection Network composed of cascaded OSA modules. **OSA means One-shot Aggregation**. It is more efficient than Denses Block in DenseNet and very optimized for GPU computation. The below image clears the concept of OSA,

Figure 14: E-ELAN Aggregation Methods (Source)

It is very similar to DensesNet, but we concatenate the features after a few conv blocks here.

Compound Model Scaling:

The authors used model scaling to adjust model parameters to generate models of different scales. By different scales, it means a change in model parameters. It helps to meet the need for different inference speeds. In EfficientDet, scaling is done in width, depth, and resolution, and Scaled-YOLOv4 scales the model in stages (No. of feature pyramid). In this model, they used **NAS (Network Architecture Search)**, which is a commonly used model scaling method, authors also showed that this method can be further improved using the compound model scaling approach.

Model Re-parameterization:

Model re-parameterization means model weight averaging. There are two ways of doing model averaging,

1. Averaging the weights of the same model trained on different folds of data.
2. Averaging the model weights of different epochs.

This is a very popular ensemble technique. Pytorch has an implementation of the same named SWA(Stochastic Weight Averaging).

Coarse for Auxiliary and Fine for Lead Loss:

As you can see, the model architecture has multiple heads predicting the same thing. The head responsible for the final output is the lead head, and the other heads assist in the model training. With the help of an assistant loss, the weights of the auxiliary heads are updated.

For further understanding of the YOLOv7 paper and its inference results, consider exploring YOLOv7 Paper Explanation. This resource provides a detailed analysis and insights into the topic.

Training YOLOv7

We have shown YOLOv7 fine-tuning on the pothole dataset, using the official YOLOv7 repository. Both fixed-resolution training as well as **Multi-resolution training** were performed along with a comprehensive study of the model inference. We have also done a comprehensive study of YOLOv7 and media pipe human pose estimation, which **compares GPU and CPU inference** performances.

YOLOv6

Paper Summary

- 2 months after the YOLOv7 release, researchers from Meituan Inc, China, released YOLOv6. Yes, you heard it right, **YOLOv7 was published before YOLOv6**. And apparently the authors took permission from the original YOLO authors. YOLOv6 was the state-of-the-art(SOTA) when it was published.
- In this paper, the authors experimented with many techniques, and the resulting successful methods were integrated with the model and published as YOLOv6. YOLOv6 is an anchor-free, decoupled head architecture that has a unique backbone named **EfficientRep**.

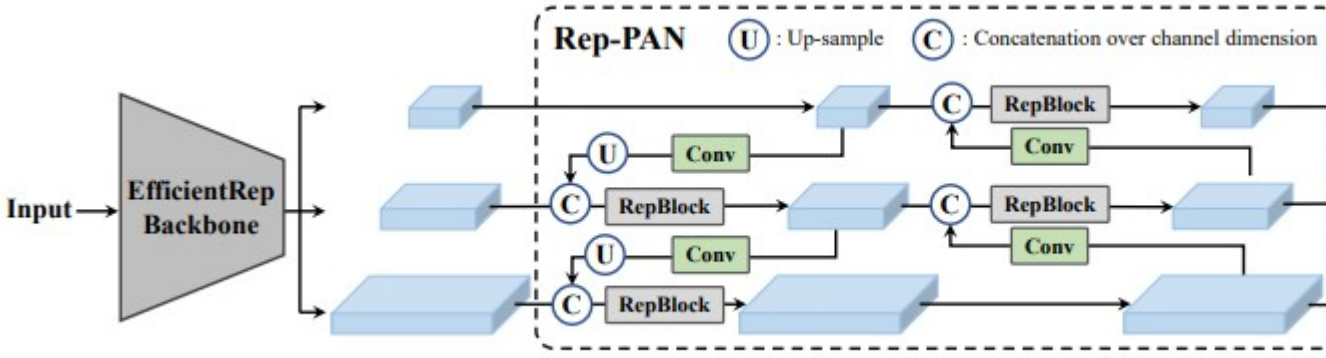


Figure 15: YOLOv6 Model Architecture (Source: [5])

- EfficientRep consists of a **RepVGGBlock stem**, followed by 4 blocks of ERBlock. ERBlock is made of a RepVGGBlock and a RepBlock. The last layer of ERBlock has a RepVGGBlock and RepBlock, with the exception of **SimSPPF**, which is nothing by the **SPPF layer with ReLU** activation. For Neck, they used Rep-PAN, which is nothing but the good old PANet from YOLOv4 and YOLOv5, it is just that we replace the CSPBlock with RepBlock (for small models) or **CSPStackRep Block** (for large models). For Head, they used a customized version of the decoupled head from YOLOX, which they named the **hybrid-channel strategy**. In this method they reduced one 3×3 conv layer and jointly scaled the width of the head by the width multiplier for the backbone and the neck.

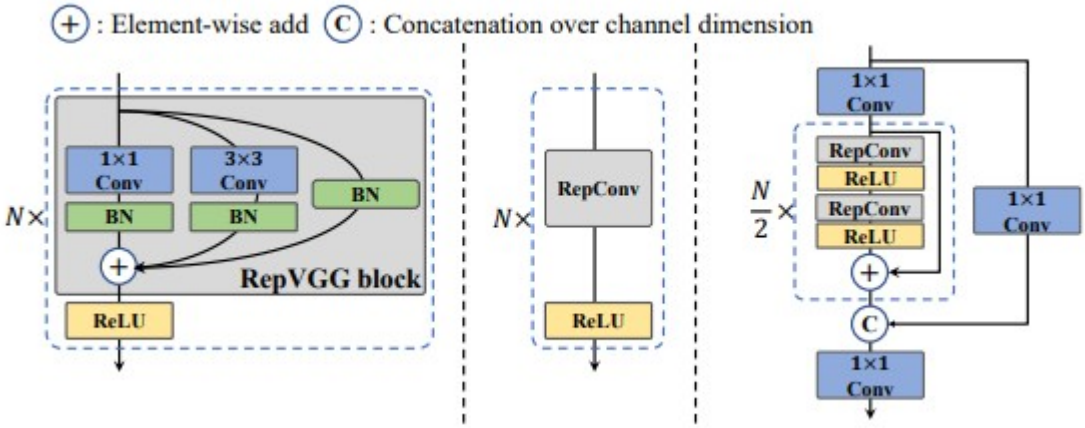


Figure 16: RepBlock and RepVGG model architecture (Source: [5])

- Except for these architectural changes, they used advanced **Lable Assignment techniques** such as **simOTA**, and **TAL**(Task alignment learning), a lot of new loss functions were introduced, and to make the model faster and more accurate, they used a lot of post-training quantization methods, such as **self-distillation**, **Reparameterizing** Optimizer, etc.

- YOLOv6 is a very important addition to the YOLO series of models, and it's very important to understand what improvements were made to make the model more efficient. YOLOv6 – Paper Explanation is a great resource to understand YOLOv6 thoroughly; it meticulously clarifies complex concepts such as RepConv, RepVGGBlock, CSPStackRep, VFL, DFL, and Self-Distillation very carefully.

Training YOLOv6

- Although it's named YOLOv6, it was the best-performing model when it was published. And it is very crucial to understand how to **train a YOLOv6 on a custom dataset**, for that, YOLOv6 custom dataset training for Underwater Trash Detection provides an excellent guide for such custom training applications.

YOLOv8

Paper Summary

- In January 2023, Ultralytics came up with YOLOv8, which was the SOTA at that time. The model was open-sourced, but the maintainers didn't publish any paper. Looking at the architecture of the YOLOv8 model, the previous model seemed over-engineered.

YOLOv8

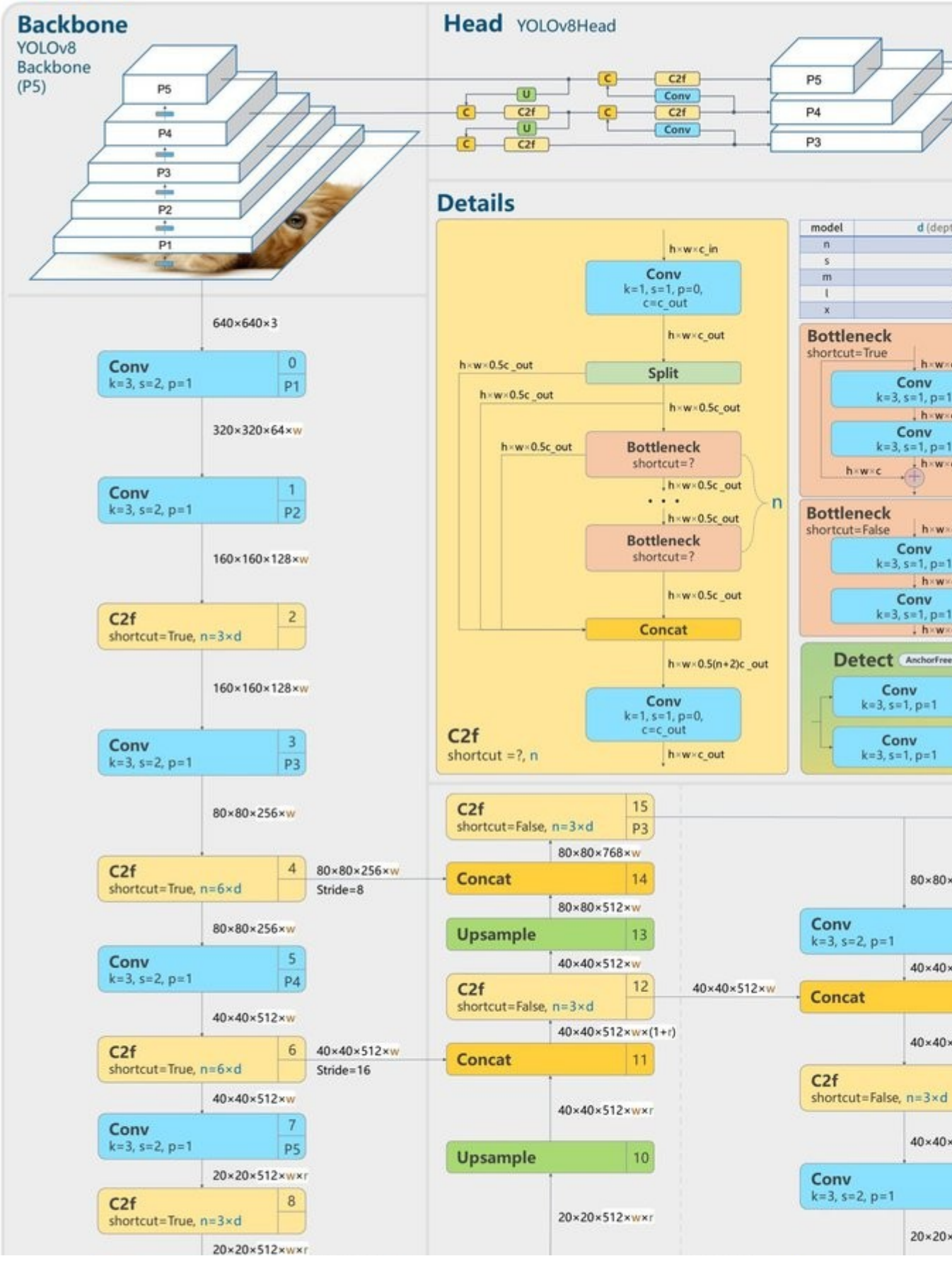

Backbone
YOLOv8
Backbone
(P5)
P5
P4
P3
P2
P1
640×640×3
Conv
k=3, s=2, p=1
0
P1
320×320×64×w
Conv
k=3, s=2, p=1
1
P2
160×160×128×w
C2f
shortcut=True, n=3×d
2
160×160×128×w
Conv
k=3, s=2, p=1
3
P3
80×80×256×w
C2f
shortcut=True, n=6×d
4
80×80×256×w
Stride=8
80×80×256×w
Conv
k=3, s=2, p=1
5
P4
40×40×512×w
C2f
shortcut=True, n=6×d
6
40×40×512×w
Stride=16
40×40×512×w
Conv
k=3, s=2, p=1
7
P5
20×20×512×w×r
C2f
shortcut=True, n=3×d
8
20×20×512×w×r
Head
YOLOv8Head
U
C
C2f
Conv
P5
P4
P3
Details
h×w×c_in
Conv
k=1, s=1, p=0,
c=c_out
h×w×c_out
Split
h×w×0.5c_out
Bottleneck
shortcut=?
n
Concat
h×w×0.5(n+2)c_out
C2f
shortcut =?, n
model
d (dept
n
s
m
l
x
Bottleneck
shortcut=True
Bottleneck
shortcut=False
Detect
AnchorFree
C2f
shortcut=False, n=3×d
15
P3
80×80×768×w
Concat
14
80×80×512×w
Upsample
13
40×40×512×w
C2f
shortcut=False, n=3×d
12
40×40×512×w×(1+r)
Concat
11
40×40×512×w×r
Upsample
10
20×20×512×w×r
40×40×512×w
Conv
k=3, s=2, p=1
Concat
C2f
shortcut=False, n=3×d
Conv
k=3, s=2, p=1

Figure 17: YOLOv8 architecture (Source)

- The YOLOv8 architecture follows the same architecture as YOLOv5, with a few slight adjustments, such as, they use **c2f module** instead of CSPNet module, which is just a variant of CSPNet, (CSPNet followed by two convolutional networks). They designed YOLOv8 **anchor-free** using **YOLOv5 as a based model**, which YOLOX authors tried but ended up using YOLOv3 as they found YOLOv5 over-optimized for anchor-based training. They also introduced **Decoupled heads** to independently process objectness, classification and bounding box prediction tasks. They used the sigmoid layer as the last layer for objectness score prediction and softmax for classification.

- YOLOv8 uses **CIoU and DFL loss functions** for bounding box loss and binary cross-entropy for classification loss. These losses have improved object detection performance, particularly when dealing with smaller objects. They also introduced the **YOLOv8-Seg model** for semantic segmentation by applying minimal changes to the original model.

- As there is no paper associated with YOLOv8, it becomes very hard to understand how the model works by looking at the code or any GitHub discussions. YOLOv8 : Comprehensive Guide is one of the best places to begin learning about YOLOV8.

Training YOLOv8

We have shown the YOLOv8 model custom training using Ultralytics repository, where we used the pothole dataset collected from Roboflow, Reasearch-Gate, manually annotated YouTube videos, and RDD2022 dataset. There is also a guide for training YOLOv8 using KerasCV. There were a total of 6962 images for training and 271 images for validation. The article also includes a comparison between different variants of YOLOv8 and inference results. We also have an in-depth article on comparing YOLOv8 models of different scales on the Global Wheat Data 2020 dataset. YOLOv8 has the

ability to train a **semantic segmentation** model having minimal changes in the original YOLOv8 detection architecture, which has been discussed here, YOLOv8 Instance Segmentation Training on Custom Data. **Pose Estimation** is a very crucial problem statement in Computer Vision; one can go through YOLOv8 Animal Pose Estimation to understand how to fine-tune YOLOv8 for pose estimation.

YOLO-NAS

Paper Summary

In May 2023, an Israel-based company Deci, published their latest YOLO variant called YOLO-NAS, the current state-of-the-art object detection model. YOLO-NAS is tailored for efficiently detecting small objects and boasts enhanced localization precision. It also improves the performance-to-compute ratio, making it ideal for real-time edge-device applications.

They introduced a few quantization-aware modules, such as QSP and QCI. The architecture was generated through AutoNAC, which is Deci's proprietary NAS technology. NAS means **Neural Architecture Search**. YOLO-NAS models incorporate **attention mechanisms and reparameterization** during inference to enhance their ability to detect objects.

- The YOLO-NAS models initially underwent pre-training on the **Object365 benchmark** dataset, which contains 2 million images across 365 categories.
- They were further pre-trained using a method called **pseudo-labeling** on 123,000 unlabeled images from the COCO dataset.
- Additionally, techniques like **Knowledge Distillation** (KD) and Distribution Focal Loss (DFL) were employed to refine and enhance the training process of the YOLO-NAS models.

- They generated different variants of YOLO-NAS by varying the depth and positions of the **QSP and QCI blocks**.
- YOLO-NAS and **AutoNAC** are very highly sophisticated algorithms, YOLO-NAS Object Detection Model serves as an informative resource to comprehensively understand the intricacies of this model.

Training YOLO-NAS

YOLO-NAS is currently the latest addition to the YOLO family of models, and one should know how to train that on a custom dataset. Here we showed how to train YOLO-NAS on Thermal Dataset. YOLO-NAS Pose is also a model that performs very well in pose estimation, Introduction to YOLO-NAS Pose is a great resource to understand the model workings.

Summary and Conclusion

The whole YOLO series is full of engineering innovations and breakthroughs, It offers numerous applied machine learning ideas that can be used in specific scenarios. As Convolution Blocks are the basis for the entire YOLO series of models and as YOLO is mainly designed for edge devices, by going through all the YOLO models, one can learn different CNN optimization techniques.

- When YOLOv1 was first introduced, it was revolutionary as it pioneered the concept of a single-stage detection model. It stood out for not incorporating any region proposal mechanisms, allowing it to analyze the entire image at once and efficiently capture both global and local context.
- Later, the YOLOv2 and YOLOv3 models integrated advanced techniques that emerged at that time, such as the concept of **Feature Pyramid Networks (FPN)**, multi-scale training, and anchor boxes.

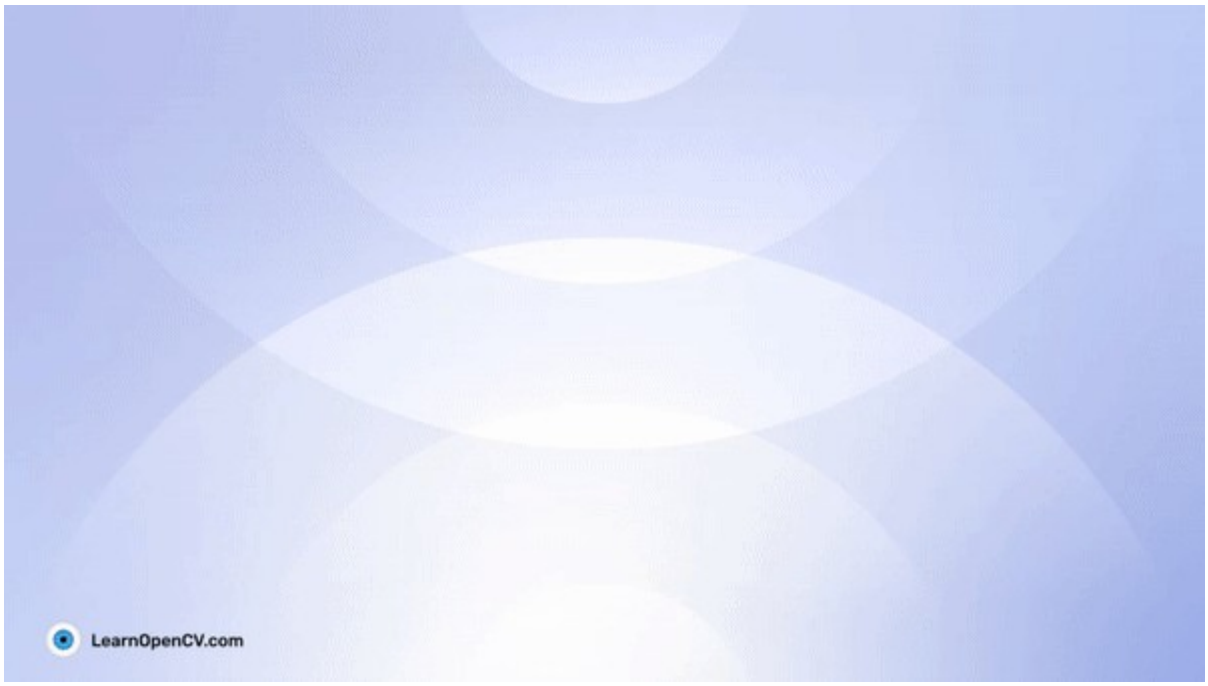
- But, YOLOv4 was different, the authors went above and beyond to bring some ground-breaking changes to YOLO. They introduced changes in the architecture, the Darknet53, which became **CSPDarknet53-PANet-SPP**, techniques such as bag of specials and freebies, Genetic Evolution algorithms, Attention modules, and so on.
- YOLOv5 brought changes that were very minimal and included most of the techniques from YOLOv4, what made YOLOv5, YOLOv5 is its Pytorch Implementation and how easy it is to train a Model using the Ultralytics.
- In YOLO-R, the authors experimented with a novel approach and worked toward evolving the model to support **multi-task learning**.
- But, in YOLO-X, it was again back to integrating new advances to the old YOLOv3 model, such as **Anchor-free, Decoupled heads**, Lable assignment and strong augmentations etc.
- In YOLOv7 and YOLOv6 authors experimented with the model architecture, YOLOv7 first brought the concept of re-parameterization in YOLO and model scaling, and YOLOv6 contributed by adding distillation and quantization techniques.
- Looking at YOLOv8 and its performance, previous models might seem over-engineered because it was built on top of YOLOv5 with minimal changes, such as replacing CSPLayer with C2f module, adding better loss functions that help the model to deal with small objects.
- YOLO-NAS was an algorithm-generated model with quantization blocks that made the model faster and more accurate.

Advanced Driver Assistance Systems (ADAS): Empowering Drivers

 Pranav Durai

OCTOBER 10, 20232 COMMENTS

Advanced Driver Assistance SystemsComputer VisionDeep LearningObject Detection



As vehicular technology continues to evolve at a rapid pace, the push towards safer, smarter, and more efficient driving experiences has been at the forefront of automotive innovation. **Advanced Driver Assistance Systems** (**ADAS**) is a key player in this technological revolution, and refers to a set of technologies and features integrated into modern vehicles to enhance driver safety, improve the driving experience, and assist in various driving tasks. It uses sensors, cameras, radar, and other technologies to monitor the vehicle's surroundings, collect data, and provide real-time feedback to the driver.

The roots of ADAS can be traced back to the early stages of automotive safety enhancements, such as the introduction of Anti-lock Braking Systems (ABS) in the late 1970s. However, the true emergence of ADAS as we understand it today began in the 2000s, with the integration of radar, cameras, and ultrasonic sensors.

Need for ADAS

In the current generation, there are multiple factors that require serious consideration:

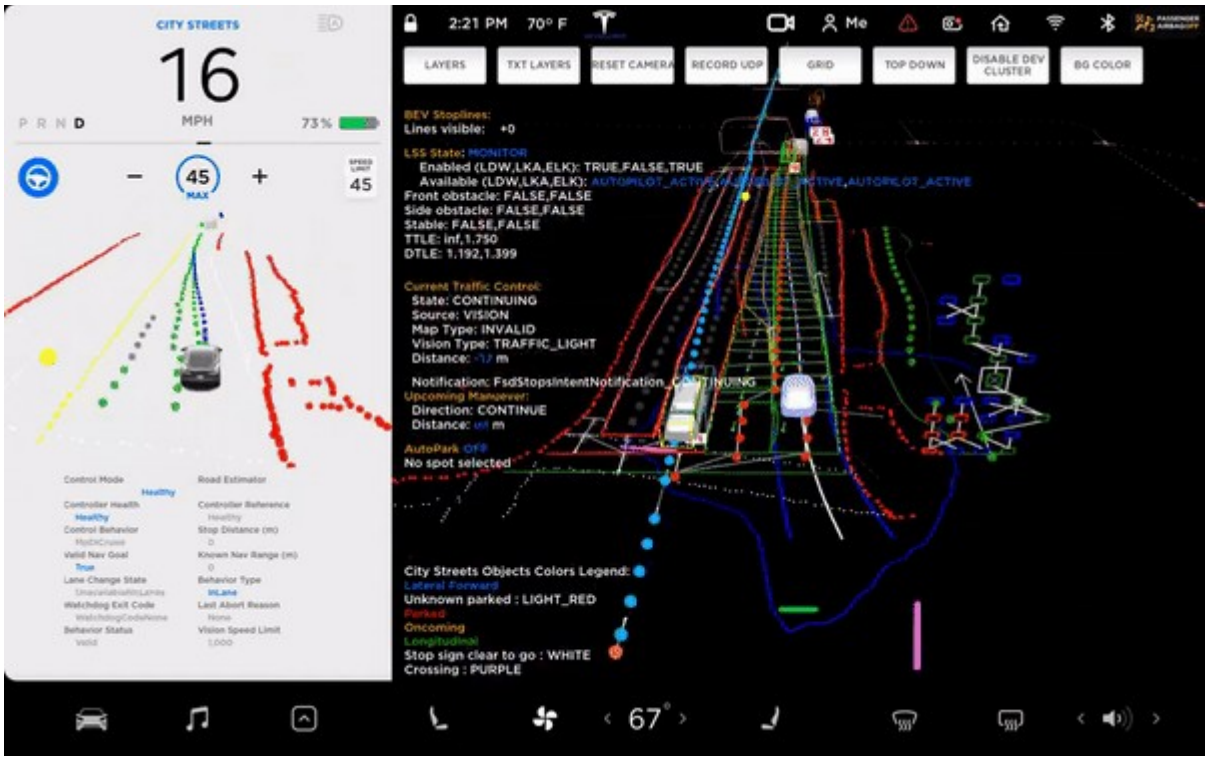


FIGURE 1: Visualization of Tesla's Autopilot Feature

Road Safety Statistics

- According to the World Health Organization (WHO), approximately 1.35 million people die in road traffic accidents yearly, making it the eighth leading cause of death globally.

- Road traffic crashes result in an estimated global economic loss of approximately 3% of GDP for most countries. This figure not only accounts for immediate medical costs but also the long-term impacts on productivity and rehabilitation costs.

ADAS can play a significant role in reducing these numbers. By integrating features like automatic emergency braking, lane departure warnings, and adaptive cruise control, the likelihood of collisions can be decreased.

Changing Urban Landscape

- With more people migrating to cities, the traffic density in urban areas is set to increase exponentially. This surge will be accompanied by new infrastructural developments and changing traffic patterns, creating more complex driving conditions.
- In many cities, road infrastructure hasn't kept pace with the rate of urbanization. Narrow roads, inadequate signage, and unpredictable traffic patterns pose challenges to even the most experienced drivers.
- To combat these challenges, ADAS tools like traffic sign recognition, pedestrian detection, and parking assistance can prove invaluable.

Environmental Considerations

- Road vehicles are responsible for significant global CO2 emissions, which play a major role in climate change. Inefficient driving patterns, like rapid acceleration and deceleration, can exacerbate these emissions.
- In urban areas, frequent stop-and-go traffic leads to excessive fuel consumption and increased emissions. This not only has environmental repercussions but also has direct health implications due to the degradation of air quality.

- Systems like adaptive cruise control can modulate speed efficiently, leading to smoother driving patterns and reduced fuel consumption.

Different levels of ADAS

There are multiple levels of advanced driver assistance systems, based on the offerings:

Level 0 [Manual]:

At this level, vehicles have no automation capabilities. The responsibility for control and decision-making rests entirely with the human driver.

Level 1 [Driver Assistance]:

This level offers limited assistance, either with steering or with acceleration / braking. Examples include adaptive cruise control and lane-keep assistance, but the driver remains largely in control. Lane detection is a crucial aspect of driver assistance.

Level 2 [Partial Automation]:

Vehicles at this level can manage both steering and acceleration/braking in certain conditions. However, the driver must always be alert and ready to take over, making sure to supervise the vehicle's actions closely.

Level 3 [Conditional Automation]:

Here, the vehicle can make dynamic decisions using AI components. It can handle some driving tasks but will prompt the driver to intervene when it encounters a scenario it can’t navigate.

Level 4 [High Automation]:

These vehicles can operate in both urban and dedicated highway settings autonomously. Nevertheless, there are still conditions or situations where human intervention might be necessary.

Level 5 [Full Automation]:

The ultimate level of automation, these vehicles don’t even need steering wheels or pedals. They’re designed to handle all driving tasks in every condition without any human input, except the final destination. This can also be a voice input.

What are the core components of ADAS?

Long-Range Radar – Continental ARS640

Highlights

- Type: Long-Range
- Dimensions: 137 x 90 x 40 mm
- Weight: ~500 grams
- Measurement Parameters: Range, Doppler, Azimuth, and Elevation
- Operating Temperature Range: -40° to +85° Celsius
- Power Consumption: 12 volts

The ARS640 Long-Range Radar is a cutting-edge automotive radar system designed to support highly automated driving scenarios, ranging from Level 3 to Level 5 autonomous systems. It boasts a compact form factor with dimensions of 137 x 90 x 40 mm (excluding the connector) and a weight of approximately 500 grams. With an impressive range of 300 meters, it enables precise and direct measurement of four critical dimensions: range, doppler, azimuth, and elevation.

FIGURE 2: Continental ARS640 Long-Range Radar

This radar system offers outstanding angular accuracy, with deviations of just ±0.1° in both azimuth and elevation, ensuring precise object detection and tracking. Its rapid update rate of 60 milliseconds enhances real-time awareness, while its operating temperature range of -40° to +85° Celsius ensures robust performance in diverse environmental conditions. With a low power dissipation of around 23 watts and a nominal supply voltage of 12 volts, it is an energy-efficient and reliable solution.

There are multiple applications for long-range radar in ADAS. Let’s explore a few:

Adaptive Cruise Control (ACC)

It is an advanced driver assistance system that automatically adjusts a vehicle’s speed to maintain a safe following distance from the vehicle ahead. Using long-range radar, and camera-based sensors, ACC continuously monitors the traffic environment and responds to changes in the flow. It intelligently decelerates the car accordingly when a slower-moving vehicle is detected in the same lane. Once the road ahead is clear, the system accelerates back to the preset speed, ensuring an optimal balance between maintaining speed and ensuring safety.

 **FIGURE 3:** Illustration of Adaptive Cruise Control

Short / Medium Range Radar – Continental SRR520

Highlights

- Type: Short-Range
- Dimensions: 83 x 68 x 22 mm
- Weight: ~135 grams
- Additional Features: Object list management, blind spot warning, lane change assist (Type IIIc), rear-cross traffic alert with braking, front-cross traffic alert with braking, etc
- Operating Temperature Range: -40°C to +85°C Celsius
- Power Consumption: 12 volts

The SRR520 short range radar is a compact and highly capable radar system designed for automotive applications. With dimensions of 83 x 68 x 22 mm and a mass of approximately 135 grams (without connectors), it is a lightweight and space-efficient solution. Operating in the 76-77 GHz frequency range, it offers an impressive range of 100 meters at a 0° angle, along with a wide field of view, featuring ± 90° detection and ± 75° measurement.

FIGURE 4: Continental SRR520 Short-Range Radar

This radar system boasts an exceptional update rate of 50 milliseconds and precise speed measurement accuracy of ± 0.07 kilometers per hour. It operates reliably in extreme temperatures ranging from -40°C to +85°C while dissipating 4.2 watts of power at a 12V supply voltage. Its advanced features include object list management, blind spot warning, lane change assist (Type IIIc), rear cross traffic alert with braking, front cross traffic alert with braking, rear pre-crash sensing, occupant safe exit support, and avoidance of lateral collisions.

There are multiple applications for short-medium range radar in ADAS. Let’s explore a few:

Cross Traffic Alert (CTA)

The Cross Traffic Alert (CTA) system becomes active when the driver shifts the vehicle into reverse. It employs short-to-medium range radar sensors, typically mounted on the rear corners of the vehicle, to scan for approaching vehicles from either side. If an approaching vehicle is detected, the system alerts the driver through visual, audible, or haptic warnings. These visual alerts often manifest in the side mirrors or on the vehicle's infotainment screen. Based on this alert, the driver can stop, continue reversing cautiously, or wait for the cross traffic to clear.

Rear Collision Warning (RTW)

It uses a fusion of multiple rear cameras with radar technology to inform the driver if another vehicle is approaching from either direction, when the vehicle is moving in reverse or backing out of a parking space.

FIGURE 5: Illustration of Rear Collision Warning – RTW

LiDAR: Light Detection and Ranging – InnovizTwo

Highlights

- Type: 3D Point-cloud Estimation
- Angular Resolution: 0.05°x0.05°
- Range of Frame Rate: 10, 15, 20 FPS
- Detection Range: 0.3 meters to 300 meters
- Maximum Field of View: 120°x43°

The InnovizTwo LiDAR system boasts impressive technical specifications, including a maximum angular resolution of 0.05°x0.05° and a configurable frame rate of 10, 15, or 20 FPS, making it a versatile and high-performance sensor for autonomous vehicles. With a detection range spanning from 0.3 meters to an impressive 300 meters, it excels in capturing objects at varying distances. Its expansive 120°x43° maximum field of view ensures comprehensive coverage of the vehicle's surroundings.

InnovizTwo's standout features include its ability to dynamically focus on four individually-controlled regions of interest within a limited FOV, enhancing visibility without compromising bandwidth, resolution, or frame rate. The system also excels at handling complex scenarios, recording multiple reflections per pixel and effectively processing laser pulses that encounter rain droplets, snowflakes, or multiple objects in their path.

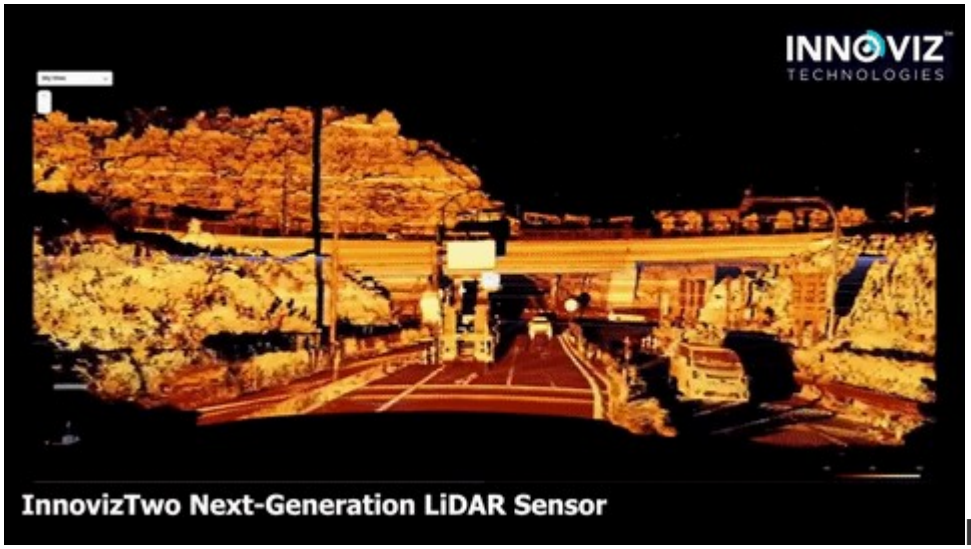


FIGURE 6: InnovizTwo High-Performance LiDAR Sensor

Its scanning pattern is characterized by contiguous pixels, eliminating data gaps crucial for building a safe autonomous vehicle perception system. This design ensures the system's capability to detect even small collision-relevant objects and pedestrians on the road surface. A vertical FOV offers high resolution evenly distributed across all regions, in contrast to other sensors that may prioritize the center and lose data towards the edges. This design provides more comprehensive data and accommodates mounting tolerances and varying driving conditions.

There are multiple applications for LiDAR in ADAS. Let's explore a few:

Emergency Braking (ER)

It utilizes advanced sensor technologies, including radar, LiDAR, and high-resolution cameras, to continuously scan the vehicle's forward environment. It employs sophisticated algorithms to process this data in real time, determining potential collision risks based on the relative speed and trajectory of detected objects. In situations where a potential frontal collision is imminent, and the driver fails to respond, the system automatically activates the braking mechanism, optimizing brake pressure to mitigate impact severity or, if possible, prevent the collision altogether.

Pedestrian Detection (PD)

PD uses LiDAR, which involves emitting laser beams and analyzing the reflected signals to identify and locate pedestrians in real-time. By capturing the 3D spatial information of a scene, LiDAR systems provide high-resolution data, enabling precise distance measurements and accurate differentiation of pedestrians from other objects in various lighting conditions.

Collision Avoidance (CA)

LiDAR technology plays a pivotal role in Collision Avoidance systems, offering high-resolution, 3D mapping of surroundings in real-time. By emitting laser beams and analyzing the reflected signals, LiDAR provides accurate distance measurements, enabling vehicles to detect obstacles and respond swiftly, even in congested traffic scenarios. This advanced sensing enhances the vehicle's safety and decision-making capabilities in complex environments.

Cameras – Continental MFC500

Highlights

- Type: Mono-Vision
- Dimensions: 88 x 70 x 38 mm
- Weight: ~200 grams
- Additional Features: Emergency braking, traffic sign assist, traffic jam assist, adaptive cruise control control, continuous lane centering, etc
- Operating Temperature Range: -40°C to +85°C Celsius
- Power Consumption: ~7 watts

The MFC500 Mono camera is a compact and lightweight automotive vision system with highly notable technical specifications. Measuring a mere 88 x 70 x 38mm and weighing less than 200g offers an unobtrusive form factor for seamless vehicle integration. The camera boasts an impressive, effective field of view, with horizontal coverage of up to 125° and vertical coverage of up to 60°, enabling comprehensive vision and perception capabilities for various driving scenarios. Operating within an extreme temperature range from -40° to +95° Celsius, it ensures reliable performance in harsh environmental conditions.

FIGURE 7: Continental MFC500 Mono-Camera

With a low power dissipation of less than 7 watts and a 12V supply voltage, the MFC500 is energy-efficient and compatible with automotive power systems. Its feature set aligns with key safety and regulatory standards, including support for General Safety Regulation (GSR 2.0) and EU-NCAP requirements. The camera offers a range of advanced driver assistance features, including emergency braking for vehicles, pedestrians, and cyclists, emergency lane keeping, speed limit information with automatic limitation, traffic sign assist, event data recording, adaptive cruise control with stop&go functionality and continuous lane centering assist. It also supports standard headlight systems up to matrix beams, enhancing nighttime visibility and safety. The MFC500 Mono camera is a powerful and versatile component for modern automotive safety and automation systems.

There are multiple applications for pure vision in ADAS. Let’s explore a few:

Traffic Sign Recognition (TSR)

Utilizing high-resolution cameras, these systems employ advanced image processing and deep learning algorithms to identify and interpret traffic signs and signals. Pure vision-based TSR provides

real-time feedback by analyzing visual data, ensuring drivers or autonomous systems adhere to road regulations. The effectiveness of such systems is enhanced by adaptive lighting conditions and the capability to distinguish between diverse sign types, colors, and shapes.

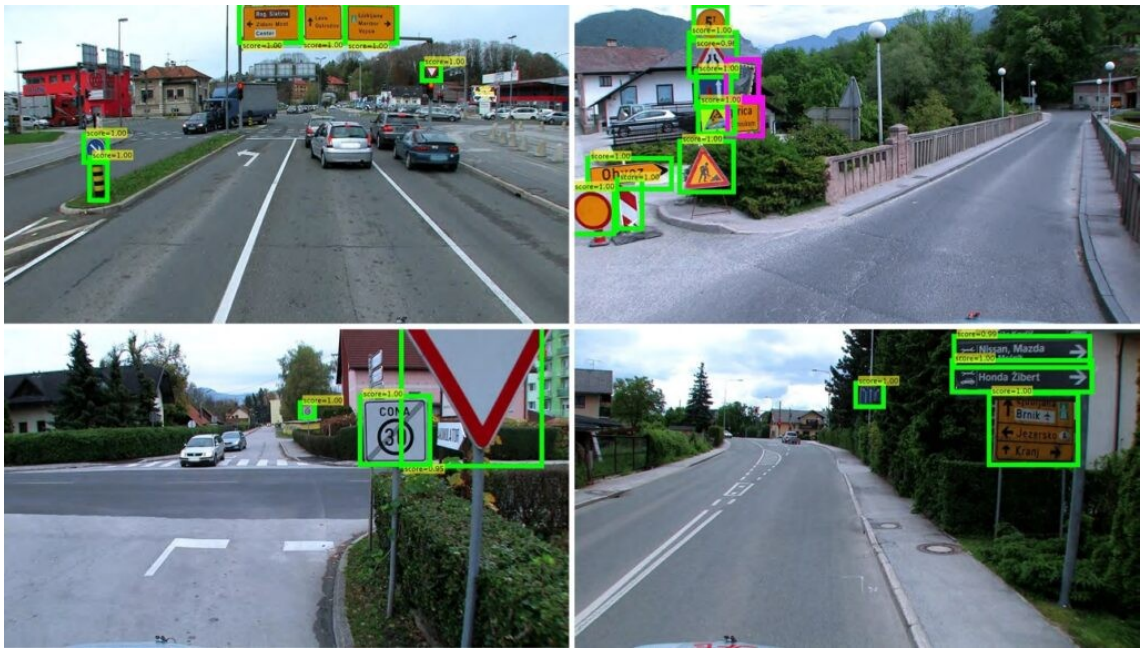


FIGURE 8: Illustration of Traffic Sign Recognition – TSR

Lane Departure Warning (LDW)

It provides dynamic alerts to the driver if the vehicle is drifting out of a specific lane on the road. LDW continuously monitors road lane markings to detect unintentional lane shifts. This technology, reliant on high-resolution cameras, processes the road's visual data in real-time to ensure the vehicle remains within its designated path, enhancing road safety and assisting drivers in maintaining proper lane discipline.

FIGURE 9: Illustration of Park Assist – PA

Surround View (SV)

SV is a critical component of advanced driver assistance systems, enhancing driver safety and awareness by offering a comprehensive 180 to 360-degree view of the vehicle's surroundings. This technology relies on the fusion of data from ultrawide-angle cameras strategically positioned on the vehicle to eliminate blind spots and provide a clear, real-time visual representation of the environment.

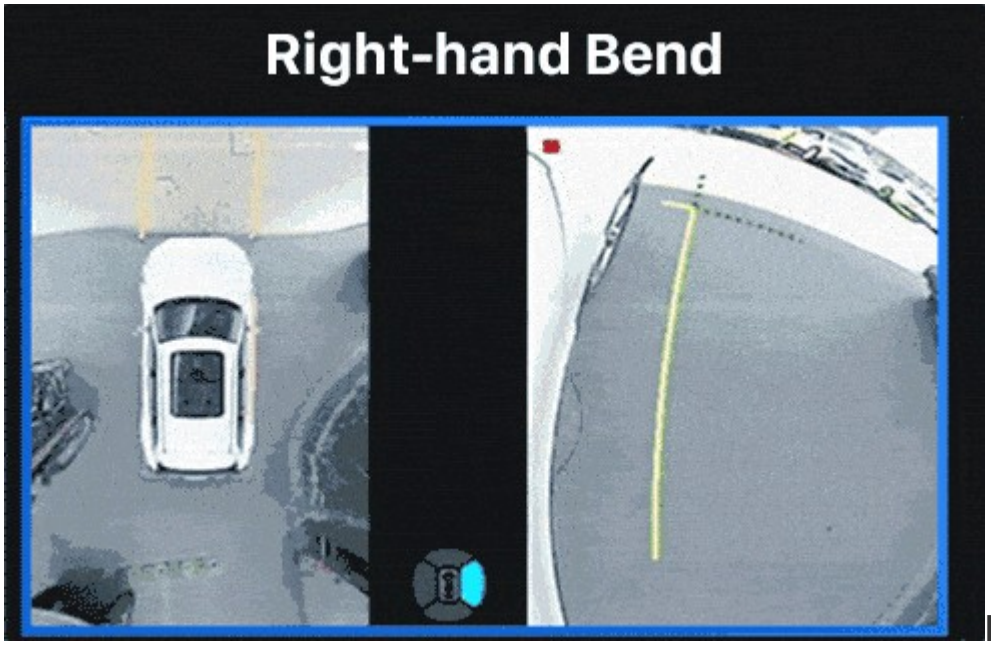


FIGURE 10: Illustration of Surround View – SV

ADAS in Action: How Does ADAS work?

The first automobile manufacturer that comes to mind when someone says ‘Self-Driving Car’ is Tesla. This bleeding-edge feature called ‘Autopilot’ relies heavily on artificial intelligence and deep learning techniques to process and interpret the data from sensors. Some of the software components and techniques involved include:

Neural Networks

Tesla uses deep neural networks for image and data analysis. Convolutional Neural Networks (CNNs) are used to recognize and track objects, such as other vehicles, pedestrians, and lane markings.

FIGURE 11: [What do Tesla's Neural Networks see?]

Sensor Fusion

The system combines data from cameras, ultrasonic sensors, and radar to create a more comprehensive and accurate understanding of the vehicle's surroundings. This sensor fusion is crucial for safe autonomous driving.

Mapping and Localization

Tesla vehicles use high-definition maps to help with localization. GPS data is combined with information from sensors to position the vehicle on the road precisely.

Control Algorithms

Sophisticated control algorithms are used to make driving decisions, such as maintaining the vehicle within lanes, changing lanes when necessary, and navigating complex traffic scenarios.

What is The Future of Advanced Driver Assistance Systems?

Technology is always evolving, so it is important to keep an eye on what the future holds for this specific technology.

Increased Automation

While Level 2 automation (e.g., Tesla's Autopilot) was common in 2021, Level 3 and Level 4 systems, which require less driver intervention, are expected to become more prevalent.

Better Sensor Technology

The future of ADAS will likely involve the integration of more advanced sensors, higher-resolution cameras, and improved LiDAR and radar systems to provide a more accurate and comprehensive view of the vehicle's surroundings.

User Experience and Human-Machine Interaction

The design of user interfaces and the interaction between humans and AI-driven vehicles will be critical to ADAS development. Natural language processing (NLP) and computer vision will play a role in creating more intuitive and user-friendly interfaces.

Environmental Impact

There will be a growing emphasis on reducing the environmental impact of transportation. ADAS may play a role in optimizing driving patterns for fuel efficiency and reducing emissions.

Conclusions

Advanced Driver Assistance Systems, uses industry-leading computer vision technology to represent a pivotal technological frontier in the automotive industry. Leveraging cutting-edge developments in artificial intelligence, deep learning, and sensor technology, it has transcended conventional driver assistance systems, evolving into a sophisticated suite of tools designed to enhance safety, automation, and user experience. The integration of high-resolution cameras, LiDAR, radar, and ultrasonic sensors, combined with the ever-advancing capabilities of neural networks and reinforcement learning algorithms, empowers ADAS to navigate complex real-world driving scenarios with unprecedented precision.

At this point, we’re evolving well beyond the scope of driver assistance and into the domain of autonomous driving. But no matter how far our solutions advance, they’ll all owe their origins to the building blocks of our lifesaving core ADAS technology.