Report Finall Exam Project

VISUAL-TRAFFIC-MONITORING-COMPUTER-VISION CHALLENGE

I. Student Information

- Name : Đinh Đức Văn
- Company Onlimical VTC

II. Problem Description

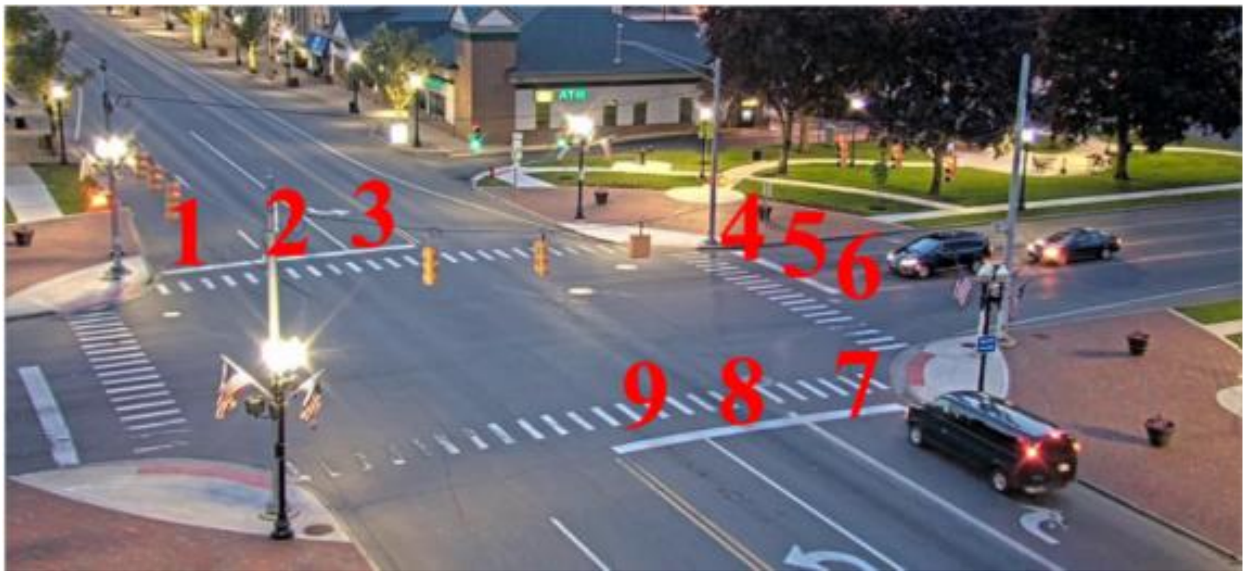
The goal of this project is to develop an automatic system for video analysis that enables visual traffic monitoring at a road intersection . The system gets as input data
the video stream from a static camera and should be able to: (1) classify road lanes as being occupied or not given a video frame; (2) track a specific vehicle in a given video; (3) count the trajectories of vehicles in the intersection in a given video

III. Logical processing and instructions for solving problems from mathematical problems

➢ Description of problem data

The release data directory (available at DATA_SOURCES) contains three directories: train, test and evaluation. The directories train and test have similar structure, although the test data will be made available after the deadline. The train directory contains the data organized in four subdirectories corresponding to the three tasks that you need to solve and the context videos shared by these tasks. The subdirectories are:

1. context videos all tasks - this directory contains 15 training videos which represents the context video data for each of the three tasks. Notice that each video is taken at a given moment of a day from T0 to T1.

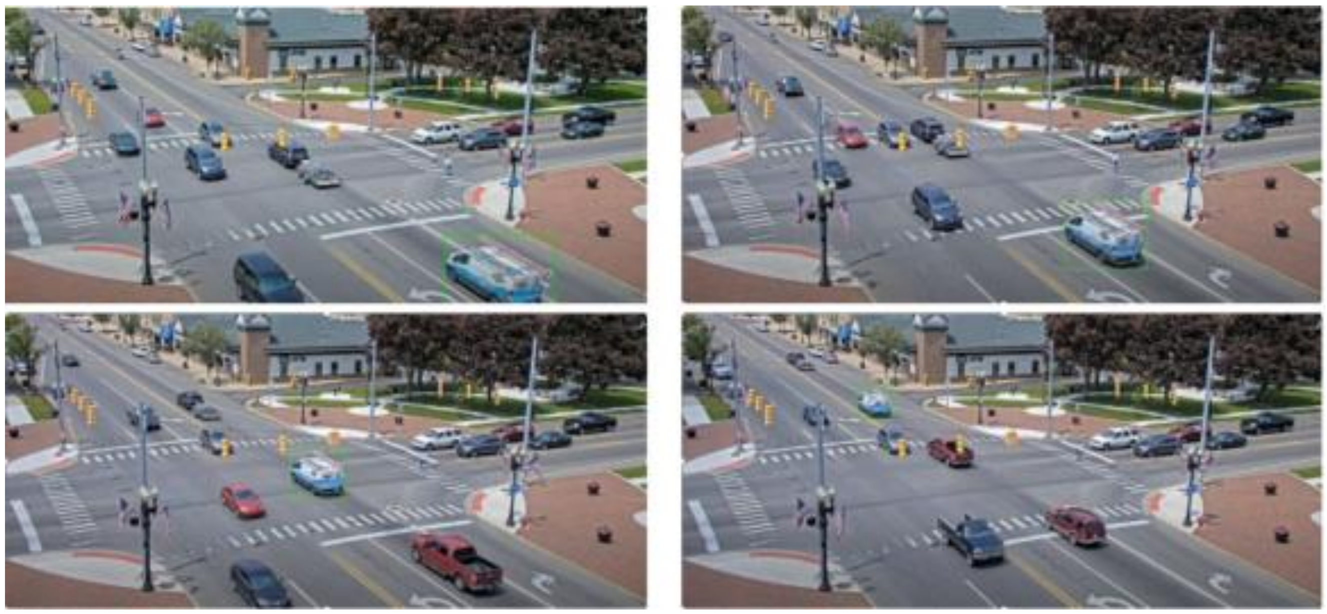


Road lanes are numbered from 1 to 3 for the upper left part of the intersection, 4 to 6 for the right part and 7 to 9 for the bottom right part of the intersection. We do not consider the left part of the intersection.

- Misstion Task1: Consists in correctly classifying the road lanes listed in the query file as being occupied (label 1) or not (label 0). The lanes are numbered:
  from 1 to 3 for the upper left part of the intersection
  4 to 6 for the right part
  and 7 to 9 for the bottom right part of the intersection.

We do not consider the left part of the intersection. The format of the query files is the following: on the first row it is specified the number L of lanes for which you have to do the binary classification and then each of the following L rows lists a lane number.

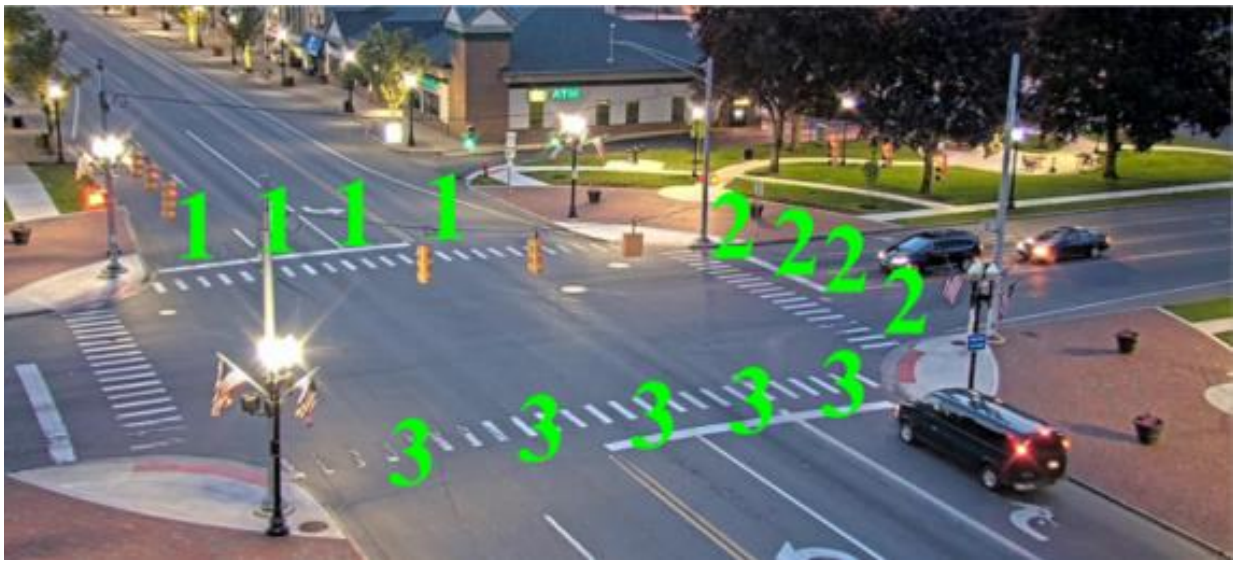
2. This directory contains 15 training videos taken in the interval from T2 to T3. The task is to track a specific vehicle from the initial frame to the final frame of the video (see Figure 4). The initial bounding box of the vehicle to be tracked is provided for the first frame (the annotation follows the format [xmin ymin xmax ymax] where (xmin,ymin) is the top left corner and (xmax,ymax) is the bottom right corner of the initial bounding-box.



The figure displays four frames of a training video containing annotation (green bounding-box) of a specific vehicle

- Misstion Task 2: In each video we will consider that your algorithm correctly tracks the vehicle if in more (greater or equal) than 80% of the video frames your algorithm correctly localizes the vehicle to be tracked. We consider that your algorithm correctly localizes the vehicle to be tracked in a specific frame if the value of the IOU (intersection over union) between the window provided by your algorithm and the ground-truth window is more than20%. The format that you need to follow is the one used in the ground-truth files (located in the subdirectory ground-truth) with the first line containing the number of frames N of the video, and each line having the format [frame index x_{min} y_{min}]

xmax ymax]. The first frame for which we provide the bounding box initialization has frame index 0, the last frame of a video with N frames has frame index N − 1. Please note that the first line of the annotation file has the format [N -1 -1 -1 -1] as it is easy to load an entire matrix M × 5 (first line is [N -1 -1 -1 -1], then the following M − 1 lines are of the form [frame index xmin ymin xmax ymax], so M ≤ N + 1) to assess the correctness of your algorithm. Notice that if the vehicle disappears from the video your algorithm should not output any detection in the corresponding frames (in this case M < N + 1).

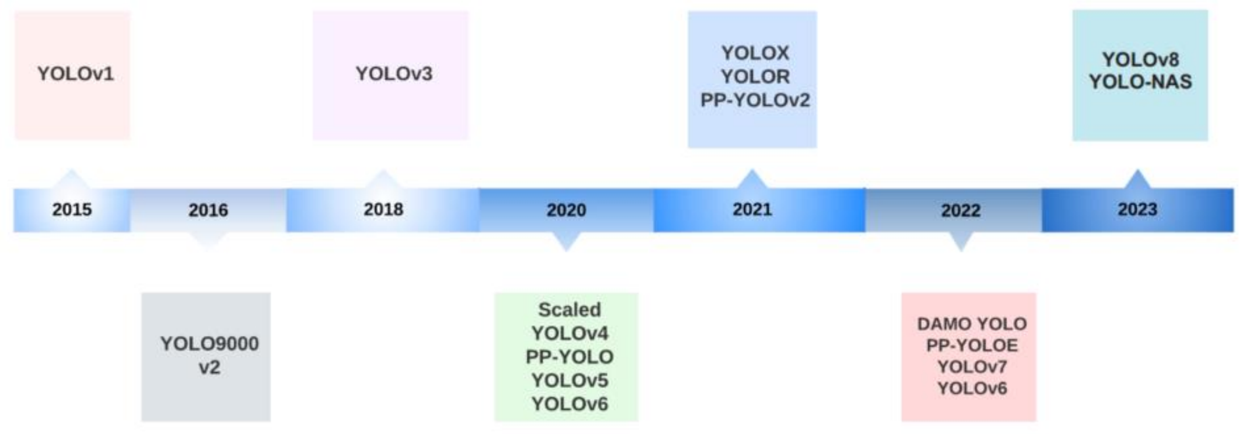


Regions of interest for Task3 are numbered with 1 (upper part of the intersection), 2(right part) and 3 (bottom part). We do not consider the left part of the intersection.

- this directory contains 15 training videos. The goal here is to count the trajectories of vehicles in the given video. We are interest to count the number of vehicles between the three regions of interest . The ground-truth files specifies the number of vehicles that go from region 1 to region 2, from region 1 to region 3, from region 2 to region 3 and so on.

IV. Applied Knowledge

- Use compate IOU evalution, Yolo$_{v3}$, Yolo$_{v4}$, Yolo$_{v7}$ application in Project Visual-Traffic-Monitoring-Computer-Vision Challenge.



1. YOLO Applications Across Drivese Fileds

- YOLO's real-time object detection capabilities have been invaluable in autonomous vehicle systems, enabling quick identification and tracking of various objects such as vehicles, pedestrians, bicycles, and other obstacles . These capabilities have been applied in numerous fields, including action recognition in video sequences for surveillance , sports analysis , and human-computer interaction .

- YOLO models have been used in agriculture to detect and classify crops , pests, and diseases , assisting in precision agriculture techniques and automating farming processes. They have also been adapted for face detection tasks in biometrics, security, and facial recognition systems .

- Security systems have integrated YOLO models for real-time monitoring and analysis of video feeds, allowing rapid detection of suspicious activities , social distancing, and face mask detection. The models have also been applied in surface inspection to detect defects and anomalies, enhancing quality control in manufacturing and production processes .

- In traffic applications, YOLO models have been utilized for tasks such as license plate detection and traffic sign recognition , contributing to the development of intelligent transportation systems and traffic management solutions. They have been employed in wildlife detection and monitoring to identify endangered species for biodiversity conservation and ecosystem management . Lastly, YOLO has been widely used in robotic applications and object detection from drones.

2. **Object Detection Metrics and Non-Maximum Suppression (NMS)**

- The Average Precision (AP), traditionally called *Mean Average Precision* (mAP), is the commonly used metric for evaluating the performance of object detection models. It measures the average precision across all categories, providing a single value to compare different models. The COCO dataset makes no distinction between AP and mAP. In the rest
of this paper, we will refer to this metric as AP. In YOLOv1 and YOLOv2, the dataset utilized for training and benchmarking was PASCAL VOC 2007, and VOC 2012 . However, from YOLOv3 onwards, the dataset used is Microsoft COCO (Common Objects in Context) . The
AP is calculated differently for these datasets. The following sections will discuss the rationale behind AP and explain how it is computed.

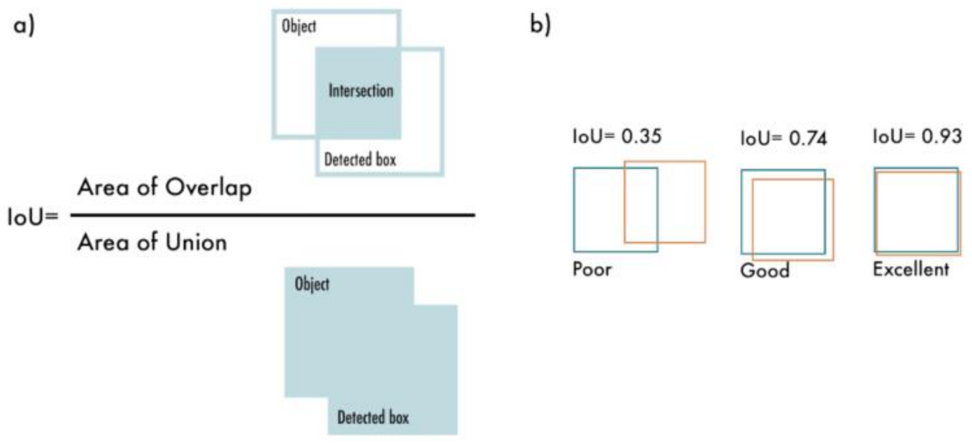
3. Evaluations

The AP metric is based on precision-recall metrics, handling multiple object categories, and defining a positive prediction using Intersection over Union (IoU).

- **Precision and Recall**: Precision measures the accuracy of the model's positive predictions, while recall measures the proportion of actual positive cases that the model correctly identifies. There is often a trade-off between precision and recall; for example, increasing the number of detected objects (higher recall) can result in more false positives (lower precision). To account for this trade-off, the AP metric incorporates the precision-recall curve that plots precision against recall for different confidence thresholds. This metric provides a balanced assessment of precision and recall by considering the area under the precision-recall curve.

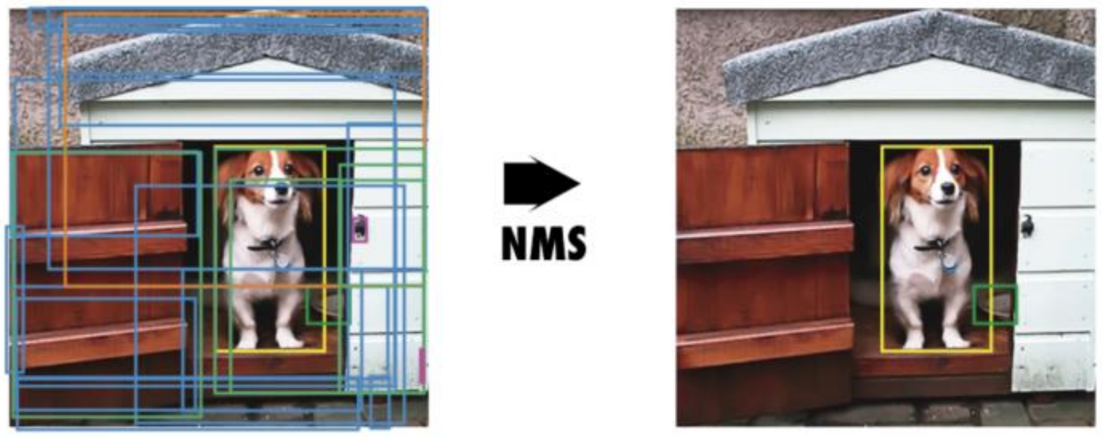
- **Handling multiple object categories**: Object detection models must identify and localize multiple object categories in an image. The AP metric addresses this by calculating each category's average precision (AP) separately and then taking the mean of these APs across all categories (that is why it is also called mean average precision). This approach ensures that the model's performance is evaluated for each category individually, providing a more comprehensive assessment of the model's overall performance.

- **Intersection over Union**: Object detection aims to accurately localize objects in images by predicting bounding boxes. The AP metric incorporates the Intersection over Union (IoU) measure to assess the quality of the predicted bounding boxes. IoU is the ratio of the intersection area to the union area of the predicted bounding box and the ground truth bounding box (see Figure . It measures the overlap between the ground truth and predicted bounding boxes. The COCO benchmark considers multiple IoU thresholds to evaluate the model's performance at different levels of localization accuracy.



YOLO: You Only Look Once

YOLO by Joseph Redmon et al. was published in CVPR 2016 . It presented for the first time a real-time end-to-end approach for object detection. The name YOLO stands for "You Only Look Once," referring to the fact that it was able to accomplish the detection task with a single pass of the network, as opposed to previous approaches that either used sliding windows followed by a classifier that needed to run hundreds or thousands of times per image or the more advanced methods that divided the task into two-steps, where the first step detects possible regions with objects or *regions proposals* and the second step run a classifier on the proposals. Also, YOLO used a more straightforward output.



Non-Maximum Suppression (NMS). a) Shows the typical output of an object detection model containing multiple overlapping boxes. b) Shows the output after NMS. based only on regression to predict the detection outputs as opposed to Fast R

CNN that used two separate outputs, a classification for the probabilities and a regression for the boxes coordinates.

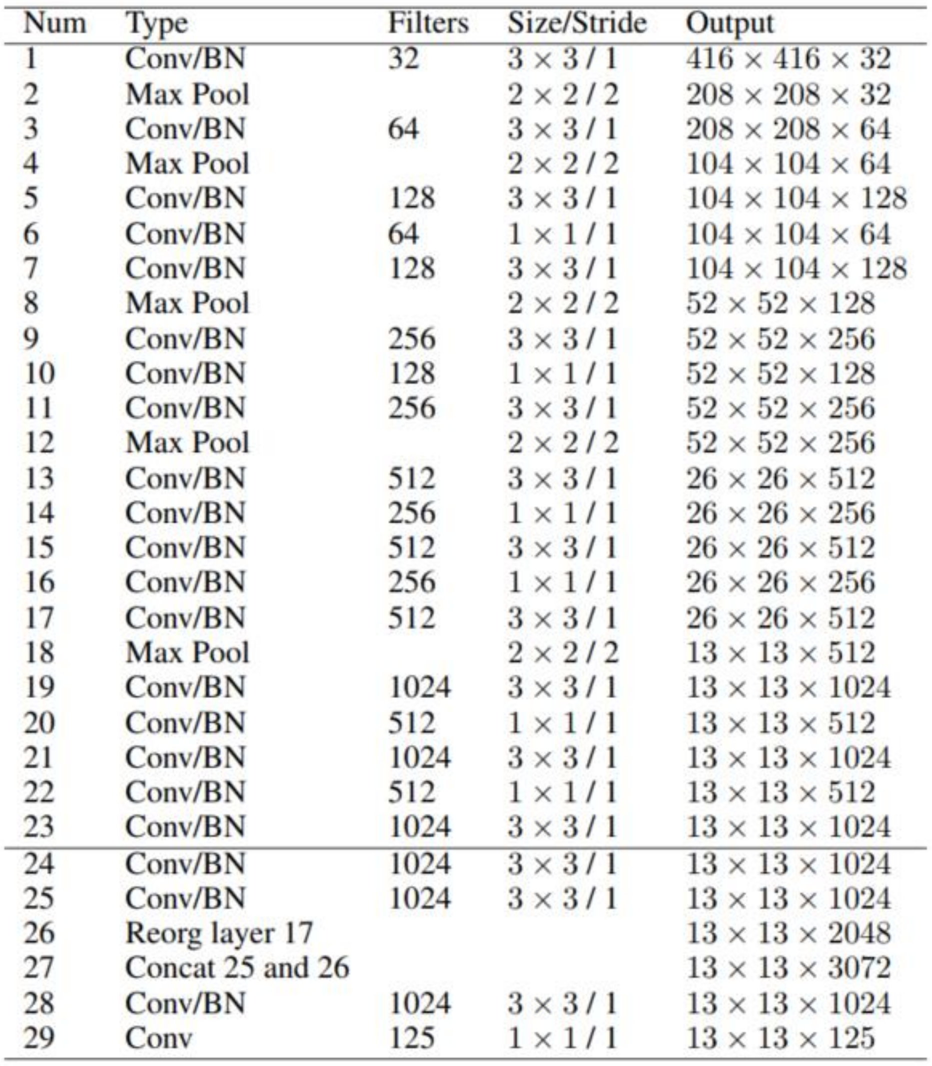
4. Yolov3

YOLOv3 [46] was published in ArXiv in 2018 by Joseph Redmon and Ali Farhadi. It included significant changes and a bigger architecture to be on par with the state-of-the-art while keeping real-time performance. In the following, we described the changes with respect to YOLOv2

1 . Bounding box prediction. Like YOLOv2, the network predicts four coordinates for each bounding box *tx*, *ty*, *tw*, and *th*; however, this time, YOLOv3 predicts an *objectness score* for each bounding box using logistic regression. This score is 1 for the anchor box with the highest overlap with the ground truth and 0 for the rest anchor boxes. YOLOv3, as opposed to Faster R-CNN , assigns only one anchor box to each ground truth object. Also, if no anchor box is assigned to an object, it only incurs in classification loss but not localization loss or confidence loss.

2. **Class Prediction**. Instead of using a softmax for the classification, they used binary cross-entropy to train independent logistic classifiers and pose the problem as a multilabel classification. This change allows assigning multiple labels to the same box, which may occur on some complex datasets with overlapping labels. For example, the same object can be a Person and a Man.

Table 2: YOLOv2 Architecture. Darknet-19 backbone (layers 1 to 23) plus the detection head composed of the last four convolutional layers and the passthrough layer that reorganizes the features of the 17th output of $26 \times 26 \times 512$ into $13 \times 13 \times 2048$ followed by concatenation with the 25th layer. The final convolution generates a grid of 13×13 with 125 channels to accommodate 25 predictions (5 coordinates + 20 classes) for five bounding boxes.



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

3. New backbone. YOLOv3 features a larger feature extractor composed of 53 convolutional layers with residual connections. Section 6.1 describes the architecture in more detail.

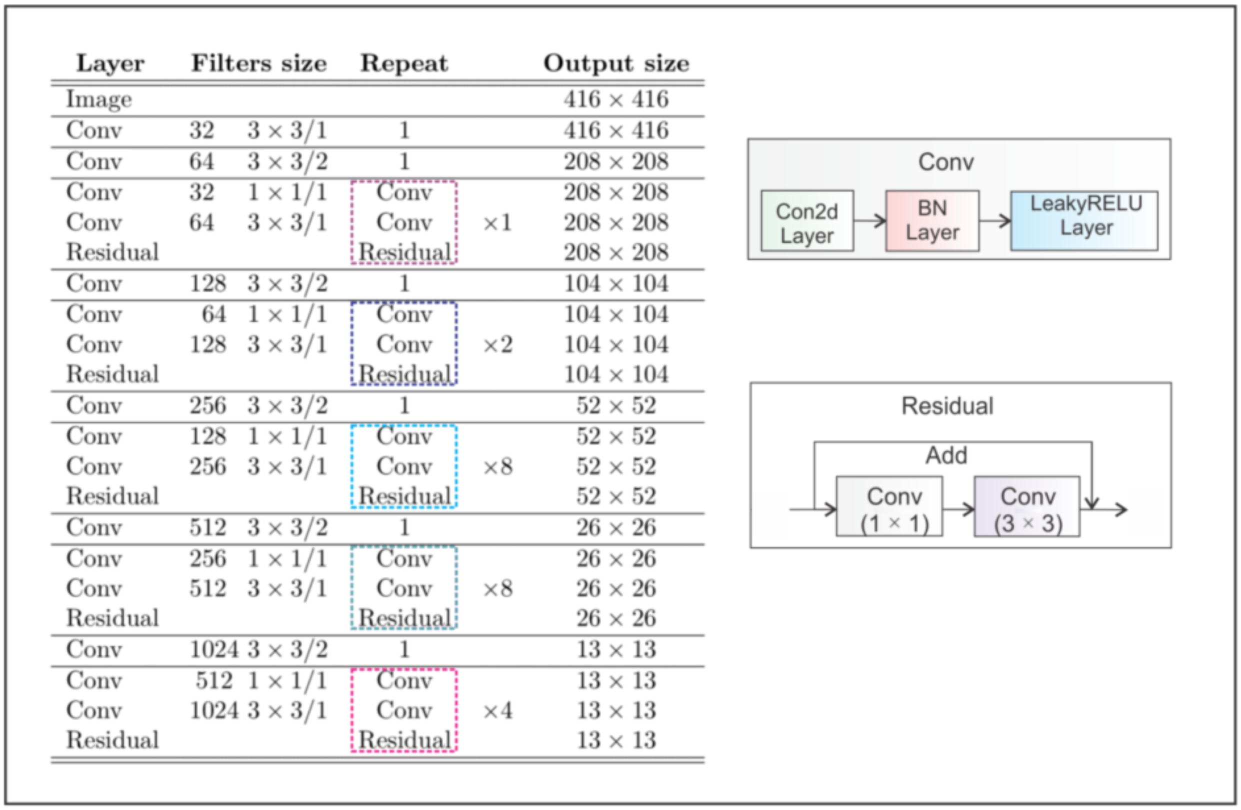
4. Spatial pyramid pooling (SPP) Although not mentioned in the paper, the authors also added to the backbone a modified SPP block that concatenates multiple max pooling outputs without subsampling (stride = 1), each with different kernel sizes k × k where k = 1, 5, 9, 13 allowing a larger receptive field. This version is called YOLOv3-spp and was the best-performed version improving the AP50 by 2.7%.

5. Multi-scale Predictions. Similar to Feature Pyramid Networks [49], YOLOv3 predicts three boxes at three different scales. Section 6.2 describes the multi-scale prediction mechanism with more details.

6. Bounding box priors. Like YOLOv2, the authors also use k-means to determine the bounding box priors of anchor boxes. The difference is that in YOLOv2, they used a total of five prior boxes per cell, and in YOLOv3, they used three prior boxes for three different scales.

1. YOLOv3 Architecture

The architecture backbone presented in YOLOv3 is called Darknet-53. It replaced all max-pooling layers with strided convolutions and added residual connections. In total, it contains 53 convolutional layers. Figure 8 shows the architecture details. The Darknet-53 backbone obtains Top-1 and Top-5 accuracies comparable with ResNet-152 but almost 2× faster.



Layer	Filters size		Repeat		Output size
Image					416×416
Conv	32	$3 \times 3/1$	1		416×416
Conv	64	$3 \times 3/2$	1		208×208
Conv	32	$1 \times 1/1$	Conv		208×208
Conv	64	$3 \times 3/1$	Conv	$\times 1$	208×208
Residual			Residual		208×208
Conv	128	$3 \times 3/2$	1		104×104
Conv	64	$1 \times 1/1$	Conv		104×104
Conv	128	$3 \times 3/1$	Conv	$\times 2$	104×104
Residual			Residual		104×104
Conv	256	$3 \times 3/2$	1		52×52
Conv	128	$1 \times 1/1$	Conv		52×52
Conv	256	$3 \times 3/1$	Conv	$\times 8$	52×52
Residual			Residual		52×52
Conv	512	$3 \times 3/2$	1		26×26
Conv	256	$1 \times 1/1$	Conv		26×26
Conv	512	$3 \times 3/1$	Conv	$\times 8$	26×26
Residual			Residual		26×26
Conv	1024	$3 \times 3/2$	1		13×13
Conv	512	$1 \times 1/1$	Conv		13×13
Conv	1024	$3 \times 3/1$	Conv	$\times 4$	13×13
Residual			Residual		13×13

Figure 8: YOLOv3 Darknet-53 backbone. The architecture of YOLOv3 is composed of 53 convolutional layers, each with batch normalization and Leaky ReLU activation. Also, residual connections connect the input of the 1 × 1 convolutions across the whole network with the output of the 3 × 3 convolutions. The architecture shown here consists of only the backbone; it does not include the detection head composed of multi-scale predictions.

2. YOLOv3 Multi-Scale Predictions

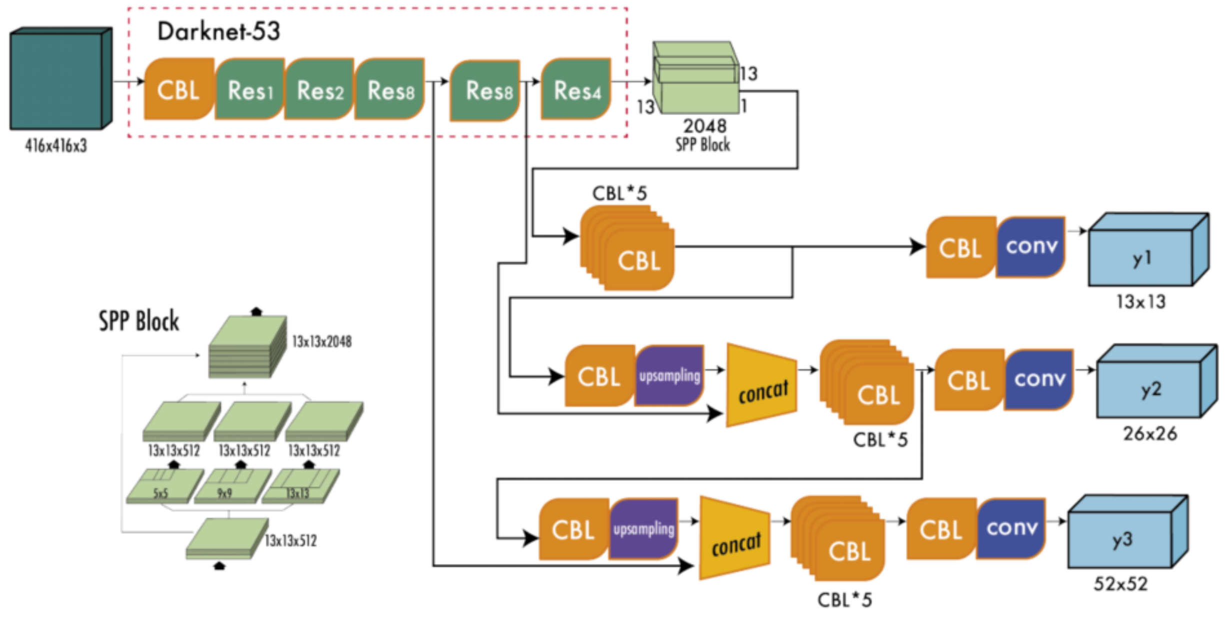
Besides a larger architecture, an essential feature of YOLOv3 is the multi-scale predictions, i.e., predictions at multiple grid sizes. This helped to obtain finer detailed boxes and significantly improved the prediction of small objects, which was one of the main weaknesses of the previous versions of YOLO.

The multi-scale detection architecture shown in Figure 9 works as follows: the first output marked as y1 is equivalent to the YOLOv2 output, where a 13 × 13 grid defines the output. The second output y2 is composed by concatenating the output after the (Res × 4) of Darknet-53 with the output after (the Res × 8). The feature maps have different sizes, i.e., 13 × 13 and 26 × 26, so there is an upsampling operation before the concatenation. Finally, using an upsampling operation, the third output y3 concatenates the 26 × 26 feature maps with the 52 × 52 feature maps.

For the COCO dataset with 80 categories, each scale provides an output tensor with a shape of N×N ×[3×(4+1+80)] where N × N is the size of the feature map (or grid cell), the 3 indicates the boxes per cell and the 4 + 1 include the four coordinates and the objectness score

3. YOLOv3 Results

When YOLOv3 was released, the benchmark for object detection had changed from PASCAL VOC to Microsoft COCO. Therefore, from here on, all the YOLOs are evaluated in the MS COCO dataset. YOLOv3-spp achieved an average precision AP of 36.2% and AP50 of 60.6% at 20 FPS, achieving state-of-the-art at the time and 2 × faster.



YOLOv3 Multi-scale detection architecture. The output of the Darknet-53 backbone is branched to three different outputs marked as y1, y2, and y3, each of increased resolution. The final predicted boxes are filtered using on-maximum suppression. The CBL (Convolution-BatchNorm-Leaky ReLU) blocks comprise one convolution layer with batch normalization and leaky ReLU. The Res blocks comprise one CBL followed by two CBL structures with a residual connection.

4. Backbone, Neck, and Head

At this time, the architecture of object detectors started to be described in three parts: the backbone, the neck, and the head. shows a high-level backbone, neck, and head diagram.

The backbone is responsible for extracting useful features from the input image. It is typically a convolutional neural network (CNN) trained on a large-scale image classification task, such as ImageNet. The backbone captures hierarchical features at different scales, with lower-level features (e.g., edges and textures) extracted in the earlier layers and higher-level features (e.g., object parts and semantic information) extracted in the deeper layers.

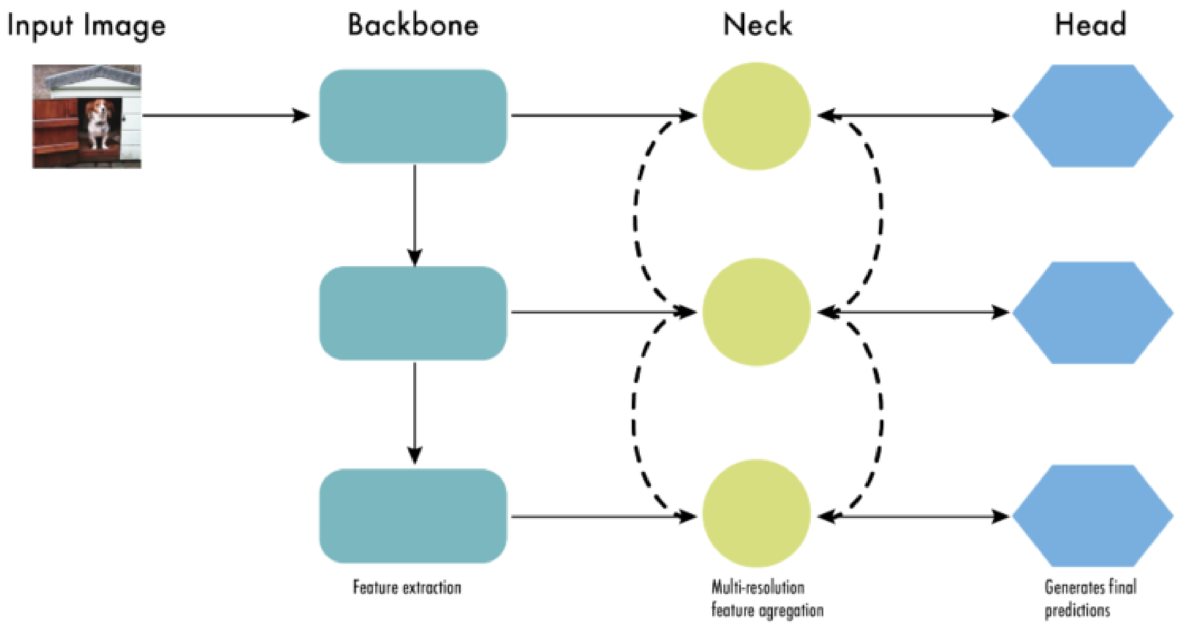
The neck is an intermediate component that connects the backbone to the head. It aggregates and refines the features extracted by the backbone, often focusing on enhancing the spatial and semantic information across different scales. The neck may include additional convolutional layers, feature pyramid networks (FPN), or other mechanisms to improve the representation of the features.

The head is the final component of an object detector; it is responsible for making predictions based on the features provided by the backbone and neck. It typically consists of one or more task-specific subnetworks that perform classification, localization, and, more recently, instance segmentation and pose estimation. The head processes the features the neck provides, generating predictions for each object candidate. In the end, a post-processing step, such as non-maximum suppression (NMS), filters out overlapping predictions and retains only the most confident detections.

In the rest of the YOLO models, we will describe the architectures using the backbone, neck, and head.

5. **YOLO v4**

Two years passed, and there was no new version of YOLO. It was until April 2020 that Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao released in ArXiv the paper for YOLOv4 . At first, it felt odd that different authors presented a new "official" version of YOLO; however, YOLOv4 kept the same YOLO philosophy —real-time, open source, single shot, and darknet framework— and the improvements were so satisfactory that the community rapidly embrace this version as the official YOLOv4


Input Image
Backbone
Neck
Head
Feature extraction
Multi-resolution feature agregation
Generates final predictions

YOLOv4 tried to find the optimal balance by experimenting with many changes categorized as bag-of-freebies and bag-of-specials. Bag-of-freebies are methods that only change the training strategy and increase training cost but do not increase the inference time, the most common being data augmentation. On the other hand, bag-of-specials are methods that slightly increase the inference cost but significantly improve accuracy.

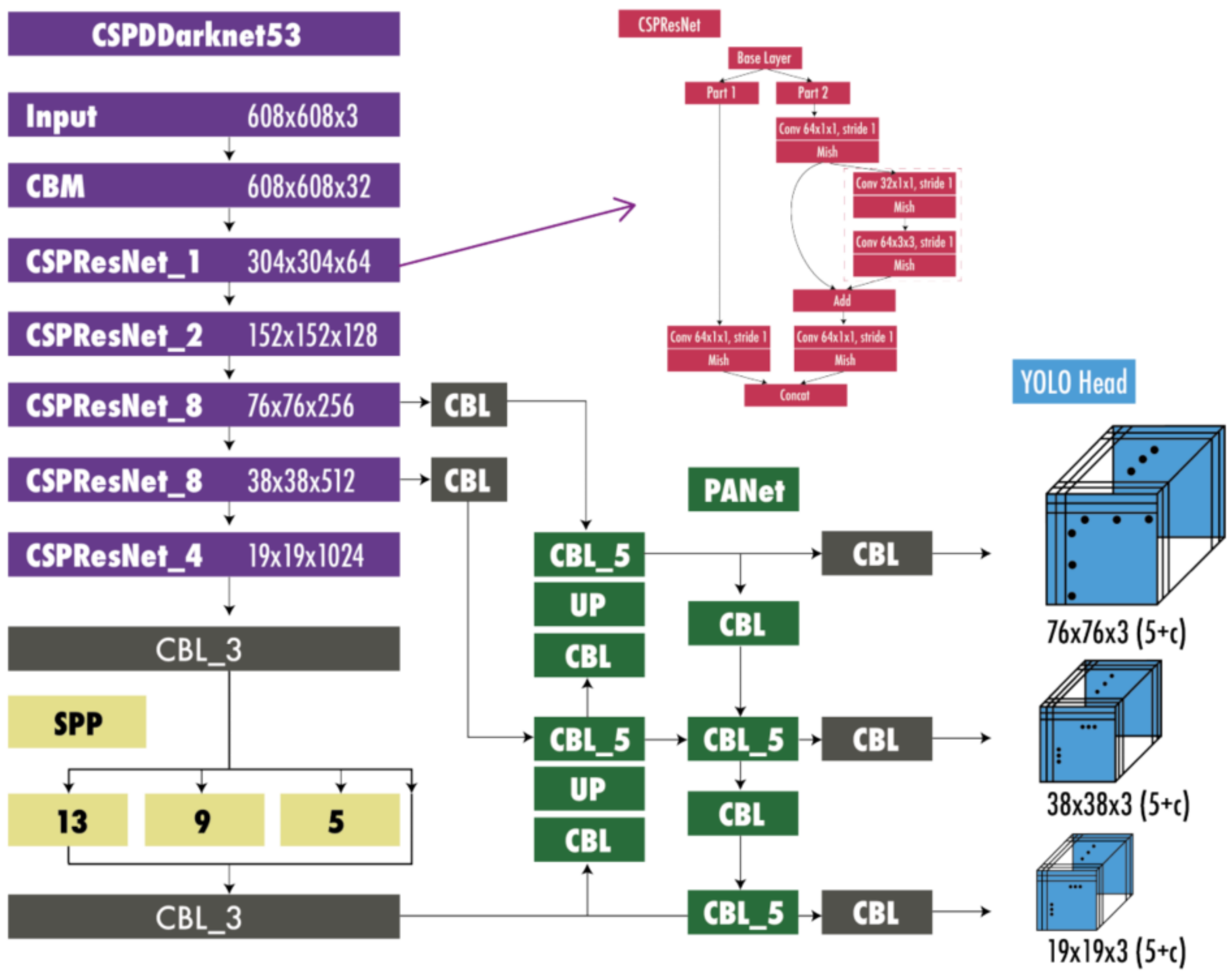
We summarize the main changes of YOLOv4 in the following points:

- An Enhanced Architecture with Bag-of-Specials (BoS) Integration. The authors tried multiple architectures for the backbone, such as ResNeXt50, EfficientNet-B3, and Darknet-53. The best-performing architecture was a modification of Darknet-53 with cross-stage partial connections (CSPNet), and Mish activation function as the backbone. For the neck, they used the modified version of spatial pyramid pooling (SPP) from YOLOv3-spp and multi-scale predictions as in YOLOv3, but with a modified version of path aggregation network (PANet) instead of FPN as well as a modified spatial attention module (SAM) . Finally, for the detection head, they use anchors as in YOLOv3. Therefore, the model was called CSPDarknet53-PANet-SPP. The cross-stage partial connections (CSP) added to the Darknet-53 help reduce the computation of the model while keeping the same accuracy. The SPP block, as in YOLOv3-spp increases the receptive field without affecting the inference speed. The modified version of PANet concatenates the features instead of adding them as in the original PANet paper.

- Integrating bag-of-freebies (BoF) for an Advanced Training Approach. Apart from the regular augmentations such as random brightness, contrast, scaling, cropping, flipping, and rotation, the authors implemented mosaic augmentation that combines four images into a single one allowing the detection of objects outside their usual context and also reducing the need for a large mini-batch size for batch normalization. For regularization, they used DropBlock that works as a replacement of Dropout but for convolutional neural networks as well as class label smoothing. For the detector, they added CIoU loss and Cross mini-bath normalization (CmBN) for collecting statistics from the entire batch instead of from single mini-batches as in regular batch normalization.

- **Self-adversarial Training (SAT)**. To make the model more robust to perturbations, an adversarial attack is performed on the input image to create a deception that the ground truth object is not in the image but keeps the original label to detect the correct object.

- **Hyperparameter Optimization with Genetic Algorithms**. To find the optimal hyperparameters used for training, they use genetic algorithms on the first 10% of periods, and a cosine annealing scheduler to alter the learning rate during training. It starts reducing the learning rate slowly, followed by a quick reduction halfway through the training process ending with a slight reduction



- YOLOv4 Architecture for object detection. The modules in the diagram are CMB: Convolution + Batch Normalization + Mish activation, CBL: Convolution + Batch Normalization + Leaky ReLU, UP: upsampling, SPP: Spatial Pyramid Pooling, and PANet: Path Aggregation Network. Diagram inspired by [71]. Lists the final selection of BoFs and BoS for the backbone and the detector. Evaluated on MS COCO dataset test-dev 2017, YOLOv4 achieved an AP of 43.5% and AP50 of 65.7% at more than 50 FPS on an NVIDIA V100

6. YOLOv7

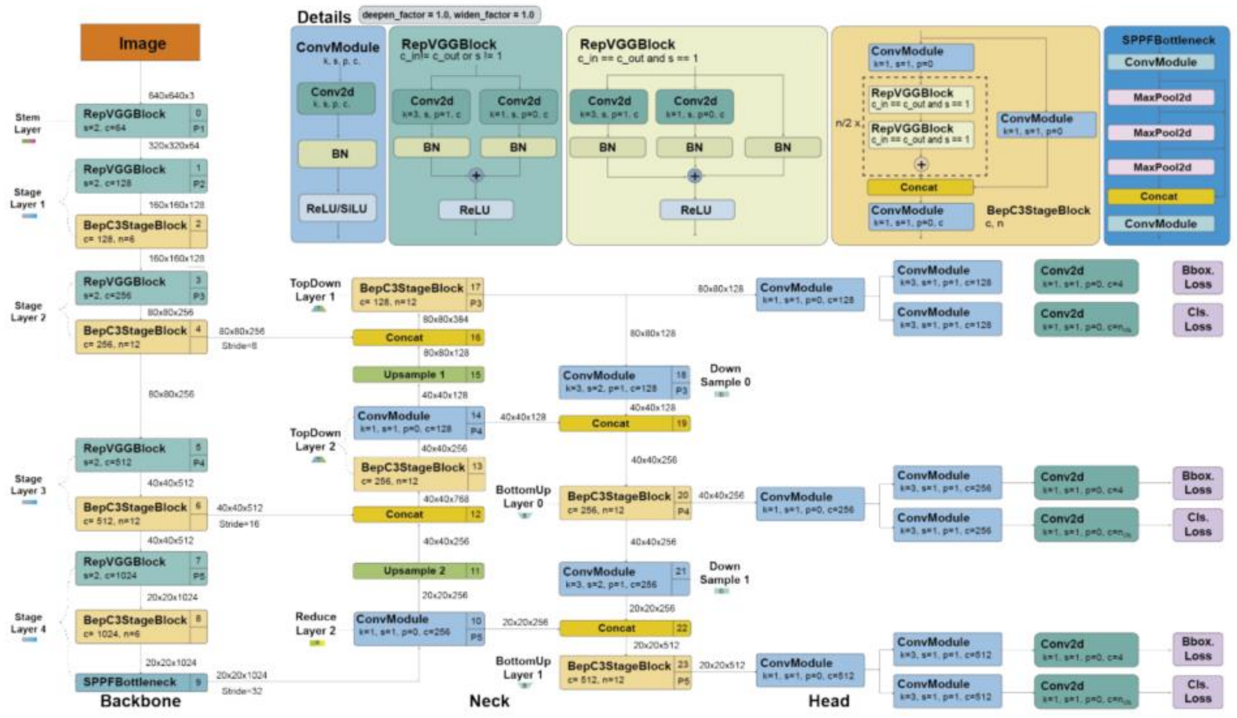
YOLOv7 was published in ArXiv in July 2022 by the same authors of YOLOv4 and YOLOR. At the time, it surpassed all known object detectors in speed and accuracy in the range of 5 FPS to 160 FPS. Like YOLOv4, it was trained using only the MS COCO dataset without pre-trained backbones. YOLOv7 proposed a couple of architecture changes and a series of bag-of-freebies, which increased the accuracy without affecting the inference speed, only the training time. Picture below shows the detailed architecture of YOLOv7.
The architecture chages of YOLOv7 are:

- **Extended efficient layer aggregation network (E-ELAN)**. ELAN [102] is a strategy that allows a deep model to learn and converge more efficiently by controlling the shortest longest gradient path. YOLOv7 proposed E-ELAN that works for models with unlimited stacked computational blocks. E-ELAN combines the features of different groups by shuffling and merging cardinality to enhance the network's learning without
destroying the original gradient path.

- **Model scaling for concatenation-based models**. Scaling generates models of different sizes by adjusting some model attributes. The architecture of YOLOv7 is a concatenation-based architecture in which standard scaling techniques, such as depth scaling, cause a ratio change between the input channel and the output channel of a transition layer which, in turn, leads to a decrease in the hardware usage of the model. YOLOv7 proposed a new strategy for scaling concatenation-based models in which the depth and width of the block are scaled with the same factor to maintain the optimal structure of the model.

The bag-of-freebies used in YOLOv7 include:

- **Planned re-parameterized convolution**. Like YOLOv6, the architecture of YOLOv7 is also inspired by parameterized convolutions (RepConv). However, they found that the identity connection in RepConv



YOLOv6 Architecture. The architecture uses a new backbone with RepVGG blocks. The spatialpyramid pooling fast (SPPF) and Conv Modules are similar to YOLOv5. However, YOLOv6 uses a decoupled head. Diagram based in destroys the residual in ResNet and the concatenation in DenseNet. For this reason, they removed the identity connection and called it RepConvN

1. Coarse label assignment for auxiliary head and fine label assignment for the lead head. The lead head is responsible for the final output, while the auxiliary head assists with the training.

2. Batch normalization in conv-bn-activation. This integrates the mean and variance of batch normalization into the bias and weight of the convolutional layer at the inference stage.

3. **Implicit knowledge** inspired in YOLOR

4. Exponential moving average as the final inference model

7. Comparison with YOLOv4 and YOLOR

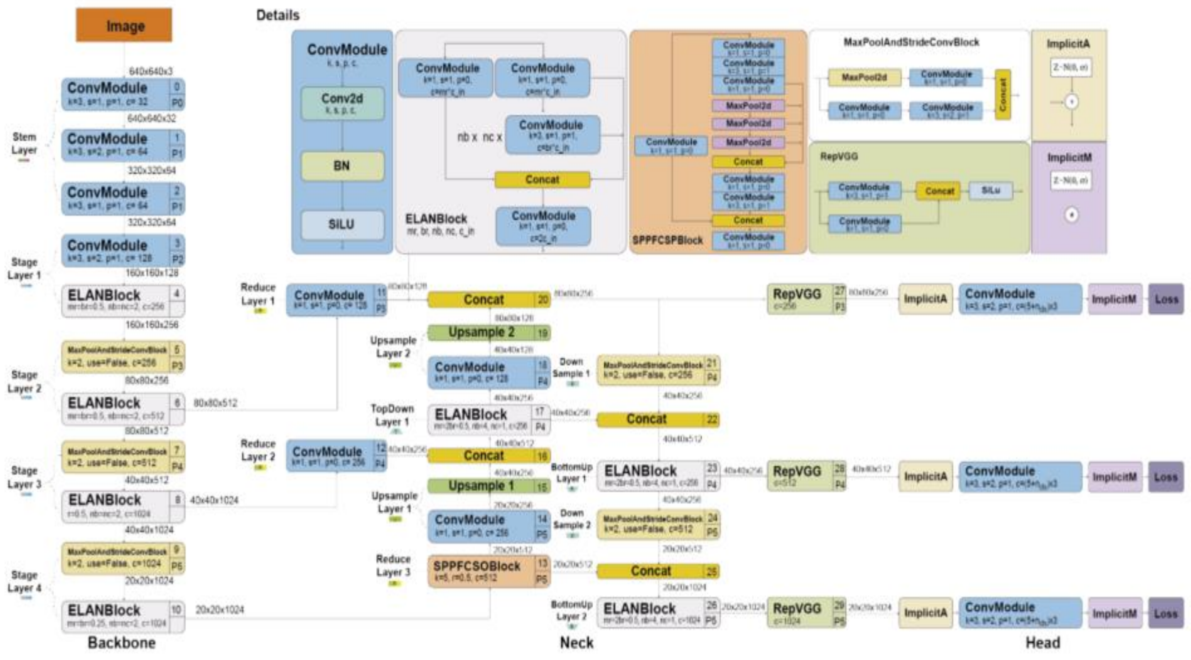
In this section, we highlight the enhancements of YOLOv7 compared to previous YOLO models developed by the same authors.

Compared to YOLOv4, YOLOv7 achieved a 75% reduction in parameters and a 36% reduction in computation while simultaneously improving the average precision (AP) by 1.5%.

In contrast to YOLOv4-tiny, YOLOv7-tiny managed to reduce parameters and computation by 39% and 49%, respectively, while maintaining the same AP.

Lastly, compared to YOLOR, YOLOv7 reduced the number of parameters and computation by 43% and 15%, respectively, along with a slight 0.4% increase in AP.

Evaluated on MS COCO dataset test-dev 2017, YOLOv7-E6 achieved an AP of 55.9% and AP50 of 73.5% with an input size of 1280 pixels with a speed of 50 FPS on an NVIDIA V100.

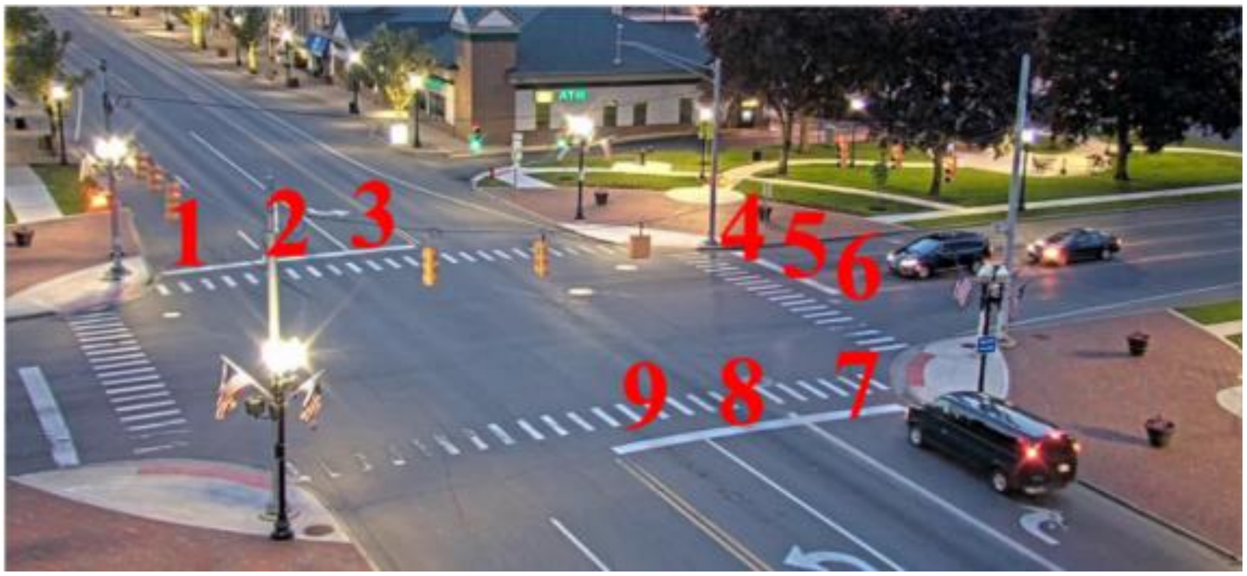


➔ YOLOv7 Architecture. Changes in this architecture include the ELAN blocks that combine features of different groups by shuffling and merging cardinality to enhance the model learning and modified RepVGG without identity connection.

V. Evaluation

Results *3 task in evaluation train*

- Task1: Consists in correctly classifying the road lanes listed in the query file as being occupied (label 1) or not (label 0). The lanes are numbered:
  from 1 to 3 for the upper left part of the intersection
  4 to 6 for the right part
  and 7 to 9 for the bottom right part of the intersection

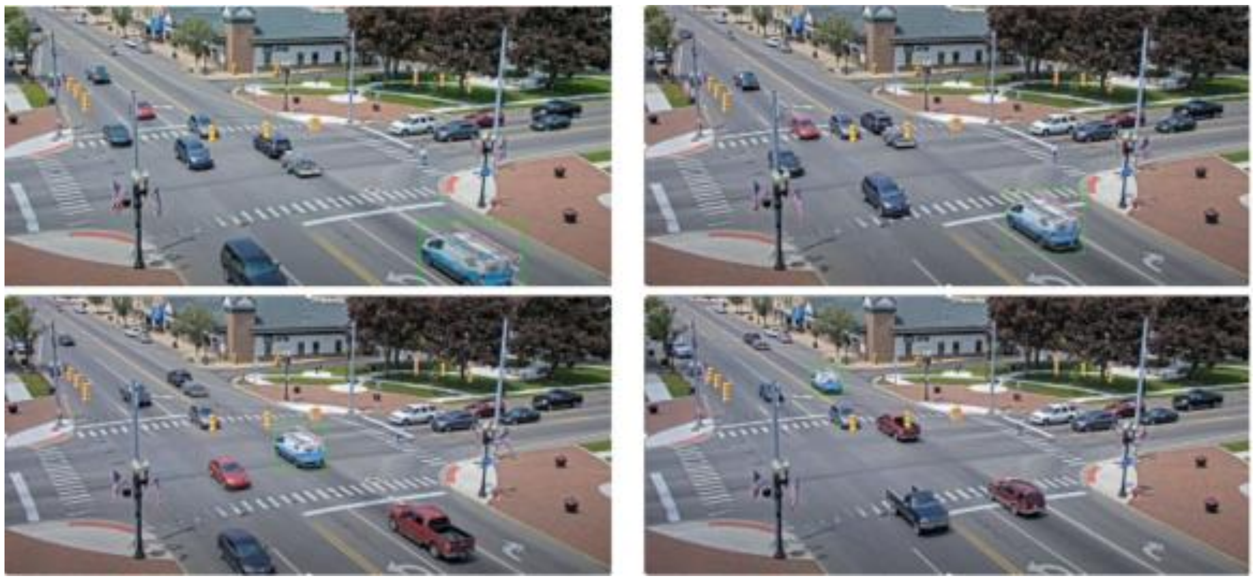


Classifying road lanes: for test example number 15 image:				
01-1 predicted	01-2 predicted	01-3 predicted	02-1 predicted	02-2 predicted
4 1 0 3 1 7 0 9 0	3 3 1 8 1 9 1	3 6 1 7 1 9 1	3 2 1 4 0 9 1	3 2 1 6 1 9 1

Train: Mistakes: 12; Total preds: 154; Score: 0.922077922077922

➔ So after training with the above set of results, it shows that the lane classification part of the algorithm reaches 92% of the lane classification level from the task we need to do in task1 . During the period from T0 to T1

- Task 2:
  - don't understand the training results
  - evaluation train task2 and ground_truth Task2
  - The task of task2 is to track a car with a blue frame in the video numbered = 1 "upper part of intersection", = 2 right side of intersection, = 3 lower part of intersection.



tp = 519, tn = 50, fp = 109,fn = 0
percentage = 0.8392330383480826
Task 2 - Tracking the vehicle: for test example number 11 the prediction is : correct

tp = 318, tn = 0, fp = 143,fn = 0
percentage = 0.6898047722342733
Task 2 - Tracking the vehicle: for test example number 12 the prediction is : incorrect

tp = 454, tn = 0, fp = 57,fn = 0
percentage = 0.8884540117416829
Task 2 - Tracking the vehicle: for test example number 13 the prediction is : correct

tp = 235, tn = 0, fp = 174,fn = 0
percentage = 0.5745721271393643
Task 2 - Tracking the vehicle: for test example number 14 the prediction is : incorrect

tp = 344, tn = 0, fp = 148,fn = 0
percentage = 0.6991869918699187
Task 2 - Tracking the vehicle: for test example number 15 the prediction is : incorrect

⇨ The requirement of task2 is that 50% of the videos track the vehicle correct and achieve a speed of [0.83 - 0.99] which is the percentage that achieves the correct tracking value of the vehicle, the rest are not tracked correctly. that car during the period from T2 to T3.

- Task 3 The goal here is to count the trajectories of vehicles in the given video. We are interest to count the number of vehicles between the three regions of interest. The ground-truth files specifies the number of vehicles that go from region 1 to region 2, region 1 to region 3, region 2 to region 3.

01_predicted.txt	02_predicted.txt	03_predicted.txt	04_predicted.txt	06_predicted.txt
1-2 0	1-2 0	1-2 0	1-2 0	1-2 0
1-3 0	1-3 0	1-3 1	1-3 6	1-3 1
2-1 0	2-1 0	2-1 0	2-1 0	2-1 1
2-3 3	2-3 0	2-3 1	2-3 0	2-3 1
3-1 0	3-1 0	3-1 5	3-1 7	3-1 1
3-2 0	3-2 0	3-2 2	3-2 0	3-2 1

- Evaluation train task3 and ground_truth Task3.

Task 3 - predicting trajectories: for test example number 1 the prediction is : correct

Task 3 - predicting trajectories: for test example number 2 the prediction is : incorrect

```
Task 3 - predicting trajectories: for test example number  3  the prediction is : correct

Task 3 - predicting trajectories: for test example number  4  the prediction is : correct

Error
Task 3 - predicting trajectories: for test example number  5  the prediction is :
incorrect

Task 3 - predicting trajectories: for test example number  6  the prediction is : correct

Task 3 - predicting trajectories: for test example number  7  the prediction is :
incorrect
Task 3 =  0.7000000000000001
```

➔ Thus, Task3 results in correctly counting 70% of the predicted results of counting the vehicle's trajectory in the video. In which, the number of vehicles in each region is accurately predicted. Shows the effectiveness of using the YoLO 7 algorithm applied to the model in each given video.

➔ Total score of each task's results on actual data train:

Task 1 = 92%   Task 2 = 50%   Task 3 = 70%.