Small Object Detection For Autonomous Vehicles

Team Members :

Abhijit rao.S SE21UARI003 N.Gagan SE21UARI094

Varshini.V SE21UARI183 Sri Ashritha SE21UARI153

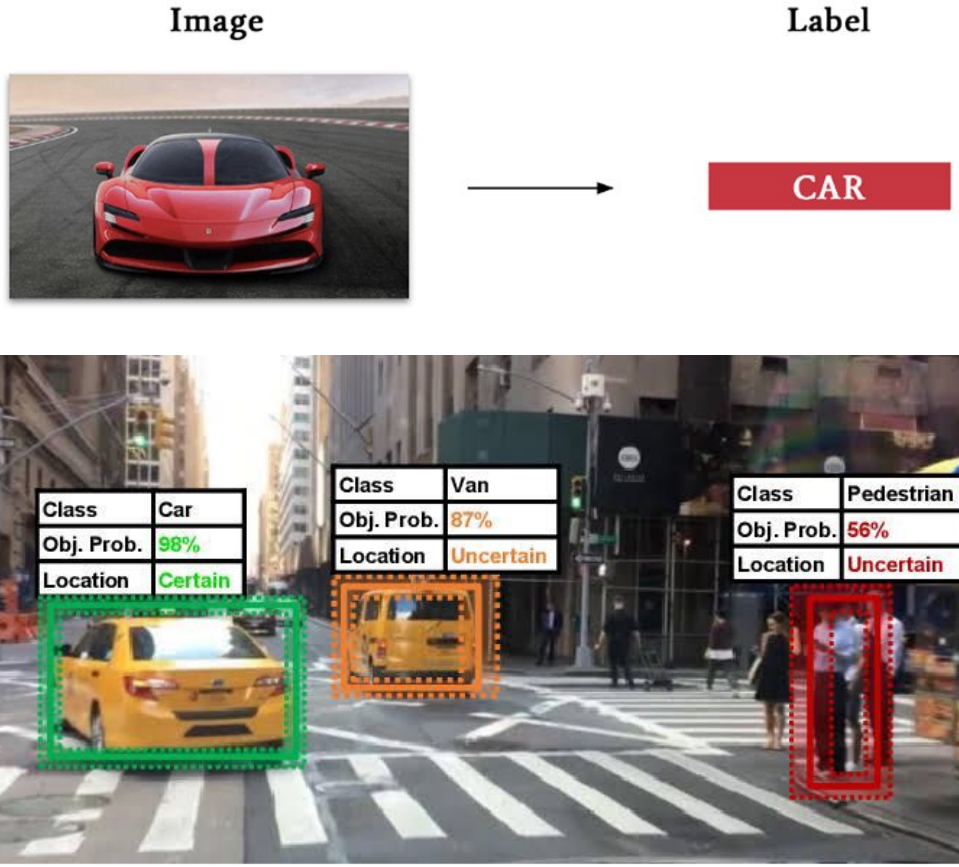
Srinivas Gurram SE21UARI190

BDD 100K Dataset

INTRODUCTION TO BDD 100K

- **Dataset Name:** Berkeley Deep Drive 100K (BDD100K)
- **Description:** BDD100K is a diverse, annotated dataset designed for autonomous driving research.
- **Applications:** This Dataset is used in Deep learning for Object Detection In Autonomous Vehicles. Here By using the Dataset we compare the 3 different architectures: #YOLOV5, #RCNN, #SSD.
- **Capturing:** Contains 100,000 HD videos, totaling over 1,100 hours of driving footage captured in various conditions and every 10th sec of the video is captured as an image.
- **Image Content:** Provides 100,000 images with labels for segmentation, detection, tracking, and lane lines.
- **Image Specifications:** Each image is a 1280x720 RGB image.
- **Availability**: The dataset, including images and annotations, is available for free.

IMAGE > LABEL ASSOCIATION



HOW IS THE DATASET COLLECTED

- Video footage collected via Nexar dashcams.
- Resolution: 720p; Frame rate: 30 fps.
- Each video clip sampled at the 10th second.
- Frame images provided.
- Images in RGB format.
- Dimensions of images: 1280 x 720 pixels.
- Dataset includes localization, timestamp, and Inertial Measurement Unit (IMU) data.

Data is collected from 3 different locations

SF, Berkeley, Bay Area & New York

Division of the Dataset

10 object categories:

1. bike
2. bus
3. car
4. motor
5. person
6. rider
7. traffic light
8. traffic sign
9. train
10. truck

Instance segmentation categories

1. banner
2. billboard
3. lane divider
4. parking sign
5. pole
6. polegroup
7. street light
8. traffic cone
9. traffic device
10. traffic light
11. traffic sign
12. sign frame
13. person
14. rider
15. bicycle
16. bus
17. car
18. caravan
19. motorcycle
20. trailer
21. train
22. truck

Lane marking categories:

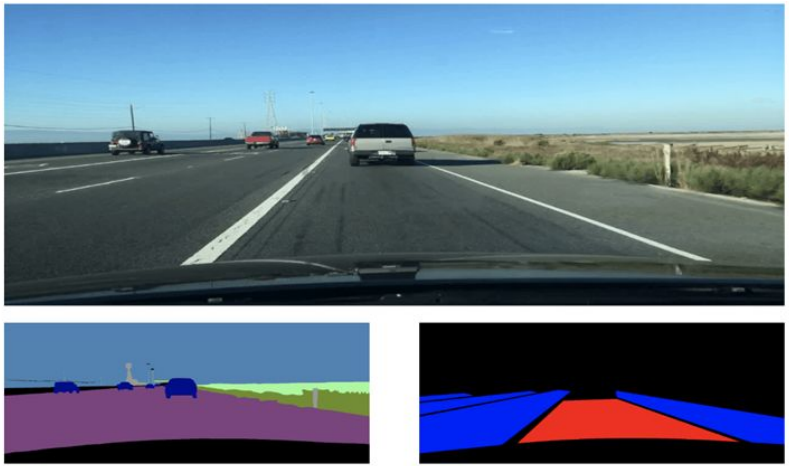
1. lane/crosswalk
2. lane/double other
3. lane/double white
4. lane/double yellow
5. lane/road curb
6. lane/single other
7. lane/single white
8. lane/single yellow

Annotations

- Bounding boxes for common objects found on the road like traffic lights and signs, buses, bikes etc.
- Two types of lane markings- vertical, which are along the driving direction of the lanes and parallel, which indicate where vehicles in the lane must stop.
- Drivable areas,

  1. Direct Drivable: Indicates that the car has priority on the indicated road.

  2. Alternative Drivable: Indicates that the car can proceed but should be cautious as other cars may have priority.



Here, the type of segmentation is full-frame segmentation. Thus you can see the road in purple, the sky in blue, the cars in marine, ...

Finally, Driveable Maps represent the lanes.

Red pixels represent the ego lane, where it's okay to drive. Blue pixels represent adjacent lanes.

How to use?

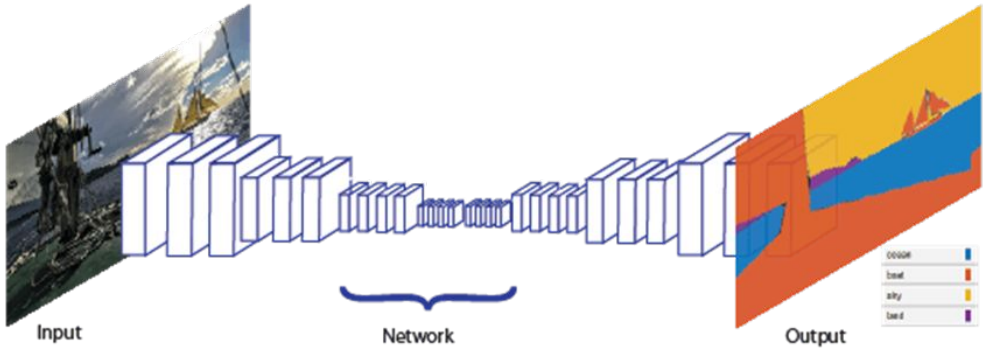
To use this type of data, we will need semantic segmentation algorithms.

This is generally done by an encoder/decoder architecture.

A JSON file is a file where you can access information in a specific format.

As Segmentation doesn't provide all the information we need we can use Bounding Boxes of each object to get more clarity

Then we use Algorithms such as YOLO v5, RCNN, SSD,... etc For Object Detection



OVERVIEW

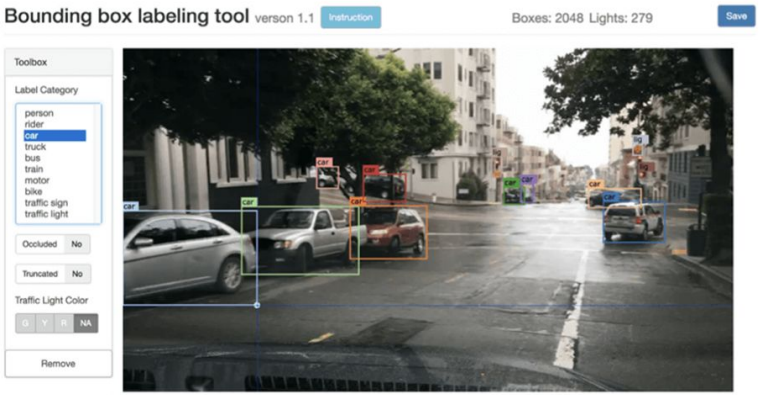
1. ***Scale and Diversity:*** 100,000 videos with varied geographic, environmental, and weather conditions.

2. ***Multitask Learning:*** Supports lane detection, drivable area segmentation, object detection, and multi-object tracking.

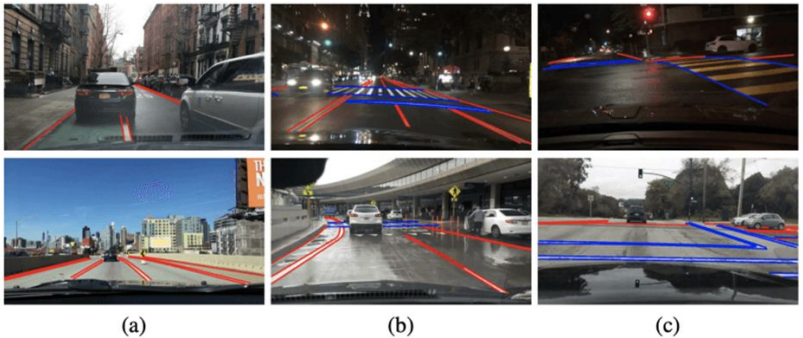
3. ***Data Quality and Annotation:*** Detailed annotations for lane markings, drivable areas, and road objects.

4. ***Research Impact:*** Advances autonomous driving with a standardized benchmark for diverse conditions.

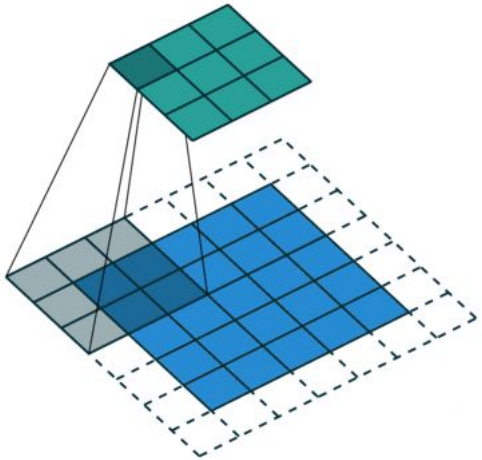
BOUNDING BOX LABELING



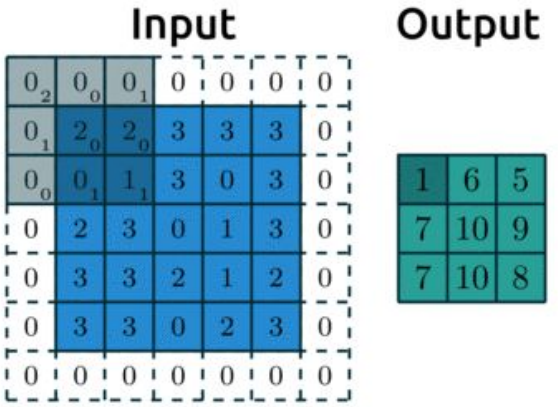
LINE MAKING



Understanding the Basics of Object Detection



Convolution

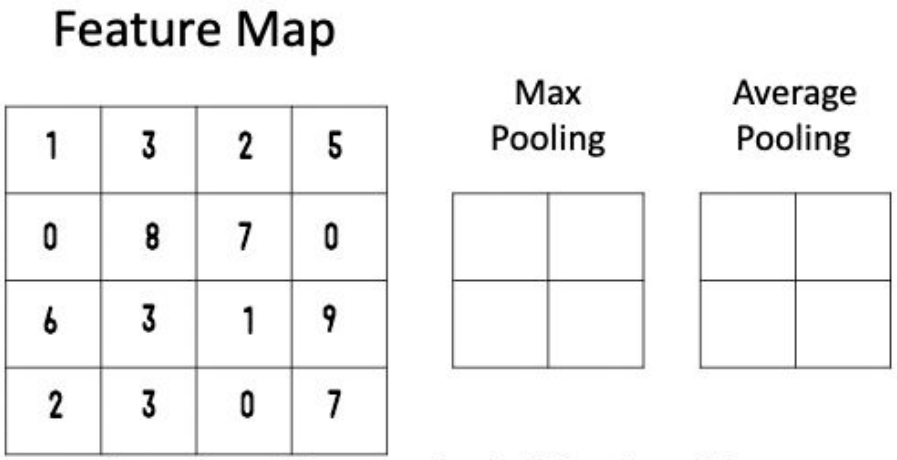


Padding - 1
Image size - 5x5
Kernel size - 3x3
Stride - 1

→ Output image : **3x3**

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Understanding the Basics of Object Detection



Padding - 0
Image size - 4x4
Kernel size - 2x2
Stride - 2

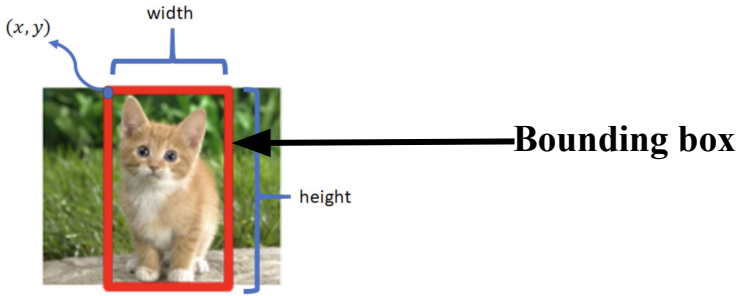
→ Output image : **2x2**

$$W_{out} = \frac{W - F}{S} + 1$$

Understanding the Basics of Object Detection

- **Segmentation :** it is the task of grouping pixels with comparable properties together instead of bounding boxes to identify objects.

- **Object localization:**Object localization seeks to identify the location of one or more objects in an image.



Understanding the Basics of Object Detection

- **Activation Function :** Activation functions determine whether or not a neuron should be activated based on its input to the network. These functions use mathematical operations to decide if the input is important for prediction. If an input is deemed important, the function “activates" the neuron.

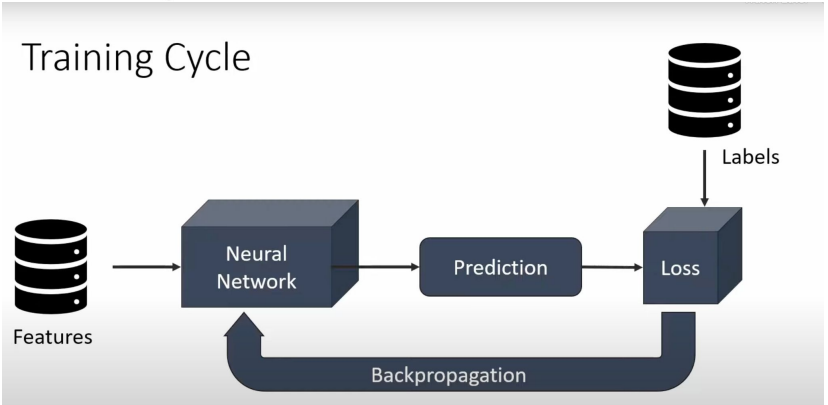
Tanh activation function

$$tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$$

ReLu activation function

$$R(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

- **Iterations:** the number of batches needed to complete one Epoch.
- **Batch Size**: The number of training samples used in one iteration.
- **Epoch:** One full cycle through the training dataset. A cycle is composed of many iterations.

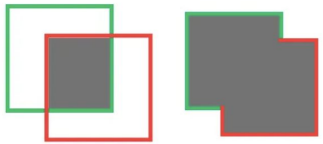


Evaluating Object Detection Models

Preparing the training data → Configuring the model → Training the model → Evaluating the model

1. **Intersection over union (IoU) :** The IoU is calculated by taking the intersection area of two bounding boxes and dividing it by the union area of these boxes.

$$IoU = \frac{(\text{Area of Intersection})}{(\text{Area of Union})} = \frac{|A \cap B|}{|A \cup B|}$$

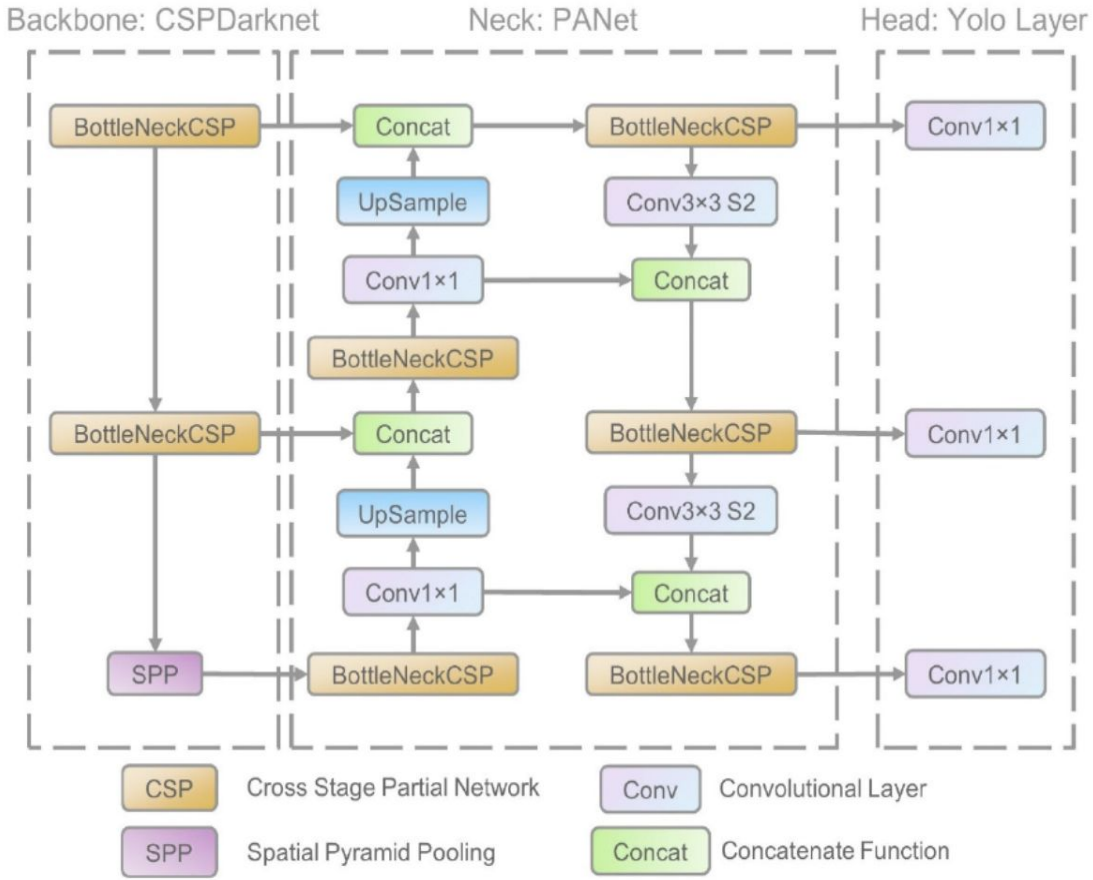


2. **Mean Average Precision (mAP) :** mAP considers both Precision (P) and Recall (R). Average Precision is the average of precision at different recall values obtained by Precision Recall (PR) curve under a given threshold.

$$Precision\ (P) = \frac{TP}{TP + FP} \quad Recall\ (R) = \frac{TP}{TP + FN} \quad mean\ Average\ Precision\ (mAP) = \frac{1}{n}\sum_{i=1}^{n} AP_i$$

$$Average\ Precision\ (AP) = \sum_{k} (R_{k+1} - R_k) \max_{R:\bar{R} \geq R_{k+1}} P(\bar{R})$$

The Network Architecture of Yolo v5.

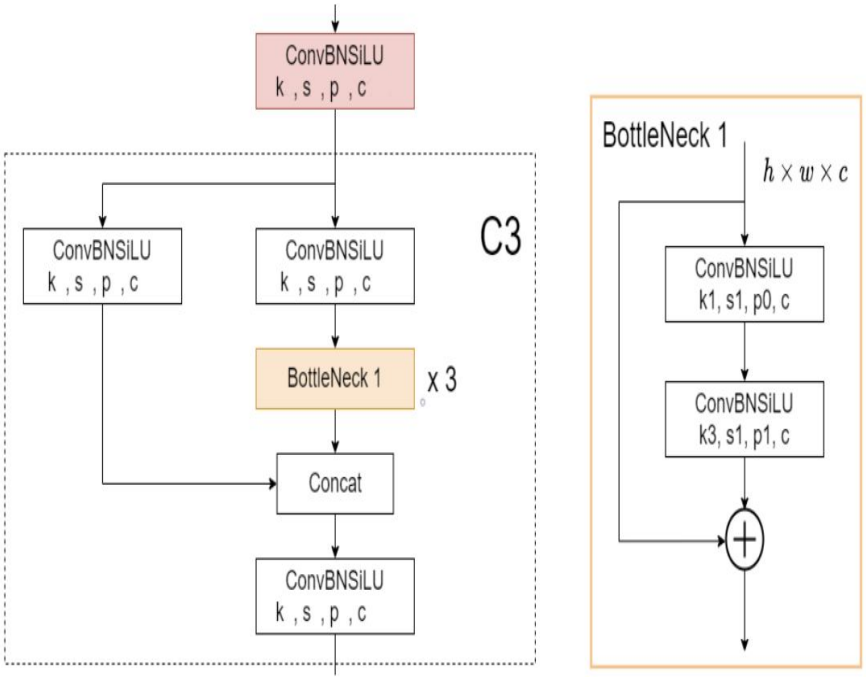


It consists of three parts:
(1) Backbone: CSPDarknet,
(2) Neck :SPP & PANet
(3) Head :Yolo Layer.

The data are first input to CSPDarknet for feature extraction,
Then fed to SPP andPANet for feature fusion.
Finally, Yolo Layer outputs detection results (class, score, location, size).

CSP-Darknet53

YOLOv5 uses CSP-Darknet53 as its backbone.



BottleNeckCSP module architecture.

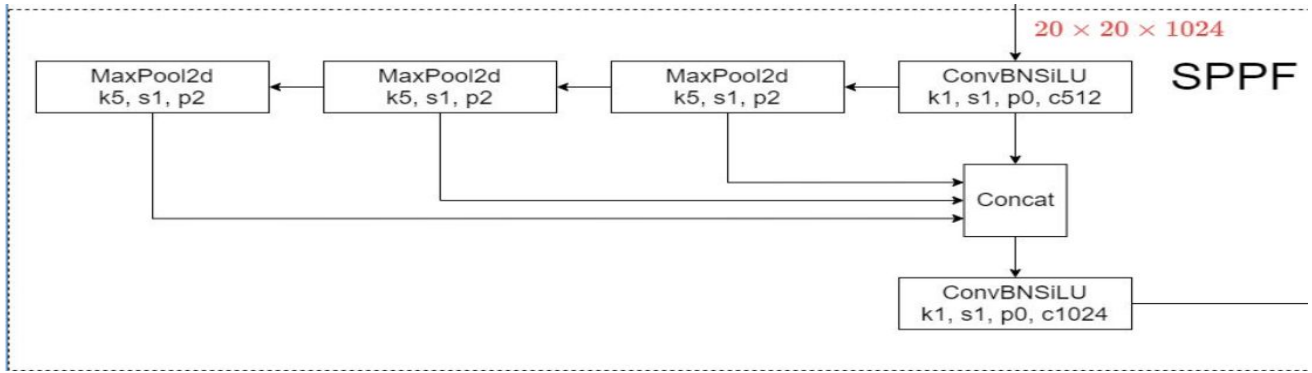
It helps reducing the number of parameters and helps reducing an important amount of computation (less FLOPS) which lead to **increasing the inference speed** that is crucial parameter in real-time object detection models.YOLOv5 employs CSPNet strategy to partition the feature map of the base layer into two parts and then merges them through a cross-stage hierarchy.

Neck of YOLOv5

YOLOv5 brought two major changes to the model neck. First a variant of **Spatial Pyramid Pooling** (SPP) has been used, and the **Path Aggregation Network** (PANet) has been modified by incorporating the **BottleNeckCSP** in its architecture.

Path Aggregation Network (PANet)PANet is a feature pyramid network, it has been used in previous version of YOLO (YOLOv4) to improve information flow and to help in the proper localization of pixels in the task of mask prediction.

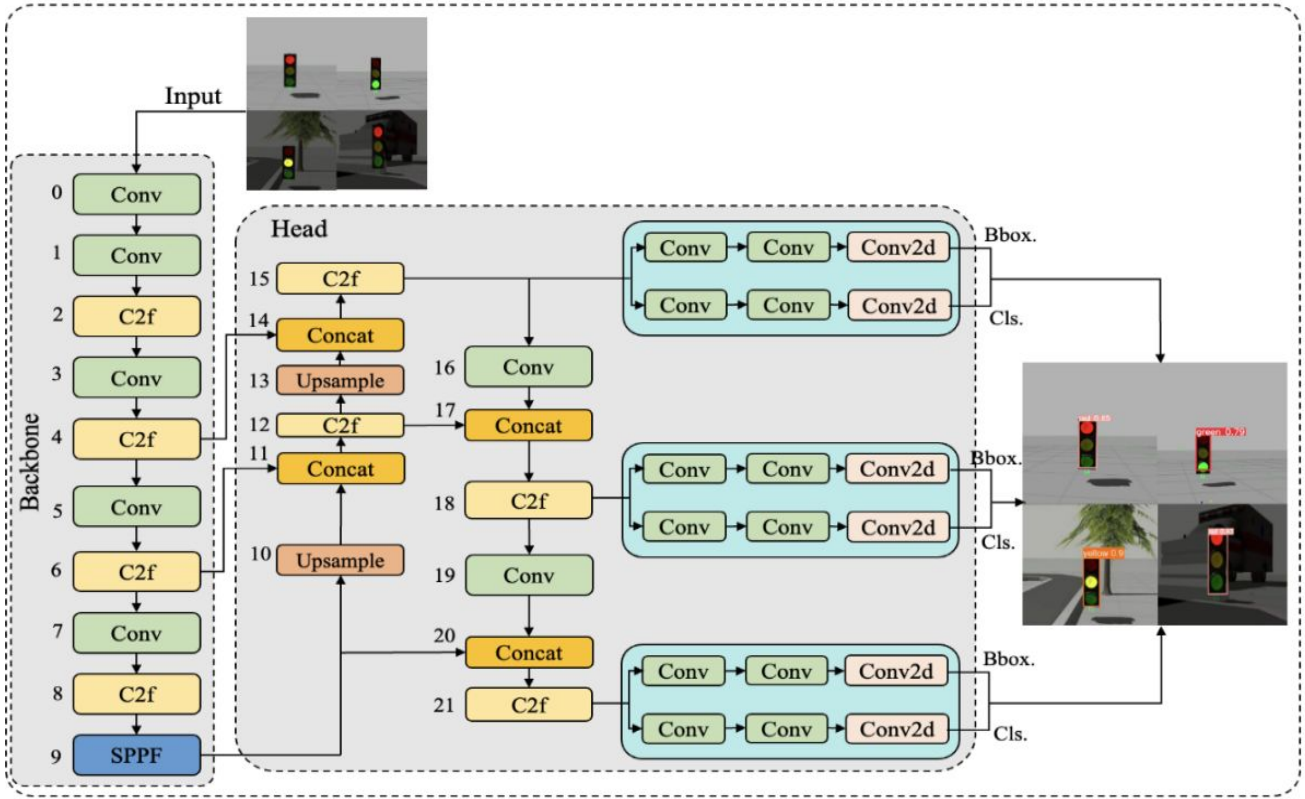
Spatial Pyramid Pooling (SPP) This block has been used in previous versions of YOLO (yolov3 and yolov4) to separate the most important features from the backbone, however in YOLOv5(6.0/6.1) **SPPF** has been used , which is just another variant of the SPP block, to **improve the speed of the network**



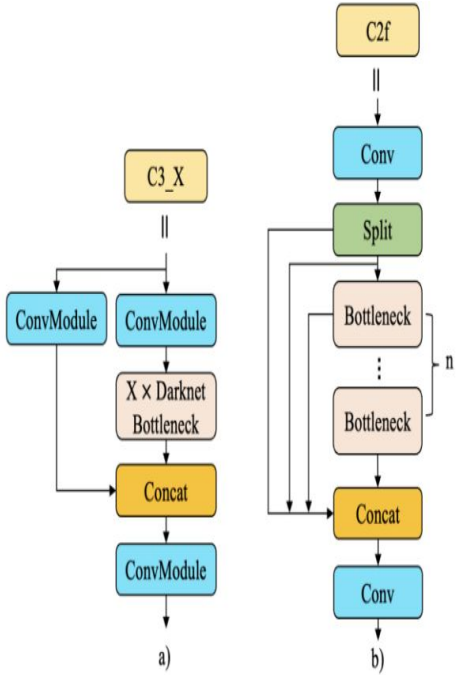
Head of the network

YOLO v5 uses the same head as YOLO v3 and YOLO v4. It is composed from three convolution layers that predicts the location of the bounding boxes (x,y,height,width), the scores and the objects classes.

YOLO V8 model Architecture



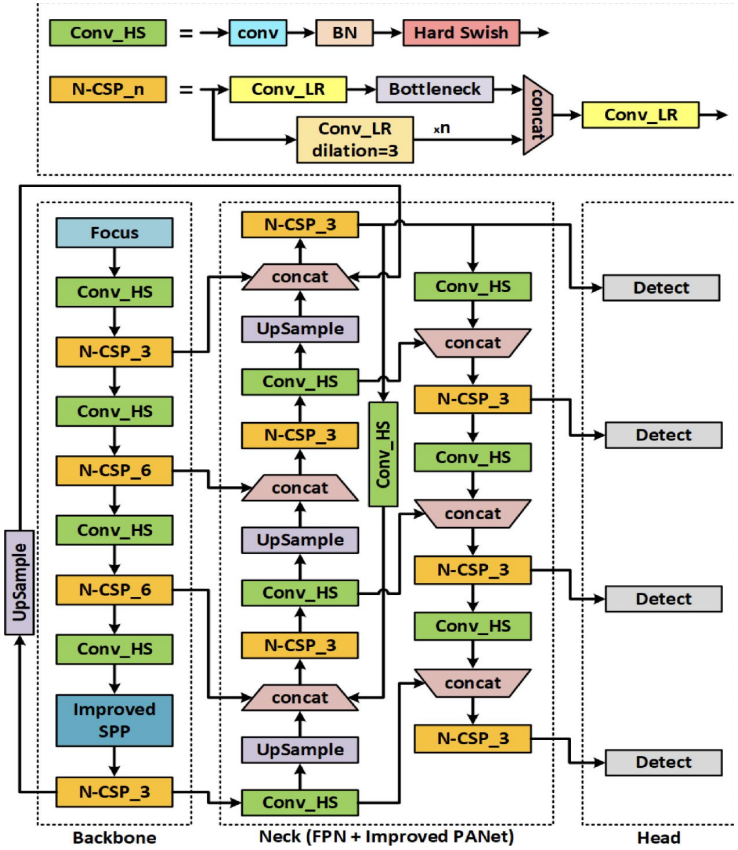
Comparison of the structure of CS_X (YOLO v5) and C2f (YOLO v8) in backbone.



The model YOLO v8 for detecting, and classification traffic lights.

Data Augmentation Techniques

YOLO v5 employs various data augmentation techniques to improve the model's ability to generalize and reduce overfitting.

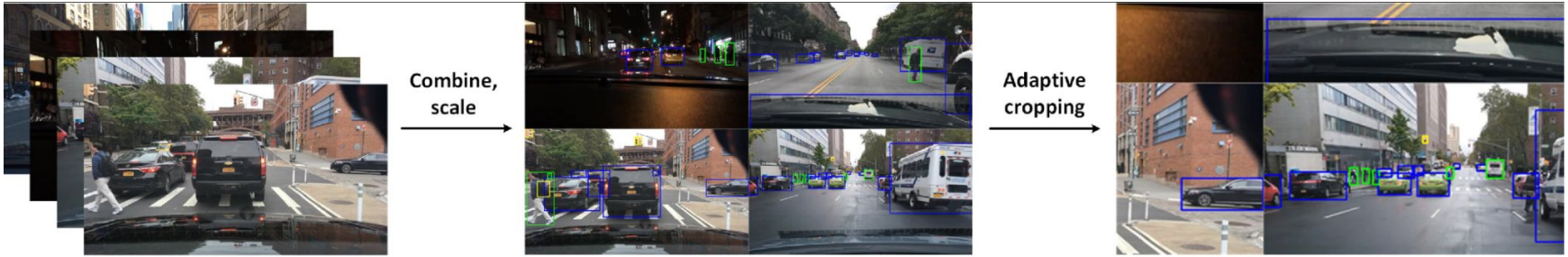


Mosaic Augmentation: An image processing technique which allows us to train four images instead of one image.

Copy-Paste Augmentation: An innovative data augmentation method that copies random patches from an image and pastes them onto another randomly chosen image generating new sample.

Random Horizontal Flip: An augmentation method that randomly flips images horizontally.

Implementation of Mosaic data augmentation

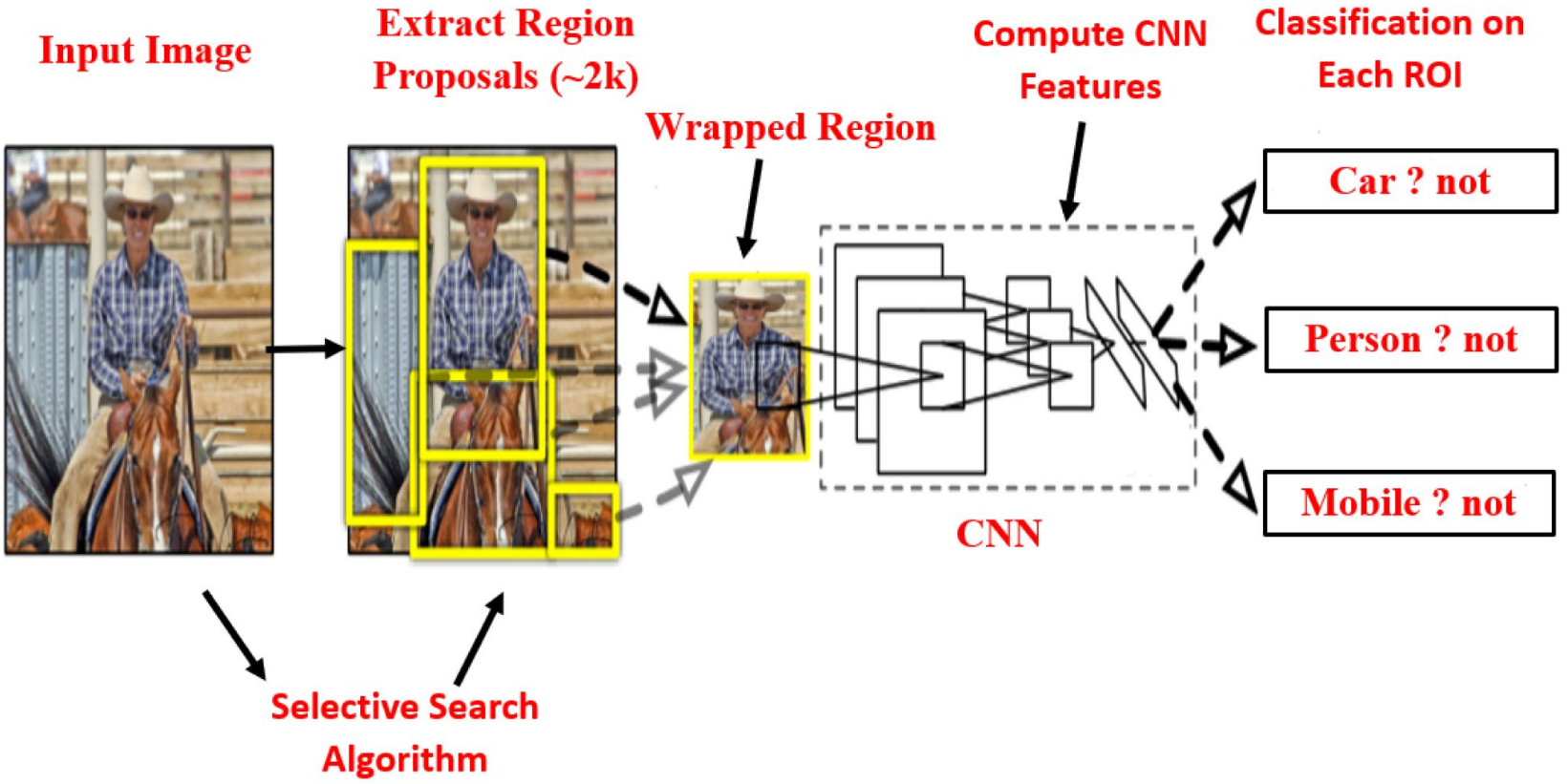


YOLO v5's results and discussion affirm its efficacy for real-time object detection. The model's
high accuracy, real-time capabilities, and robustness in challenging scenarios make it a compelling choice for
improving the perception and safety of autonomous vehicles

when the traffic density increases (top to bottom), the prediction confidence of YOLOv5 decreases and starts to miss targets. In contrast, the proposed iS-YOLOv5 model detects traffic signs and traffic lights with high confidence, even in high traffic scenarios.

RCNN (Region based Convolutional Neural Network)

RCNN Architecture



Step 1: Region proposals

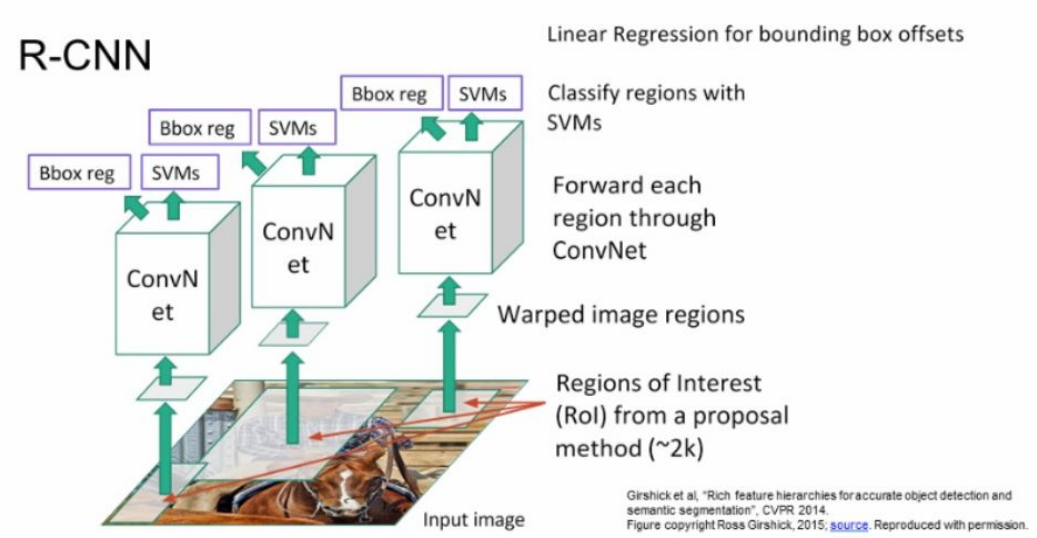
- The image is passed through a selective search algorithm, which identifies potential regions that might contain objects.
- This step generates a large number of region proposals (around 2,000) – potentially with many redundant or irrelevant ones.

Step 2: Wrapping the proposed regions

- Each proposed region is extracted from the original image and resized to a standard size.
- Without performing this step we can't achieve accurate results in RCNN because it requires same sized inputs.

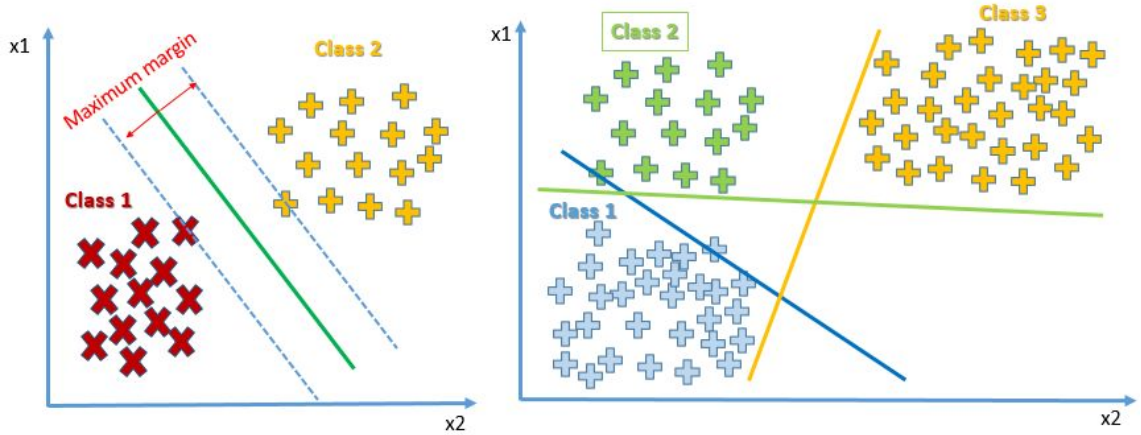
Step 3 : Converting warped images into feature vectors

- These warped images are now converted into feature vectors when passed through a specific trained CNN network
- This step is used so that the complexity of the data will be reduced and there are lesser elements to process than a whole image.



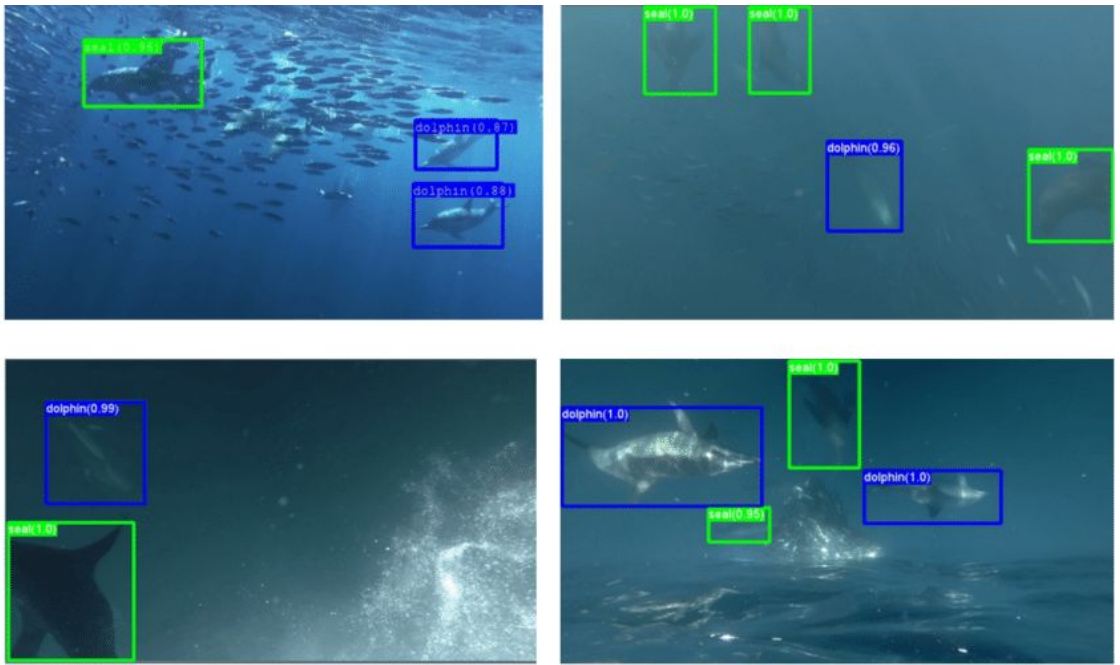
Step 4 : Segregation of feature vectors using SVM

- SVM is nothing but support vector machine which uses hyperplanes to distinguish any objects or classes.
- SVM is a supervised machine learning algorithm unlike neural network in our previous step.
- When our feature vectors are passed through SVMs they get classified like shown below



Final result

After SVM giving results the proposed regions are classified as their respective objects.



Pros

- **Accuracy**: R-CNN excels at precise object localization and recognition.
- **Robustness**: R-CNN models can handle variations in objects, including different sizes, orientations, and scales.

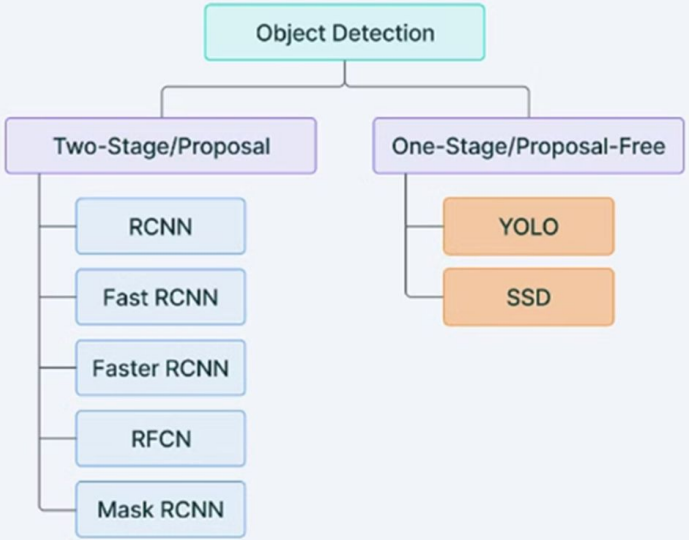
Cons

- **Computational Cost**: R-CNN is computationally expensive. It requires processing image regions multiple times, which can be resource-intensive.
- **Slow Inference**: Due to the sequential processing of region proposals, R-CNN is relatively slow, making it unsuitable for real-time applications.
- **Redundancy**: R-CNN may generate overlapping region proposals, leading to unnecessary computations and potentially impacting detection performance.

Single Shot Detector (SSD)

Single shot detectors (SSDs) are a class of object detection models that can identify and locate multiple objects in an image with a single forward pass through a neural network. This efficient approach makes SSDs a popular choice for real-time applications

One and two stage detectors



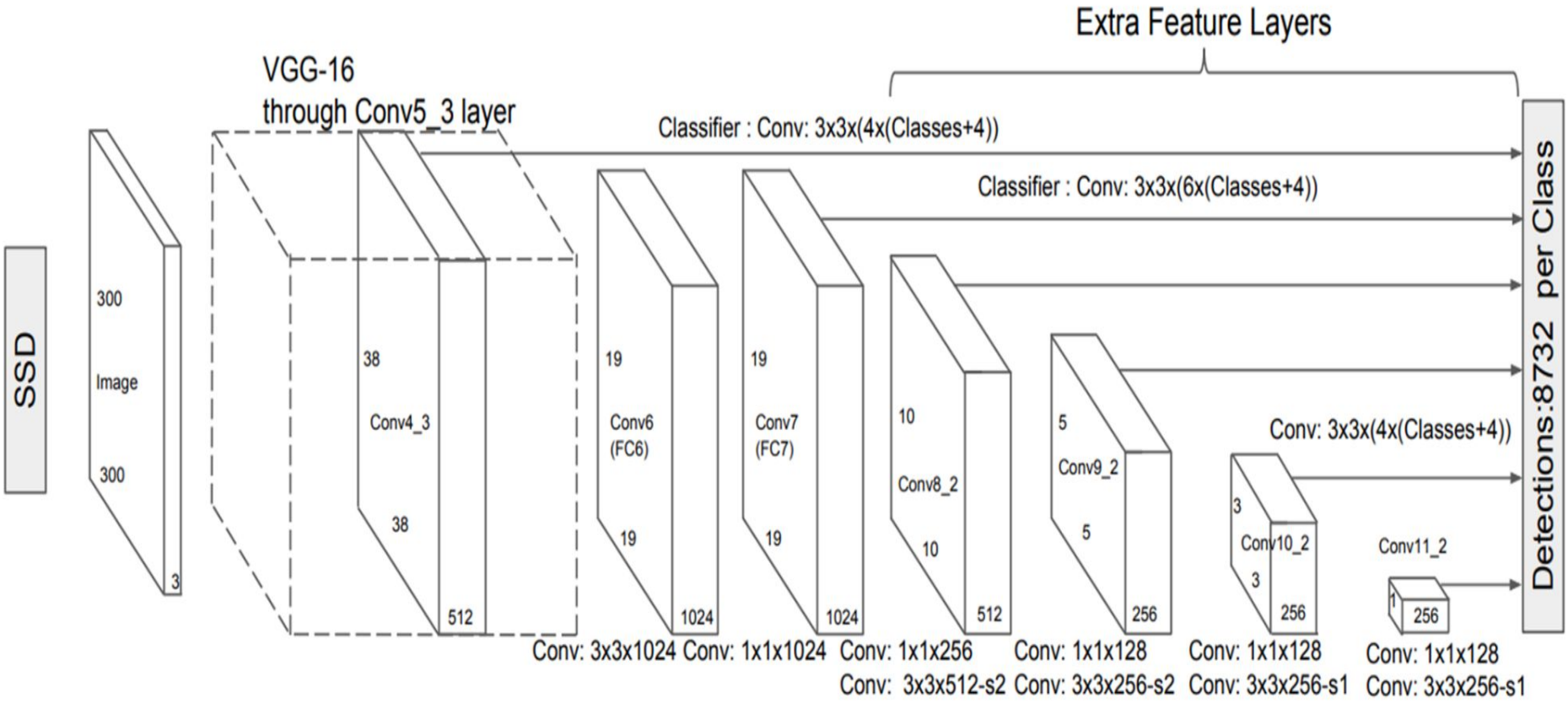
V7 Labs

Advantages of Single Shot Detectors

1. High efficiency

2. Accurate predictions

3. Scalable Architectures

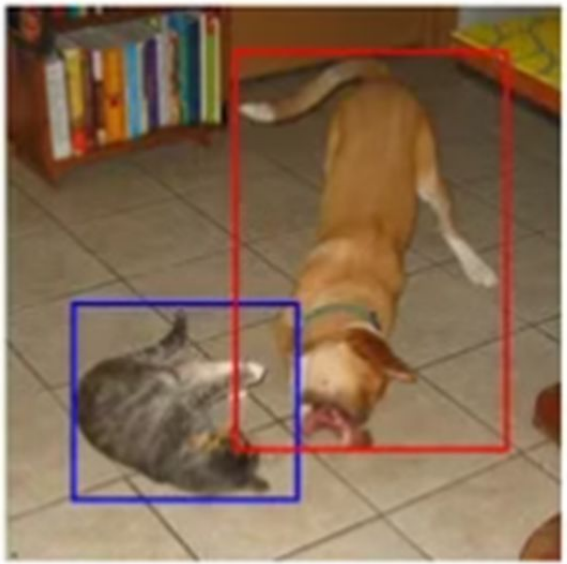

SSD
300
Image
300
3
VGG-16
through Conv5_3 layer
38
Conv4_3
38
512
19
Conv6
(FC6)
19
1024
19
Conv7
(FC7)
19
1024
10
Conv8_2
10
512
5
Conv9_2
5
256
3
Conv10_2
3
256
Conv11_2
1
256
Extra Feature Layers
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
Conv: 3x3x(4x(Classes+4))
Detections:8732 per Class
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256
Conv: 3x3x512-s2
Conv: 1x1x128
Conv: 3x3x256-s2
Conv: 1x1x128
Conv: 3x3x256-s1
Conv: 1x1x128
Conv: 3x3x256-s1

Architecture of Single Shot Detector

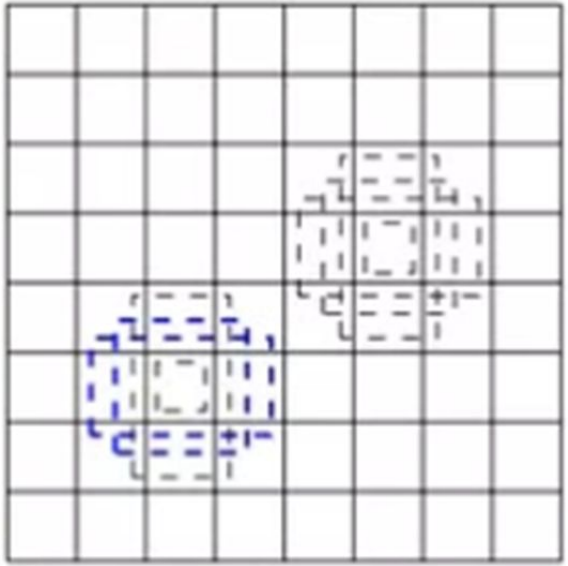
1. Feature extractor

2. Feature pyramid

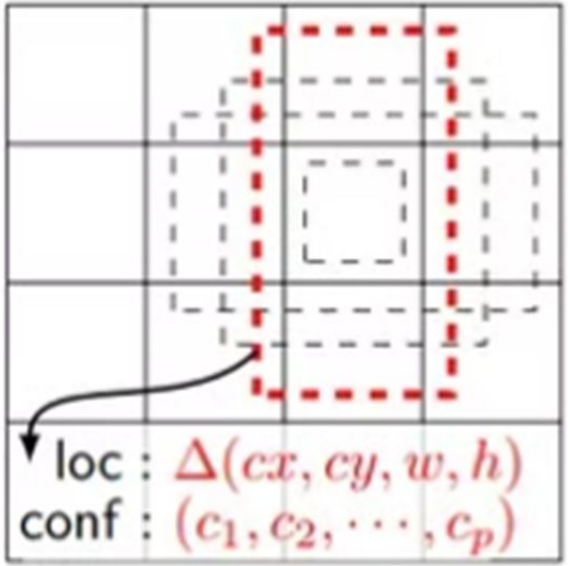
3. Prediction Head



(a) Image with GT boxes



(b) 8×8 feature map



(c) 4×4 feature map


HUMAN
HUMAN
HUMAN
HUMAN
HUMAN
HUMAN
HUMAN
HUMAN
Pushcart
Bag
Bag
Bicycle

Conclusion and Future direction

1. Improved efficiency

2. Novel architecture

3. Real world Deployment

COMPARISON OF ALL THE ARCHITECTURES

Model	Small Object Detection Capability	Speed	Accuracy	Flexibility	Implementation
YOLO v5	Effective	Fast	High	Moderate	Single-stage, end-to-end
RCNN	Moderate	Moderate	Very High	High	Multi-stage, region-based
SSD	Moderate	Fast	Moderate	Moderate	Single-stage, end-to-end

Review