Advanced Deep Learning

CSA401

Single Shot MultiBox Object Detection (SSD)



Learning Outcomes

- Architecture of SSD
- Feature extraction backbones
- Working of SSD
- Applications

https://arxiv.org/pdf/1512.02325

Objection Detection Milestones

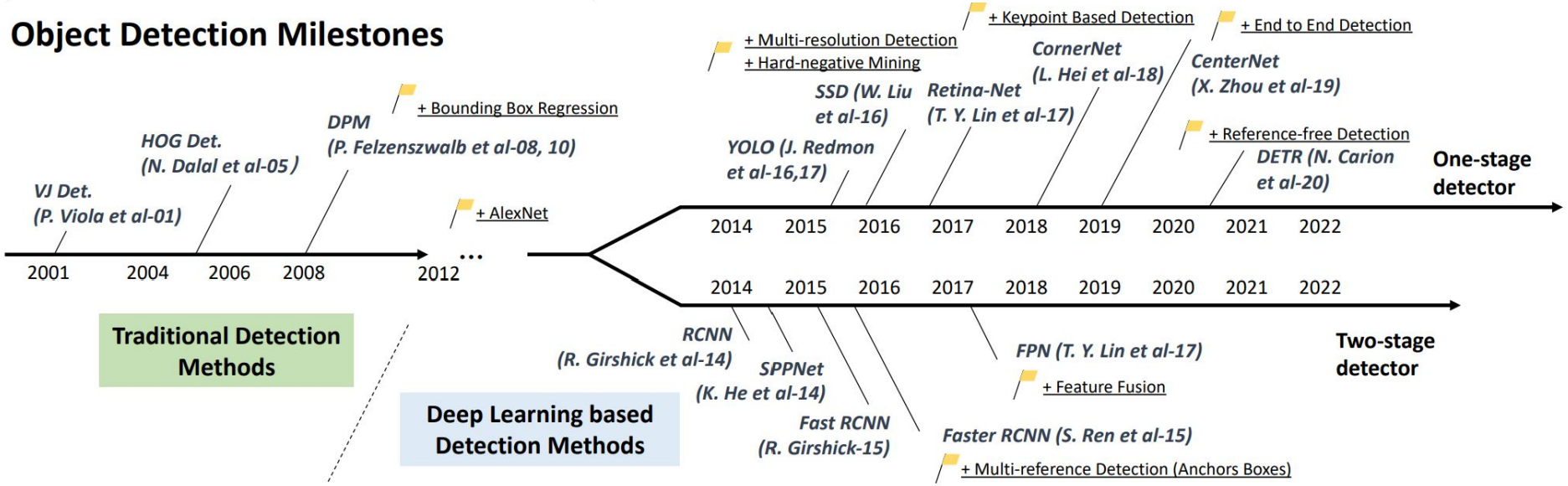


Fig. 2: A road map of object detection. Milestone detectors in this figure: VJ Det. [10, 11], HOG Det. [12], DPM [13–15], RCNN [16], SPPNet [17], Fast RCNN [18], Faster RCNN [19], YOLO [20–22], SSD [23], FPN [24], Retina-Net [25], CornerNet [26], CenterNet [27], DETR [28].

https://arxiv.org/pdf/1905.05055

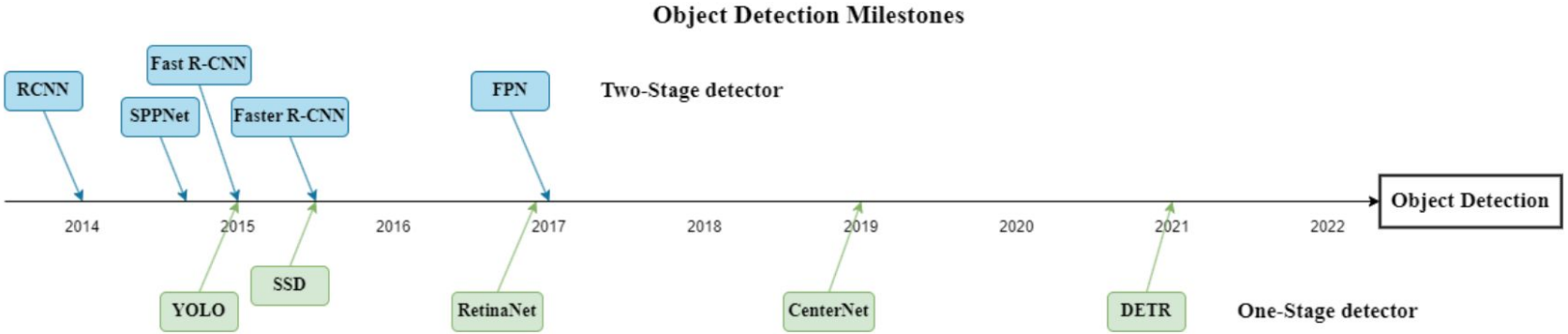


Figure 2: One-phase and two-phase detection algorithms for object detection classics published in chronological order in the last 10 years. Including one-stage detector algorithms YOLO, SSD, RetinaNet, CenterNet, and DETR; two-stage detector algorithms RCNN, SPPNet, Fast R-CNN, Faster R-CNN, and FPN

source

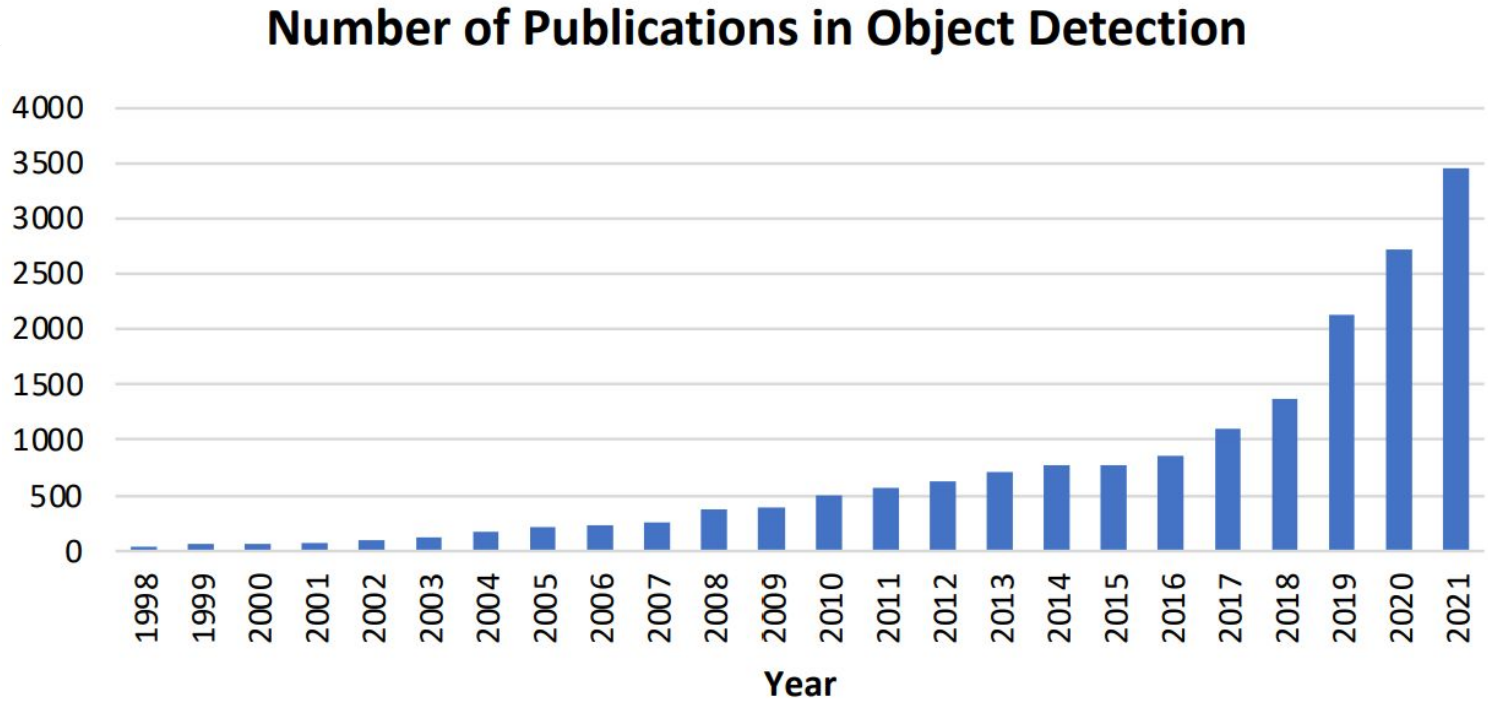


Fig. 1: The increasing number of publications in object detection from 1998 to 2021. (Data from Google scholar advanced search: *allintitle: "object detection" OR "detecting objects"*.)

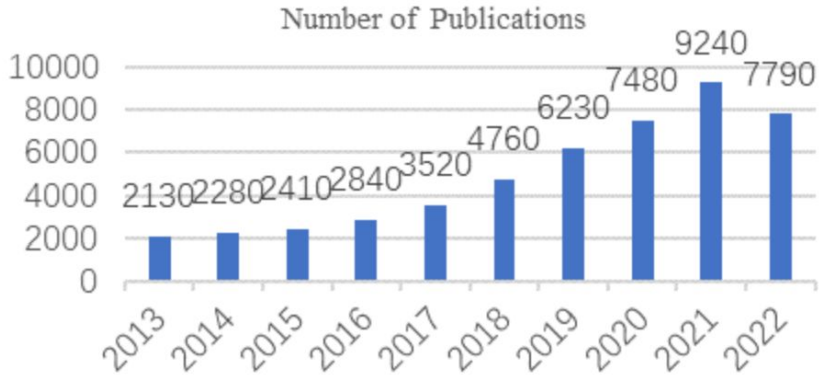


Figure 1: Number of published articles on object detection in the last 10 years. (Data from Google Scholar advanced search on the keyword "object detection")

source

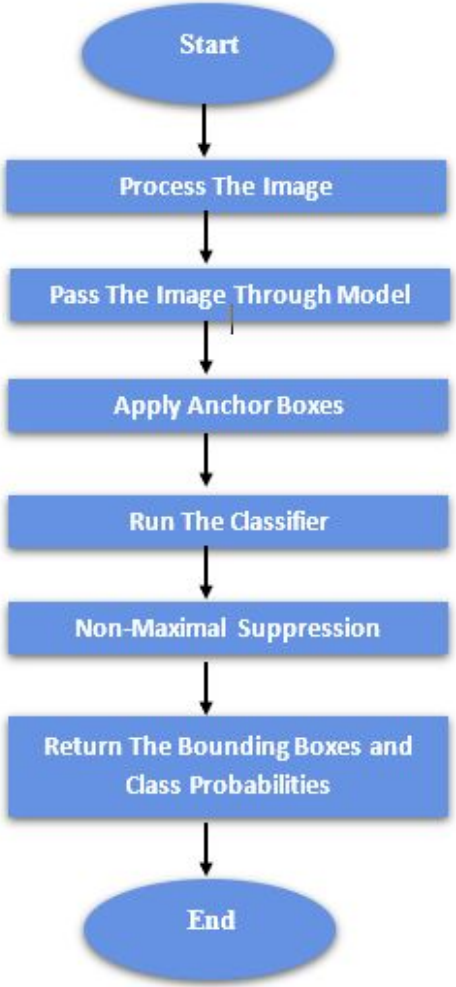
Introduction to SSD

- Single Shot MultiBox Object Detection (SSD) model was first published by **Wei Liu et al** in 2015 (https://arxiv.org/abs/1512.02325).
- SSD algorithm is based on deep CNN that takes an image as input and produces a set of bounding boxes and class predictions for all objects in the image.
- Unlike other object detection algorithms, SSD performs **both localization** and **classification** in a **single forward pass** of the network, hence the name "**single shot**".

Key features of SSD

- **Single Shot**: SSD performs object detection in a **single pass** through the network. It directly predicts the presence of objects and their bounding box coordinates in a single shot, making it faster and more efficient.
- **MultiBox:** SSD uses a **set of default bounding boxes** (anchor boxes) of different **scales** and **aspect ratios** at multiple locations in the input image. These default boxes serve as prior knowledge about where objects are likely to appear. SSD predicts adjustments to these default boxes to locate objects accurately.
- **Multi-Scale Detection**: SSD operates on **multiple feature maps** (eg. 38x38, 19x19) with different resolutions, allowing it to detect objects of various sizes. Predictions are made at different scales to capture objects at varying levels of granularity.

SSD Flow

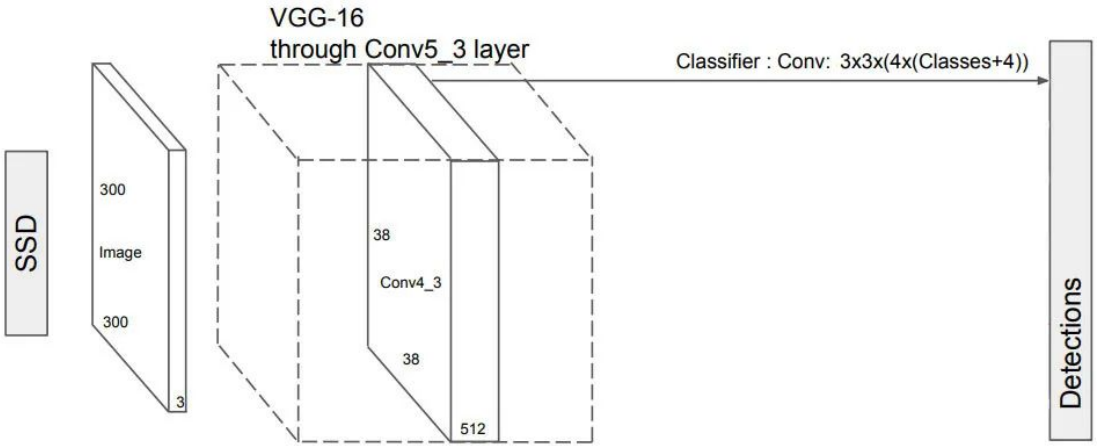

Start
Process The Image
Pass The Image Through Model
Apply Anchor Boxes
Run The Classifier
Non-Maximal Suppression
Return The Bounding Boxes and Class Probabilities
End

SSD Architecture

- SSD variants: **SSD300 and SSD512 (**300 and 512 refers to **image dimensions).** Although there is a difference in the input size, the underlying architecture does not change.
- The SSD model is a feed-forward network that outputs a set of bounding boxes and the scores for each of the classes.
- The most common **SSD300 is with VGG16** as the backbone. **ResNet18**, **ResNet34**, or even **EfficientNet** models can be also used.
- The architecture of the SSD is a deep CNN for real-time object detection.
- It combines various layers to perform localization (bounding box prediction) and classification (object category prediction) in a single forward pass.

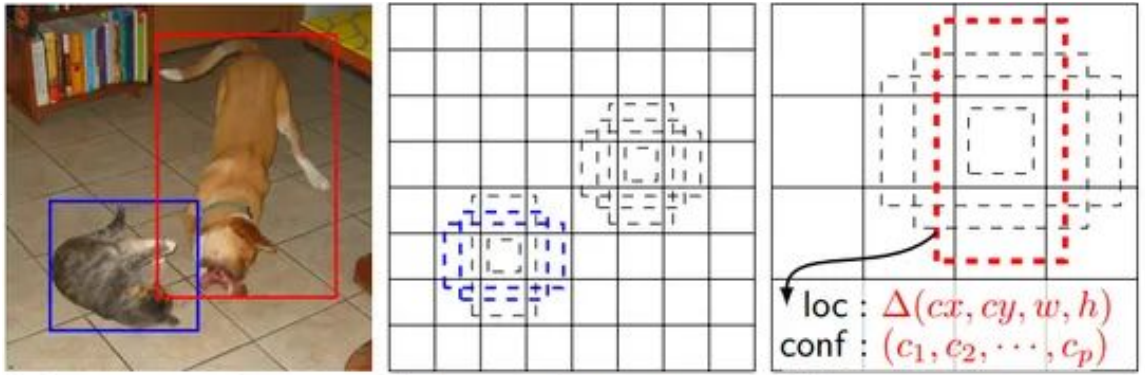
SSD Architecture

- The SSD object detection composes of 2 parts:
  - Extract feature maps, and
  - Apply convolution filters to detect objects.



Key concepts of SSD

- **Default Bounding Boxes (Anchor Boxes)**: SSD uses a **predefined** set of default bounding boxes, also known as **anchor boxes**.
- These boxes come in various **scales** and **aspect ratios**, providing prior knowledge about where objects are likely to be located in the image. SSD predicts adjustments to these default boxes to localize objects accurately.

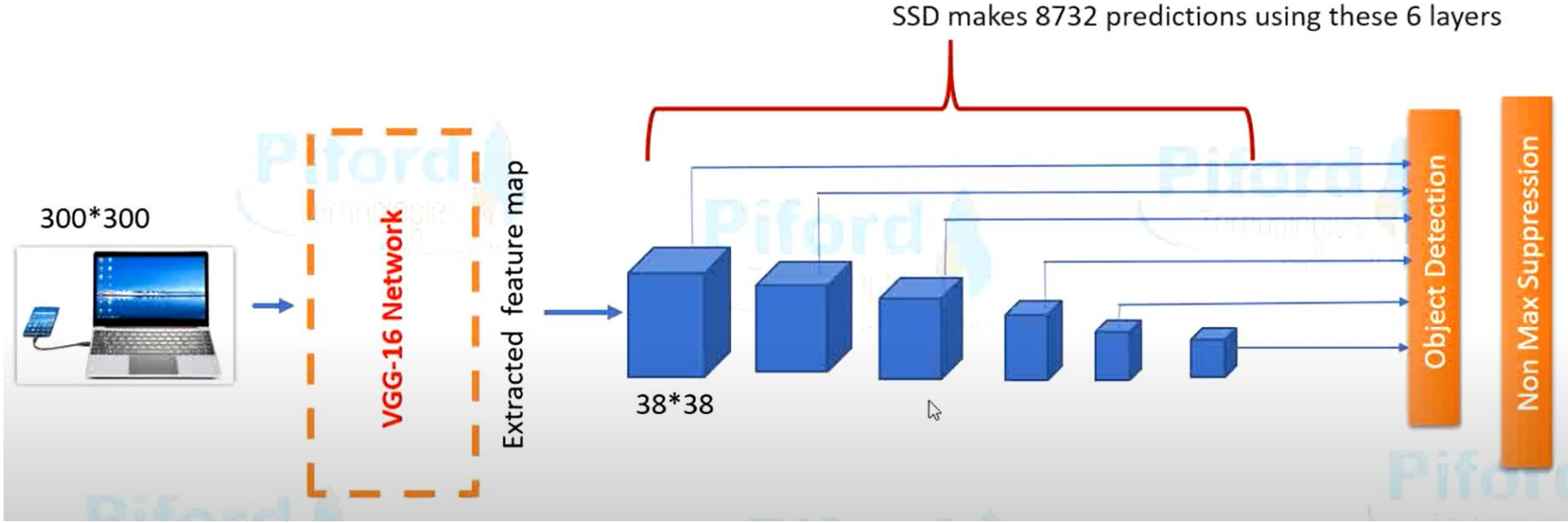


(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Key concepts of SSD

- **Multi-Scale Feature Maps**: SSD operates on **multiple feature maps** at different resolutions (eg. 38x38, 19x19). Which allows SSD to detect objects of different sizes.
- **Multi-Scale Predictions**: For each default bounding box, SSD makes predictions (classification + box offsets) at each of those feature maps. **This enables the model to capture objects at various scales.** Different feature maps cover different sizes of receptive fields, *Shallower layers-good for small objects and Deeper layers - good for large objects.*
- **Aspect Ratio Handling**: SSD uses multiple predefined default boxes with different aspect ratios at each feature map cell, and makes separate predictions for each — using a shared convolutional predictor.

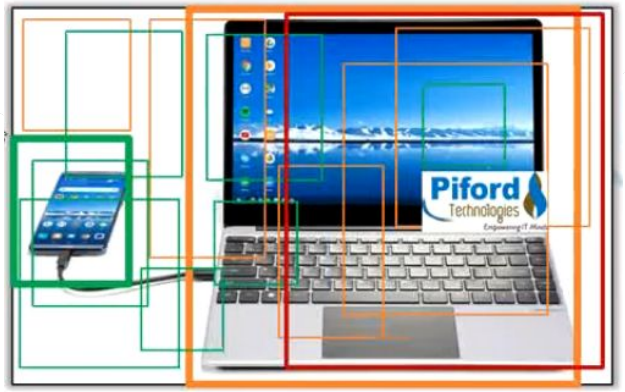
SSD300 Architecture



SSD- Matching the boxes

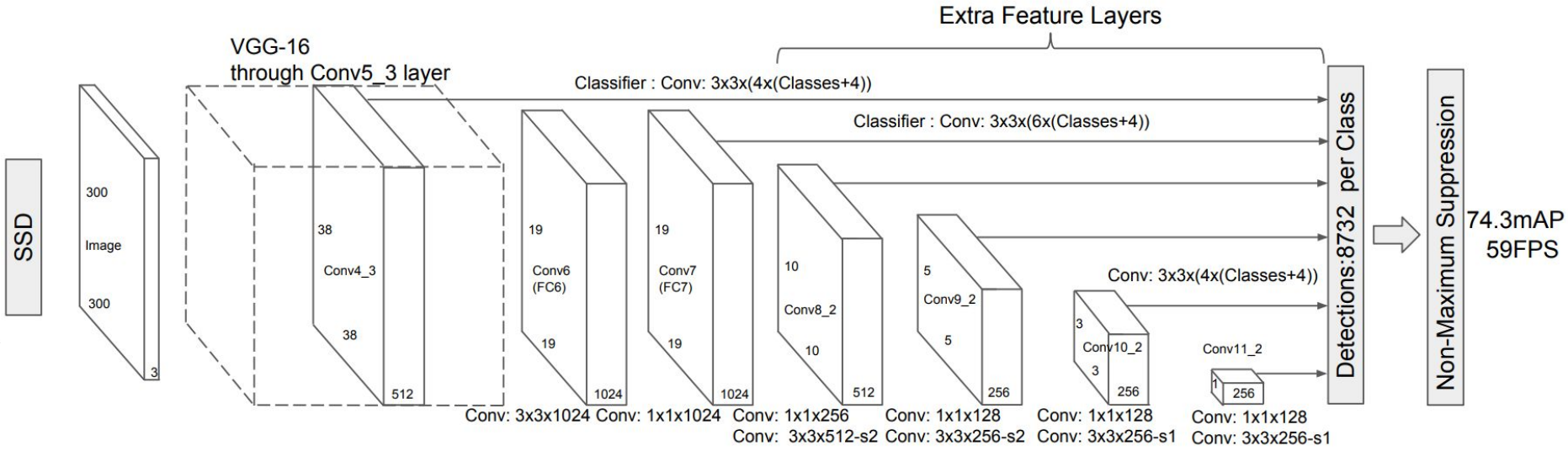
Input with GT



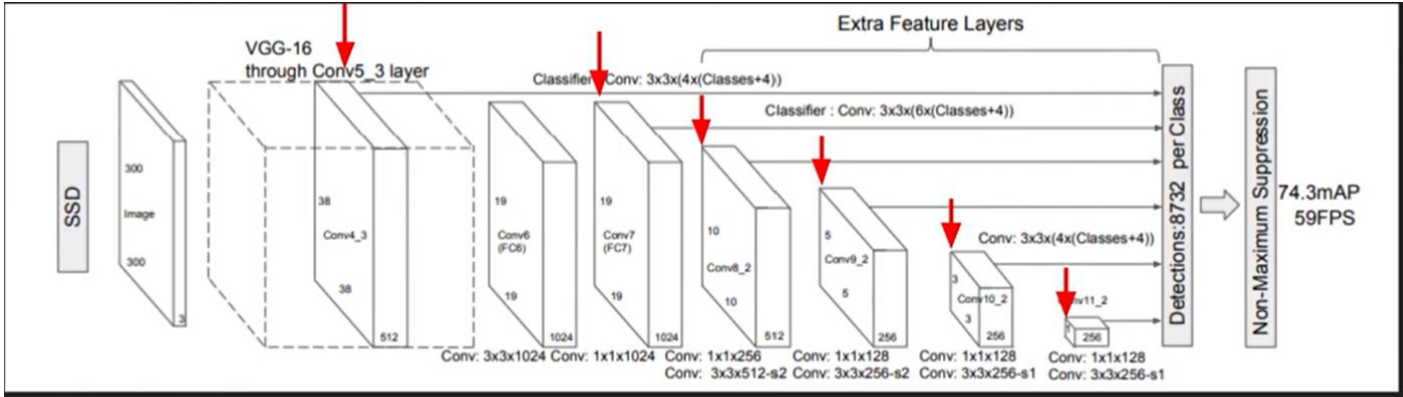


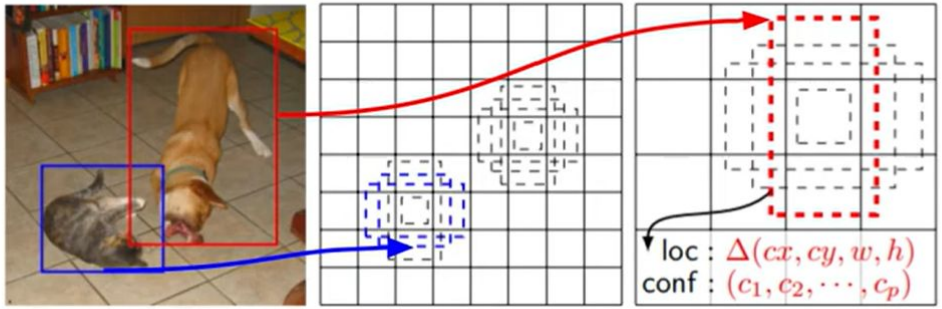
- SSD match the **8732** default bbx (anchor box) with the ground truth.
- Multiple default boxes (**8732**) of different sizes and aspect ratio across the entire image.
- Using **IoU**, find the box with highest overlap. IoU must be greater than **50%**. We will take those bbx that has the highest IoU.
- Pick top **200** boxes and perform **non-max suppression** on it.

SSD300 Architecture



Single shot object detection model architecture with VGG16 backbone.





(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

- **Bounding box offset** from predefined boxes and **class score** of the object
- Featuremap produced by using a 3 x 3 conv filter

Components of SSD

Base Network/backbone

- SSD begins with a standard CNN(eg. VGG16) architecture, which is typically used for high-quality image classification tasks.
- However, in SSD, this base network is truncated before any classification layers.
- In SSD, the focus is on **object detection**, not classification, so the network is **truncated** before the fully connected layers (typically fc6, fc7).
- The base network is responsible for extracting essential features from the input image.

Components of SSD

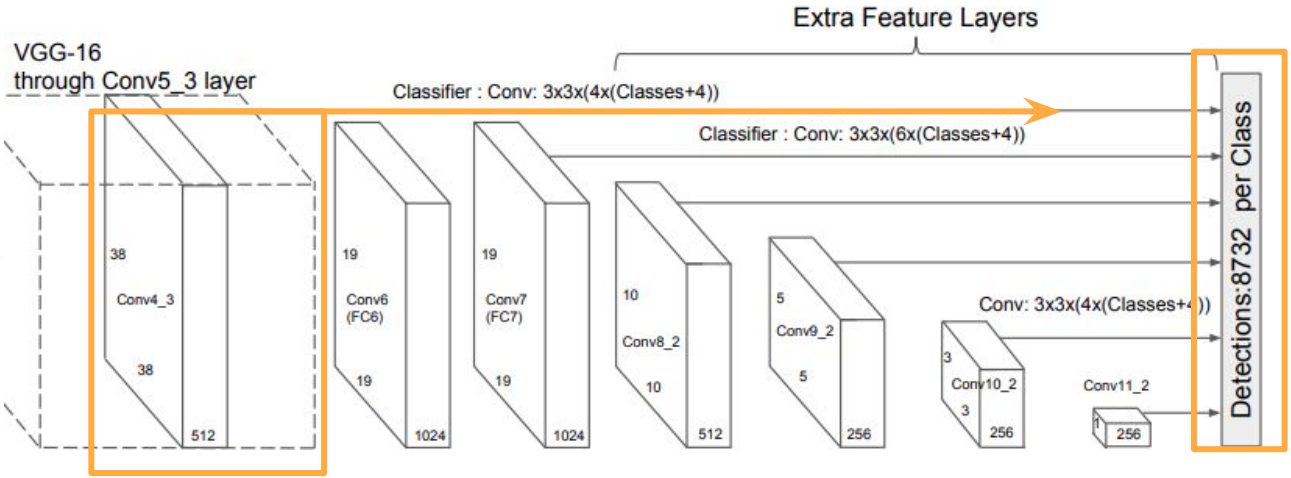
Multi-Scale Feature Maps

- Additional convolutional layers are added to the truncated base network. SSD employs **convolutional layers** to perform **dense predictions** (multiple boxes across the image).
- These layers progressively reduce the spatial dimensions while increasing the number of channels (feature channels).
- This design allows SSD to produce feature maps at multiple scales. *Each scale's feature map is suitable for detecting objects of different sizes.*

Components of SSD

Multi-Scale Feature Maps

- From the diagram, we can see that each convolutional layers has names, such as **Conv4_3, Conv6** and so on.
- We can see the dimension of each of the feature maps.
- For instance, **Conv4_3** has a dimension of **38*38** and a line from **Conv4_3** to final detection block.



Components of SSD

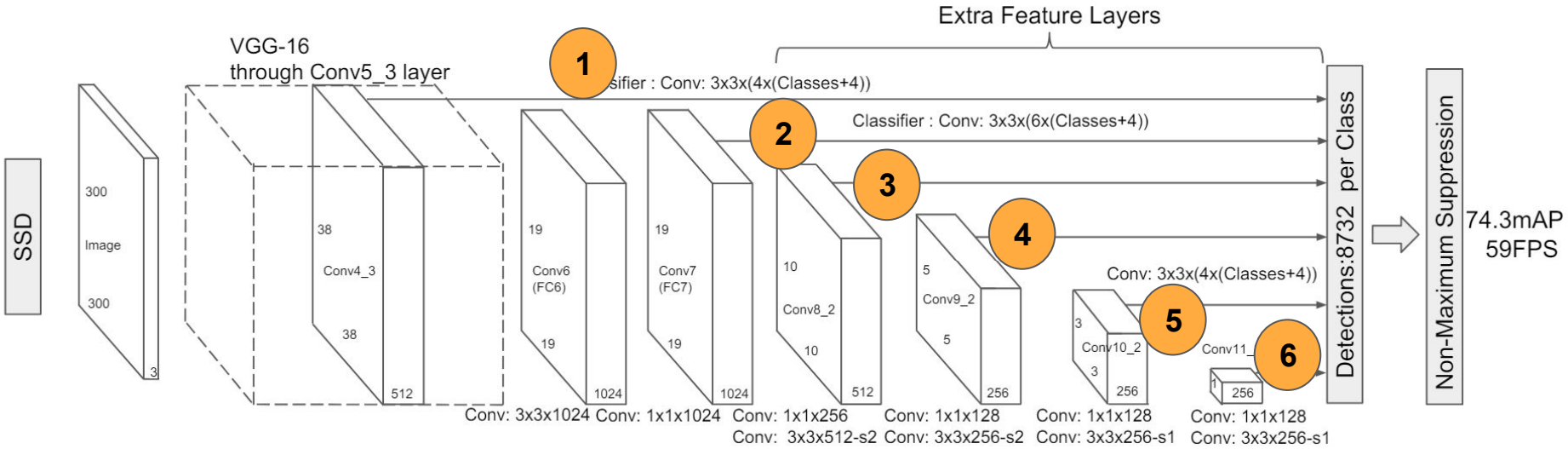
Multi-Scale Feature Maps

- Even **Conv_7, Conv8_2, Conv9_2, Conv10_2** and **Conv11_2** has a line from each of these to the final blocks.
- This indicates that the **bounding box predictions from each of these layers are used in the final predictions**.
- A total of **6** layers with different feature map resolutions are used to predict the **8732** bounding boxes and their scores.

Components of SSD

Multi-Scale Feature Maps

- SSD predicts bounding box locations and their scores at **6** different **layers**.
- These layers result in feature maps with different resolutions **(38×38 to 1×1)** which in turn helps in multi-scale predictions and detection of objects of varying sizes.



Working of SSD

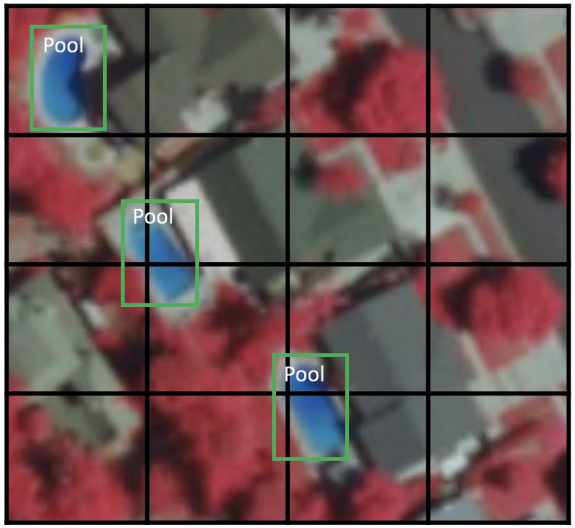
- SSD has two components: a **backbone model** and **SSD head.**
- ***Backbone*** model usually is a pre-trained image classification network as a feature extractor.
- This is typically a network like **ResNet** or **VGG16** trained on ImageNet from which the final fully connected classification layer has been removed.
- The **SSD head** is just one or more convolutional layers added to this backbone and the outputs are interpreted as the bounding boxes and classes of objects in the spatial location of the final layers activations.

source

Working of SSD

1. Grid Cell

- SSD divides the image using a **grid** and have each grid cell be responsible for detecting objects in that region of the image.
- Detection objects simply means predicting the class and location of an object within that region.
- If **no object** is present, we consider it as the **background** class and the location is **ignored**.
- For instance, we could use a 4x4 grid in the example below. Each grid cell is able to output the position and shape of the object it contains.

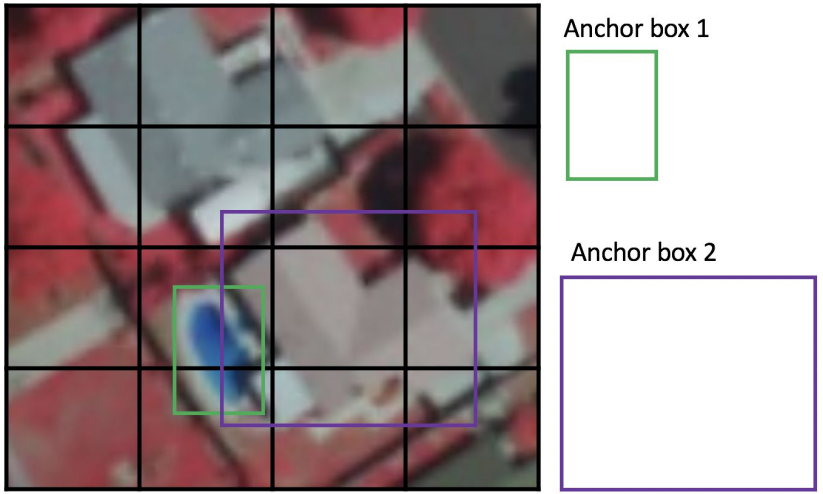


Working of SSD

The anchor box with the **highest degree of overlap with an object** is responsible for predicting that object's class and its location.

2. Anchor Box

- If there are multiple objects in one grid cell or we need to detect multiple objects of different shapes. There is where **anchor box** into play.
- Each grid cell in SSD can be assigned with multiple anchor/prior boxes.
- These anchor boxes are **predefined** and each one is responsible for a size and shape within a grid cell.
- For eg:, the swimming pool in the image corresponds to the taller anchor box while the building corresponds to the wider box.
- SSD uses a **matching phase** while **training**, to match the appropriate anchor box with the bounding boxes of each ground truth object within an image.



This property is used for training the network and for predicting the detected objects and their locations once the network has been trained.

Working of SSD

Aspect ratio is choosing the **shape of the window**, and **scale** is choosing how **big** the window is.

3. Aspect Ratio

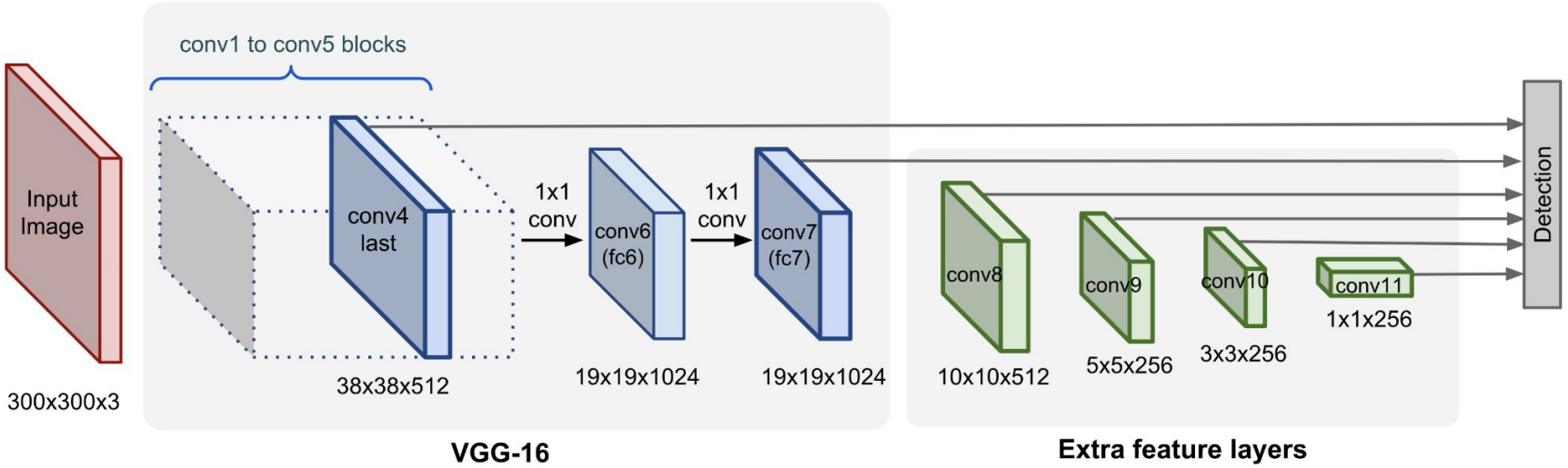
- Not all objects are square in shape. Some are longer and some are wider, by varying degrees.
- The SSD architecture allows pre-defined aspect ratios of the **anchor boxes** to account for this.
- The **ratios parameter(eg;** 1:1, 2:1, and 1:2**)** can be used to specify the different aspect ratios of the anchor boxes associates with each grid cell at each **zoom/scale level.**



SSD Architecture

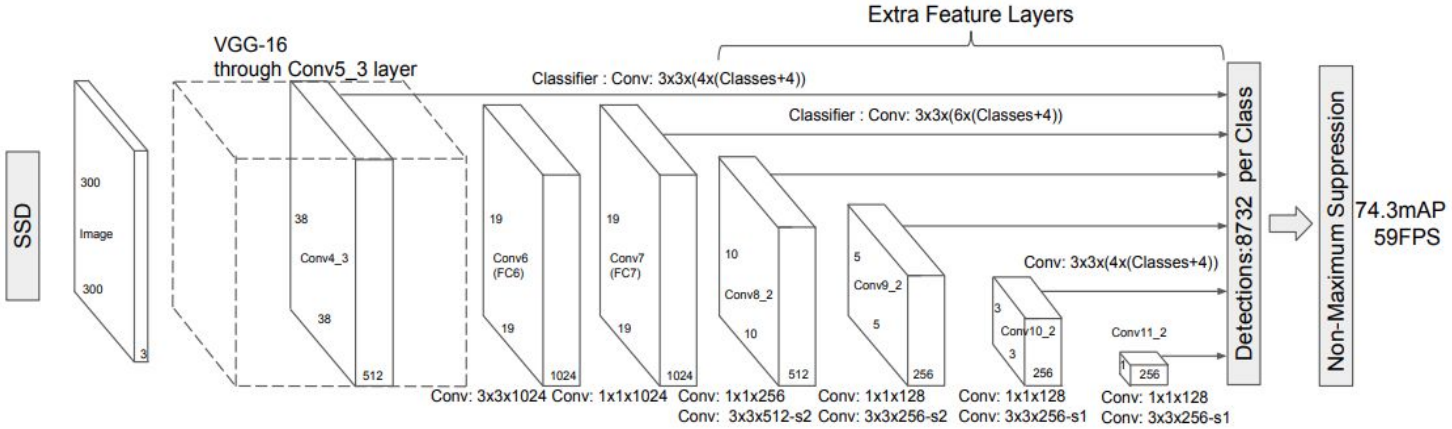
The total number of anchor boxes (default boxes) in SSD depends on:

- The number of feature maps used
- The size (resolution) of each feature map
- The number of default boxes per location (usually 4 to 6)



- SSD makes many prediction for every single class.
- SSD uses **6** CNN layers and it makes **8732** predictions (bounding boxes) for **every single objects**.
- Using **NMS**, top **200** predictions will be taken based on **confidence score**.

SSD300 Architecture



The number of anchor boxes in a cell

conv4_3	→	38x38x(4x(Classes+4))	=	5776x(Classes+4)
conv7	→	19x19x(6x(Classes+4))	=	2166x(Classes+4)
conv8_2	→	10x10x(6x(Classes+4))	=	600x(Classes+4)
conv9_2	→	5x5x(6x(Classes+4))	=	150x(Classes+4)
conv10_2	→	3x3x(4x(Classes+4))	=	36x(Classes+4)
conv11_2	→	1x1x(4x(Classes+4))	=	4x(Classes+4)

\+ = 8732x(Classes+4)

Anchor boxes **coordinates**

8732 total anchor boxes
For eg, **20** classes:
8372×(20+4)=8372×24= 200,928
This means there would be 200,928 predicted values in total for all anchor boxes across the feature map.

Working of SSD

What really happens during training?

1. **Before training**, for each **grid cell** on each **feature map**, the model generates a **fixed set of anchor boxes**.
    - E.g., for each cell: 6 boxes of different shapes and sizes.
    - This gives you **thousands of anchor boxes total** per image.
2. During training:
    - For each **ground truth object**, the model tries to find the **best-matching anchor box** using **IoU (Intersection over Union)**.
    - If IoU > threshold (e.g., 0.5), that anchor is labeled as **positive** (i.e., it should learn to predict that object).
    - All other anchors are either **ignored** or labeled **negative** (background).

SSD

Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

Table 7: **Results on Pascal VOC2007 `test`.** SSD300 is the only real-time detection method that can achieve above 70% mAP. By using a larger input image, SSD512 outperforms all methods on accuracy while maintaining a close to real-time speed.

https://arxiv.org/pdf/1512.02325

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast [6]	07	66.9	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8
Fast [6]	07+12	70.0	77.0	78.1	69.3	59.4	38.3	81.6	78.6	86.7	42.8	78.8	68.9	84.7	82.0	76.6	69.9	31.8	70.1	74.8	80.4	70.4
Faster [2]	07	69.9	70.0	80.6	70.1	57.3	49.9	78.2	80.4	82.0	52.2	75.3	67.2	80.3	79.8	75.0	76.3	39.1	68.3	67.3	81.1	67.6
Faster [2]	07+12	73.2	76.5	79.0	70.9	65.5	52.1	83.1	84.7	86.4	52.0	81.9	65.7	84.8	84.6	77.5	76.7	38.8	73.6	73.9	83.0	72.6
Faster [2]	07+12+COCO	78.8	84.3	82.0	77.7	68.9	65.7	88.1	88.4	88.9	63.6	86.3	70.8	85.9	87.6	80.1	82.3	53.6	80.4	75.8	86.6	78.9
SSD300	07	68.0	73.4	77.5	64.1	59.0	38.9	75.2	80.8	78.5	46.0	67.8	69.2	76.6	82.1	77.0	72.5	41.2	64.2	69.1	78.0	68.5
SSD300	07+12	74.3	75.5	80.2	72.3	66.3	47.6	83.0	84.2	86.1	54.7	78.3	73.9	84.5	85.3	82.6	76.2	48.6	73.9	76.0	83.4	74.0
SSD300	07+12+COCO	79.6	80.9	86.3	79.0	**76.2**	57.6	87.3	88.2	88.6	60.5	85.4	**76.7**	**87.5**	**89.2**	84.5	81.4	55.0	81.9	**81.5**	85.9	78.9
SSD512	07	71.6	75.1	81.4	69.8	60.8	46.3	82.6	84.7	84.1	48.5	75.0	67.4	82.3	83.9	79.4	76.6	44.9	69.9	69.1	78.1	71.8
SSD512	07+12	76.8	82.4	84.7	78.4	73.8	53.2	86.2	87.5	86.0	57.8	83.1	70.2	84.9	85.2	83.9	79.7	50.3	77.9	73.9	82.5	75.3
SSD512	07+12+COCO	**81.6**	**86.6**	**88.3**	**82.4**	76.0	**66.3**	**88.6**	**88.9**	**89.1**	**65.1**	**88.4**	73.6	86.5	88.9	**85.3**	**84.6**	**59.1**	**85.0**	80.4	**87.4**	**81.2**

Table 1: **PASCAL VOC2007 `test` detection results.** Both Fast and Faster R-CNN use input images whose minimum dimension is 600. The two SSD models have exactly the same settings except that they have different input sizes (300×300 vs. 512×512). It is obvious that larger input size leads to better results, and more data always helps. Data: "07": VOC2007 `trainval`, "07+12": union of VOC2007 and VOC2012 `trainval`. "07+12+COCO": first train on COCO `trainval35k` then fine-tune on 07+12.

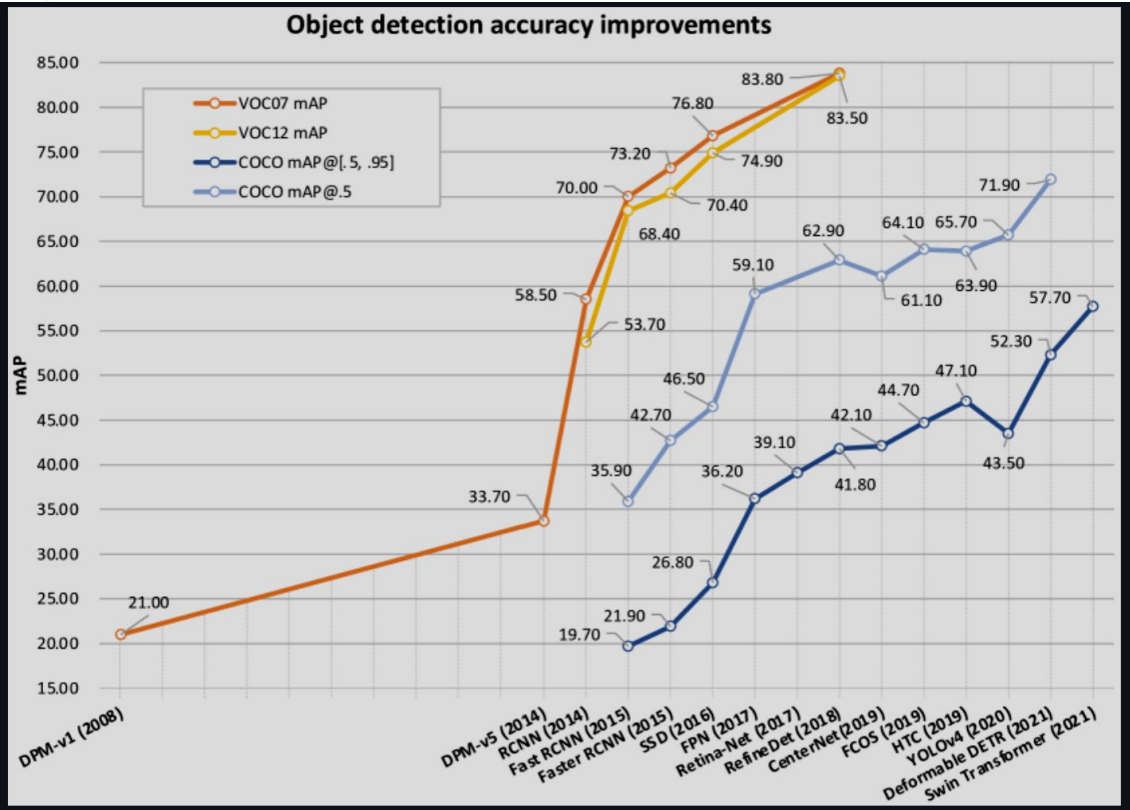


Figure 3: Accuracy improvement of object detection on VOC07, VOC12 and MS-COCO datasets. Detectors in this figure: DPM-v1 [13], DPM-v5 [37], RCNN [16], SPPNet [17], Fast RCNN [18], Faster RCNN [19], SSD [23], FPN [24], Retina-Net [25], RefineDet [38], TridentNet [39] CenterNet [40], FCOS [41], HTC [42], YOLOv4 [22], Deformable DETR [43], Swin Transformer [44].

source

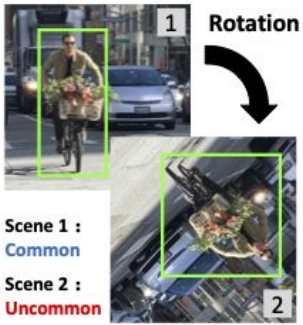
Applications

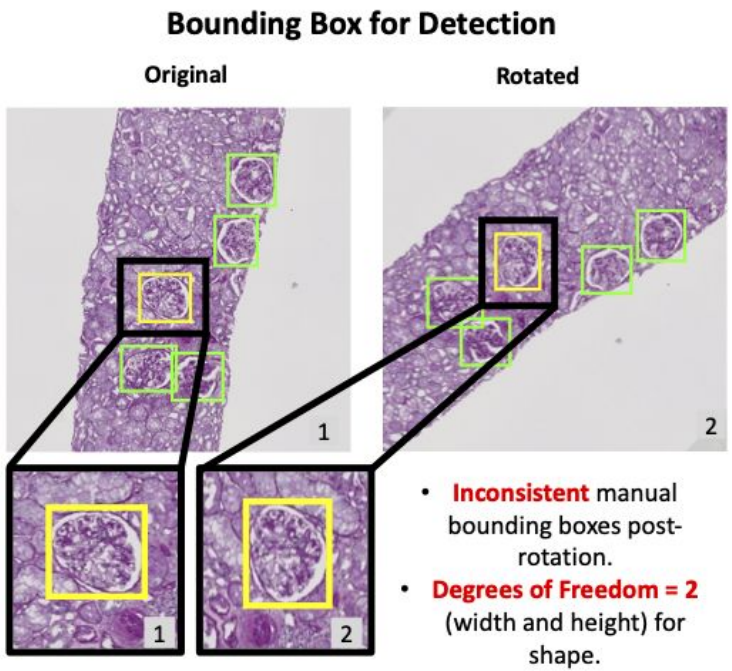
- Google – Mobile object detection in TensorFlow Lite.
- Tesla – Autonomous vehicle object detection.
- NVIDIA – Real-time video surveillance with Jetson platform.
- Amazon – Retail analytics and object recognition in Amazon Go stores.
- Huawei – Smart city surveillance and traffic monitoring.
- CureMetrix – Real-time medical imaging for breast cancer detection.

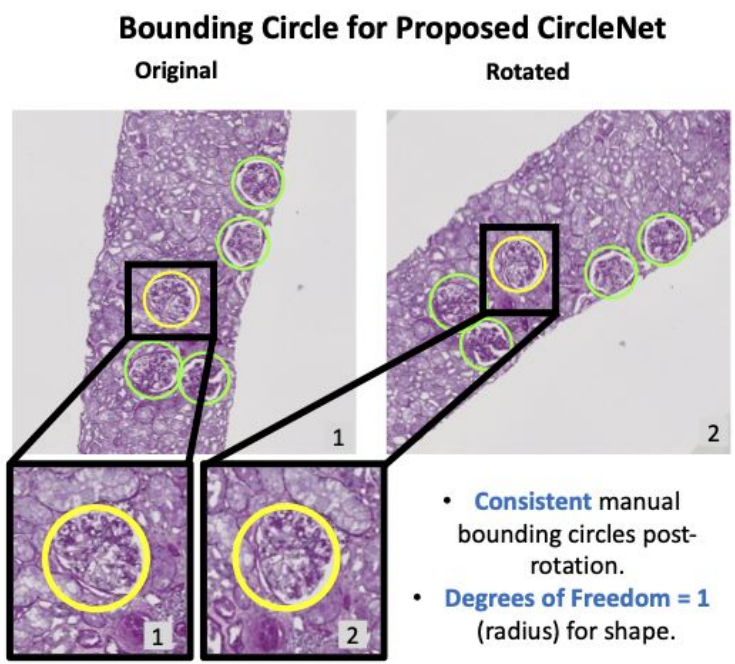
Limitations

- SSD's primary limitation is its difficulty in **accurately** detecting tiny objects, heavily **occluded** objects, or objects with **extreme aspect ratios** (very wide, very tall), which can impact its performance in certain scenarios.
- Models like **YOLOv5/YOLOv8** (with learnable or anchor-free detection), **Faster R-CNN**, **RetinaNet**, and **EfficientDet** offer significant improvements.
- Additionally, anchor-free models like **FCOS** (Fully Convolutional One-Stage Object Segmenter), **CenterNet**, and **DETR**(DEtection TRansformer) excel at handling extreme aspect ratios and occlusions without relying on fixed anchor boxes, making them ideal for complex detection scenarios.

CircleNet - Anchor free Model







When detecting glomeruli and nuclei on pathological images, the proposed circle representation achieved superior detection performance and be more rotation-invariant, compared with the bounding box

https://github.com/hrlblab/CircleNet

PolyYolo

Polygonal bounding boxes for objects

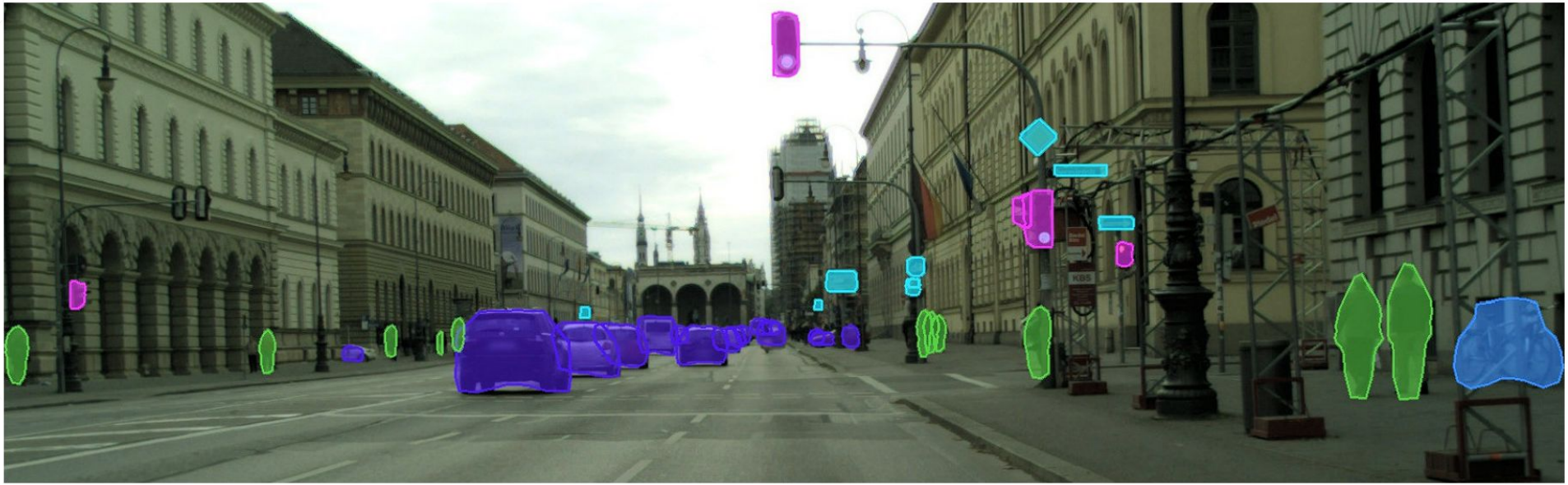


Figure 1: The figure shows instance segmentation performance of the proposed Poly-YOLO algorithm applied on Cityscapes dataset and running 22FPS on a mid-tier graphic card. Image was cropped due to visibility.

https://arxiv.org/pdf/2005.13243

GYALPOZHING
COLLEGE OF INFORMATION TECHNOLOGY



Thank You