UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack:Anashua Krittika Dastidar, Teaching Assistant

Introduction

1. Object detection is a vital computer vision task focusing on identifying instances like humans, animals, or cars in digital images.

1. Its goal is to provide essential knowledge for computer vision applications by determining what objects are present and where they are located.

1. Key metrics for object detection include accuracy, encompassing both classification and localization accuracy, as well as speed.

1. Object detection serves as the foundation for other computer vision tasks such as instance segmentation, image captioning, and object tracking.

1. Recent years have seen significant progress in object detection, driven by rapid developments in deep learning techniques.Fig. 1 illustrates the growing number of publications associated with "object detection" over the past two decades.

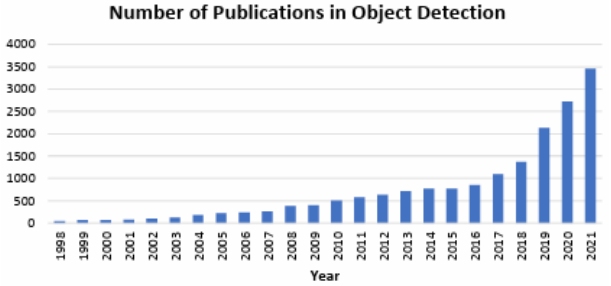
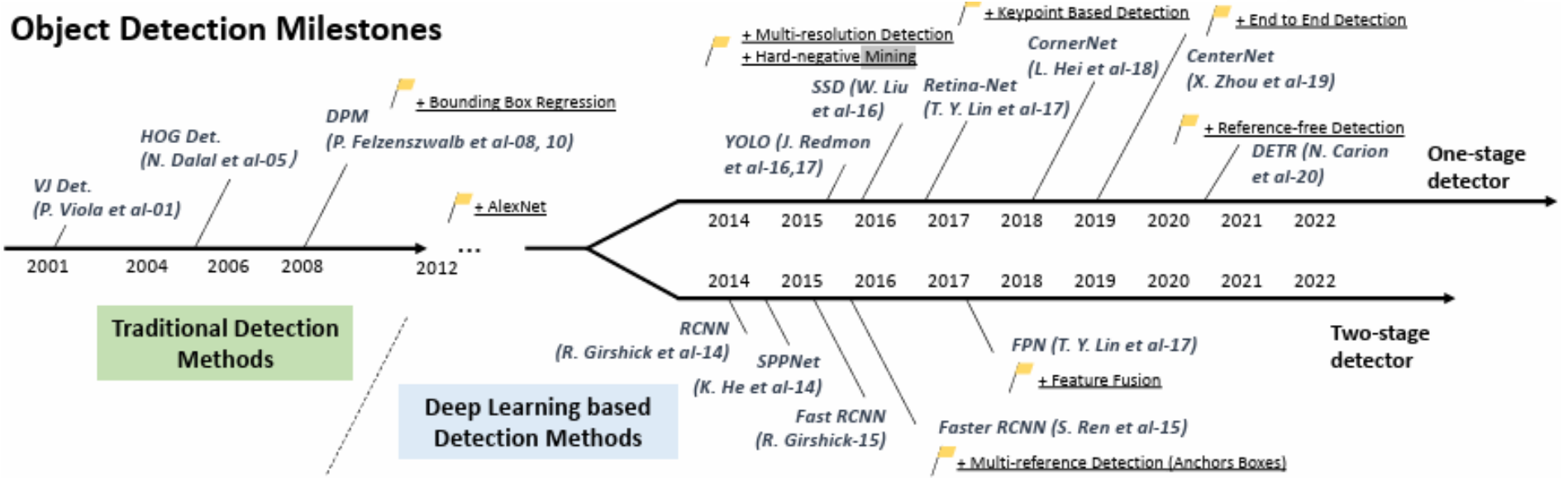


Fig. 1: The increasing number of publications in object detection from 1998 to 2021. (Data from Google scholar advanced search: *allintitle: "object detection" OR "detecting objects".*)

Source [1]



Source [1]

In the past two decades, it is widely accepted that the progress of object detection has generally gone through two historical periods: “traditional object detection period (before 2014)” and “deep learning based detection period (after 2014)”.

Traditional Detectors

Most of the early object detection algorithms were built based on handcrafted features. Due to the lack of effective image representation at that time, people have to design sophisticated feature representations and a variety of speed-up skills. Some of them are :

1. **Viola Jones Detectors:** In 2001, P. Viola and M. Jones achieved real-time detection of human faces for the first time without any constraints (e.g., skin color segmentation) [4,5].

1. **HOG Detector**: In 2005, N. Dalal and B. Triggs proposed Histogram of Oriented Gradients (HOG) feature descriptor [6].

1. **Deformable Part-based Model (DPM)**: DPM, as the winners of VOC-07,-08, and-09 detection challenges, was the epitome of the traditional object detection methods. DPM was originally proposed by P. Felzenszwalb [7] in 2008 as an extension of the HOG detector.

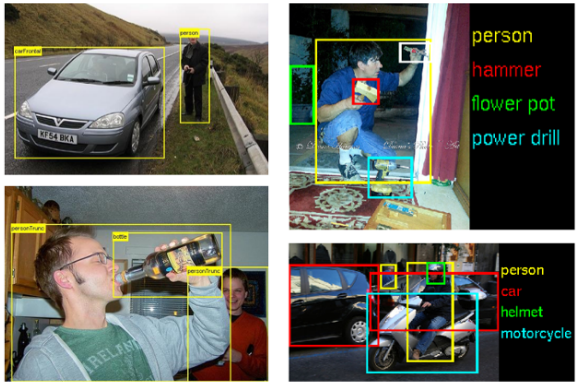
Datasets for Object Detection

Datasets for Object Detection

Building larger datasets with less bias is essential for developing advanced detection algorithms. A number of well-known detection datasets have been released in the past 10 years:

1. **Pascal VOC:** The **PASCAL Visual Object Classe**s (VOC) Challenges (from 2005 to 2012) was one of the most important competitions in the early computer vision community. Two versions of Pascal-VOC are mostly used in object detection: VOC07 and VOC12, where the former consists of 5k training images + 12k annotated objects, and the latter consists of 11k training images + 27k annotated objects. 20 classes of objects that are common in life are annotated in these two datasets.
    a. Link to challenges

1. **ILSVRC**: The **ImageNet Large Scale Visual Recognition Challenge (ILSVRC)** was organized each year from 2010 to 2017. It contains a detection challenge using ImageNet images. The ILSVRC detection dataset contains 200 classes of visual objects. The number of its images/ object instances is two orders of magnitude larger than VOC.
    a. Link to challegege
    b. More info



Example images and annotations
1. Pascal VOC
2. ILSVRC
Source [1]

Datasets for Object Detection

MS-COCO: **MS-COCO** is one of the most challenging object detection dataset available today. It has less number of object categories than ILSVRC, but more object instances. For example, MS-COCO-17 contains 164k images and 897k annotated objects from 80 categories.

→ Compared with VOC and ILSVRC, the biggest progress of MS-COCO is that apart from the bounding box annotations, each object is further labeled using per-instance segmentation to aid in precise localization.

→ In addition, MS-COCO contains more small objects (whose area is smaller than 1% of the image) and more densely located objects. Just like ImageNet in its time, MS-COCO has become the de facto standard for the object detection community.

→ Link to challenge

→ More information

Example images and annotation
MS COCO source1 [1] , source2

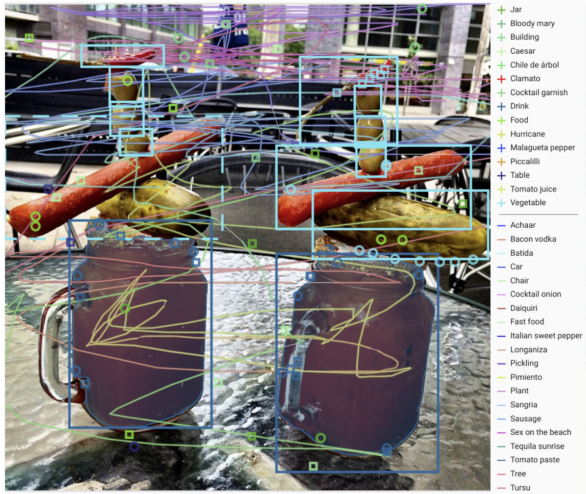
Datasets for Object Detection

Open Images: The year of 2018 sees the introduction of the Open Images Detection (OID) challenge, following MS-COCO but at an unprecedented scale.
There are two tasks in Open Images:

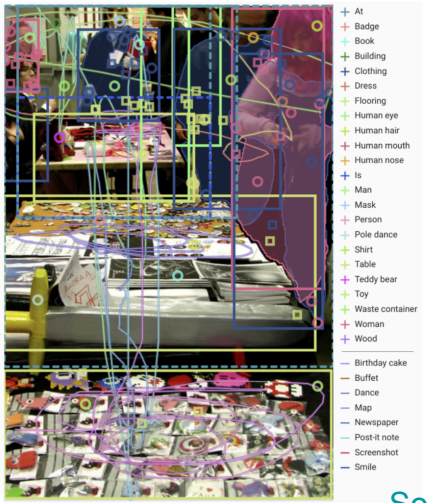
1. The standard object detection
2. Visual relationship detection which detects paired objects in particular relations.

For the standard detection task, the dataset consists of 1,910k images with 15,440k annotated bounding boxes on 600 object categories.

→ Link to dataset



"In the foreground of the picture there are jars, drink, sausage and other food items placed on a table.
In the center of the picture there are chairs, tables and other objects.
In the background there are buildings, trees, cars, footpath and other objects."



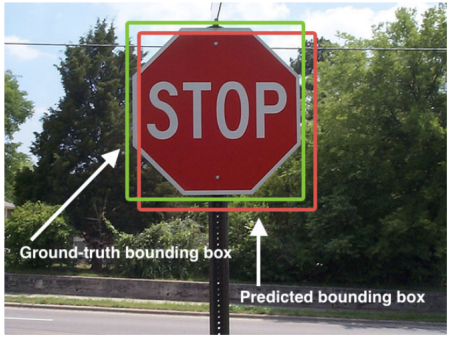
"As we can see in the image there are few people wall, mirror and tables.
On tables there are books, papers and covers."

Source

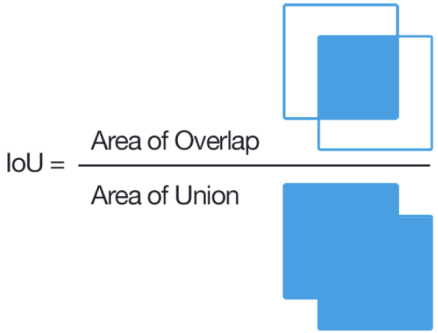
Metrics for Object Detection

Intersection over Union (IoU)

Intersection over Union indicates the overlap of the predicted bounding box coordinates to the ground truth box. Higher IoU indicates the predicted bounding box coordinates closely resembles the ground truth box coordinates.

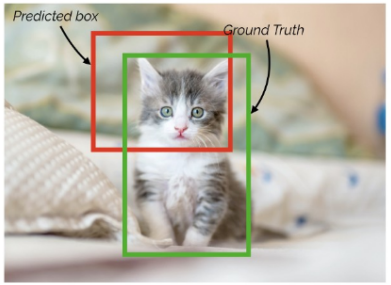


$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



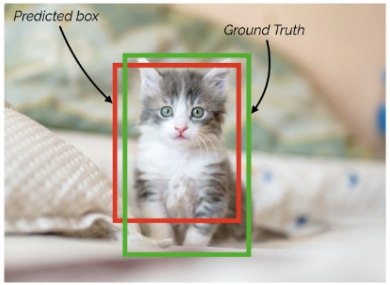
If IoU threshold = 0.5

False Positive (FP)



IoU = ~0.3

True Positive (TP)



IoU = ~0.7

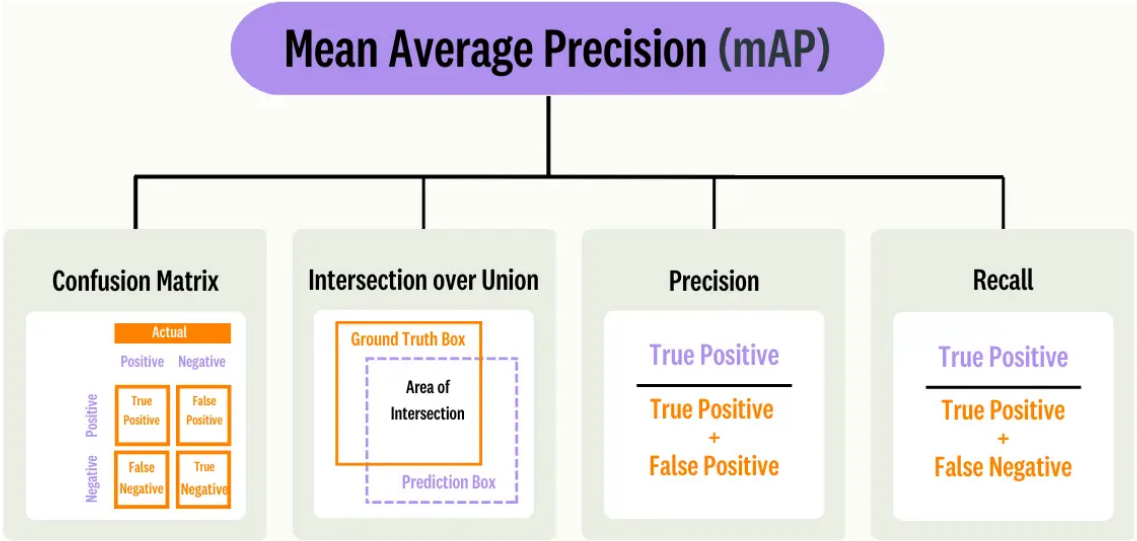
To measure the object localization accuracy, the IoU between the predicted box and the ground truth is used to verify whether it is greater than a predefined threshold, say, 0.5. If yes, the object will be identified as “detected”, otherwise, “missed”.

Source

What is mAP?

Mean Average Precision(mAP) is a metric used to evaluate object detection models such as Fast R-CNN, YOLO, Mask R-CNN, etc. The mean of average precision(AP) values are calculated over recall values from 0 to 1.

mAP formula is based on the following sub metrics Confusion Matrix, Intersection over Union(IoU), Recall and Precision



Source

How to calculate AP?

→ AP@α is Area Under the Precision-Recall Curve(AUC-PR) evaluated at α IoU threshold.

$$AP@\alpha = \int_0^1 p(r)\, dr$$

→ Notation: AP@α or APα means that AP precision is evaluated at α IoU threshold. If you see metrics like AP50 and A75, they mean AP calculated at IoU=0.5 and IoU=0.75, respectively.

→ A high Area Under the PR Curve means high recall and high precision which is preferred.

→ AP is calculated individually for each class. This means that there are as many AP values as the number of classes.

→ These AP values are averaged to obtain the mean Average Precision (mAP) metric.

Source

What is mAP?

The mAP is calculated by finding Average Precision(AP) for each class and then average over a number of classes.The formula for mAP essentially tells us that, for a given class, i we need to calculate its corresponding AP. The mean of these collated AP scores will produce the mAP and inform us how well the model performs.

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i$$

mAP50: Mean average precision calculated at an intersection over union (IoU) threshold of 0.50. It's a measure of the model's accuracy considering only the "easy" detections.

mAP50-95: The average of the mean average precision calculated at varying IoU thresholds, ranging from 0.50 to 0.95. It gives a comprehensive view of the model's performance across different levels of detection difficulty.

The 0.5-IoU mAP has then become the de facto metric for object detection. After 2014, due to the introduction of MS-COCO datasets, researchers started to pay more attention to the accuracy of object localization. Instead of using a fixed IoU threshold, MS-COCO AP is averaged over multiple IoU thresholds between 0.5 and 0.95, which encourages more accurate object localization and may be of great importance for some real world applications

Source

Frames Per Second (fps)

- When it comes to object detection algorithms, processing speed is of paramount importance. The most common metric that is used to measure the detection speed is the number of frames per second (FPS).
- A high **FPS** value indicates that the model can process frames quickly, making it suitable for time-sensitive applications like autonomous vehicles, surveillance systems, robotics, and more.
- On the other hand, a low FPS value implies that the model is slower, which might limit its applicability in certain real-time scenarios.
- For example, Faster R-CNN operates at only 7 frames per second (FPS), whereas SSD operates at 59 FPS.
- In benchmarking experiments, you will see the authors of a paper stating their network results as: “Network X achieves mAP of Y% at Z FPS”. Where X is the network name, Y is the mAP percentage, and Z is the FPS.

Two Stage Detectors vs One Stage Detectors

Two stage Object Detector

The two-stage object detector divides the whole process into 2 steps:

1. It first extracts the features using a CNN
2. It then extracts a series of regions of interest called object proposals and then the classification and localization happens only on the object proposals.

Two-stage object detectors are very powerful and extremely accurate having very high values of mAP. Hence, they are mostly used in the medical domain where classification accuracy is more important than speed. Examples of two-stage object detectors are the R-CNN family, SPP-Net, etc.

Single stage Object Detector

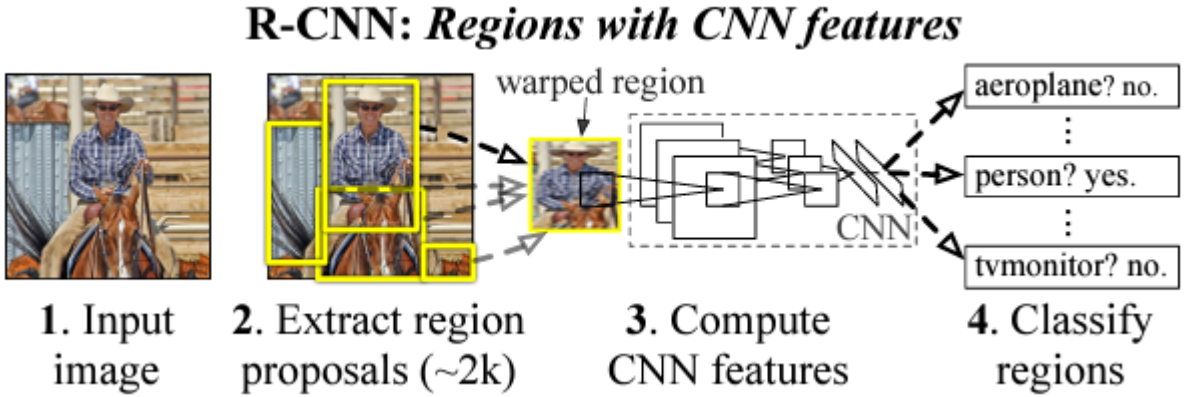
- In a single-stage object detector, we go directly from the image to classification and bounding box coordinates.
- The images are fed into a feature extractor using a CNN and then the extracted features are directly used for classification and regression for bounding box coordinates.
- Single-stage object detectors are very very fast and can be used in real-time object detection but sometimes their performance is poorer than two-stage object detectors.
- Examples are the YOLO family, SSD, RetinaNet, etc

Models for Object Detection :

Two Stage Detectors

R-CNN

Region-based Convolutional Neural Network (R-CNN) is a type of deep learning architecture used for object detection in computer vision tasks. R-CNN was one of the pioneering models that helped advance the object detection field by combining the power of convolutional neural networks and region-based approaches. Let's dive deeper into how R-CNN works, step by step.



Source[9]

R-CNN- Region Proposal

R-CNN starts by dividing the input image into multiple regions or subregions. These regions are referred to as "region proposals" or "region candidates." The region proposal step is responsible for generating a set of potential regions in the image that are likely to contain objects. R-CNN does not generate these proposals itself; instead, it relies on external methods like Selective Search or EdgeBoxes to generate region proposals.

Selective Search : The selective search algorithm [9] works by generating sub-segmentations of the image that could belong to one object — based on color, texture, size and shape — and iteratively combining similar regions to form objects. This gives ‘object proposals’ of different scales.

The authors use the selective search algorithm to generate 2000 category independent region proposals (usually indicated by rectangular regions or ‘bounding boxes’) for each individual image.

source

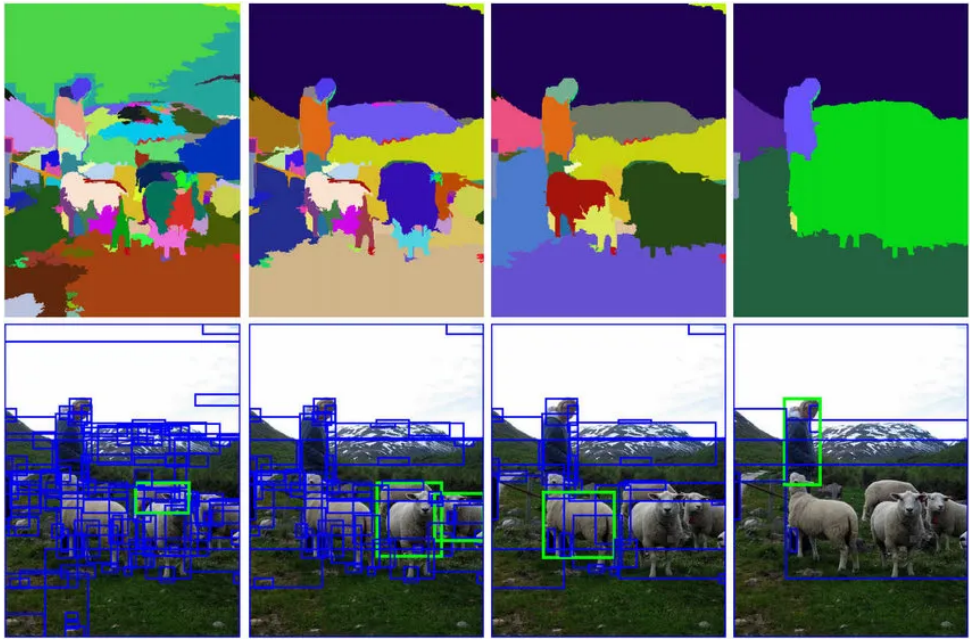


Figure 1: How the selective search algorithm iteratively obtains “region proposals”

R-CNN- Feature extraction from Region Proposals

- In this stage, each region proposal is warped or cropped into a fixed resolution, and the CNN module is utilized to extract a 4096-dimensional feature as the final representation from each of the 2000 region proposals for every image.
- The CNN used is of the same architecture as AlexNet by Krizhevsky et al pre-trained on a ILSVRC 2012 classification using image-level annotations only (to achieve results comparable to AlexNet) and the fine tuned on warped region proposals only.
- Due to large learning capacity, dominant expressive power, and hierarchical structure of CNNs, a high-level, semantic, and robust feature representation for each region proposal can be obtained.

source

R-CNN- SVM for object classification

This stage consists of learning an **individual** linear SVM (Support Vector Machine) classifier for each class, that detects the presence or absence of an object belonging to a particular class.

Inputs: The 4096-d feature vector for each region proposal.

Labels for training: The features of all region proposals that have an IoU overlap of less than 0.3* with the ground truth bounding box are considered negatives for that class during training. The positives for that class are simply the features from the ground truth bounding boxes itself. All other proposals (IoU overlap greater than 0.3, but not a ground truth bounding box) are ignored for the purpose of training the SVM.

*This number 0.3 was found using grid search on a validation set

source

R-CNN- Bounding Box Regression and Non Maximum Suppression

Bounding Box Regression

In addition to classifying objects, R-CNN also performs bounding box regression. For each class, a separate regression model is trained to refine the location and size of the bounding box around the detected object. The bounding box regression helps improve the accuracy of object localization by adjusting the initially proposed bounding box to better fit the object's actual boundaries.

Non-Maximum Suppression (NMS)

After classifying and regressing bounding boxes for each region proposal, R-CNN applies non-maximum suppression to eliminate duplicate or highly overlapping bounding boxes. NMS ensures that only the most confident and non-overlapping bounding boxes are retained as final object detections.

source

R-CNN - Disadvantages

Disadvantages of R-CNN.

- **Multi-stage, expensive training:** The separate training processes are required for all the stages of the network i.e fine-tuning a CNN on object proposals, learning an SVM to classify the feature vector of each proposal from the CNN and learning a bounding box regressor to fine-tune the object proposals proves to be a burden in terms of time, computation and resources. This multi-stage process can be slow and resource-demanding.
- **Slow Inference**: Due to its sequential processing of region proposals, R-CNN is relatively slow during inference. Real-time applications may find this latency unacceptable. Detection using a simple VGG network as the backbone CNN takes 47s/image.
- **R-CNN is Not End-to-End**: Unlike more modern object detection architectures like Faster R-CNN, R-CNN is not an end-to-end model. It involves separate modules for region proposal and classification, which can lead to suboptimal performance compared to models that optimize both tasks jointly.

SPPNet

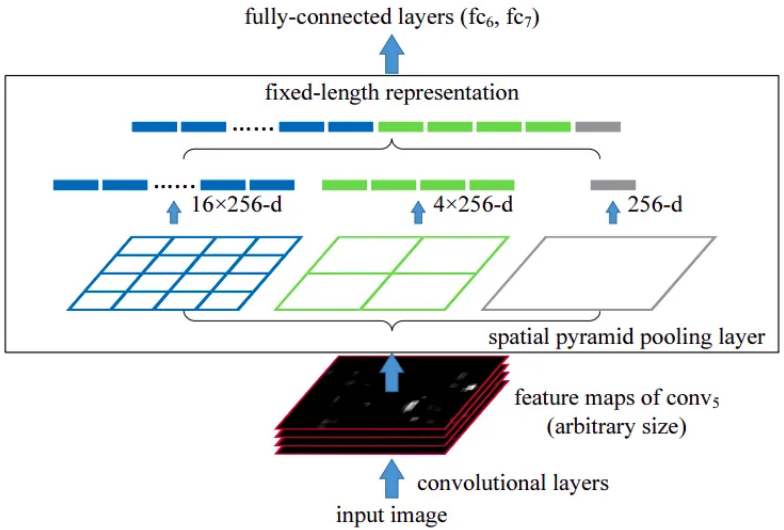
In the paper "Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition", a technique called the **Spatial Pyramid Pooling layer** was introduced, which makes the CNN model agnostic of input image size. This drastically improved the bounding box prediction speed as compared to the R-CNN, without compromising on the mAP.

The fixed size constraint of a CNN network is not because of the convolution layer but because of the Fully Connected (FC) layer. A single convolution layer or a set of convolution layers takes an image and produces a feature map proportional to a particular ratio (called the sub-sampling ratio) w.r.t the input image. But for a fully connected layer, the input has to be a fixed-length vector. To overcome this issue, the authors replaced the last pooling layer (the one just before the FC layer) with a **Spatial Pyramid Pooling(SPP) layer**.

This is done by making the pooling window and stride proportional to the input image, such that a fixed-sized output can always be obtained.

Moreover, the SPP layers do not just apply one pooling operation, it applies a couple of different output sized pooling operations (that's where then same comes from — Spatial Pyramid Pooling) and combines the results before sending them to the next layer.

SPPNet



Considering there are 256 feature maps from the last Conv layer,

1. Each feature map is pooled into 1 value forming a 256-d vector
2. Each feature map is pooled into 4 values forming a 4×256-d vector
3. Each feature map is pooled into 16 values forming a 16×256-d vector

The SPP Layer output is flattened to form a 1-dimensional vector and sent to the FC Layer. This eliminates the cropping of the input image to a fixed size before inputting to a CNN. One can apply the SPP Layer to any CNN architecture

Source

A numerical example..

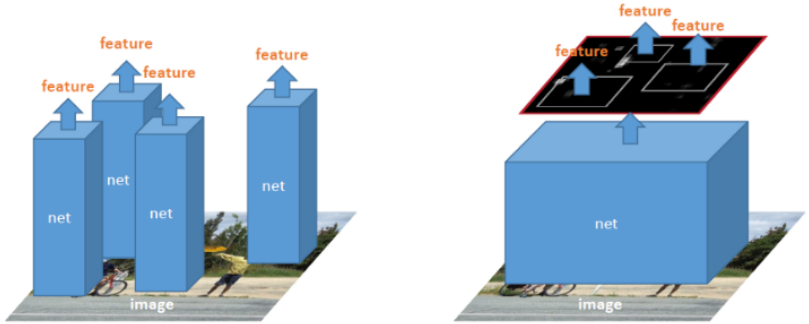
Let the feature map input to the SPP layer be of size [a x a] where a= 13 , we need the output size for single map be [n x n] say n=4 . Thus the kernel size and stride of the pooling layer will be :

- Kernel size = ceiling(a/n) = ceiling(13/4) **= 4**
- stride = floor(a/n) = floor(13/4) **= 3**

Now on a [13 x 13] map, apply the above window([4 x 4]) and stride of 3 , we get the output to be [4 x 4]. This operation is applied to all the feature maps and for 256 feature maps we get an output of [4 x 4 x 256]. So now we can change the [n x n] grid size to get the desired output size.

How does this help object detection?

The authors used the SPP mechanism for object detection in an improved approach. Rather than sending the 2000 region proposals one by one to the CNN model, they projected the regions onto the Feature map obtained from the 5th Conv Layer.
This eliminates 2000 passes the CNN needs to undergo for each image. Now, suddenly from 2000, it is just 1. However, the Selective Search continues to be the bottleneck because it needs to generate 2000 proposals. These regions are sent forward to the SPP Layer for pooling into a 1-d vector. This reduced the computation time to a great extent. The time taken for a test image inference on a GPU was well within 1s and was massively fast in comparison to R-CNN and on par with accuracy too.



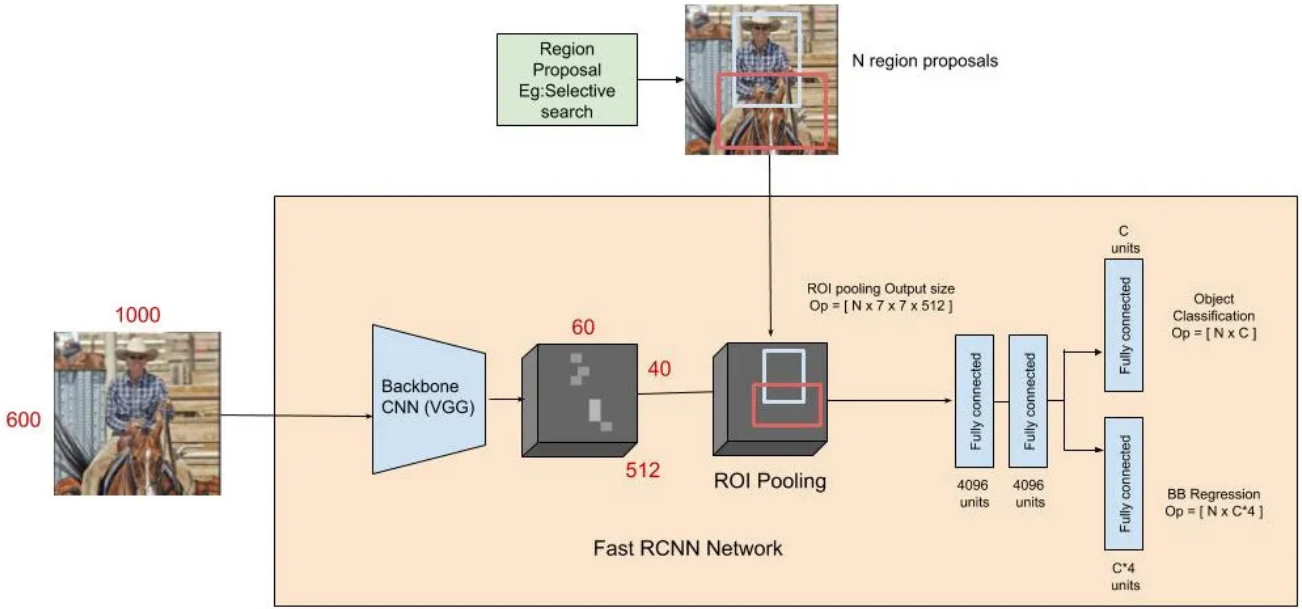
R-CNN vs SPP-net. Image Credits – Towardsdatascience

Source

Fast R-CNN

Published in 2015 one year after the SPPNet paper Fast RCNN is a popular successor to these models improving both efficiency and performance of R-CNN and SPP Networks.

The Fast R-CNN consists of a CNN (usually pre-trained on the ImageNet classification task) with its final pooling layer replaced by an “ROI pooling” layer and its final FC layer is replaced by two branches — a (K + 1) category softmax layer branch and a category-specific bounding box regression branch.



Source

Fast R-CNN Architecture

1. The entire image is fed into the backbone CNN and the features from the last convolution layer are obtained. Depending on the backbone CNN used, the output feature maps are much smaller than the original image size. This depends on the stride of the backbone CNN, which is usually 16 in the case of a VGG backbone.
2. Meanwhile, the object proposal windows are obtained from a region proposal algorithm like selective search.
3. The portion of the backbone feature map that belongs to this window is then fed into the ROI Pooling layer. The ROI pooling layer is a special case of the spatial pyramid pooling (SPP) layer with just one pyramid level. The layer basically divides the features from the selected proposal windows (that come from the region proposal algorithm) into sub-windows of size *h/H* by *w/W* and performs a pooling operation in each of these sub-windows. This gives rise to fixed-size output features of size (H x W) irrespective of the input size. H and W are chosen such that the output is compatible with the network's first fully-connected layer. The chosen values of H and W in the Fast R-CNN paper is 7. Like regular pooling, ROI pooling is carried out in every channel individually.
4. The output features from the ROI Pooling layer (N x 7 x 7 x 512) where N is the number of proposals) are then fed into the successive FC layers, and the softmax and BB-regression branches. The softmax classification branch produces probability values of each ROI belonging to K categories and one catch-all background category. The BB regression branch output is used to make the bounding boxes from the region proposal algorithm more precise.

Faster RCNN

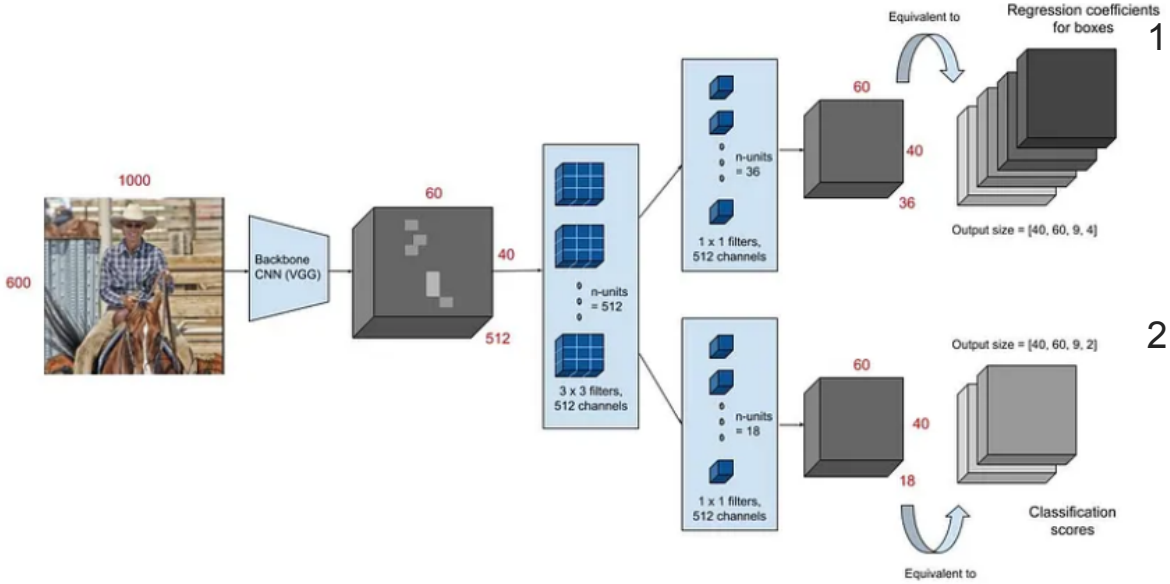
In the R-CNN family of papers, the evolution between versions was usually in terms of computational efficiency (integrating the different training stages), reduction in test time, and improvement in performance (mAP). These networks usually consist of:

1. A region proposal algorithm to generate “bounding boxes” or locations of possible objects in the image.
2. A feature generation stage to obtain features of these objects, usually using a CNN.
3. A classification layer to predict which class this object belongs to; and d) A regression layer to make the coordinates of the object bounding box more precise.

Both R-CNN and Fast R-CNN use CPU based region proposal algorithms, Eg- the Selective search algorithm which takes around 2 seconds per image and runs on CPU computation.
The Faster R-CNN [10] paper fixes this by using another convolutional network (the RPN) to generate the region proposals. This not only brings down the region proposal time from 2s to 10 ms per image but also allows the region proposal stage to share layers with the following detection stages, causing an overall improvement in feature representation.

Source

Region Proposal Network (RPN)



1. The region proposal network (RPN) starts with the input image being fed into the backbone convolutional neural network. The input image is first resized such that it's shortest side is 600px with the longer side not exceeding 1000px.
2. The output features of the backbone network (indicated by H x W) are usually much smaller than the input image depending on the stride of the backbone network. For both the possible backbone networks used in the paper (VGG, ZF-Net) the network stride is 16.

Source

Region Proposal Network (RPN)

3. For every point in the output feature map, the network has to learn whether an object is present in the input image at its corresponding location and estimate its size. This is done by placing a set of “Anchors” on the input image for each location on the output feature map from the backbone network. These anchors indicate possible objects in various sizes and aspect ratios at this location. The figure below shows 9 possible anchors in 3 different aspect ratios and 3 different sizes placed on the input image for a point A on the output feature map. For the PASCAL challenge, the anchors used have 3 scales of box area 128^2, 256^2, 512^2 and 3 aspect ratios of 1:1, 1:2 and 2:1.

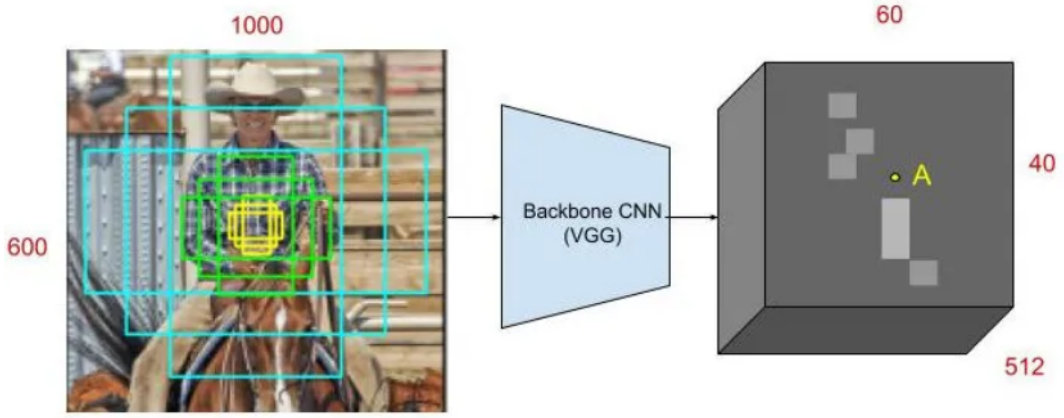
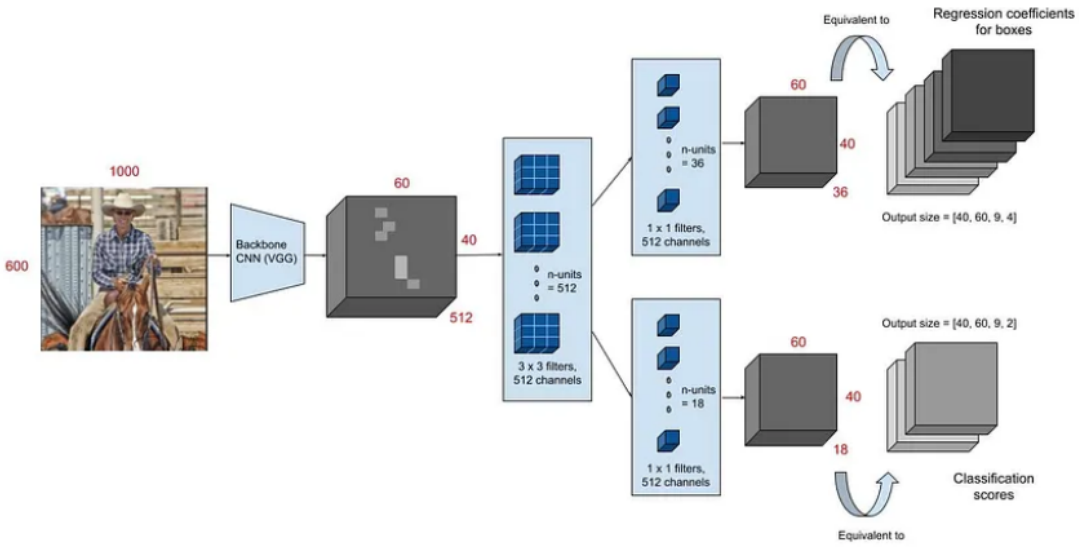


Figure 2: The possible anchors in the input image in a location corresponding to point A in the feature map.

4. As the network moves through each pixel in the output feature map, it has to check whether these *k* corresponding anchors spanning the input image actually contain objects, and refine these anchors’ coordinates to give bounding boxes as “Object proposals” or regions of interest.

Source

Region Proposal Network (RPN)



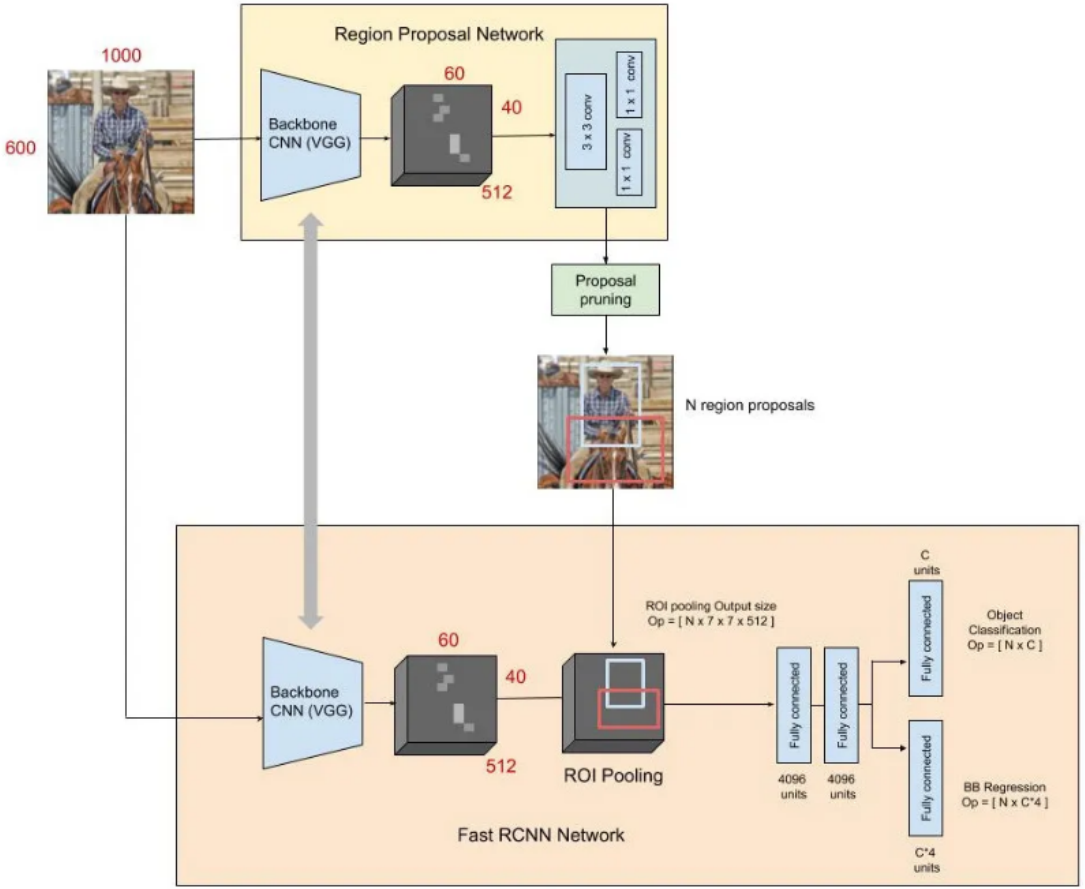
5. First, a 3 x 3 convolution with 512 units is applied to the backbone feature map as shown in Figure , to give a 512-d feature map for every location. This is followed by two sibling layers: a 1 x 1 convolution layer with 18 units for object classification, and a 1 x 1 convolution with 36 units for bounding box regression.

6. The 18 units in the classification branch give an output of size (H, W, 18). This output is used to give probabilities of whether or not each point in the backbone feature map (size: H x W) contains an object within all 9 of the anchors at that point.

7. The 36 units in the regression branch give an output of size (H, W, 36). This output is used to give the 4 regression coefficients of each of the 9 anchors for every point in the backbone feature map (size: H x W). These regression coefficients are used to improve the coordinates of the anchors that contain objects.

Source

Faster R-CNN

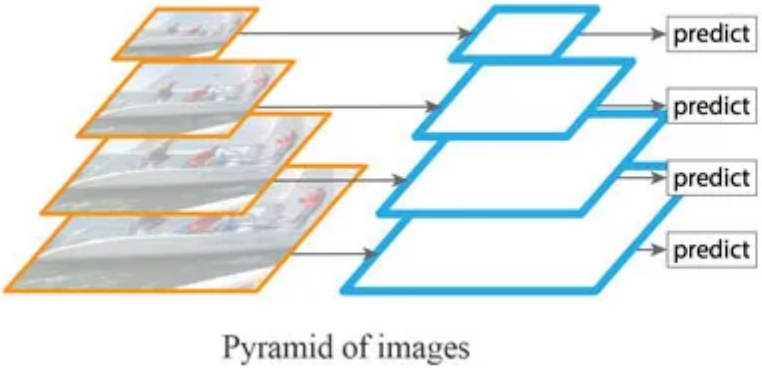


Source

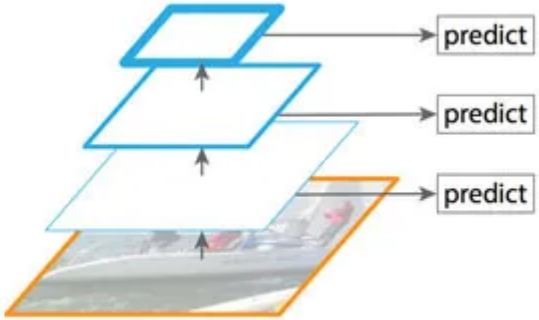
Feature Pyramid Networks (FPN)

Detecting objects in different scales is challenging in particular for small objects. We can use a pyramid of the same image at different scale to detect objects (left diagram below). However, processing multiple scale images is time consuming and the memory demand is too high to be trained end-to-end simultaneously.Alternatively, we create a pyramid of feature and use them for object detection (the right diagram).

Feature Pyramid Network (**FPN**) is a feature extractor designed for such pyramid concept with accuracy and speed in mind. It replaces the feature extractor of detectors like Faster R-CNN and generates multiple feature map layers (**multi-scale feature maps**) with better quality information than the regular feature pyramid for object detection.



Pyramid of images



Pyramid of feature maps

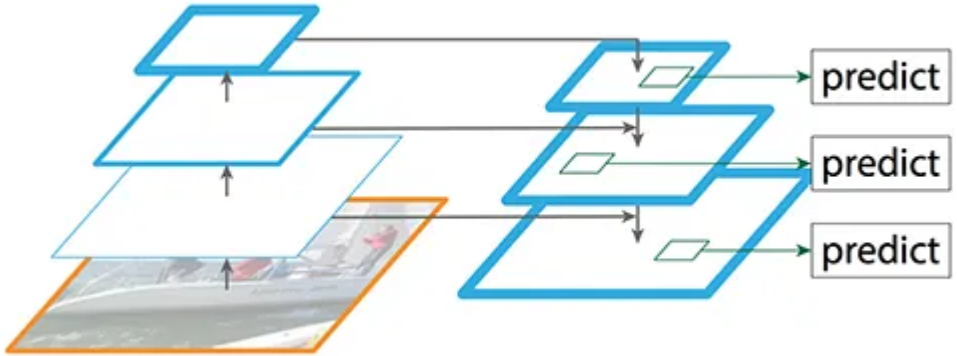
Source

Feature Pyramid Networks (FPN)

- FPN composes of a **bottom-up** and a **top-down** pathway. The bottom-up pathway is the usual convolutional network for feature extraction. As we go up, the spatial resolution decreases. With more high-level structures detected, the **semantic value** for each layer increases.

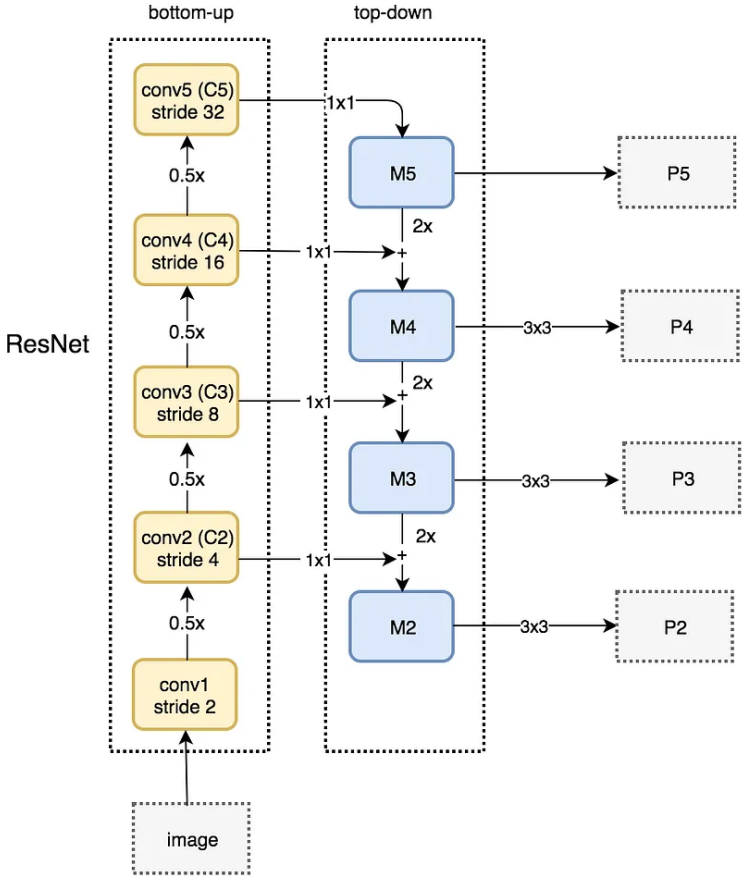
- FPN provides a top-down pathway to construct higher resolution layers from a semantic rich layer.

- While the reconstructed layers are semantic strong but the locations of objects are not precise after all the downsampling and upsampling. We add lateral connections between reconstructed layers and the corresponding feature maps to help the detector to predict the location betters. It also acts as skip connections to make training easier (similar to what ResNet does).



Source

FPN- Data Flow



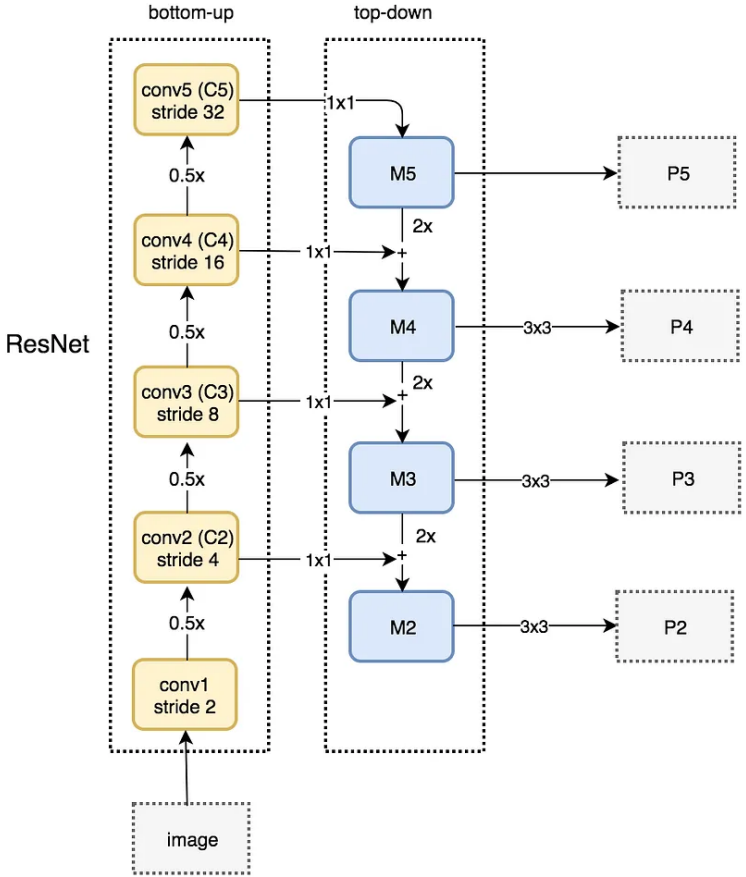
The **bottom-up pathway** uses ResNet to construct the bottom-up pathway. It composes of many convolution modules (conv*i* for i equals 1 to 5) each has many convolution layers. As we move up, the spatial dimension is reduced by 1/2 (i.e. double the stride). The output of each convolution module is labeled as C*i* and later used in the top-down pathway.

We apply a 1 × 1 convolution filter to reduce C5 channel depth to 256-d to create M5. This becomes the first feature map layer used for object prediction.

As we go down the **top-down path**, we upsample the previous layer by 2 using nearest neighbors upsampling. We again apply a 1 × 1 convolution to the corresponding feature maps in the bottom-up pathway. Then we add them element-wise. We apply a 3 × 3 convolution to all merged layers. This filter reduces the aliasing effect when merged with the upsampled layer.

Source

FPN- Data Flow



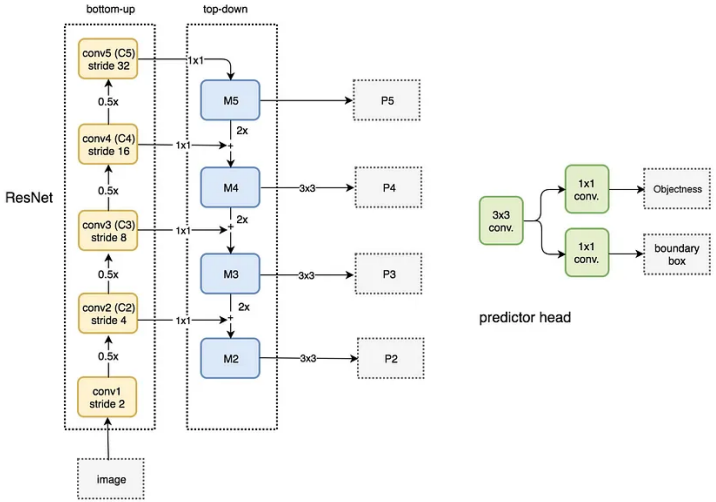
We repeat the same process for P3 and P2. However, we stop at P2 because the spatial dimension of C1 is too large. Otherwise, it will slow down the process too much. Because we share the same classifier and box regressor of every output feature maps, all pyramid feature maps (P5, P4, P3 and P2) have 256-d output channels.

Source

FPN with RPN (Region Proposal Network)

FPN extracts feature maps and later feeds into a detector, says RPN, for object detection. RPN applies a sliding window over the feature maps to make predictions on the objectness (has an object or not) and the object boundary box at each location.

In the FPN framework, for each scale level (say P4), a 3 × 3 convolution filter is applied over the feature maps followed by separate 1 × 1 convolution for objectness predictions and boundary box regression. These 3 × 3 and 1 × 1 convolutional layers are called the RPN **head**. The same head is applied to all different scale levels of feature maps.

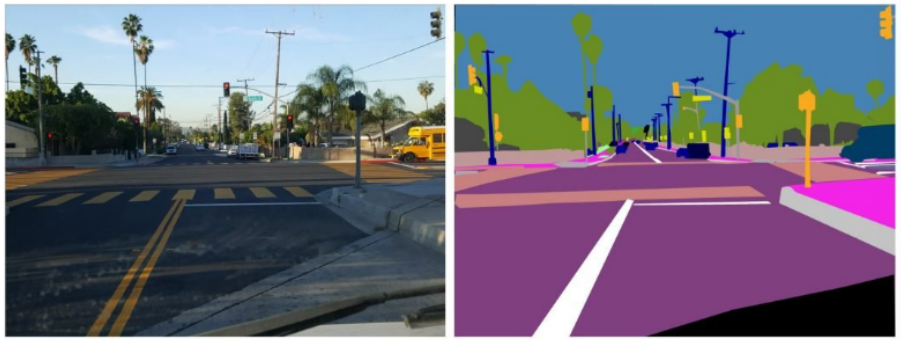


Source

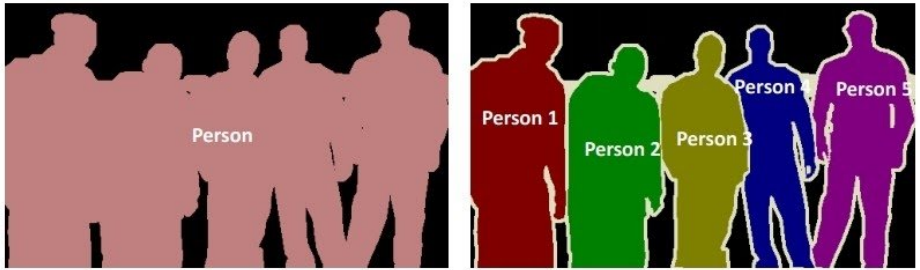
What is Image Segmentation?

One of the most important operations in Computer Vision is Segmentation. Image segmentation is the process of dividing an image into multiple parts or regions that belong to the same class. This task of clustering is based on specific criteria, for example, color or texture. This process is also called pixel-level classification. In other words, it involves partitioning images (or video frames) into multiple segments or objects.
There are 2 main types of image segmentation that fall under Mask R-CNN:

1. Semantic Segmentation: Semantic segmentation classifies each pixel into a fixed set of categories without differentiating object instances.
2. Instance Segmentation: Instance Segmentation deals with the correct detection of all objects in an image while also precisely segmenting each instance.



Annotated image for semantic image segmentation tasks in autonomous driving – Source: Sample from the Mapillary Vistas Dataset



Semantic Segmentation **Instance Segmentation**

Read more at: https://viso.ai/deep-learning/mask-r-cnn/

Mask RCNN

- Mask R-CNN, or Mask RCNN, is a Convolutional Neural Network (CNN) and state-of-the-art in terms of image segmentation and instance segmentation.

- Mask R-CNN was built using Faster R-CNN. While Faster R-CNN has 2 outputs for each candidate object, a class label and a bounding-box offset, Mask R-CNN is the addition of a third branch that outputs the object mask. The additional mask output is distinct from the class and box outputs, requiring the extraction of a much finer spatial layout of an object.

- The key element of Mask R-CNN is the pixel-to-pixel alignment, which is the main missing piece of Fast/ Faster R-CNN. Mask R-CNN adopts the same two-stage procedure with an identical first stage (which is RPN). In the second stage, in parallel to predicting the class and box offset, Mask R-CNN also outputs a binary mask for each RoI. This is in contrast to most recent systems, where classification depends on mask predictions.

- Furthermore, Mask R-CNN is simple to implement and train given the Faster R-CNN framework, which facilitates a wide range of flexible architecture designs. Additionally, the mask branch only adds a small computational overhead, enabling a fast system and rapid experimentation.

Models for Object Detection :

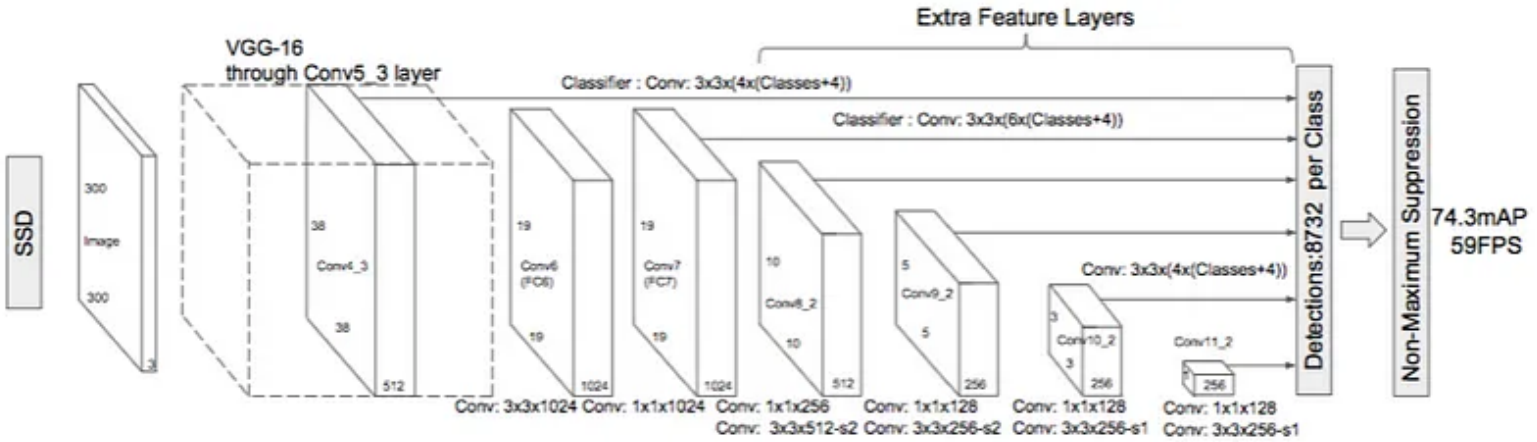
Single shot Detectors

Single-shot object detection is a type of object detection algorithm that is able to detect objects within an image or video in a single pass without the need for multiple stages or region proposals.

Single-shot object detectors, such as YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector), use a single convolutional neural network (CNN) to directly predict the class labels and bounding boxes of objects within an image or video. These models are trained end-to-end using a large dataset of labeled images and their associated object-bounding boxes

SDD

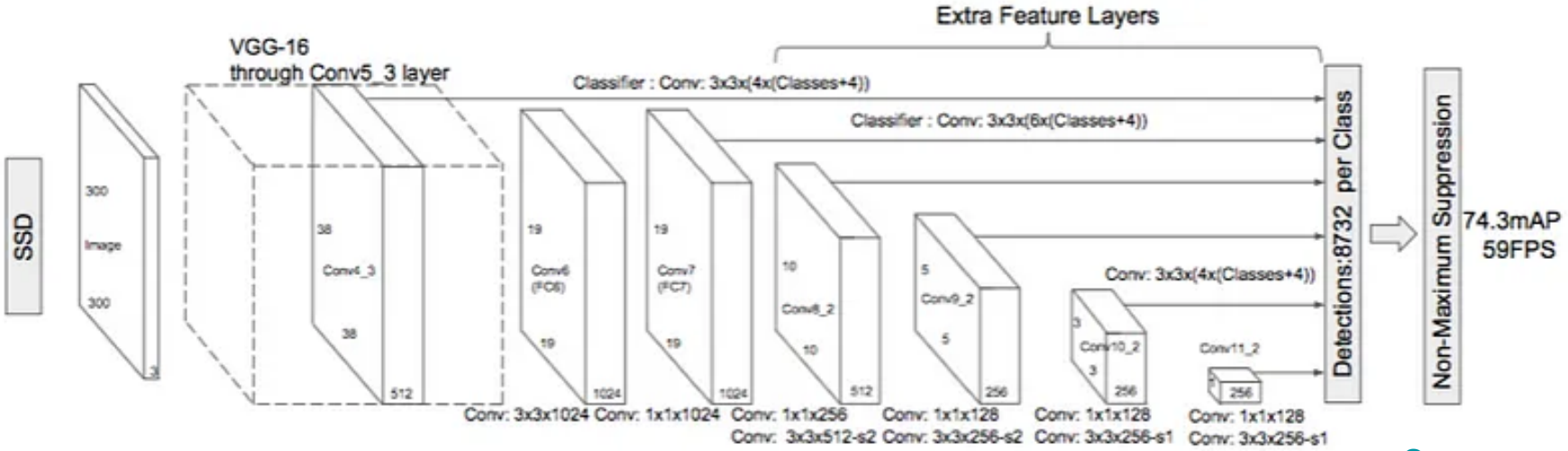
The paper about SSD: Single Shot MultiBox Detector (by C. Szegedy et al.) was released at the end of November 2016 and reached new records in terms of performance and precision for object detection tasks, scoring over 74% mAP (*mean Average Precision*) at 59 frames per second on standard datasets such as Pascal VOC and COCO.



Source

SDD

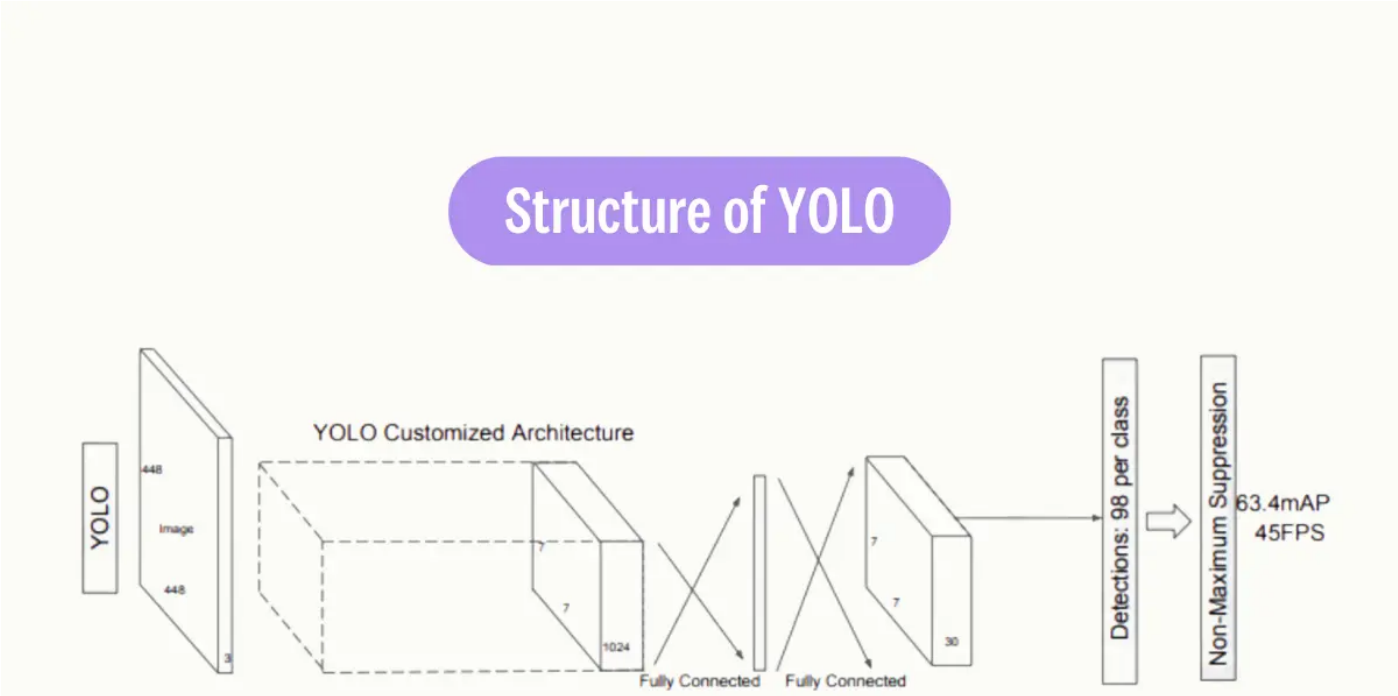
As you can see from the diagram below, SSD's architecture builds on the venerable VGG-16 architecture, but discards the fully connected layers. The reason VGG-16 was used as the *base network* is because of its strong performance in high quality image classification tasks and its popularity for problems where *transfer learning* helps in improving results. Instead of the original VGG fully connected layers, a set of *auxiliary* convolutional layers (from *conv6* onwards) were added, thus enabling to extract features at multiple scales and progressively decrease the size of the input to each subsequent layer.



Source

YOLO v1

YOLO (You Only Look Once) is a real-time object detection algorithm developed by Joseph Redmon and Ali Farhadi in 2015. It is a single-stage object detector that uses a convolutional neural network (CNN) to predict the bounding boxes and class probabilities of objects in input images.



Source

YOLO Algorithm

The basic idea behind YOLO is to divide the input image into a grid of cells and, for each cell, predict the probability of the presence of an object and the bounding box coordinates of the object. The process of YOLO can be broken down into several steps:

1. Input image is passed through a CNN to extract features from the image.

2. The features are then passed through a series of fully connected layers, which predict class probabilities and bounding box coordinates.

3. The image is divided into a grid of cells, and each cell is responsible for predicting a set of bounding boxes and class probabilities.

4. The output of the network is a set of bounding boxes and class probabilities for each cell.

5. The bounding boxes are then filtered using a post-processing algorithm called non-max suppression to remove overlapping boxes and choose the box with the highest probability.

6. The final output is a set of predicted bounding boxes and class labels for each object in the image.

One of the key advantages of YOLO is that it processes the entire image in one pass, making it faster and more efficient than two-stage object detectors such as R-CNN and its variants.

Source

YOLO v2

- One of the main differences between YOLO v2 and the original YOLO is the use of anchor boxes. In YOLO v2, CNN predicts not only the bounding box coordinates but also the anchor boxes. Anchor boxes are pre-defined boxes of different aspect ratios and scales, which are used to match the predicted bounding boxes with the actual objects in the image. This allows YOLO v2 to handle objects of different shapes and sizes better.
- Another key difference is the use of a multi-scale approach. In YOLO v2, the input image is fed through CNN at multiple scales, which allows the model to detect objects at different sizes. This is achieved by using a feature pyramid network (FPN), which allows the model to extract features at different scales from the same image.
- Additionally, YOLO v2 uses a different loss function than the original YOLO, called the sum-squared error (SSE) loss function. The SSE loss function is more robust and helps the model to converge faster.
- In terms of architecture, YOLO v2 uses a slightly deeper CNN than YOLO, which allows it to extract more powerful features from the image. The CNN is followed by several fully connected layers, which predict class probabilities and bounding box coordinates.

Source

YOLO v3

- YOLO v3 is the third version of the YOLO object detection algorithm. The first difference between YOLO v3 and previous versions is the use of multiple scales in the input image. YOLO v3 uses a technique called "feature pyramid network" (FPN) to extract features from the image at different scales. This allows the model to detect objects of different sizes in the image.
- Another important difference is the use of anchor boxes. In YOLO v3, anchor boxes are used to match the predicted bounding boxes with the actual objects in the image. Anchor boxes are pre-defined boxes of different aspect ratios and scales, and the model predicts the offset of the anchor boxes relative to the bounding boxes. This helps the model to handle objects of different shapes and sizes better.
- In terms of architecture, YOLO v3 is built on a deep convolutional neural network (CNN) that is composed of many layers of filters. The CNN is followed by several fully connected layers, which predict class probabilities and bounding box coordinates.
- YOLO v3 also uses a different loss function than previous versions. It uses a combination of classification loss and localization loss, which allows the model to learn both the class probabilities and the bounding box coordinates.

Source

YOLO v4

- A key distinction between YOLO v4 and previous versions is using a more advanced neural network architecture. YOLO v4 uses a technique called "Spatial Pyramid Processing" (SPP) to extract features from the image at different scales and resolutions. This allows the model to detect objects of different sizes in the image.

- Additionally, YOLO v4 also uses a technique called "Cross-stage partial connection" (CSP) to improve the model's accuracy. It uses a combination of multiple models with different architectures and scales and combines their predictions to achieve better accuracy.

Source

YOLO v5

- YOLO v5 was introduced in 2020 with a key difference from the previous versions, which is the use of a more efficient neural network architecture called EfficientDet, which is based on the EfficientNet architecture. EfficientDet is a family of image classification models that have achieved state-of-the-art performance on a number of benchmark datasets. The EfficientDet architecture is designed to be efficient in terms of computation and memory usage while also achieving high accuracy.

- Another important difference is the use of anchor-free detection, which eliminates the need for anchor boxes used in previous versions of YOLO. Instead of anchor boxes, YOLO v5 uses a single convolutional layer to predict the bounding box coordinates directly, which allows the model to be more flexible and adaptable to different object shapes and sizes.

- YOLO v5 also uses a technique called "Cross mini-batch normalization" (CmBN) to improve the model's accuracy. CmBN is a variant of the standard batch normalization technique that is used to normalize the activations of the neural network.

Source

YOLO v6

- A notable contrast between YOLO v6 and previous versions is the use of a more efficient and lightweight neural network architecture; this allows YOLO v6 to run faster and with fewer computational resources. The architecture of YOLO v6 is based on the "Efficient Net-Lite" family, which is a set of lightweight models that can be run on various devices with limited computational resources.

- YOLO v6 also incorporates data augmentation techniques to improve the robustness and generalization of the model. This is done by applying random transformations to the input images during training, such as rotation, scaling, and flipping.

Source

YOLO v7

- YOLO v7 boasts a number of enhancements compared to previous versions. A key enhancement is the implementation of anchor boxes. These anchor boxes, which come in various aspect ratios, are utilized to identify objects of various shapes. The use of nine anchor boxes in YOLO v7 enables it to detect a wider range of object shapes and sizes, leading to a decrease in false positives.

- In YOLO v7, a new loss function called "focal loss" is implemented to enhance performance. Unlike the standard cross-entropy loss function used in previous versions of YOLO, focal loss addresses the difficulty in detecting small objects by adjusting the weight of the loss on well-classified examples and placing more emphasis on challenging examples to detect.

Source

Conclusion

This lecture aims to summarize the most popular CNN architectures for object detection , however there are a plethora of other architectures which may or may not be CNN based which the students may wish to explore . A comprehensive list is provided in the research papers listed in reference [1],[2] and [3].

Reading the research papers cited in the references section is also encouraged.

References

[1]Z. Zou, K. Chen, Z. Shi, Y. Guo and J. Ye, "Object Detection in 20 Years: A Survey," in Proceedings of the IEEE, vol. 111, no. 3, pp. 257-276, March 2023, doi: 10.1109/JPROC.2023.3238524.
keywords: {Object detection;Detectors;Computer vision;Feature extraction;Deep learning;Convolutional neural networks;Computer vision;convolutional neural networks (CNNs);deep learning;object detection;technical evolution}

[2]Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, 'Object Detection with Deep Learning: A Review', *arXiv [cs.CV]*. 2019.

[3]X. Wu, D. Sahoo, and S. C. H. Hoi, 'Recent Advances in Deep Learning for Object Detection', *arXiv [cs.CV]*. 2019.

[4] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in CVPR, vol. 1. IEEE, 2001, pp. I–I.

[5] P. Viola and M. J. Jones, "Robust real-time face detection," International journal of computer vision, vol. 57, no. 2, pp. 137–154, 2004.

[6] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in CVPR, vol. 1. IEEE, 2005, pp. 886–893.

[7]P. Felzenszwalb, D. McAllester, and D. Ramanan, "A discriminatively trained, multiscale, deformable part model," in CVPR. IEEE, 2008, pp. 1–8.

[8]R. Girshick, J. Donahue, T. Darrell, and J. Malik, 'Rich feature hierarchies for accurate object detection and semantic segmentation', *arXiv [cs.CV]*. 2014.

[9]Uijlings, J. R. R. et al. "Selective Search for Object Recognition." International Journal of Computer Vision 104.2 (2013)

[10]Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks, NIPS'15 Proceedings

UE21CS343BB2

Topics in Deep Learning

Dr. Shylaja S S
Director of Cloud Computing & Big Data (CCBD), Centre for Data Sciences & Applied Machine Learning (CDSAML)
Department of Computer Science and Engineering
shylaja.sharath@pes.edu

Ack:Anashua Krittika Dastidar, Teaching Assistant