Object Detection

Halil Eralp Koçaş

February 2, 2020

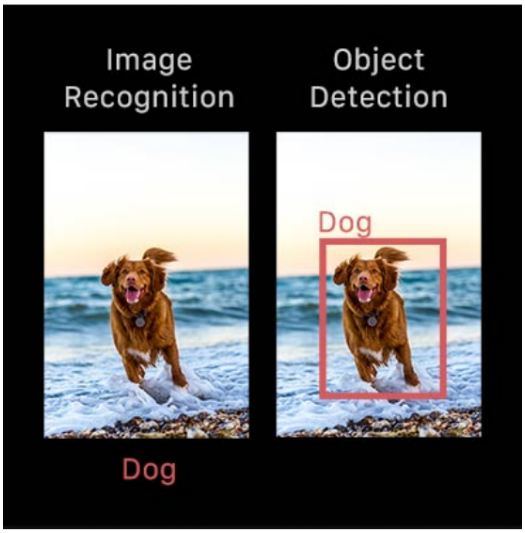

Figure 1: Difference between recognition and detection

1 Object Detection

Object detection is a concept of computer vision in which trained models detect objects with their category and location in given images or videos. As you can see in figure 1, object detection is both classifying and localizing the object instances in given images.

1.1 Motivation

Various models are developed through years in the field of object detection. All these models are developed using different methods. Although all of these models are based on convolutional neural networks, their way of identifying objects in given images or frames are varies. Also, their backbone networks, scanning methods, multi-scale object detection, error functions vary. These varieties affect the performance of models in different aspects such as small and large object detections, speed of the model, etc.

This article aims to investigate, analyze, and compare the performances of various state-of-the-art object detectors through the time trained on video data. Then, promising detectors will be selected and these selected detectors will be analyzed on Video Object Detection dataset on both mean average precision and frame rate per second basis.

1.2 Literature in Static Object Detection

Static object detection refers to object detection in a single image. In this context, object detectors do not use temporal information about objects in given image or frame sequences. Each given images are considered separately. The aim of investigating static object detection is to make a generalization from single frame to multiple frames. Understanding the pros and cons of each object detector in various static object detection cases are important to generalize and create a baseline for video object detection which requires more features to be considered for a well-performing detector.

2 Detector Features

As mentioned in the Motivation section, in this article, the aim is to analyze object detectors based on various detector features. In the following five subsections, the following features will be examined:

- Backbone Networks
- Scanning Methods
- Multi-scale Handling
- Loss Functions

These listed features are the essential features for detectors. When one studies the object detectors, one can see that major increases in the performances of object detectors are caused by changes in these features. These features will be analyzed and shown in the following subsections of this section.

2.1 Backbone Networks

Backbone networks are the initial part of a detector's architecture. The given frame is processed for the first time in backbone networks. These backbone networks are implemented as Fully Convolutional Neural Networks. The used networks are the ones that are already proven to perform well in images. The most used backbone networks are VGG, ResNet, DarkNet, and Hourglass networks.

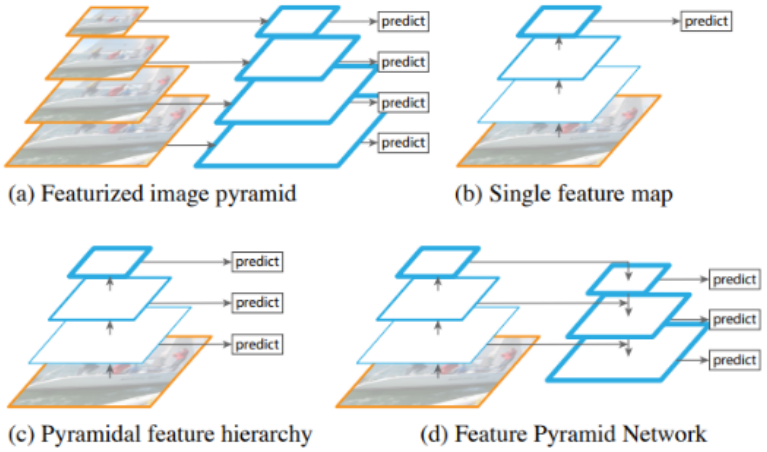

Figure 2: Different methods to use feature maps

The main function of these networks is to compute convolutional feature maps over the given frame, so that, these extracted feature maps are used to detect and localize objects in given frames.

2.2 Scanning Methods

There are many methods developed to scan given images through the years. The most used scanning methods are Sliding Windows, Region Proposal, Grid Cells, and Anchor Boxes.

All these methods have different advantages against each other. Also, Anchor Boxes are used together with other methods.

2.3 Multi-scale Handling

Multi-scale handling refers to detect objects that have different sizes in a given frame. Multi-scale handling is a crucial feature for having a well performing detector since most of the objects in images have different sizes. There are various ways to handle this problem:

One of the solutions (Fig. 2(a)) is to run detection multiple times that in each run, the resolution of the input frame has to be changed. Then, all the detected objects have to be combined after all iterations are completed. In the case of multiple detections for the same object, a suppression has to

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Figure 3: Smooth L1 Loss Function

be performed to reduce single detection. Although this method works well, its runtime is slow.

Another solution (Fig. 2(b)) is to use single feature map. In this method, a single feature map is extracted from given frame and this feature map is passed through multiple convolutional layers to obtain a final feature map with more fine-grained features. Then, this feature map is used to predict the objects in the given frame. This method is used to obtain fast detection but its performance is relatively worse than other methods.

Another solution (Fig. 2(c)) is to use a pyramidal feature hierarchy in which multiple feature maps are used to make a prediction. In this method, a feature map is extracted from the given frame and as it is in the second solution, this feature map is passed through multiple convolutional layers. However, in each layer, the extracted feature map is used to make a prediction. This method is relatively slower than the second solution but it performs better.

Another solution (Fig. 2(d)) is to use feature pyramid network but this method will be examined in section 3.2.6.

2.4 Loss Functions

Loss functions are used to measure the difference between ground-truth values and the predicted values in machine learning problems. Loss functions can also be called error functions. Thus, the aim is to minimize the value of loss function during training to have a better detector. In object detection problems, loss functions are composed of two parts: regression difference and classification difference. Loss functions are the measure of the difference between predicted regression box and ground-truth regression box and also, the difference between the correct class for object and the predicted class for the object. These functions can vary based on design choices and different loss functions have different advantages and disadvantages. The most frequent used loss functions as following:

- L1 Loss Function

  L1 loss function minimizes the absolute differences between the predicted value and ground-truth value.

$$\sum_{i=1}^{n}(|y_t - y_p|)$$

- L2 Loss Function

  L2 loss function minimizes the squared differences between the predicted value and ground-truth value.

$$\sum_{i=1}^{n}(y_t - y_i)^2$$

- Smooth L1 Loss Function

  The function of the smooth L1 loss function can be seen in figure 3.

- Cross-Entropy Loss Function

$$\sum_{i=1}^{n}(y_t * \log(y_i))$$

Since the L1 loss function operates on the absolute values, it does not affect by outliers. Therefore, L1 is more robust to outliers. On the other hand, L2 loss function operates on the squared values, therefore, L2 is not robust to outliers since outliers will cause a huge error value. However, if you analyze and see there are not many outliers in data, then, using the L2 loss function leads to better training. Smooth L1 loss function is a combination of L1 and L2 loss functions. When absolute value of loss is small, it behaves similar to L2 loss but when absolute value of loss is high, it behaves similar to L1 loss. Thus, it shares the advantages of both loss functions. Cross-entropy loss function measures the performance of classification.

3 Static Detector Types

In this section, state-of-the-art object detectors will be examined. The following detectors are the best-performing detectors in their publication

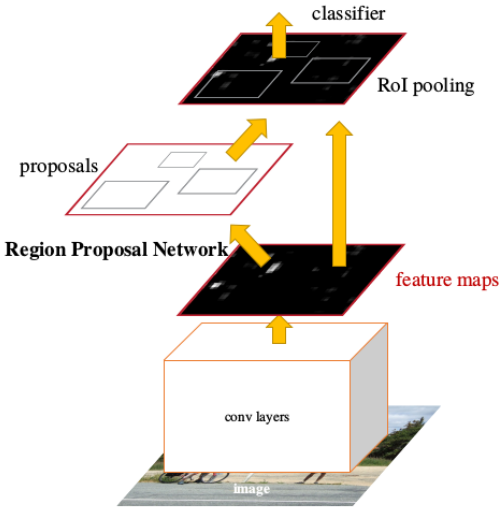

Figure 4: Network of Faster R-CNN

time. These detectors are divided into two subsections that are two-stage detectors and one-stage detectors. The difference between these one-stage and two-stage detectors is their way to process a given frame to detect objects.

In two-stage detectors, the detections process takes two stages that are region proposal and object detection. In the first phase, the detector generates a region of interest that are the regions in which objects can be found most likely. Then, in the second phase, a classifier processes these regions to detect objects.

In one-stage detectors, the detection process takes only one stage. The detector runs over a dense sampling of possible locations in the given frame. This approach converges faster but its performance might be worse than two-stage detectors.

3.1 Two-Stage Detectors

3.1.1 Faster R-CNN

Faster R-CNN [11] consists of a region proposal network and a detection network in which network is a single, unified network. The main improvement in Faster R-CNN is the shared convolutional layers between

the RPN and the detection network as you can see in figure 4, so that, the cost of detection is drastically reduced.

The RPN provides a simultaneous prediction of object bounds and objectness scores. This is done by adding two convolutional layers after the shared convolutional layers. The first one converts extracted feature map into a feature vector and the second one generates an objectness score and regressed bounds for let's assume k region proposals as in figure 5. To generate region proposals, a small convolutional network slides over the shared feature map and each of these sliding windows is mapped to a lower-dimensional feature.

These k region proposals are fed into the detection network with the extraced feature map from shared convolutional layers. Then, the region of interest pooling extract region proposals from the extracted feature map. Predictions are made by using this final feature map. Thus, RPN says the detector network where to look.

Training of Faster R-CNN consists of four steps. First, RPN is trained. Since negative samples dominate positive samples in region proposals, 256 samples are sampled to train RPN by using mini-batches from a single image. Then, in the second step, Fast R-CNN is trained separately by using the region proposals generated by the RPN in step 1. So far, two networks are separate and they do not share convolutional layers. As a step 3, the shared convolutional layers are held fix and RPN is fine-tuned by using the detector network. As a final step, the layers of the detection network is fine-tuned by holding the shared convolutional layers fixed.

3.2 One-Stage Detectors

3.2.1 You Only Look Once: Unified, Real-Time Object Detection

You Only Look Once (YOLO) [10] refers to unified, real-time object detection. YOLO is called unified since it performs simultaneous prediction overall image using its fixed number of bounding boxes in each grid cell. Also, the whole detection pipeline is a single network, thus YOLO can be optimized end-to-end directly from a given image to its detection performance.

The process of prediction can be examined as follows:

1. The given image is split into the SxS grid.

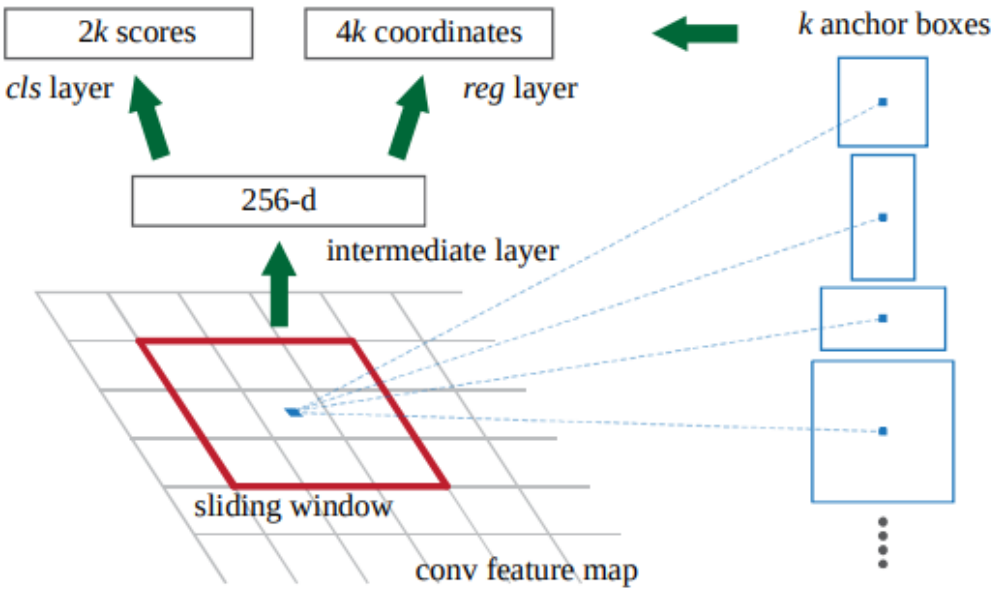

Figure 5: An example of RPN

- Each grid cell is responsible for the prediction of one object. The selection of which cell is responsible is chosen by the center of the object. If an object's center falls into a cell, that cell is responsible to predict that object.

2. Each grid cell has B bounding boxes to make a prediction.

   - One prediction for each grid cell, independent of the number of bounding boxes, causes YOLO to predict a limited number of objects when there are multiple objects present in a grid cell.

3. Thus, YOLO predicts B bounding boxes and one confidence score for each grid cell. Now, let's get into more detail for these predictions:

   - For each of these bounding boxes, five values are predicted: x, y, w, h, and confidence score. The values x and y are the corresponding object's center coordinates on the given image. The values w and h are corresponding object's width and height. The confidence score refers to detector's confidence whether an object exists in

the corresponding bounding box and how accurate the bounding box is.

- The values x and y are normalized relative to the cell they belong and the values w and h are normalized relative to the given image, so that, all these x, y, w, and h values are between 0 and 1. Also, a class conditional probability for each grid cell is predicted. Thus, each category in the possible class set has one class conditional probability in each grid cell.
- Thus, for SxS grid and B bounding boxes for each grid cell, predictions are encoded as S x S x (B * 5 + C) tensor.

3.2.2 YOLO9000: Better, Faster, Stronger

Before introducing YOLO9000, one has to explain YOLOv2 since YOLOv2 is the improved version of YOLO in detection performance and speed while keeping the real-time speed of detection. YOLO9000 refers to a joint training method on object detection and classification. Using this joint training method, one can detect objects that do not have any labelled detection data. Now, YOLOv2 will be explained and later, YOLO9000 will be explained.

As the name of the article suggested, YOLOv2 will be explained in three different perspectives: Better, faster, stronger [9]. While better and faster refer to a performance increase of YOLO which is YOLOv2, stronger refers to YOLO9000.

1. Better

    - Analyzing the sufferings of YOLO, one can see that YOLO suffers from localization and having a low recall rate. Thus, the aim is to improve recall and localization while maintaining classification accuracy.

    (a) Batch Normalization

        - Applying batch normalization leads to significant improvement in performance and also it reduces the need for other forms of regularizations. One can remove dropout from the model without overfitting.
        - Batch normalization leads to a 2% increase in mAP.

(b) High-Resolution Classifier

- Training of YOLO occurs in two phases: first, convolutional layers are trained on ImageNet classification task on image resolution of 224 x 224. Then, the convolutional layers are trained for detection on image resolution of 448 x 448. This causes the detector to switch itself for learning detection for new resolution simultaneously.
- In YOLOv2, before the detector is switched for learning detection, the classifier network is trained for 448 x 448 resolution images for 10 epochs on ImageNet. Thus, a time is provided for the network to adjust itself to work better on high resolution.
- High-Resolution Classifier leads to a 4% increase in mAP.

(c) Convolutional with Anchor Boxes

- The Fully-connected layers in the architecture of YOLO are removed and instead of fully-connected layers, anchor boxes are used to predict bounding boxes. Input resolution is decreased from 448x448 to 416x416 to have a single center feature map: 13x13 feature map. In YOLO, predictions are made with 98 bounding boxes, however, in YOLOv2, predictions are made with more than a thousand bounding boxes. Thus, the mean average precision is decreased from 69.5 to 69.2 but the recall is increased from 81% to 88%. This tradeoff is worth to do.
- There are two issues using anchor boxes with YOLO. The first one is that box dimensions are hand picked which can lead the network to learn hard. If initialization may be done better, the network can learn better. The other one is model instability especially in early iterations of training. Unconstrained anchor boxes can cause a box to appear anywhere in the given image.

(d) Dimension Cluster

- The dimension cluster is developed to handle hand-picked dimensions of anchor boxes. To find good anchor boxes for the network, k-means clustering is run over training set bounding boxes.

(e) Direct Location Prediction

- Direct Location Prediction is developed to handle model instability. The anchor boxes are constrained as bounding boxes in YOLO. Thus, predictions are made relative to the location of the grid cell each anchor boxes belong. This bounds the predicted values 0 and 1. Also, using offsets from the top-left of the image, the correct and constrained locations can be found.
- Using dimension cluster with direct location prediction leads to a 5% increase over the version with anchor boxes.

(f) Fine-grained Features

- As mentioned above, YOLOv2 predicts with 13x13 feature maps. While this is enough for large objects, it may perform worse in small objects. To handle this problem, a passthrough layer is connected from a 26x26 feature map to a 13x13 feature map. By reshaping 26x26 map to 13x13 map and concatenating them leads to 1% performance increase.

(g) Multi-scale Training

- Multi-scale training aims to make the detector more robust to running images of different sizes. Thus, the detector chooses a new image dimension size in every 10 epochs during training from a list of sizes {320, 352, ..., 608}.
- There is a tradeoff between speed and accuracy since the detector performs faster when image size is smaller but accuracy is worse when image size is smaller.

2. Faster

The main motivation of YOLO is to design a detector that performs real-time object detection while maintaining the detection accuracy as high as possible. To increase detection performance, some features of the detector have to be more complex but designing a more complex architecture causes slower detection speed. To create space for more complex features, in YOLOv2, speed of the backbone network is increased as follows:

- VGG16 requires 30.69 billion floating-point operations for a single pass over a single image at 224 x 224 resolution [9].

- YOLO framework uses a custom backbone network based on GoogleNet architecture. This network uses 8.52 billion operations, however, it works slightly worse than VGG16.
- Thus, in YOLOv2, a new classification model is used as a backbone network which is called DarkNet19. This network requires 5.58 billion of operations.
- Their accuracy on ImageNet as follows:
  (a) VGG16: 90.0% top-5 accuracy.
  (b) YOLO: 88% top-5 accuracy.
  (c) DarkNet19: 91.2% top-5 accuracy.

3. Stronger

   As mentioned before YOLO9000 is a joint training method on classification and detection data. Detection dataset is used to learn information required to detect such as bounding box coordinate prediction, objectness, etc. Classification dataset is used to expand the number of category detector can detect. During training, detection and classification datasets are used together. When an image is labeled for detection, then, backpropagation can be done overall architecture. Yet, if an image is labeled for classification, then, backpropagation can be done over classification parts of the architecture. The problem in this approach is detection datasets have common labels such as dog, cat, etc but classification datasets have more specific labels such as Norfolk terrier, Yorkshire terrier, etc. WordNet is used to pull labels. WordNet is a language database that structures and relate concepts. As you can see in figure 6, WordTree is used to combine detection (COCO) and classification (ImageNet) datasets. Conditional probabilities are used to predict classifications. An example can be seen in figure 7.

3.2.3 SSD: Single Shot MultiBox Detector

Single Shot MultiBox Detector [7] is a single deep neural network which aims to detect objects in real-time. The main improvement in speed comes from eliminating region proposals as Faster R-CNN does. Although eliminating region proposals increases the speed of detection, it reduces the accuracy significantly. Thus, three main improvements are applied to increase accuracy:

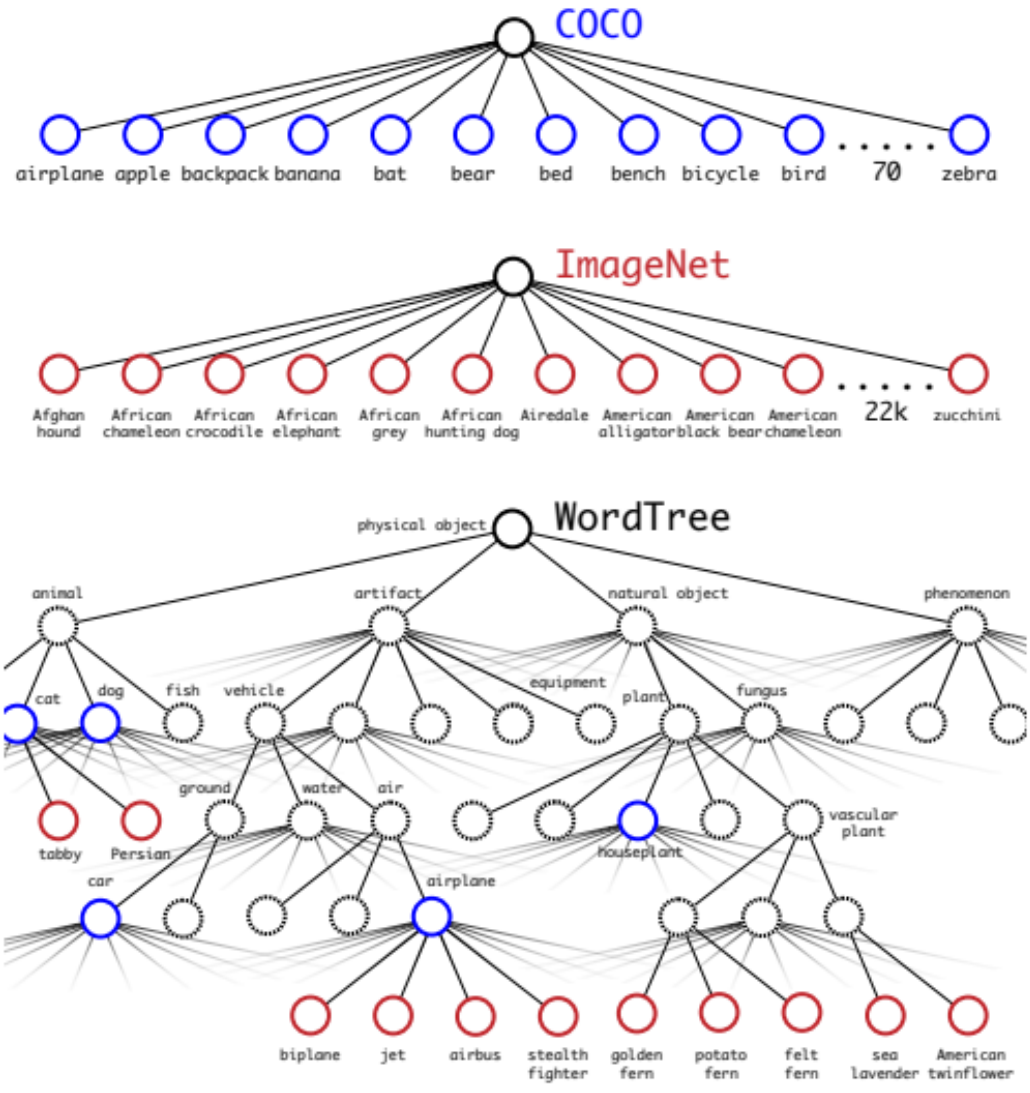

Figure 6: Combining ImageNet and COCO using WordTree hierarchy

$$
\begin{aligned}
Pr(\text{Norfolk terrier}) &= Pr(\text{Norfolk terrier}|\text{terrier}) \\
&\quad * Pr(\text{terrier}|\text{hunting dog}) \\
&\quad\quad * \ldots * \\
&\quad * Pr(\text{mammal}|Pr(\text{animal}) \\
&\quad * Pr(\text{animal}|\text{physical object})
\end{aligned}
$$

Figure 7: Classification calculation example for WordTree

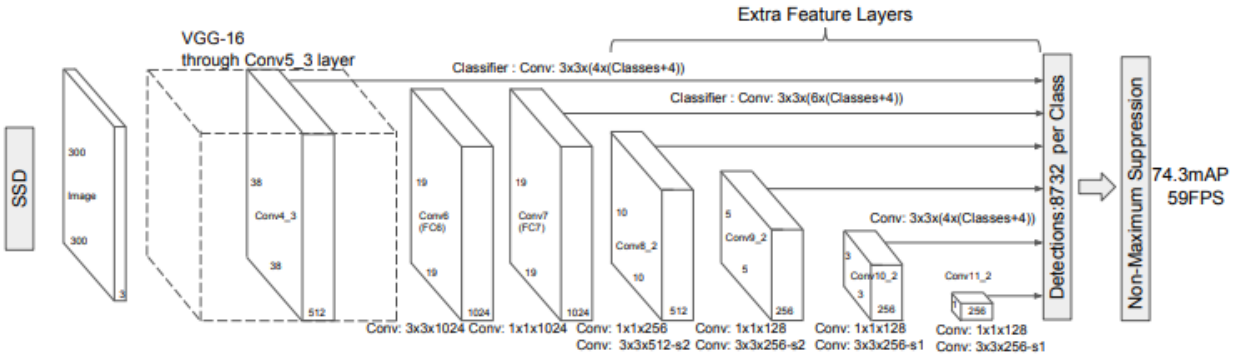

Figure 8: Network of SSD

1. Multi-Scale Feature Maps for Detection

    - As mentioned in section 2.3, handling multi-scale is an important feature for object detectors. It can reduce the mAP for detector's performance since objects of different sizes in a given image cannot be detected well. To handle multi-scale detection, SSD uses multi-scale feature maps instead of using different sizes of input images. The size of feature maps decreases through the network's architecture. You can see in figure 8 the SSD architecture in which layers are getting smaller from left to right and each of these convolutional layers are connected to the prediction module, so that, multi-scale feature maps can be used in prediction. For larger feature maps, SSD can detect smaller objects and for smaller feature maps, smaller objects can be detected. You can see in figure 9, detecting the cat in the image is done by an 8x8 feature map, although, detecting the dog is done by a 4x4 feature map.

2. Convolutional Predictors for Detection

    - Small-size convolutional filters are used to perform object detection. These convolutional filters are applied to extracted feature maps to compute both localization and class scores. Each filter computes a bounding box and corresponding class scores for each category.

3. Default Bounding Boxes and Aspect Ratios

    - SSD divides its feature maps into a grid and each feature map cells are associated with a set of default bounding boxes. These bounding boxes have a fixed position relative to its corresponding grid cell. The aim of using multiple bounding boxes in each grid cell is to detect different-shape objects such as cars and people. You can see in figure 10, there are four different bounding boxes for each grid cells. These bounding boxes are shaped differently than the others to increase the chance of detection of different-shaped objects. For example, bounding box 1 in figure 10 can be used to detect cars but bounding box 3 is a better option for detecting people.

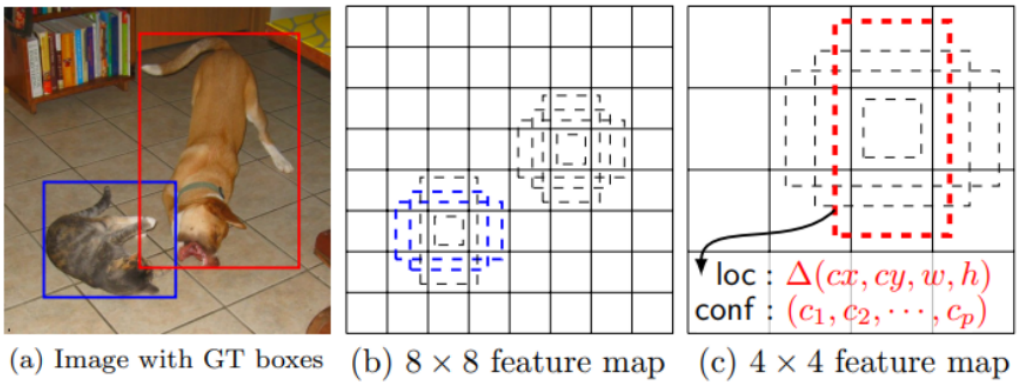

Figure 9: Multi-scale feature maps

- For different layers of SSD, default bounding boxes are customized for different resolutions. There are target aspect ratios and corresponding to these aspect ratios, default bounding boxes are calculated.

3.2.4 DSSD: Deconvolutional Single Shot Detector

Deconvolutional Single Shot Detector [2] aims to contribute a new approach for object detection. The major changes in DSSD are to change the base network from VGG16 to ResNet101, using prediction modules, and adding deconvolution layers after convolutional layers of SSD. In figure 11, you can see the DSSD layers and the way they are connected with the layer before and corresponding size SSD layer. Also, the change in the prediction layer from SSD to DSSD can be seen. So, these changes will be examined as follows:

1. ResNet101

   - The aim of changing the base network from VGG16 to ResNet101 is to increase accuracy. However, it does not improve accuracy by itself. That's why the prediction module is used to increase performance.

2. Prediction Module

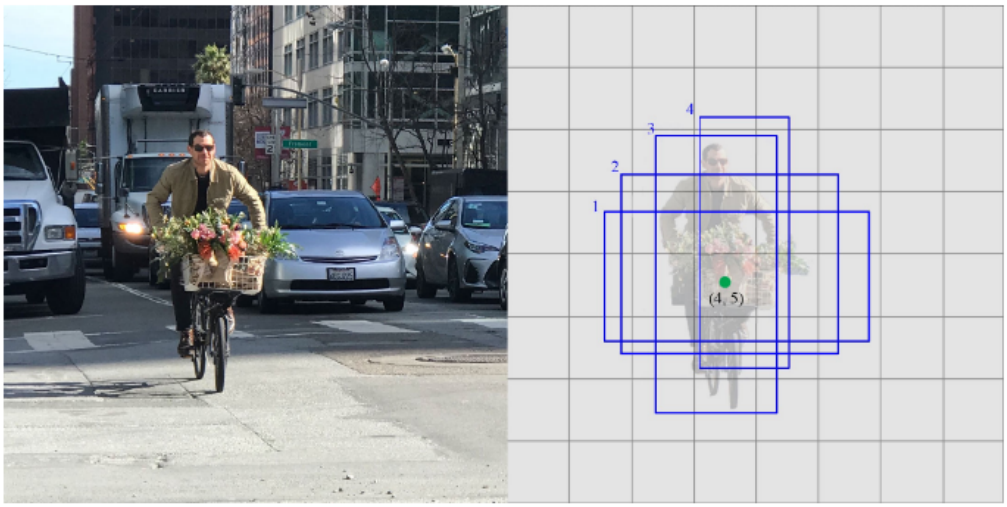

Figure 10: Default Bounding Boxes

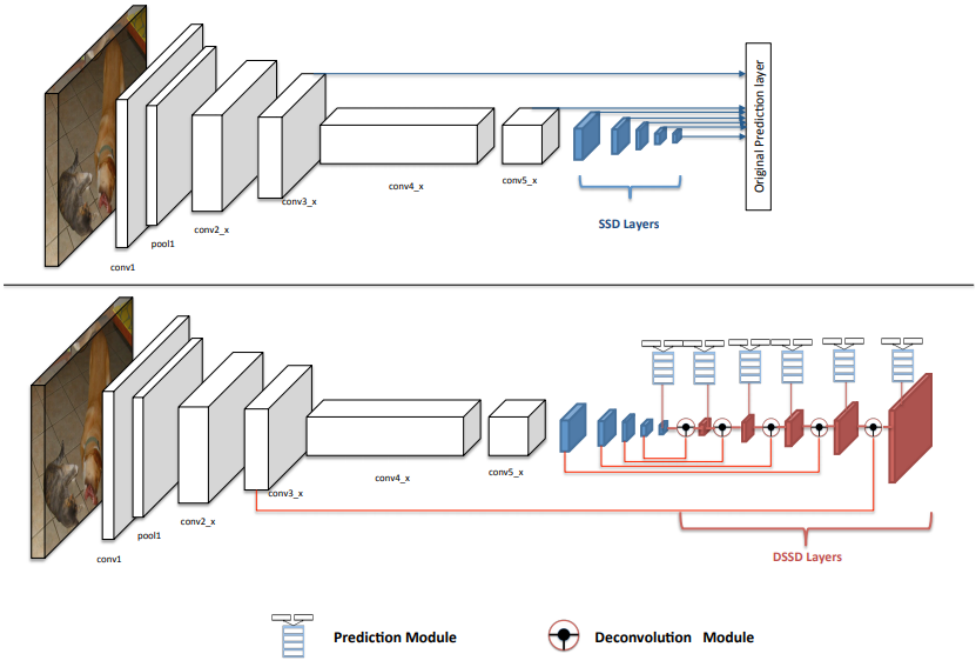

Figure 11: Networks of SSD and DSSD on ResNet

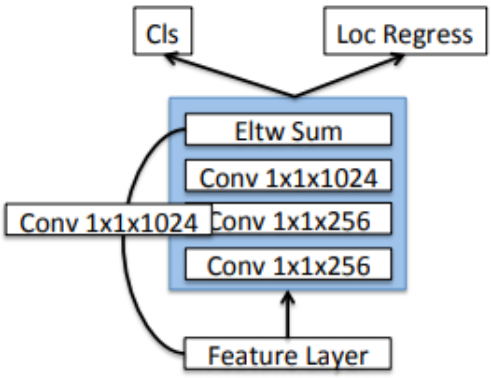

Figure 12: Prediction Module

- Due to the principle of improving sub-network can improve accuracy, the original SSD approach for parameter prediction is replaced with a prediction module that consists of a residual block. Thus, using ResNet101 with a prediction module performs better than VGG16. The implementation of the prediction module can be seen in figure 12.

3. Deconvolution Layers

- The aim of using deconvolutional layers is to include more high-level context in detection, so that, deconvolution module integrates information from both earlier feature maps and earlier deconvolution layers. Instead of using the deconvolution module, upsampling layers would have been used to increase the resolution of feature maps such as in hourglass networks. However, using the deconvolution module provides learnable parameters which perform better. You can see the implementation of the deconvolution module in figure 13.

3.2.5 CornerNet: Detecting Objects as Paired Keypoints

CornerNet [3] is a new approach to object detection. It detects objects using paired key points which means detecting bounding boxes using top-left and bottom-right corners of objects. In addition to this new approach, corner

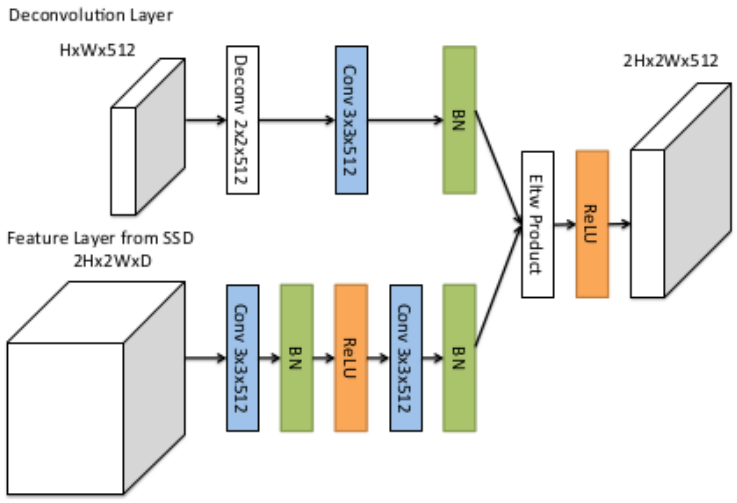

Figure 13: Deconvolution Module

pooling is introduced. Corners are better localized using corner pooling. Using paired keypoints eliminates the need for using anchor boxes.

There are two drawbacks of using anchor boxes:

1. Anchor boxes require a huge set. Since most of the anchor boxes do not overlap with ground-truth bounding boxes, a huge imbalance between positive and negative anchor boxes is created and handling this imbalance slows down training.

2. Anchor boxes introduce many hyperparameters and design choices to make, it may even introduce more if a single network makes separate predictions at multiple resolutions.

Corner pooling is applied by taking the maximum values in two directions (horizontal and vertical) for each channel and adding these two values together.

In CornerNet, a single neural network predicts a heatmap for the top-left corners of all instances of the same object category, a heatmap for all bottom-right corners, and an embedding vector for each detected corner [3].

3.2.6 FPN

Multi-scale handling is mentioned in section 2.3. For multi-scale input

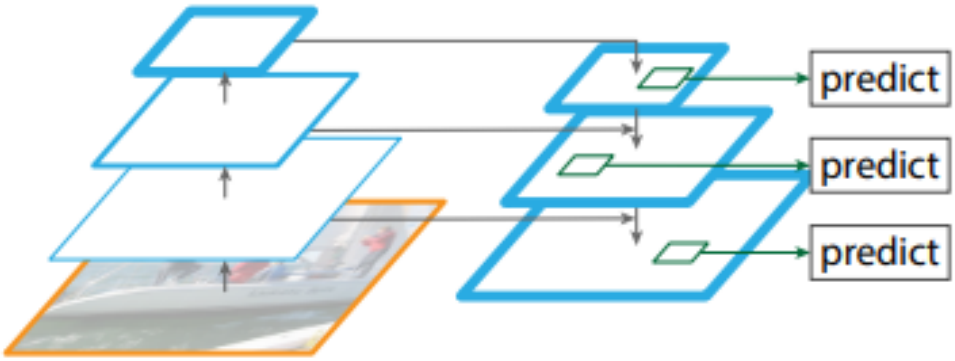

Figure 14: FPN Architecture

images, the required time and memory is too high to be trained end-to-end simultaneously. Also, the pyramidal feature hierarchy in figure 2 is not effective for accurate object detection, specifically in small objects due to the resolution of feature maps. Feature Pyramid Network (FPN) [4] is designed as a feature extractor keeping accuracy and speed in mind.

As you can see in figure 14, FPN consists of a bottom-up and top-down pathway. In the bottom-up pathway, as in pyramidal feature hierarchy (Fig. 2(c)), a convolutional network is used as a feature extractor. As layers go through, the resolution for feature maps are getting smaller, however, the detected features is more high level which leads to an increase of semantic value for each layer. In the top-down pathway, higher resolution feature maps are constructed using a semantic rich layer and corresponding layer from the bottom-up pathway. The reason for the lateral connection from the bottom-up pathway is to obtain location information of objects since after bottom-up and top-down pathways, the locations of objects not precise. Thus, connection to corresponding feature maps may help to increase the possibility of finding correct locations for objects.

3.2.7 Focal Loss and RetinaNet

In general, two-stage detectors have better accuracy than one-stage detectors. This article aims to find out why this is the case. Two-stage detectors are applied to a sparse set of candidate object locations. However, one-stage detectors are applied over a dense sampling of possible object locations. Then, the obtained result is the foreground-background class im-

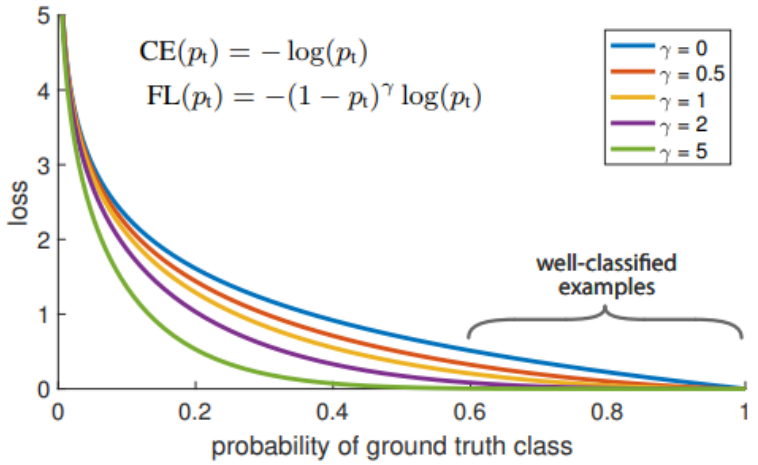

Figure 15: Focal Loss

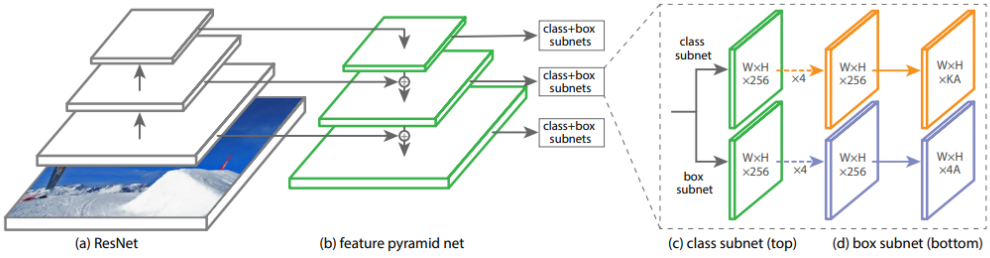

Figure 16: Network of RetinaNet

balance in the training of dense detectors. The improved solution to this problem is called focal loss (Fig. 15) which is introduced by this article.

Focal loss [5] is the reshaped version of cross-entropy loss. The aim of this change in cross-entropy is to down-weights the loss calculated for well-classified examples. Thus, the detector is trained on a sparse set of hard examples. Also, the class imbalance is handled by focal loss and sampling examples are not required since well-learned examples do not overwhelm loss during training. You can see the change in the loss by looking at figure 15. When γ equals to zero, the focal loss is equivalent to cross-entropy loss. For higher values of γ, the convergence of loss values changes as in the figure 15. The best-performing γ value equals to two according to the article.

RetinaNet [5] is designed to show the effectiveness of Focal Loss. As you can see in figure 16, RetinaNet uses ResNet as its backbone network and FPN to obtain a multi-scale convolutional feature pyramid. Then, two subnetworks are used to obtain detection results. One of them is used to classify anchor boxes to obtain the classification of objects in anchor boxes and the other one is used to regress bounding boxes from anchors in which the aim is to obtain ground-truth bounding boxes.

3.2.8 EfficientDet: Scalable and Efficient Object Detection

This article aims to study model efficiency and increase model efficiency. Therefore, building a scalable architecture with both higher accuracy and efficiency [12] is studied. As you can see in figure 17, variations of FPN are studied (only the chosen one is put) and BiFPN is chosen as the best-performing one. Also, as you can see in figure 18, EfficientDet is developed using EfficientNet as a backbone, BiFPN as a feature extractor, and two subnetworks one responsible for class prediction, the other one responsible for bounding box prediction.

There are two main ideas in the design of BiFPN: efficient bidirectional cross-scale connections and weighted feature fusion. Difference between FPN and BiFPN can be observed in figure 17. FPN has one-way data flow but bidirectional data flow is observed to be better. Also, the nodes that have only one input node are removed since there is no feature fusion and that node will have less contribution to feature network. In addition, an extra edge is added from input nodes to output nodes. The reason is to fuse more features. Finally, to enable more high-level feature, same layer is applied multiple times as you can see in 18. The importance of multi-scale feature fusion is mentioned before. The novel contribution here is using weights for feature fusion. Previous fusing operations on feature maps does not include weights until BiFPN, therefore, all the input features contribute equally. The problem here is that different input feature on different resolutions does not contribute output feature equally. Therefore, a weighted feature fusion is needed to obtain a better performance. Thus, learnable parameters are present to learn the importance of different features.

Bigger backbone networks or higher input resolutions can lead to an

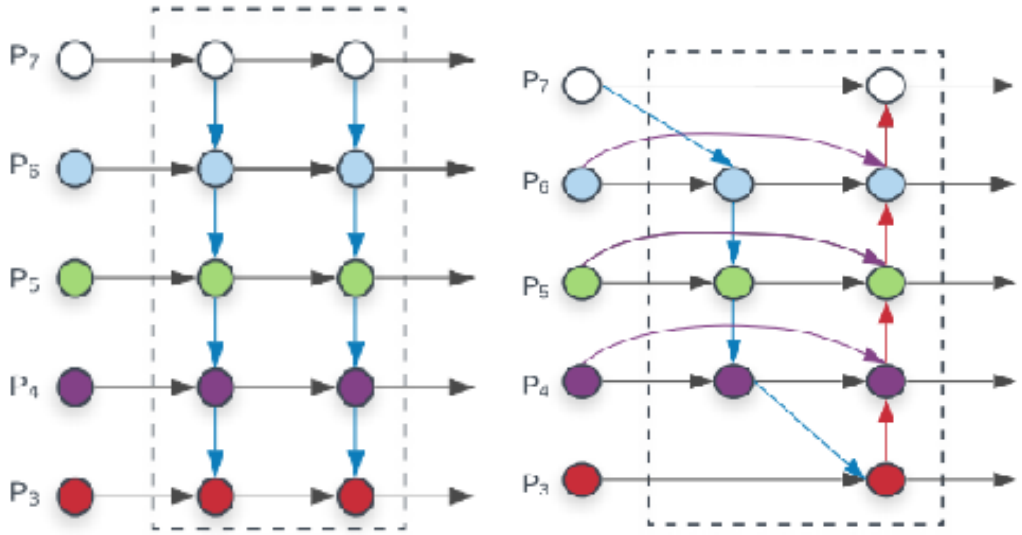

Figure 17: FPN (left) and BiFPN (right)

increase in accuracy but also, can lead to a decrease in efficiency due to the high number of operations required. Therefore, scaling up the network is crucial to obtain both accuracy and efficiency. The scaling component ϕ is used. You can see the configurations in figure 19. When detector configured from D0 to D7, mean average precision increases but also the inference time for the detector to run increases. The required model can be chosen based on the task performed.

4 Datasets and Metrics

In this section datasets and performance metrics will be explained.

Datasets are used in training and testing of detectors. Datasets consist of some specific type of objects, in object detection case they are images, and the corresponding ground-truth information about the categories in images that detector performs to detect. So, the mainly used datasets are Pascal Visual Object Classes (VOC) [1] and Microsoft COCO: Common Objects in Context [6].

Pascal VOC datasets are formed based on two challenges: classification and detection. From the beginning year of 2005 to 2012, Pascal VOC chal-

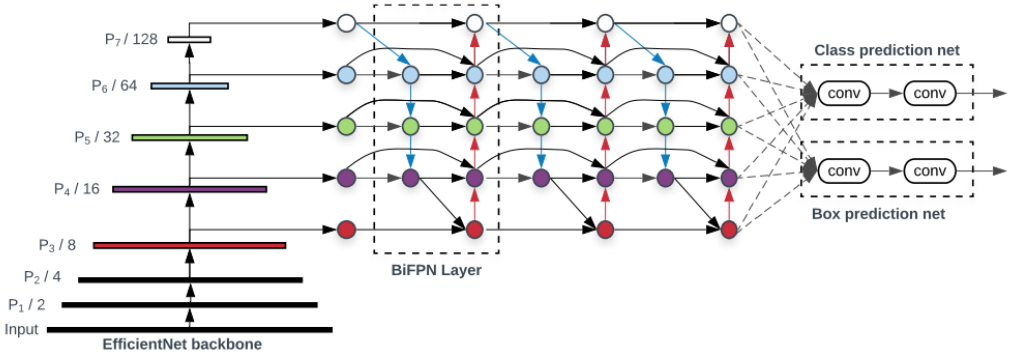

Figure 18: Network of EfficientDet

	Input size R_{input}	Backbone Network	BiFPN #channels W_{bifpn}	BiFPN #layers D_{bifpn}	Box/class #layers D_{class}
D0 ($\phi = 0$)	512	B0	64	2	3
D1 ($\phi = 1$)	640	B1	88	3	3
D2 ($\phi = 2$)	768	B2	112	4	3
D3 ($\phi = 3$)	896	B3	160	5	4
D4 ($\phi = 4$)	1024	B4	224	6	4
D5 ($\phi = 5$)	1280	B5	288	7	4
D6 ($\phi = 6$)	1408	B6	384	8	5
D7	1536	B6	384	8	5

Figure 19: Scaling Table of EfficientDet D0-D7

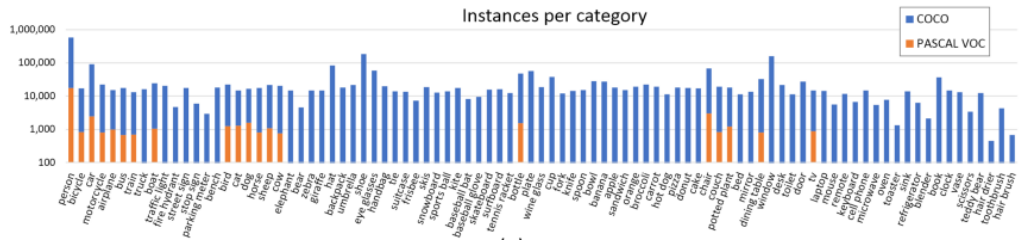

Figure 20: Instances per category

		Ground-truth Value	
		Positive	Negative
Predicted Value	Positive	True Positive	False Positive
	Negative	False Negative	True Negative

Table 1: Table to understand concepts of classification of a prediction

lenges are developed. Number of classes and images are increased through challenges. The last challenge of Pascal VOC is occured in 2012. Dataset of 2012 consists of 20 categories in 11.530 images.

MS COCO aims to present a dataset in which common objects are in their natural contexts. The data are taken from complex everyday scenes. COCO consists of 91 object categories in 328k images. A comparison between Pascal VOC and MS COCO of instances per category is shown in figure 20.

Another dataset we used to test analyzed detectors is Multiple Object Tracking Benchmark (MOT) [8]. Although MOT is a dataset for tracking, in challenge MOT17Det, they provide a challenge for Pedestrian Detection. There are seven different video data provided for training set with their ground-truth labels. We used this data as test set. There are 5316 frame to process with total of 112297 pedestrians annotated.

Performance metrics are the way to measure the accuracy of detectors. In object detection, improving precision and recall yields to better test performance. The difference lies under the calculation of these metrics for object detection since correctly classifying the object in the image is not enough, detector also correctly localize the object in the image. So, to measure the correctness of each detection, a metric called Intersection over Union is used:

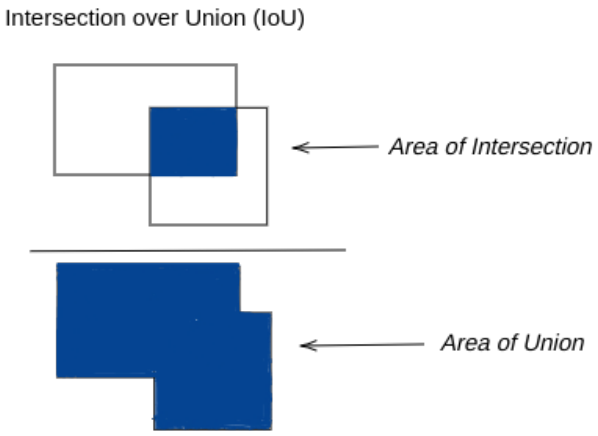

Figure 21: A visualization of IoU

- Intersection over Union (IoU) IoU is the ratio between the intersection of predicted and ground-truth bounding box divided by union of predicted and ground-truth bounding box. The figure 21 is a visualization of IoU to help to have a better intuition.
- Correctness of Detection IoU is used as a threshold to identify a detection as positive or negative. The most commonly used IoU threshold is 0.5 which means if IoU value of a detection is bigger than 0.5, that detection is counted as True Positive; otherwise, False Positive.
- The following concepts are used by the metrics:
  - True Positive (TP): A correct detection, IoU $\geq$ threshold.
  - False Positive (FP): A wrong detection, IoU $\leq$ threshold.
  - False Negative (FN): A ground-truth not detected.
  - True Negative (TN): Since there are many possible bounding boxes that do not contain any object, counting corrected misdetection is not necessarily needed.
  - A better intuition can be obtained by looking into table 1.
- These calculated concepts above are used to measure two main metrics: precision and recall. Precision is the ratio of true positives to the sum

Stage	Model-Input Resolution	Backbone	mAP	FPS
Two Stage	Faster R-CNN	VGG16	39.3	5
Two Stage	Faster R-CNN + FPN	ResNet101	58.5	6.75
One Stage	YOLOv2	DarkNet19	21.6	40
One Stage	YOLOv3-320/416/608	DarkNet53	28.2/31.0/33.0	45.45/34.48/19.60
One Stage	SSD-300/512	VGG16	25.1/28.8	59.0/22.0
One Stage	SSD-513	ResNet101	31.2	8
One Stage	DSSD-513	ResNet101	33.2	6.41
One Stage	RetinaNet500/800	ResNet50/101	32.5/37.8	13.88/5.05
One Stage	RetinaNet	ResNet101/X	39.1/40.8	8.19
One Stage	EfficientDet-D3	EfficientNet	44.3	23.81
One Stage	CornerNet(s/m)	Hourglass	40.5/42.1	4.09

Table 2: Performance table on COCO dataset

of true positives and false positives which provides the ratio of how accurate is your predictions. Recall is the ratio of true positives to the sum of true positives and false negatives which indicates the ratio of finding positives in all positives.

- The performance indicator for detectors is mean average precision. Average precision of a category can be calculated by calculating the area under precision-recall curve. Then, the mean average precision is calculated by calculating average precision for all categories and divide this total AP by the number of categories. In Pascal VOC, AP is calculated by using interpolation. 11-point interpolated precision-recall curve is used. Corresponding precision values of 11 recall points are summed up and this value is divided by 11. Another way is to calculate all area under the precision-recall curve which is used in later year competetions of Pascal VOC. In MS COCO, 101-point interpolated AP calculation is used. In MOT17Det, 11-point interpolated AP is used.

5 Performance Comparison

Understanding the performance of detectors based on different cases and different features is important knowledge to decide on the usage of detectors in real-life problems. Therefore, a table to show the performance of

detectors is prepared which can be seen in table 2.

5.1 Accuracy and Real-Time Applicable

Obtained accuracy and frame rate per second performances of detectors are visible in table 2. These results are valid on COCO dataset. Since most of the detectors perform well on Pascal VOC dataset and COCO is the more hard dataset to obtain better accuracy, only the results on COCO dataset are shown. Also, the aim of this study is to find out which detectors can be applied in real-time. Therefore, the detectors that are shown in table 2 are chosen in a specific form in which they are most close to their real-time applicable specifications.

Based on the results in table 2, YOLOv2, YOLOv3 for 320x 320 image resolution, YOLOv3 for 416x416 image resolution, SSD for both 300x300 and 512x512 image resolution, EfficientDet-D3 can be used in real-time applications since videos require 24 FPS to play without any distortion in human perception. Although the rest of the detectors cannot be applicable in real-time, their accuracy is better than these real-time applicable detectors. Thus, these detectors can be used in different cases than real-time applications, so that, better accuracy in objective can be obtained.

6 Conclusion

In this article, a detailed study of object detectors is introduced. Object detectors are used to detect object instances in given images. Detector features are important when a detector is designed since these features are the main factors of detectors' performance. State-of-the-art detectors through the years 2015 to 2019 are introduced in this article. These detectors can be used according to the application needed since they all have different advantages and disadvantages. Most commonly used datasets and performance metrics are introduced and the introduced detectors' performances are compared. Based on this comparison, one can choose a detector that is most applicable to the needed application.

References

[1] M. Everingham et al. "The Pascal Visual Object Classes (VOC) Challenge". In: *International Journal of Computer Vision* 88.2 (June 2010), pp. 303–338.

[2] Cheng-Yang Fu et al. "DSSD : Deconvolutional Single Shot Detector". In: *CoRR* abs/1701.06659 (2017). arXiv: 1701.06659. URL: http://arxiv.org/abs/1701.06659.

[3] Hei Law and Jia Deng. "CornerNet: Detecting Objects as Paired Keypoints". In: *CoRR* abs/1808.01244 (2018). arXiv: 1808.01244. URL: http://arxiv.org/abs/1808.01244.

[4] Tsung-Yi Lin et al. "Feature Pyramid Networks for Object Detection". In: *CoRR* abs/1612.03144 (2016). arXiv: 1612.03144. URL: http://arxiv.org/abs/1612.03144.

[5] Tsung-Yi Lin et al. "Focal Loss for Dense Object Detection". In: *CoRR* abs/1708.02002 (2017). arXiv: 1708.02002. URL: http://arxiv.org/abs/1708.02002.

[6] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *CoRR* abs/1405.0312 (2014). arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[7] Wei Liu et al. "SSD: Single Shot MultiBox Detector". In: *Lecture Notes in Computer Science* (2016), pp. 21–37. ISSN: 1611-3349. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[8] A. Milan et al. "MOT16: A Benchmark for Multi-Object Tracking". In: *arXiv:1603.00831 [cs]* (Mar. 2016). arXiv: 1603.00831. URL: http://arxiv.org/abs/1603.00831.

[9] Joseph Redmon and Ali Farhadi. "YOLO9000: Better, Faster, Stronger". In: *CoRR* abs/1612.08242 (2016). arXiv: 1612.08242. URL: http://arxiv.org/abs/1612.08242.

[10] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *CoRR* abs/1506.02640 (2015). arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[11] Shaoqing Ren et al. “Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks”. In: *CoRR* abs/1506.01497 (2015). arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[12] Mingxing Tan, Ruoming Pang, and Quoc V. Le. *EfficientDet: Scalable and Efficient Object Detection*. 2019. arXiv: 1911.09070 [cs.CV].