Object Detection

Halil Eralp Koçaş

February 12, 2020

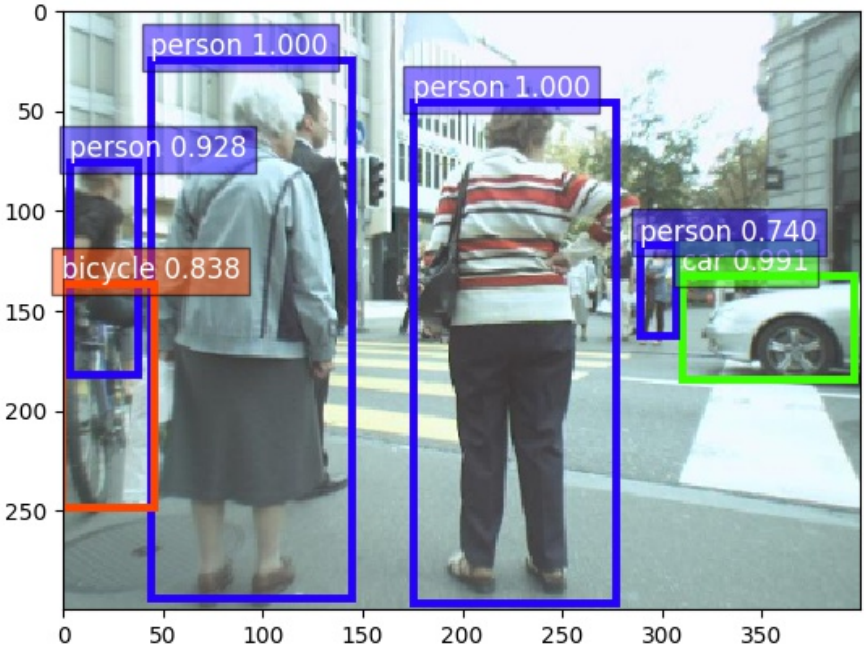


Figure 1: Example of Object Detection. Bounding boxes are drawn around objects. Also, the predicted class and the prediction confidence scores are indicated in the upper parts of bounding boxes.

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. Also, the parts of the given image that contain no object is called background.

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 image is processed for the first time in backbone networks. These backbone networks are implemented as Fully Convolutional Neural Networks. Since these backbone networks are full convolutional neural networks, one of the main benefits of using these is to train our detectors end-to-end. The used networks are the ones that are already proven to perform classification well in large image datasets. The most used backbone networks are VGG [17], ResNet [3], DarkNet [13], and Hourglass [12] networks.

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

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

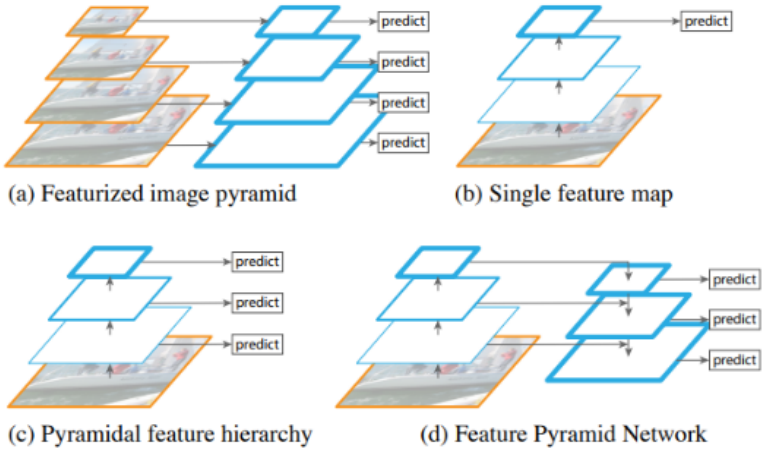


Figure 2: Different methods to use feature maps [7]. (a) Image pyramid is used to build a feature pyramid. Input image is scaled down and feature maps are computed independently which is slow. (b) Initially calculated feature map from input image is downscaled using convolutional layers and single scale feature map is used. This method is used for faster convergence. (c) The idea is to use feature pyramid instead of image pyramid as (a). This is slower than (a) but more accurate. (d) will be explained in section 3.2.6.

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.

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

- L2 Loss Function

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

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

- Smooth L1 Loss Function

  Smooth L1 loss function minimizes whether half of the square of the predicted value or difference of absolute value of predicted value and 0.5.

$$\sum_{i=1}^{n} \begin{cases} 0.5y_i^2 & \text{if } |y_i| < 1 \\ |y_i| - 0.5 & \text{otherwise} \end{cases} \tag{3}$$

- Cross-Entropy Loss Function

  Cross-Entropy loss function minimizes the product of ground-truth value and logarithm of predicted value.

$$\sum_{i=1}^{n} (y_t * \log(y_i)) \tag{4}$$

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

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 [16] 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 3, 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 4. 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

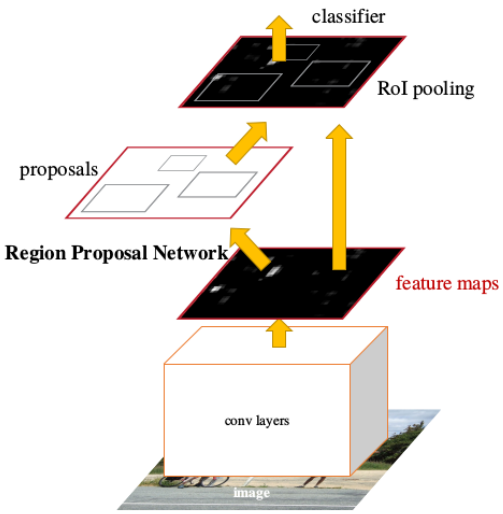


Figure 3: Network of Faster R-CNN [16]. Faster R-CNN is single and a unified network. The RPN extracts region of interests (possible locations of objects in the given image) from the extracted feature map from the given image. Then, the classifier network operates on the final feature map to classify objects.

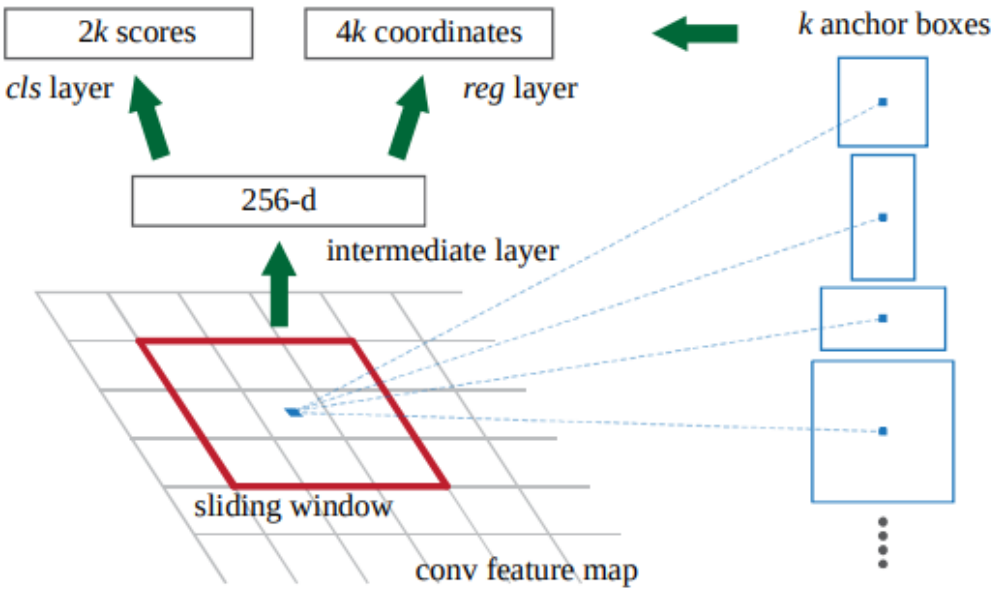


Figure 4: Region Proposal Network (RPN) [16]. RPN takes a convolutional feature map as input. Then, it outputs a set of rectangular region proposals with objectness score.

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) [15] 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.
    - 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 [14]. 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 [14].
- 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**

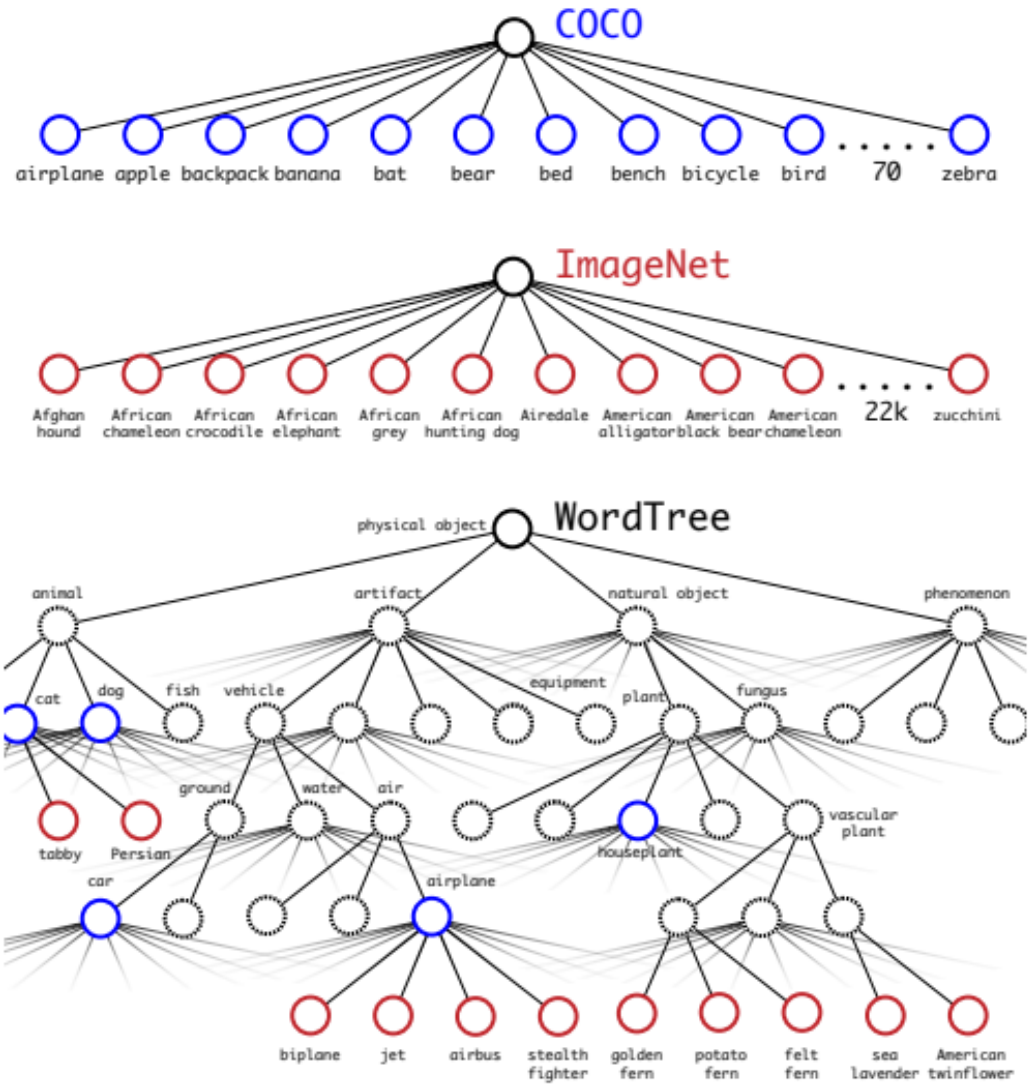


Figure 5: Combining ImageNet and COCO using WordTree hierarchy [14]. This is a hierarchical tree of visual concepts which is created using WordNet concept. The classes are mapped to their synsets, so that, datasets are merged.

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

Figure 6: Classification calculation example for WordTree [14]

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. Detection and classification datasets are used together during training. 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.

This approach requires a modification in prediction of detections. 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 5, 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 6.

3.2.3 SSD: Single Shot MultiBox Detector

Single Shot MultiBox Detector [10] 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:

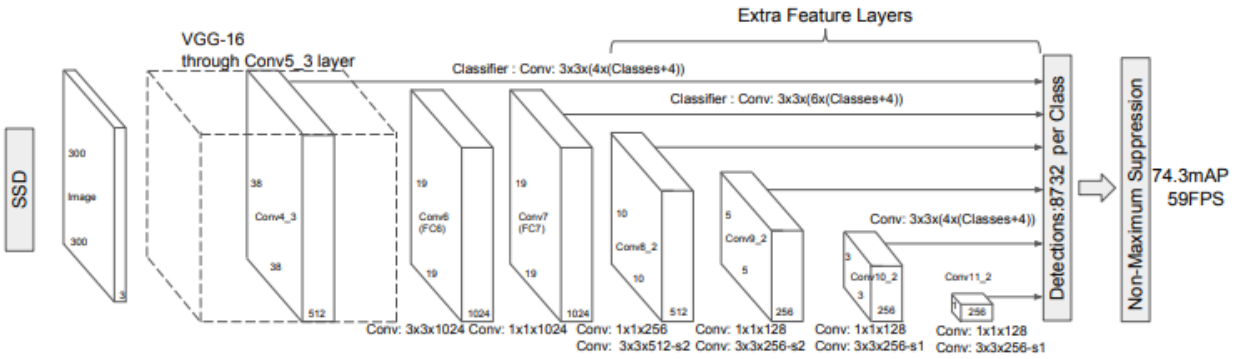


Figure 7: Network of SSD [10]. The pyramidal feature hierarchy can be seen on the figure. After the base model, VGG-16, extra feature layers and VGG-16 itself are connected to the detection layer by using 3x3 convolutional layers with different numbers of filters. For instance, there are 4 filters connected to detection layer from VGG-16. By using these multiple filters from the multiple convolutional feature layers provide more accurate object detection.

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 7 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 8, 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

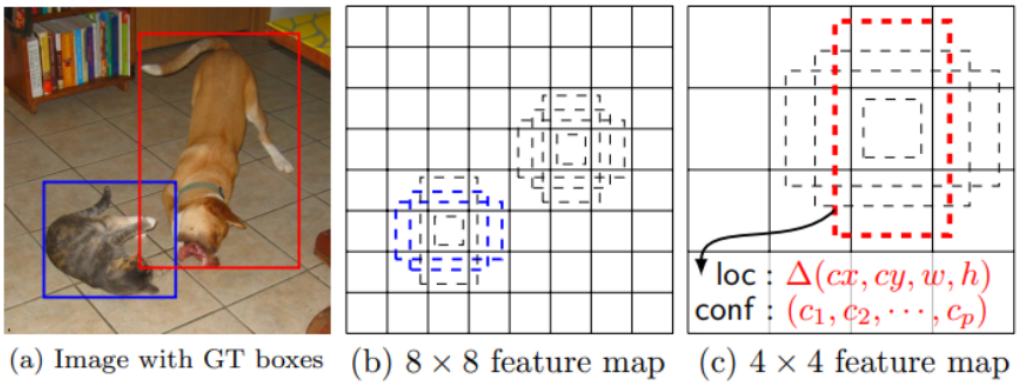


Figure 8: Multi-scale feature maps [10]. Different sizes of feature maps are used to detect different sizes of objects in the given images. For instance, while 8x8 feature map can be able to detect the cat in the image, 4x4 feature map can be able to detect the dog in the image.

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 9, 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 9 can be used to detect cars but bounding box 3 is a better option for detecting people.
   - 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.

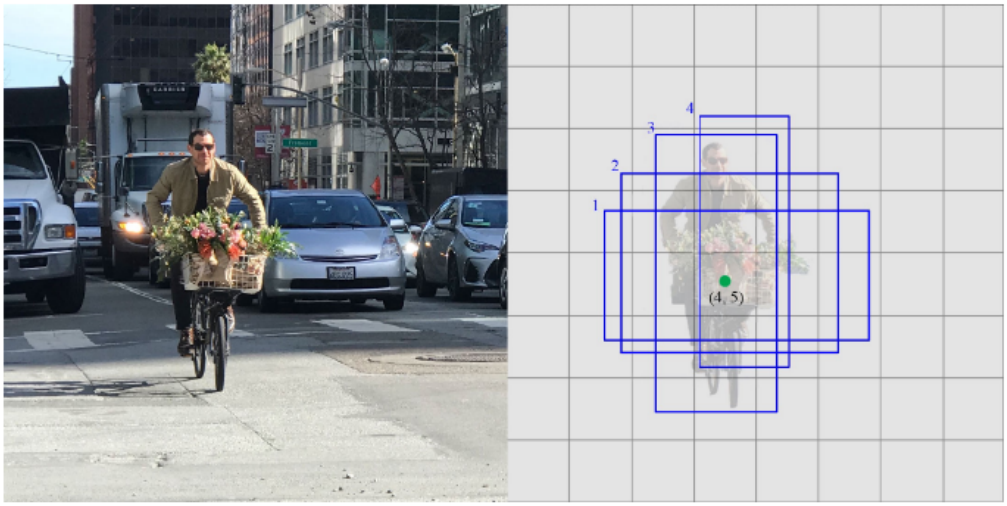


Figure 9: Default Bounding Boxes [4]. This figure is the visualization of Conv4_3 and it is 8x8 spatially (it should be 38x38). As it can be seen in the figure, there are four differently-shaped default bounding boxes to make object prediction. The reason for the different shapes of the default bounding boxes is to increase the possibility of detecting an object. For instance, the forth box in the figure can be used to detect people, yet, the first box can be used to detect cars in the given images.

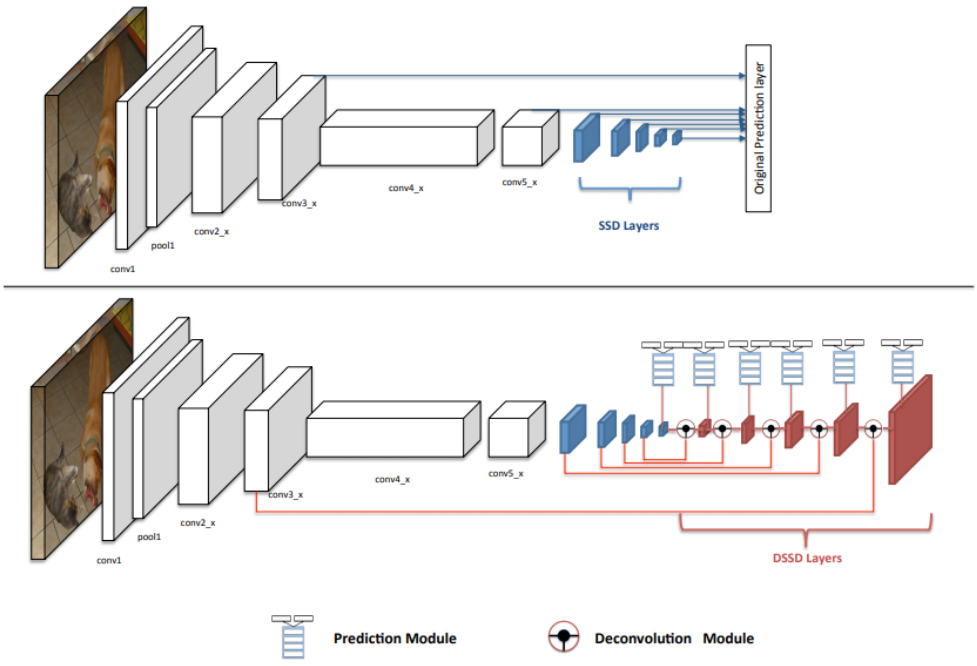


Figure 10: Networks of SSD and DSSD on ResNet [2]. The differences between SSD and DSSD networks are visible in the figure. Deconvolution modules are added after SSD layers. Also, the prediction layer in SSD is replaced with multiple prediction layers, so that, predictions are made after each deconvolution layers.

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 10, 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

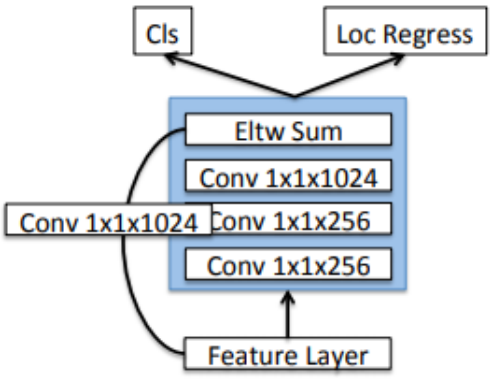


Figure 11: Prediction Module [2].

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

   - 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 11.

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 12.

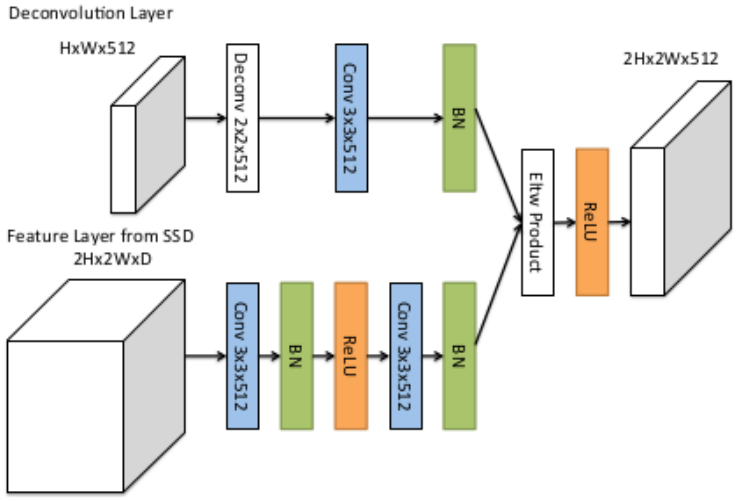


Figure 12: Deconvolution Module [2]. Feature layer is taken from the corresponding SSD layer which can be seen in figure 10. Also, the input deconvolution layer's dimensions are doubled with a deconvolution layer. Then, the layers in the deconvolution module are applied over the input layers.

3.2.5 CornerNet: Detecting Objects as Paired Keypoints

CornerNet [6] 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 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

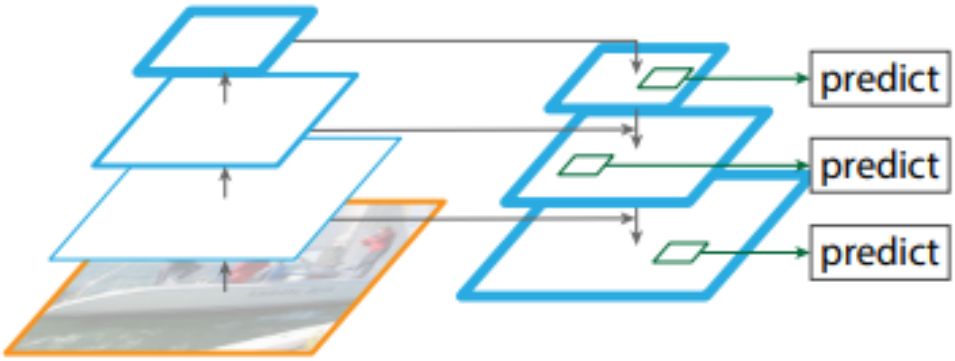


Figure 13: FPN Architecture [7]. It is a top-down architecture with skip connections. Feature pyramid is built and predictions are made independently at all levels.

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 [6].

3.2.6 Feature Pyramid Network (FPN)

Multi-scale handling is mentioned in section 2.3. For multi-scale input 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) [7] is designed as a feature extractor keeping accuracy and speed in mind.

As you can see in figure 13, 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.

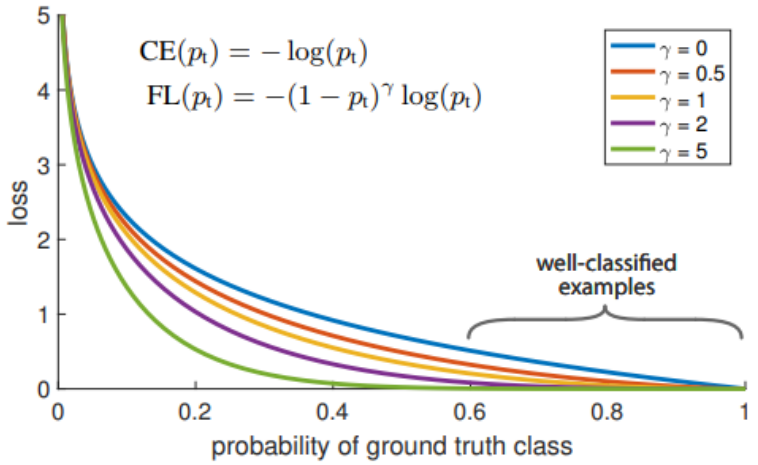


Figure 14: Focal Loss [8]. The effect of gamma can be observed. The contribution of well-classified examples to loss function reduces when gamma increases. Detectors are more focused on hard, misclassified examples. Thus, the accuracy of detectors are increased since they learn less from the well-classified examples, yet, more from the hard, misclassified examples.

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 imbalance in the training of dense detectors. The improved solution to this problem is called focal loss (Fig. 14) which is introduced by this article.

Focal loss [8] 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 14.

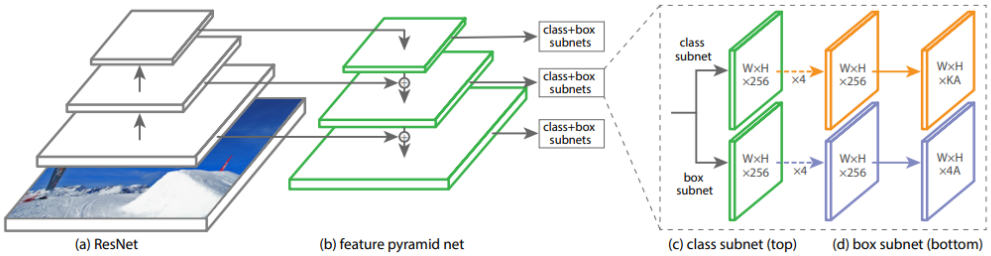


Figure 15: Network of RetinaNet [8]. Feature pyramid network is used after ResNet to generate a fine-grained, multi-scale convolutional feature pyramid. Then, two subnetworks are attached after RetinaNet in which one for classifying anchor boxes and the other one for regressing final bounding boxes from anchor boxes.

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 14. The best-performing γ value equals to two according to the article.

RetinaNet [8] is designed to show the effectiveness of Focal Loss. As you can see in figure 15, 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 [18] is studied. As you can see in figure 16, variations of FPN are studied and BiFPN is chosen as the best-performing one. Also, as you can see in figure 17, 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 16. FPN has one-way data flow but

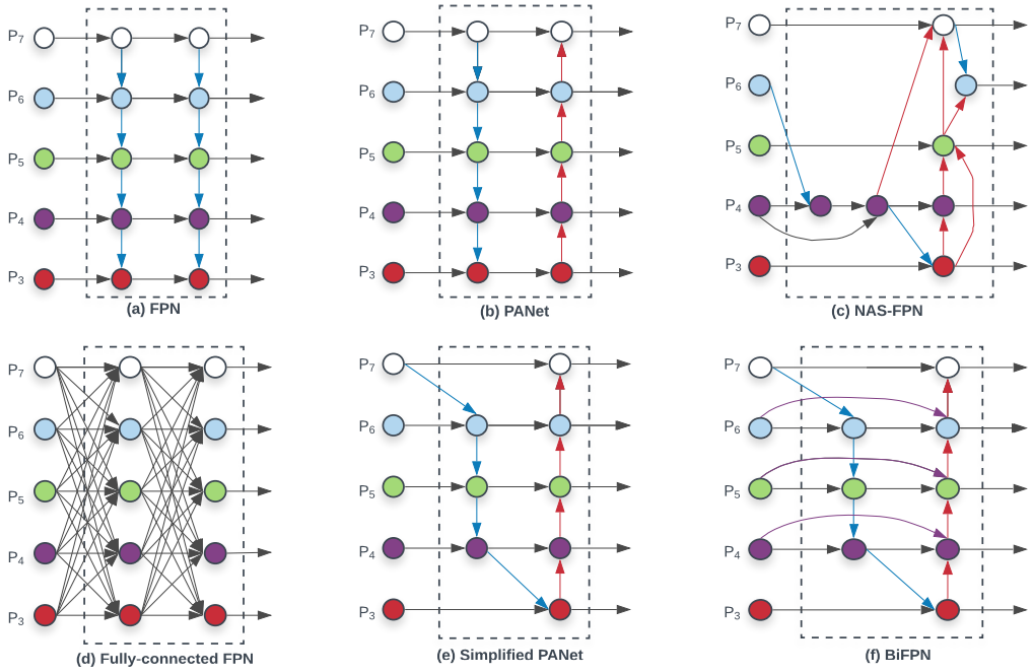


Figure 16: Feature Network Design [18]. As it can be seen in this figure, FPN introduces a top-down pathway to fuse multi-scale features from level 3 to level 7 (P3-P7). Then, PANet introduces an additional bottom-up pathway. NAS-FPN introduces a method in which a neural architecture is used to find an irregular network topology. Fully-connected FPN adds expensive connections from all input features to output features. Simplified PANet removes the nodes if they only have one input edge. BiFPN is discussed in this paper and it is introduced with better accuracy and efficieny trade-offs.

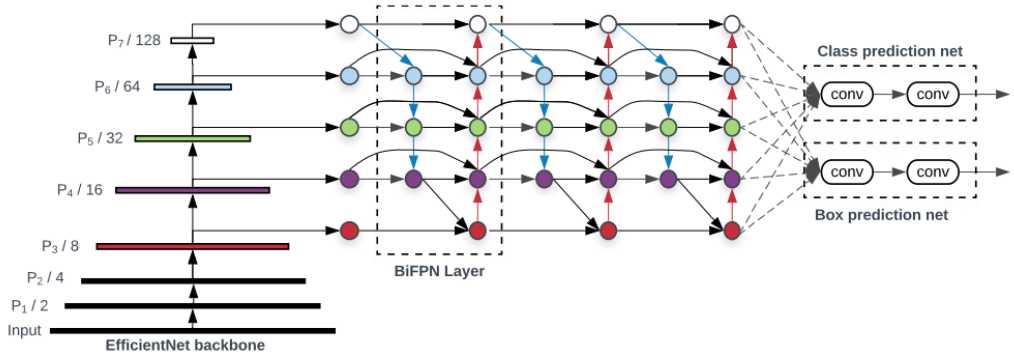


Figure 17: Network of EfficientDet [18]. Its backbone network is EfficientNet and BiFPN is used as feature network. After the backbone and feature networks, shared class and box prediction networks are used to predict. Both BiFPN and class/box prediction layers are used multiple times based on the constraints as shown in figure 18.

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 17. 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 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 18. 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.

	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 18: Scaling Table of EfficientDet D0-D7 [18]. ϕ is used as the compound coefficient that controls all other scaling dimensions.

4 Datasets and Metrics

In this section datasets and performance metrics will be explained.

Datasets are used in the training and testing of detectors. Datasets consist of some specific types 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 [9].

Pascal VOC datasets are formed based on two challenges: classification and detection. From the beginning year of 2005 to 2012, Pascal VOC challenges are developed. The number of classes and images is increased through challenges. The last challenge of Pascal VOC occurred 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 19.

Another dataset we used to test analyzed detectors is the Multiple Object Tracking Benchmark (MOT) [11]. Although MOT is a dataset for

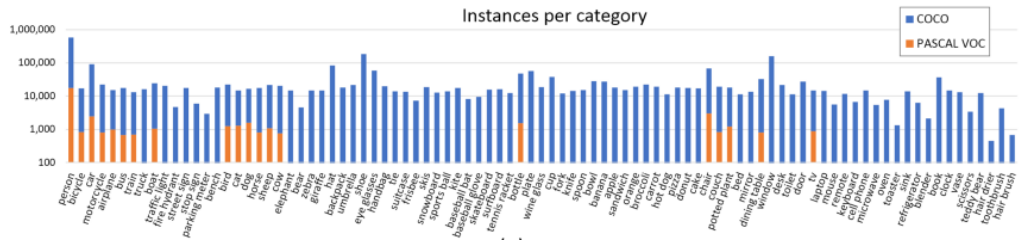


Figure 19: Instances per category [9]. As it can be seen in the figure, MS COCO dataset dominates both the number of categories and the number of instances per category over Pascal VOC dataset.

		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

tracking, in challenge MOT17Det, they provide a challenge for Pedestrian Detection. There are seven different video data provided for the training set with their ground-truth labels. We used this data as a test set. There are 5316 frames to process with a 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, the detector also correctly localizes the object in the image. So, to measure the correctness of each detection, a metric called Intersection over Union is used.

- Intersection over Union (IoU) IoU is the ratio between the intersection of predicted and ground-truth bounding box divided by the union of predicted and ground-truth bounding box. The figure 20 is a visualization of IoU to help to have a better intuition.

- Correctness of Detection IoU is used as a threshold to identify a detec-

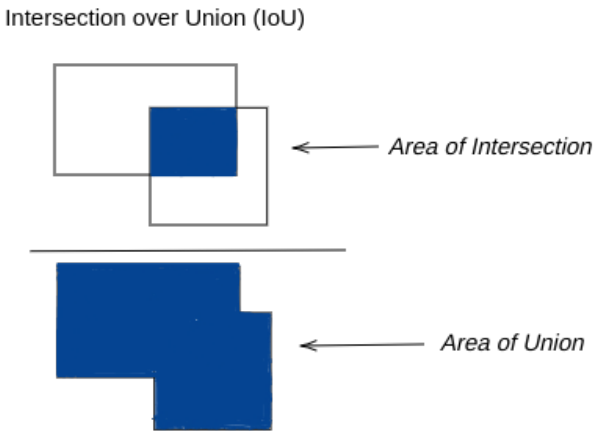


Figure 20: A visualization of IoU. The intersection area of two boxes is divided by the union area of these two boxes.

tion 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:
  - Ground-truth (GT): The true annotations on a given frame.
  - 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 many possible bounding boxes 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 of true positives and false positives which provides the ratio of how accurate are your predictions. The 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.

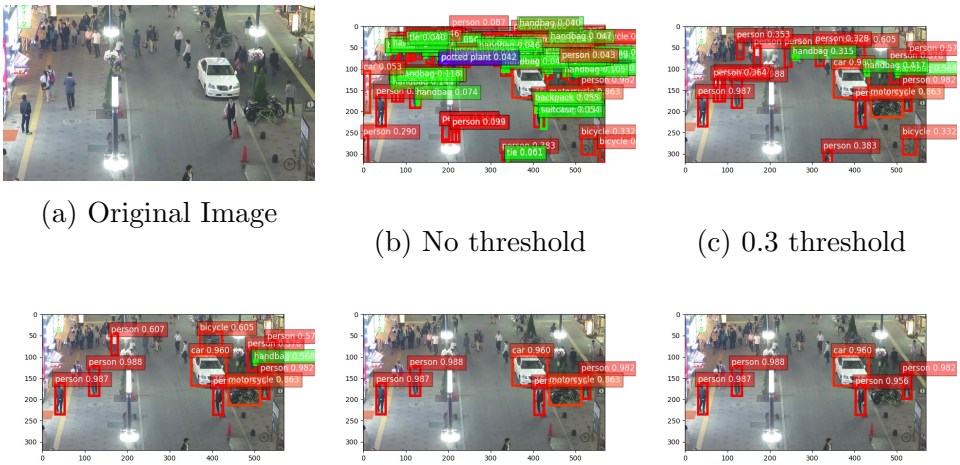
- The performance indicator for detectors is mean average precision. The average precision of a category can be calculated by calculating the area under the precision-recall curve. Then, the mean average precision is calculated by calculating average precision for all categories and divide this total AP by the number of categories. In Pascal VOC, AP is calculated by using interpolation. The 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 competitions of Pascal VOC. In MS COCO, 101-point interpolated AP calculation is used. In MOT17Det, 11-point interpolated AP is used.

- False Alarm Rate (FAR) is calculated per frame. A false alarm is equivalent to a false positive. Detector assumes an object is present in some location, although there is no object in that same position. So, False Alarm Rate per frame is calculated by dividing the number of false positives to the number of frames in the dataset. The relation of FAR and FP can be observed in table 3.

- Multiple Object Detection Accuracy (MODA) [5] is calculated by using weighted-sum of FN and FP per frame and this sum is divided by the number of ground-truth objects in that frame. So, the higher the MODA value corresponds to a better performance in a detector. In the equation 5, m_t refers to the number of false negatives and f_{p_t} refers to the number of false positives for each frame t. Also, c_m and c_f are the cost functions for the false negatives and false positives and $N_G^{(t)}$ refers to the number of ground-truth objects in the tth frame.

$$MODA(t) = 1 - \frac{c_m\left(m_t\right) + c_f\left(fp_t\right)}{N_G^{(t)}} \tag{5}$$

- Multiple Object Detection Precision (MODP) [5] is calculated by using the overlap between ground-truth and predicted object. In other terms, for a given frame, all objects' IoU values are calculated with corresponding to their ground-truth boxes and these values are summed up. Then, this sum is divided by the number of total ground-truth



(a) Original Image (b) No threshold (c) 0.3 threshold

(d) 0.5 threshold (e) 0.7 threshold (f) 0.9 threshold

Figure 21: Examples of detected boxes for different thresholds in YOLOv3 for input's short dimension is 320. As it can be seen, the number of detected objects reduces from 0 threshold to 0.9 threshold.

boxes in that frame. Thus, the higher MOTP value corresponds to a better performance in a detector.

$$\text{Mapped Overlap Ratio} = \sum_{i=1}^{N^{(t)}-1} \frac{\left|G_i^{(t)} \cap D_i^{(t)}\right|}{\left|G_i^{(t)} \cup D_i^{(t)}\right|} \tag{6}$$

$$\text{MODP}(t) = \frac{(\text{Mapped Overlap Ratio})}{N_{\text{mapped}}^{(t)}} \tag{7}$$

In the equation 6, $G_i^{(t)}$ refers to the ith ground-truth object in the tth frame, D_i^t refers to the detected object for G_i^t, and N_{mapped}^t is the number of mapped object pairs in the frame t.

5 Performance Comparison

Understanding the performance of detectors based on different cases

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

and different features is important knowledge to decide on the usage of detectors in real-life problems. Therefore, tables to show the performance of detectors are prepared which can be seen in table 2, 3, 4, and 5.

Model	Average Precision	Recall	Precision	FAR	GT	TP	FP	FN	MODA	MODP
Faster R-CNN + FPN	0.5893	67.5	77.5	2.44	66393	44792	12988	21601	47.9	79.4
SSD512	0.4546	56.5	75.2	2.33	66393	37521	12367	28872	37.9	80.0
YOLOv3_320	0.4642	51.7	68.5	2.97	66393	34311	15782	32082	27.9	80.2

Table 3: Performance table of different detectors on MOT dataset by holding detection threshold 0.5

Model	Average Precision	Recall	Precision	FAR	GT	TP	FP	FN	MODA	MODP
YOLOv3_320	0.4642	51.7	68.5	2.97	66393	34311	15782	32082	27.9	80.2
YOLOv3_416	0.5236	60.2	68.4	3.48	66393	39938	18481	26455	32.3	80.8

Table 4: Performance table to show the effect of input image size on MOT dataset by holding detection threshold 0.5

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 the COCO dataset. Since most of the detectors perform well on the Pascal VOC dataset and COCO is the more hard dataset to obtain better accuracy, only the results on the COCO dataset are shown. Also, the aim of this study is to find out which

Model	Avg. Precision	Recall	Precision	FAR	GT	TP	FP	FN	MODA	MODP	Threshold
YOLOv3	0.5532	80.0	12.4	70.32	66393	53116	373818	13277	-483.0	77.4	0.0
YOLOv3	0.5172	61.0	56.9	5.78	66393	40509	30712	25884	14.8	79.3	0.3
YOLOv3	0.4642	51.7	68.5	2.97	66393	34311	15782	32082	27.9	80.2	0.5

Table 5: Performance table to show the effect of detection threshold on MOT dataset by using 320 as the short dimension of the input image

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 320 as short dimension of input image resolution, YOLOv3 for 416 as short dimension of input image resolution, SSD for both 300 and 512 as short dimension of input 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.

5.2 Accuracy and Class Score Threshold

The Faster R-CNN with FPN, SSD for 512 input's short dimension resolution, and YOLOv3 for 320 and 416 input's short dimension resolution are analyzed with the MOT dataset and the results can be observed in table 3. The train set of MOT dataset is used as a test set on these models since their ground-truth values are provided. So, these models are not trained on the MOT dataset. They only trained on the COCO dataset. The class score threshold is a threshold for an object is accepted as its founded class. For instance, if SSD detects an object as a pedestrian with a class score of 0.85 and if the threshold is equal to 0.5, then, this object will be accepted as pedestrian and it will be included in the detected objects' list. However, if the prediction score is 0.3, then, the prediction will be rejected. The threshold for the COCO dataset is 0.5. So, to understand the effect of threshold on detection, YOLOv3 is analyzed on the MOT dataset for different threshold values: 0.0, 0.3, 0.5 as can be observed in table 3. Also, a selected image is visualized with detected bounding boxes on figure 21 for different threshold values: 0.0, 0.3, 0.5, 0.7, 0.9.

The results in table 3 show us that the performances of the analyzed detectors are compatible with their performances in table 2. In addition, in table 3, you can see the effect of input resolution and threshold.

Reducing the resolution of the input image may lead to a loss of information. Therefore, using a bigger input image resolution may increase the average precision of detectors. As you can see in table 3, the average precision is increased from 0.4642 to 0.5236 when you use YOLOv3 with 416 input image resolution instead of 320 input image resolution. This change increases the number of true positives and false positives, however, the number of false negatives is reduced. Therefore, recall is increased significant amount which means a performance increase in detectors.

Reducing the threshold for class prediction may lead to an increase in the number of true positives. Yet, you can notice that it also leads to a huge increase in false positives. The reason for this increase can be observed in figure 21. You can see the difference in the number of detected objects in given thresholds for the given image. For instance, having a 0.0 threshold leads our detector to accept a detection as a true detection even the score of classification is around 0.02. Therefore, detectors predict so many detections. The numbers of true and false positives and false negatives can be observed in table 3. You can notice that the number of true positives reduces from 0.0 to 0.9 threshold but the number of false positives also reduces drastically. In addition, the number of false negatives increases in the corresponding order. The reason for this flow is that having a lower threshold leads to detect every part of the given image even though there is no object present. So, in the end, the detector covers every object of the image even with the parts that have no object. Therefore, all the objects are detected and TP is close to GT. Thus, FN is less. Also, since the detector detects many backgrounds as positive, the FP is also high. As a result, keeping the threshold low leads to a better average precision, yet, the precision reduces significant amount and the recall increases significant amount. This means that our detector is good at finding positives but our detector is not accurate since it detects almost anything as positive.

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. These analyzed 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. The tradeoff between precision and recall is a design choice and it can be chosen based on the task required. For instance, you can lower the threshold if the task is dangerous and even though it is a false alarm, you want to get alarmed in any small change. On the other hand, if the cost of a false alarm is high and the task allows you to miss some events, then, using a high threshold to lower cost and miss some events is suitable. Based on these design choices, 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] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *CoRR* abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/1512.03385.

[4] Jonathan Hui. *SSD object detection: Single Shot Multibox Detector for real-time processing*. 2018. URL: https://medium.com/@jonathan_hui/ssd-object-detection-single-shot-multibox-detector-for-real-time-processing-9bd8deac0e06.

[5] Rangachar Kasturi et al. "Framework for Performance Evaluation of Face, Text, and Vehicle Detection and Tracking in Video: Data, Metrics, and Protocol". In: *IEEE transactions on pattern analysis and machine intelligence* 31 (Mar. 2009), pp. 319–36. DOI: 10.1109/TPAMI.2008.57.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

[11] 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.

[12] Alejandro Newell, Kaiyu Yang, and Jia Deng. "Stacked Hourglass Networks for Human Pose Estimation". In: *CoRR* abs/1603.06937 (2016). arXiv: 1603.06937. URL: http://arxiv.org/abs/1603.06937.

[13] Joseph Redmon. *Darknet: Open Source Neural Networks in C*. http://pjreddie.com/darknet/. 2013–2016.

[14] 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.

[15] 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.

[16] 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.

[17] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2014. arXiv: 1409.1556 [cs.CV].

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