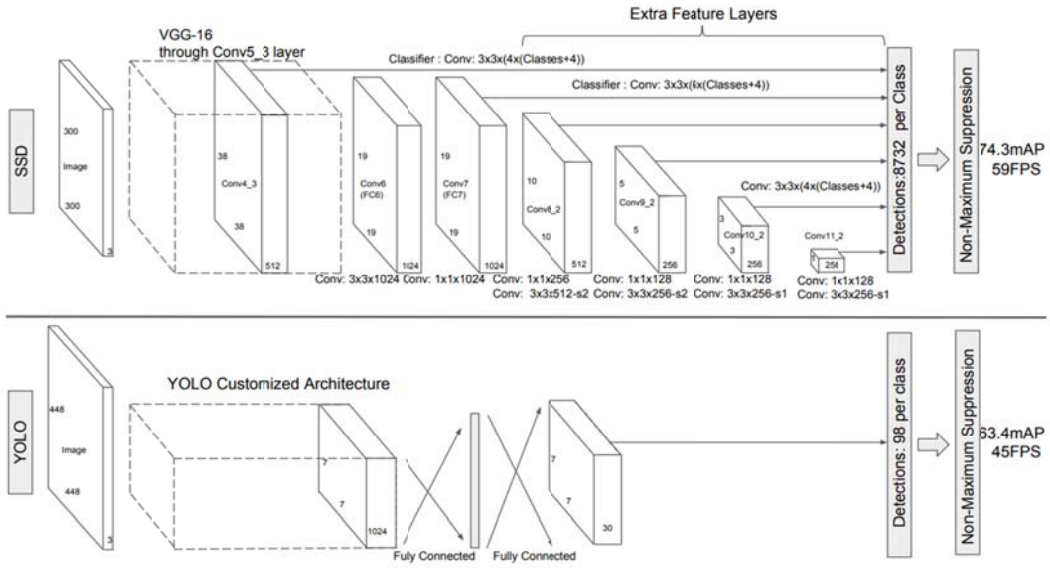
Reading Report based on "CornerNet: Detecting Objects as Paired Keypoints" by Law H, Deng J., "Feature Pyramid Networks for Object Detection" by W. Liu et al. and "SSD: Single Shot MultiBox Detector" by Lin T Y et al.

Author: Golobokov Mikhail

1. SSD

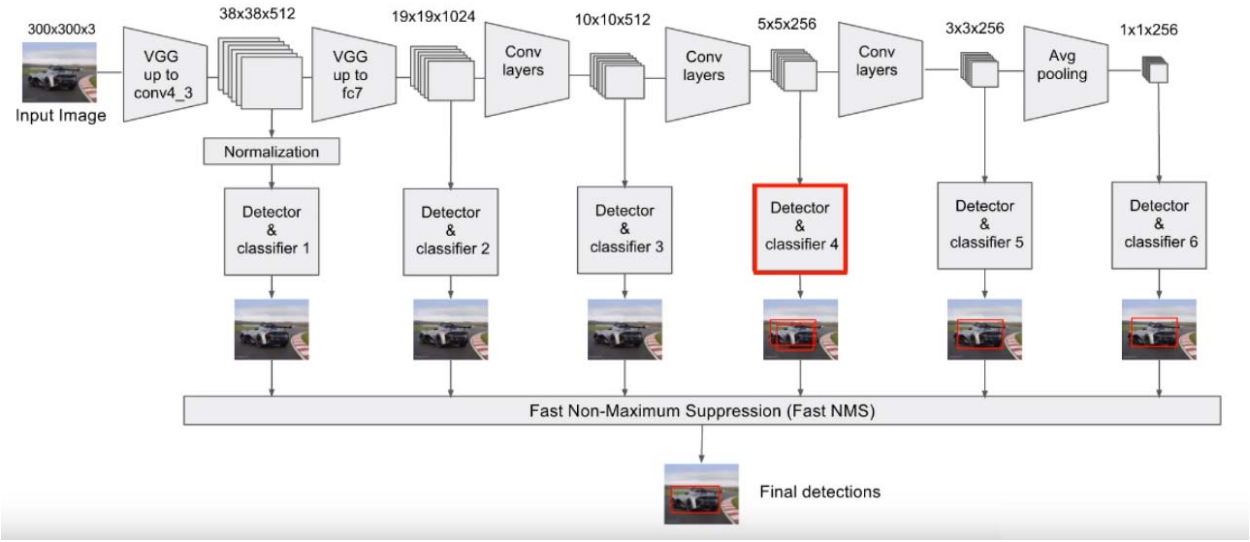
This method based on ideas and approaches taken from the method named YOLO.

The pic below shows a comparison between two single shot detection models SSD and YOLO:

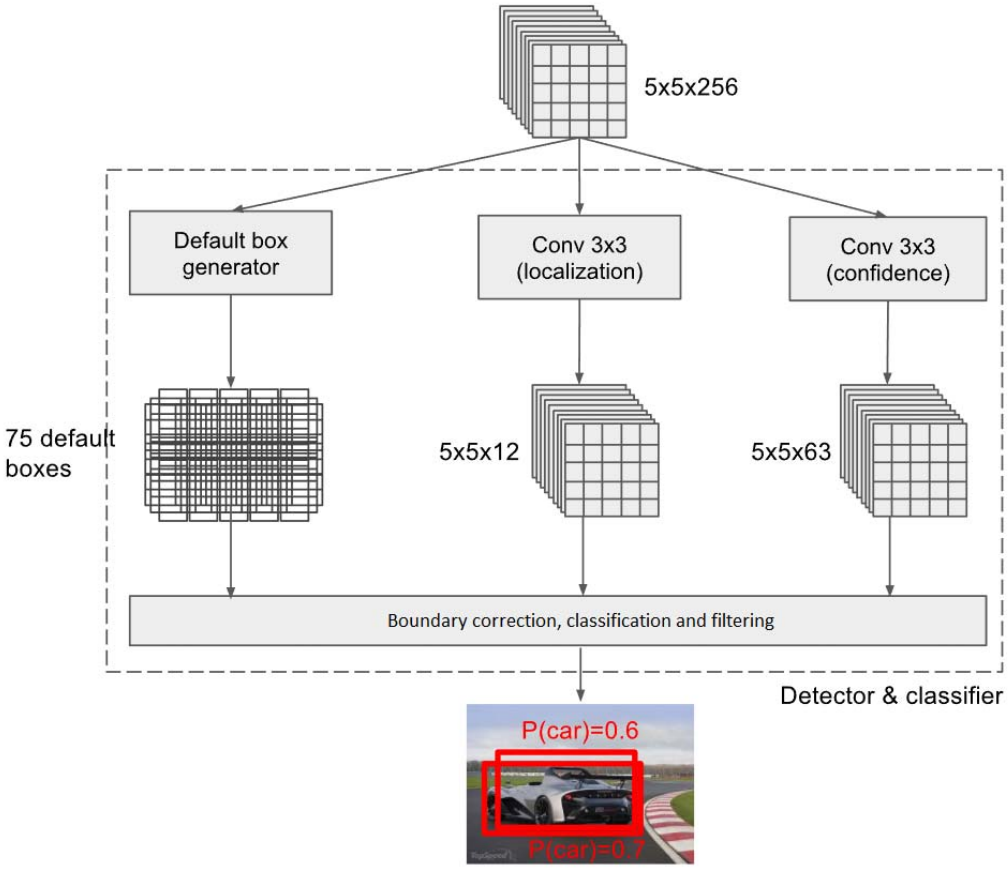


Let's introduce the base architecture of this method. We have an RGB image as input, then we use certain number of convolution layers, part of this conv layers are taken from VGG16, another ones are from SSD model. The spatial dimension slowly wanes until it becomes 1. Also, each middle feature map has normalization and «detector and classifier» block. The output of this block is detection – rectangles with class. After this, all outputs from «detector and classifier» blocks transfer to fast non-maximum suppression (Fast NMS). Fast NMS unites all these

rectangles to give final detections as the output of SSD method.



To find objects with different scale and size this method use about 6 «detector and classifier» blocks in case of slow decrease in spatial dimension. The closer block is to the beginning of the network, the smaller objects it can see or better to say, that chance of the detection small object is bigger when the mid-conv size layer is bigger. «Detector and classifier» block is a feature map and consists of default box generator, conv 3x3 (localization) layer and conv 3x3 (confidence) layer. This shown on pic below:



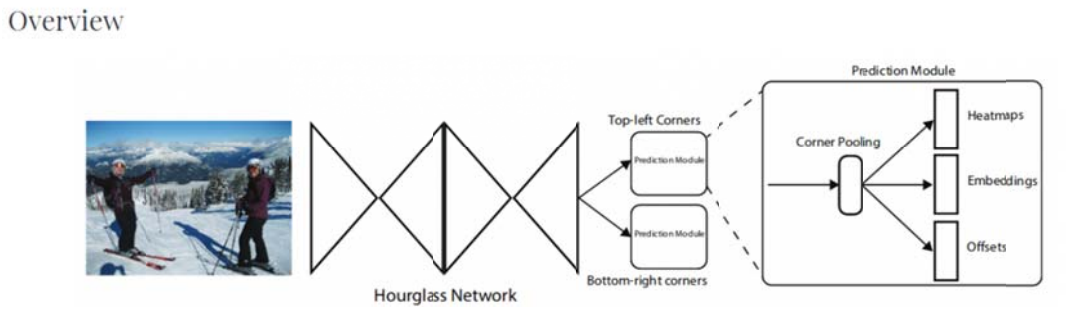
So, there are three main operations:

- Creating an anchor box
- To perform a convolution → positioning (localization)
- To make a convolution → classification (confidence)

To sum up, we can say that SSD method is able to work with speed of 58 fps with mAP 72%(7308 detections at a time to get final detection), which is nearly close to the cutting-edge two-stage Faster R-CNN.

2. CornerNet

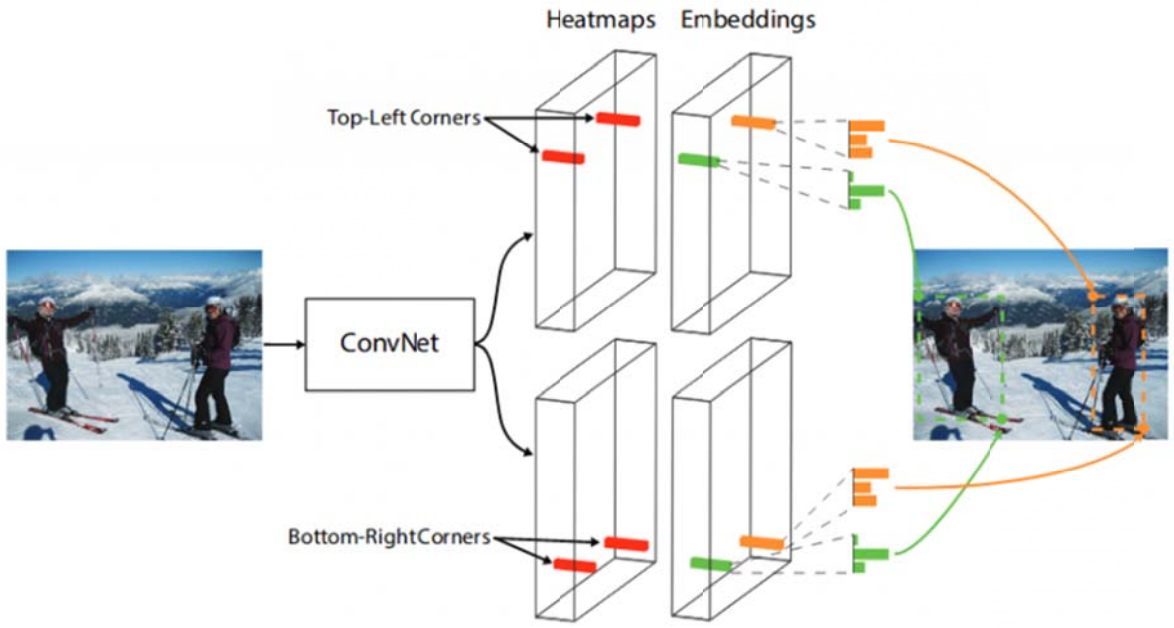
CornerNet is another single shot method of object detection. The idea of this method is to use corners (top-left corner and bottom-right corner as a pair of keypoints.



As we can see from pic above, CornerNet uses hourglass network as the backbone, followed by two prediction modules. One module is for the top-left corners, while the other one is for the bottom-right corners. Each module consists of its own corner pooling module to pool features from the hourglass network before predicting the heatmaps, embeddings, and offsets.

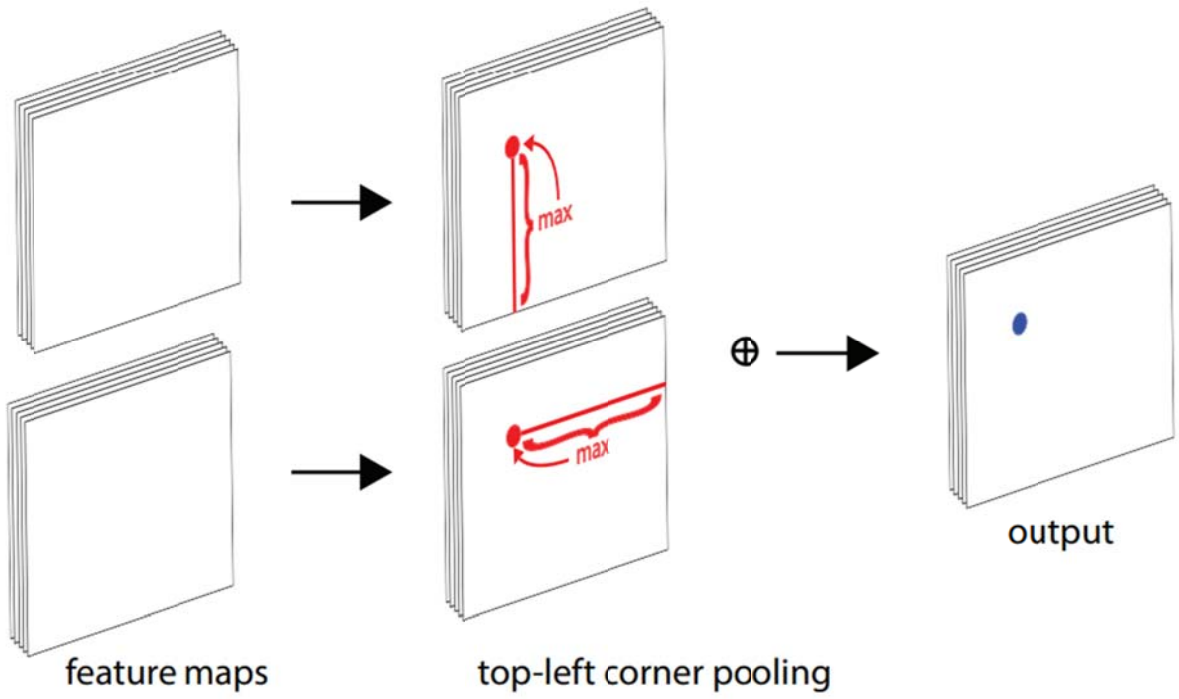
Both predicted sets of heatmaps have N channels, where N is the number of categories, and is of size HW. There is no background channel. Each channel is a binary mask indicating the locations of the corners for a class.

Additionally, for each detected corner the network predicts an embedding vector such that the distance between the embeddings of two corners from the same object is small (the pic below). An object bounding box is generated if the distance is less than a threshold. The bounding box is assigned a confidence score, which is equal to the average score of the corner pair.



To produce tighter bounding boxes, the network also predicts offsets to slightly adjust the locations of the corners. After that, a simple post-processing algorithm including NMS (Non-maximum Suppression) is used to generate bounding boxes from the heatmaps, embeddings, and offsets.

As we can see from the name of method: «Corner» means that we'll localize corners. For each channel, corner pooling takes the maximum values (red dots) in two directions (red lines), each from a separate feature map, and add the two maximums together (blue dot). From the obtained heatmaps the network is trained to predict similar embeddings for corners that belong to the same object.



In conclusion, CornerNet outperforms all one-stage detectors and achieves results competitive to two-stage detectors. The results are below.

Method	Backbone	AP	AP^{50}	AP^{75}	AP^{s}	AP^{m}	AP^{l}	AR^{1}	AR^{10}	AR^{100}	AR^{s}	AR^{m}	AR^{l}
Two-stage detectors													
DeNet (Tychsen-Smith and Petersson, 2017a)	ResNet-101	33.8	53.4	36.1	12.3	36.1	50.8	29.6	42.6	43.5	19.2	46.9	64.3
CoupleNet (Zhu et al., 2017)	ResNet-101	34.4	54.8	37.2	13.4	38.1	50.8	30.0	45.0	46.4	20.7	53.1	68.5
Faster R-CNN by G-RMI (Huang et al., 2017)	Inception-ResNet-v2 (Szegedy et al., 2017)	34.7	55.5	36.7	13.5	38.1	52.0	-	-	-	-	-	-
Faster R-CNN+++ (He et al., 2016)	ResNet-101	34.9	55.7	37.4	15.6	38.7	50.9	-	-	-	-	-	-
Faster R-CNN w/ FPN (Lin et al., 2016)	ResNet-101	36.2	59.1	39.0	18.2	39.0	48.2	-	-	-	-	-	-
Faster R-CNN w/ TDM (Shrivastava et al., 2016)	Inception-ResNet-v2	36.8	57.7	39.2	16.2	39.8	52.1	31.6	49.3	51.9	28.1	56.6	71.1
D-FCN (Dai et al., 2017)	Aligned-Inception-ResNet	37.5	58.0	-	19.4	40.1	52.5	-	-	-	-	-	-
Regionlets (Xu et al., 2017)	ResNet-101	39.3	59.8	-	21.7	43.7	50.9	-	-	-	-	-	-
Mask R-CNN (He et al., 2017)	ResNeXt-101	39.8	62.3	43.4	22.1	43.2	51.2	-	-	-	-	-	-
Soft-NMS (Bodla et al., 2017)	Aligned-Inception-ResNet	40.9	62.8	-	23.3	43.6	53.3	-	-	-	-	-	-
LH R-CNN (Li et al., 2017)	ResNet-101	41.5	-	-	25.2	45.3	53.1	-	-	-	-	-	-
Fitness-NMS (Tychsen-Smith and Petersson, 2017b)	ResNet-101	41.8	60.9	44.9	21.5	45.0	57.5	-	-	-	-	-	-
Cascade R-CNN (Cai and Vasconcelos, 2017)	ResNet-101	42.8	62.1	46.3	23.7	45.5	55.2	-	-	-	-	-	-
D-RFCN + SNIP (Singh and Davis, 2017)	DPN-98 (Chen et al., 2017)	45.7	67.3	51.1	29.3	48.8	57.1	-	-	-	-	-	-
One-stage detectors													
YOLOv2 (Redmon and Farhadi, 2016)	DarkNet-19	21.6	44.0	19.2	5.0	22.4	35.5	20.7	31.6	33.3	9.8	36.5	54.4
DSOD300 (Shen et al., 2017a)	DS/64-192-481	29.3	47.3	30.6	9.4	31.5	47.0	27.3	40.7	43.0	16.7	47.1	65.0
GRP-DSOD320 (Shen et al., 2017b)	DS/64-192-481	30.0	47.9	31.8	10.9	33.6	46.3	28.0	42.1	44.5	18.8	49.1	65.0
SSD513 (Liu et al., 2016)	ResNet-101	31.2	50.4	33.3	10.2	34.5	49.8	28.3	42.1	44.4	17.6	49.2	65.8
DSSD513 (Fu et al., 2017)	ResNet-101	33.2	53.3	35.2	13.0	35.4	51.1	28.9	43.5	46.2	21.8	49.1	66.4
RefineDet512 (single scale) (Zhang et al., 2017)	ResNet-101	36.4	57.5	39.5	16.6	39.9	51.4	-	-	-	-	-	-
RetinaNet800 (Lin et al., 2017)	ResNet-101	39.1	59.1	42.3	21.8	42.7	50.2	-	-	-	-	-	-
RefineDet512 (multi scale) (Zhang et al., 2017)	ResNet-101	41.8	62.9	45.7	25.6	45.1	54.1	-	-	-	-	-	-
CornerNet511 (single scale)	Hourglass-104	40.6	56.4	43.2	19.1	42.8	54.3	35.3	54.7	59.4	37.4	62.4	77.2
CornerNet511 (multi scale)	Hourglass-104	42.2	57.8	45.2	20.7	44.8	56.6	36.6	55.9	60.3	39.5	63.2	77.3

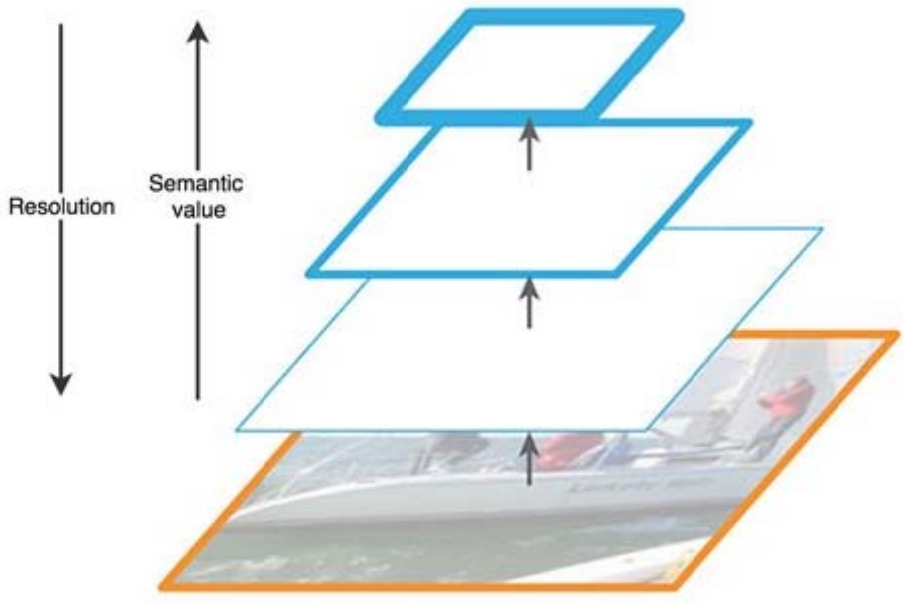
3. *Feature Pyramid Networks*

While experiments with Mask R-CNN along with the usual CNN ResNet-50/101 as backbone there were research of the feasibility of using Feature Pyramid Network. Theese experiments shown, that usage of FPN in backbone give Mask R-CNN gains in both accuracy and performance.

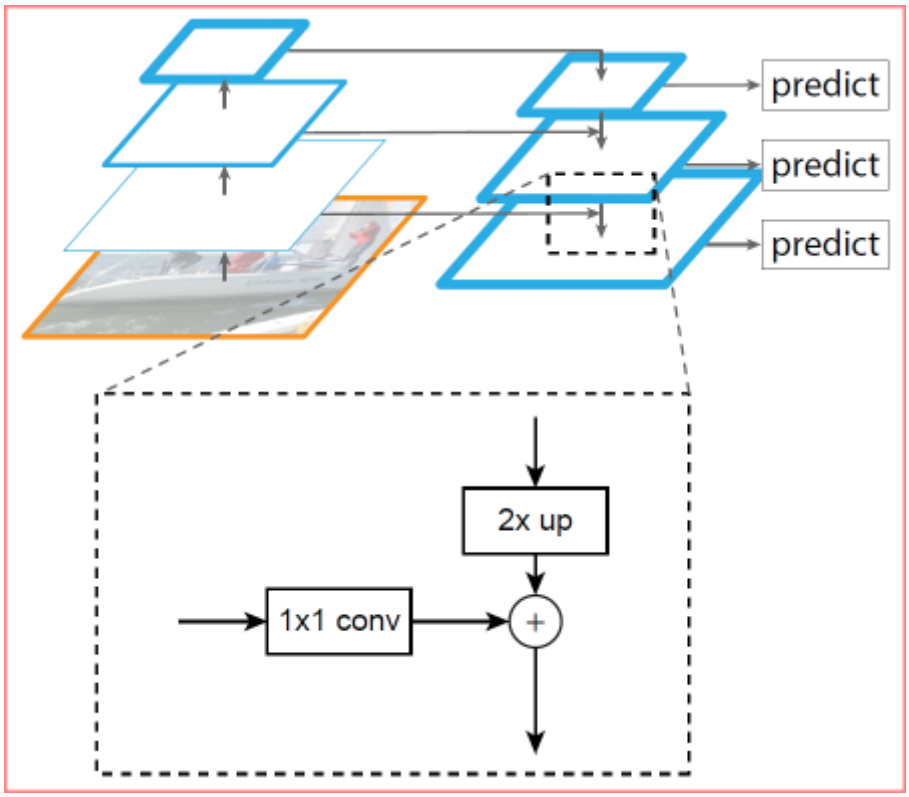
The appointment of Feature Pyramids - improving the quality of object detection, taking into account the large range of possible sizes.

The heat map in FPN, extracted by successive layers with CNN with decreasing dimension, considered as a kind of hierarchical «pyramid» called bottom-up pathway.

At the same time, feature maps of both the lower and upper levels of the pyramid have their advantages and disadvantages: the first have a high resolution, but low semantic, generalizing ability; the second-on the contrary:



The FPN architecture allows us to combine the advantages of the upper and lower levels by adding a top-down pathway and lateral connections. To do this, the map of each overlying layer is enlarged to the size of the underlying layer, and their contents are pieced together. The final predictions use the resulting maps of all levels. It looks like in pic below:



4. Comparison between one-stage and two-stage detectors

One-Stage detectors make the predictions about the object in the image on the grid, there is no intermediary task. So, they take an image as the input and pass it through a certain number of convolutional layers and find bounding boxes which are likely to contain the object and then do the prediction. These models use already trained image classifiers as their backbone network to identify the objects in the image. This results in a simpler and faster model, but lack the accuracy in comparison with two-stage detectors. Examples: SSD, YOLO.

In contrast with one-stage detectors, two-stage detectors use two stages to identify the objects in the image.

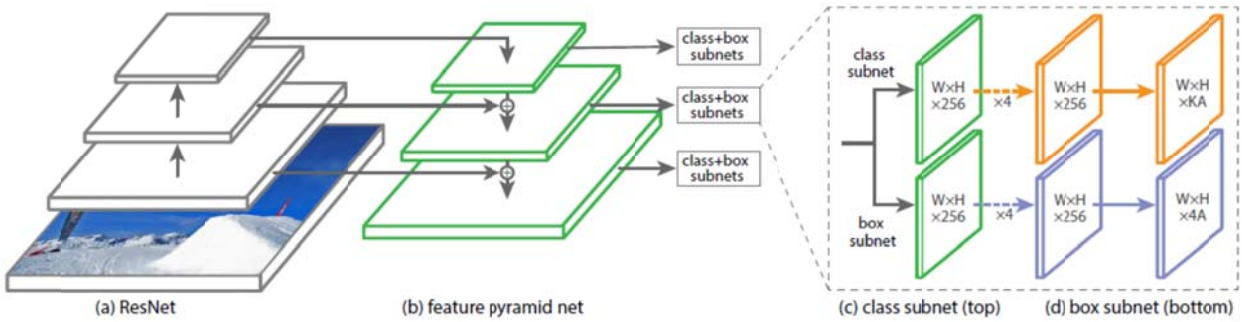
- The first stage contains some region proposal networks (RPN) which reduces the number the locations which are likely to contain the objects (sometimes also called Region of Interest (ROI)). So, in the second stage, we don't have to search over all the locations over the image to find the objects in the image but just the ones which are proposed by RPNs.

- Two-stage detectors also use some pre-trained image classifier as the backbone network

- In the second stage, classification is performed on the object locations and label the objects based on the confidence of the model
- In the second stage, classification is performed on the object locations and label the objects based on the confidence of the model

Speaking about ideas on how to combine these methods for better detection performance I'd like to say, that one excellent one-stage detector already exists.

RetinaNet is a single, unified network composed of a backbone network and two task-specific subnetworks: ResNet and Feature Pyramid Network (FPN) as the backbone networks.



Results of RetinaNet, a one-stage detector using focal loss were significant even on the challenging COCO dataset and beat every one-stage and two-stage detectors by a significant margin and delivered the state-of-the-art performance.

depth	scale	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	time
50	400	30.5	47.8	32.7	11.2	33.8	46.1	64
50	500	32.5	50.9	34.8	13.9	35.8	46.7	72
50	600	34.3	53.2	36.9	16.2	37.4	47.4	98
50	700	35.1	54.2	37.7	18.0	39.3	46.4	121
50	800	35.7	55.0	38.5	18.9	38.9	46.3	153
101	400	31.9	49.5	34.1	11.6	35.8	48.5	81
101	500	34.4	53.1	36.8	14.7	38.5	49.1	90
101	600	36.0	55.2	38.7	17.4	39.6	49.7	122
101	700	37.1	56.6	39.8	19.1	40.6	49.4	154
101	800	37.8	57.5	40.8	20.2	41.1	49.2	198