Deep Learning & Engineering Applications

5. Object Detection

JIDONG J. YANG

COLLEGE OF ENGINEERING

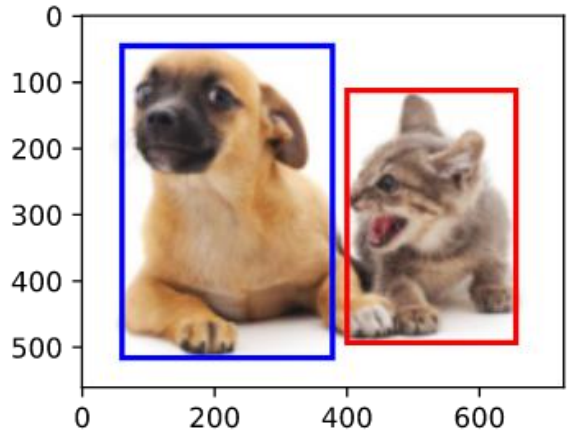
UNIVERSITY OF GEORGIA

Object Detection

A core problem in computer vision.

Object detection not only recognizes all the objects of interest in the image, but also their positions.

The position is generally represented by a rectangular bounding box.



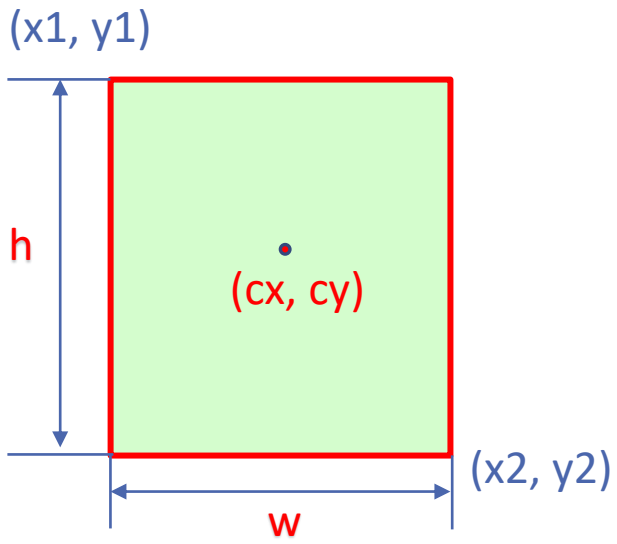
Applications

- Security systems, self-driving cars, drones, robots, etc.

Two Bounding Box Representations

(cx, cy, w, h)

(x1, y1, x2, y2)




```python
def box_corner_to_center(boxes):
    """Convert from (upper-left, lower-right) to (center, width, height)."""
    x1, y1, x2, y2 = boxes[:, 0], boxes[:, 1], boxes[:, 2], boxes[:, 3]
    cx = (x1 + x2) / 2
    cy = (y1 + y2) / 2
    w = x2 - x1
    h = y2 - y1
    boxes = torch.stack((cx, cy, w, h), axis=-1)
    return boxes

def box_center_to_corner(boxes):
    """Convert from (center, width, height) to (upper-left, lower-right)."""
    cx, cy, w, h = boxes[:, 0], boxes[:, 1], boxes[:, 2], boxes[:, 3]
    x1 = cx - 0.5 * w
    y1 = cy - 0.5 * h
    x2 = cx + 0.5 * w
    y2 = cy + 0.5 * h
    boxes = torch.stack((x1, y1, x2, y2), axis=-1)
    return boxes
```

How to detect objects in image?

Extract a set of robust features from input images (e.g., Haar, SIFT, HOG, and **convolutional features**).

Use classifiers or localizers to identify objects in the feature space.

- Sliding Window Search?
- Branch & Bound Search [e.g., Lampert et al., 2009]
- Selective Search [e.g., Uijlings et al., 2013]

[Lampert et al., 2009] https://pub.ist.ac.at/~chl/papers/lampert-pami2009.pdf

[Uijlings et al., 2013] https://ivi.fnwi.uva.nl/isis/publications/bibtexbrowser.php?key=UijlingsIJCV2013&bib=all.bib

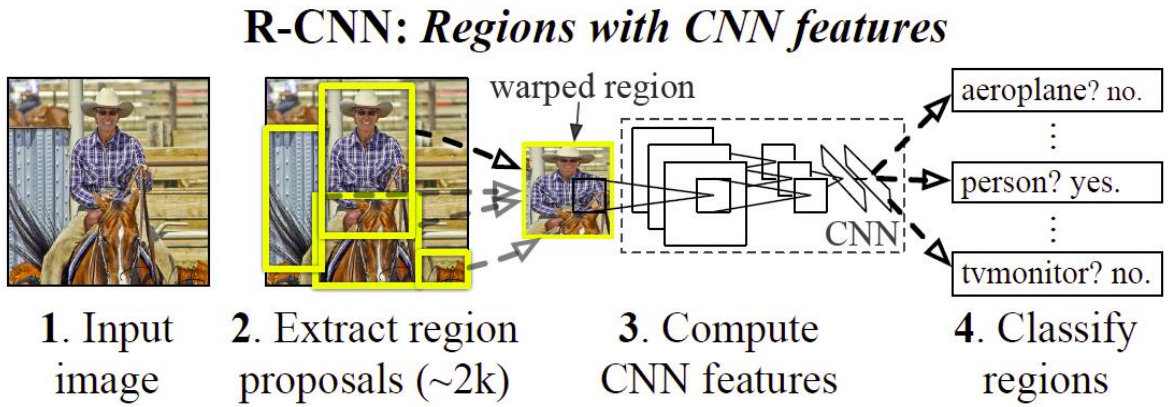
R-CNN

Creates region proposals or bounding boxes through selective search (step 2).

After the proposals are created, R-CNN warps the region to a standard square size (224 x 224) and passes it to a CNN, a modified version of AlexNet, to compute features (step 3).

A Support Vector Machine (SVM) is then applied on the final layer of the CNN to classify the regions by types of objects (step 4).

Once a region is classified to a certain type of objects, a linear regression is performed on the coordinates of the region to output a tighter bounding box.



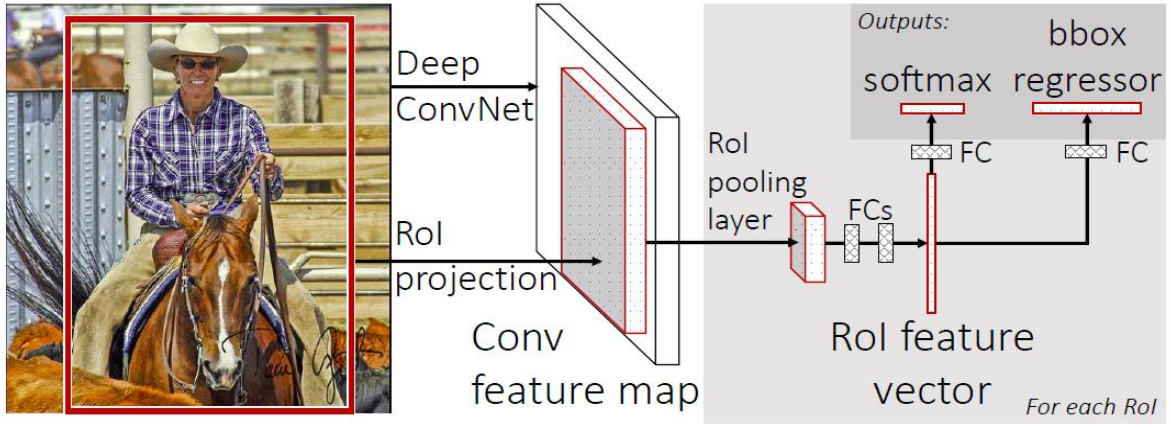
- In R-CNN, three different models are trained separately:
  1. A CNN to extract image features
  2. A SVM for classification
  3. A regression model to tighten the bounding boxes.

Complex pipelines

- Slow and hard to optimize because each individual component must be trained separately.

[Girshick, et al., 2014] https://arxiv.org/pdf/1311.2524.pdf

Fast R-CNN



Instead of feeding each warped region into CNN, feed the input image directly to the CNN to generate a convolutional feature map (i.e., **one-single forward pass**).

Extract features from the feature map **per the region of proposals**. Then, the features inside any valid region of interest are max-pooled into a smaller feature map with a fixed spatial extent (e.g., 7x7).

Fast R-CNN replaced the SVM classifier with a softmax layer for the classification task and added a linear regression layer in parallel to the softmax layer to generate bounding box coordinates.

Use multi-task loss (combine classification loss and localization loss). By doing so, a single network can handle both classification and localization.

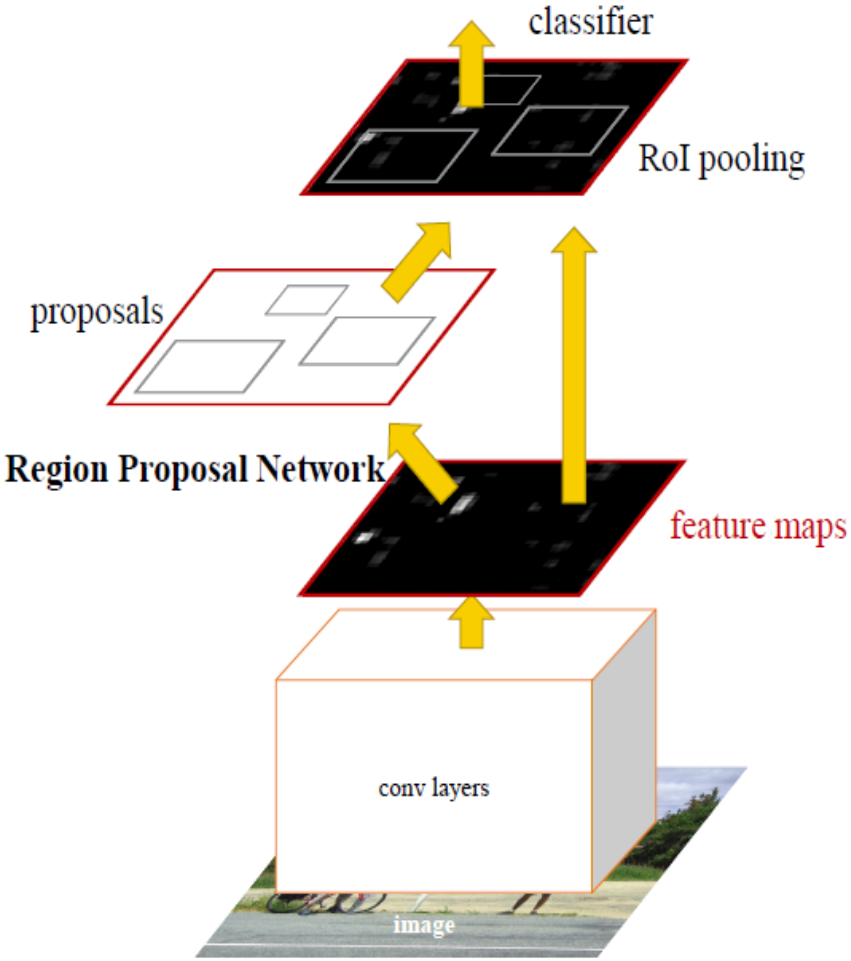
[Girshick, et al., 2015] https://arxiv.org/pdf/1504.08083.pdf

Faster R-CNN

Regardless of all the improvements, there is still one bottleneck in the Fast R-CNN, which is the region proposal by selective search.

Instead of executing a separate algorithm for selective search, **use the image features extracted from the forward pass of the CNN to generate region proposals**. This results in nearly cost-free region proposals.

A single, unified network for object detection.



[Ren, et al., 2015] https://arxiv.org/abs/1506.01497

Anchor Boxes

Usually, a large number of regions are sampled in the input image to determine whether these regions contain objects of interest. (Region of Interest, ROI).

Adjust the boundaries of the regions so as to predict the ground-truth bounding boxes of the objects more accurately.

Different models may adopt different region sampling schemes.

Anchor Boxes is one of popular approaches that have been used in Faster R-CNN, Single Shot MultiBox Detector and YOLO.

Anchor Boxes are multiple bounding boxes with **varying scales and aspect ratios centered on each pixel**.

Anchor Boxes

Scale, $s \in (0,1]$

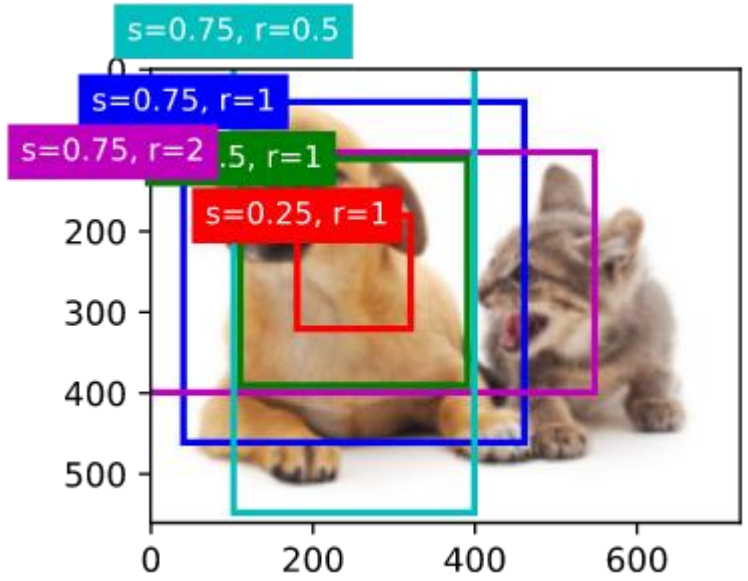
Aspect ratio (w/h), $r > 0$

Apply s , r, we have

- $Anchor\ box\ width = ws\sqrt{r}$
- $Anchor\ box\ height = \frac{hs}{\sqrt{r}}$

Pick series of $(s_1, s_2, \ldots, s_n)\ and\ (r_1, r_2, \ldots, r_m)$

The results in a total of $whnm$ anchor boxes for the input image.



Intersection over Union (IoU)

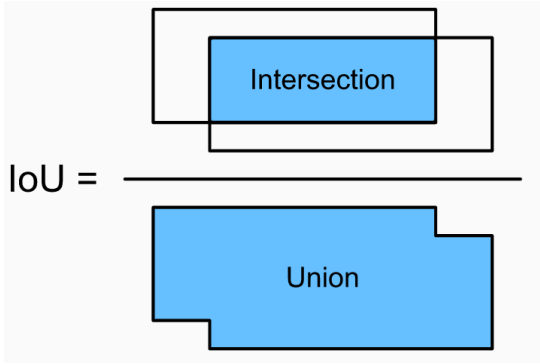
Quantify the quality of detection by measuring the “similarity” between the anchor box and the ground-truth bounding box.

Jaccard index for measuring the similarity between two sets, A and B.

IoU is just an application of Jaccard index to bounding boxes.

IoU: [0,1]

$$J(\mathcal{A},\mathcal{B}) = \frac{|\mathcal{A}\cap\mathcal{B}|}{|\mathcal{A}\cup\mathcal{B}|}$$

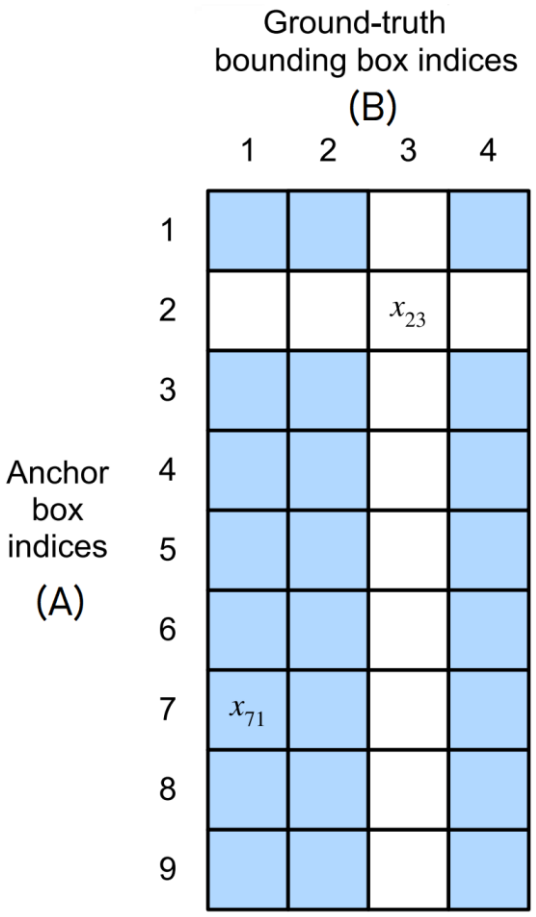


Assign ground-truth bounding boxes (B) to anchor boxes (A)

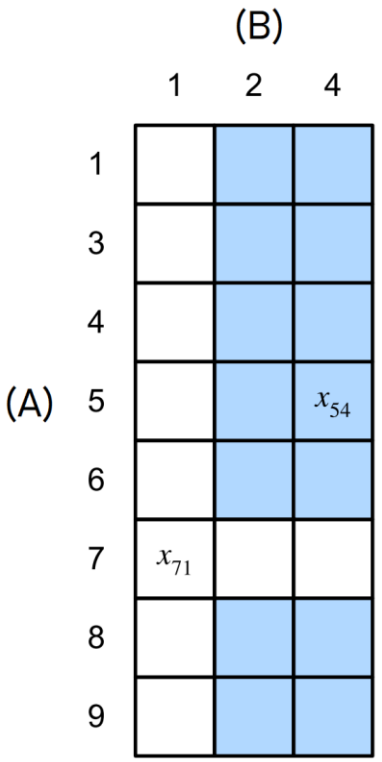
Positive anchor boxes

Negative anchor boxes

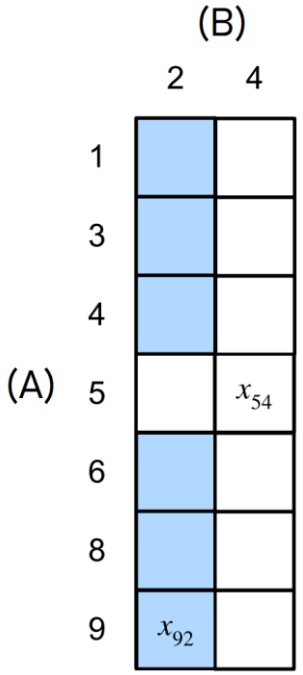
$x_{23} > \cdots > x_{71} > \cdots > x_{54} > \cdots > x_{92}$



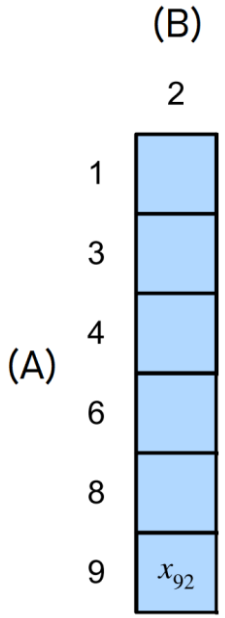
$B_3 \rightarrow A_2$



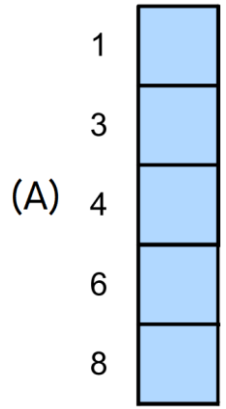
$B_1 \rightarrow A_7$



$B_4 \rightarrow A_5$



$B_2 \rightarrow A_9$



For A_i, find the ground-truth bounding box B_j with the largest IoU and assign B_j to A_i only if the IoU > a predefined threshold.

Labeling classes and offsets for each anchor box

Class Labeling

- The class of the anchor box A will be labeled as that of the ground-truth box B assigned to A.

Offset Labeling

- Given varying positions and sizes of different boxes in the dataset, transformations are applied to those relative positions and sizes.
- This leads to more uniformly distributed offsets that are easier to fit.
- For example, the following transformation can be used.

$$\left(\frac{\frac{x_b - x_a}{w_a} - \mu_x}{\sigma_x}, \frac{\frac{y_b - y_a}{h_a} - \mu_y}{\sigma_y}, \frac{\log \frac{w_b}{w_a} - \mu_w}{\sigma_w}, \frac{\log \frac{h_b}{h_a} - \mu_h}{\sigma_h} \right)$$

Default values

$$\mu_x = \mu_y = \mu_w = \mu_h = 0$$

$$\sigma_x = \sigma_y = 0.1 \quad \sigma_w = \sigma_h = 0.2$$

Loss function

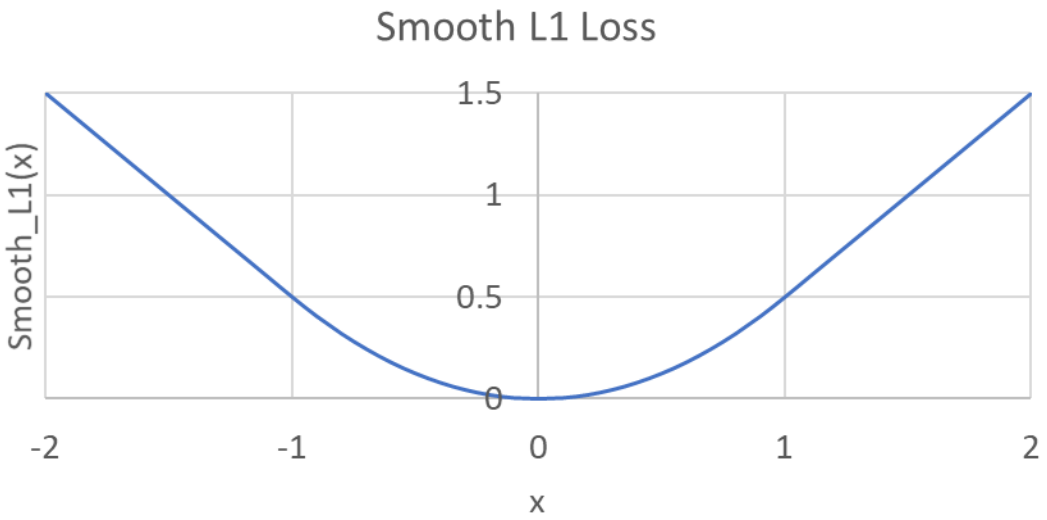
$$L(x, c, l, g) = \frac{1}{N}(L_{conf}(x, c) + \alpha L_{loc}(x, l, g))$$

where,

N = *the number of matched default boxes.*

L_{conf} = *cross_entropy loss*

L_{loc} = ***Smooth L_1 loss***



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

[Liu et al., 2016] https://arxiv.org/pdf/1512.02325.pdf

Predicting Bounding Boxes with Non-Maximum Suppression

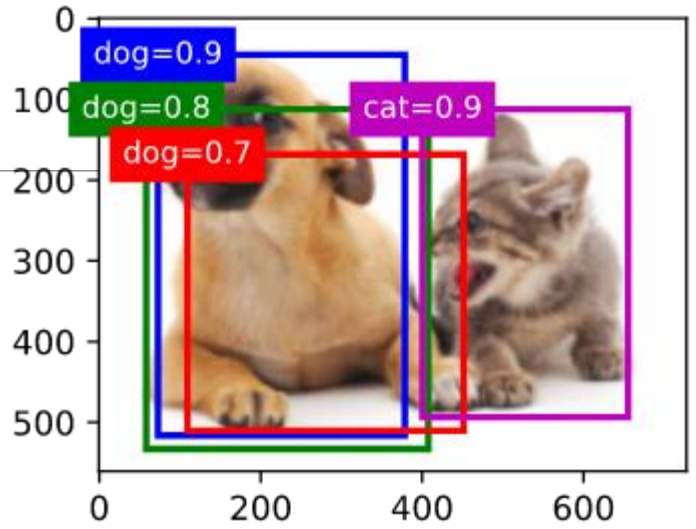
For an image, all the predicted positive bounding boxes are sorted by confidence (class probability) in descending order. This gives a sorted list L.

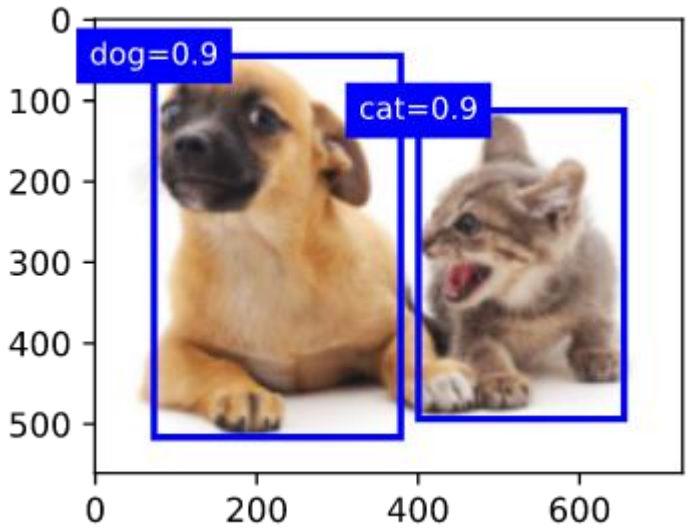
The bbox with the highest confidence (B1) is used as a basis to remove all non-basis predicted bounding boxes whose IoU with B1 exceeds a predefined threshold ϵ from L. (i.e., the boxes with non-maximum confidence scores are suppressed).

Do same with the bbox of the 2nd highest confidence.

Repeat the above process through all the predicted bboxes in L.

Output all bboxes in L.





Multiscale Anchor Boxes

Too many anchor boxes to compute if they are generated for every pixel.

We could uniformly sample a small portion of pixels to generate anchor boxes.

Intuitively, smaller objects are more likely (more ways) to appear on an image than larger ones.

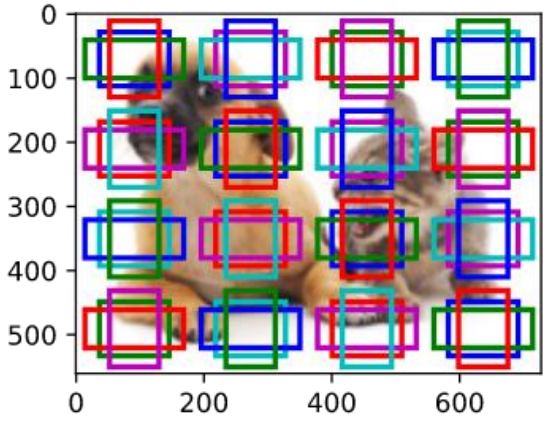
Sample more regions using smaller anchor boxes to detect smaller objects, while sample fewer regions for larger objects.

Sample in feature maps.

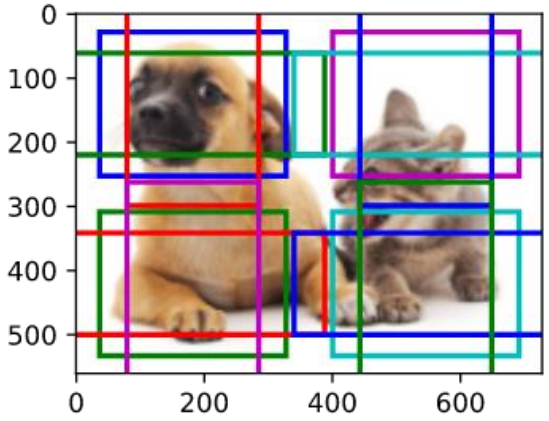
Leverage layer-wise representations of images at multiple levels by CNNs for multiscale object detection.

Multiscale Anchor Boxes

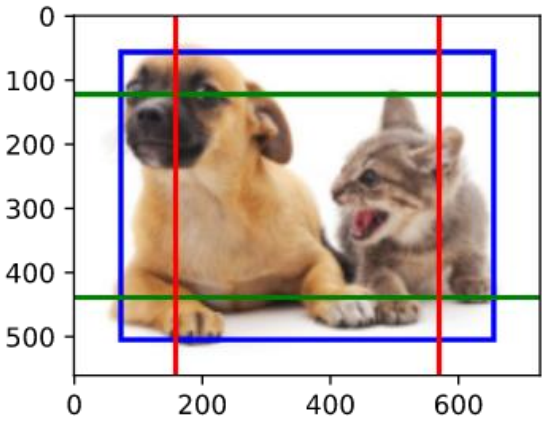
s=0.15



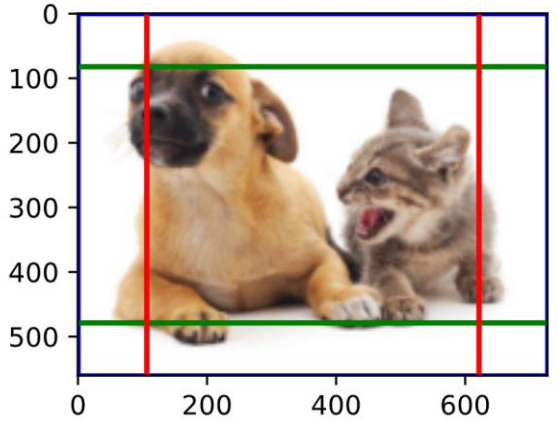
s=0.4



s=0.8



s=1.0



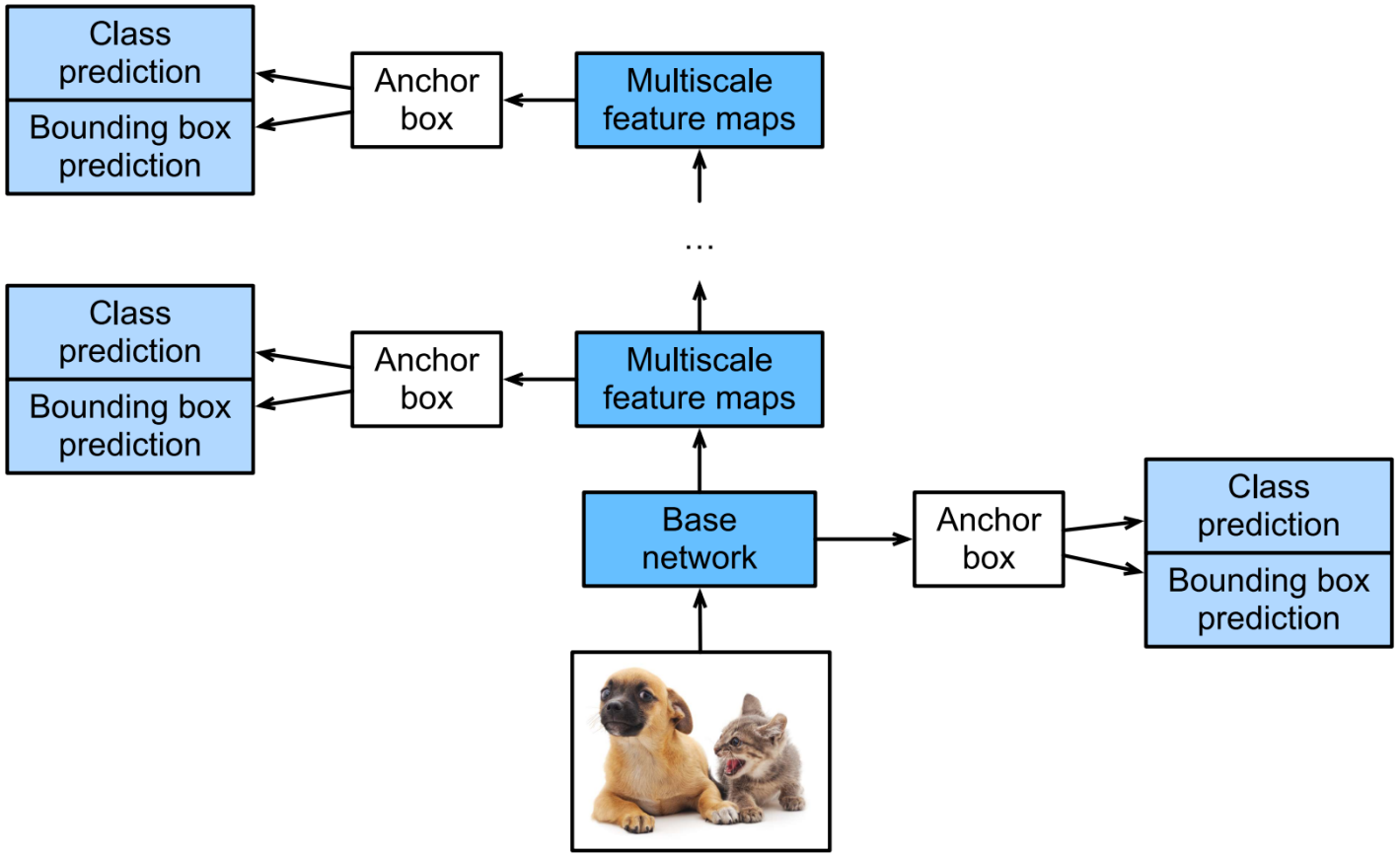
r = [1, 2, 0.5]

Single Shot Multibox Detection (SSD)

A multiscale object detection model

A base network is used to extract features from input images.

followed by several multiscale feature map blocks



[Liu et al., 2016] https://arxiv.org/pdf/1512.02325.pdf

Class Prediction Layer

A convolutional layer without altering width or height of feature maps (e.g., kernel size = 3, padding =1)

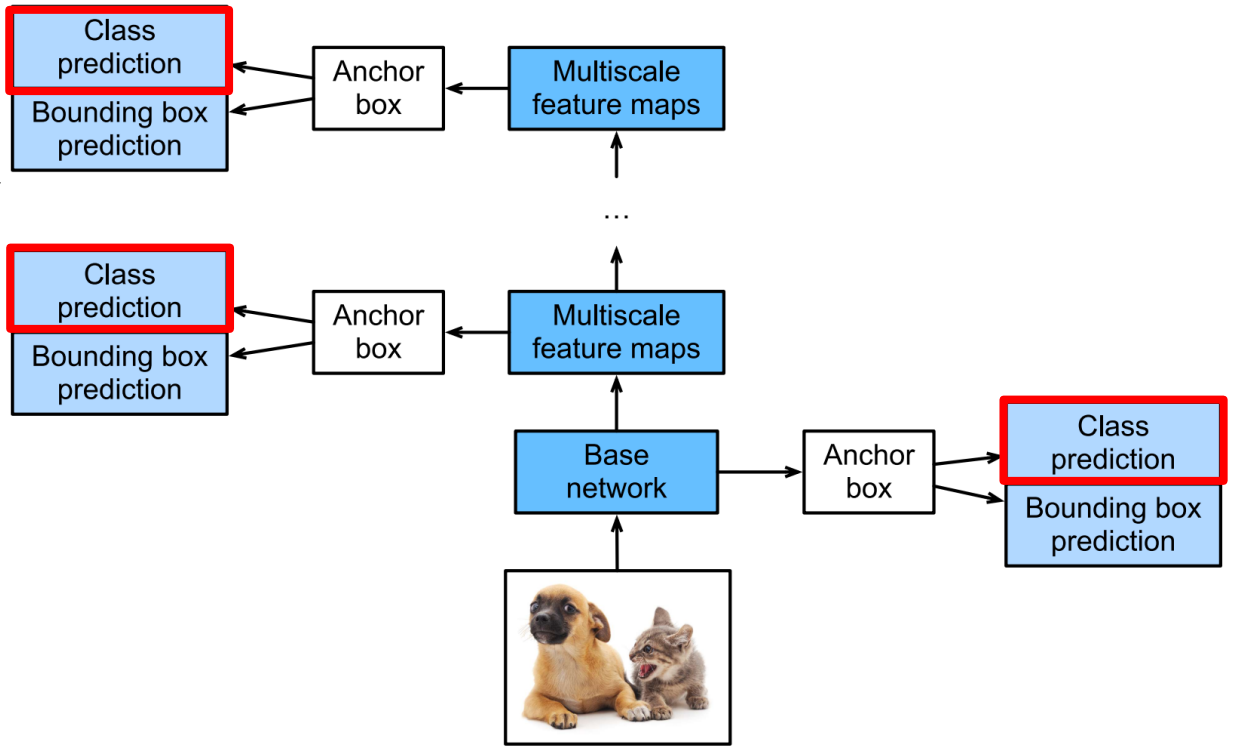
num_classes = q

num_anchors = a

Each anchor box has q+1 classes (class 0 is background)

Total output channels = a(q+1)

Channels of the output feature maps at any spatial position (x, y) represent class predictions for all the anchor boxes centered on (x, y) of the input feature maps.

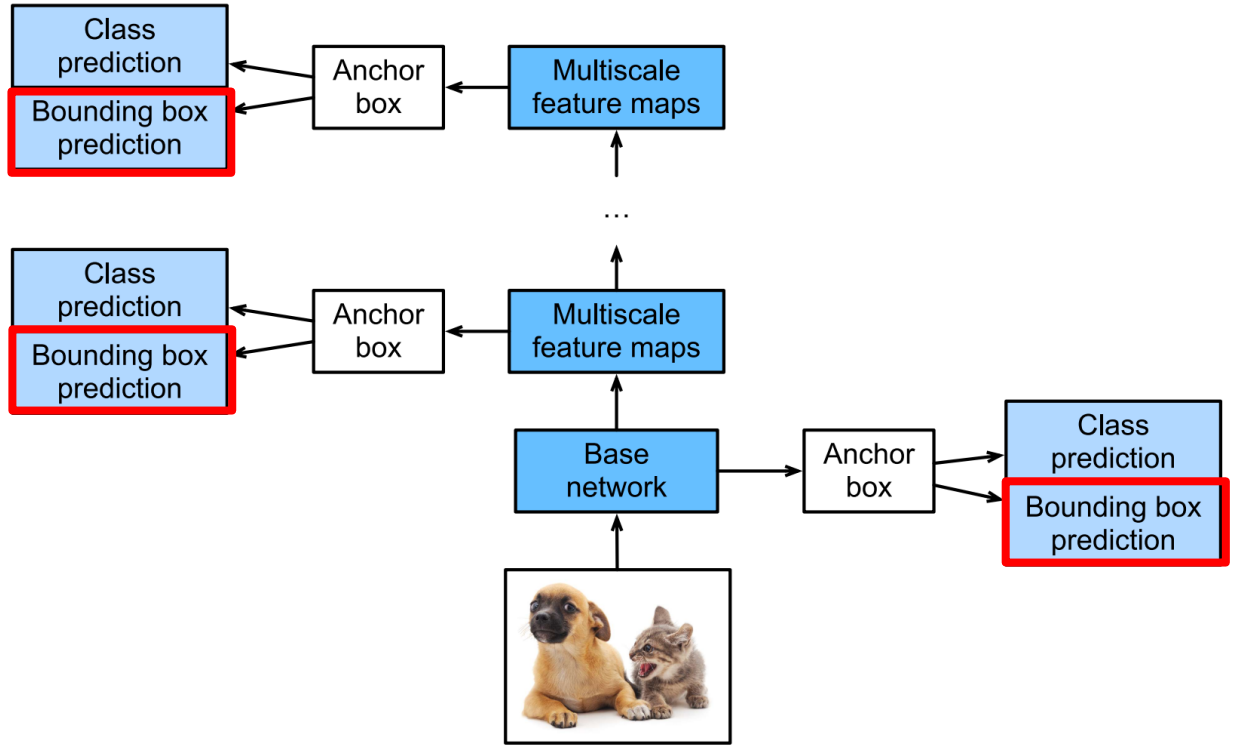



```
def cls_predictor(num_inputs, num_anchors, num_classes):
    return nn.Conv2d(num_inputs, num_anchors * (num_classes + 1),
                     kernel_size=3, padding=1)
```

Bounding Box Prediction Layer

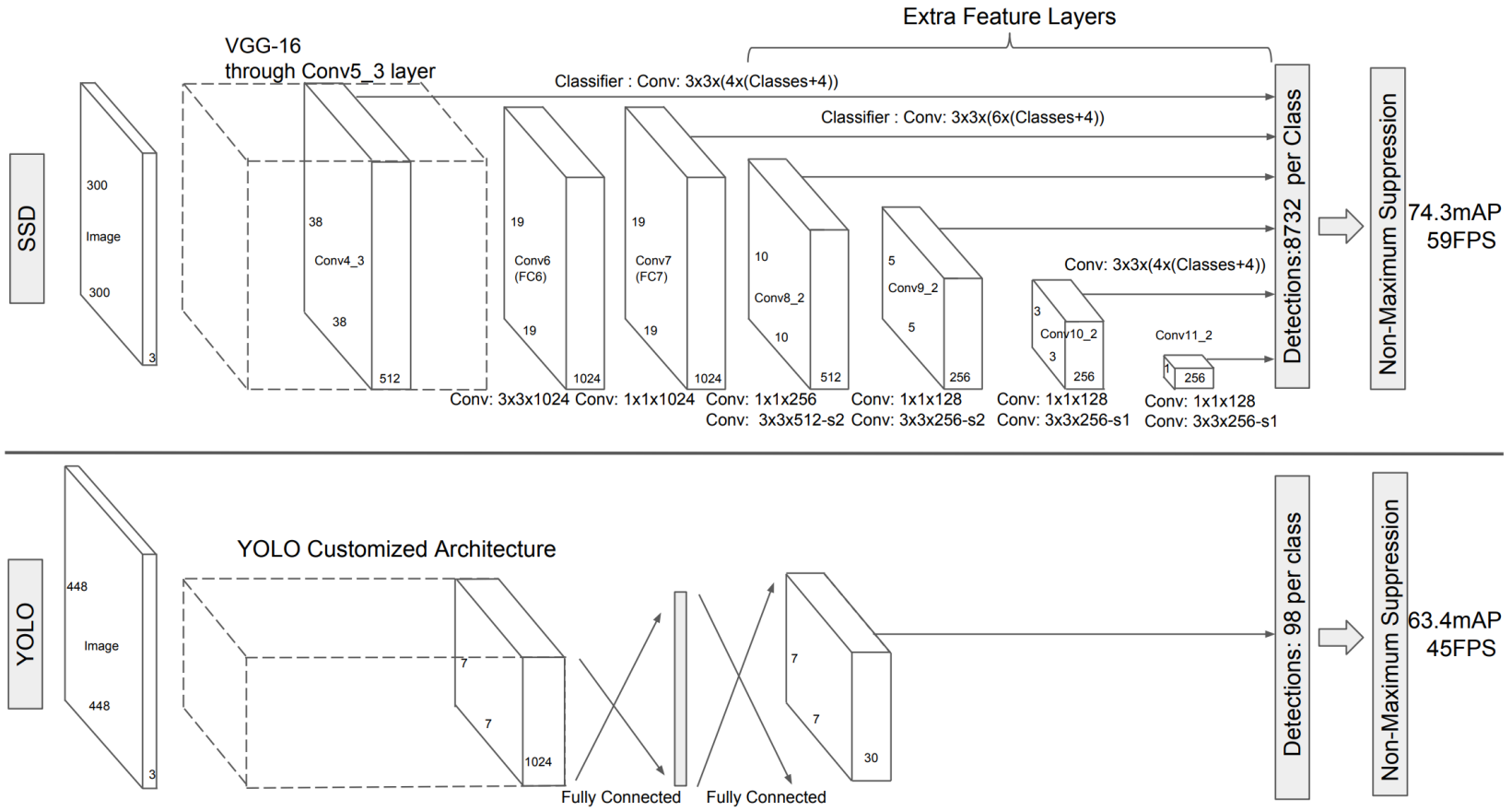
Similar to that of the class prediction layer.

Instead of q+1 classes, the number of outputs for each anchor box = 4 offsets




```
def bbox_predictor(num_inputs, num_anchors):
    return nn.Conv2d(num_inputs, num_anchors * 4, kernel_size=3, padding=1)
```

SSD vs YOLO



[Liu et al., 2016] https://arxiv.org/pdf/1512.02325.pdf

You Only Look Once (YOLO)

You Only Look Once (YOLO) at an image to predict what objects are present and where they are.

Reframe object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities.

Extremely fast at test time since it only requires a single network evaluation, unlike classifier-based methods.

Implicitly encodes contextual information about classes as well as their appearance.

[Redmon et al., 2015] https://arxiv.org/pdf/1506.02640.pdf

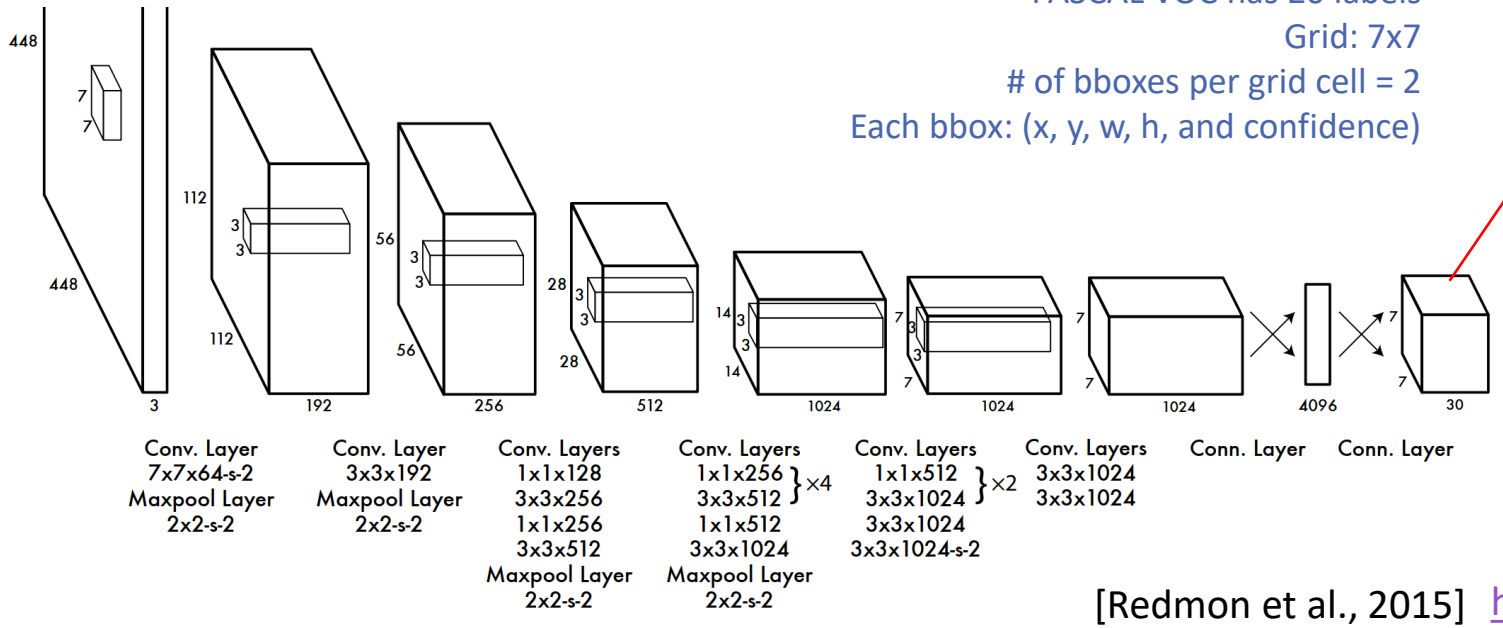
YOLO

Inspired by the GoogLeNet

24 convolutional layers followed by 2 fully connected layers

Instead of the inception modules used by GoogLeNet, use 1 × 1 convolution followed by 3 × 3 convolutional layers

Alternating 1 × 1 conv_layers reduce the features space.



PASCAL VOC has 20 labels
Grid: 7x7
# of bboxes per grid cell = 2
Each bbox: (x, y, w, h, and confidence)

$$7 \times 7 \times (2 * 5 + 20)$$

$$S \times S \times (B * 5 + C)$$

[Redmon et al., 2015] https://arxiv.org/pdf/1506.02640.pdf

YOLO Training

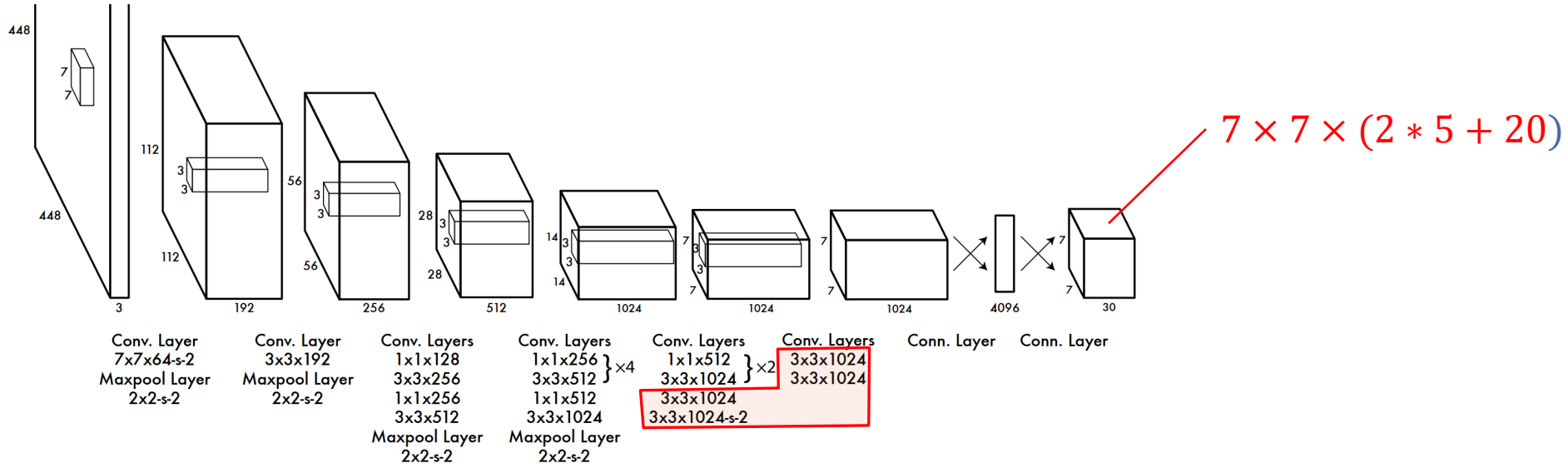
Pretrain the convolutional layers (first 20) on the ImageNet 1000-class dataset (with a 224x224 input resolution, note: this is for classification).

Add four convolutional layers and two fully connected layers with randomly initialized weights.

A dropout layer (p = 0.5) after the first connected layer.

The final layer predicts both class probabilities and bounding box coordinates.

Data augmentation: random scaling, translations (up to 20% of the original image size), randomly adjusted exposure and saturation (up to a factor of 1.5) in the HSV color space.



Limitations of YOLO

Imposes strong spatial constraints on bounding box predictions since each grid cell only predicts two boxes and can only have one class.

Uses relatively coarse features for predicting bounding boxes since the architecture has multiple down-sampling layers from the input image

Main source of error is incorrect localizations

The loss function treats errors the same in small bounding boxes versus large bounding boxes. A small error in a small box has a much greater effect on IOU than a large box.

Struggles with small objects that appear in groups (e.g., groups of pedestrians, flocks of birds).

YOLO v2

Batch normalization. with batch normalization the dropout can be removed without overfitting.

Fine tune the classification network at the full 448 × 448 resolution for 10 epochs on ImageNet.

Remove the FC layers and use anchor boxes to predict bounding boxes.

Shrink the network to operate on 416 input images instead of 448 (to get an odd number of location, i.e. 416/32 = 13)

Predict class and objectness for every anchor box.

Run k-means clustering on the training set bounding boxes to automatically find good anchor box priors (k = 5).

Use fine grain features for detection by adding a passthrough layer that brings features from an earlier layer at 26 × 26 resolution. (In contrast to the multiscale feature maps used by SSD)

Multi-Scale Training

- The network can be dynamically adjusted because it only has convolutional and pooling layers.
- Since the network downsample by 32, randomly chooses a new image dimension size (a multiple of 32: 320, 352, …, 608) every 10 batches.

https://arxiv.org/pdf/1612.08242.pdf

YOLO v2 (2016)

Pretrain Darknet-19 Backbone for classification with ImageNet.

Modify the Network for detection:

- Remove the last convolutional layer
- Add three 3 × 3 conv_layers with 1024 filters
- Add a passthrough layer from the final 3 × 3 × 512 layer to use fine grain features.
- Add a final 1 × 1 conv_layer with the number of outputs needed for detection [i.e., 5 anchor boxes x (20 classes + 5 coordinates) =125].

Darknet-19

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

YOLOv3

Predicts boxes at 3 different scales (32, 16, 8)

Upsampling + feature merging

Residual/skip connections

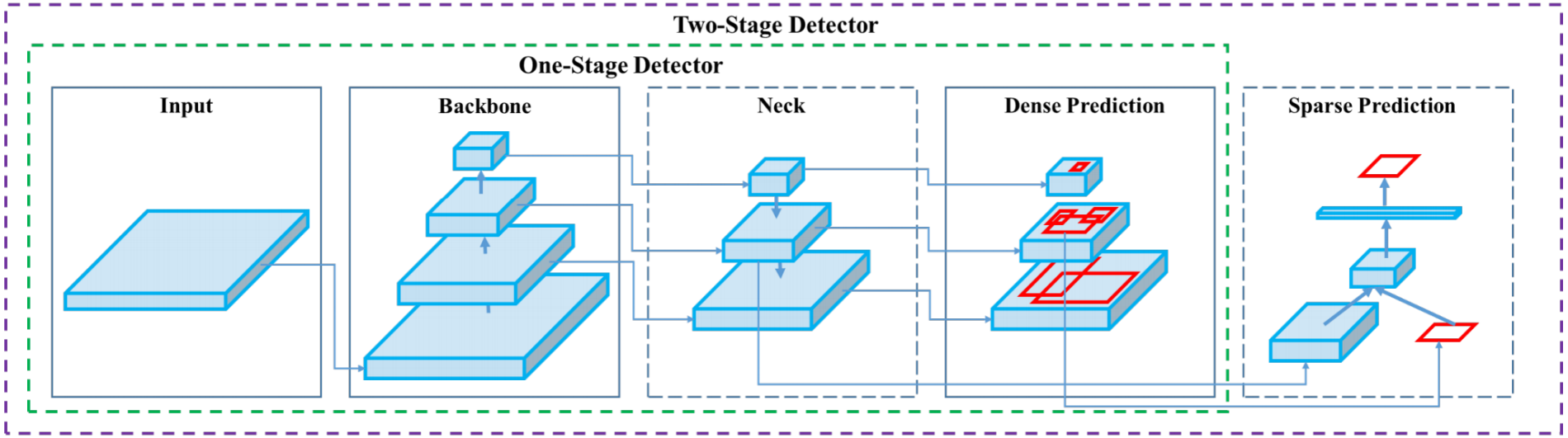
Use independent logistic classifiers (not softmax). Use binary cross-entropy loss for the class predictions during training.

Feature Extractor (Darknet – 53)

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

[Redmon & Farhadi, 2018] https://arxiv.org/pdf/1804.02767.pdf

Object Detector



Input: { Image, Patches, Image Pyramid, … }

Backbone: { VGG16 [68], ResNet-50 [26], ResNeXt-101 [86], Darknet53 [63], … }

Neck: { FPN [44], PANet [49], Bi-FPN [77], … }

Head:

Dense Prediction: { RPN [64], YOLO [61, 62, 63], SSD [50], RetinaNet [45], FCOS [78], … }

Sparse Prediction: { Faster R-CNN [64], R-FCN [9], … }

{Bochkovskiy et al., 2020] https://arxiv.org/pdf/2004.10934.pdf

YOLOv4

Training requires only one conventional GPU.

Backbone: CSPDarknet53

Neck: SPP, PAN

Head: YOLOv3

Bag of Freebies (BoF) for backbone: CutMix and Mosaic data augmentation, DropBlock regularization, Class label smoothing

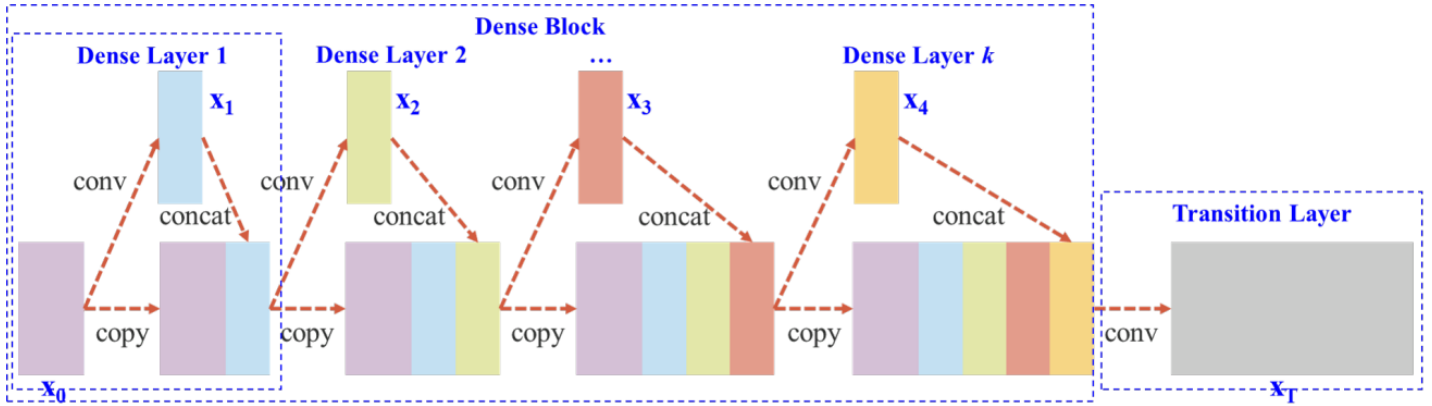
Bag of Specials (BoS) for backbone: Mish activation, Cross-stage partial connections (CSP), Multiinput weighted residual connections (MiWRC)

Bag of Freebies (BoF) for detector: CIoU-loss, CmBN, DropBlock regularization, Mosaic data augmentation, Self-Adversarial Training, Eliminate grid sensitivity, Using multiple anchors for a single ground truth, Cosine annealing scheduler, Optimal hyperparameters, Random training shapes

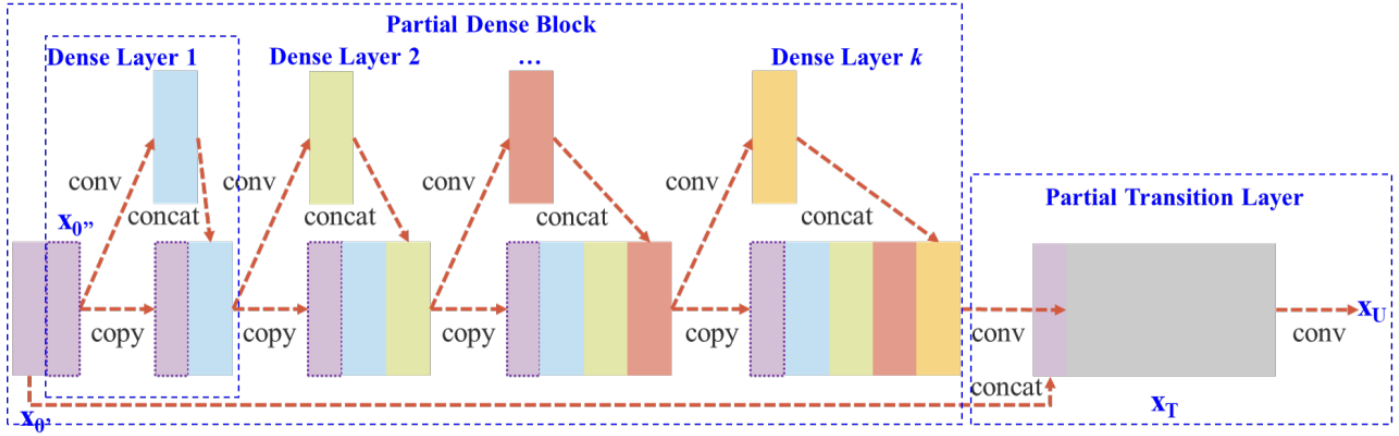
Bag of Specials (BoS) for detector: Mish activation, SPP-block, SAM-block, PAN path-aggregation block, DIoU-NMS

{Bochkovskiy et al., 2020] https://arxiv.org/pdf/2004.10934.pdf

CSPDenseNet



(a) DenseNet



(b) Cross Stage Partial DenseNet

[Wang et al., 2019] https://arxiv.org/pdf/1911.11929.pdf

Feature Pyramid Network (FPN)

The goal is to leverage a ConvNet's pyramidal feature hierarchy, which has semantics from low to high levels, and build a feature pyramid with high-level semantics throughout.

To achieve this goal, we rely on an architecture that combines low-resolution, semantically strong features with high-resolution, semantically weak features via a top-down pathway and lateral connections

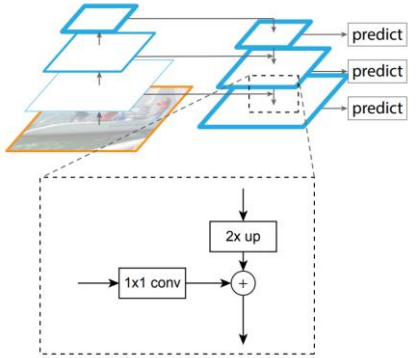
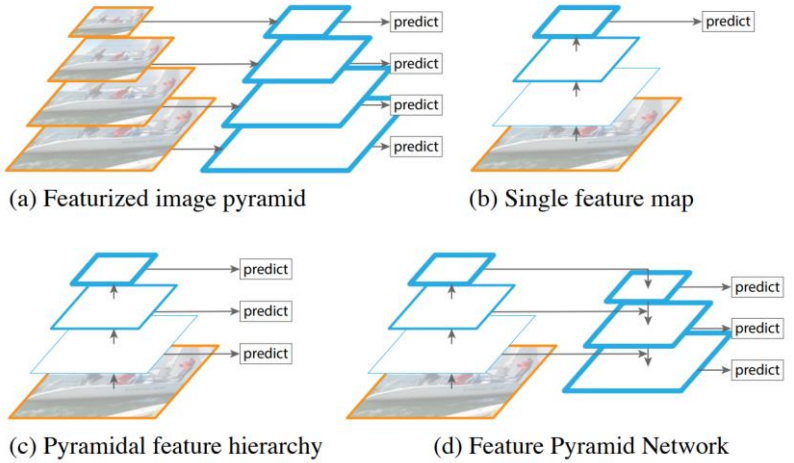


Figure 3. A building block illustrating the lateral connection and the top-down pathway, merged by addition.



(a) Featurized image pyramid (b) Single feature map

(c) Pyramidal feature hierarchy (d) Feature Pyramid Network

Figure 1. (a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Recent detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy computed by a ConvNet as if it were a featurized image pyramid. (d) Our proposed Feature Pyramid Network (FPN) is fast like (b) and (c), but more accurate. In this figure, feature maps are indicate by blue outlines and thicker outlines denote semantically stronger features.

https://arxiv.org/pdf/1612.03144.pdf

Path Aggregation Network (PAN)

Utilize information from all feature levels in the in-network feature hierarchy with **single-scale input**.

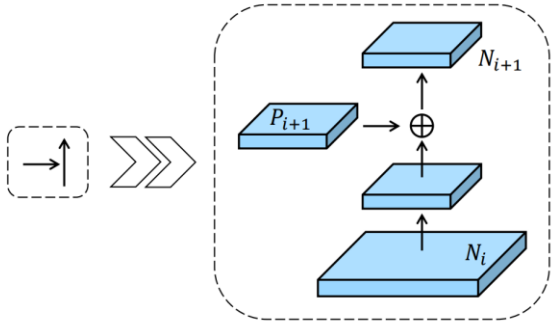


Figure 2. Illustration of our building block of bottom-up path augmentation.

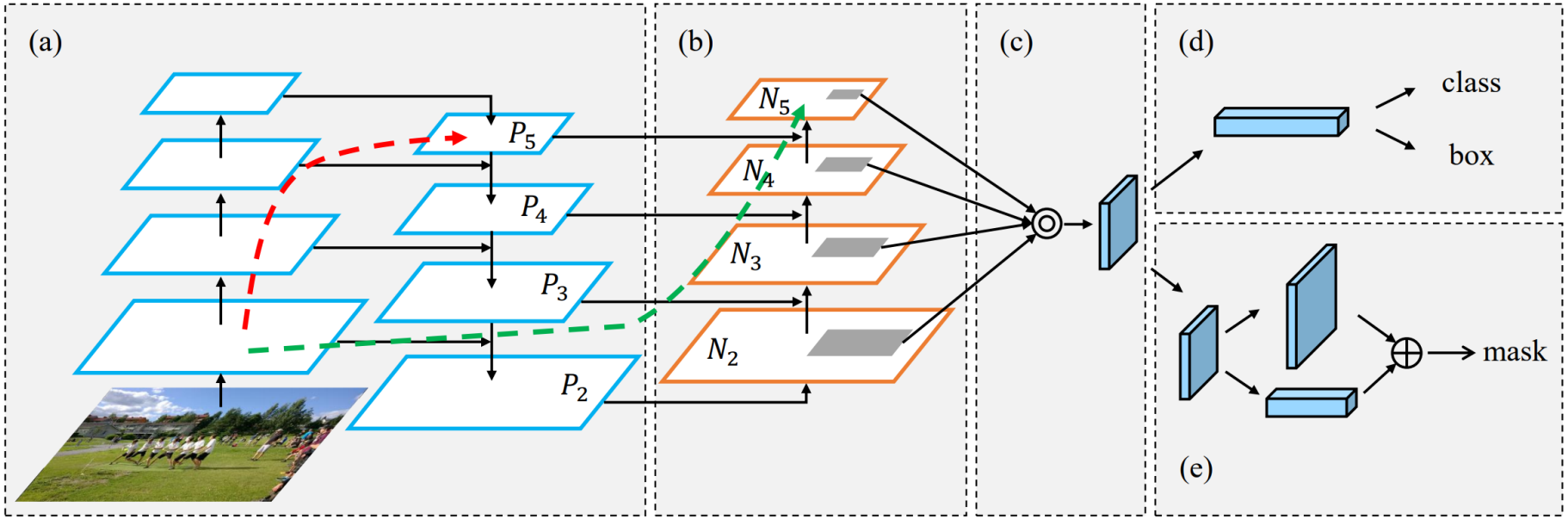


Figure 1. Illustration of our framework. (a) FPN backbone. (b) Bottom-up path augmentation. (c) Adaptive feature pooling. (d) Box branch. (e) Fully-connected fusion. Note that we omit channel dimension of feature maps in (a) and (b) for brevity.

https://arxiv.org/pdf/1803.01534.pdf

Spatial Pyramid Pooling (SPP)

Maintain spatial information by pooling in local spatial bins.

These spatial bins have sizes proportional to the image size, so the number of bins is fixed regardless of the image size.

Different than the sliding window pooling, where the number of sliding windows depends on the input size.

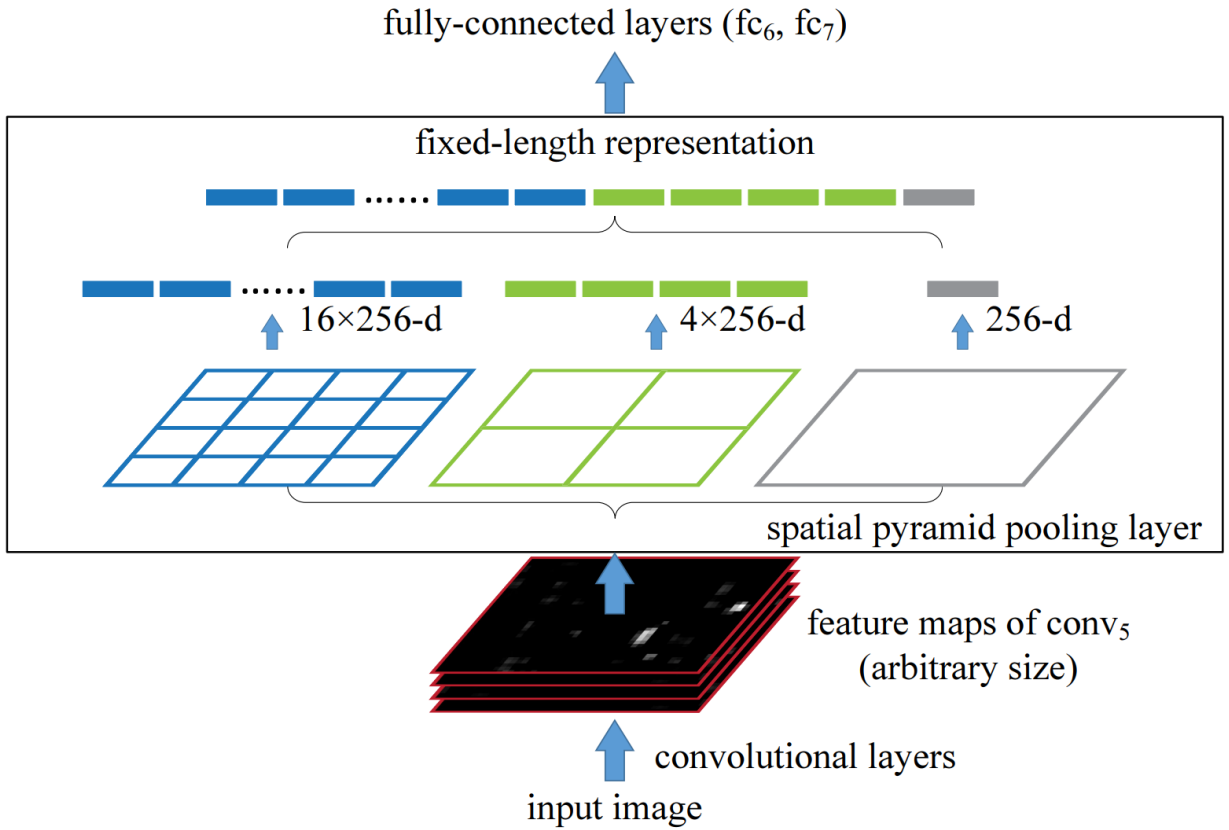


Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

[He et al., 2015] https://arxiv.org/pdf/1406.4729.pdf