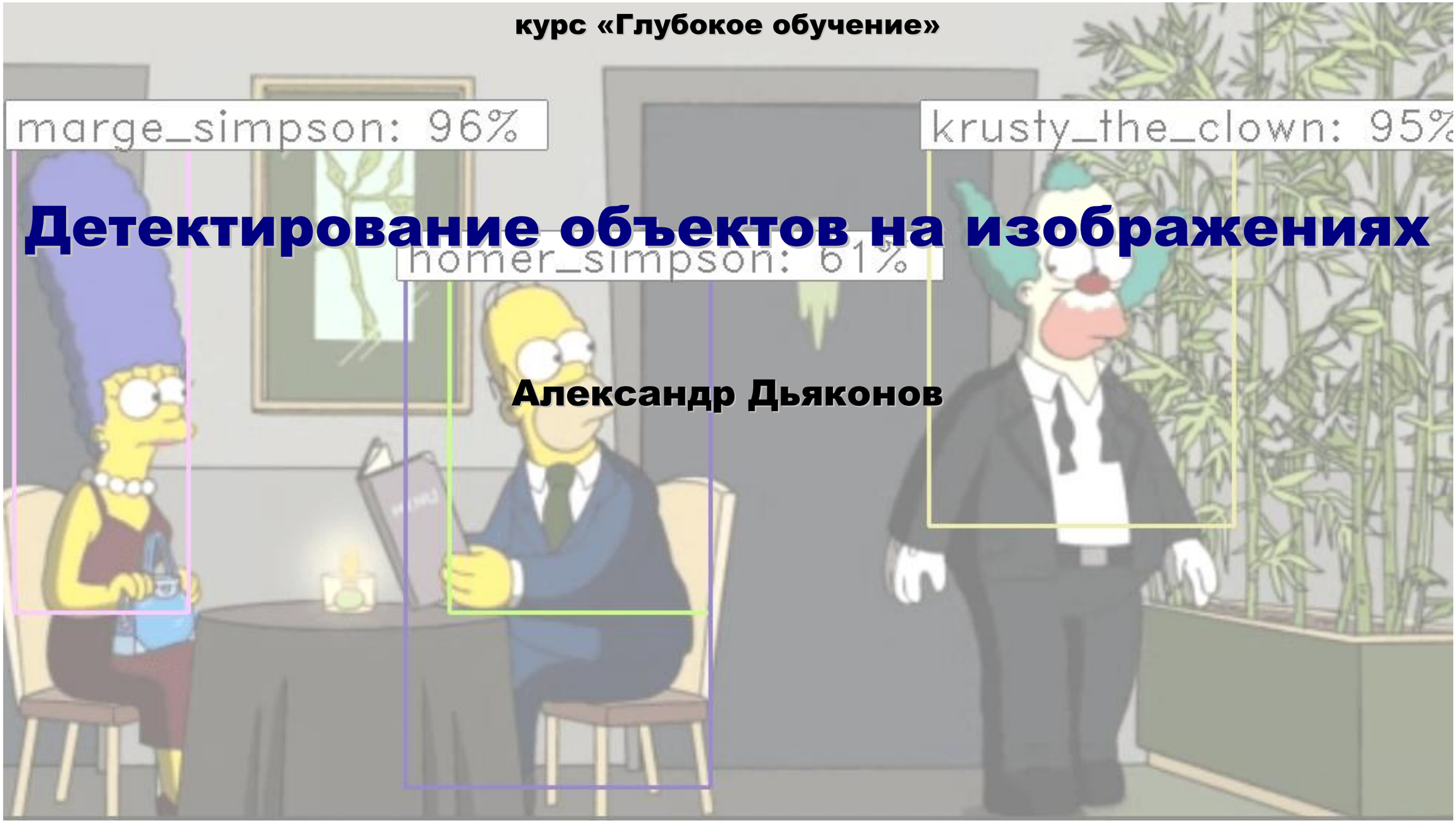
курс «Глубокое обучение»



Детектирование объектов на изображениях

Александр Дьяконов

Напоминание: задачи с изображениями

Классификация + локализация

«cat»

Семантическая сегментация

(Semantic Segmentation)

GRASS, CAT, TREE, SKY

Детектирование объектов

(Object Detection)

DOG, DOG, CAT

Сегментация объектов

(Instance Segmentation)

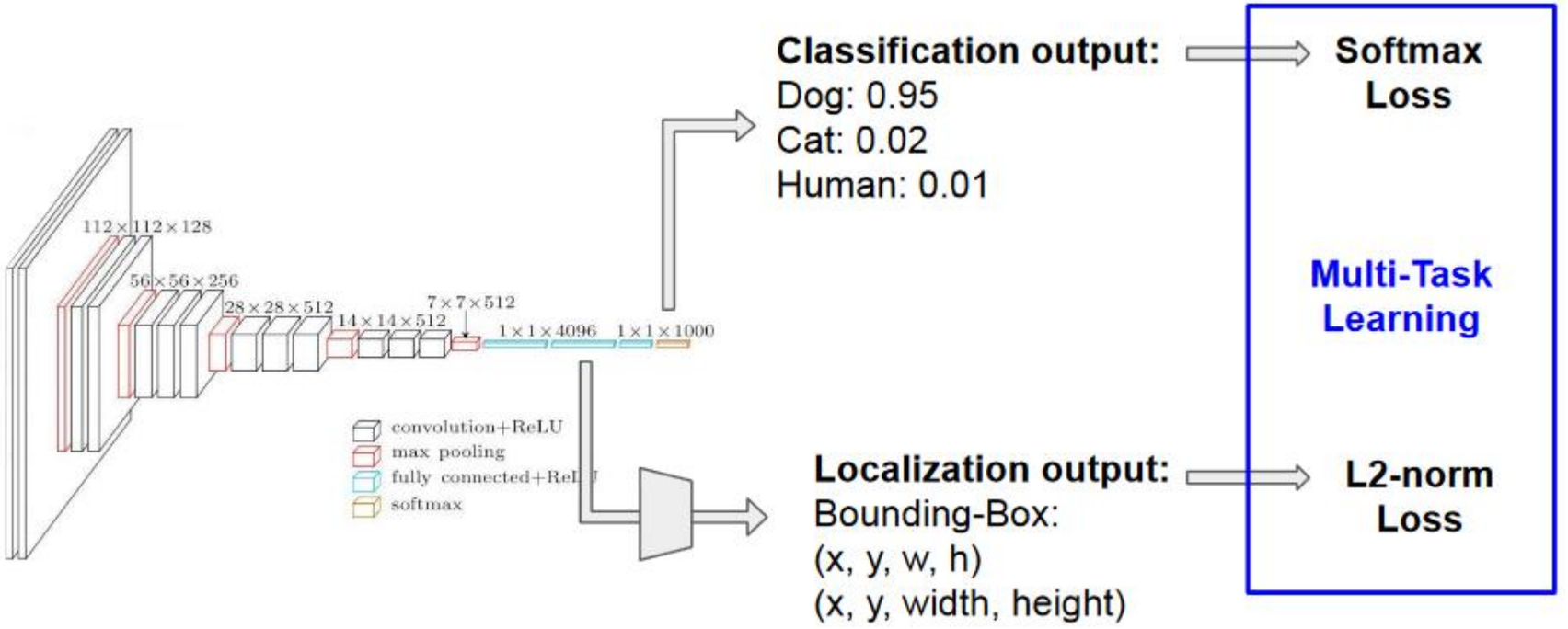
DOG, DOG, CAT

http://cs231n.stanford.edu/2017/syllabus.html

Локализация + Классификация

Классификация – что изображено

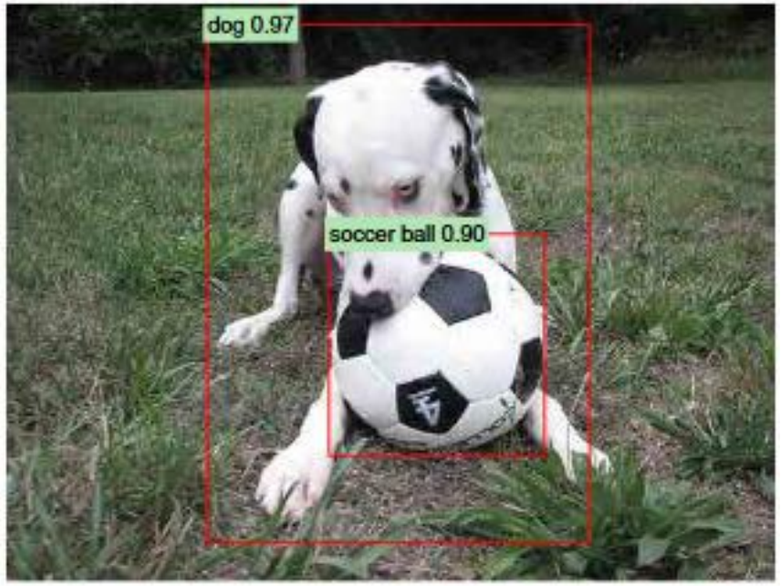
Локализация (localization) объекта – где изображено



идея решения довольно проста

[CS109B - 2021]

Детектирование объектов = (Локализация + Классификация) × число объектов



Локализация (localization) объекта – где

Классификация – что

м.б. ещё определяем параметры объекта

Данные для детектирования объектов

image	X1	X2	Y1	Y2	label
img1.jpg	7	39	3	57	car
img1.jpg	50	157	100	203	dog
Img2.jpg	40	105	207	389	car

выделенное (X1, X2, Y1, Y2, label) – предсказывать

http://cv-tricks.com/object-detection/faster-r-cnn-yolo-ssd/

Детектирование объектов: немного об области

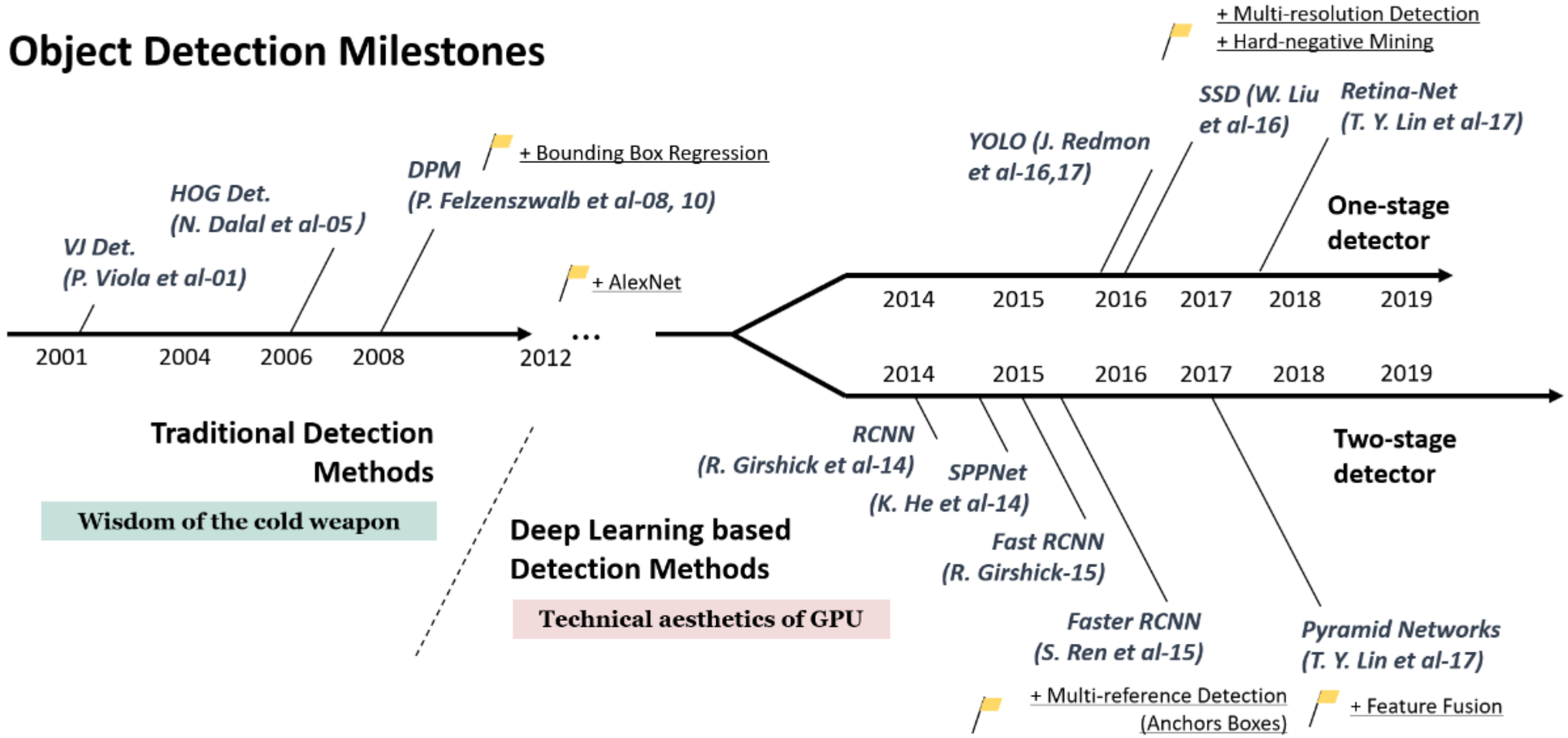


Fig. 2. A road map of object detection. Milestone detectors in this figure: VJ Det. [10, 11], HOG Det. [12], DPM [13–15], RCNN [16], SPPNet [17], Fast RCNN [18], Faster RCNN [19], YOLO [20], SSD [21], Pyramid Networks [22], Retina-Net [23].

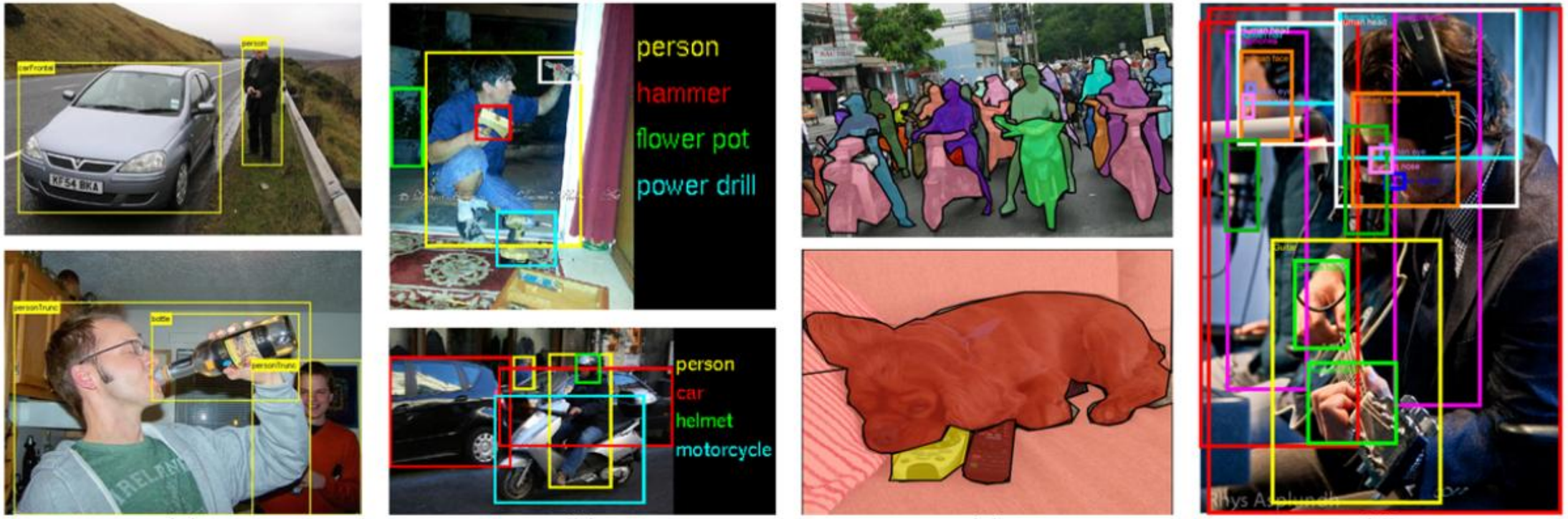
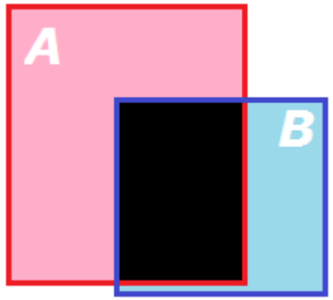


Fig. 4. Some example images and annotations in (a) PASCAL-VOC07, (b) ILSVRC, (c) MS-COCO, and (d) Open Images.

Dataset	train		validation		trainval		test	
	images	objects	images	objects	images	objects	images	objects
VOC-2007	2,501	6,301	2,510	6,307	5,011	12,608	4,952	14,976
VOC-2012	5,717	13,609	5,823	13,841	11,540	27,450	10,991	-
ILSVRC-2014	456,567	478,807	20,121	55,502	476,688	534,309	40,152	-
ILSVRC-2017	456,567	478,807	20,121	55,502	476,688	534,309	65,500	-
MS-COCO-2015	82,783	604,907	40,504	291,875	123,287	896,782	81,434	-
MS-COCO-2018	118,287	860,001	5,000	36,781	123,287	896,782	40,670	-
OID-2018	1,743,042	14,610,229	41,620	204,621	1,784,662	14,814,850	125,436	625,282

Метрики качества

Правильное детектирование, если



$$\mathrm{IoU} = \frac{|A \cap B|}{|A \cup B|} \geq \alpha$$

Intersection Over Union

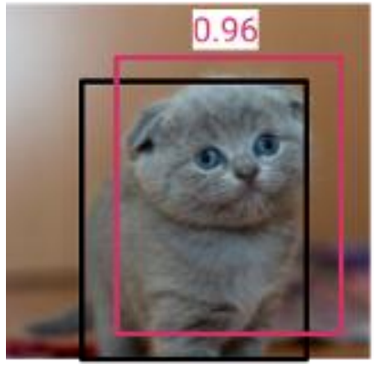
TP, FN, FP

Precision / Recall

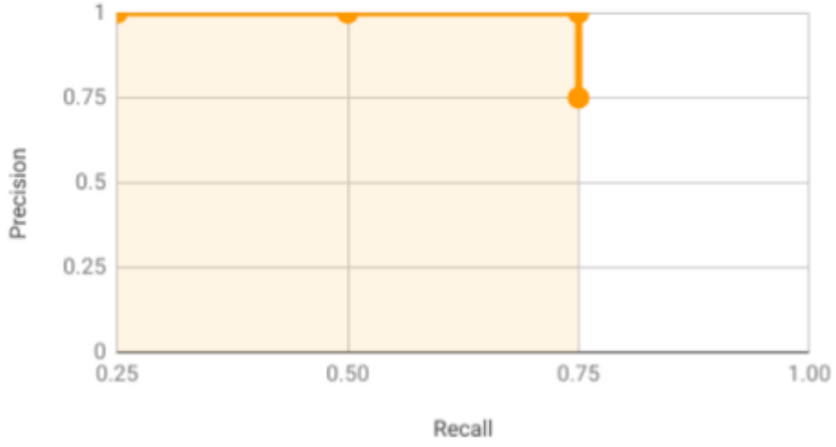
Average Precision (AP)

~ по задаче классификации региона

решение «Positive», если правильный класс и IoU > 0.5, иначе «Negative»



Confidence	Cat Number	isCorrect?	Precision	Recall
0.96	Cat 1	1	1	0.25
0.88	Cat 3	1	1	0.5
0.82	Cat 2	1	1	0.75
0.64	Cat 4	0	0.75	0.75



Упорядочиваем по confidence (оценке класса «кот»)

Площадь под кривой.

Mean Average Precision (mAP) – **усреднение AP по всем классам**

Panoptic Quality

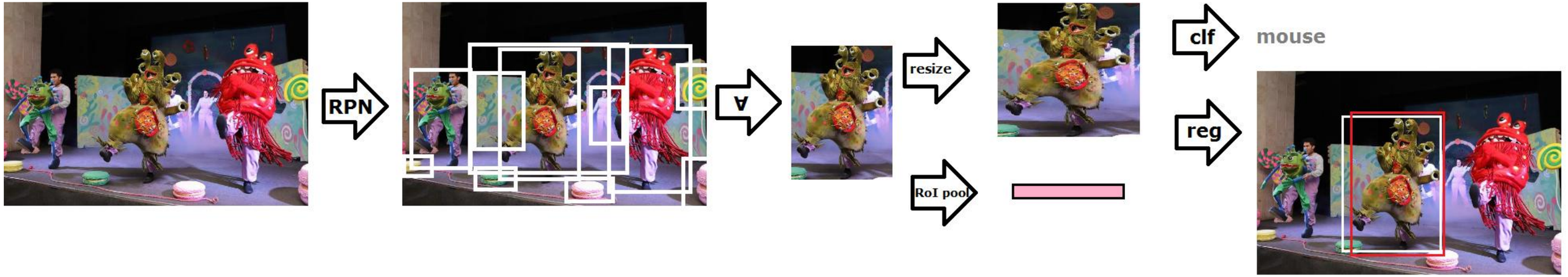
$$PQ = \frac{\sum\limits_{(x,z)\in \mathrm{TP}} \mathrm{IoU}(x,z)}{|\,\mathrm{TP}\,| + \frac{1}{2}|\,\mathrm{FP}\,| + \frac{1}{2}|\,\mathrm{FN}\,|} = \underbrace{\frac{\sum\limits_{(x,z)\in \mathrm{TP}} \mathrm{IoU}(x,z)}{|\,\mathrm{TP}\,|}}_{\mathrm{SQ}} \underbrace{\frac{|\,\mathrm{TP}\,|}{|\,\mathrm{TP}\,| + \frac{1}{2}|\,\mathrm{FP}\,| + \frac{1}{2}|\,\mathrm{FN}\,|}}_{\mathrm{RQ}}$$

SQ – **segmentation quality**

RQ – **recognition quality**

https://medium.com/@danielmechea/panoptic-segmentation-the-panoptic-quality-metric-d69a6c3ace30

Двухстадийные детекторы – общее описание



1. Сгенерировать регионы-кандидаты (эволюция: SS → RPN)

2. Каждый регион подготовить

- под формат CNN

- (или) в признаковое описание (более быстрое решение)

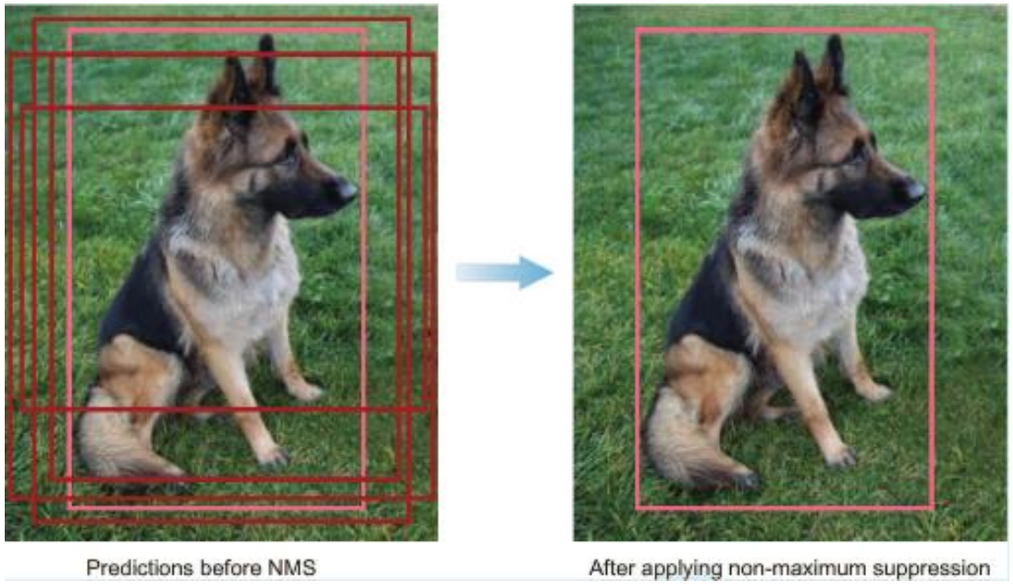
3. Провести классификацию – «какой объект»

4. Провести 4D-регрессию – «уточнить где»

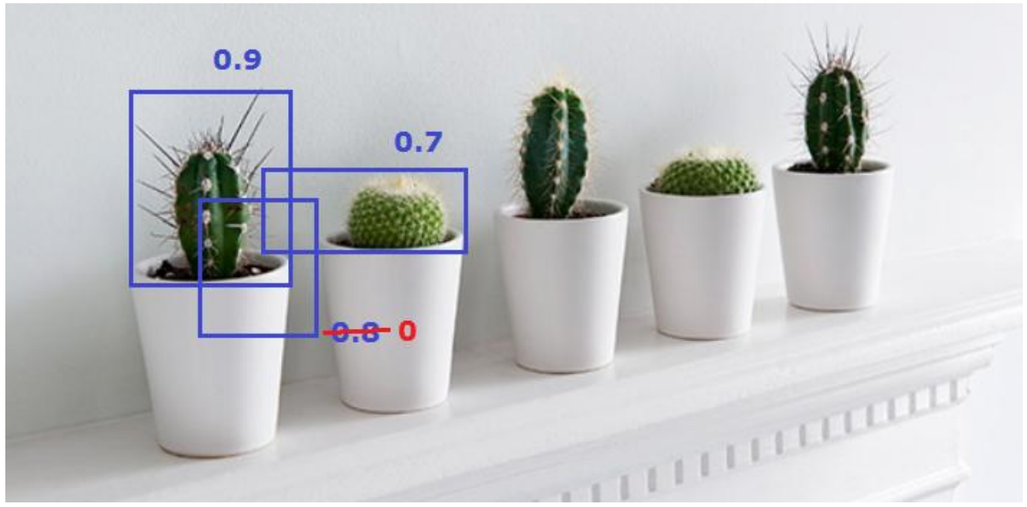
в результате много уточнённых рамок с классами

+ ещё устранить объекты-дубликаты (NMS)

NMS = Non Maximum Suppression



[Mohamed Elgendy]



Из семейства похожих рамок надо оставить одну для каждого объекта

Класс «Кактус»

Упорядочим регионы: 0.9, 0.8, 0.7, …

(ниже порога сразу занулить)

Идём от MAX (i):

Идём от текущего (j):

регион j имеет большое пересечение с i ⇒ зануляем

Минутка кода: NMS

`torchvision.ops.nms(`*`boxes: torch.Tensor, scores: torch.Tensor, iou_threshold: float`*`)` → torch.Tensor [SOURCE]

Performs non-maximum suppression (NMS) on the boxes according to their intersection-over-union (IoU).

NMS iteratively removes lower scoring boxes which have an IoU greater than iou_threshold with another (higher scoring) box.

If multiple boxes have the exact same score and satisfy the IoU criterion with respect to a reference box, the selected box is not guaranteed to be the same between CPU and GPU. This is similar to the behavior of argsort in PyTorch when repeated values are present.

Parameters

- **boxes** (*Tensor*[*N*, 4]) – boxes to perform NMS on. They are expected to be in `(x1, y1, x2, y2)` format with `0 <= x1 < x2` and `0 <= y1 < y2`.
- **scores** (*Tensor*[*N*]) – scores for each one of the boxes
- **iou_threshold** (*float*) – discards all overlapping boxes with IoU > iou_threshold

Returns

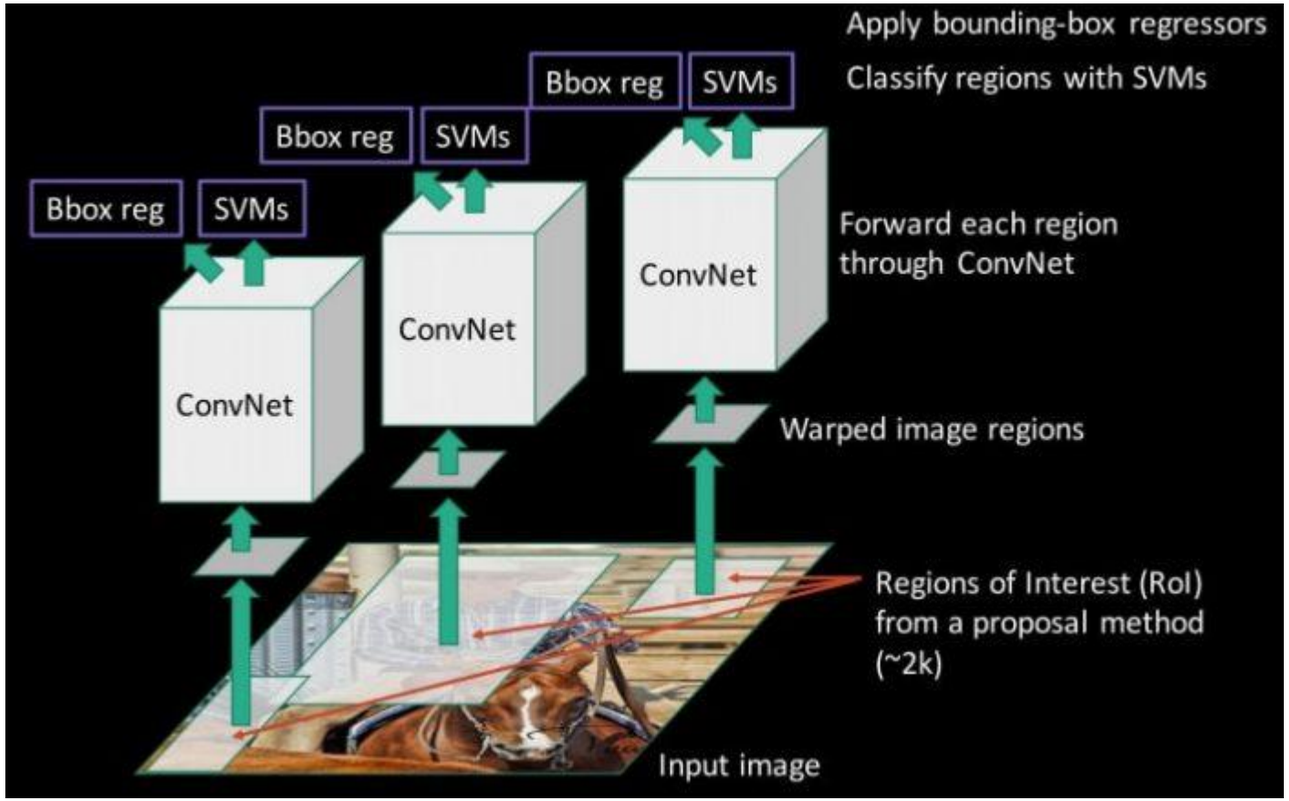
int64 tensor with the indices of the elements that have been kept by NMS, sorted in decreasing order of scores

Return type

Tensor

https://pytorch.org/vision/stable/ops.html

R-CNN (=Regions with CNN features)



- **Selective Search для генерации регионов (гипотез, где объекты)**

  Есть много методов просто выбран этот 2000 раз
- **Подгонка региона (resize) для подачи в CNN: 277×277**

  Сеть имеет вход фиксированного размера – натренирована (ILSVRC2012 classification)
- **Классификация регионов (CNN + SVM)**

  Krizhevsky Caffe выдаёт 4096 признаков + линейный SVM

  + дотренировка на данных
- **Оптимизация регионов с помощью регрессии**

  Bounding-box regression

R-CNN: дотренировка

Если 30% (важный коэффициент) пересечения с истинным объектом «машина», то это класс машина

Классов = число объектов детектирования + 1 (фон) – теперь столько нейронов в последнем слое

batch = 32 окна с объектом + 96 с фоном

выбирается из 2000 регионов, сгенерированных по изображению

это решение проблемы дисбаланса (фона больше), ещё используют (не здесь) focal loss

R-CNN: регрессия

Для обучения только регионы $(\bar{x}, \bar{y}, \bar{h}, \bar{w})$

с 30% пересечением с истинным (x, y, h, w)

целевые значения – $\left(\frac{x - \bar{x}}{w}, \frac{y - \bar{y}}{h}, \ln(\bar{w} / w), \ln(\bar{h} / h) \right)$

линейная регрессия на CNN-признаках, MSE

ясно, как пересчитать в координаты регионов ответы

SPP-net: Spatial Pyramid Pooling

Разделение свёрточных слоёв для всех регионов одного изображения

Полносвязной НС нужен вход определённых размеров.
Чисто свёрточная НС может работать со входом любых размеров!

Идея – добавим к свёрточной сети слой: может генерировать фиксированный размер

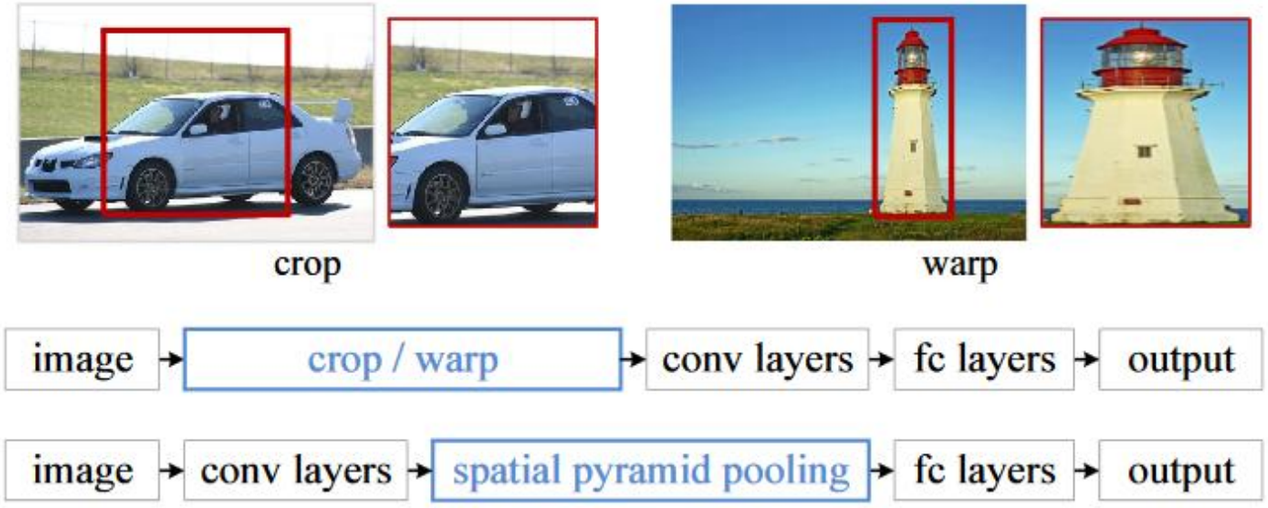


Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

SPP-net: The Spatial Pyramid Pooling Layer

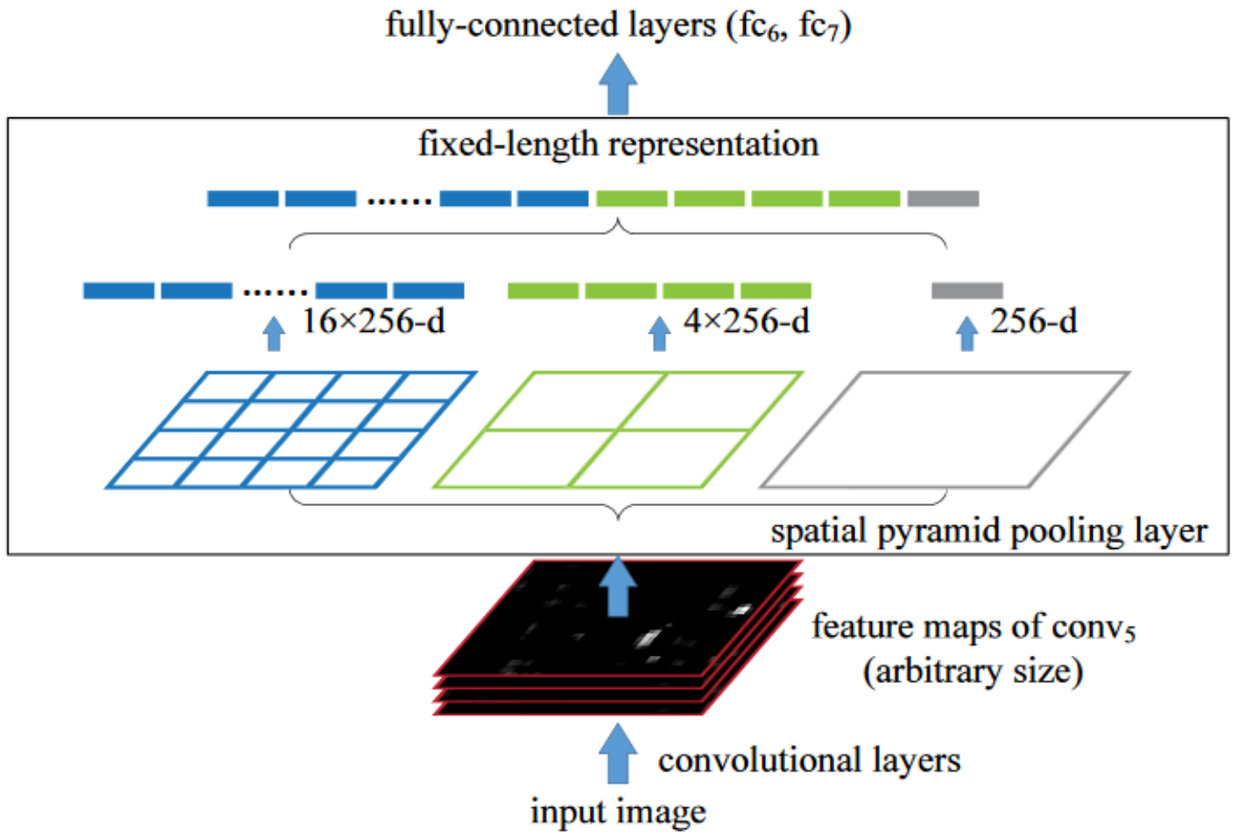
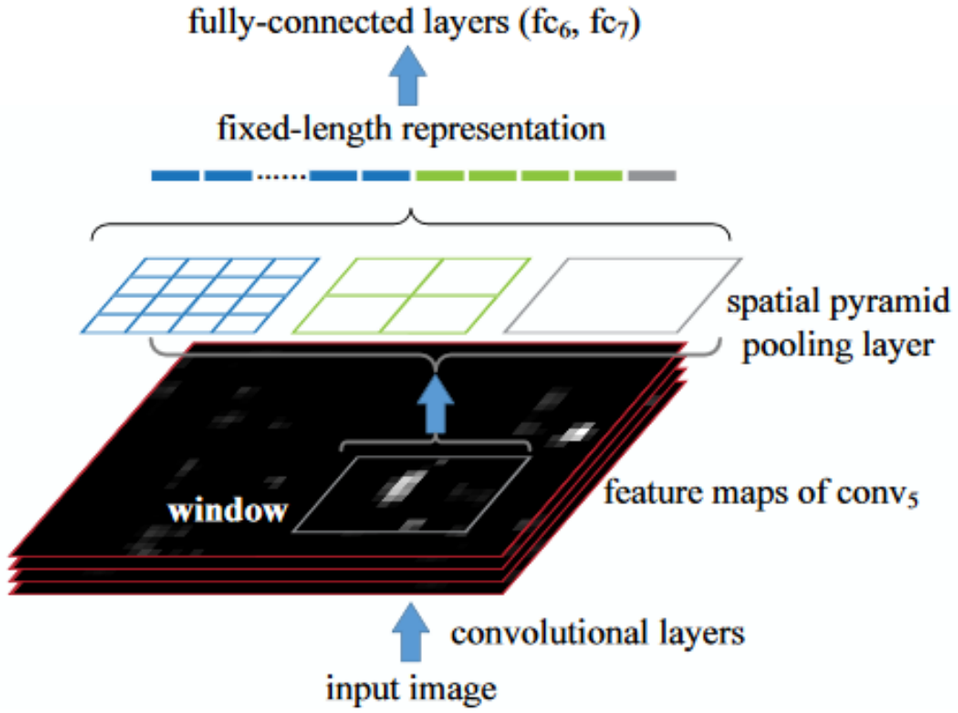


Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the $conv_5$ layer, and $conv_5$ is the last convolutional layer.

He и др. «Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition» https://arxiv.org/pdf/1406.4729.pdf]

The Spatial Pyramid Pooling Layer



Считаем CNN-представление для изображений один раз

С помощью SPP-net для каждого региона получаем признаковое представление (фиксированной длины)

Выигрыш по скорости 10x – 100x!

Быстрое решение проблемы того, что регионы разных размеров!
+ сначала-то мы считаем CNN-представление!
а SPP-net просто эффектный способ перевода его в признаки
Сеть не работает на каждом регионе! Сразу на изображении!

Минутка кода

```
import math
def spatial_pyramid_pool(self,previous_conv, num_sample, previous_conv_size, out_pool_size):
    '''
    previous_conv: a tensor vector of previous convolution layer
    num_sample: an int number of image in the batch
    previous_conv_size: an int vector [height, width] of the matrix features size of previous convolution layer
    out_pool_size: a int vector of expected output size of max pooling layer
    returns: a tensor vector with shape [1 x n] is the concentration of multi-level pooling
    '''
     for i in range(len(out_pool_size)):
         h_wid = int(math.ceil(previous_conv_size[0] / out_pool_size[i]))
         w_wid = int(math.ceil(previous_conv_size[1] / out_pool_size[i]))
         h_pad = (h_wid*out_pool_size[i] - previous_conv_size[0] + 1)/2
         w_pad = (w_wid*out_pool_size[i] - previous_conv_size[1] + 1)/2
         maxpool = nn.MaxPool2d((h_wid, w_wid), stride=(h_wid, w_wid), padding=(h_pad, w_pad))
         x = maxpool(previous_conv)
         if(i == 0):
             spp = x.view(num_sample,-1)
         else:
             spp = torch.cat((spp, x.view(num_sample,-1)), 1)
     return spp
```

https://github.com/yueruchen/sppnet-pytorch/blob/master/spp_layer.py

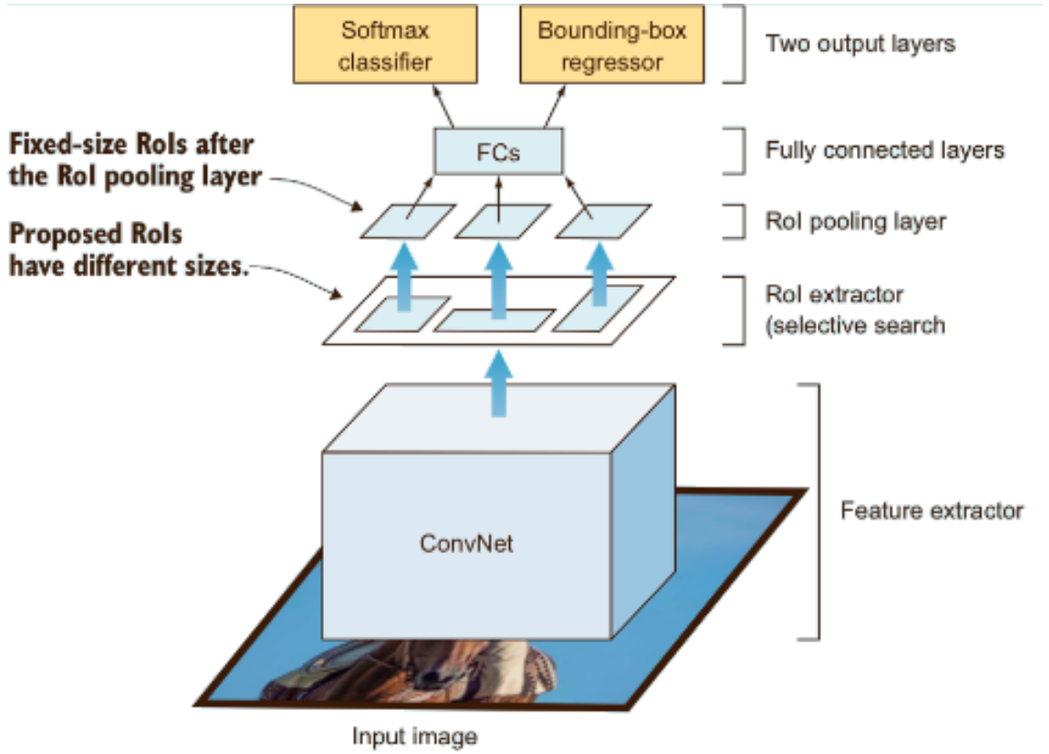
есть адаптивный пулинг – выход конкретно размера!

Замечание: есть ещё Temporal Pyramid Pooling (т.е. 1D)

Fast R-CNN

Fast R-CNN = R-CNN + SPP (только по-другому: RoI-pool) + регрессию встроили в НС

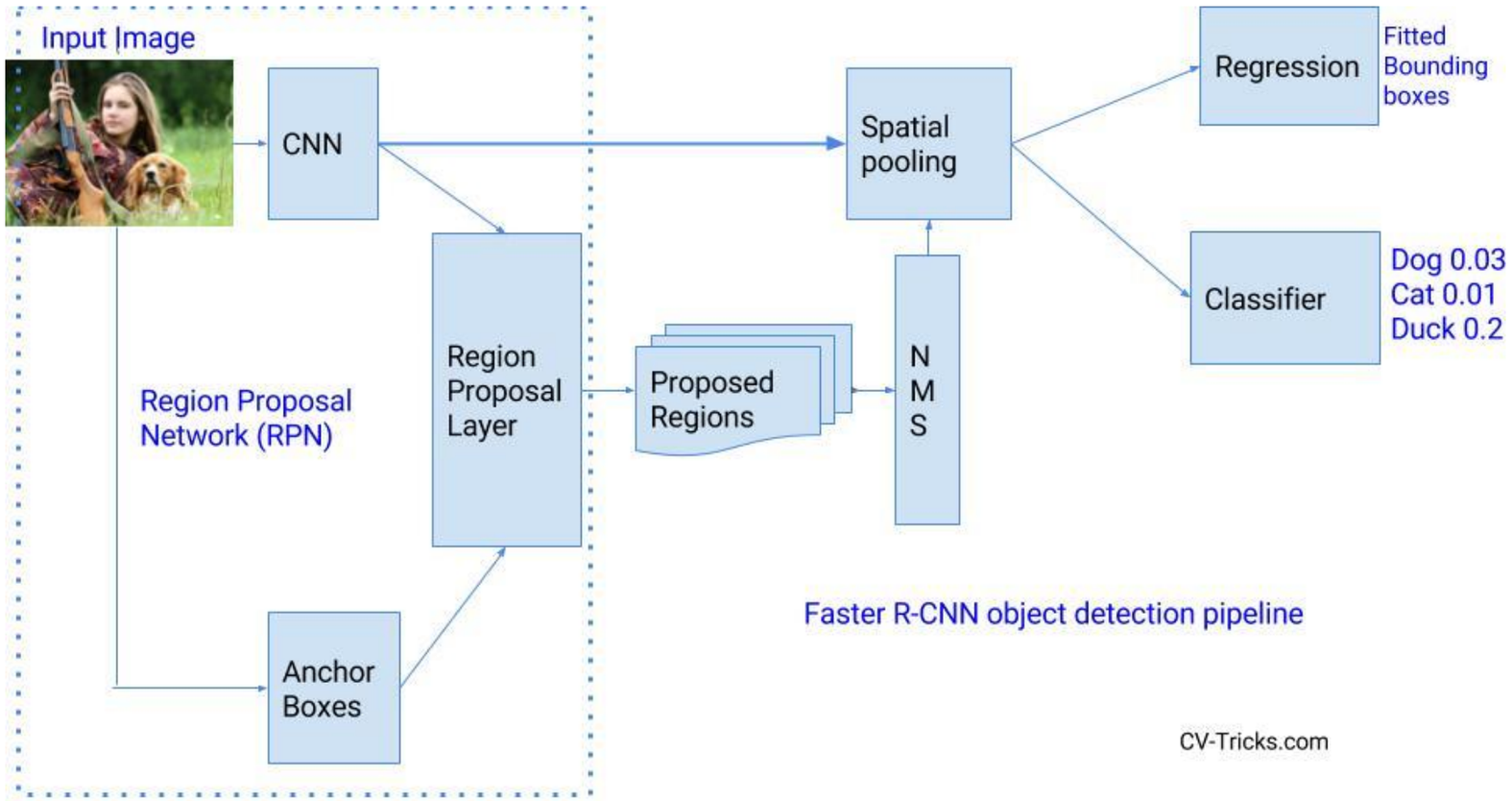
Обучение в одну стадию (раньше CNN → SVM → regression)!



Вход: изображение + параметры регионов (регионы тоже с помощью SS)

end2end: ошибка = сумма ошибок классификатора и регрессора

Faster R-CNN: SS заменяем на Region Proposal Network (RPN)

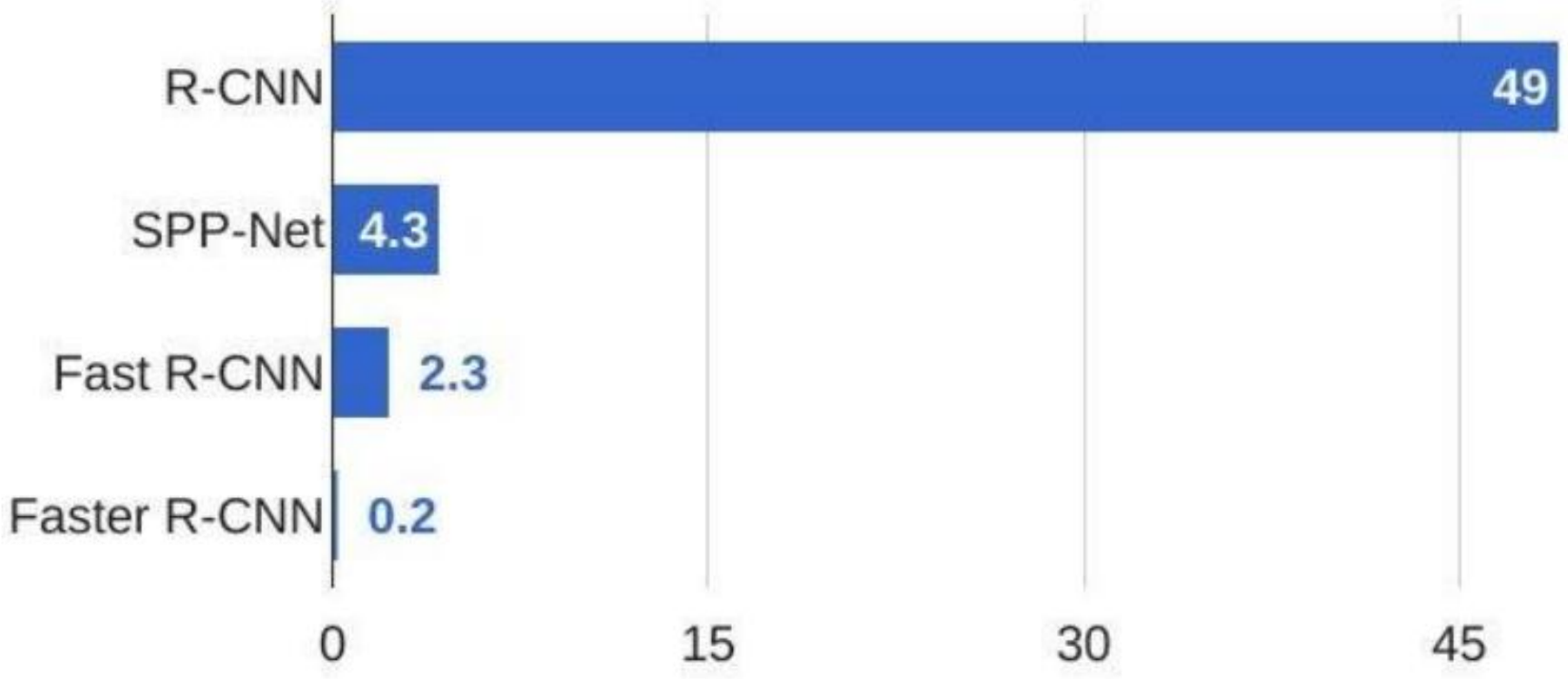


Полностью end2end: «где смотреть» решает НС

DL-генерация регионов

картинка → свёрточные слои, потом Region Proposal Network (RPN)

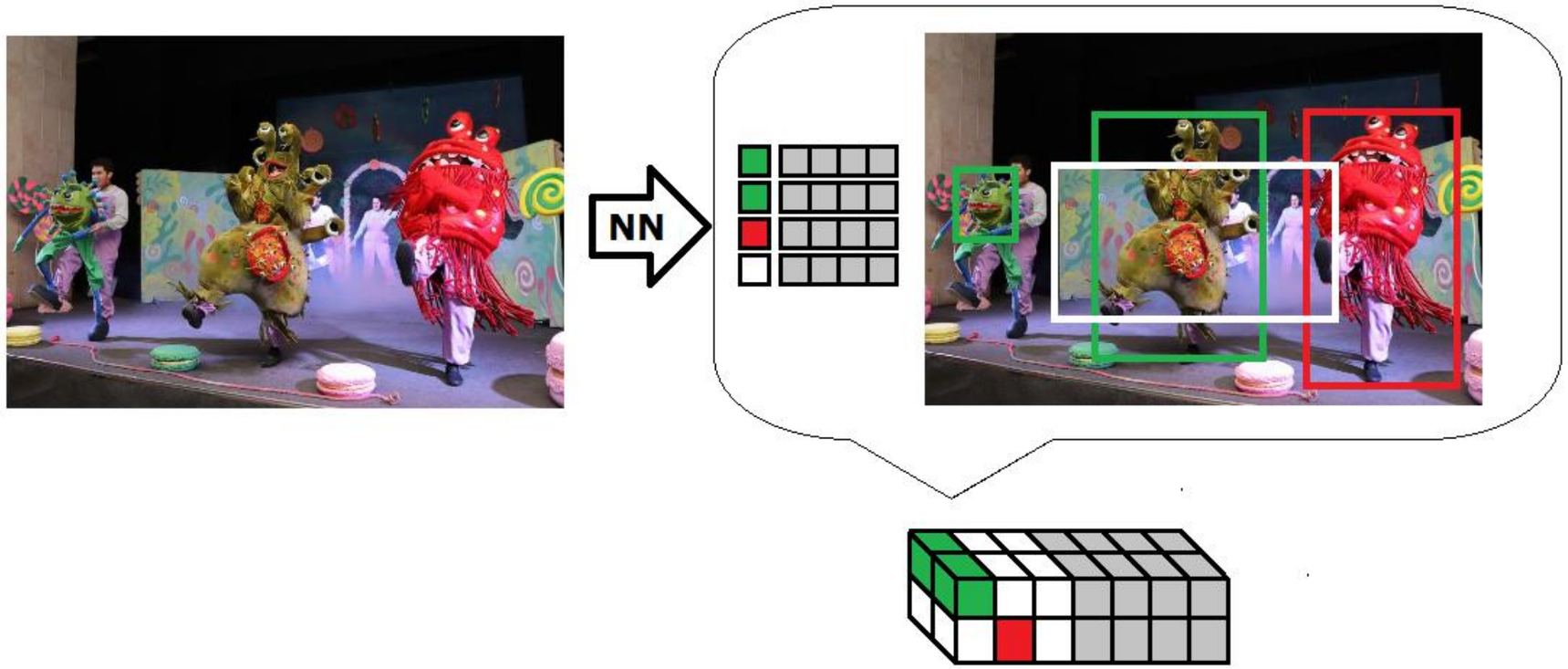
Скорость *-R-CNN сетей



указано сек / на 1 изображение

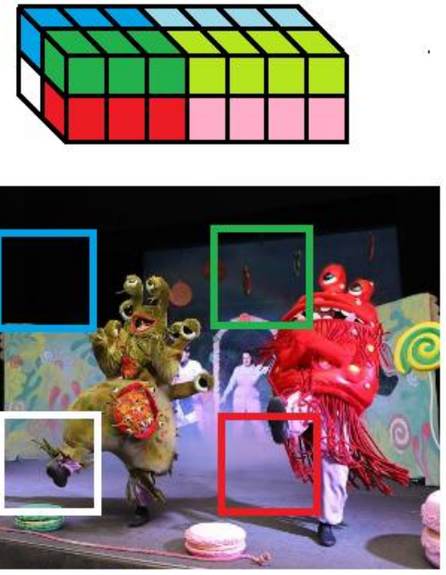
Одностадийные детекторы – общее описание

Получаем перечень регионов (фиксированное количество) в виде тензора



всё делается «в одну стадию» – одной нейронкой

Привязка элементов тензора к якорям

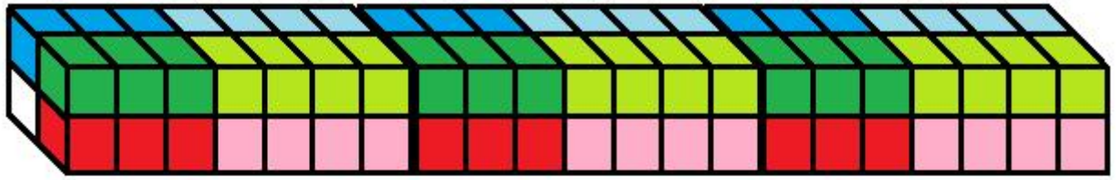


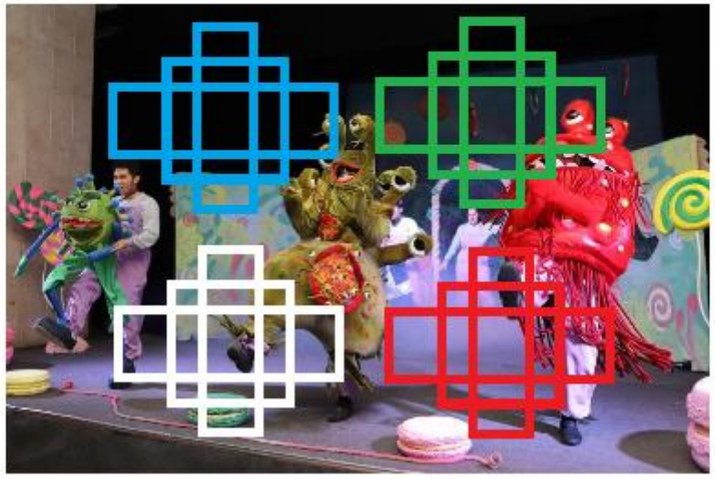
расположение якорей соответствует индексам [h, w] строк тензора

Одностадийные детекторы – общее описание

якорей м.б. много

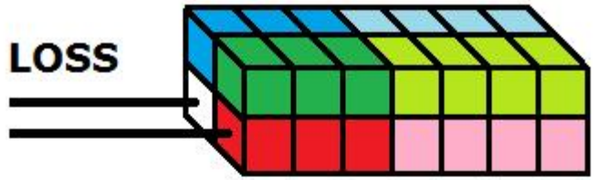
(можно и без явных якорей делать)

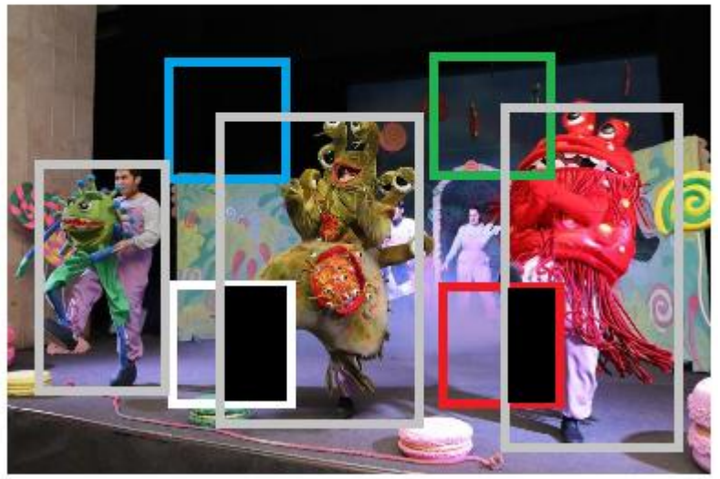




больше якорей – больше объектов сможем детектировать

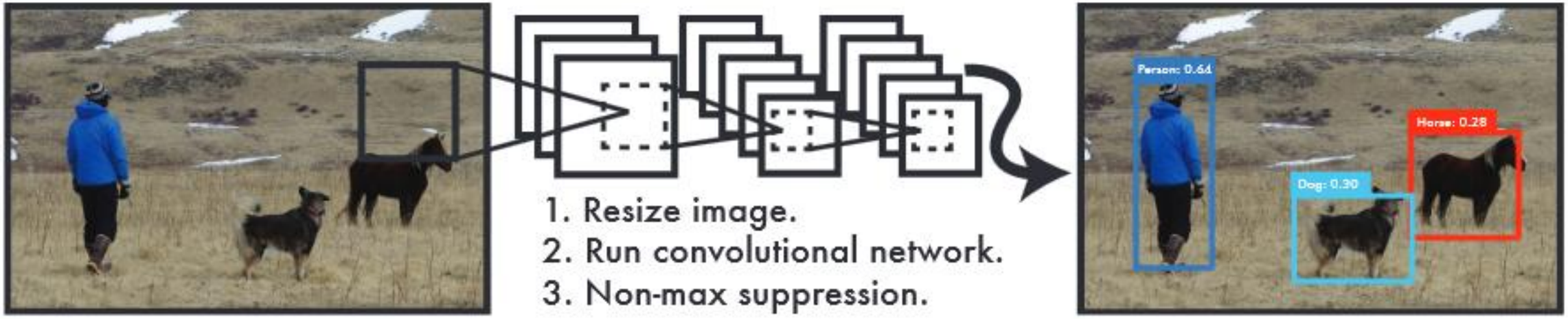
как делается привязка к правильным рамкам





где большие пересечения

YOLO: You only ~~Live~~ Look Once (2016)



- **Изменение масштаба → 448×448**
- **CNN (одна!)**
- **Детектирование – задача регрессии;)**
- **Пороговое принятие решения**
- **Запуск сразу на всём изображении – очень быстро**

Сеть видит изображение целиком, а не регионами

Очень быстрая, но точность хуже (особенно для мелких объектов)

https://pjreddie.com/media/files/papers/yolo_1.pdf

YOLO: изображение → тензор (с вероятностями классов и координатами рамок)

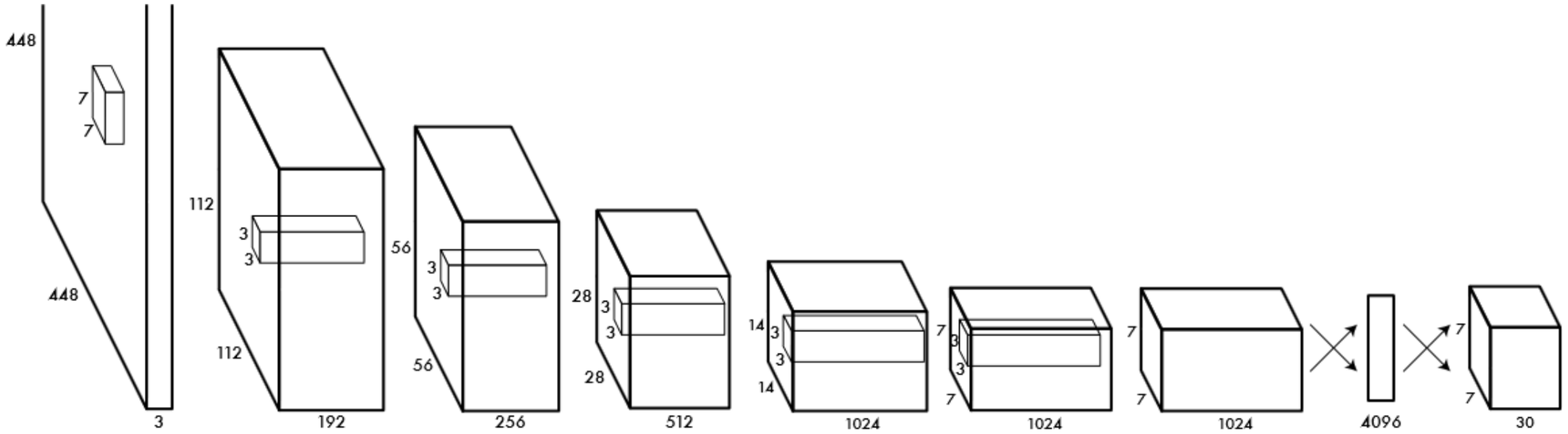
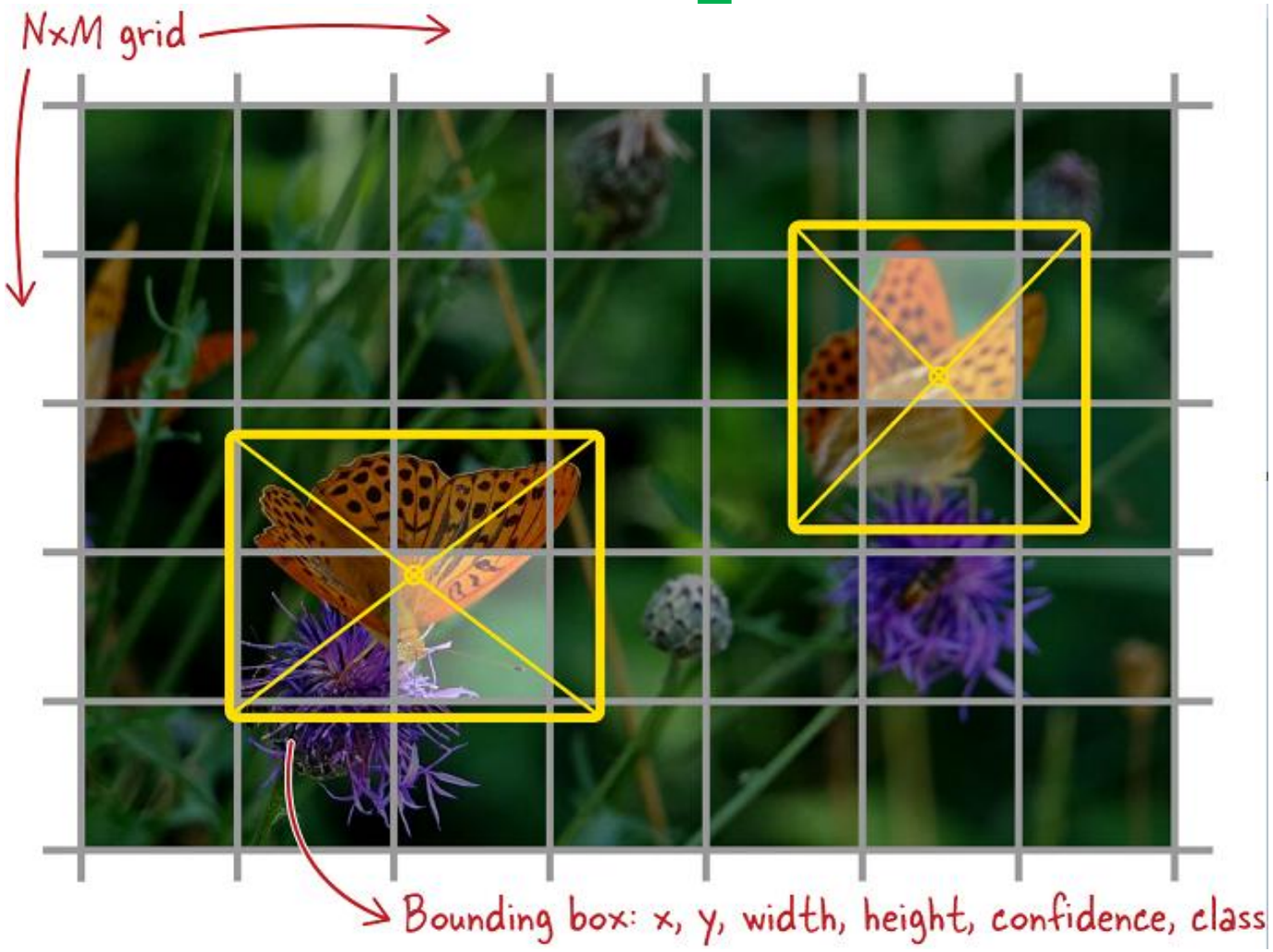


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

Классика: свёртки + полносвязные слои

YOLO



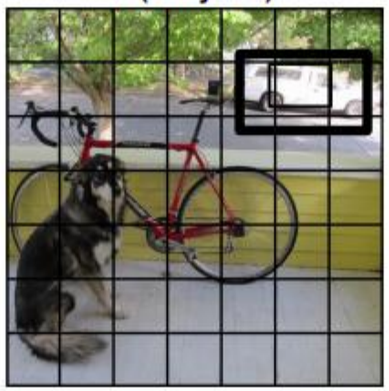
предсказание региона (соотв. ячейке) – предсказание 6 чисел

[Practical Machine Learning for Computer Vision]

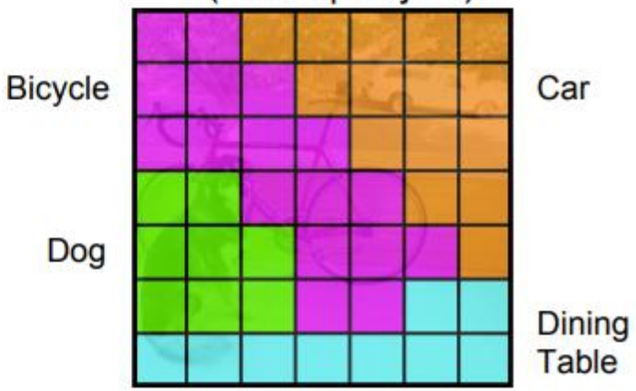
YOLO

Split the image into a grid

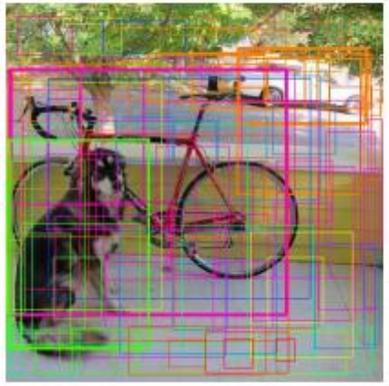
P(Object)



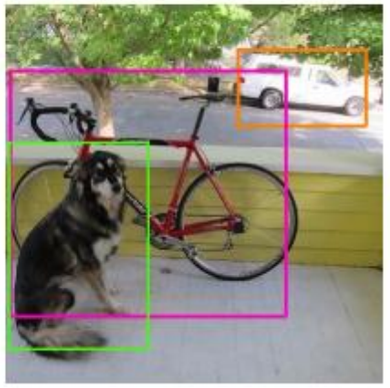
P(Class | Object)



Combine the box and class predictions

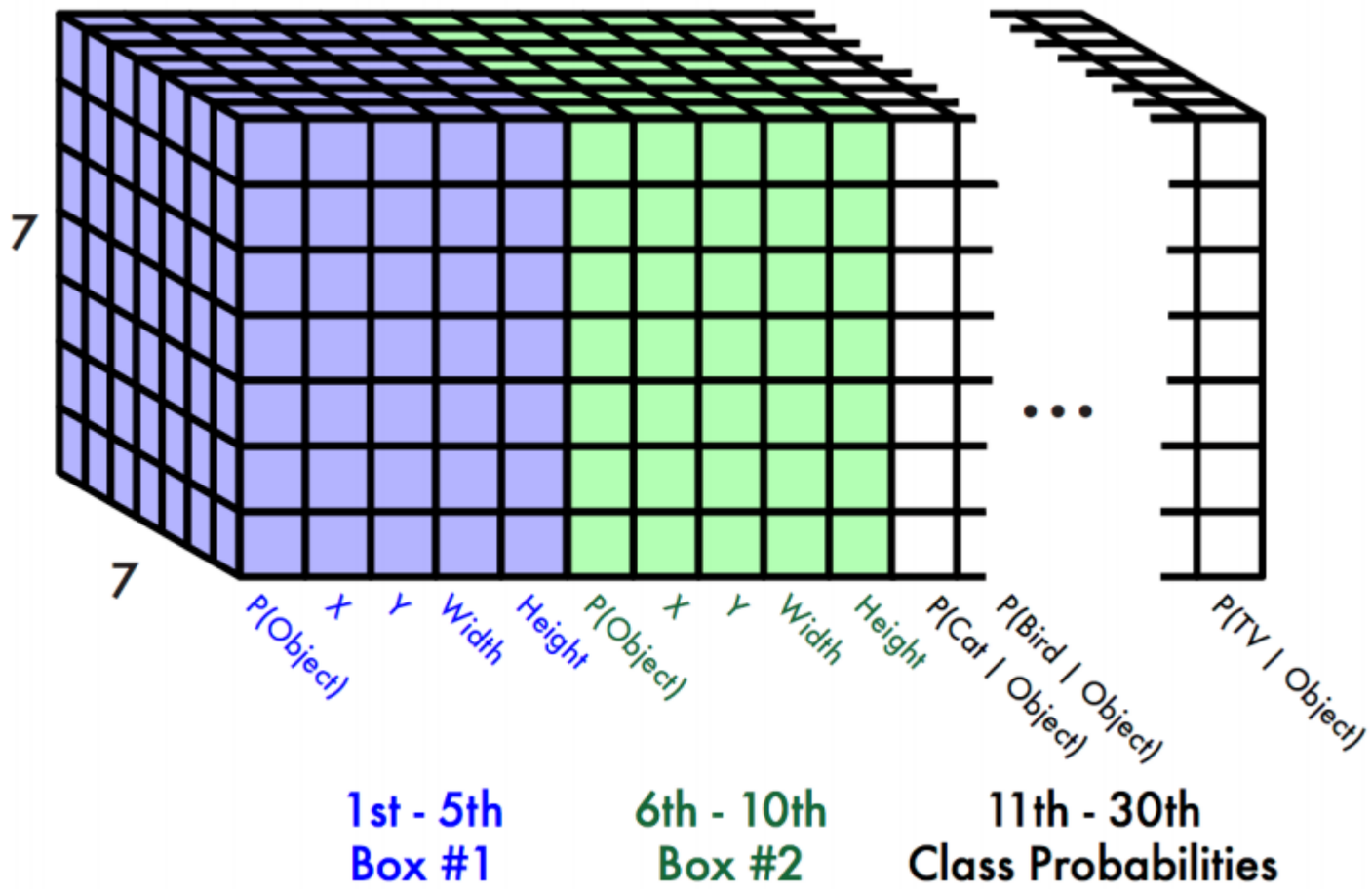


Finally do NMS and threshold detections



каждая клетка – свой класс, для каждого бокса своя вероятность, что там есть объект

YOLO: выход модели



В тензоре 7×7×30 закодированы все регионы оценки за классы

7×7 – это сетка;)

30 = 5 + 5 +

(почему-то 2 региона для ∀ ячейки)

+ 20 (# классов ~ Pascal VOC)

5 = |(x, y, w, h, c)|

x, y – координаты в центре соотв. ячейки

c – уверенность, что регион правильный

YOLO: ошибка регрессионная

$$\lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right]$$

$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2$$

$$+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2$$

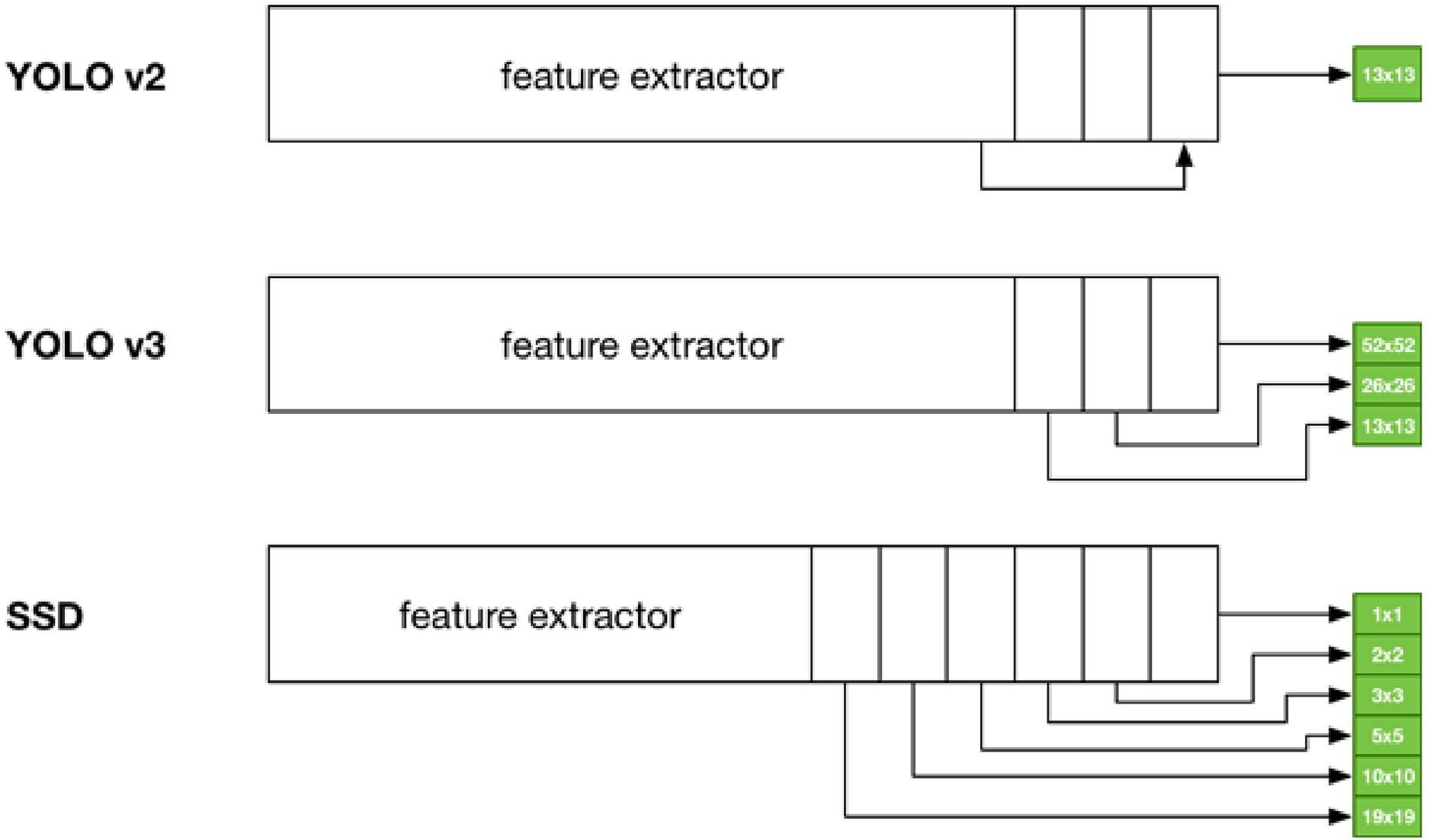
$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2$$

YOLO

- Предположение: в каждой ячейке центры максимум 2х классов
- Первые 20 свёрхточных слоёв предтренированы на ImageNet
- 448 × 448 вместо 224 × 224,
- Leaky ReLU (везде)
- 1 × 1 bottleneck filters
- dropout после 1го полносвязного слоя
- квадратичная ошибка (для координат, уверенности и оценок!)
- борьба с очень большими регионами и пустыми регионами (...)
- momentum 0.9, decay $5 \cdot 10^{-4}$
- аугментация (scaling, translation, HSV transformation)
- быстрая 45 fps, YOLO-tiny – 155 fps, хотя качество и хуже SOTA
- проблема с детекцией маленьких объектов и толпы (ограничение 2 объекта в ячейке)

J. Redmon, S. Divvala, R. Girshick, A. Farhadi «You Only Look Once: Unified, Real-Time Object Detection» https://arxiv.org/abs/1506.02640

Эволюция YOLO + SSD



https://machinethink.net/blog/object-detection/

YOLOv2 + YOLO9000 (2016)

YOLO9000 модификация на базе архитектуры YOLOv2 (была SoTA)

усложнили, предобучили на ImageNet-е на разрешении 448×448 + COCO

убрали полносвязные слои, теперь свёртки + якоря (идея из RPN)

якоря (их формы и размеры) **с помощью k-means**
(для определения форм специфичных для данного датасета)

детектирование ~9000 объектов (классов)
(качество не очень, но быстрая)

определение более 1000 регионов у YOLO Было 98

иерархия классов

BN-слои

Redmon and Farhadi «YOLO9000: Better, Faster, Stronger» // https://arxiv.org/pdf/1612.08242.pdf

YOLOv2: кластеризация якорей

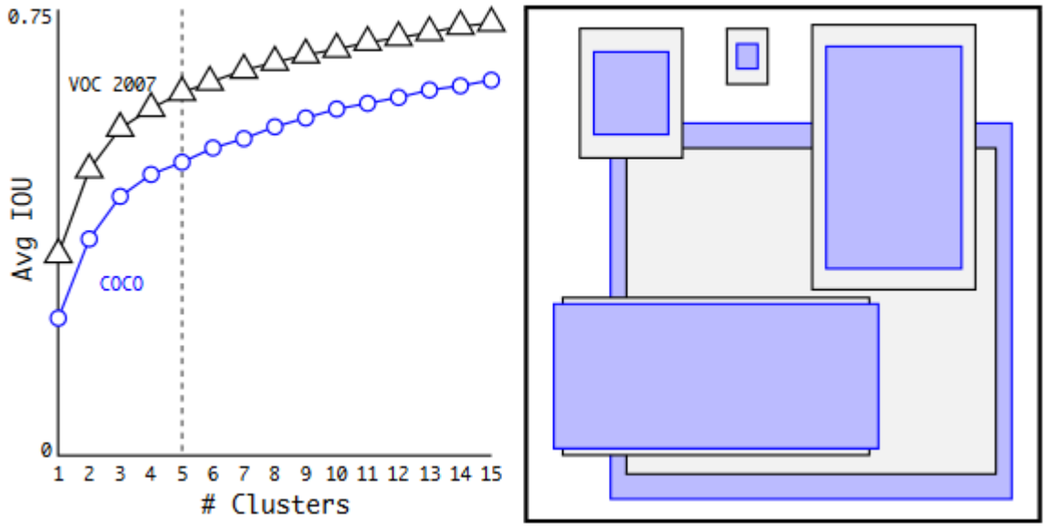
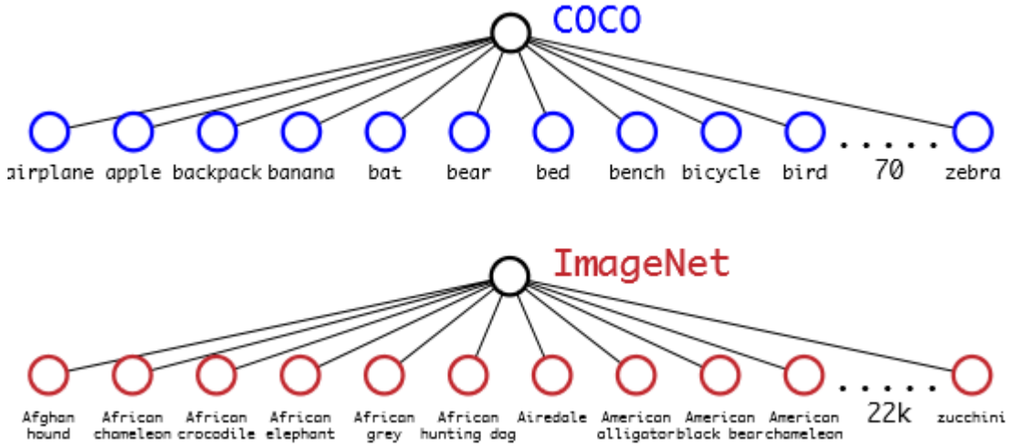


Figure 2: Clustering box dimensions on VOC and COCO. We run k-means clustering on the dimensions of bounding boxes to get good priors for our model. The left image shows the average IOU we get with various choices for k. We find that $k = 5$ gives a good tradeoff for recall vs. complexity of the model. The right image shows the relative centroids for VOC and COCO. Both sets of priors favor thinner, taller boxes while COCO has greater variation in size than VOC.

YOLOv2: иерархия классов



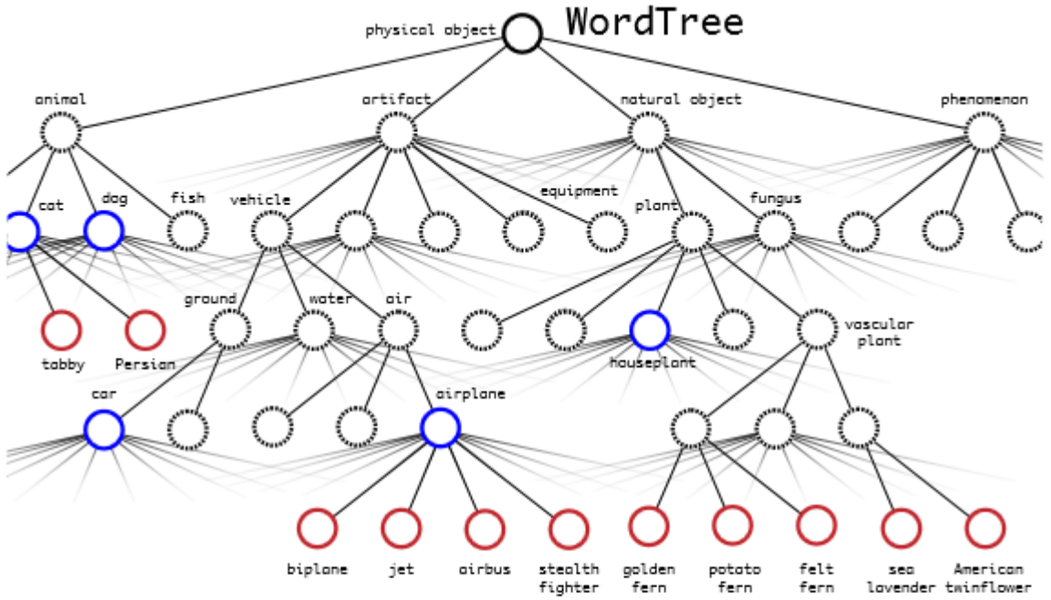
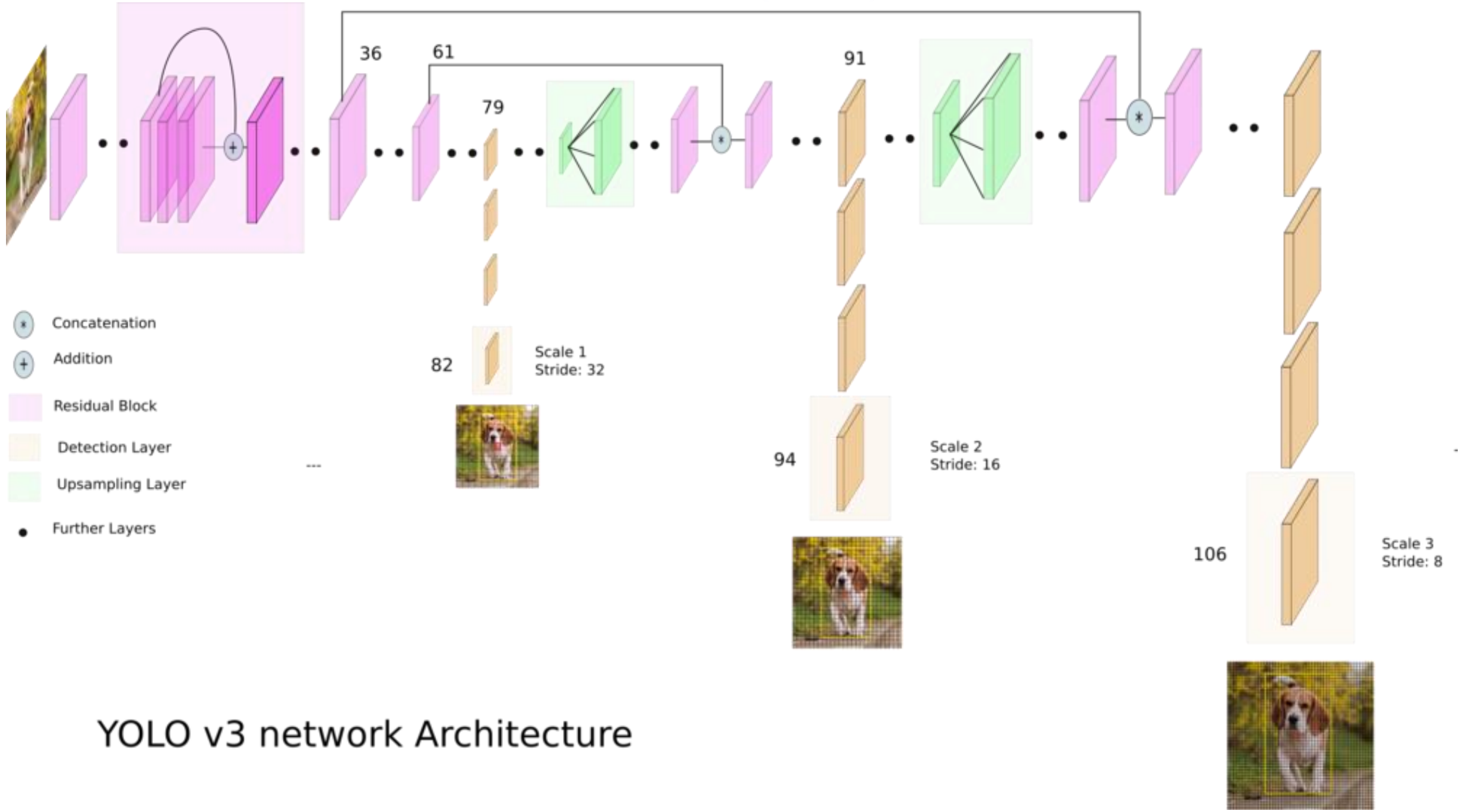


Figure 6: Combining datasets using WordTree hierarchy. Using the WordNet concept graph we build a hierarchical tree of visual concepts. Then we can merge datasets together by mapping the classes in the dataset to synsets in the tree. This is a simplified view of WordTree for illustration purposes.

WordTree для обучения на разных датасетах (с разными классами)

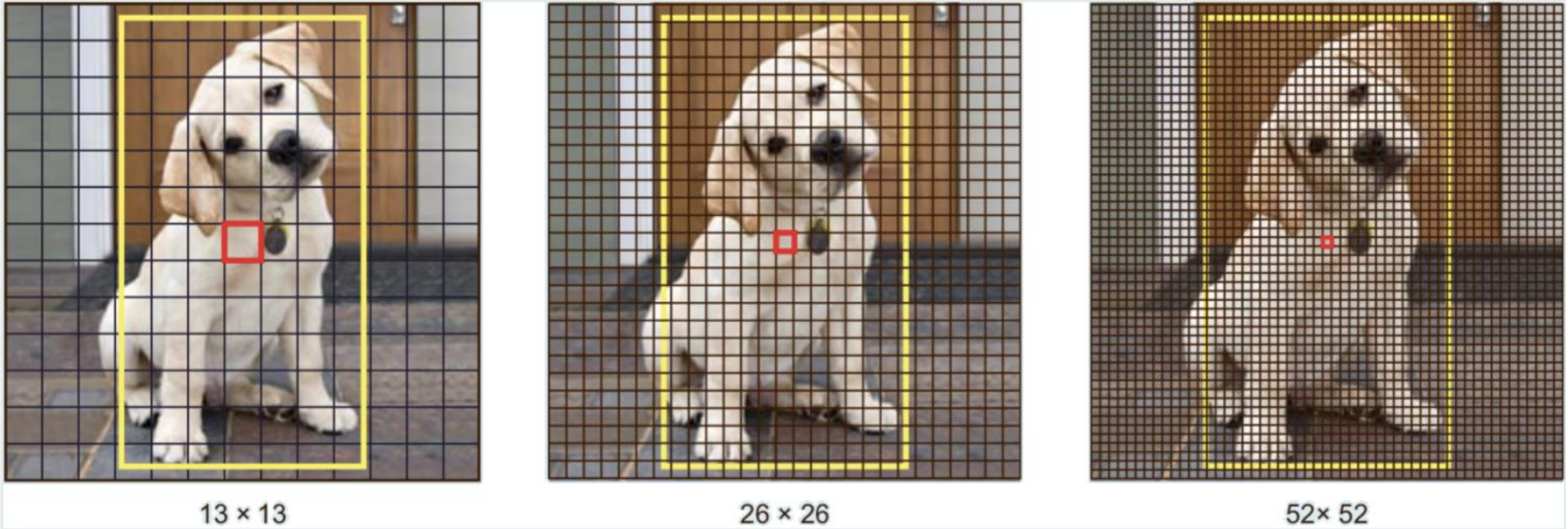
YOLOv3 (2018)



YOLO v3 network Architecture

на выходе есть три слоя – для обнаружения объектов разного размера, нет pool (есть stride)

YOLOv3



работа на разных разрешениях

[Mohamed Elgendy]

YOLOv3

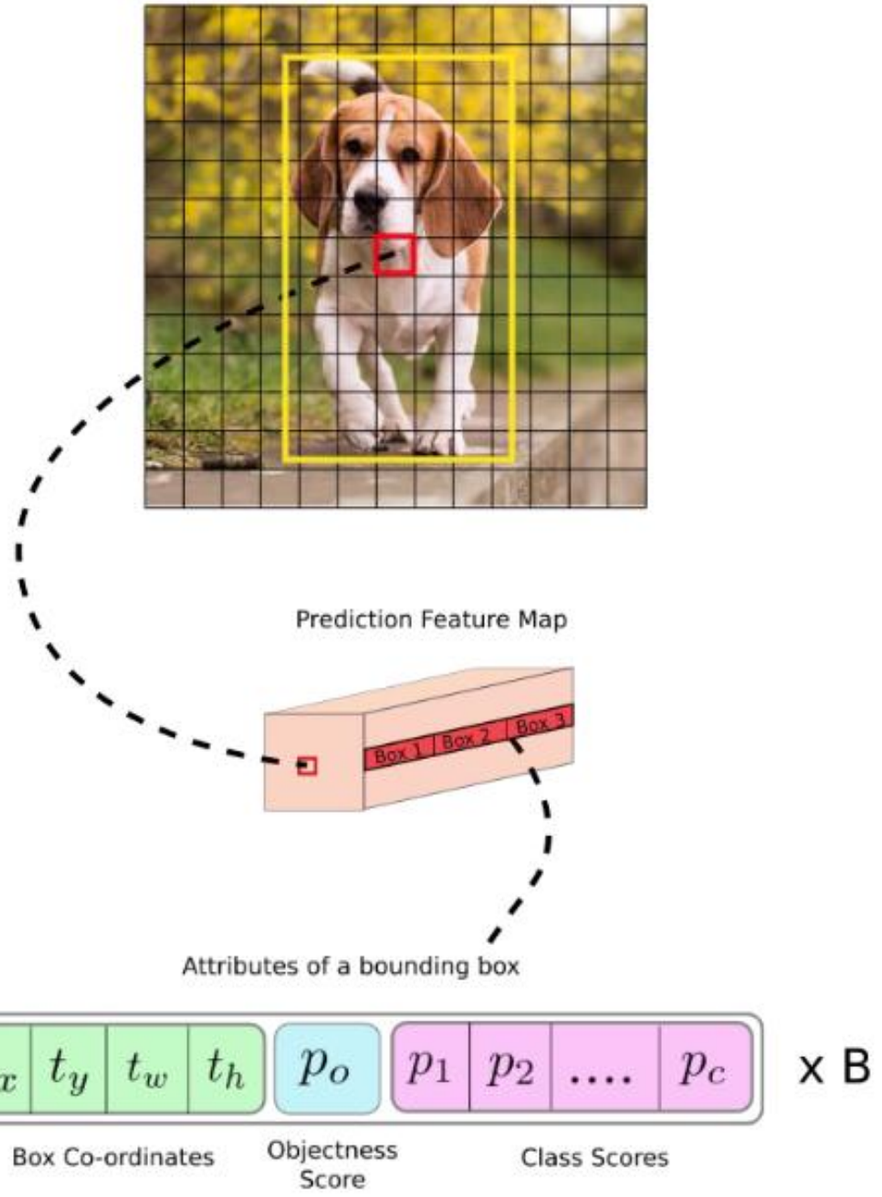
на вход сети изображение

проходит через предобученную (на классификации) свёрточную часть Darknet-53

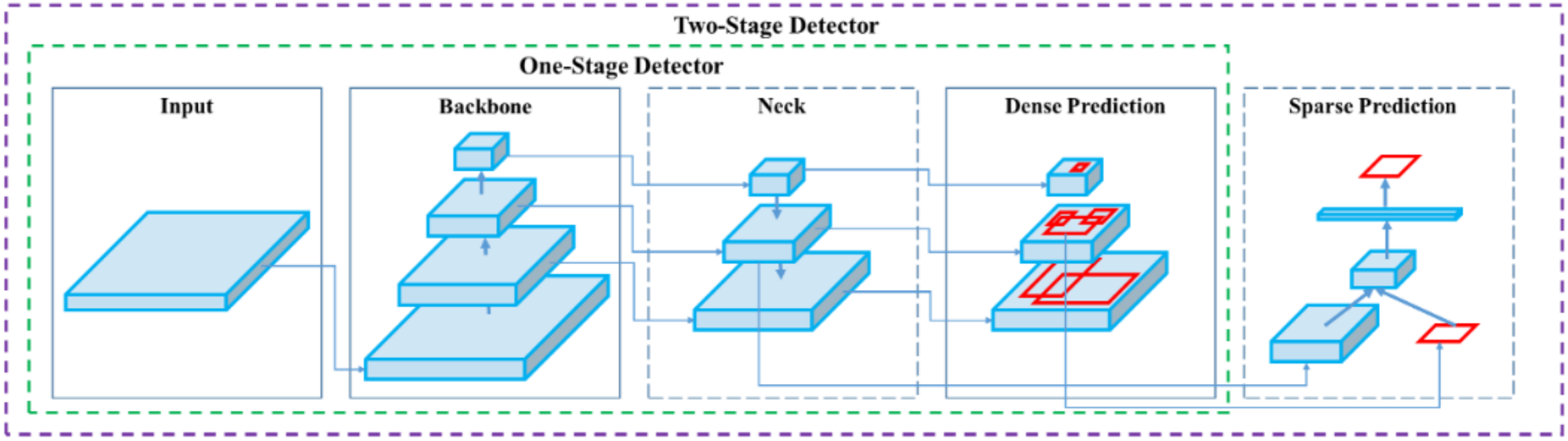
на выходе тензор размера 13×13×B(5+C)

выучиваются смещение якорей

отсечение по порогу objectness score и NMS (на всех размерах)



YOLOv4



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], … }

Figure 2: Object detector.

С подобной архитектурой познакомимся дальше, на примере других сетей

Alexey Bochkovskiy «YOLOv4: Optimal Speed and Accuracy of Object Detection» // https://arxiv.org/pdf/2004.10934.pdf

Single Shot MultiBox Detector (SSD)

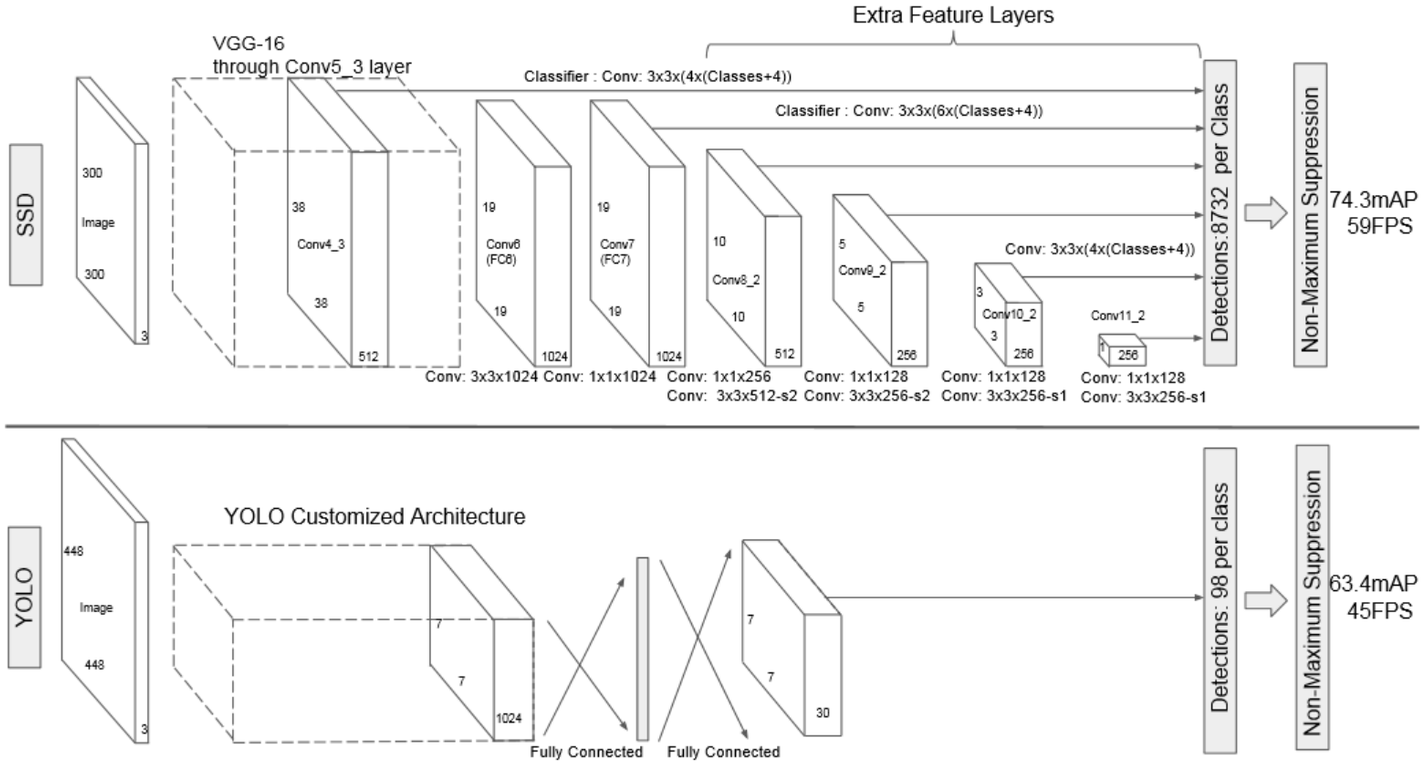
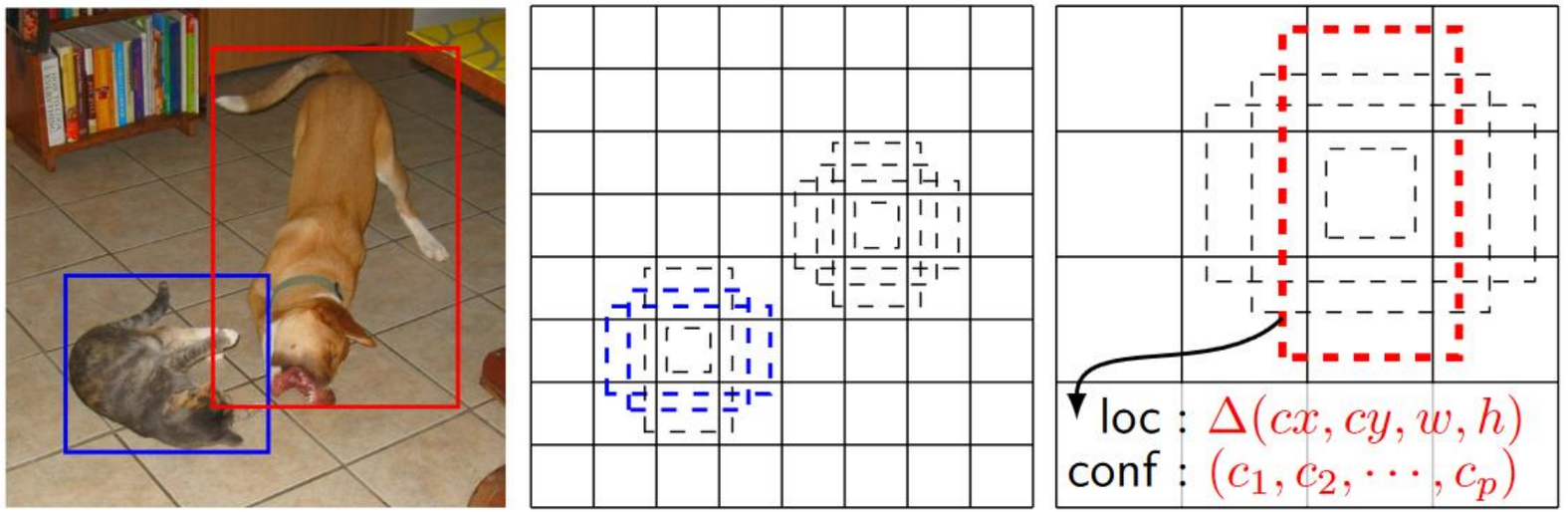


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300×300 input size significantly outperforms its 448×448 YOLO counterpart in accuracy on VOC2007 `test` while also improving the speed.

[W. Liu и др. https://arxiv.org/pdf/1512.02325.pdf]

Single Shot MultiBox Detector (SSD)



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

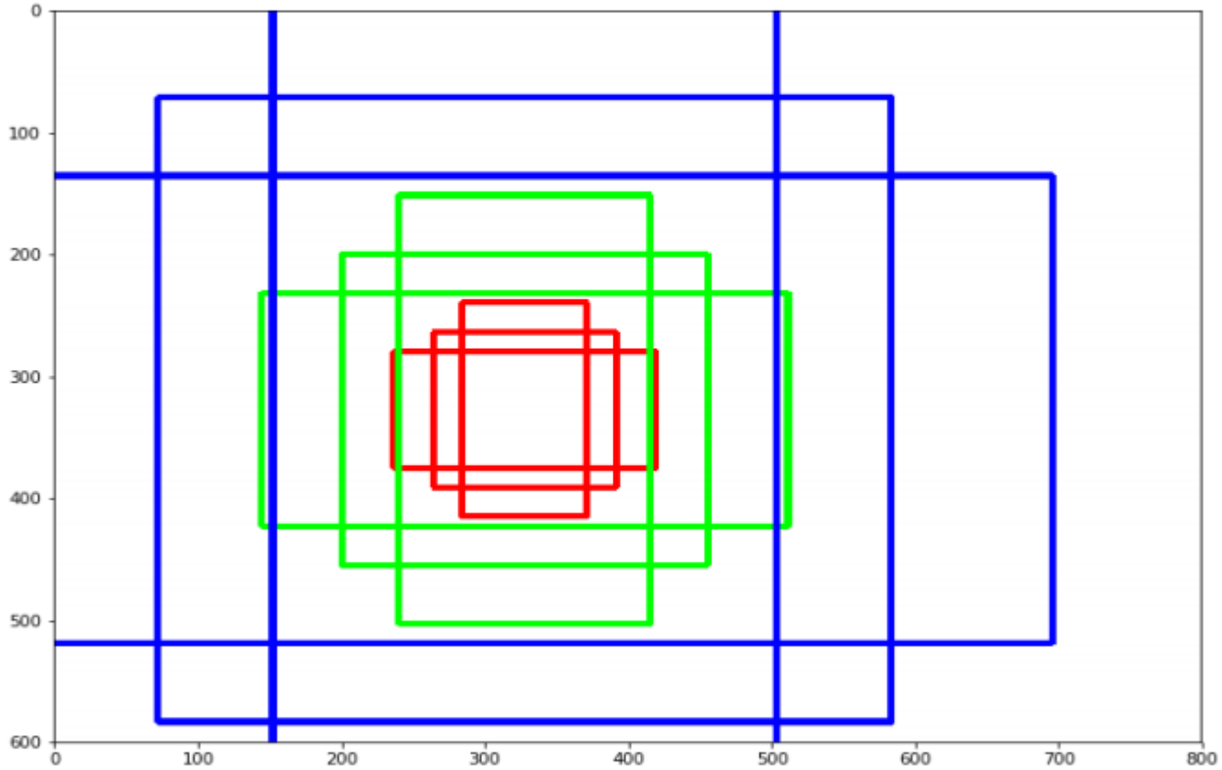
Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 in (b) and (c)). For each default box, we predict both the shape offsets and the confidences for all object categories ($(c_1, c_2, \cdots, c_p)$). At training time, we first match these default boxes to the ground truth boxes. For example, we have matched two default boxes with the cat and one with the dog, which are treated as positives and the rest as negatives. The model loss is a weighted sum between localization loss (e.g. Smooth L1 [6]) and confidence loss (e.g. Softmax).

SSD: якоря (anchors)

base anchor (пусть 128 × 128)

anchor scales (пусть {1, 2, 3})

anchors aspect ratios (пусть 1:1, 1:2, 2:1)



Идея как в Region Proposal Network (RPN), но там для каждого региона «есть объект / нет», а тут сразу класс определяем

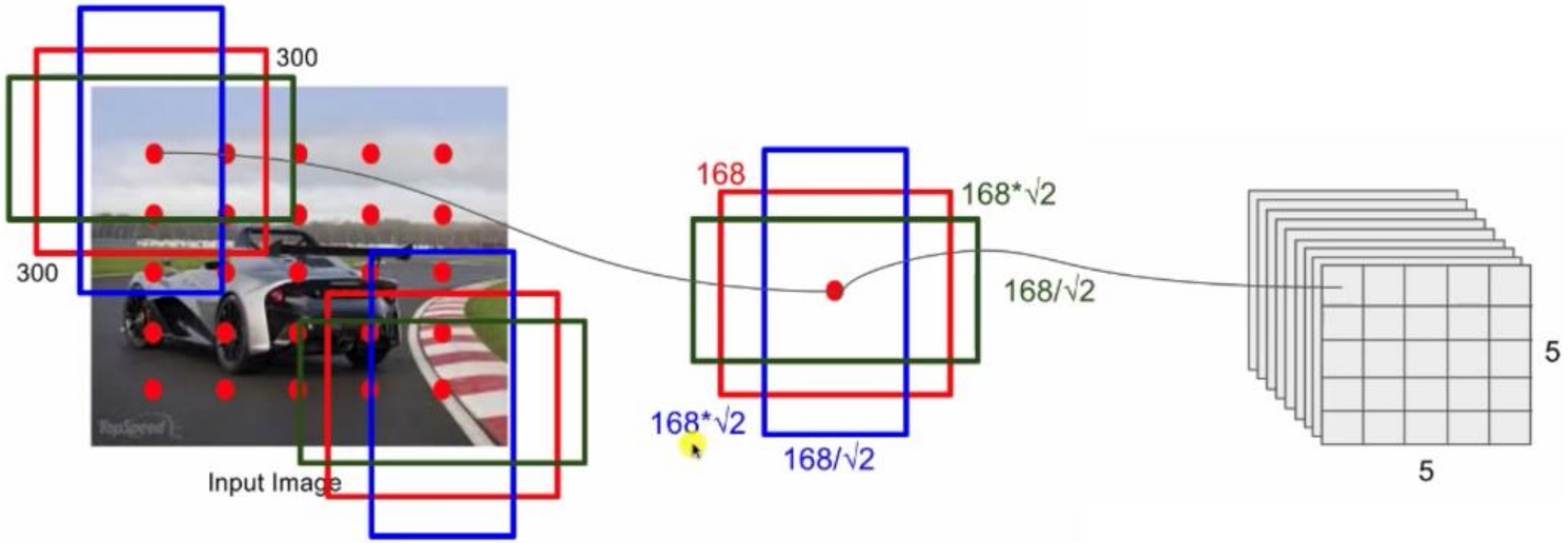
Single Shot MultiBox Detector (SSD)

Вход 300×300 изображение

Нужны истинные рамки (на обучении)

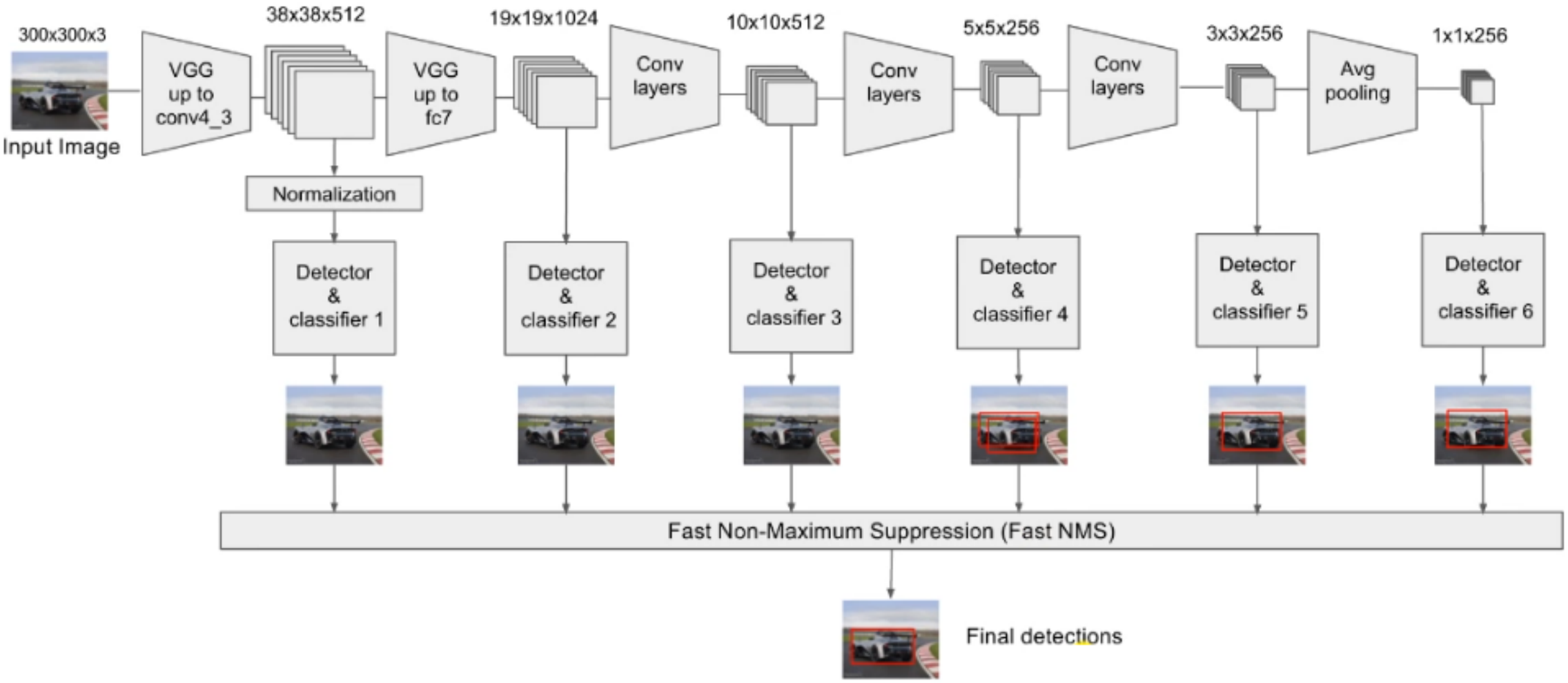
Задаём несколько (на картинке 3) «default boxes» – для каждого

- **коррекция положения**
- **оценки принадлежности к классам**



https://deepsystems.ai

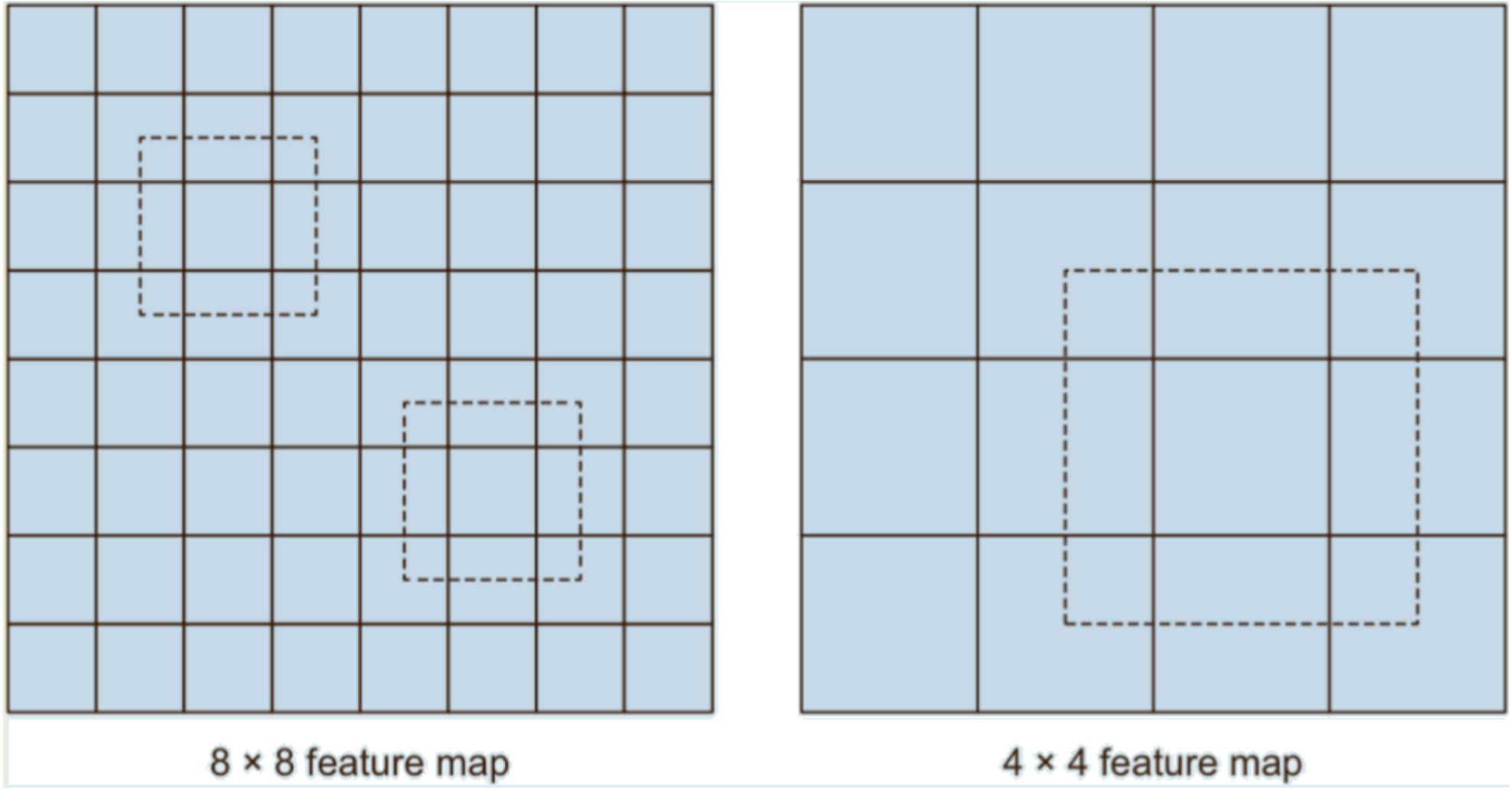
Single Shot MultiBox Detector (SSD)



Много регионов «default boxes» на разных масштабах

https://deepsystems.ai

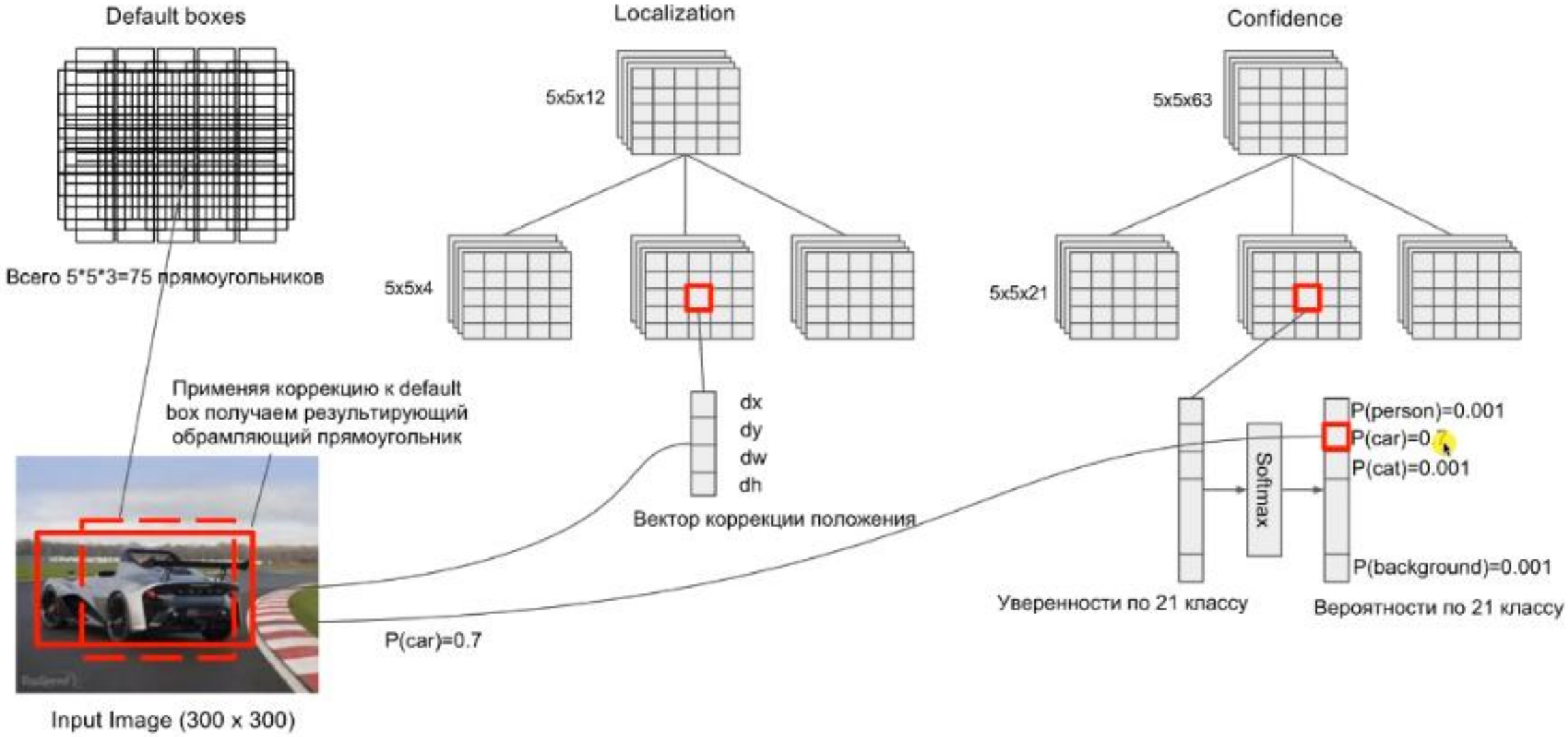
Single Shot MultiBox Detector (SSD)



детектирование на разных масштабах

[Mohamed Elgendy]

Single Shot MultiBox Detector (SSD)



https://deepsystems.ai

Single Shot MultiBox Detector (SSD)

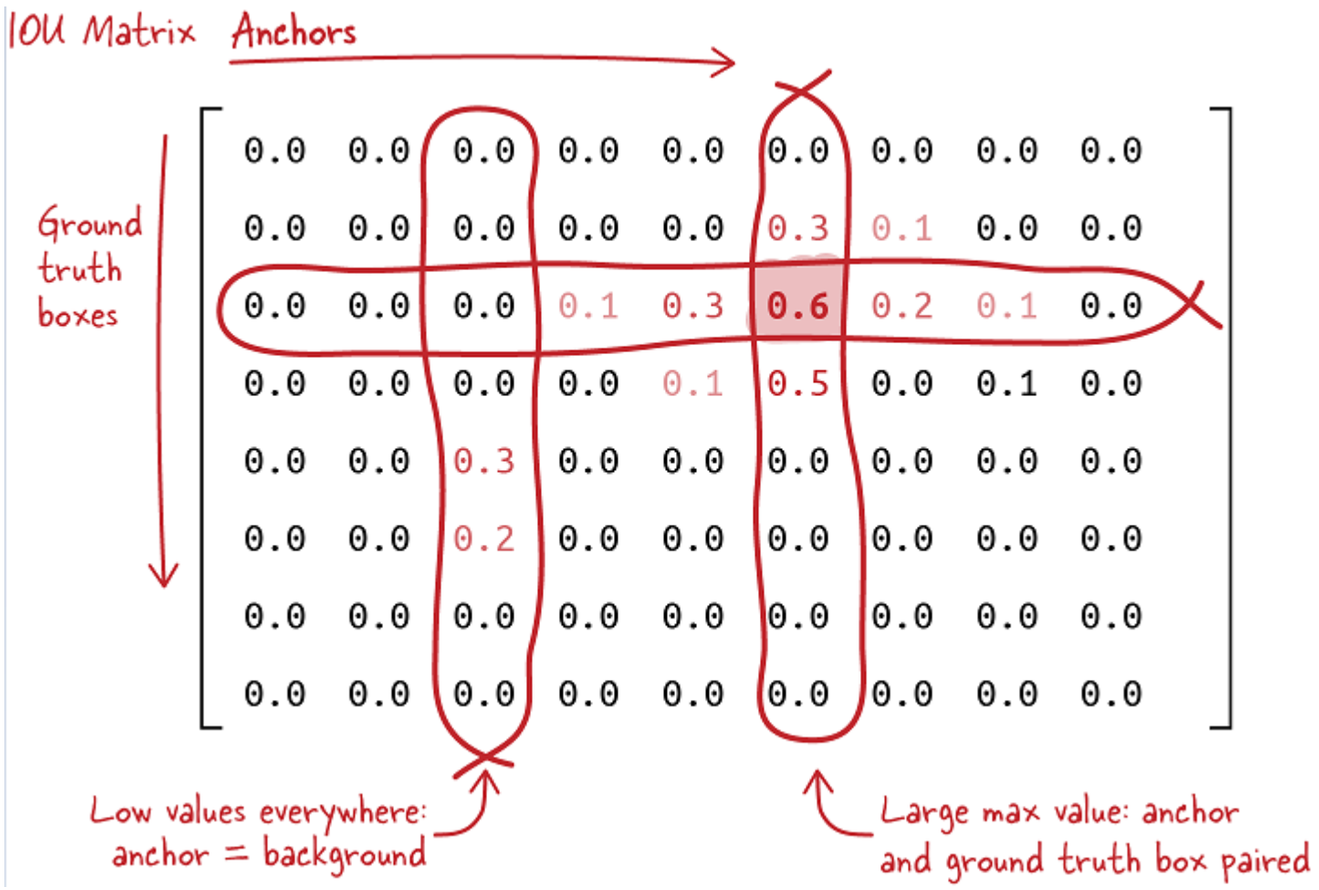
размеры выхода: 5×5 (сетка) × #якорей (default boxes) × (# классов + 1)
5×5 (сетка) × #якорей (default boxes) × 4 (регрессия для регионов)

- **очень быстро с хорошим качеством**
- **детектирование на разных масштабах**
- **большое число default boxes на разных масштабах**

быстрая, почти как YOLO
но снимается ограничение на 2 объекта в ячейке

в отличии от YOLO
- **нет p_obj – а только вероятности**
- **VGG в основе, а не Darknet**
- **hard-negative mining (отношение +/– = 1/3)**
- **другая ошибка (без подробностей)**

Как сопоставить якоря ↔ истинные регионы

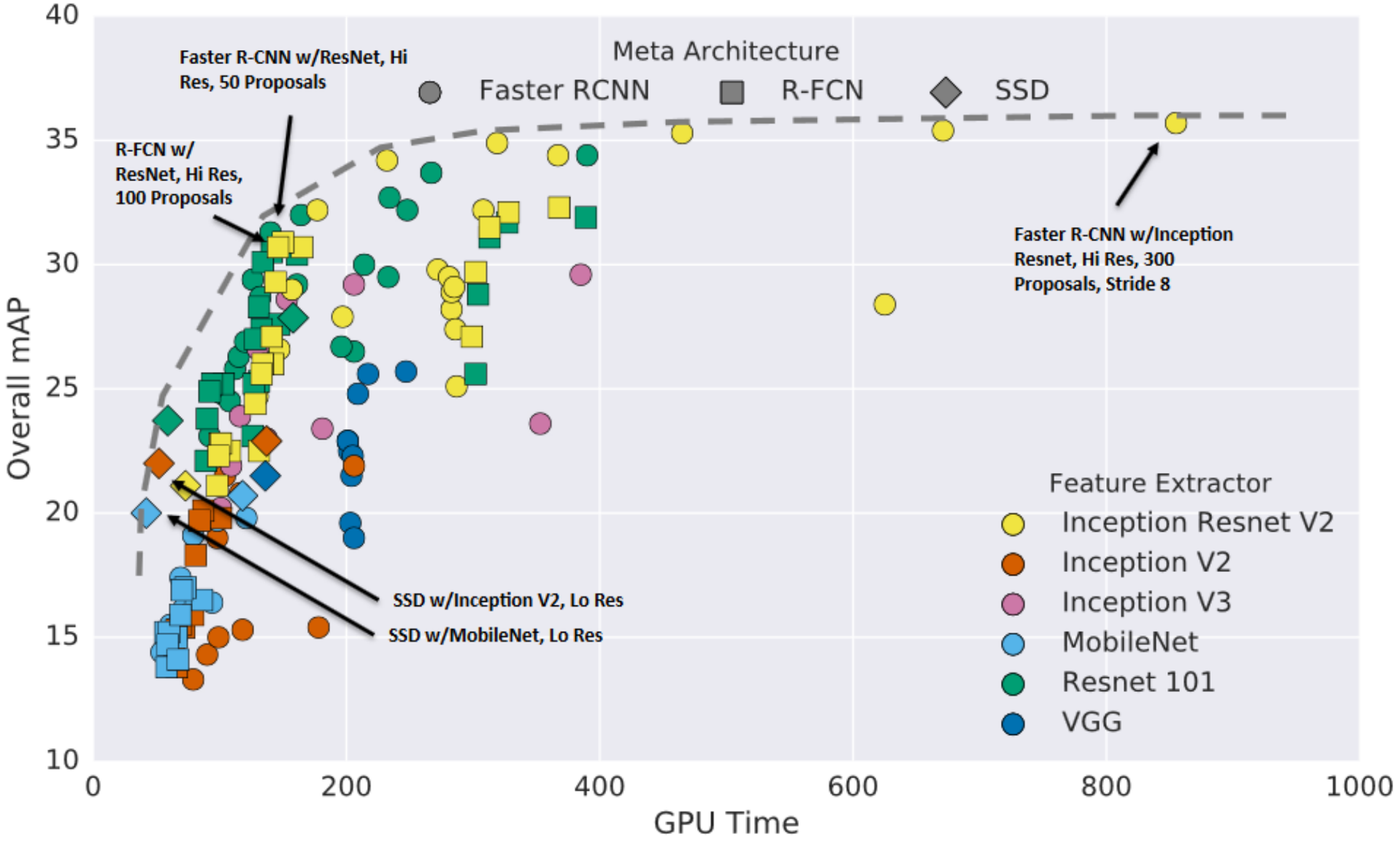


надо найти наиболее подходящие

(и пытаться «модифицировать» форму и размеры к ним)

так точно в RetinaNet делается

Детектирование объектов: сравнение



Speed/accuracy trade-offs for modern convolutional object detectors

[Jonathan Huang и др. 2017 https://arxiv.org/pdf/1611.10012.pdf]

Итог

история перехода к полностью НС-решениям

локализация объектов не обязательно должна быть классоориентированной
«class-specific»

в одном регионе можно одновременно детектировать несколько объектов!

есть способ генерирования «якорей» – большого набора потенциальных регионов
есть способ обойтись без генерации регионов!

«Пирамидные сети» – способ формирования признакового пространства

Хорошая идея: каждая точка потенциальный представитель рамки
(оцениваем её параметры)