

marge_simpson: 96%

krusty_the_clown: 95%

Детектирование объектов на изображениях

Александр Дьяконов

16 марта 2020 года

План: задачи с изображениями

Классификация – что изображено

Локализация – где изображено

Детектирование – что и где

Сегментация – матрица меток сегментов

Преобразование изображений

- удаление шума
- стилизация

Восстановление объектов (ex: 3D-модели)

Классификация изображений – почему нетривиальная задача

освещение меняется



ракурсы / деформации / позы



[<http://cs231n.stanford.edu/2017/syllabus.html>]

Классификация изображений – почему нетривиальная задача

Окклюзия



Сливание с фоном



Классификация изображений



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

Формально уже почти решённая задача!

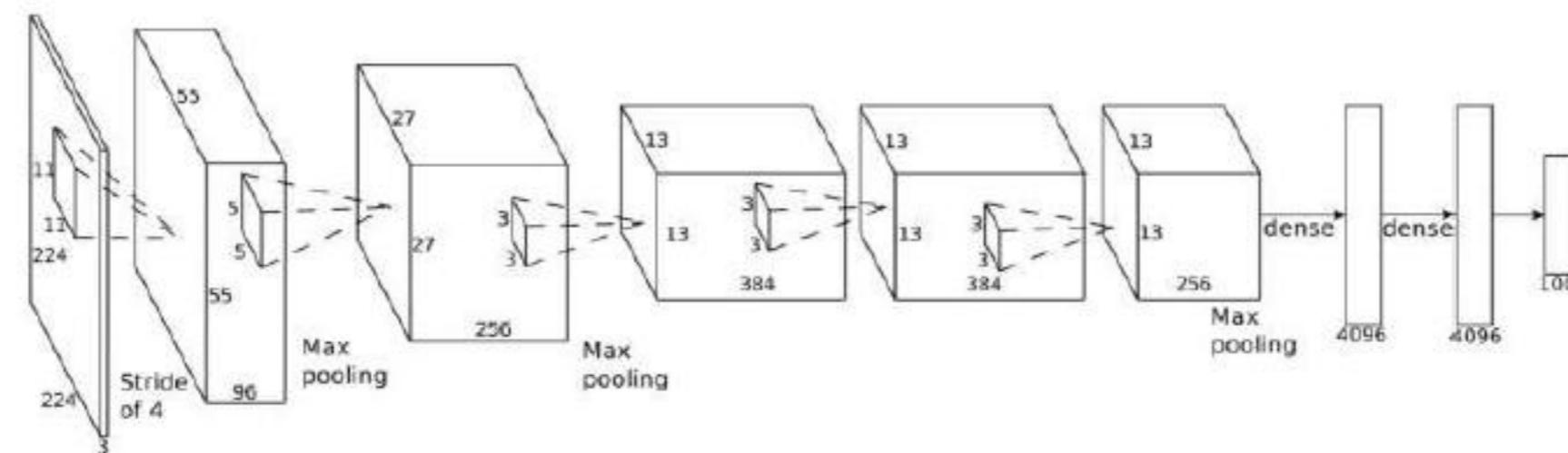
Можно использовать любую архитектуру / преднастроенную НС

Вход – изображение

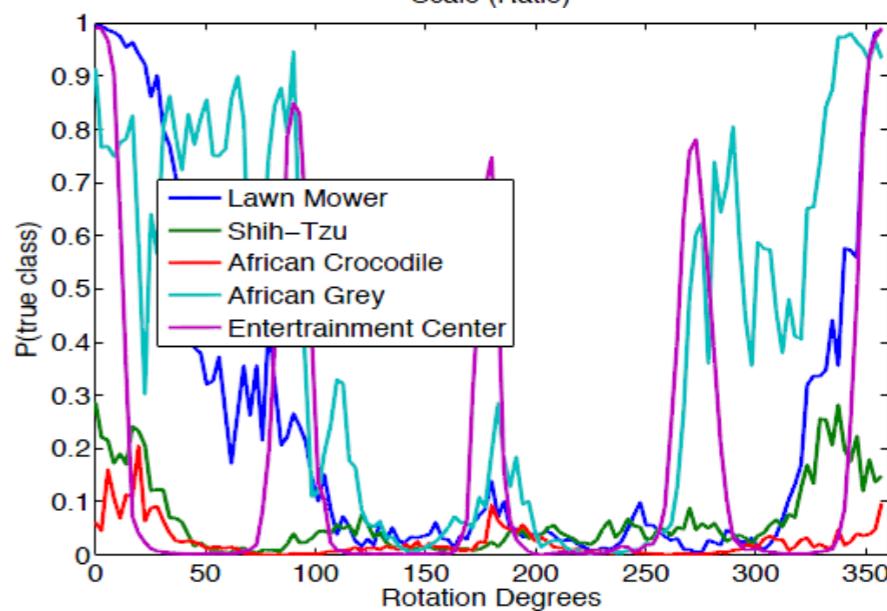
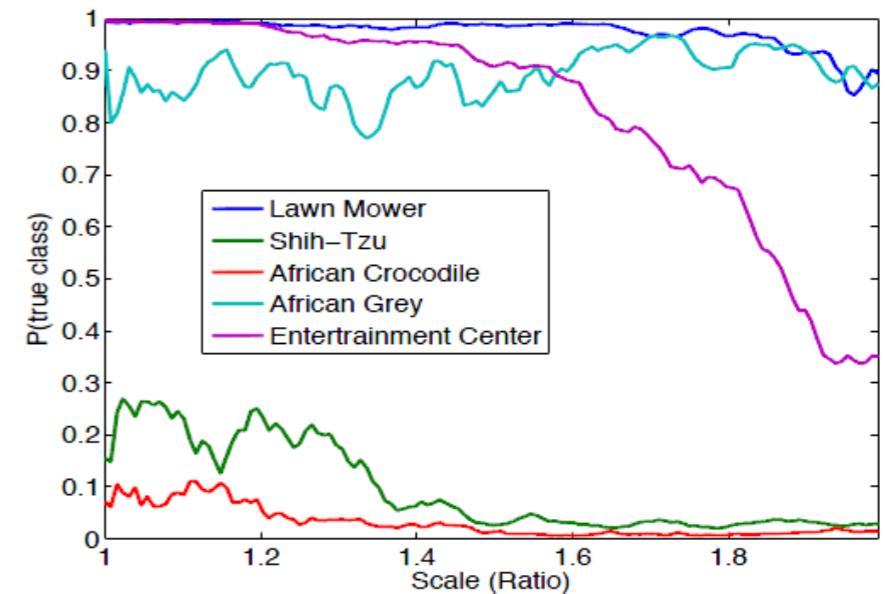
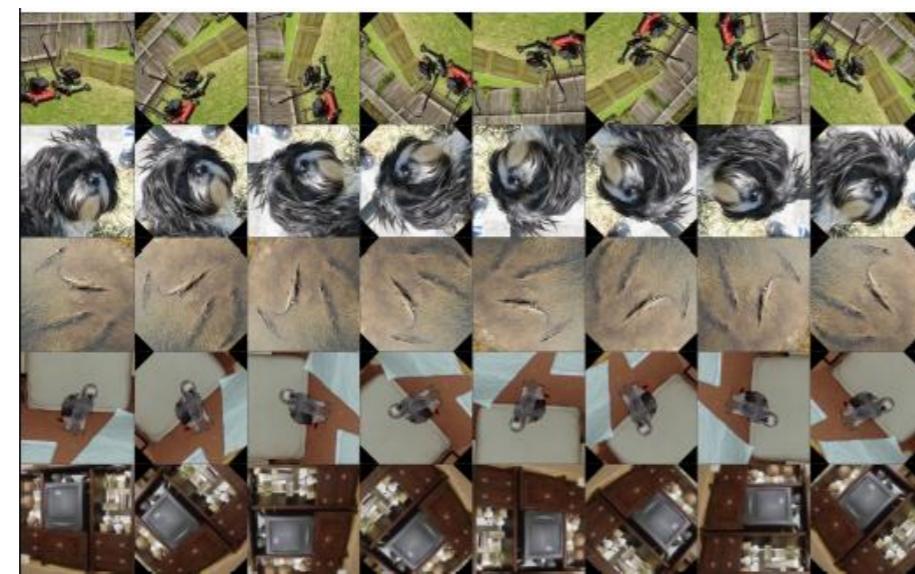
Выходы – классы

Ошибка – LogLoss

Сначала свёрточные слои, потом – полносвязные



Проблемы классификации: если менять масштаб и вращать



Другие задачи типа «что и где изображено»

Классификация + локализация



«cat»

Детектирование объектов (Object Detection)



DOG, DOG, CAT

Семантическая сегментация (Semantic Segmentation)



GRASS, CAT, TREE, SKY

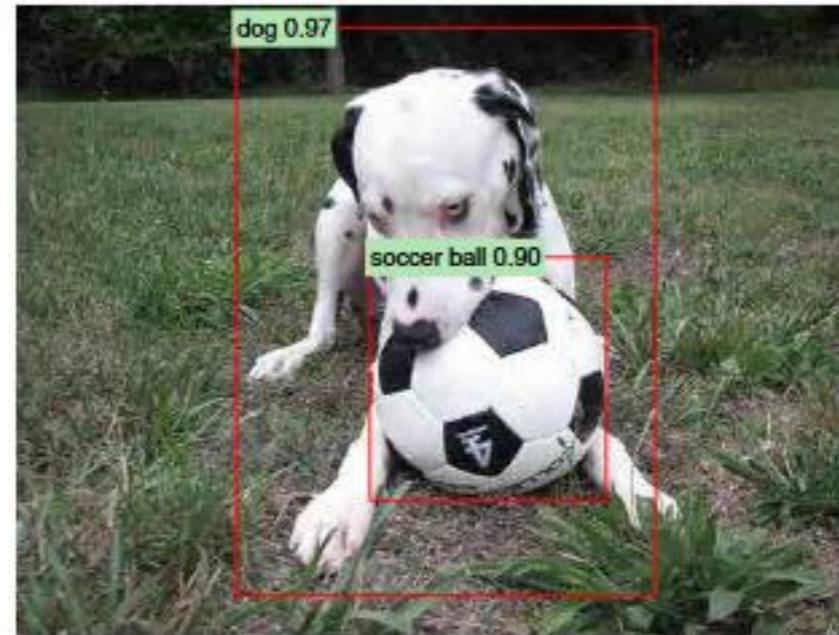
Сегментация объектов (Instance Segmentation)



DOG, DOG, CAT

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

Детектирование объектов = Локализация + Классификация



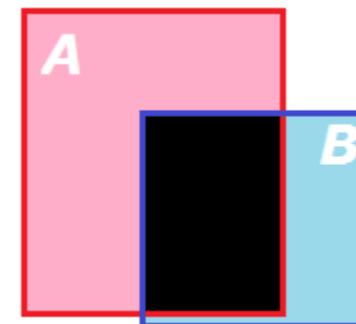
Локализация (localization) объекта – где

Классификация – что
м.б. ещё определяем параметры объекта

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

Метрики качества

Правильное детектирование, если



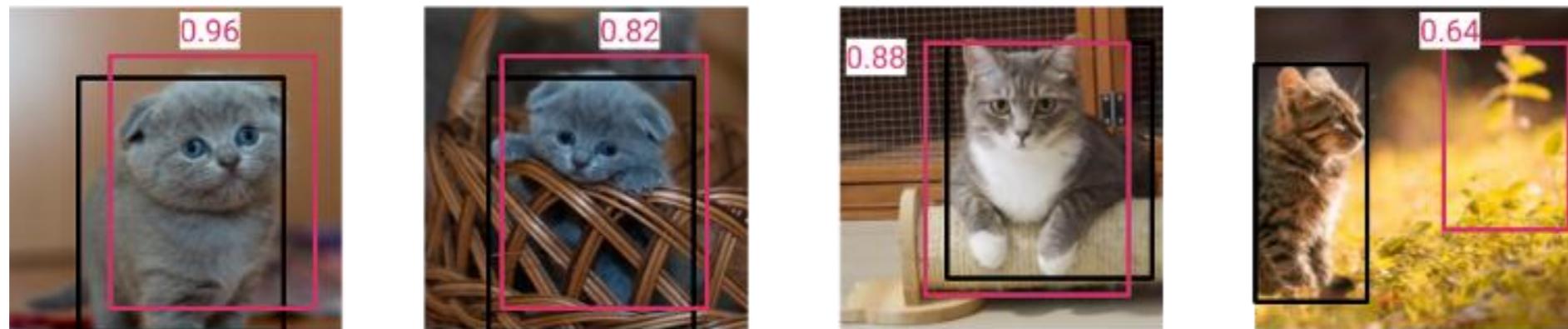
$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \geq \alpha$$

Intersection Over Union

TP, FN, FP
Precision / Recall

Average Precision (AP)

~ по задаче классификации региона



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

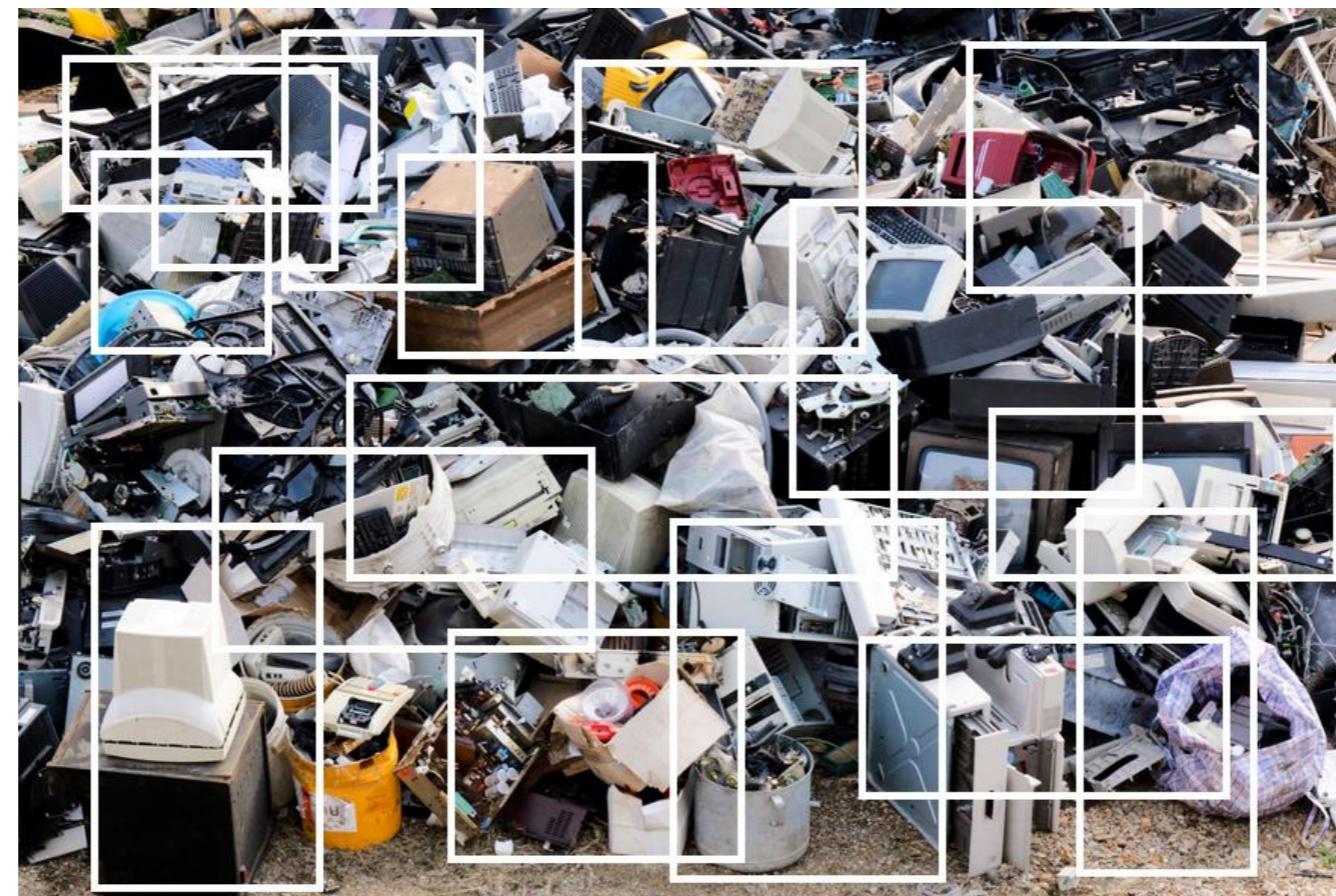
$$PQ = \frac{\sum_{(p,g) \in TP} IoU(p, g)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}$$

$$PQ = \underbrace{\frac{\sum_{(p,g) \in TP} IoU(p, g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}}$$

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

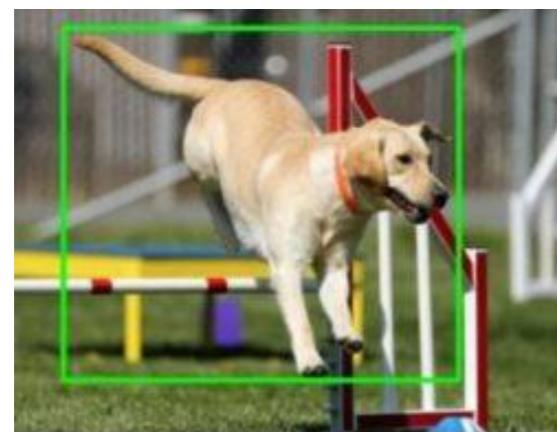
Детектирование объектов

**Формально – перебрать разные локализации
(чаще всего используют прямоугольники)
и для каждого – классификация**

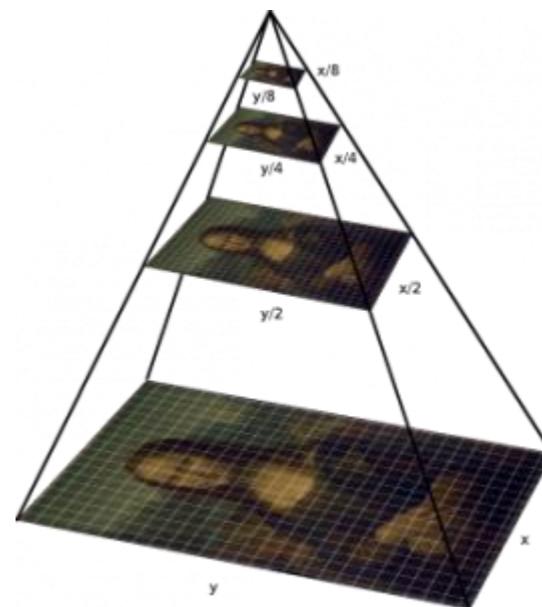


Детектирование объектов

**Реально –
проблема размера**



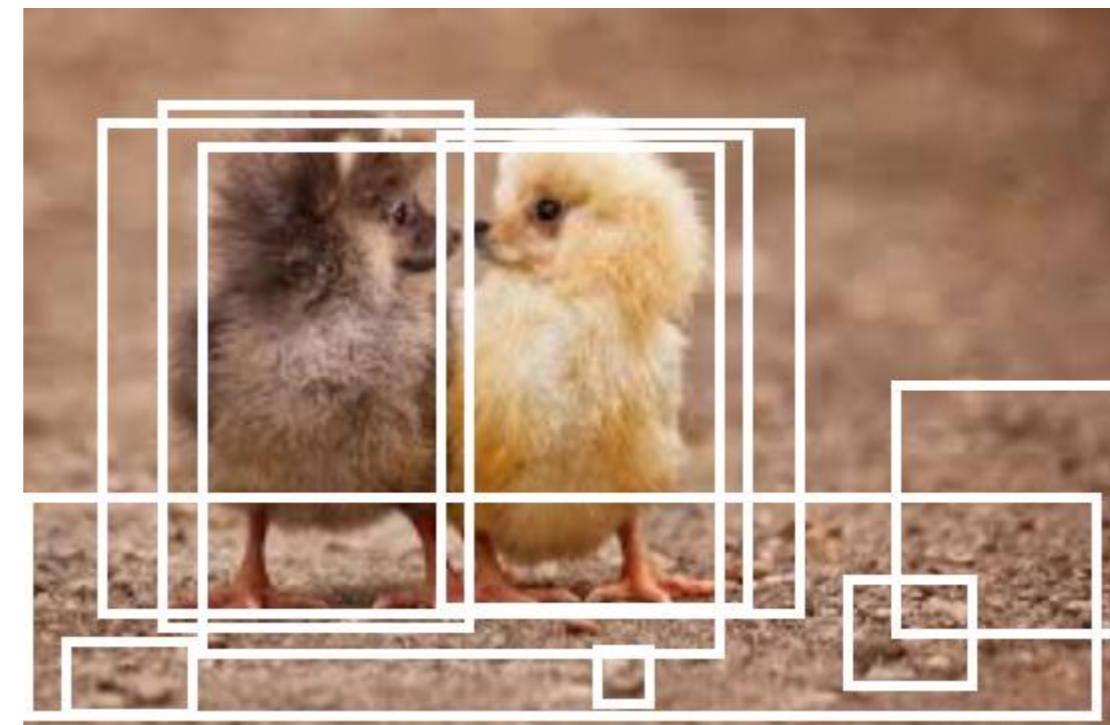
Решение –



**Использовать изображения разных масштабов +
фиксированный размер области**

Генерация регионов (Region Proposals)

**Много разных методов генерации областей,
где потенциально есть объекты**



Работают очень быстро

Selective Search

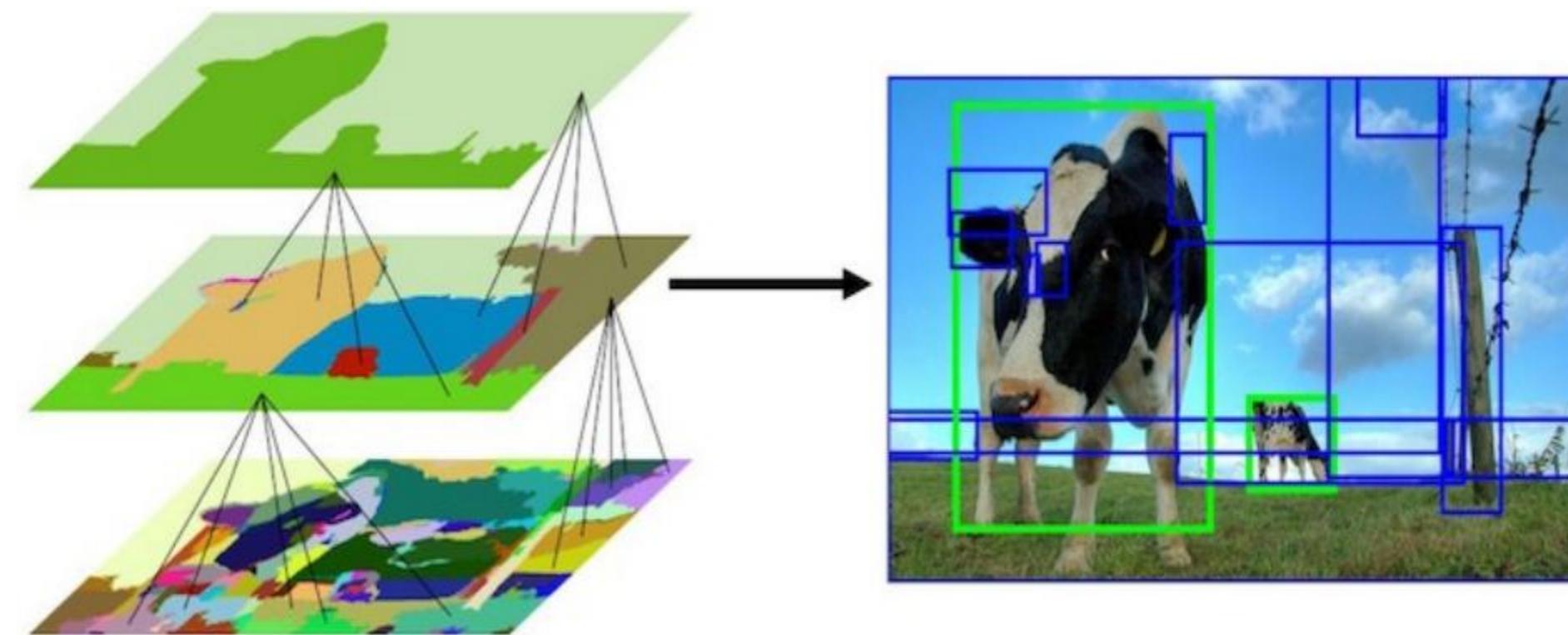


Figure 2: Two examples of our selective search showing the necessity of different scales. On the left we find many objects at different scales. On the right we necessarily find the objects at different scales as the girl is contained by the tv.

<http://www.huppelen.nl/publications/selectiveSearchDraft.pdf>

Selective Search

- 0. список = []**
- 1. Разбить изображение на сегменты**
- 2. Для каждого сегмента в список его обрамляющую рамку**
- 3. Объединить соседние похожие сегменты**
- 4. Если число сегментов > 1 перейти к 2**
- 5. Для каждой рамки из списка оценить наличие в ней объекта**



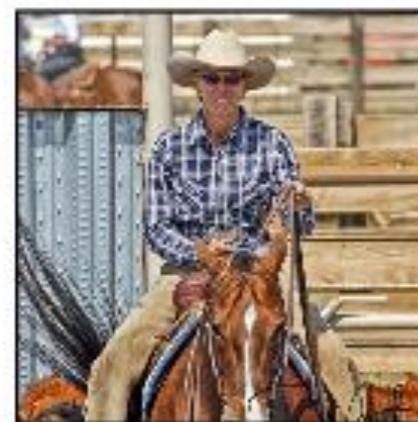
Детектирование объектов

«R-CNN – Object detection and semantic segmentation

[Girshick et al., 2013 <https://arxiv.org/pdf/1311.2524.pdf>]

«Fast R-CNN» – [Girshick 2015 <https://arxiv.org/pdf/1504.08083.pdf>]

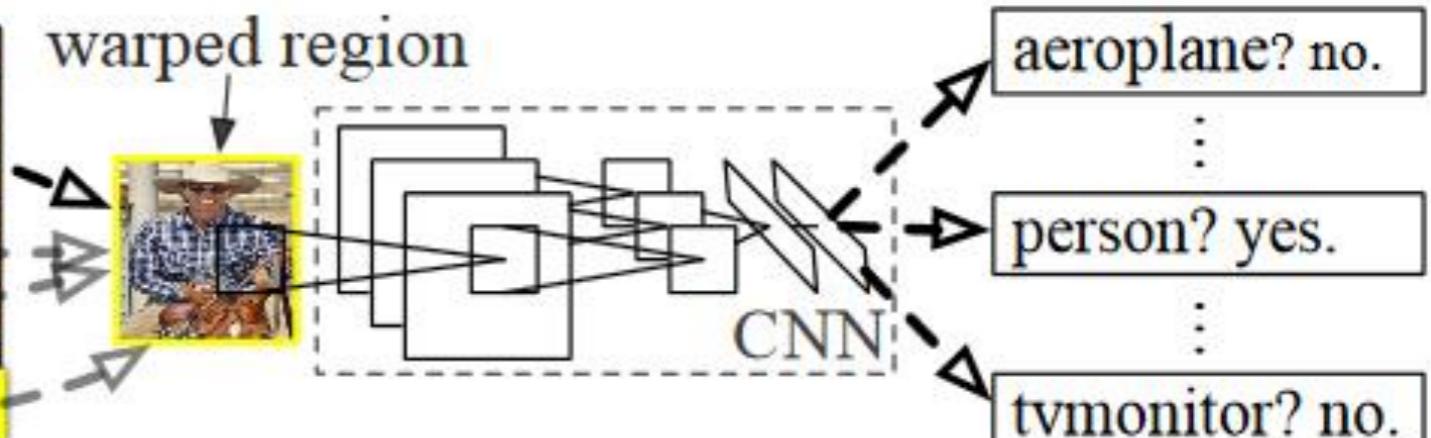
«Faster R-CNN» – [Ren et al., 2015 <https://arxiv.org/abs/1506.01497>]



1. Input
image



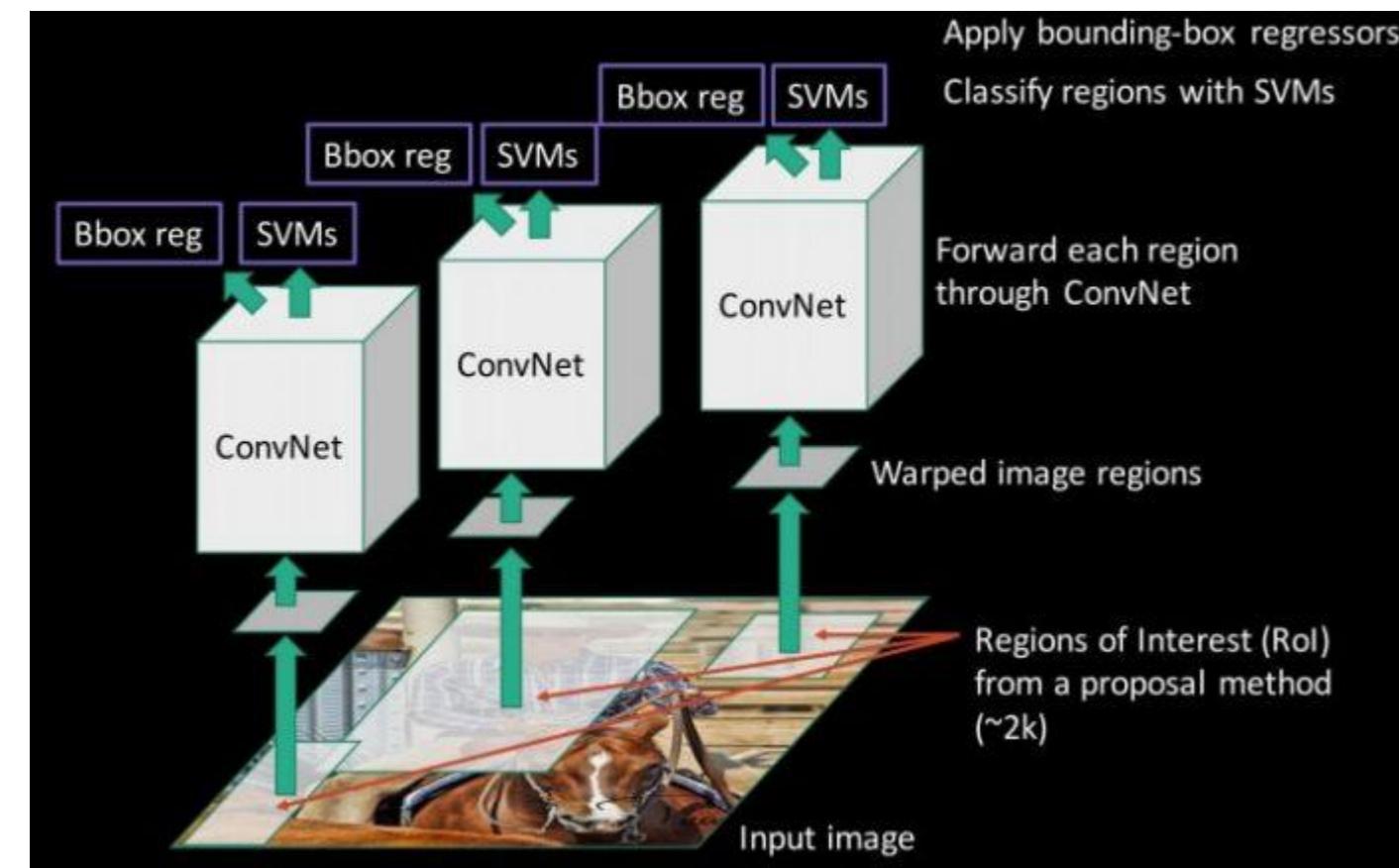
2. Extract region
proposals (~2k)



3. Compute
CNN features

4. Classify
regions

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**

R-CNN: Визуализация

Как понять, что сеть учится...

**Для нейрона (в последних слоях) упорядочить все входы по его значению,
посмотреть на Топ входов**



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

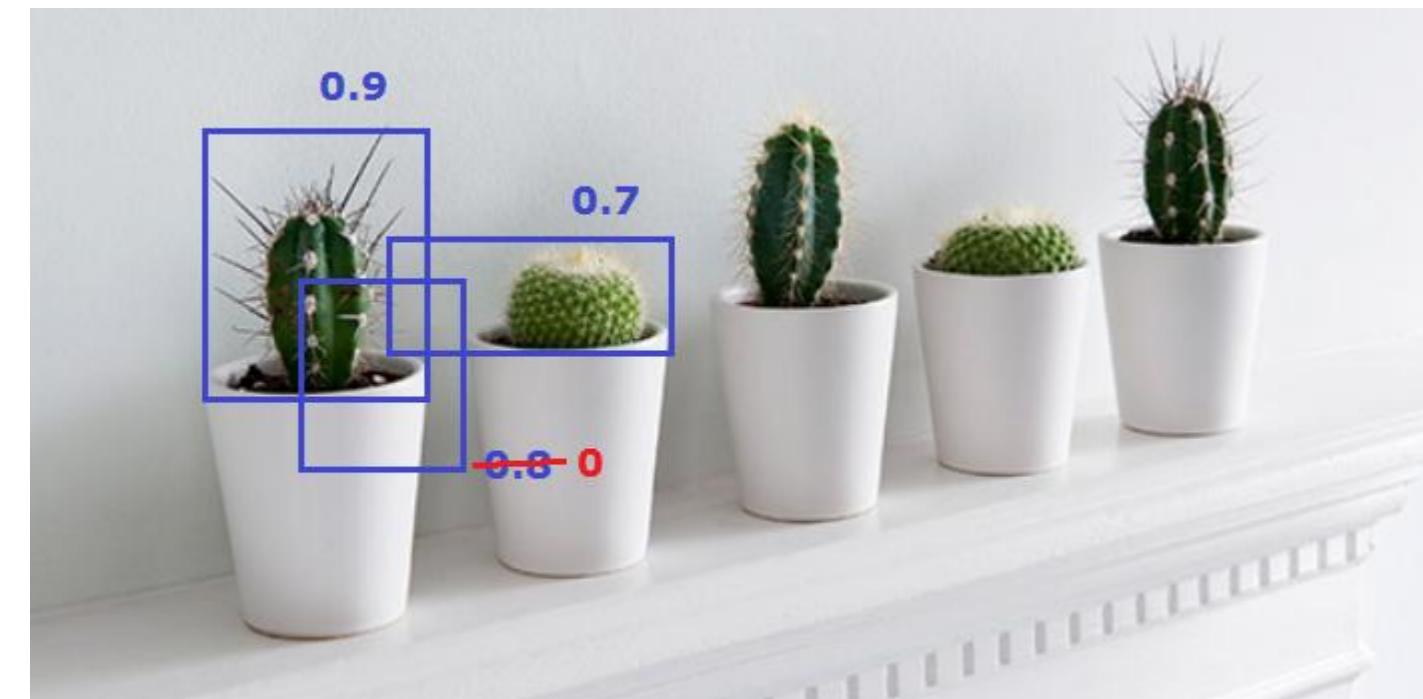
R-CNN: работа сети

- Выбор 2000 регионов (**Selective Search**)
- Регионы → 224×224
- регионы → CNN → SVM/Regression
- non maximum suppression (NMS) **далше**

R-CNN: Недостатки

- Дообучиваем CNN (**log_loss**), потом ещё SVM, потом bounding-box-регрессию...
очень долгое обучение
 - Для каждого региона запускаем CNN

Non Maximum Suppression (NMS)



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**

Spatial Pyramid Pooling (SPP-net)

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

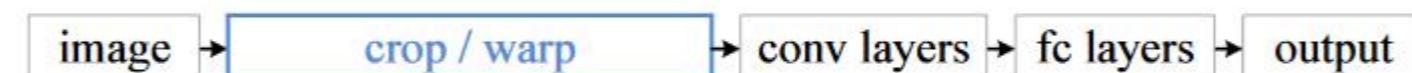


Figure 1: Top: cropping or warping to fit a fixed size. Middle: a conventional CNN. Bottom: our spatial pyramid pooling network structure.

The Spatial Pyramid Pooling Layer

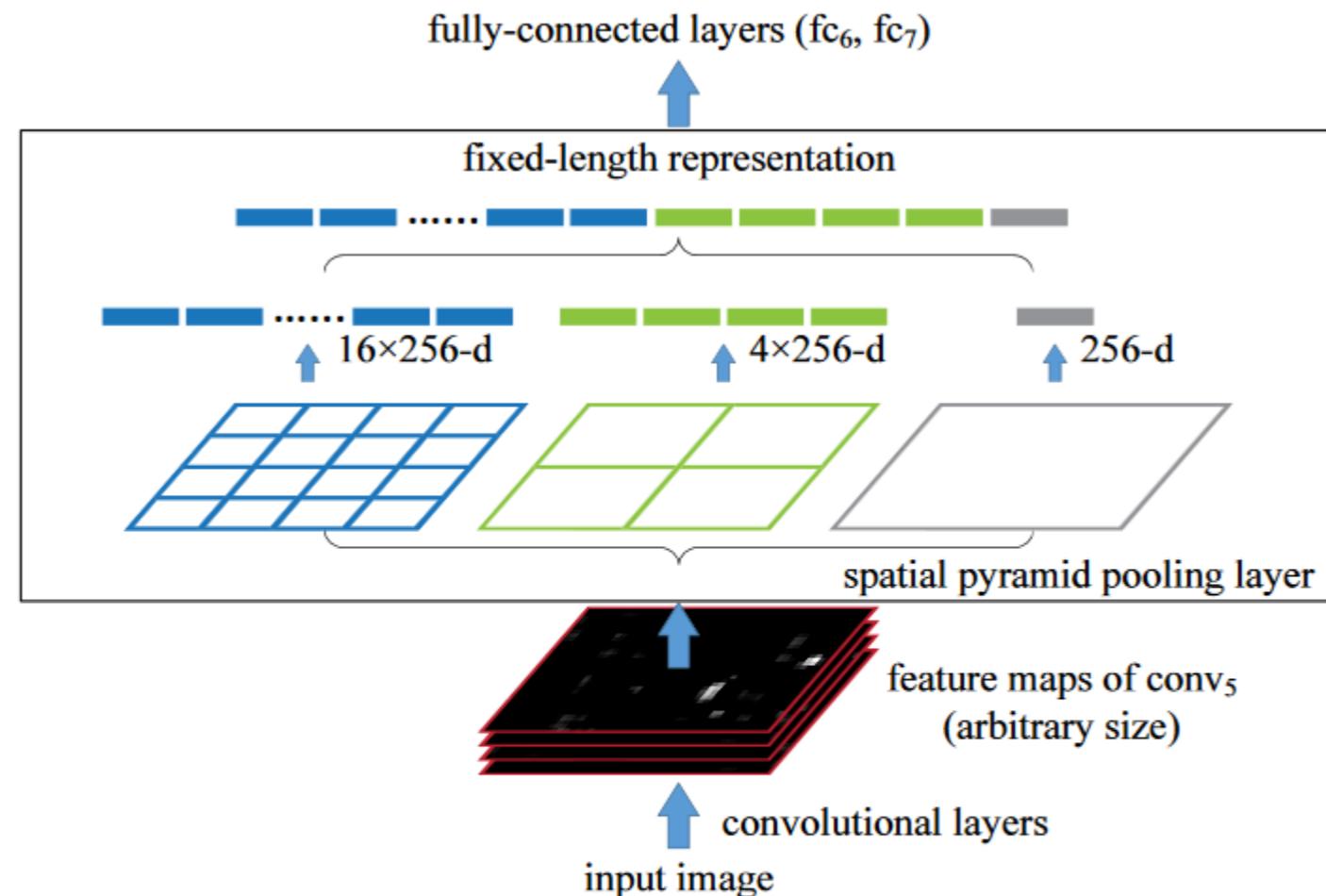
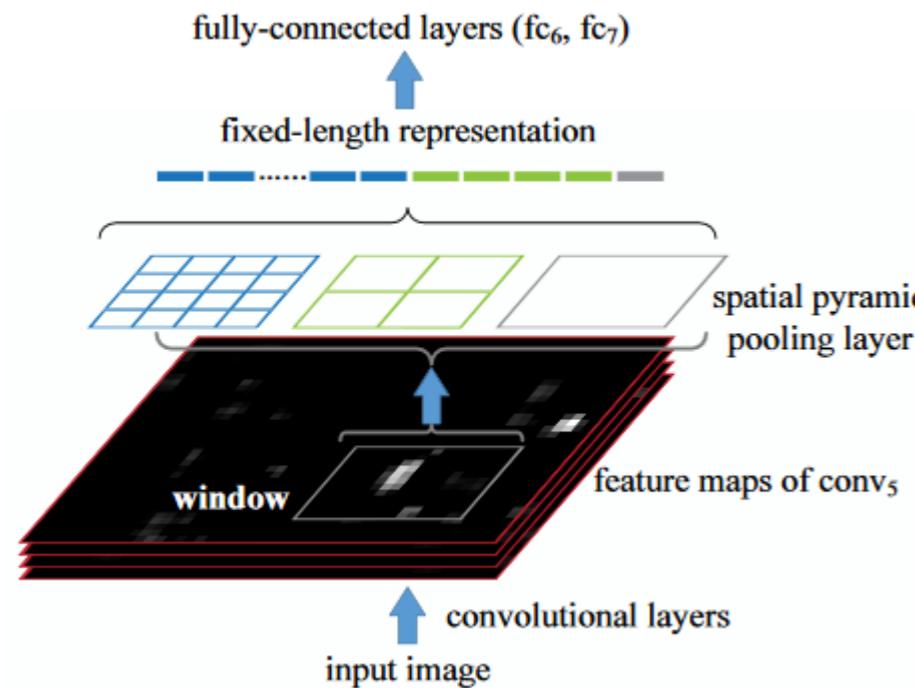


Figure 3: A network structure with a **spatial pyramid pooling layer**. Here 256 is the filter number of the conv₅ layer, and conv₅ is the last convolutional layer.

Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition [He и др. 2015,
<https://arxiv.org/pdf/1406.4729.pdf>]

The Spatial Pyramid Pooling Layer



**Считаем CNN-представление для изображений
один раз**

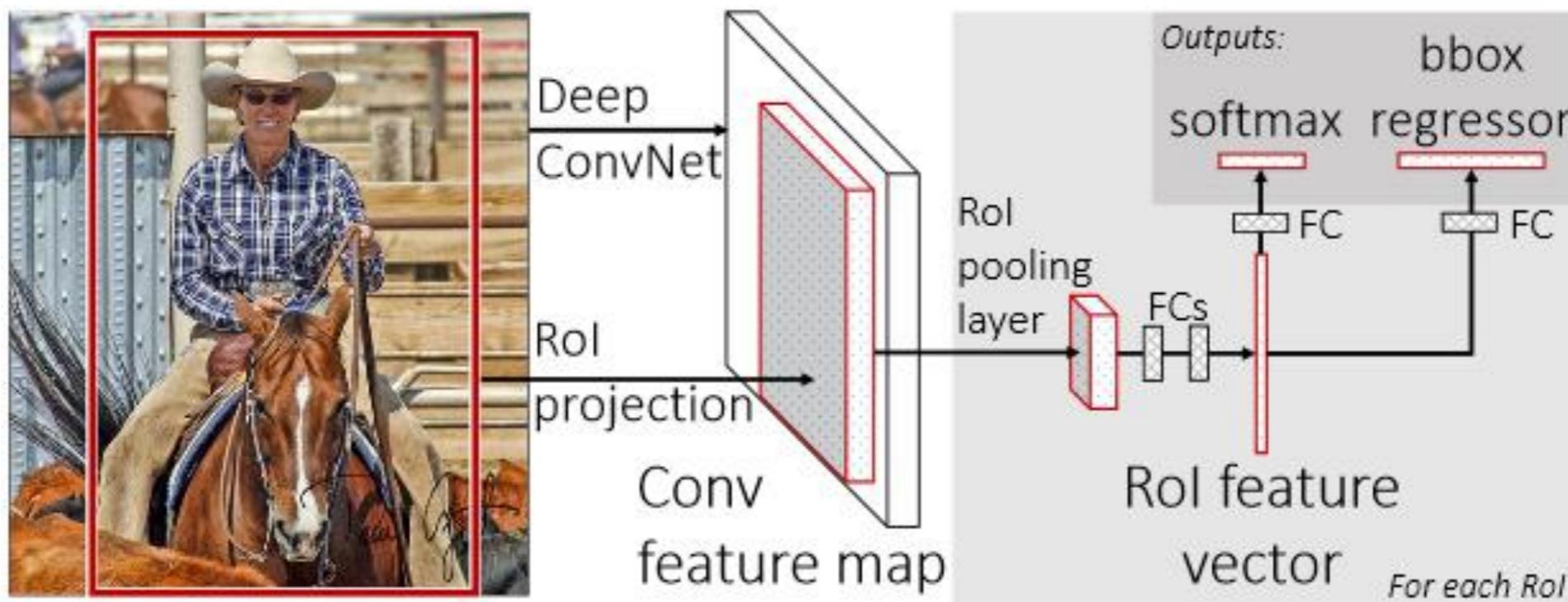
**С помощью SPP-net для каждого региона
получаем признаковое представление
(фиксированной длины)**

**Быстрое решение проблемы того, что регионы разных размеров!
+ сначала-то мы считаем CNN-представление!
а SPP-net просто эффективный способ перевода его в признаки
Сеть не работает на каждом регионе! Сразу на изображении!
Выигрыш по скорости 10x – 100x!**

Fast R-CNN

Fast R-CNN = R-CNN + SPP + 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

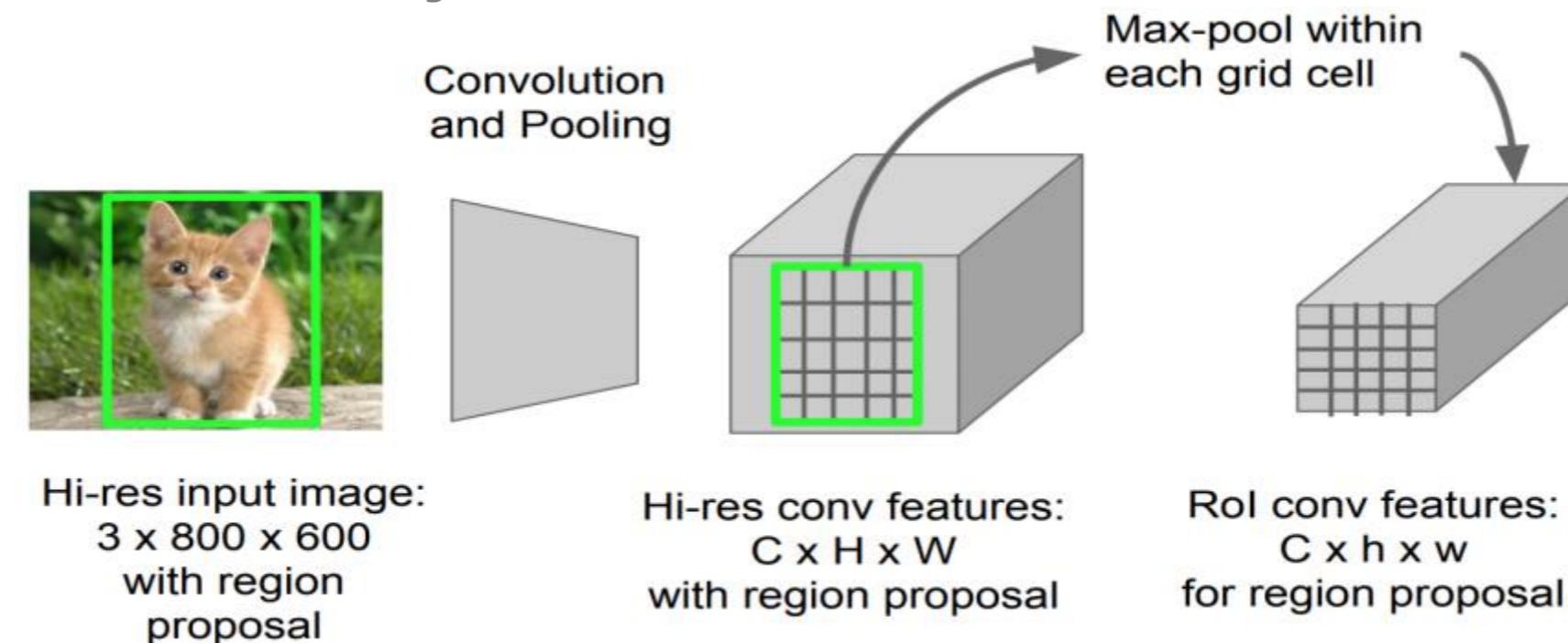
Fast R-CNN

**пропускаем через CNN-сеть изображение целиком
(а не каждый регион в отдельности, как раньше)**

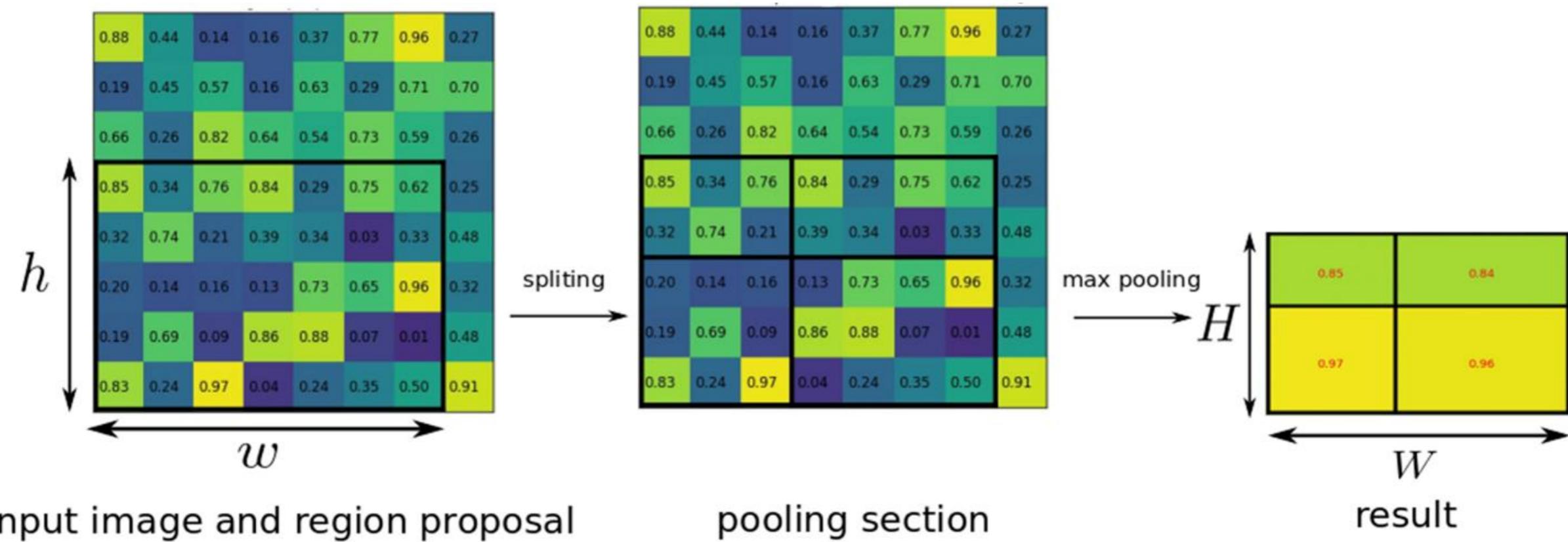
а регионы накладываются на полученную карту признаков

признаки из разных регионов приводятся в одну размерность с помощью ROI Pooling

$H \times W \rightarrow$ пулинг по сетке $H/7 \times W/7 \rightarrow$ сетка 7×7



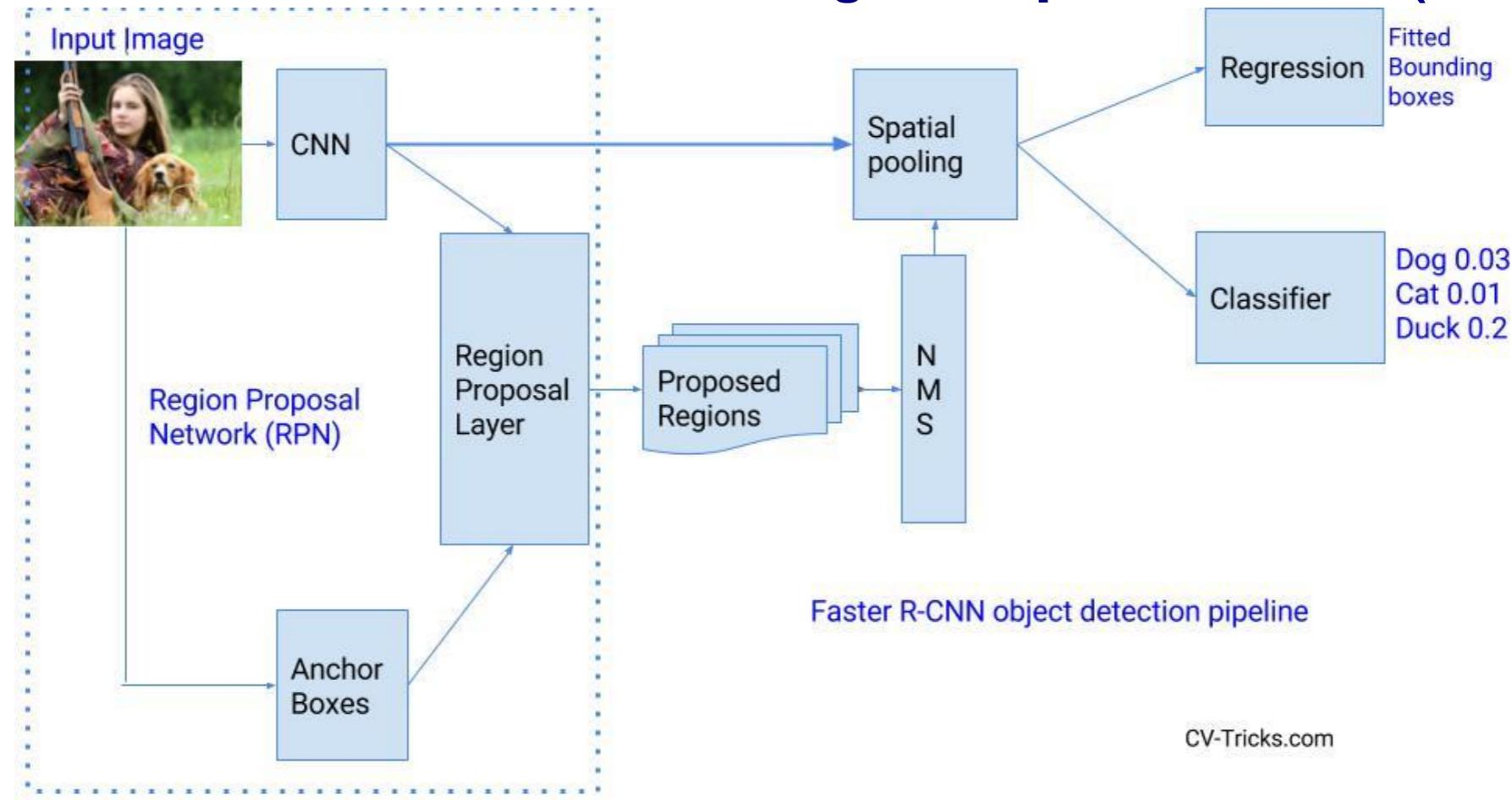
Fast R-CNN: ROI Pooling Layer



~ SPP-слой в SPPNet с одной пирамидой

+ небольшая тонкость с обучением (ошибка = сумма ошибок классификации и регрессии)

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



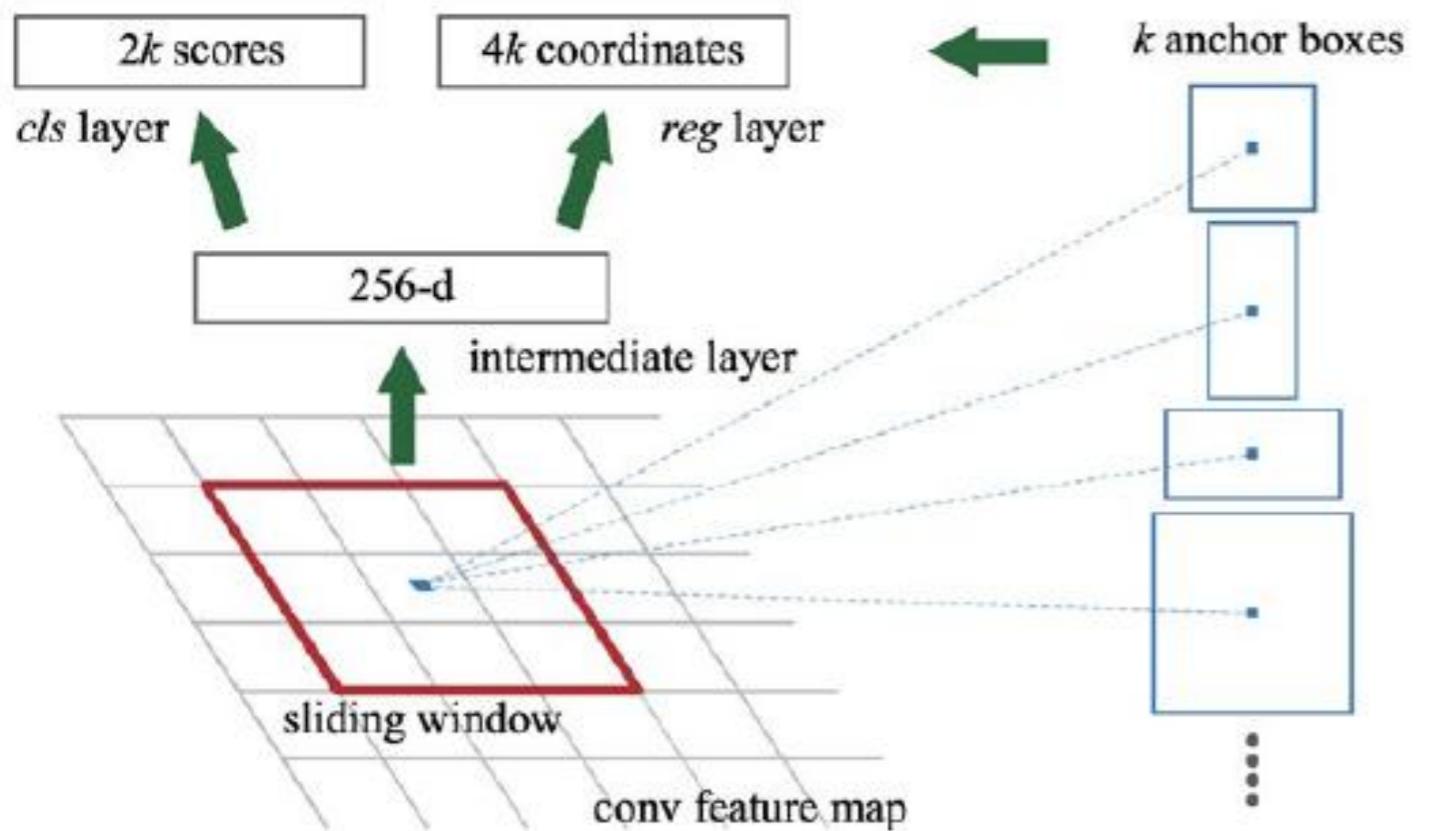
Полностью end2end: «Где смотреть» решает НС

Решили убрать не DL-генерацию регионов!

картина → свёрточные слои

Потом Region Proposal Network (RPN)

Faster R-CNN: Region Proposal Network (RPN)



хотим, чтобы генерация регионов – обучаемая процедура

проходимся окном 3×3 и генерируем k кандидатов на роль регионов

k anchor boxes – какие-то заданные шаблоны (размеры 128, 256, 512 и соотношения 1:1, 1:2, 2:1)

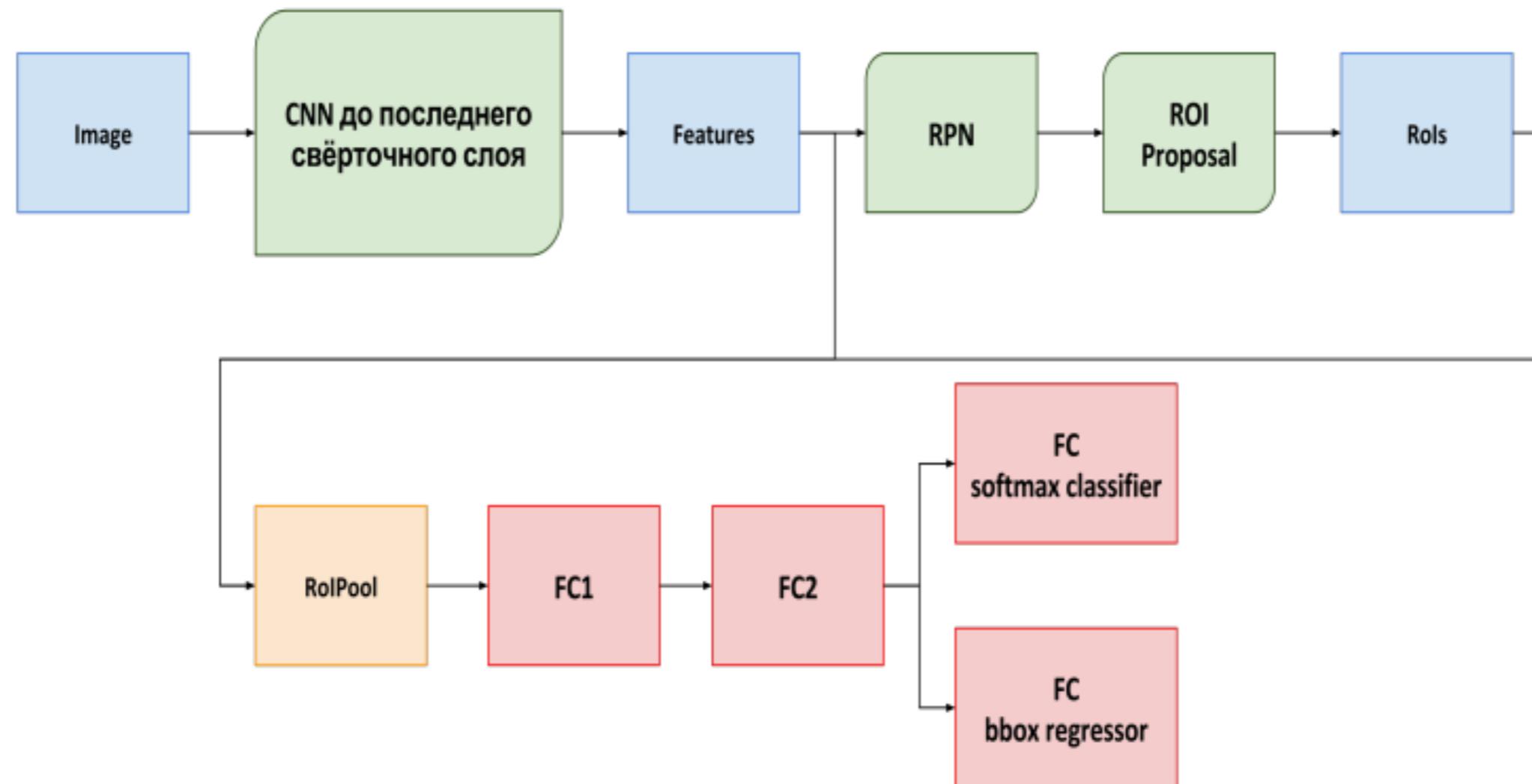
каждая точка – регион

Регионы с большой вероятностью содержания объекта передаются дальше для уточнения региона и детектирования

2k scores – 2 = объект или нет

Для $W \times H$ -карты признаков получается $W \cdot H \cdot k$ регионов (anchors)

Faster R-CNN: Region Proposal Network (RPN)



Faster R-CNN: Bounding Box Regression

термин для уточнения региона или его генерации

$$\sum_{i \text{ in } I} [(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + (\sqrt{w_i} - \sqrt{\hat{w}_i})^2 + (\sqrt{h_i} - \sqrt{\hat{h}_i})^2]$$

Faster R-CNN: Обучение

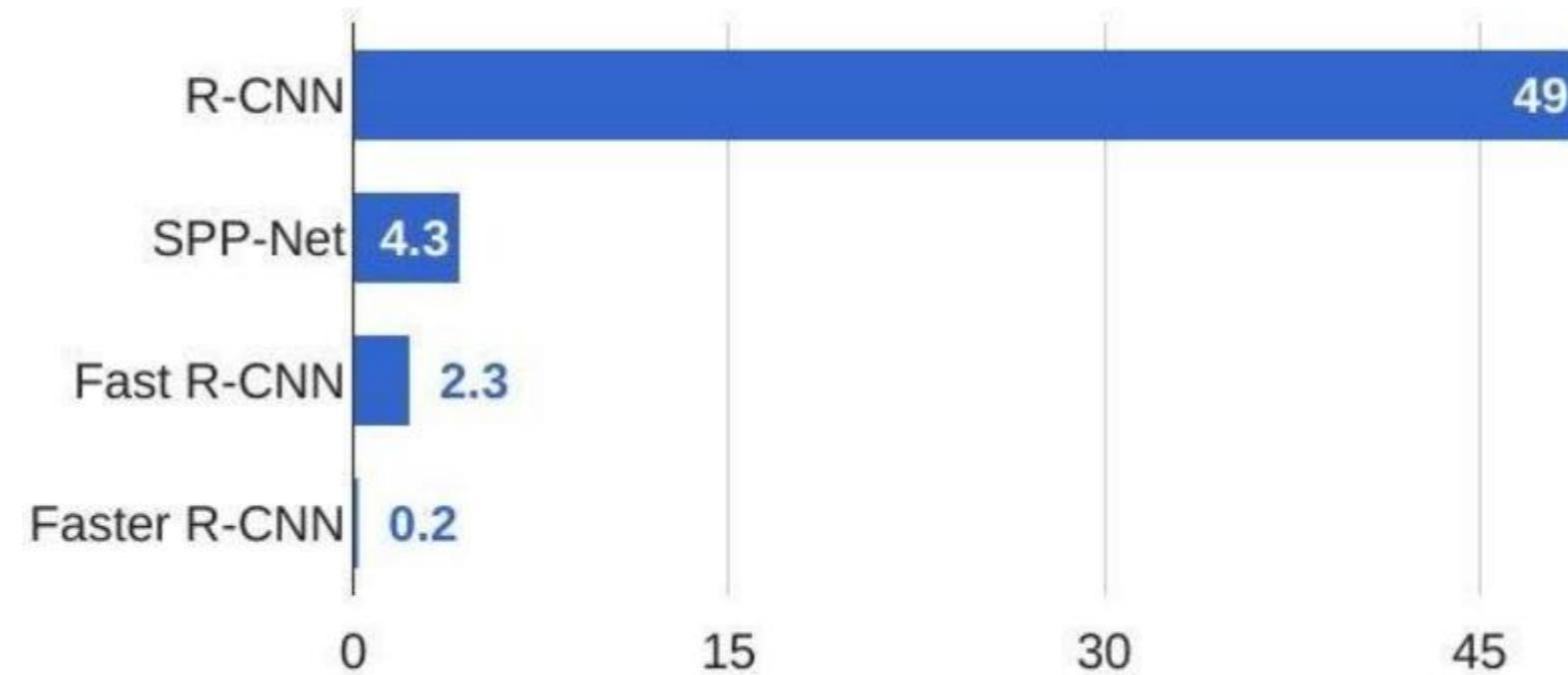
Обучение довольно нетривиальное

(сначала RPN-часть, потом Fast-RCNN-часть – можно итерационно)

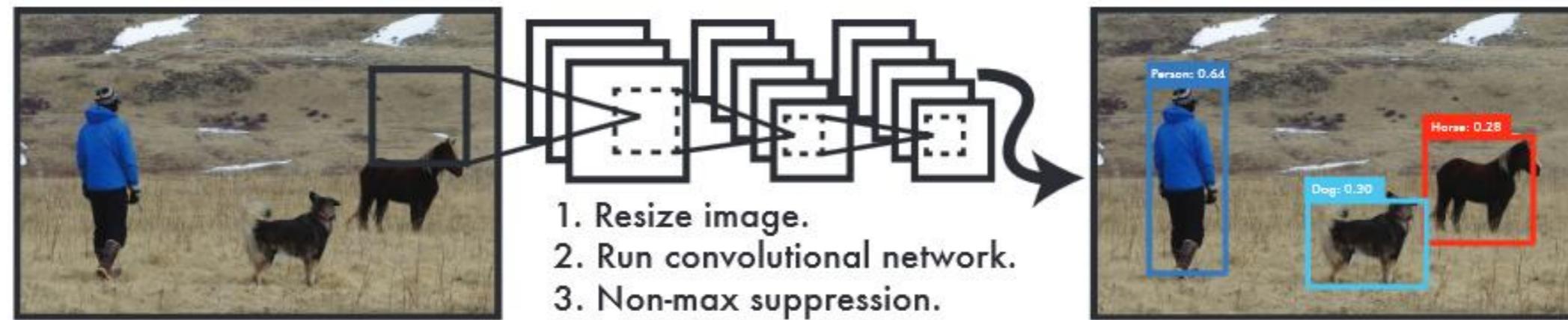
Faster R-CNN: Работа (inference)

- изображение → CNN + RPN
- оставляем 6000 перспективных регионов
- удаляем очень узкие и выходящие за изображения
- NMS с большим порогом IoU=0.9
- регионы → RoI pooling + регрессия + классификация
- NMS по классам

Скорость *-R-CNN сетей



YOLO: You only Look Once



- **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

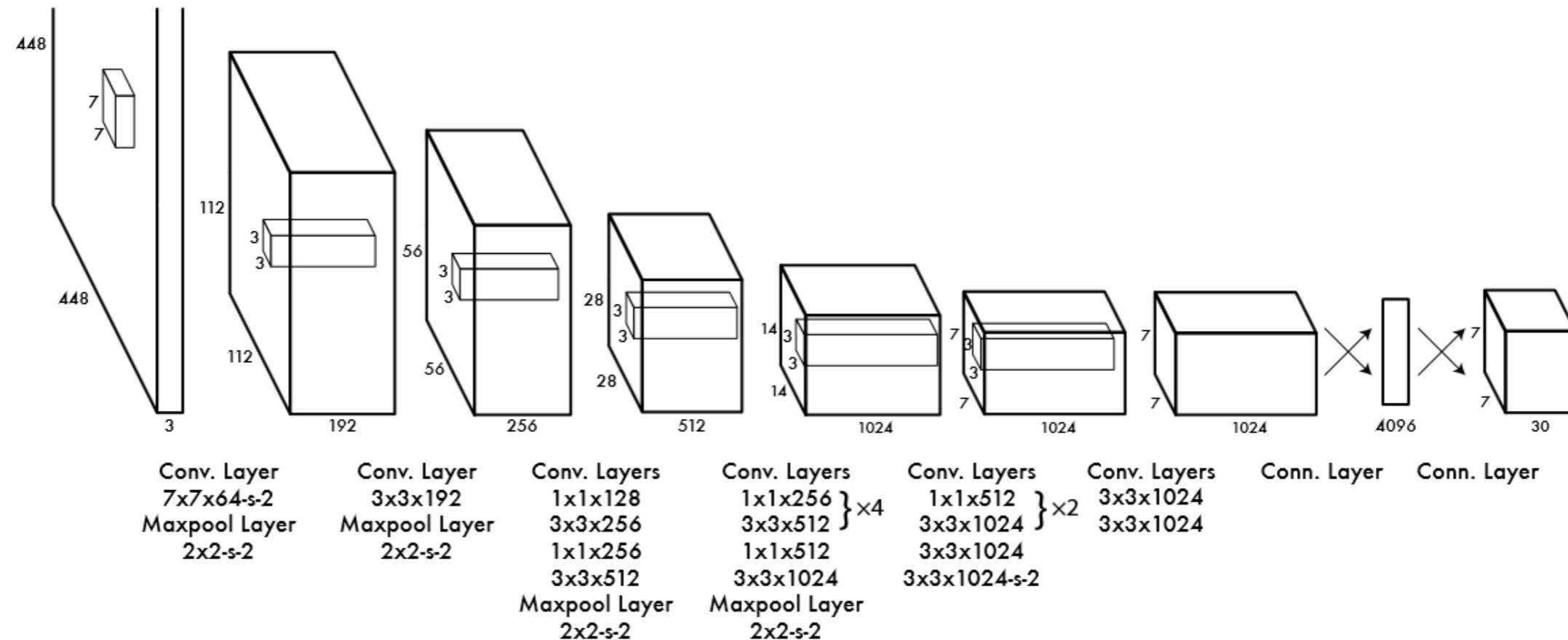


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224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

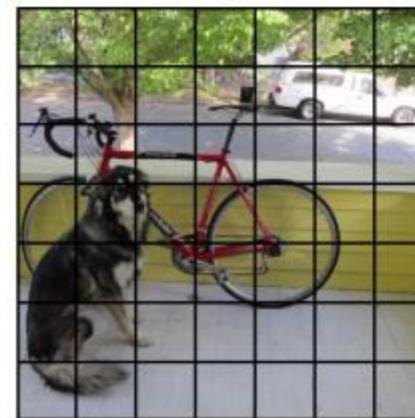
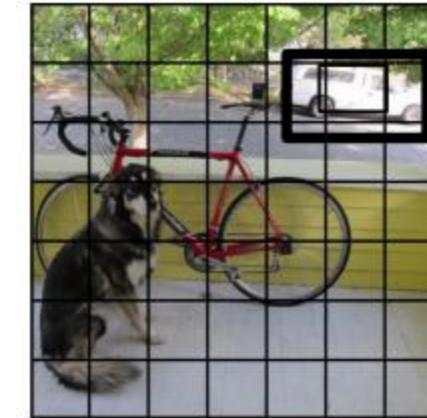
Вход: Изображение

Выход: класс (вероятности) + координаты рамки

Классика: свёртки → полно связные слои

YOLO

Split the image into a grid

 $P(\text{Object})$  $P(\text{Class} \mid \text{Object})$

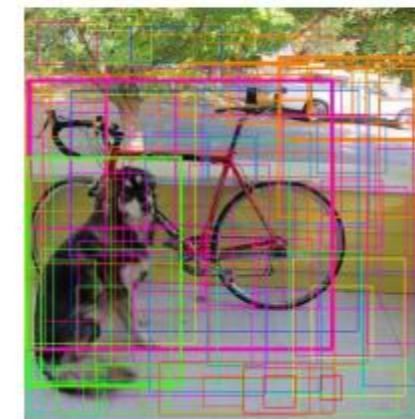
Bicycle

Dog

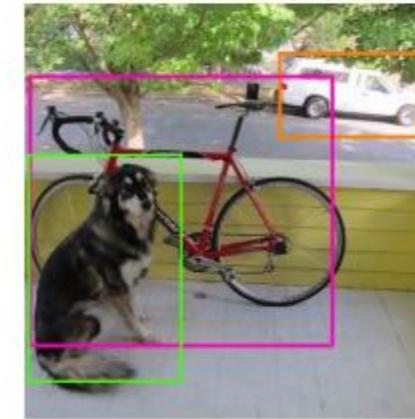
Car

Dining
Table

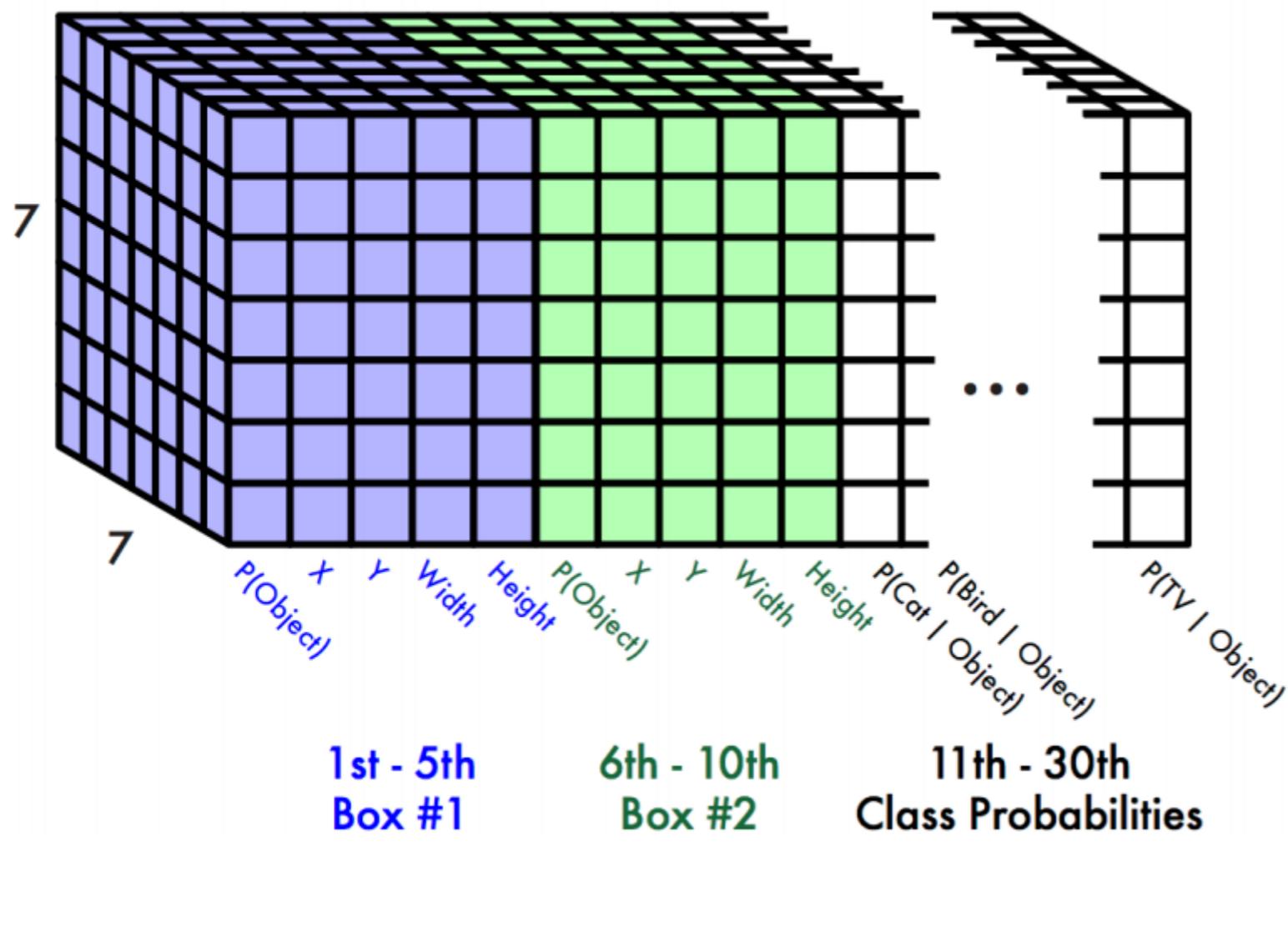
Combine the box and class predictions



Finally do NMS and threshold detections

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

YOLO: выход модели



В тензоре $7 \times 7 \times 30$ закодированы все регионы оценки за классы

7×7 – это сетка;)

$30 = 5 + 5 +$
(почему-то 2 региона для каждой ячейки)

+ 20 (# классов ~ Pascal VOC)

$5 = |(x, y, w, h, c)|$
 x, y – координаты в центре соотв. ячейки

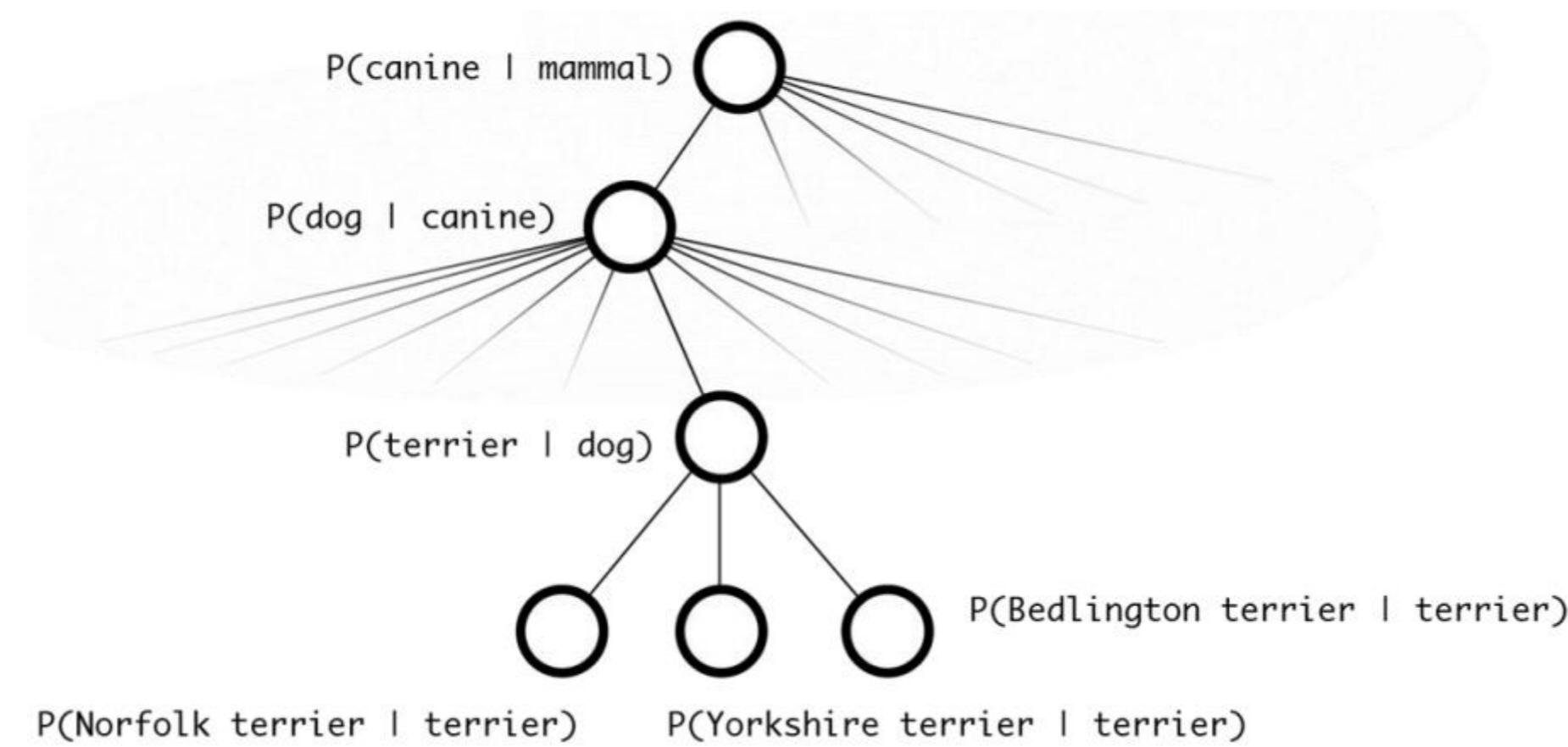
c – уверенность, что регион правильный

YOLO

- **Предположение: в каждой ячейке центры максимум 2x классов**
- **Первые 20 свёрточных 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**

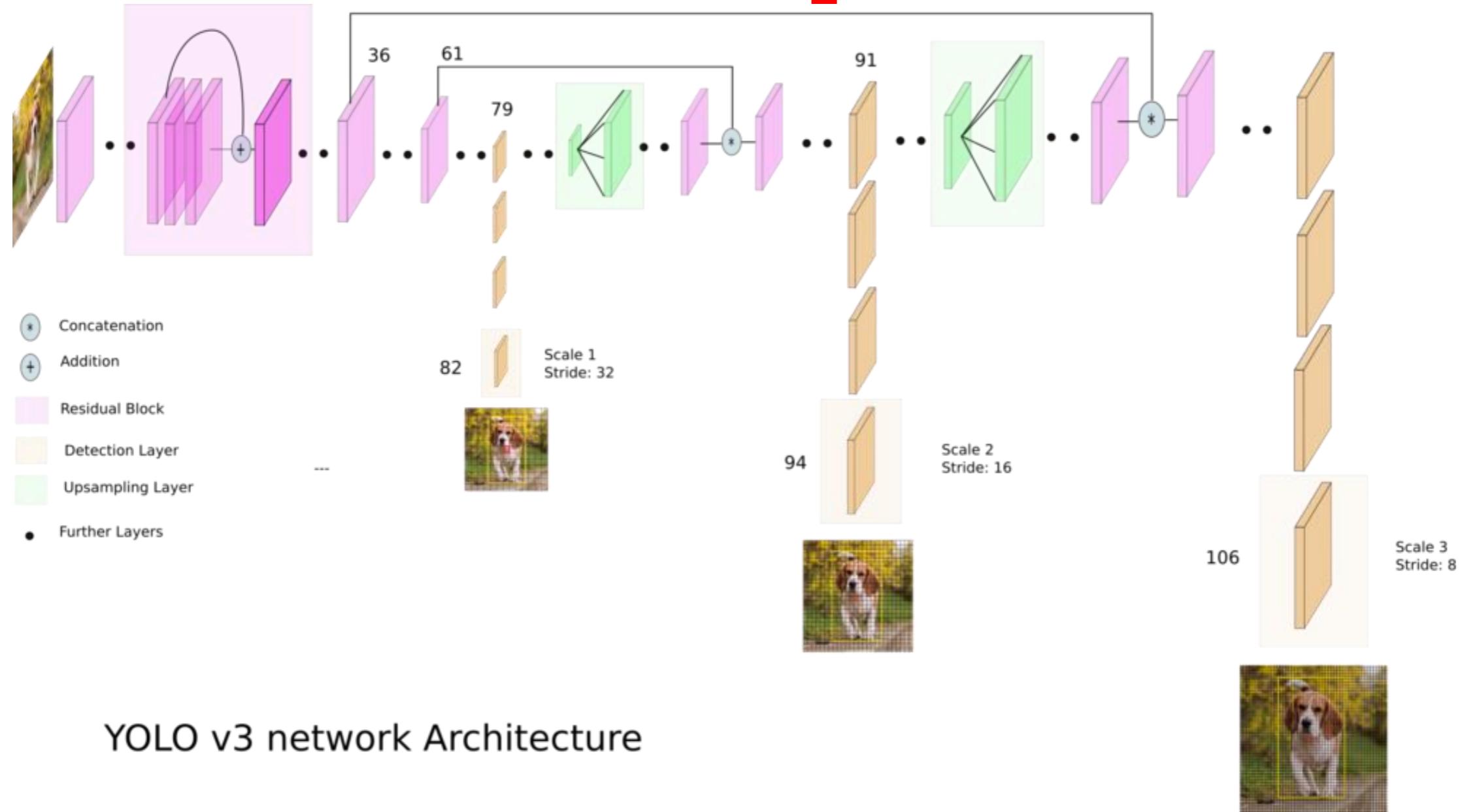
J. Redmon, S. Divvala, R. Girshick, A. Farhadi «You Only Look Once: Unified, Real-Time Object Detection» <https://arxiv.org/abs/1506.02640>

YOLO: иерархия в классах



Redmon and Farhadi «YOLO9000: Better, Faster, Stronger», CVPR 2017

YOLOv3



на выходе есть три слоя – для обнаружения объектов разного размера

Single Shot MultiBox Detector (SSD)

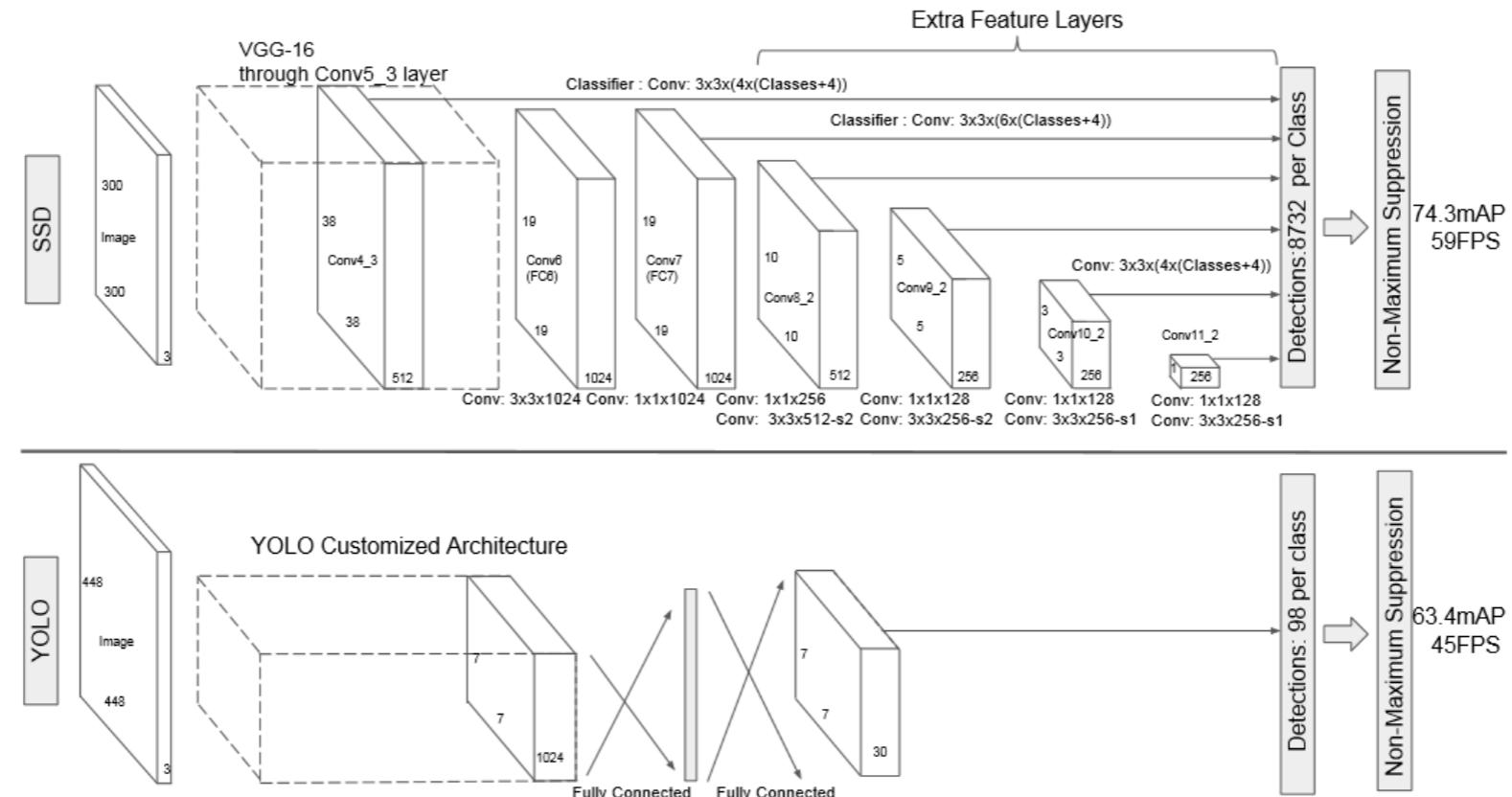


Fig. 2: A comparison between two single shot detection models: SSD and YOLO [5]. Our SSD model adds several feature layers to the end of a base network, which predict the offsets to default boxes of different scales and aspect ratios and their associated confidences. SSD with a 300×300 input size significantly 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)

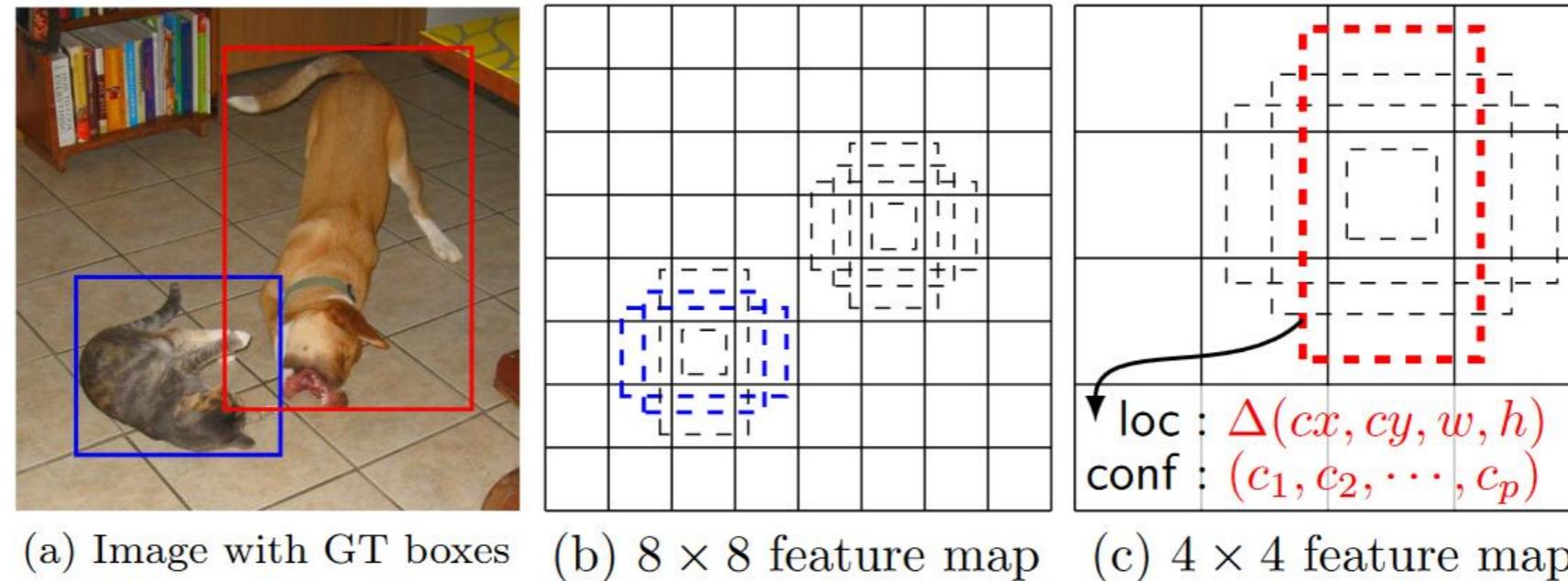


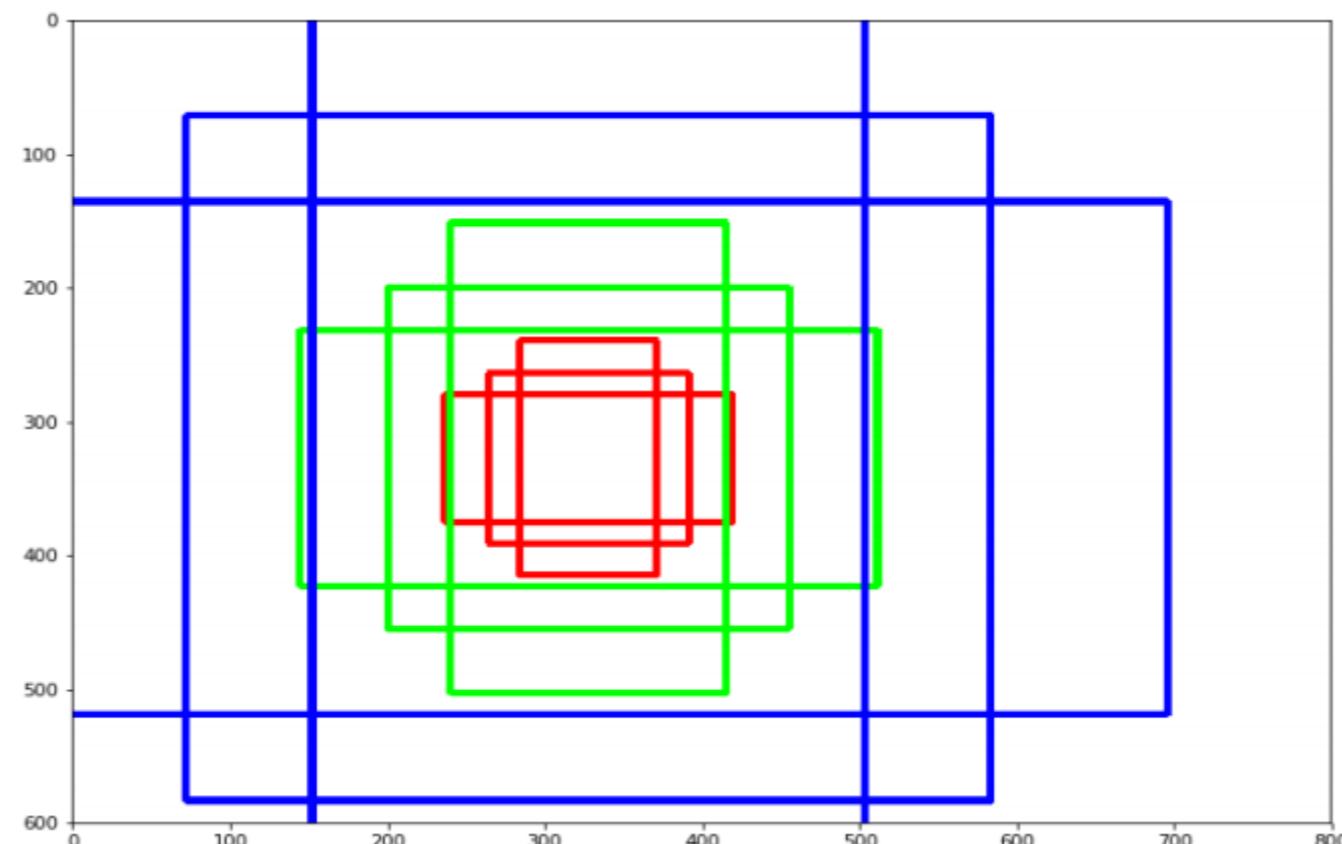
Fig. 1: **SSD framework.** (a) SSD only needs an input image and ground truth boxes for each object during training. In a convolutional fashion, we evaluate a small set (e.g. 4) of default boxes of different aspect ratios at each location in several feature maps with different scales (e.g. 8×8 and 4×4 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, \dots, 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 $1:1, 1:2, 2:1$)



Single Shot MultiBox Detector (SSD)

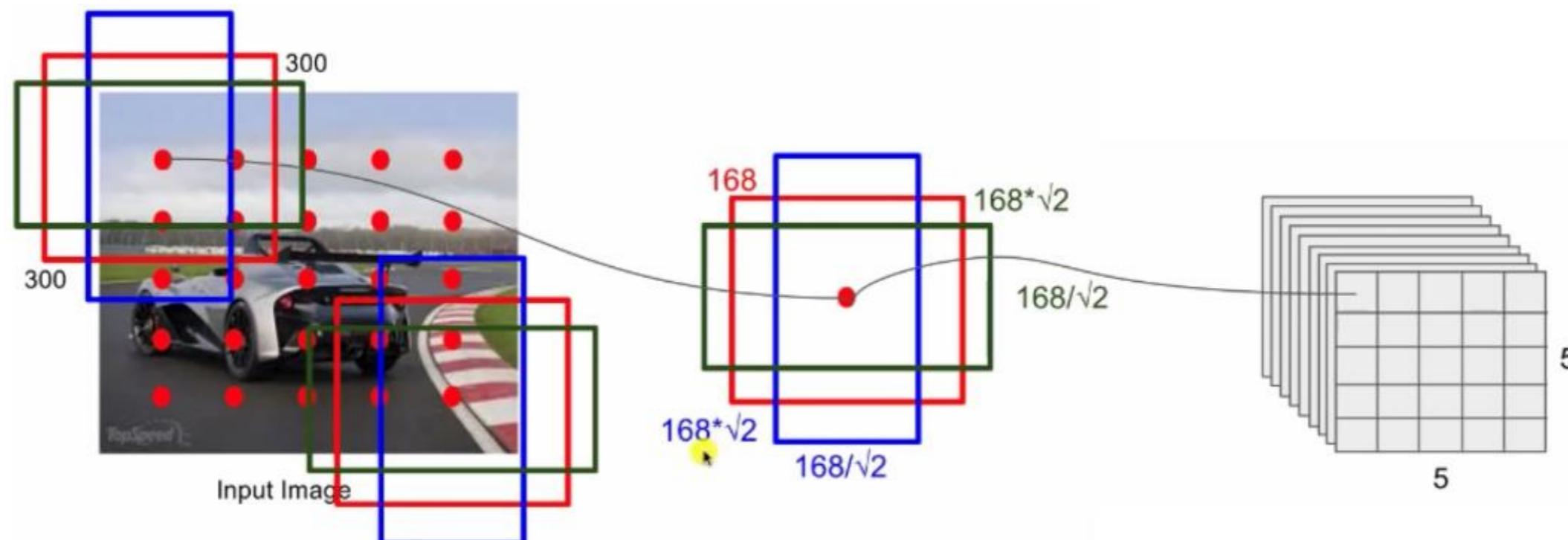
Вход 300×300 изображение

Нужны истинные рамки (на обучении)

Задаём несколько (на картинке 4) «default boxes» – для каждого

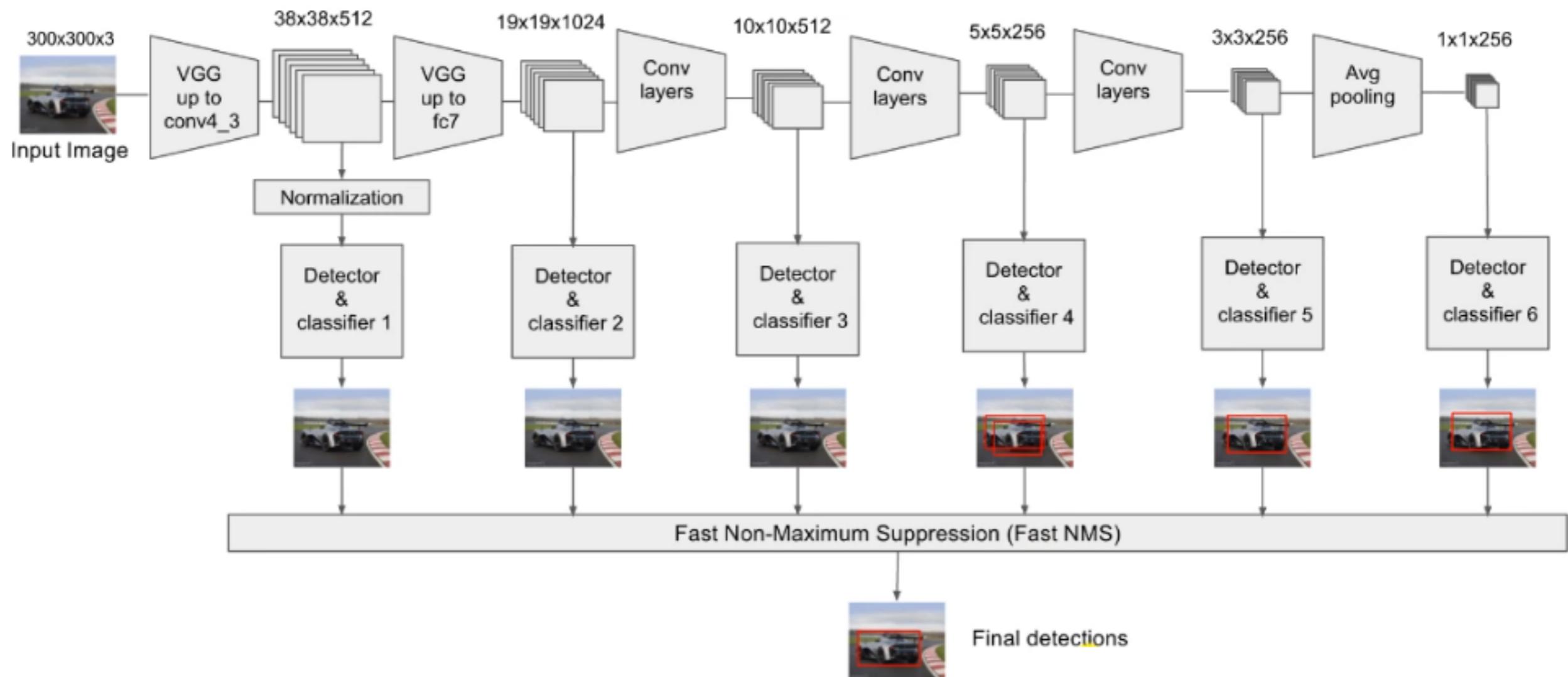
- коррекция положения

- оценки принадлежности к классам



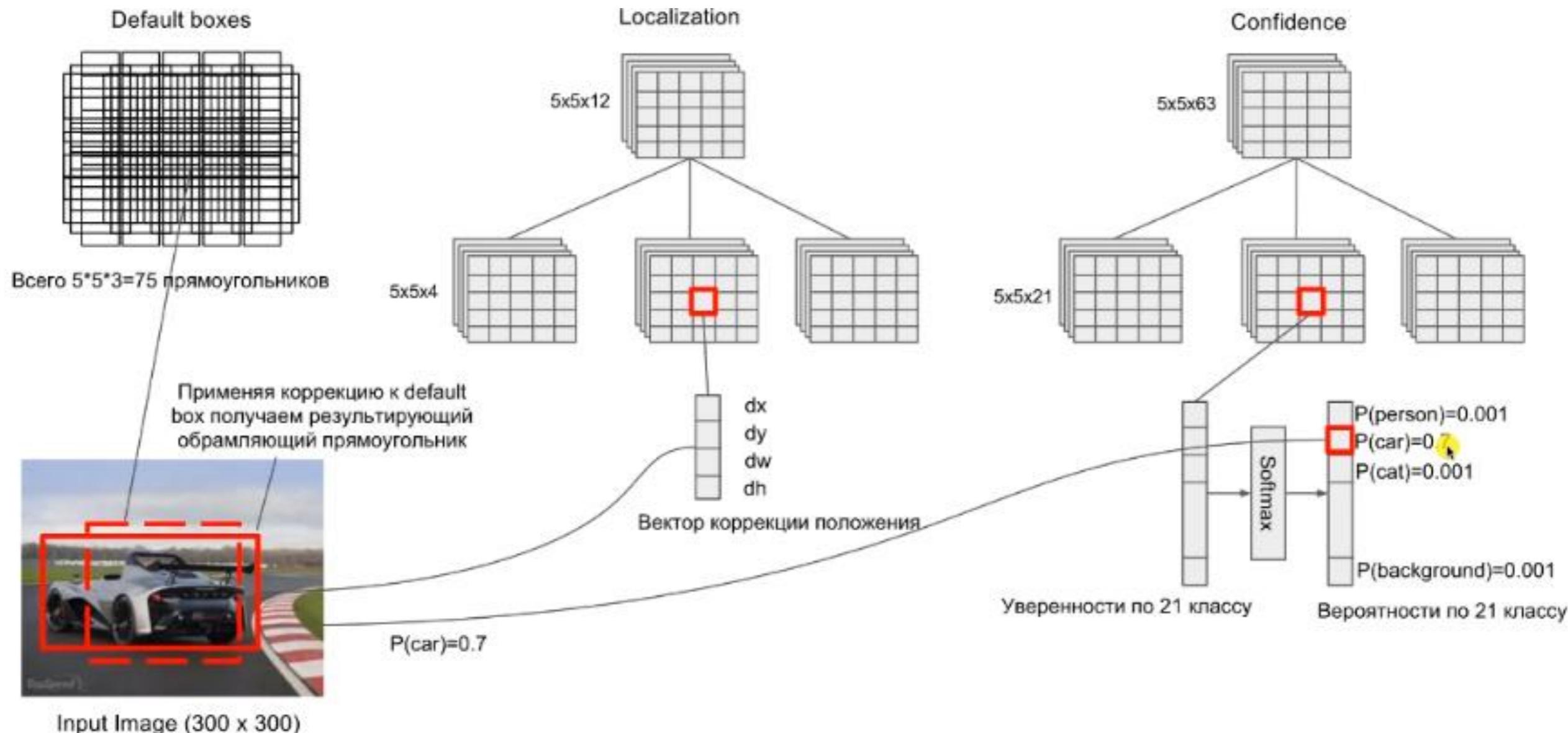
<https://deepsystems.ai>

Single Shot MultiBox Detector (SSD)



Много регионов «default boxes» на разных масштабах
<https://deepsystems.ai>

Single Shot MultiBox Detector (SSD)



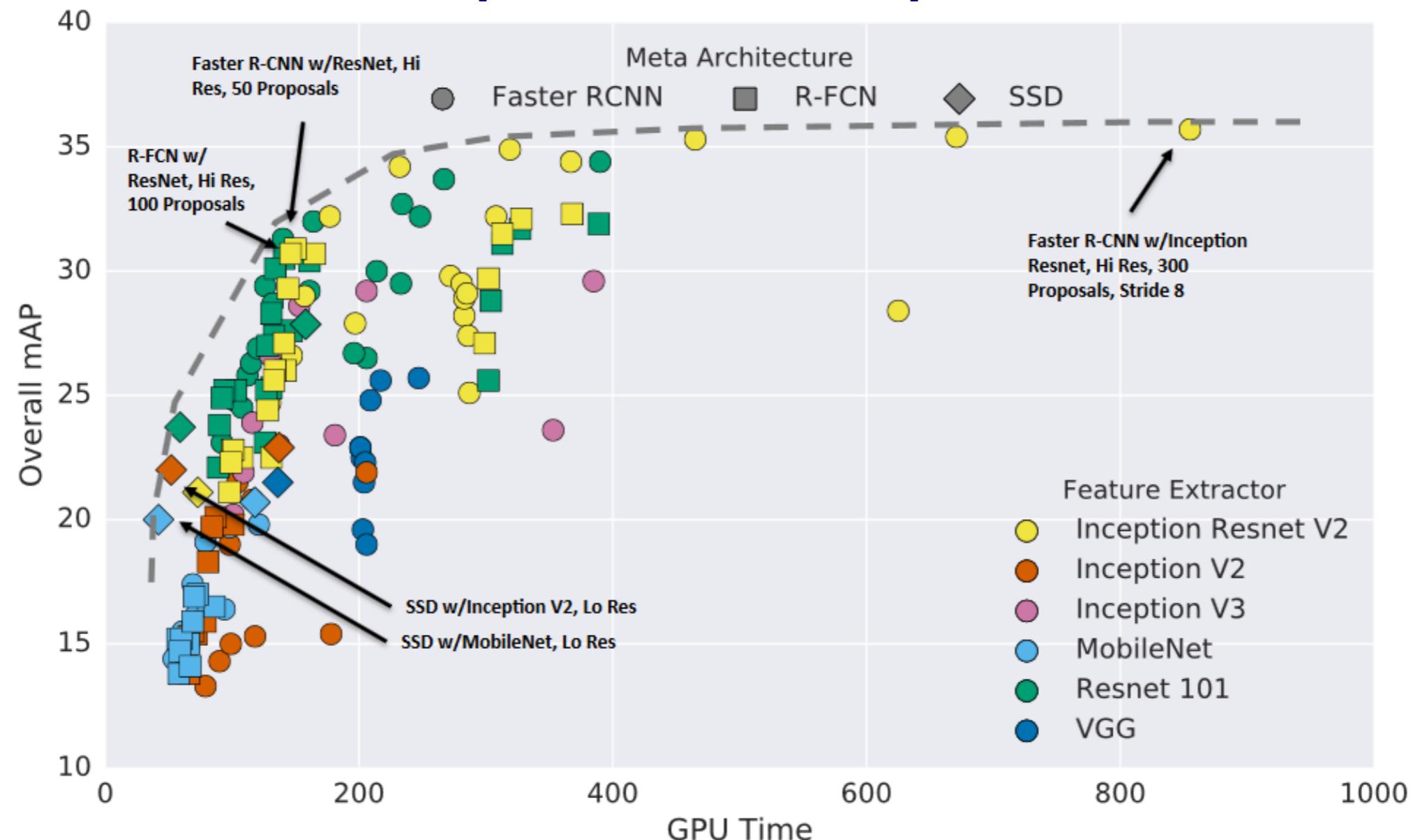
<https://deepsystems.ai>

Single Shot MultiBox Detector (SSD)

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

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

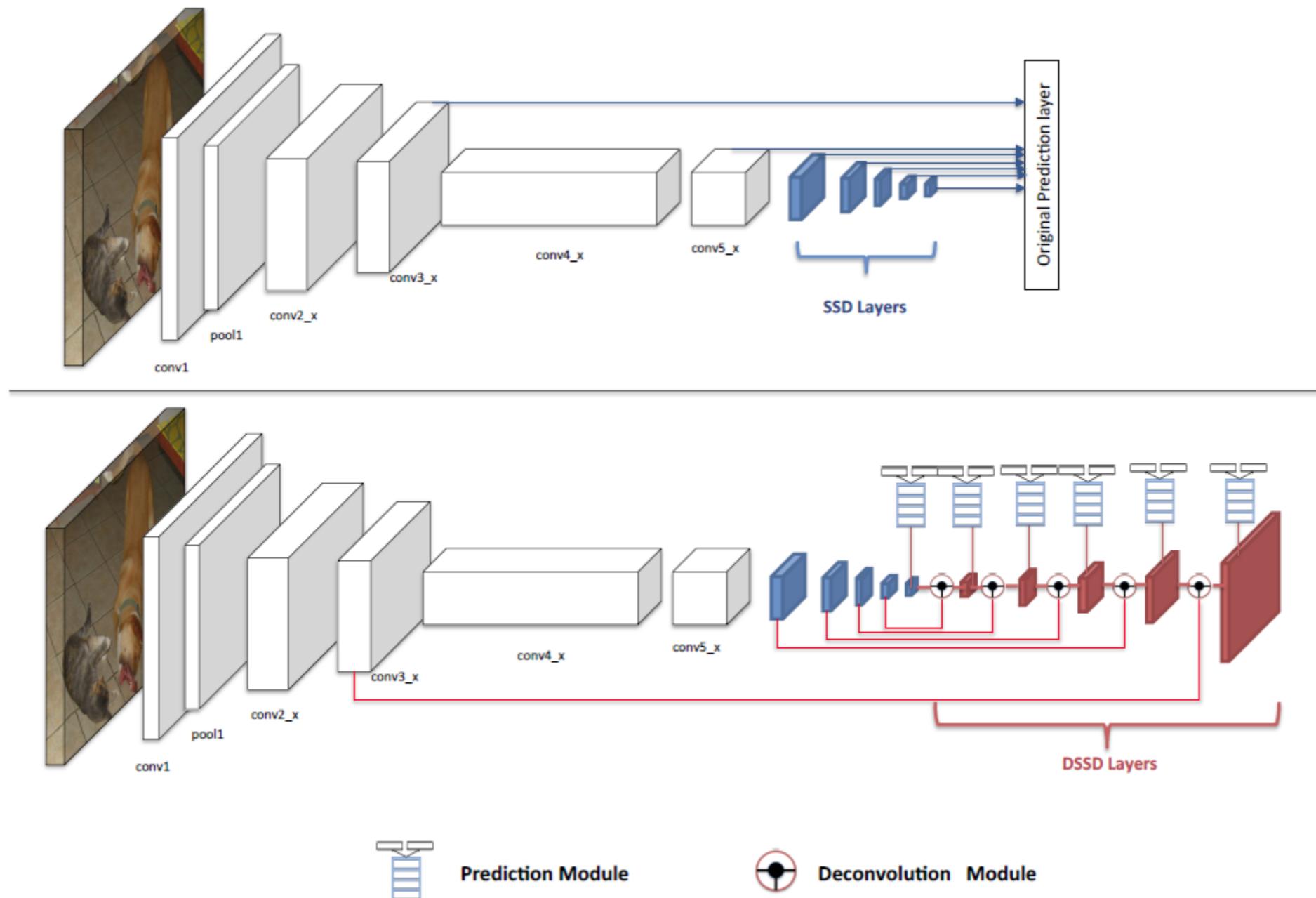
Детектирование объектов: сравнение



Speed/accuracy trade-offs for modern convolutional object detectors

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

DSSD: Deconvolutional Single Shot Detector (Object Detection)



DSSD: Deconvolutional Single Shot Detector (Object Detection)

На базе VGGNet / ResNet – белым на рисунке

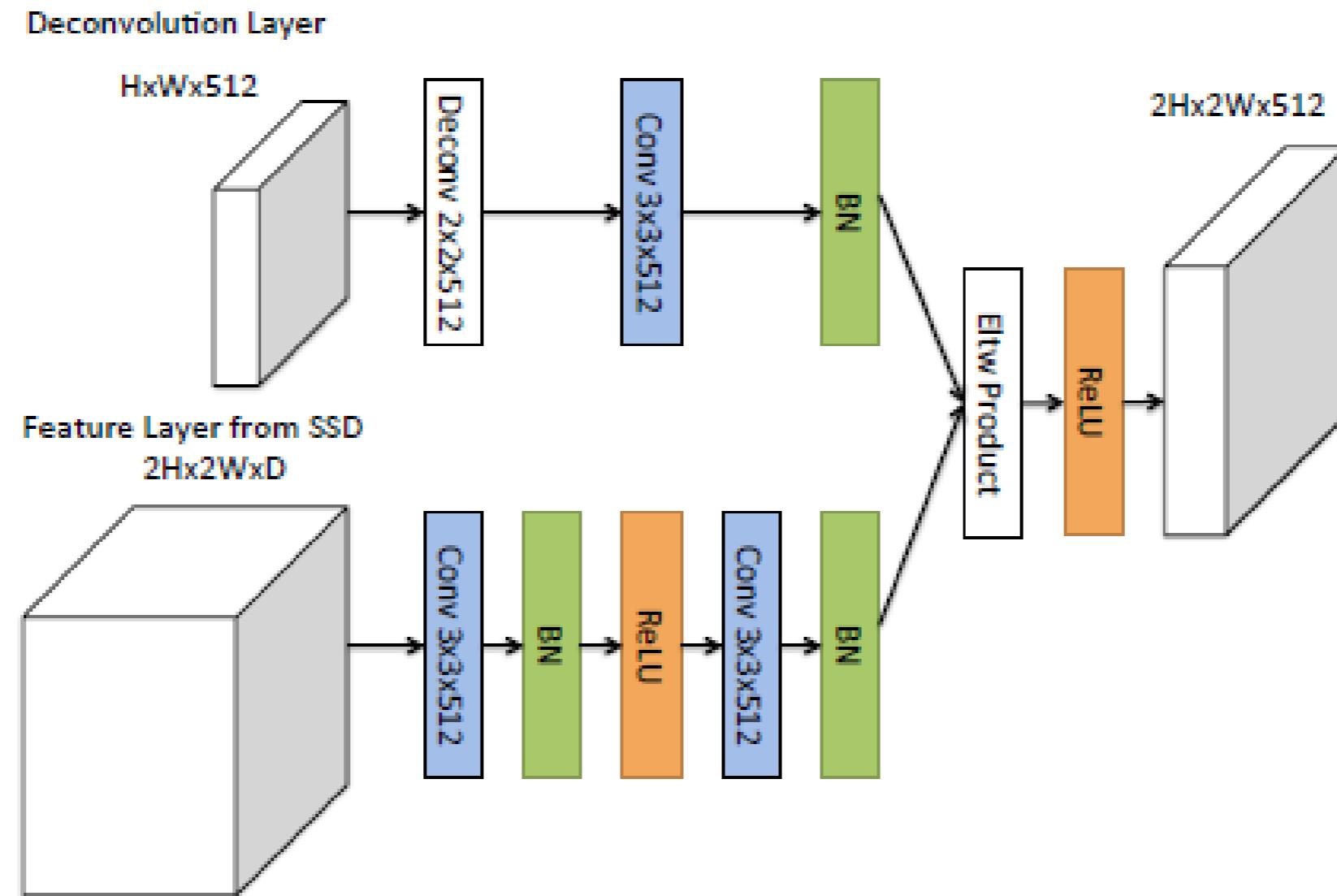
Свёртки из SSD – синим

Deconvolution Module – ниже

Потом вернёмся к этой архитектуре

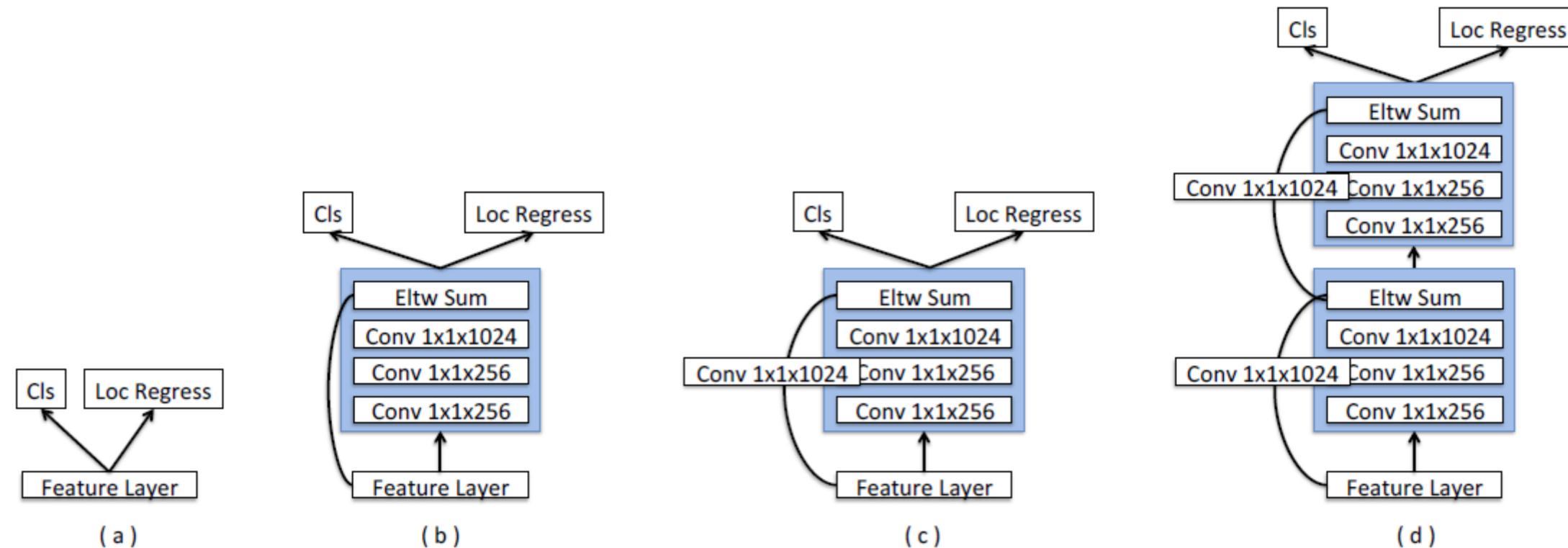
<https://towardsdatascience.com/review-dssd-deconvolutional-single-shot-detector-object-detection-d4821a2bbeb5>

DSSD: Deconvolution Module



Eltw Product – поэлементное умножение

DSSD: Prediction Module



Были попробованы разные варианты

Детектирование объектов: Feature Pyramid Networks (FPN)

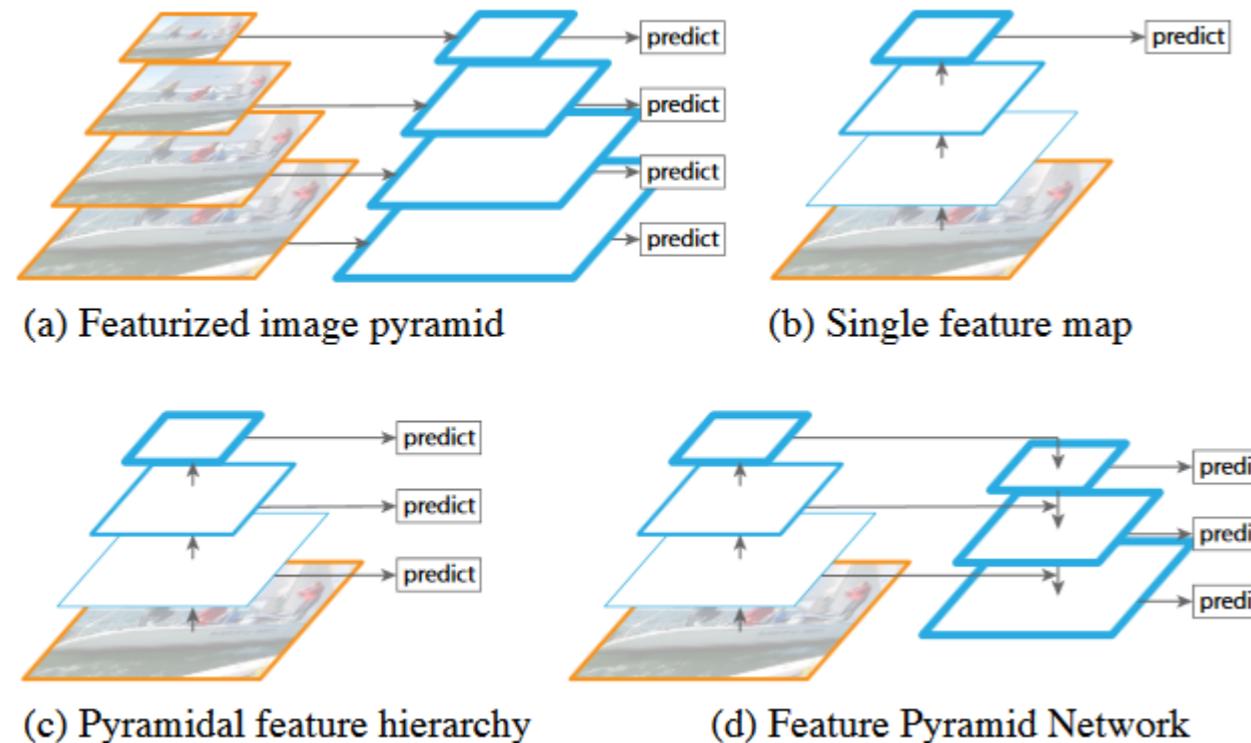


Figure 1. (a) Using an image pyramid to build a feature pyramid. Features are computed on each of the image scales independently, which is slow. (b) Recent detection systems have opted to use only single scale features for faster detection. (c) An alternative is to reuse the pyramidal feature hierarchy 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 indicated by blue outlines and thicker outlines denote semantically stronger features.

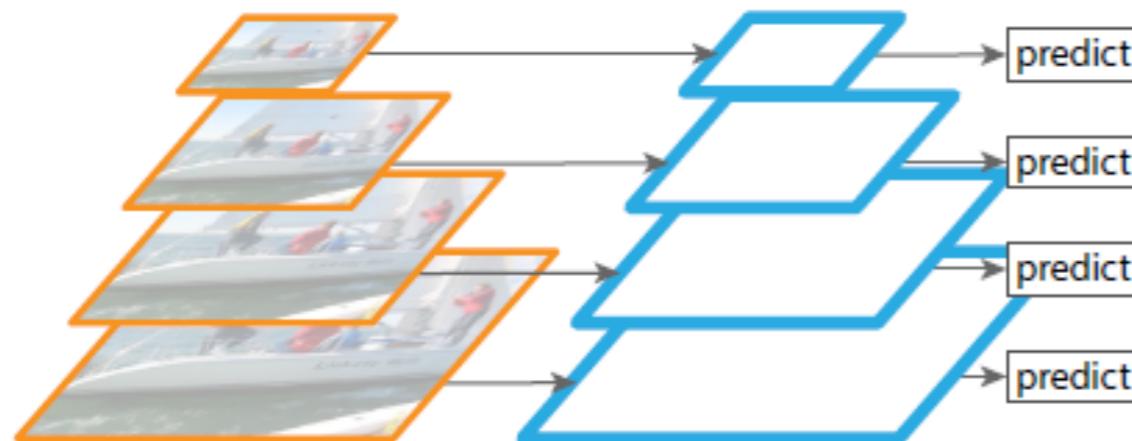
для инвариантности к масштабу делают «пирамиды изображений»

обычно используются только во время тестирования (т.к. долгое end2end-обучение)

но свёрточные сети получают «пирамиды признаков»

это всё – способы получения карт признаков. Можно использовать с любыми архитектурами НС! <https://arxiv.org/abs/1612.03144>

Разные архитектуры

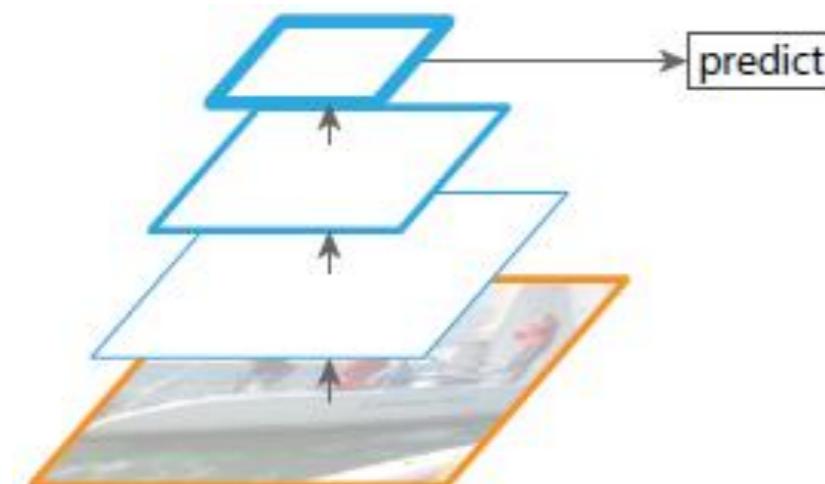


(a) Featurized image pyramid

Использовались ещё до эры DL

Точные

Медленные



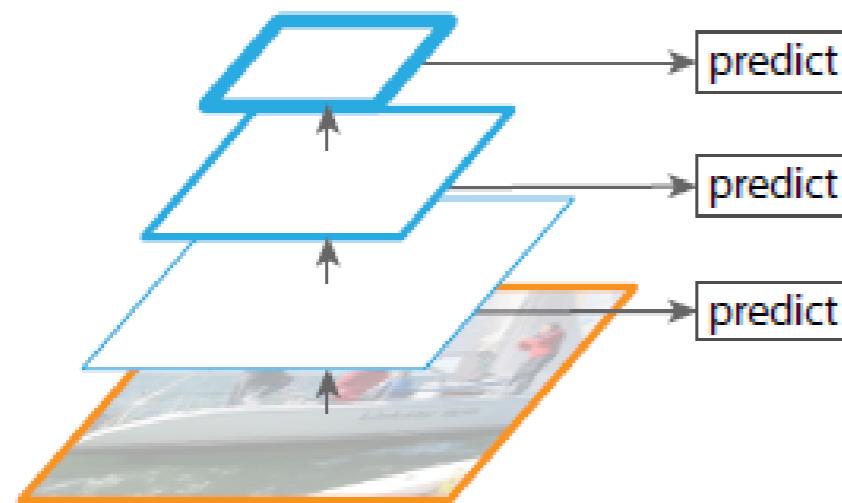
(b) Single feature map

Стандартное CNN-решение

Реализовано в YOLO

**Не учитывают низкоуровневую
информацию**

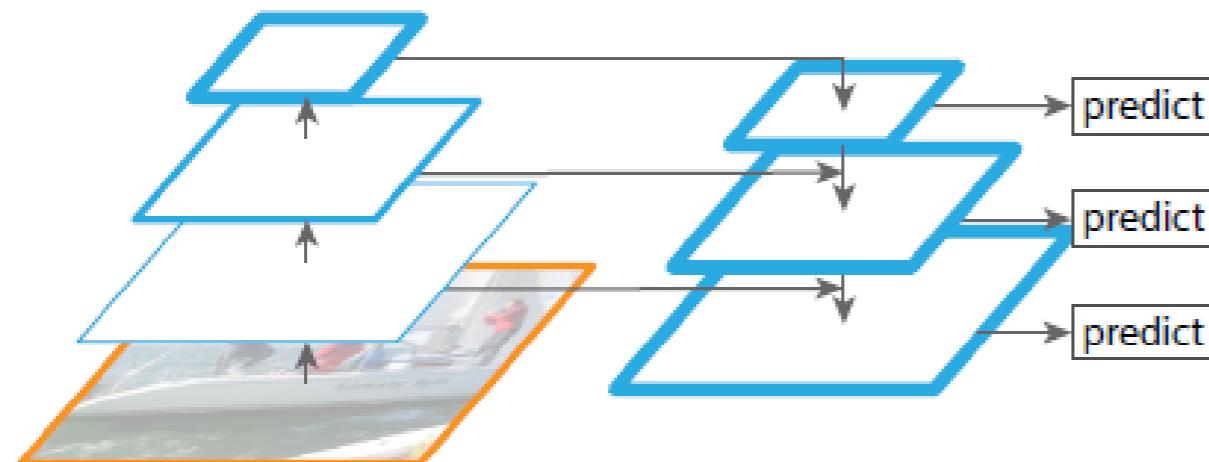
Разные архитектуры



(c) Pyramidal feature hierarchy

Реализовано в SSD

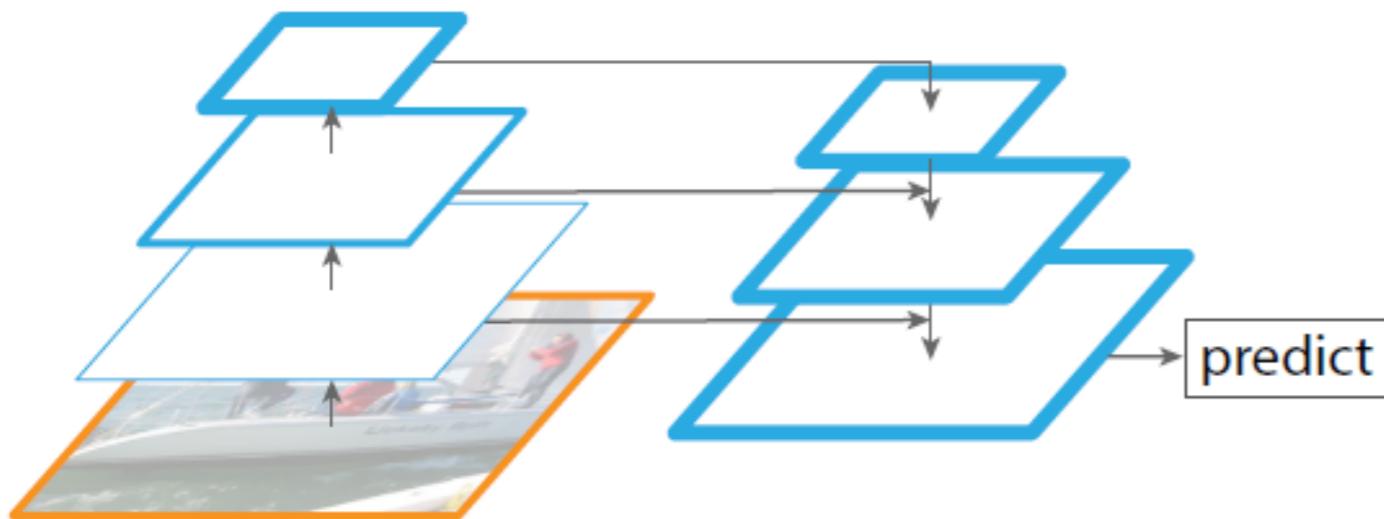
Не учитывают контекст на низком уровне



(d) Feature Pyramid Network

точные и быстрые

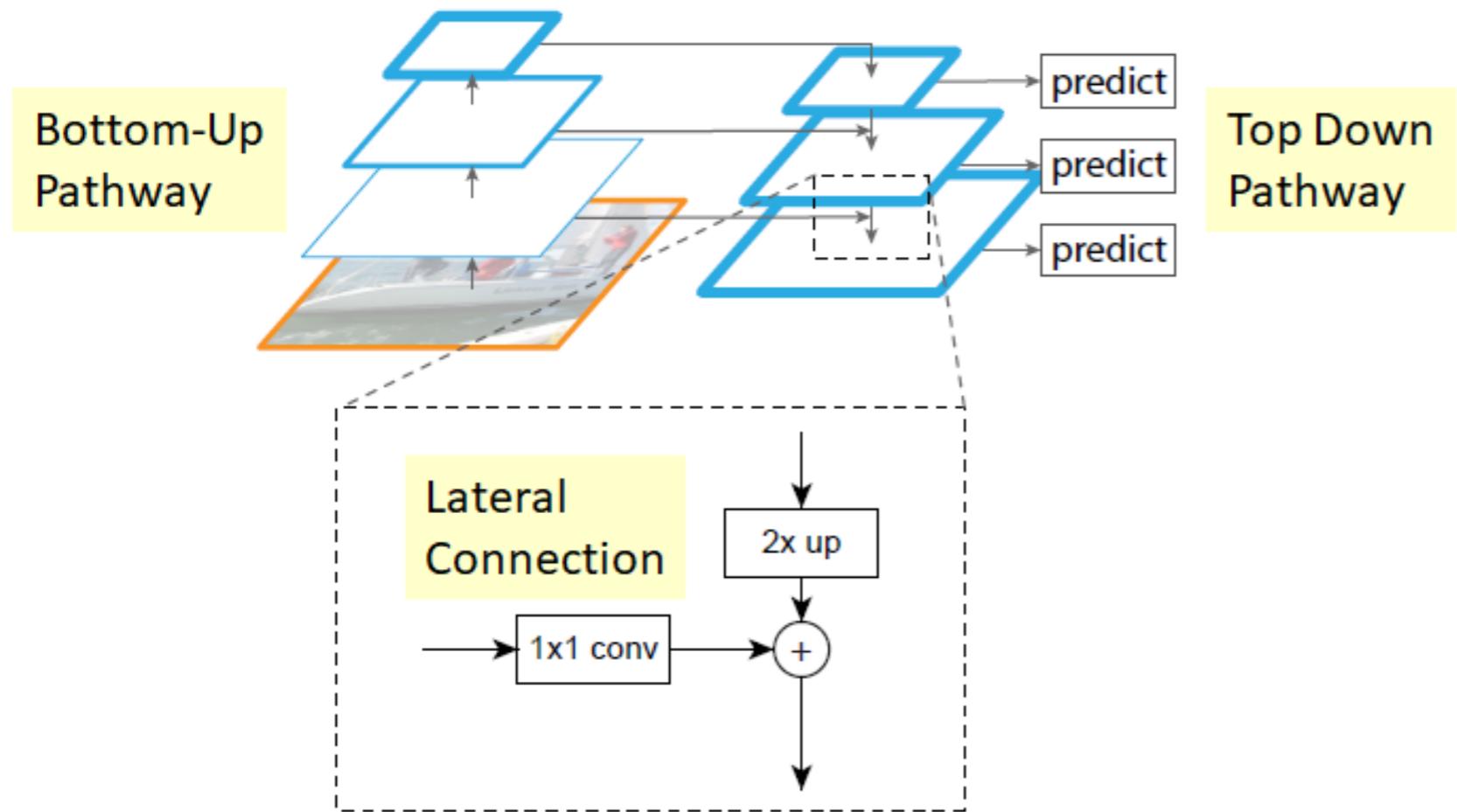
Разные архитектуры



Иногда применяется схожая архитектура

(e) Similar Structure with (d)

Feature Pyramid Networks (FPN)

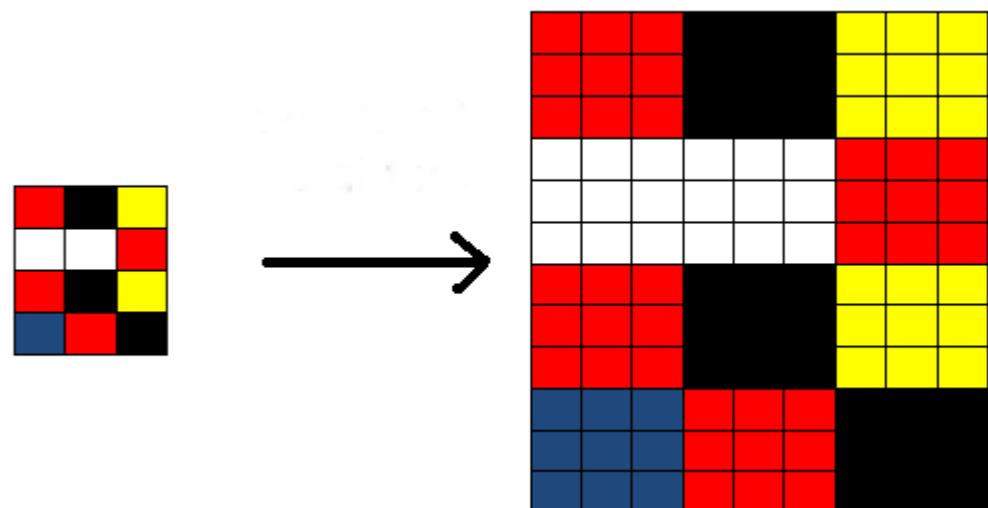


предсказания на разных уровнях

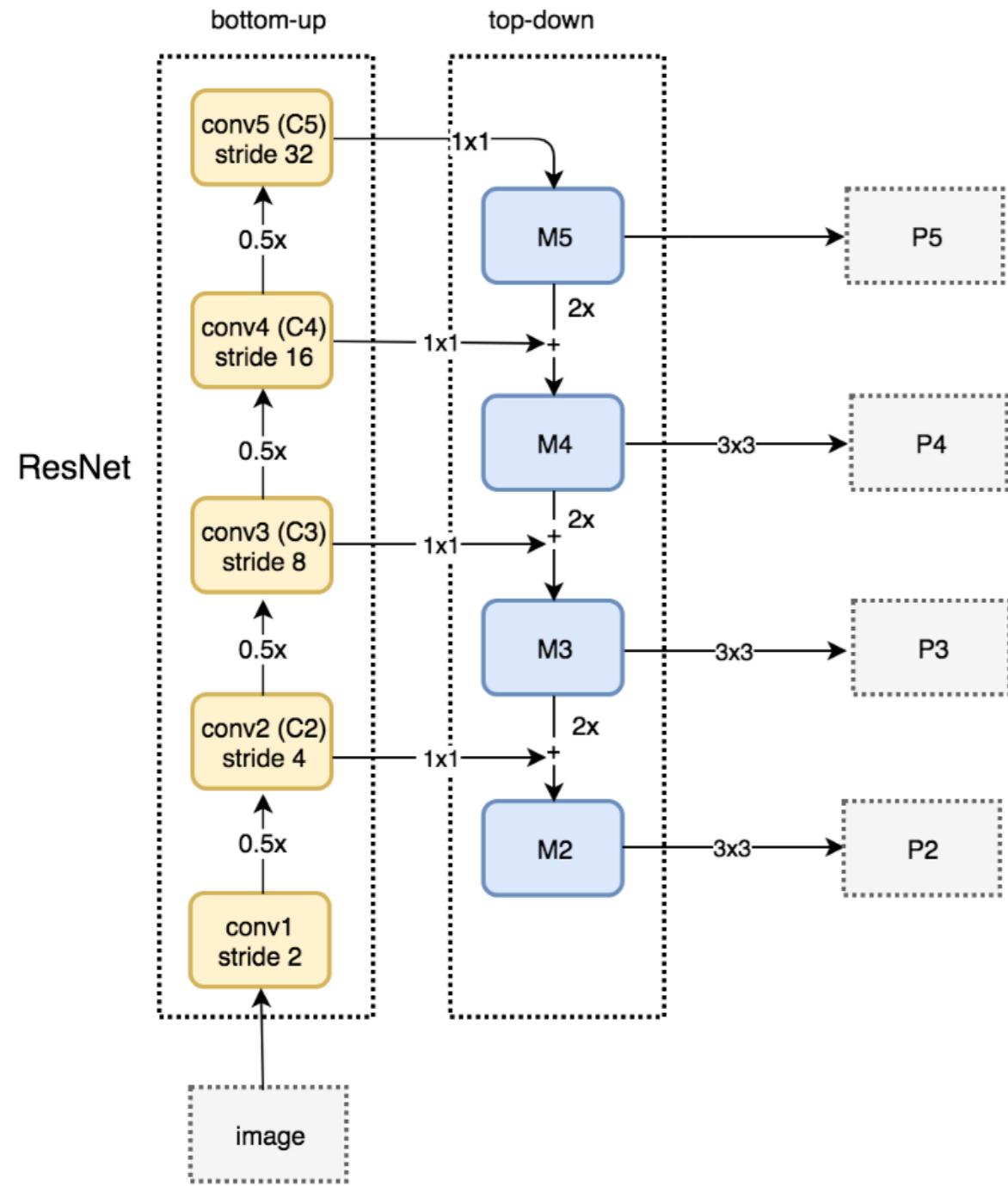
для одинаковости размера (256) свёртки 1×1

Top-down

**upsampling $\times 2$ с помощью
ближайшего соседа**



**lateral connection складывает
признаковые карты одинакового
пространственного размера**



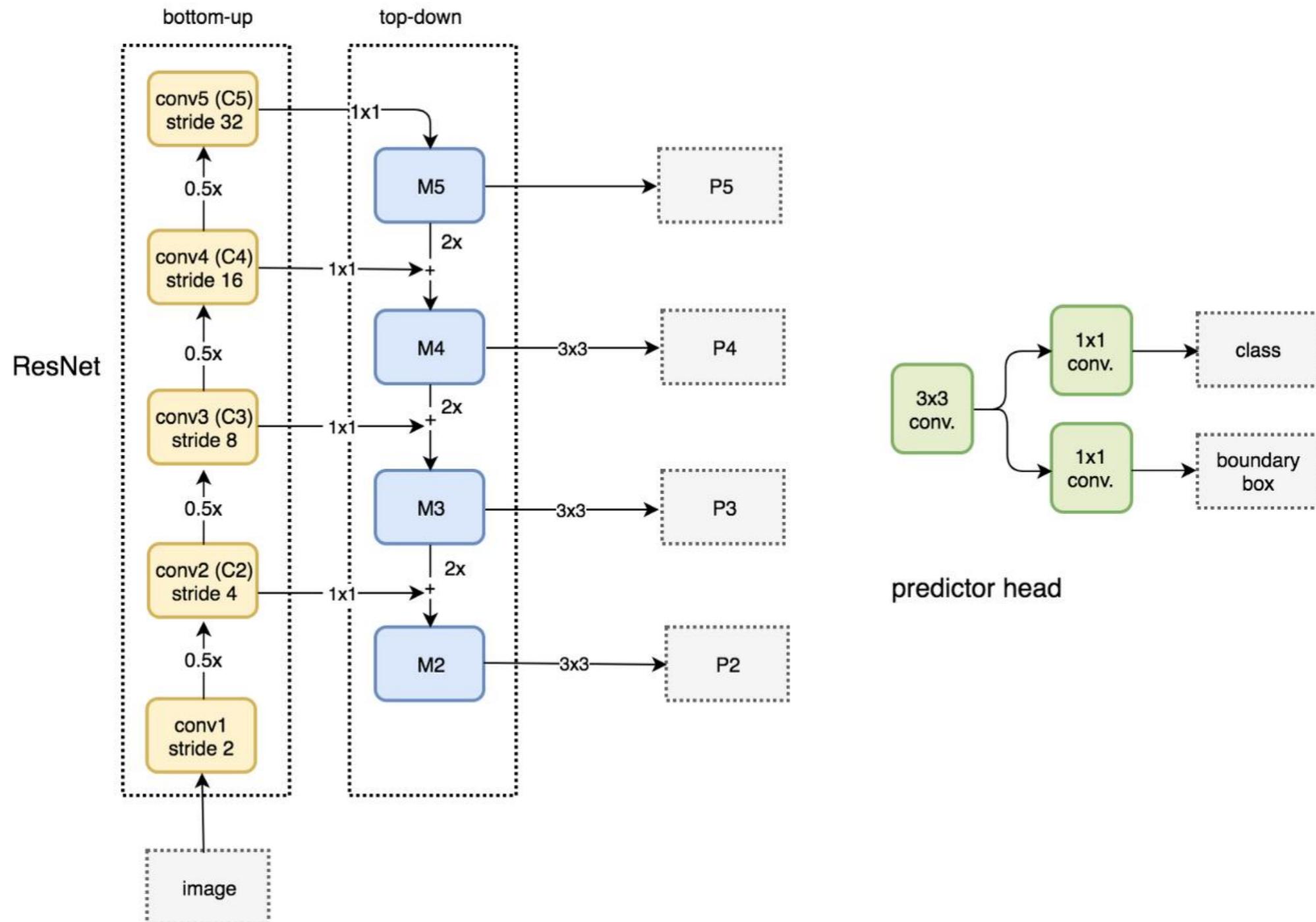
Bottom-up

Используется ResNet

На уровнях пространственное разрешение уменьшается в 2 раза

P1 нет из-за слишком большой пространственной размерности

Это не object-detector, а построение признаков



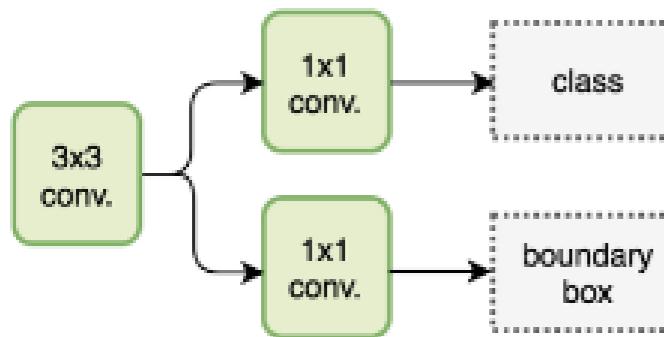
Feature Pyramid Networks (FPN)

**До этого момента – общий способ генерации карт признаков
подходит для любой сети**

Применено для

- **bounding box proposal generation в RPN**
- **object detection в Fast R-CNN**

FPN + RPN (bounding box proposal generation)



3×3-свёртки на каждой карте из lateral connection для получения финальной признаковой карты
1×1-свёртки для классификаторов и регрессоров
– это называется «RPN head»

На всех уровнях одинаковые (shared) классификаторы (есть или нет объект) / регрессоры!

Они действуют по отношению к набору anchors
На каждом уровне 1:2, 1:1, 2:1
по площади { 32^2 , 64^2 , 128^2 , 256^2 , 512^2 } пикселей на каждом уровне

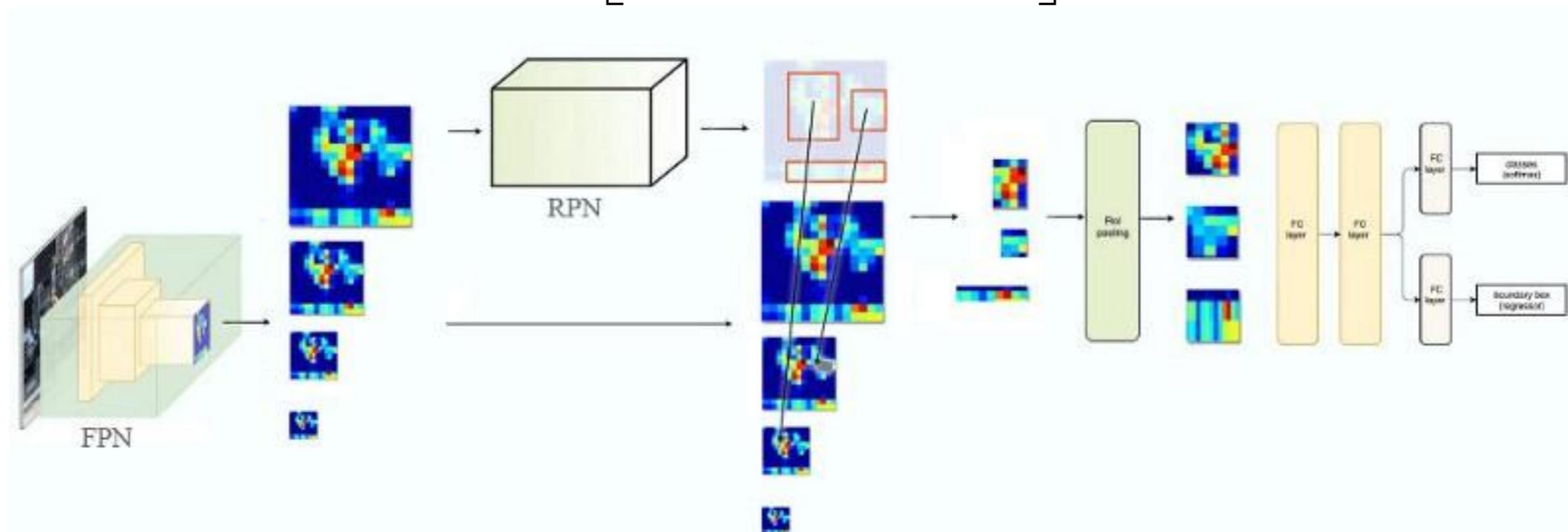
anchor позитивный, если IoU > 0.7
негативный, если IoU < 0.3

FPN + Fast R-CNN (object detection) используем Region-of-Interest (RoI) pooling из Fast R-CNN

но есть тонкий момент – если есть регион, какому уровню его отнести?

и взять с этого уровня описание

$$k = \left\lfloor k_0 + \log_2(\sqrt{wh} / 224) \right\rfloor$$



R-FCN: Region-based Fully Convolutional Networks

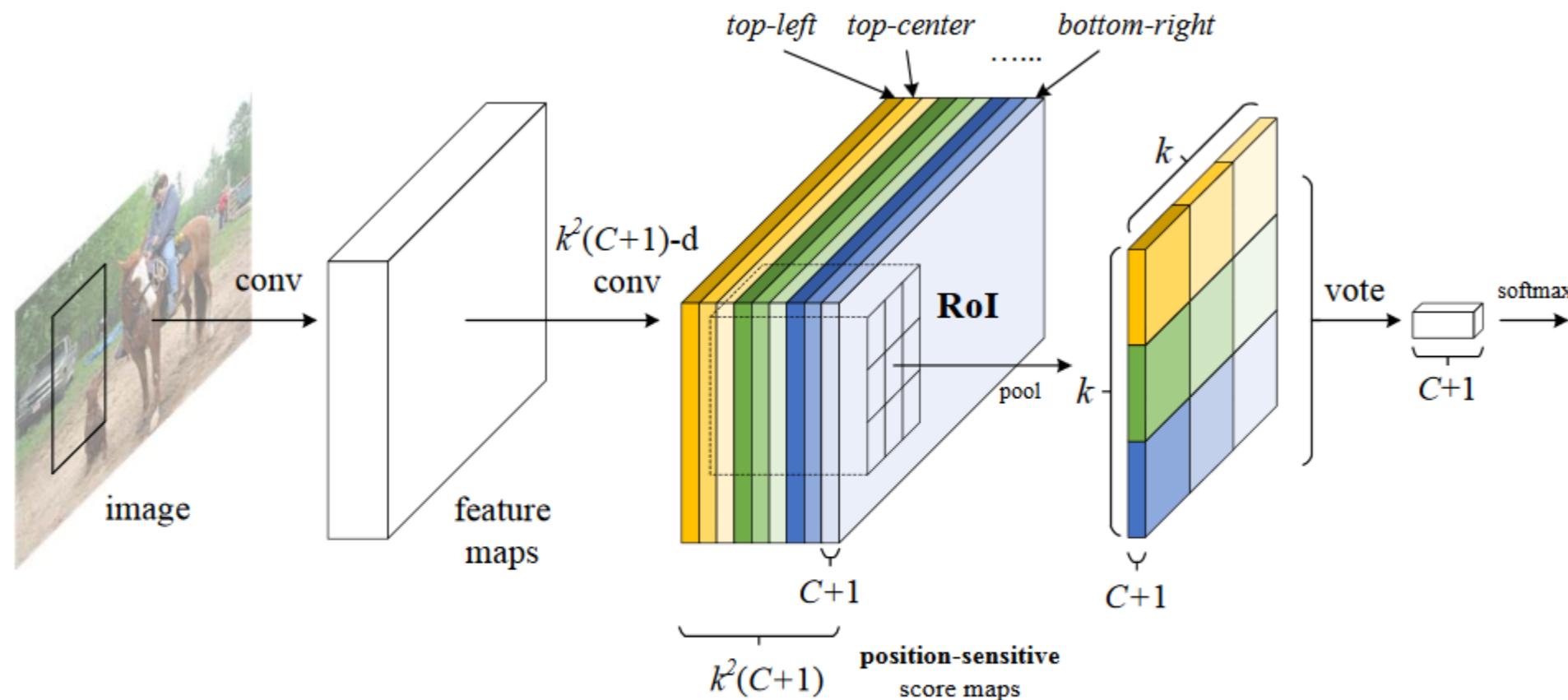


Figure 1: Key idea of **R-FCN** for object detection. In this illustration, there are $k \times k = 3 \times 3$ position-sensitive score maps generated by a fully convolutional network. For each of the $k \times k$ bins in an RoI, pooling is only performed on one of the k^2 maps (marked by different colors).

каждая группа ($C+1$) каналов в последнем слое – регион метки $k \times k$

Jifeng Dai, Yi Li, Kaiming He, Jian Sun «R-FCN: Object Detection via Region-based Fully Convolutional Networks» <https://arxiv.org/pdf/1605.06409v2.pdf>

R-FCN: Region-based Fully Convolutional Networks

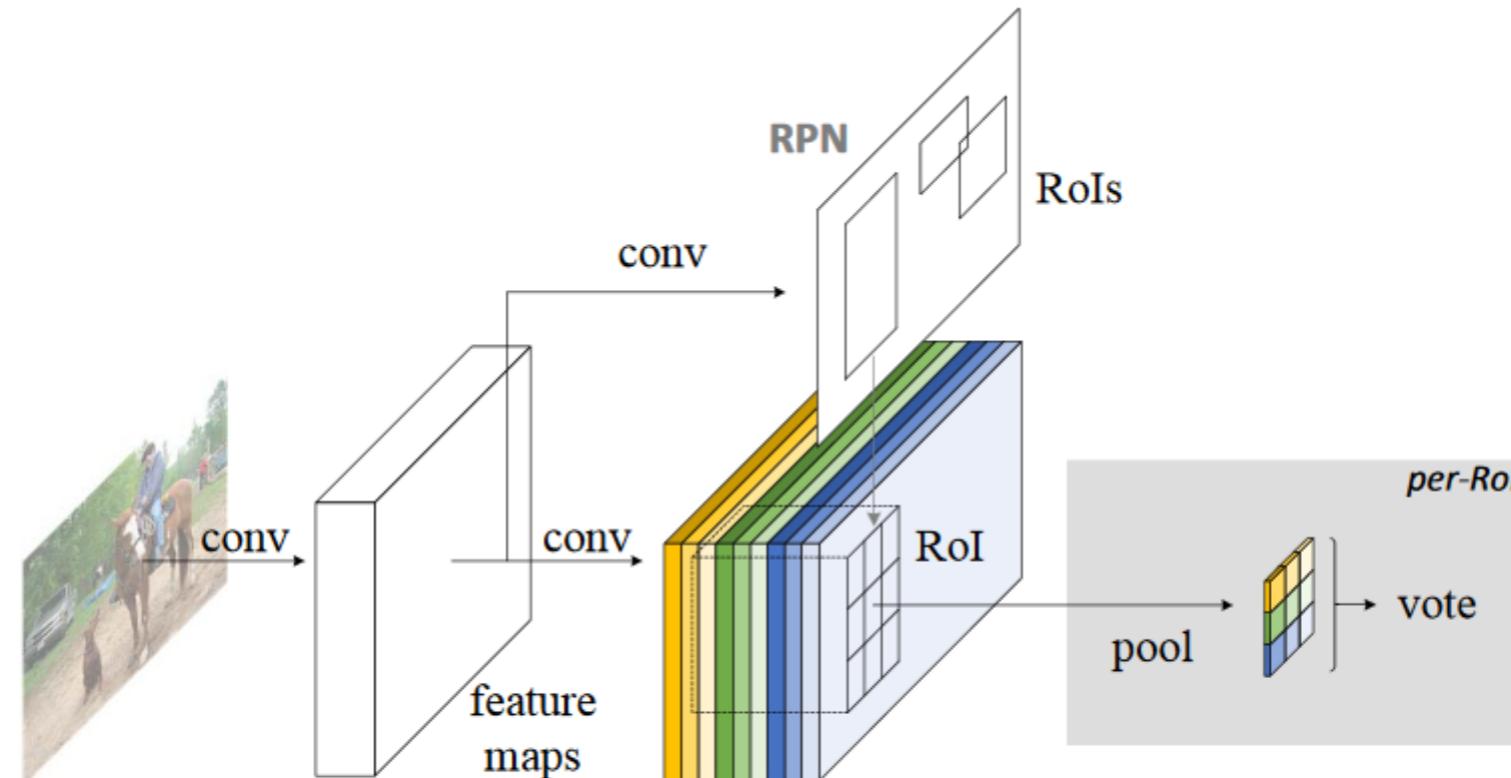


Figure 2: Overall architecture of R-FCN. A Region Proposal Network (RPN) [18] proposes candidate RoIs, which are then applied on the score maps. All learnable weight layers are convolutional and are computed on the entire image; the per-RoI computational cost is negligible.

**на основе ResNet-101 (предтренировка на ImageNet)
с помощью RPN регионы – они делятся на сетку $k \times k$
далее – пулинг по каналам, softmax по категориям → $k \times k$ -сетка**

R-FCN: Region-based Fully Convolutional Networks

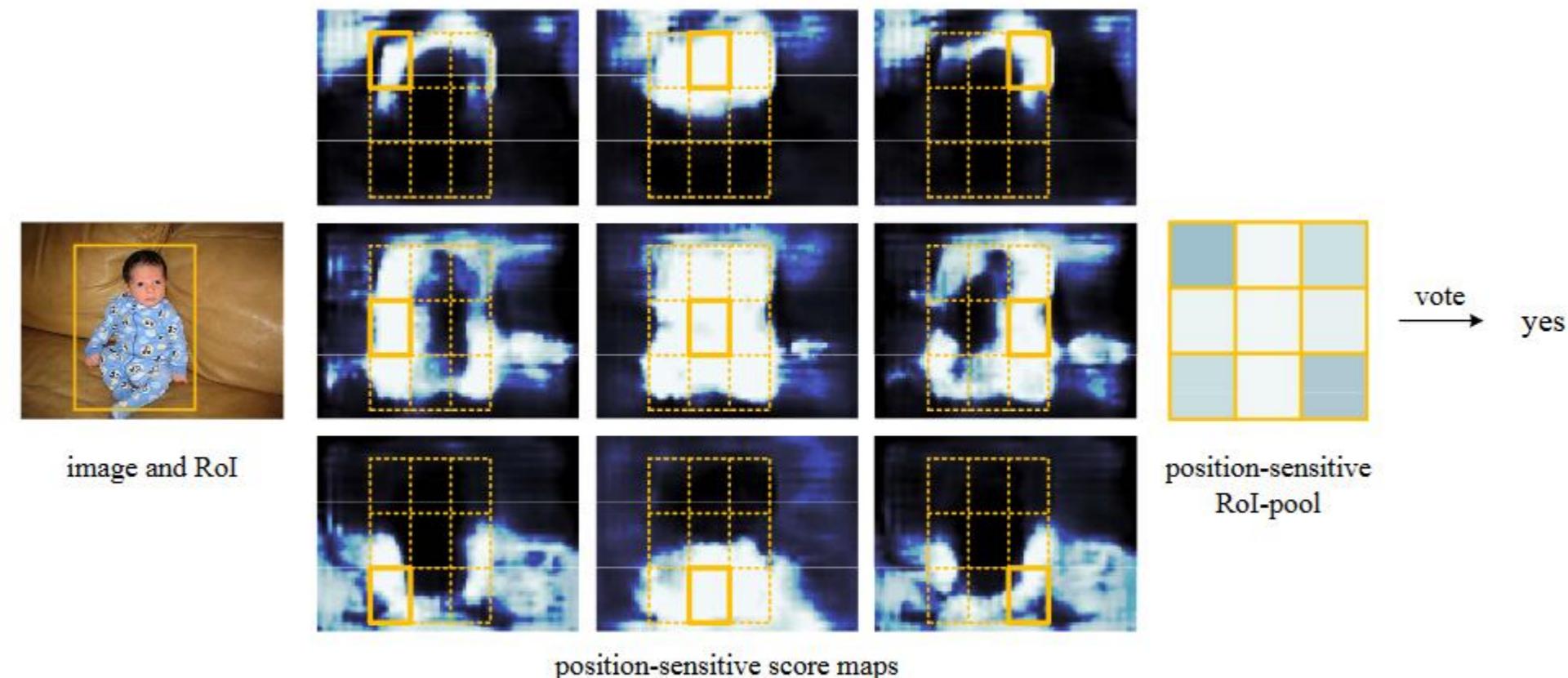


Figure 3: Visualization of R-FCN ($k \times k = 3 \times 3$) for the *person* category.

FCOS: Fully Convolutional One-Stage Object Detection

anchor box free

это затратный модуль

есть гиперпараметры, от которых сильно зависит качество
все регионы-кандидаты при обучении надо размечать
(опять же – долго + гиперпараметры)

proposal free

Zhi Tian, Chunhua Shen, Hao Chen, Tong He «FCOS: Fully Convolutional One-Stage Object Detection» // <https://arxiv.org/pdf/1904.01355.pdf>

FCOS: Fully Convolutional One-Stage Object Detection

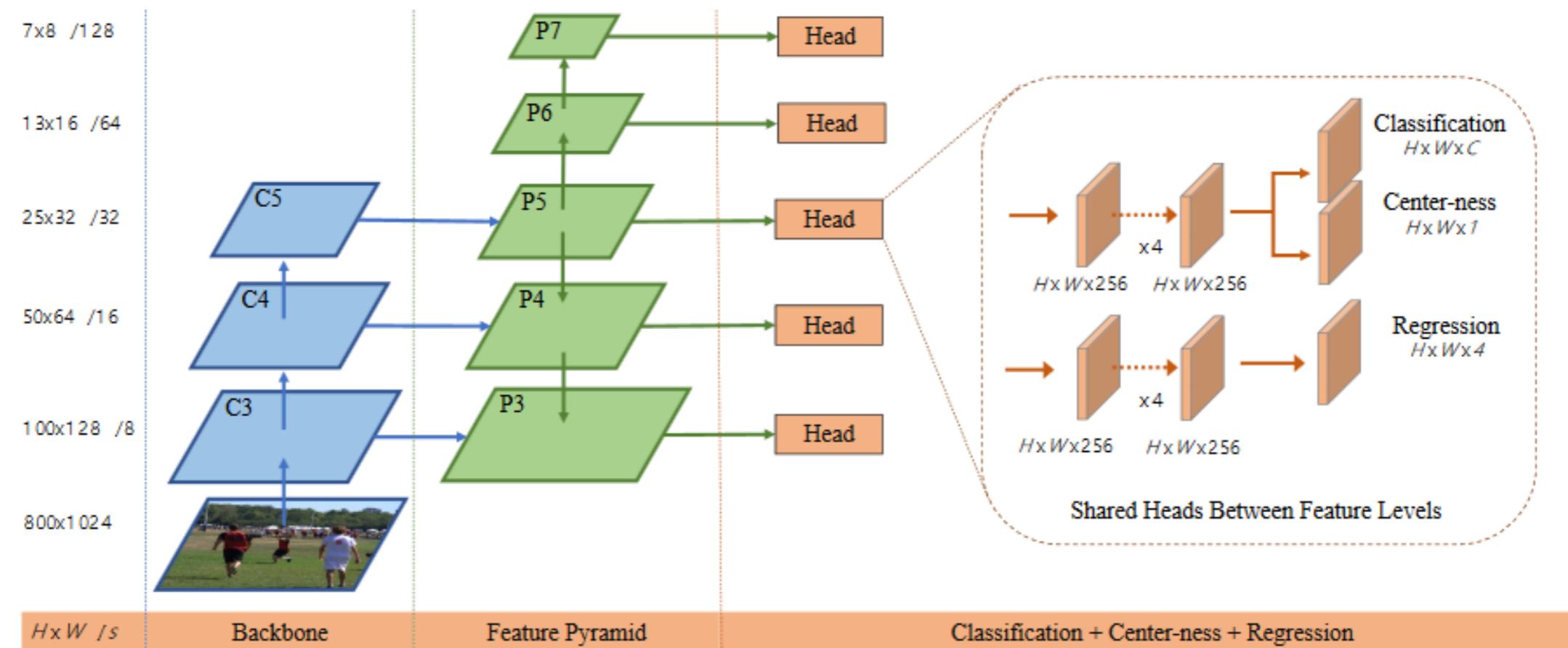


Figure 2 – The network architecture of FCOS, where C3, C4, and C5 denote the feature maps of the backbone network and P3 to P7 are the feature levels used for the final prediction. $H \times W$ is the height and width of feature maps. ‘ s ’ ($s = 8, 16, \dots, 128$) is the down-sampling ratio of the feature maps at the level to the input image. As an example, all the numbers are computed with an 800×1024 input.

Feature Pyramid Network(FPN) + выходы для центра, регрессии и класса

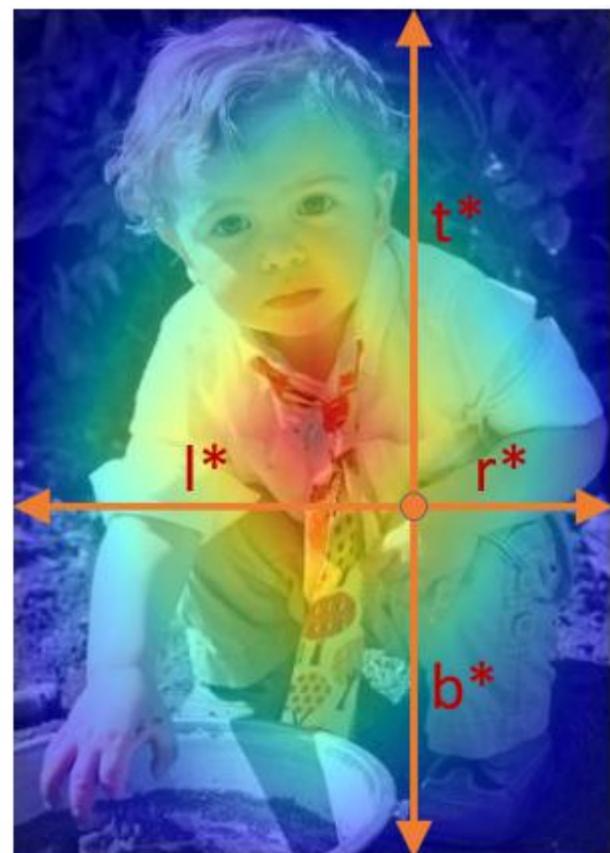
FCOS: Fully Convolutional One-Stage Object Detection

**идея: есть целевые регионы – им сразу и обучаемся
«One-Stage» – нет как раньше отдельных подзадач**

**для любой точки на изображении:
если она попадает в целевой регион – «позитивная»
если в несколько – «двузначная» (относим к минимальному)
не проблема**

основа: ResNet-50

FCOS: Center-ness



$$\text{centerness}^* = \sqrt{\frac{\min(l^*, r^*)}{\max(l^*, r^*)} \times \frac{\min(t^*, b^*)}{\max(t^*, b^*)}}.$$

при работе к точкам с большим значением
можно применить NMS

Figure 3 – Center-ness. Red, blue, and other colors denote 1, 0 and the values between them, respectively. Center-ness is computed by Eq. (3) and decays from 1 to 0 as the location deviates from the center of the object. When testing, the center-ness predicted by the network is multiplied with the classification score thus can down-weight the low-quality bounding boxes predicted by a location far from the center of an object.

FCOS: Fully Convolutional One-Stage Object Detection (l, t, r, b) а не (x, y, x, y)

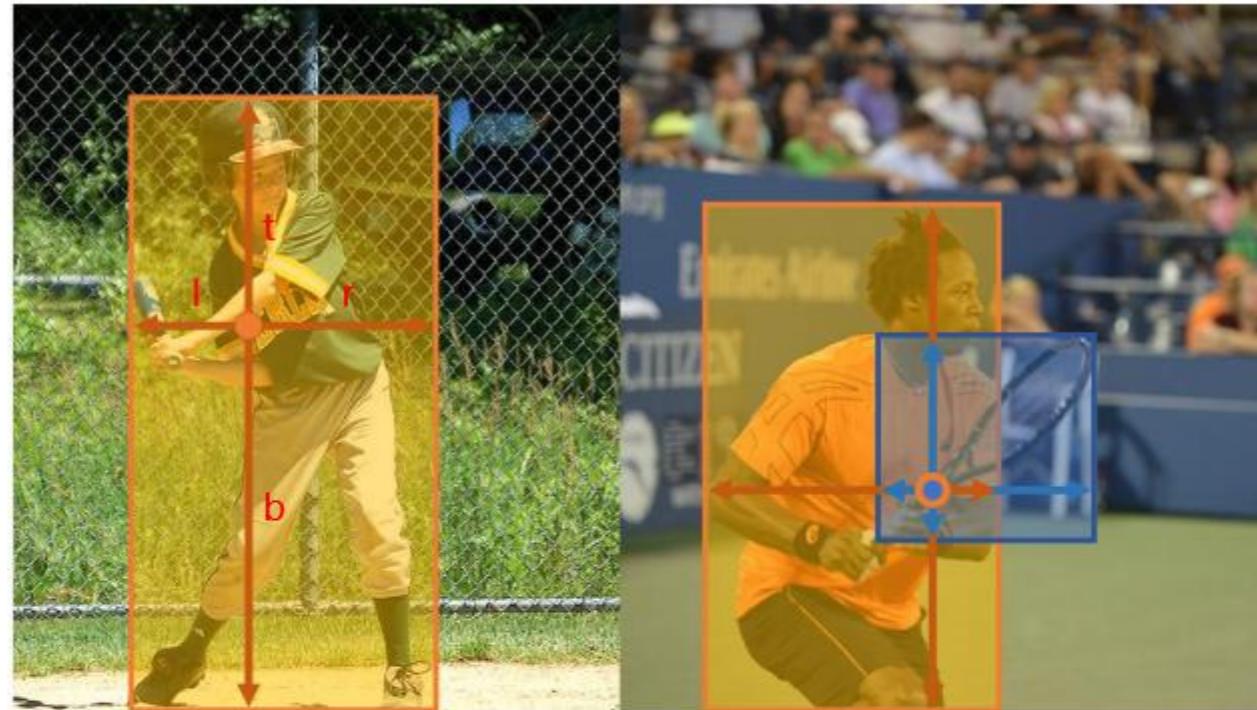


Figure 1 – As shown in the left image, FCOS works by predicting a 4D vector (l, t, r, b) encoding the location of a bounding box at each foreground pixel (supervised by ground-truth bounding box information during training). The right plot shows that when a location residing in multiple bounding boxes, it can be ambiguous in terms of which bounding box this location should regress.

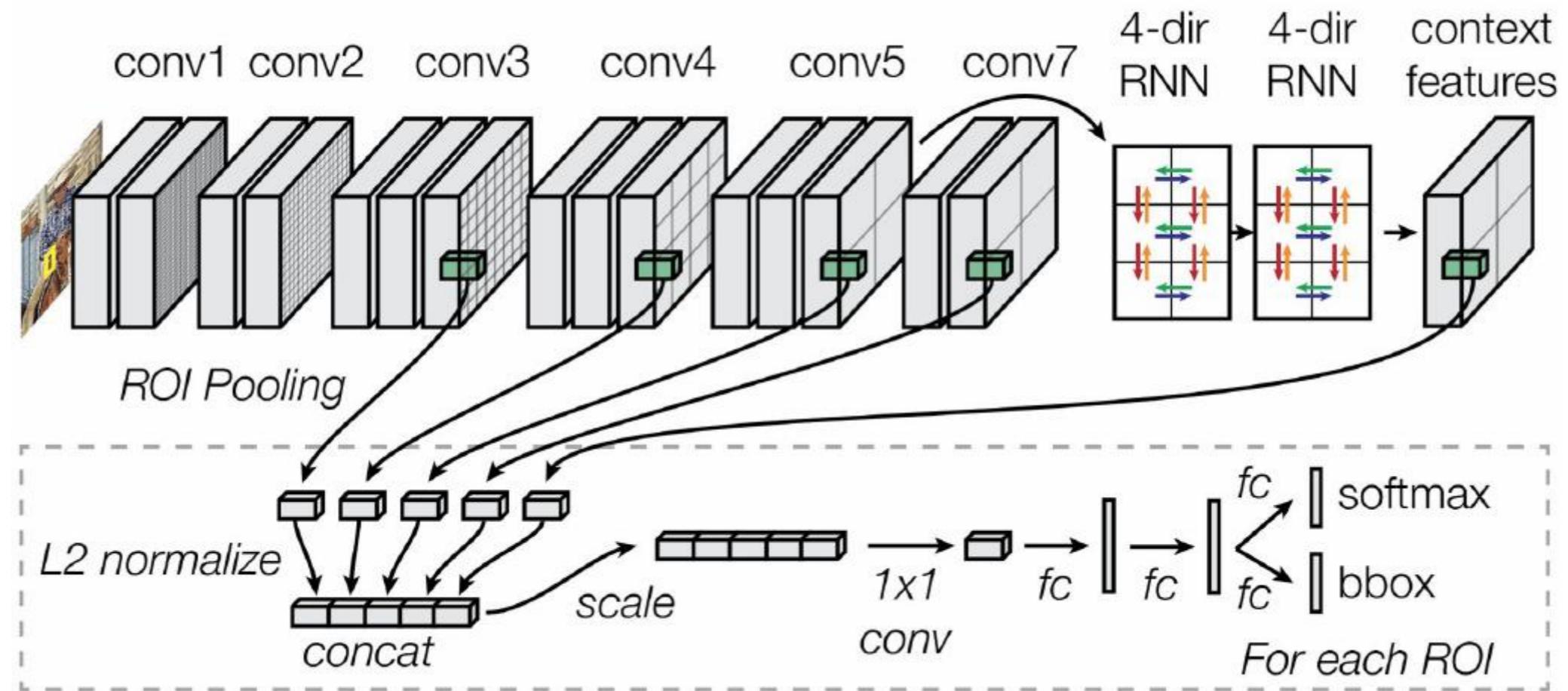
FCOS: Fully Convolutional One-Stage Object Detection

Method	Backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
Two-stage methods:							
Faster R-CNN w/ FPN [14]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [11]	Inception-ResNet-v2 [27]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w/ TDM [25]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
One-stage methods:							
YOLOv2 [22]	DarkNet-19 [22]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [18]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [5]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [15]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
CornerNet [13]	Hourglass-104	40.5	56.5	43.1	19.4	42.7	53.9
FSAF [34]	ResNeXt-64x4d-101-FPN	42.9	63.8	46.3	26.6	46.2	52.7
FCOS	ResNet-101-FPN	41.5	60.7	45.0	24.4	44.8	51.6
FCOS	HRNet-W32-51 [26]	42.0	60.4	45.3	25.4	45.0	51.0
FCOS	ResNeXt-32x8d-101-FPN	42.7	62.2	46.1	26.0	45.6	52.6
FCOS	ResNeXt-64x4d-101-FPN	43.2	62.8	46.6	26.5	46.2	53.3
FCOS w/ improvements	ResNeXt-64x4d-101-FPN	44.7	64.1	48.4	27.6	47.5	55.6

Table 5 – FCOS vs. other state-of-the-art two-stage or one-stage detectors (*single-model and single-scale results*). FCOS outperforms the anchor-based counterpart RetinaNet by 2.4% in AP with the same backbone. FCOS also outperforms the recent anchor-free one-stage detector CornerNet with much less design complexity. Refer to Table 3 for details of “improvements”.



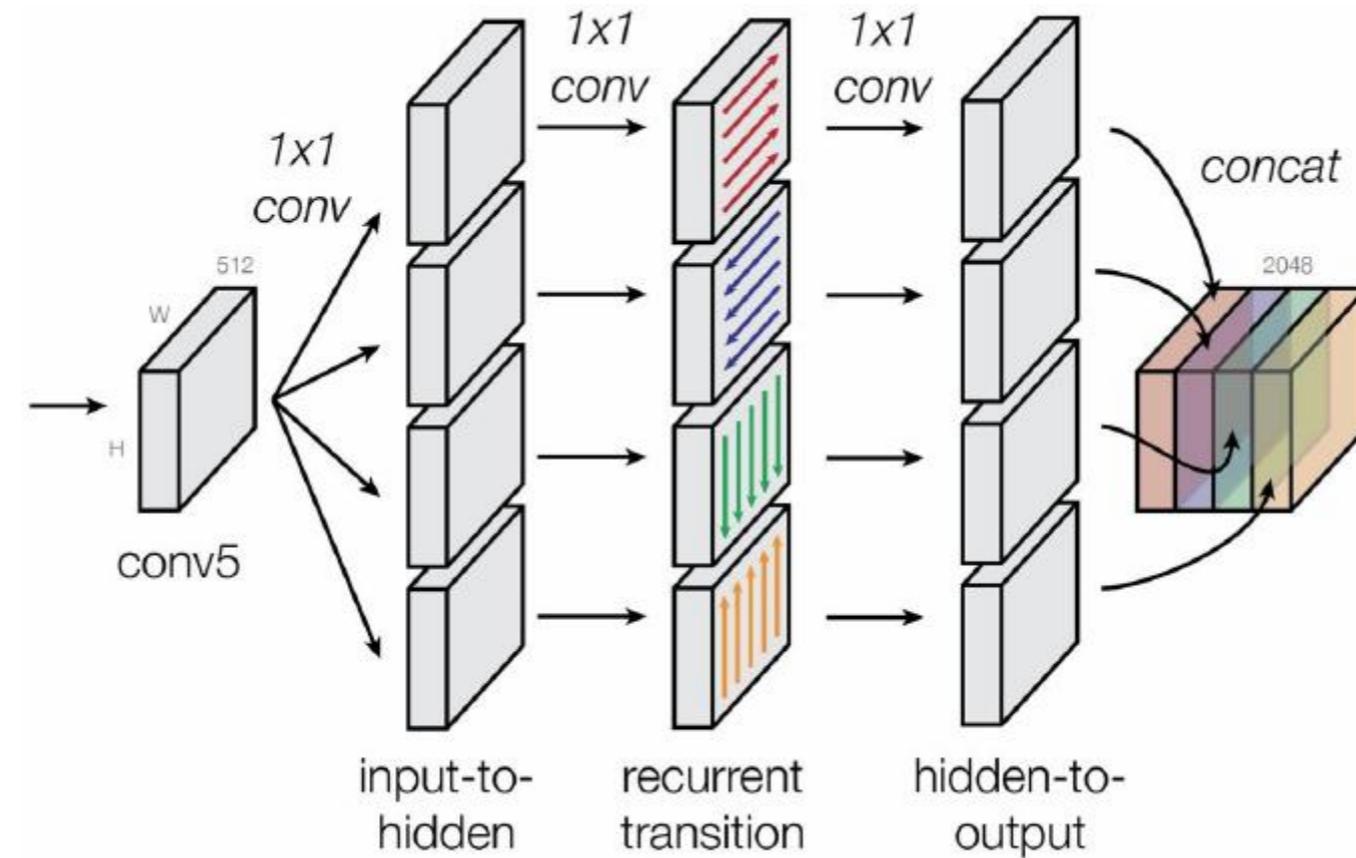
Детектирование объектов: ION (Inside-Outside Net)



<https://people.eecs.berkeley.edu/~rbg/papers/r-cnn-cvpr.pdf>

<https://www.robots.ox.ac.uk/~vgg/rg/slides/ion-coco.pdf>

Использование RNN для извлечение контекста с разных сторон региона



Skip connection with L2 normalization

Stacked 4-direction RNNs for context

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