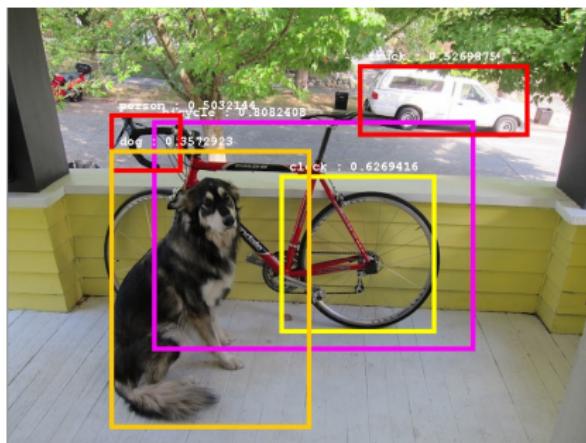
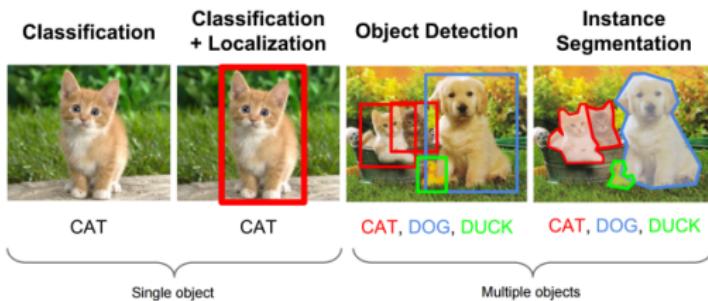


# Детекция объектов

Виктор Китов  
[victorkitov.github.io](http://victorkitov.github.io)



# Локализация объектов



- Локализация: не только классифицировать объект (единственный), но и выделить рамкой.
  - два выхода - вероятности классов и  $(x, y, w, h)$
- Детекция: объектов может не быть или быть несколько разных классов.
- Instance-сегментация: выделить не рамкой, а маской.

## Приложения: подсчёт числа людей

Подсчёт числа людей:

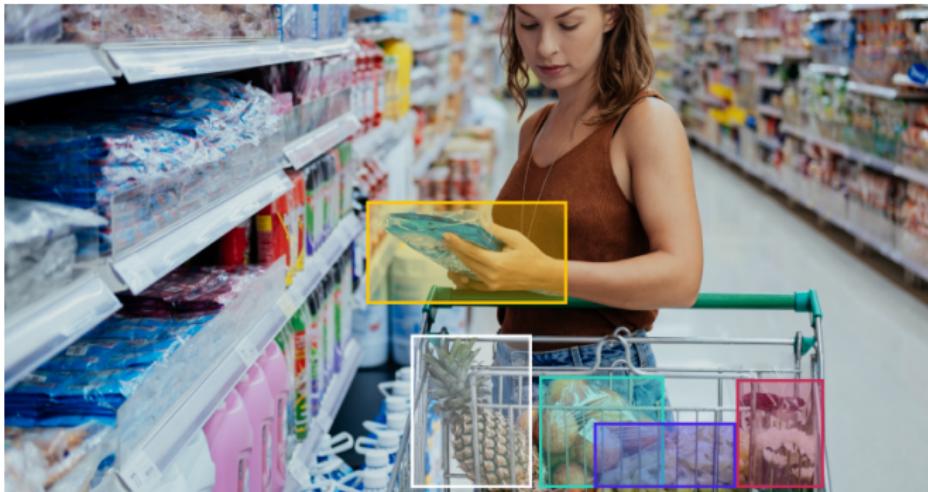
- оценка популярности мероприятий, управление потоками, чтобы не было заторов



## Приложения: поведение покупателей в магазине

Поведение покупателей в магазине:

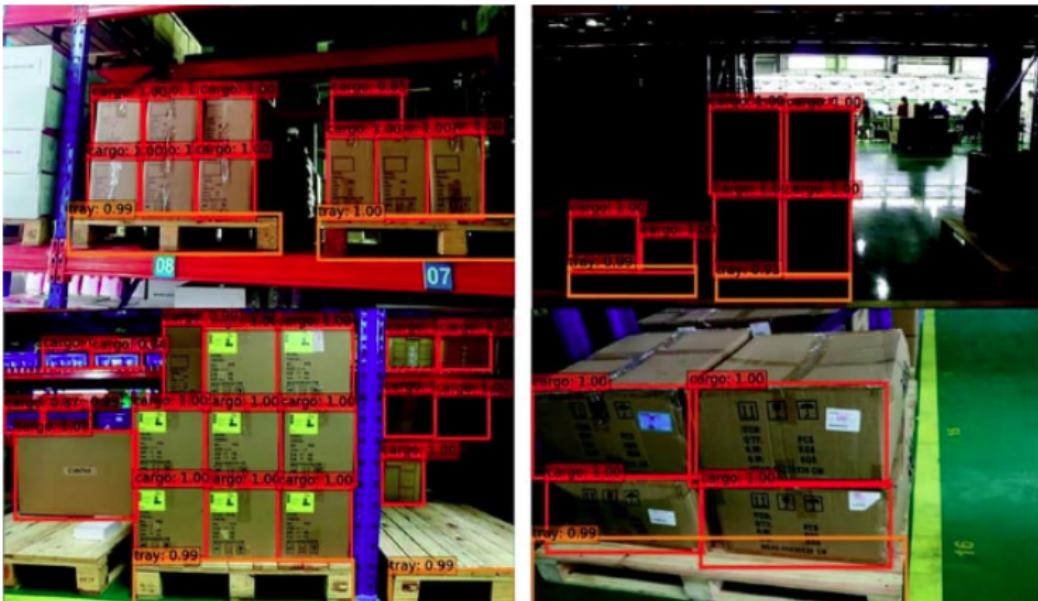
- безопасность, оптимизация расположения товаров на полках, автоматическое выписывание чека



# Приложения: управление складом

Управление складом:

- распределение товаров по складу, учёт товаров.



## Приложения: безопасность

Безопасность:

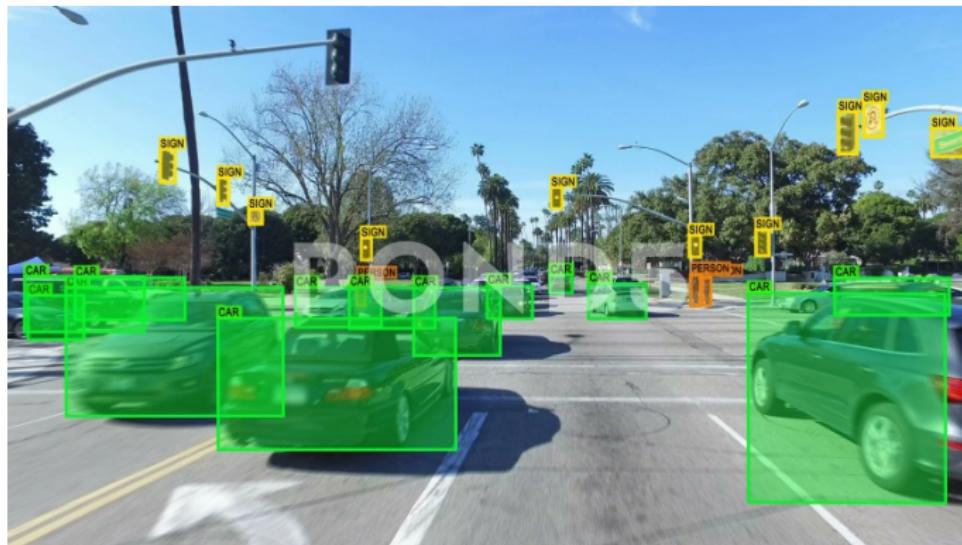
- обнаружение запрещенных предметов, незаконных действий, краж, идентификация человека



## Приложения: self-driving cars

Self-driving cars:

- обнаружение светофоров, знаков, съездов, стоянок, пешеходов, др. машин



## Приложения



ball tracking



object identification

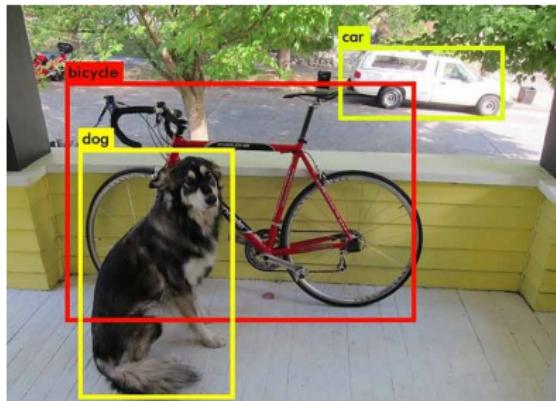


activity recognition



face identification,  
emotion identification

## Постановка задачи



Обучающая выборка:

- img1.jpg: X(1), Y(1), X(2), Y(2), класс
- img2.jpg: X(1), Y(1), X(2), Y(2), класс
- .....

# Популярные датасеты для детекции объектов

| 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 |

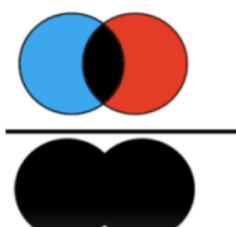
# Меры качества

- Intersection over union (IoU) = близость Жаккарда (в детекции - только для прямоугольников)

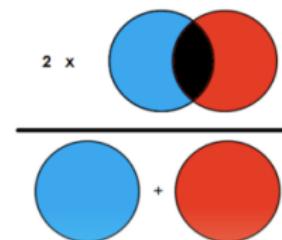
$$IoU = \frac{TP}{TP + FP + TN} = \frac{a}{b}$$

$$Dice = \frac{TP + TP}{TP + TP + FP + TN} = \frac{2a}{a + b}$$

$$Dice = \frac{\frac{2a}{b}}{\frac{a+b}{b}} = \frac{2 \cdot \frac{a}{b}}{\frac{a}{b} + 1} = \frac{2 \cdot IoU}{IoU + 1}$$



IoU



Dice

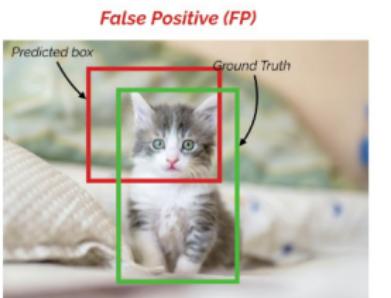
# Меры качества

- Точность =  $TP/\hat{P}$ , Полнота =  $TP/P$

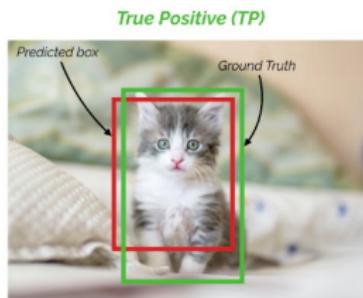
$$\text{Precision} = \frac{\text{Green}}{\text{Red} + \text{Green}}$$


$$\text{Recall} = \frac{\text{Green}}{\text{Red} + \text{Green}}$$


- TP: вероятность класса  $\geq$  порога и IoU  $\geq$  порога
- FP: вероятность класса  $<$  порога или IoU  $<$  порога



$IoU = \sim 0.3$



$IoU = \sim 0.7$

## Кривая точности-полноты (precision-recall curve)

- Задаём порог на минимальное пересечение  $IoU_{min}$ 
  - верная детекция - верный класс и  $IoU \geq IoU_{min}$

## Кривая точности-полноты (precision-recall curve)

- Задаём порог на минимальное пересечение  $IoU_{min}$ 
  - верная детекция - верный класс и  $IoU \geq IoU_{min}$
- Для порога уверенности  $t = 1, \dots, 0$  (по  $\downarrow$  уникальных значений уверенности):
  - выделяем объекты
  - считаем  $Pr(t), Rec(t)$ ; достраиваем  $Prec(Recall)$
- Выход:  $Prec(Recall)$  - зависимость точности от полноты

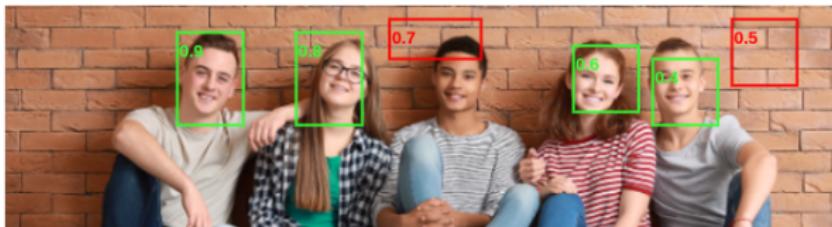
# Кривая точности-полноты (precision-recall curve)

- Задаём порог на минимальное пересечение  $IoU_{min}$ 
  - верная детекция - верный класс и  $IoU \geq IoU_{min}$
- Для порога уверенности  $t = 1, \dots, 0$  (по  $\downarrow$  уникальных значений уверенности):
  - выделяем объекты
  - считаем  $Pr(t), Rec(t)$ ; достраиваем  $Prec(Recall)$
- Выход:  $Prec(Recall)$  - зависимость точности от полноты
- Пересчитываем интерпolatedированную точность (*interpolated precision*)

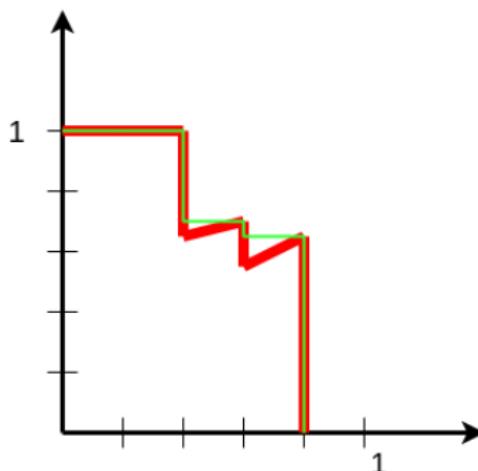
$$\overline{Pr}(i) = \max_{j: Rec(j) \geq Rec(i)} \{Pr(j)\}$$

- Получим сглаженный график  $\overline{Prec}(Recall)$ .

# Кривая точности-полноты (precision-recall curve)



| порог | precision | recall |
|-------|-----------|--------|
| 0.9   | 1         | 1/5    |
| 0.8   | 1         | 2/5    |
| 0.7   | 2/3       | 2/5    |
| 0.6   | 3/4       | 3/5    |
| 0.5   | 3/5       | 3/5    |
| 0.4   | 4/6       | 4/5    |
| 0.3   | 4/6       | 4/5    |
| ...   | ...       | ...    |
| 0     | 4/6       | 4/5    |



## Average precision, mean average precision

- Average precision (AP) -площадь под  $\overline{Prec}(Recall)$ :

$$AP = \int_0^1 \overline{Prec}(Recall) d(Recall)$$

- Mean average precision - макроусреднённая AP по классам:

$$mAP = \frac{1}{C} \sum_{c=1}^C AP(c)$$

# Содержание

- 1 Двухстадийные детекторы
- 2 Одностадийные детекторы (one stage, proposal free)

## Детекция объектов: простейший подход



Простейший подход:

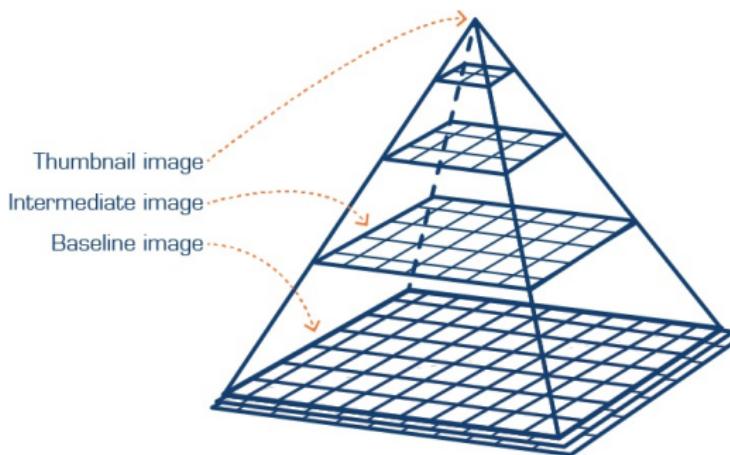
- ① извлечь участки на всех позициях и разных размеров
- ② применить классификатор к каждому участку.
- неэффективно, но отражает идеологию др. методов.

## Извлечение участков разного размера

Проблема: детектор натренирован под фикс. разрешение.

Решение: извлекаем участки одного размера с разных масштабов исх. изображения.

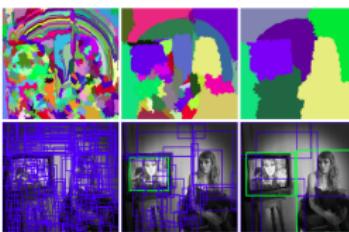
Гауссова пирамида



Также нужно извлекать участки разной формы.

## Генерация участков-кандидатов

- Алгоритм selective search<sup>1</sup> генерирует участки-кандидаты.

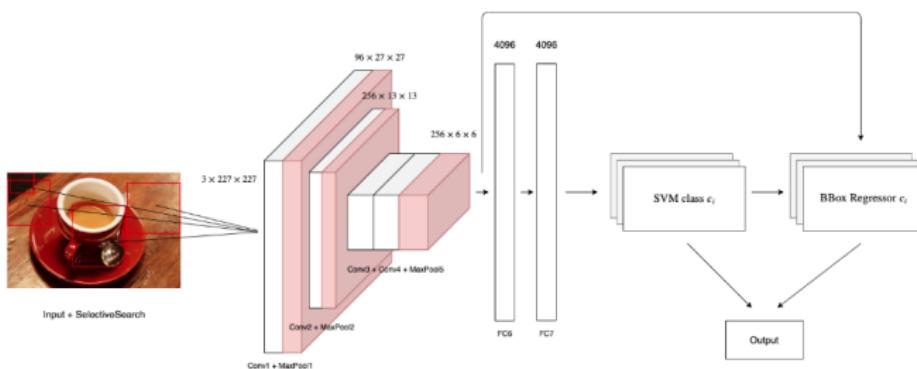


- Алгоритм:
  - объединяем RGB+XY объекты в суперпиксели (например, используя SLIC).
  - пока  $\#\text{суперпикселей} > 1$ :
    - рисуем прямоугольник вокруг каждого суперпикселя, запоминаем.
    - объединяем соседние кластеры, похожие по цвету.
- Выход алгоритма: множество прямоугольников, в которых потенциально может быть объект.

<sup>1</sup><http://www.huppelen.nl/publications/selectiveSearchDraft.pdf>

# R-CNN<sup>2</sup>

R-CNN scheme:



<sup>2</sup><https://arxiv.org/pdf/1311.2524.pdf>

## Алгоритм R-CNN

- ❶ SelectiveSearch генерирует  $\sim 2000$  регионов-кандидатов
- ❷ Регионы-кандидаты масштабируются к 224x224 (AlexNet)
- ❸ Кодировщик (AlexNet) сверточные слои (но не полно связные) извлекают 4096 признаков для каждого региона.
- ❹ SVM классификатор обучается на  $C + 1$  класс (+1 для фона)
- ❺ Регрессия обучается уточнять координаты регионов, предложенных SelectiveSearch:

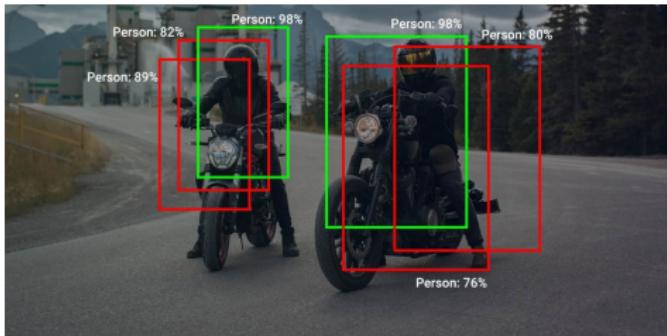
$(\hat{x}, \hat{y}, \hat{h}, \hat{w})$  -предсказанный регион,  $(x, y, h, w)$  -реальный регион

регрессия предсказывает:  $\left( \frac{x - \hat{x}}{w}, \frac{y - \hat{y}}{h}, \ln \left( \frac{\hat{w}}{w} \right), \ln \left( \frac{\hat{h}}{h} \right) \right)$

регрессия обучается на регионах с  $\text{IoU} > 0.3$  с истинным регионом

- ❻ Убираем лишние выделения (non-maximum suppression)

## Подавление не-максимумов (non-maximum suppression)

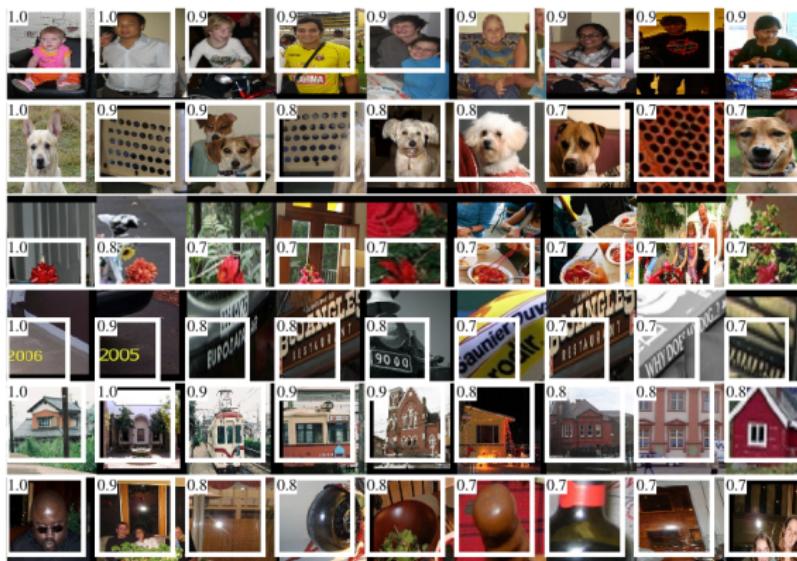


Алгоритм подавления не-максимумов:

- отбрасываем регионы с низкой уверенностью
- упорядочиваем регионы по убыванию уверенности
- последовательно, начиная с региона максимальной уверенности, к менее уверенным:
  - если регион имеет высокий IoU с др. регионом более низкой уверенности, отбрасываем 2й регион.

## Интерпретация нейронов

Визуализация участков, наиболее активирующих определенные нейроны внутри кодировщика R-CNN.



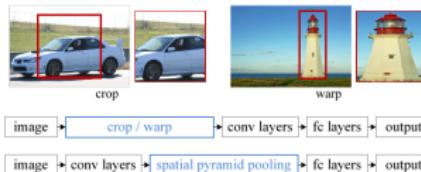
## Недостатки R-CNN

Недостатки R-CNN:

- требуется отдельная процедура обучения для
  - кодировщика (донастройка по log-loss)
  - классификатора (SVM)
  - регрессии
- для каждого региона кандидата нужно применять полносверточный энкодер
  - регионов-кандидатов много и они сильно пересекаются

## SPP-net<sup>3</sup>

- Проблемы R-CNN:
  - CNN переприменяется к каждому региону
  - необходимо приводить разрешение к 224x224 (деформация либо обрезка краёв с потерей информации)

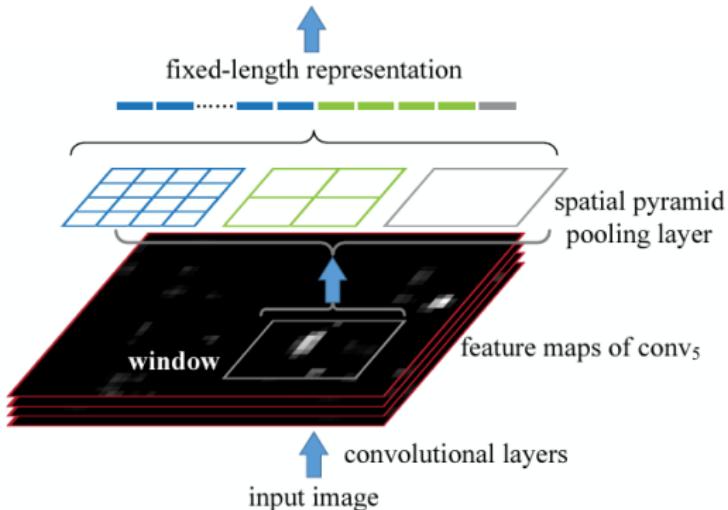


Идеи SPP-net (spatial pyramid pooling net):

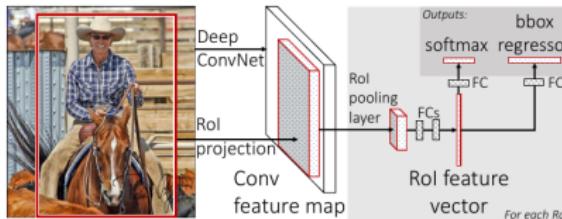
- применить CNN ко всему изображению 1 раз
  - ускорение 10x - 100x
- spatial pyramid pooling приводит к фикс. размеру изображение произвольного размера

<sup>3</sup><https://arxiv.org/pdf/1406.4729v4.pdf>

# Spatial pyramid pooling



## Fast R-CNN<sup>4</sup>

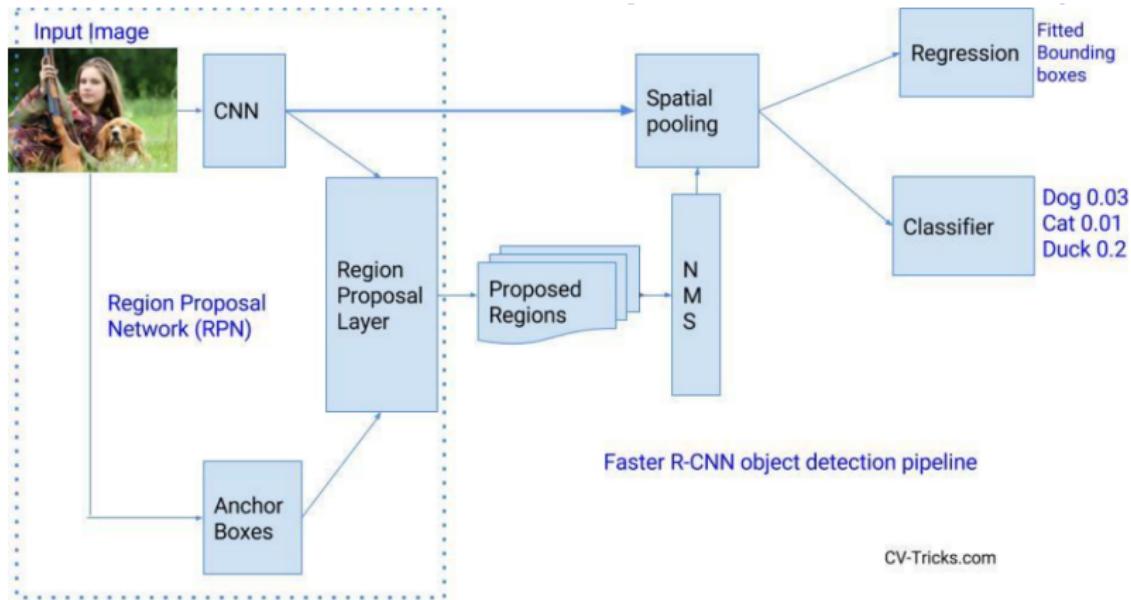


Fast R-CNN: SPP-net со следующими отличиями:

- Однослойный spatial pyramid max pooling на фикс. сетке 7x7 (названный ROI pooling).
  - произвольный тензор (выделяющий область) -> 49C признаков
- Классификатор и bbox-регрессия реализованы доп. слоями сети.
  - потери=классификационные+регрессионные (от локализации)

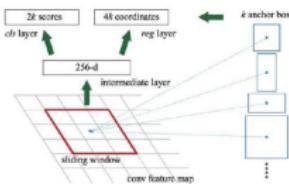
<sup>4</sup><https://arxiv.org/pdf/1504.08083.pdf>

# Faster R-CNN



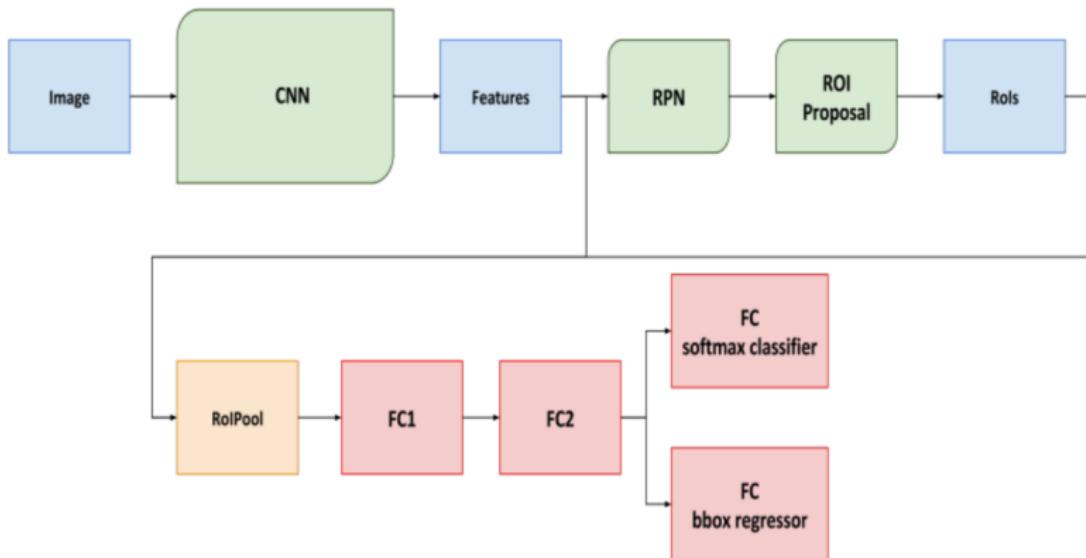
- Полностью нейросетевое решение:
  - SelectiveSearch->генерация регионов-кандидатов через Region Proposal Network (RPN).

# Faster R-CNN



- Region proposal network: 1 этап
- Проходим скользящим окном  $3 \times 3$  по карте признаков, генерируем  $k=9$  регионов-кандидатов.
  - регионы-кандидаты - шаблоны размера 128, 256, 512 и соотношением сторон 1:1, 1:2, 2:1
- Выход RPN:
  - вероятность присутствия/отсутствия объекта:  $2k$
  - положение объекта:  $4k$  координаты (уточненных регрессией)
- Постпроцессинг регионов-кандидатов: удаляем слишком узкие или выходящие за пределы изображения регионы, NMS с  $\text{IoU}=0.9$

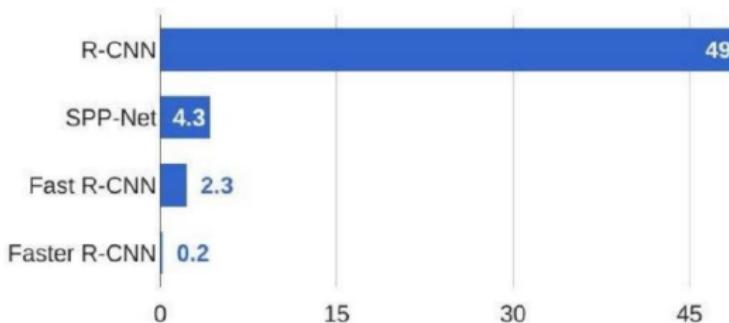
# Faster R-CNN



# Faster R-CNN

- Обучение: учим поочередно итеративно RPN и оставшуюся часть сети.
- Применение:
  - изображение->CNN+RPN
  - оставляем 6000 перспективных регионов
  - удаляем слишком тонкие регионы или попавшие за пределы изображения
  - подавление не-максимумов с высоким  $\text{IoU}=0.9$
  - оставшиеся регионы->RoI  
 $\text{pooling}$ ->классификация+регрессия
  - итоговое подавление не-максимумов

## Сравнение скорости работы



- По скорости работы Faster-RCNN уступает одностадийным детекторам, но превосходит их по точности.

## Содержание

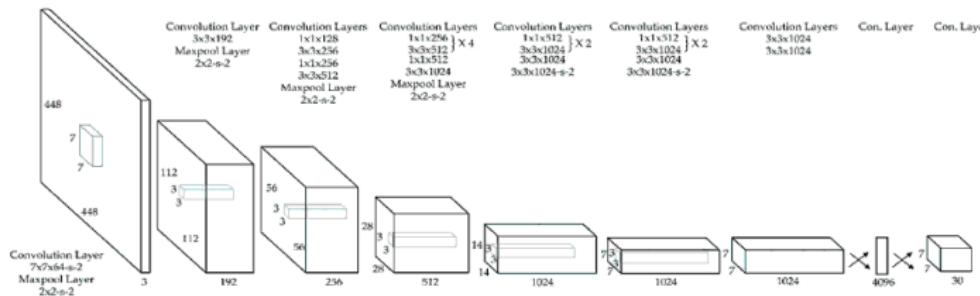
- 1 Двухстадийные детекторы
- 2 Одностадийные детекторы (one stage, proposal free)
  - Методы с шаблонными рамками (anchor based)
  - Методы без шаблонных рамок (anchor free)

2 Одностадийные детекторы (one stage, proposal free)

- Методы с шаблонными рамками (anchor based)
- Методы без шаблонных рамок (anchor free)

# YOLO<sup>5</sup>

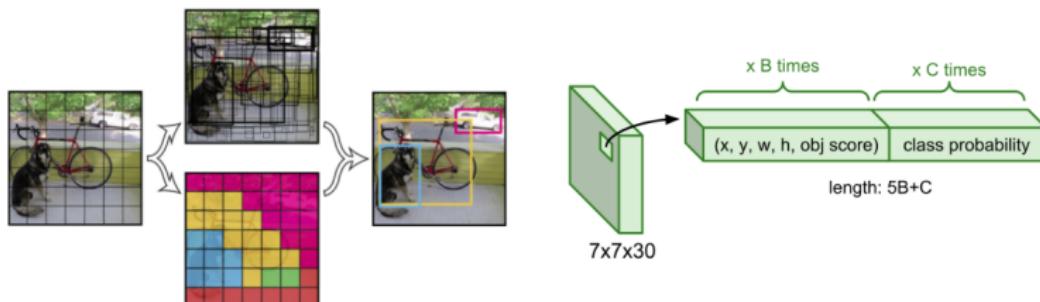
- YOLO (You look only once) одностадийный детектор
  - сразу предсказывает выход, без этапа предсказания регионов-кандидатов (proposal free, one stage)



- Вход 448x448.
- CNN, 2 полно связных слоя в конце
  - за счёт них прогнозы "видят" всё изображение
- DropOut после 1го полно связного слоя, везде LeakyReLU.

<sup>5</sup><https://arxiv.org/pdf/1506.02640.pdf>

# YOLO: выход

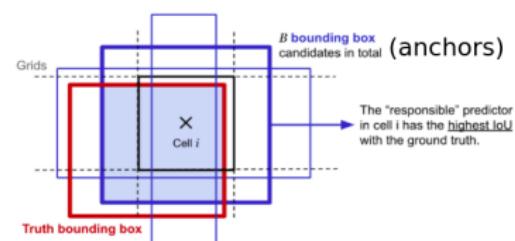


- Выход - фикс. сетка  $S \times S$ ,  $S = 7$ , в каждом элементе сетки:
  - $p(y = 1|obj), p(y = 2|obj), \dots p(y = C|obj)$
  - $B$  раз (чтобы  $\uparrow$ recall):
    - $x, y, w, h$  - относит. координаты рамки
    - $\text{obj\_score} = p(\text{class}_i) \cdot IoU_{\text{pred}}^{\text{truth}}$  - степень присутствия объекта (0, если отсутствует)

# YOLO: функция потерь

$$\begin{aligned}
 & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
 & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
 & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left( C_i - \hat{C}_i \right)^2 \\
 & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} \left( C_i - \hat{C}_i \right)^2 \\
 & + \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
 \end{aligned}$$

where  $\mathbb{1}_i^{\text{obj}}$  denotes if object appears in cell  $i$  and  $\mathbb{1}_{ij}^{\text{obj}}$  denotes that the  $j$ th bounding box predictor in cell  $i$  is “responsible” for that prediction.

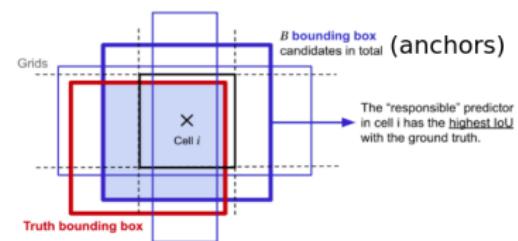


- Для каждой ячейки  $B$  прогнозов рамки. Выбиралась одна (responsible), имеющая макс.  $IoU_{\text{pred}}^{\text{truth}}$ .
- $\hat{C}_i = p(\text{class}_i) \cdot IoU_{\text{pred}}^{\text{truth}}$  - confidence score для макс. релевантного прогноза из  $B$  для рамки.
- $p_i(c) = p(y = c | obj)$  - в рамке  $i$ .

# YOLO: функция потерь

$$\begin{aligned}
 & \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ (x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\
 & + \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left[ \left( \sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left( \sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\
 & + \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{obj}} \left( C_i - \hat{C}_i \right)^2 \\
 & + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^B \mathbb{1}_{ij}^{\text{noobj}} \left( C_i - \hat{C}_i \right)^2 \\
 & + \sum_{i=0}^{S^2} \mathbb{1}_i^{\text{obj}} \sum_{c \in \text{classes}} (p_i(c) - \hat{p}_i(c))^2
 \end{aligned}$$

where  $\mathbb{1}_i^{\text{obj}}$  denotes if object appears in cell  $i$  and  $\mathbb{1}_{ij}^{\text{obj}}$  denotes that the  $j$ th bounding box predictor in cell  $i$  is "responsible" for that prediction.



- $\lambda_{\text{noobj}} = 0.5$  - для выравнивания ситуаций "объект присутствует/отсутствует"
- Штрафуется расхождение между  $\sqrt{w}$  и  $\sqrt{h}$ , т.к. одинаковая ошибка на малых  $w, h$  более важна, чем на больших.
- Везде использовалась MSE как более стабильная в обучении.

# YOLO<sup>6</sup>

Особенности обучения:

- Претренировка первых слоев на классификации ImageNet.  
Затем добавление новых свёрточных и полносвязных слоев. Дообучение.
- Сначала  $\uparrow$  learning rate (иначе нестабильное обучение),  
потом плавно  $\downarrow$ .
- Расширение выборки: случайные перенос,  
масштабирование, изменение экспозиции и насыщенности.

Итог:

- Быстрая одностадийная архитектура детекции за счет отсутствия Region Proposal Net.

---

<sup>6</sup>Улучшения YOLO в новых версиях.

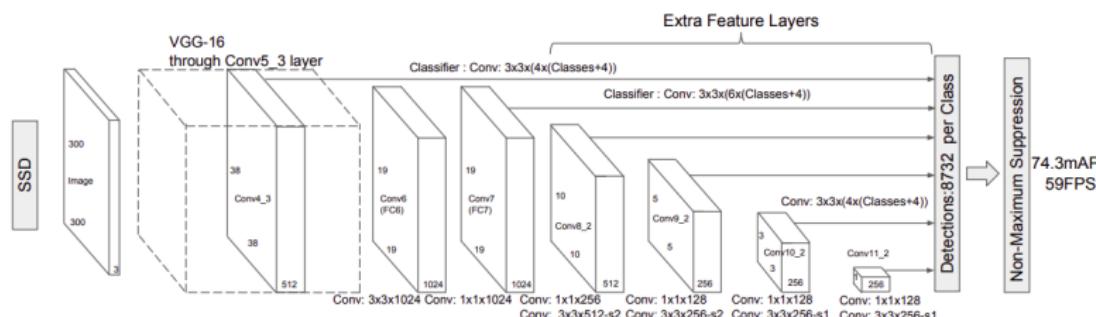
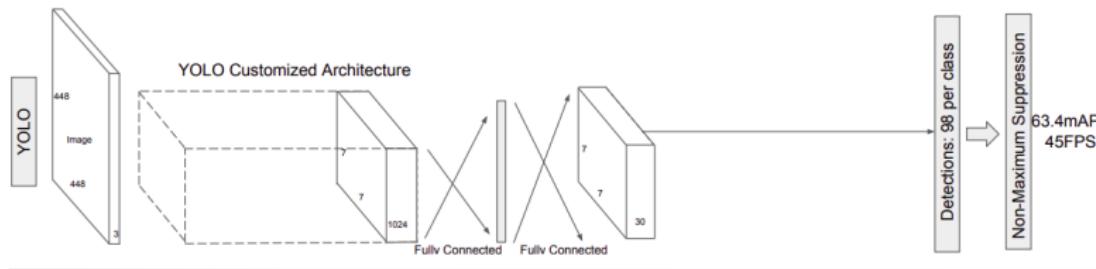
## Недостатки YOLO, SSD

- Недостаток YOLOv1:
  - $\#\text{обнаружений} \leq S \times S = 49$
  - не может извлекать малые объекты (теряются в грубой пространств. сетке)
- SSD (Single Shot MultiBox Detector<sup>7</sup>) решает проблемы YOLOv1:
  - за счёт детекторов, применяемых к разным простр. разрешениям
  - в конце - подавление не-максимумов (убираем дубликаты)

---

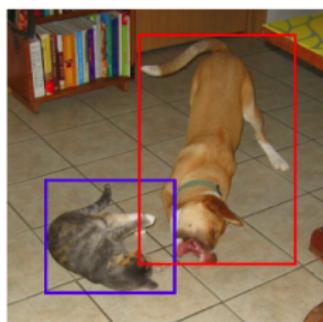
<sup>7</sup><https://arxiv.org/pdf/1512.02325.pdf>

# Сравнение архитектур YOLO и SSD

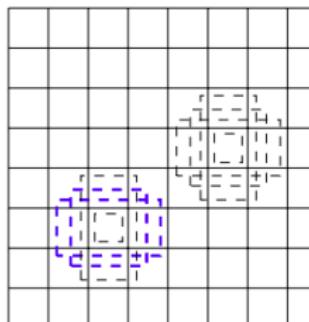
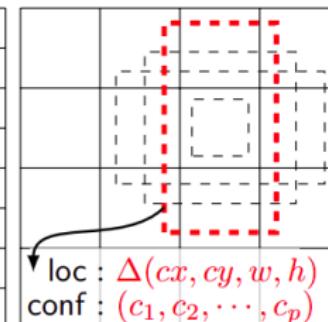


# SSD

- На каждом слое для каждой позиции, используя conv 3x3, предсказываем  $K$  раз (для разных anchor boxes):  
 $(dx, dy, w, h, p(class_1), \dots p(class_C))$ 
  - $K$  - #шаблонных рамок (anchor boxes) задающих шаблонные размеры и соотношения сторон
  - $dx, dy, w, h$  - относительные уточнения соотв. anchor box.
- Всего предсказаний на слое  $H \times W$ :  $HWK(4 + C)$



(a) Image with GT boxes

(b)  $8 \times 8$  feature map(c)  $4 \times 4$  feature map

# SSD: обучение

$$\mathcal{L} = \frac{1}{N} (\mathcal{L}_{\text{cls}} + \alpha \mathcal{L}_{\text{loc}})$$

where  $N$  is the number of matched bounding boxes and  $\alpha$  balances the weights between two losses, picked by cross validation.

The *localization loss* is a smooth L1 loss between the predicted bounding box correction and the true values. The coordinate correction transformation is same as what R-CNN does in bounding box regression.

$$\begin{aligned}\mathcal{L}_{\text{loc}} &= \sum_{i,j} \sum_{m \in \{x,y,w,h\}} \mathbb{1}_{ij}^{\text{match}} L_1^{\text{smooth}}(d_m^i - t_m^j)^2 \\ L_1^{\text{smooth}}(x) &= \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases} \quad \begin{aligned} t_x^j &= (g_x^j - p_x^i)/p_w^i & t_w^j &= \log(g_w^j/p_w^i) \\ t_y^j &= (g_y^j - p_y^i)/p_h^i & t_h^j &= \log(g_h^j/p_h^i) \end{aligned}\end{aligned}$$

where  $\mathbb{1}_{ij}^{\text{match}}$  indicates whether the  $i$ -th bounding box with coordinates  $(p_x^i, p_y^i, p_w^i, p_h^i)$  is matched to the  $j$ -th ground truth box with coordinates  $(g_x^j, g_y^j, g_w^j, g_h^j)$  for any object.  $d_m^i, m \in \{x, y, w, h\}$  are the predicted correction terms.

The *classification loss*  $\mathcal{L}_{\text{cls}} = - \sum_{i \in \text{pos}} \mathbb{1}_{ij}^k \log(\hat{c}_i^k) - \sum_{i \in \text{neg}} \log(\hat{c}_i^0)$ , where  $\hat{c}_i^k = \text{softmax}(c_i^k)$

where  $\mathbb{1}_{ij}^k$  indicates whether the  $i$ -th bounding box and the  $j$ -th ground truth box are matched for an object in class  $k$ . pos is the set of matched bounding boxes ( $N$  items in total) and neg is the set of negative examples. SSD uses hard negative mining to select easily misclassified negative examples

## SSD: обучение

Функция потерь=ошибка классификации+ошибка локализации

- Pos - положительные примеры:  $i$ -ый шаблон рамки примерно соответствует  $j$ -ой реальной рамке на категории  $p$
- Neg - негативные примеры, их  $\gg \#\text{положительных}$  примеров
  - поэтому используется hard negative mining: сэмплируются отрицательные примеры с максимальной уверенностью алгоритма, чтобы  $\#neg/\#pos \approx 3$ ,

Расширение обучающей выборки:

- сэмплируются случайные фрагменты изображения, содержащие объекты
- отражения вдоль горизонтали

SSD работает точнее YOLO v1, но потом была предложена YOLO v2.

## YOLO v2<sup>8</sup>

- YOLO v2 - быстрее и точнее SSD.
- Добавлена BatchNorm в каждую свёртку. Работает в разрешении 416x416.
- Предобучается на классификаторе целевого разрешения, а не 224x224.
- Прогнозы-для шаблонных рамок (как в SSD) разных размеров и соотношений сторон,  $(x, y, w, h)$  - поправочные коэффициенты к заданным anchor boxes.
  - можно задать априорные anchor boxes, например вертикальные, если детектируем людей.

---

<sup>8</sup><https://arxiv.org/pdf/1612.08242.pdf>

## YOLO v2

- Размеры anchor boxes выбираются не вручную, а путем кластеризации реальных размеров и соотношений сторон реальных рамок в обучающей выборке.
- Поправки пропускаются через  $\sigma(\cdot)$  и  $\exp(\cdot)$  чтобы итоговые размеры принимали разумные значения:

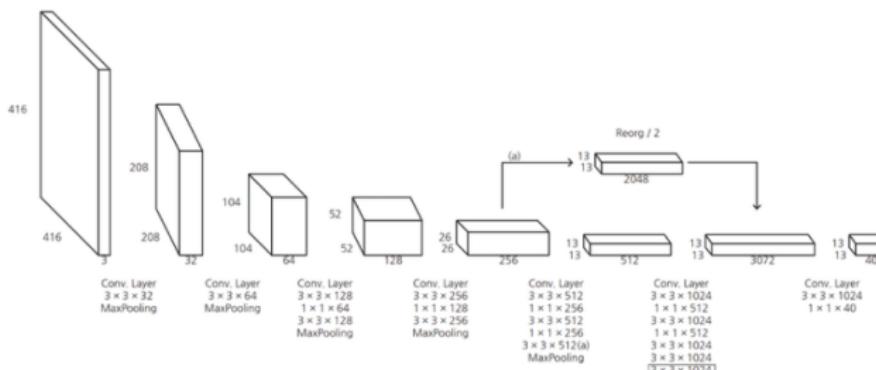
$$b_x = \sigma(t_x) + c_x \quad b_w = p_w e^{t_w}$$

$$b_y = \sigma(t_y) + c_y \quad b_h = p_h e^{t_h}$$

$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$

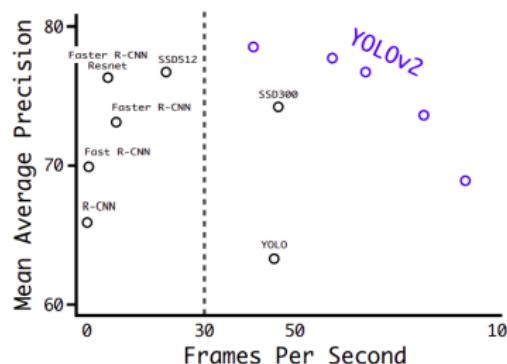
## YOLO v2

- Добавлен тождественный слой (с  $\downarrow$  разрешения), конкатенирующий выходы раннего слоя с поздним.
- Убраны полносвязные слои (предсказываем коррекции свёртками, как в SSD)  $\Rightarrow$  возможность работать со входами разного разрешения



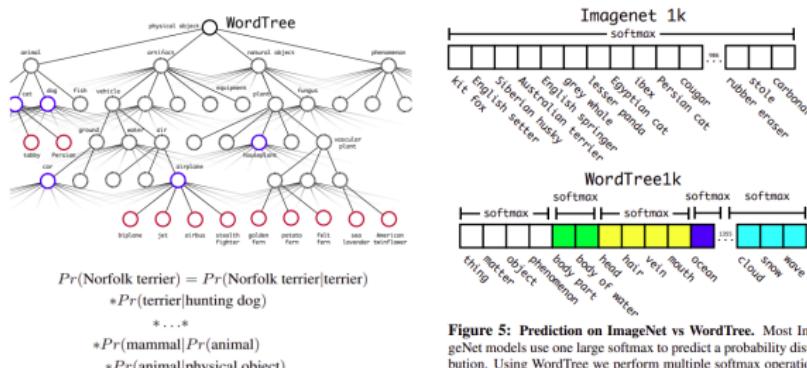
## YOLO v2

- Каждые 10 минибатчей изменяется входное разрешение объектов, чтобы сеть умела работать с разными входными разрешениями.
- в итоге получаем более кастомизируемую модель прогнозирования:
  - получаем возможность управлять противоречием скорость-точность, задавая разрешение входа



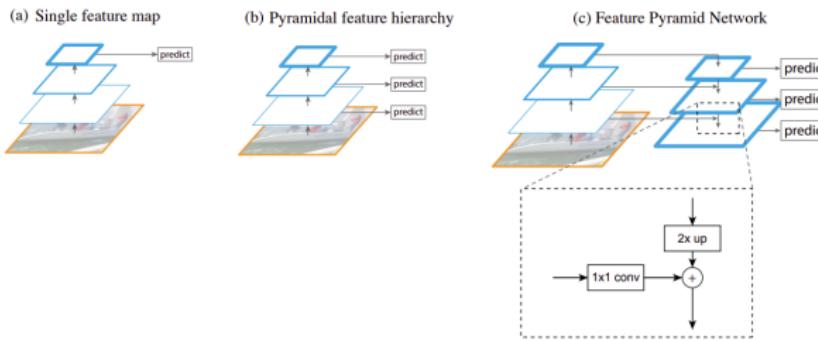
# YOLO v2

- Чтобы согласовать классы ImageNet и детекции строится дерево понятий, каждый итоговый класс - путь в этом дереве.
  - используем максимально вероятный переход, пока не дойдем до листа
- Это ↑ устойчивость классификации (модель не уверена в породе собаки, зато уверенно предсказывает, что это собака)



**Figure 5: Prediction on ImageNet vs WordTree.** Most ImageNet models use one large softmax to predict a probability distribution. Using WordTree we perform multiple softmax operations over co-hyponyms.

# Feature Pyramid Networks<sup>9</sup>

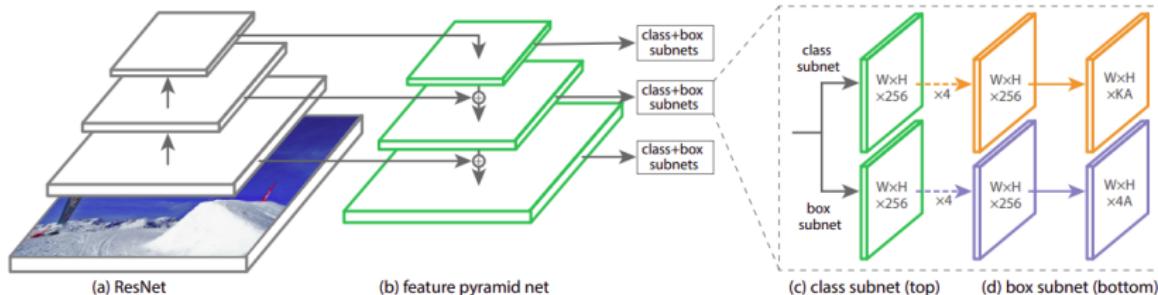


- (a)-YOLO v1, (b)-SSD
- Feature Pyramid Network (c) использует высокоуровневые признаки на каждом простр. разрешении.
- Для выравнивания #каналов перед суммированием используется conv1x1.

<sup>9</sup><https://arxiv.org/pdf/1612.03144.pdf>

# RetinaNet<sup>10</sup>

RetinaNet (точность выше) использует Feature Pyramid Network архитектуру и ResNet как CNN кодировщик:



<sup>10</sup><https://arxiv.org/pdf/1708.02002.pdf>

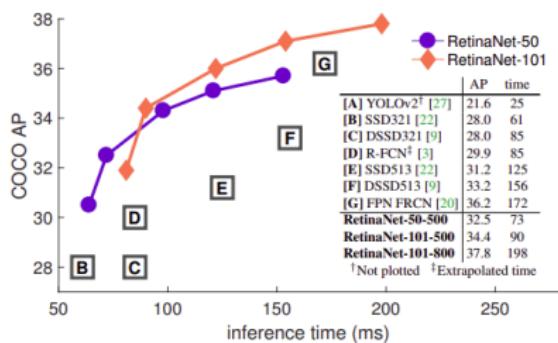
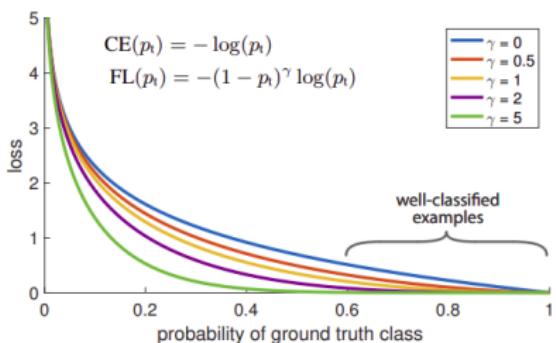
# RetinaNet

- Мотивация новой ф-ции потерь в RetineNet:
  - ↓ влияние уверенных классификаций (для фона) в большинстве случаев.
  - тогда ↑ влияние неуверенных классификаций самих объектов
- Используется новый вид потерь (Focal loss):
  - $-\log(p_t) \longrightarrow -(1 - p_t)^\gamma \log p_t$ , в работе  $\gamma = 2$
  - для уверенных классификаций  $p_t \approx 1$  и  $(1 - p_t) \approx 0$  - занижаем их вклад.
- В работе FocalLoss совмещается с взвешиванием классов (напр.  $=1/\text{частотность класса}$ )

$$-\alpha_t(1 - p_t)^\gamma \log p_t$$

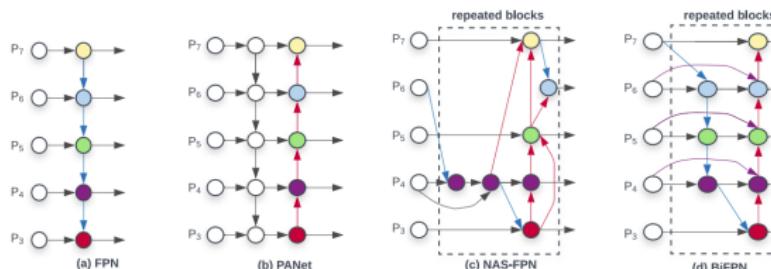
# RetinaNet

Функция потерь и ↑ качества:



# Развитие Feature Pyramid Network: EfficientDet<sup>11</sup>

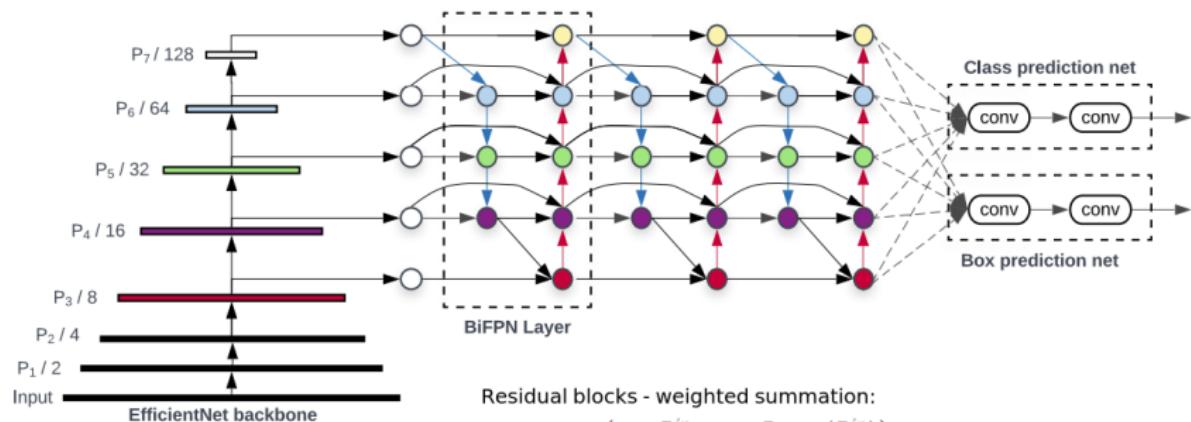
- Обозначим  $P_1, P_2, P_3, \dots$  - последовательные признаковые представления (feature maps)
- PANet - признаки, по которым строятся прогнозы, агрегируют информацию снизу-вверх и сверху-вниз.
- NAS-FPN (др. метод) - связи получены автоматическим нейросетевым перебором архитектур.
- BiFPN блок (в модели EfficientDet) - дополнительный проброс связей от исходных признаков ( $\sim$ ResNet).



<sup>11</sup><https://arxiv.org/pdf/1911.09070.pdf>

# EfficientDet

EfficientDet содержит несколько BiFPN блоков, проброс связи происходит взвешенным образом (веса настраиваются автоматически)



Residual blocks - weighted summation:

$$P_6^{td} = \text{Conv} \left( \frac{w_1 \cdot P_6^{in} + w_2 \cdot \text{Resize}(P_7^{in})}{w_1 + w_2 + \epsilon} \right)$$

$$P_6^{out} = \text{Conv} \left( \frac{w'_1 \cdot P_6^{in} + w'_2 \cdot P_6^{td} + w'_3 \cdot \text{Resize}(P_5^{out})}{w'_1 + w'_2 + w'_3 + \epsilon} \right)$$

Одностадийные детекторы (one stage, proposal free)

Методы без шаблонных рамок (anchor free)

2 Одностадийные детекторы (one stage, proposal free)

- Методы с шаблонными рамками (anchor based)
- Методы без шаблонных рамок (anchor free)

## CornerNet<sup>12</sup>

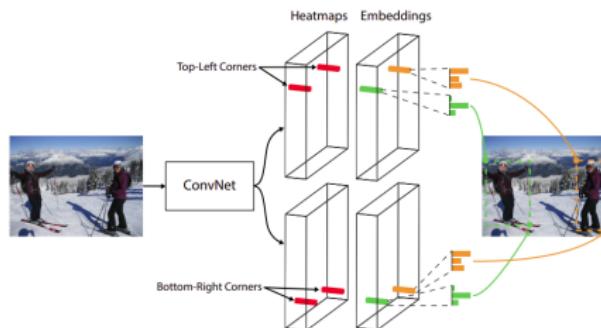
- YOLO, SSD, RetinaNet используют заданный набор anchor boxes (центры, размеры).
  - ограничивает #выделяемых объектов
- CornerNet выделяет объекты, предсказывая расположения верхнего-левого и нижнего правого угла рамки.
- Сеть не зависит от шаблонных рамок (anchor boxes), ↑ гибкость, нет ограничений на #объектов



<sup>12</sup><https://arxiv.org/pdf/1808.01244.pdf>

# CornerNet

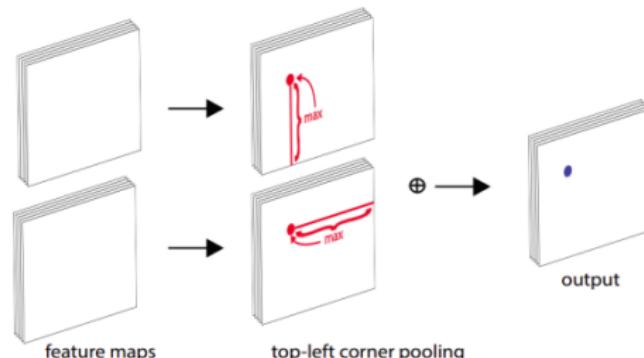
- Помимо углов каждая точка предсказывает векторное представление (embedding) объекта.
  - эмбеддинг свой для каждого объекта (на рисунке 1й человек-зеленый embedding, 2й-оранжевый)
  - углы образуют рамку, если  $\rho(embedding_1, embedding_2) < t$ .



- При обучении: для одинаковых объектов эмбеддинги должны быть близки, для различных - далеки.

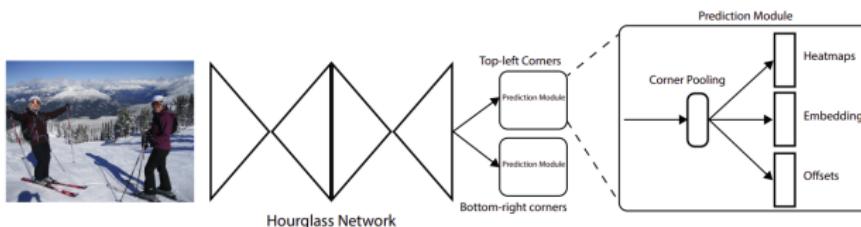
## CornerNet: pooling

- Проблема: крайние точки не попадают в объект, им сложно понять расположение рамки.
- Решение: специальный тах-пулинг
  - для ↗ угла (для ↘ угла - наоборот):
    - для y: агрегируем снизу-вверх до y
    - для x: агрегируем справа-налево до x



## CornerNet: архитектура

- Архитектура - 2 последовательных hourglass блока (сначала  $\downarrow$ , потом  $\uparrow$  разрешение с пробросом связей)  $\Rightarrow$  высокоуровневые признаки с широкой областью видимости.

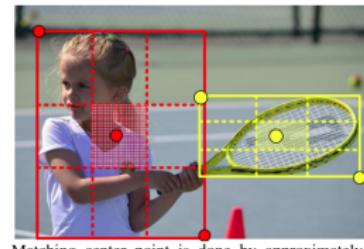
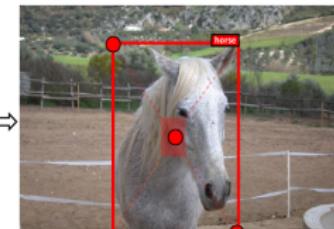
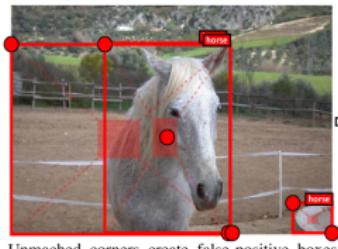


- Дополнительный выход смещений (offsets) - уточняет расположение углов рамок при экстраполяции heatmap-расположений из низкого разрешения в высокое.

$$\mathbf{o}_k = \left( \frac{x_k}{n} - \left\lfloor \frac{x_k}{n} \right\rfloor, \frac{y_k}{n} - \left\lfloor \frac{y_k}{n} \right\rfloor \right)$$

# Идея CenterNet<sup>13</sup>

- CornerPooling детектирует много ложно-положительных рамок.
  - концентрация на краях, не хватает информации изнутри рамки.
- CenterNet помимо углов детектирует центры объектов.
  - выделяем рамкой, если углам примерно соответствует центр того же класса.



<sup>13</sup><https://arxiv.org/pdf/1904.08189.pdf>

# Архитектура CenterNet

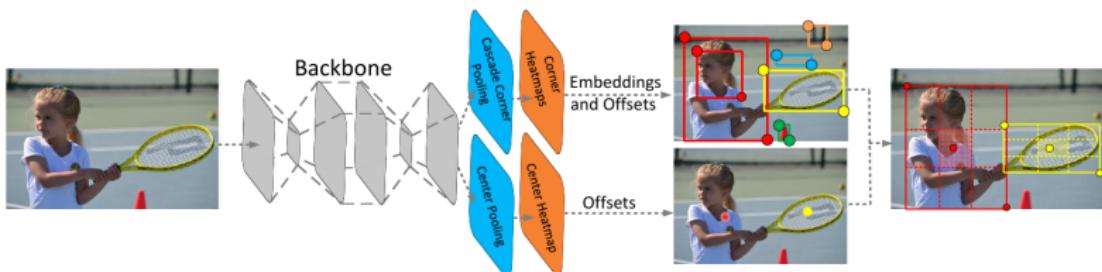
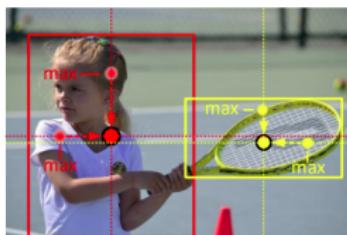


Figure 2: Architecture of CenterNet. A convolutional backbone network applies cascade corner pooling and center pooling to output two corner heatmaps and a center keypoint heatmap, respectively. Similar to CornerNet, a pair of detected corners and the similar embeddings are used to detect a potential bounding box. Then the detected center keypoints are used to determine the final bounding boxes.

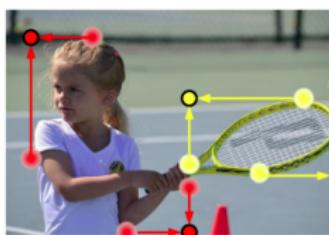
- Учёт центров CornerNet  $\uparrow$  точность.
- CenterNet - самый точный одностадийный метод детекции среди представленных на выборке MS-COCO.

## Специальный пулинг в CenterNet

- В CenterNet используется специальный пулинг для лучшей локализации:
  - CenterPooling (a) - сумма максимумов вдоль  $X$  и  $Y$ .
  - CascadeCenterPooling (c) - сумма 2x максимумов. Для верхнего левого угла:
    - максимума вдоль  $X$  вправо (достигаемого на  $X^*$ ) (b)
    - и максимума вдоль  $Y$  для  $X^*$  вниз (c)
  - Для у верхнего левого угла: максимум вниз+максимум оттуда вправо.
  - Для нижнего правого угла - наоборот.



(a)



(b)



(c)

## Заключение

- Детекция: обнаружение, классификация, выделение об-тов.
- Балансируем 2 ошибки - идентификации и локализации.
- Проблемы:
  - один и тот же объект идентифицируется многими рамками  
-> подавление не-максимумов
  - много позиций, где объект отсутствует, мало - где присутствует
    - при обучении формируем батч с заданными пропорциями случаев, hard negative mining (SSD), перевзвешивание классов
- Подходы:
  - two-stage (R-CNN, fast, faster R-CNN)
  - one-stage
    - anchor-based: размеры неизвестны (YOLO), размеры-уточнения шаблонных (SSD, YOLO v2)
    - anchor-free (CornerNet, CenterNet)
- Прогнозы: на последнем слое, на каждом слое отдельно, на композиции разных слоёв.