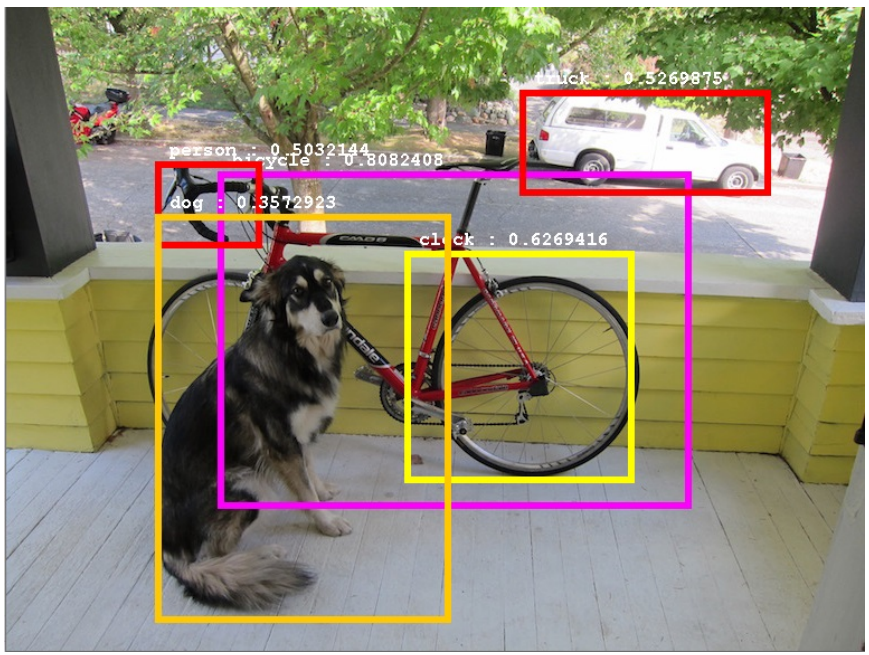
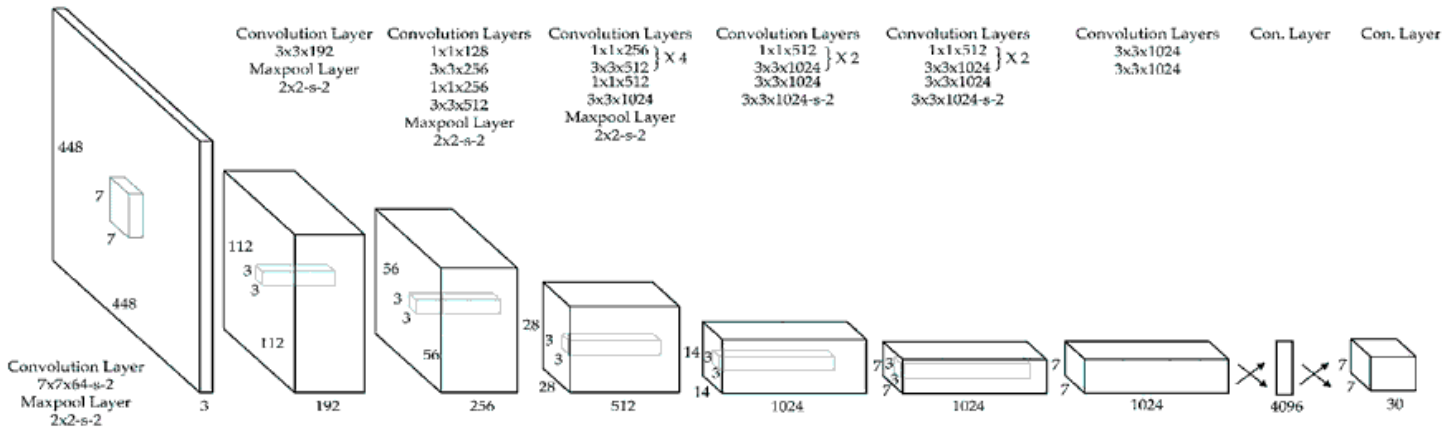
Одностадийная детекция объектов

Виктор Китов
victorkitov.github.io



YOLO[1]

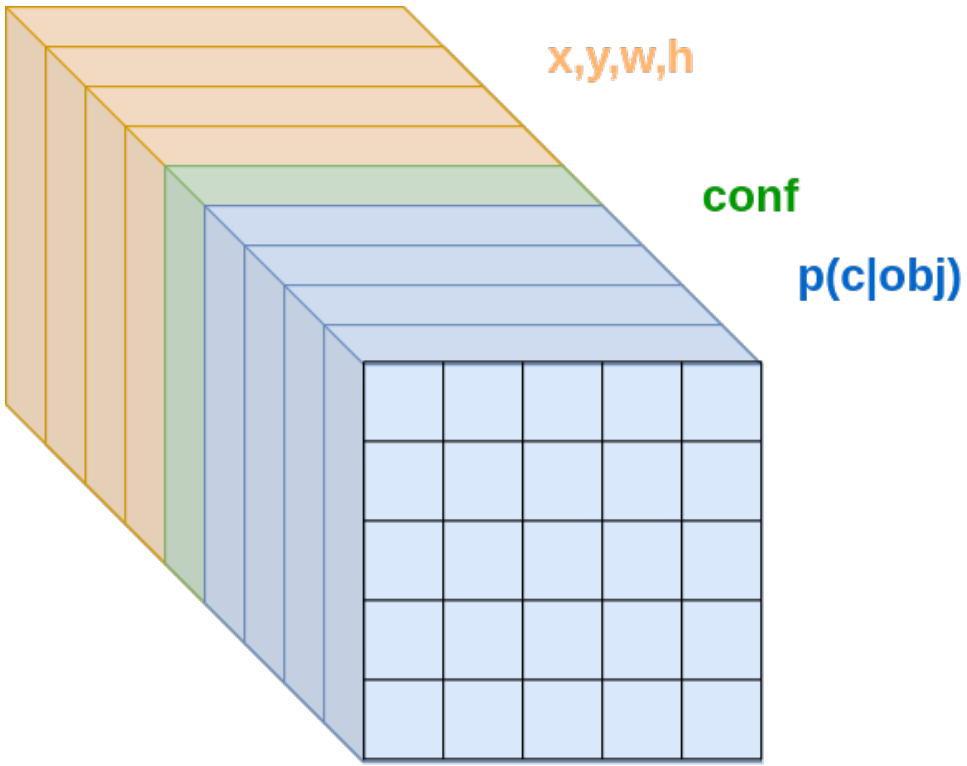
- YOLO (You look only once) одностадийный детектор
  - сразу предсказывает выход, без этапа предсказания регионов-кандидатов (proposal free, one stage)



- Вход 448x448. ↓ разрешения: stride=2.
- Первая свёртка 7x7, затем 3x3 (с conv 1x1 для ↓ числа каналов).
- 2 FC слоя в конце, DropOut после 1го.
- Активации LeakyReLU.

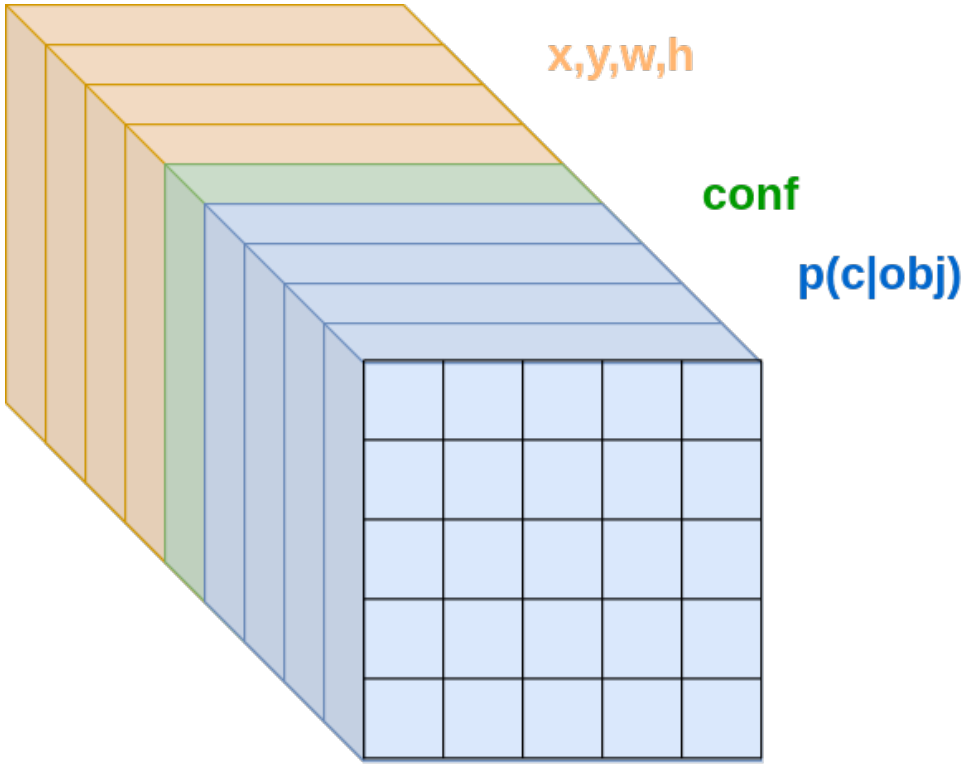
[1] https://arxiv.org/pdf/1506.02640.pdf

YOLO: выход



- Выход - фикс. тензор $S \times S \times (B * 5 + C)$,
  - SxS пространственное разрешение (в статье 7x7)
  - C - #классов
  - B - #предсказываемых рамок в каждой ячейке
    - $B > 1$ для $\uparrow$ recall извлекаемых рамок, в статье $B = 2$.

YOLO: выход



- Выход - фикс. тензор $S \times S \times (B * 5 + C)$, в каждом элементе предсказываются
  - B раз:
    - $conf = p(obj) \cdot IoU_{true}^{pred}$ - степень присутствия объекта (любого класса)
    - (x, y) - координаты центра выделяющей рамки (отн.

Настройка YOLO

- Настройка по сумме 3-х ф-ций потерь:
  - точность определения, что объект присутствует
  - точность определения класса объекта
  - точность локализации объекта

Точность обнаружения

- Точность обнаружения:
  - $(1 - conf)^2$ - если объект есть
  - $(0 - conf)^2$ - если объекта нет
    - учитывалось с меньшим весом, т.к. случаев отсутствия объекта гораздо больше
- Среди B предсказаний присутствия объекта объект считался присутствующим только для 1 предсказания с максимальным IoU с истинным выделением.

Точность локализации

Точность локализации:

$$(\hat{x}-x)^2+(\hat{y}-y)^2+(\sqrt{\hat{w}}-\sqrt{w})^2+(\sqrt{\hat{h}}-\sqrt{h})^2$$

- $\sqrt{\cdot}$ для ширины высоты, чтобы повысить внимание модели к точности локализации малых объектов.
- для больших объектов та же величина ошибки меньше важна.

YOLO: выход

- class probabiliy (conditional):
  $p\left(y=1|obj\right), p\left(y=2|obj\right), ...p\left(y=C|obj\right)$
  - B раз, чтобы ↑recall, (но все разы для 1го класса)
    - x, y, w, h - относит. координаты рамки
    - obj_score = $p(object) \cdot IoU_{pred}^{truth}$ - степень присутствия объекта (0, если отсутствует)

YOLO: функция потерь

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

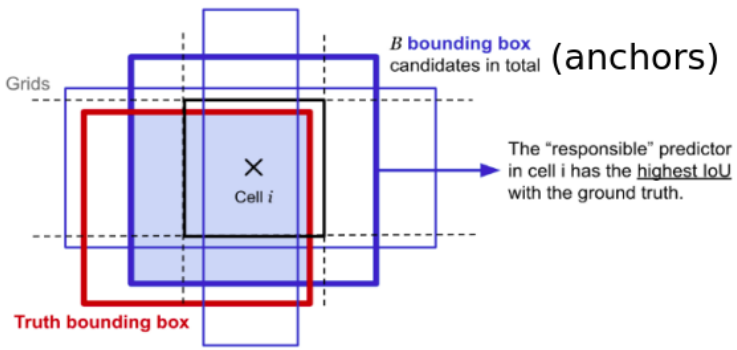
$$+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right]$$

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

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

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

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



- Потери классификации - MSE (в более поздних версиях - cross-entropy).

Анализ

- Особенности обучения:
  - Претренировка первых слоев на ImageNet-классификации. 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.
- Во время применения
  - для каждой ячейки предсказывались рамки, на которых

$$p(class_i|obj)p(obj)IoU^{true}_{pred} = p(class_i)IoU^{true}_{pred} > threshold$$

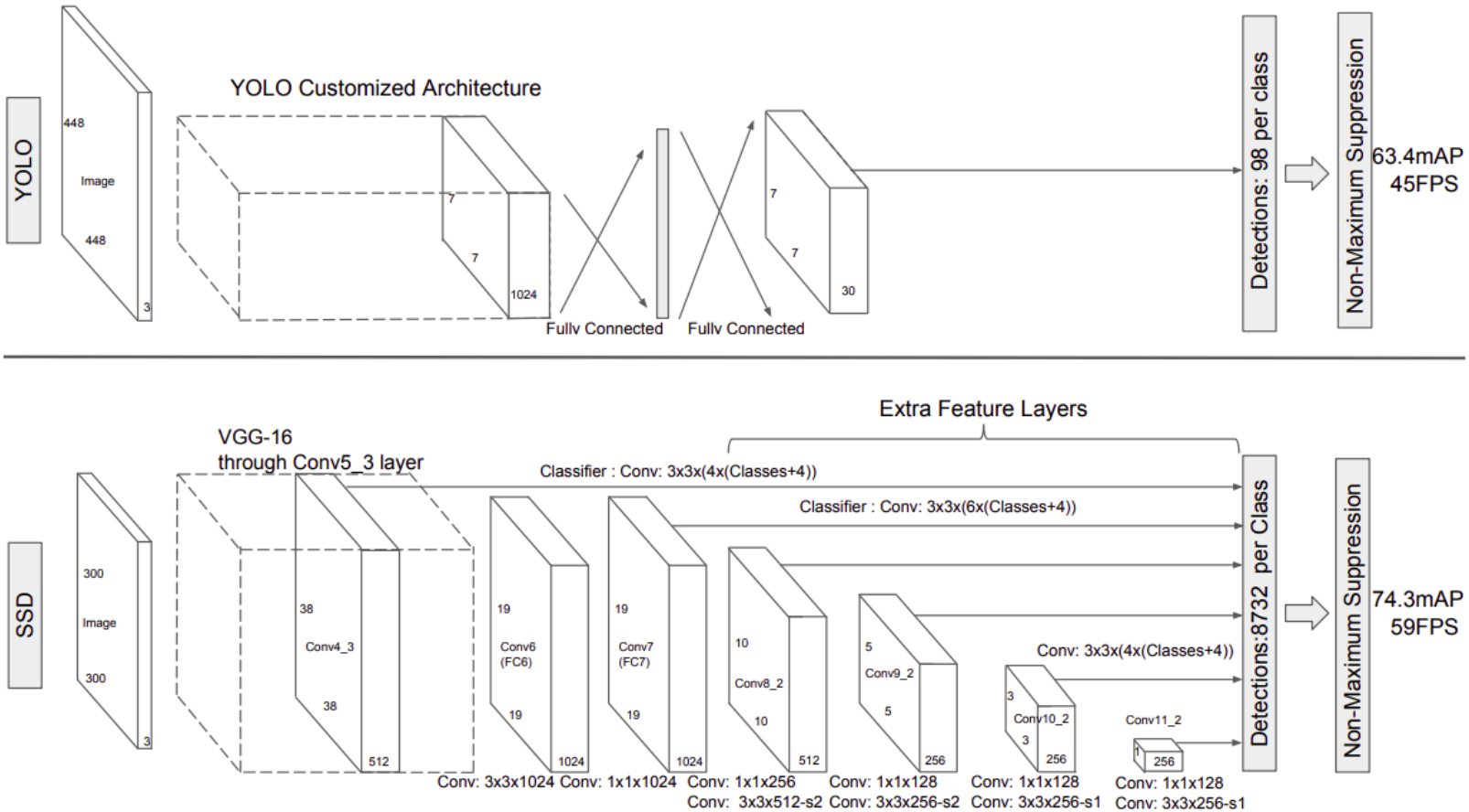
  - применялось подавление не-максимумов

Недостатки YOLO

- Недостаток YOLOv1:
  - #обнаружений $\leq S \times S$=49
  - не может извлекать малые объекты (теряются в грубой пространств. сетке)
- Новые версии YOLO исправляли эти недостатки.
- SSD (Single Shot MultiBox Detector[2]) решает проблемы YOLOv1:
  - за счёт детекторов, применяемых к разным простр. разрешениям
  - в конце - подавление не-максимумов (убираем дубликаты)

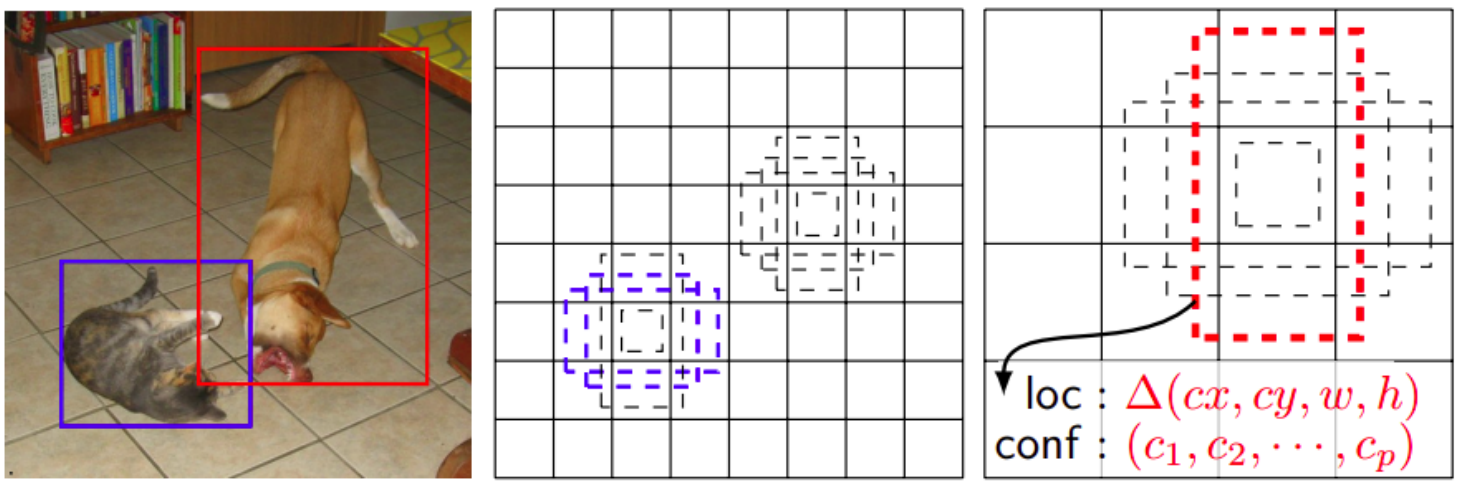
[2] https://arxiv.org/pdf/1512.02325.pdf

Сравнение архитектур YOLO и SSD



SSD

- На каждом слое для каждой позиции, используя conv 3x3, предсказываем K раз (для разных anchor boxes): $(dx, dy, w, h, p(class_1), ...p(class_C))$
  - 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) 8×8 feature map (c) 4×4 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 α 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.

$$\mathcal{L}_{\text{loc}} = \sum_{i,j} \sum_{m\in\{x,y,w,h\}} \mathbb{1}_{ij}^{\text{match}} L_1^{\text{smooth}}(d_m^i - t_m^j)^2$$

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

$$t_x^j = (g_x^j - p_x^i)/p_w^i \quad t_w^j = \log(g_w^j/p_w^i)$$
$$t_y^j = (g_y^j - p_y^i)/p_h^i \quad t_h^j = \log(g_h^j/p_h^i)$$

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

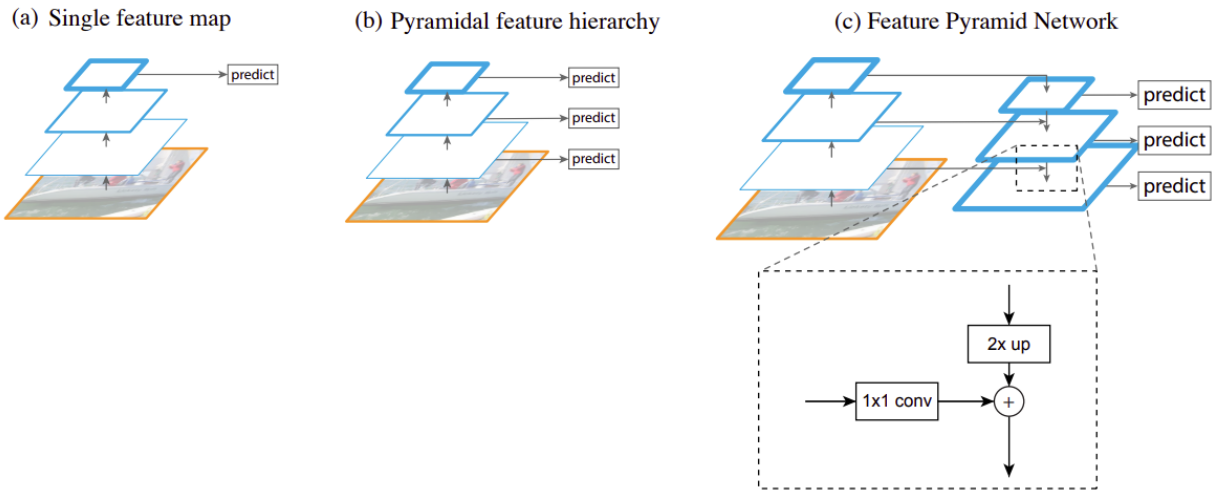
- Pos - положительные примеры: i-ый шаблон рамки примерно соотвествует j-ой реальной рамке на категории p
- Neg - негативные примеры, их $\gg$ #положительных примеров
  - поэтому используется hard negative mining: сэмплируются отрицательные примеры с максимальной уверенностью алгоритма, чтобы $\#neg/\#pos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

Расширение обучающей выборки:

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

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

Feature Pyramid Networks[3]

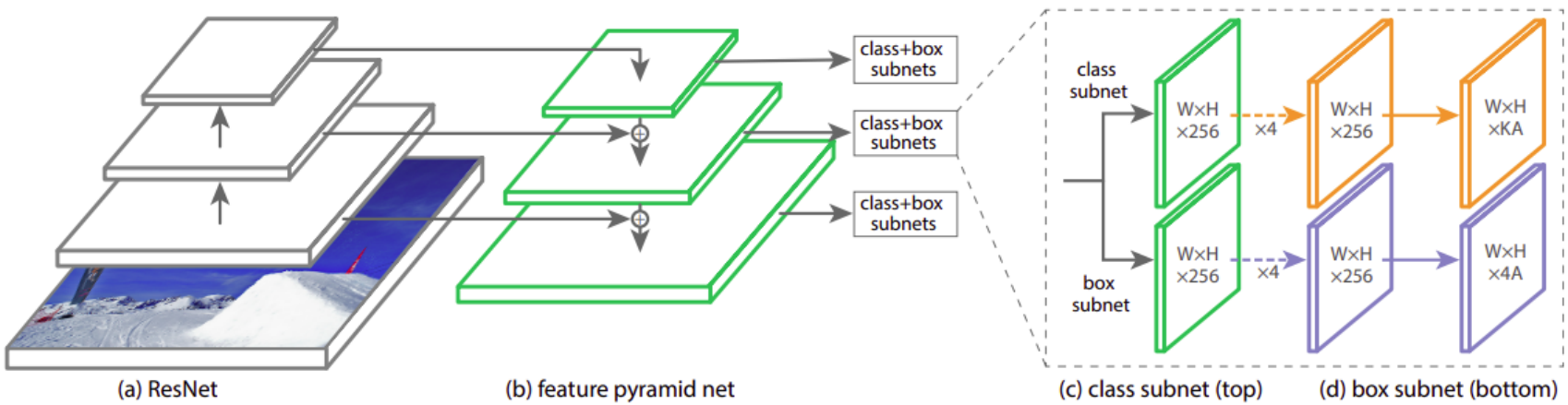


- (a)-YOLO v1, (b)-SSD
- Feature Pyramid Network (c) использует высокоуровневые признаки на каждом простр. разрешении.
- Для выравнивания #каналов перед суммированием используется conv1x1.

[3] https://arxiv.org/pdf/1612.03144.pdf

RetinaNet[4]

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



[4] https://arxiv.org/pdf/1708.02002.pdf

RetinaNet

- Мотивация новой ф-ции потерь в RetineNet:
  - $\downarrow$ влияние уверенных классификаций (для фона) в большинстве случаев.
  - тогда $\uparrow$ влияние неуверенных классификаций самих объектов
- Используется новый вид потерь (Focal loss):

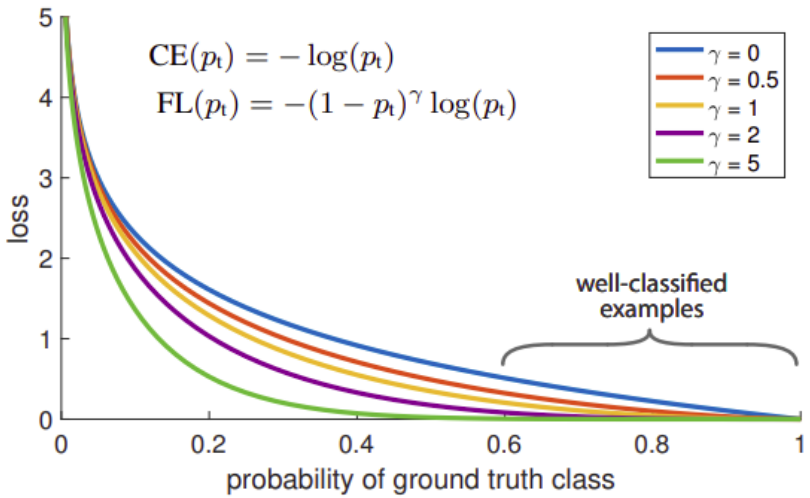
$$-\log(p_t) \longrightarrow -(1-p_t)^{\gamma}\log p_t, \quad \text{в работе } \gamma = 2$$

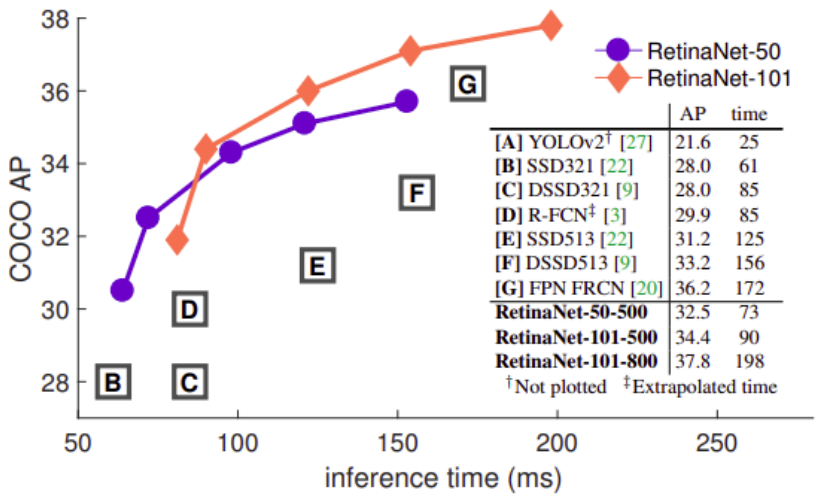
  - для уверенных классификаций $p_t \approx 1$ и $(1-p_t) \approx 0$ - занижаем их вклад.
- В работе FocalLoss совмещается с взвешиванием классов (напр. =1/частотность класса)

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

RetinaNet

Функция потерь и ↑ качества:



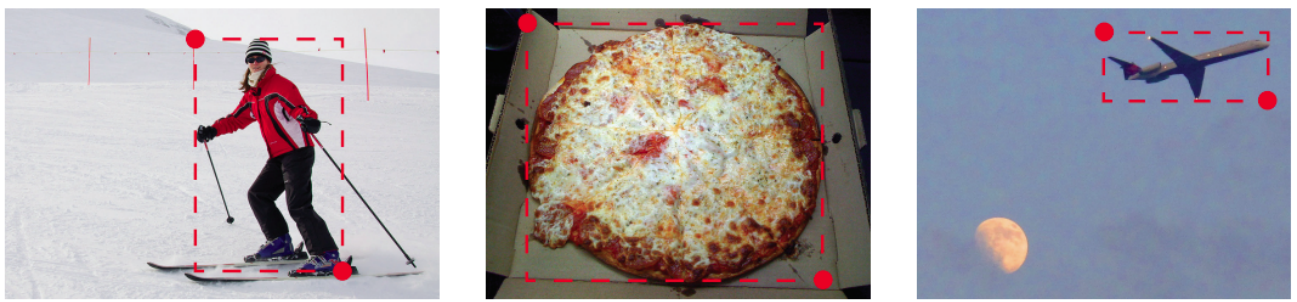


	AP	time
[A] YOLOv2† [27]	21.6	25
[B] SSD321 [22]	28.0	61
[C] DSSD321 [9]	28.0	85
[D] R-FCN‡ [3]	29.9	85
[E] SSD513 [22]	31.2	125
[F] DSSD513 [9]	33.2	156
[G] FPN FRCN [20]	36.2	172
RetinaNet-50-500	32.5	73
RetinaNet-101-500	34.4	90
RetinaNet-101-800	37.8	198

†Not plotted ‡Extrapolated time

CornerNet[5]

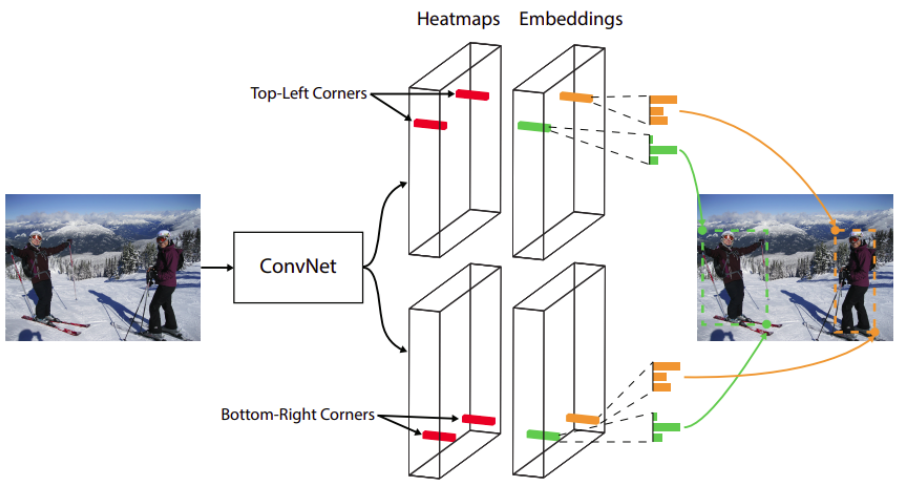
- YOLO, SSD, RetinaNet используют заданный набор anchor boxes (центры, размеры).
  - ограничивает #выделяемых объектов
- CornerNet выделяет объекты, предсказывая расположения верхнего-левого и нижнего правого угла рамки.
- Сеть не зависит от шаблонных рамок (anchor boxes), ↑ гибкость, нет ограничений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



[5] https://arxiv.org/pdf/1808.01244.pdf

CornerNet

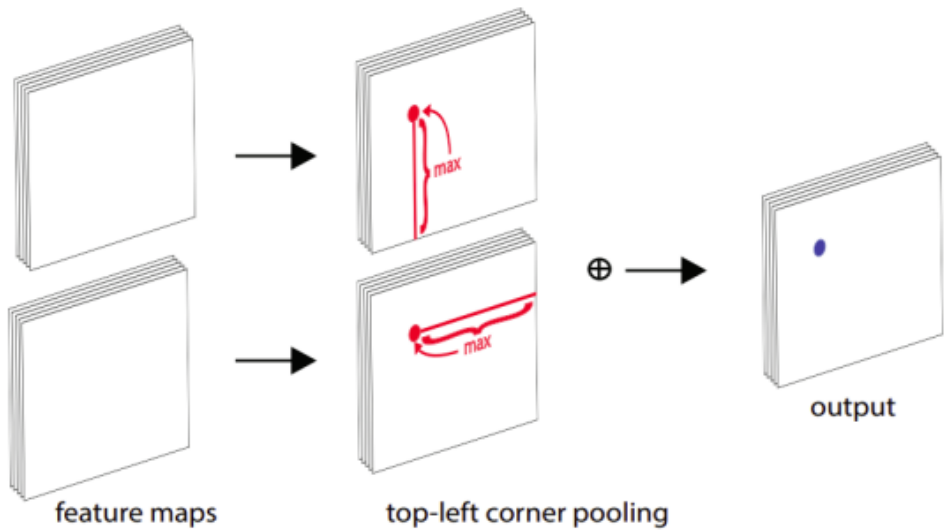
- Помимо углов каждая точка предсказывает векторное представление (embedding) объекта.
  - эмбеддинг свой для каждого объекта (на рисунке 1й человек-зеленый embedding, 2й-оранжевый)
  - углы образуют рамку, если $\rho\left(embedding_1, embedding_2\right) < t$.



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

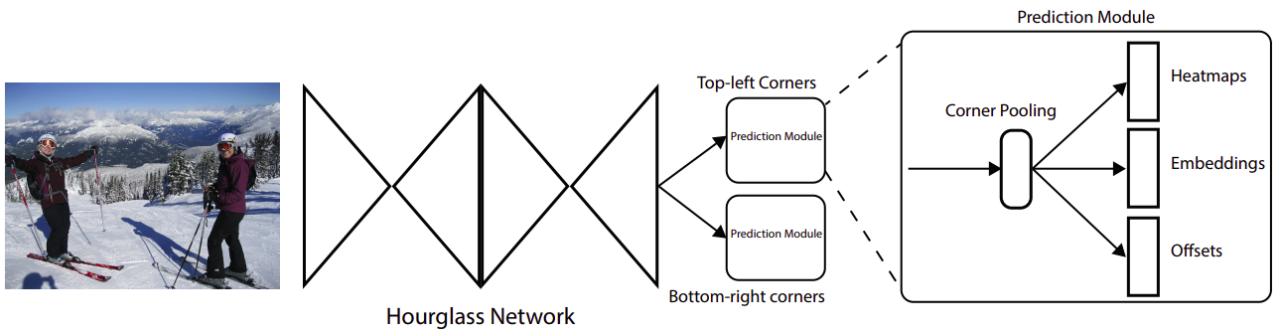
CornerNet: pooling

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



CornerNet: архитектура

- Архитектура - 2 последовательных hourglass блока (сначала ↓, потом ↑ разрешение с пробросом связей) => высокоуровневые признаки с широкой областью видимости.

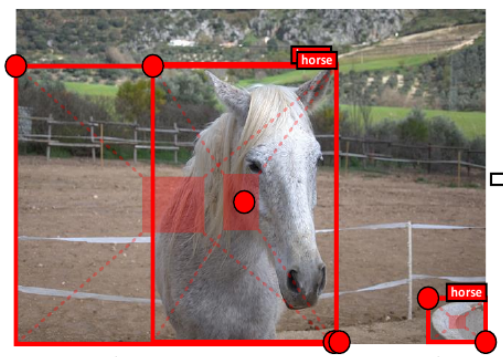


- Дополнительный выход смещений (offsets) - уточняет расположение углов рамок при экстраполяции heatmap-расположений из низкого разрешения в высокое.

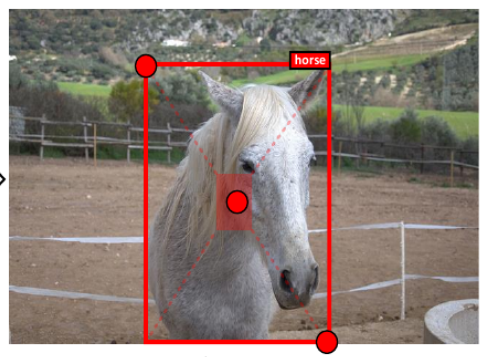
$$\boldsymbol{o}_k = \left(\frac{x_k}{n} - \left\lfloor \frac{x_k}{n} \right\rfloor, \frac{y_k}{n} - \left\lfloor \frac{y_k}{n} \right\rfloor \right)$$

Идея CenterNet[6]

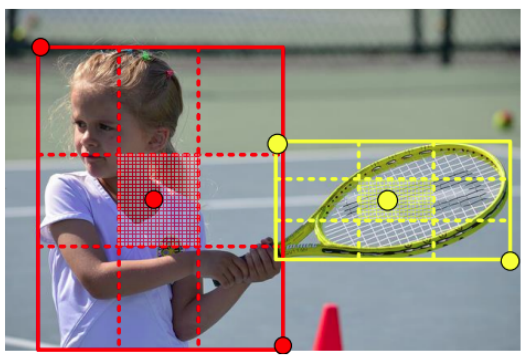
- CornerPooling детектирует много ложно-положительных рамок.
  - концентрация на краях, не хватает информации изнутри рамки.
- CenterNet помимо углов детектирует центры объектов.
  - выделяем рамкой, если углам примерно соответствует центр того же класса.



Unmached corners create false-positive boxes.



Matching centers decreases false-positive rate.



Matching center point is done by approximately.

[6] https://arxiv.org/pdf/1904.08189.pdf

Архитектура CenterNet

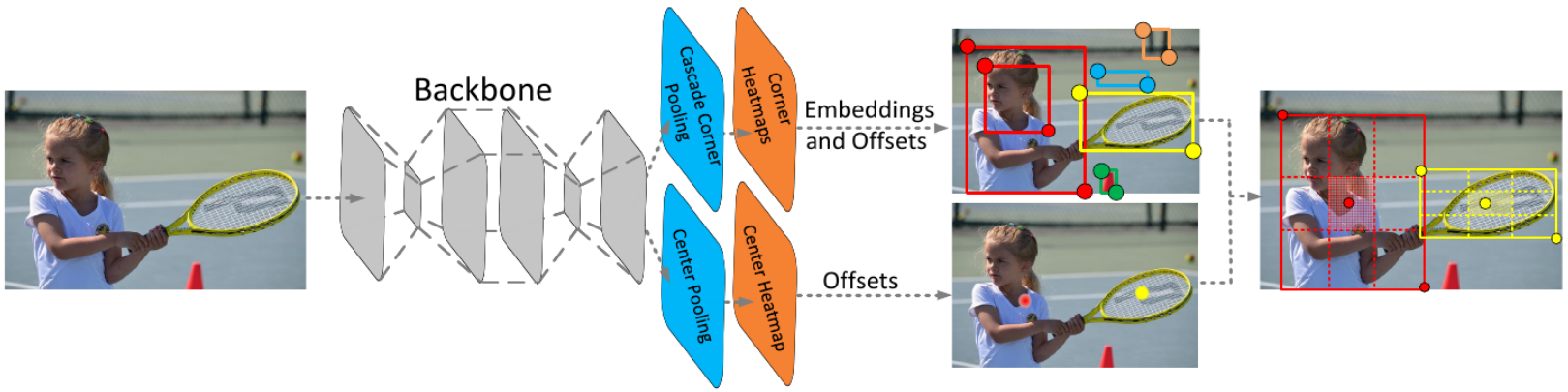
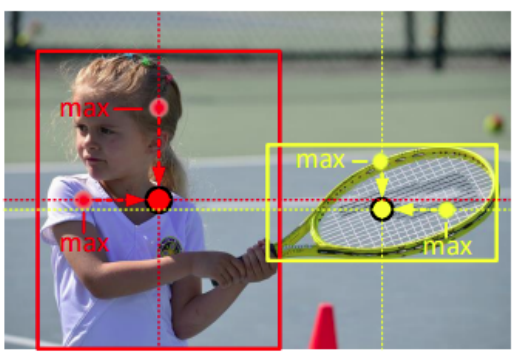


Figure 2: Architecture of CenterNet. A convolutional backbone network applies cascade corner pooling and center pooling to output two corner heatmaps and a center keypoint heatmap, respectively. Similar to CornerNet, a pair of detected corners and the similar embeddings are used to detect a potential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

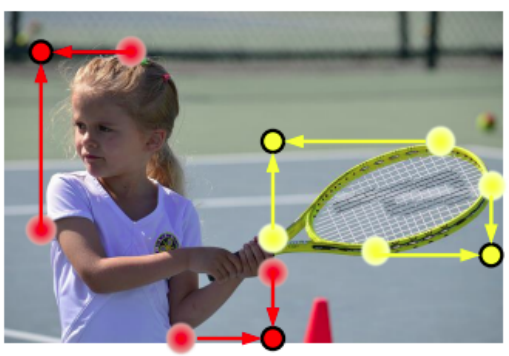
- Учёт центров CornerNet ↑ точность.
- CenterNet - самый точный одностадийный метод детекции среди представленных на выборке MS-COCO.

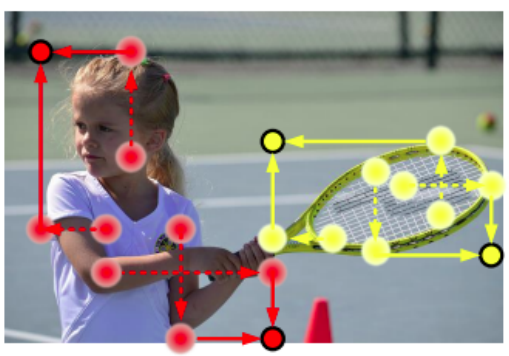
Специальный пулинг в CenterNet

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



(a)


(b)


(c)

Заключение

- Одностадийные детекторы детектируют объекты за один проход по нейросети.
- YOLO: прогнозы по последнему слою.
- SSD: прогнозы по разным слоям разной силы.
- FPN: слои признаков одинаковой выразительной силы.
- RetinaNet: $\downarrow$ вклада прогнозов, в которых сильно уверены (фон).
- CornerNet, CenterNet предсказывают рамки через тепловые карты.
  - специальные пулинги для извлечения признаков
  - сопоставление - по близости эмбеддингов