Лекция 5. Задачи компьютерного зрения и сверточные нейронные сети

Бабич Павел Васильевич

ИММиКН

2026

Детекция среди задач компьютерного зрения

Задача	Выход модели	
Классификация	Метка класса изображения	
Локализация	Класс и рамка одного объекта	
Детекция	Набор рамок, классов и оценок уверенности	Детектор
Семантическая сегментация	Класс каждого пикселя	
Сегментация экземпляров	Отдельная маска каждого объекта	

должен распознавать **несколько экземпляров одного класса** и определять их положение.

Формальная постановка

Изображение $x \in \mathbb{R}^{H \times W \times 3}$ и его разметка:

$$Y = \{(b_j, c_j)\}_{j=1}^{N}, \qquad b_j = (x_j^{\min}, y_j^{\min}, x_j^{\max}, y_j^{\max}), \quad c_j \in \{1, \ldots, C\}.$$

Модель с параметрами θ предсказывает:

$$f_\theta(x) = \widehat{Y} = \{(\widehat{b}_i, \widehat{c}_i, s_i)\}_{i=1}^{M}, \qquad s_i \in [0, 1].$$

- Число объектов N неизвестно и может быть нулевым.
- Порядок элементов множества несущественен.
- s_i служит для ранжирования. Это не обязательно калиброванная вероятность.

Геометрия и основные трудности

Две эквивалентные параметризации рамки:

$$w = x^{\max} - x^{\min}, \quad h = y^{\max} - y^{\min}, \quad x_c = \frac{x^{\min} + x^{\max}}{2}, \quad y_c = \frac{y^{\min} + y^{\max}}{2}.$$

Для непрерывных координат площадь равна $|b| = wh$.

- Масштаб: маленький объект занимает мало ячеек карты признаков.
- Окклюзия и плотные сцены: объекты перекрывают друг друга.
- Фон: отрицательных кандидатов гораздо больше, чем положительных.
- Геометрия: точное положение границ важно наряду с классом.

При изменении размера изображения преобразуют и координаты разметки.

Intersection over Union

Для предсказанной рамки A и истинной рамки B:

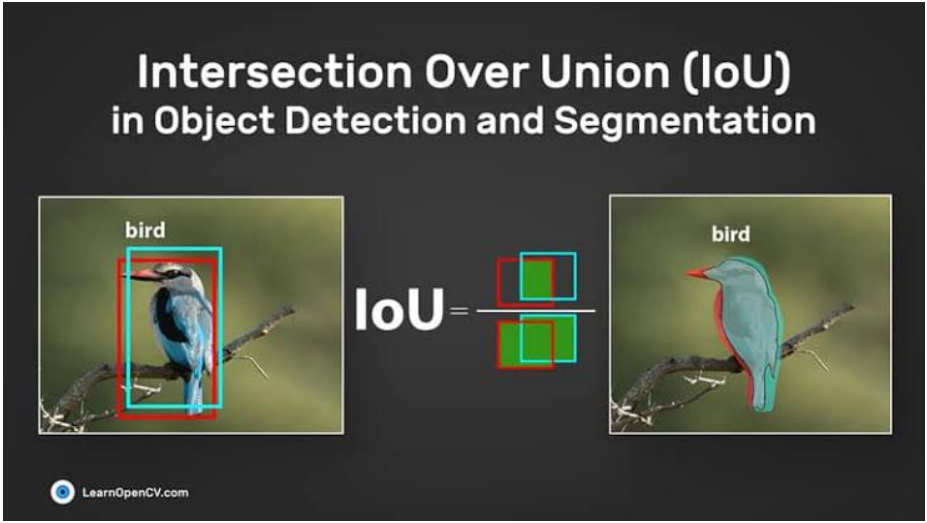
$$\boxed{(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}} \in [0,1].$$

$$\begin{aligned} w_{\cap} &= \max\bigl(0, \min(x_A^{\max}, x_B^{\max}) - \max(x_A^{\min}, x_B^{\min})\bigr), \\ h_{\cap} &= \max\bigl(0, \min(y_A^{\max}, y_B^{\max}) - \max(y_A^{\min}, y_B^{\min})\bigr), \\ |A \cap B| &= w_{\cap} h_{\cap}. \end{aligned}$$

Пример: $|A| = |B| = 100$, $|A \cap B| = 60$:

$$= \frac{60}{100 + 100 - 60} = \frac{3}{7} \approx 0.429.$$

Intersection over Union



Сопоставление предсказаний с разметкой

Фиксируем класс c и порог совпадения τ_{IoU}.

1. В каждом изображении сортируем предсказания по убыванию s_i.
2. Для очередного предсказания ищем ещё не сопоставленную истинную рамку того же класса с максимальным IoU.
3. Если $\geq \tau_{\mathrm{IoU}}$, считаем предсказание **TP** и занимаем рамку. Иначе получаем **FP**.
4. Несопоставленные истинные объекты дают **FN**.

Второе предсказание одного объекта обычно становится FP.

Precision, recall и F_β

При фиксированных порогах уверенности q и IoU τ:

$$P(q;\tau) = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}, \qquad R(q;\tau) = \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}.$$

$$F_\beta = (1+\beta^2)\frac{PR}{\beta^2 P + R}, \qquad F_1 = \frac{2\mathrm{TP}}{2\mathrm{TP}+\mathrm{FP}+\mathrm{FN}}.$$

- Precision: какая доля выданных рамок правильная.
- Recall: какая доля истинных объектов найдена.
- $\beta > 1$ повышает относительную значимость полноты.

При нулевом знаменателе нужна явная договорённость. Число TN для детекции обычно не определяют.

Intersection over Union



PR-кривая и Average Precision

Для одного класса объединяем предсказания **всех изображений** и сортируем по уверенности. Пусть $z_k \in \{0, 1\}$ отмечает TP:

$$P_k = \frac{\sum_{i=1}^{k} z_i}{k}, \qquad R_k = \frac{\sum_{i=1}^{k} z_i}{N_{\mathrm{GT}}}.$$

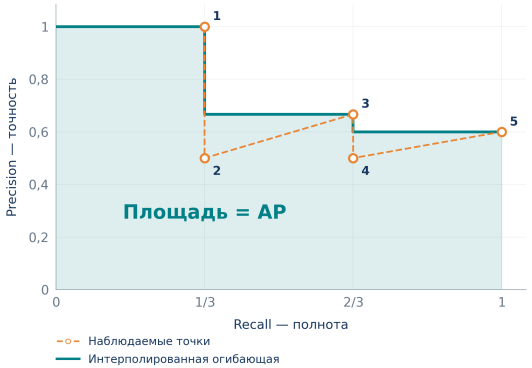
Интерполированная точность и площадь под её огибающей:

$$P_{\mathrm{interp}}(r) = \max_{k:R_k \geq r} P_k, \qquad = \int_0^1 P_{\mathrm{interp}}(r)\, dr.$$

Если достижимого $R_k \geq r$ нет, полагаем $P_{\mathrm{interp}}(r) = 0$. AP оценивает **ранжирование**, а F_1 оценивает одну рабочую точку. В бенчмарках интеграл заменяют конкретным правилом дискретизации.

PR-кривая и Average Precision

Один класс · 3 истинных объекта · фиксированный порог IoU



Как строится кривая

Предсказания по убыванию score:

TP	FP	TP	FP	TP
0,95	0,90	0,80	0,70	0,60

$$P_{\text{interp}}(r) = \max_{k:R_k \geq r} P_k$$

$$\text{AP} = \int_0^1 P_{\text{interp}}(r)\,dr$$

$$= \frac{1}{3}\cdot 1 + \frac{1}{3}\cdot\frac{2}{3} + \frac{1}{3}\cdot\frac{3}{5}$$

$$= \frac{34}{45} \approx 0{,}756$$

FP снижает precision без роста recall.
TP увеличивает recall.

Учебный пример: AP как интеграл по огибающей. В COCO используется усреднение на 101 уровне recall.

mAP

Среднее по классам при фиксированном IoU:

$$\mathrm{mAP}_\tau = \frac{1}{|\mathcal{C}|} \sum_{c \in \mathcal{C}} {}_{c,\tau}.$$

В COCO используют $\mathcal{T} = \{0.50, 0.55, \ldots, 0.95\}$ и 101 уровень recall:

$$\overset{101}{}_{c,\tau} = \frac{1}{101} \sum_{r \in \{0,.01,\ldots,1\}} P^{c,\tau}_{\mathrm{interp}}(r), \qquad \boxed{\mathrm{COCO} = \frac{1}{10|\mathcal{C}|} \sum_{\tau \in \mathcal{T}} \sum_{c \in \mathcal{C}}^{101} {}_{c,\tau}}.$$

- AP50 и AP75: отдельные пороги $\tau = 0.50$ и 0.75.
- Стандартная bbox AP использует максимум 100 детекций на изображение.
- $\mathcal{C}$: категории с учитываемой разметкой в выбранном срезе.

MS COCO 2017: данные и разметка

Common Objects in Context: объекты в повседневных сценах.
80 классов детекции, например person, car, cat, chair.

Количество изображений

Выборка	Изображений
Train	118 287
Validation	5 000
Test	40 670

Разметка train/val доступна. GT для test скрыта, оценку выполняет сервер.

Дополнительные задачи: маски экземпляров, ключевые точки людей, описания изображений.

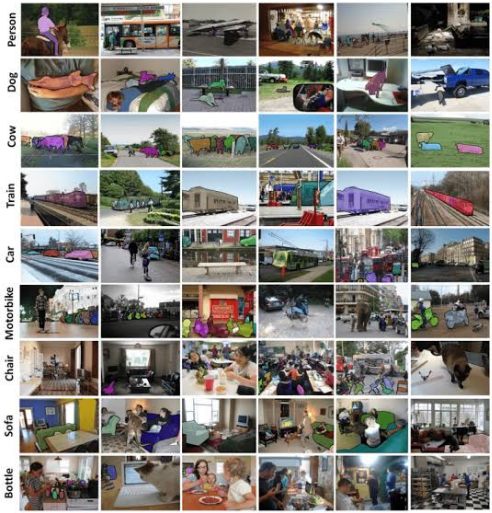
MS COCO 2017: данные и разметка

Пример разметки MS COCO

```
"annotations": [
  {
    "id": 1,
    "image_id": 1,
    "category_id": 1,
    "bbox": [120, 150, 200, 180],
    "area": 36000,
    "segmentation": [[120, 150, 320, 150, 320, 330, 120, 330]],
    "iscrowd": 0
  }
  ...
]
```

В MS COCO ‘bbox‘ в формате ‘xywh‘, segmentation в формате ‘xyxyxy‘.

MS COCO



Масштаб объектов, recall и скорость

- AP_S, AP_M, AP_L: качество на малых, средних и больших объектах. Границы площади COCO: 32^2 и 96^2 пикселей.
- AR_K: усреднённая по классам и порогам IoU полнота при ограничении числа предсказаний K.

$$\kappa = \frac{1}{10|\mathcal{C}|} \sum_{\tau \in \mathcal{T}} \sum_{c \in \mathcal{C}} R_{c,\tau,K}, \qquad K \in \{1, 10, 100\}.$$

Общие компоненты детектора

Компонент	Назначение
Backbone	CNN, например ResNet: извлечение карт признаков
Neck	Объединение масштабов и уровней, например FPN
Head	Классы, координаты и при необходимости objectness
Assignment	Назначение обучающих целей кандидатам
Postprocessing	Декодирование рамок, фильтрация и обычно NMS

$$\{(\widehat{b}_i, \widehat{p}_i)\}_i = h_\theta(n_\theta(\phi_\theta(x)))\,.$$

Эти компоненты можно менять независимо. FPN совместима с одностадийными и двухстадийными детекторами.

Selective Search: основная идея

Selective Search — алгоритм генерации кандидатов областей (*region proposals*) для детекции объектов.

Вместо полного перебора всех возможных bounding boxes:

1. Выполняется начальная сегментация изображения на небольшие регионы.
2. Для соседних регионов вычисляется мера сходства.
3. Наиболее похожие регионы объединяются.
4. Процесс повторяется и формирует иерархию регионов.
5. Bounding boxes полученных регионов используются как кандидаты объектов.

$$\text{Image} \rightarrow \text{Segmentation} \rightarrow \text{Region Merging} \rightarrow \text{Object Proposals}$$

Selective Search: сходство регионов

Для соседних регионов r_i и r_j вычисляется суммарная мера сходства:

$$S(r_i, r_j) = S_{\text{color}} + S_{\text{texture}} + S_{\text{size}} + S_{\text{fill}}.$$

$$S_i(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k)$$

- c_i^k, c_j^k — значение зерен гистрограм для регионов i и j.
- **Color similarity** — близость цветовых гистограмм.
- **Texture similarity** — близость текстурных признаков.
- **Size similarity** — предпочтение небольших регионов.
- **Fill similarity** — насколько компактно регионы заполняют общий bounding box.

На каждой итерации объединяется наиболее похожая пара:

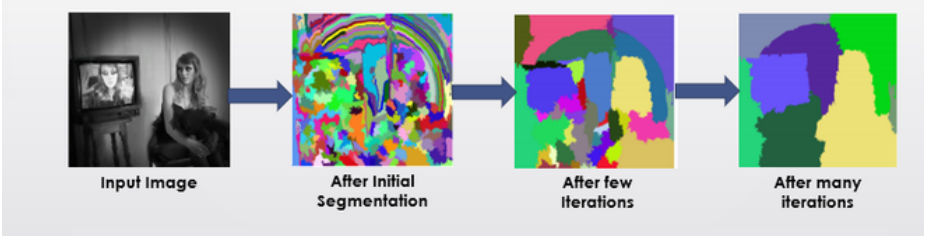
$$(r_i, r_j) = \arg \max_{(r_p, r_q)} S(r_p, r_q).$$

Selective Search

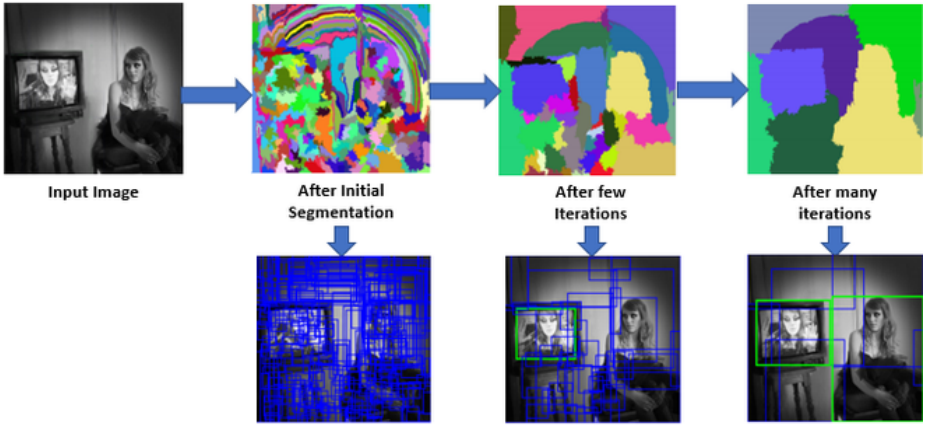




Selective Search



Selective Search



R-CNN: основная идея

Алгоритм состоит из трех основных этапов:

1. **Region proposals:** Selective Search генерирует около 2000 областей-кандидатов.
2. **Feature extraction:** каждая область приводится к фиксированному размеру и независимо пропускается через CNN.
3. **Classification:** полученные признаки используются для определения класса объекта и уточнения bounding box.

$$\text{Image} \rightarrow \text{Region Proposals} \rightarrow \text{CNN} \rightarrow \text{Classification} + \text{BBox}$$

R-CNN: обучение и ограничения

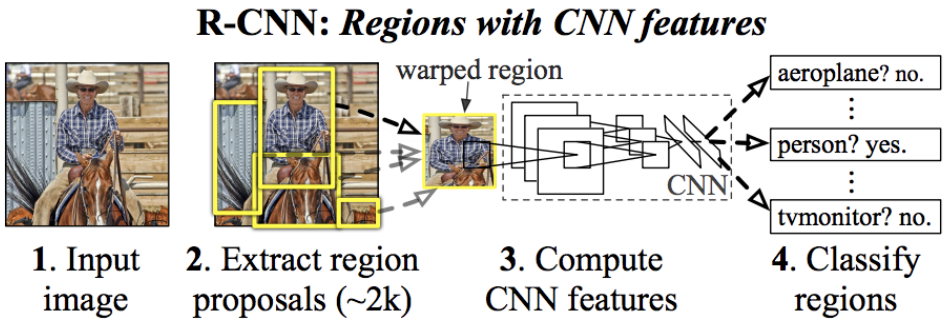
Для каждого region proposal CNN извлекает вектор признаков:

$$\mathbf{x}_i = \mathrm{CNN}(r_i).$$

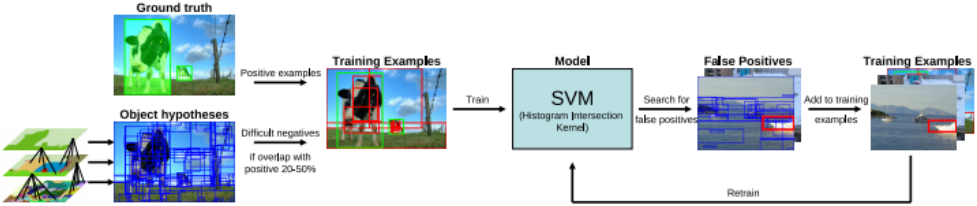
Преимущество: R-CNN значительно улучшил качество детекции благодаря использованию глубоких CNN-признаков.
Главный недостаток: каждый из ~ 2000 регионов независимо проходит через CNN:

$$\boxed{\text{2000 proposals} \;\Rightarrow\; \text{около 2000 запусков CNN}}$$

Это делает R-CNN очень медленным и приводит к следующему развитию идеи: **Fast R-CNN**.


R-CNN: Regions with CNN features
warped region
CNN
aeroplane? no.
person? yes.
tvmonitor? no.
1. Input image
2. Extract region proposals (~2k)
3. Compute CNN features
4. Classify regions

R-CNN обучение



Fast R-CNN: основная идея

Fast R-CNN ускоряет R-CNN за счет совместного вычисления CNN-признаков для всех region proposals.
https://arxiv.org/pdf/1504.08083

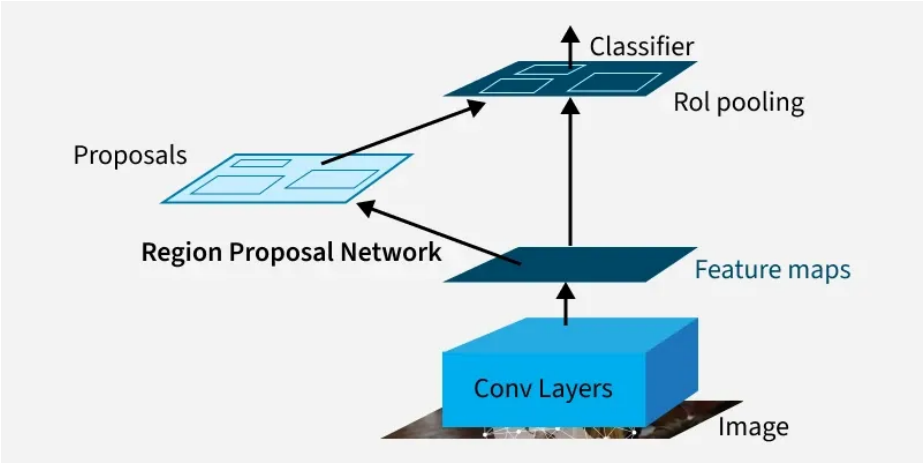
$$\text{R-CNN:} \qquad \text{2000 regions} \Rightarrow \text{2000 CNN forward passes.}$$

$$\text{Fast R-CNN:} \qquad \text{Image} \xrightarrow{\text{CNN}} \text{Feature Map.}$$

1. Selective Search генерирует region proposals.
2. CNN строит общую feature map всего изображения.
3. Каждый proposal проецируется на feature map.
4. **RoI Pooling** преобразует признаки региона к фиксированному размеру.

$$\boxed{\text{Image} \to \text{CNN} \to \text{RoI Pooling} \to \text{Detection}}$$

Fast R-CNN


Classifier
RoI pooling
Proposals
Region Proposal Network
Feature maps
Conv Layers
Image

Fast R-CNN: RoI Pooling и предсказание

Для каждого региона r_i слой **RoI Pooling** извлекает из общей feature map тензор фиксированного размера:

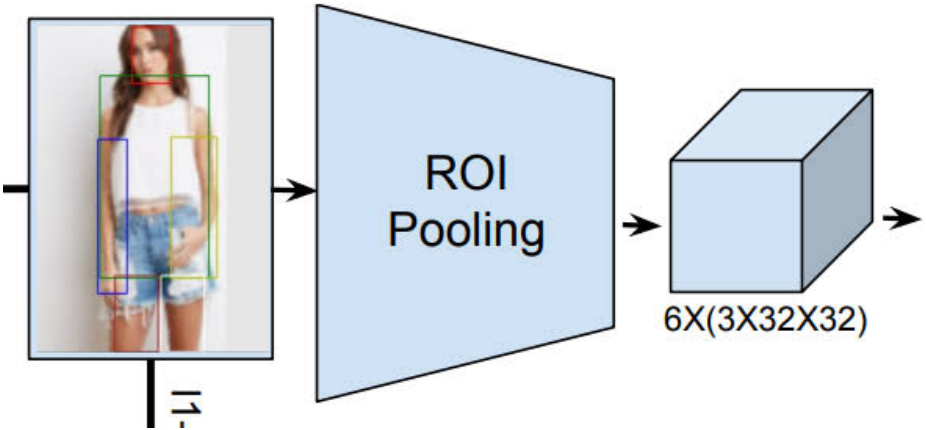
$$\mathbf{x}_i = \mathrm{RoIPool}(\mathbf{F}, r_i),$$

где $\mathbf{F}$ — общая feature map изображения.
Полученные признаки передаются в две выходные ветви:

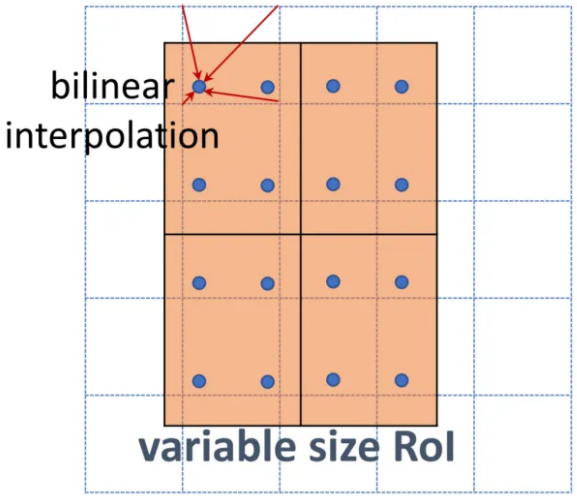
$$\mathbf{x}_i \longrightarrow \begin{cases} \text{Softmax} & \to \text{класс объекта}, \\ \text{BBox Regression} & \to \text{координаты рамки}. \end{cases}$$

Но остается узкое место: region proposals по-прежнему генерируются внешним алгоритмом **Selective Search**.

RoI Pooling



RoI Pooling



Bounding Box Regression

Пусть исходная рамка задана параметрами

$$(x, y, w, h),$$

где (x, y) — центр рамки, а w, h — её ширина и высота.
Сеть предсказывает четыре поправки:

$$(t_x, t_y, t_w, t_h).$$

После применения поправок получаем новую рамку:

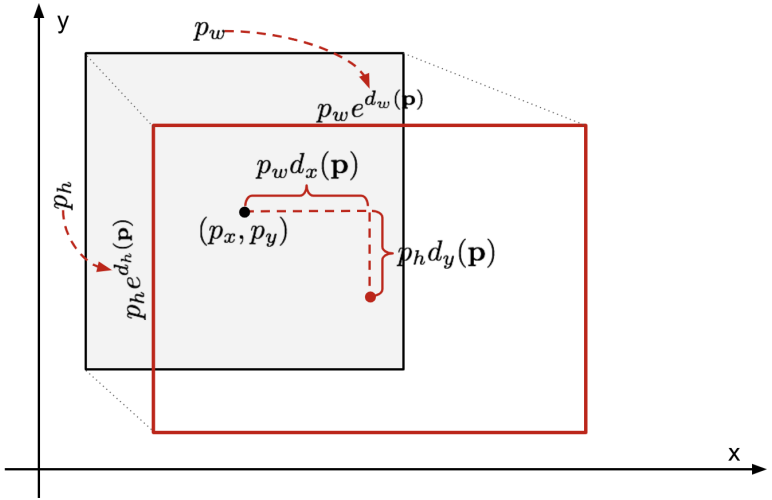
$$\hat{x} = x + wt_x, \qquad \hat{y} = y + ht_y,$$

$$\hat{w} = we^{t_w}, \qquad \hat{h} = he^{t_h}.$$

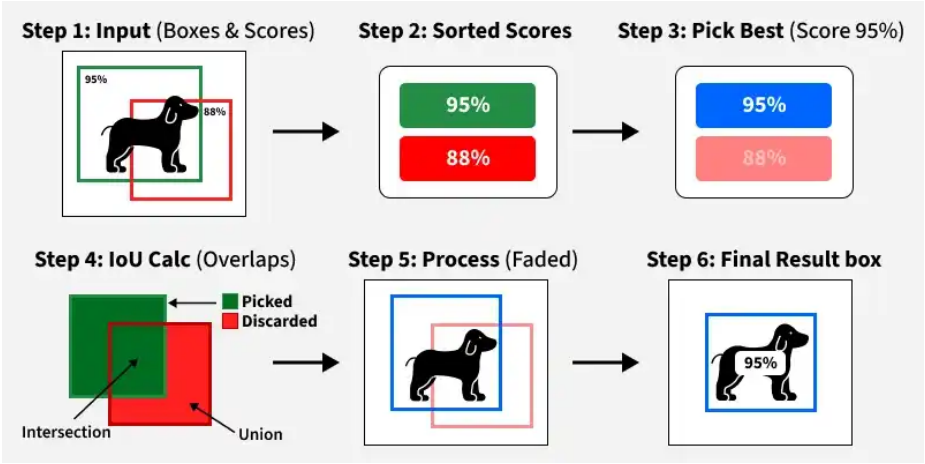
Таким образом,

$$\boxed{\text{Proposal} \; + \; (t_x, t_y, t_w, t_h) \; \longrightarrow \; \text{Refined Bounding Box}}$$

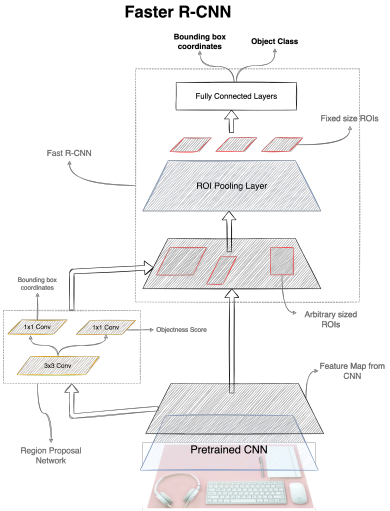
Bounding Box Regression



Non Maximum Suppression



Faster R-CNN


Faster R-CNN
Bounding box coordinates
Object Class
Fully Connected Layers
Fixed size ROIs
Fast R-CNN
ROI Pooling Layer
Bounding box coordinates
1x1 Conv
1x1 Conv
Objectness Score
3x3 Conv
Arbitrary sized ROIs
Feature Map from CNN
Region Proposal Network
Pretrained CNN

YOLO: You Only Look Once

YOLO рассматривает детекцию объектов как единую задачу регрессии, выполняемую за один проход нейронной сети.

$$\boxed{\text{Image} \xrightarrow{\text{CNN}} \{\text{Bounding Boxes},\ \text{Classes},\ \text{Confidence}\}}$$

- нет отдельного алгоритма генерации region proposals;
- признаки вычисляются сразу для всего изображения;
- координаты рамок и классы предсказываются одновременно;
- вся модель обучается end-to-end.

YOLO: архитектура

$$\text{Input} \to \underbrace{\text{Backbone}}_{\text{features}} \to \underbrace{\text{Neck}}_{\text{multi-scale}} \to \underbrace{\text{Detection Head}}_{\text{predictions}}$$

Backbone извлекает признаки изображения:

$$I \in \mathbb{R}^{3\times H\times W} \longrightarrow F_1, F_2, \ldots, F_L.$$

Neck объединяет признаки разных разрешений:

$$F_1, \ldots, F_L \longrightarrow P_1, \ldots, P_K.$$

Это позволяет одновременно находить маленькие и большие объекты.
Detection Head для каждой пространственной позиции предсказывает:

$$\boxed{(t_x, t_y, t_w, t_h) \;+\; p_{\text{obj}} \;+\; (p_1, \ldots, p_C)}$$

YOLO: предсказание объектов

Feature map разбивает изображение на пространственную сетку:

$$H \times W \quad \longrightarrow \quad S_h \times S_w.$$

Каждая позиция (i, j) отвечает за предсказание объектов, центр которых находится в соответствующей области изображения.

$$\mathbf{y} = (t_x, t_y, t_w, t_h,\ p_{\text{obj}},\ p_1, \ldots, p_C).$$

$$x = \frac{i + \sigma(t_x)}{S_w}, \qquad y = \frac{j + \sigma(t_y)}{S_h},$$

$$w = f_w(t_w), \qquad h = f_h(t_h),$$

$$\boxed{s_c = p_{\text{obj}}\, p_c}$$

YOLO: обучение

$$\mathcal{L} = \lambda_{\text{box}}\mathcal{L}_{\text{box}} + \lambda_{\text{obj}}\mathcal{L}_{\text{obj}} + \lambda_{\text{cls}}\mathcal{L}_{\text{cls}}.$$

1. Localization loss. Для предсказанной рамки B и ground truth B^*:

$$\text{IoU}(B, B^*) = \frac{|B \cap B^*|}{|B \cup B^*|}, \qquad \mathcal{L}_{\text{box}} = 1 - \text{IoU}(B, B^*).$$

На практике часто используются улучшенные варианты: GIoU, DIoU, CIoU.

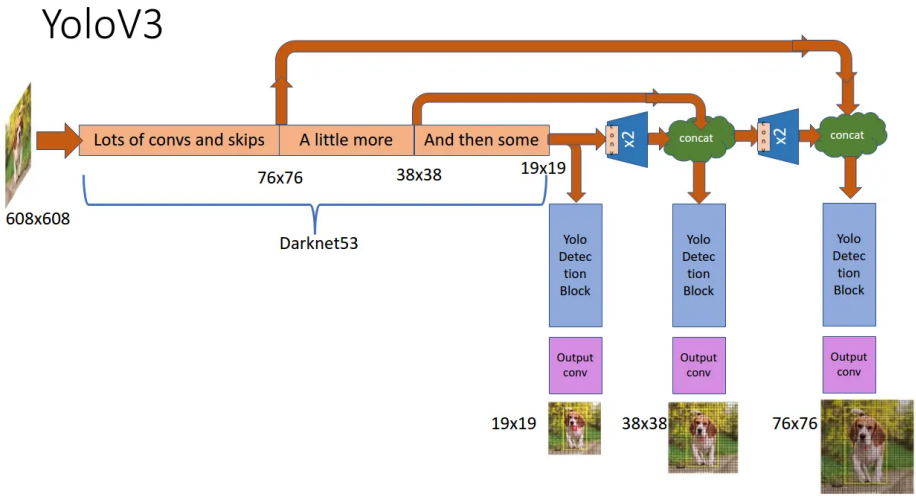
2. Objectness loss. Пусть $y_i \in \{0, 1\}$ показывает наличие объекта:

$$\mathcal{L}_{\text{obj}} = -\sum_i \left[y_i \log p_i + (1 - y_i) \log(1 - p_i) \right].$$

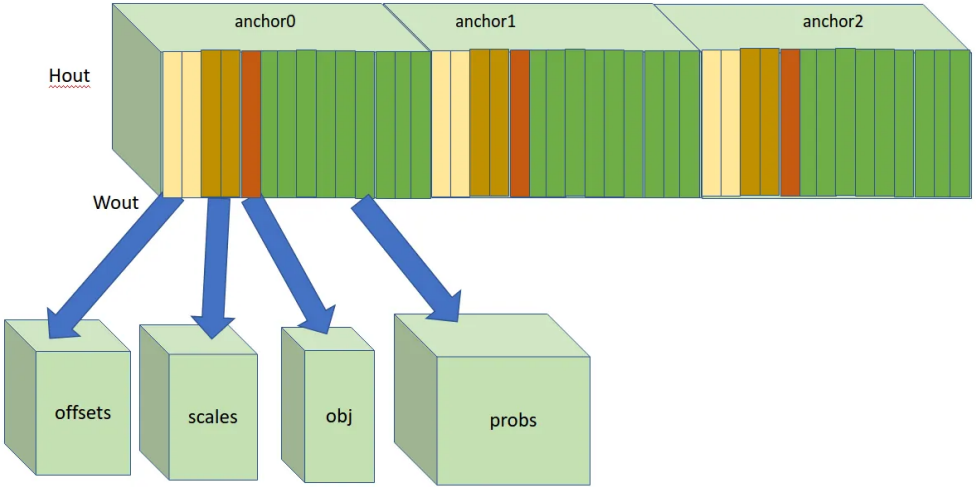
3. Classification loss. Для C классов:

$$\mathcal{L}_{\text{cls}} = -\sum_i \sum_{c=1}^{C} \left[y_{ic} \log p_{ic} + (1 - y_{ic}) \log(1 - p_{ic}) \right].$$

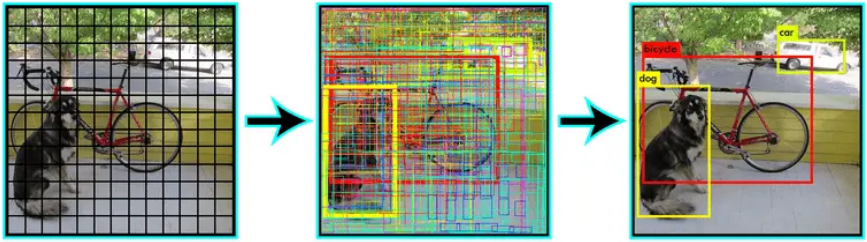
YOLO


YoloV3
Lots of convs and skips
A little more
And then some
608x608
76x76
38x38
19x19
Darknet53
x2
concat
x2
concat
Yolo Detection Block
Yolo Detection Block
Yolo Detection Block
Output conv
Output conv
Output conv
19x19
38x38
76x76

YOLO


anchor0
anchor1
anchor2
Hout
Wout
offsets
scales
obj
probs

YOLO



SSD: Single Shot MultiBox Detector

Главная идея — выполнять детекцию на **нескольких feature maps** разного разрешения:

$$\boxed{\text{Image} \to \text{Backbone} \to F_1 \to F_2 \to \ldots \to F_L}$$

Например:

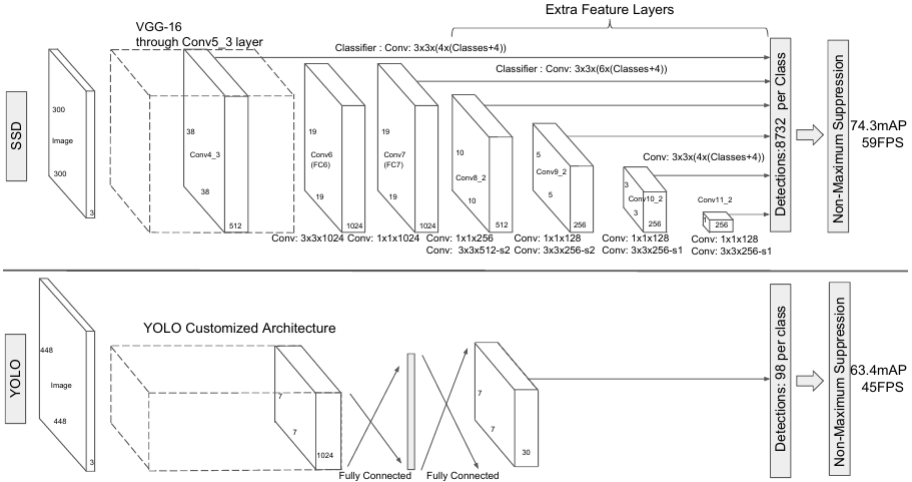
$$38 \times 38 \to 19 \times 19 \to 10 \times 10 \to 5 \times 5 \to 3 \times 3 \to 1 \times 1.$$

На каждом уровне сеть независимо предсказывает:

$$\text{BBox offsets} + \text{Class scores}.$$

- feature maps высокого разрешения $\to$ маленькие объекты;
- feature maps низкого разрешения $\to$ большие объекты;
- region proposals и отдельный RoI Pooling не нужны.

SSD


Extra Feature Layers
VGG-16
through Conv5_3 layer
Classifier : Conv: 3x3x(4x(Classes+4))
Classifier : Conv: 3x3x(6x(Classes+4))
Conv: 3x3x(4x(Classes+4))
SSD
300
Image
300
3
38
Conv4_3
38
512
19
Conv6
(FC6)
19
1024
19
Conv7
(FC7)
19
1024
10
Conv8_2
10
512
5
Conv9_2
5
256
3
Conv10_2
3
256
Conv11_2
1
256
Conv: 3x3x1024
Conv: 1x1x1024
Conv: 1x1x256
Conv: 3x3x512-s2
Conv: 1x1x128
Conv: 3x3x256-s2
Conv: 1x1x128
Conv: 3x3x256-s1
Conv: 1x1x128
Conv: 3x3x256-s1
Detections:8732 per Class
Non-Maximum Suppression
74.3mAP
59FPS
YOLO
448
Image
448
3
YOLO Customized Architecture
7
7
1024
Fully Connected
Fully Connected
7
7
30
Detections: 98 per class
Non-Maximum Suppression
63.4mAP
45FPS

SSD: Default Boxes и обучение

$$d = (d_x, d_y, d_w, d_h), \qquad t = (t_x, t_y, t_w, t_h), \qquad p = (p_0, p_1, \ldots, p_C),$$

$$\mathrm{IoU}(d, g) = \frac{|d \cap g|}{|d \cup g|}.$$

$$\boxed{\mathcal{L} = \frac{1}{N}\left(\mathcal{L}_{\mathrm{cls}} + \alpha \mathcal{L}_{\mathrm{loc}}\right)}$$

$$\mathcal{L}_{\mathrm{loc}} = \sum_{i \in \mathrm{Pos}} \sum_{m \in \{x,y,w,h\}} \mathrm{smooth}_{L_1}\left(t_i^m - \hat{t}_i^m\right),$$

$$\mathrm{smooth}_{L_1}(z) = \begin{cases} \frac{1}{2}z^2, & |z| < 1, \\ |z| - \frac{1}{2}, & |z| \geq 1. \end{cases}$$

а $\mathcal{L}_{\mathrm{cls}}$ — softmax cross-entropy по классам и background.

FPN: Feature Pyramid Network

ранние слои $\rightarrow$ высокое разрешение, слабая семантика

глубокие слои $\rightarrow$ низкое разрешение, сильная семантика

Например, backbone строит иерархию признаков:

$$C_2,\ C_3,\ C_4,\ C_5,$$

$$\frac{H}{4} \times \frac{W}{4} \rightarrow \frac{H}{8} \times \frac{W}{8} \rightarrow \frac{H}{16} \times \frac{W}{16} \rightarrow \frac{H}{32} \times \frac{W}{32}.$$

FPN строит новую пирамиду

$$P_2,\ P_3,\ P_4,\ P_5,$$

$$\begin{aligned} P_2, P_3 &\rightarrow \text{маленькие объекты}, \\ P_4 &\rightarrow \text{средние объекты}, \\ P_5 &\rightarrow \text{крупные объекты}. \end{aligned}$$

FPN: Top-Down Pathway и Lateral Connections

FPN объединяет глубокие семантические признаки с feature maps высокого разрешения.

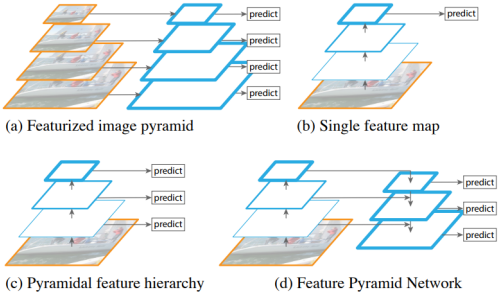
$$P_5 = \mathrm{Conv}_{1\times1}(C_5).$$

$$\tilde{P}_l = \mathrm{UpSample}(P_{l+1}),$$

$$\boxed{P_l = \mathrm{Conv}_{3\times3}\left(\mathrm{Conv}_{1\times1}(C_l) + \mathrm{UpSample}(P_{l+1})\right)}$$

- $\mathrm{Conv}_{1\times1}$ приводит число каналов C_l к одинаковой размерности;
- UpSample увеличивает пространственное разрешение в 2 раза;
- сложение называется **lateral connection**;
- $\mathrm{Conv}_{3\times3}$ сглаживает результат после объединения.

FPN



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

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

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

https://arxiv.org/pdf/1612.03144

Anchor-Free Detection

Например, в **FCOS** каждая точка (x, y) на feature map может предсказывать bounding box непосредственно через расстояния до его границ. Для каждой позиции сеть предсказывает

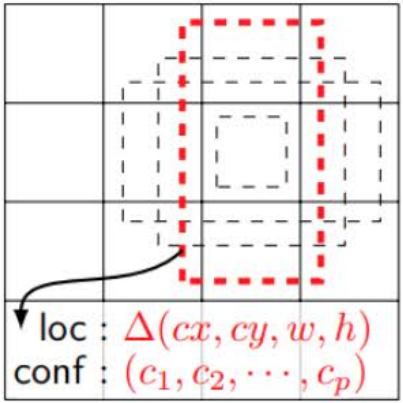
$$\text{class probabilities} \; + \; (l, t, r, b) \; + \; \text{centerness}.$$

$$\boxed{x_1 = x - l, \qquad y_1 = y - t, \qquad x_2 = x + r, \qquad y_2 = y + b.}$$

Centerness уменьшает score для точек далеко от центра объекта:

$$c = \sqrt{\frac{\min(l, r)}{\max(l, r)} \cdot \frac{\min(t, b)}{\max(t, b)}}.$$

Anchor based detection using SSD



Anchor free detection using FCOS



Эволюция CNN-детекторов

R-CNN $\rightarrow$ Fast R-CNN $\rightarrow$ Faster R-CNN

SSD / YOLO $\rightarrow$ FPN $\rightarrow$ Anchor-Free Detectors $\rightarrow$ Modern YOLO