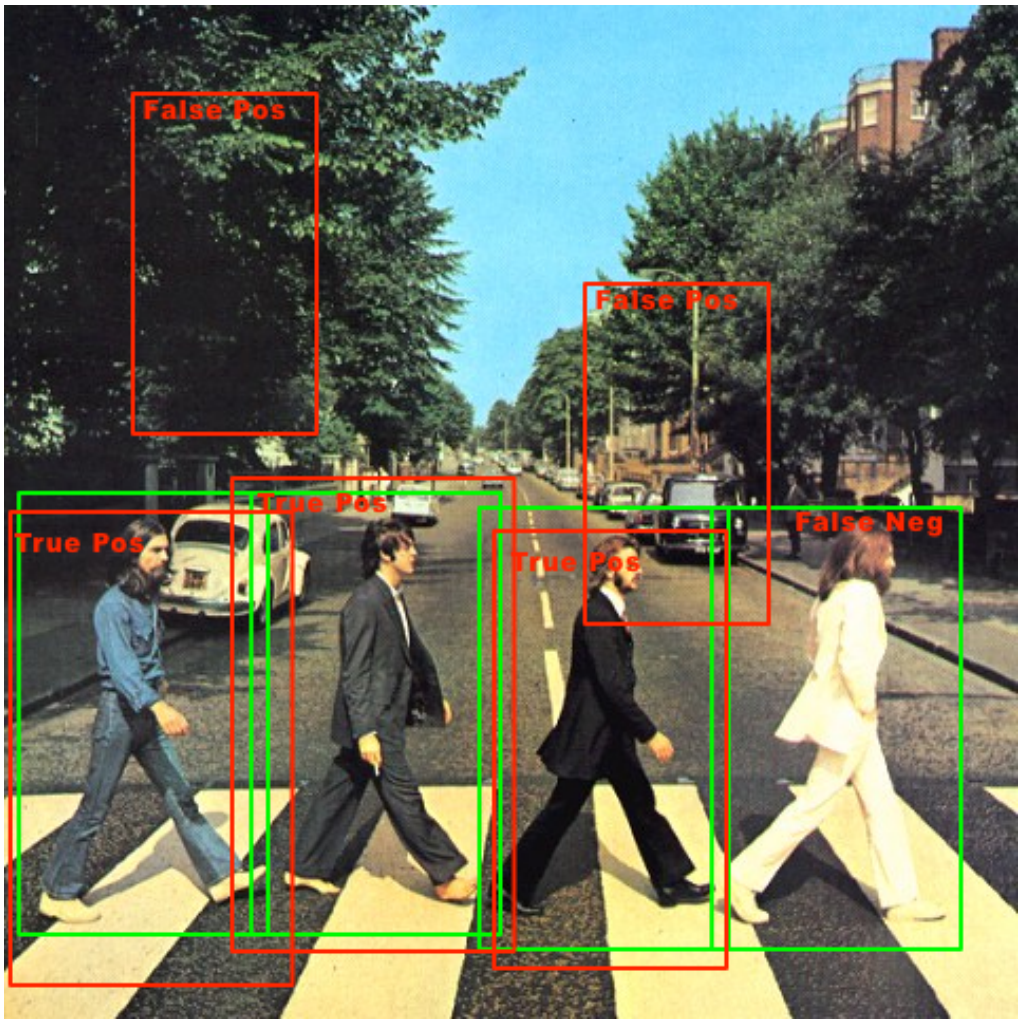
Современные архитектуры для детекции объектов на изображении

Пономарева Любовь

МГУ имени М. В. Ломоносова, факультет ВМК, кафедра ММП

17 мая 2018 г.

Постановка задачи



Требуется определить, присутствуют ли на входном изображении объекты заданного класса, и уточнить их положение.

- Для каждого объекта на изображении мы должны предсказать: $class$, $x_{\ top\ left}$, $y_{\ top\ right}$, $width$, $height$;
- Sliding Windows Detection:
  - Обучим CNN предсказывать класс объекта для уменьшенного изображения;
  - Проходим по нему скользящим окном (с различным масштабом и шагом), отвечая на вопрос, попал ли наш объект в данную область;
  - Это очень неэффективно.

Algorithm 1: Hierarchical Grouping Algorithm

Input: (colour) image
Output: Set of object location hypotheses L

Obtain initial regions $R = \{r_1, \cdots, r_n\}$ using [13]
Initialise similarity set $S = \emptyset$
foreach *Neighbouring region pair* (r_i, r_j) **do**
    Calculate similarity $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$

while $S \neq \emptyset$ **do**
    Get highest similarity $s(r_i, r_j) = \max(S)$
    Merge corresponding regions $r_t = r_i \cup r_j$
    Remove similarities regarding $r_i : S = S \setminus s(r_i, r_*)$
    Remove similarities regarding $r_j : S = S \setminus s(r_*, r_j)$
    Calculate similarity set S_t between r_t and its neighbours
    $S = S \cup S_t$
    $R = R \cup r_t$

Extract object location boxes L from all regions in R

Selective Search: пример того, как получаются признаки для регионов

- $s_{colour}(r_i, r_j)$: для каждого региона получаем цветовой вектор-гистограмму длины 25 из 0 и 1 для каждого канала, нормализуем его ($L_1\ norm$):

$$C_i = \{c_i^1, \ldots, c_i^n\},\ n = 75,\ s_{colour}(r_i, r_j) = \sum_{k=1}^{n} \min(c_i^k, c_j^k)$$

$$C_t = \frac{size(r_i) * C_i + size(r_j) * C_j}{size(r_i) + size(r_j)}$$

- $s_{texture}(r_i, r_j) = \sum_{k=1}^{n} \min(t_i^k, t_j^k)$

$$\frac{\partial G_\sigma(x,y)}{\partial x} \propto xe^{-\frac{x^2+y^2}{2\sigma^2}} \quad \frac{\partial G_\sigma(x,y)}{\partial y} \propto ye^{-\frac{x^2+y^2}{2\sigma^2}}$$

Производная вычисляется по 8 направлениям, для каждого направления для каждого цветового канала извлекаем вектор-гистограмму длины 10 из 0 и 1.
$T_i = \{t_i^1, \ldots, t_i^n\},\ n = 240$

Selective Search: пример того, как получаются признаки для регионов

$$s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(im)}$$

$$s_{fill}(r_i, r_j) = 1 - \frac{size(BoundingBox_{ij}) - size(r_i) - size(r_j)}{size(im)}$$

$$\begin{aligned} s(r_i, r_j) \quad = \quad & a_1 s_{colour}(r_i, r_j) + a_2 s_{texture}(r_i, r_j) + \\ & a_3 s_{size}(r_i, r_j) + a_4 s_{fill}(r_i, r_j), \end{aligned}$$

$$IoU = \frac{Img_1 \cap Img_2}{Img_1 \cup Img_2}$$

Если $IoU > 0.5$: региону присваивается класс детектируемого объекта;

$$Precision_C = \frac{N(TruePositives)_C}{N(TotalObjects)_C}$$

$$AveragePrecision_C = \frac{\sum Precision_C}{N(TotalImages)_C}$$

$$MeanAveragePrecision = \frac{AveragePrecision_C}{N(Classes)}$$

Non-Max Suppression

- $B = b_1, \ldots, b_N$ — набор прямоугольников-претендентов для одного объекта;
- $S = s_1, \ldots, s_N$ — соответствующие им оценки

$$s_i = \begin{cases} s_i & IoU(b, b_i) < T \\ 0 & IoU(b, b_i) \geq T \end{cases}$$

- $s_i = f(IoU(b, b_i), s_i)$
- Например:

$$s_i = \begin{cases} s_i & IoU(b, b_i) < T \\ s_i(1 - IoU(b, b_i)) & IoU(b, b_i) \geq T \end{cases}$$

  Плохо: не является непрерывной.

- $s_i = s_i \exp \frac{-IoU(b,b_i)^2}{\sigma}$

Navaneeth Bodla, Bharat Singh, Rama Chellappa, Larry S. Davis, «Improving Object Detection With One Line of Code», 2017

R-CNN

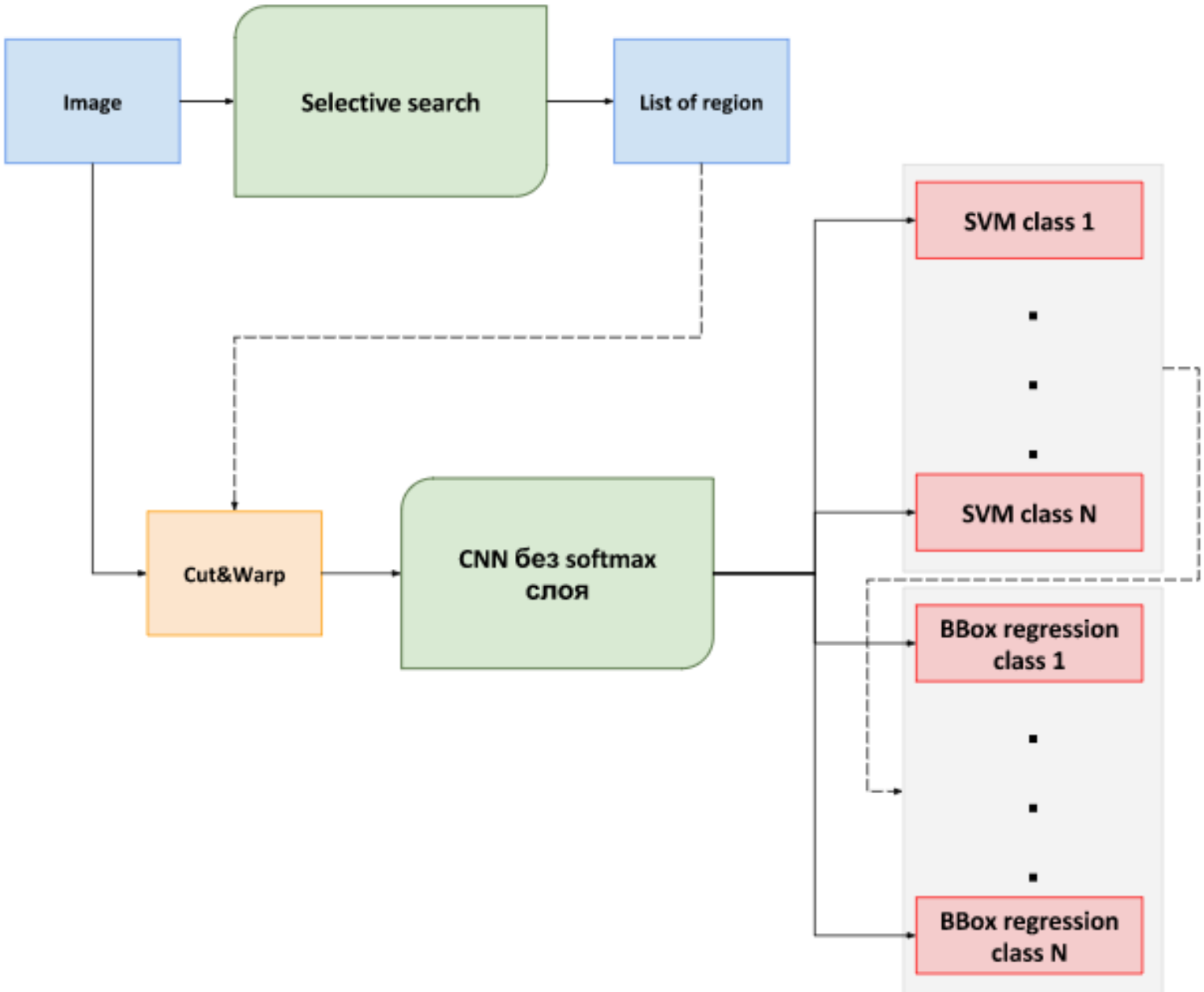


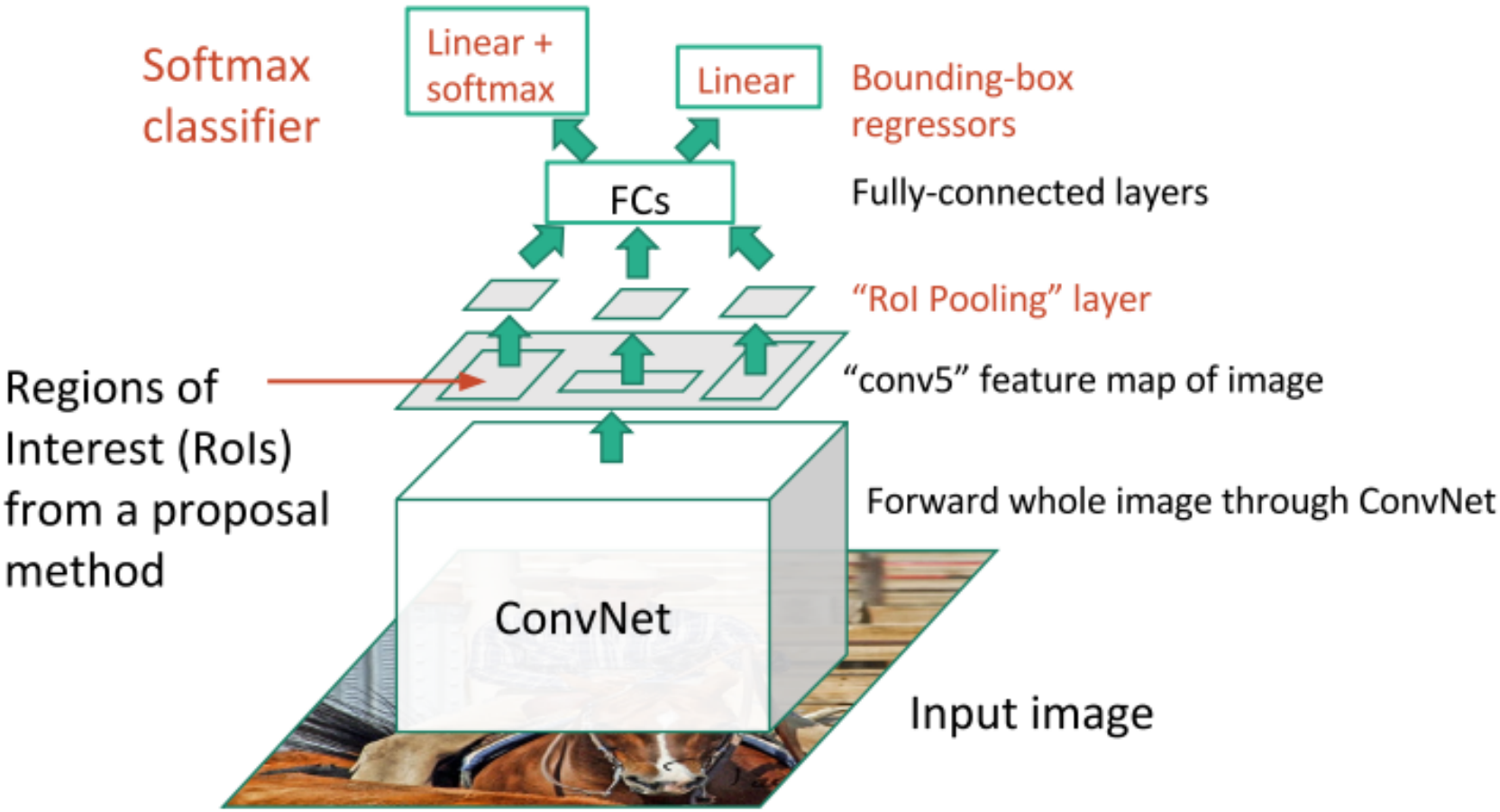
Image
Selective search
List of region
SVM class 1
SVM class N
Cut&Warp
CNN без softmax слоя
BBox regression class 1
BBox regression class N

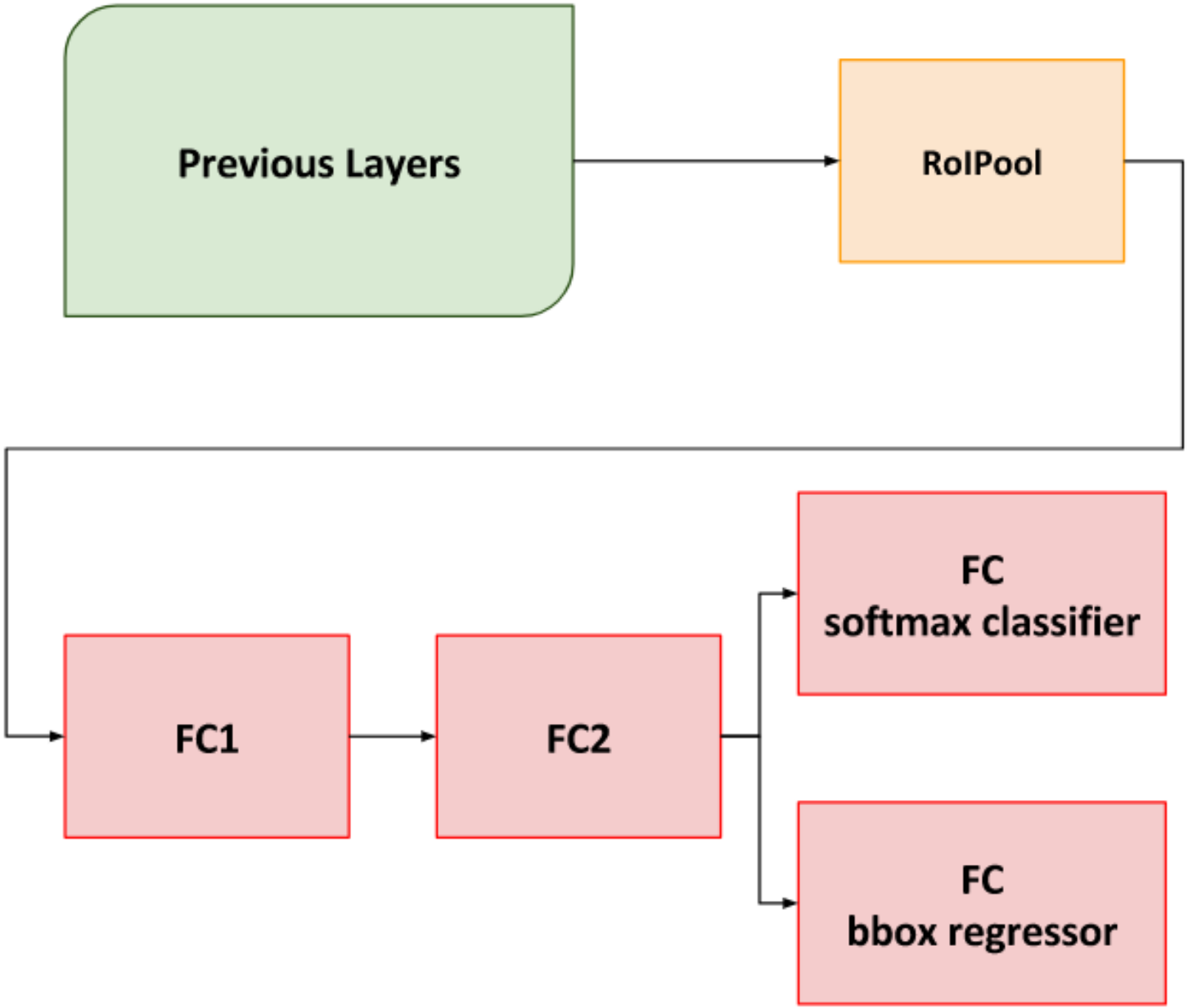
Проблемы R-CNN

- Нужно натренировать три модели отдельно:
  Fine-tune network with softmax classifier (log loss)
  Train linear SVMs (hinge loss) - только после того, как мы натренировали сеть;
  Train bounding-box regressions (least squares) - только после того, как мы натренировали классификатор.
- Обучение медленное (84 часа), занимает много дискового пространства;
- Медленная детекция на этапе тестирования.

Fast R-CNN

1. Higher detection quality (mAP) than R-CNN;
2. Training is single-stage, using a multi-task loss;
3. No disk storage is required for feature caching.




Previous Layers
RoIPool
FC
softmax classifier
FC1
FC2
FC
bbox regressor

Избавляемся от Selective Search

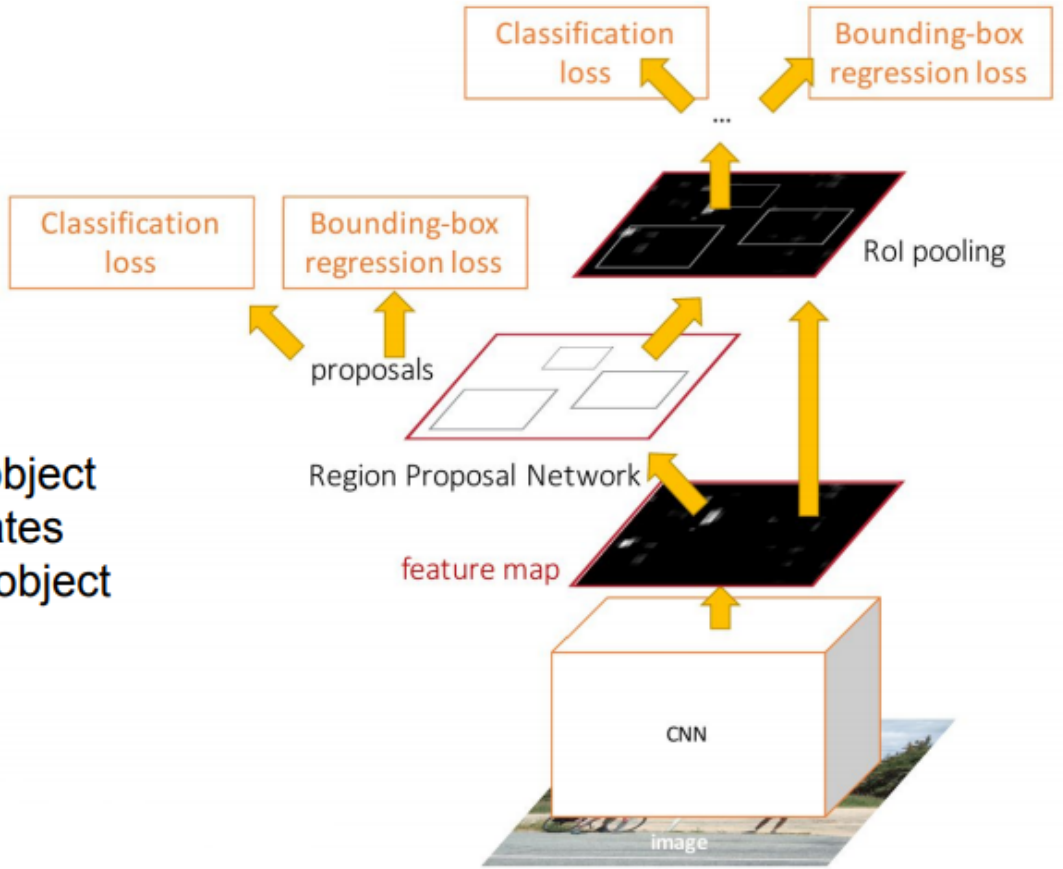
Faster R-CNN:

Make CNN do proposals!

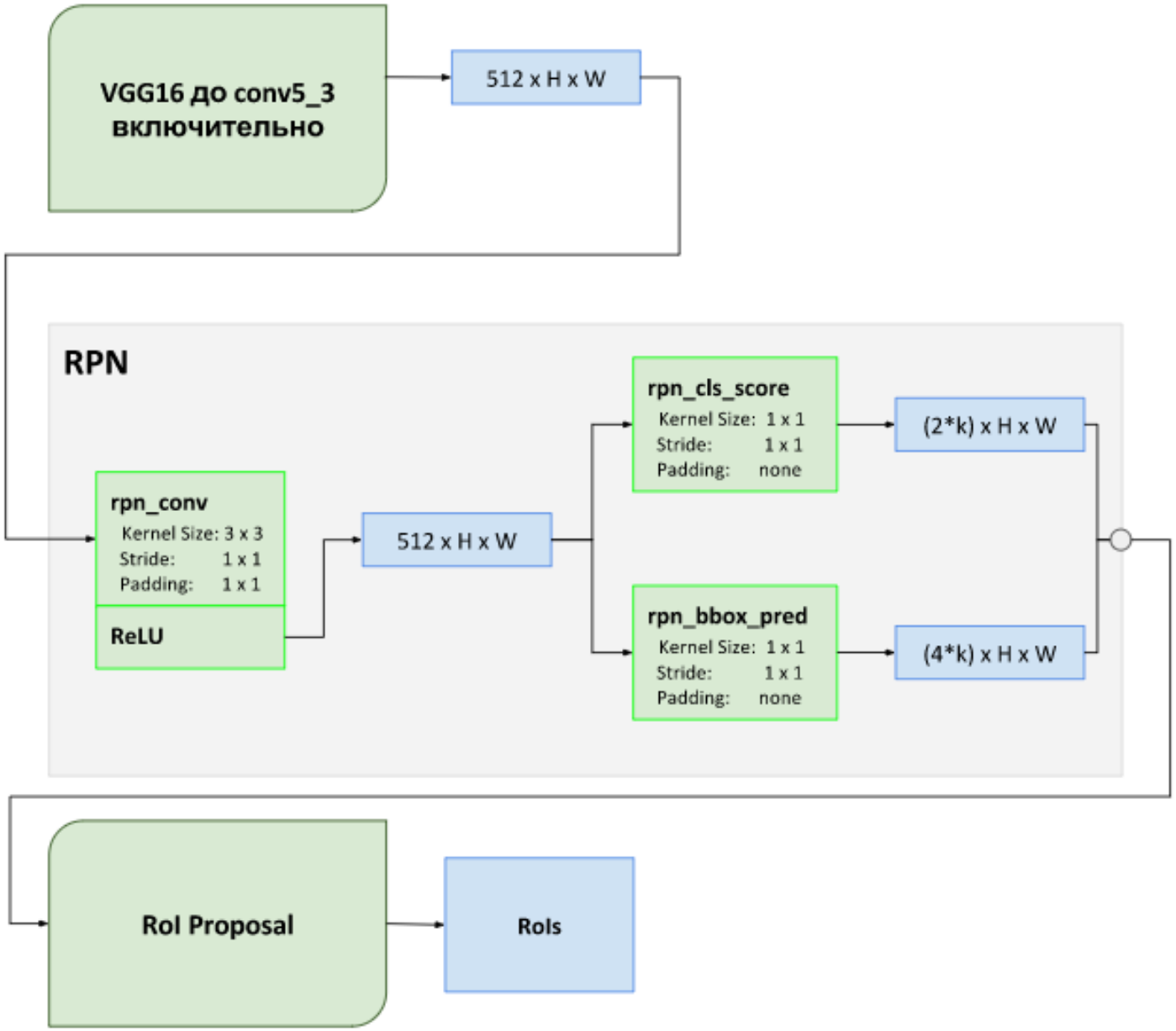
Insert **Region Proposal Network (RPN)** to predict proposals from features

Jointly train with 4 losses:

1. RPN classify object / not object
2. RPN regress box coordinates
3. Final classification score (object classes)
4. Final box coordinates



Faster R-CNN


VGG16 до conv5_3 включительно
512 x H x W
RPN
rpn_conv
Kernel Size: 3 x 3
Stride: 1 x 1
Padding: 1 x 1
ReLU
512 x H x W
rpn_cls_score
Kernel Size: 1 x 1
Stride: 1 x 1
Padding: none
(2*k) x H x W
rpn_bbox_pred
Kernel Size: 1 x 1
Stride: 1 x 1
Padding: none
(4*k) x H x W
RoI Proposal
RoIs

Ground Truth Bounding Box

w^*: Box width
h^*: Box height
x^*, y^*: Box center

Anchor's properties

w_a: width
h_a: height
x_a, y_a: center

$p^* \in \{0, 1, -1\}$
based on IoU between GT-Box and Anchor

Classification ***cls***

p

Regressor ***reg***

w: width
h: height
x, y: center

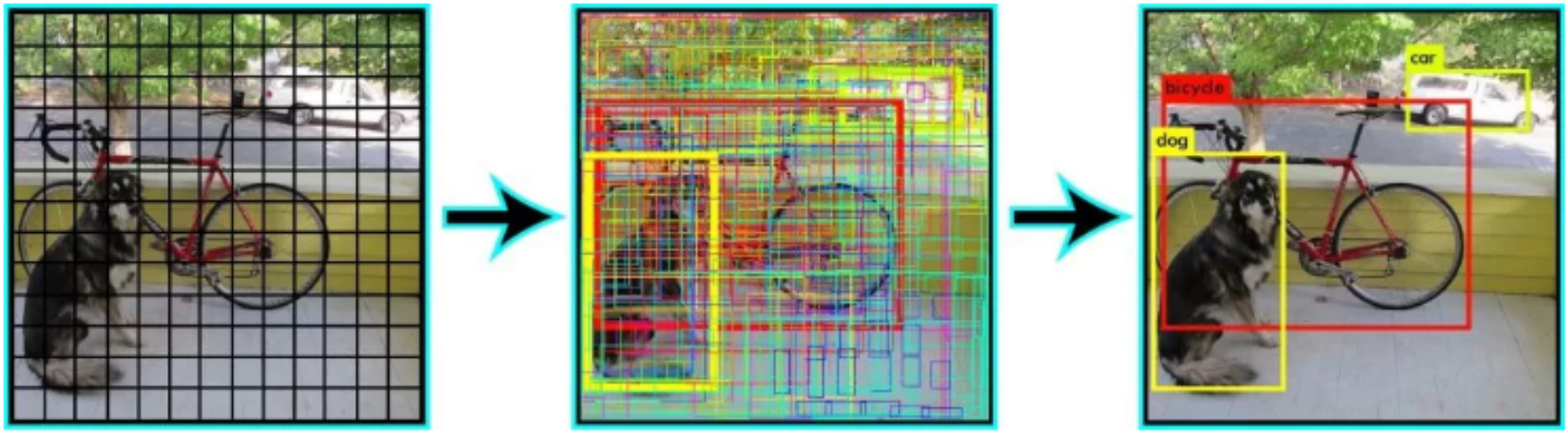
$$t = [(x - x_a)/w_a,\ (y - y_a)/h_a,\ \log w/w_a,\ \log h/h_a]$$

$$t^* = [(x^* - x_a)/w_a,\ (y^* - y_a)/h_a,\ \log w^*/w_a,\ \log h^*/h_a]$$

Loss Function

$$L(\{p_i\}, \{t_i\}) = 1/N_{cls}\ L_{cls}(p_i, p_i^*) + \lambda\ 1/N_{reg}\ L_{reg}(t_i, t_i^*)$$

You Only Look Once



- $S \times S$ ячеек сетки; каждая предсказывает положение B bounding boxes и C вероятностей принадлежности объекта внутри к классам; для каждой ячейки:
  $(x, y, w, h, confidence)$,
  $confidence = P(Object) * IoU(pred, truth)$;
- Total: $S \times S \times B * 5$ предсказаний для положения, $S \times S \times C$ предсказаний классов — в результате $S \times S \times (B * 5 + C)$ тензор на выходе.

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

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

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

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

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

Model	mAP	FPS	Real Time speed
Fast YOLO	52.7%	**155**	Yes
YOLO	**63.4%**	45	Yes
YOLO VGG-16	66.4%	21	No
Fast R-CNN	70.0%	0.5	No
Faster R-CNN VGG-16	73.2%	7	No
Faster R-CNN ZF	62.1%	18	No

Real Time Systems on PASCAL VOC 2007. Comparison of speeds and performances for models trained with the 2007 and 2012 PASCAL VOC datasets. The published results correspond to the implementations of J. Redmon and al. (2016).

Источники

- R-CNN;
- Fast R-CNN;
- Faster R-CNN;
- Лекция Джастина Джонсона, Детекция и локализация объектов, Стэнфордский университет, 2017;
- YOLO.