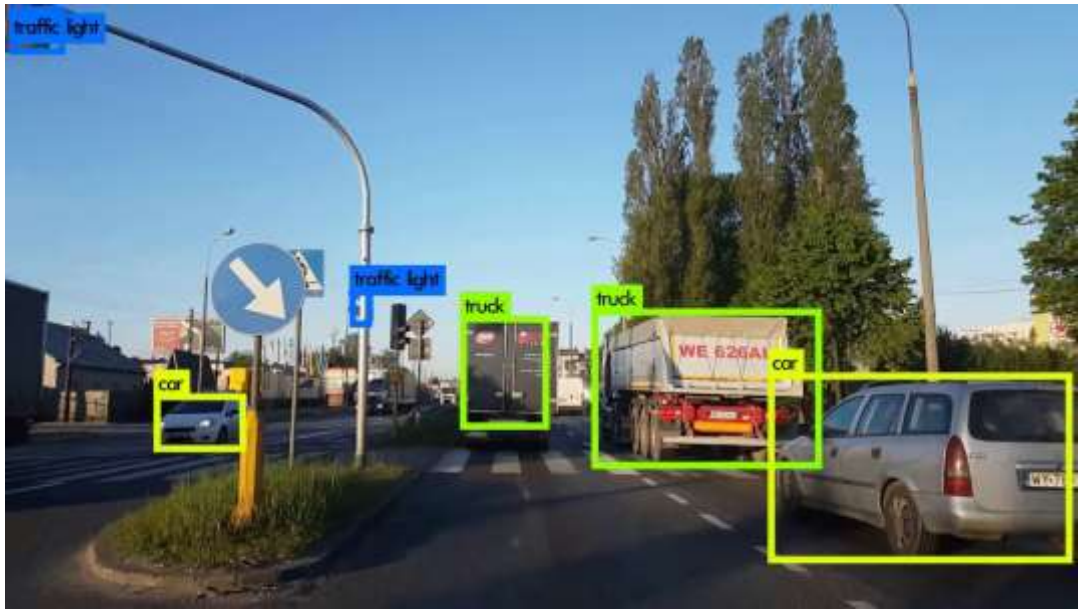
Real-Time Object Detection:
Fast R-CNN, You Only Look Once

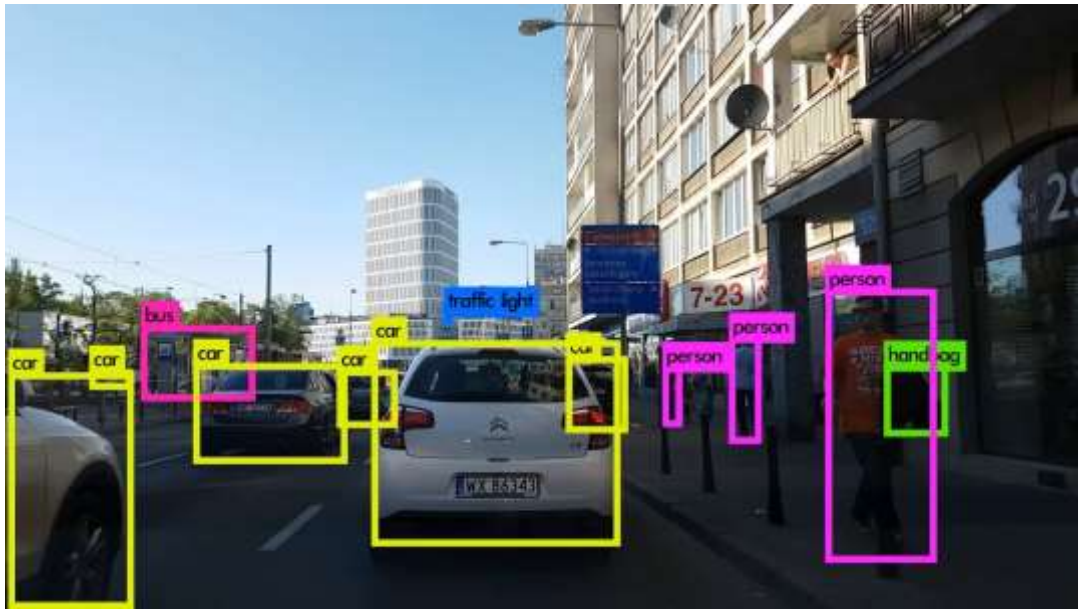
Симкин Алексей

22 февраля 2019 г.

Object detection

- Задача автоматического выделения объектов определенного класса(люди, автомобили, здания) на изображениях





Object detection

- С помощью нейросетей можно достигнуть хороших результатов, но это требует много ресурсов

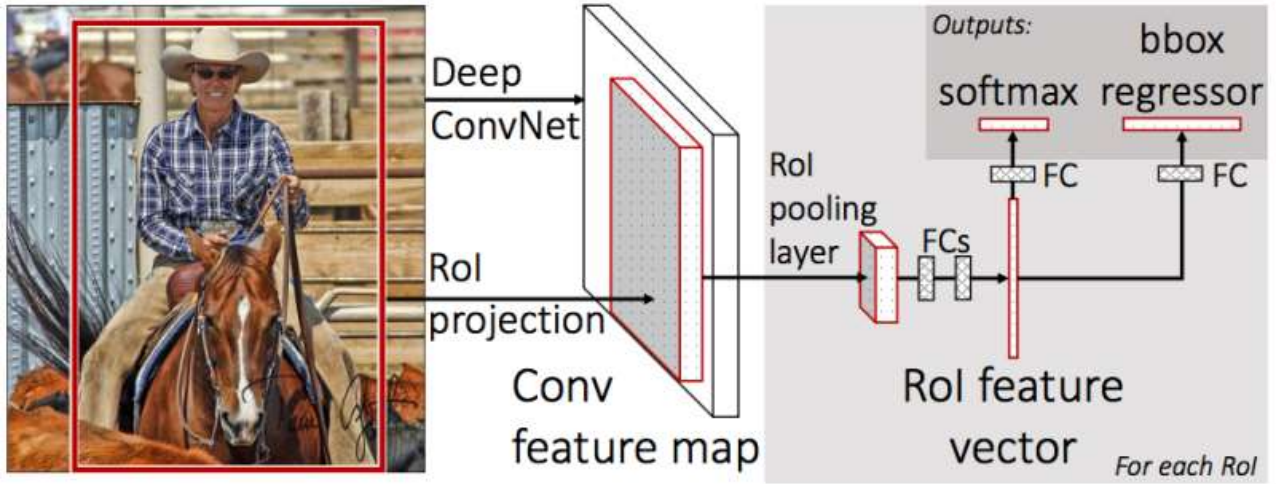
1. **Training is a multi-stage pipeline.** R-CNN first fine-tunes a ConvNet on object proposals using log loss. Then, it fits SVMs to ConvNet features. These SVMs act as object detectors, replacing the softmax classifier learnt by fine-tuning. In the third training stage, bounding-box regressors are learned.
2. **Training is expensive in space and time.** For SVM and bounding-box regressor training, features are extracted from each object proposal in each image and written to disk. With very deep networks, such as VGG16, this process takes 2.5 GPU-days for the 5k images of the VOC07 trainval set. These features require hundreds of gigabytes of storage.
3. **Object detection is slow.** At test-time, features are extracted from each object proposal in each test image. Detection with VGG16 takes 47s / image (on a GPU).

Fast R-CNN

- Higher detection quality (mAP) than R-CNN, SPPnet
- Training is single-stage, using a multi-task loss
- Training can update all network layers
- No disk storage is required for feature caching
- Fast R-CNN trains the very deep VGG16 network 9× faster than R-CNN, is 213× faster at test-time, and achieves a higher mAP on PASCAL VOC 2012
- Compared to SPPnet, Fast R-CNN trains VGG16 3× faster, tests 10× faster, and is more accurate. Fast R-CNN is implemented in Python and C++ (using Caffe)

Fast R-CNN

- На вход подается изображение и предполагаемые локации объекта
- Изображение обрабатывается сверточной нейросетью (например VGG16)
- Для каждой предполагаемой локации извлекается вектор признаков фиксированной длина с помощью region of interest (RoI) pooling layer
- Полученные векторы проходят через несколько полносвязных слоев
- Один из выходных слоев выдает softmax вероятности для K классов объектов и одного класса для фона
- Второй выходной слой выдает четыре числа для каждого из K классов. Каждый такой набор задает положение bounding-box для одного класса



Fast R-CNN

Region of Interest (RoI) pooling layer:

- Каждый регион задается значениями (r, c, h, w), где (r, c) задают координаты левого верхнего угла, а его высоту и ширину задают (h, w)
- Регион размером h × w делится на сетку, имеющую H×W ячеек размера h/H × w/W
- В статье использовались значения H = W = 7
- В каждой ячейке используется max-pooling, получая на выходе матрицу размера H×W
- Для каждого канала изображения max-pooling делается независимо

Fast R-CNN

Сверточная нейросеть:

- CaffeNet(AlexNet)
- VGG CNN M 1024
- VGG16

Предобученная на ImageNet

Fast R-CNN

- Multi-task loss:

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1] L_{loc}(t^u, v)$$

- $p = (p_0, \ldots, p_K)$ — вероятности классов
- u — истинный класс
- $L_{cls}(p, u) = -\log p_u$ — log loss for the true class

Fast R-CNN

- $v = (v_x, v_y, v_w, v_h)$ — значения bounding-box для класса u
- $t^u = (t_x^u, t_y^u, t_w^u, t_h^u)$ — предсказанные значения
-
- $L_{loc}(t^u, v) = \sum_{i \epsilon \{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i)$

- $smooth_{L_1} = \begin{cases} 0.5x^2, \text{если } |x| < 1 \\ |x| - 0.5, \text{иначе} \end{cases}$

Fast R-CNN

- Мини-батч составляется на основе N = 2 случайных изображений
- Размер мини-батча равен R = 128, то есть от каждого изображения выбирается sampling 64 RoI
- Такое сэмплирование позволяет более эффективно обучать модель по сравнению с методами, где RoI берутся от случайного числа изображений
- ”*Critically, RoIs from the same image share computation and memory in the forward and backward passes*”

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
SPPnet BB [11]†	07 \ diff	73.9	72.3	62.5	51.5	44.4	74.4	73.0	74.4	42.3	73.6	57.7	70.3	74.6	74.3	54.2	34.0	56.4	56.4	67.9	73.5	63.1
R-CNN BB [10]	07	73.4	77.0	63.4	45.4	**44.6**	75.1	78.1	79.8	40.5	73.7	62.2	79.4	78.1	73.1	64.2	**35.6**	66.8	67.2	70.4	**71.1**	66.0
FRCN [ours]	07	74.5	78.3	69.2	53.2	36.6	77.3	78.2	82.0	40.7	72.7	67.9	79.6	79.2	73.0	69.0	30.1	65.4	70.2	75.8	65.8	66.9
FRCN [ours]	07 \ diff	74.6	**79.0**	68.6	57.0	39.3	79.5	**78.6**	81.9	**48.0**	74.0	67.4	80.5	80.7	74.1	69.6	31.8	67.1	68.4	75.3	65.5	68.1
FRCN [ours]	07+12	**77.0**	78.1	**69.3**	**59.4**	38.3	**81.6**	**78.6**	**86.7**	42.8	**78.8**	**68.9**	**84.7**	**82.0**	**76.6**	**69.9**	31.8	**70.1**	**74.8**	**80.4**	70.4	**70.0**

Table 1. **VOC 2007 test** detection average precision (%). All methods use VGG16. Training set key: **07**: VOC07 trainval, **07 \ diff**: **07** without "difficult" examples, **07+12**: union of **07** and VOC12 trainval. †SPPnet results were prepared by the authors of [11].

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	77.7	73.8	62.3	48.8	45.4	67.3	67.0	80.3	41.3	70.8	49.7	79.5	74.7	78.6	64.5	36.0	69.9	55.7	70.4	61.7	63.8
R-CNN BB [10]	12	79.3	72.4	63.1	44.0	44.4	64.6	66.3	84.9	38.8	67.3	48.4	82.3	75.0	76.7	65.7	35.8	66.2	54.8	69.1	58.8	62.9
SegDeepM	12+seg	**82.3**	75.2	67.1	50.7	**49.8**	71.1	69.6	88.2	42.5	71.2	50.0	85.7	76.6	81.8	69.3	**41.5**	**71.9**	62.2	73.2	**64.6**	67.2
FRCN [ours]	12	80.1	74.4	67.7	49.4	41.4	74.2	68.8	87.8	41.9	70.1	50.2	86.1	77.3	81.1	70.4	33.3	67.0	63.3	77.2	60.0	66.1
FRCN [ours]	07++12	82.0	**77.8**	**71.6**	**55.3**	42.4	**77.3**	**71.7**	**89.3**	**44.5**	**72.1**	**53.7**	**87.7**	**80.0**	**82.5**	**72.7**	36.6	68.7	**65.4**	**81.1**	62.7	**68.8**

Table 2. **VOC 2010 test** detection average precision (%). BabyLearning uses a network based on [17]. All other methods use VGG16. Training set key: **12**: VOC12 trainval, **Prop.**: proprietary dataset, **12+seg**: **12** with segmentation annotations, **07++12**: union of VOC07 trainval, VOC07 test, and VOC12 trainval.

method	train set	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	persn	plant	sheep	sofa	train	tv	mAP
BabyLearning	Prop.	78.0	74.2	61.3	45.7	42.7	68.2	66.8	80.2	40.6	70.0	49.8	79.0	74.5	77.9	64.0	35.3	67.9	55.7	68.7	62.6	63.2
NUS_NIN_c2000	Unk.	80.2	73.8	61.9	43.7	**43.0**	70.3	67.6	80.7	41.9	69.7	51.7	78.2	75.2	76.9	65.1	**38.6**	**68.3**	58.0	68.7	63.3	63.8
R-CNN BB [10]	12	79.6	72.7	61.9	41.2	41.9	65.9	66.4	84.6	38.5	67.2	46.7	82.0	74.8	76.0	65.2	35.6	65.4	54.2	67.4	60.3	62.4
FRCN [ours]	12	80.3	74.7	66.9	46.9	37.7	73.9	68.6	87.7	41.7	71.1	51.1	86.0	77.8	79.8	69.8	32.1	65.5	63.8	76.4	61.7	65.7
FRCN [ours]	07++12	**82.3**	**78.4**	**70.8**	**52.3**	38.7	**77.8**	**71.6**	**89.3**	**44.2**	**73.0**	**55.0**	**87.5**	**80.5**	**80.8**	**72.0**	35.1	**68.3**	**65.7**	**80.4**	**64.2**	**68.4**

Table 3. **VOC 2012 test** detection average precision (%). BabyLearning and NUS_NIN_c2000 use networks based on [17]. All other methods use VGG16. Training set key: see Table 2, **Unk.**: unknown.

You Only Look Once(YOLO)

- Задача детекции рассматривается как задача регрессии для нескольких отдельных bounding boxes и связанных с ними вероятностей классов, а не классификации, как это делается многими
- Все это делается при помощи одной нейросети, использующей в качестве входа все изображение

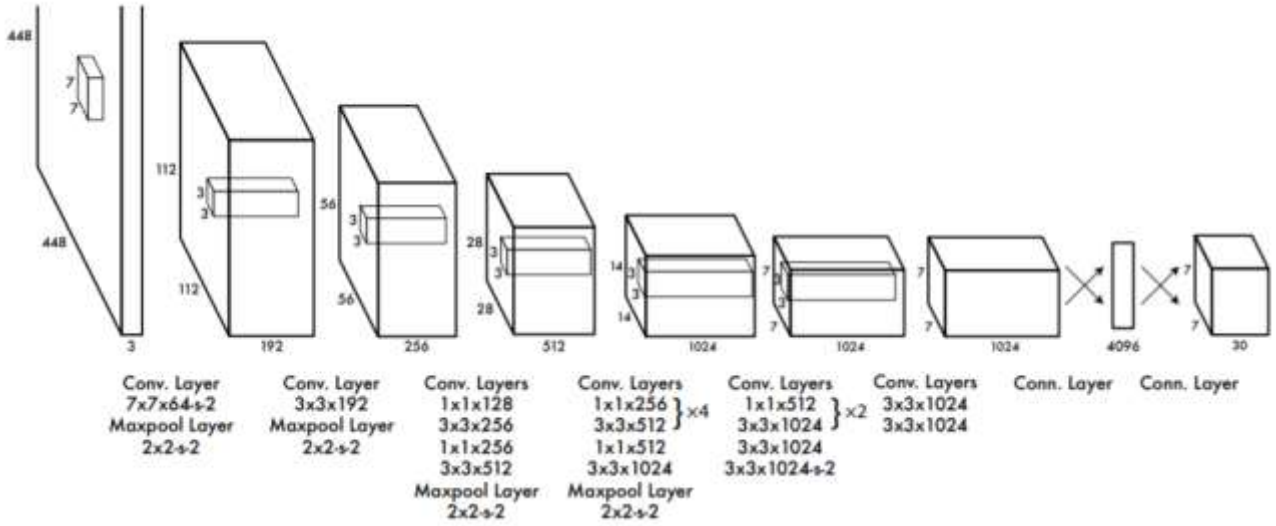
YOLO

- Входное изображение делится на сетку размером S × S
- Если центр объекта попадает в клетку сетки, то эта клетка отвечает за его детекцию
- Каждая клетка предсказывает B bounding boxes и значения того, насколько вероятно, что выделен объект и насколько точно это сделано
- Формально это определяется как $Pr(Object) * IOU_{pred}^{truth}$
- Если в клетке нет объектов значение должно быть нулевым. Иначе, значение должно равняться intersection over union (IOU) между предсказанным и истинным значением.

YOLO

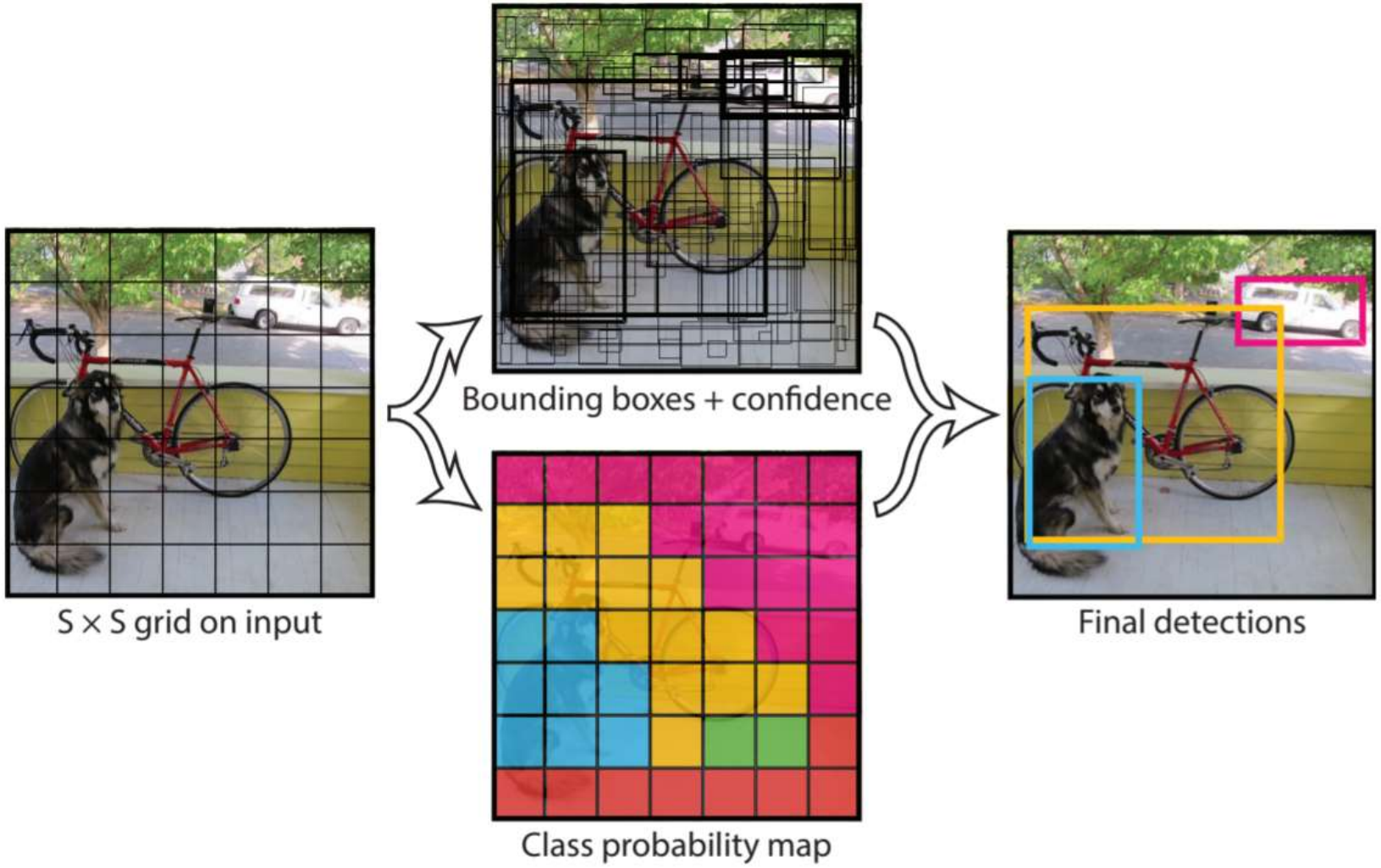
- Каждый bounding box состоит из 5 значений: x, y, w, h, и confidence
- Координаты (x, y) задают центр, по отношению к клетке сетки.
- Значения w и h задают ширину и длину рамки.
- Confidence представляет IOU между предсказанным и реальным значениями. Также в каждой клетке предсказывается вероятность $Pr(Class_i | Object)$.

YOLO



- Нейросеть состоит из 24 сверточных слоев, после которых идут 2 полносвязных
- Сверточные слои предобучаются на ImageNet на изображениях размером 224 × 224 пикселей, после чего разрешение удваивается для детекции.
- Fast YOLO использует 9 сверточных слоев.

YOLO


Bounding boxes + confidence
S × S grid on input
Class probability map
Final detections

YOLO

loss function:

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

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

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

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

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

YOLO

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

- 1_{ij}^{obj} — Индикатор того, что объект находится в i-ой клетке и j-й bounding box predictor ”отвечает” за его обнаружение
- Слагаемое считает насколько точно предсказано положение bounding box

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

- Слагаемое считает насколько точно предсказаны размеры bounding box

YOLO

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

- $\hat{C}_i$ — intersection over union
- C_i — confidence
- 1_{ij}^{noobj} — индикатор отсутствия объекта

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

- Classification loss

YOLO

- Fast R-CNN достигает лучших значений mAP на VOC 2007 test set в 71.8%. При комбинировании с YOLO, mAP увеличивается на 3.2%, достигая 75.0%.

	mAP	Combined	Gain
Fast R-CNN	71.8	-	-
Fast R-CNN (2007 data)	**66.9**	72.4	.6
Fast R-CNN (VGG-M)	59.2	72.4	.6
Fast R-CNN (CaffeNet)	57.1	72.1	.3
YOLO	63.4	**75.0**	**3.2**

Table 2: Model combination experiments on VOC 2007. We examine the effect of combining various models with the best version of Fast R-CNN. Other versions of Fast R-CNN provide only a small benefit while YOLO provides a significant performance boost.

Real-Time Detectors	Train	mAP	FPS
100Hz DPM [31]	2007	16.0	100
30Hz DPM [31]	2007	26.1	30
Fast YOLO	2007+2012	52.7	**155**
YOLO	2007+2012	**63.4**	45
Less Than Real-Time			
Fastest DPM [38]	2007	30.4	15
R-CNN Minus R [20]	2007	53.5	6
Fast R-CNN [14]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[28]	2007+2012	73.2	7
Faster R-CNN ZF [28]	2007+2012	62.1	18
YOLO VGG-16	2007+2012	66.4	21

YOLO

- В отличие от подходов, основанных на классификации, YOLO обучается, используя loss функцию, прямо соотносящуюся с результатами детекции (detection performance).
- Вся модель обучается целиком.
- YOLO также хорошо справляется с обобщением в новых областях, что делает возможным его применения в задачах быстрого и достаточно точного обнаружения объектов.

Ссылки

1. Fast R-CNN

   https://arxiv.org/pdf/1504.08083.pdf

2. You Only Look Once: Unified, Real-Time Object Detection

   https://arxiv.org/pdf/1506.02640.pdf