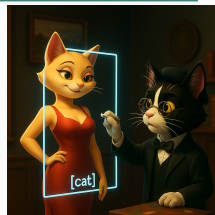
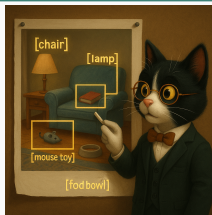


Сергей Николенко

СПбГУ — Санкт-Петербург

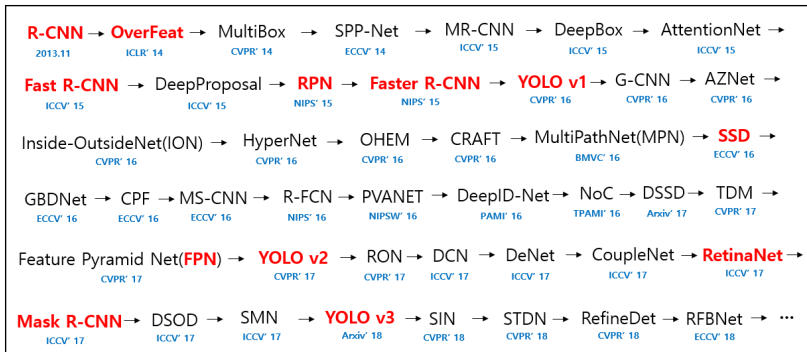
18 сентября 2025 г.

Random facts:



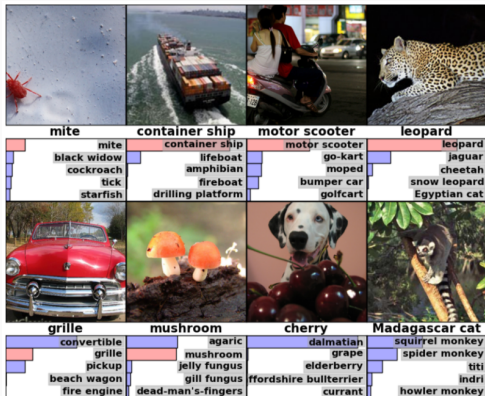
- 18 сентября 1066 г. Гаральд Гардерада вместе с Тостигом Годвинсоном высадились в устье реки Хамбер; через два дня он разгромил английские войска в битве у Фулфорда, но через пять дней потерпел сокрушительное поражение при Стэмфорд-Бридж
- 18 сентября 1698 г. в Бастилию был переведён таинственный узник под номером 64489001, известный как «Железная маска»; после его смерти в 1703 г. всю мебель и одежду специально уничтожили, закрасив стены и расплавив все металлические вещи
- 18 сентября 1830 г. лошадь победила в гонке на 9 миль от Райлиз Таверн до Балтимора между лошадью и первым американским паровозом, а 18 сентября 1893 г. было завершено строительство Великой Северной железной дороги между Миссисипи и Тихим океаном
- 18 сентября 1870 г. Генри Уошберн открыл гейзер Old Faithful, который впоследствии стал источником одного из самых известных датасетов для кластеризации
- 18 сентября 1886 г. поэт Жан Мореас опубликовал в газете «Фигаро» статью, ставшую манифестом литераторов-символистов; сами термины «символист» и «символизм» были предложены для замены слова «декадент», которую раньше применяли критики для Бодлера и Малларме

РАСПОЗНАВАНИЕ ОБЪЕКТОВ



ПОСТАНОВКА ЗАДАЧИ И ДАТАСЕТЫ

- Мы обсуждали классификацию: Imagenet ($> 15\text{M}$ картинок, $\approx 22\text{K}$ категорий).
- Размечены вручную (Amazon Mechanical Turk). Стандартный датасет для обучения всех свёрточных сетей для обработки фотографий.



ПОСТАНОВКА ЗАДАЧИ И ДАТАСЕТЫ

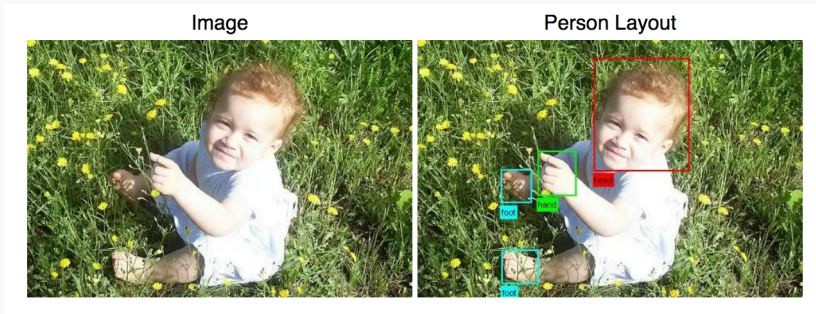
- Но ведь надо же ещё и понять, где эти объекты (и распознавать несколько объектов на одной и той же фотографии).
- PASCAL VOC для сегментации:



- Относительно маленький, так что сначала всё предобучается на *ImageNet*.

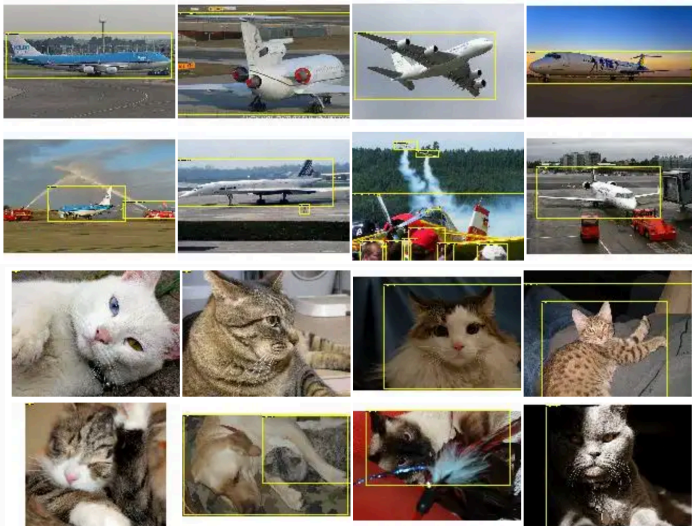
ПОСТАНОВКА ЗАДАЧИ И ДАТАСЕТЫ

- Классические статьи используют VOC07 (9963 изображений, 24640 объектов):



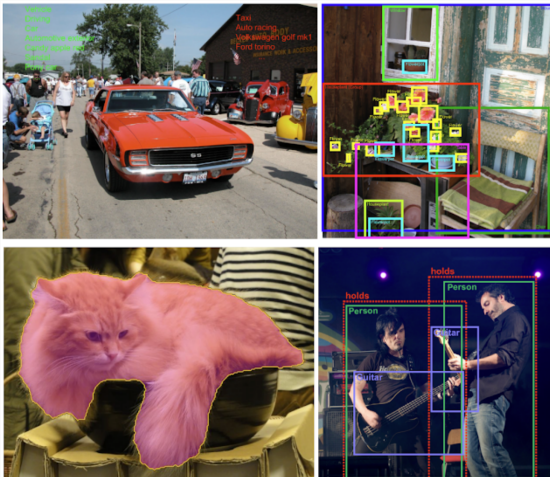
ПОСТАНОВКА ЗАДАЧИ И ДАТАСЕТЫ

- Или VOC12 (11540 изображений, 27450 объектов):



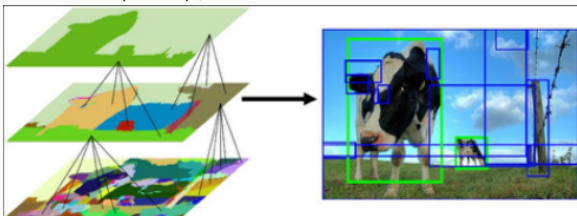
ПОСТАНОВКА ЗАДАЧИ И ДАТАСЕТЫ

- Сейчас главный датасет — наверное, *OpenImages* от *Google*
- OpenImages V6: 15M bbox'ов, 2.7M сегментированных изображений, 60M меток

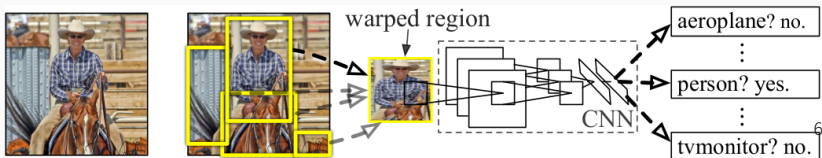


ДВУХЭТАПНОЕ РАСПОЗНАВАНИЕ ОБЪЕКТОВ

- R-CNN: Region-based Convolutional Network method (Girshick et al., 2014).
- Идея: будем выделять участки каким-то внешним алгоритмом (например, selective search).



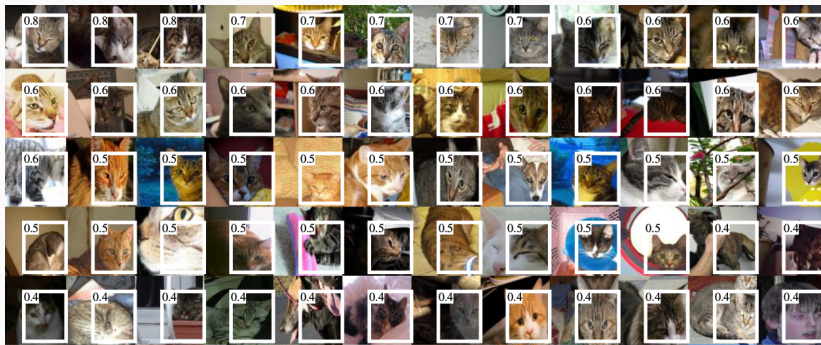
- Затем на них выделять признаки CNN (предобученной на ImageNet и fine-tuned на нужном) и классифицировать.



- Можно визуализировать: берём нейрон высокого уровня, показываем, на каких участках он активируется сильнее.



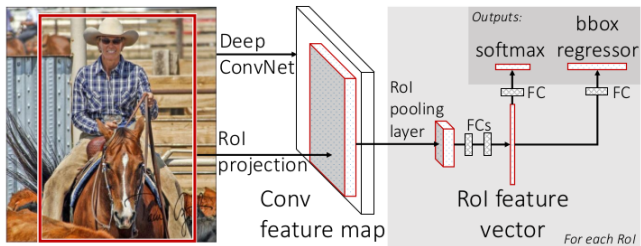
- Можно визуализировать: берём нейрон высокого уровня, показываем, на каких участках он активируется сильнее.



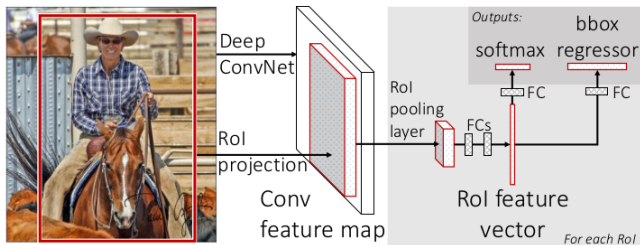
- Но у R-CNN есть проблемы:
 - надо обучать в несколько приёмов (сначала CNN, потом SVM на её признаках, потом bounding box регрессоры)
 - обучать долго (надо выделить признаки из каждого участка в тренировочном датасете и записать на диск для обучения SVM и регрессоров);
 - распознавать очень медленно (47с на картинку на GPU).
- Причина в том, что для каждого участка надо делать проход по CNN.

FAST R-CNN

- Поэтому – *Fast R-CNN*.
- RoI (region of interest) projection собирает признаки из заданного участка.
- Теперь нужно считать признаки для всей картинки только один раз.

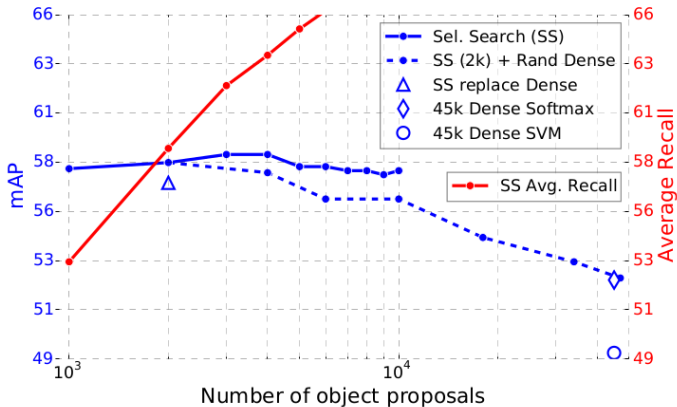


- Функция потерь складывается из двух: ошибка классификации и ошибка bounding box регрессии.
- Получается на два порядка быстрее и ничем не хуже (иногда лучше) по качеству.



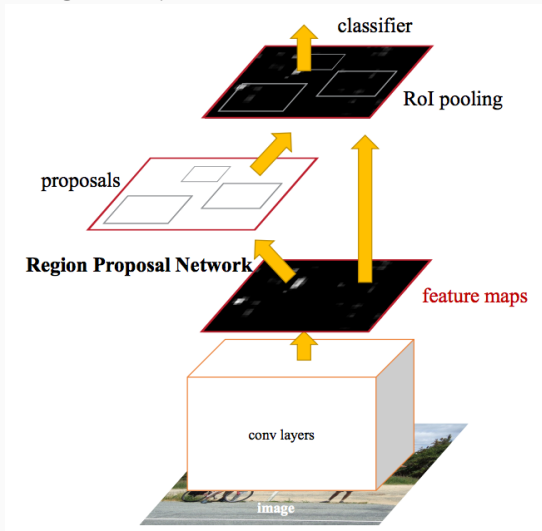
FAST R-CNN

- Сами кандидаты в участки по-прежнему из selective search.
- Выяснилось, что добавлять больше кандидатов не очень помогает.



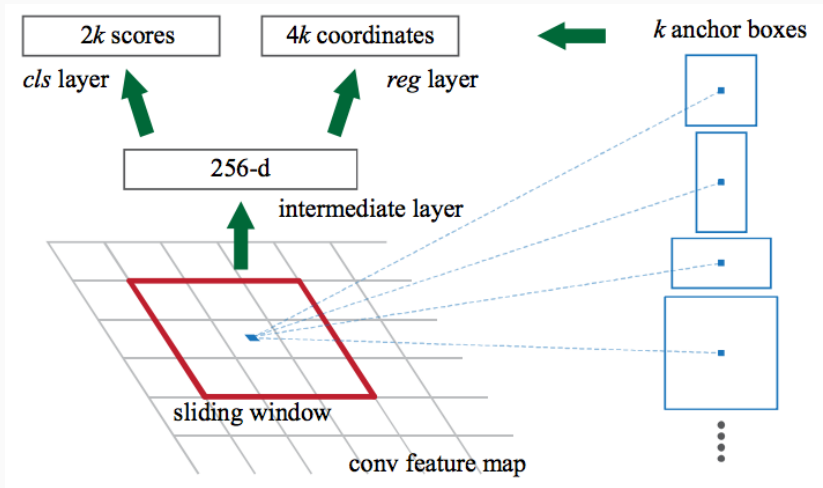
- Осталось одно узкое место: selective search для выбора участков.
- Оказывается, это тоже можно вставить в сеть!
- Получается *Faster R-CNN* (Ren et al., 2015).

- Отдельная Region Proposal Network:

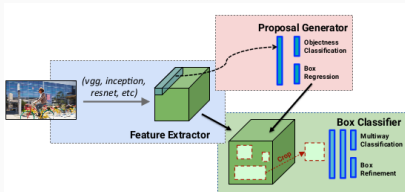


FASTER R-CNN

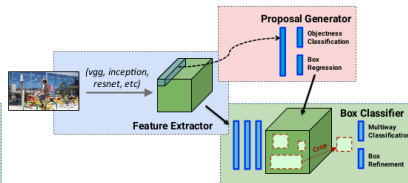
- Оценивает каждое отдельное окно из существующих anchor boxes:



- Можно и ещё больше сэкономить, не вычисляя на каждом участке вообще никаких сложных слоёв.
- R-FCN (Region-based Fully Convolutional Network) вырезает признаки из самого последнего перед классификацией слоя (Dai et al., 2016).
- Вот в чём разница:



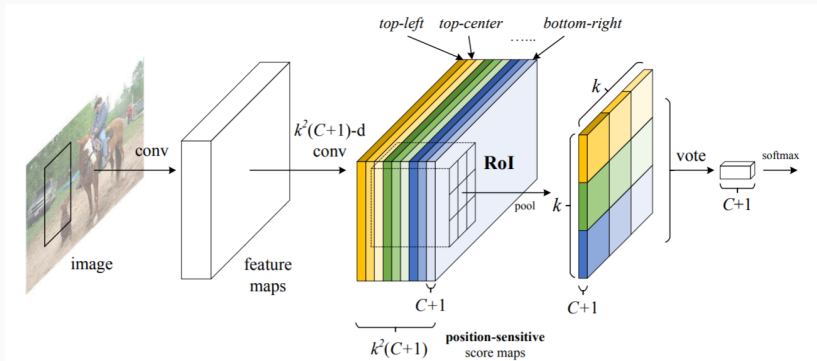
(b) Faster RCNN.



(c) R-FCN.

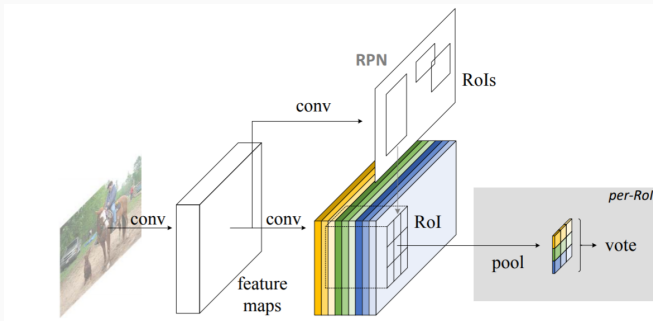
- (Dai et al., 2016), R-FCN (Region-based Fully Convolutional Network)
- Можно ещё больше сэкономить в Faster R-CNN, не вычисляя на каждом участке вообще никаких сложных слоёв.
- Но как? Идея такая:
 - лучшие CNN для классификации (ResNet, GoogLeNet) давно fully convolutional, т.е. полносвязный слой только в конце
 - но с object detection почему-то это не работает
 - Faster R-CNN пришлось вставить противоестественный region proposal слой
 - проблема в том, что CNN дают translation invariance, а для object detection этого как раз не надо

- Решение R-FCN в том, чтобы сделать специальные карты признаков, которые как раз position-sensitive:

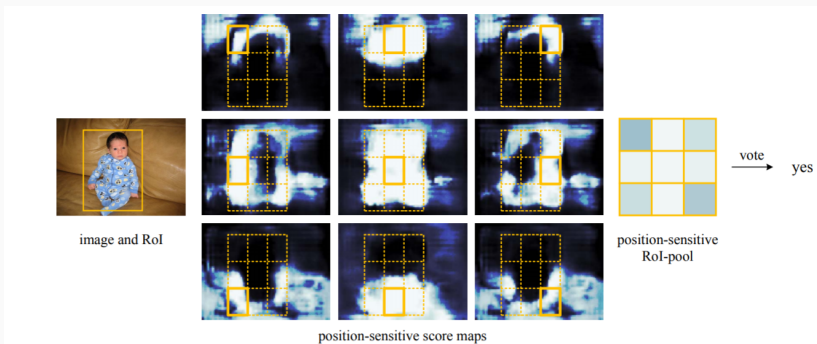


- Пытаемся их обучить так, чтобы они кодировали информацию относительно конкретной позиции («левая часть объекта», «правая часть объекта»)

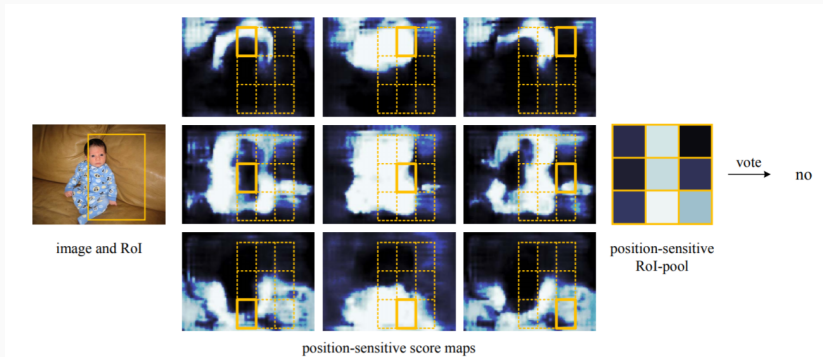
- В остальном как Faster R-CNN – есть отдельная RPN
- А потом RoI pooling, который учитывает наши новые feature maps:



- Визуализация R-FCN активаций для 3×3 position-sensitive карт признаков:



- А вот что будет для неправильного окна:



- Это неплохая архитектура и по сегодняшним меркам:



- Циферки тоже отличные:

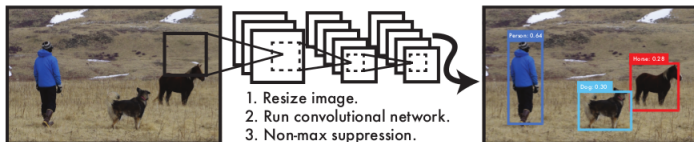
	depth of per-RoI subnetwork	training w/ OHEM?	train time (sec/img)	test time (sec/img)	mAP (%) on VOC07
Faster R-CNN	10		1.2	0.42	76.4
R-FCN	0		0.45	0.17	76.6
Faster R-CNN	10	✓ (300 RoIs)	1.5	0.42	79.3
R-FCN	0	✓ (300 RoIs)	0.45	0.17	79.5
Faster R-CNN	10	✓ (2000 RoIs)	2.9	0.42	N/A
R-FCN	0	✓ (2000 RoIs)	0.46	0.17	79.3

	training data	mAP (%)	test time (sec/img)
Faster R-CNN [9]	07+12	76.4	0.42
Faster R-CNN +++ [9]	07+12+COCO	85.6	3.36
R-FCN	07+12	79.5	0.17
R-FCN multi-sc train	07+12	80.5	0.17
R-FCN multi-sc train	07+12+COCO	83.6	0.17

	training data	mAP (%)	test time (sec/img)
Faster R-CNN [9]	07++12	73.8	0.42
Faster R-CNN +++ [9]	07++12+COCO	83.8	3.36
R-FCN multi-sc train	07++12	77.6 [†]	0.17
R-FCN multi-sc train	07++12+COCO	82.0[‡]	0.17

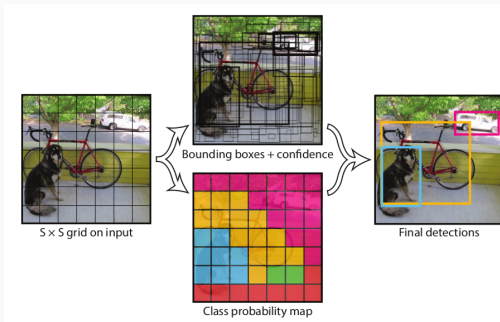
Одноэтапное распознавание объектов

- You only look once (YOLO; Redmon et al., 2016).
- За один проход ищет и прямоугольники с объектами (bounding boxes), и сами объекты.

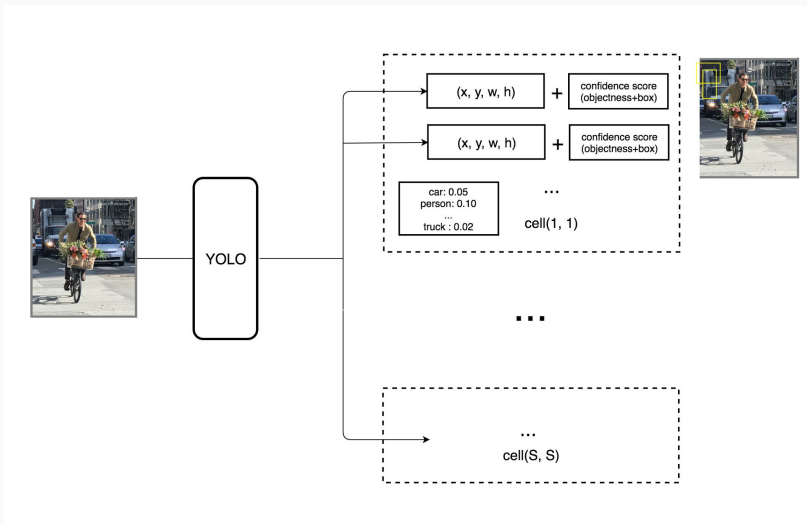


- Разбивает картинку на решётку $S \times S$.
- В каждой ячейке предсказывает и прямоугольники, и вероятности; и потом просто

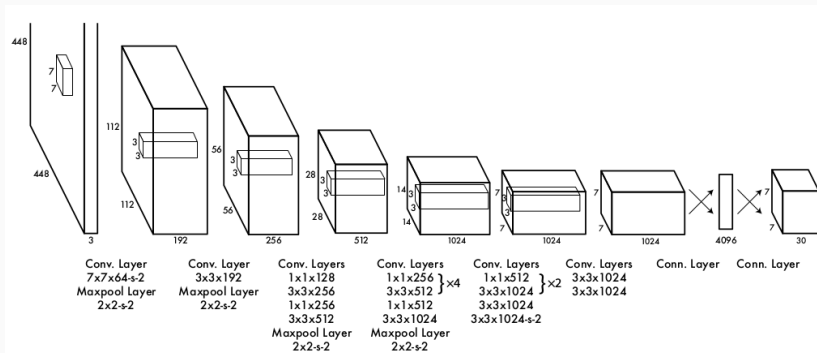
$$p(\text{class}_i \mid \text{obj})p(\text{obj})p(\text{bbox}).$$



- Вот так:

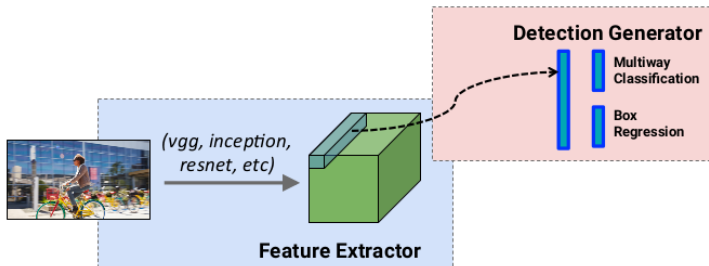


- Архитектура YOLO.

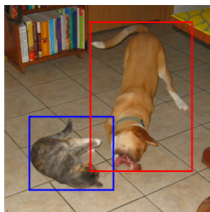


- Первый YOLO уступал по точности, но работал быстро.

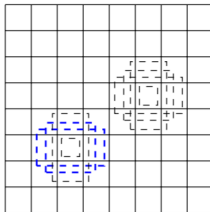
- Дальнейшее развитие идеи предсказывать всё сразу – single-shot detectors (SSD; Liu et al., 2016).
- Одна сеть, которая предсказывает
 - несколько меток классов;
 - несколько соответствующих позиций для anchor boxes (bounding boxes нескольких подходящих заранее выбранных размеров).



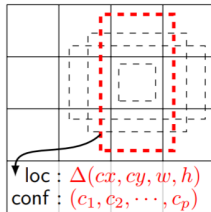
- Чтобы получить правильные ответы, мы сопоставляем стандартные anchor boxes с настоящими, и это правильные значения offsets и всего такого:



(a) Image with GT boxes



(b) 8×8 feature map

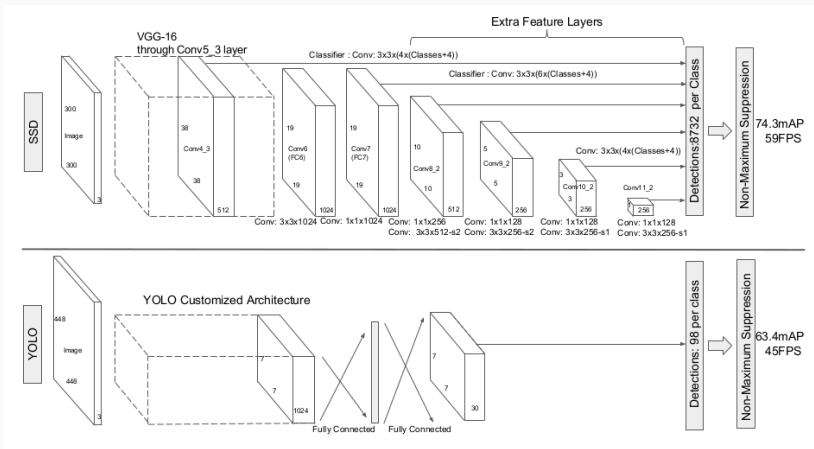


loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \dots, c_p)$

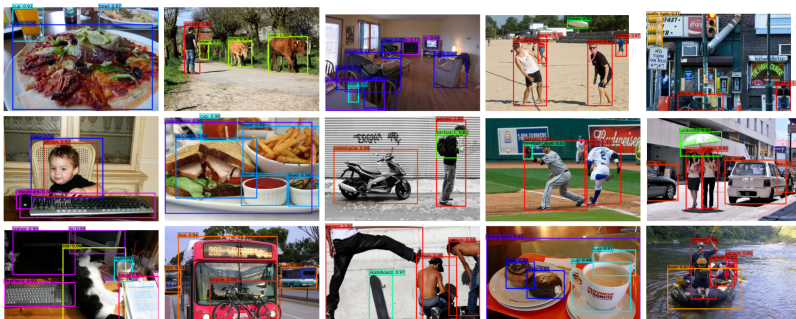
(c) 4×4 feature map

- Целевая функция – как MultiBox, но с учётом классов

- Архитектура чуть посложнее YOLO, очень много выходов:



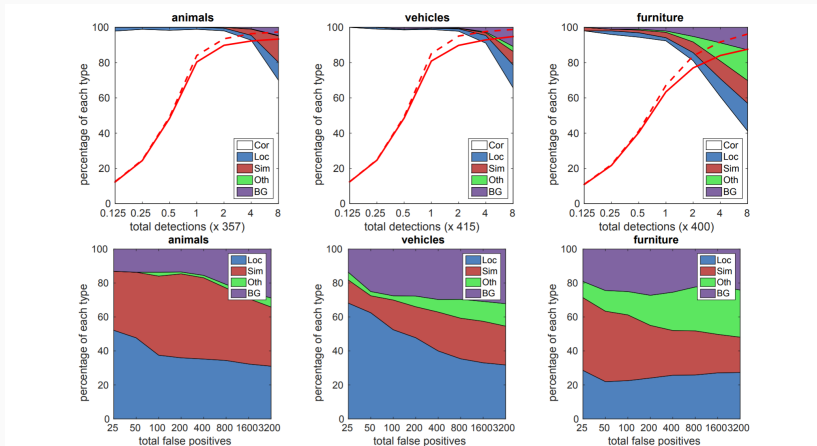
- Опять же, всё неплохо работает, но теперь уже везде всё будет неплохо работать:



- На тот момент был единственным real-time детектором, который давал $> 70\%$ mAP на Pascal VOC2007:

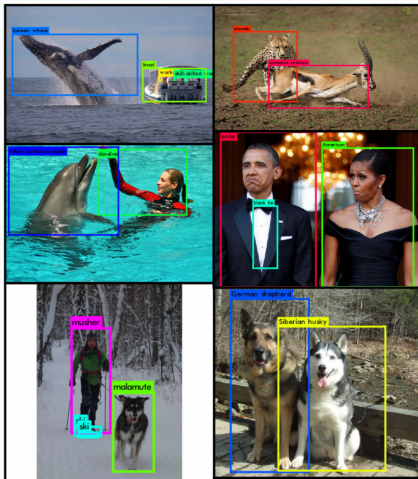
Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

- Можно проанализировать, где ошибки возникают:



YoLo v2, YOLO9000, YoLo v3

- (Redmon, Farhadi, 2016): следующая итерация YoLo, гораздо больше классов (9000):



- Что было не так: много ошибок локализации и низкий recall по сравнению с Faster R-CNN
- Что сделали:
 - добавили batchnorm на всех свёрточных уровнях, +2% mAP сразу
 - сделали отдельные классификатор высокого разрешения (448×448) вместо предобученного на ImageNet (256×256), ещё +4% mAP
 - YoLo предсказывал координаты bbox'ов, а лучше предсказывать offsets в каждой позиции свёрточным образом; класс теперь для каждого anchor box тоже предсказывается
 - размеры anchor boxes выбраны по bbox'ам из тренировочного набора, кластеризацией
 - добавили skip connection признаков с более раннего слоя, чтобы увеличить геометрическое разрешение
 - multi-scale training: обучаемся на разных размерах картинок

- Получается очень круто:

Detection Frameworks	Train	mAP	FPS
Fast R-CNN [5]	2007+2012	70.0	0.5
Faster R-CNN VGG-16[15]	2007+2012	73.2	7
Faster R-CNN ResNet[6]	2007+2012	76.4	5
YOLO [14]	2007+2012	63.4	45
SSD300 [11]	2007+2012	74.3	46
SSD500 [11]	2007+2012	76.8	19
YOLOv2 288 × 288	2007+2012	69.0	91
YOLOv2 352 × 352	2007+2012	73.7	81
YOLOv2 416 × 416	2007+2012	76.8	67
YOLOv2 480 × 480	2007+2012	77.8	59
YOLOv2 544 × 544	2007+2012	78.6	40

YoLo v2, YOLO9000, YoLo v3

- И всё это реально помогает:

	YOLO									YOLOv2
batch norm?		✓	✓	✓	✓	✓	✓	✓	✓	✓
hi-res classifier?			✓	✓	✓	✓	✓	✓	✓	✓
convolutional?				✓	✓	✓	✓	✓	✓	✓
anchor boxes?				✓	✓					
new network?					✓			✓	✓	✓
dimension priors?						✓	✓	✓	✓	✓
location prediction?						✓	✓	✓	✓	✓
passthrough?							✓	✓	✓	✓
multi-scale?								✓	✓	✓
hi-res detector?									✓	✓
VOC2007 mAP	63.4	65.8	69.5	69.2	69.6	74.4	75.4	76.8	78.6	

- По категориям:

Method	data	mAP	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv
Fast R-CNN [5]	07++12	68.4	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
Faster R-CNN [15]	07++12	70.4	84.9	79.8	74.3	53.9	49.8	77.5	75.9	88.5	45.6	77.1	55.3	86.9	81.7	80.9	79.6	40.1	72.6	60.9	81.2	61.5
YOLO [14]	07++12	57.9	77.0	67.2	57.7	38.3	22.7	68.3	55.9	81.4	36.2	60.8	48.5	77.2	72.3	71.3	63.5	28.9	52.2	54.8	73.9	50.8
SSD300 [11]	07++12	72.4	85.6	80.1	70.5	57.6	46.2	79.4	76.1	89.2	53.0	77.0	60.8	87.0	83.1	82.3	79.4	45.9	75.9	69.5	81.9	67.5
SSD512 [11]	07++12	74.9	87.4	82.3	75.8	59.0	52.6	81.7	81.5	90.0	55.4	79.0	59.8	88.4	84.3	84.7	83.3	50.2	78.0	66.3	86.3	72.0
ResNet [6]	07++12	73.8	86.5	81.6	77.2	58.0	51.0	78.6	76.6	93.2	48.6	80.4	59.0	92.1	85.3	84.8	80.7	48.1	77.3	66.5	84.7	65.6
YOLOv2 544	07++12	73.4	86.3	82.0	74.8	59.2	51.8	79.8	76.5	90.6	52.1	78.2	58.5	89.3	82.5	83.4	81.3	49.1	77.2	62.4	83.8	68.7

- А как сделать много классов? Новые проблемы: например, классы больше не взаимоисключающие («собака» и сотни пород собак)
- ImageNet берёт классы из WordNet – сети понятий, в которой есть иерархия, в виде направленного графа
- В YOLO сначала сделали из WordNet дерево, WordTree, а потом предсказывают условные вероятности на каждом уровне:

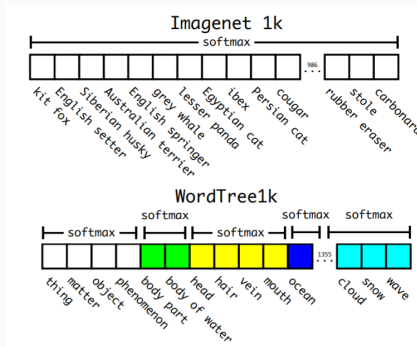
$$\begin{aligned} &Pr(\text{Norfolk terrier}|\text{terrier}) \\ &Pr(\text{Yorkshire terrier}|\text{terrier}) \\ &Pr(\text{Bedlington terrier}|\text{terrier}) \\ &\dots \end{aligned}$$

И ПОТОМ

$$\begin{aligned} Pr(\text{Norfolk terrier}) &= Pr(\text{Norfolk terrier}|\text{terrier}) \\ &\quad * Pr(\text{terrier}|\text{hunting dog}) \\ &\quad * \dots * \\ &\quad * Pr(\text{mammal}|Pr(\text{animal})) \\ &\quad * Pr(\text{animal}|\text{physical object}) \end{aligned}$$

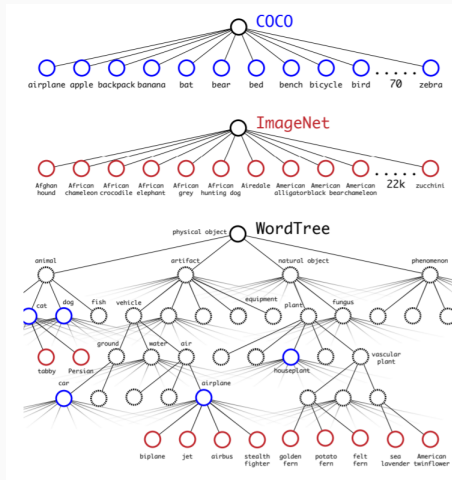
YoLo v2, YOLO9000, YoLo v3

- Это позволяет строить много маленьких softmax'ов, которые проще обучать:



YoLo v2, YOLO9000, YoLo v3

- И датасеты можно объединить даже



СПАСИБО!

Спасибо за внимание!

