

РАСПОЗНАВАНИЕ ОБЪЕКТОВ

Сергей Николенко

СПбГУ — Санкт-Петербург

25 сентября 2026 г.

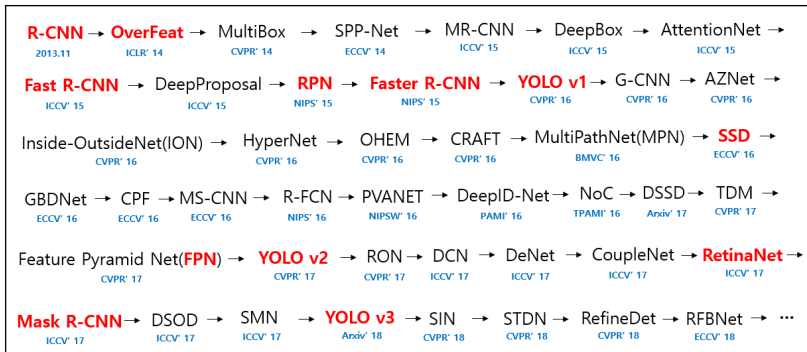
Random facts:

- 25 сентября 1066 г. битва при Стамфорд-Бридже положила конец вторжению викингов в Англию
- 25 сентября 1396 г. крестоносцы были разбиты при Никополе; последний большой крестовый поход закончился ничем
- 25 сентября 1818 г. английский врач Джеймс Бландел впервые перелил кровь от человека человеку
- 25 сентября 1926 г. Ford перевёл заводы на пятидневную рабочую неделю; восьмичасовой день там ввели ещё в 1914-м
- 25 сентября 1949 г. ТАСС сообщило, что атомная бомба у СССР уже есть; испытание провели месяцем раньше
- 25 сентября 1957 г. девять чернокожих школьников под охраной федеральных войск отучились полный день в школе Литл-Рока
- 25 сентября 1981 г. Сандра Дэй О'Коннор стала первой женщиной в Верховном суде США



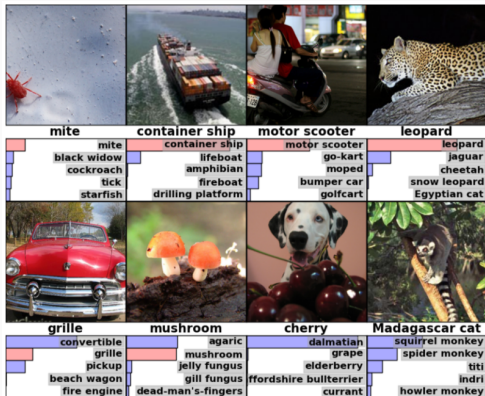
- В прошлый раз мы прошли зоопарк свёрточных архитектур: от VGG и Inception до ResNet, SNet, MobileNet и NAS — то есть научились извлекать из картинки хорошие признаки.
- Сегодня применим их к двум задачам, где ответ сложнее одной метки: *распознавание объектов* (где на картинке что) и *сегментация* (какому объекту принадлежит каждый пиксель).
- Материала много, так что идём быстро: двухэтапные детекторы, одноэтапные, пирамиды признаков — а потом посмотрим, что стало с YOLO за последние шесть лет.

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

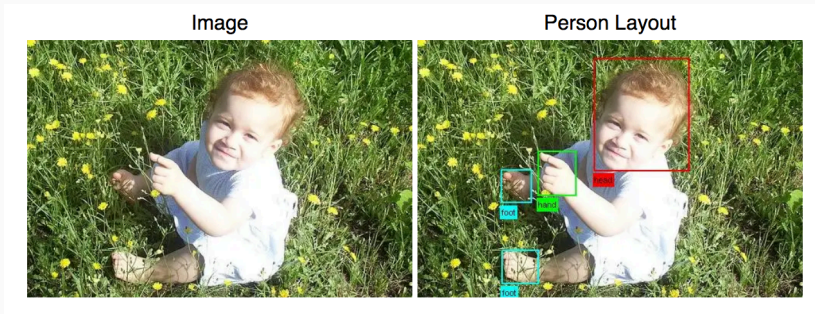
- Но ведь надо же ещё и понять, где эти объекты (и распознавать несколько объектов на одной и той же фотографии).
- PASCAL VOC для сегментации:



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

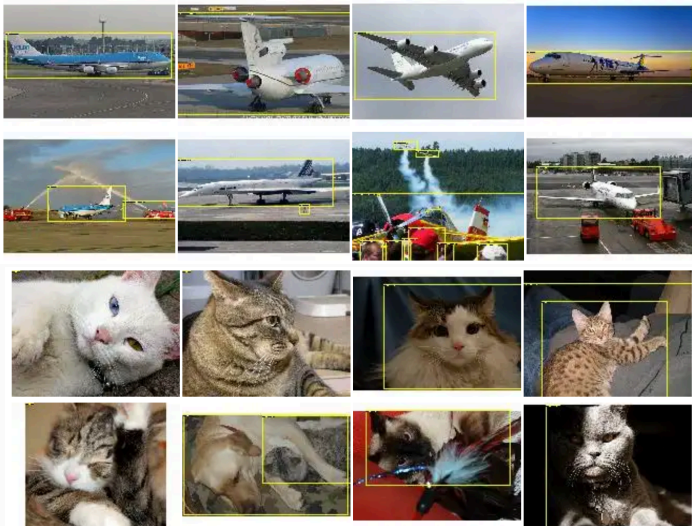
ПОСТАНОВКА ЗАДАЧИ И ДАТАСЕТЫ

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



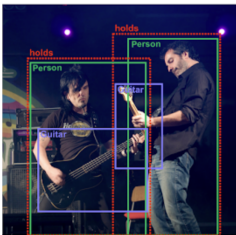
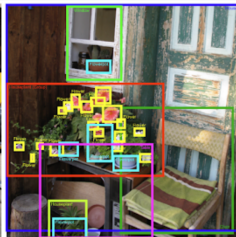
ПОСТАНОВКА ЗАДАЧИ И ДАТАСЕТЫ

- Или VOC12 (11540 изображений, 27450 объектов):



ПОСТАНОВКА ЗАДАЧИ И ДАТАСЕТЫ

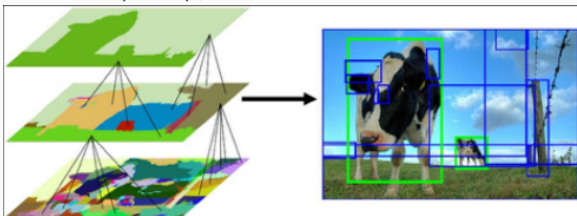
- Сейчас главный датасет — наверное, *OpenImages* от *Google*
- OpenImages V6: 15M bbox'ов, 2.7M сегментированных изображений, 60M меток



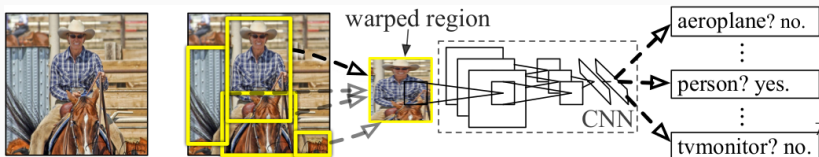
ДВУХЭТАПНОЕ РАСПОЗНАВАНИЕ ОБЪЕКТОВ

R-CNN

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



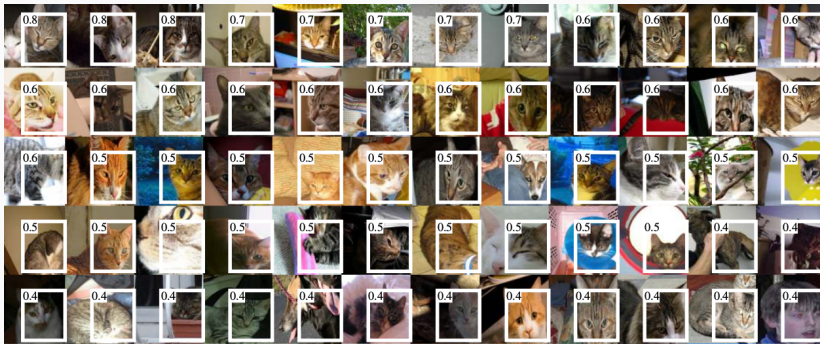
- Затем на них выделять признаки CNN (предобученной на ImageNet и fine-tuned на нужном) и классифицировать.



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



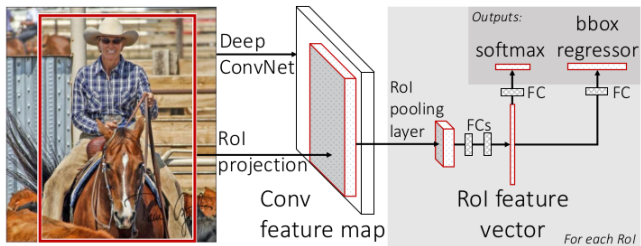
- Можно визуализировать: берём нейрон высокого уровня, показываем, на каких участках он активируется сильнее.



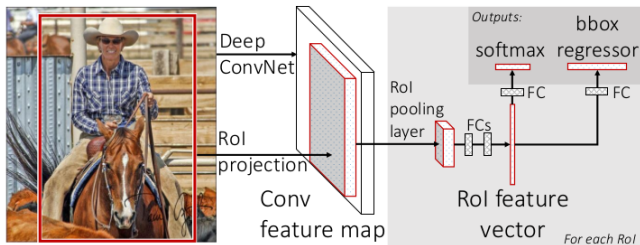
- Но у R-CNN есть проблемы:
 - надо обучать в несколько приёмов (сначала CNN, потом SVM на её признаках, потом bounding box регрессоры)
 - обучать долго (надо выделить признаки из каждого участка в тренировочном датасете и записать на диск для обучения SVM и регрессоров);
 - распознавать очень медленно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

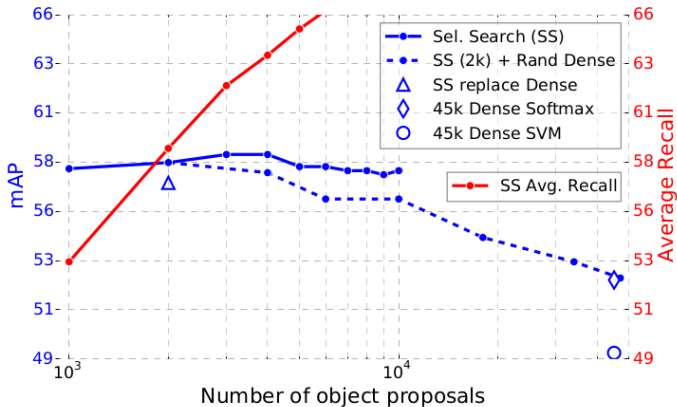


- Функция потерь складывается из двух: ошибка классификации и ошибка bounding box регрессии.
- Получается на два порядка быстрее и ничем не хуже (иногда лучше) по качеству.



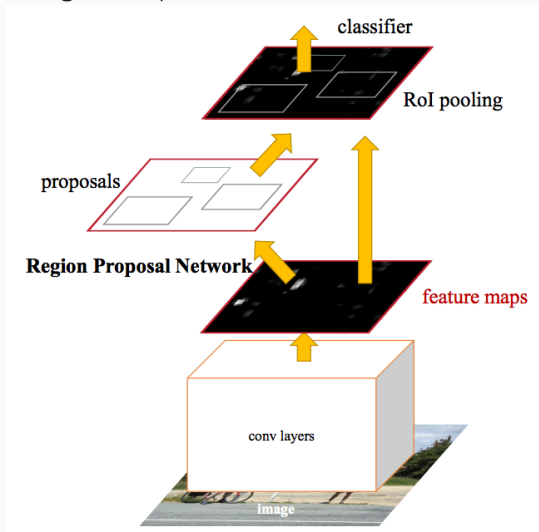
FAST R-CNN

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



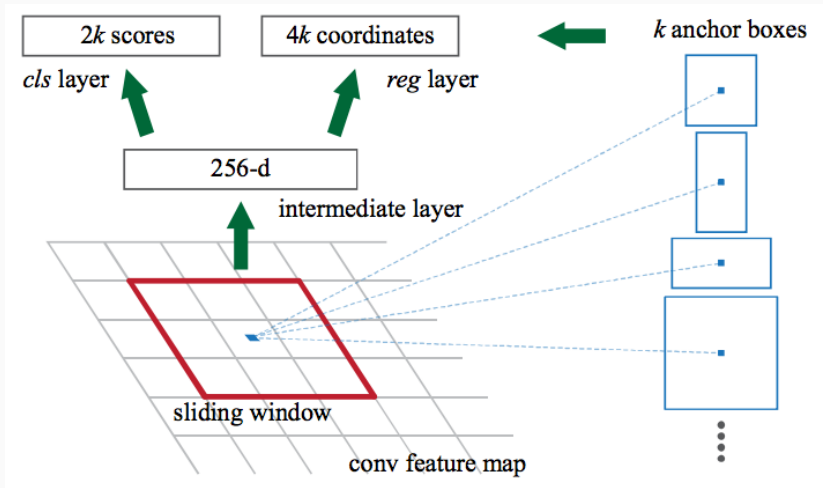
- Осталось одно узкое место: selective search для выбора участков.
- Оказывается, это тоже можно вставить в сеть!
- Получается *Faster R-CNN* (Ren et al., 2015).

- Отдельная Region Proposal Network:



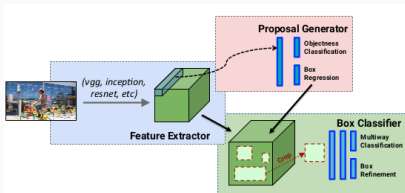
FASTER R-CNN

- Оценивает каждое отдельное окно из существующих anchor boxes:

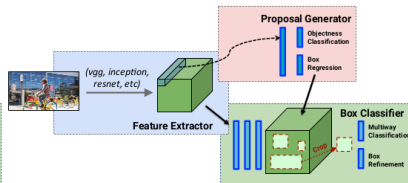


FASTER R-CNN

- 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:



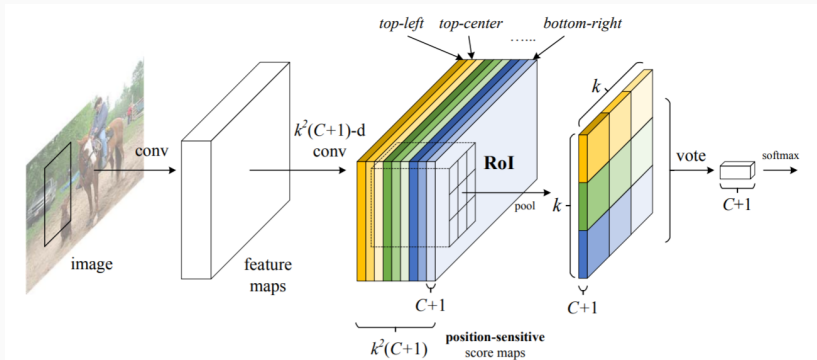
(b) Faster RCNN.



(c) R-FCN.

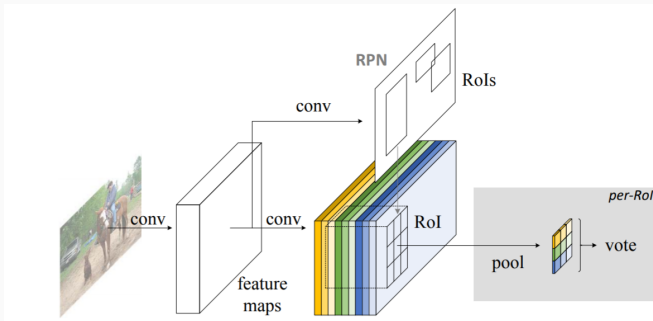
- (Dai et al., 2016), R-FCN (Region-based Fully Convolutional Network)
- Можно ещё больше сэкономить в Faster R-CNN, не вычисляя на каждом участке вообще никаких сложных слоёв.
- Но как? Идея такая:
 - лучшие CNN для классификации (ResNet, GoogLeNet) давно fully convolutional, т.е. полносвязный слой 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:

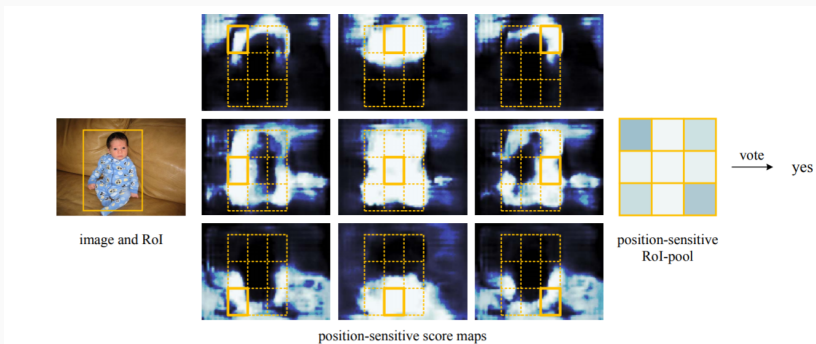


- Пытаемся их обучить так, чтобы они кодировали информацию относительно конкретной позиции («левая часть объекта», «правая часть объекта»)

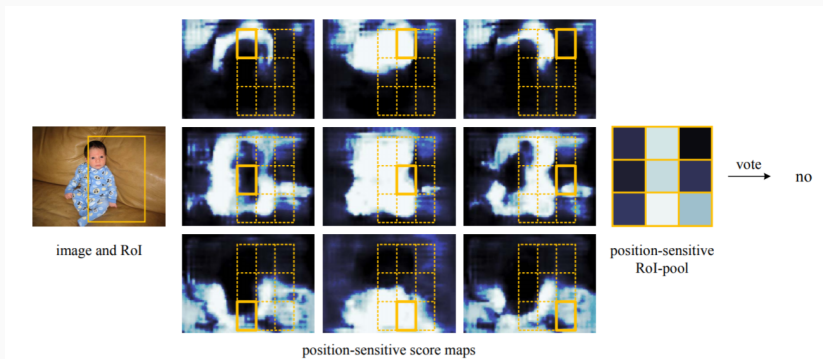
- В остальном как Faster R-CNN – есть отдельная RPN
- А потом RoI pooling, который учитывает наши новые feature maps:



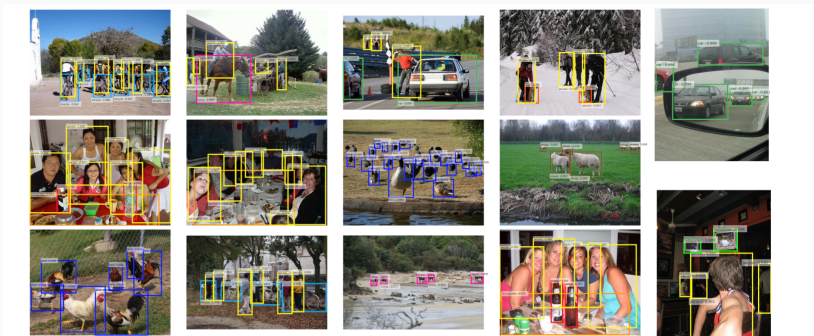
- Визуализация R-FCN активаций для 3×3 position-sensitive карт признаков:



- А вот что будет для неправильного окна:



- Это неплохая архитектура и по сегодняшним меркам:



- Циферки тоже отличные:

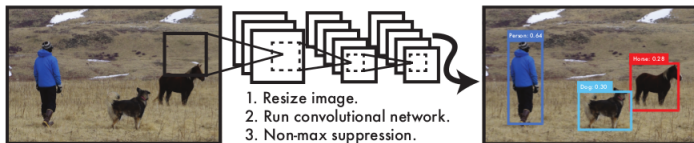
	depth of per-RoI subnetwork	training w/ OHEM?	train time (sec/img)	test time (sec/img)	mAP (%) on VOC07
Faster R-CNN	10		1.2	0.42	76.4
R-FCN	0		0.45	0.17	76.6
Faster R-CNN	10	✓ (300 RoIs)	1.5	0.42	79.3
R-FCN	0	✓ (300 RoIs)	0.45	0.17	79.5
Faster R-CNN	10	✓ (2000 RoIs)	2.9	0.42	N/A
R-FCN	0	✓ (2000 RoIs)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

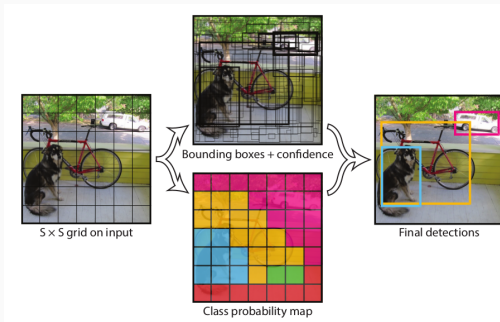
Одноэтапное распознавание объектов

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

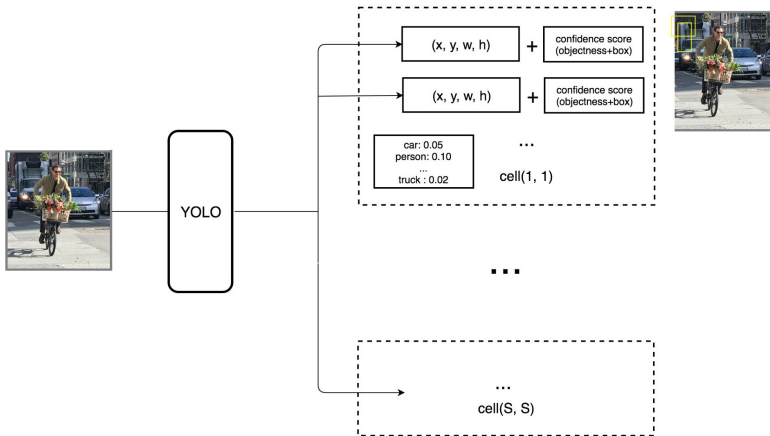


- Разбивает картинку на решётку $S \times S$.
- В каждой ячейке предсказывает и прямоугольники, и вероятности; и потом просто

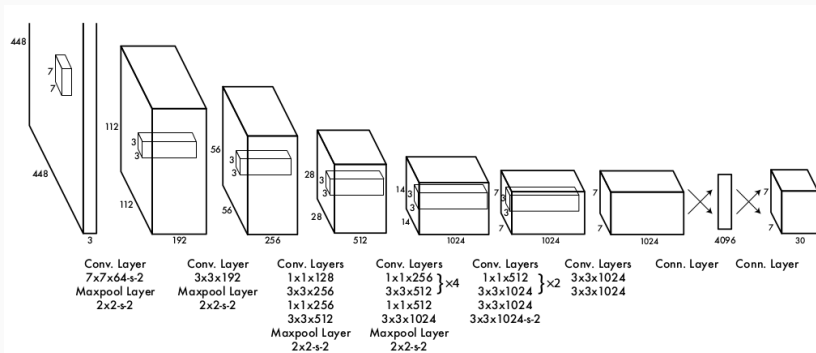
$$p(\text{class}_i \mid \text{obj})p(\text{obj})p(\text{bbox}).$$



- Вот так:

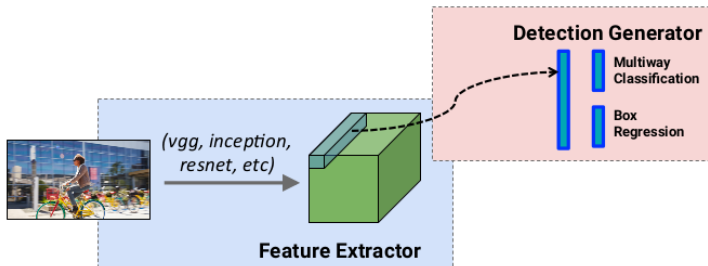


- Архитектура YOLO.

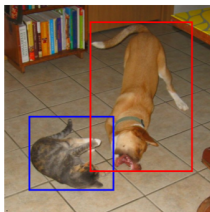


- Первый YOLO уступал по точности, но работал быстро.

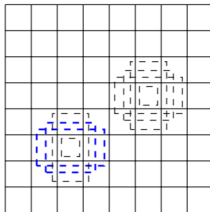
- Дальнейшее развитие идеи предсказывать всё сразу – single-shot detectors (SSD; Liu et al., 2016).
- Одна сеть, которая предсказывает
 - несколько меток классов;
 - несколько соответствующих позиций для anchor boxes (bounding boxes нескольких подходящих заранее выбранных размеров).



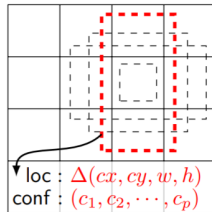
- Чтобы получить правильные ответы, мы сопоставляем стандартные anchor boxes с настоящими, и это правильные значения offsets и всего такого:



(a) Image with GT boxes



(b) 8×8 feature map

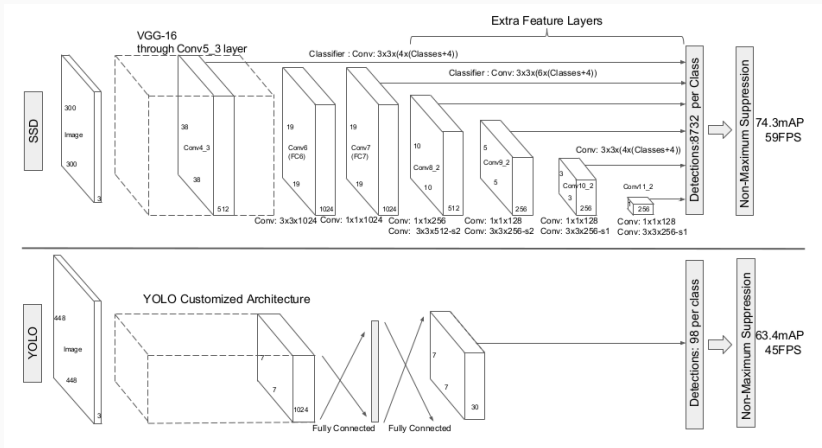


loc : $\Delta(cx, cy, w, h)$
conf : $(c_1, c_2, \dots, c_p)$

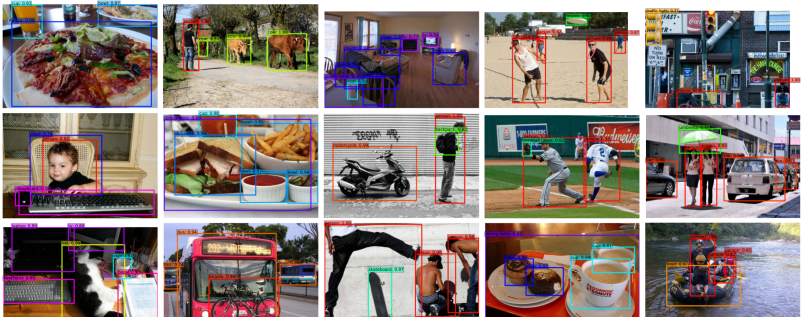
(c) 4×4 feature map

- Целевая функция – как MultiBox, но с учётом классов

- Архитектура чуть посложнее YOLO, очень много выходов:



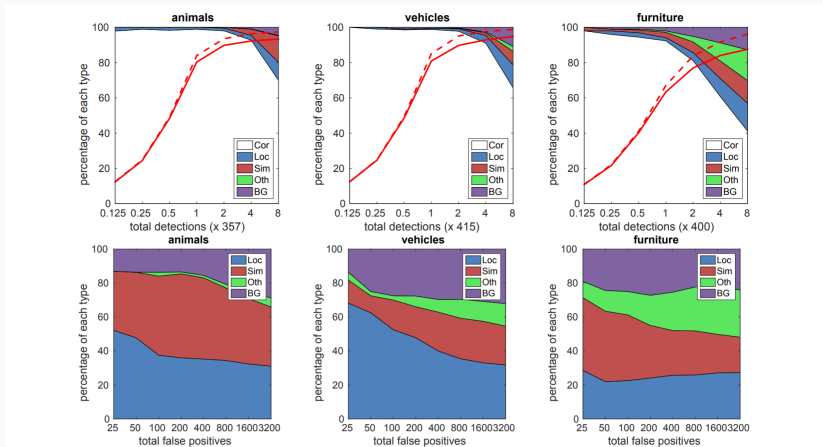
- Опять же, всё неплохо работает, но теперь уже везде всё будет неплохо работать:



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

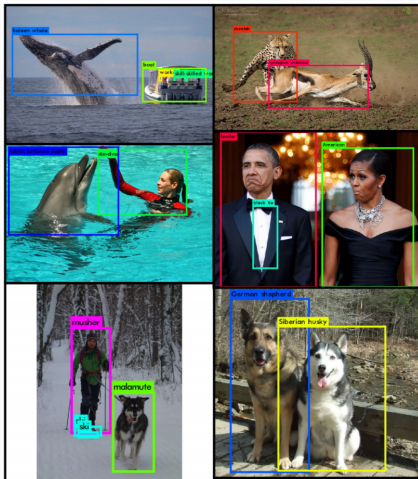
Method	mAP	FPS	batch size	# Boxes	Input resolution
Faster R-CNN (VGG16)	73.2	7	1	~ 6000	$\sim 1000 \times 600$
Fast YOLO	52.7	155	1	98	448×448
YOLO (VGG16)	66.4	21	1	98	448×448
SSD300	74.3	46	1	8732	300×300
SSD512	76.8	19	1	24564	512×512
SSD300	74.3	59	8	8732	300×300
SSD512	76.8	22	8	24564	512×512

- 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

YoLo v2, YOLO9000, YoLo v3

- 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:

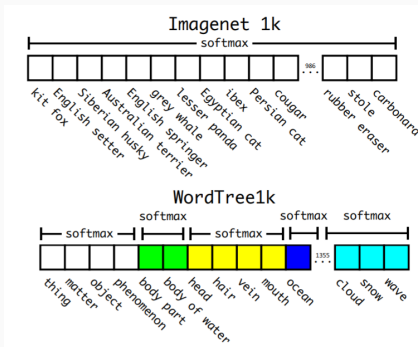
$$\begin{aligned} &Pr(\text{Norfolk terrier}|\text{terrier}) \\ &Pr(\text{Yorkshire terrier}|\text{terrier}) \\ &Pr(\text{Bedlington terrier}|\text{terrier}) \\ &\dots \end{aligned}$$

И ПОТОМ

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

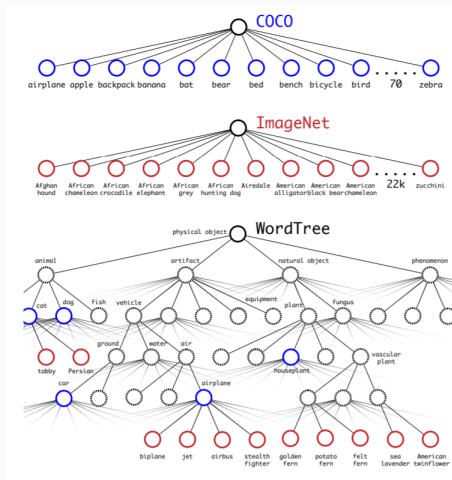
YoLo v2, YOLO9000, YoLo v3

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



YoLo v2, YOLO9000, YoLo v3

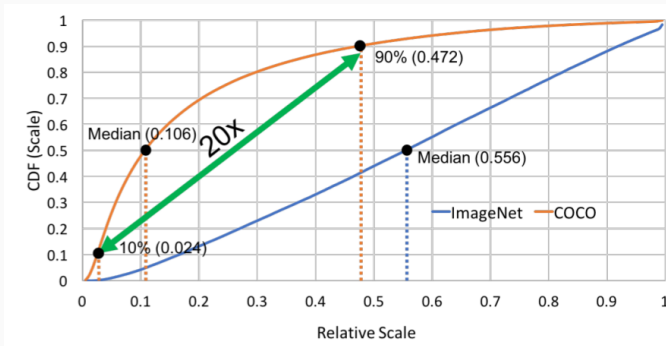
- И датасеты можно объединить даже



ПРОБЛЕМЫ С МАСШТАБОМ

ПРОБЛЕМЫ С МАСШТАБОМ

- Детекторы все хорошие, но почему-то распознают максимум 50-60% объектов. Почему на COCO такие плохие результаты?
- Потому что маленькие объекты плохо распознаются, а их в жизни много, и разброс масштабов очень большой.



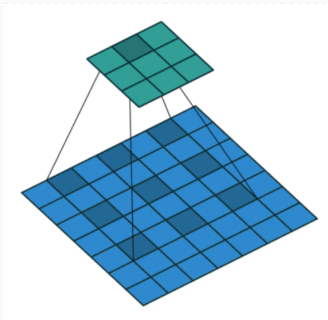
- Почему так?

ПРОБЛЕМЫ С МАСШТАБОМ

- Effective stride: к моменту region proposal network уже геометрия исходной картинки сжимается.
- В базовой конструкции Faster R-CNN получается effective stride 16, т.е. от объекта 16×16 пикселей остаётся 1 пиксель.
- А если пытаться уменьшить stride, то сети выходят совсем гигантские и неподъёмные. Что делать?..



- Во-первых, свёртки в таких сетях часто немного другие.
Dilated/atrous convolutions:



- Кстати:
https://github.com/vdumoulin/conv_arithmetic
- Зачем нужны dilated convolutions?

ПРОБЛЕМЫ С МАСШТАБОМ

- А чтобы не повторяясь покрыть гораздо большее receptive field за меньшее число слоёв:

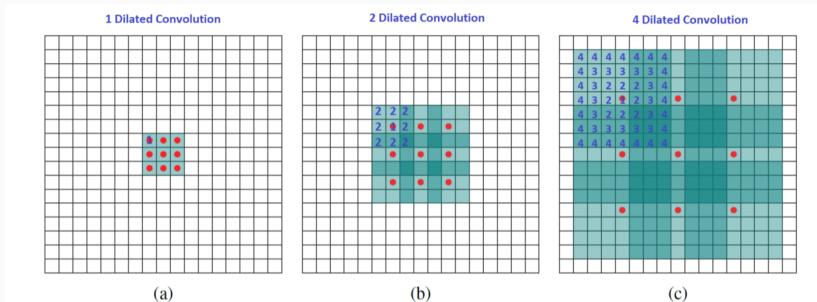
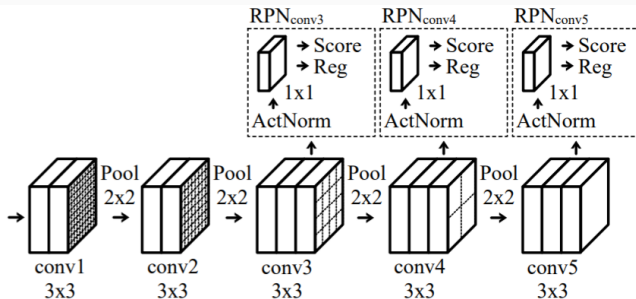


Figure 1: Systematic dilation supports exponential expansion of the receptive field without loss of resolution or coverage. (a) F_1 is produced from F_0 by a 1-dilated convolution; each element in F_1 has a receptive field of 3×3 . (b) F_2 is produced from F_1 by a 2-dilated convolution; each element in F_2 has a receptive field of 7×7 . (c) F_3 is produced from F_2 by a 4-dilated convolution; each element in F_3 has a receptive field of 15×15 . The number of parameters associated with each layer is identical. The receptive field grows exponentially while the number of parameters grows linearly.

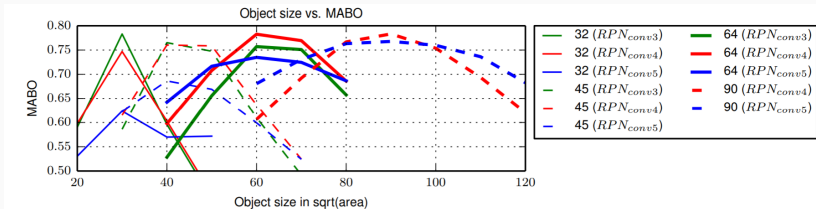
ПРОБЛЕМЫ С МАСШТАБОМ

- Dilated convolutions используются в Faster R-CNN, R-FCN и других.
- (Yu, Koltun, 2016): Multi-scale context aggregation by dilated convolutions; собирают весь контекст с разных dilated convolutions вместе, применяют к сегментации.
- Можно ещё пытаться сделать несколько разных RPN; (Eggert et al., 2017) – возьмём базовый Faster R-CNN и встроим в него три RPN с разным масштабом.

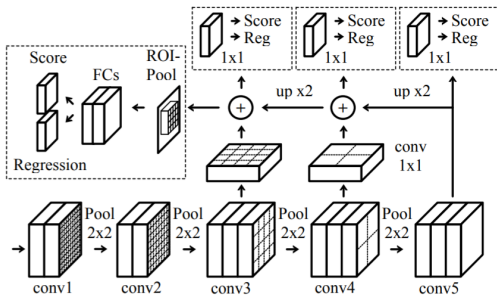


ПРОБЛЕМЫ С МАСШТАБОМ

- Разные RPN нужны для объектов разного размера:

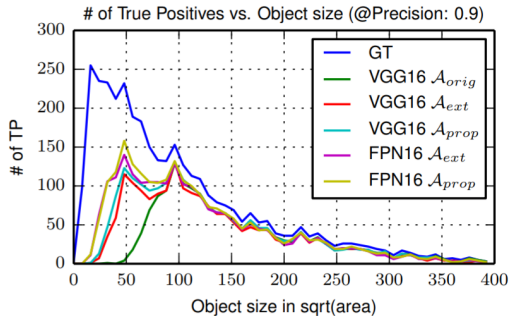


- И делают нечто похожее на FPN, чтобы объединить признаки:



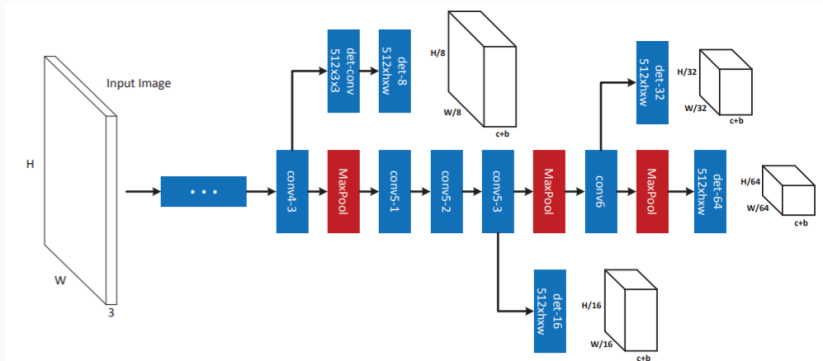
ПРОБЛЕМЫ С МАСШТАБОМ

- Получается хорошо именно за счёт маленьких объектов:



ПРОБЛЕМЫ С МАСШТАБОМ

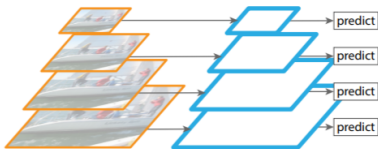
- (Cai et al., 2016): A unified multiscale deep convolutional neural network for fast object detection, аналогичная идея
- У object proposal network несколько веток с выходами разных масштабов:



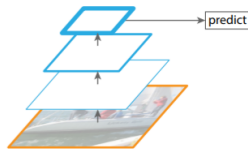
ПИРАМИДЫ ПРИЗНАКОВ

FEATURE PYRAMID NETWORKS

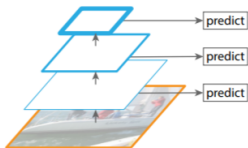
- (Lin et al., 2017): Feature Pyramid Networks for Object Detection
- Базовая идея – строим пирамиду признаков и используем их все:



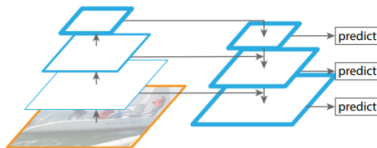
(a) Featurized image pyramid



(b) Single feature map



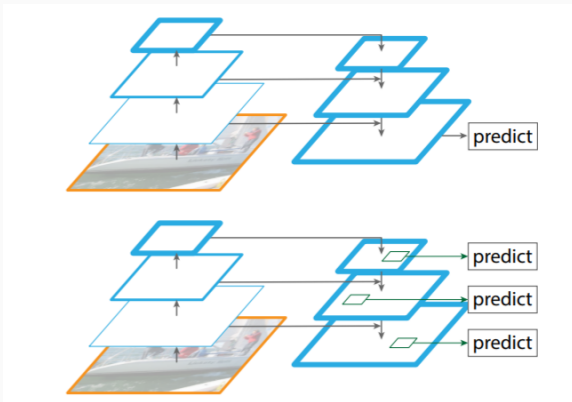
(c) Pyramidal feature hierarchy



(d) Feature Pyramid Network

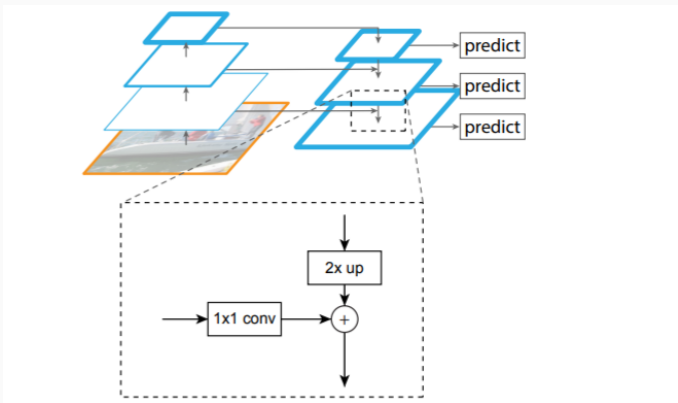
FEATURE PYRAMID NETWORKS

- Раньше так не делали, потому что очень долго и сложно
- FPN пытается сделать пирамиду признаков, у которой было бы много семантики на каждом уровне; сверху обычная архитектура, снизу FPN:



FEATURE PYRAMID NETWORKS

- Когда идём сверху вниз, добавляем признаки сверху, в них сила, то бишь семантика:



- Эту идею можно куда угодно применить, и в RPN, и в Faster R-CNN

FEATURE PYRAMID NETWORKS

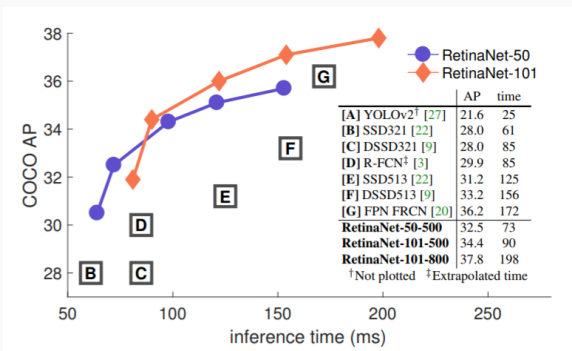
- И, опять же, очень круто получается:

method	backbone	competition	image pyramid	test-dev					test-std				
				AP@.5	AP	AP _s	AP _m	AP _l	AP@.5	AP	AP _s	AP _m	AP _l
ours, Faster R-CNN on FPN	ResNet-101	-		59.1	36.2	18.2	39.0	48.2	58.5	35.8	17.5	38.7	47.8
<i>Competition-winning single-model results follow:</i>													
G-RMI [†]	Inception-ResNet	2016		-	34.7	-	-	-	-	-	-	-	-
AttractionNet [‡] [10]	VGG16 + Wide ResNet [§]	2016	✓	53.4	35.7	15.6	38.0	52.7	52.9	35.3	14.7	37.6	51.9
Faster R-CNN +++ [16]	ResNet-101	2015	✓	55.7	34.9	15.6	38.7	50.9	-	-	-	-	-
Multipath [40] (on minival)	VGG-16	2015		49.6	31.5	-	-	-	-	-	-	-	-
ION [‡] [2]	VGG-16	2015		53.4	31.2	12.8	32.9	45.2	52.9	30.7	11.8	32.8	44.8

- И можно убедиться, что действительно все эти пирамиды нужны, без них хуже:

Fast R-CNN	proposals	feature	head	lateral?	top-down?	AP@0.5	AP	AP _s	AP _m	AP _l
(a) baseline on conv4	RPN, $\{P_k\}$	C_4	conv5			54.7	31.9	15.7	36.5	45.5
(b) baseline on conv5	RPN, $\{P_k\}$	C_5	2fc			52.9	28.8	11.9	32.4	43.4
(c) FPN	RPN, $\{P_k\}$	$\{P_k\}$	2fc	✓	✓	56.9	33.9	17.8	37.7	45.8
<i>Ablation experiments follow:</i>										
(d) bottom-up pyramid	RPN, $\{P_k\}$	$\{P_k\}$	2fc	✓		44.9	24.9	10.9	24.4	38.5
(e) top-down pyramid, w/o lateral	RPN, $\{P_k\}$	$\{P_k\}$	2fc		✓	54.0	31.3	13.3	35.2	45.3
(f) only finest level	RPN, $\{P_k\}$	P_2	2fc	✓	✓	56.3	33.4	17.3	37.3	45.6

- (Lin et al., 2018): Focal Loss for Dense Object Detection
- Изменили функцию ошибки, стали работать гораздо быстрее и лучше:



- Как это?

- Что мешает one-shot detectors? Почему они хуже, чем two-stage (типа Faster R-CNN)?
- Lin et al. утверждают, что мешает class imbalance: надо очень много выходов сразу предсказать, и в основном это очень простые примеры из чистого background
- В two-stage детекторах есть фильтрация, region proposal networks, т.е. у них imbalance совсем не такой сильный

- Чтобы с этим справиться, предлагается focal loss. Начинаем с перекрёстной энтропии:

$$\text{CE}(p, y) = \begin{cases} -\log(p), & \text{если } y = 1, \\ -\log(1 - p), & \text{в противном случае;} \end{cases}$$

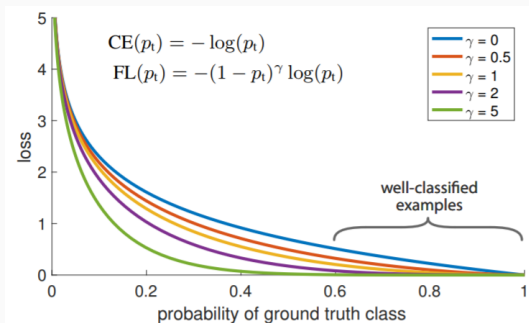
переобозначим $p_t = [y = 1]p + [y = 0](1 - p)$, и тогда $\text{CE}(p, y) = -\log(p_t)$.

- Проблема тут в том, что даже простые примеры с $p_t > 0.8$ дают нетривиальную потерю, и если всё просуммировать, то забивают «сигнал».

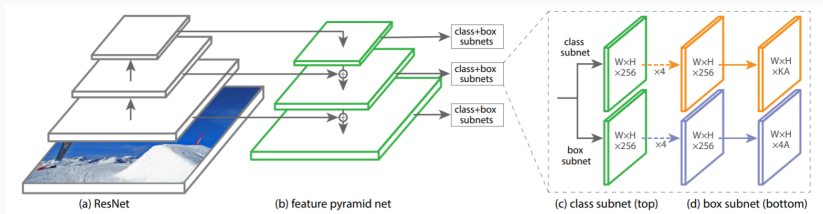
- Focal loss:

$$\text{FL}(p, y) = -(1 - p_t)^\gamma \log(p_t).$$

- Просто перевзвешиваем так, чтобы потерь было меньше на хорошо классифицируемых примерах.



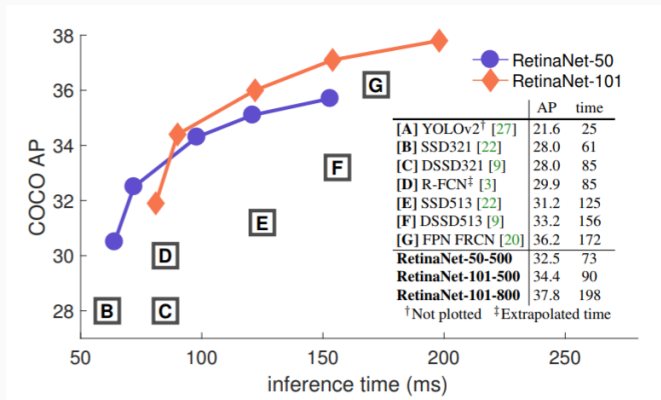
- RetinaNet: FPN c focal loss



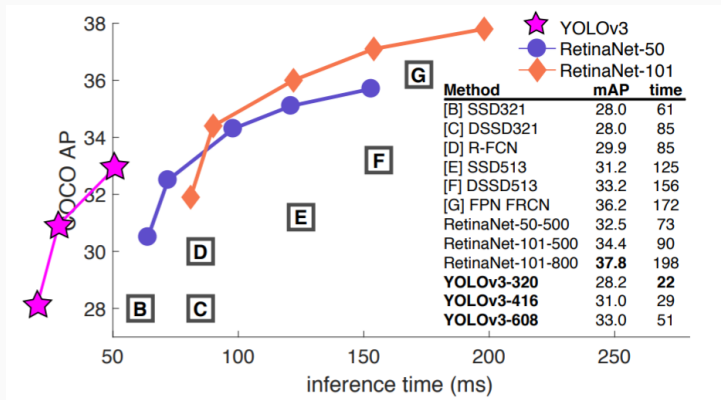
- Получается очень хорошо:

	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
<i>Two-stage methods</i>							
Faster R-CNN+++ [16]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [20]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [17]	Inception-ResNet-v2 [34]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [32]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
<i>One-stage methods</i>							
YOLOv2 [27]	DarkNet-19 [27]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [22, 9]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [9]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet (ours)	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet (ours)	ResNeXt-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2

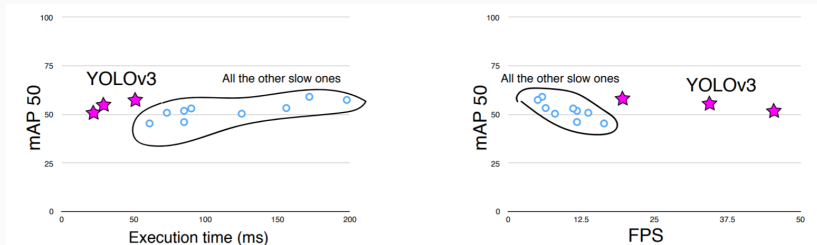
- Очень хорошо...



- ...но YOLO всё равно лучше

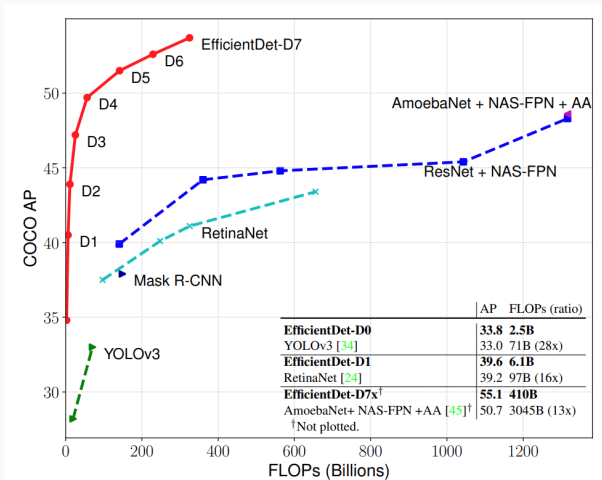


- YOLOv3 – это новый вариант YOLO, в котором использовали focal loss.
- Быстро:

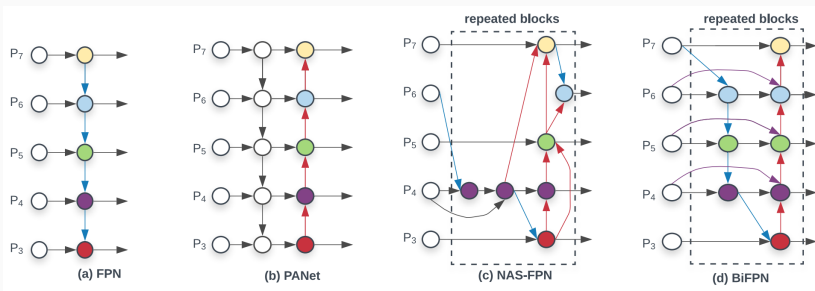


- И хорошо: https://www.youtube.com/watch?time_continue=2&v=MPU2HistivI

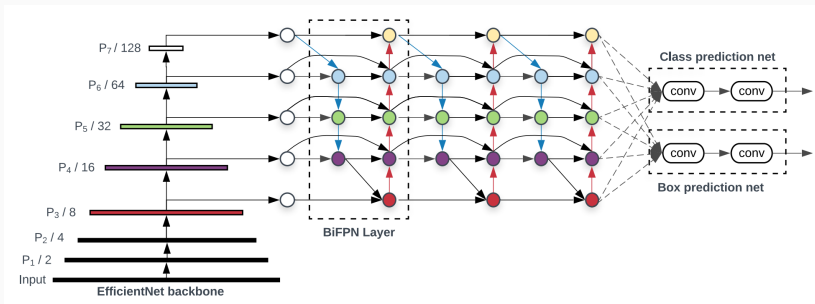
- (Tan et al., 2019): EfficientDet – neural architecture search добрался и до object detection, конечно



- Новые типы блоков для пирамид признаков:

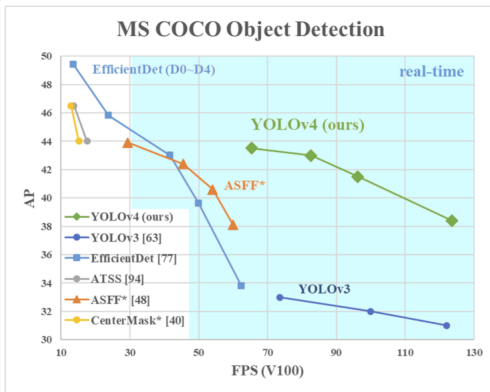


- А дальше просто подставим EfficientNet в качестве backbone:



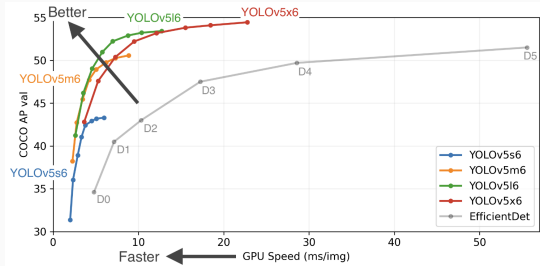
YOLOv4, YOLOv5, PP-YOLO...

- YOLOv4 (Bochkovskiy et al., 2020): again a new iteration with lots of small improvements
- Bag-of-freebies (augmentation incl. self-adversarial training, label smoothing, objective function etc.) vs. bag-of-specials (activation functions, attention mechanism, weighted skip connections etc.)

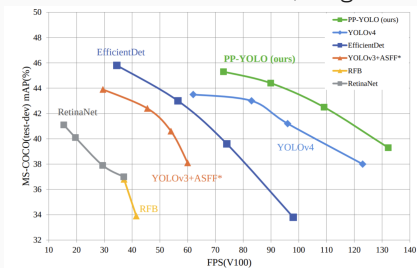


YOLOv4, YOLOv5, PP-YOLO...

- YOLOv5 only took a couple of months, but there is a story here...



- Things get even murkier with PP-YOLO (Long et al., 2020)



YOLO сегодня

А что с YOLO сейчас?

- Мы остановились на YOLOv4/v5; с тех пор прошло шесть лет, и «YOLO» из одной архитектуры превратилось в бренд.
- Джозеф Редмон, автор v1–v3, ушёл из computer vision в 2020 году по этическим соображениям и больше к YOLO отношения не имеет.
- Имя подхватили все кому не лень:
 - v4 и v7 — Бочковский, Ванг и Ляо;
 - v5, v8, YOLO11 — компания Ultralytics (статей нет, есть репозиторий);
 - v6 — Meituan, v9 — снова Ванг и Ляо, v10 — Университет Цинхуа.

А что с YOLO сейчас?

- Две идеи из этого потока действительно важны.
- *NMS-free детекция* (YOLOv10, 2024): две головы при обучении, one-to-many для сигнала и one-to-one для инференса — и non-maximum suppression больше не нужен, а значит, детектор становится честным end-to-end.
- *Внимание просочилось внутрь*: YOLOv12 (Tian et al., 2025) называет себя attention-centric — area attention A^2 режет карту признаков на полосы, чтобы внимание стоило линейно, плюс R-ELAN для стабильного обучения; 40.6% mAP при 1.64 мс.
- YOLOv13 (2025) пошёл дальше и собирает контекст гиперграфом (HyperACE).
- YOLOv12: <https://arxiv.org/abs/2502.12524>
- YOLOv13: <https://arxiv.org/abs/2506.17733>

- YOLO26 (Ultralytics, январь 2026) — то, что сейчас стоит брать по умолчанию:
 - end-to-end без NMS прямо из коробки;
 - выкинули distribution focal loss — регрессия рамки стала проще и дружелюбнее к железу;
 - обучение на MuSGD (SGD, скрещённый с Muon из мира LLM), ProgLoss и STAL — последнее специально про мелкие объекты.
- 40.9–57.5% mAP на COCO при 1.7–11.8 мс на T4; на CPU до 43% быстрее YOLO11.
- <https://docs.ultralytics.com/models/yolo26>

- Одна модель теперь закрывает семь задач: детекция, instance- и semantic-сегментация, классификация, позы, повёрнутые рамки (OBB) и даже монокулярная глубина.
- А 13 сентября 2026 года на YOLO Vision анонсировали YOLO27: четыре размера, те же семь задач, NMS-free; YOLO27l заявлен как первая модель Ultralytics выше 60 mAP на COCO (61.2). Скачать пока нельзя — обещают «до конца года».
- <https://docs.ultralytics.com/models/yolo27>

МОЖНО ЛИ ЕЩЁ БЕЗ ТРАНСФОРМЕРОВ?

- Короткий ответ: да, и это по-прежнему выбор по умолчанию, если вам нужен real-time на видеопотоке или деплой на edge.
- Свёрточные детекторы выигрывают по задержке, по размеру и по тому, насколько предсказуемо они квантуются в INT8 и экспортируются в ONNX/TensorRT.
- Даже открытый словарь теперь доступен без трансформерного детектора: YOLOE (Wang et al., ICCV 2025) добавляет к YOLO три режима — текстовый промпт (RepRTA), визуальный промпт-пример (SAVPE) и вообще без промпта (LRPC со встроенным словарём на 4585 имён).
- YOLOE: <https://arxiv.org/abs/2503.07465>

МОЖНО ЛИ ЕЩЁ БЕЗ ТРАНСФОРМЕРОВ?

- Но честно скажем и обратное: на верхней полке качества трансформеры.
- RF-DETR первым перешагнул 60 AP на COCO, оставаясь real-time; это DINOv2-бэкбон плюс деформируемое внимание в декодере.
- SAM 3 (Meta, ноябрь 2025) сегментирует по текстовому понятию сразу все экземпляры на картинке и в видео, а SAM 3.1 (март 2026) добавил трекинг нескольких объектов с общей памятью.
- Так что граница проходит не по «свёртки против внимания», а по тому, что вам важнее: миллисекунды и джоули или открытый словарь и качество масок.
- <https://ai.meta.com/blog/segment-anything-model-3/>

СПАСИБО!

Спасибо за внимание!

