



Онлайн-образование

Онлайн-образование

Проверить, идет ли запись!



Меня хорошо видно && слышно?

Ставьте + , если все хорошо
Напишите в чат, если есть проблемы

Преподаватель



Витвицкий Антон Александрович

- 8+ лет опыта в Computer Vision / Deep Learning
- 10+ лет опыта в коммерческой разработке связанной с RnD
- Опыт участия в американских и европейских стартапах
- Разработчик приложения Mirror-AI ( и )
- PhD in maths

Правила вебинара



Активно участвуем



Задаем вопрос в чат или голосом



Off-topic обсуждаем в Slack #канал группы или #general



Вопросы вижу в чате, могу ответить не сразу

Object Detection: R-CNN, Fast/Faster R-CNN, YOLO v1/v2/v3



Витвицкий Антон

Head of Computer Vision

Boost Technology Inc, CA

darkangel-nwo@ya.ru

Цели вебинара | После занятия вы

1

Поймете базовые концепции *Object Detection*

2

Узнаете о метриках оценки качества
Object Detection моделей и о датасетах

3

Изучите как устроены *R-CNN*, *Fast R-CNN* и
Faster R-CNN детекторы

4

Изучите как устроены *YOLO*,
YOLOv2 / YOLO9000 и *YOLOv3* детекторы

Object Detection: подходы, метрики, датасеты

Object Detection vs Object Recognition

?

Object Detection vs Object Recognition

Object Recognition:



Dog (0.97)

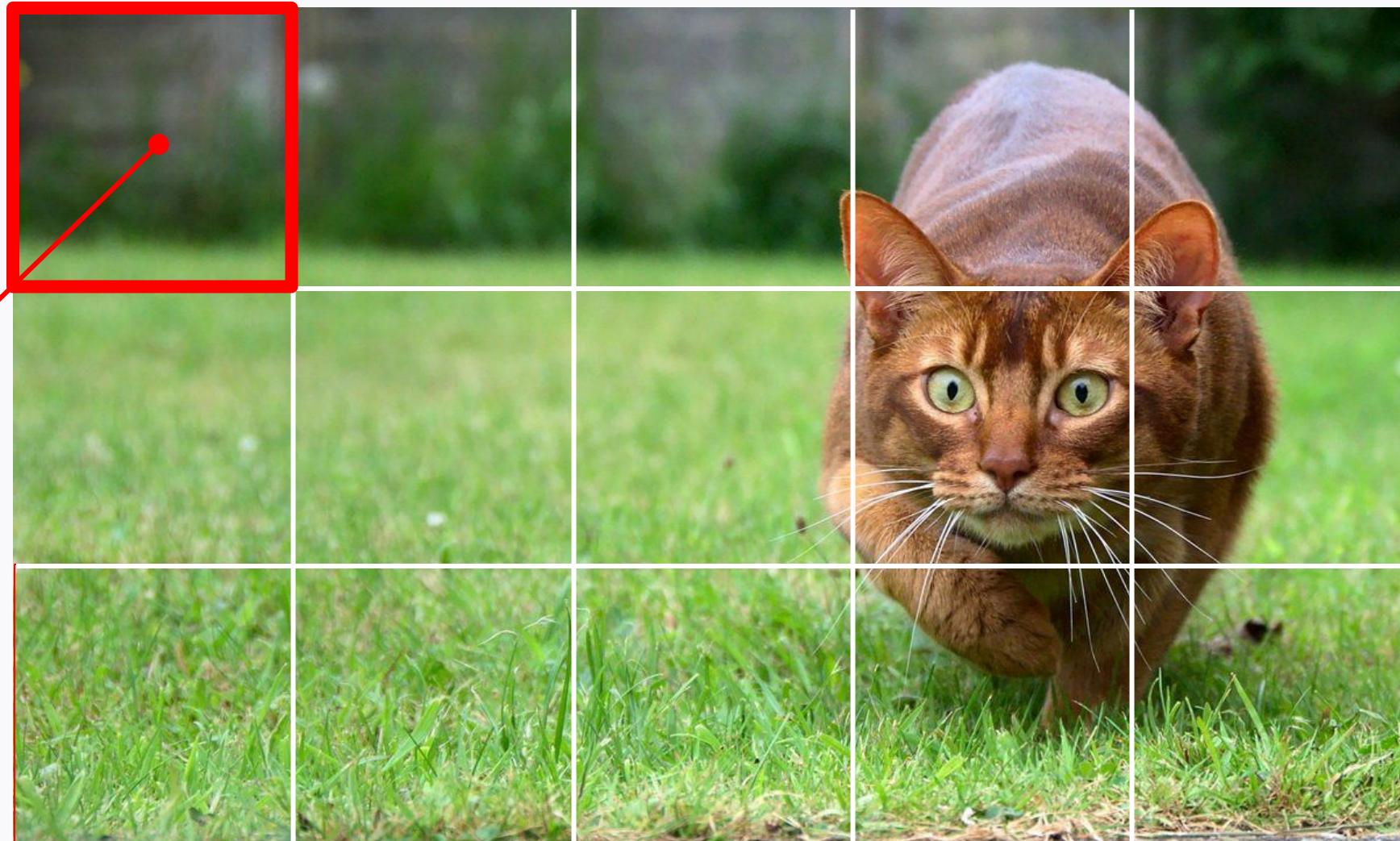
⊓

Object Detection:

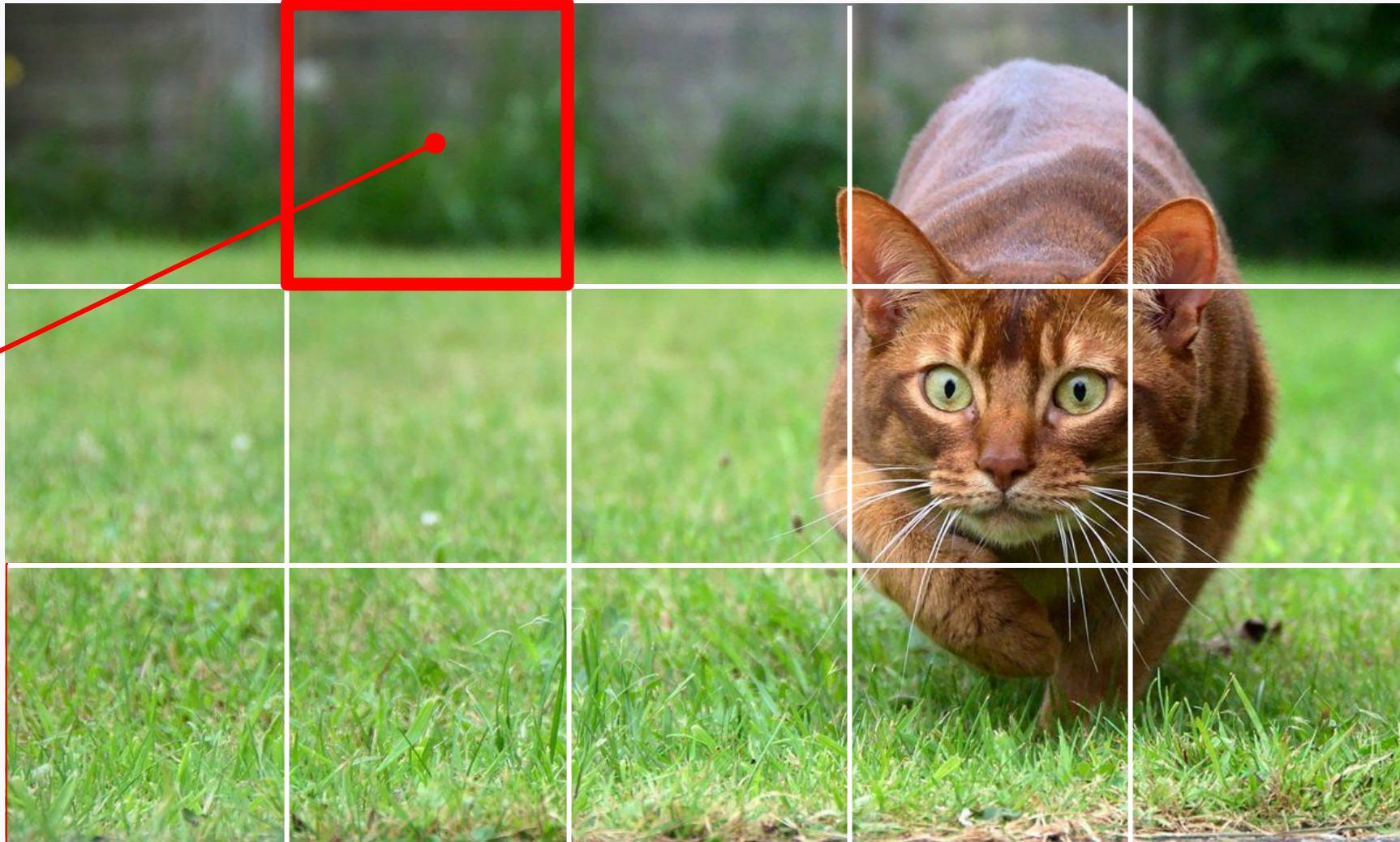


Dog (0.97)
x,y,w,h

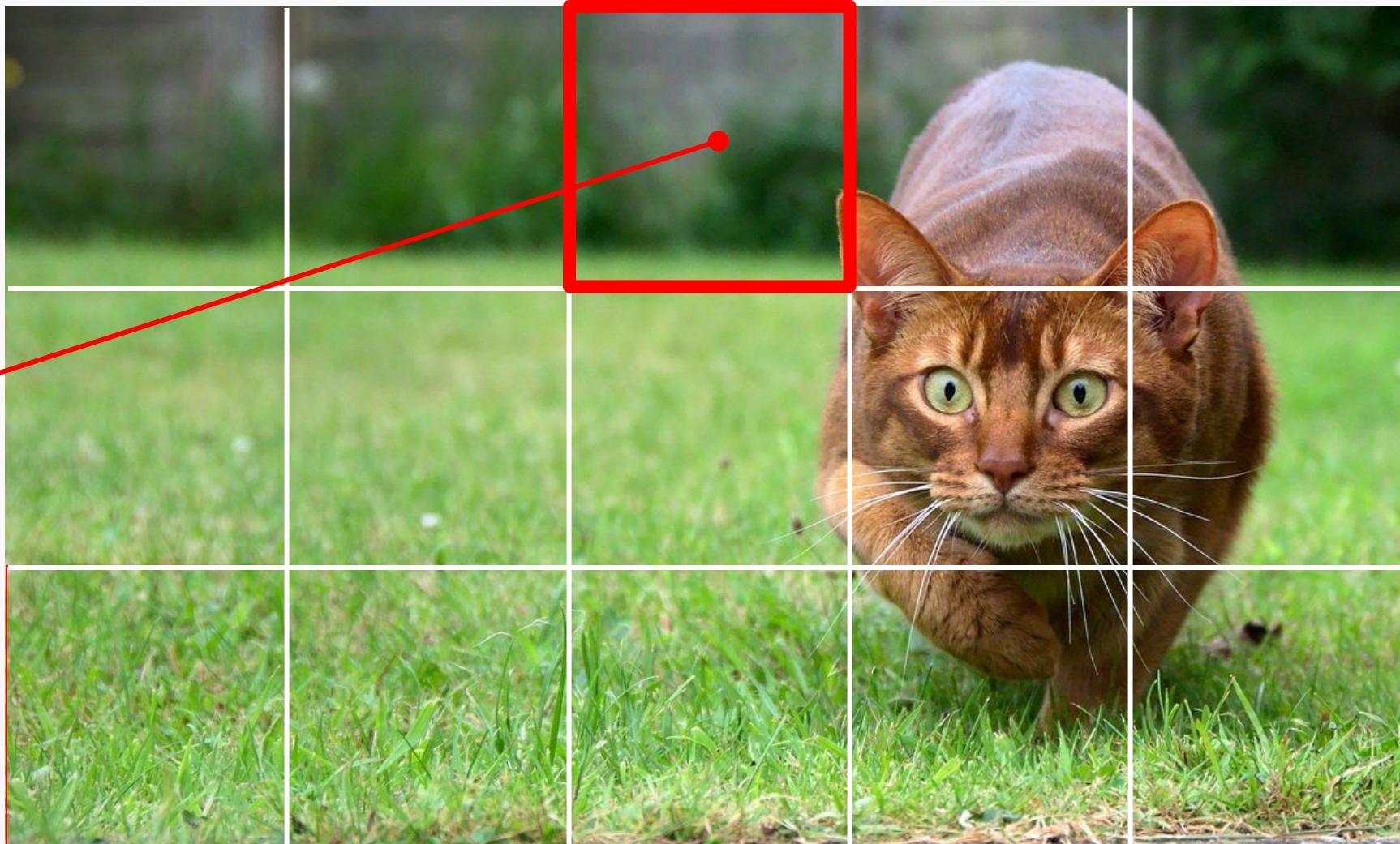
Метод Скользящего Окна (Sliding Window)



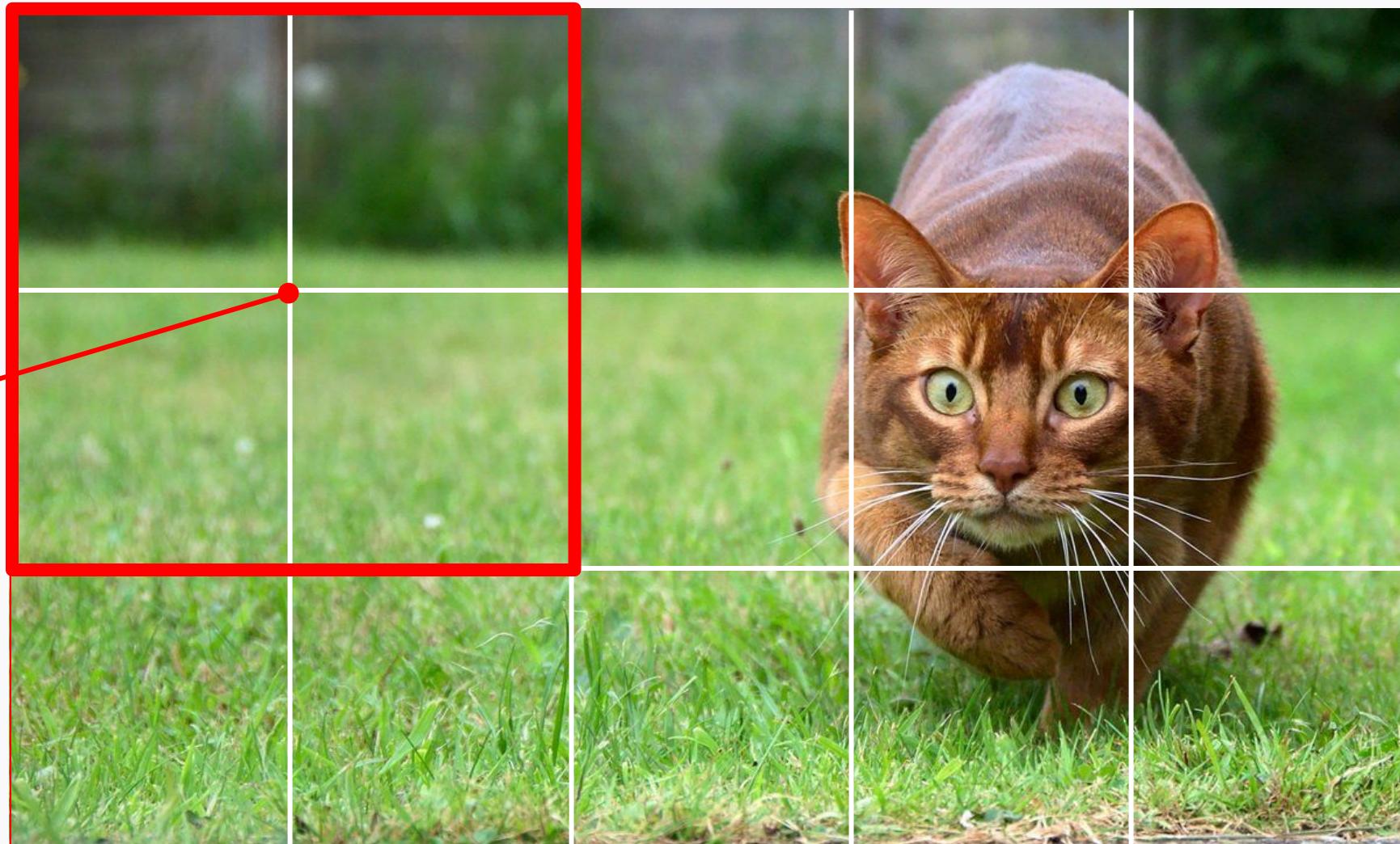
Метод Скользящего Окна (Sliding Window)



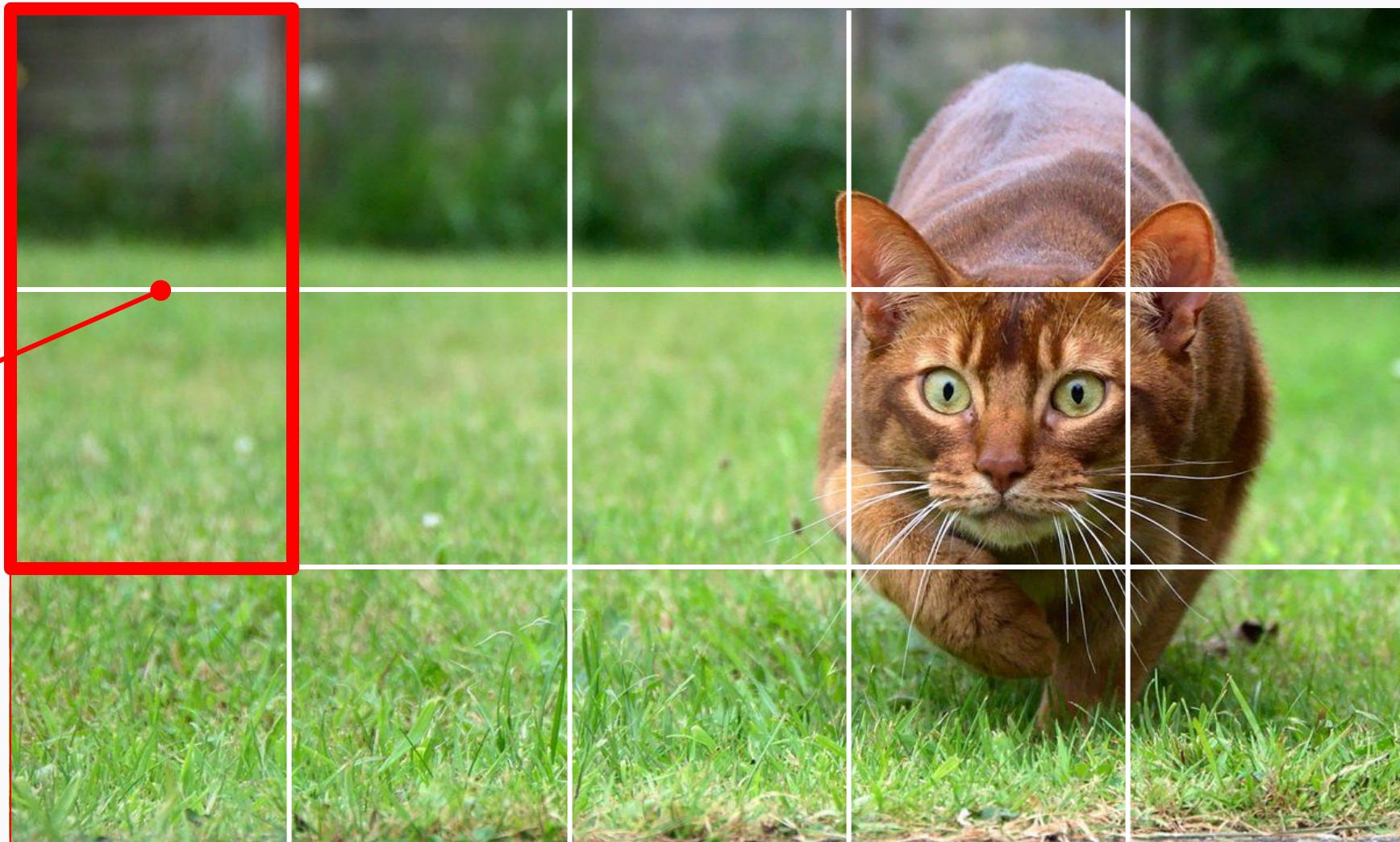
Метод Скользящего Окна (Sliding Window)



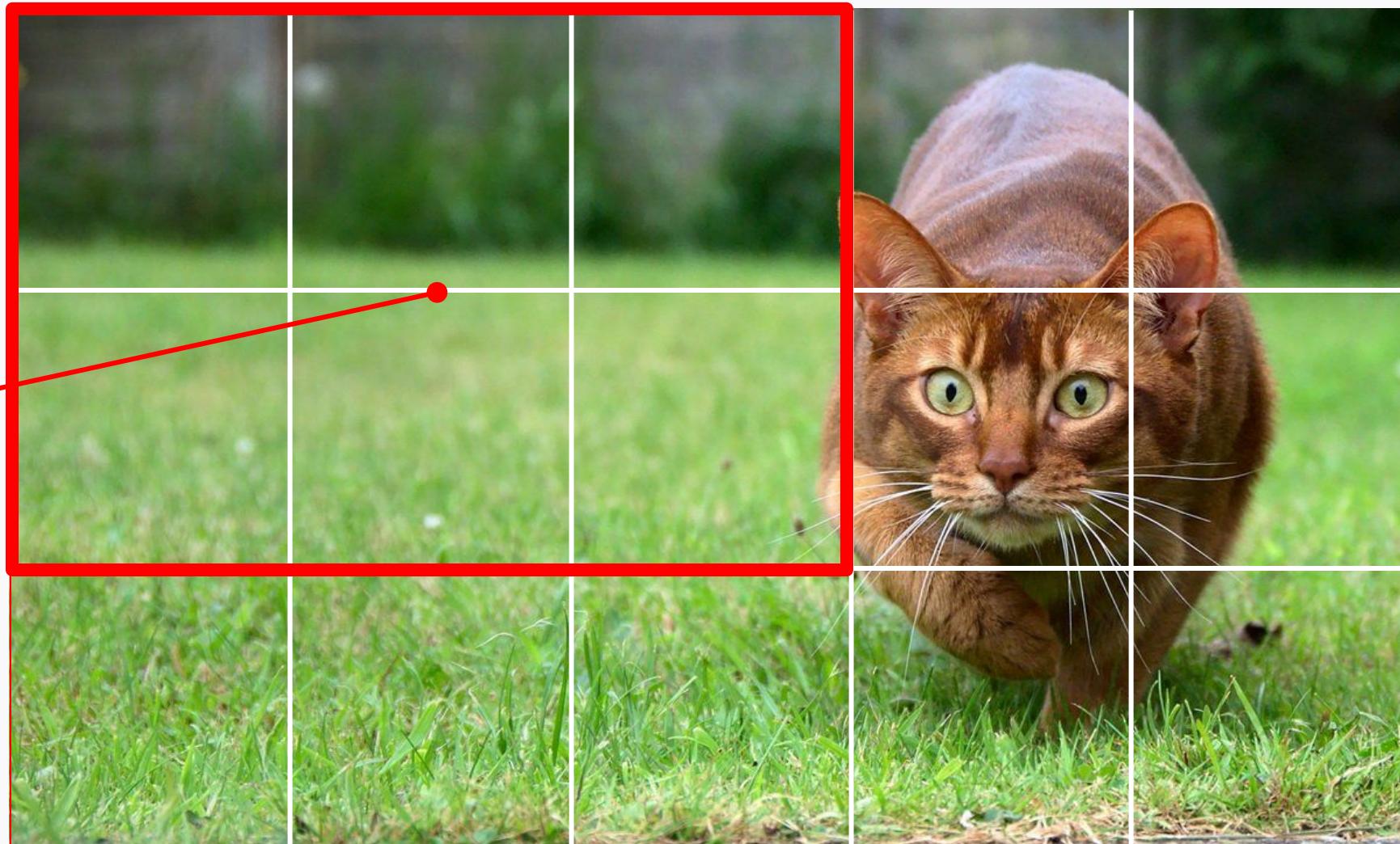
Метод Скользящего Окна (Sliding Window)



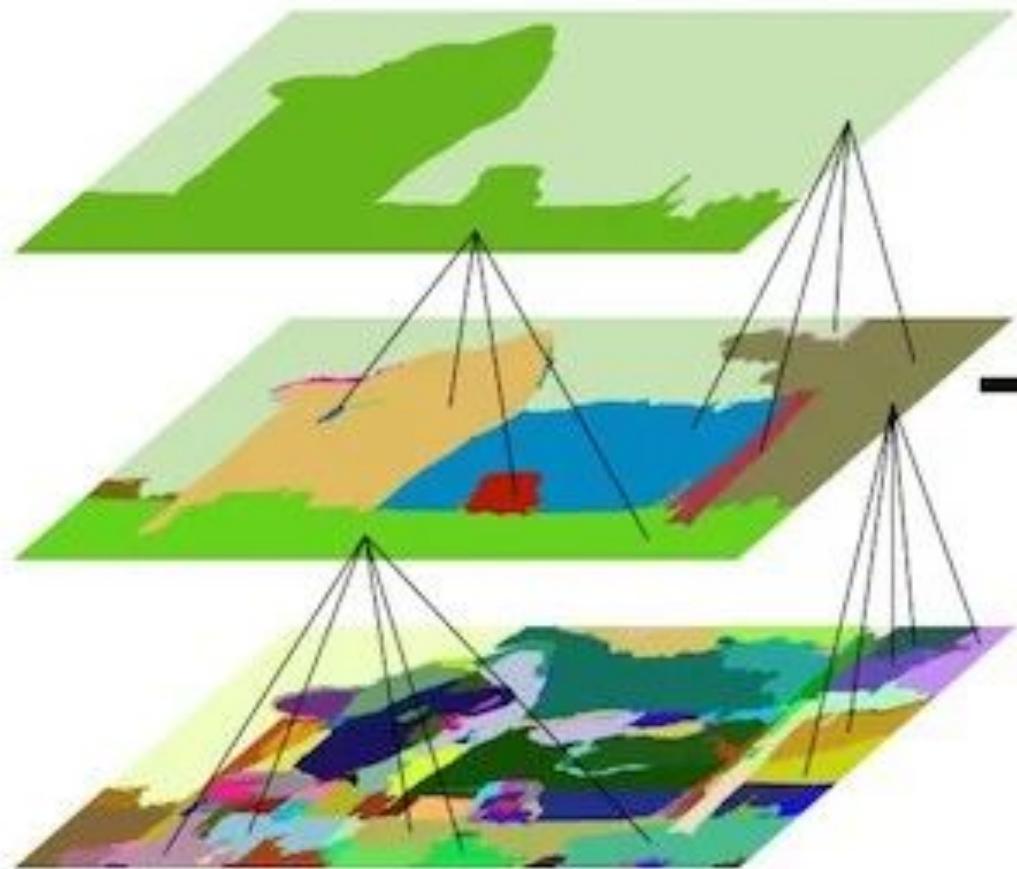
Метод Скользящего Окна (Sliding Window)



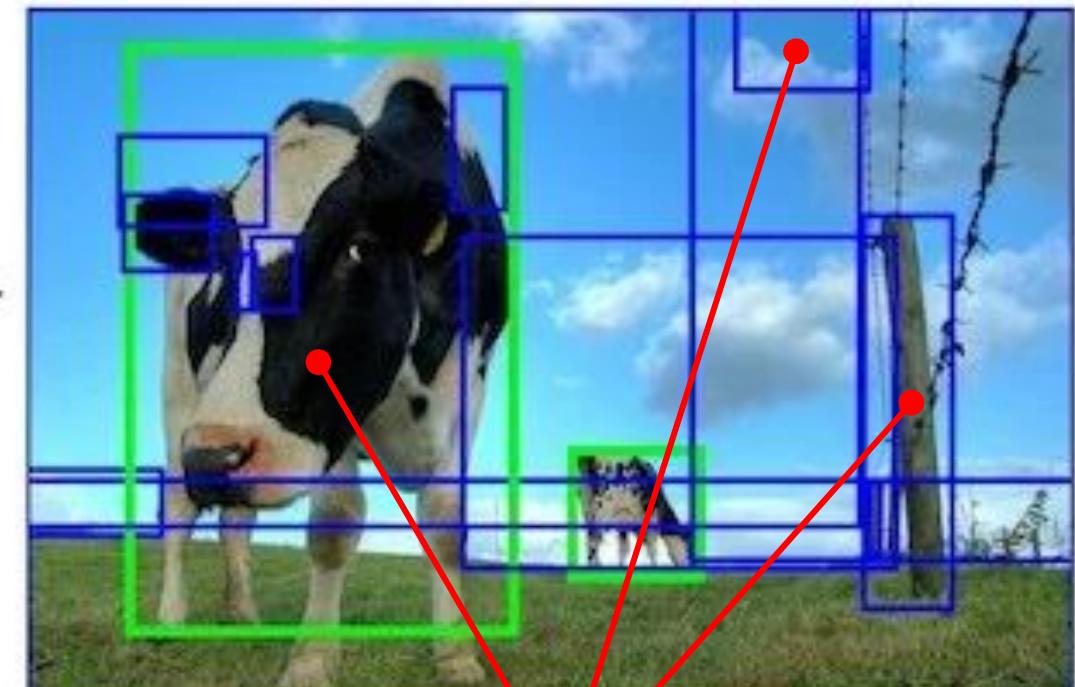
Метод Скользящего Окна (Sliding Window)



Подход Region Proposals



Region proposals (регионы-кандидаты):



Object
Recognition

→ Yes / No

Sliding Window vs Region Proposals

Sliding Window:

- Работают на уровне пикселей
- Используют окна разных масштабов и пропорций
- Генерируют десятки тысяч патчей для *Object Recognition*

Region Proposals:

- Группируют пиксели в сегменты/кластеры и работают с ними
- Генерируют на порядок меньше патчей для *Object Recognition*
- Обладают высокой оценкой полноты (*recall*)

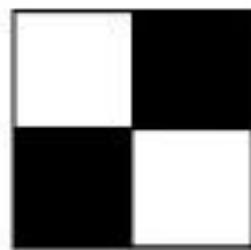
Признаки Хаара (Haar-like features)



(a) Edge Features



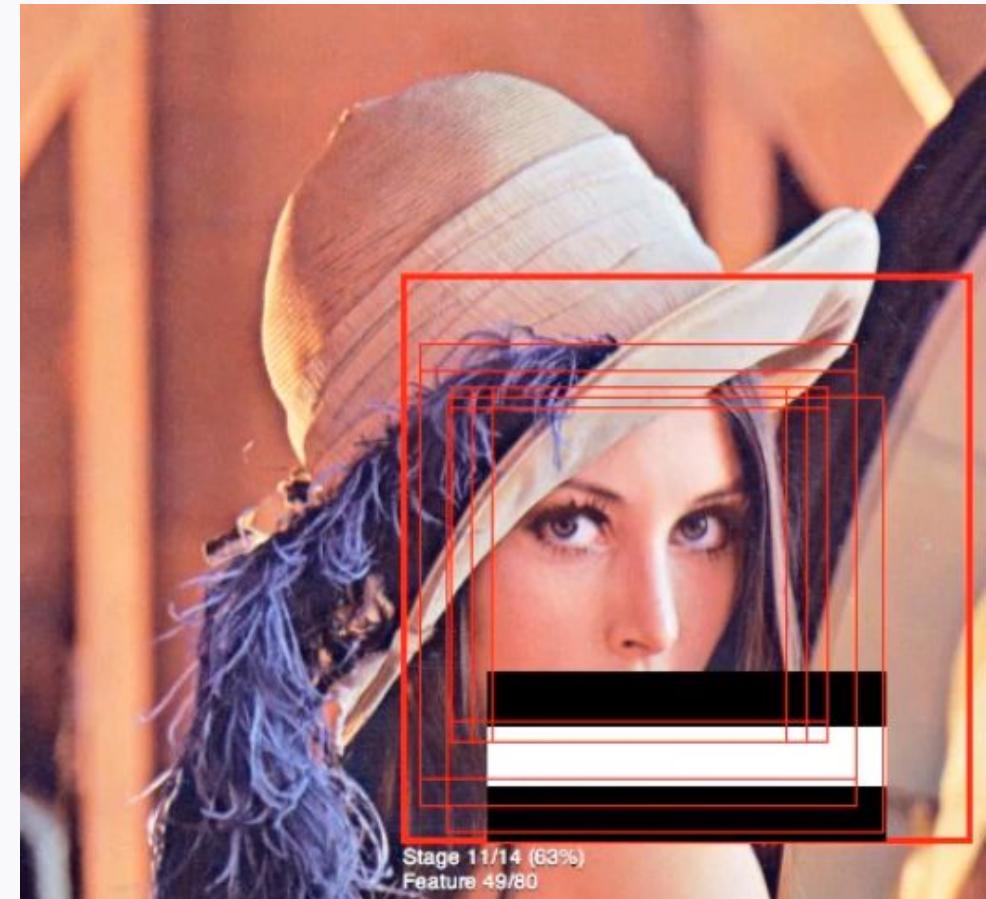
(b) Line Features



(c) Four-rectangle features

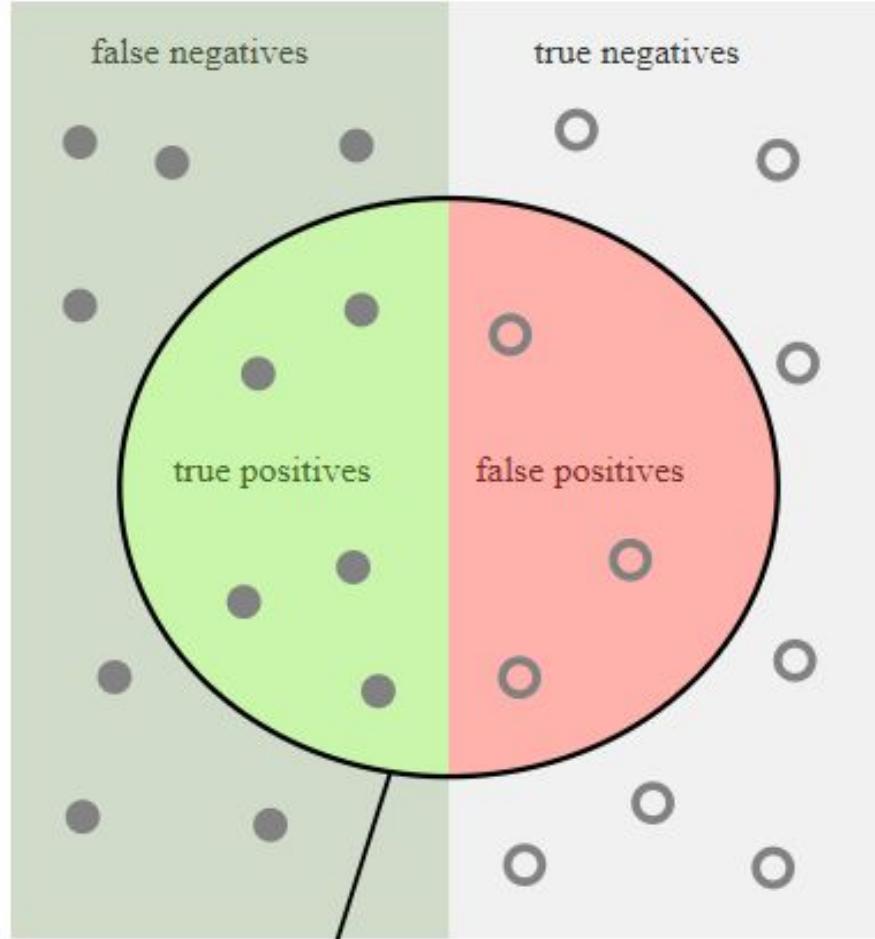
Признаки Хаара вычисляется как разность суммы интенсивностей пикселей под черной областью и суммой под белой областью.

Каскад классификаторов (метод Виолы-Джонса)



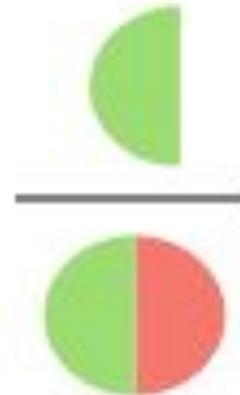
<https://vimeo.com/12774628>

Метрики: Precision and Recall



How many selected items are relevant?

Precision = $\frac{\text{true positives}}{\text{true positives} + \text{false positives}}$



How many relevant items are selected?

Recall = $\frac{\text{true positives}}{\text{true positives} + \text{false negatives}}$



Метрики: Intersection over Union (IoU)

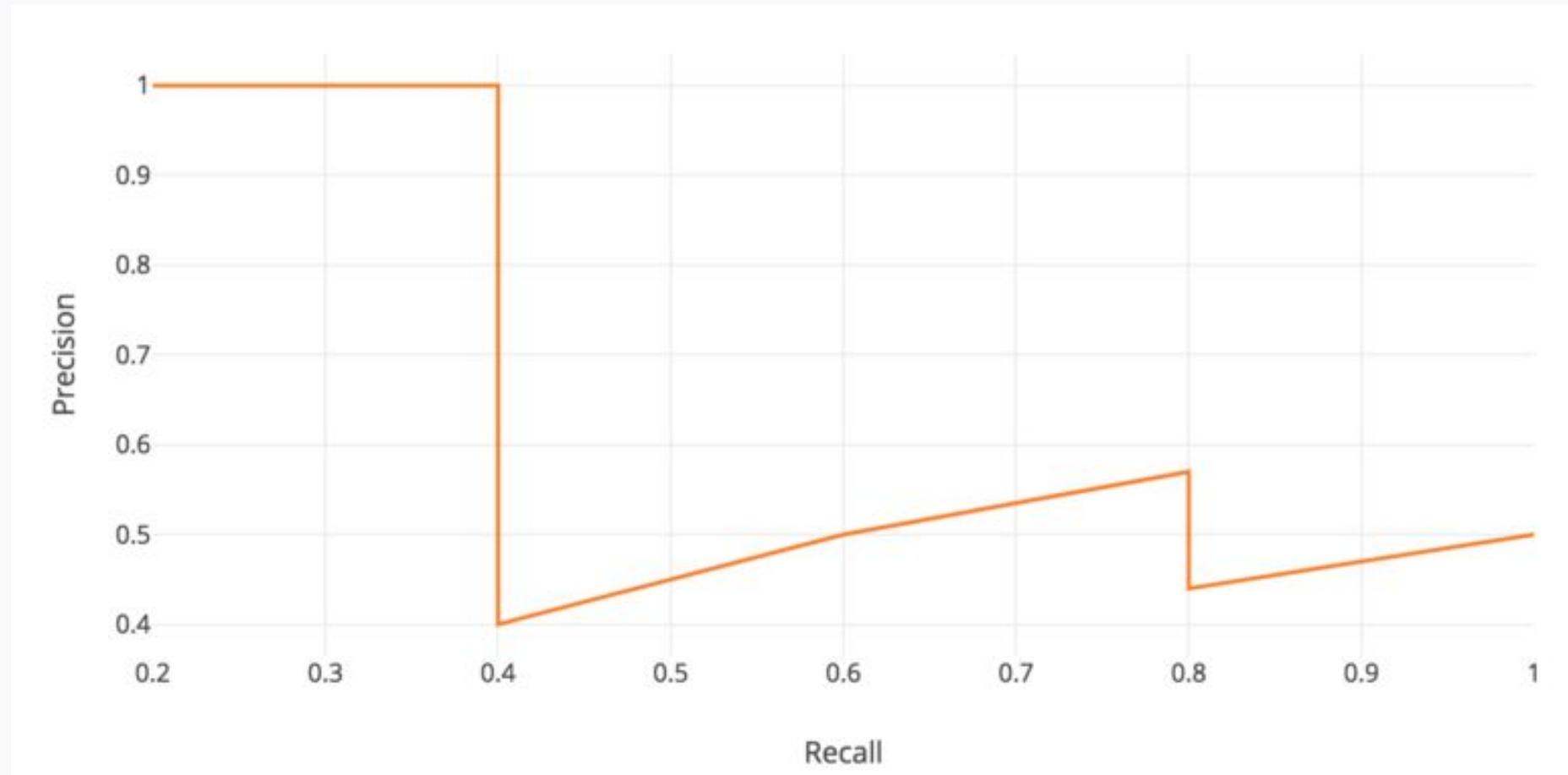
$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$



Метрики: Average Precision (AP)

AP - площадь под *Precision-Recall* кривой. Вычисляется как:

$$AP = \int_0^1 p(r)dr$$



Метрики: Average Precision (AP)

Полезно сгладить зигзагообразную форму кривой:

$$p_{\text{interp}}(r) = \max_{\tilde{r}: \tilde{r} \geq r} p(\tilde{r})$$



Метрики: Mean Average Precision (mAP)

$$\text{mAP} = \frac{1}{\#\text{classes}} \sum_{\text{class} \in \text{classes}} AP[\text{class}]$$

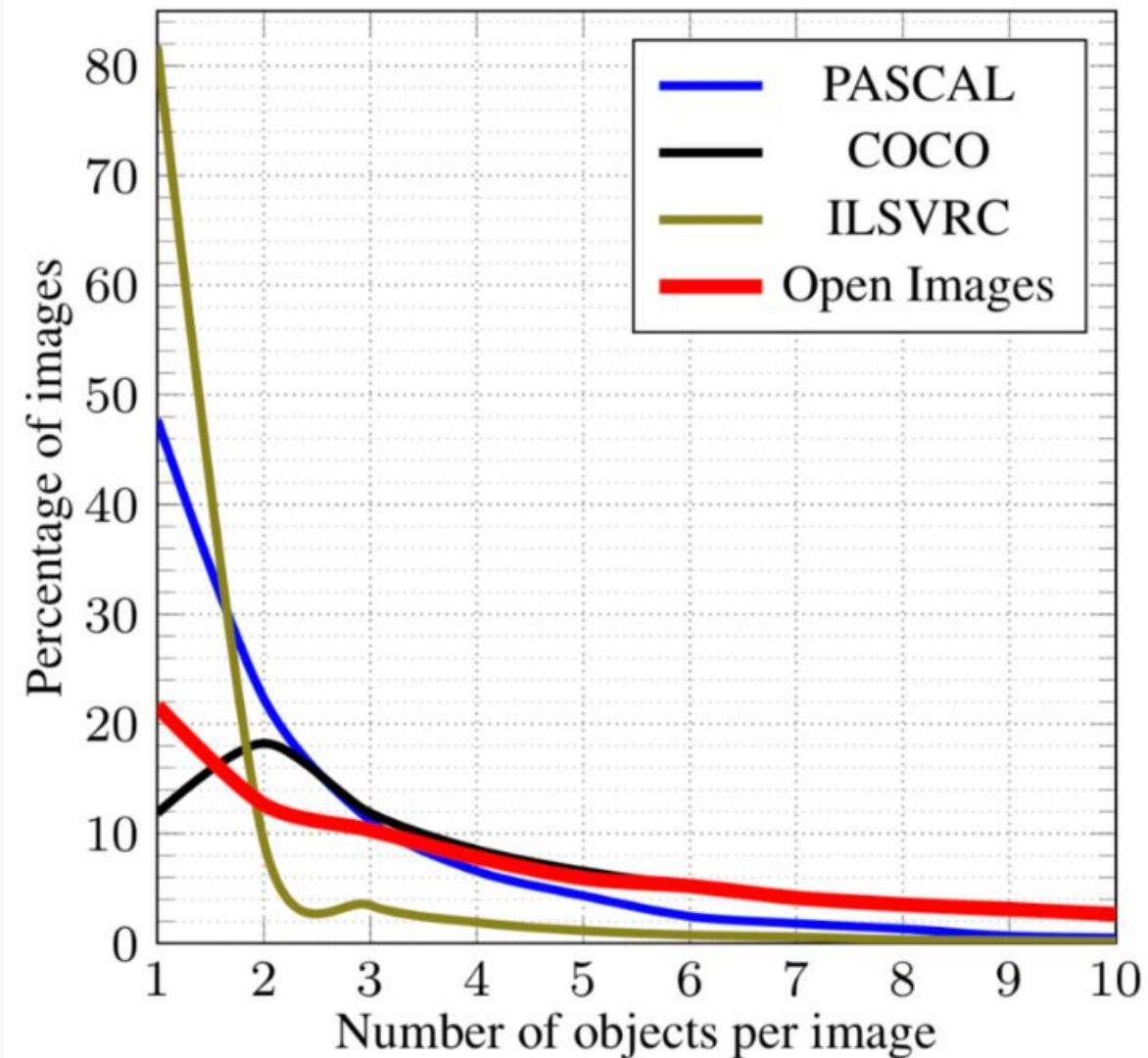
Метрики: Mean Average Precision (mAP)

- **PASCAL VOC 2008** использует AP вычисленный на 11 точках.
- **PASCAL VOC 2010-2012** использует AUC (т.е. все точки P-R кривой).
- **MS COCO** использует AP вычисленный на 101 точке и 10 порогах IoU .
- **ImageNet** используется AUC.

Датасеты для Object Detection.

	Число изображений	Число классов	Разметка	Размер
MS COCO	330k	80	1.5м объектов размечено боксами (200k изображений)	25 ГБ
Open Images Dataset	9м	600	16м объектов размечено боксами (1.7м изображений)	500 ГБ
ImageNet	14м	20 000	На 1.5м изображений объекты размечены боксами (всего 3к классов объектов)	150 ГБ

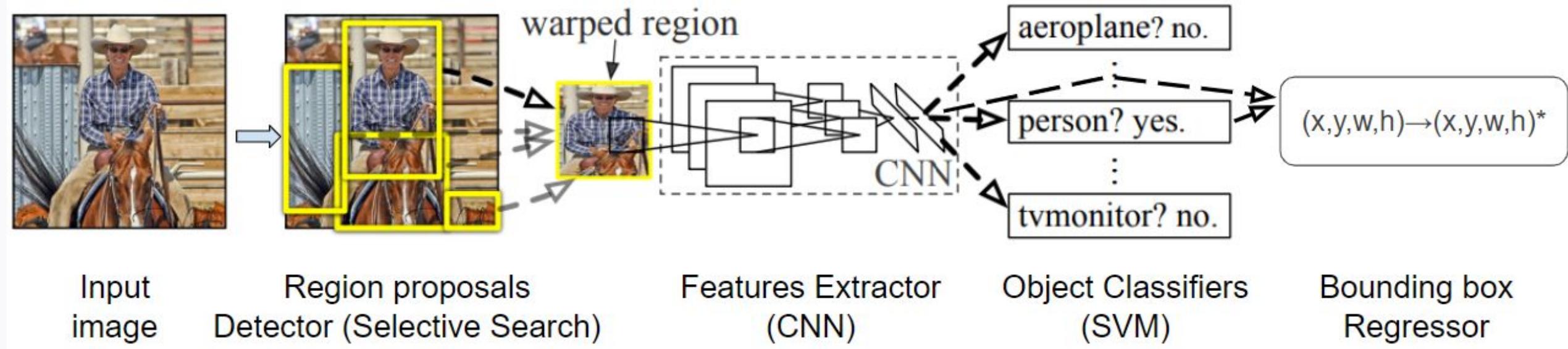
Датасеты для Object Detection.



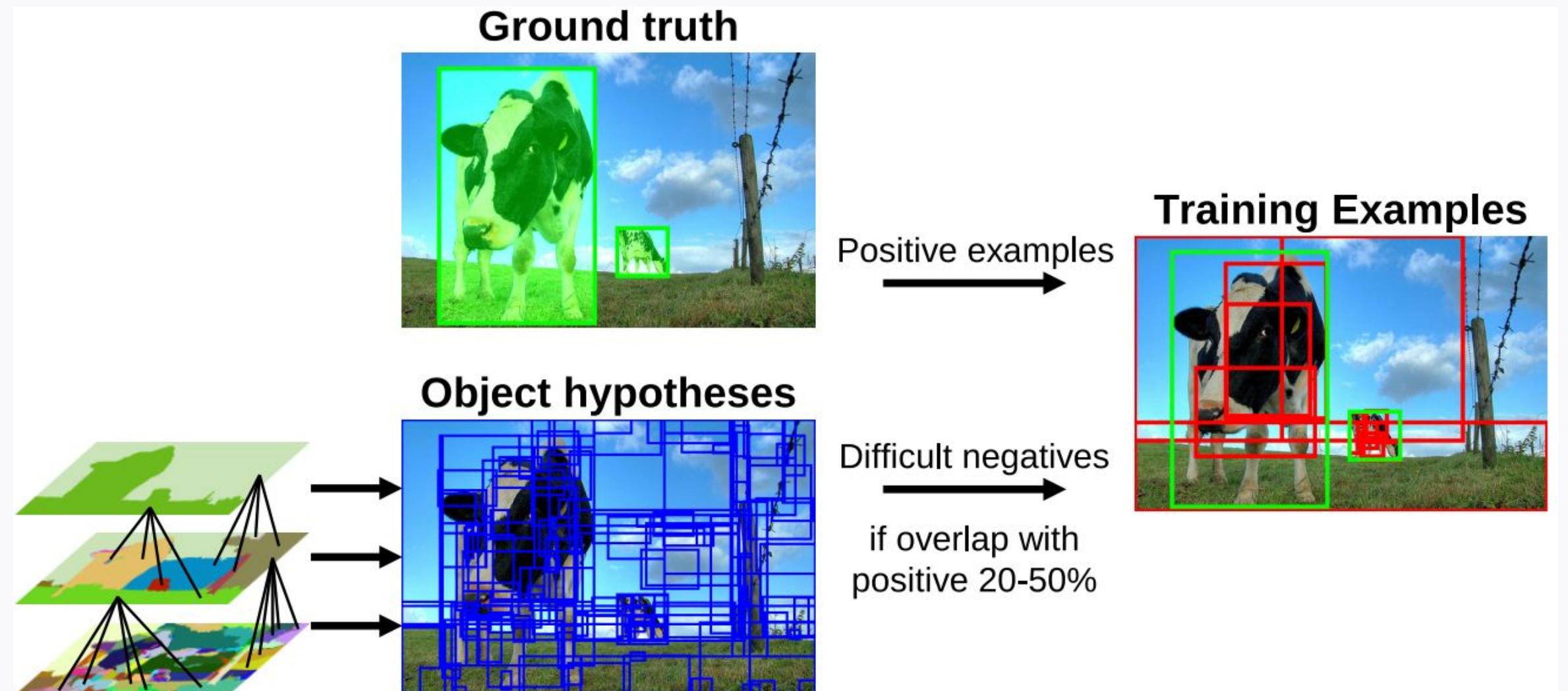
R-CNN

(Regions with CNN features)

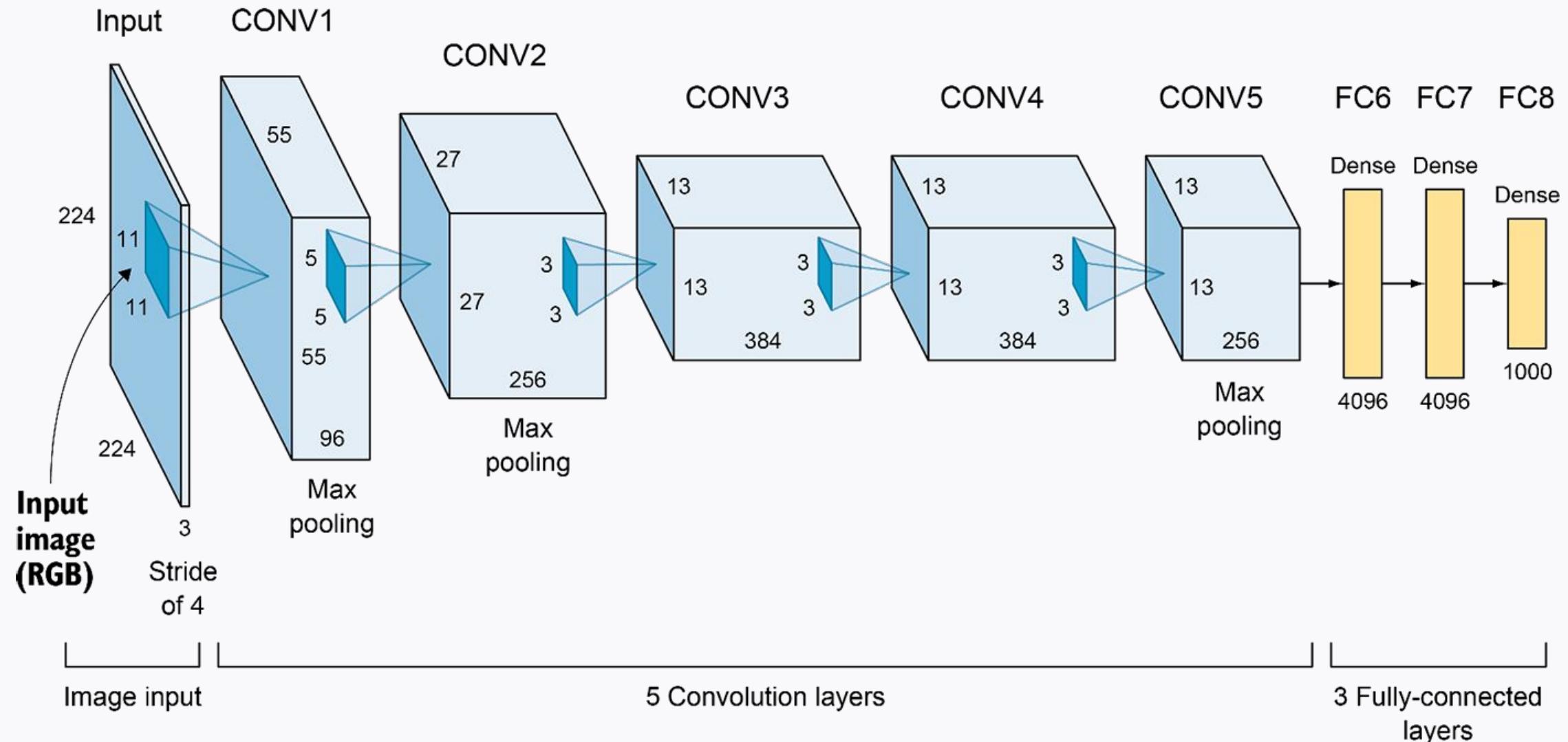
Object detection with R-CNN



Region proposals Detector (Selective Search)



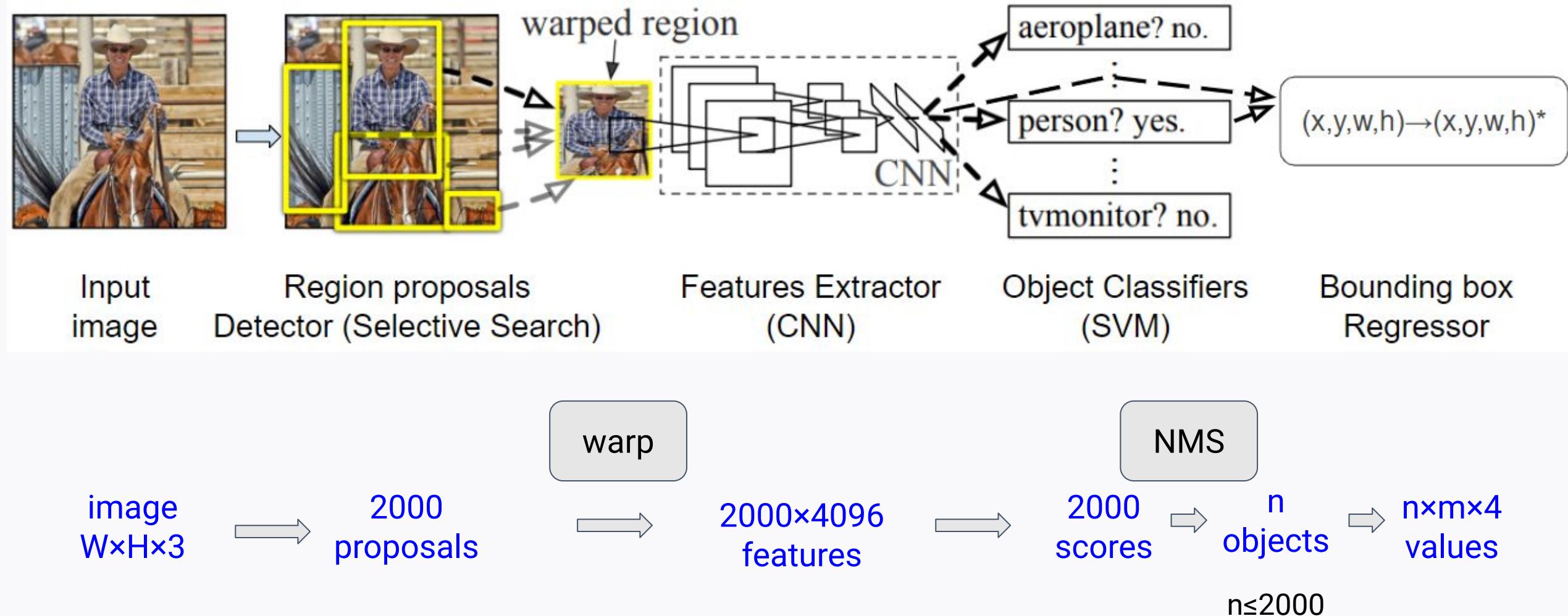
Feature Extractor (AlexNet)



Different object proposal transformations



Test-time detection



Training

Training steps:

- Supervised pre-training
- Domain-specific fine-tuning
- Object category classifiers

Bounding box regression

Regularized least squares objective (ridge regression):

$$\mathbf{w}_\star = \operatorname{argmin}_{\hat{\mathbf{w}}_\star} \sum_i^N (t_\star^i - \hat{\mathbf{w}}_\star^\top \phi_5(P^i))^2 + \lambda \|\hat{\mathbf{w}}_\star\|^2.$$

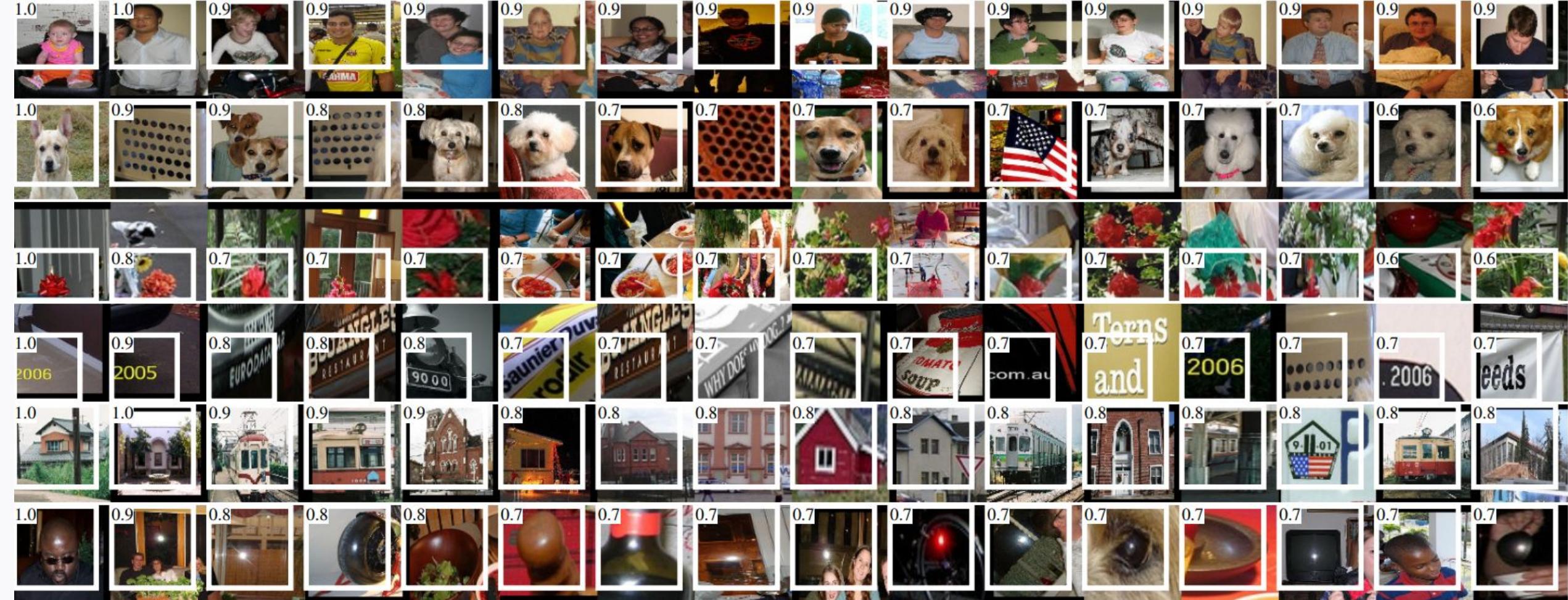
где

$$t_x = (G_x - P_x)/P_w$$
$$t_y = (G_y - P_y)/P_h$$
$$t_w = \log(G_w/P_w)$$
$$t_h = \log(G_h/P_h).$$

Proposal бокса -> gt бокс:

$$\begin{aligned}\hat{G}_x &= P_w d_x(P) + P_x \\ \hat{G}_y &= P_h d_y(P) + P_y \\ \hat{G}_w &= P_w \exp(d_w(P)) \\ \hat{G}_h &= P_h \exp(d_h(P)).\end{aligned}$$

Visualizing learned features



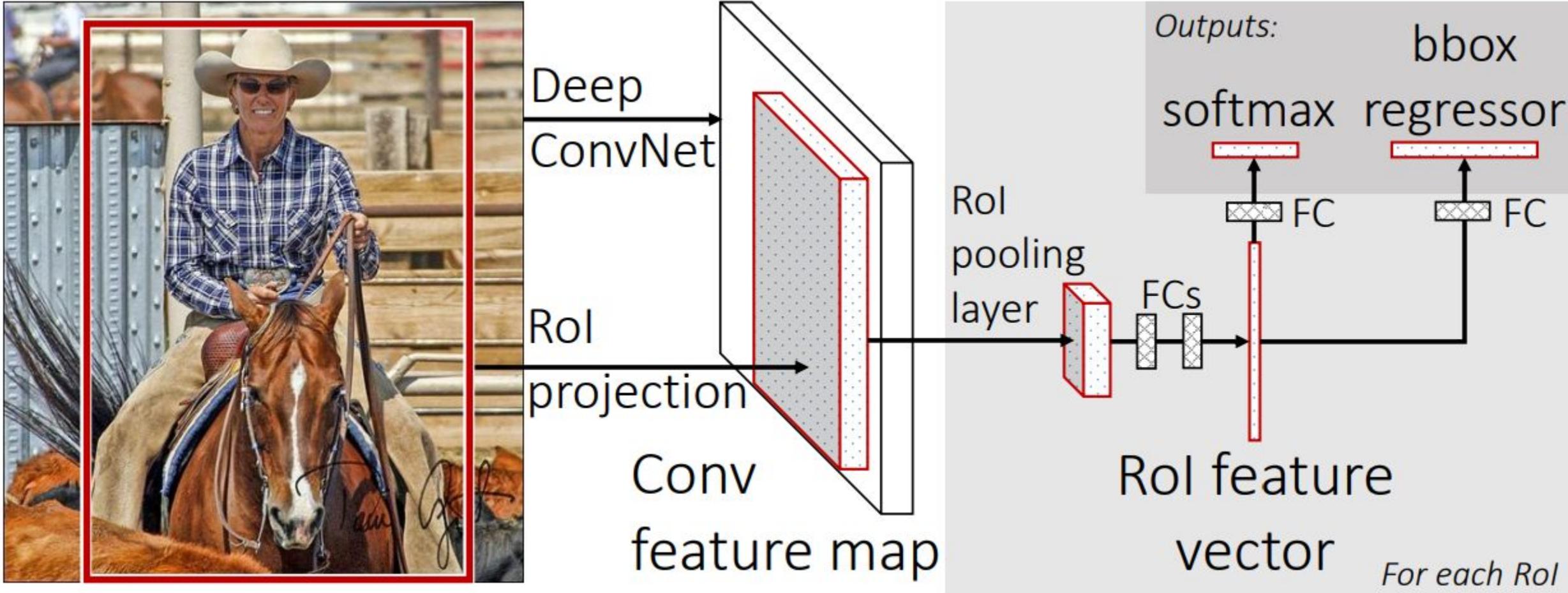
R-CNN: VOC 2010 results

VOC 2010 test	aero	bike	bird	boat	bottle	bus	car	cat	chair	cow	table	dog	horse	mbike	person	plant	sheep	sofa	train	tv	mAP
DPM v5 [20] [†]	49.2	53.8	13.1	15.3	35.5	53.4	49.7	27.0	17.2	28.8	14.7	17.8	46.4	51.2	47.7	10.8	34.2	20.7	43.8	38.3	33.4
UVA [39]	56.2	42.4	15.3	12.6	21.8	49.3	36.8	46.1	12.9	32.1	30.0	36.5	43.5	52.9	32.9	15.3	41.1	31.8	47.0	44.8	35.1
Regionlets [41]	65.0	48.9	25.9	24.6	24.5	56.1	54.5	51.2	17.0	28.9	30.2	35.8	40.2	55.7	43.5	14.3	43.9	32.6	54.0	45.9	39.7
SegDPM [18] [†]	61.4	53.4	25.6	25.2	35.5	51.7	50.6	50.8	19.3	33.8	26.8	40.4	48.3	54.4	47.1	14.8	38.7	35.0	52.8	43.1	40.4
R-CNN	67.1	64.1	46.7	32.0	30.5	56.4	57.2	65.9	27.0	47.3	40.9	66.6	57.8	65.9	53.6	26.7	56.5	38.1	52.8	50.2	50.2
R-CNN BB	71.8	65.8	53.0	36.8	35.9	59.7	60.0	69.9	27.9	50.6	41.4	70.0	62.0	69.0	58.1	29.5	59.4	39.3	61.2	52.4	53.7

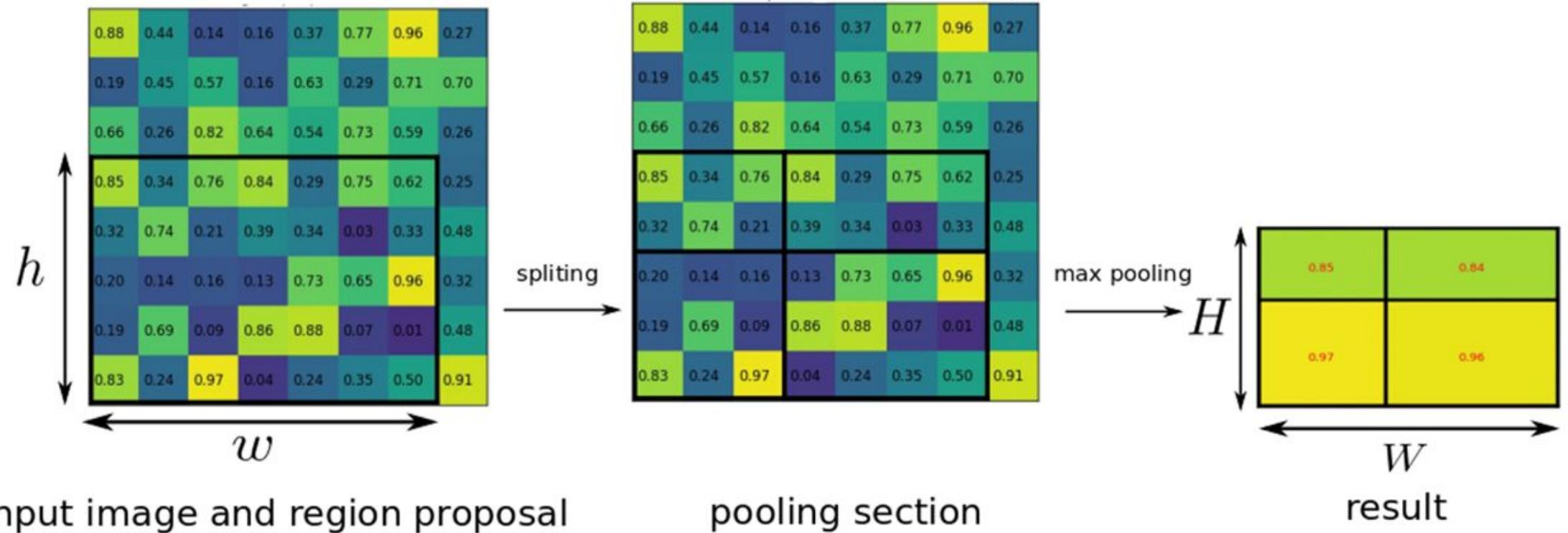
Fast R-CNN

(Fast Region-based CNN)

Fast R-CNN architecture



RoI Pooling Layer



Пример: $\text{RoI} = 5 \times 7$ и $\text{output} = 2 \times 2$, тогда $\text{pooling area} = 2 \times 3$ (3×3 после округления)

Training

- VGG16 обучается на ImageNet, а затем дотюнивается на PASCAL VOC / MS COCO.
- Обучение end-to-end, классификатор и регрессор учатся совместно.
- Батч состоит из N изображений и R/N RoI ($N=2, R=128$).
- 25% RoI положительные ($IoU \geq 0.5$), остальные - негативные ($IoU \in [0.1;0.5]$).
- RoI с $IoU < 0.1$ отбрасываются (*hard negative mining*).

Multi-task loss

u - gt класс объекта,

$v = (v_x, v_y, v_w, v_h)$ - gt бокс,

p - предсказанный класс объекта,

t^u - предсказанный бокс.

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

где $L_{\text{cls}}(p, u) = -\log p_u$ - *loss log* для классификатора объектов,

$L_{\text{loc}}(t^u, v)$ - *smooth L1 loss* для регрессии боксов.

Multi-task loss: Bounding box regression loss

$$L_{\text{loc}}(t^u, v) = \sum_{i \in \{\text{x}, \text{y}, \text{w}, \text{h}\}} \text{smooth}_{L_1}(t_i^u - v_i)$$

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Fast R-CNN detection

Вывод модели:

- 1) На вход подается изображение и 2000 proposals (из Selective Search).
- 2) Получаем набор RoI, которые содержат объект с некоторой вероятностью.
- 3) Для RoI одного класса применяем Non-Maximum suppression (NMS).

Fast R-CNN: VOC 2007 results

S=AlexNet

M=VGG-like версия S

L=VGG16

	S				M				L			
multi-task training?		✓		✓		✓		✓		✓		✓
stage-wise training?			✓			✓		✓		✓		✓
test-time bbox reg?			✓		✓		✓	✓		✓		✓
VOC07 mAP	52.2	53.3	54.6	57.1	54.7	55.5	56.6	59.2	62.6	63.4	64.0	66.9

SVM vs Softmax

	method	classifier	S	M	L
SVM	R-CNN [9, 10]	SVM	58.5	60.2	66.0
	FRCN [ours]	SVM	56.3	58.7	66.8
softmax	FRCN [ours]	softmax	57.1	59.2	66.9

Fast R-CNN: VOC 2010/2012 results

Pascal VOC 2010:

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

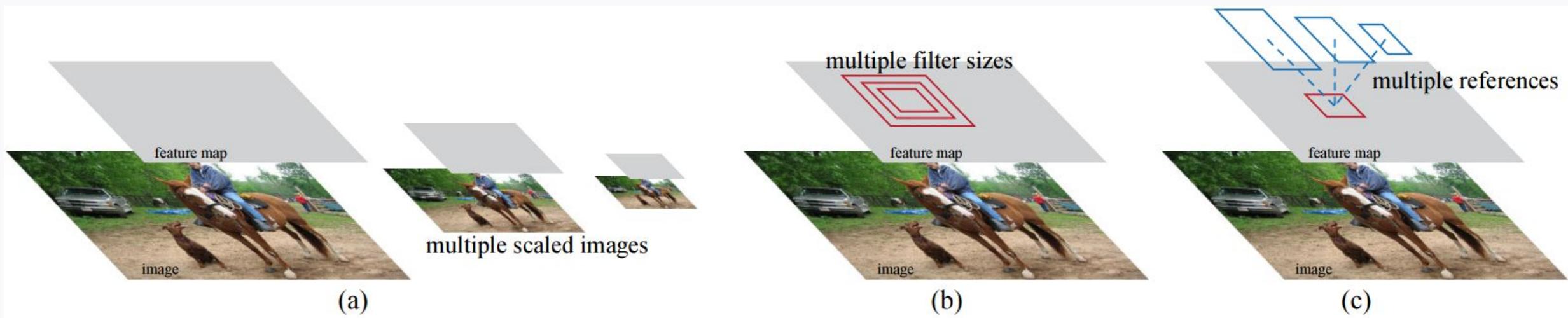
Pascal VOC 2012:

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

Faster R-CNN

(Region Proposal Networks + Fast R-CNN)

Scale invariant

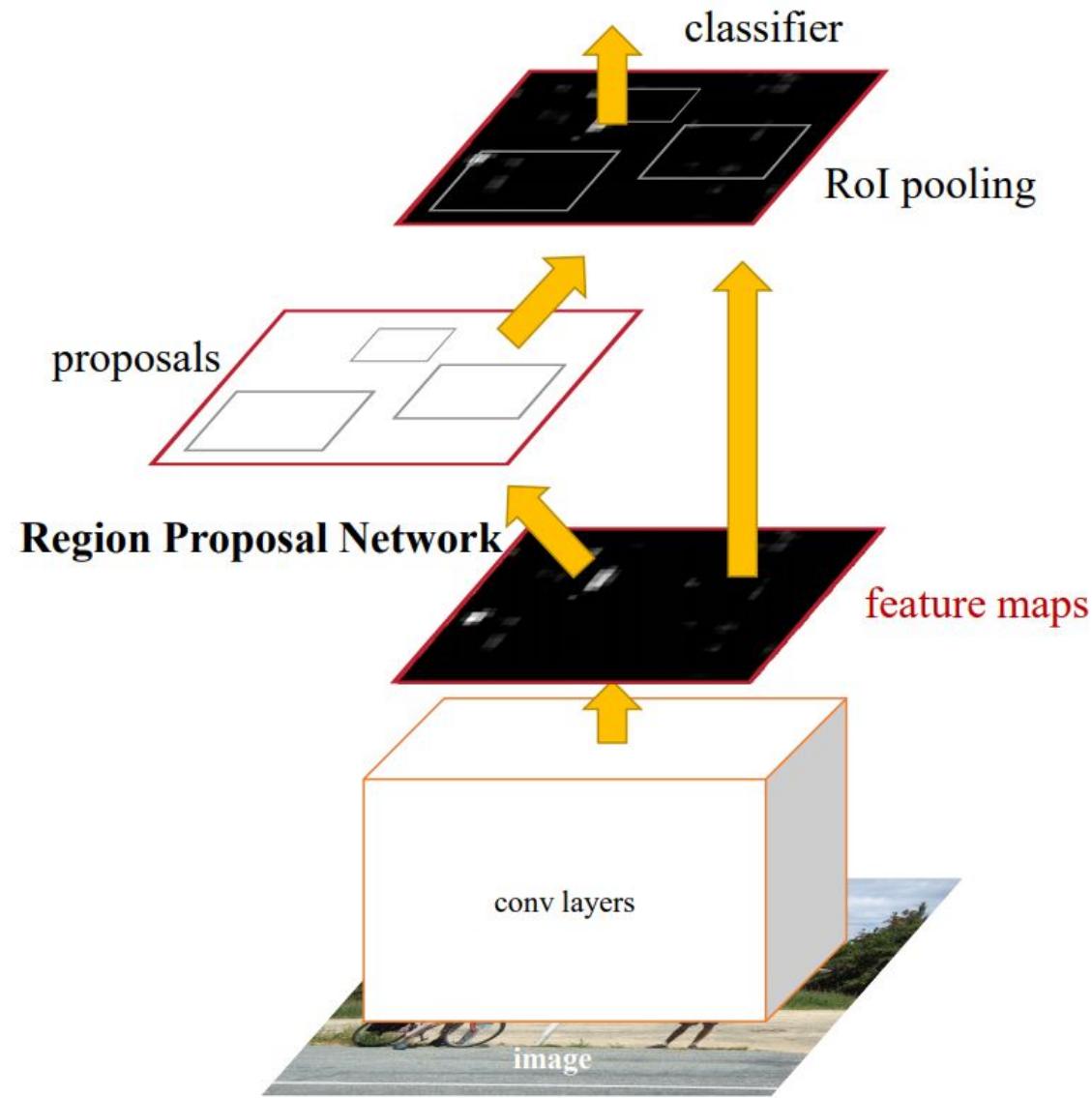


Images
pyramid

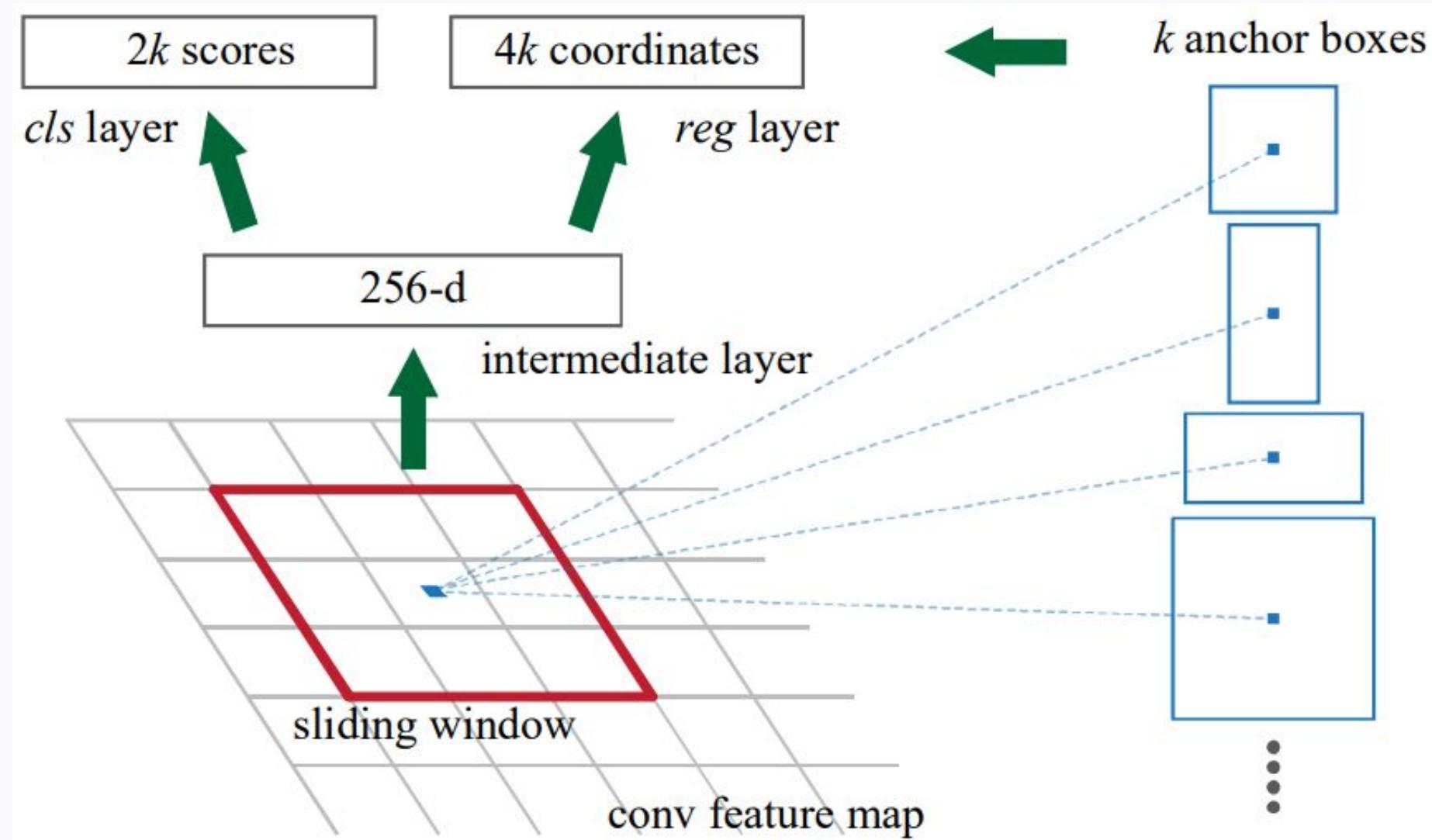
Filters
pyramid

Pyramid of
reference boxes

FASTER R-CNN



Region Proposal Networks (RPN)



Loss Function

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

где i - индекс в батче, p_i - предсказанная вероятность, p^* - бинарная метка, t_i - предсказанный бокс, t^* - gt бокс.

$$L_{reg}(t_i, t_i^*) = R(t_i - t_i^*) \quad \text{где } R:$$

$L_{cls}(p_i, p_i^*)$ - 2-way softmax,

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise} \end{cases}$$

Bounding Box Regression

$$\begin{aligned} t_x &= (x - x_a) / w_a, & t_y &= (y - y_a) / h_a, \\ t_w &= \log(w / w_a), & t_h &= \log(h / h_a), \\ t_x^* &= (x^* - x_a) / w_a, & t_y^* &= (y^* - y_a) / h_a, \\ t_w^* &= \log(w^* / w_a), & t_h^* &= \log(h^* / h_a), \end{aligned}$$

- predictions
- ground truth

где x, y, w, h - центр бокса и его ширина/высота,

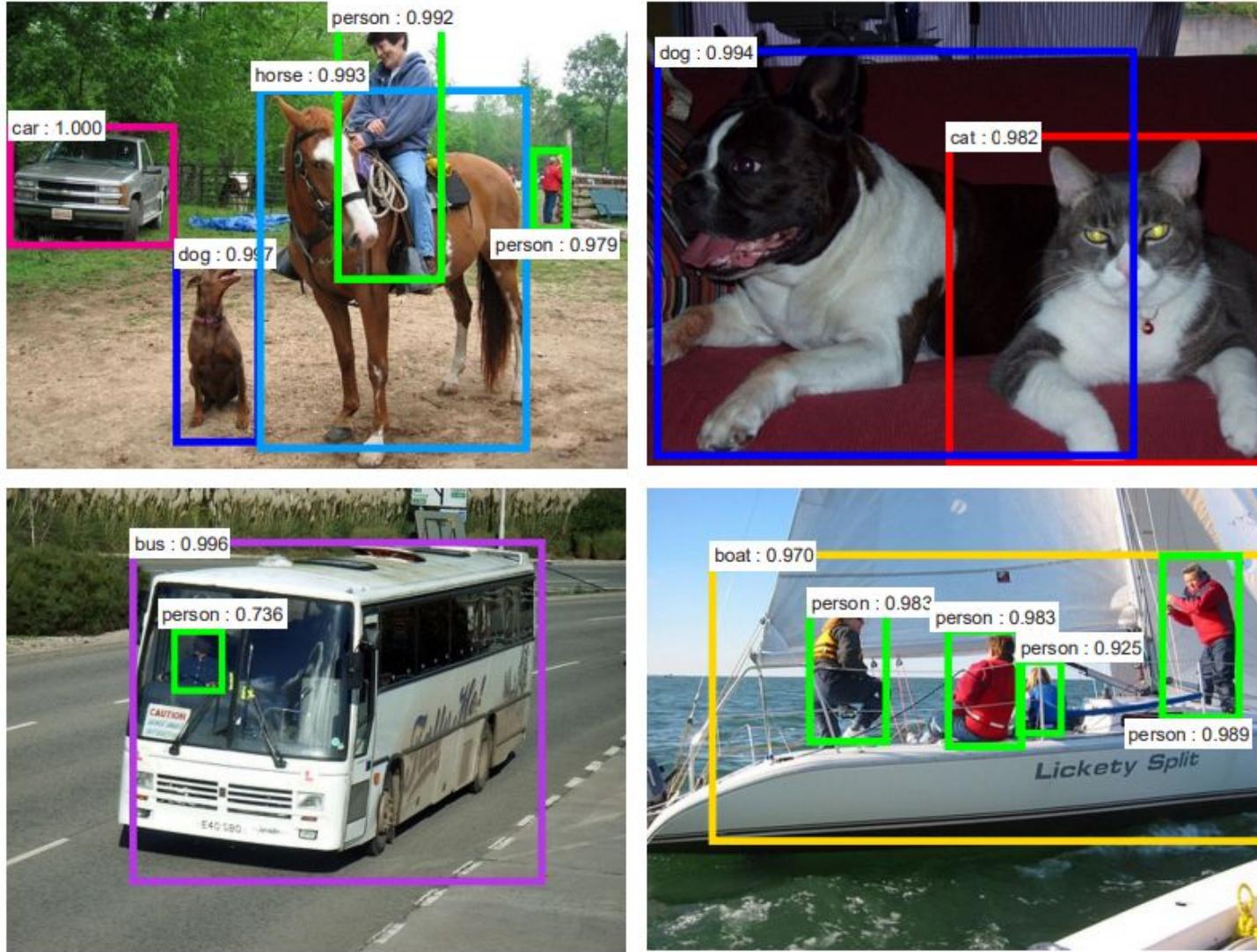
x, x_a, x^* - предсказанный бокс, *anchor* бокс и *gt* бокс, соответственно
(аналогично для y, w, h).

Training RPN

Обучение происходит в четыре этапа:

- 1) *RPN*, предобученная на *ImageNet*, обучается на *region proposals* задаче.
- 2) Детектор, предобученный на *ImageNet*, обучается на *proposals* из 1.
- 3) *RPN* инициализируется из 2, расшариваемые веса замораживаются, и *RPN* дотюнивается.
- 4) Детектор инициализируется из 3, расшариваемые веса замораживаются, и детектор дотюнивается.

Implementation details

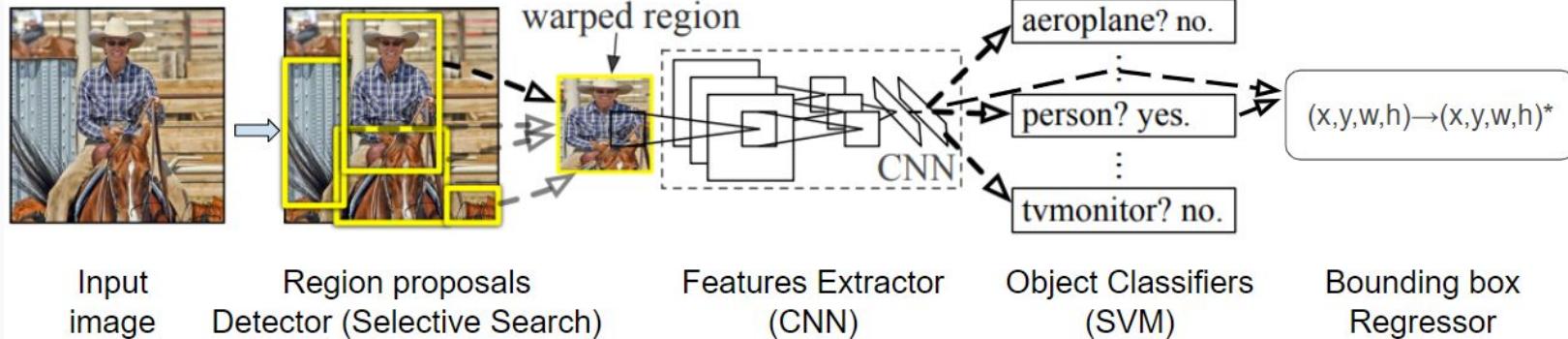


MS COCO: Fast R-CNN vs Faster R-CNN

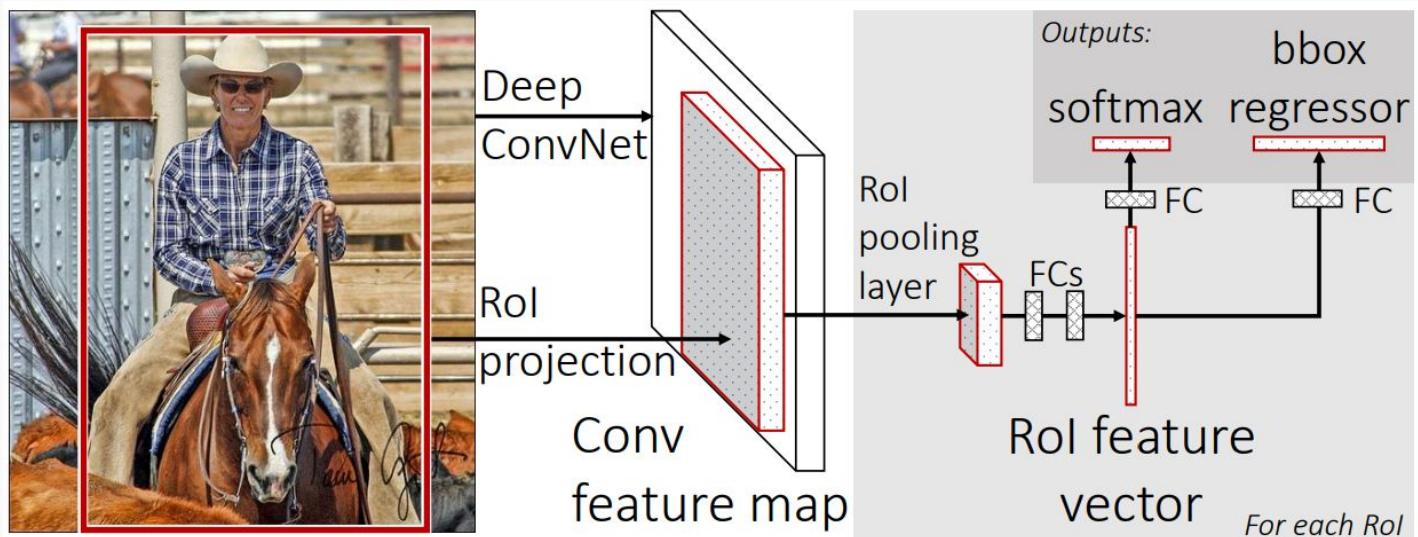
method	proposals	training data	COCO val		COCO test-dev	
			mAP@.5	mAP@[.5, .95]	mAP@.5	mAP@[.5, .95]
Fast R-CNN [2]	SS, 2000	COCO train	-	-	35.9	19.7
Fast R-CNN [impl. in this paper]	SS, 2000	COCO train	38.6	18.9	39.3	19.3
Faster R-CNN	RPN, 300	COCO train	41.5	21.2	42.1	21.5
Faster R-CNN	RPN, 300	COCO trainval	-	-	42.7	21.9

R-CNN vs Fast R-CNN vs Faster R-CNN

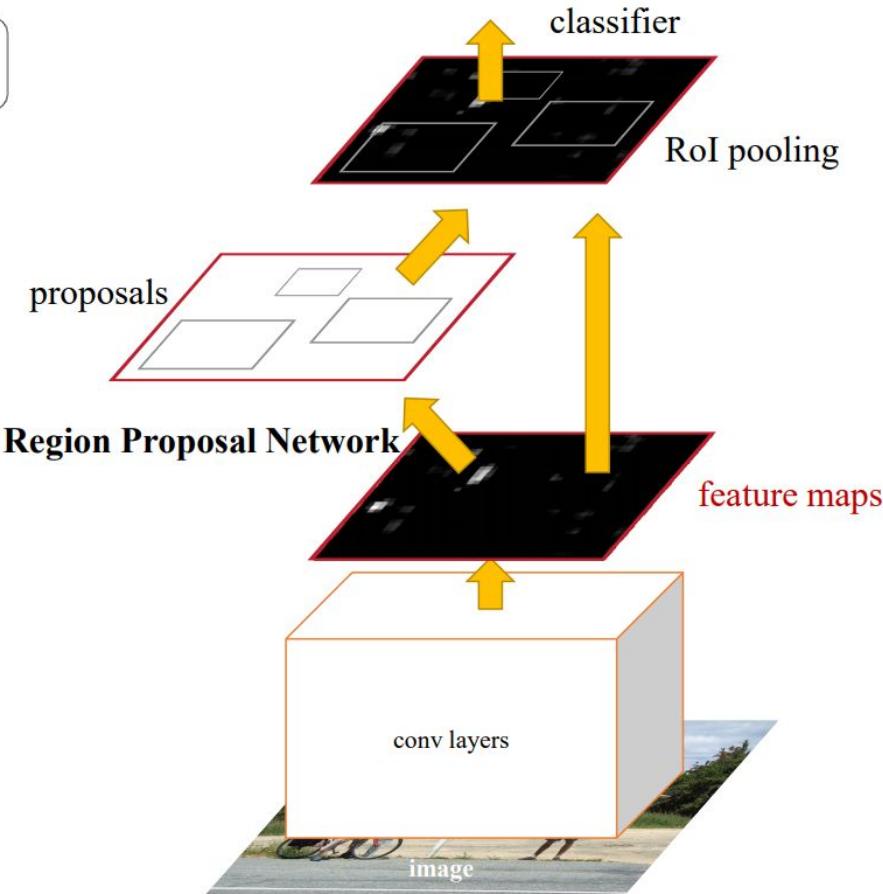
R-CNN



Fast R-CNN



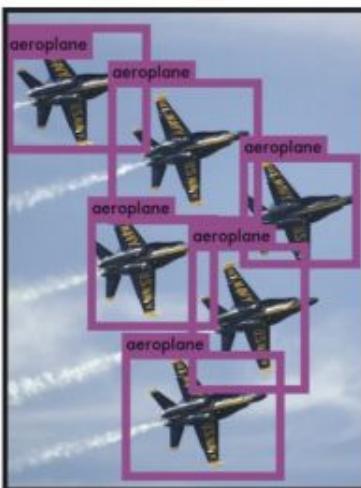
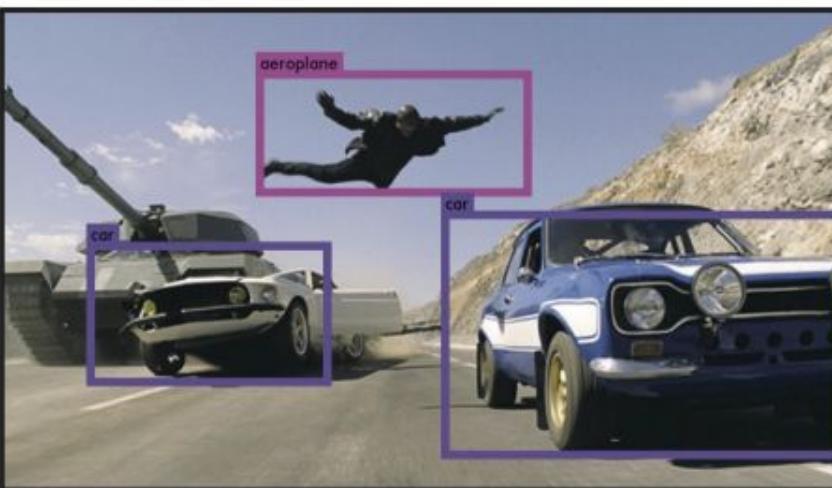
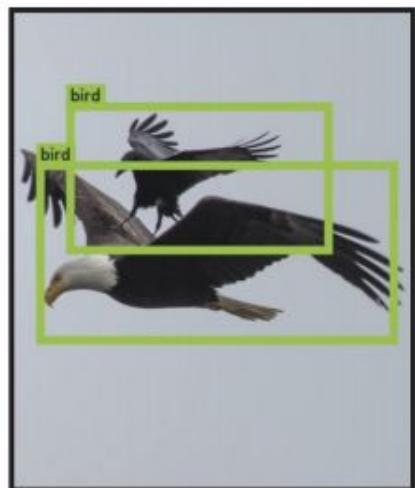
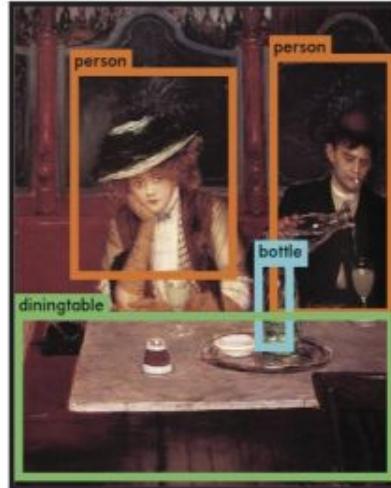
Faster R-CNN



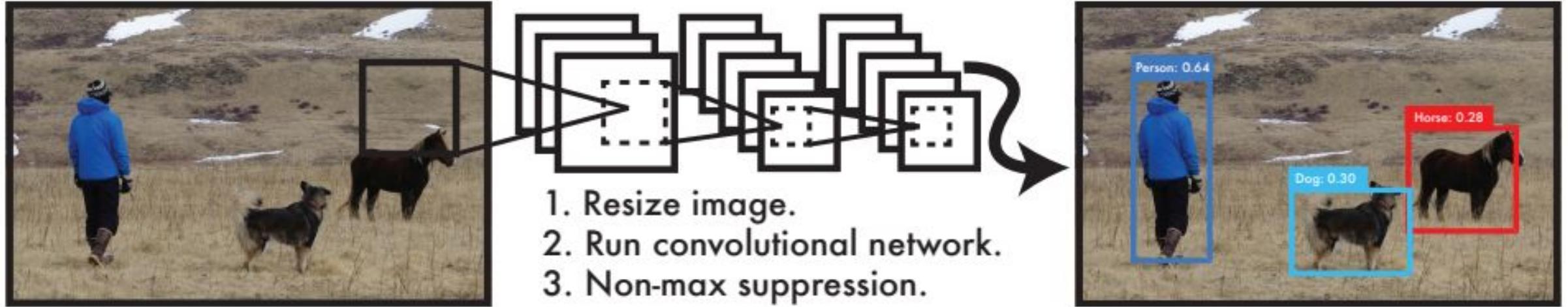
YOLO

(You Only Look Once)

YOLO: You Only Look Once



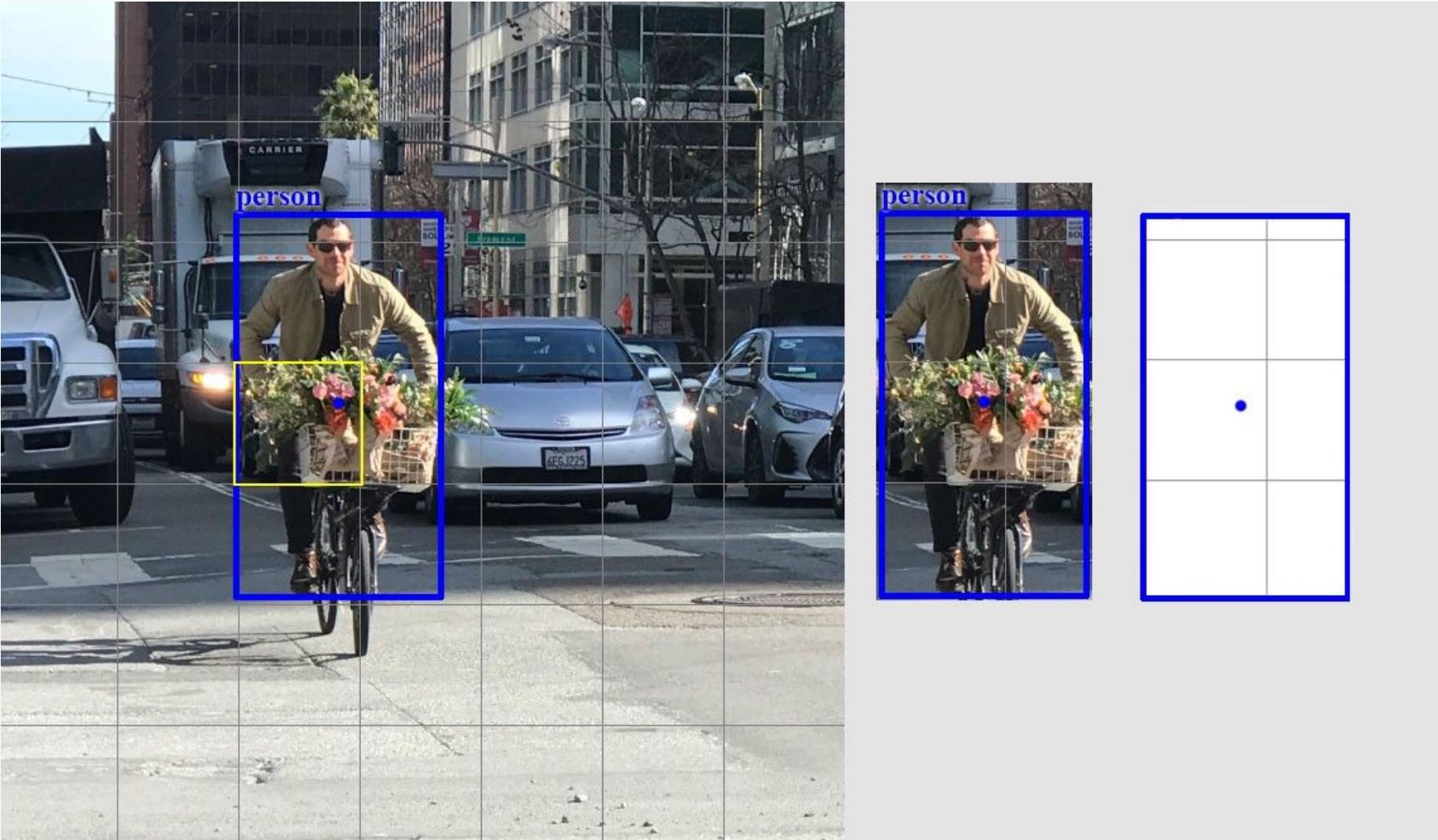
Unified Detection



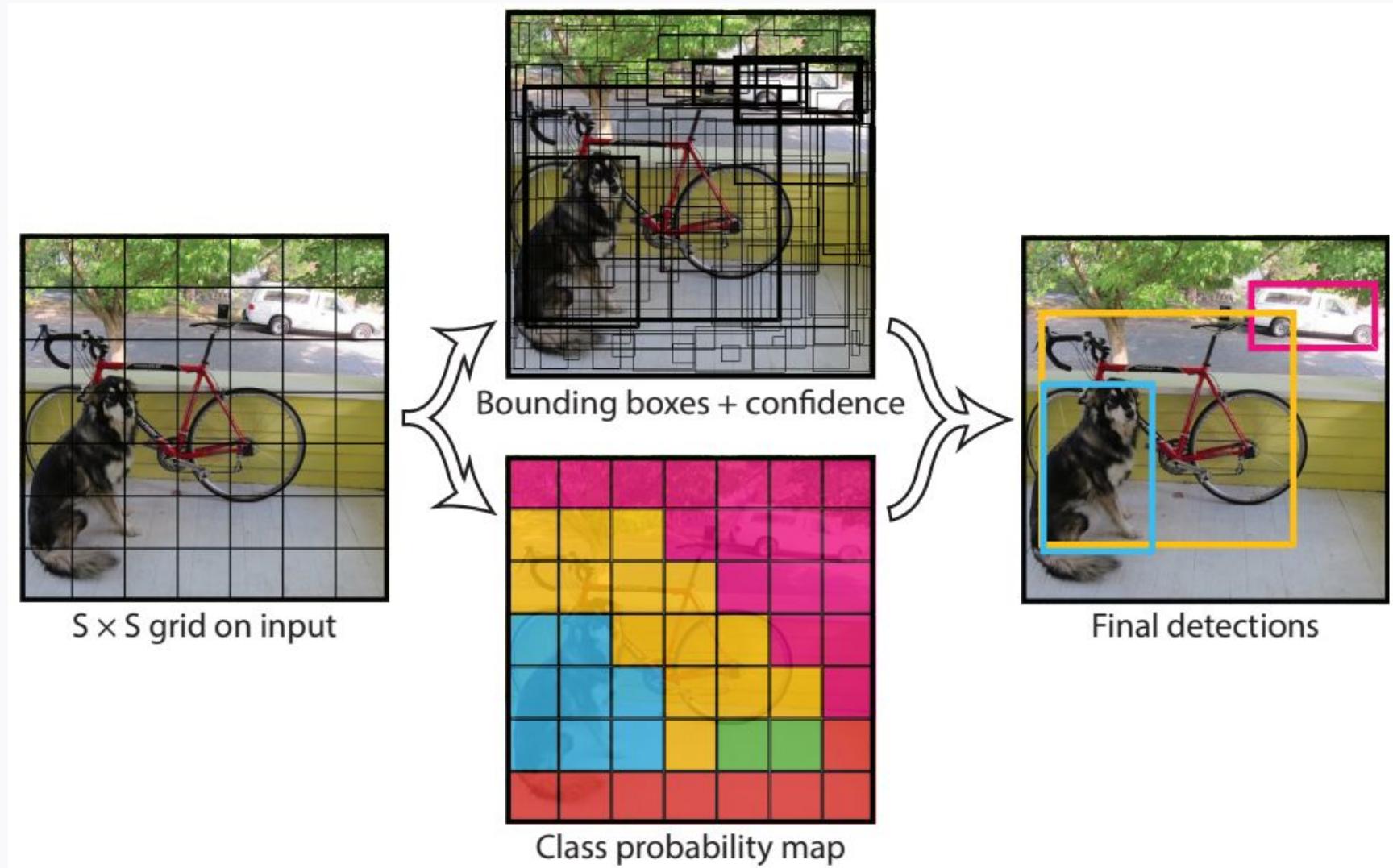
Объединение всех компонентов детектора в единую сеть позволяет:

- обучать модель end-to-end
- делать предсказания за один проход
- использовать контекстную информацию со всего изображения

Grid Cell



Grid Cell



Bounding Boxes

Каждый **bounding box** состоит из $[x, y, w, h, \text{box confidence}]$,

где (x, y) - центр бокса относительно ячейки (смещение),

(w, h) - ширина и высота нормализованы относительно всего изображения.

Все значения находятся в диапазоне $[0;1]$.

Box confidence score, Class probability, Class confidence score

box confidence score $\equiv P_r(\text{object}) \cdot IoU$

conditional class probability $\equiv P_r(\text{class}_i | \text{object})$

class confidence score $\equiv P_r(\text{class}_i) \cdot IoU$

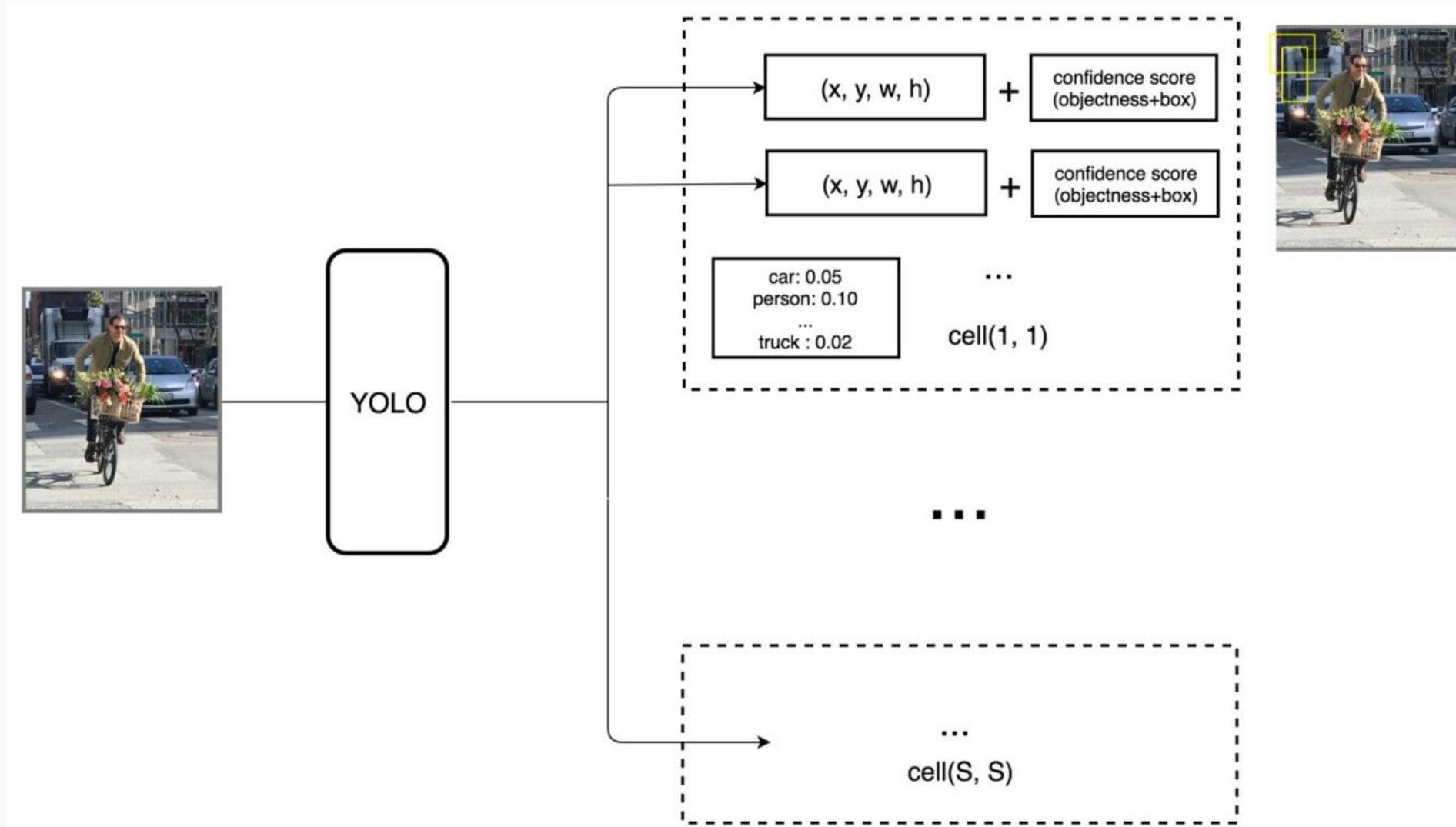
$=$ box confidence score \times conditional class probability

Box confidence score - уверенность модели в том, что бокс содержит объект и как точно этот объект локализован.

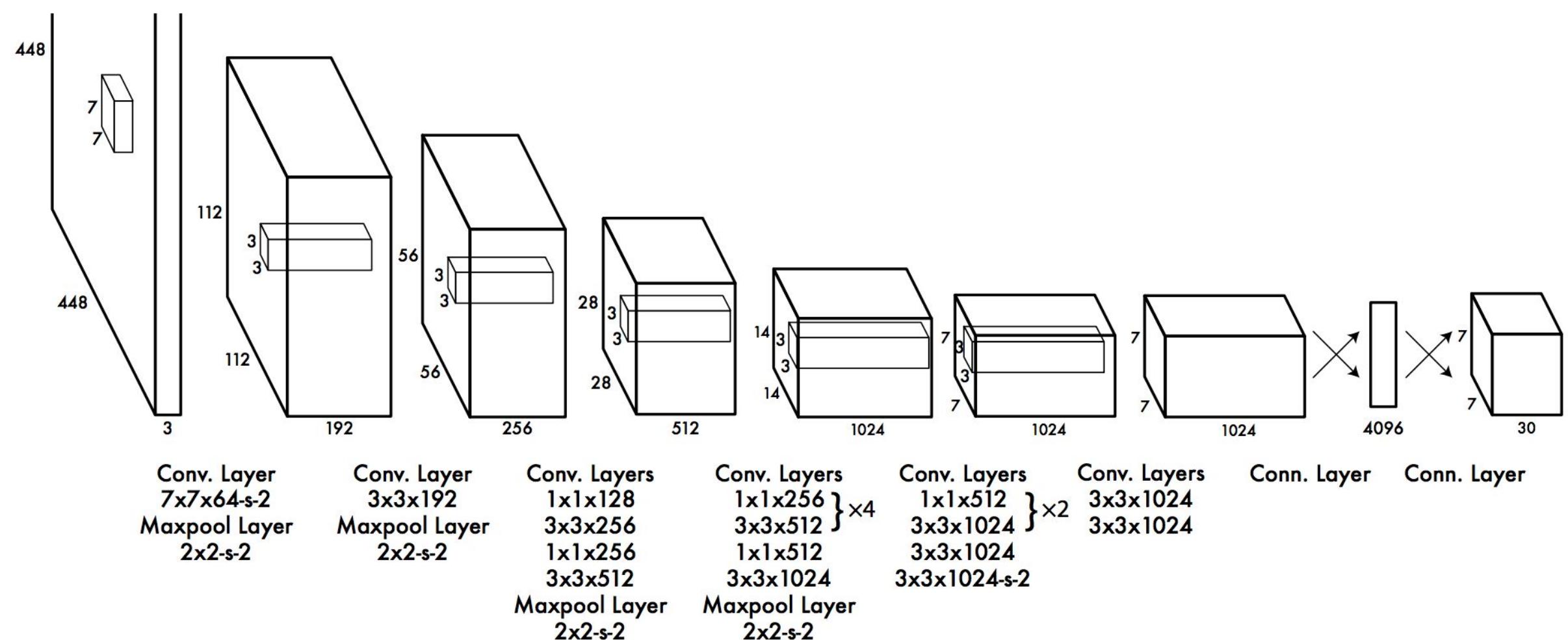
Conditional class probabilities - список вероятностей классов объекта.

Class confidence score - итоговая оценка отражающая точность классификации объекта и точность локализации бокса для него.

YOLO pipeline



Network Design



Loss Function

“Ответственный” предиктор - предиктор, чей бокс имеет максимальный IoU с ground truth.

YOLO использует ***Sum of Squared Errors (SSE)*** в качестве loss function.

Полный loss модели состоит из:

- 1) ***Classification loss,***
- 2) ***Localization loss,***
- 3) ***Confidence loss.***

Classification loss

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

где $\mathbb{1}_i^{\text{obj}}$ - равен 1, если объект присутствует в ячейке i , иначе 0,
 $\hat{p}_i(c)$ - вероятность того, что объект класса c присутствует в ячейки i .

Localization loss

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

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

где $\mathbb{1}_{ij}^{\text{obj}}$ - равен 1 если предиктор j в ячейке i “ответственен” за детекцию этого объекта, иначе 0,

λ_{coord} - нормировочный коэффициент (равен 5).

Confidence loss

Для “ответственного” предиктора:

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

где \hat{C}_i - *box confidence score* предиктора j в ячейке i .

Для предикторов, которые не содержат объект:

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

где $\mathbb{1}_{ij}^{\text{noobj}}$ - равен 1 если предиктор j в ячейке i не содержит объект,
 λ_{noobj} - нормировочный коэффициент (равен 0.5).

Full loss

Localization loss

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

Confidence loss

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

Classification loss

—

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

Training

Обучение YOLO происходит в два этапа:

- 1) Сеть предобучается на задаче классификации (*ImageNet*) со входом 224×224 .
- 2) Затем полносвязные слои заменяются сверточными, вход увеличивается до 448×448 , и сеть дотюнивается на задаче детекции (*VOC* или *COCO*).

Limitations of YOLO

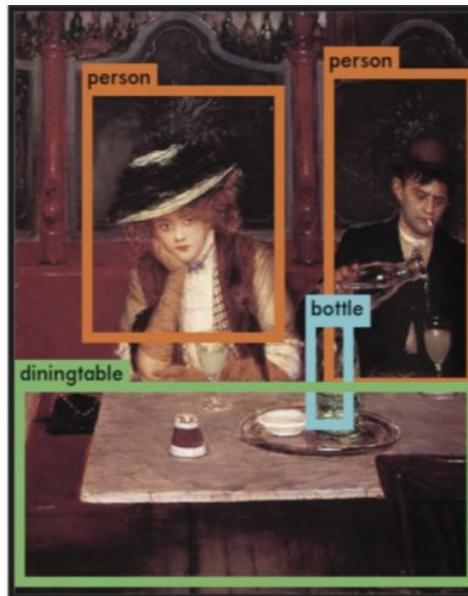


Real-Time Detectors on PASCAL VOC 2007

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

Benefits of YOLO

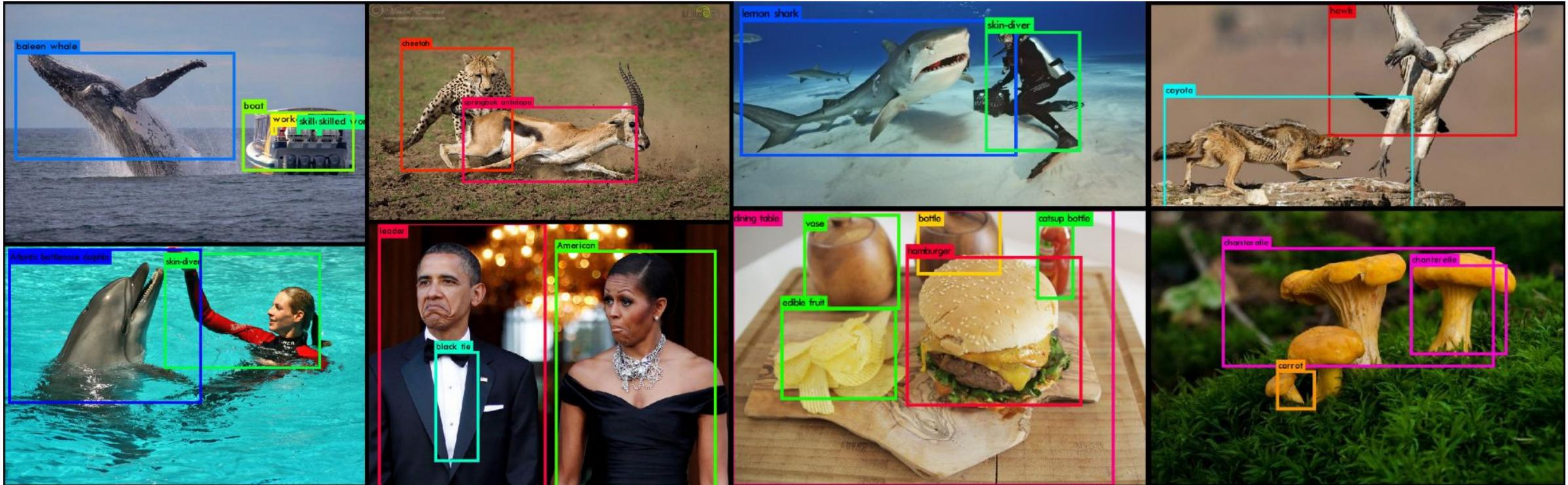
- Очень быстрая детекция.
- Модель может быть обучена end-to-end.
- Используется контекст объектов.
- Хорошее обобщение на объекты из других доменов:



YOLOv2 / YOLO9000

(Better, Faster, Stronger)

Accuracy improvements



2 High-resolution classifier

Обучение YOLO:

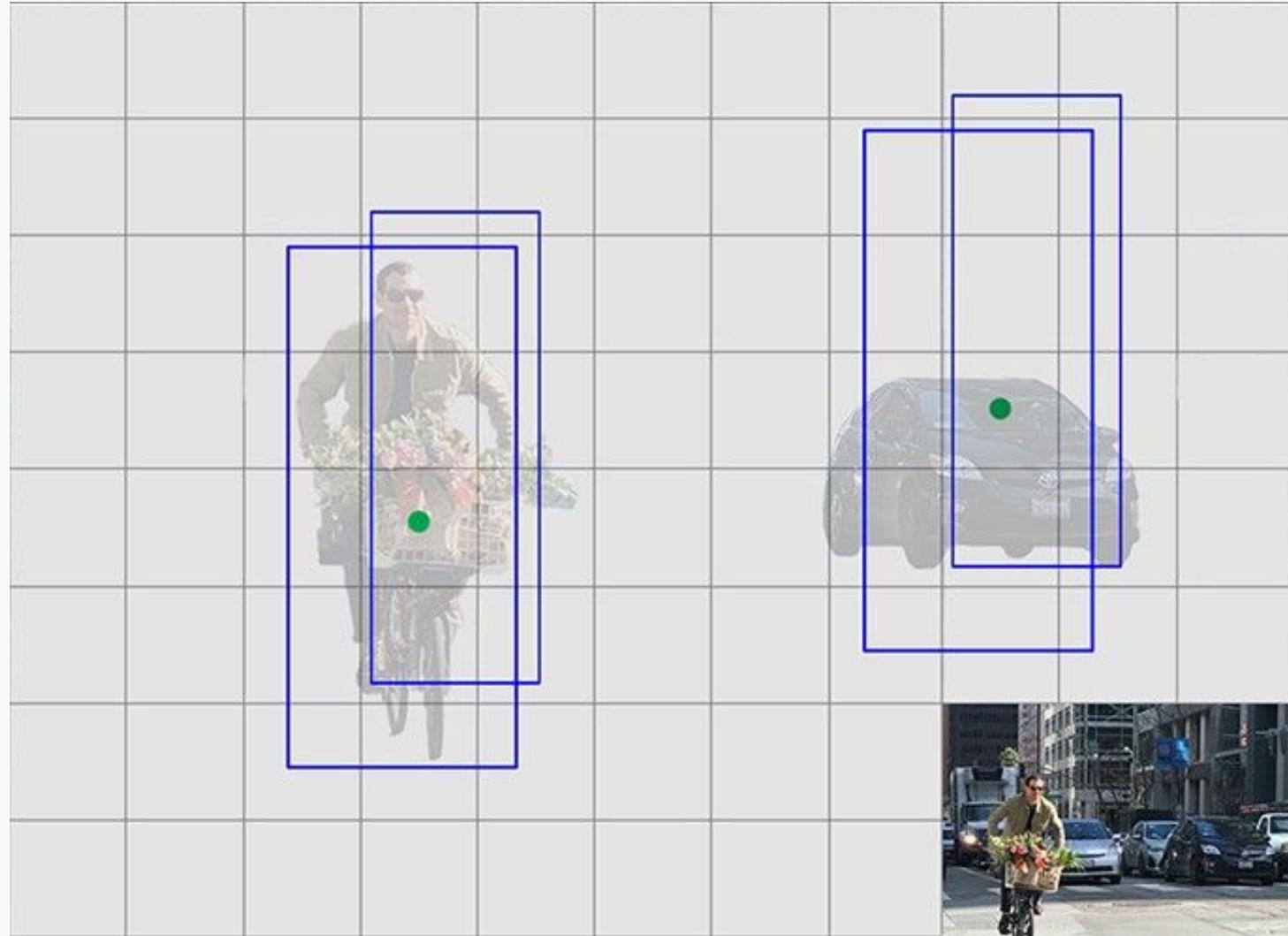
- 1) Сеть обучается на задаче классификации (*ImageNet*) со входом 224×224 .
- 2) Затем вход увеличивается до 448×448 , и сеть дотюнивается на задаче детекции (*VOC* или *COCO*).

Обучение YOLOv2:

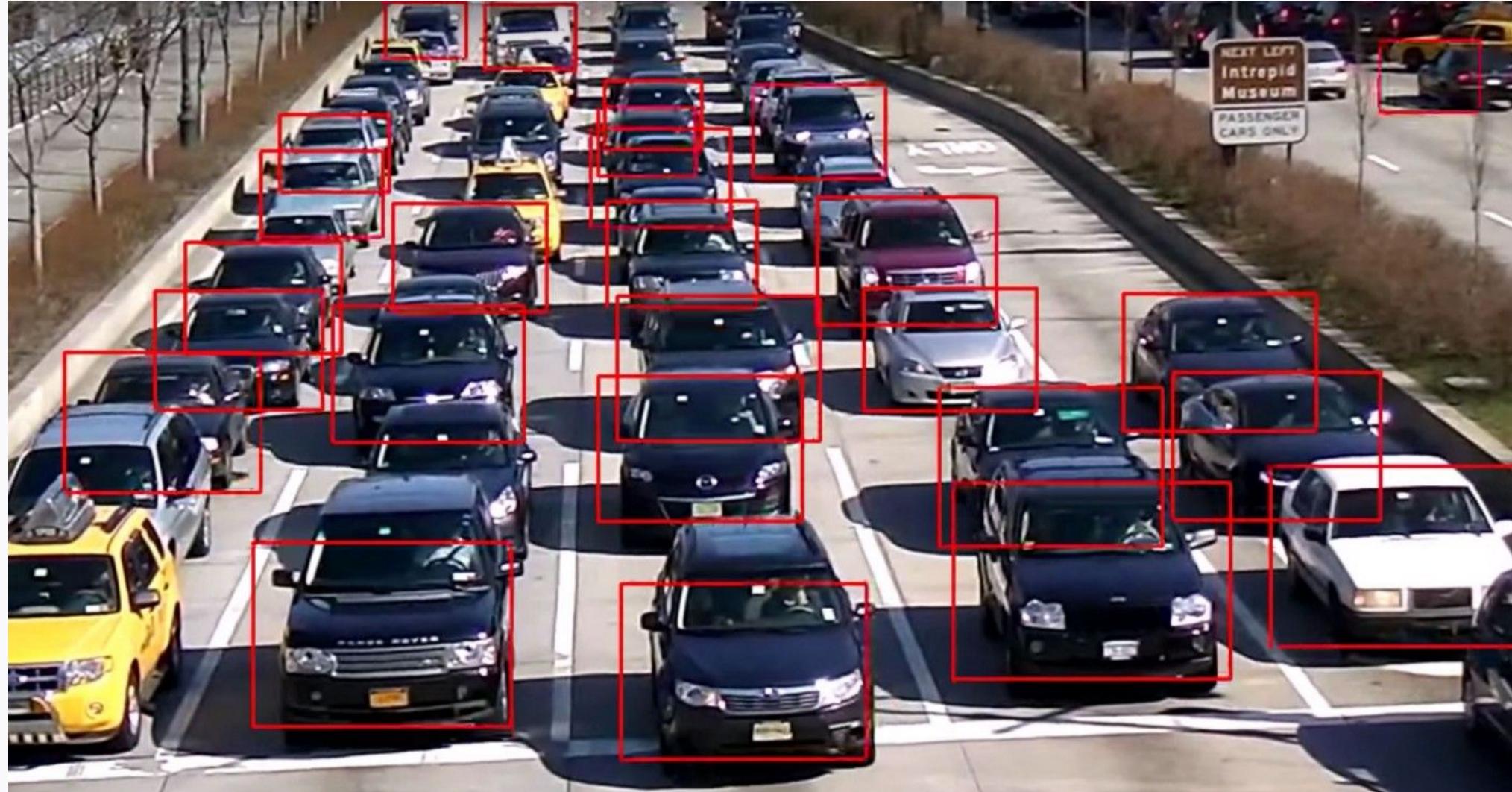
- 1) Сеть обучается на задаче классификации (*ImageNet*) со входом 224×224 .
- 2) Затем вход увеличивается до 448×448 и сеть обучается еще несколько эпох.
- 3) После сеть дотюнивается на задаче детекции (*VOC* или *COCO*).

Такая схема обучения увеличивает *mAP* на 4%.

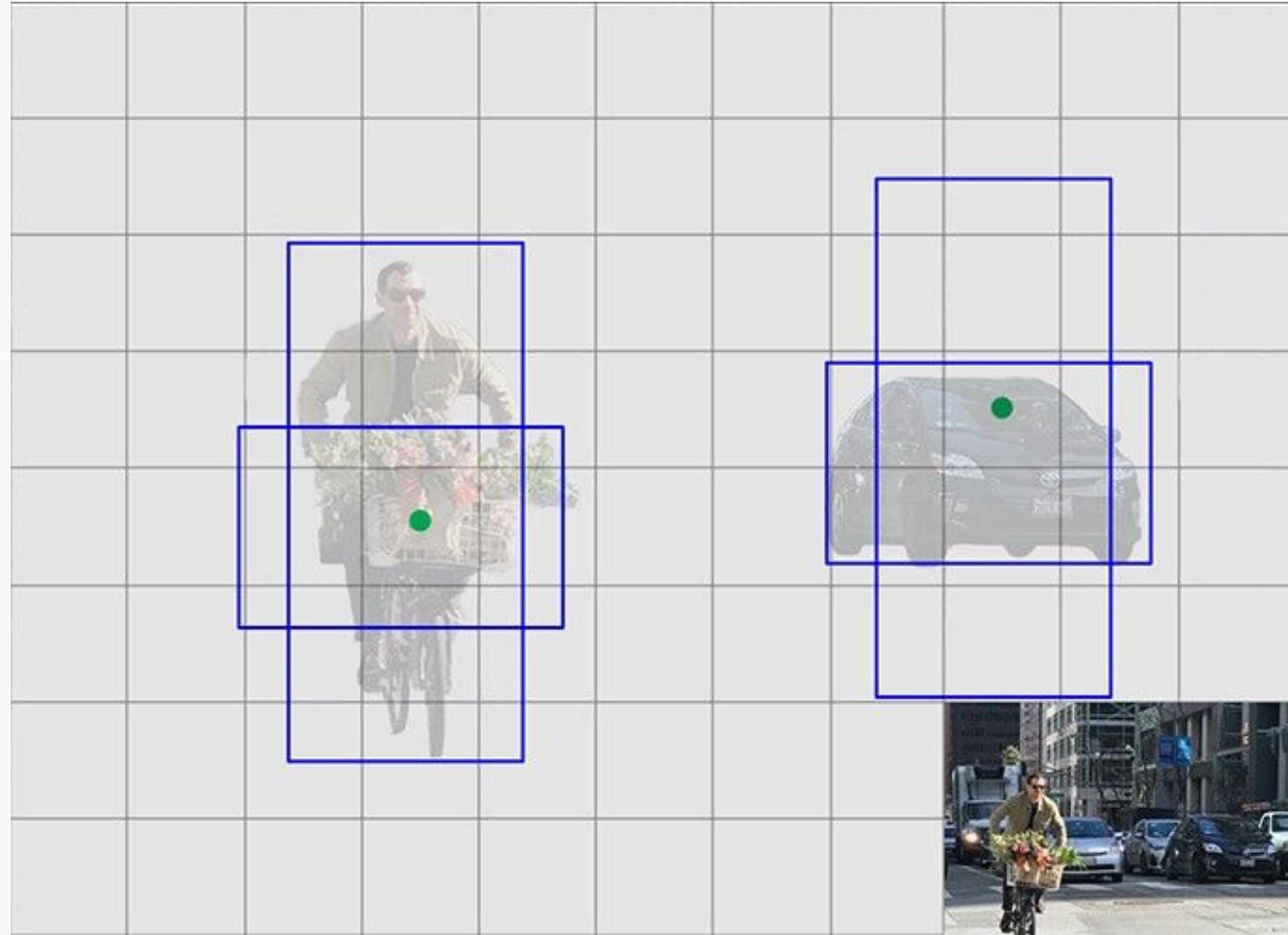
3 Convolutional with Anchor Boxes



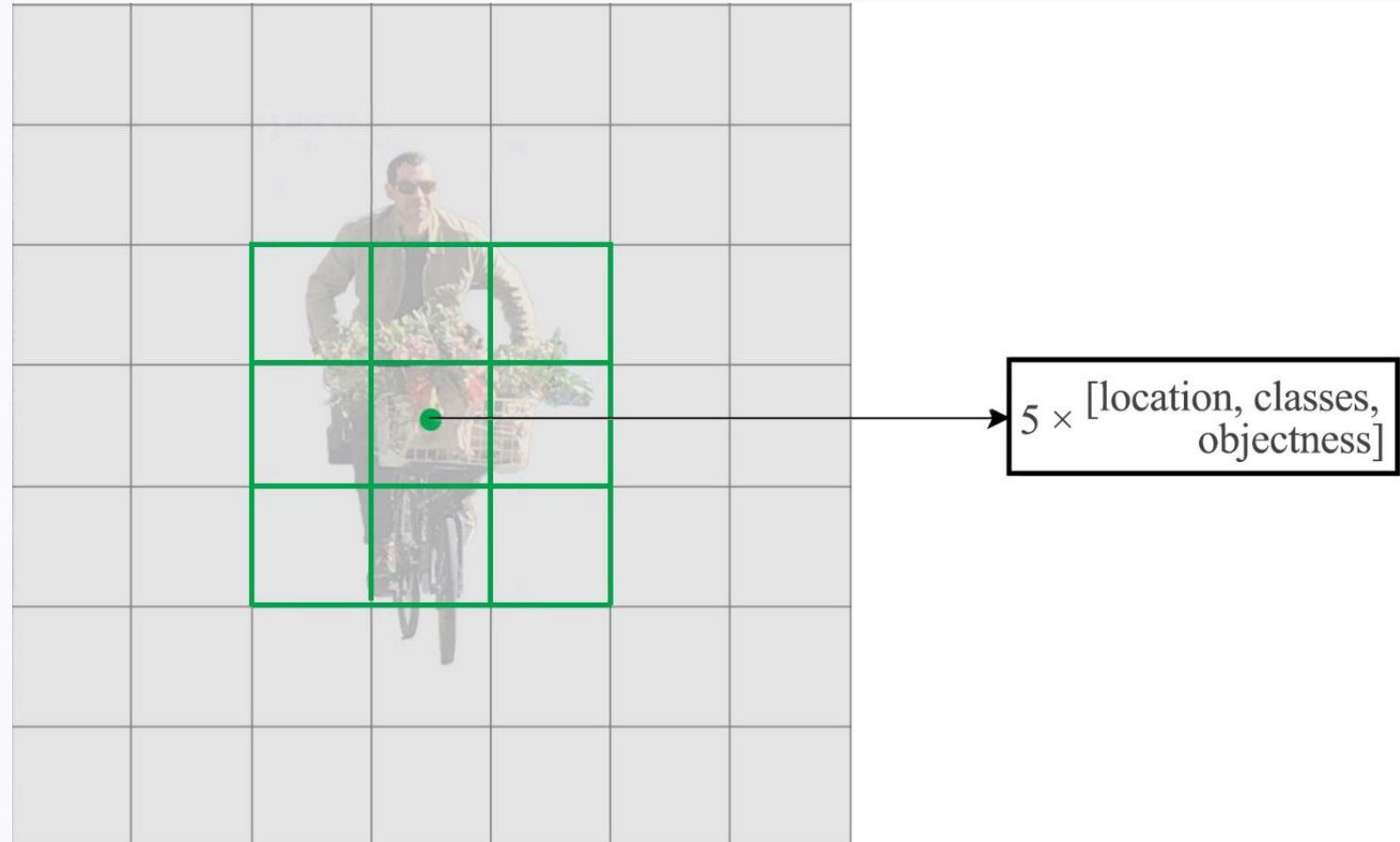
3 Convolutional with Anchor Boxes



3 Convolutional with Anchor Boxes



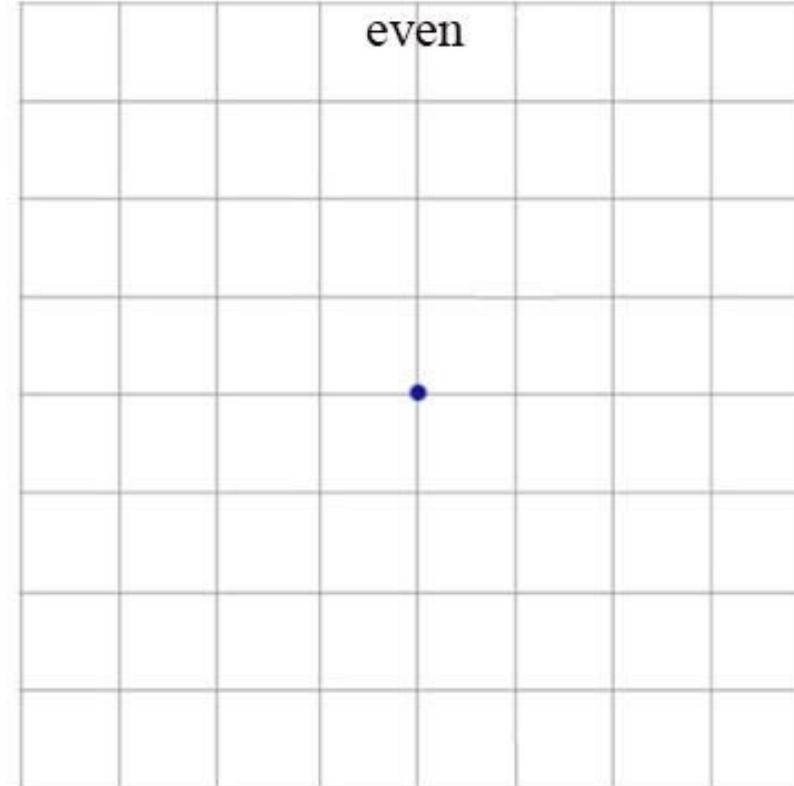
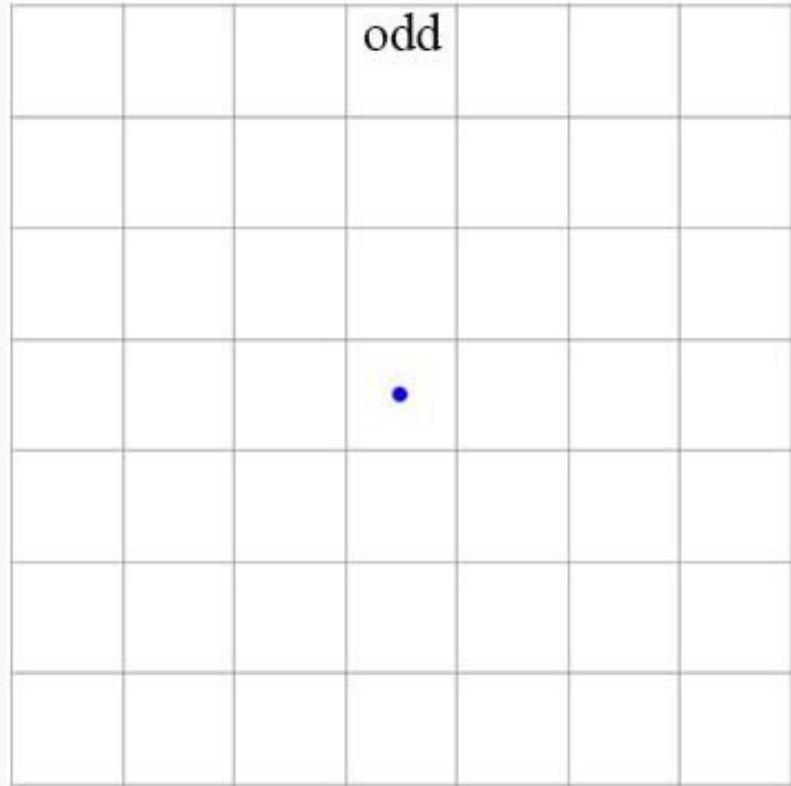
3 Convolutional with Anchor Boxes



Значений в каждой ячейке: $5 \times (4 + 1 + 20) = 125$.

Выходной тензор $(7 \times 7 \times 125)$.

3 Convolutional with Anchor Boxes



Размер входа сети изменяется с 448×448 на 416×416 для нечетного числа ячеек.

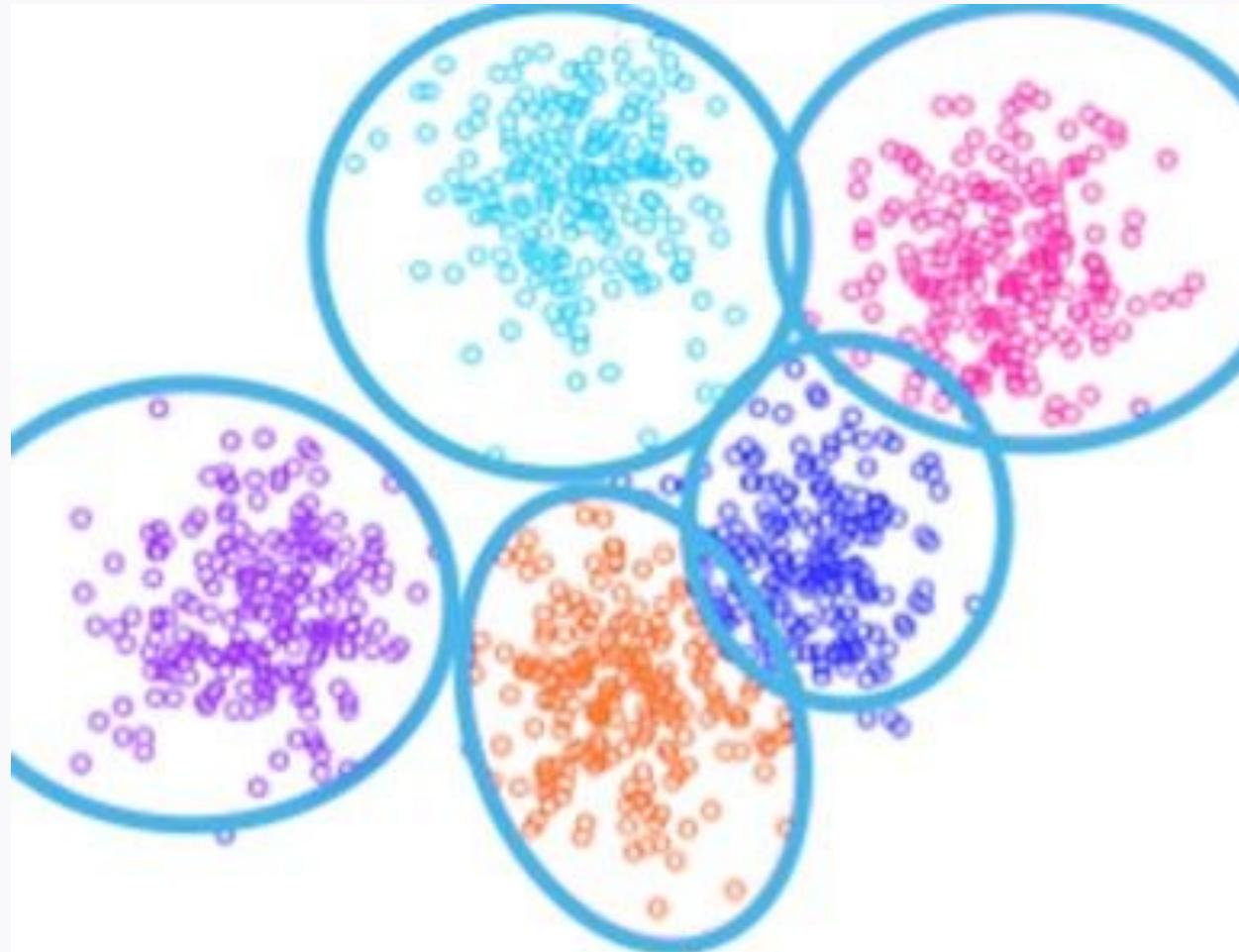
Размер выхода сети увеличивается с 7×7 на 13×13 за счет удаления *pooling* слоя.

4 ?

Как выбрать якорные боксы (*anchors*)?

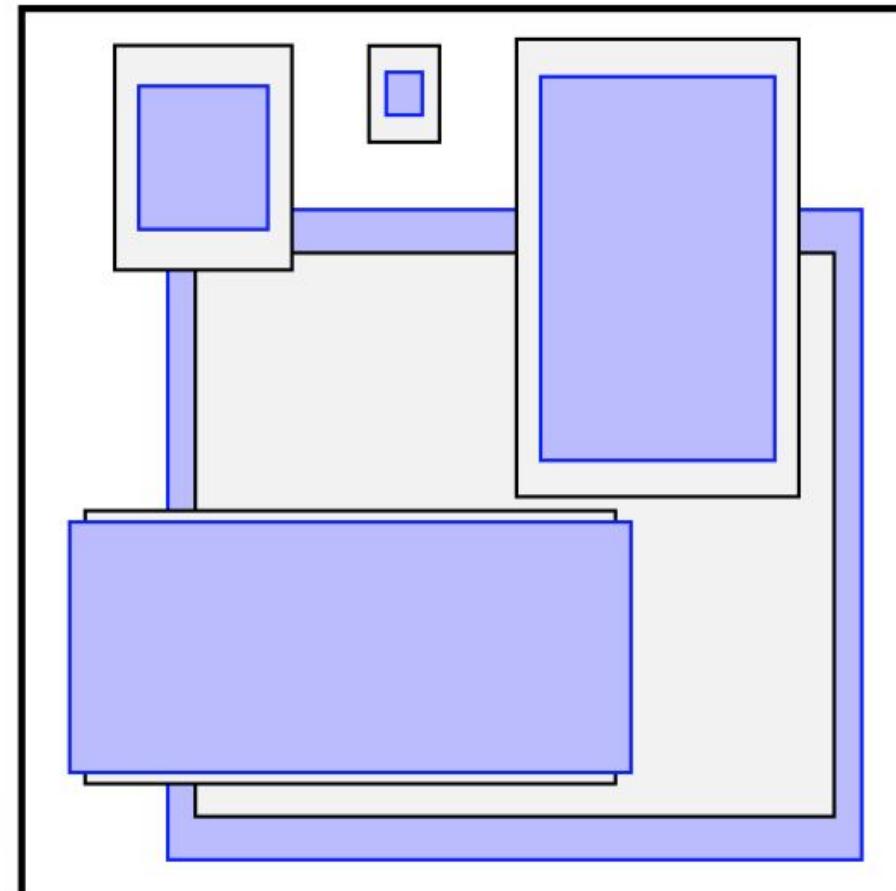
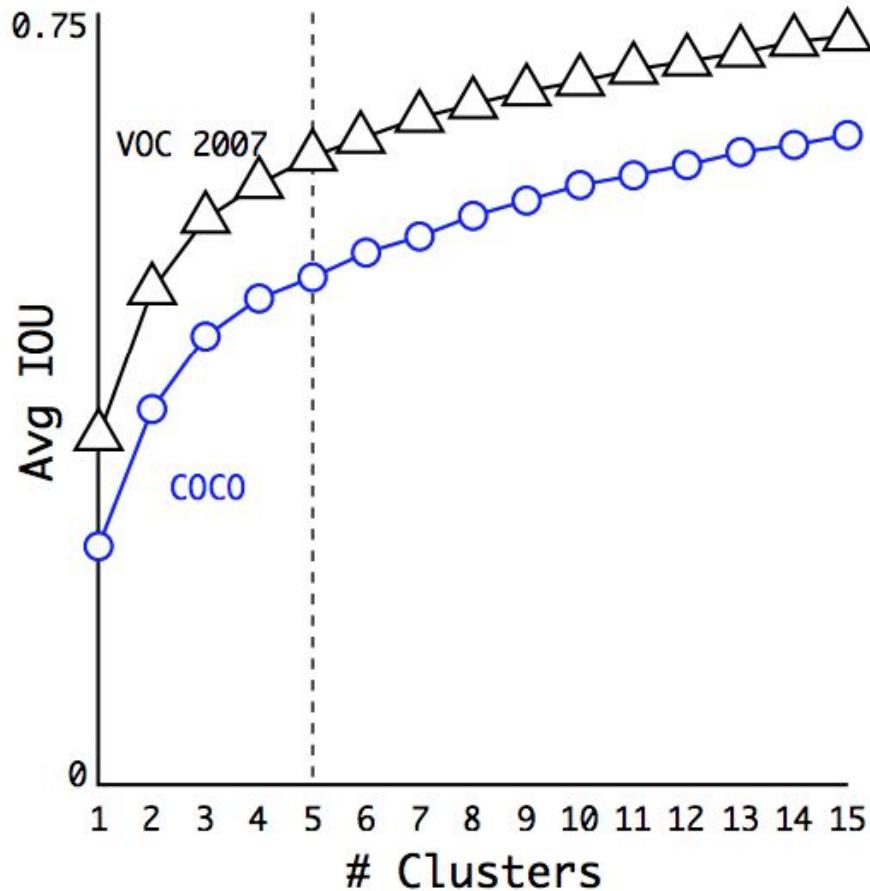


4 Dimension Clusters



$$d(\text{box, centroid}) = 1 - \text{IOU}(\text{box, centroid})$$

4 Dimension Clusters



5 Direct location prediction

B Region Proposals Network (RPN):

$$x = (t_x * w_a) - x_a$$

$$y = (t_y * h_a) - y_a$$

В YOLOv2 предсказывается:

$$t_x, t_y, t_w, t_h, t_o$$

где

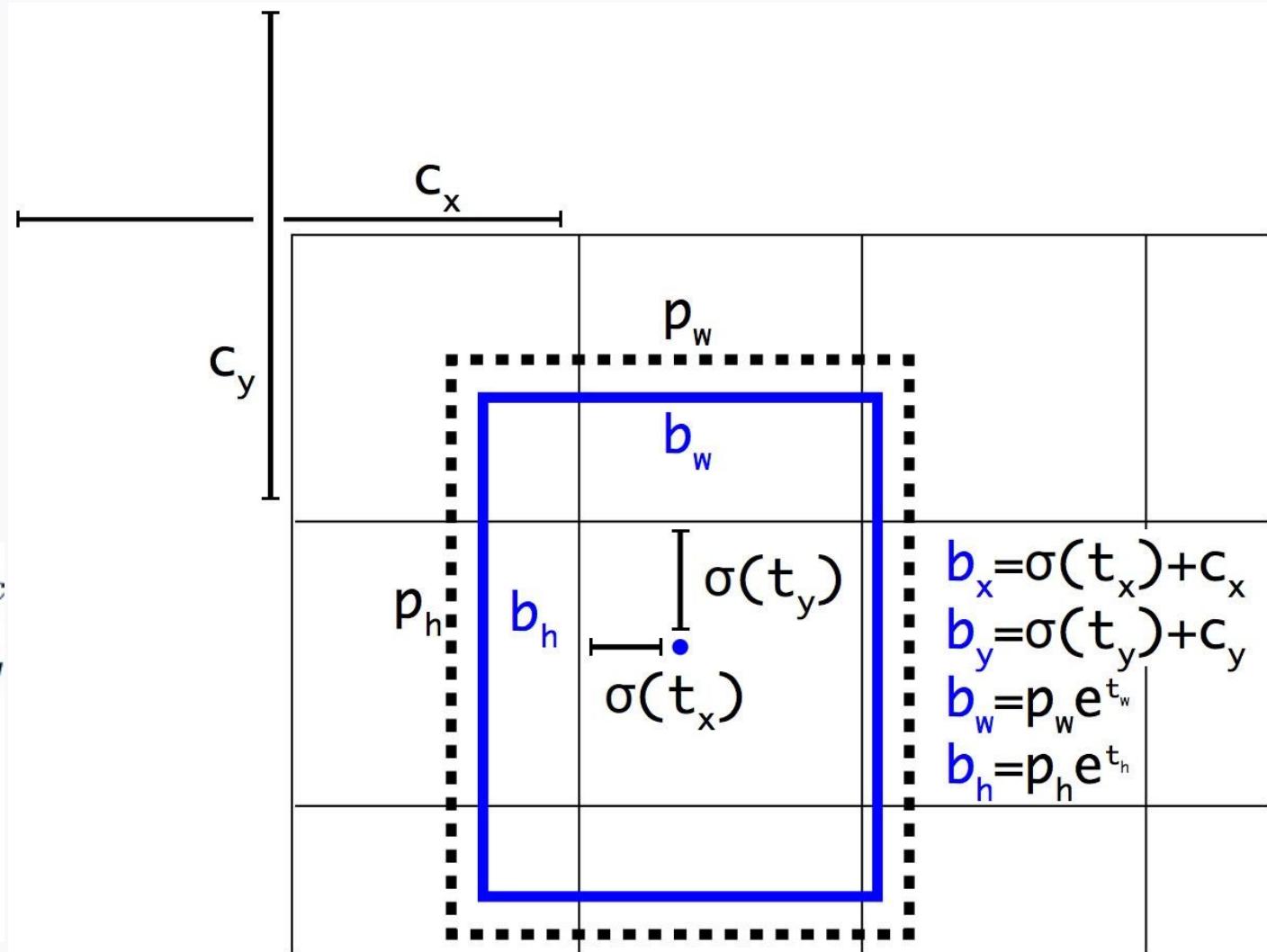
$$b_x = \sigma(t_x) + c_x$$

$$b_y = \sigma(t_y) + c_y$$

$$b_w = p_w e^{t_w}$$

$$b_h = p_h e^{t_h}$$

$$Pr(\text{object}) * IOU(b, \text{object}) = \sigma(t_o)$$



7 Multi-Scale Training

Сеть YOLOv2 является ***Fully Convolutional Network (FCN)***,
а ее коэффициент ***downsampling*** равен 32.

Во время обучения каждые 10 батчей случайно выбирается входное разрешение из списка:

$\{320, 352, 384, \dots, 576, 608\}$ с шагом 32.

На низких разрешениях модель работает менее точно, но быстрей.

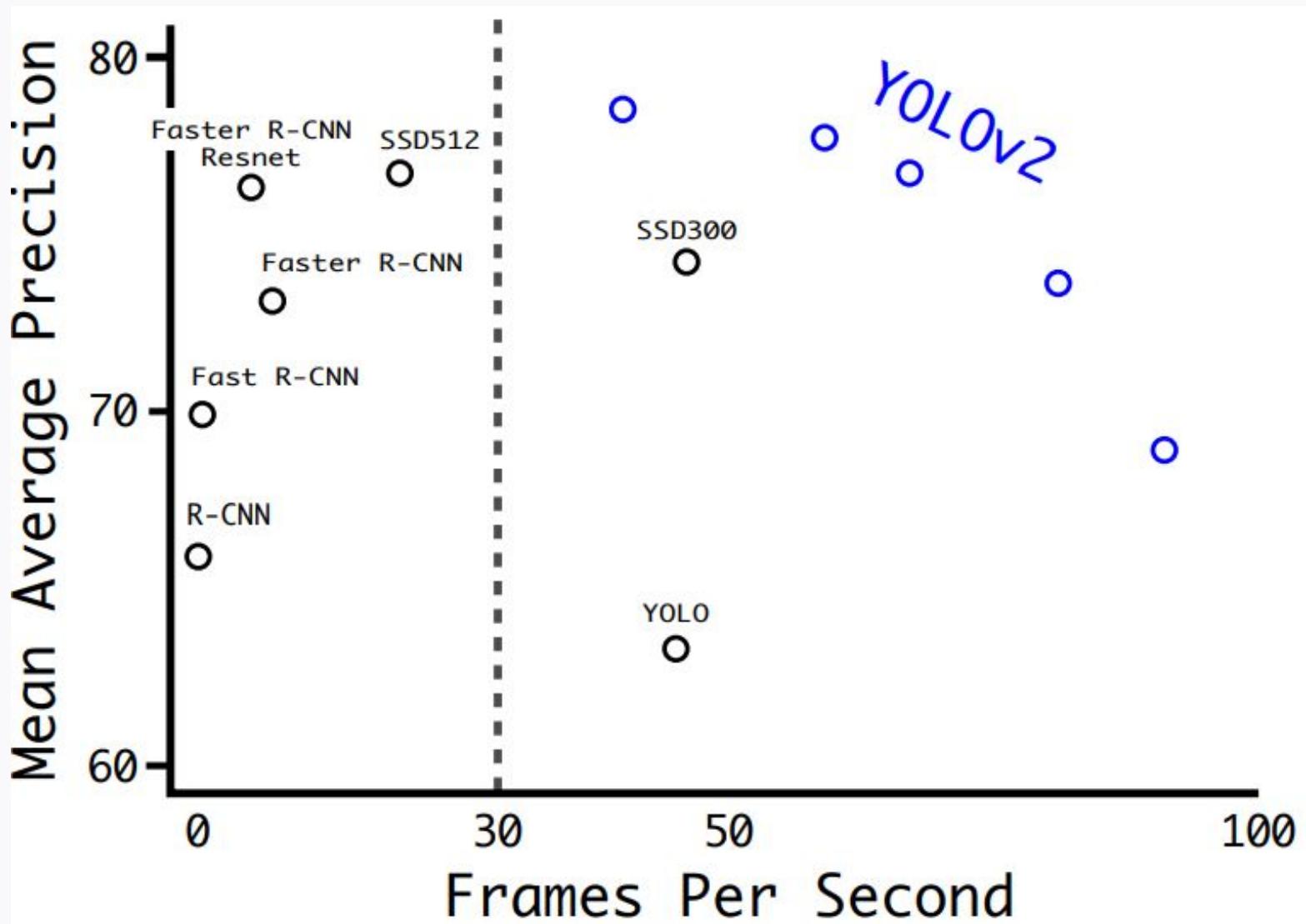
7 Multi-Scale Training

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

Accuracy Improvement

	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

PASCAL VOC 2007



DarkNet-19 for classification

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2/2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2/2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2/2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2/2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2/2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	
Softmax			1000

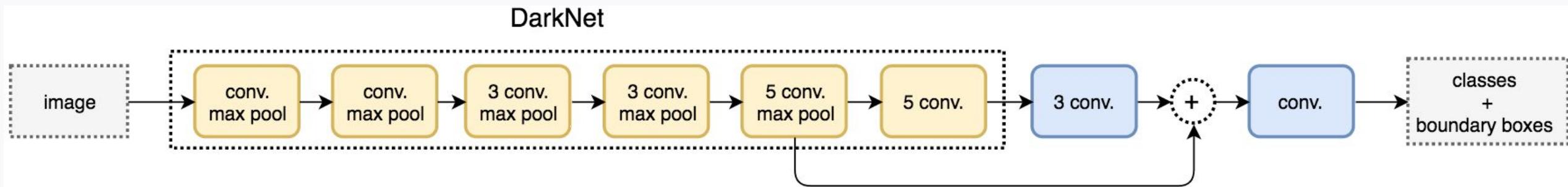
DarkNet-19 имеет 5.8 млрд параметров,
кастомная сеть **YOLO** - 8.52 млрд.

DarkNet-19 достигает на **ImageNet**
точности 91.2% (top-5).

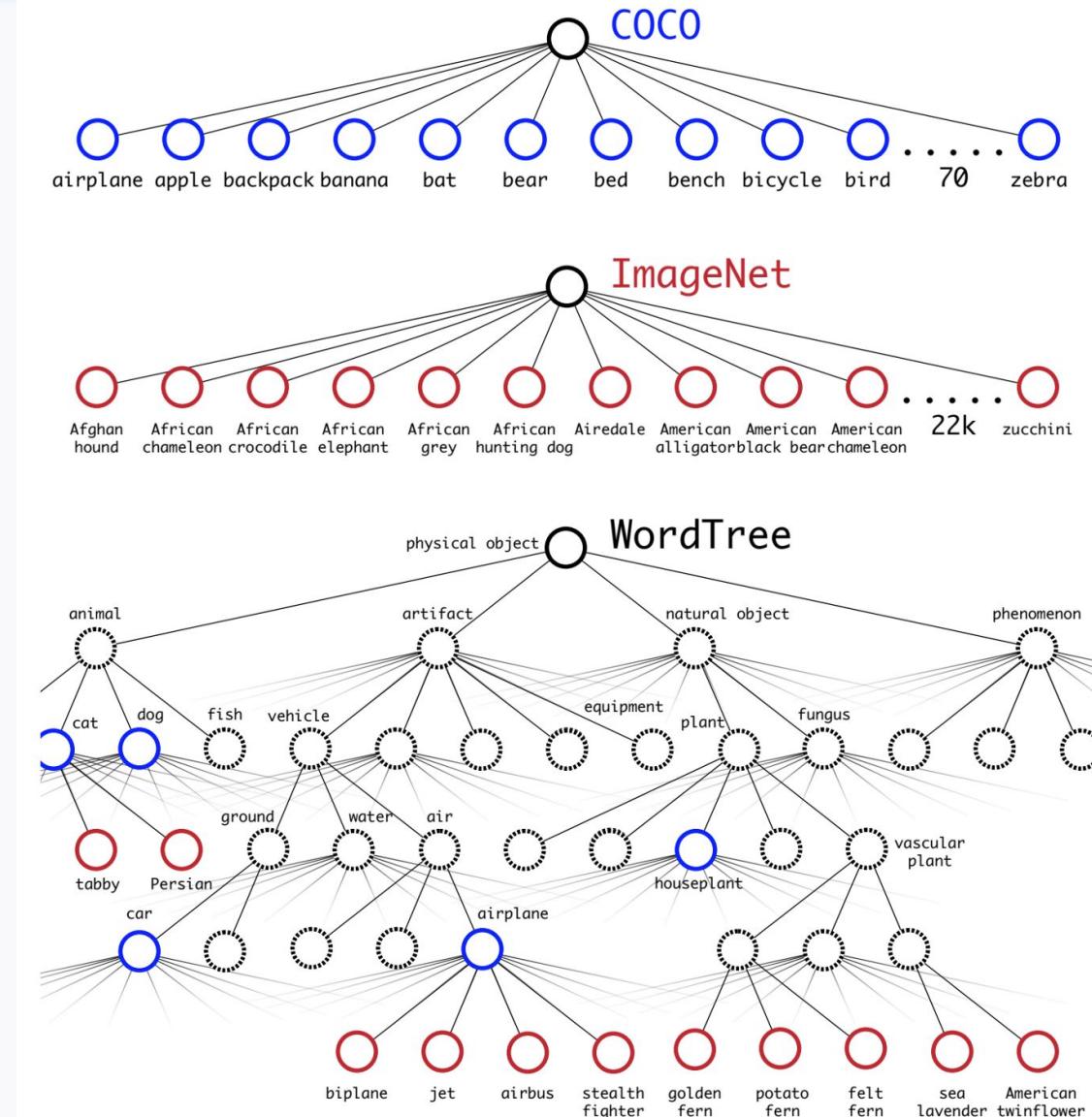
DarkNet-19 for detection

DarkNet-19 для задачи детекции на VOC использует выходной тензор ($7 \times 7 \times 125$):

- 5 боксов с 4 координатами +
- 1 *confidence score* +
- 20 *conditional class probabilities*.



Classification



YOLOv3

(An Incremental Improvement)

Class Prediction

YOLO использует взаимоисключающие метки классов, YOLOv2 - их иерархию, а YOLOv3 - не взаимоисключающие метки.

Предыдущие YOLO и YOLOv2 решают задачу ***multi-class classification*** и используют ***softmax*** с ***cross-entropy loss***.

YOLOv3 решает задачу ***multi-label classification*** и использует ***logistic classifiers*** с ***binary cross-entropy loss***.

Predictions Across Scales

YOLOv3 делает три предсказания в каждой ячейке. Предсказание состоит из:

- ***bounding box offset***
- ***objectness score***
- ***class probability*** для N классов.

Выходной тензор YOLOv3: $N \times N \times [3 * (4 + 1 + 80)]$,

где $N \times N$ - число ячеек в *grid cell*,

3 предсказания на ячейку,

4 смещения (*offsets*) для бокса,

1 *objectness score*,

80 классов объектов.

Predictions Across Scales

YOLOv3 делает предсказания на 3 масштабах:

- 1) На *feature map* последнего сверточного слоя.
- 2) Для двух предпоследних слоев делается *upsampling* x2, и их *feature maps* конкатенируются с *feature maps* более раннего слоя.
- 3) Действие 2 повторяется еще раз.

YOLOv3 используется *k-means* для получения *anchors*.

Девять *anchors* (кластеров) для COCO:

(10×13),(16×30),(33×23),(30×61),(62×45),(59×119),(116×90),(156×198),(373×326).

DarkNet-53 vs. DarkNet-17

DarkNet-53:

Type	Filters	Size	Output
Convolutional	32	3×3	256×256
Convolutional	64	$3 \times 3 / 2$	128×128
1x Convolutional	32	1×1	
1x Convolutional	64	3×3	
Residual			128×128
Convolutional	128	$3 \times 3 / 2$	64×64
Convolutional	64	1×1	
2x Convolutional	128	3×3	
Residual			64×64
Convolutional	256	$3 \times 3 / 2$	32×32
Convolutional	128	1×1	
8x Convolutional	256	3×3	
Residual			32×32
Convolutional	512	$3 \times 3 / 2$	16×16
Convolutional	256	1×1	
8x Convolutional	512	3×3	
Residual			16×16
Convolutional	1024	$3 \times 3 / 2$	8×8
Convolutional	512	1×1	
4x Convolutional	1024	3×3	
Residual			8×8
Avgpool		Global	
Connected		1000	
Softmax			

DarkNet-19:

Type	Filters	Size/Stride	Output
Convolutional	32	3×3	224×224
Maxpool		$2 \times 2 / 2$	112×112
Convolutional	64	3×3	112×112
Maxpool		$2 \times 2 / 2$	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	56×56
Maxpool		$2 \times 2 / 2$	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	28×28
Maxpool		$2 \times 2 / 2$	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		$2 \times 2 / 2$	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	
Softmax			1000

DarkNet-53 vs. DarkNet-17 vs. ResNets

Backbone	Top-1	Top-5	Bn Ops	BFLOP/s	FPS
Darknet-19 [15]	74.1	91.8	7.29	1246	171
ResNet-101[5]	77.1	93.7	19.7	1039	53
ResNet-152 [5]	77.6	93.8	29.4	1090	37
Darknet-53	77.2	93.8	18.7	1457	78

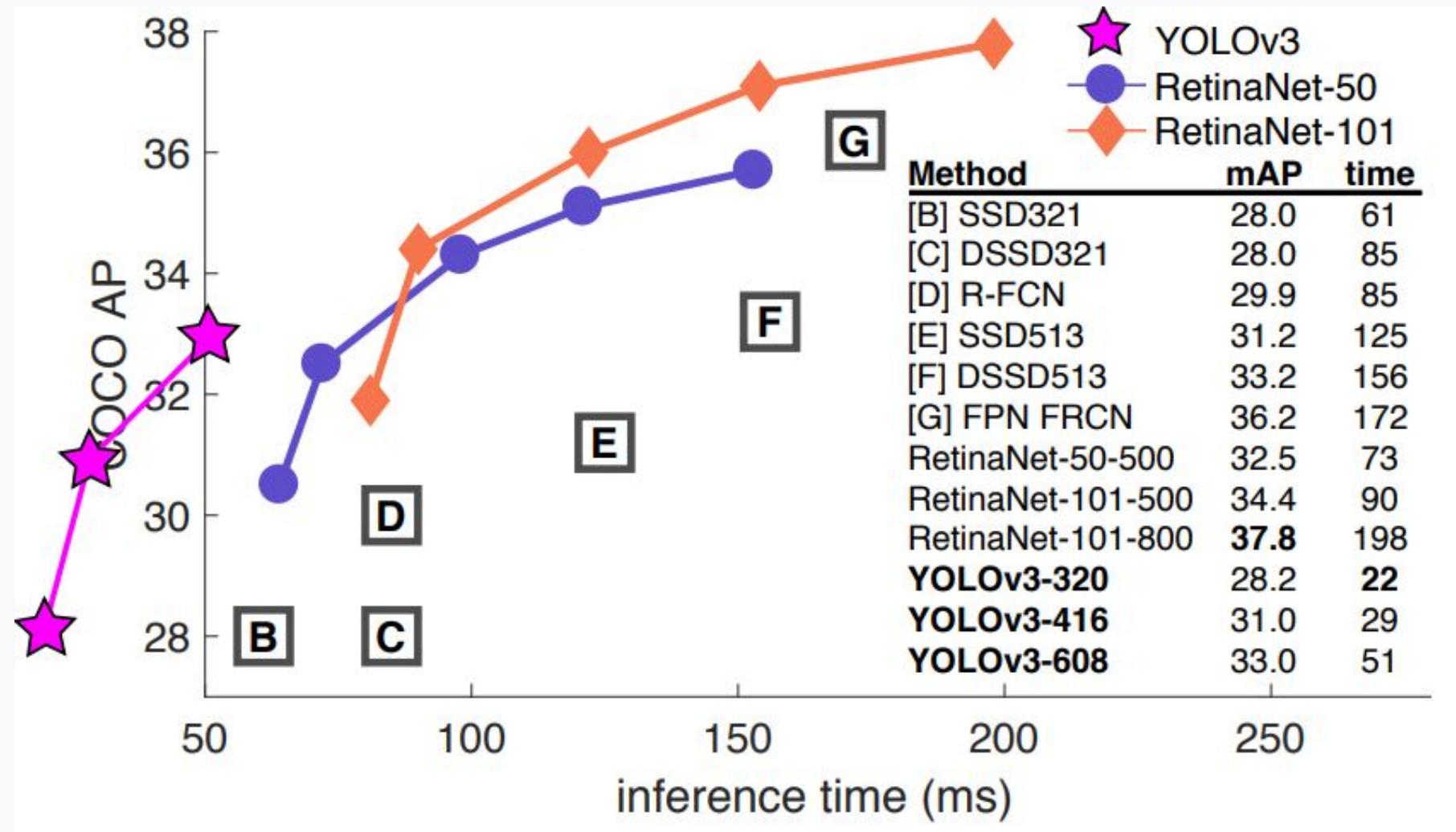
Bn Ops - billions of operations.

BFLOP/s - billion floating point operations per second.

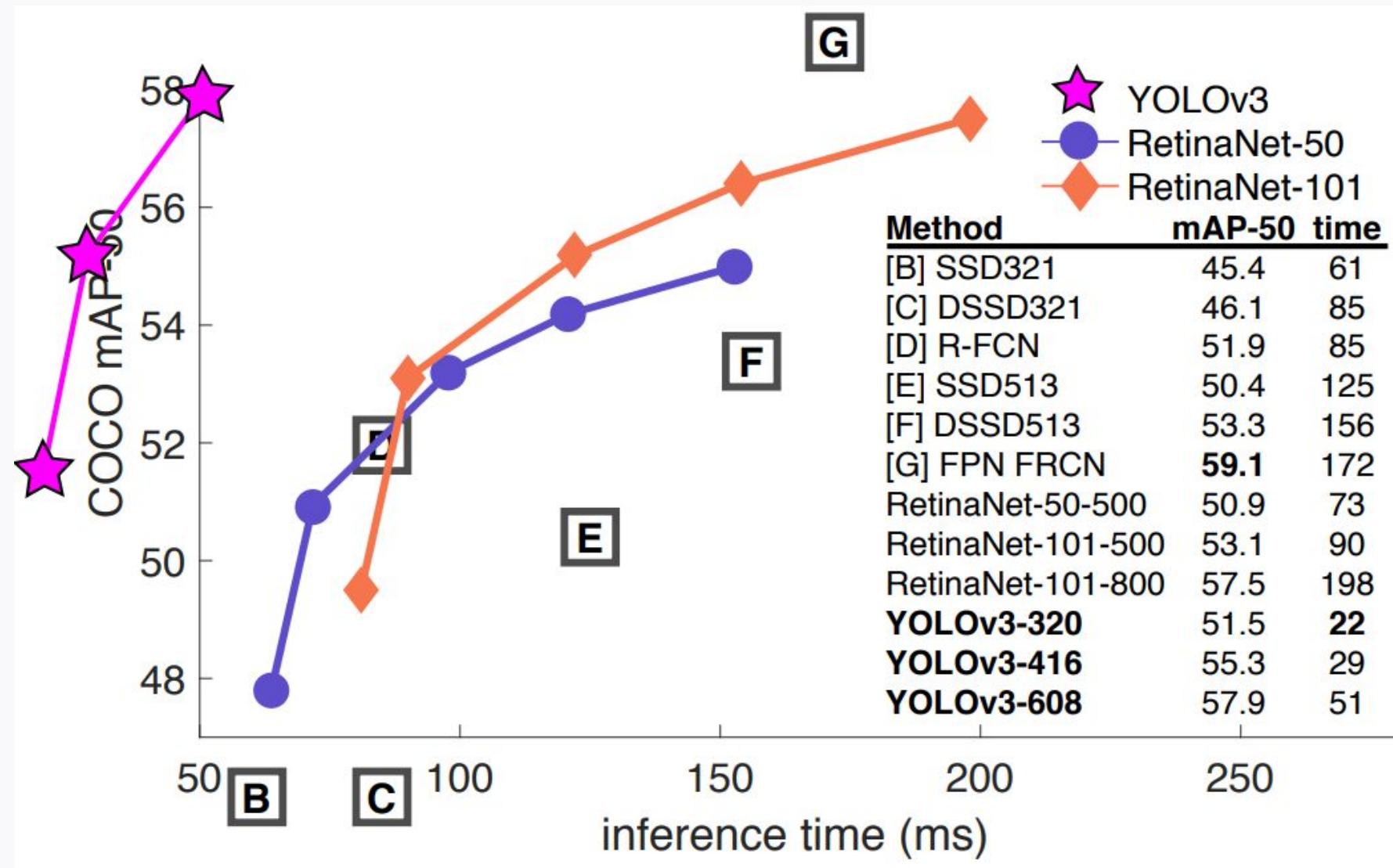
MS COCO AP

	backbone	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
<i>Two-stage methods</i>							
Faster R-CNN+++ [3]	ResNet-101-C4	34.9	55.7	37.4	15.6	38.7	50.9
Faster R-CNN w FPN [6]	ResNet-101-FPN	36.2	59.1	39.0	18.2	39.0	48.2
Faster R-CNN by G-RMI [4]	Inception-ResNet-v2 [19]	34.7	55.5	36.7	13.5	38.1	52.0
Faster R-CNN w TDM [18]	Inception-ResNet-v2-TDM	36.8	57.7	39.2	16.2	39.8	52.1
<i>One-stage methods</i>							
YOLOv2 [13]	DarkNet-19 [13]	21.6	44.0	19.2	5.0	22.4	35.5
SSD513 [9, 2]	ResNet-101-SSD	31.2	50.4	33.3	10.2	34.5	49.8
DSSD513 [2]	ResNet-101-DSSD	33.2	53.3	35.2	13.0	35.4	51.1
RetinaNet [7]	ResNet-101-FPN	39.1	59.1	42.3	21.8	42.7	50.2
RetinaNet [7]	ResNeXt-101-FPN	40.8	61.1	44.1	24.1	44.2	51.2
YOLOv3 608 × 608	Darknet-53	33.0	57.9	34.4	18.3	35.4	41.9

MS COCO AP@IoU=.75



MS COCO AP@IoU=.5



Further fate of YOLO

But maybe a better question is: “What are we going to do with these detectors now that we have them?” A lot of the people doing this research are at Google and Facebook. I guess at least we know the technology is in good hands and definitely won’t be used to harvest your personal information and sell it to.... wait, you’re saying that’s exactly what it will be used for?? Oh.

Well the other people heavily funding vision research are the military and they’ve never done anything horrible like killing lots of people with new technology oh wait.....¹

I have a lot of hope that most of the people using computer vision are just doing happy, good stuff with it, like counting the number of zebras in a national park [13], or tracking their cat as it wanders around their house [19]. But computer vision is already being put to questionable use and as researchers we have a responsibility to at least consider the harm our work might be doing and think of ways to mitigate it. We owe the world that much.

In closing, do not @ me. (Because I finally quit Twitter).

Joseph Redmon



YOLOv3 demo

<https://youtu.be/MPU2HistivI>



LIVE

Подведение итогов

1

Понимаем основные концепции
Object Detection

2

Знаем метрики для оценки качества Object
Detection моделей

3

Понимаем как устроены
R-CNN, Fast R-CNN, Faster R-CNN

4

Знаем YOLO, YOLOv2, YOLOv3

без картинок

0-9



Рефлексия



Какие мысли и идеи вам запомнились из сегодняшнего вебинара?



Что планируете применять в работе из этого?

Список материалов для изучения

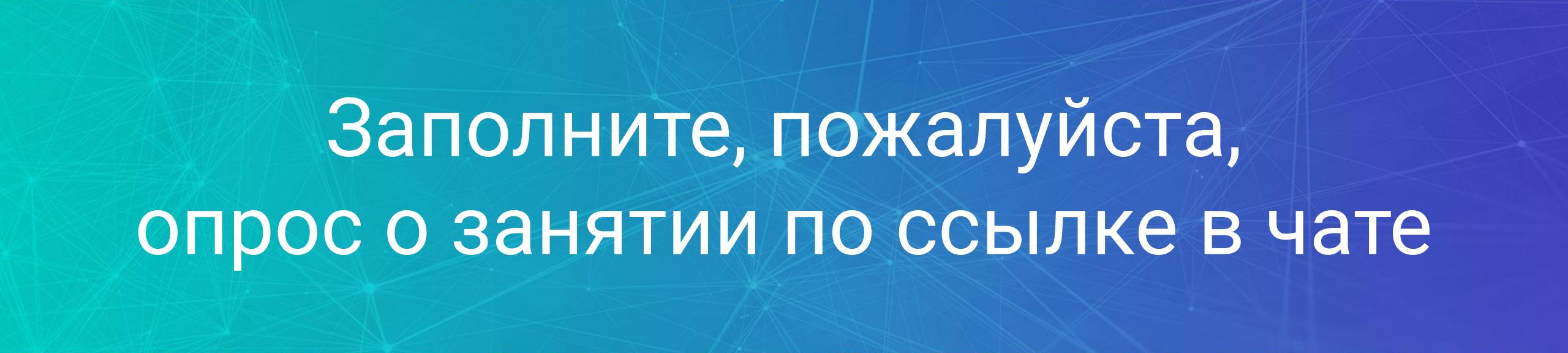
- Evaluation metrics for object detection and segmentation (mAP):
<https://kharshit.github.io/blog/2019/09/20/evaluation-metrics-for-object-detection-and-segmentation>
- mAP (mean Average Precision) for Object Detection:
https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173
- Rapid Object Detection using a Boosted Cascade of Simple Features:
<https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf>
- OpenCV: Cascade Classifier: https://docs.opencv.org/master/db/d28/tutorial_cascade_classifier.html
- Интегральное представление изображения:
<https://xn--90abkegh7abpe7afy.xn--p1ai/digital-vision/integralnoe-predstavlenie-izobrazheniya/>
- OpenCV Face Detection – Visualized: <https://vimeo.com/12774628>
- Selective Search for Object Recognition: <http://www.huppelen.nl/publications/selectiveSearchDraft.pdf>
- Selective Search for Object Detection: <https://arthurdouillard.com/post/selective-search/>
<https://www.learnopencv.com/selective-search-for-object-detection-cpp-python/>

Список материалов для изучения

- R-CNN: <https://arxiv.org/pdf/1311.2524.pdf>
- Fast R-CNN: <https://arxiv.org/pdf/1504.08083.pdf>
- Faster R-CNN: <https://arxiv.org/pdf/1506.01497.pdf>
- Review: Fast R-CNN: <https://medium.com/coinmonks/review-fast-r-cnn-object-detection-a82e172e87ba>
- Detectron2: <https://github.com/facebookresearch/detectron2>
- Feature Pyramid Networks for Object Detection: <https://arxiv.org/pdf/1612.03144.pdf>
- Spatial Pyramid Pooling / RoI Pooling: <https://arxiv.org/abs/1406.4729>
- Selective Search for Object Recognition: <http://www.huppelen.nl/publications/selectiveSearchDraft.pdf>
- Deep Sliding Shapes for Amodal 3D Object Detection: <http://dss.cs.princeton.edu/paper.pdf>

Список материалов для изучения

- YOLO: <https://arxiv.org/abs/1506.02640>
- YOLOv2 / YOLO9000: <https://arxiv.org/abs/1612.08242>
- YOLOv3: <https://arxiv.org/abs/1804.02767>
- Feature Pyramid Networks for Object Detection: <https://arxiv.org/abs/1612.03144>
- YOLO: Real-Time Object Detection: <https://pjreddie.com/darknet/yolo/>
- Real-time Object Detection with YOLO, YOLOv2 and YOLOv3:
https://medium.com/@jonathan_hui/real-time-object-detection-with-yolo-yolov2-28b1b93e2088
- Understanding Categorical Cross-Entropy Loss: https://gombru.github.io/2018/05/23/cross_entropy_loss/
- darknet github: <https://github.com/pjreddie/darknet>
- ultralytics/yolov3: <https://github.com/ultralytics/yolov3>



Заполните, пожалуйста,
опрос о занятии по ссылке в чате

Спасибо за внимание!
Приходите на следующие вебинары



Витвицкий Антон

Head of Computer Vision

Boost Technology Inc, CA

darkangel-nwo@ya.ru