

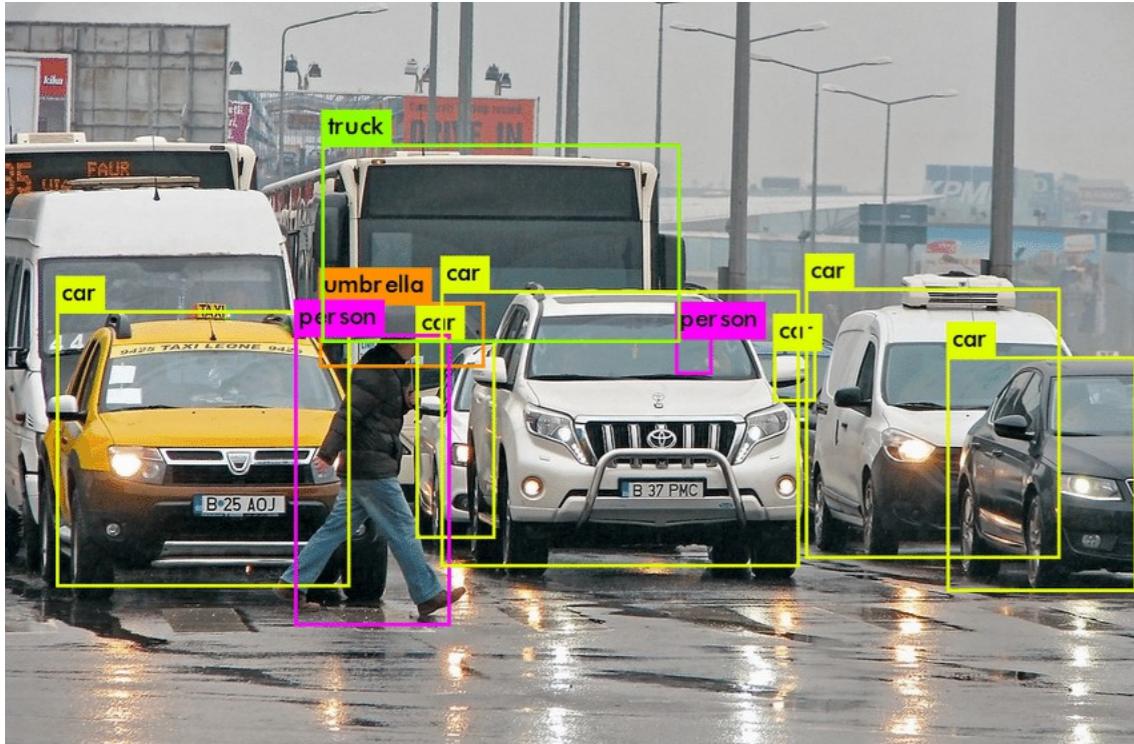
# Машинное обучение

## DS-поток



# План лекции

1. Детектирование объектов
2. Одностадийные детекторы
3. Двухстадийные детекторы
4. Объектная / Инстанс сегментация
5. Оценка ключевых точек



# Детектирование объектов



# Задача детектирования объектов на изображениях

**Задача:**

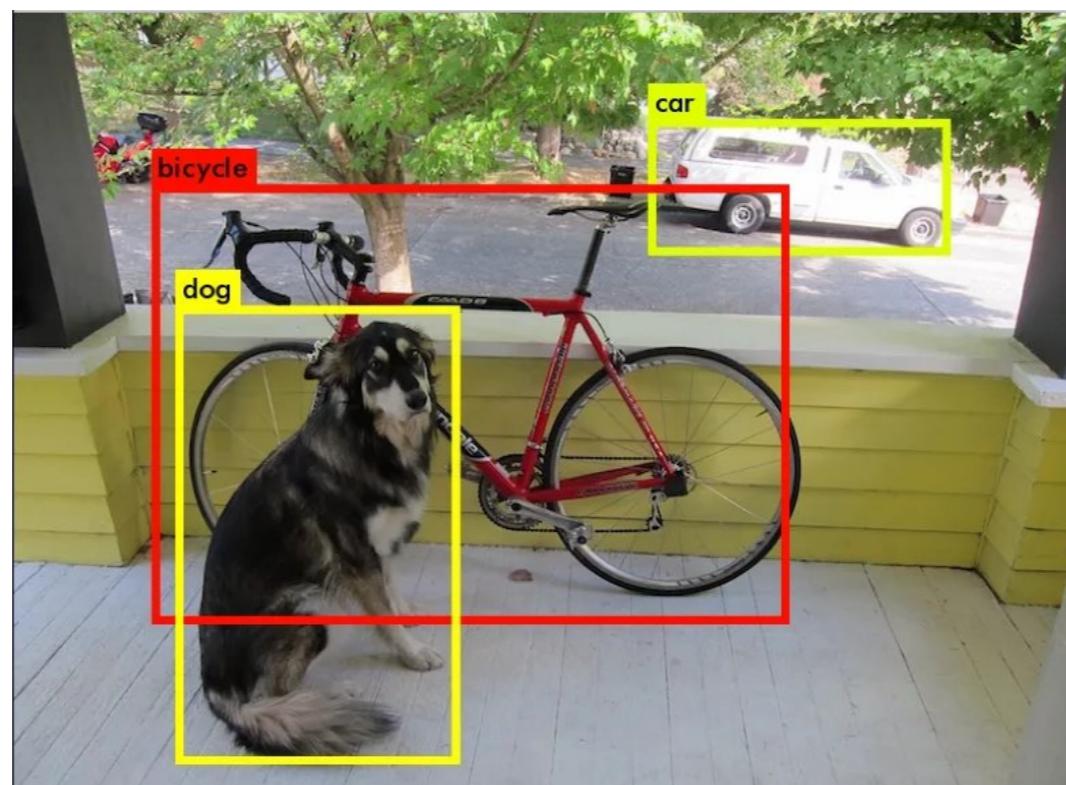
Пусть  $\mathbb{X}$  - пространство картинок.

$\mathbb{Y}$  - набор классов, например: {собака, велосипед, автомобиль}.

$\mathbb{Z}$  - пространство векторов из пятерок

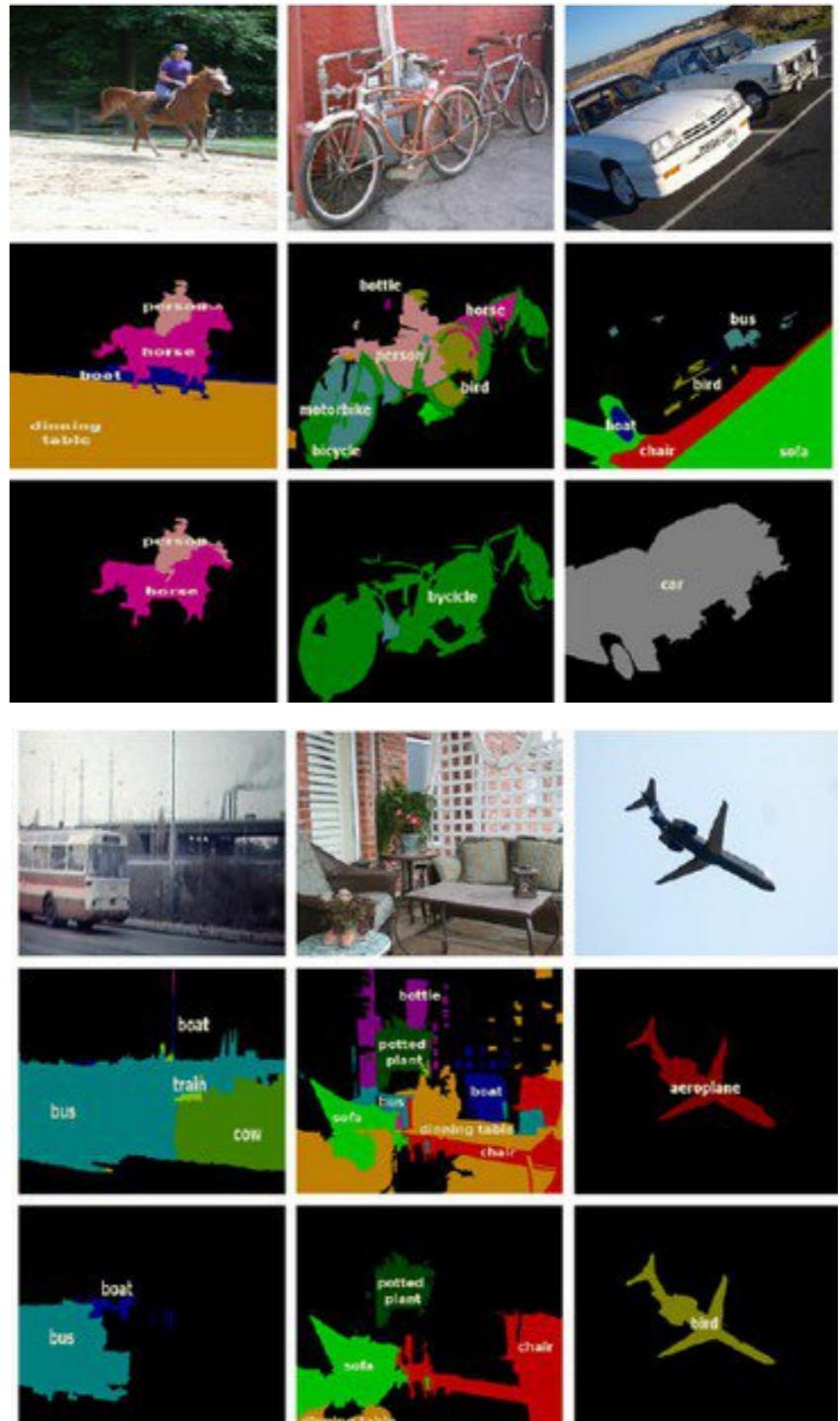
$$[(x_{11}, x_{12}, x_{13}, x_{14}, y_1), \dots, (x_{k1}, x_{k2}, x_{k3}, x_{k4}, y_k)] : x_{j1:4} \in \mathbb{R}, y_j \in \mathbb{Y}$$

Требуется построить модель  $f: \mathbb{X} \rightarrow \mathbb{Z}$ , определяющую координаты всех объектов из  $\mathbb{Y}$ , присутствующих на изображении.



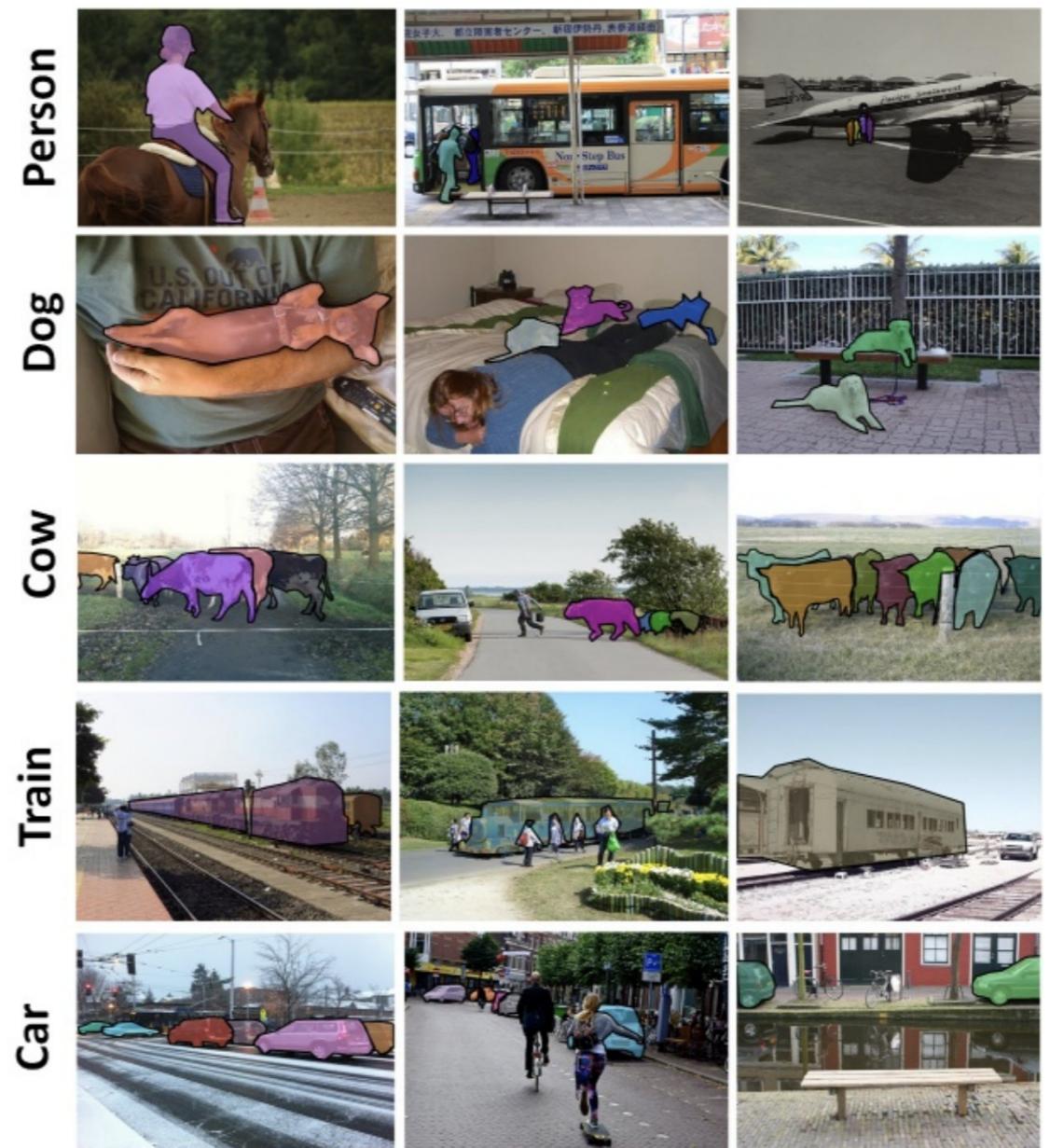
# Данные: PASCAL VOC

- PASCAL Visual Object Classes (2005)
- Боксы и маски объектов
- Классов: 20
- Train + Val: ~11k картинок, ~27k боксов
- Картинки варьируются по размеру и качеству



# Данные: MS COCO

- Microsoft Common Objects in COntext (2015)
- Классов: 80 (для детекции)
- Train: ~120k картинок (18GB)
- Val: ~5k картинок (1GB)
- Test: ~40k картинок (6GB)
- Картинки варьируются по размеру и качеству



# Данные: Google Open Images (v6)

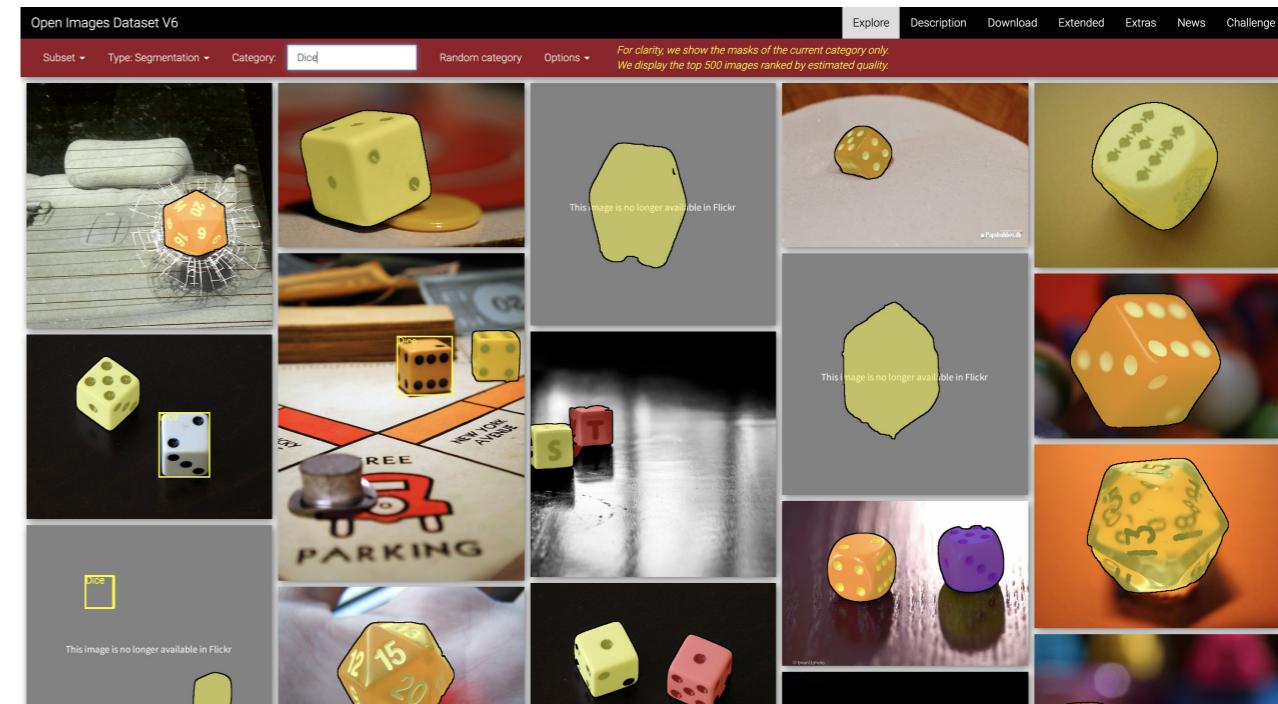
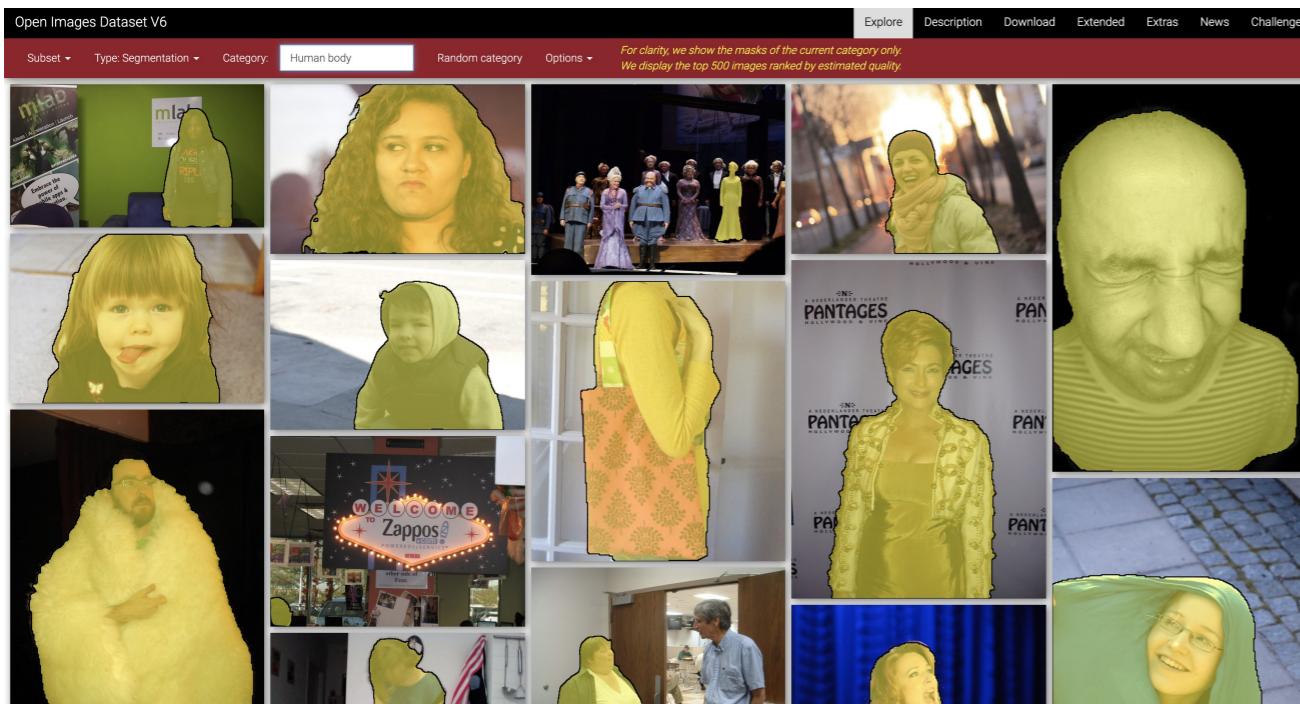


Table 2: Boxes.

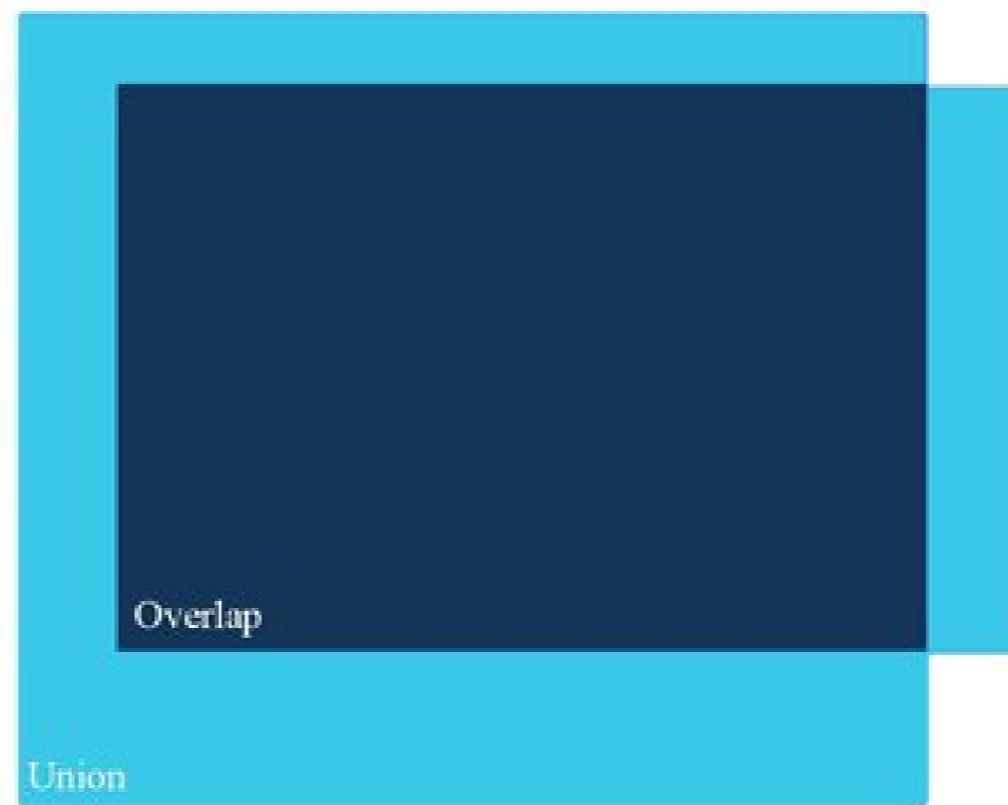
	Train	Validation	Test	# Classes
Images	1,743,042	41,620	125,436	-
Boxes	14,610,229	303,980	937,327	600

# Метрика качества: Intersection over Union



- Ground truth
- Prediction

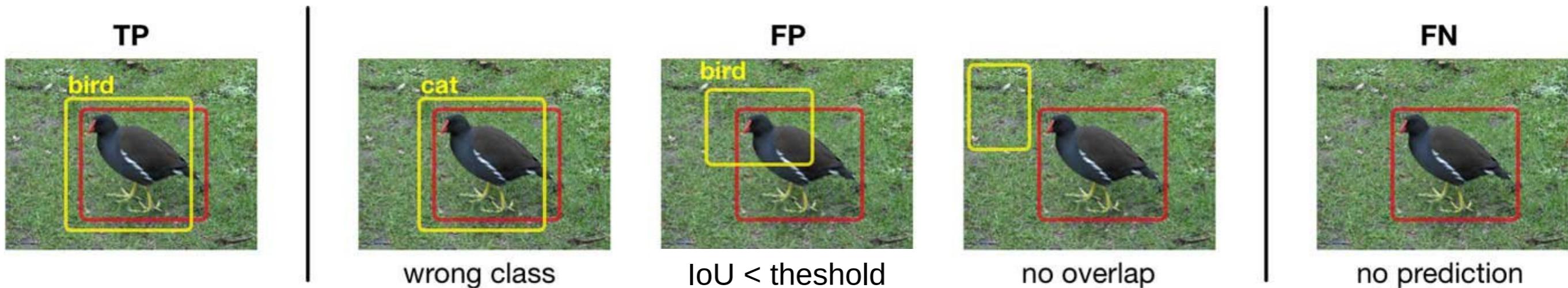
$$IoU = \frac{\text{area of overlap}}{\text{area of union}}$$



# Positives and Negatives

Пример вычисления ТР, FP и FN для оценки детекции

Установим порог порог (threshold) по IoU.



- **TP** — пересечение предсказания и разметки больше порога для объекта заданного класса.
- **FP** — предсказание есть, но либо перепутан класс, либо пересечение мало.
- **FN** — нет предсказания.

# Метрики качества: Precision, Recall

$$\begin{aligned}\textbf{Precision} &= \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Positives}} \\ \textbf{Recall} &= \frac{\textit{True Positives}}{\textit{True Positives} + \textit{False Negatives}}\end{aligned}$$

# Метрики качества: Mean Average Precision

Всего 5 яблок. Для них получилось 10 предсказаний.

Установили порог IoU = 0.5. По нему определили корректность предсказания.

Отсортируем предсказания для них по оценке уверенности в них.

Для первых k предсказаний в строчке k будем считать precision и recall.

Rank	Correct?	Precision	Recall
1	True		
2	True		
3	False		
4	False		
5	False	?	?
6	True		
7	True		
8	False		
9	False		
10	True		

# Метрики качества: Mean Average Precision

$k = 3$ :  $TP = 2$ ,  $FP = 1$ ,  $FN = 5$  -  $TP = 3$

$Precision = TP / (TP + FP) = 2/3 = 0.67$

$Recall = TP / (TP + FN) = 2 / 5 = 0.4$

Rank	Correct?	Precision	Recall
1	True		
2	True		
3	False		
4	False		
5	False		
6	True		
7	True		
8	False		
9	False		
10	True		



?

?

# Метрики качества: Mean Average Precision

$k = 3$ :  $TP = 2$ ,  $FP = 1$ ,  $FN = 5 - TP = 3$

$Precision = TP / (TP + FP) = 2/3 = 0.67$

$Recall = TP / (TP + FN) = 2 / 5 = 0.4$



Rank	Correct?	Precision	Recall
1	True	1.0	0.2
2	True	1.0	0.4
3	False	0.67	0.4
4	False	0.5	0.4
5	False	0.4	0.4
6	True	0.5	0.6
7	True	0.57	0.8
8	False	0.5	0.8
9	False	0.44	0.8
10	True	0.5	1.0

# Метрики качества: Mean Average Precision

**Recall** не убывает всегда.

**Precision** в среднем убывает, но с некоторыми скачками.

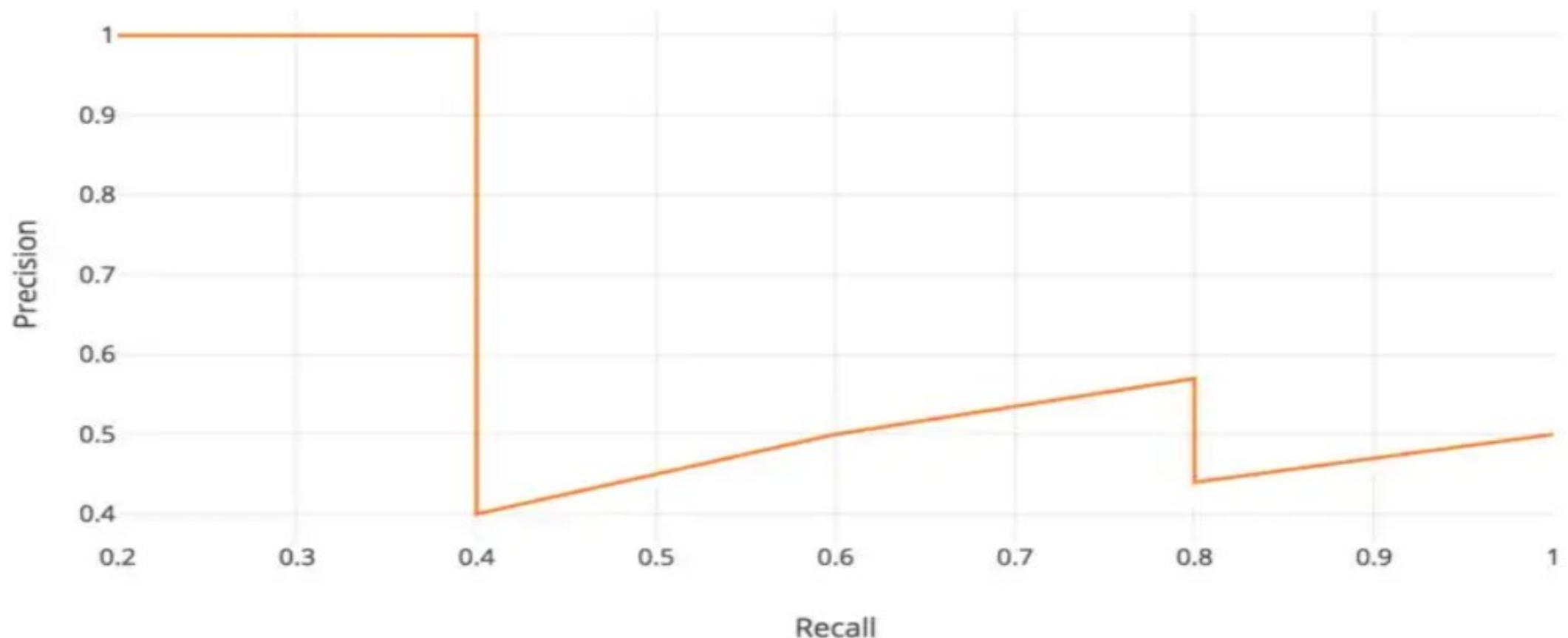


Rank	Correct?	Precision	Recall
1	True	1.0 ↑	0.2 ↑
2	True	1.0 –	0.4 ↑
3	False	0.67 ↓	0.4 –
4	False	0.5 ↓	0.4 –
5	False	0.4 ↓	0.4 –
6	True	0.5 ↑	0.6 ↑
7	True	0.57 ↑	0.8 ↑

# Метрики качества: Mean Average Precision

**Recall** не убывает всегда

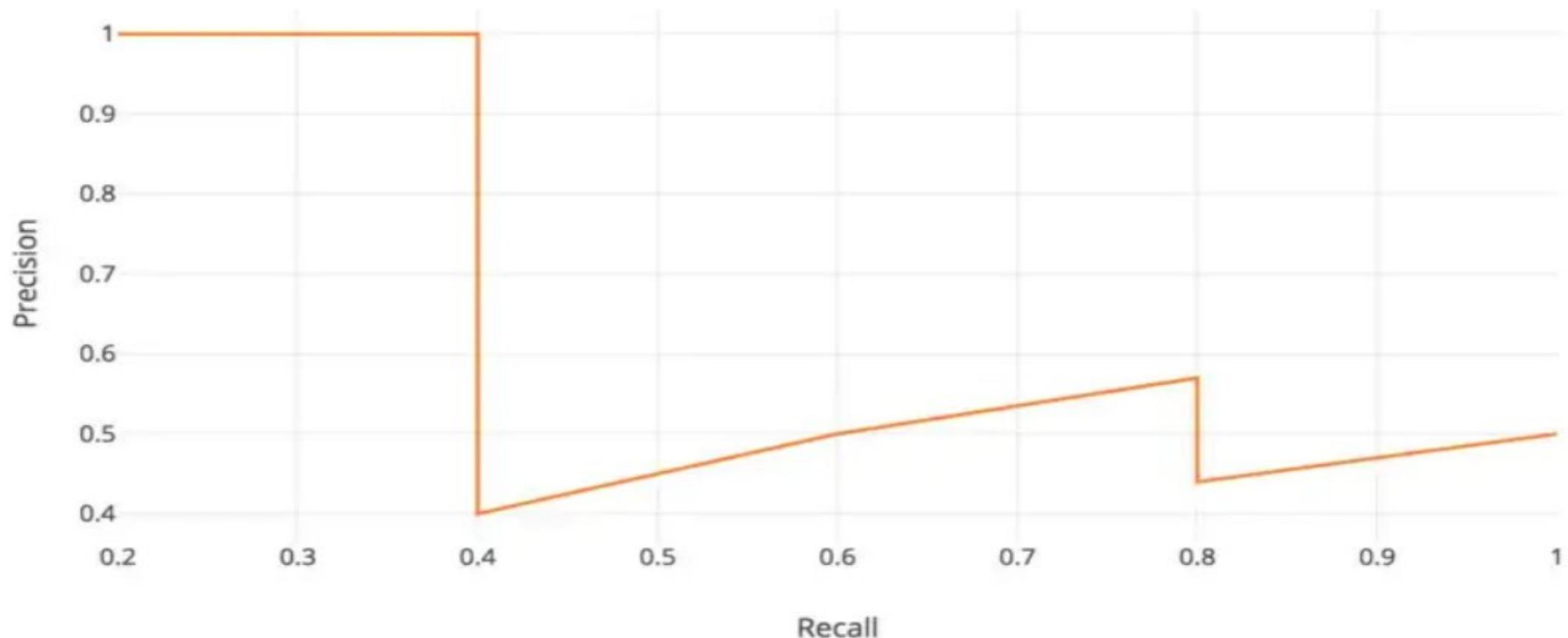
**Precision** в среднем убывает, но с некоторыми скачками



# Метрики качества: Mean Average Precision

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

**Mean Average Precision**  
усредняем значения precision для  
каждого значения recall.

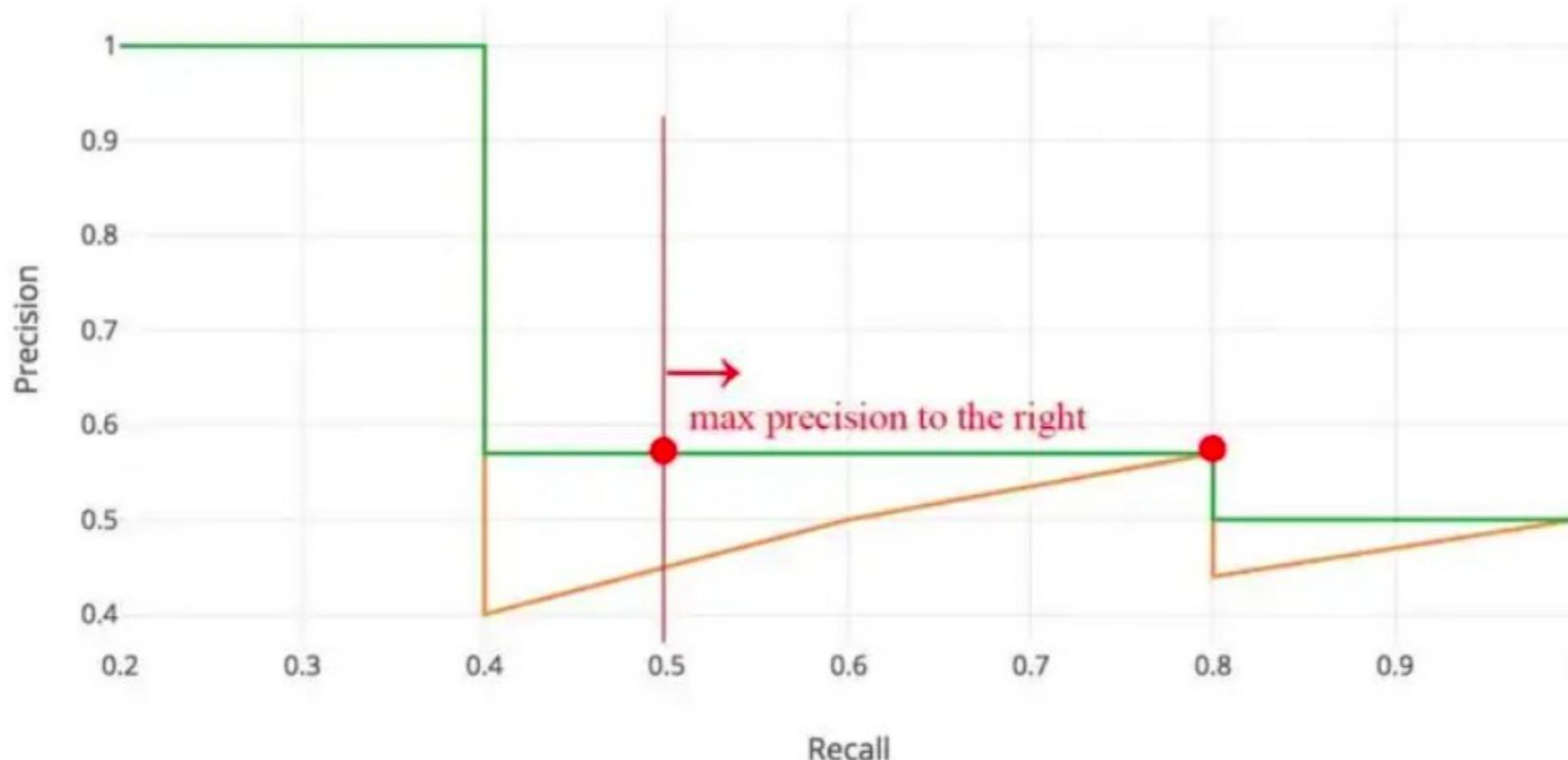


# Метрики качества: Mean Average Precision

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

## Mean Average Precision

Заменим  $p(r)$  на максимальный precision для всех  $r$  больших данного recall.



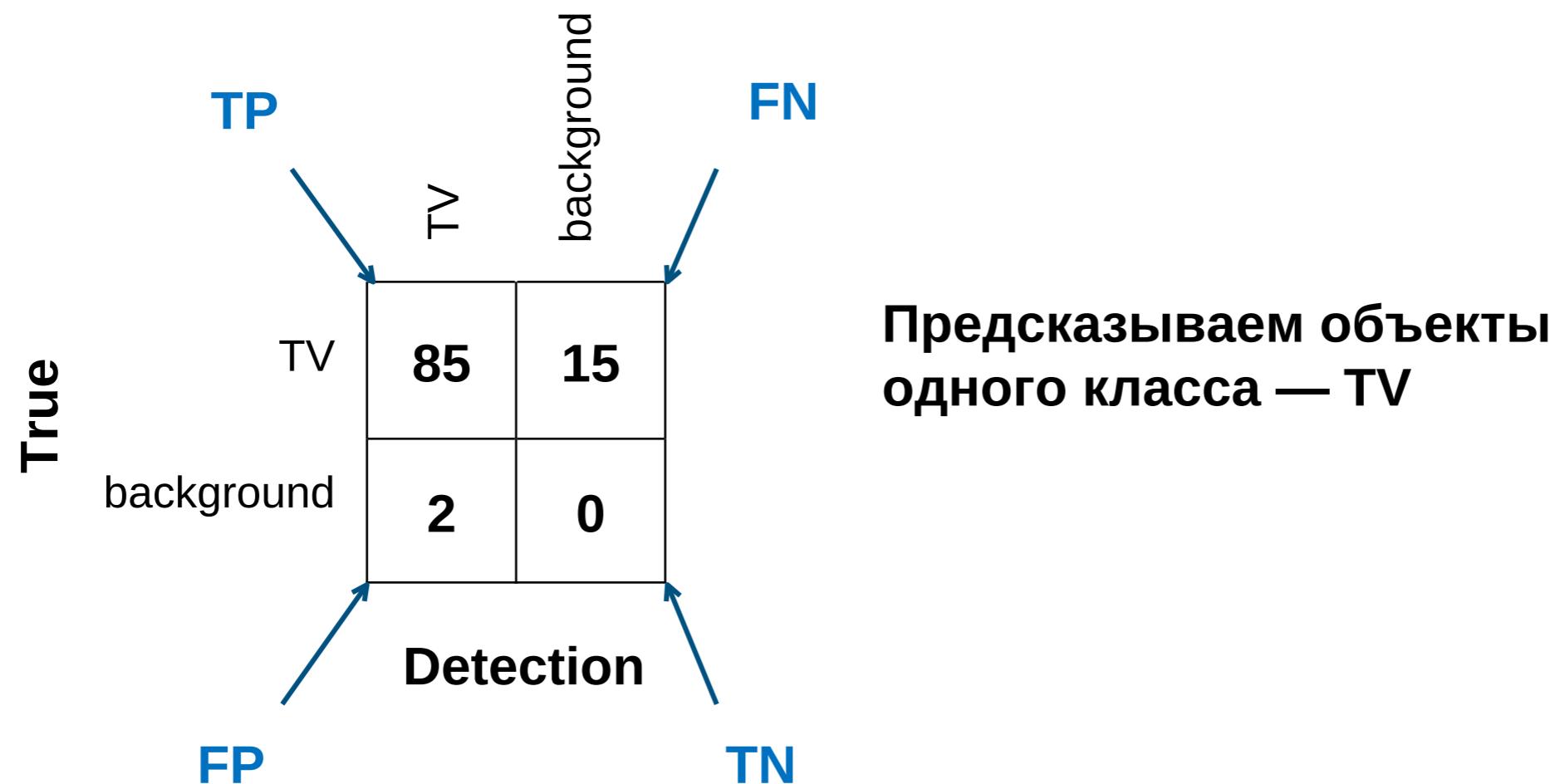
# Метрики качества: Confusion matrix

		TV	background
True	TV	85	15
	background	2	0
Detection			

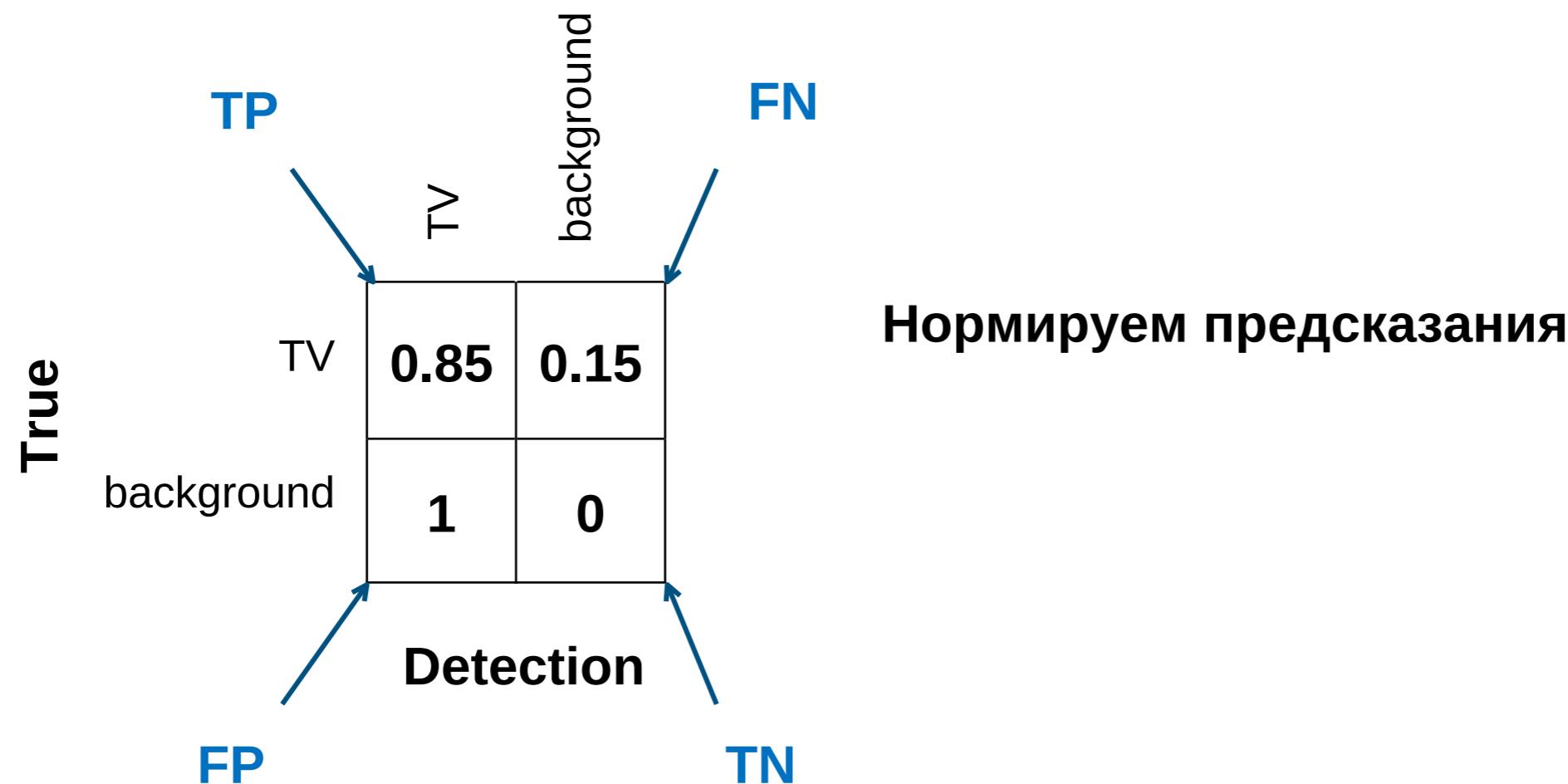
**Предсказываем объекты  
одного класса — TV**

Где TP, FP, FN и TN?

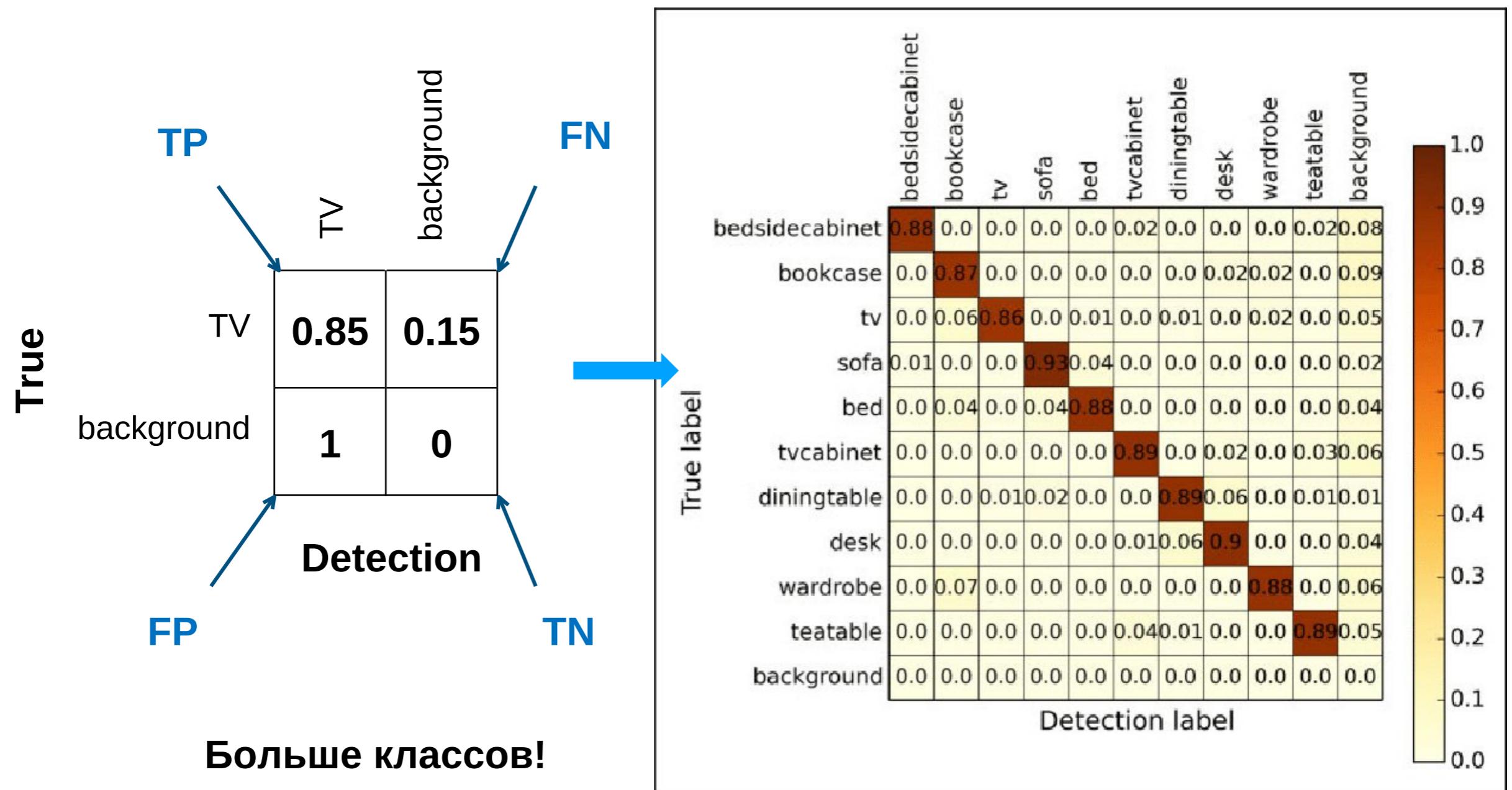
# Метрики качества: Confusion matrix



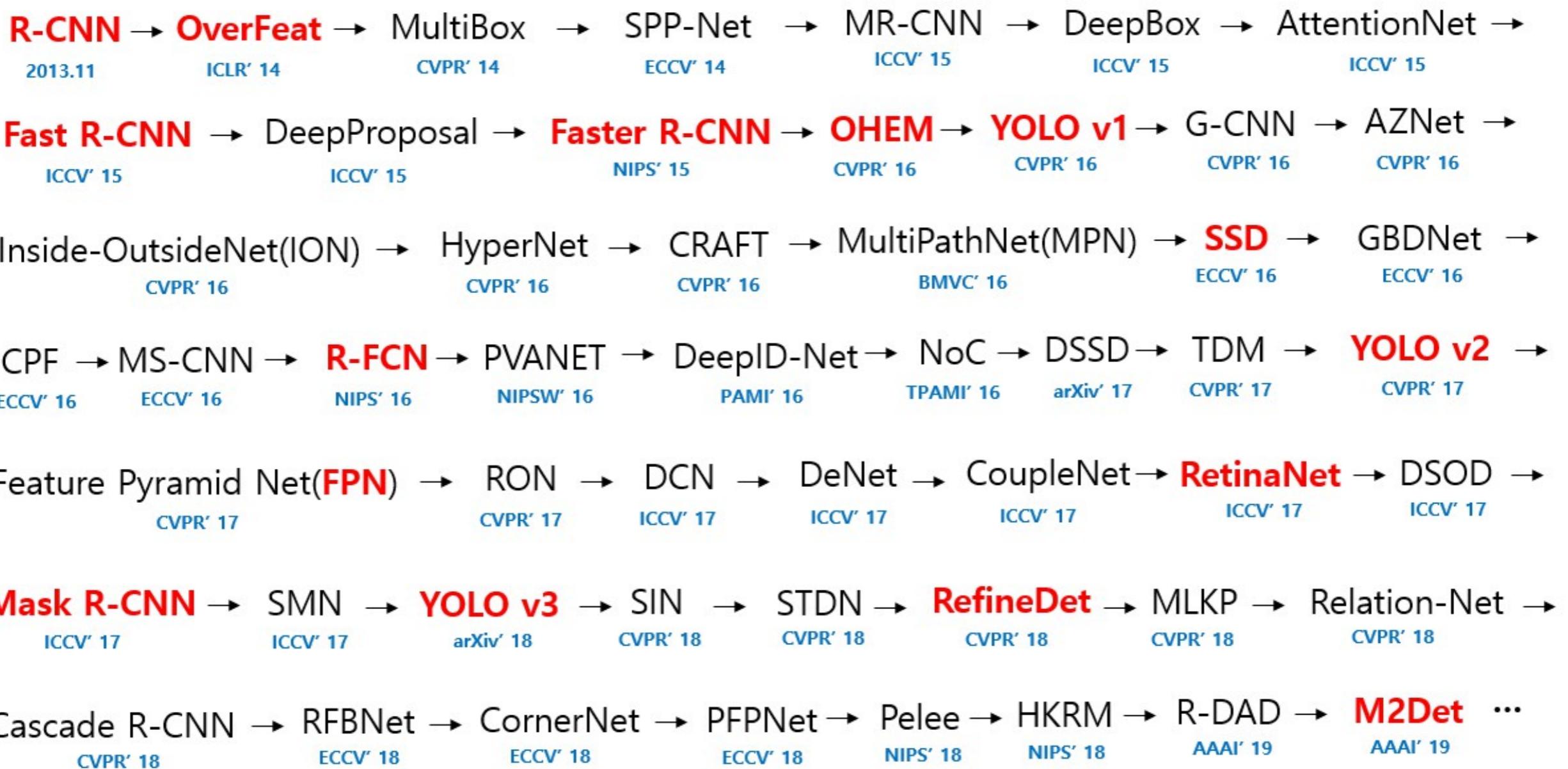
# Метрики качества: Confusion matrix



# Метрики качества: Confusion matrix

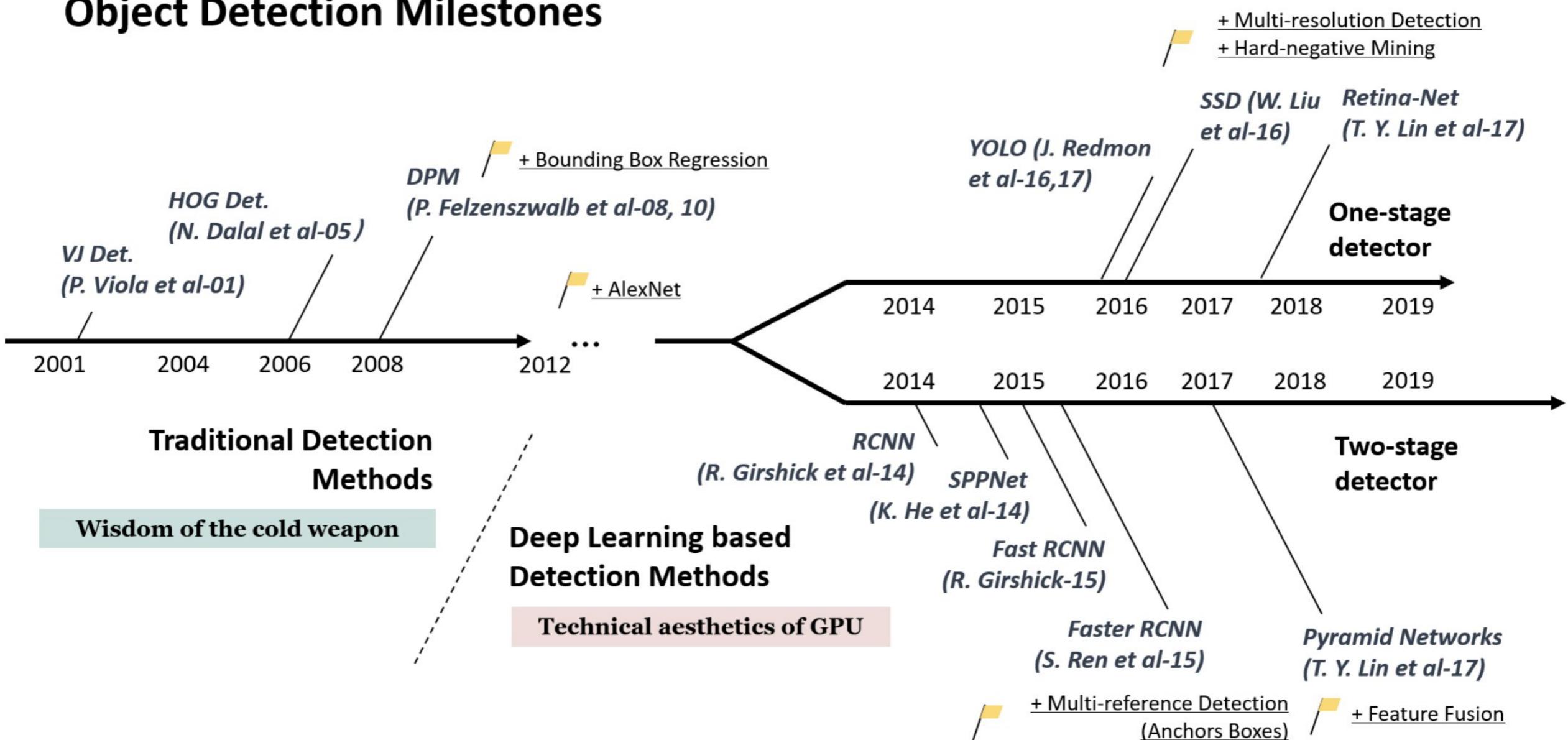


# Модели детектирования объектов



# Модели детектирования объектов

## Object Detection Milestones



# Модели детектирования объектов

## Двухстадийные

1. RCNN and SPPNet (2014)
2. Fast RCNN and Faster RCNN (2015)
3. Mask R-CNN (2017)
4. Pyramid Networks/FPN (2017)
5. G-RCNN (2021)

## Одностадийные

1. YOLO (2016)
2. SSD (2016)
3. RetinaNet (2017)
4. YOLOv3 (2018)
5. YOLOv4 (2020)
6. YOLOR (2021)
7. YOLOv7 (2022)

# Модели детектирования объектов

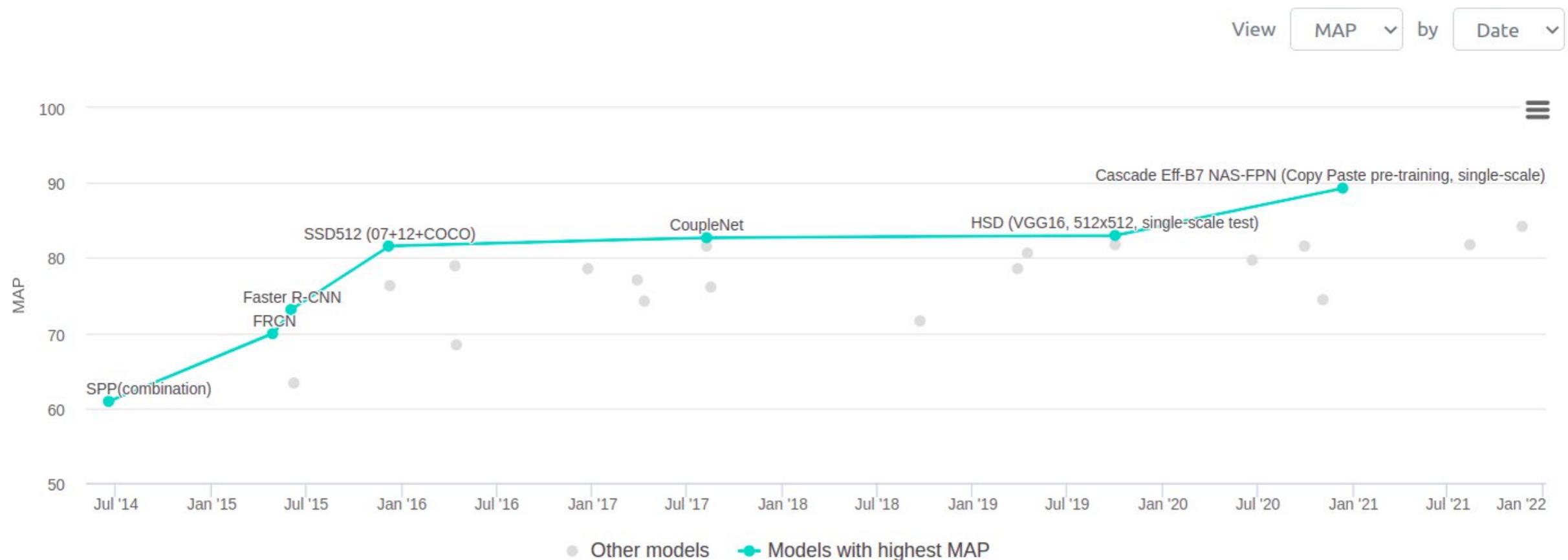
## Object Detection on COCO test-dev



[Papers with code](#)

# Модели детектирования объектов

## Object Detection on PASCAL VOC 2007



[Papers with code](#)

# Детекция: простой подход

- Будем фиксированным окном проходить по всему изображению и предсказывать вероятность того, что в этом окне есть определенный объект.
- Можем делать предсказания для всех объектов сразу.
- Можем варьировать параметры окна — будем перебирать окна различных размеров.

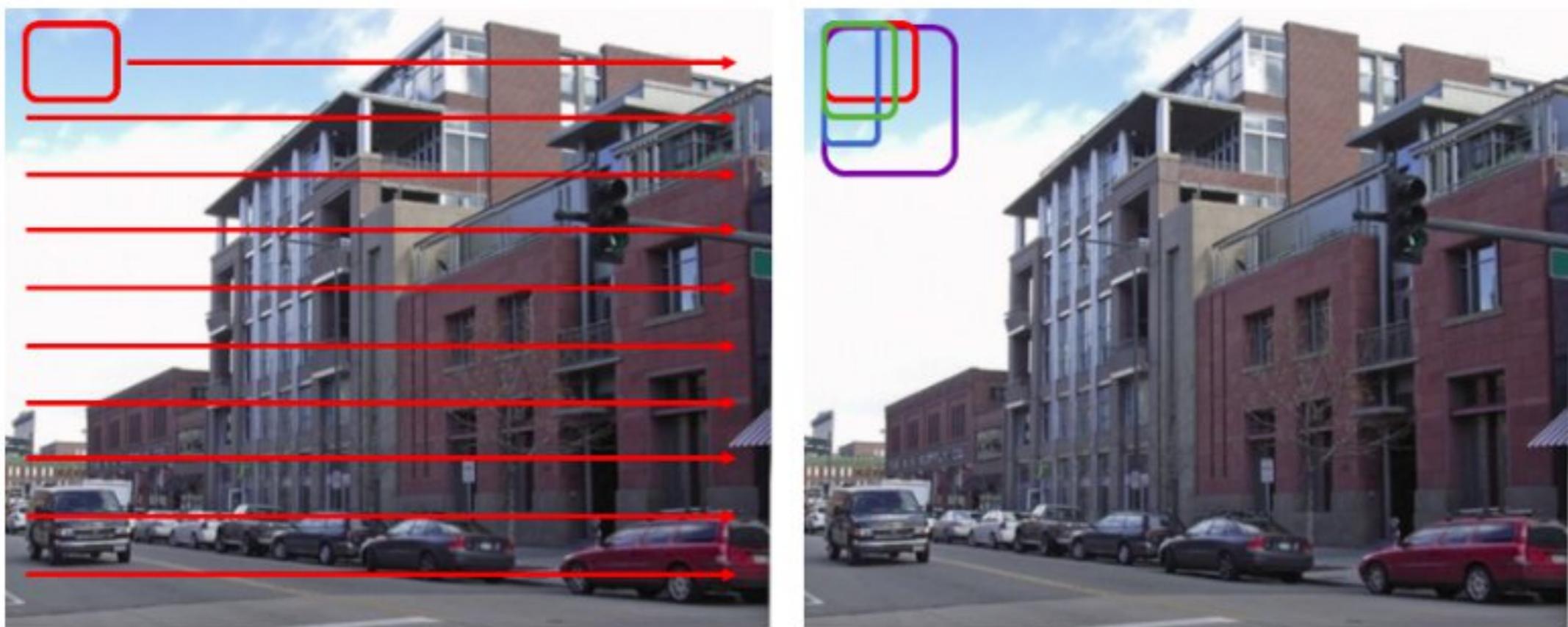


Illustration of Sliding Window (Left) with Different Aspect Ratios and Sizes (Right)

# Детекция: простой подход

- Будем фиксированным окном проходить по всему изображению и предсказывать вероятность того, что в этом окне есть определенный объект.
- Можем делать предсказания для всех объектов сразу.
- Можем варьировать параметры окна — будем перебирать окна различных размеров.

## Проблема

Вычислительно затратно!

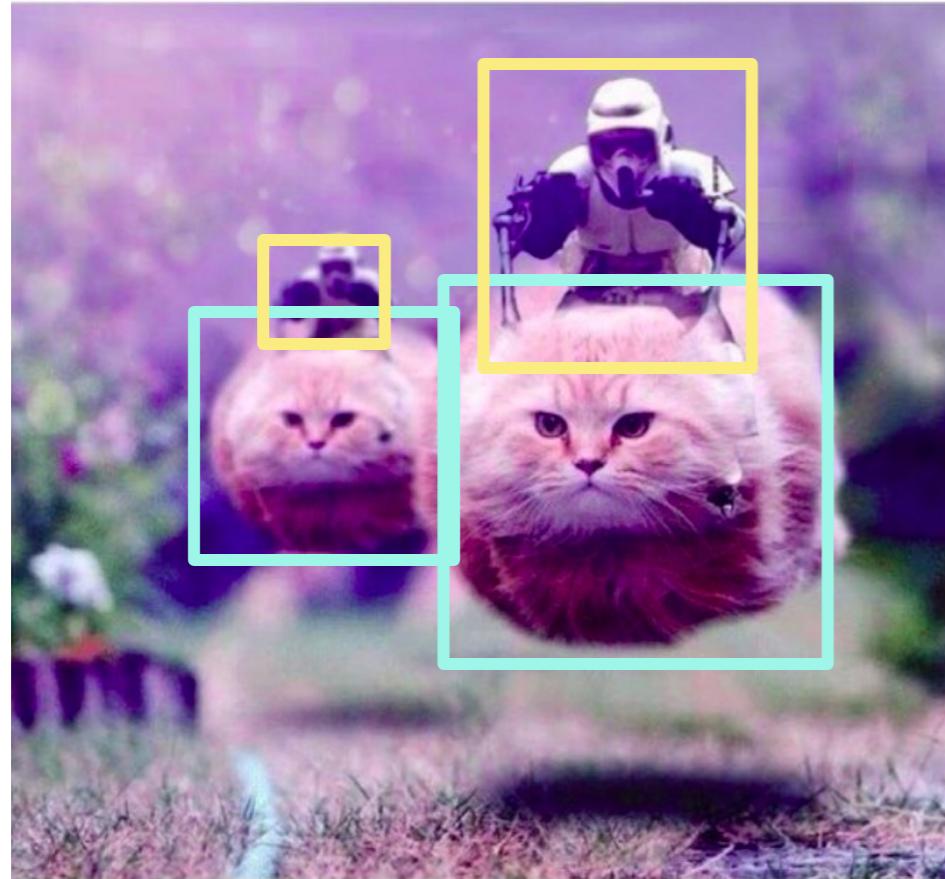
Нужно будет делать очень много проходов по изображению.



Illustration of Sliding Window (Left) with Different Aspect Ratios and Sizes (Right)

# Одностадийные детекторы

# Одностадийные детекторы



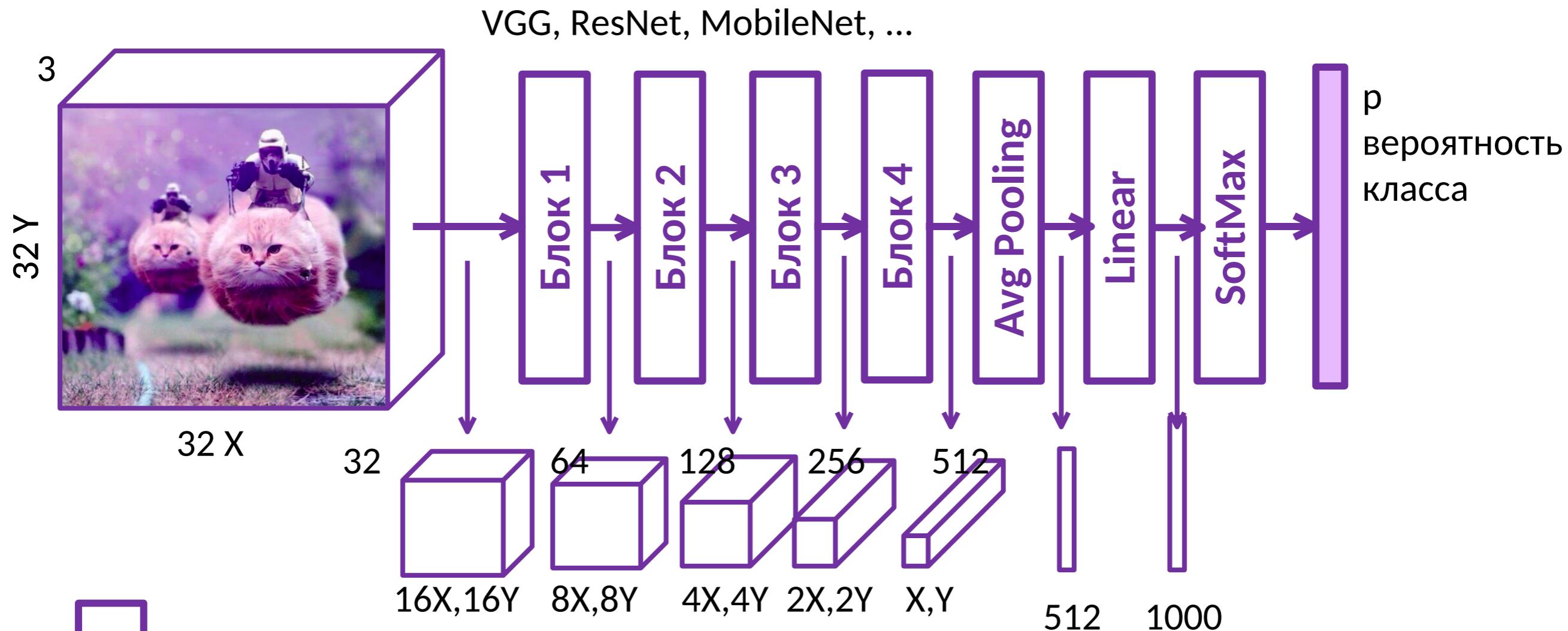
## Задача

Хотим научиться обнаруживать котов и штурмовики.

А именно мы хотим найти координаты и размеры **bouding box**-в для всех котов и штурмовиков на картинке.

# Классификация

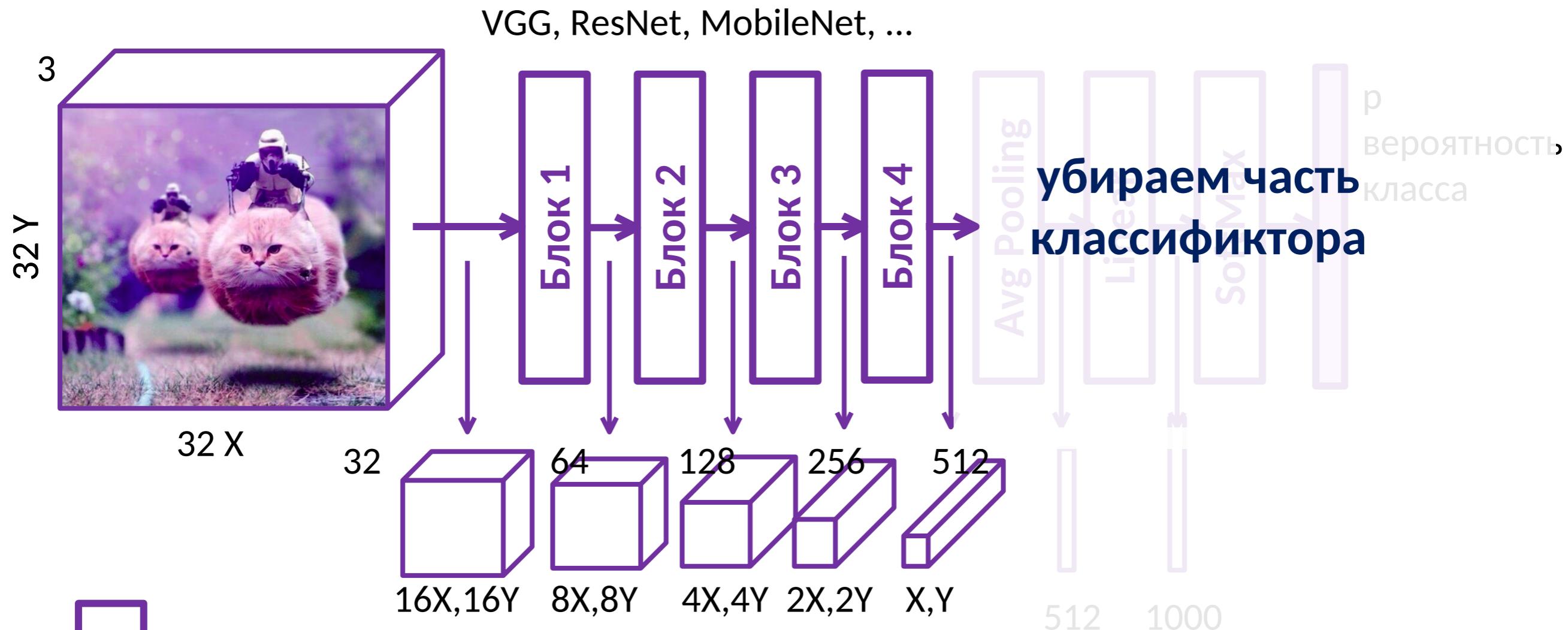
Вспомним задачу классификации: рассмотрим стандартный классификатор.



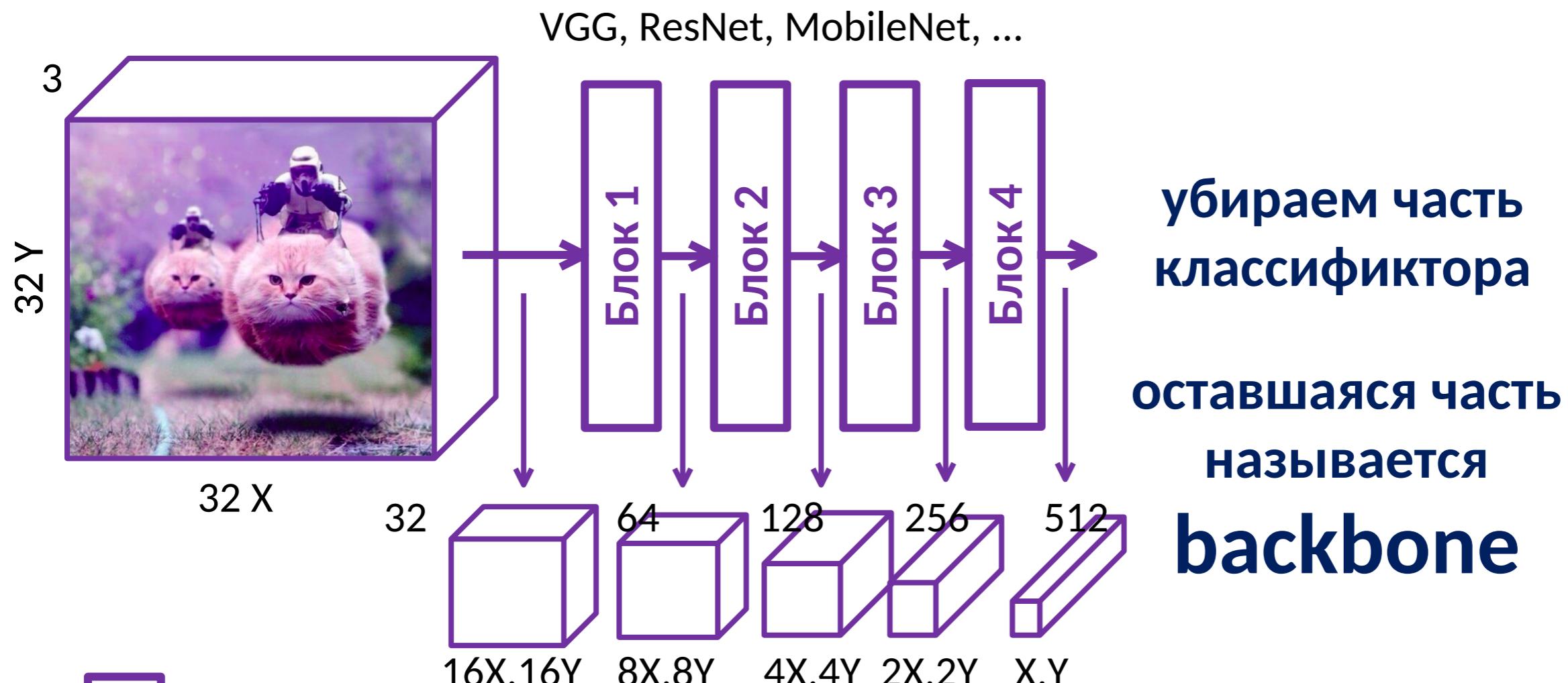
Свертка, нормализация,  
активация, пулинг ...  
Зависит от конкретной  
backbone модели.

# Классификация

Вспомним задачу классификации: рассмотрим стандартный классификатор.

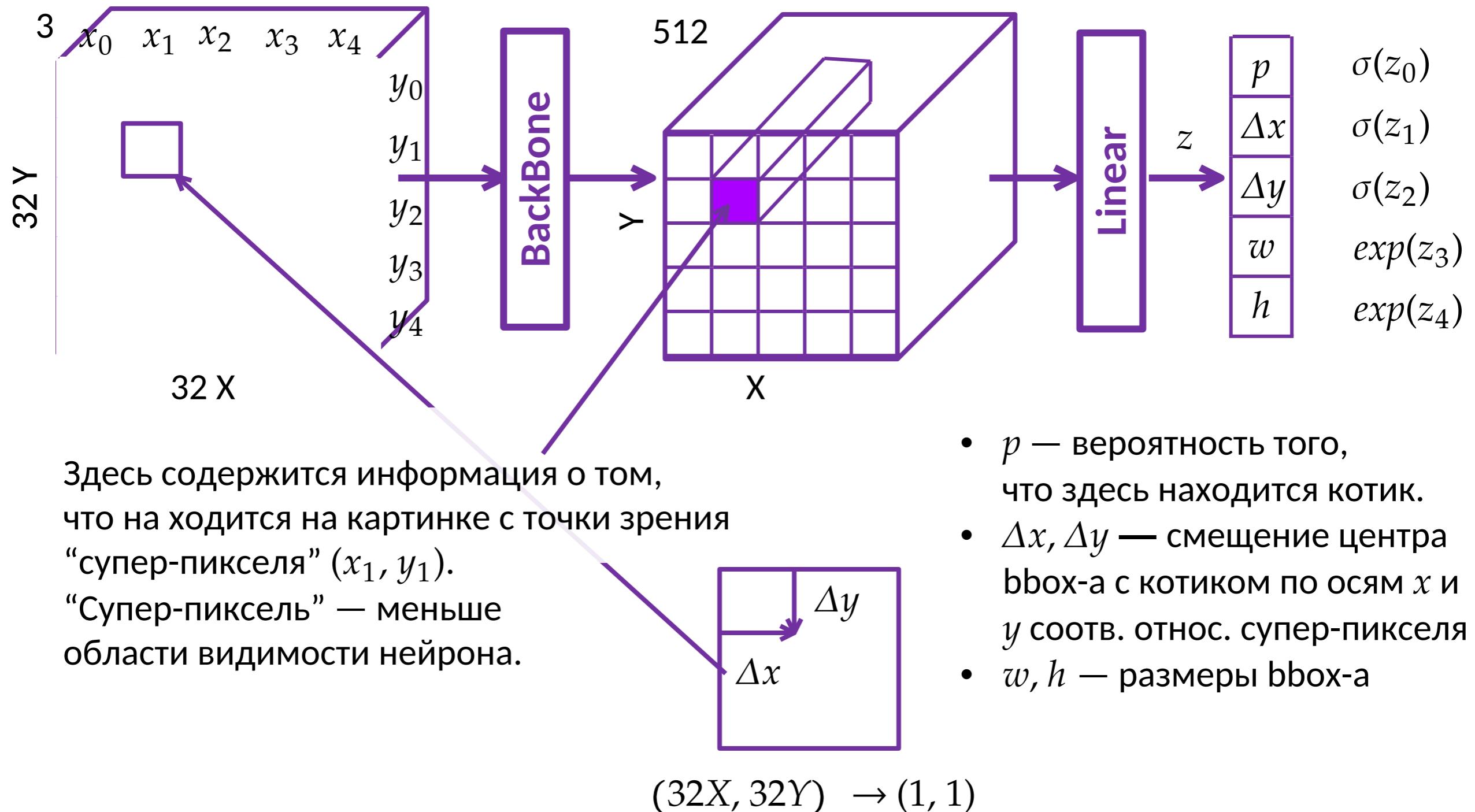


# BackBone-модель



# Модель детектора

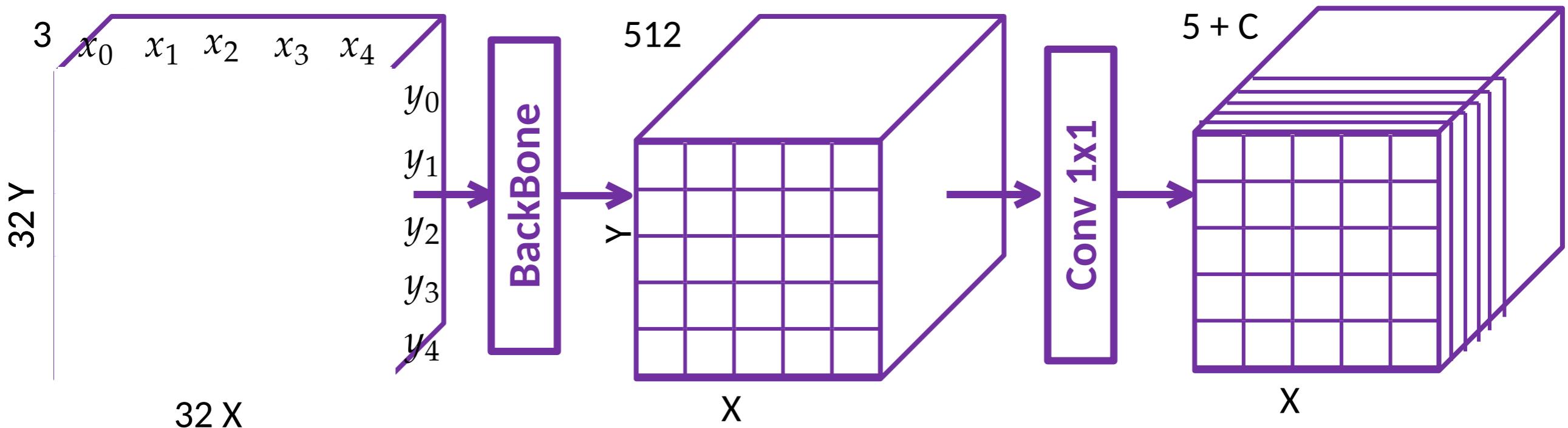
Будем предсказывать расположение котика в супер-пикселе



- $p$  — вероятность того, что здесь находится котик.
- $\Delta x, \Delta y$  — смещение центра bbox-а с котиком по осям  $x$  и  $y$  соотв. относ. супер-пикселя
- $w, h$  — размеры bbox-а

# Модель детектора: YOLO

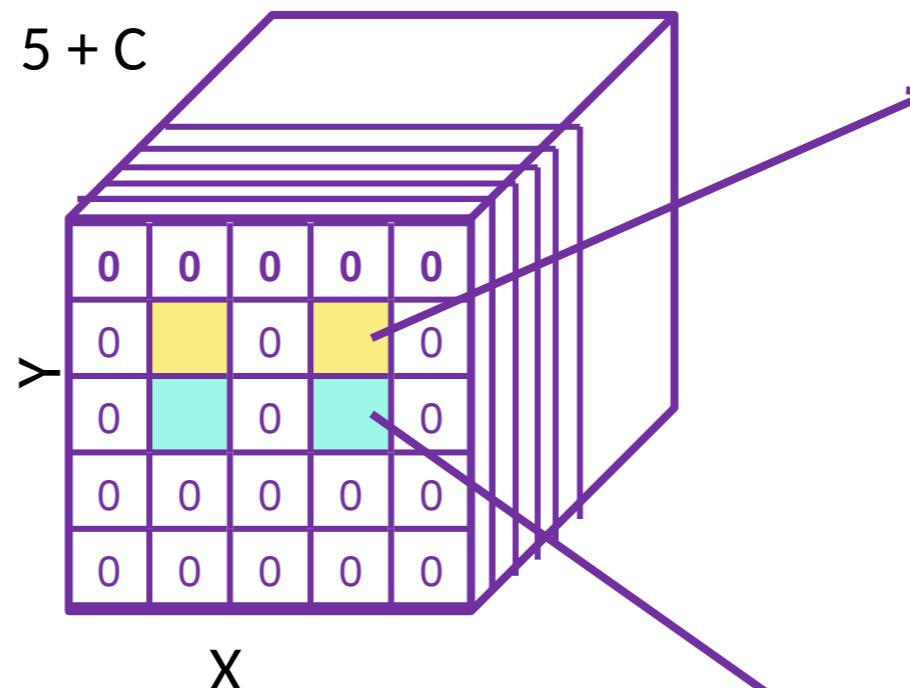
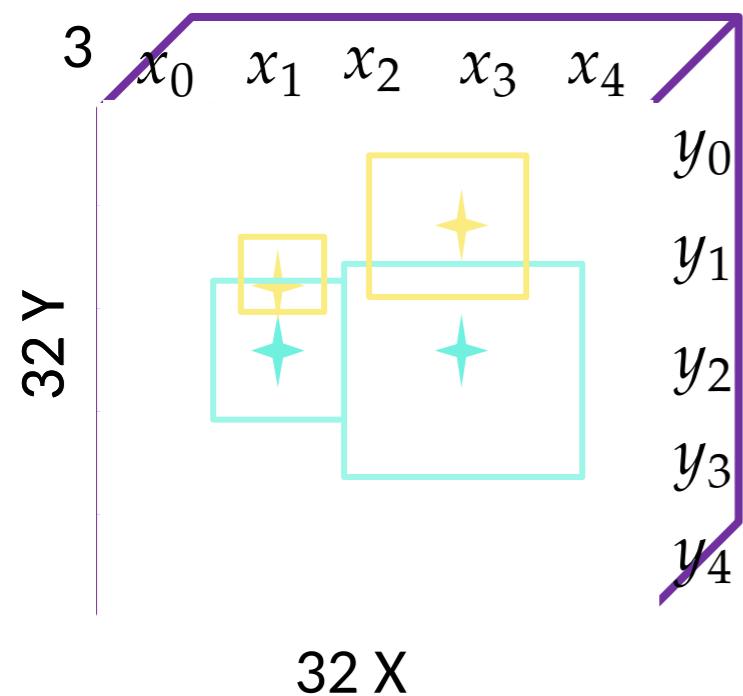
Будем предсказывать расположение объектов из C классов по всему изображению.



- Канал №0  $\rightarrow \sigma \rightarrow p$  — вероятность того, что здесь находится один из нужных объектов.
- Каналы №1, 2  $\rightarrow \sigma \rightarrow \Delta x, \Delta y$  — смещение центра bbox-а с по осям  $x$  и  $y$  соотв. относ. супер-пикселя.
- Каналы №3, 4  $\rightarrow \exp \rightarrow w, h$  — размеры bbox-а.
- Каналы №6, ...  $\rightarrow \text{SoftMax} \rightarrow p_c$  — вероятность класса  $c$ .

# Модель детектора. Обучение

Как подготовить данные?



$I = 1$  – индикатор наличия объекта

$$\Delta x' = 0.25$$

$$\Delta y' = 0.15$$

$$\log w' = \log 1.7$$

$$\log h' = \log 1.6$$

$$c' = 0$$

$I = 1$  – индикатор наличия объекта

$$\Delta x' = 0.25$$

$$\Delta y' = 0.5$$

$$\log w' = \log 2.5$$

$$\log h' = \log 2.3$$

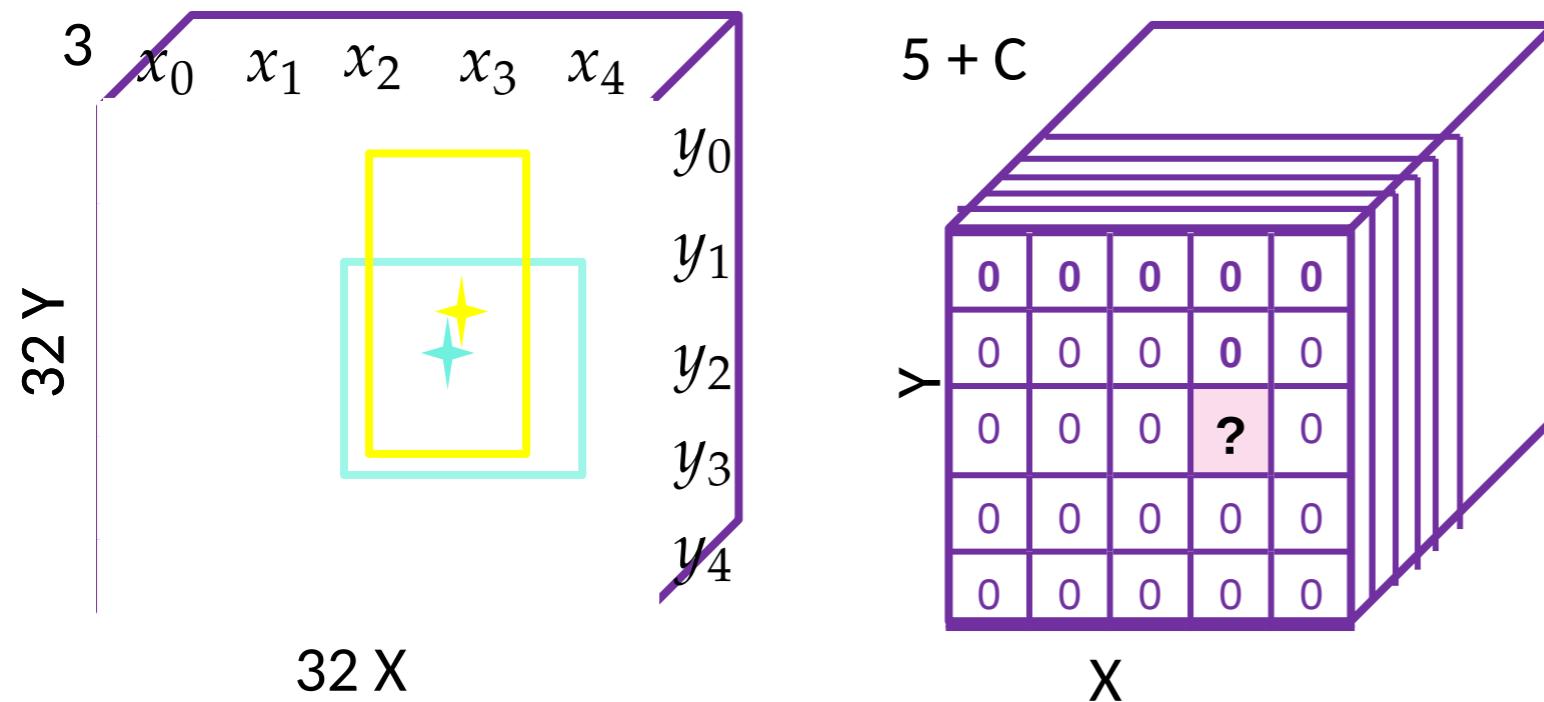
$$c' = 1$$

Как посчитать лосс?

$$\begin{aligned} L &= BCE(p, I') \\ &+ I'(BCE(\Delta x, \Delta x') + BCE(\Delta y, \Delta y') \\ &+ (\log w - \log w')^2 + (\log h - \log h')^2 - \log p_c] \end{aligned}$$

# Модель детектора

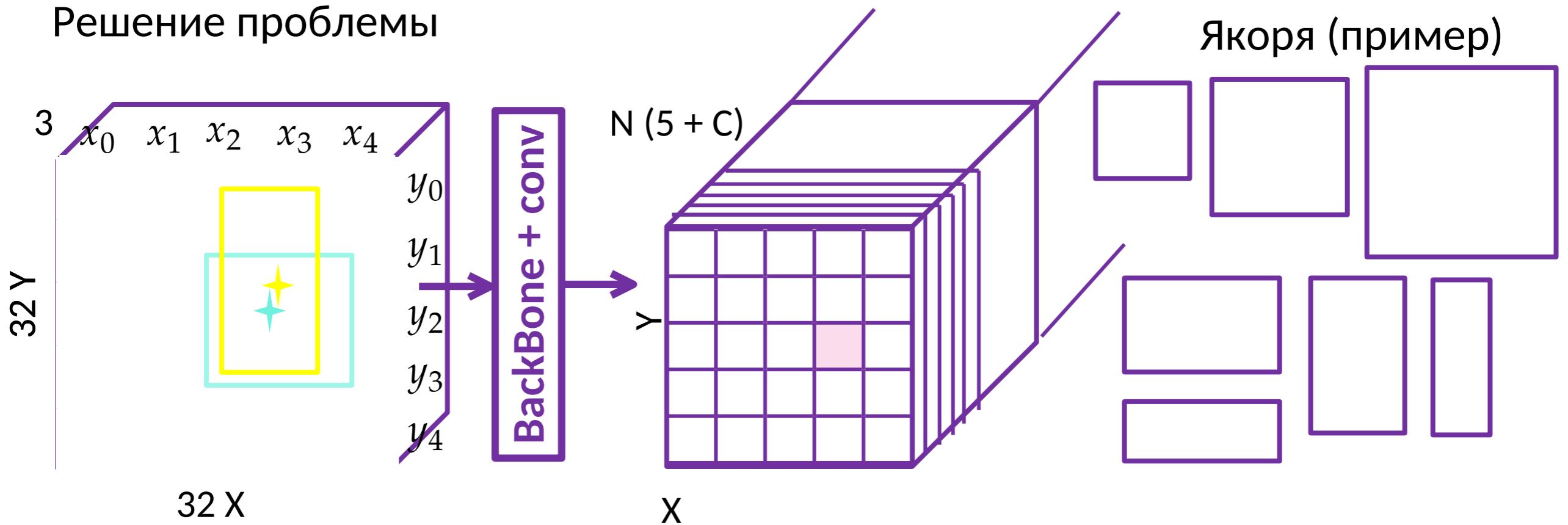
Есть проблема :(



Если 2 или больше объектов находятся в одном супер-пикселе,  
то мы не сможем детектировать оба объекта.

# Модель детектора: Якоря

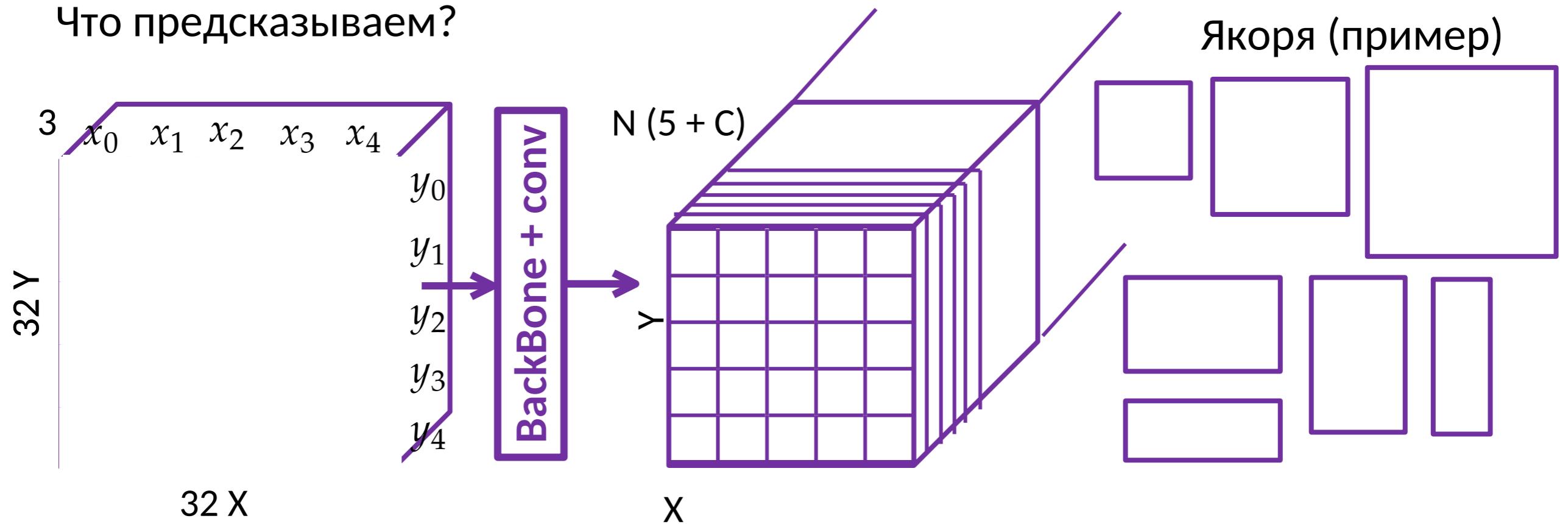
Решение проблемы



Для каждого нейрона будем генерировать предсказание не для одного box-а, а для **целого набора box-в с характерными размерами**. Набор якорей задает характерные размеры.

# Модель детектора: Якоря

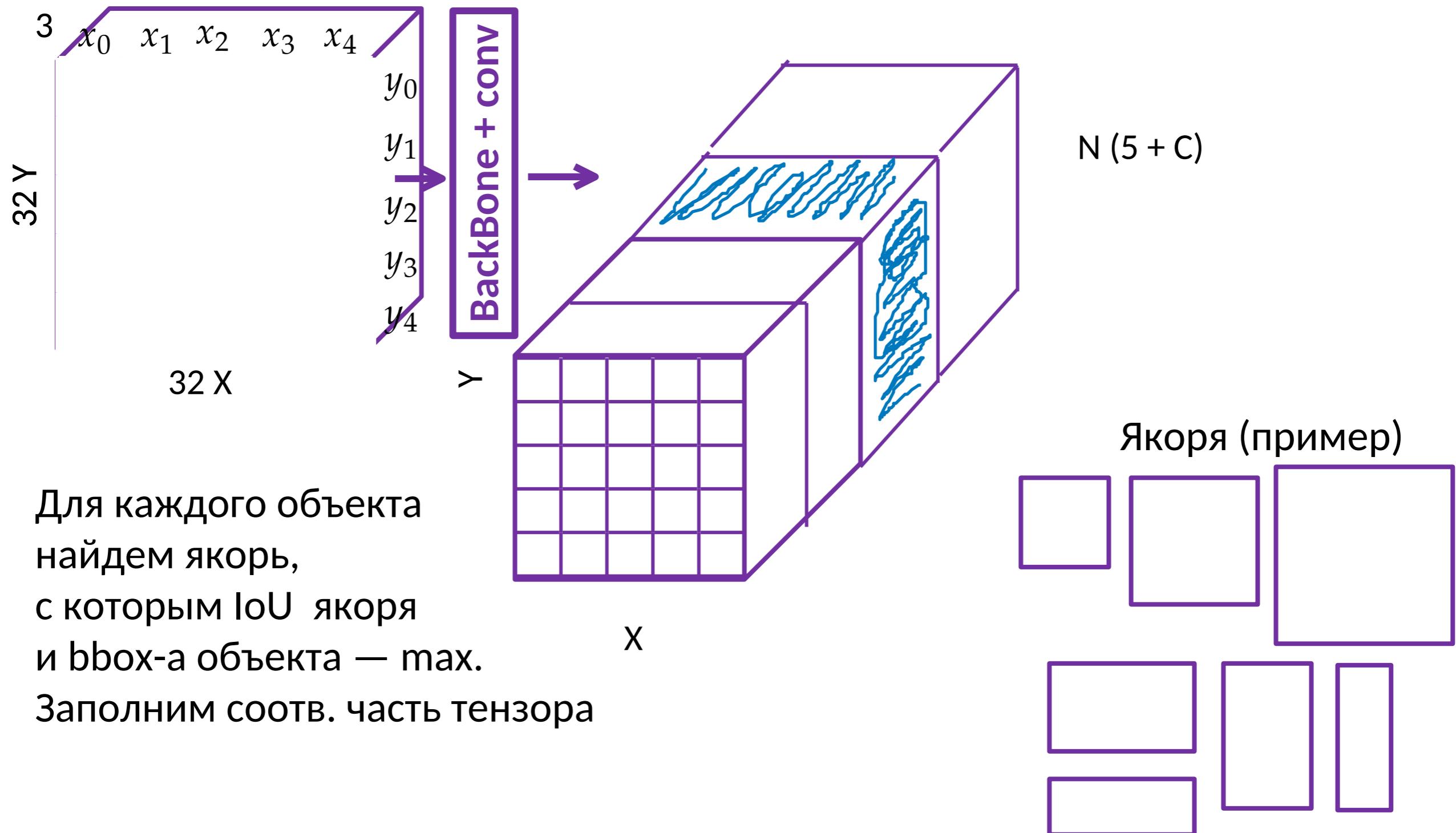
Что предсказываем?



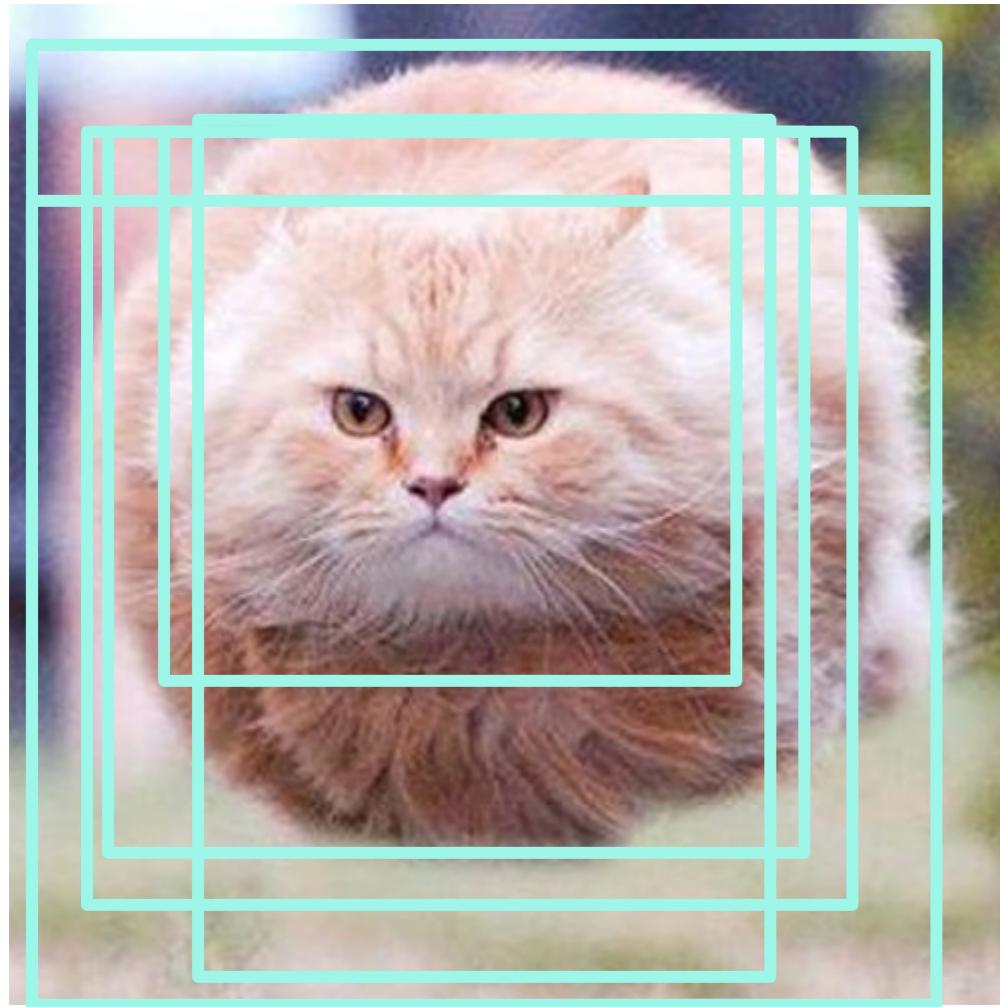
- Пусть  $a$  — номер якоря,  $W_a$  — ширина якоря  $H_a$  — высота якоря.
- Канал №  $a(5 + c) + 0 \rightarrow \sigma \rightarrow p_a$  — вероятность того, что здесь находится один из нужных объектов.
- Каналы №  $a(5 + c) + 1, a(5 + c) + 2 \rightarrow \sigma \rightarrow \Delta x_a, \Delta y_a$  — смещение центра bbox-а с по осям  $x$  и  $y$  соотв. относ. супер-пикселя.
- Каналы №  $a(5 + c) + 3, a(5 + c) + 4 \rightarrow \exp \rightarrow w_a/W_a, h_a/H_a$  — размеры bbox-а, относительно размеров якоря
- Каналы №  $a(5 + c) + 6, \dots \rightarrow \text{SoftMax} \rightarrow p_{ac}$  — вероятность класса  $C$ .

# Модель детектора: Якоря. Обучение

Как подготовить данные?

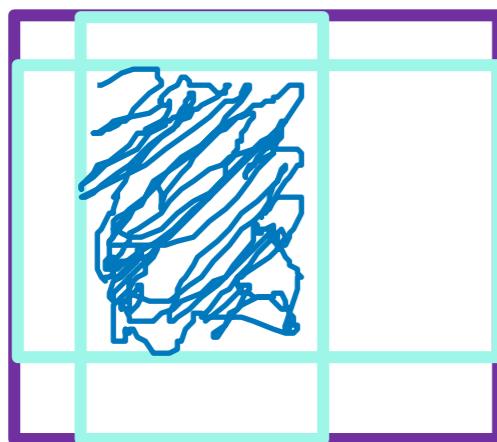


# Модель детектора: NMS. Применение

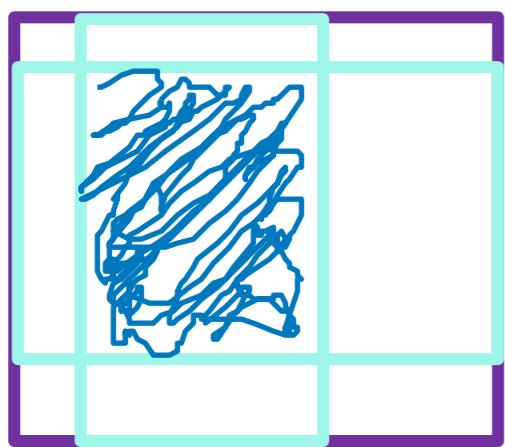
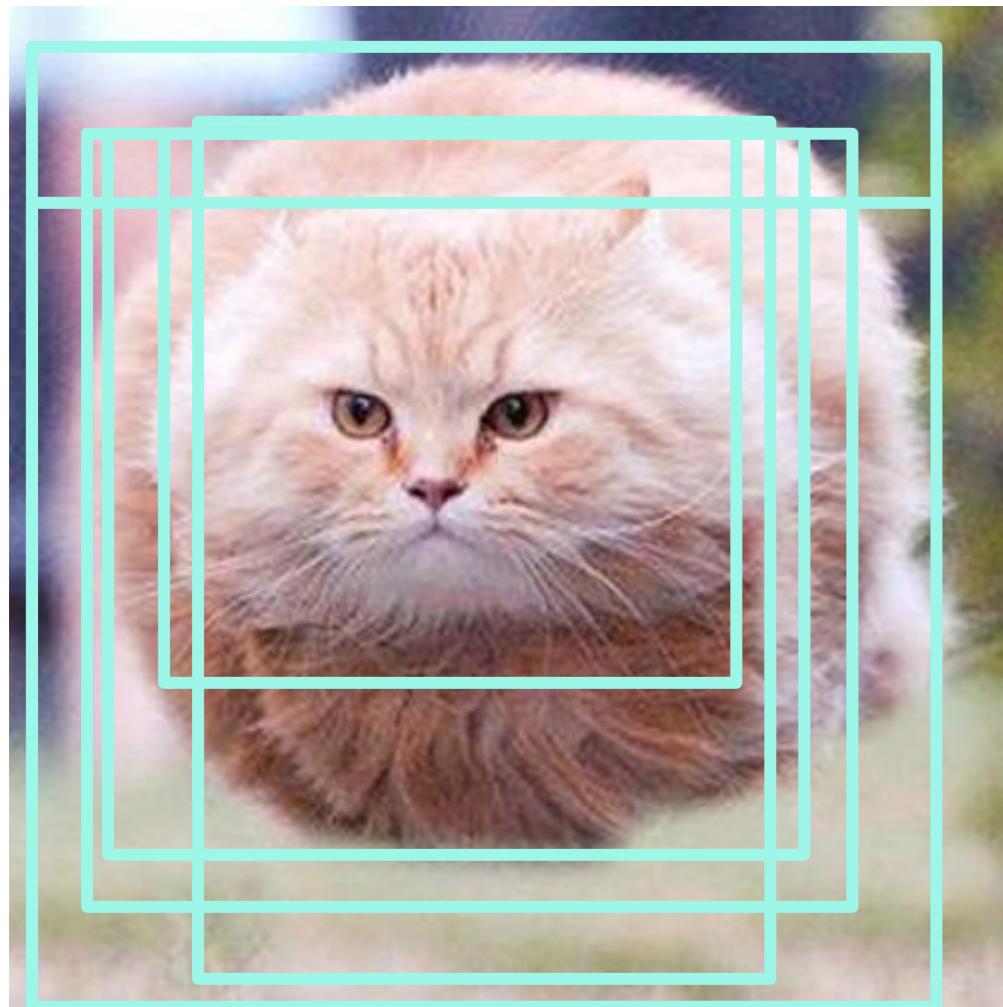


Для каждого объекта будет генерироваться очень много bbox-в. Нужно делать отбор.

Эту задачу решает алгоритм **Non Maximum Supression (NMS)**.



# Модель детектора: NMS. Применение



## **Non Maximum Supression (NMS).**

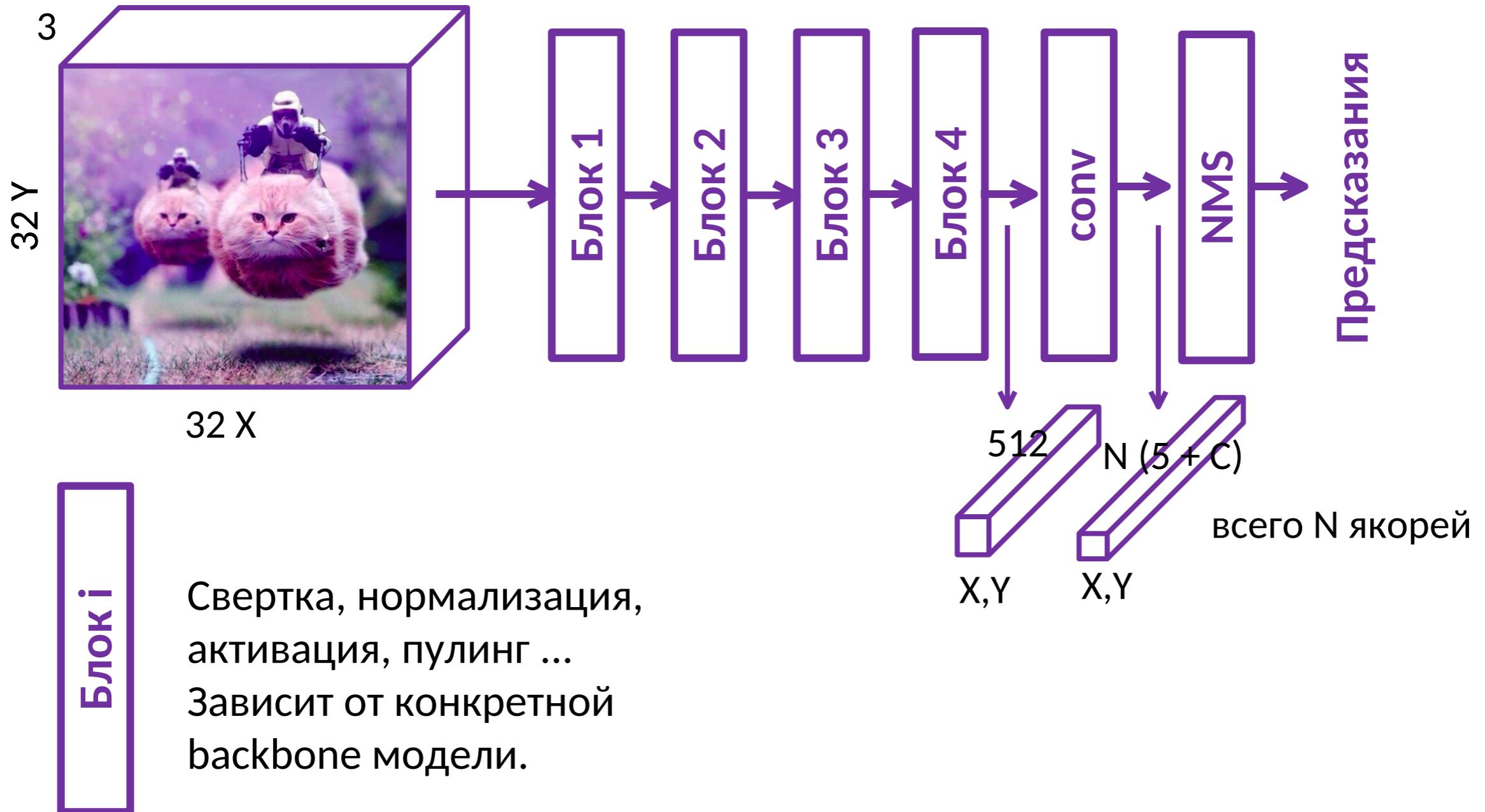
Сортируем предсказ. боксы заданного класса по степени уверенности.

Задаем пороги  $r_{min}$  и  $threshold$ .

В цикле пока не кончатся боксы.

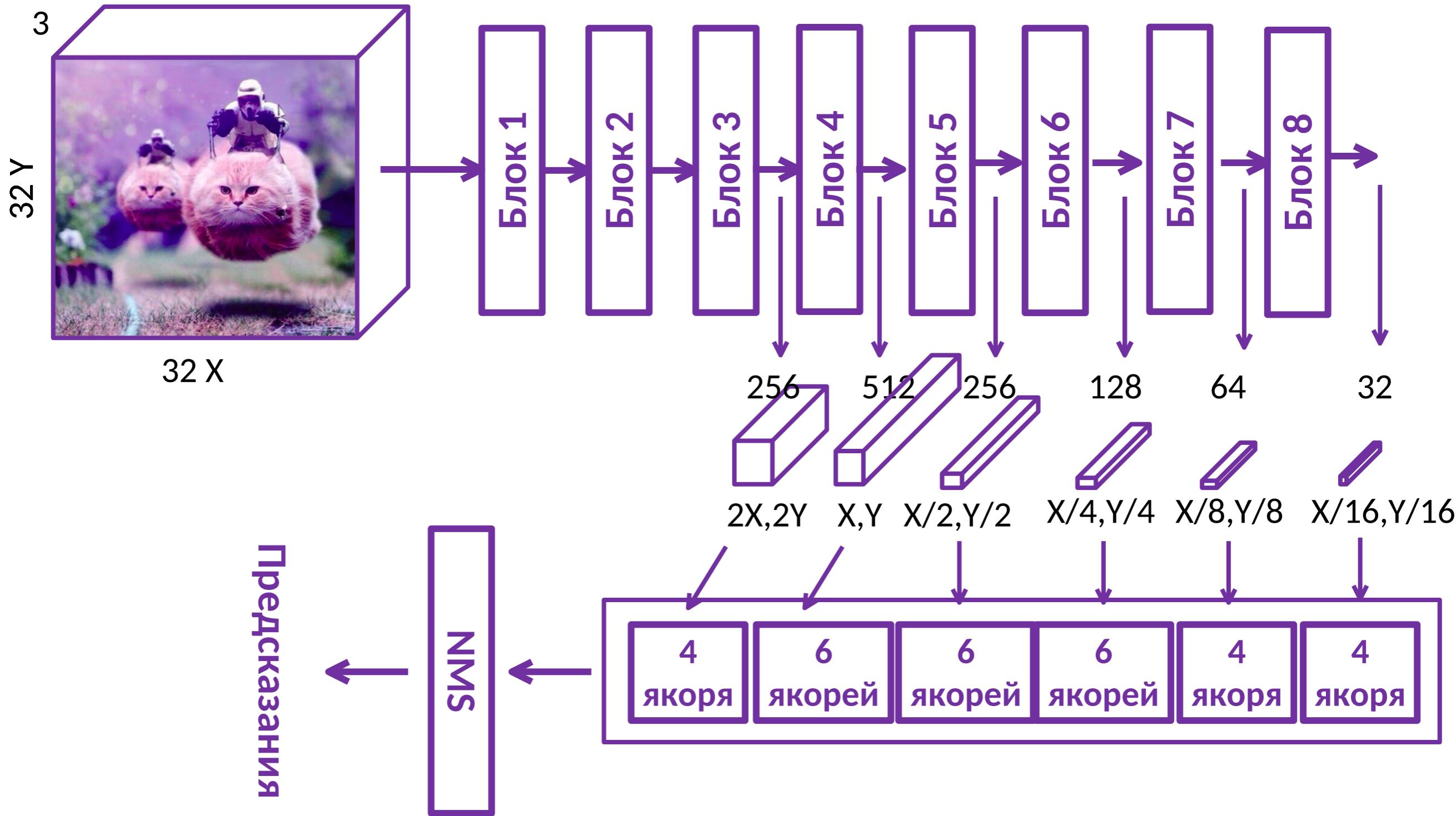
- Если уверенность в боксе  $< r_{min}$ , останавливаемся.
- Выбираем бокс с макс. уверенностью и кладем в итоговый список.
- Берем следующий с макс. уверенностью,
  - уверенность в боксе  $< r_{min}$ , останавливаемся
  - если  $IoU > threshold$ , то bbox-ы совпадают, выкидываем второй бокс и берем новый
  - иначе добавляем его.

# Итоговая архитектура YOLO

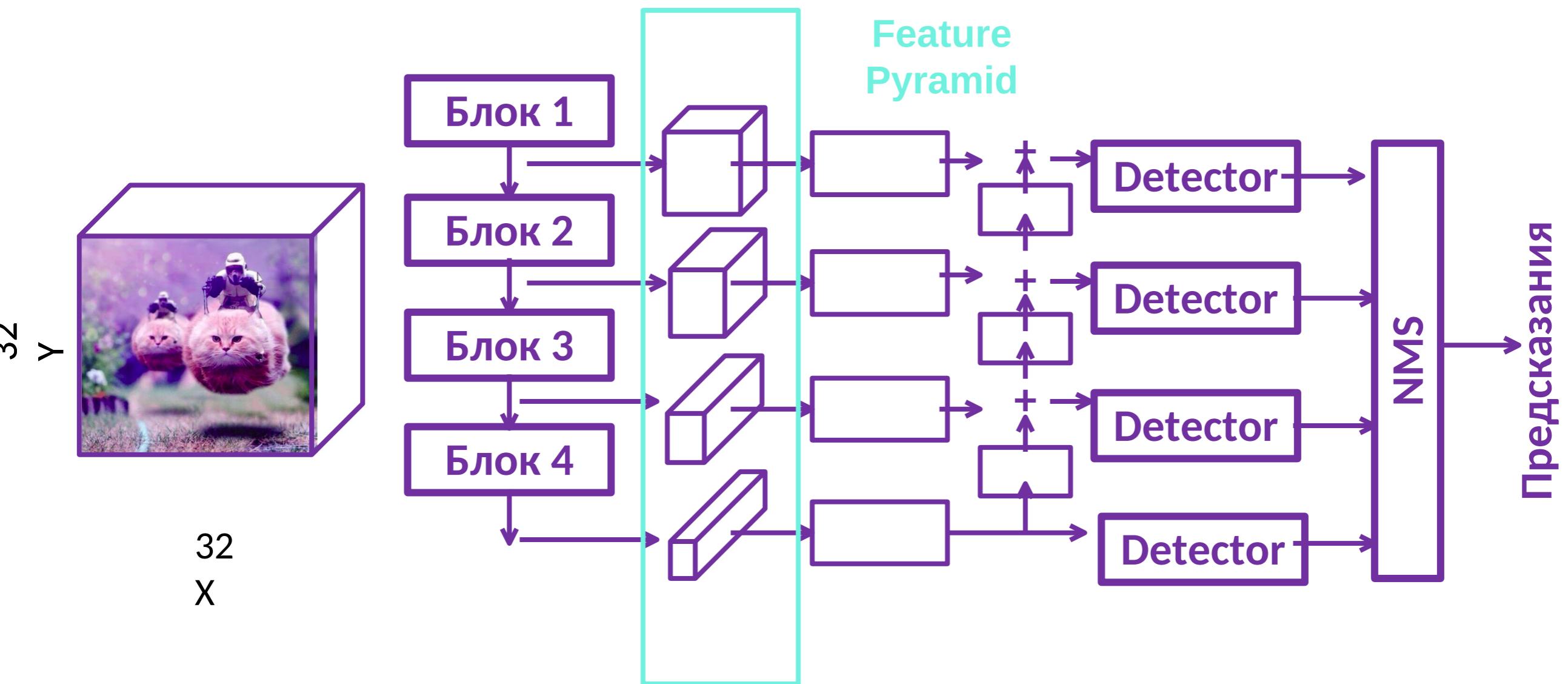


# Докрутим YOLO: SSD

Мы делали предсказания только на основе последней карты активации.  
Но мы можем использовать менее глубокие карты,  
чтобы предсказывать более мелкие объекты. Получим модель SSD.



# Докрутим SSD: RetinaNet



Чтобы детекторы более мелких объектов лучше понимали контекст, используется feature pyramid.

# Двухстадийные детекторы

# Двухстадийные детекторы

**Стадия 1:** генерируем регионы интереса, или Regions of Interest (RoI)

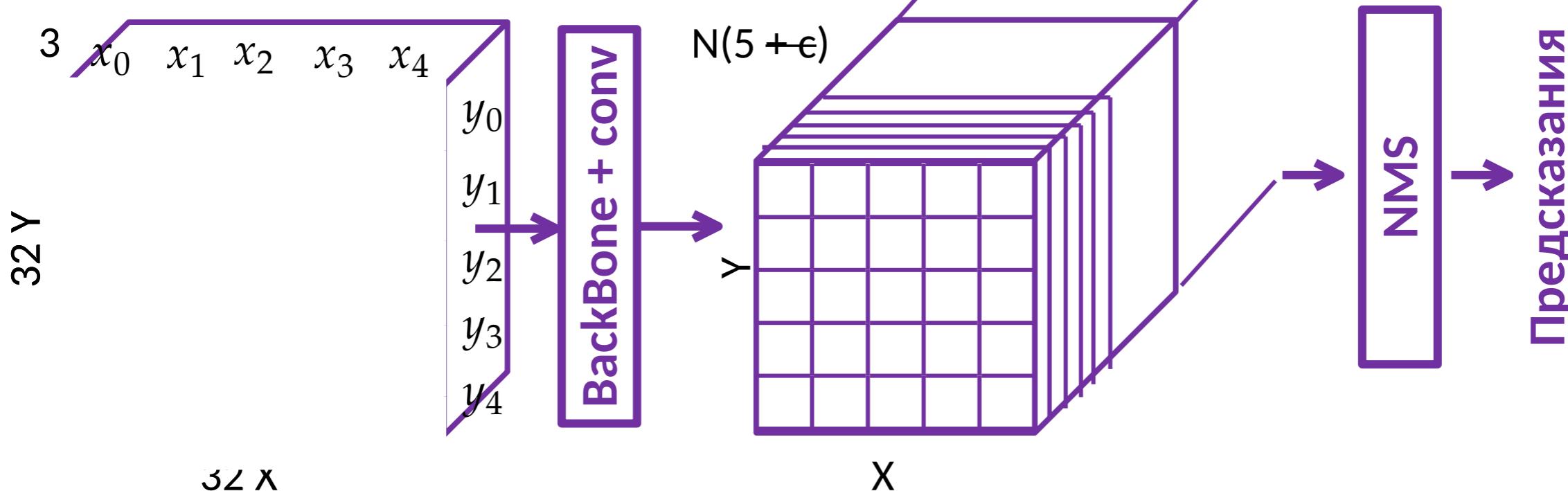
**Стадия 2:** уточняем их координаты и предсказываем класс внутри каждого RoI

# Region Proposal Network (RPN)

Все то же самое, что в YOLO, кроме классов.

Будем предсказывать расположение объектов на изображении.

**Классы не предсказываем.**



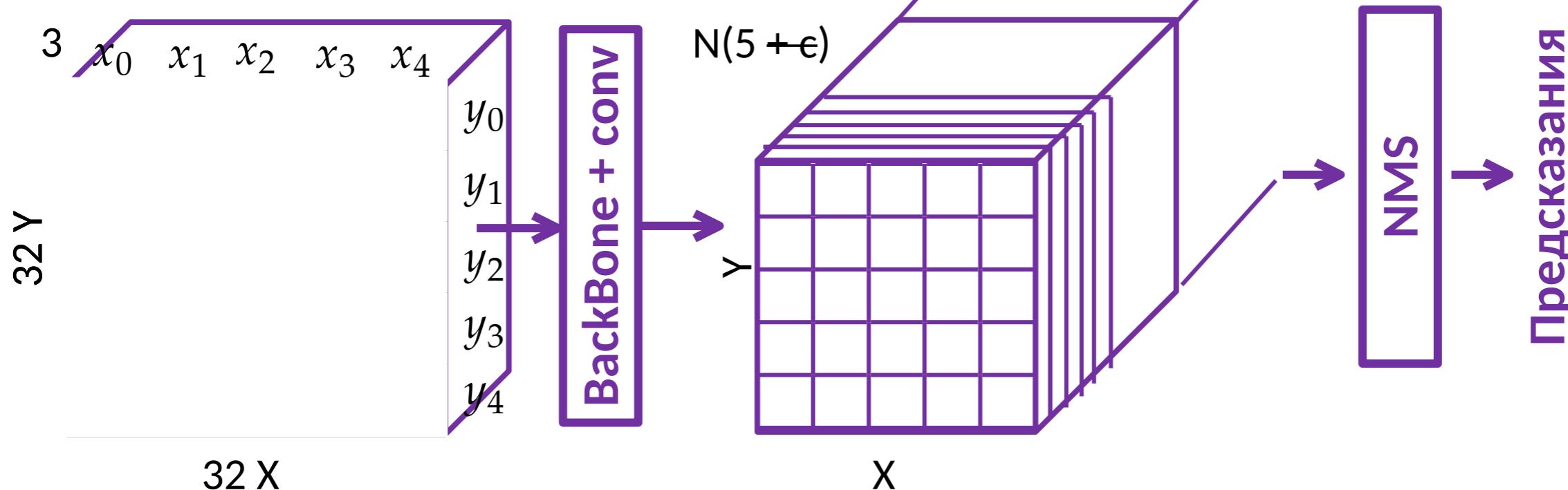
- Пусть  $a$  — номер якоря,  $W_a$  — ширина якоря  $H_a$  — высота якоря.
- Канал №  $a(5 + \epsilon) + 0 \rightarrow \sigma \rightarrow p_a$  — вероятность того, что здесь находится один из нужных объектов.
- Каналы №  $a(5 + \epsilon) + 1, a(5 + \epsilon) + 2 \rightarrow \sigma \rightarrow \Delta x_a, \Delta y_a$  — смещение центра bbox-а с по осям  $x$  и  $y$  соотв. относ. супер-пикселя.
- Каналы №  $a(5 + \epsilon) + 3, a(5 + \epsilon) + 4 \rightarrow \exp \rightarrow w_a/W_a, h_a/H_a$  — размеры bbox-а, относительно размеров якоря
- Каналы №  $a(5 + \epsilon) + 6, \dots \rightarrow \text{SoftMax} \rightarrow p_{ae}$  — вероятность класса  $e$ .

# Region Proposal Network (RPN)

Все то же самое, что в YOLO, кроме классов.

Будем предсказывать расположение объектов на изображении.

**Классы не предсказываем.**



Как посчитать лосс?

$$\begin{aligned} L &= BCE(p, I') \\ &+ I'(BCE(\Delta x, \Delta x') + BCE(\Delta y, \Delta y')) \\ &+ (\log w - \log w')^2 + (\log h - \log h')^2 - \log p_\epsilon] \end{aligned}$$

# Двухстадийные детекторы

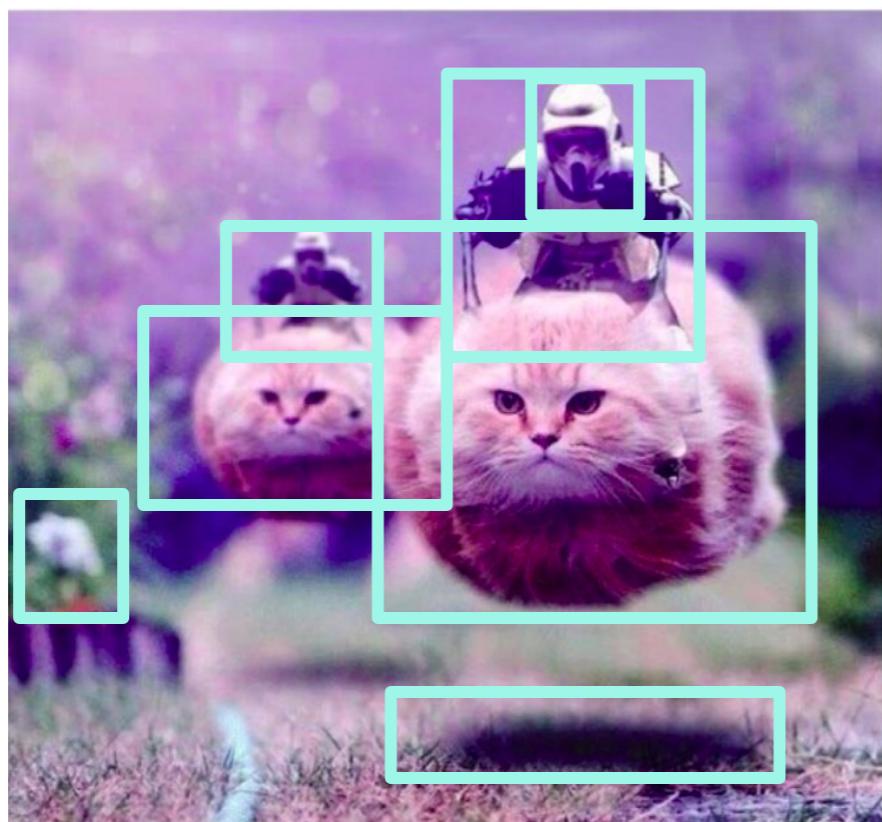
Будем работать с предсказанными областями из RPN.

Для всех областей будем уточнять размеры bbox-в.

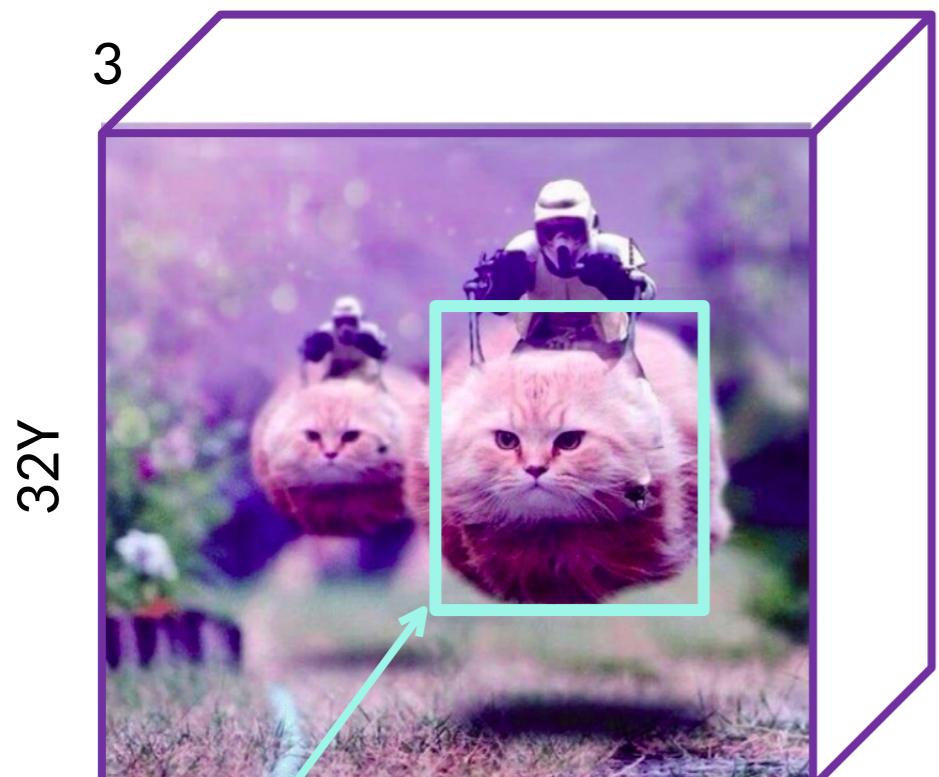
- RPN отбрала для нас интересные области.
- В некоторых из областей есть интересные нужные нам объекты.
- Мы применили NMS, который мог привести к растягиванию рамок.

Теперь рассмотрим каждую область по отдельности.

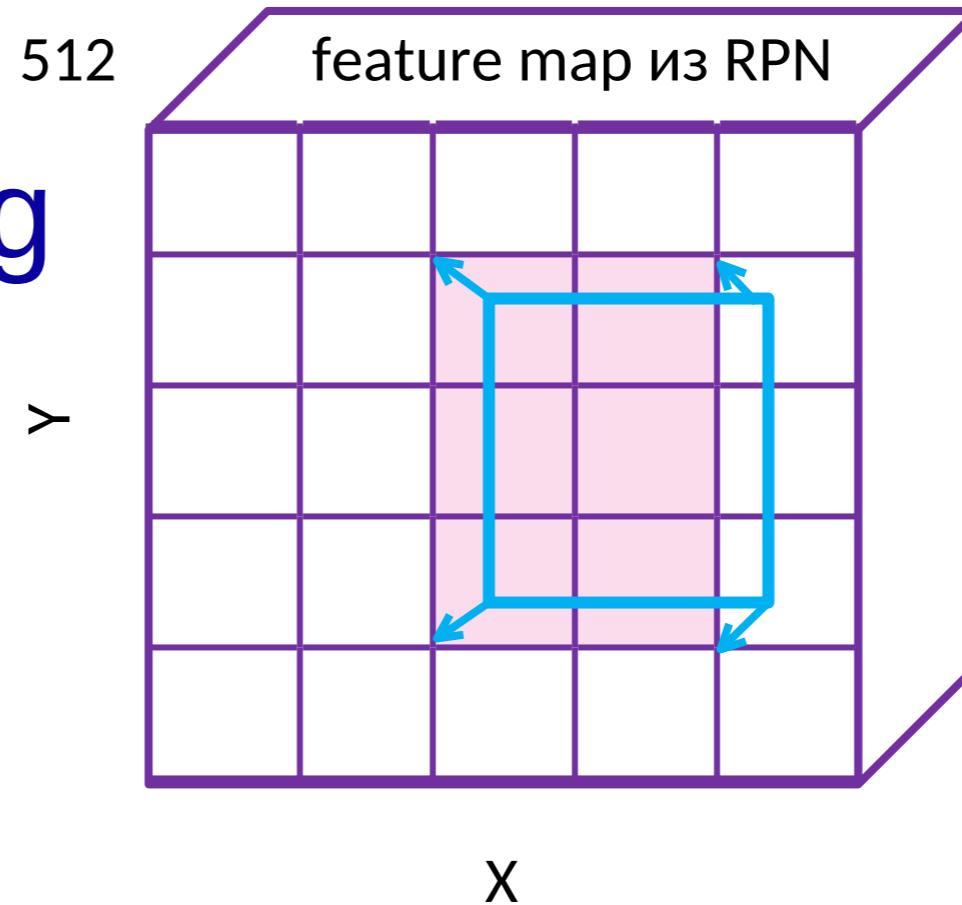
Классифицируем ее и уточним границы.



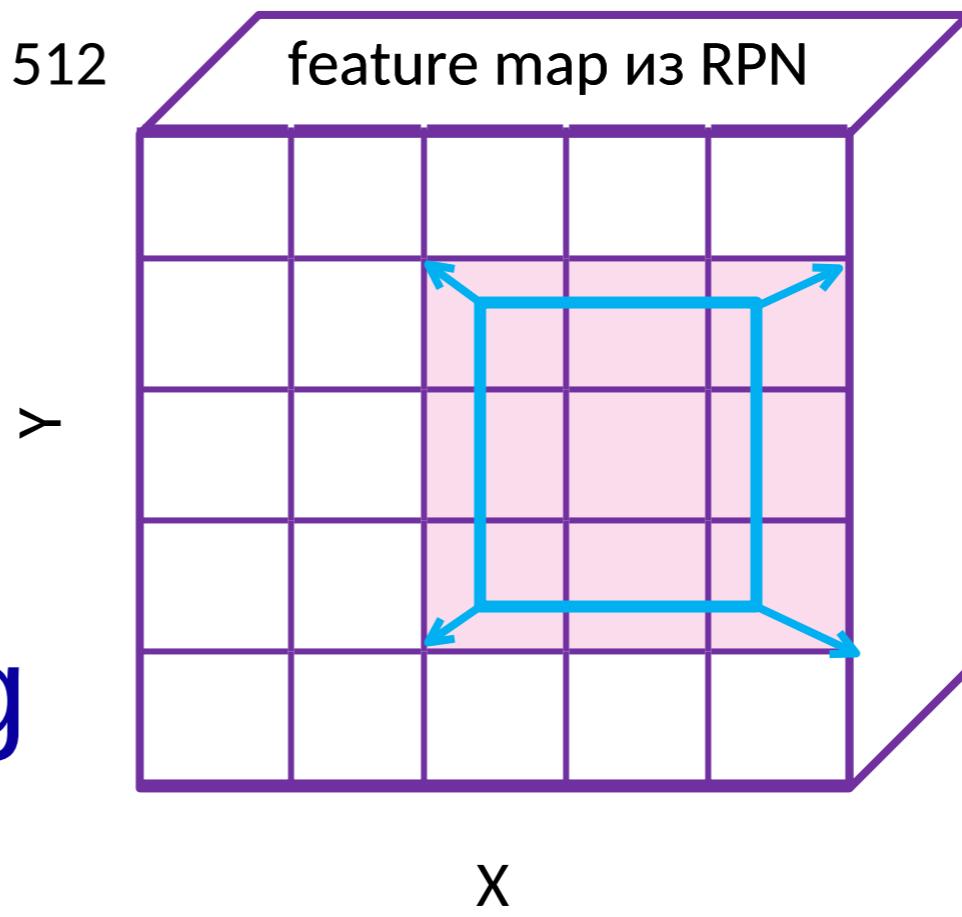
# ROI Pooling



# ROI Align

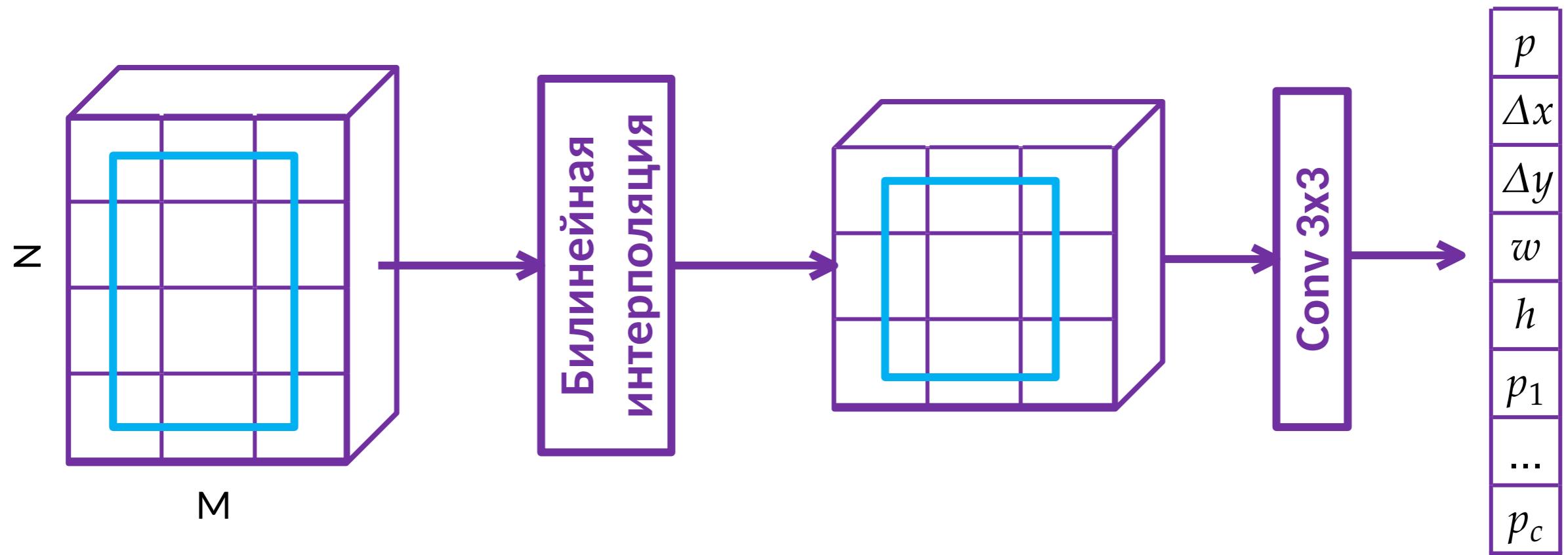


[Подробнее  
про ROI  
Pooling](#)



[Подробнее  
про ROI  
Align](#)

# Финальный детектор

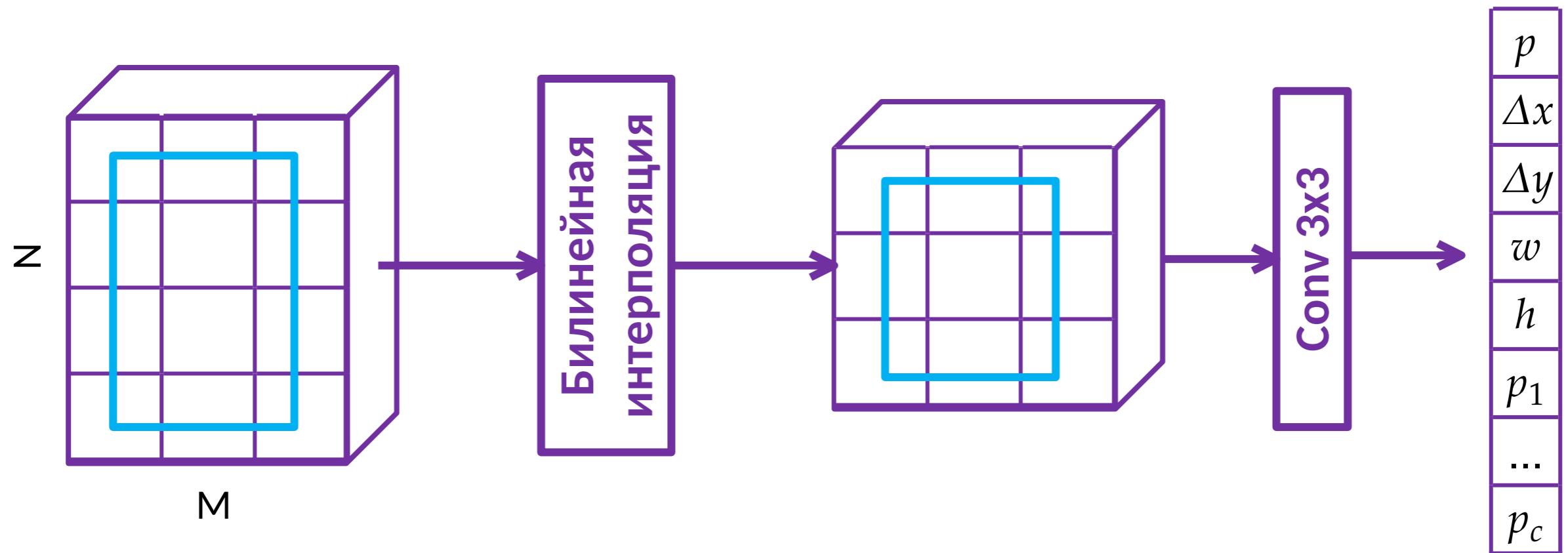


**Будем предсказывать**

- $p$  — вероятность того, что здесь находится один из нужных объектов.
- $\Delta x, \Delta y$  — смещение объекта по осям  $x$  и  $y$  соотв. относ. выделенной области.
- $w, h$  — размеры относ. выделенной области.

Финальный детектор — небольшая нейронная сеть, которая классифицирует и выдает характеристики тензорам размера 3x3.

# Финальный детектор

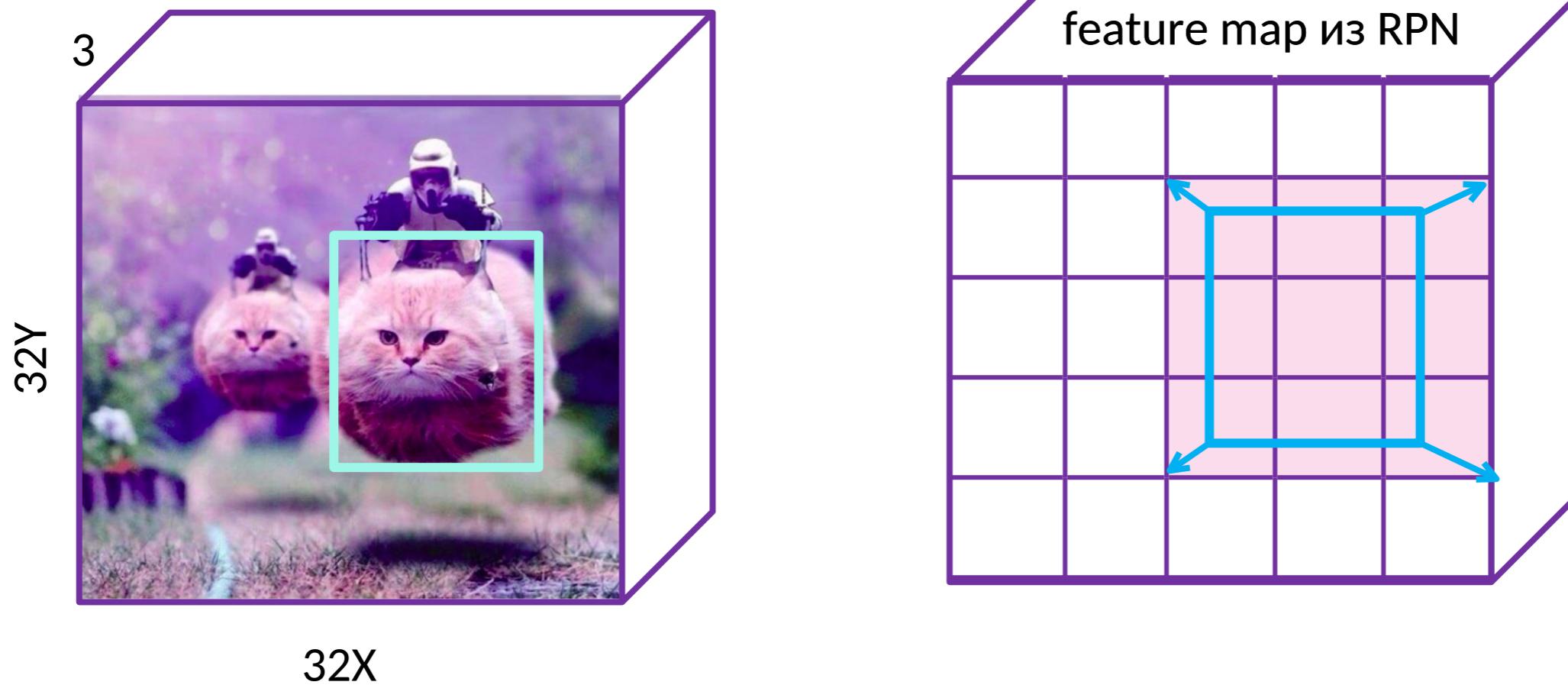


**Что делать с разметкой?**

Мы не можем предсказать размеры исходного bbox-а, потому что в сеть эта информация не поступает ни в каком виде. Поэтому будем интерполировать размер таргетного bbox-а и предсказывать интеполированные координаты.

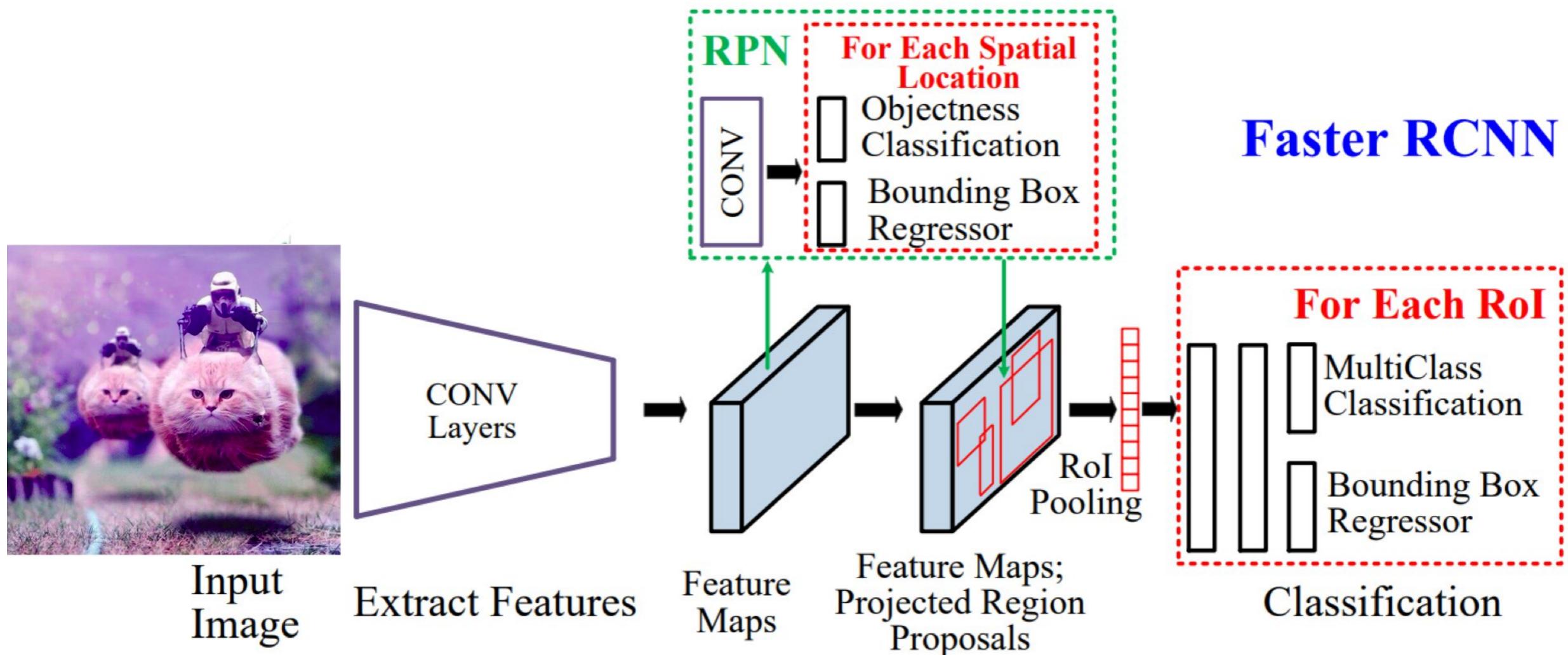
# Двухстадийные детекторы

Мы могли бы просто вырезать предсказанный bbox из картинки и пропускать его через нейронную сеть, чтобы получать предсказания. Но такие регионы будут гораздо больше тех, что в feature-карт, а значит параметров для такой сети нужно гораздо больше.



# Двухстадийные детекторы: Faster-RCNN

Мы получили модель Fast-RCNN

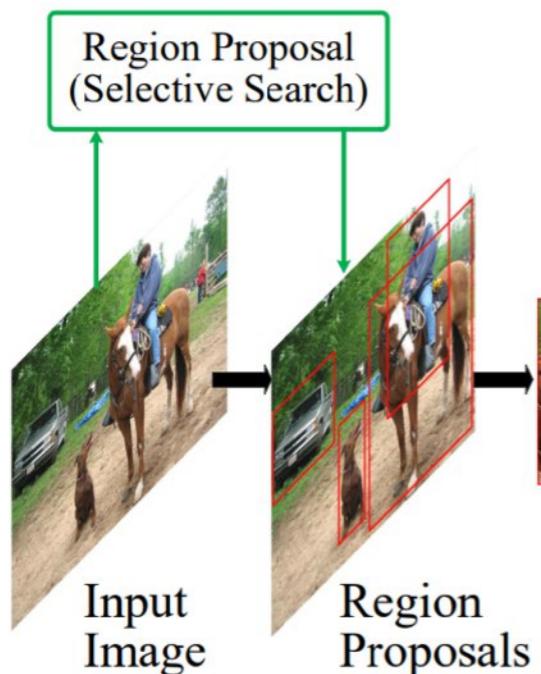


# Двухстадийные детекторы: история

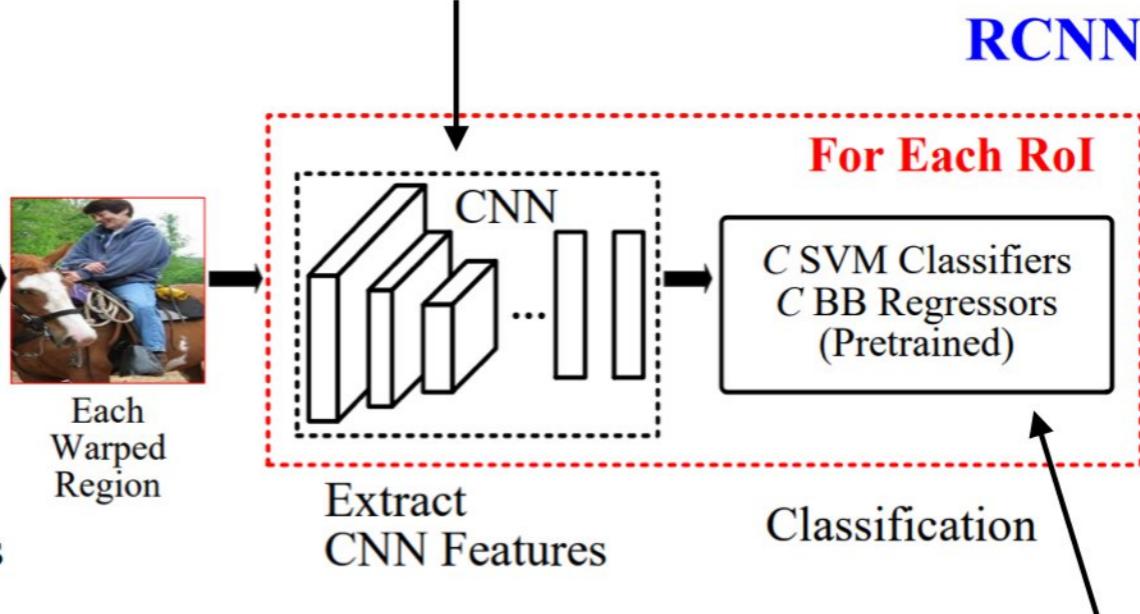
- RCNN
- Fast RCNN
- Faster RCNN
- ...

# Модели: Region-proposal CNN

1. Сгенерировать предположения боксов



3. Извлечь признаки из каждого бокса нейросетью



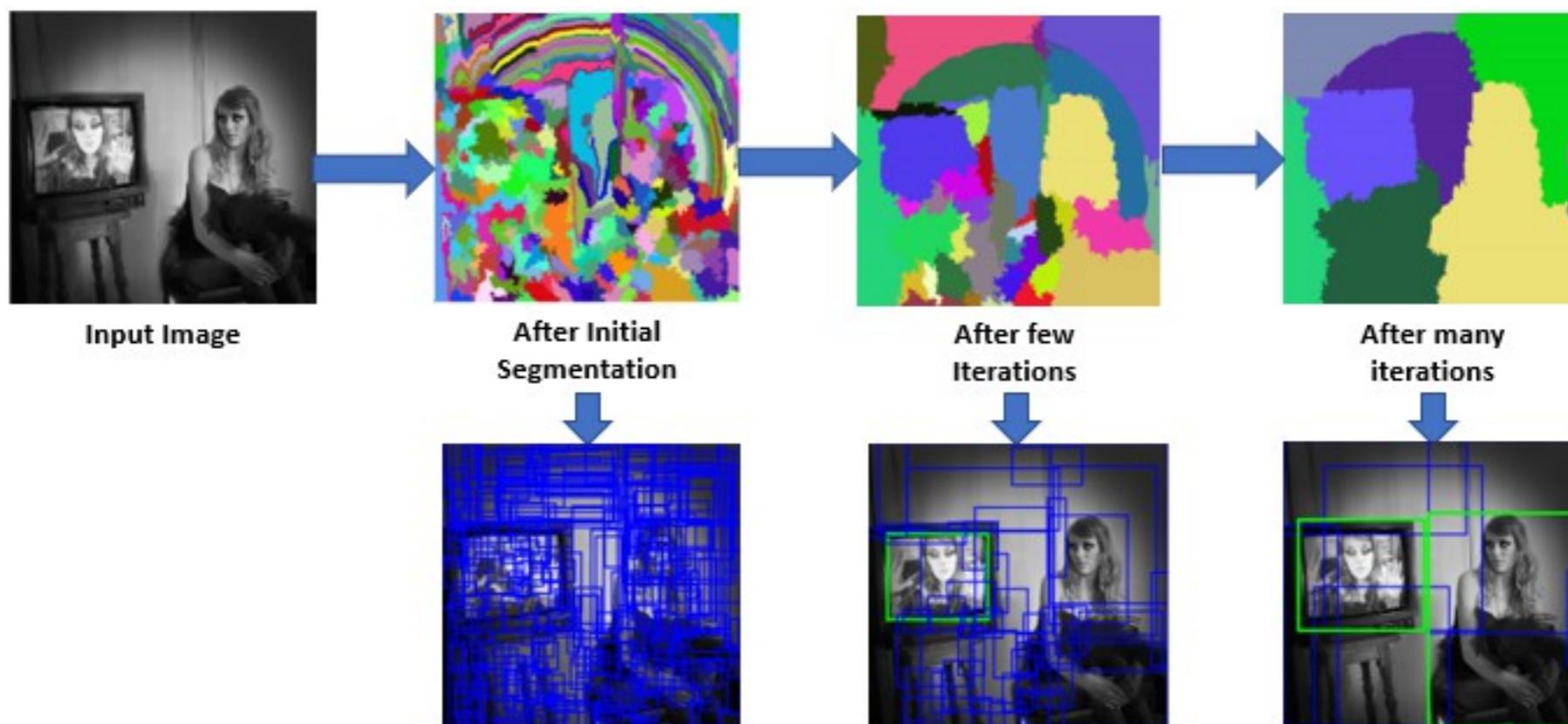
2. Привести боксы к одному размеру

4. Уточнить координаты  
и классифицировать бокс

# Selective search

Алгоритм Selective Search:

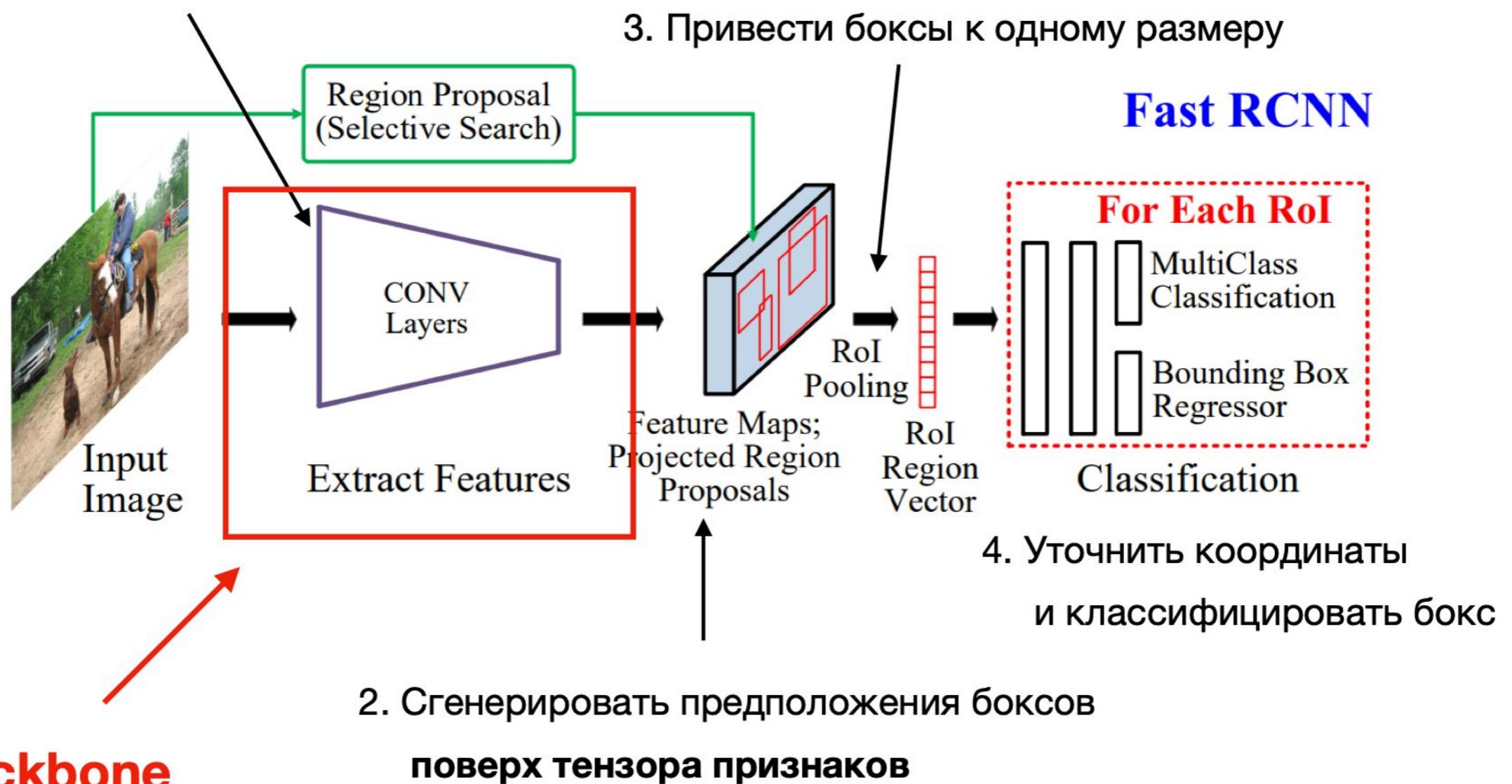
1. Сгенерировать начальную сегментацию картинки с помощью метода из статьи “Efficient Graph-Based Image Segmentation”;
2. Рекурсивно объединить маленькие участки в большие жадным алгоритмом;
3. Используя границы сегментации на последнем шаге извлечь bounding box’ы



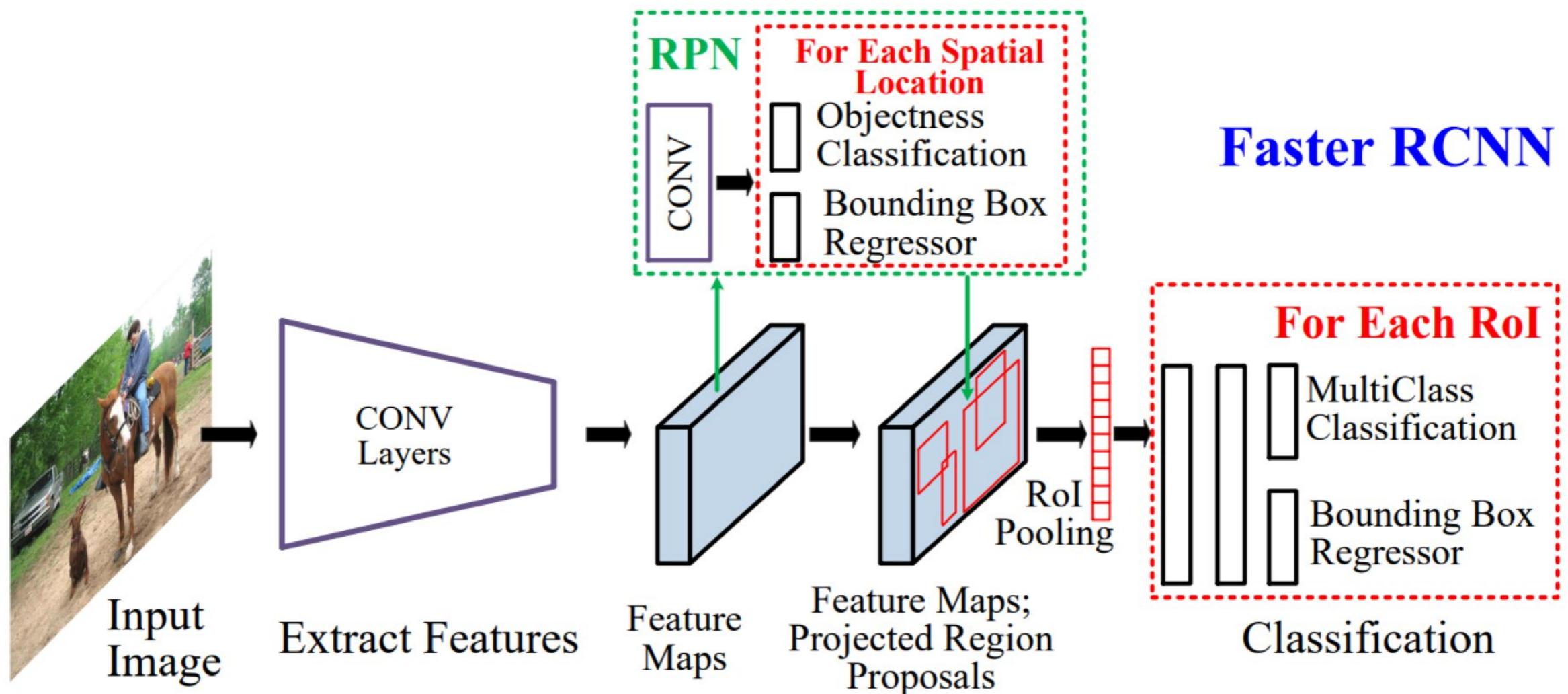
[source](#)

# Модели: Fast RCNN

1. Извлечь признаки из всей картинки



# Модели: Faster RCNN

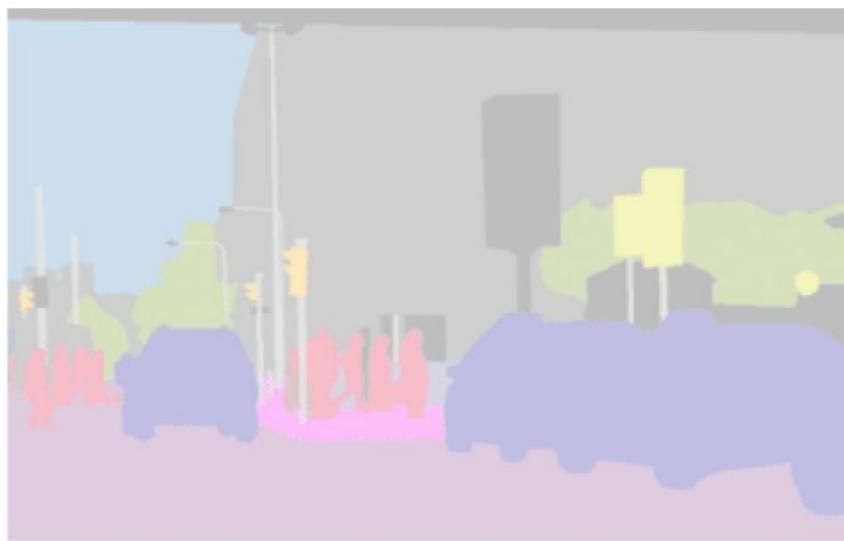


# Инстанс-сегментация

# Задача инстанс-сегментации картинок



(a) Image



(b) Semantic Segmentation

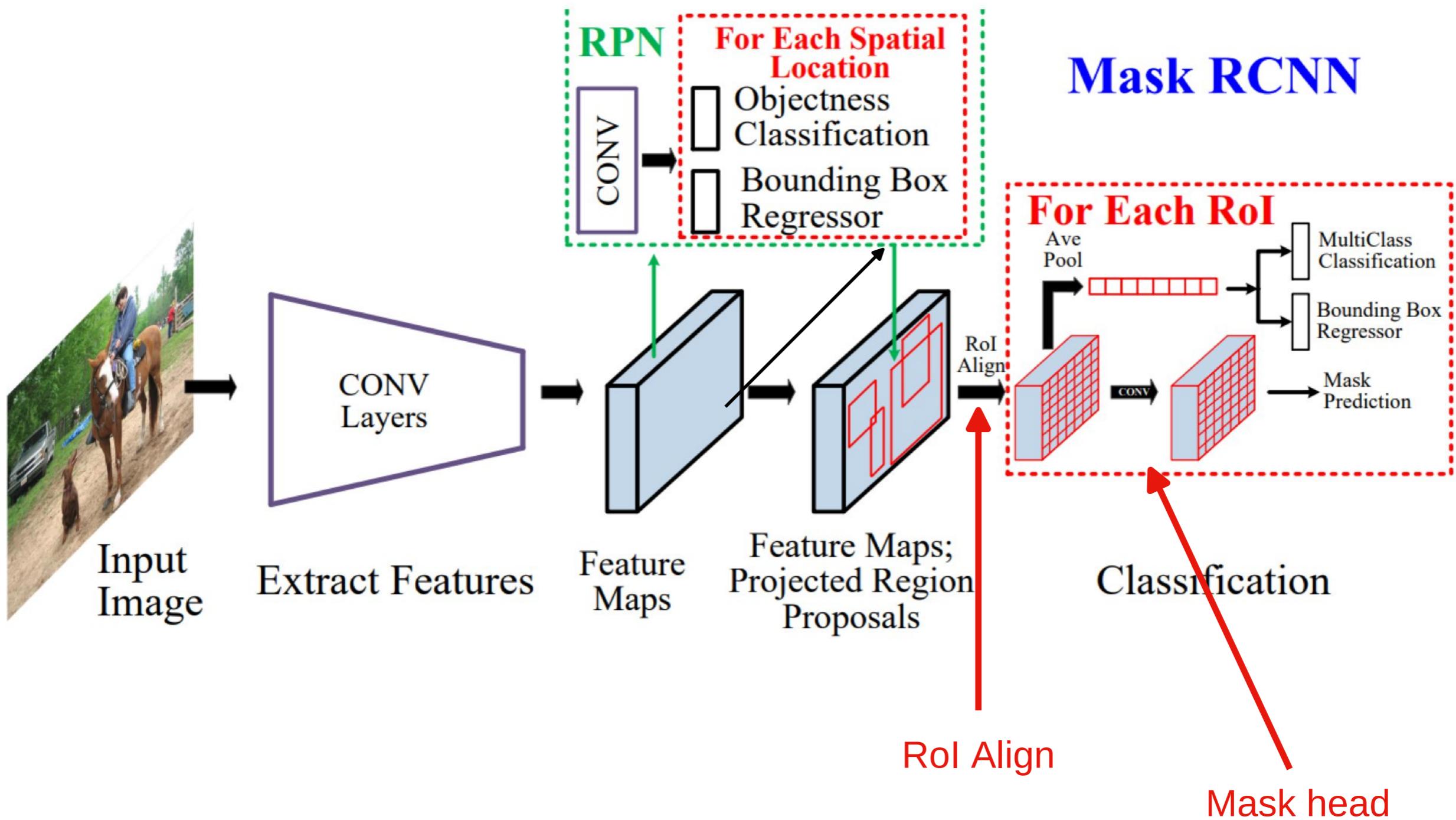


(c) Instance Segmentation



(d) Panoptic Segmentation

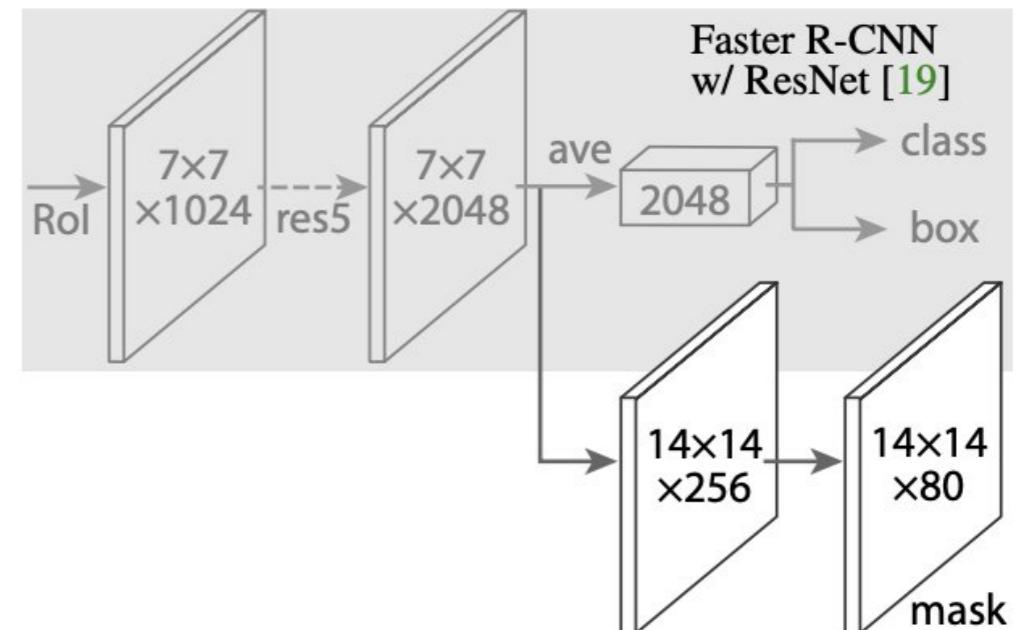
# Детекция + Сегментация: Mask RCNN



# Mask-RCNN: Mask head

- Вход: Feature map размера  $7 \times 7 \times 2048$
- Выход:  $y_{pred}^{small}$  размера  $14 \times 14 \times K$
- $y_{pred} = \text{resize}(y_{pred}^{small}, (H_{RoI}, W_{RoI}))$
- Пусть  $k_{pred}$  - предсказанный класс
- Пусть  $y \in \mathcal{Y}$  - истинная маска размера  $H_{RoI} \times W_{RoI} \times K$ ,  $k \in K$  - истинный класс
- Тогда функции потерь - попиксельный LogLoss:

$$\mathcal{L}(y, y_{pred}, k, k_{pred}) = -\frac{1}{H_{RoI} * W_{RoI}} \sum_{i=1}^{H_{RoI}} \sum_{j=1}^{W_{RoI}} y[i, j, k] \log(y_{pred}[i, j, k_{pred}]) + (1 - y[i, j, k]) (1 - \log(y_{pred}[i, j, k_{pred}])))$$



# Задача оценки ключевых точек

**Задача :**

Пусть  $\mathcal{X}$  - пространство картинок.

$\mathcal{Y}$  - набор классов, например {грудь, левое плечо, правое плечо}.

$\widehat{\mathcal{X}}$  - пространство картинок, где каждый пиксель имеет значение из  $\mathcal{Y}$ .

Требуется построить модель  $f: \mathcal{X} \rightarrow \widehat{\mathcal{X}}$ , определяющую к какому классу из  $\mathcal{Y}$  принадлежит каждый пиксель изображения  $\mathcal{X}$ .

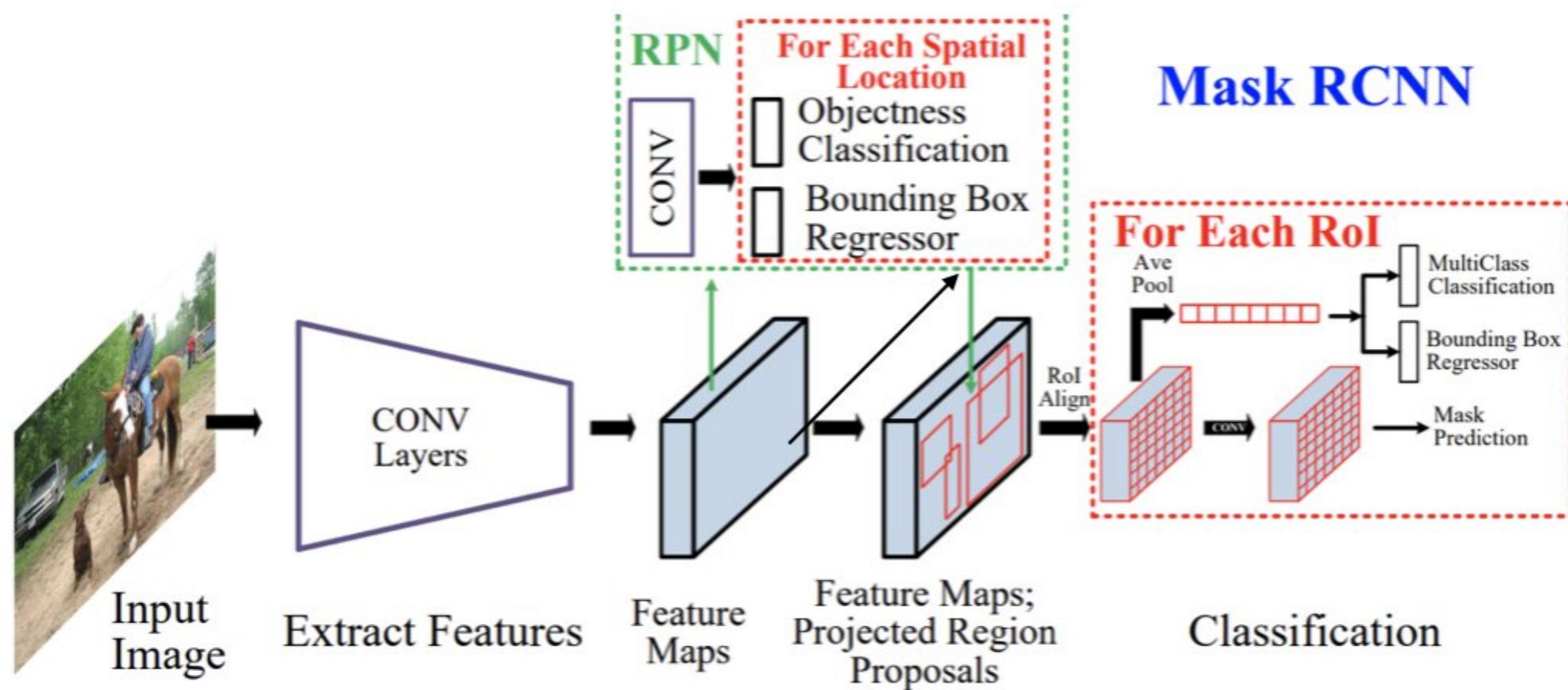


COCO KeyPoints

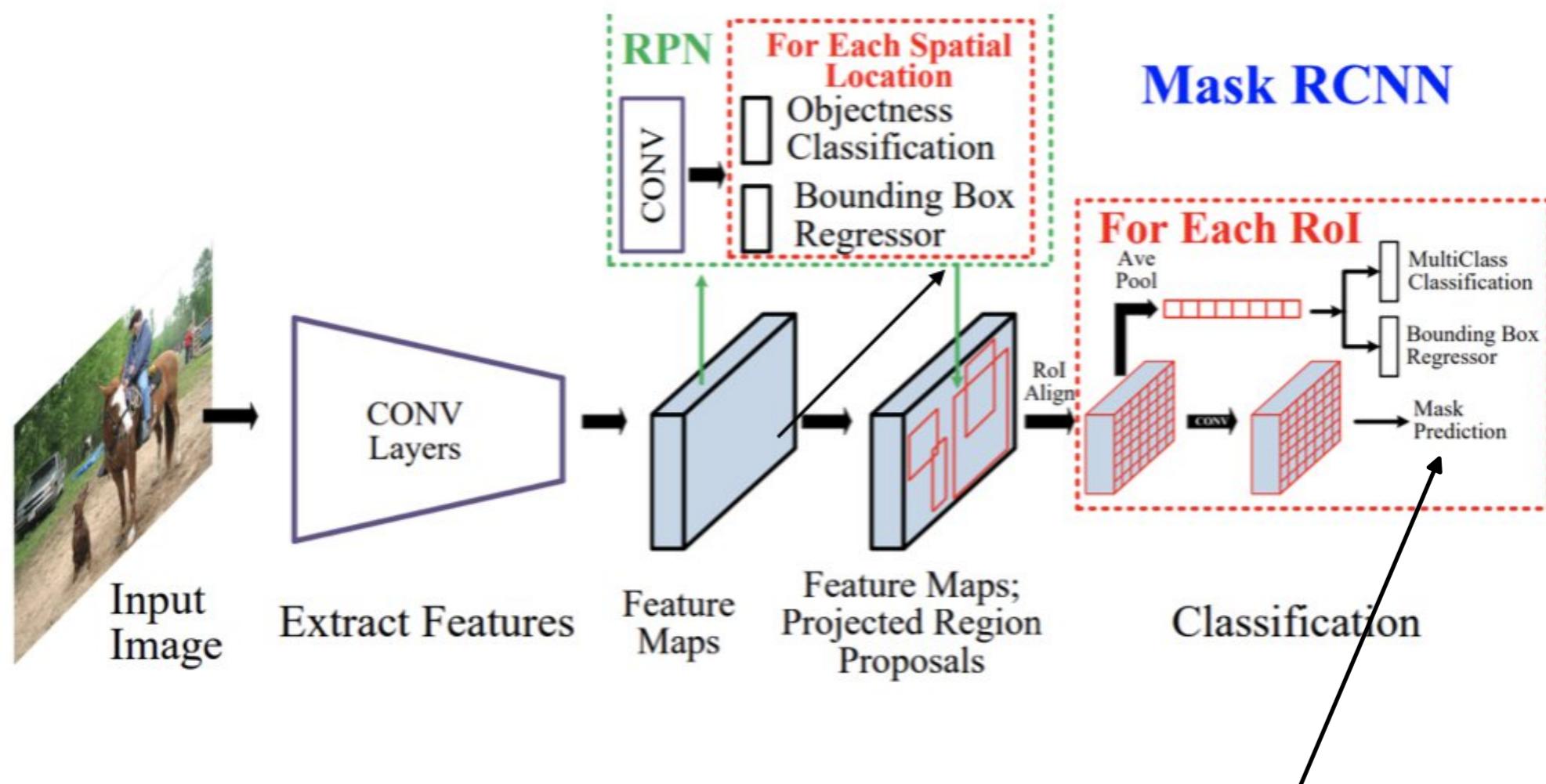


MPII KeyPoints

# Keypoint-RCNN



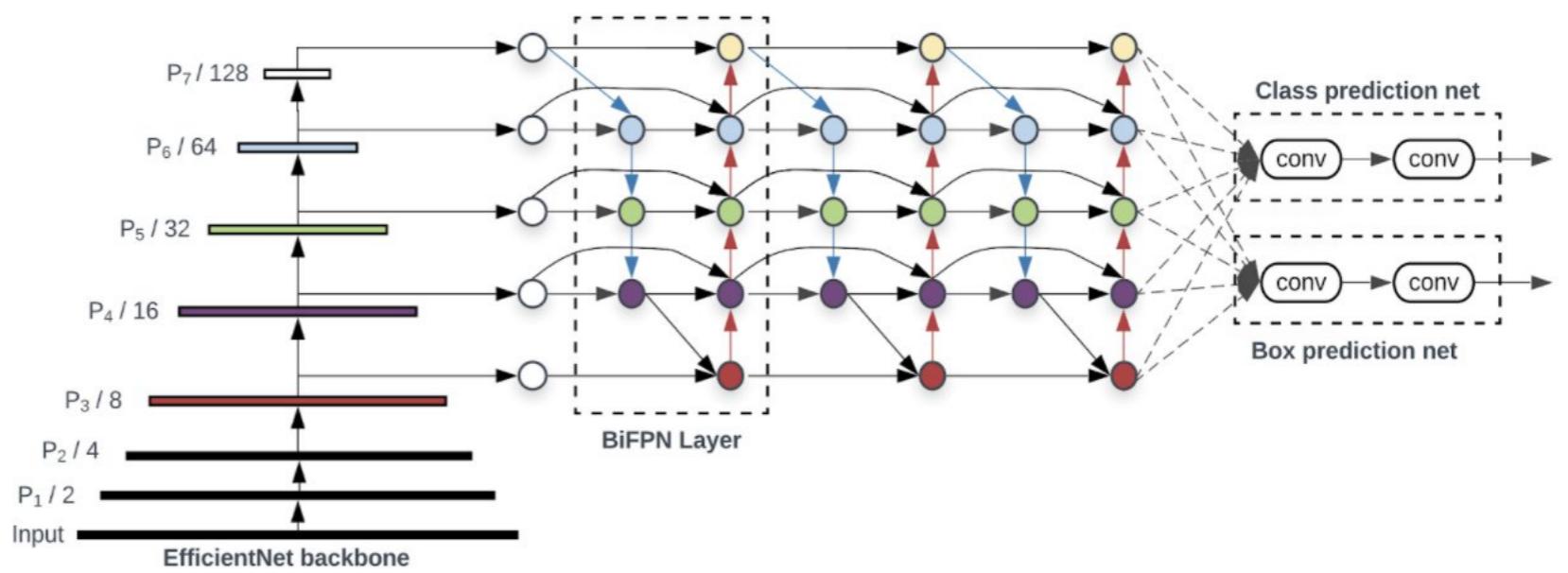
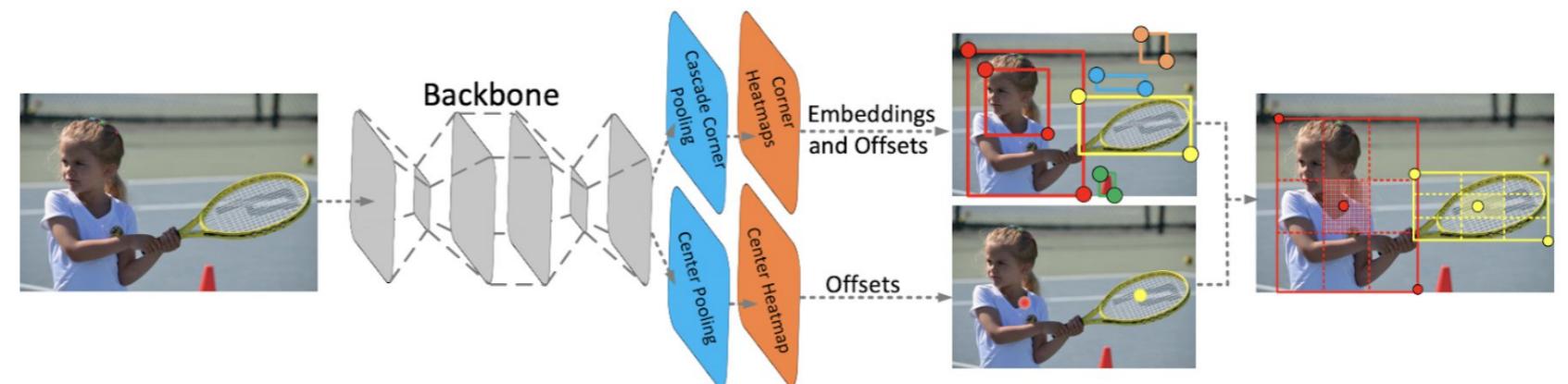
# Keypoint-RCNN



Считаем одну ключевую точку одним классом:  
один канал для предсказания карты вероятности  
одной ключевой точки

# Современные задачи и решения

- CenterNet
- YOLOv4
- EfficientDet





BGE!