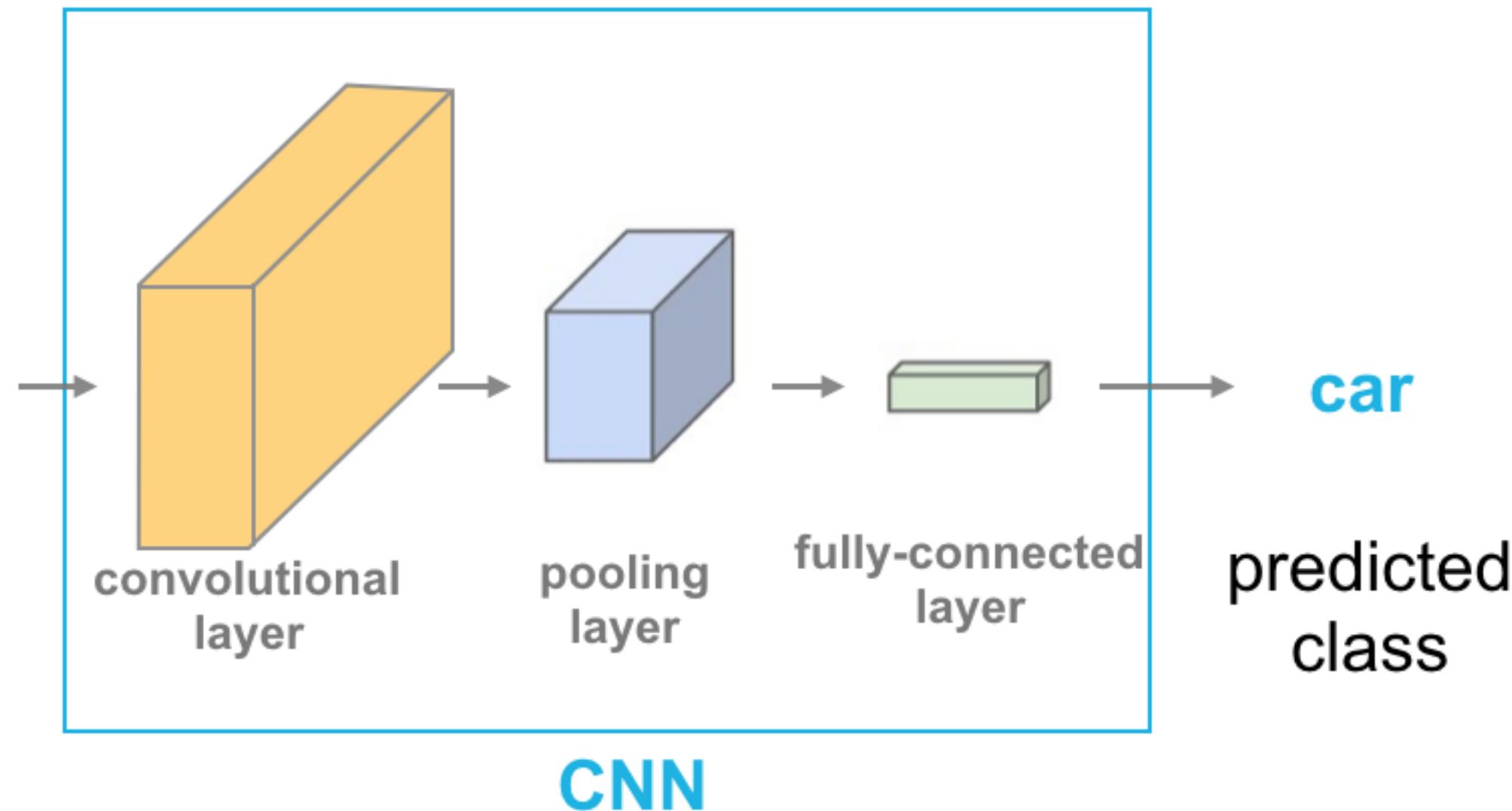


# Задачи компьютерного зрения

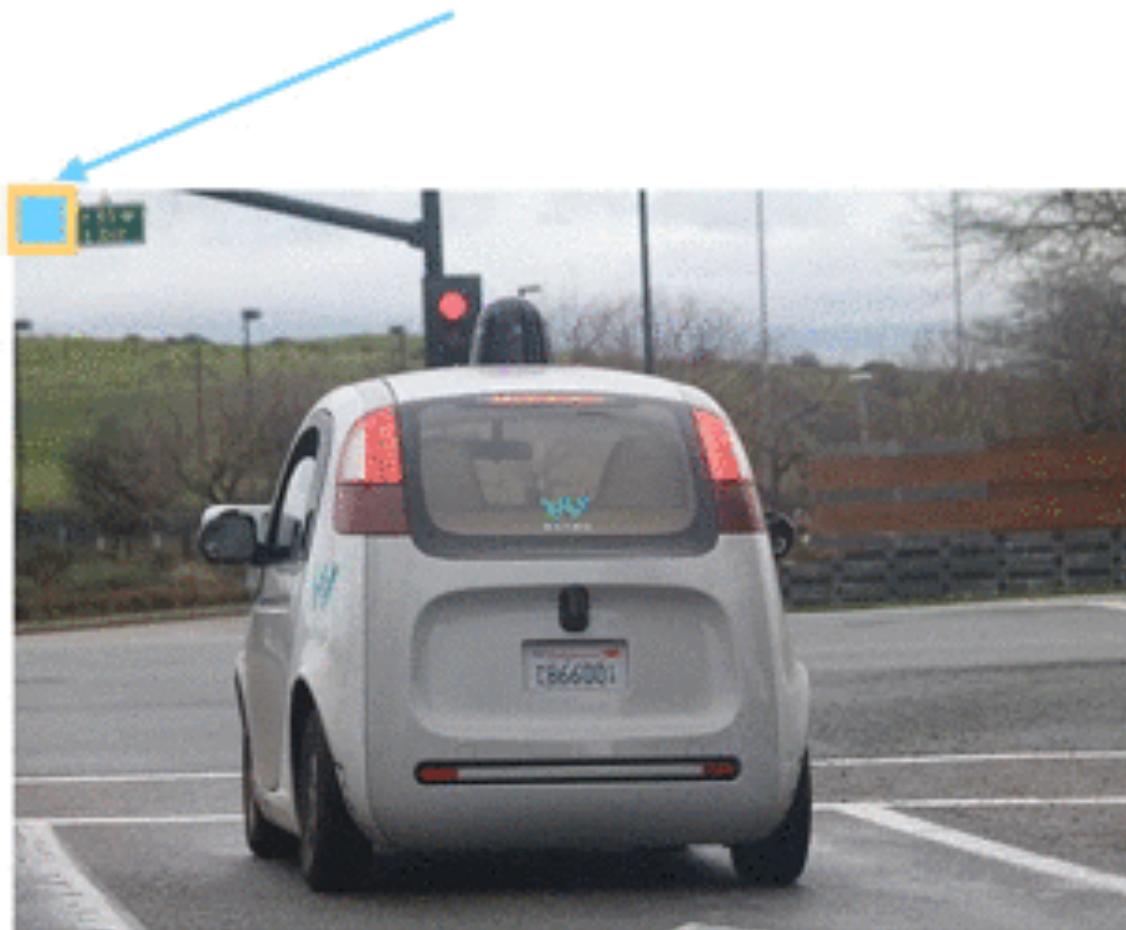
6 Ноября Tinkoff.Generation



input image



**convolutional kernel  
(image filter)**



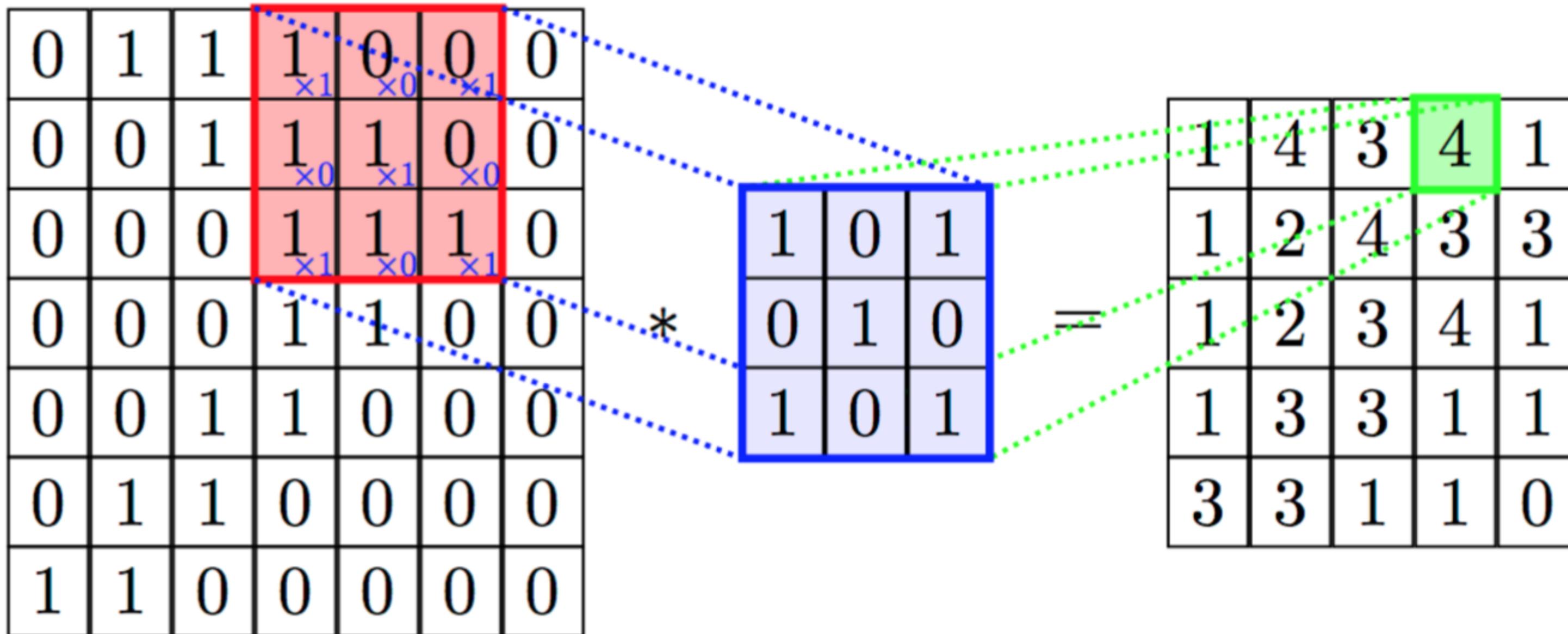
input image

Свёрточные сети – это просто нейронные сети, в которых вместо общей операции умножения на матрицу, по крайней мере в одном слое, используется операция свёртки.

$$(\mathbf{I} * \mathbf{G})[x, y] = \sum_{a=0}^{w-1} \sum_{b=0}^{h-1} \sum_{c \in \{1 \dots D\}} I_c[x + a, y + b] \cdot G_c[a, b]$$

**I** – вход размерностью  $W \times H \times D$

**G** – ядро(kernel/filter) размерностью  $w \times h \times D$



Выход  $L = I * G$  размерностью  $(W - w + 1) \times (H - h + 1) \times 1$

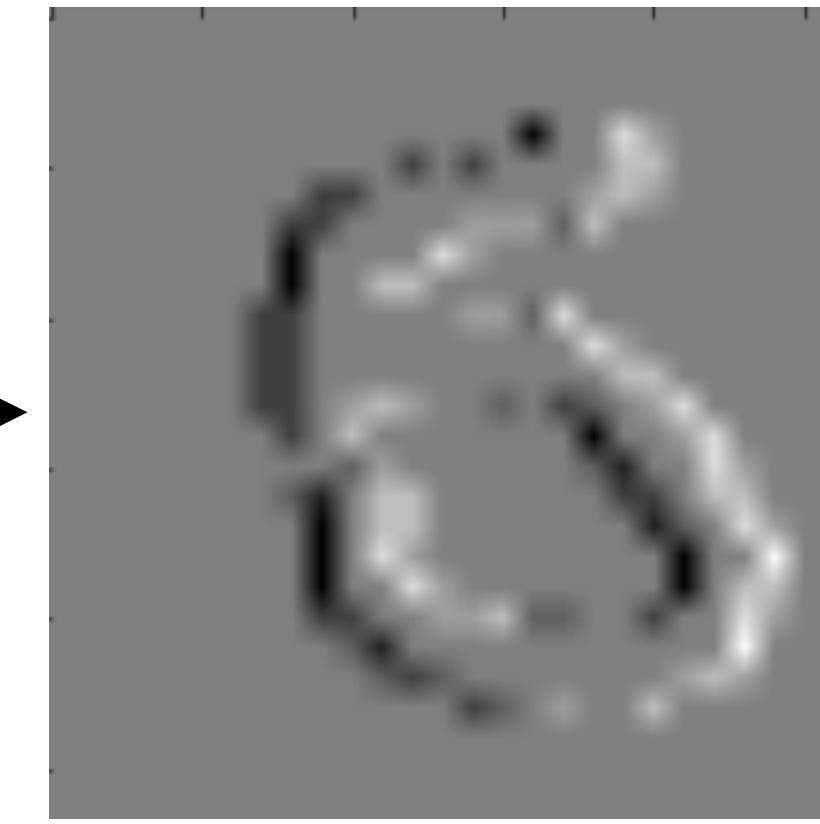


Remember this guy?

Плоский вектор размерностью 784



Свёртка [1 -1]



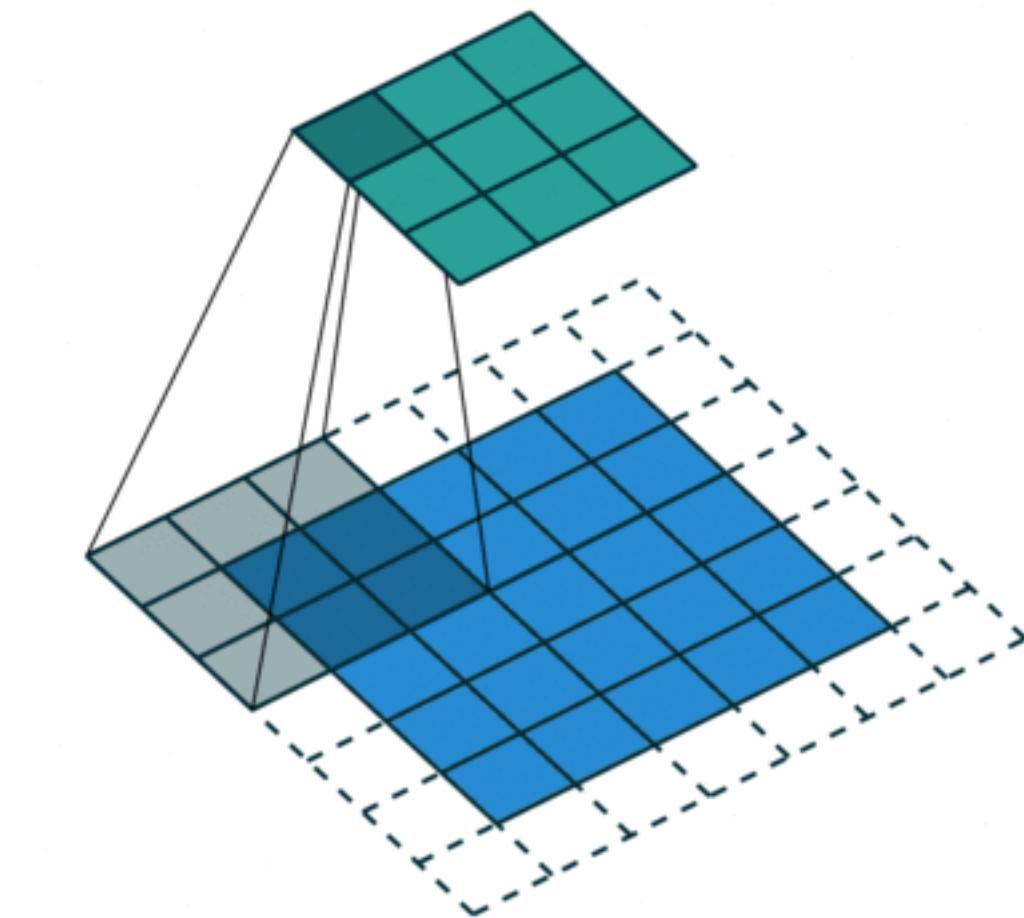
This is he now

Детекция угла  $\pi/4$

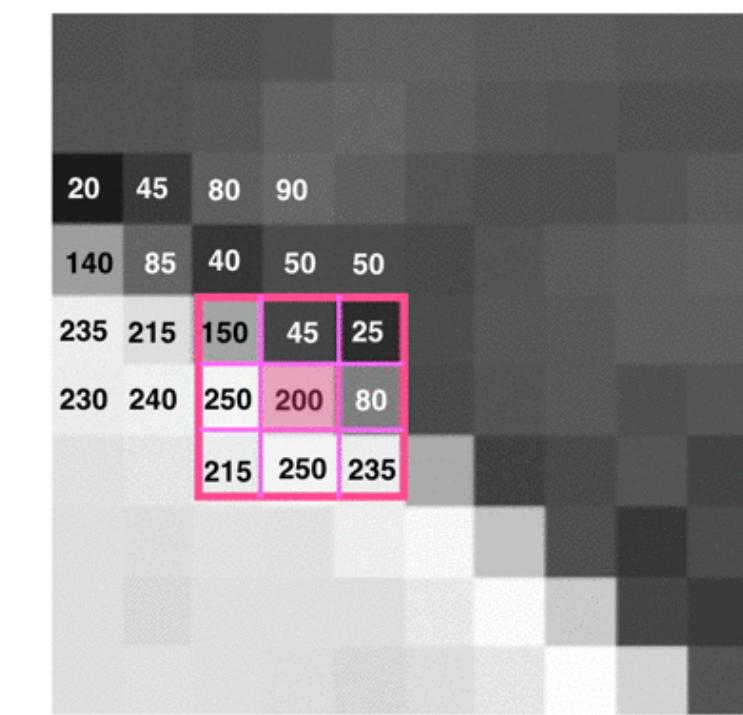
$$\begin{bmatrix} 0.6 & 0.2 & 0 \\ 0.2 & 0 & 0.2 \\ 0 & 0.2 & 0.6 \end{bmatrix}$$

# Какие у свёртки параметры?

- Количество ядер
- Размерность ядра
- Stride
- Padding

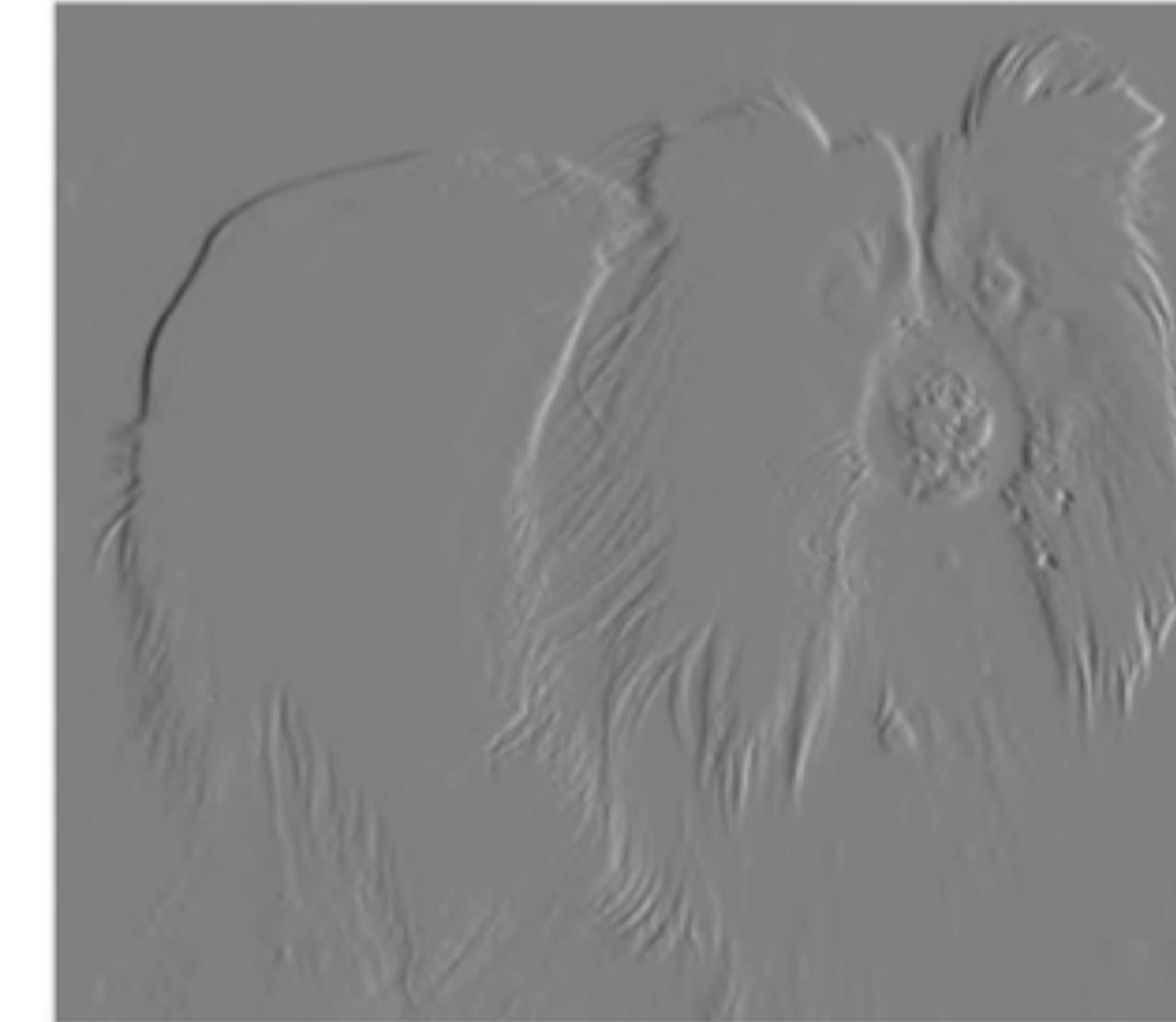


0	-1	0
-1	4	-1
0	-1	0



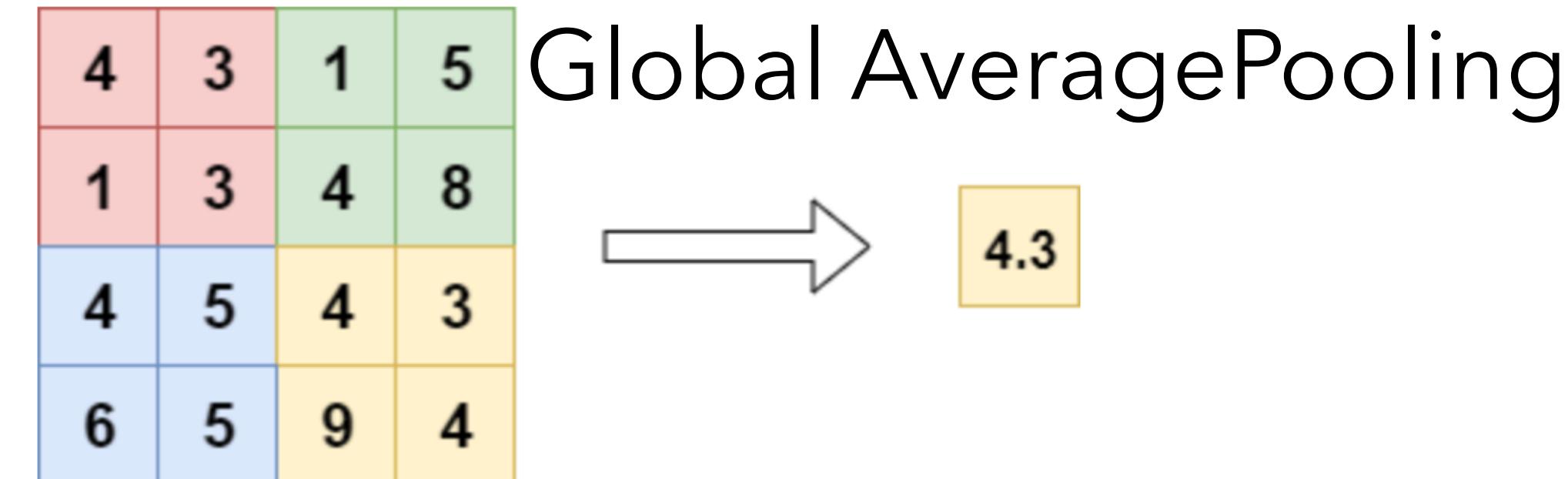
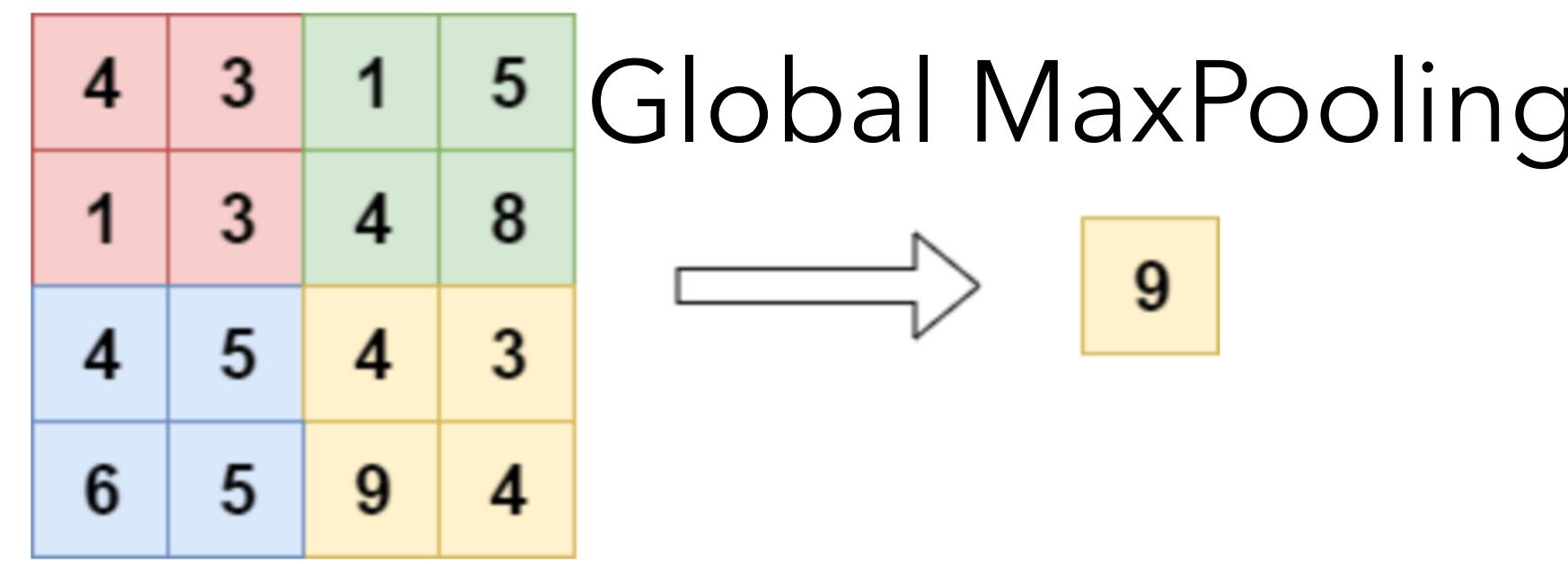
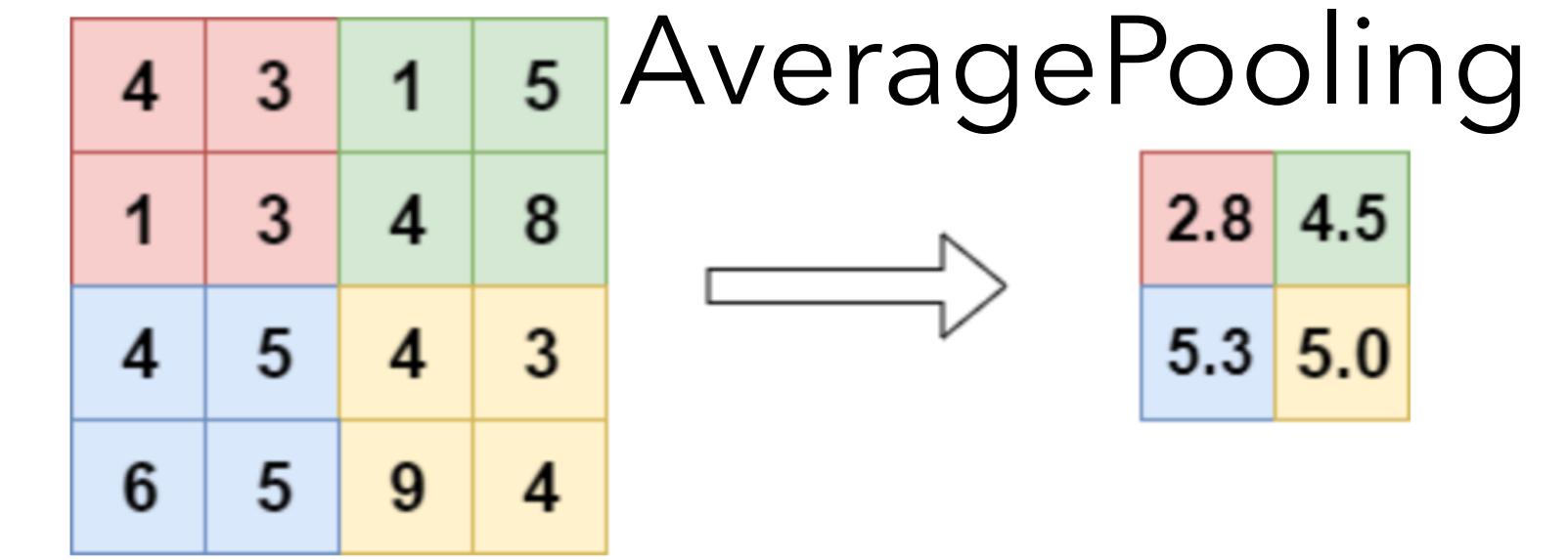
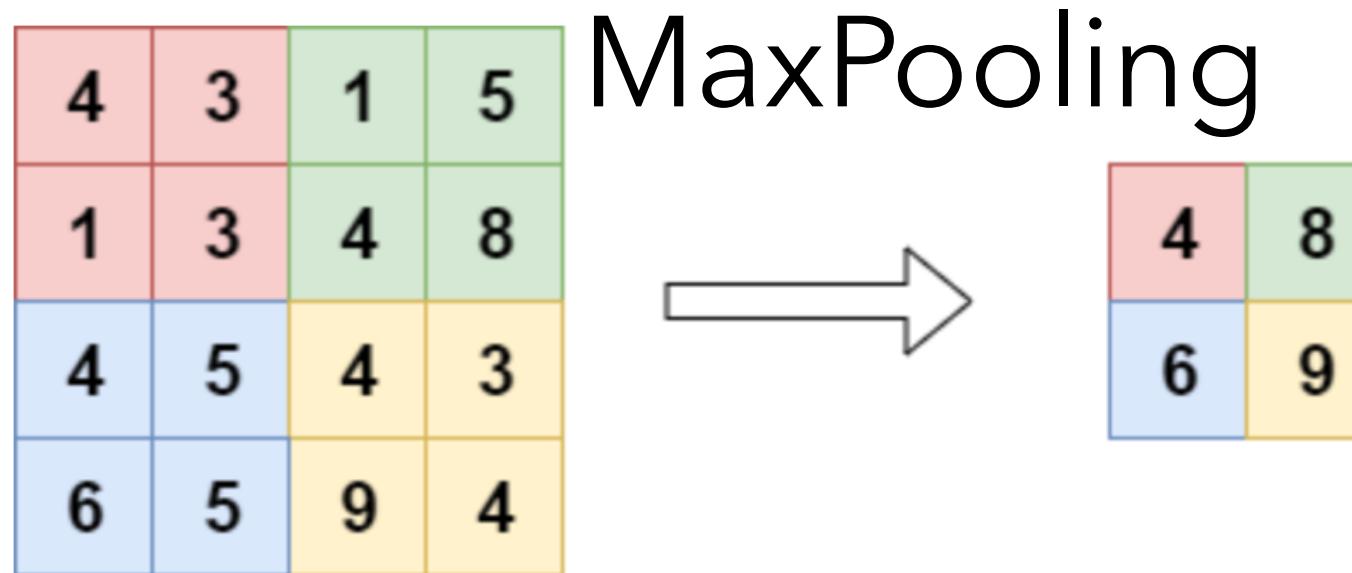
20	45	80	90	
140	85	40	50	50
235	215	150	45	25
230	240	250	200	80
		215	250	235

# Примеры свёрток



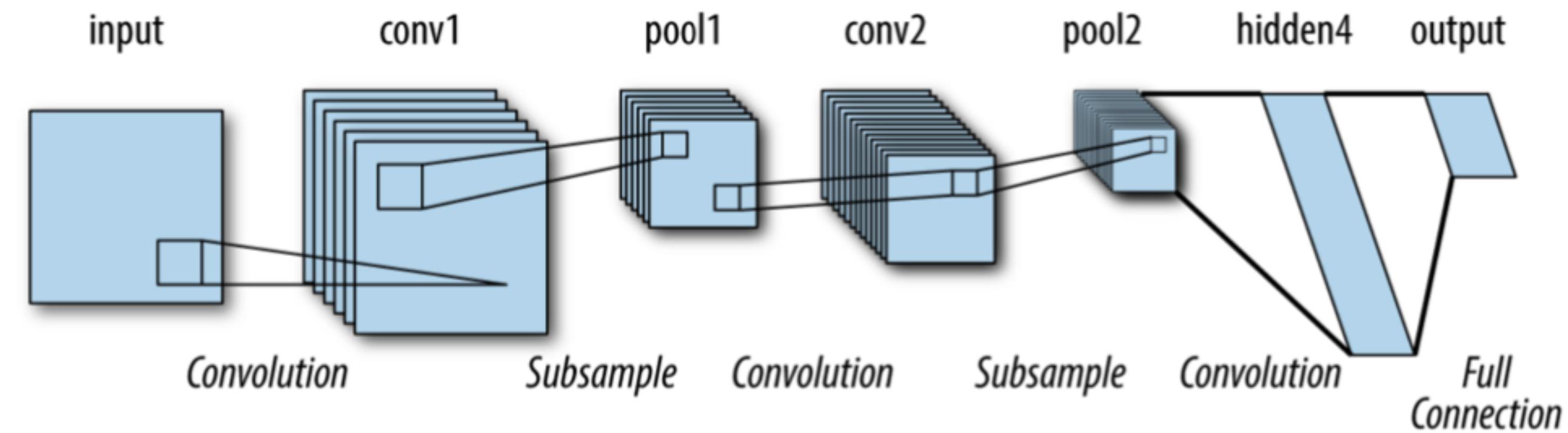
Выделение границ

- Что-то ещё?
- Да, **Pooling**



# Архитектуры

## LeNet

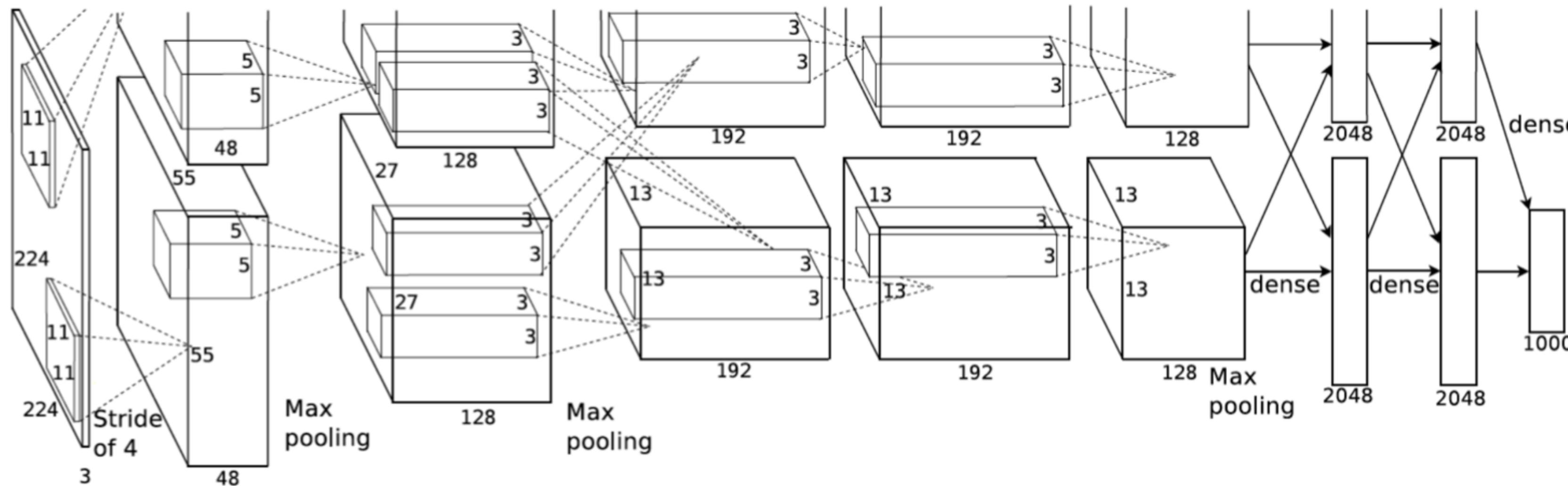


**LeCun et al, 1998, [yann.lecun.com](http://yann.lecun.com)**

Классификация рукописных символов

# Архитектуры

## AlexNet



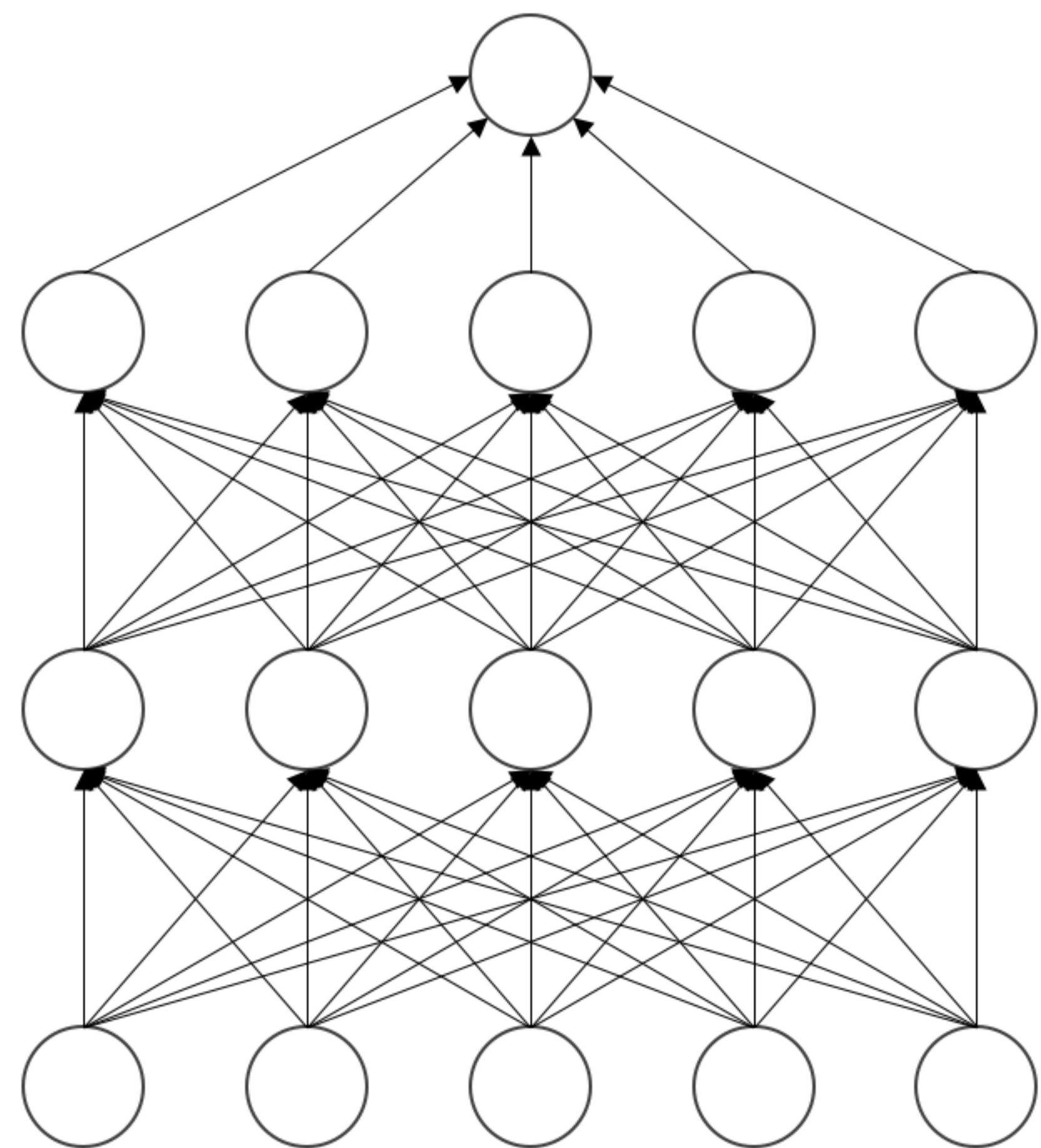
**Krizhevsky et al, 2012**

Разные свёртки

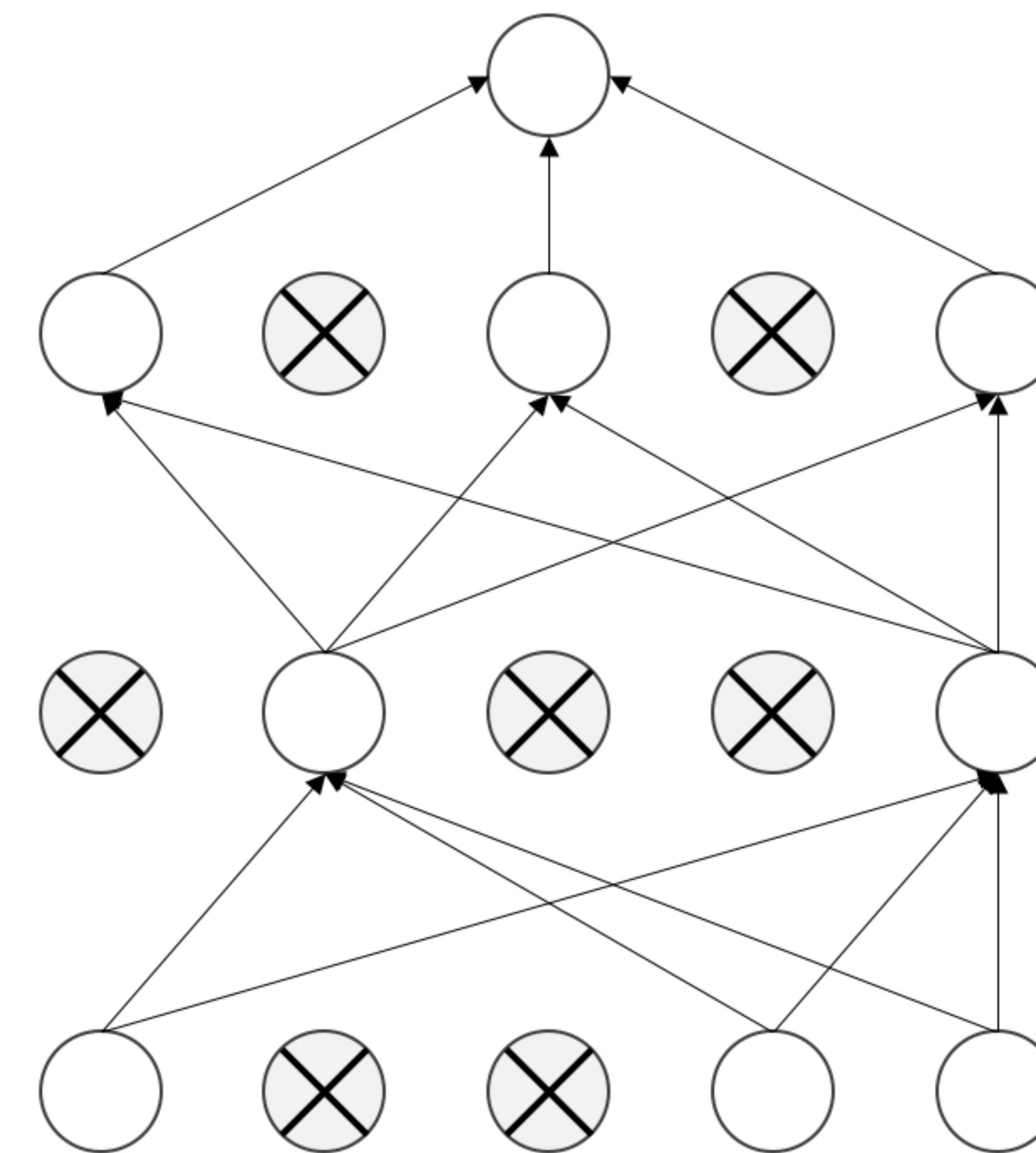
60 Миллионов весов

Есть Dropout

# Dropout



Standard Neural Net



After applying dropout



memegenerator.net

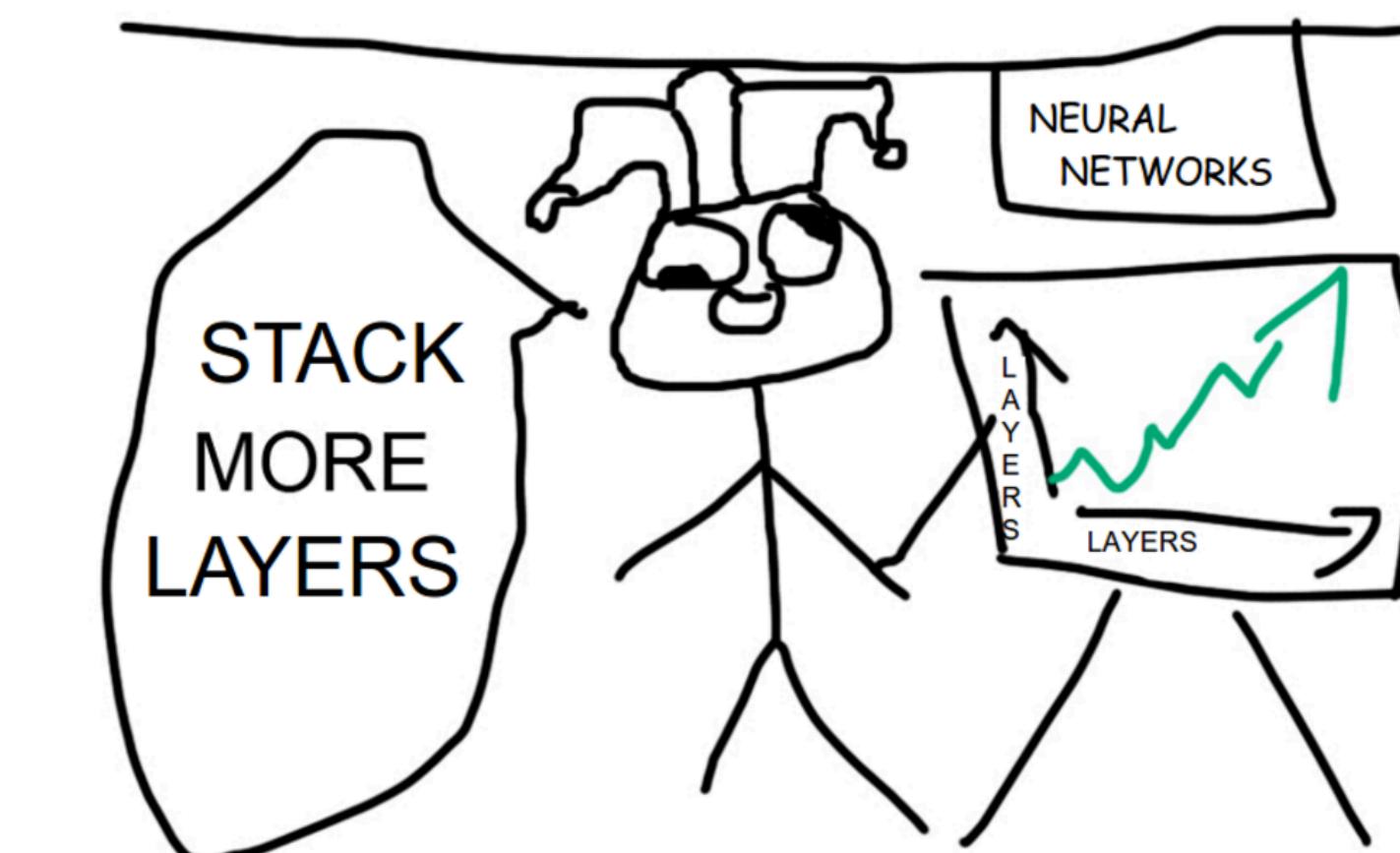
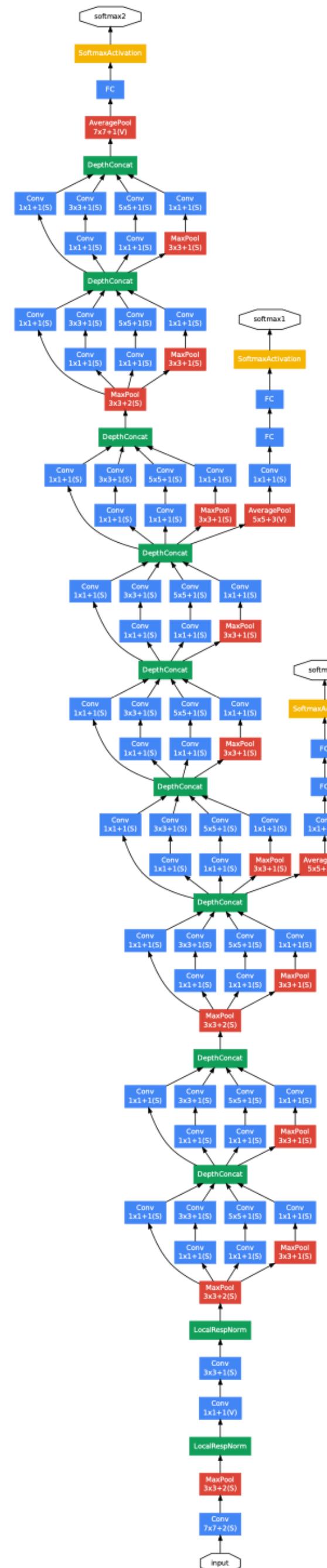
# Архитектуры

## GoogLeNet

**Szegedy et al, 2014**

Очень глубокая (даже на слайд не  
поместились...), но меньше весов, всего 5  
миллионов

Появились Inception Modules



# Inception block

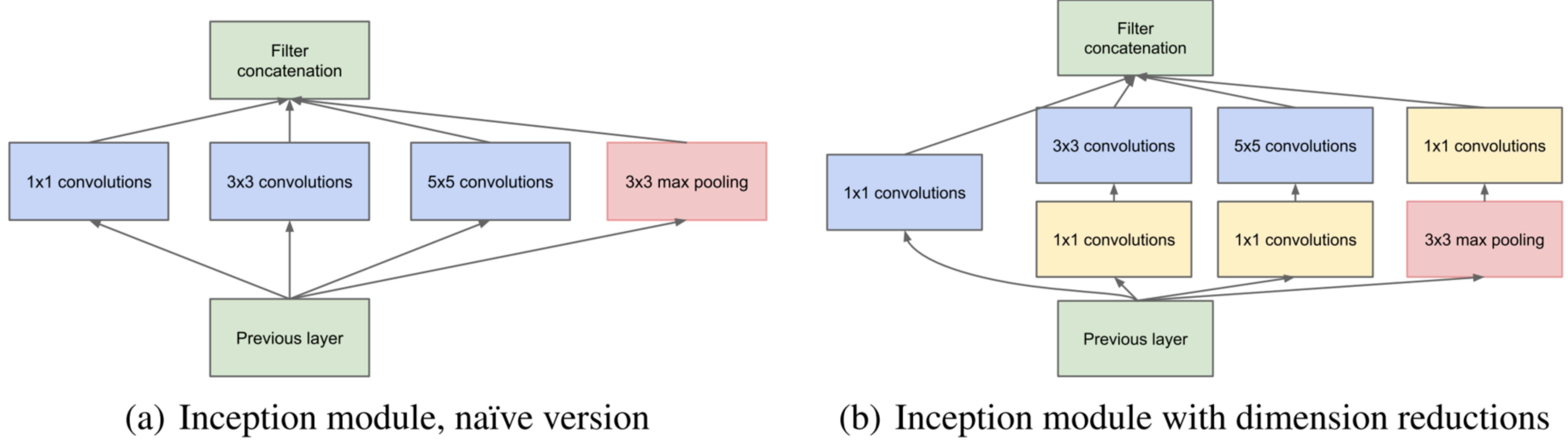
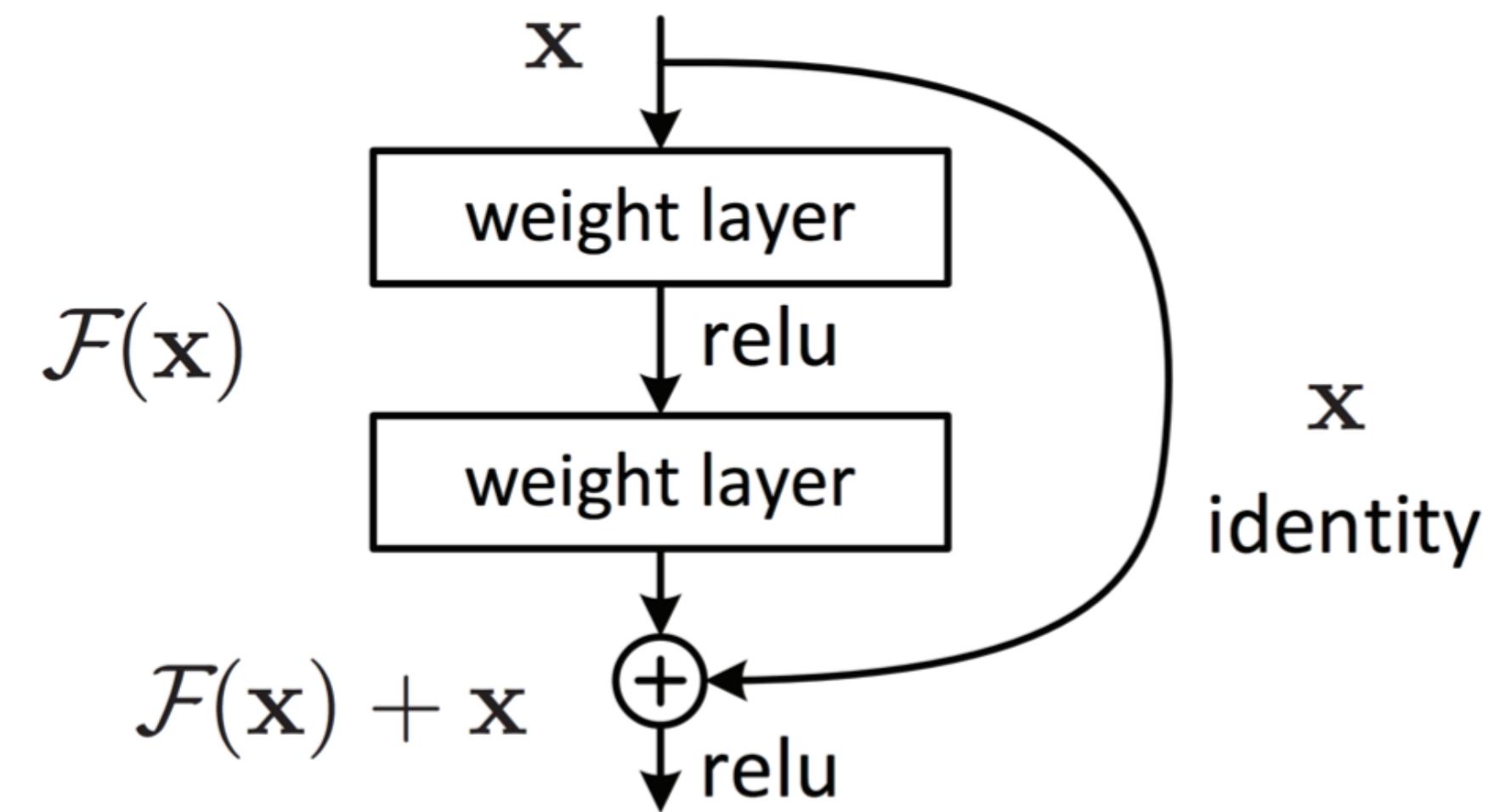


Figure 2: Inception module

# Архитектуры

## ResNet



**He et al, 2015**

152 Слоя!!

Решается проблема затухающего градиента

# А как проблема vanishing gradient то решается?

$$\frac{\partial L}{\partial x} = \frac{\partial L}{\partial Y} \frac{\partial Y}{\partial x}$$



**L** – Loss

**Y** – Выход со слоя

**x** – Вход

Когда применяем Residual Block,  
где **Y(x)=F(x)+x**, то получаем

$$\frac{\partial Y}{\partial x} = \frac{\partial F(x)}{\partial x} + 1$$

То есть за счёт этого  $+x$  появляется дефолтный градиент сигнал, по которому всегда можно сделать backprop

# Итого, с какими данными теперь можем работать

	<b>Одноканальные</b>	<b>Многоканальные</b>
1D	Аудиосигналы: ось свертки соответствует времени. Мы дискретизируем время и измеряем амплитуду сигнала один раз в каждом временном интервале	Данные анимации «скелета»: анимации трехмерного отрисовываемого компьютером персонажа генерируются путем изменения положения «скелета» со временем. В каждый момент положение скелета описывается углами сочленения костей в каждом суставе. Каждый канал данных, подаваемых на вход сверточной сети, представляет угол относительно одной оси одного сустава
2D	Аудиоданные, предварительно обработанные с помощью преобразования Фурье. Мы можем преобразовать аудиосигнал в двумерный тензор, строки которого соответствуют частотам, а столбцы – моментам времени. Применение свертки по времени делает модель эквивариантной относительно временных сдвигов. Применение свертки по оси частот делает модель эквивариантной относительно частоты, т. е. одна и та же мелодия в разных октавах порождает на выходе сети одно и то же представление, но с разной высотой	Данные цветного изображения: один канал содержит красные пиксели, другой – зеленые, третий – синие. Ядро свертки сдвигается по двум осям изображения, обеспечивая эквиварантность относительно параллельного переноса в обоих направлениях
3D	Объемные данные: типичным источником таких данных являются технологии медицинской интроскопии, например компьютерной томографии	Данные цветного видео: одна ось соответствует времени, другая – высоте кадра, третья – ширине кадра

# Задачи компьютерного зрения

**1. Задачи  
классификации**

**3. Задачи  
сегментации**

**5. Задачи  
captioning**

**2. Задачи  
детекции**

**4. Задачи  
идентификации  
и матчинга**

# Задачи классификации

# Классический вариант, single label

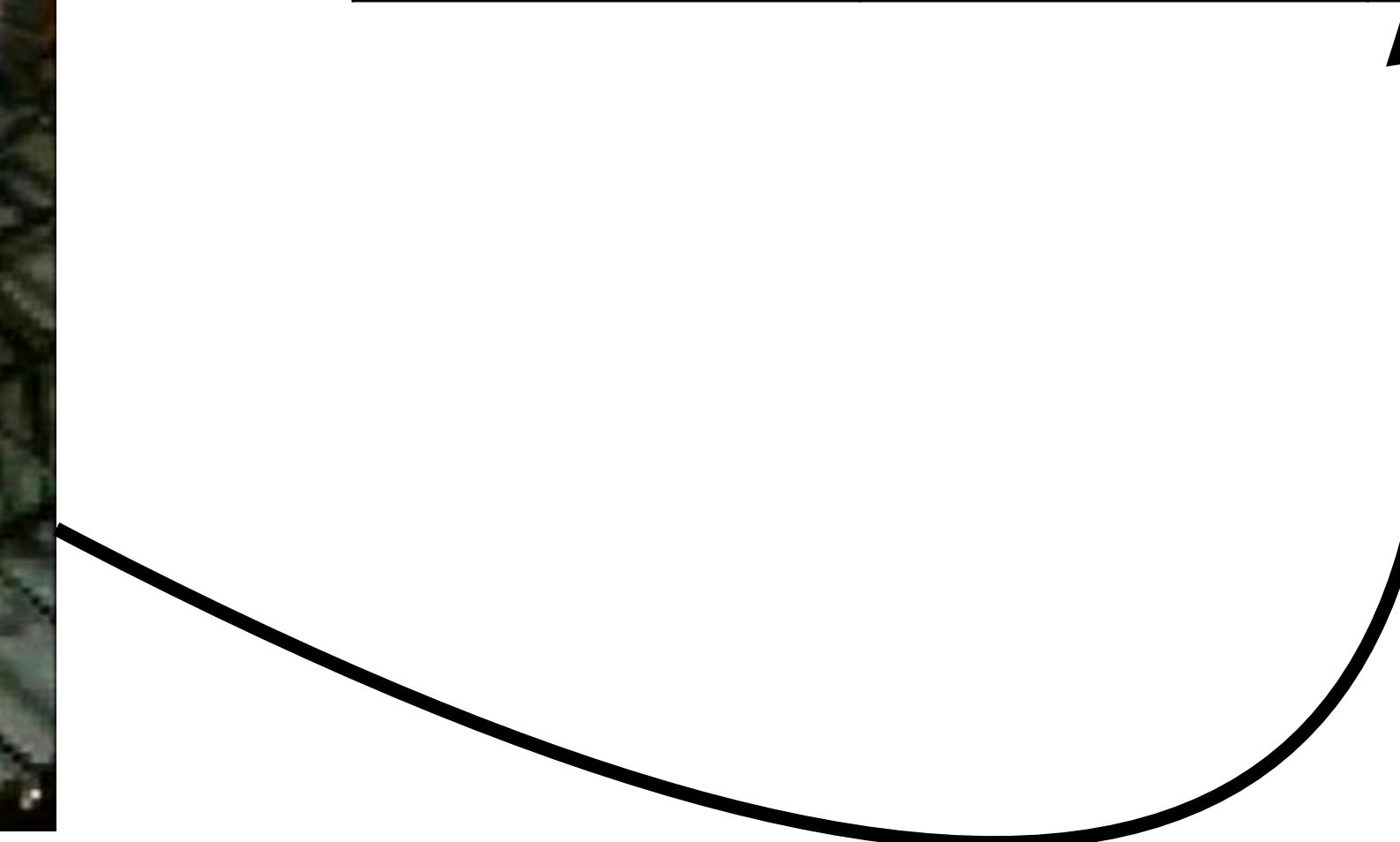


Класс	Вероятность принадлежности
Кот	0.70
Хлеб	0.20
...	...

# Навороченный вариант, multi label



Животное	Млекопитаю	Кот	Коричневый
Машина	Рыба	Собака	Глубой
...	...	...	...



# Общий pipeline



Данные:

Cat
Cat
Dog
Cat
...

# Общий pipeline

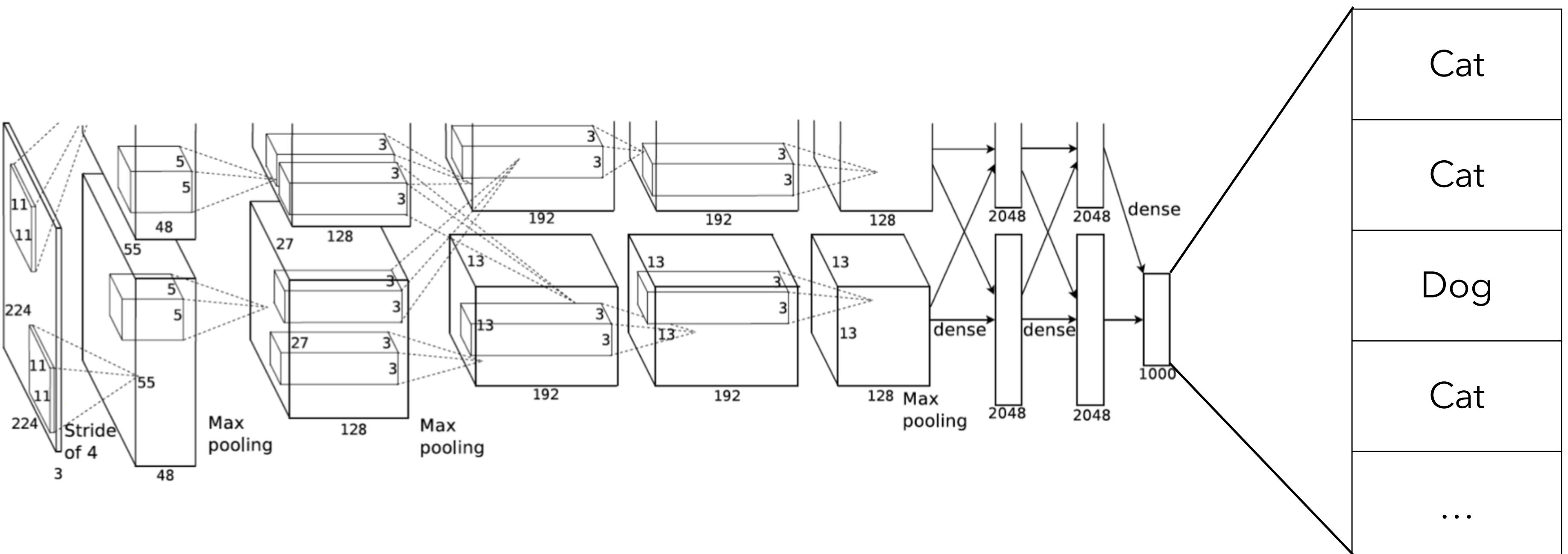


# Преобразование данных:



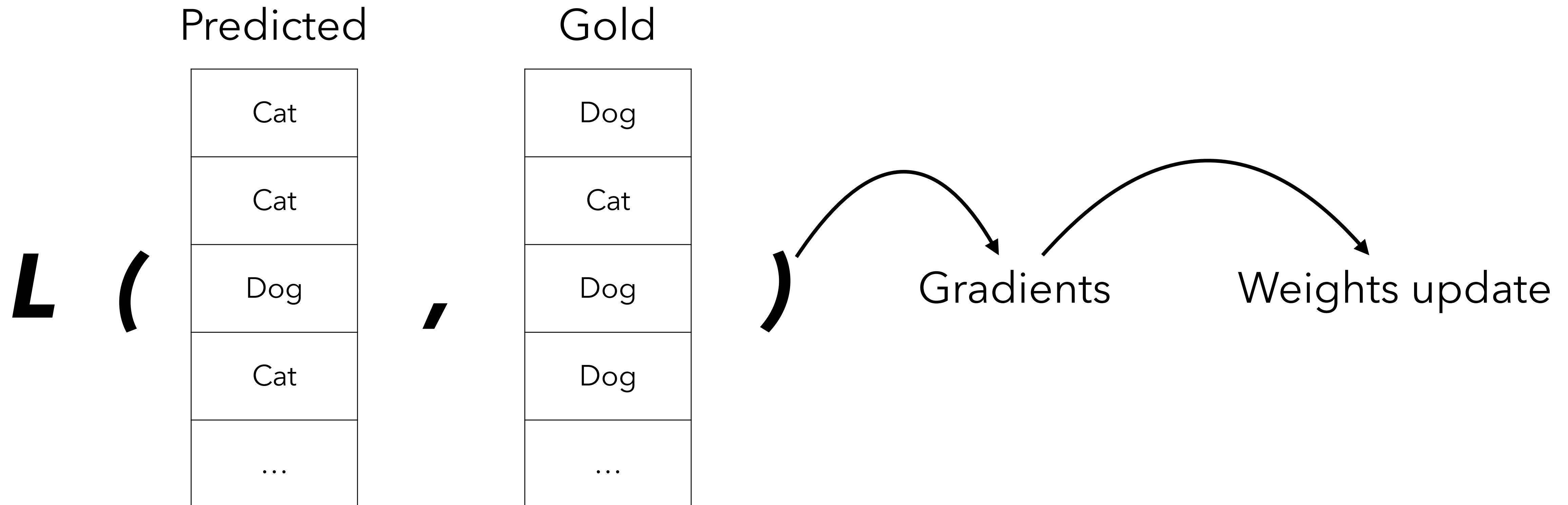
# Общий pipeline

Построение архитектуры:



# Общий pipeline

# Обучение:



# Общий pipeline

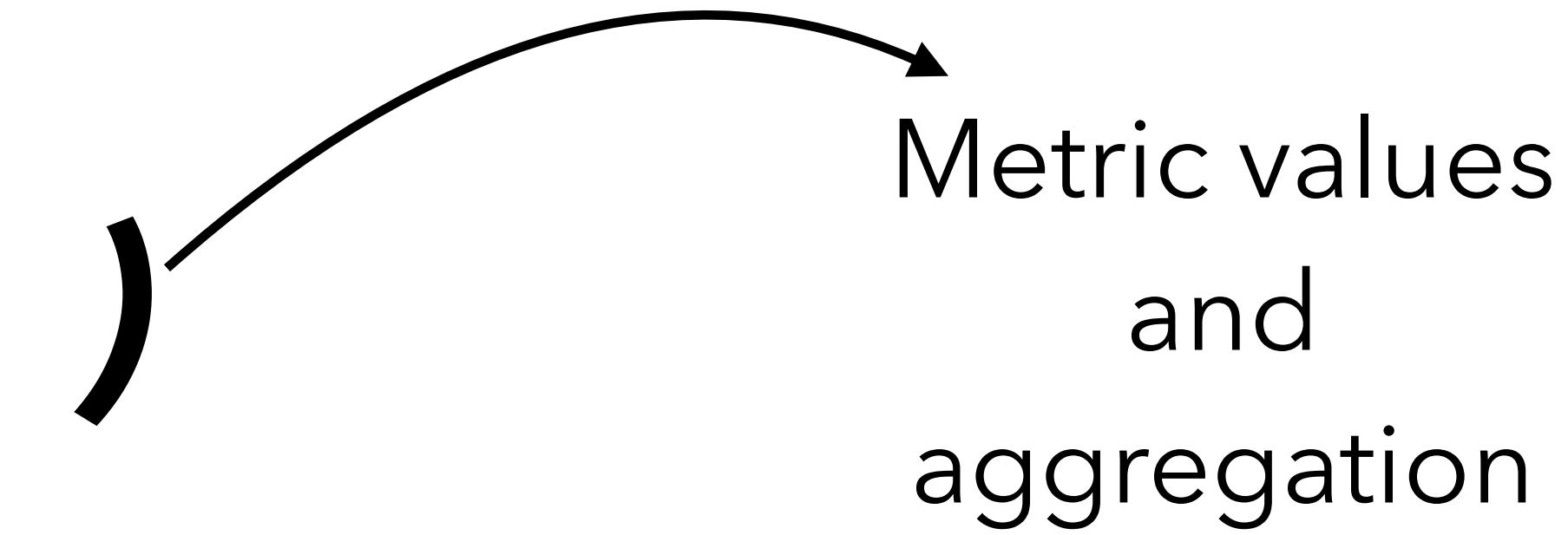
**M** (

Predicted
Cat
Cat
Dog
Cat
...

,

Gold
Dog
Cat
Dog
Dog
...

## Evaluation:



# Общий pipeline

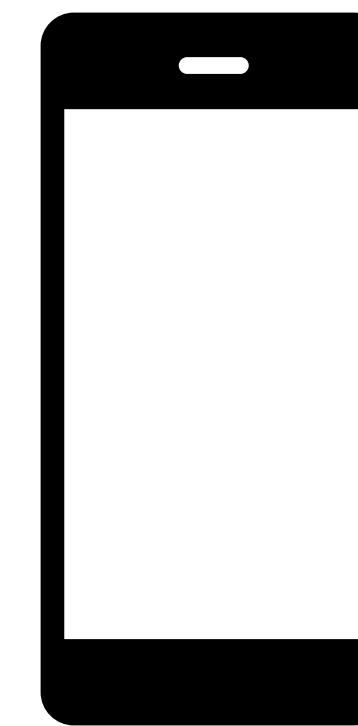
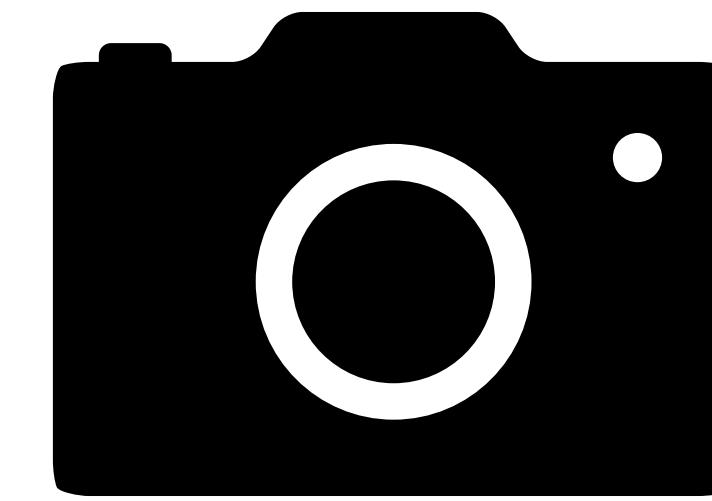
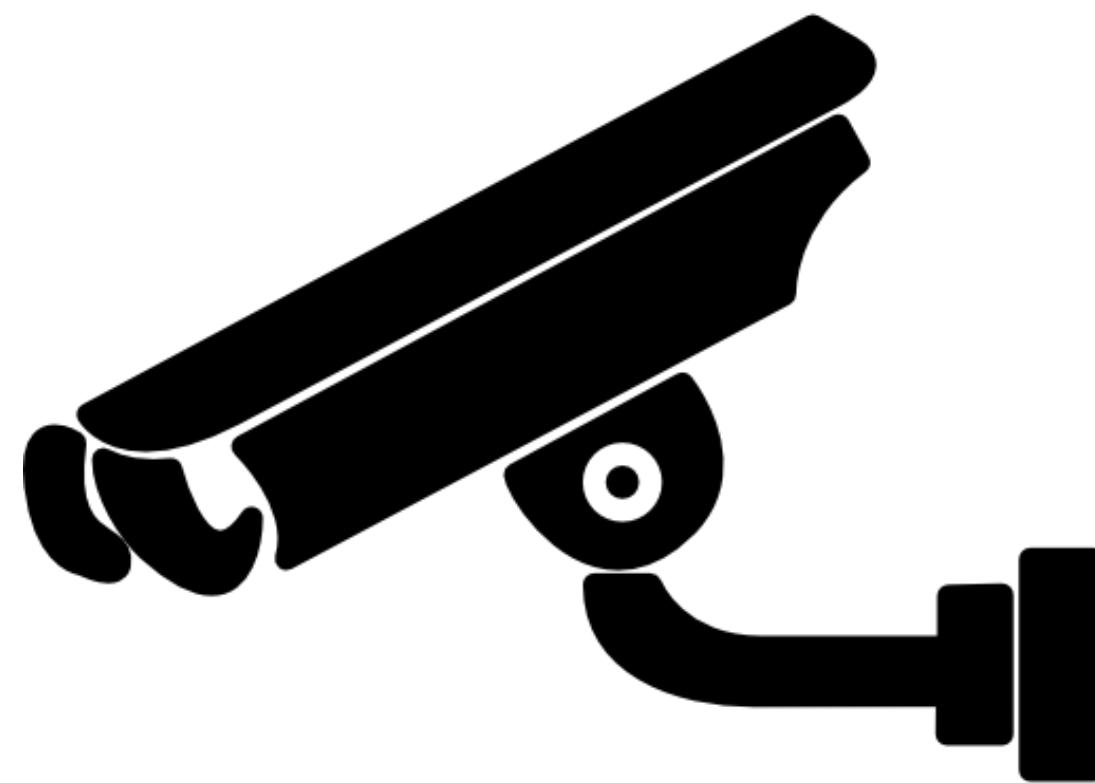
## Evaluation:

<b>Team</b>	<b>Year</b>	<b>Place</b>	<b>Error (top-5)</b>	<b>Uses external data</b>
SuperVision	2012	1st	16.4%	no
SuperVision	2012	1st	15.3%	Imagenet 22k
Clarifai	2013	1st	11.7%	no
Clarifai	2013	1st	11.2%	Imagenet 22k
MSRA	2014	3rd	7.35%	no
VGG	2014	2nd	7.32%	no
GoogLeNet	2014	1st	6.67%	no

Table 2: Classification performance

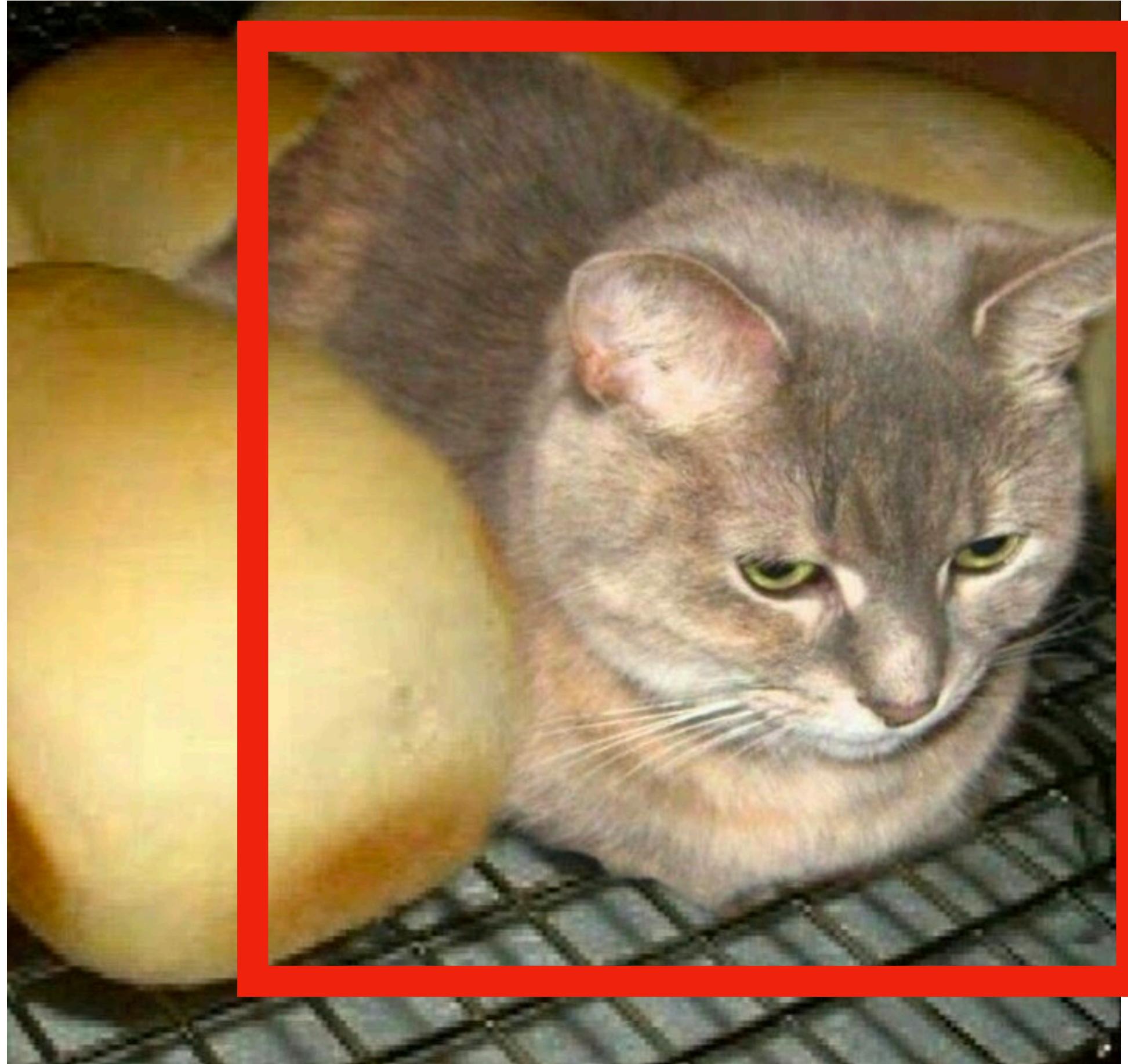
# Общий pipeline

Inference:



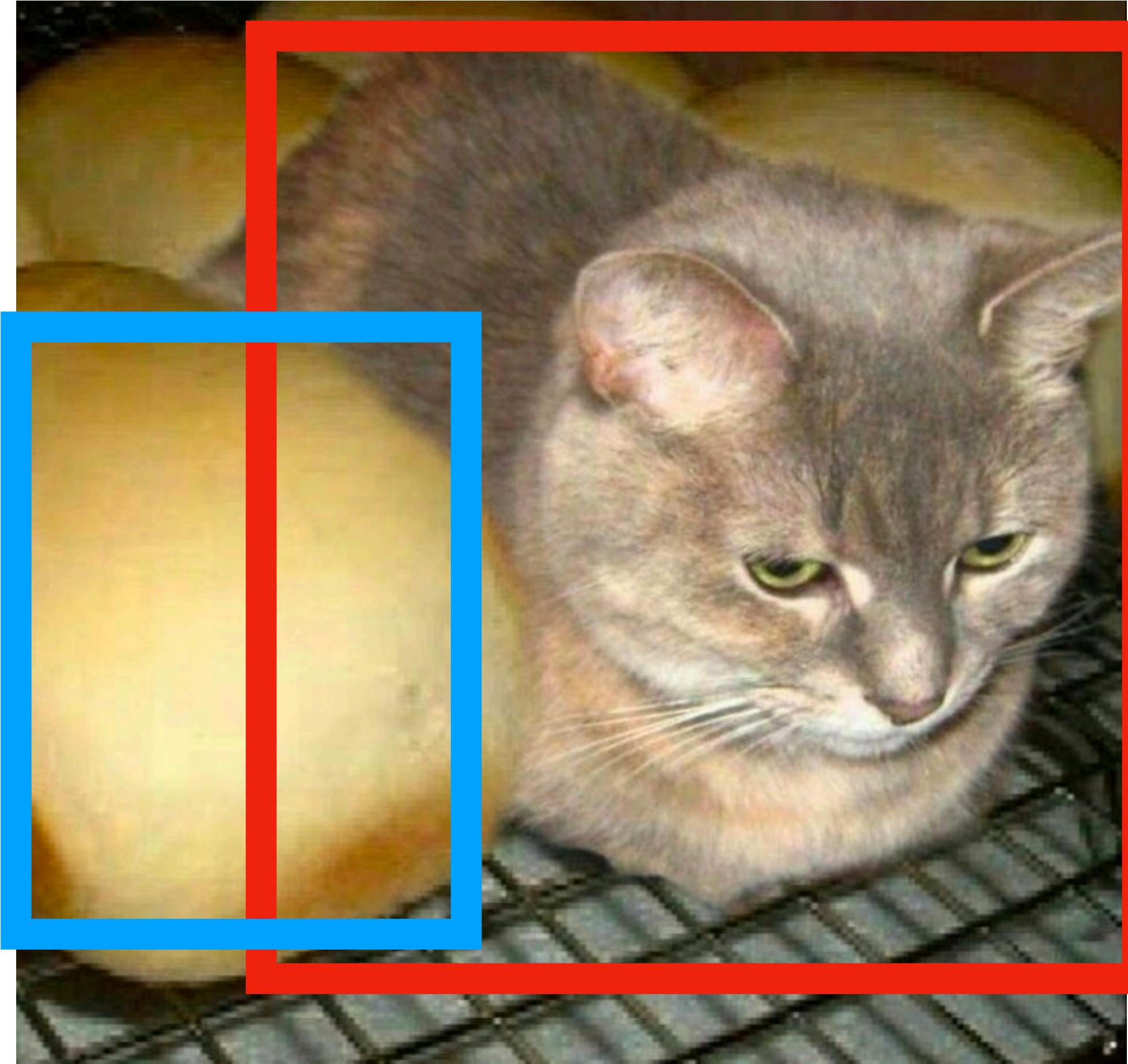
# Задачи детекции и локализации

# Localisation

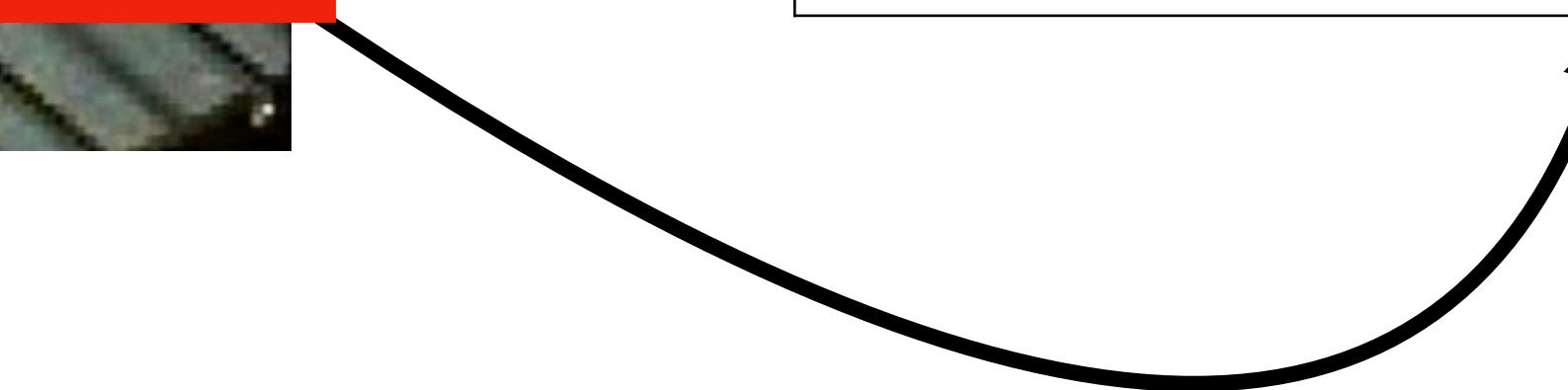


Ограничивающий прямоугольник	Значение
X_1	100
Y_2	200
...	...

# Classification + localisation



Ограничивающий прямоугольник	Класс
$[x, y, w, h]$	Cat
$[x, y, w, h]$	Dog
...	...



# Расчёт ошибки

Predicted

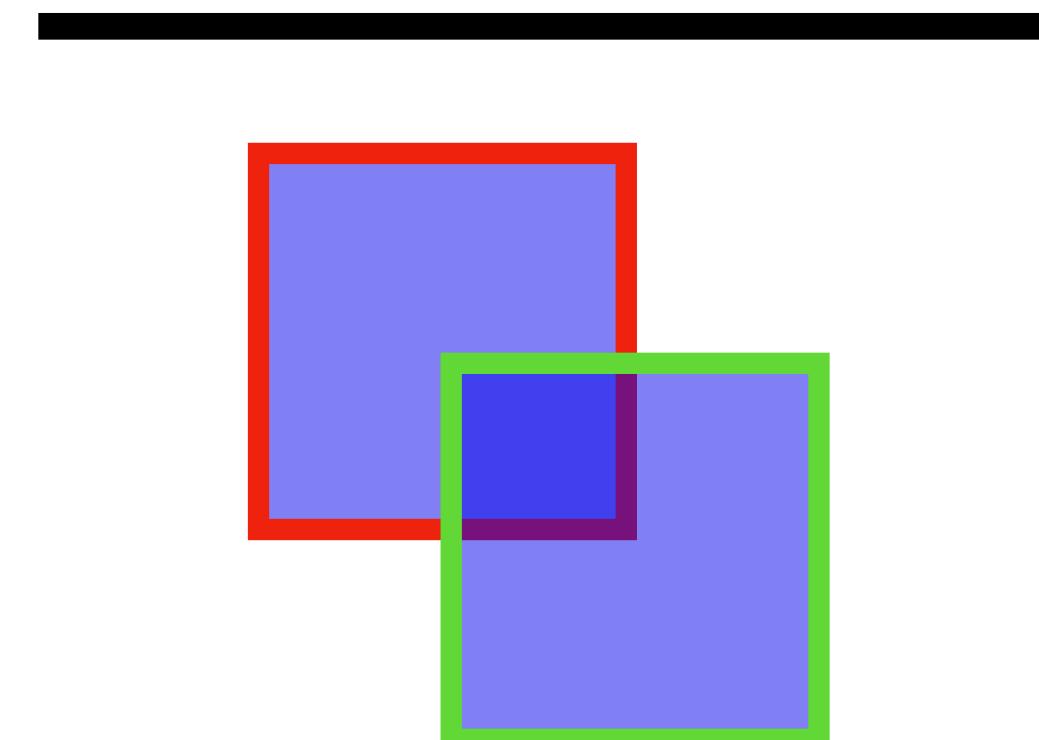
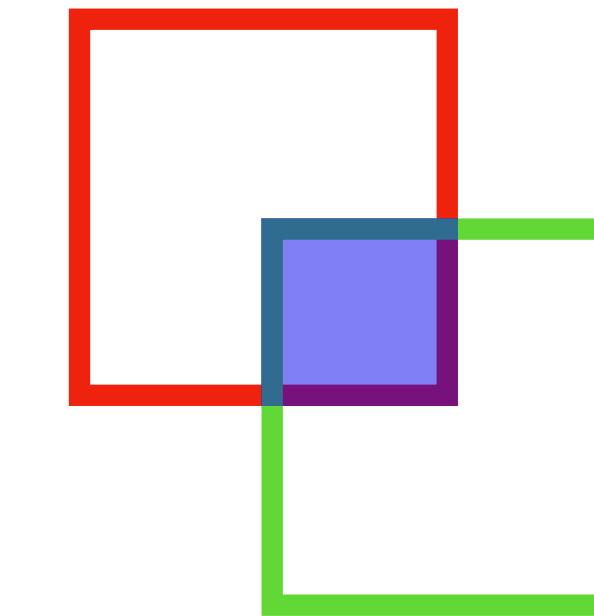
Ограничиваю щий прямоугольник , (b, boxes)	Класс (l, class labels)
[x, y, w, h]	Cat
[x, y, w, h]	Dog
...	...

Gold

Ограничиваю щий прямоугольник , (z, gt boxes)	Класс (g, gt class labels)
[x, y, w, h]	Cat
[x, y, w, h]	Dog
...	...

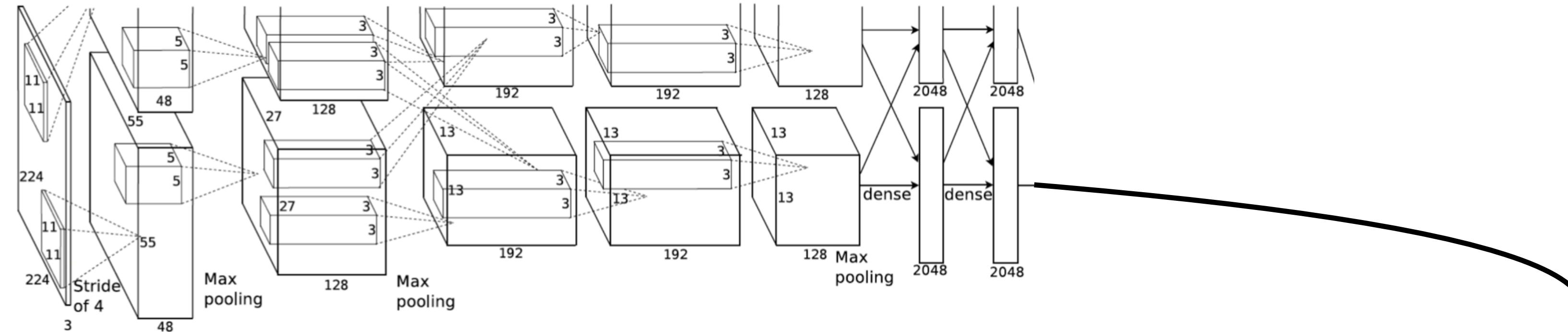
$$e = \frac{1}{n} \cdot \sum_k \min_j \min_m^M \max\{d(l_j, g_k), f(b_j, z_{km})\},$$

IoU



# Общий pipeline

# Построение архитектуры:

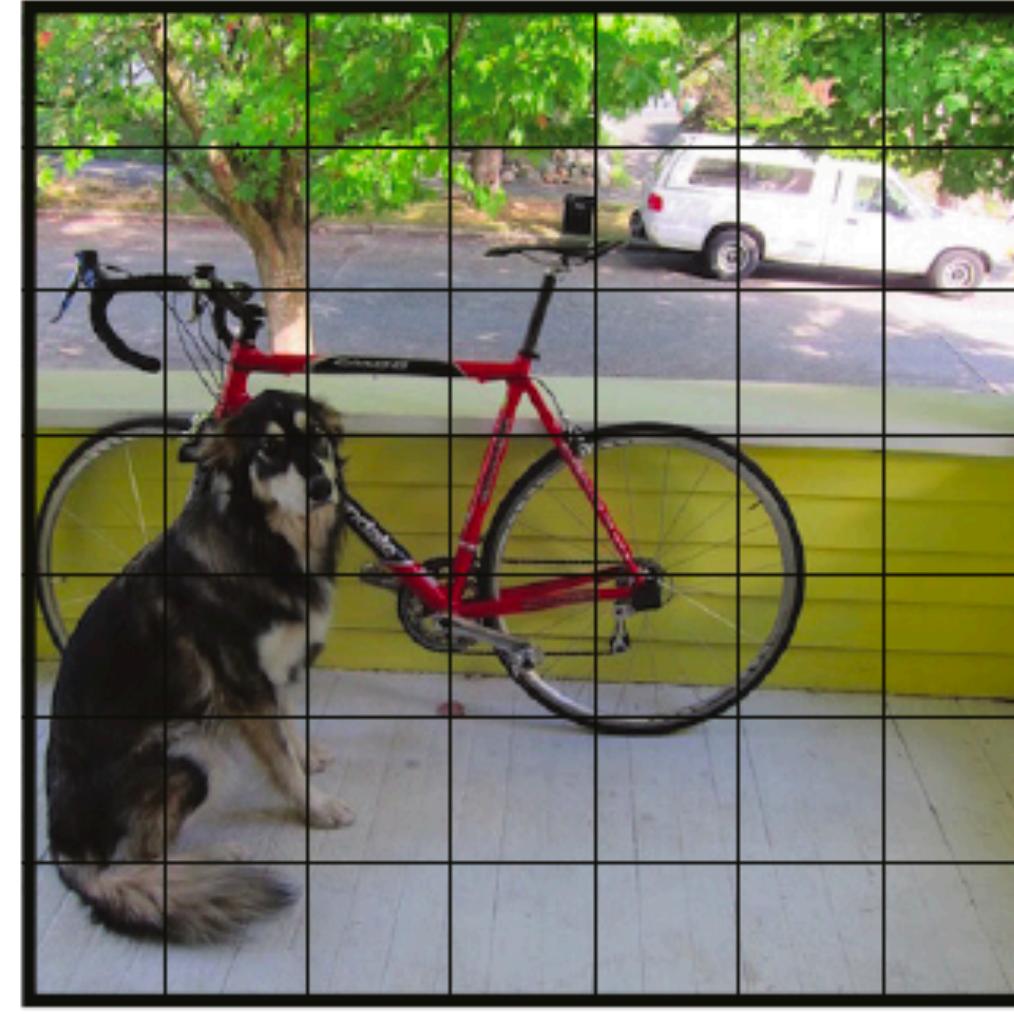


Classification head

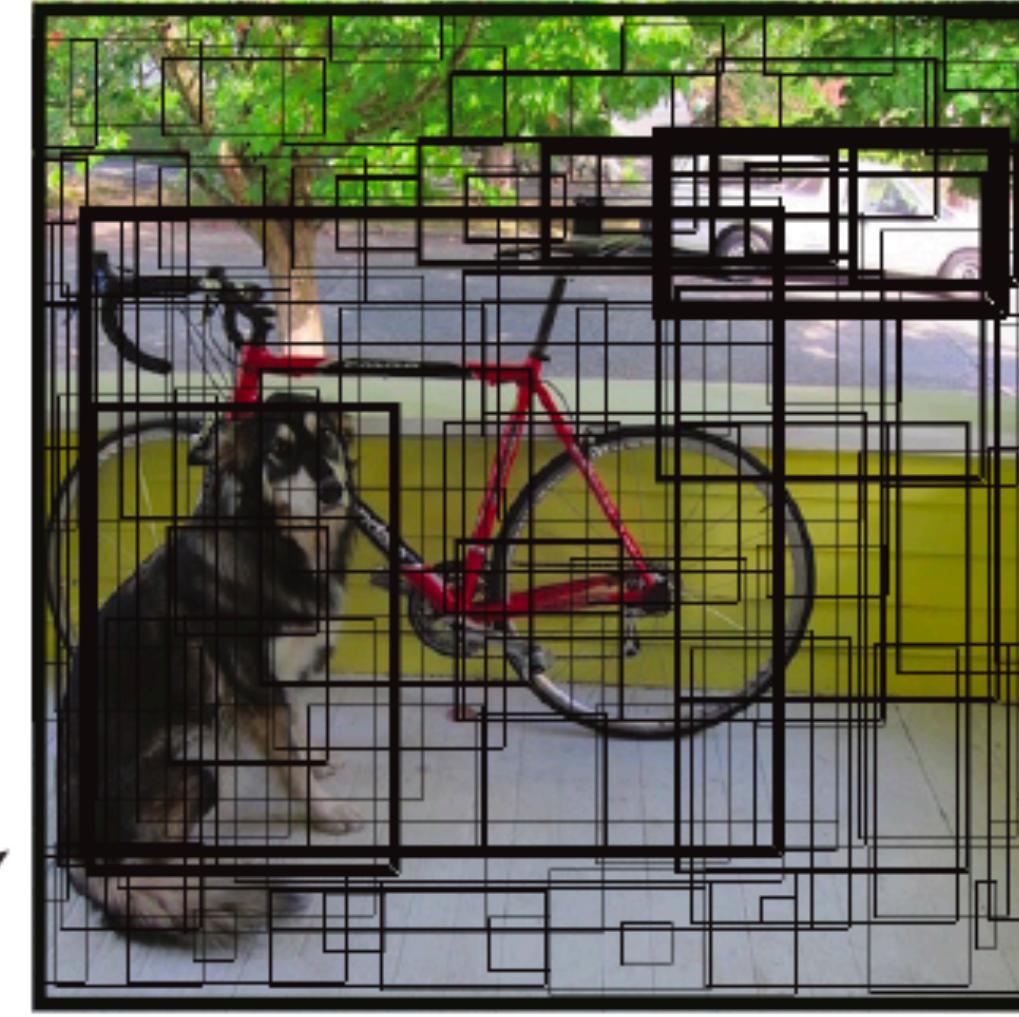
V  
e  
c  
t  
o  
r  
4  
0  
9  
6

Regression head

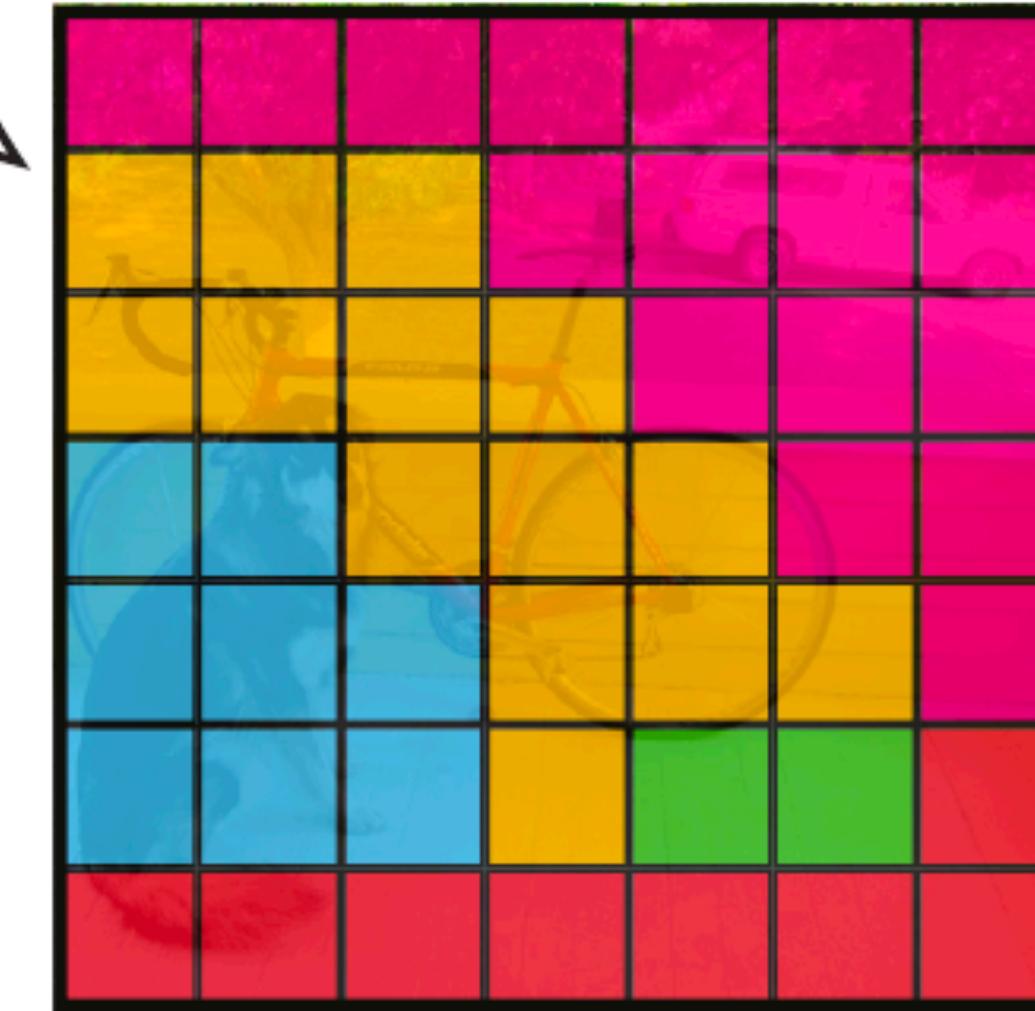
Ограничивающий прямоугольник, (b, boxes)	Класс (l, class labels)
[x, y, w, h]	Cat
[x, y, w, h]	Dog
...	...



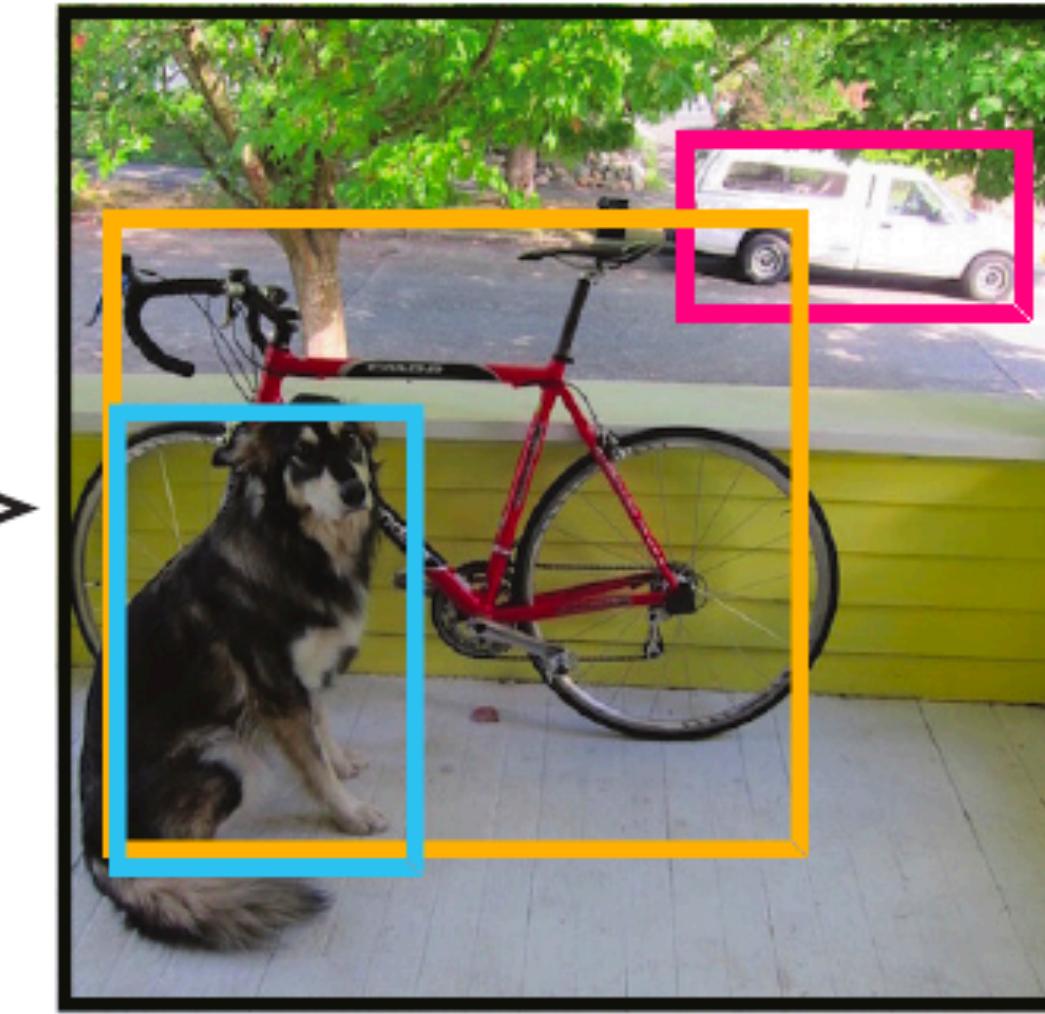
Входное изображение  
с решеткой  $S \times S$



ВВох-ы и уверенность

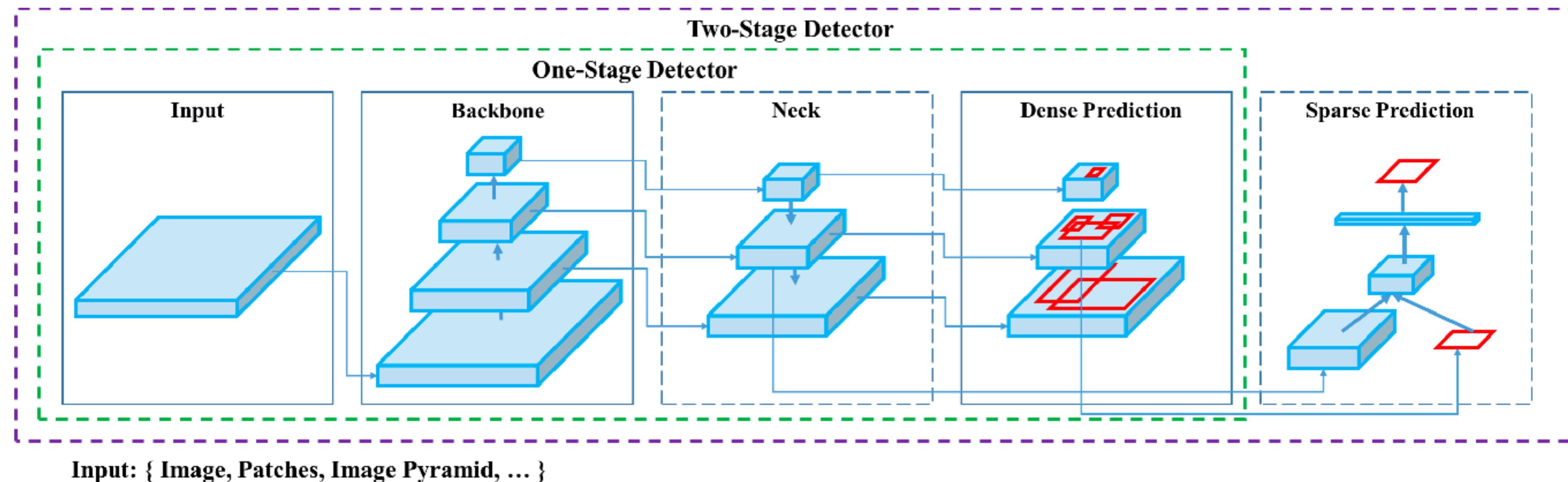


Вероятности классов для ячеек

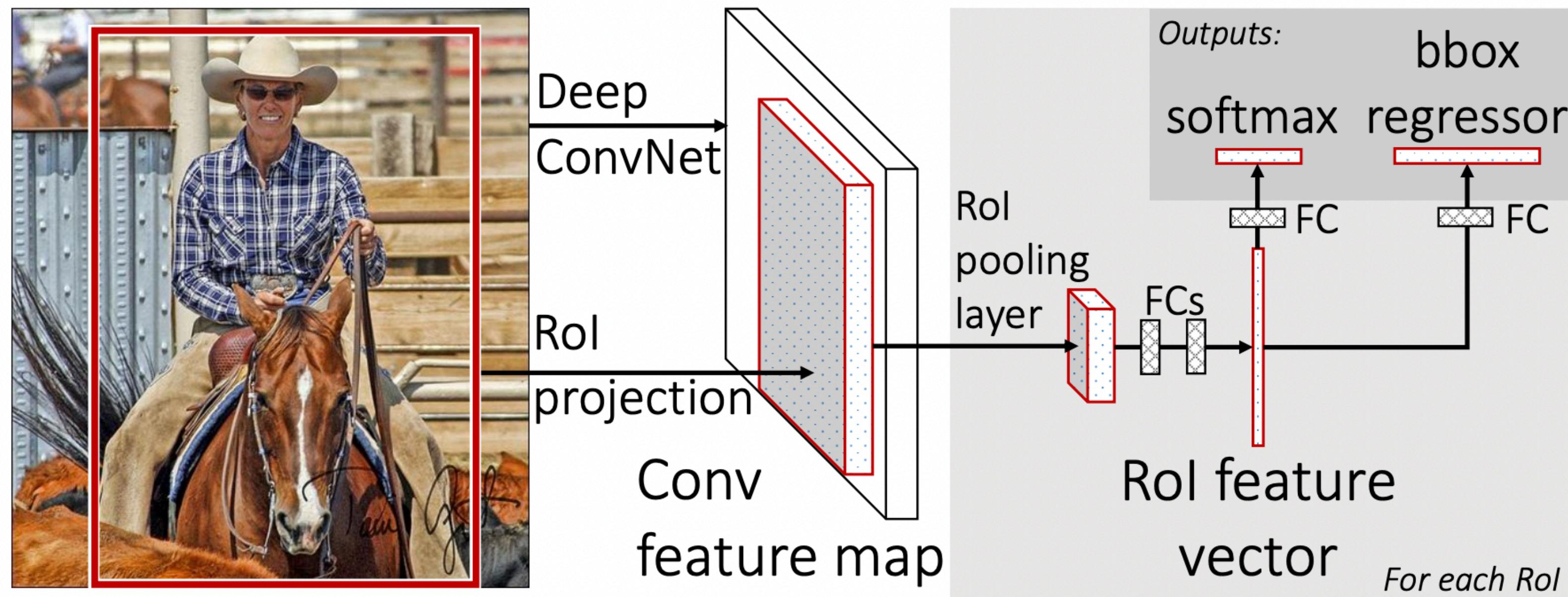


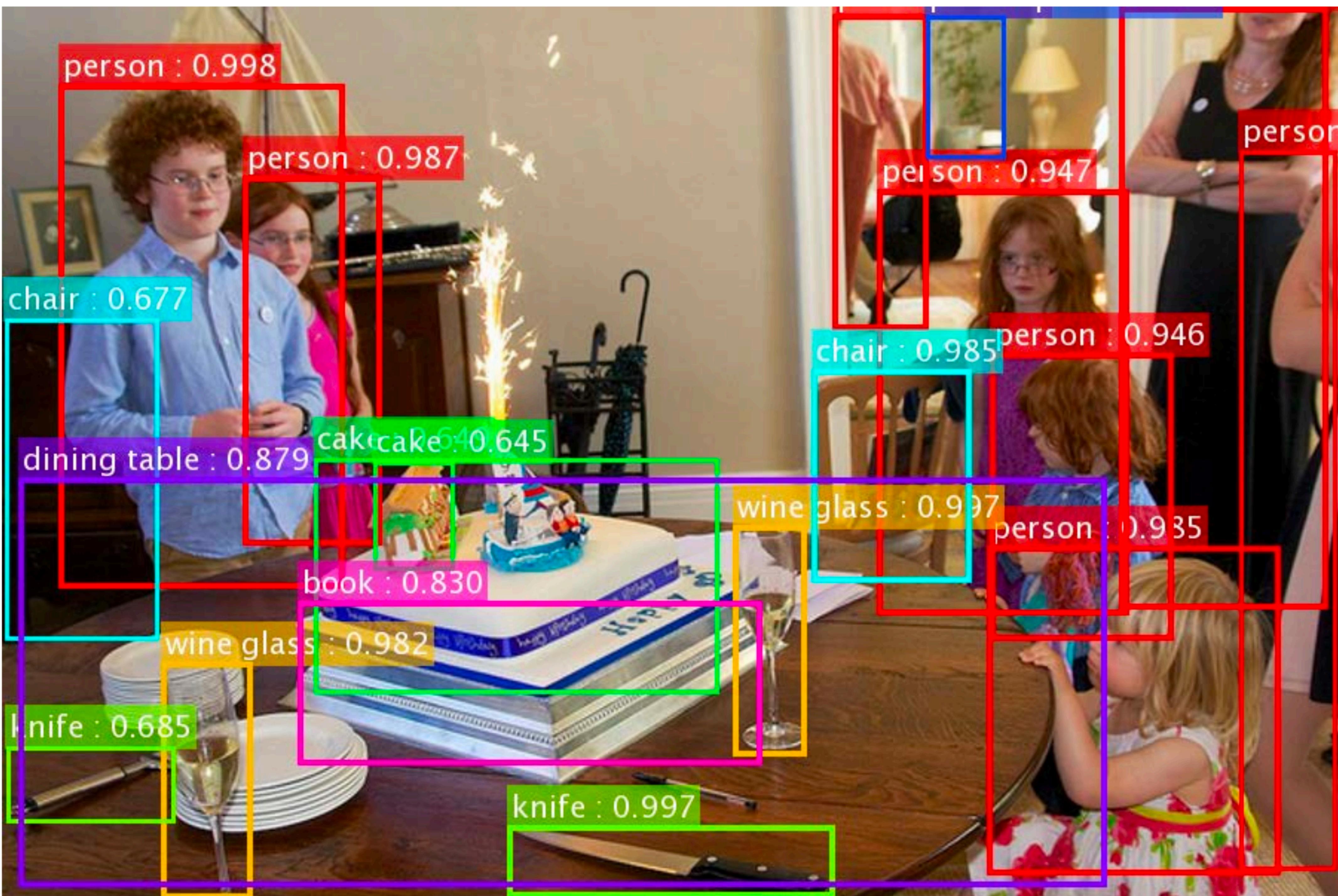
Результаты детекции

# YoloV\*



# RCNN





Задачи сегментации (semantic + instance)

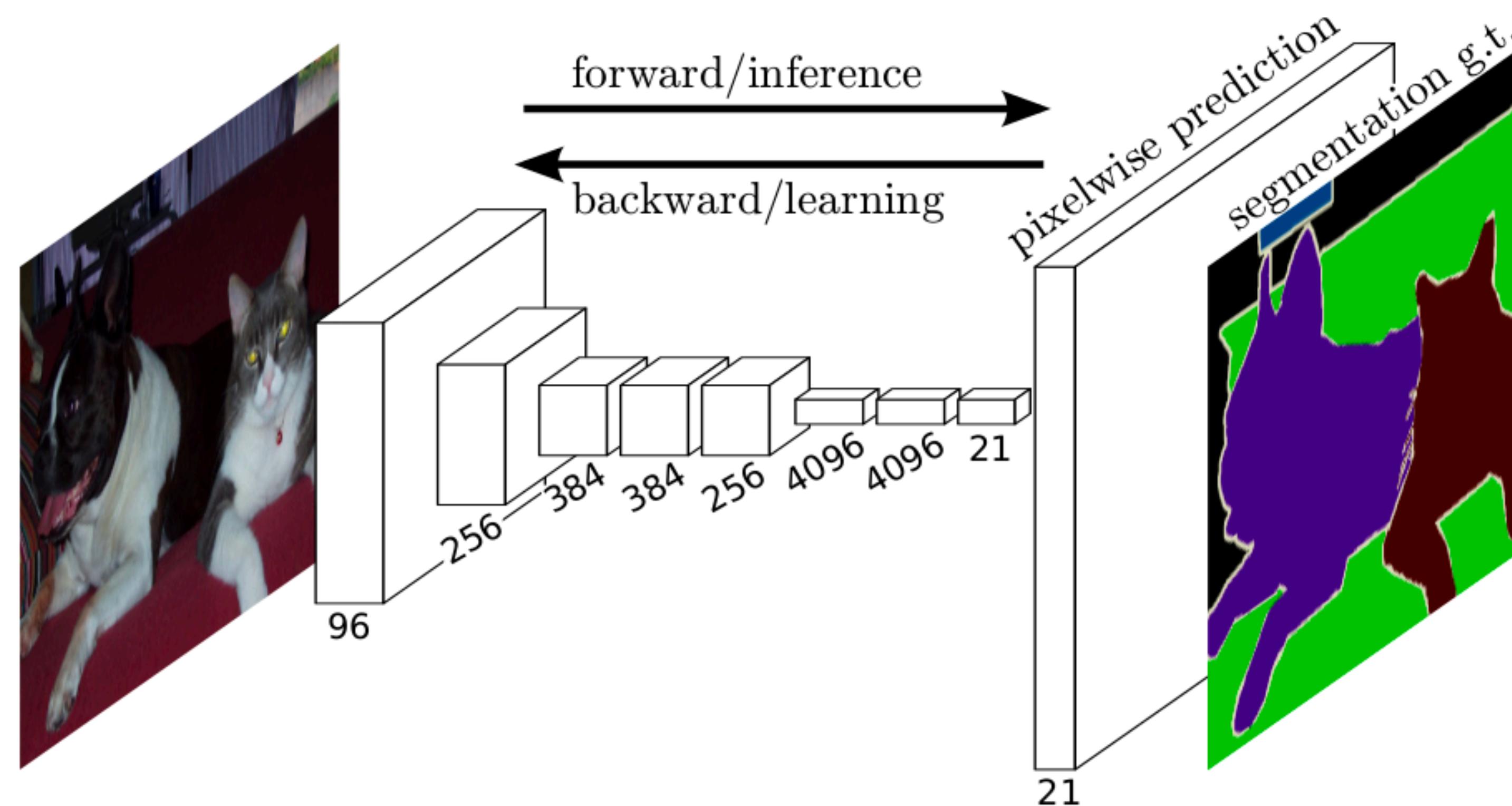
# Semantic segmentation



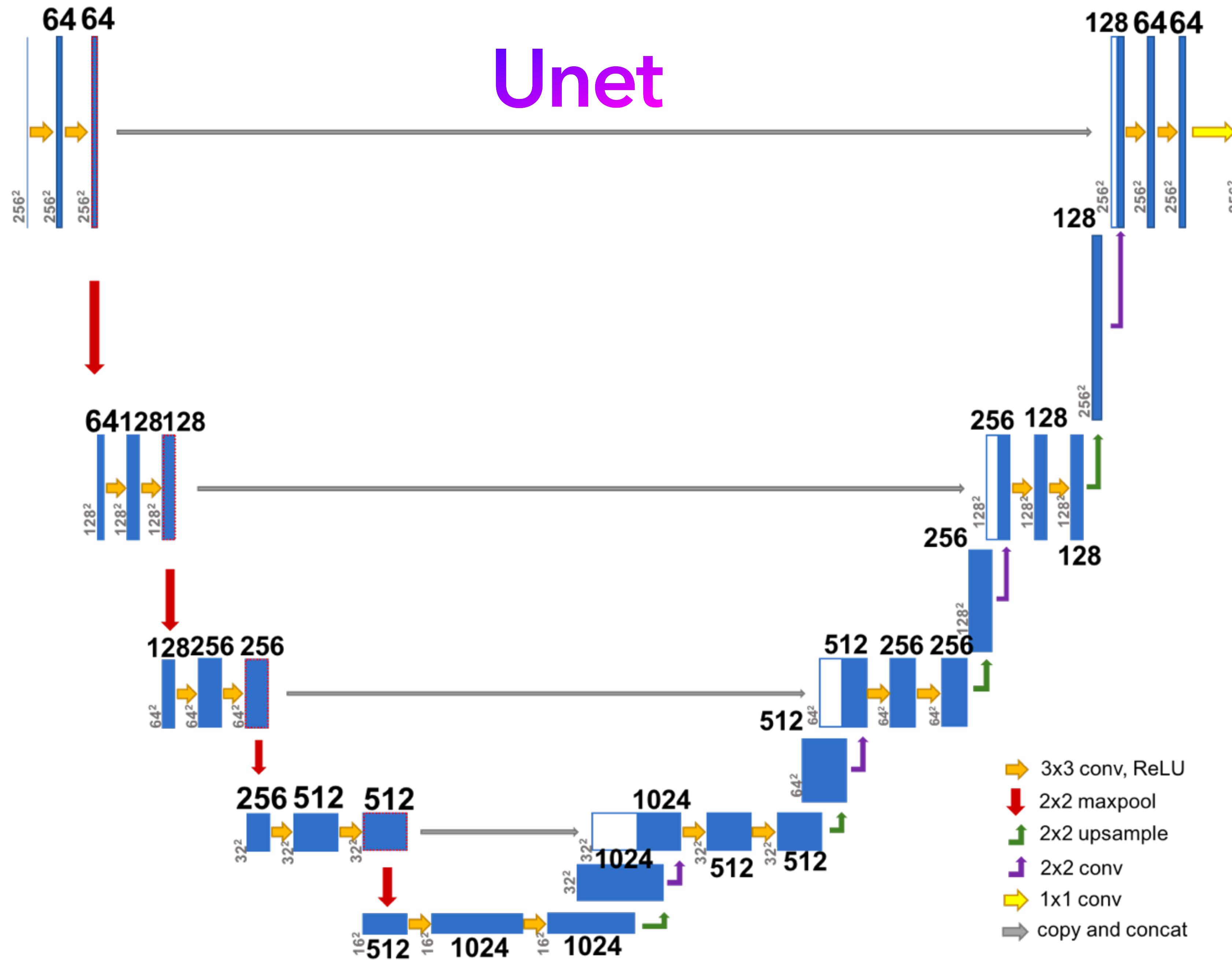
# Instance segmentation



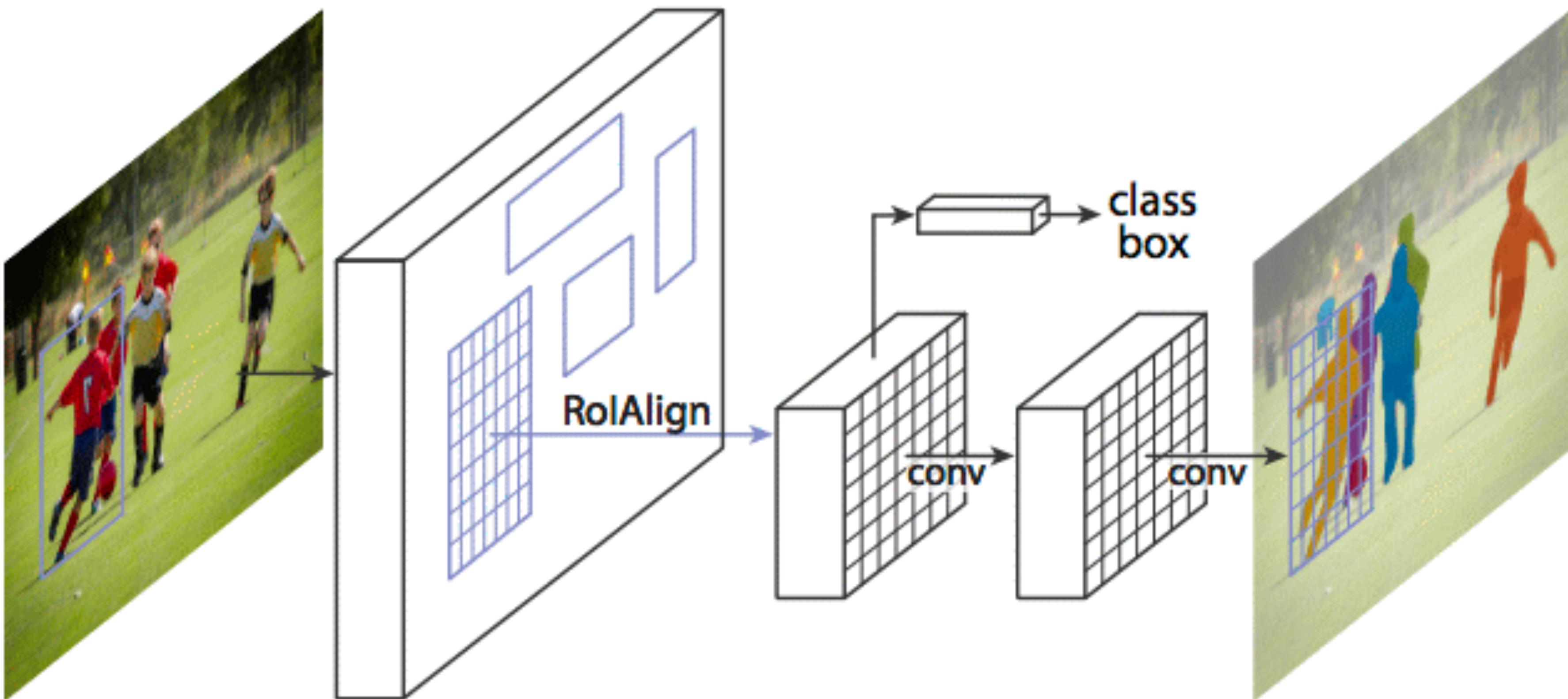
# Pipeline обучения



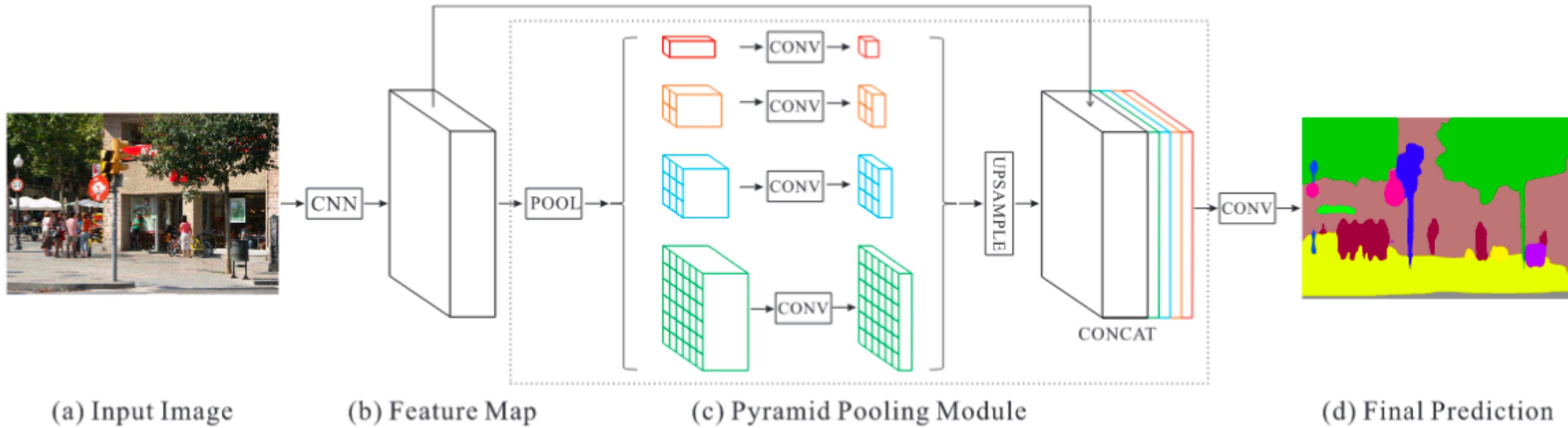
# Unet



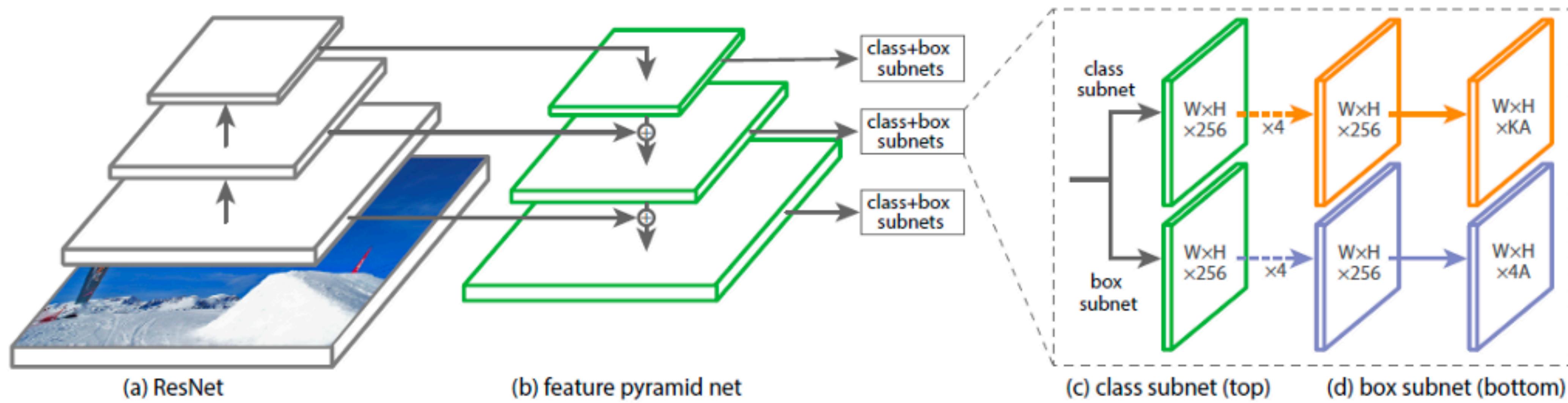
# Mask RCNN



# Multiscale models



# RetinaNet



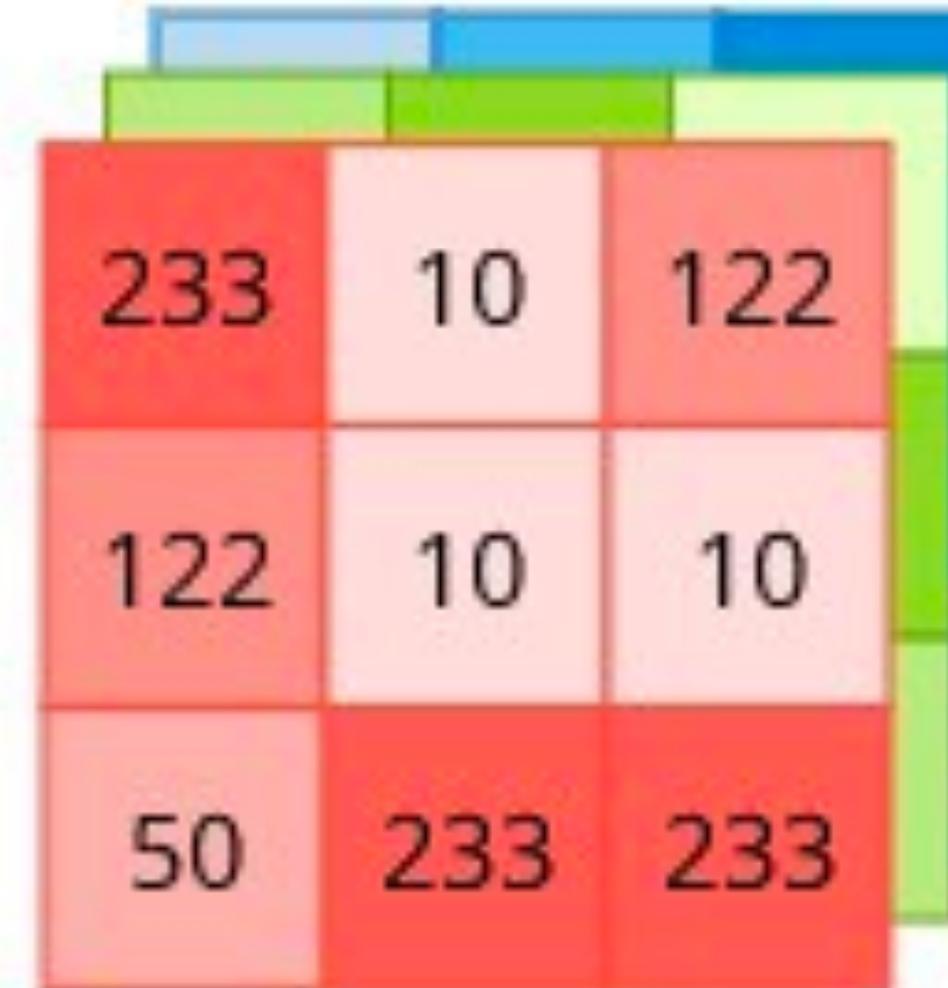
# Loss Functions

## PixWise softmax + CE

A. Class Labels

1	2	0
0	2	2
3	1	1

B. RGB mappings



C. One Hot Vectors

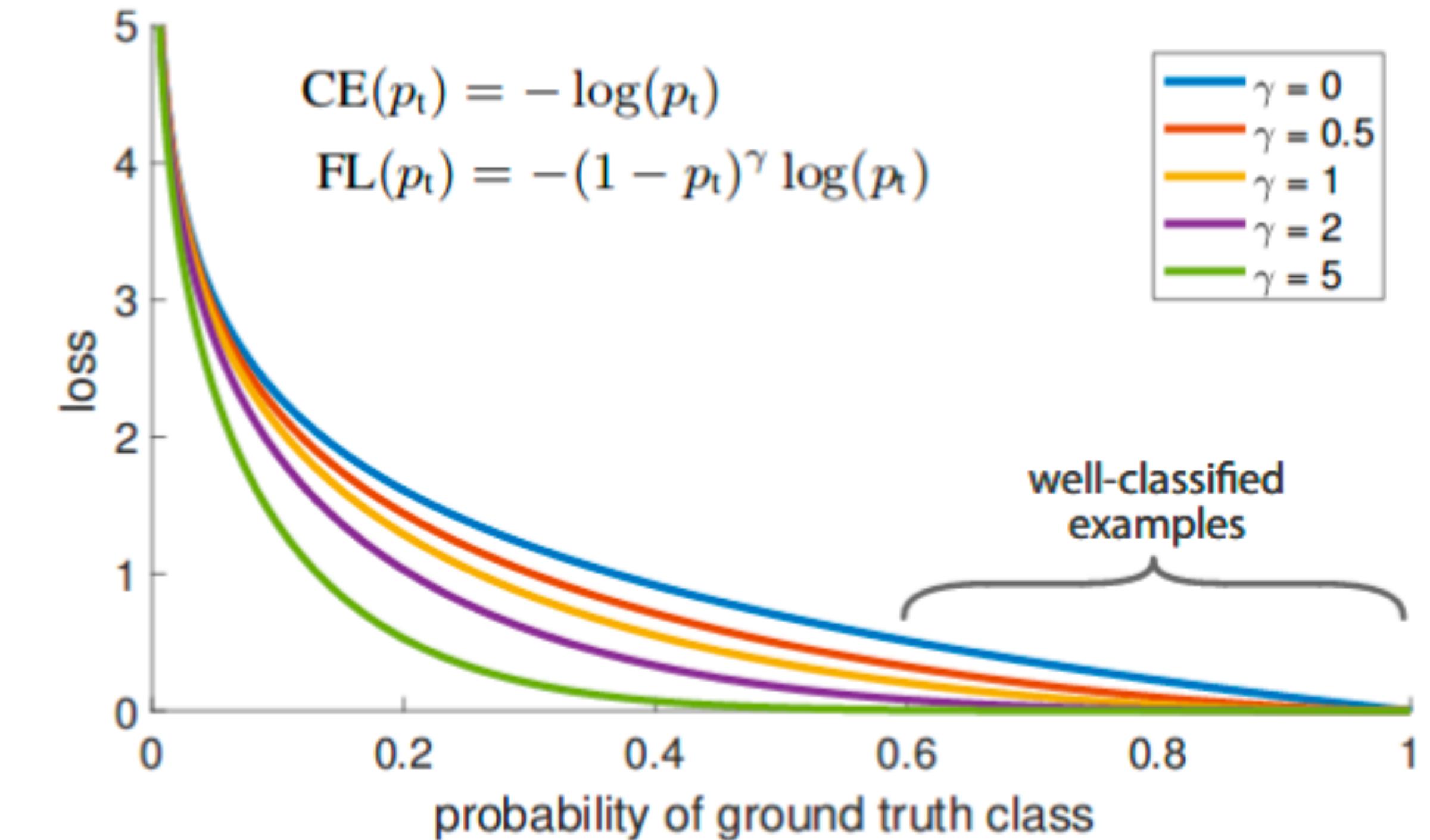
0	0	1
1	0	0
0	0	0

# Loss Functions

## Focal Loss

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

$$p_t = \begin{cases} p & \text{if } y = 1 \\ 1 - p & \text{otherwise,} \end{cases}$$



# Loss Functions

Dice loss (IoU но дифференцируемый)

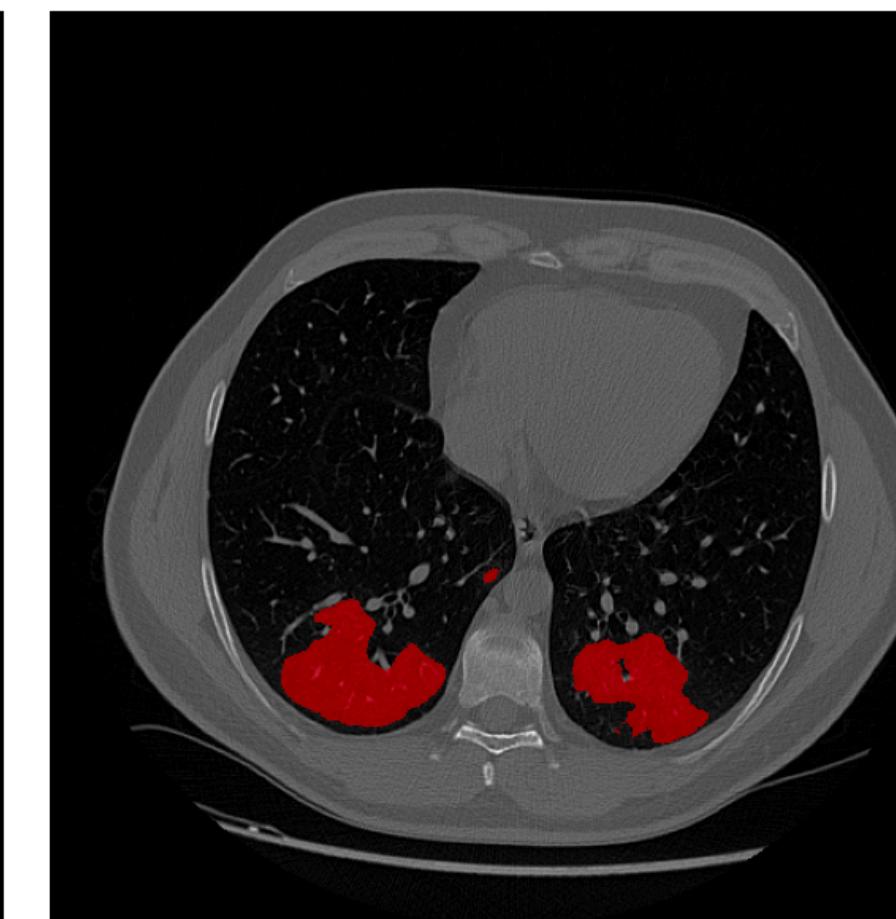
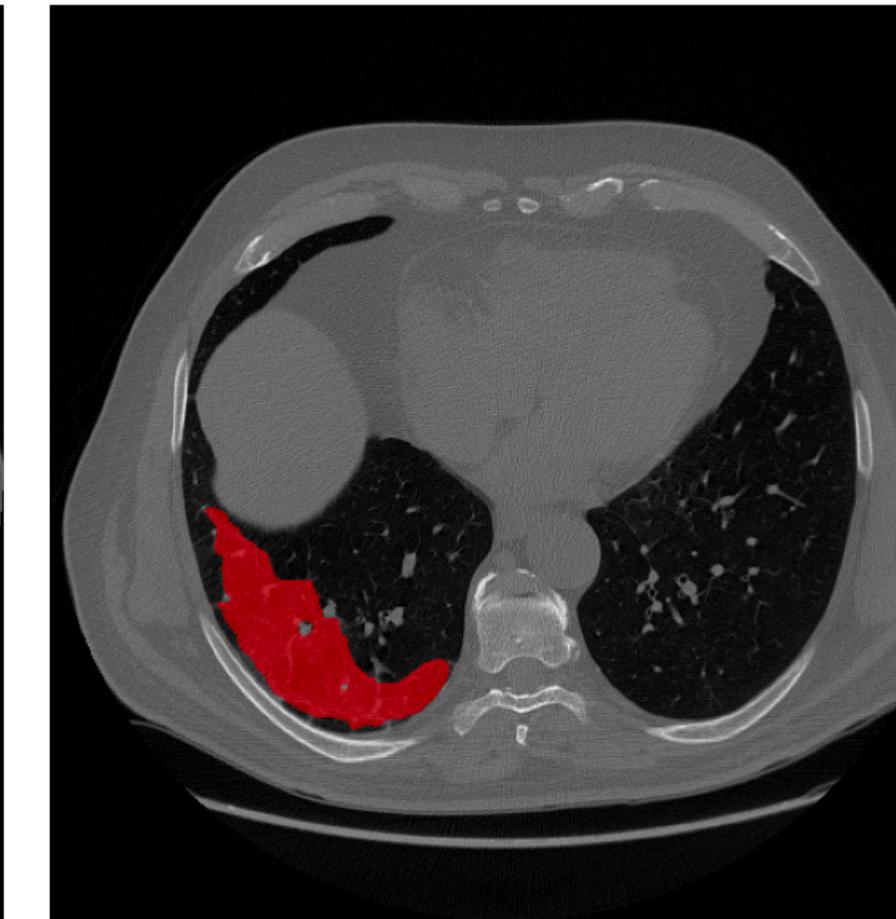
$$D = \frac{2 \sum_i^N p_i g_i}{\sum_i^N p_i^2 + \sum_i^N g_i^2}$$

$$\frac{\partial D}{\partial p_j} = 2 \left[ \frac{g_j \left( \sum_i^N p_i^2 + \sum_i^N g_i^2 \right) - 2p_j \left( \sum_i^N p_i g_i \right)}{\left( \sum_i^N p_i^2 + \sum_i^N g_i^2 \right)^2} \right]$$

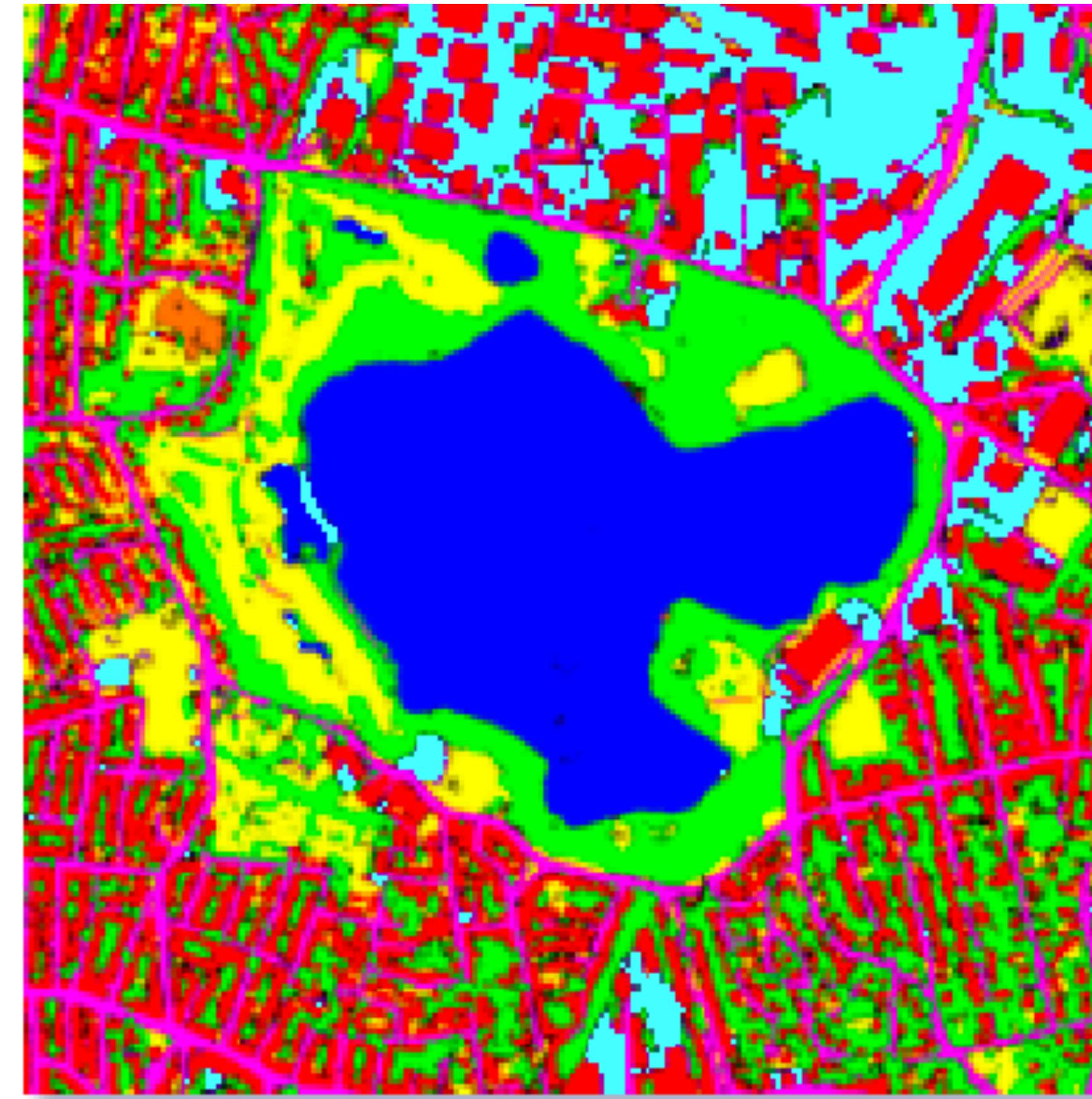
# Область применения?



# Область применения?



# Область применения?



# Captioning



```
CPU times: user 2 µs, sys: 1 µs, total: 3 µs
Wall time: 6.2 µs
100%[██████████] 1/1 [00:00<00:00, 2.59it/s]
```

или другой мужчина смотрит на другого человека, сидящего рядом с ним в кино



```
CPU times: user 4 µs, sys: 1e+03 ns, total: 5 µs
Wall time: 11 µs
100% [██████████] 1/1 [00:02<00:00, 2.68s/it]
```

Кухня с зелеными окнами и стульями, сидящая на столе.



```
CPU times: user 3 µs, sys: 1e+03 ns, total: 4 µs
Wall time: 8.11 µs
100%|██████████| 1/1 [00:01<00:00, 1.50s/it]
```

красный велосипед, сидящий на обочине дороги.

```
25 print('\n')
26 print(generated_text_prefix)
```



```
CPU times: user 3 µs, sys: 0 ns, total: 3 µs
Wall time: 7.63 µs
100%|██████████| 1/1 [00:03<00:00,  3.40s/it]
```

Молодой человек, который стоит перед входом в отель и разговаривает с кем-то по телефону.

```
25 print('\n')
26 print(generated_text_prefix)
```

└→



```
CPU times: user 3 µs, sys: 1e+03 ns, total: 4 µs
Wall time: 7.63 µs
100%|██████████| 1/1 [00:02<00:00,  2.48s/it]
```

Шумные игрушки, похожие на маленьких собак, играют с ними.