



Тверской
государственный
технический
университет

Интеллектуальные информационные системы

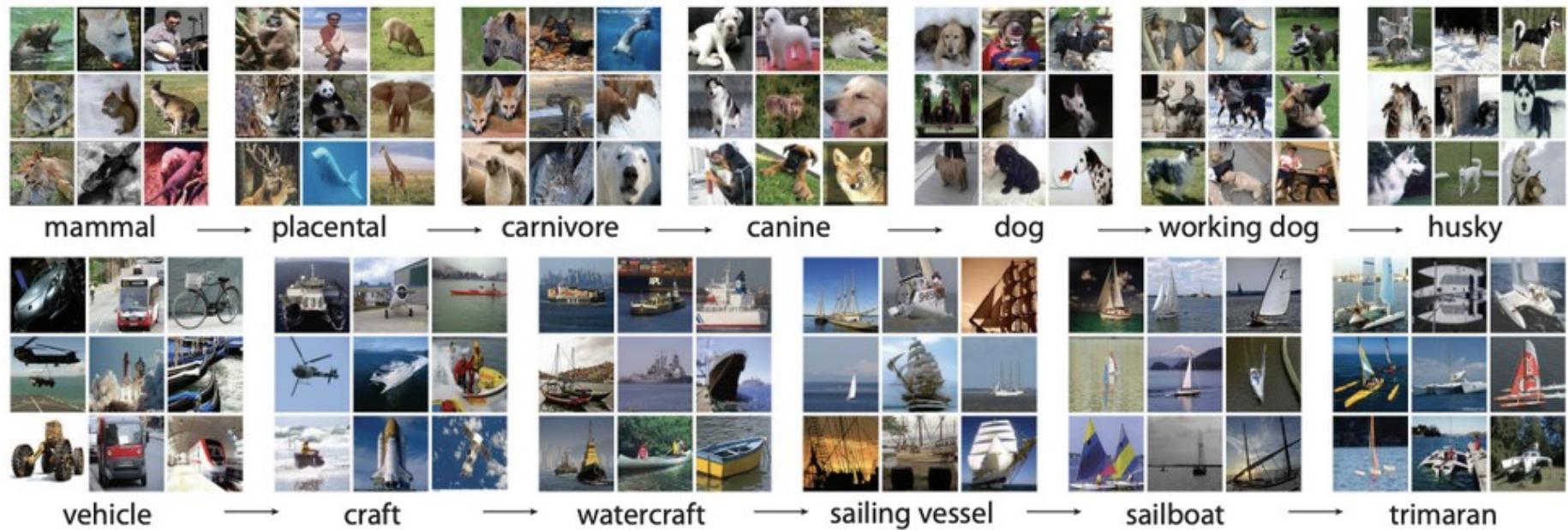
Сверточные нейронные сети

2025 г.

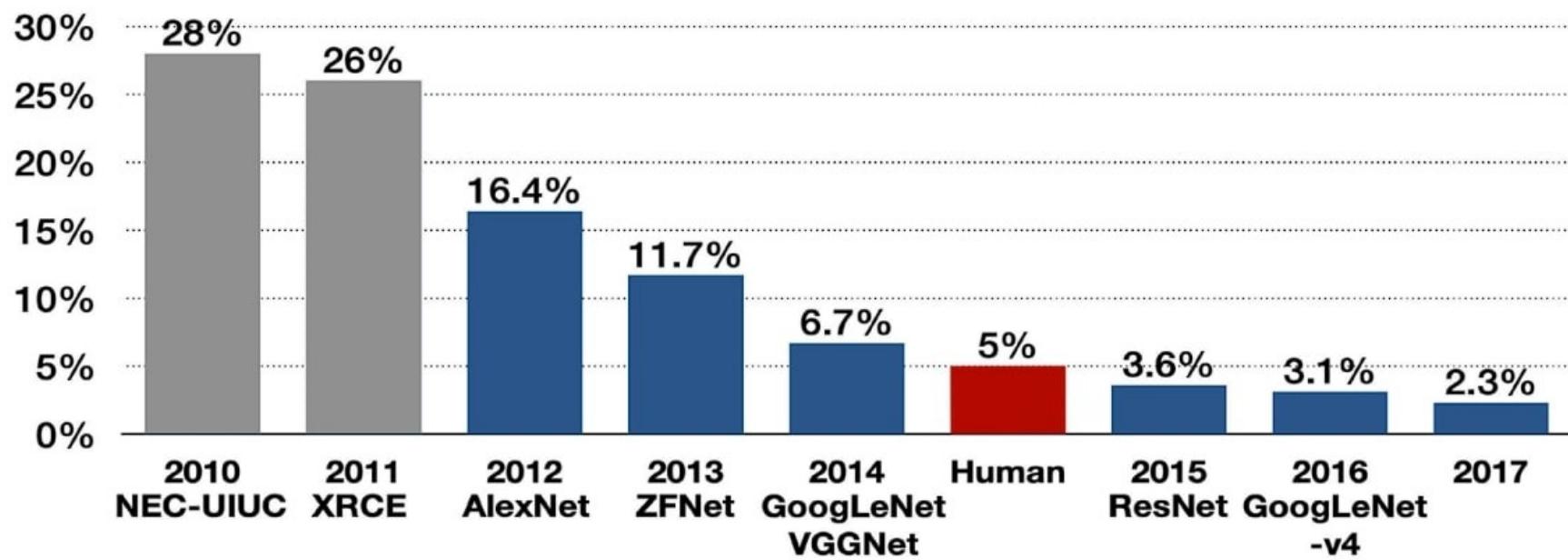
The **ImageNet** project is a large visual database designed for use in visual object recognition software research

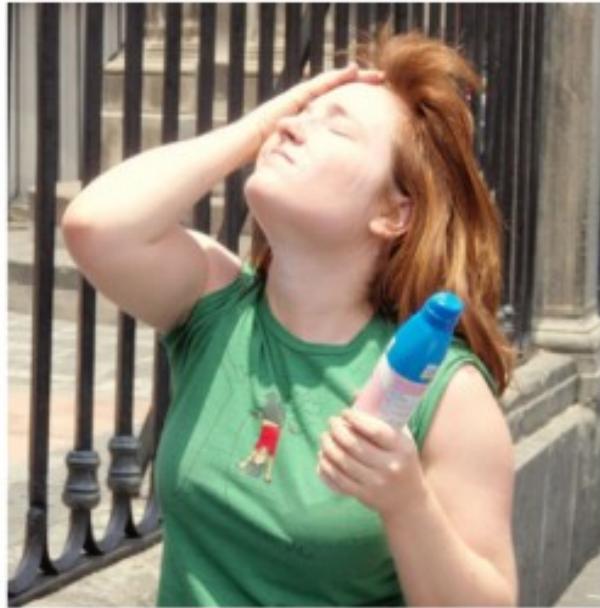


ImageNet: A Large-Scale Hierarchical Image Database



Top-5 error





GT: sunscreen

- 1: hair spray
- 2: ice lolly
- 3: sunscreen**
- 4: water bottle
- 5: lotion



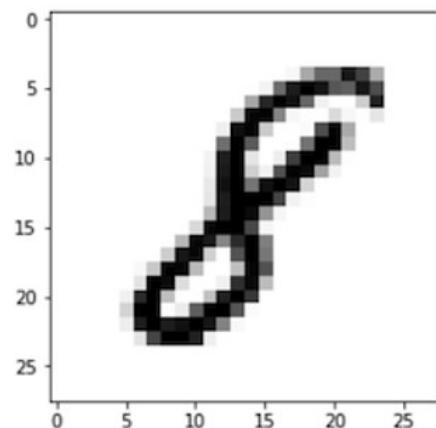
GT: flute

- 1: flute**
- 2: oboe
- 3: panpipe
- 4: trombone
- 5: bassoon



GT: wooden spoon

- 1: wok
- 2: frying pan
- 3: spatula
- 4: wooden spoon**
- 5: hot pot



=

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0														
0	0	0	0	1	12	0	11	39	137	37	0	152	147	84	0	0	0	0	0	0	0	0	0	0	0	0	0	0														
0	0	1	0	0	0	41	160	250	255	235	162	255	238	286	11	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0	0	0	15	9	9	158	251	45	21	184	159	154	255	233	40	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
10	0	0	0	0	0	145	146	3	10	0	11	124	253	255	187	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0												
0	0	0	3	9	4	15	236	216	0	0	38	109	247	240	169	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
1	0	2	0	0	0	253	253	23	62	224	241	255	164	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0											
6	0	0	4	0	0	3	252	250	238	255	255	234	127	28	0	2	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
0	2	1	4	0	0	21	255	253	251	255	172	31	8	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0										
0	0	4	0	153	225	251	255	229	120	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
0	0	0	21	162	235	235	254	255	126	6	0	18	14	6	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0									
3	79	242	255	141	66	255	245	189	7	8	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0							
26	221	237	98	0	67	251	255	144	0	8	0	0	7	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0						
125	255	141	0	87	244	255	268	3	0	0	13	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0				
145	248	228	116	235	235	141	34	0	11	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0			
05	237	253	246	235	216	21	1	0	1	0	0	0	2	0	0	0	7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0		
6	23	112	157	114	32	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

32

Grayscale

223	150	91
39	71	150
221	150	221

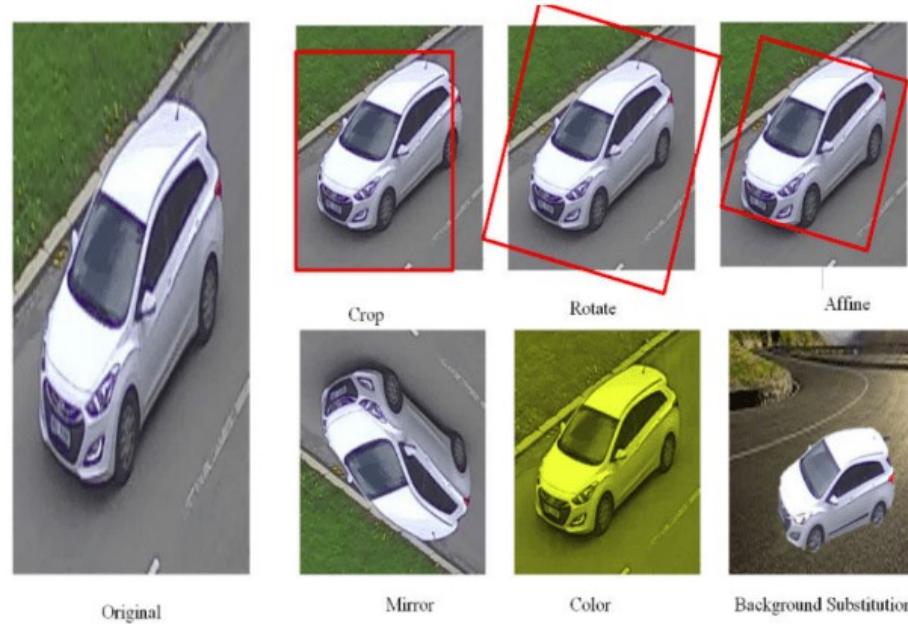
RGB

255	79	42			
79	255	42	181		
42	79	255	102	63	
102	255	29	105		
63	105	253			

32

Классификация с помощью полносвязной НС с кросс-энтропийной функцией потерь:

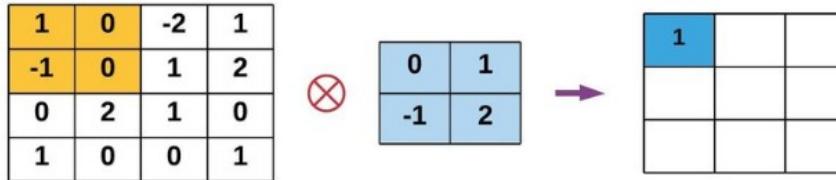
- Никак не учитывается структура данных - хочется получить инвариантность к различным преобразованиям исходного изображения



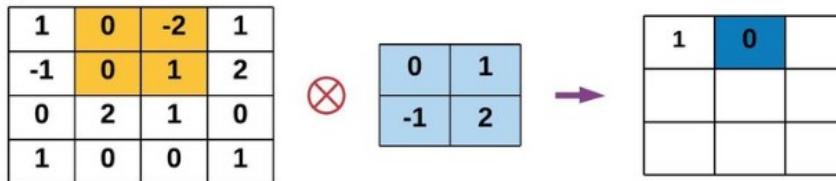
- Большое количество параметров (нейронов в первом слое сети). Например, RGB изображение размером 128x128 это вектор из 49152 элементов

Операция свертки - поэлементное умножение и суммирование

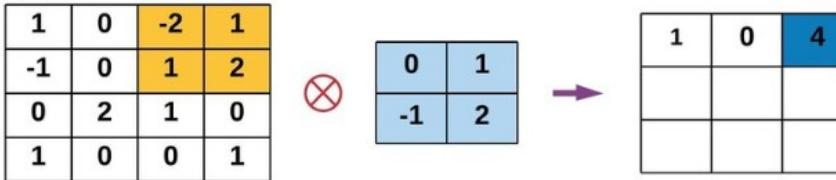
Step-1



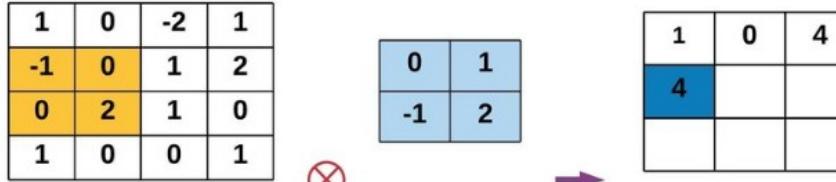
Step-2



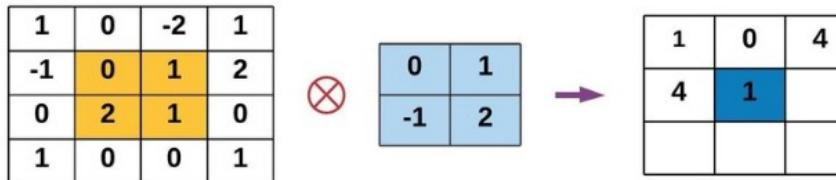
Step-3



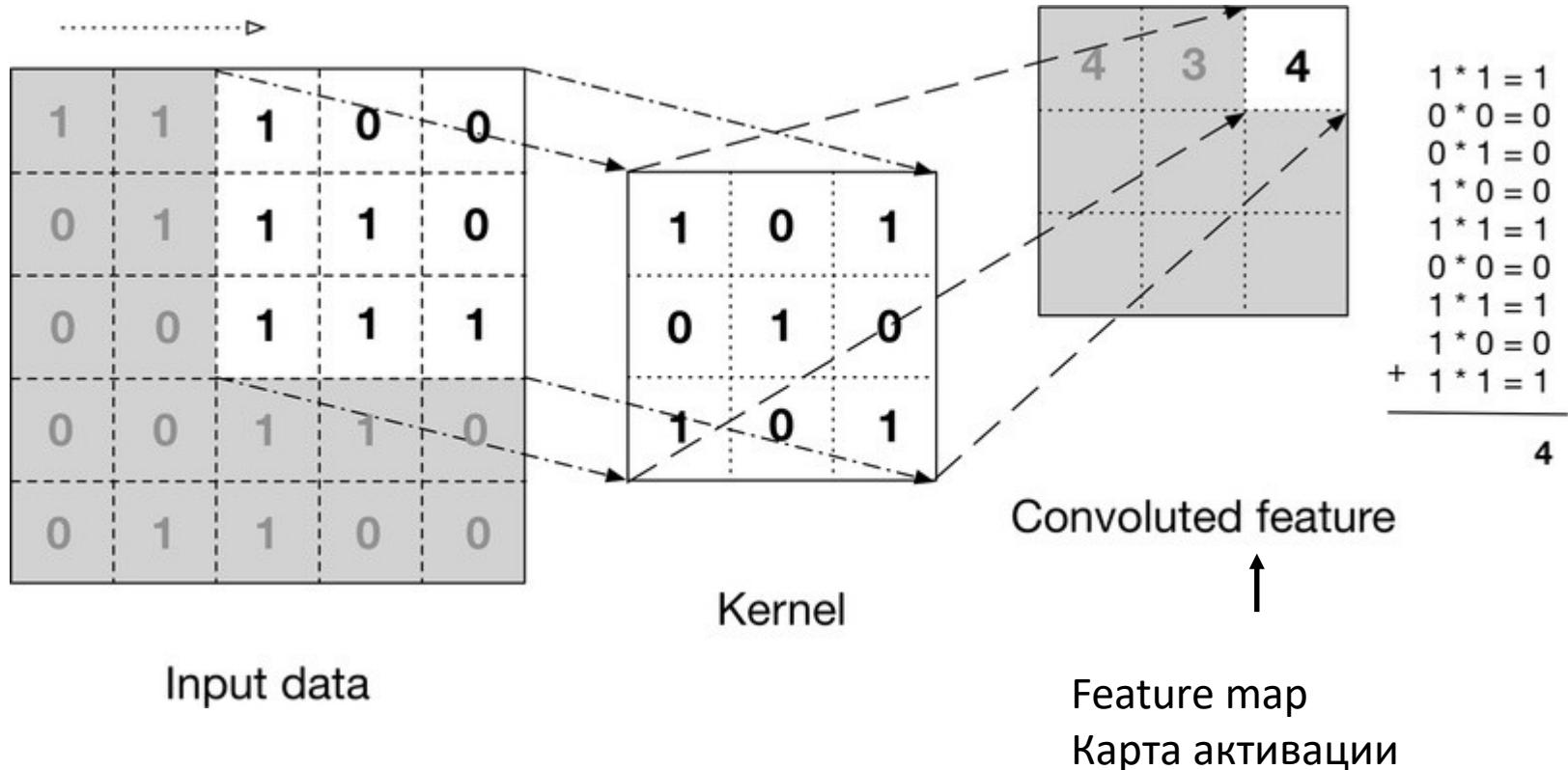
Step-4



Step-5



С чем сворачиваем? С ядром свертки (фильтром)



Strided Convolution

Input Image

1	9	8	4	4	5	7
4	8	6	7	9	1	7
4	0	5	9	3	8	4
7	3	5	9	0	5	4
7	4	1	1	8	1	2
7	6	6	9	8	7	6
3	6	3	5	4	2	7

Kernel

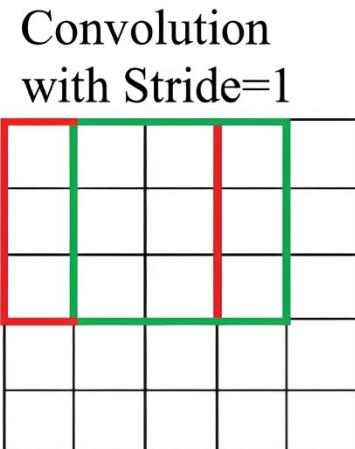
*

0	-1	0
-1	5	-1
0	-1	0

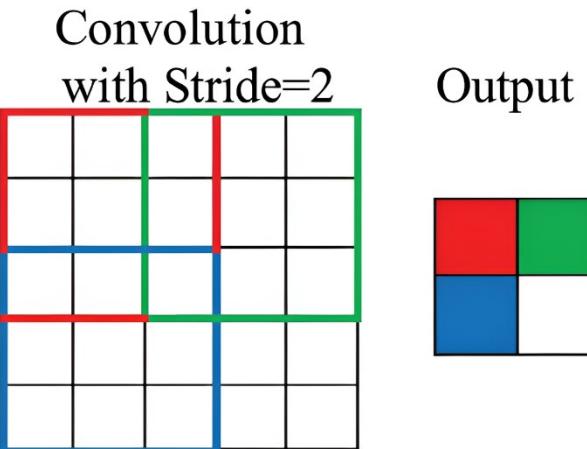
Feature Map

=

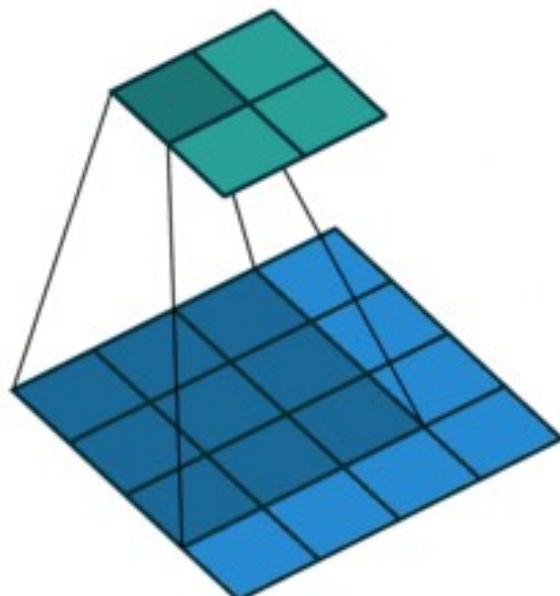
● Stride – величина смещения фильтра



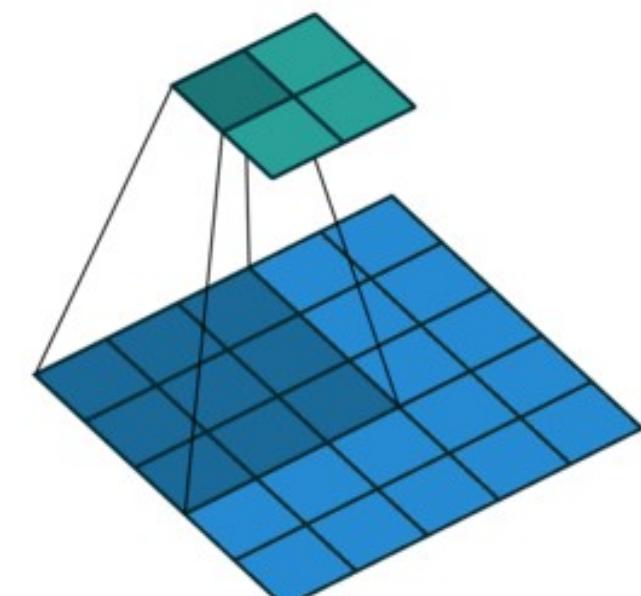
Output



Output



Stride = 1



Stride = 2

- Padding – дополнение исходного изображения пикселями (рамка вокруг изображения)

0	0	0	0	0	0	0	0	0
0	1	2	3	4	5	6	0	
0	7	8	9	10	11	12	0	
0	1	2	3	4	5	6	0	
0	7	8	9	10	11	12	0	
0	1	2	3	4	5	6	0	
0	7	8	9	10	11	12	0	
0	0	0	0	0	0	0	0	0

Zero padding with a one-pixel thick boundary

1	1	1	2	3	4	4	4	4
1	1	1	2	3	4	4	4	4
1	1	1	2	3	4	4	4	4
5	5	5	6	7	8	8	8	8
1	1	1	2	3	4	4	4	4
5	5	5	6	7	8	8	8	8
5	5	5	8	9	8	8	8	8
5	5	5	8	9	8	8	8	8



zero padding

11	10	9	10	11	12	11	10
7	6	5	6	7	8	7	6
3	2	1	2	3	4	3	2
7	6	5	6	7	8	7	6
11	10	9	10	11	12	11	10
15	14	13	14	15	16	15	14
11	10	9	10	11	12	11	10
7	6	5	6	7	8	7	6

Mirror padding with a two-pixel thick boundary



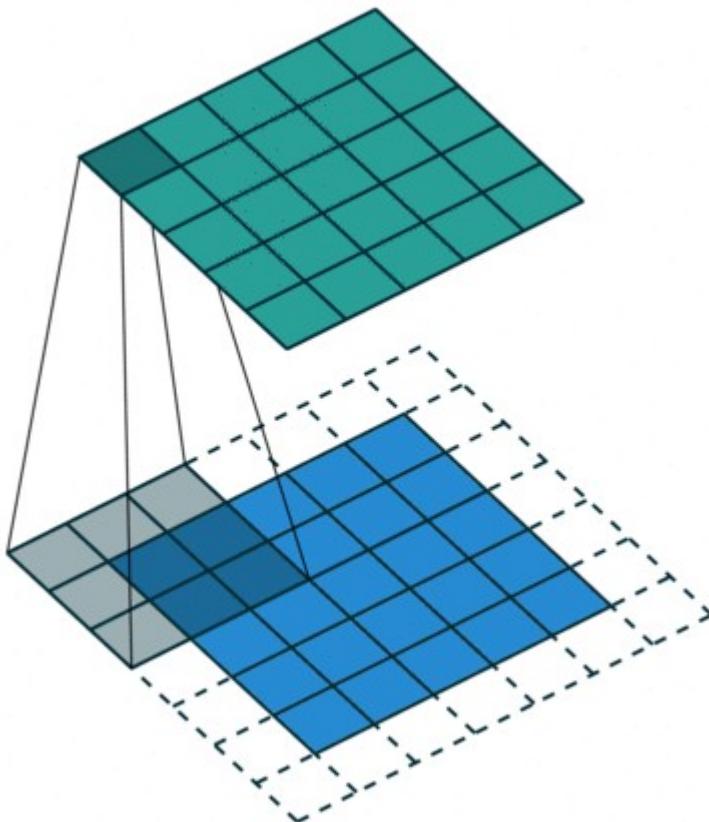
mirror padding



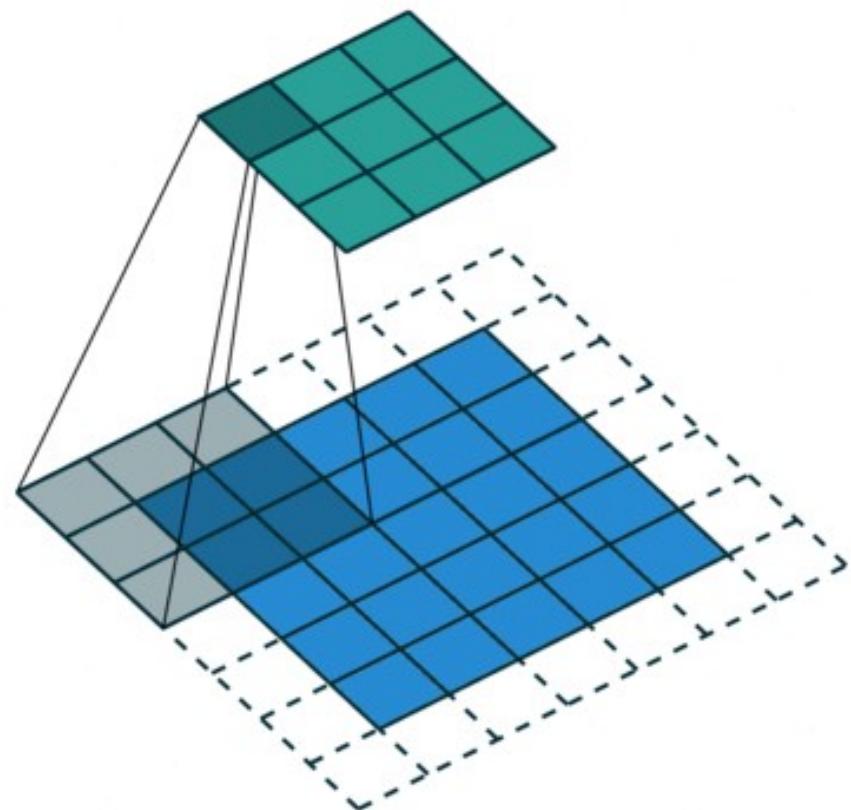
circular padding

Replicate padding with a two-pixel thick boundary

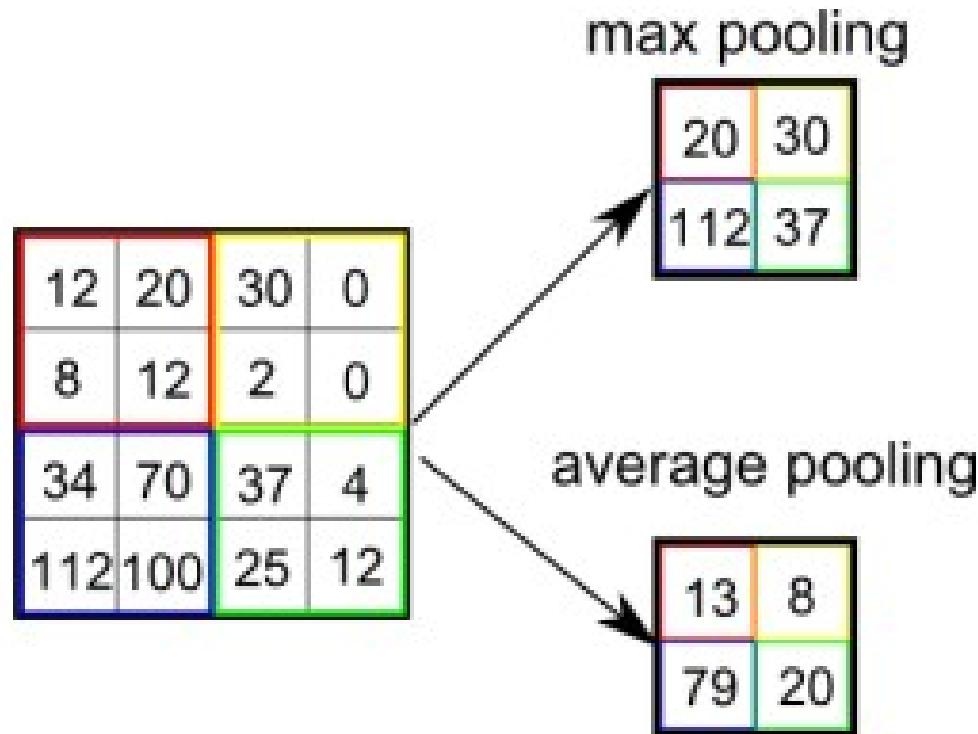
Padding = 1
Stride = 1



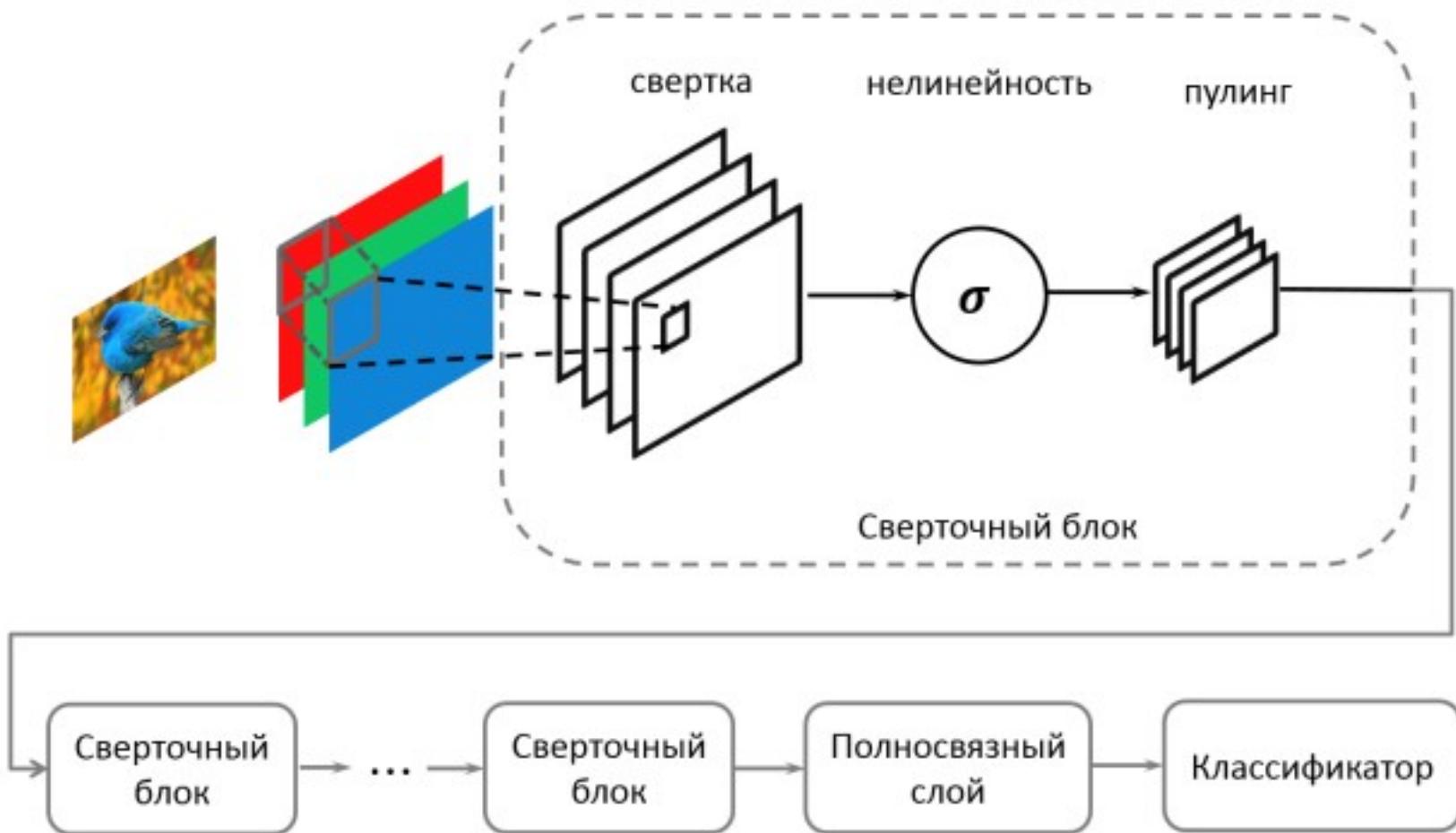
Padding = 1
Stride = 2



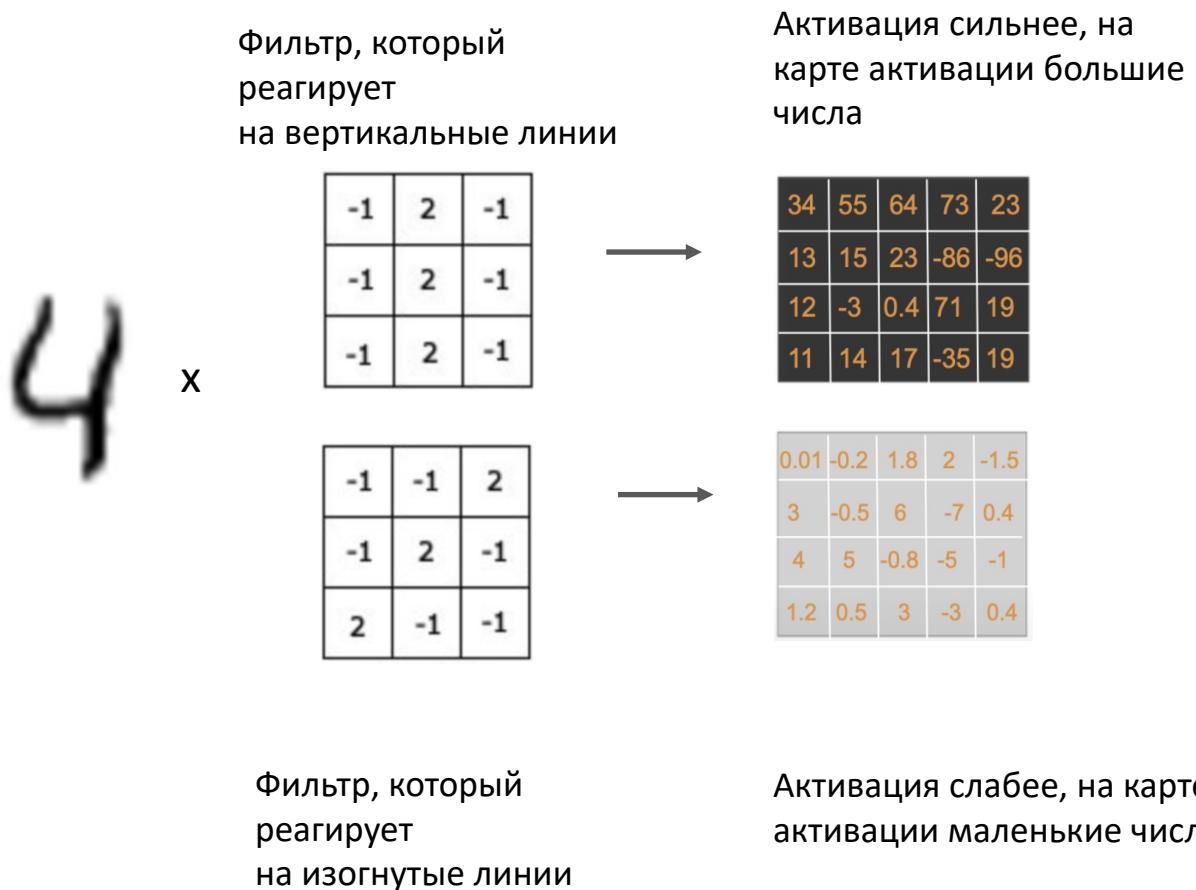
- Pooling – техника уменьшения размерности (downsampling) карт активации. Если некая область содержит ярко выраженные свойства, то мы можем отказаться от поиска других свойств в этой области



Базовая схема сверточной сети



Фильтры “реагируют” на паттерны на изображении. Если паттерн присутствует на изображении, то карта активации после соотв. фильтра будет содержать большие числа



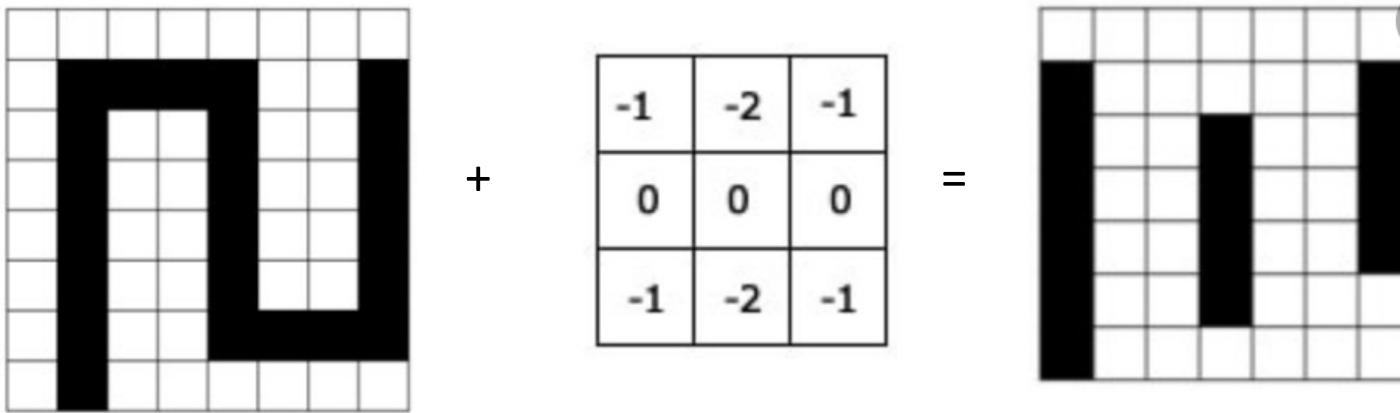
Как мог бы выглядеть фильтр, реагирующий на горизонтальные линии

The diagram illustrates a convolution operation. On the left is an input image consisting of a black 'L' shape on a white background, represented on a 5x5 grid. In the center is a 3x3 kernel with the following values:

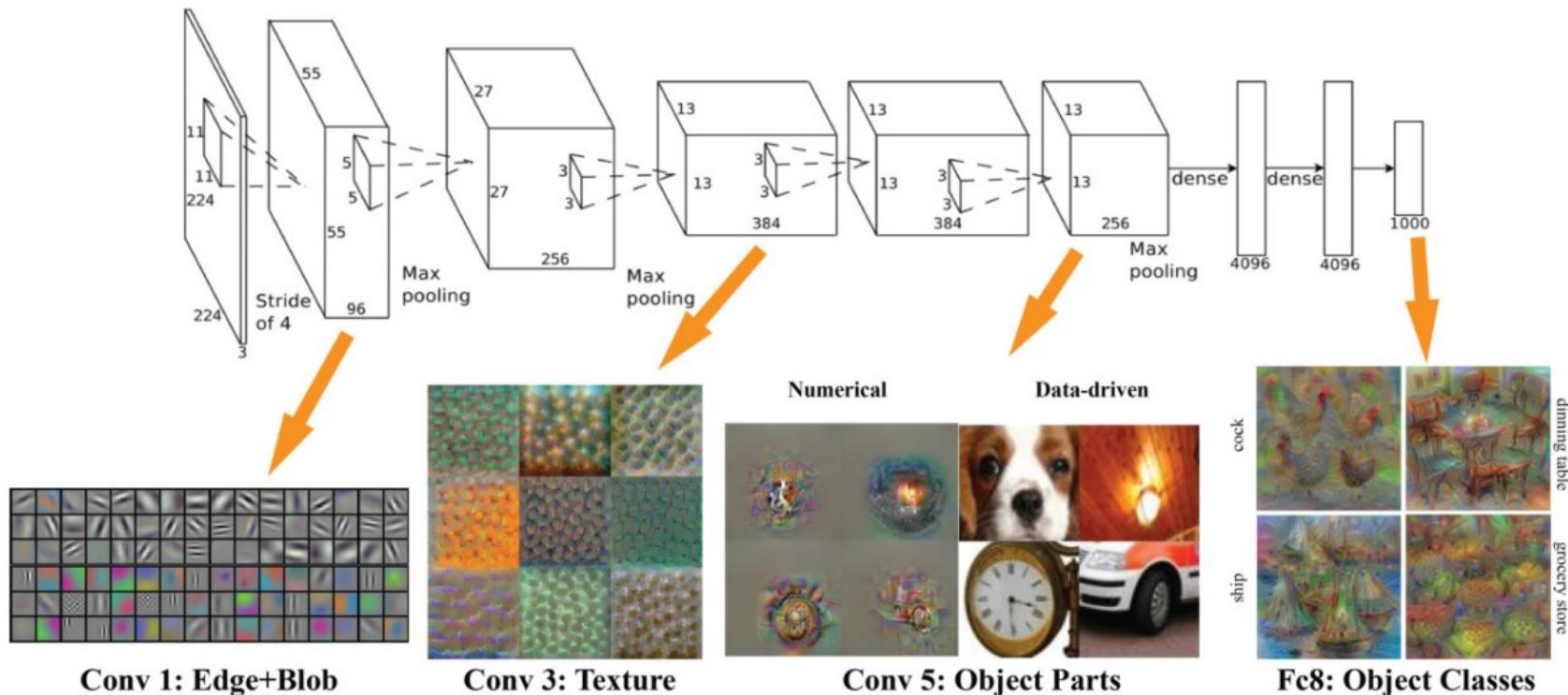
-1	0	1
-1	0	1
-1	0	1

Between the input and the kernel is a plus sign (+), indicating addition. To the right of the kernel is an equals sign (=). The resulting output image on the far right shows a single black pixel at the top-left position, representing the result of applying the kernel to the input image.

Как мог бы выглядеть фильтр, реагирующий на вертикальные линии

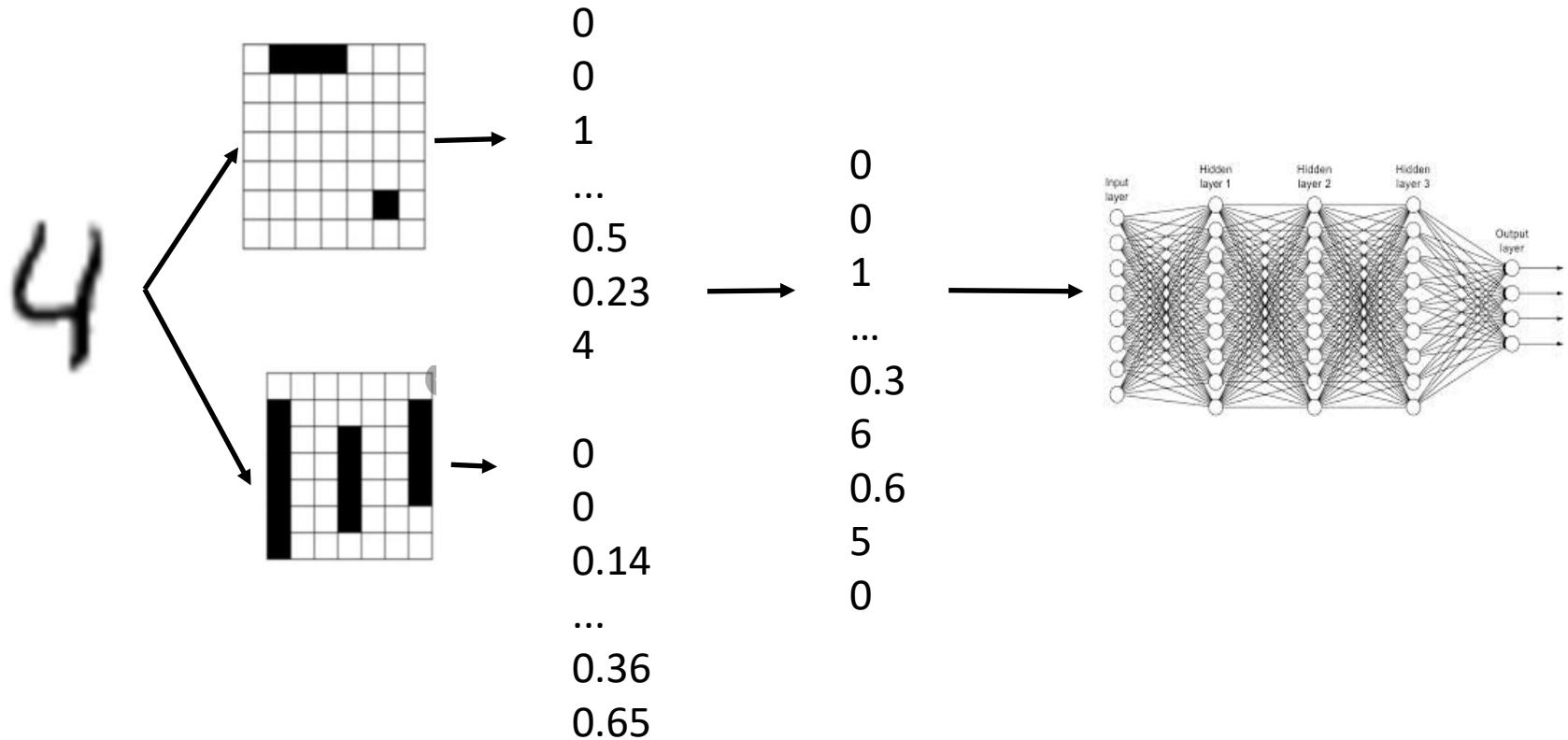


- Сверточная сеть обучается извлечению признаков. Чем выше слой, тем более крупные и сложные элементы изображений он способен распознавать

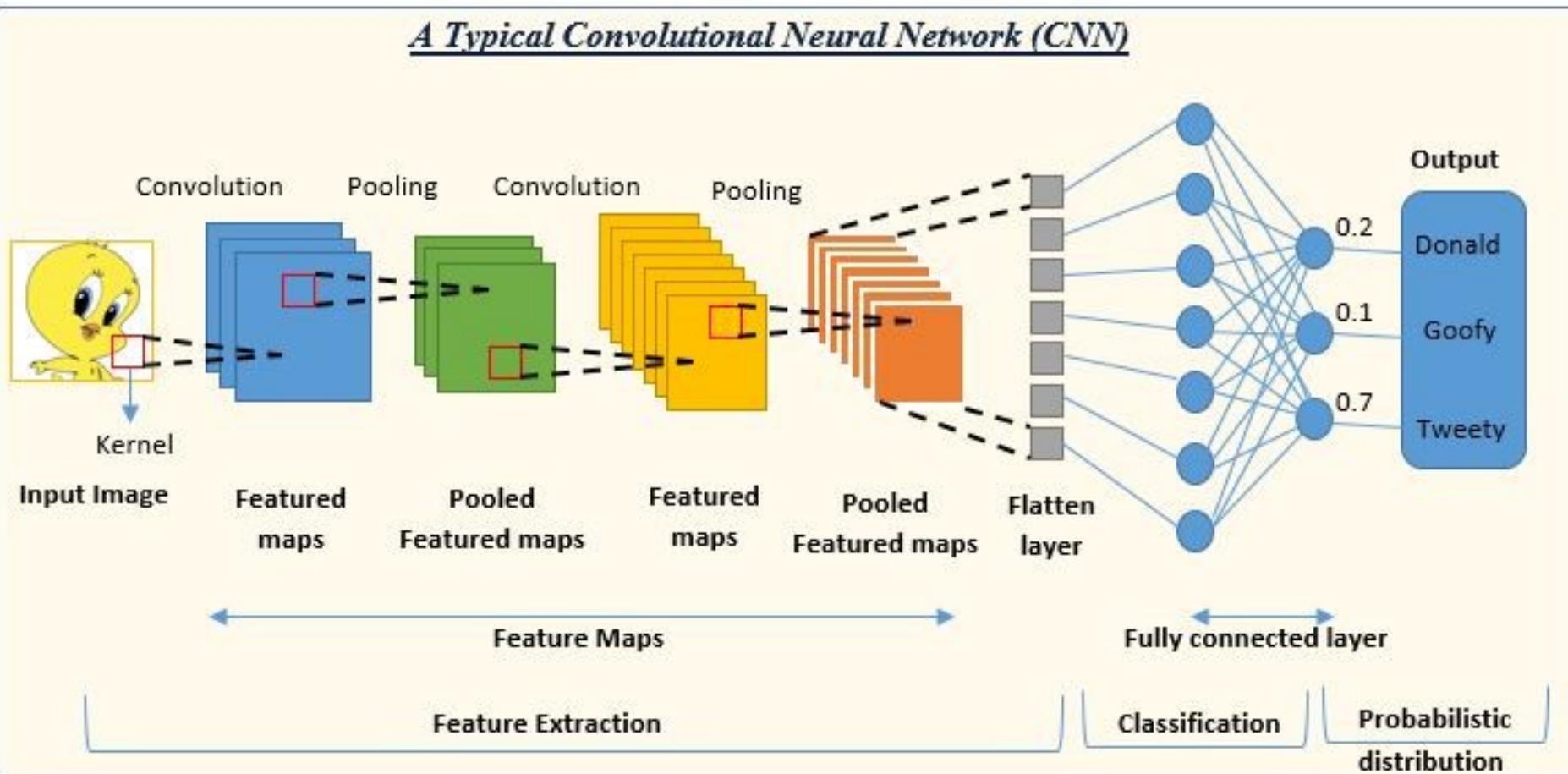


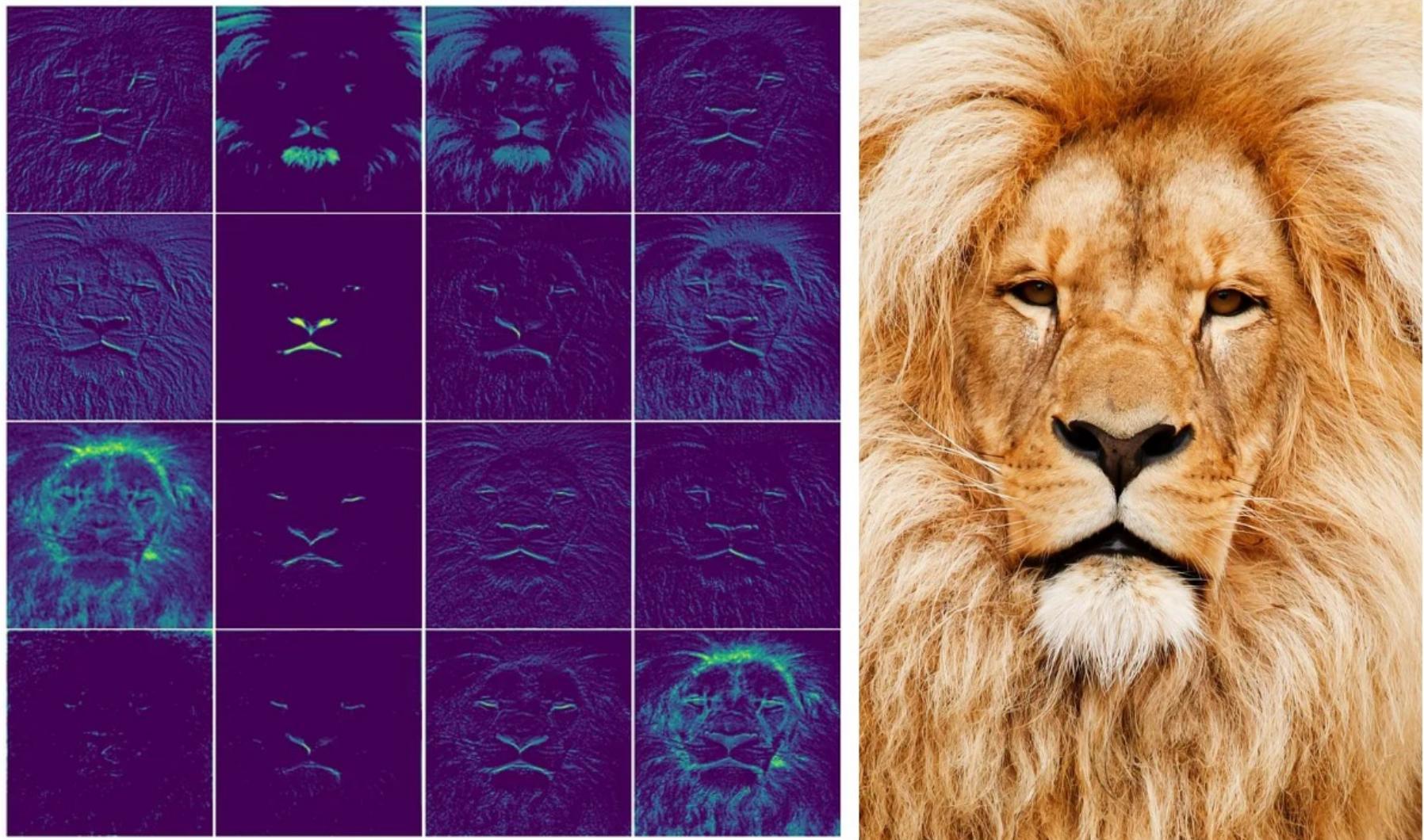
Krizhevsky A., Sutskever I., Hinton G. ImageNet classification with deep convolutional neural networks. 2012.

После получения карт активаций, мы **развернем все карты в векторы, сконкатенируем и подадим на вход полно связной сети**



Вместо ручного конструирования признаков изображения с помощью операции свертки НС сама обучается извлечению признаков. Но для комплексных изображений необходимо много больше промежуточных представлений





(Left) Feature extraction performed over the image of a lion using vgg19 CNN architecture (image by author). (Right)
Original picture of the lion (public domain, available at [Pexels](#)).

Свертка:

- Выявляет наличие на изображении паттерна, который задается ядром свертки
- Результат операции свертки – промежуточное представление (новое изображение)
- Чем сильнее на исходном изображении представлен паттерн, тем выше будут значения свертки
- Хотим много различных паттернов – используем много сверток

А что вообще обучать?

- Ядро свертки – подбирается в процессе обучения
- Веса полносвязного слоя – подбираются в процессе обучения

- Первые свертки выделяют какие-то низкоуровневые паттерны (изгибы, край, линии)
- Последние сверточные слои выделяют части изображений. Выход последнего слоя - признаковое описание изображения
- Полученные представления (признаковые описания) подаются на вход полносвязной сети. Например, для задачи многоклассовой классификации: размер выходного слоя - количество классов, функция активации - softmax

- Зная параметры сети (размер ядра свертки, паддинг, пуллинг и страйд) можно посчитать размер выходной матрицы признаков:

$$n_{out} = \left\lfloor \frac{n_{in} + 2p - k}{s} \right\rfloor + 1$$

n_{in} : number of input features

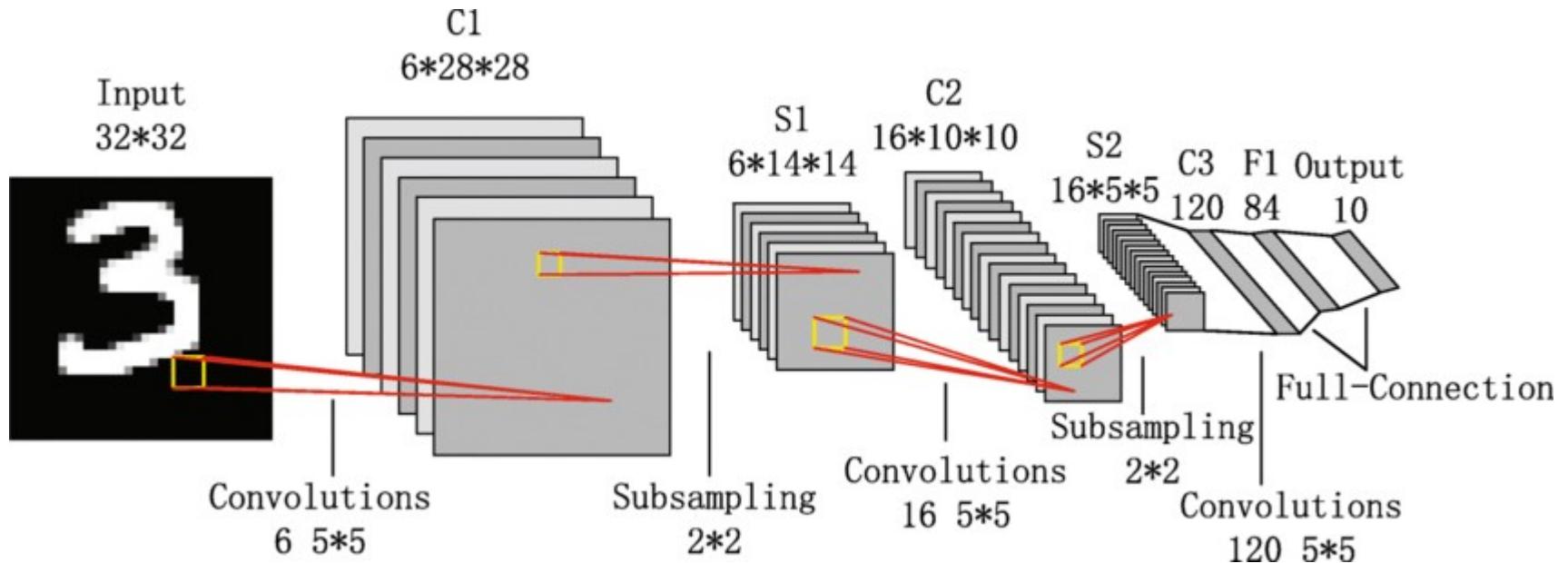
n_{out} : number of output features

k : convolution kernel size

p : convolution padding size

s : convolution stride size

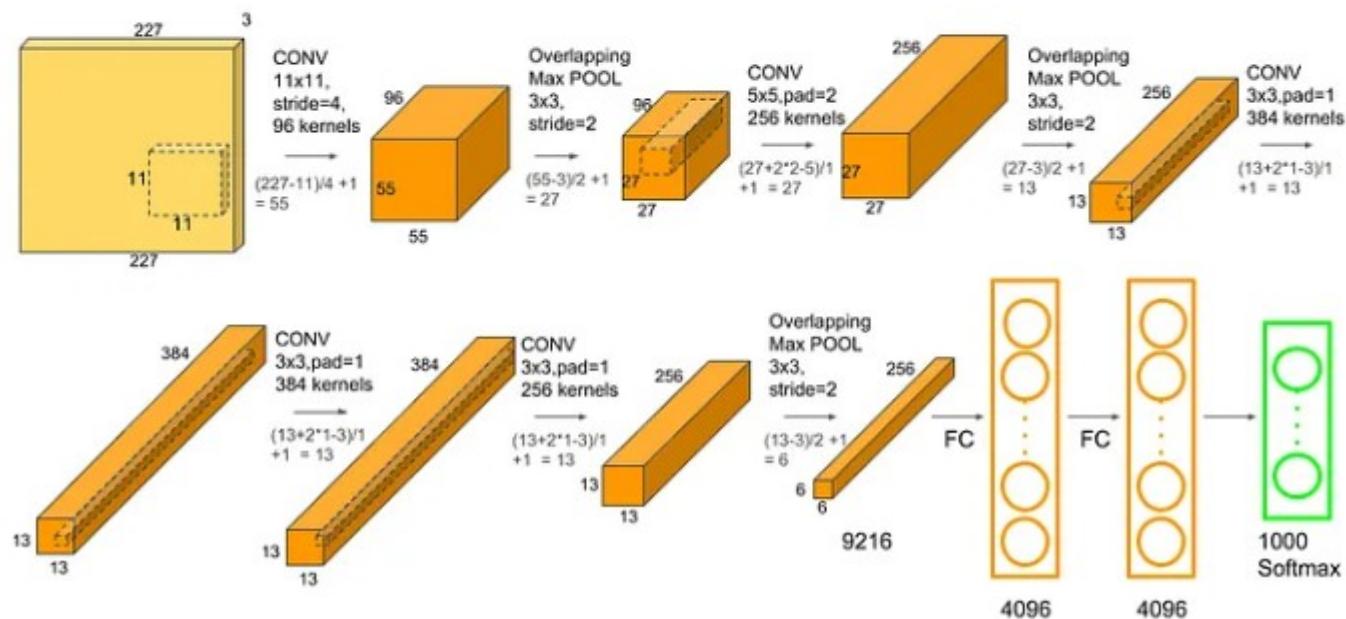
Насколько глубокие должны быть сети?



Архитектура нейронной сети LeNet-5, 1998 год.
Первая версия LeNet-1 была представлена в 1989
году

Насколько глубокие должны быть сети?

- AlexNet – 2012. Uses max pooling instead of average pooling. Uses ReLU activation instead of tanh. Takes in 3 channels (RGB) as input. Obtained significantly better results in ImageNet Large Scale Visual Recognition Challenge compared to other models at the time.



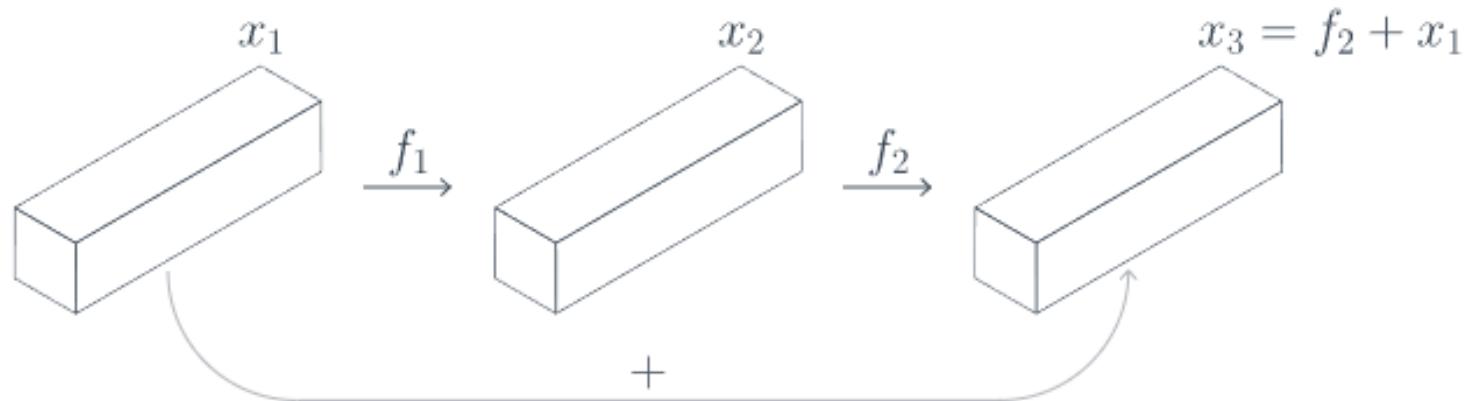
Layer	Type	Maps	Size	Kernel size	Stride	Padding	Activation
out	Fully connected	--	1000	--	--	--	Softmax
F9	Fully connected	--	4096	--	--	--	ReLU
F8	Fully connected	--	4096	--	--	--	ReLU
C7	Convolution	256	13*13	3*3	1	Same	ReLU
C6	Convolution	384	13*13	3*3	1	Same	ReLU
C5	Convolution	384	13*13	3*3	1	Same	ReLU
S4	Max Pooling	256	13*13	3*3	2	Valid	--
C3	Convolution	256	27*27	5*5	1	Same	ReLU
S2	Max Pooling	96	27*27	3*3	2	Valid	--
C1	Convolution	96	55*55	11*11	4	Valid	ReLU
In	Input	3(RGB)	227*227	--	--	--	--

AlexNet Summary

- Чем больше сверточ, тем лучше?

Если мы будем бесконтрольно добавлять сверточные слои, то, несмотря на использование ReLU и batchNorm, градиенты все равно будут затухать и на первых слоях будут близки к нулю

- Residual connection (архитектура ResNet, Residual NN) – прокидываем признаки на предыдущем слое мимо сверточ на следующем

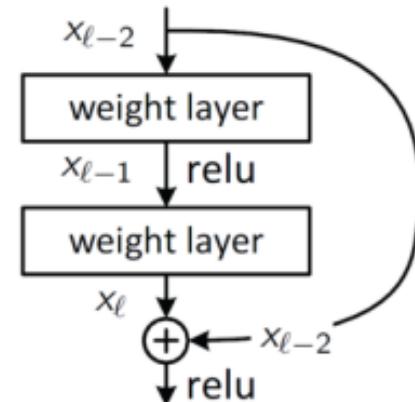


- DenseNet – вместо суммы конкатенация, результаты лучше, но вычислительно сложно

Сквозная связь (skip connection) слоя ℓ
с предшествующим слоем $\ell - d$:

$$x_\ell = \sigma(Wx_{\ell-1}) + x_{\ell-d}$$

Слой ℓ выучивает не новое векторное представление x_ℓ , а его приращение $x_\ell - x_{\ell-d}$



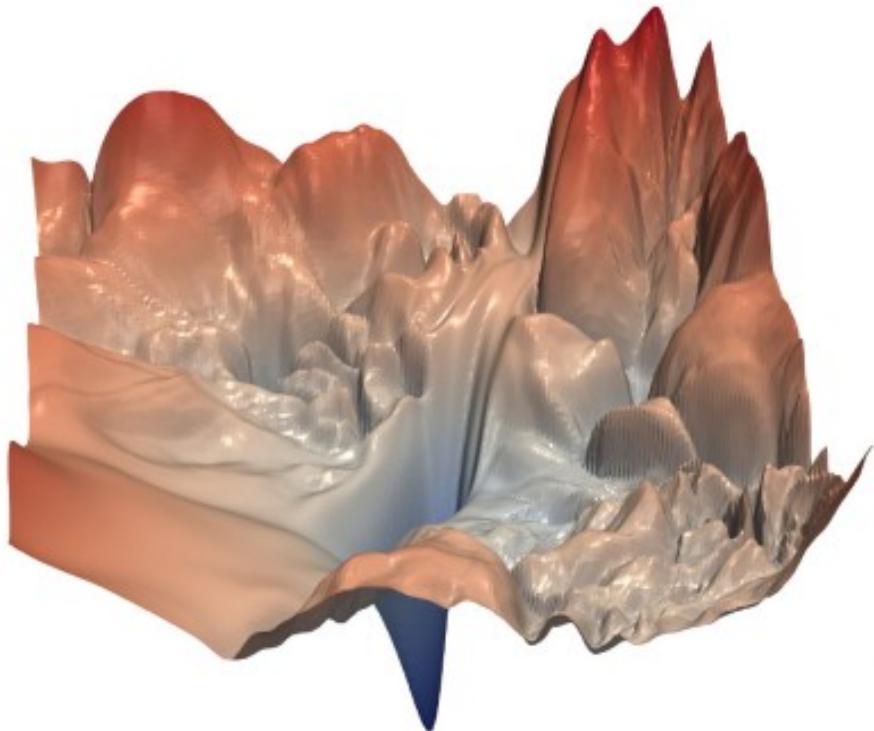
- Приращения более устойчивы \Rightarrow улучшается сходимость
- Появляется возможность увеличивать число слоёв
- Обобщение — Highway Networks:

$$x_\ell = \sigma(Wx_{\ell-1}) \underbrace{\tau(W'x_{\ell-1})}_{\text{transform gate}} + x_{\ell-d} \underbrace{(1 - \tau(W'x_{\ell-1}))}_{\text{carry gate}}$$

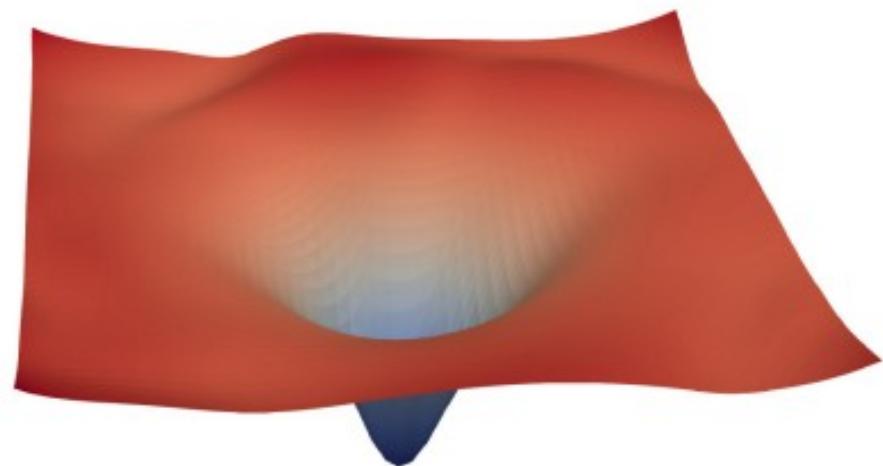
Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. 2015

R.K.Srivastava, K.Greff, J.Schmidhuber. Highway Networks. 2015

Сквозные связи (skip connection) упрощают оптимизируемый критерий, устранивая локальные экстремумы и седловые точки:



without skip connections



with skip connections

Hao Li et al. Visualizing the Loss Landscape of Neural Nets. 2018

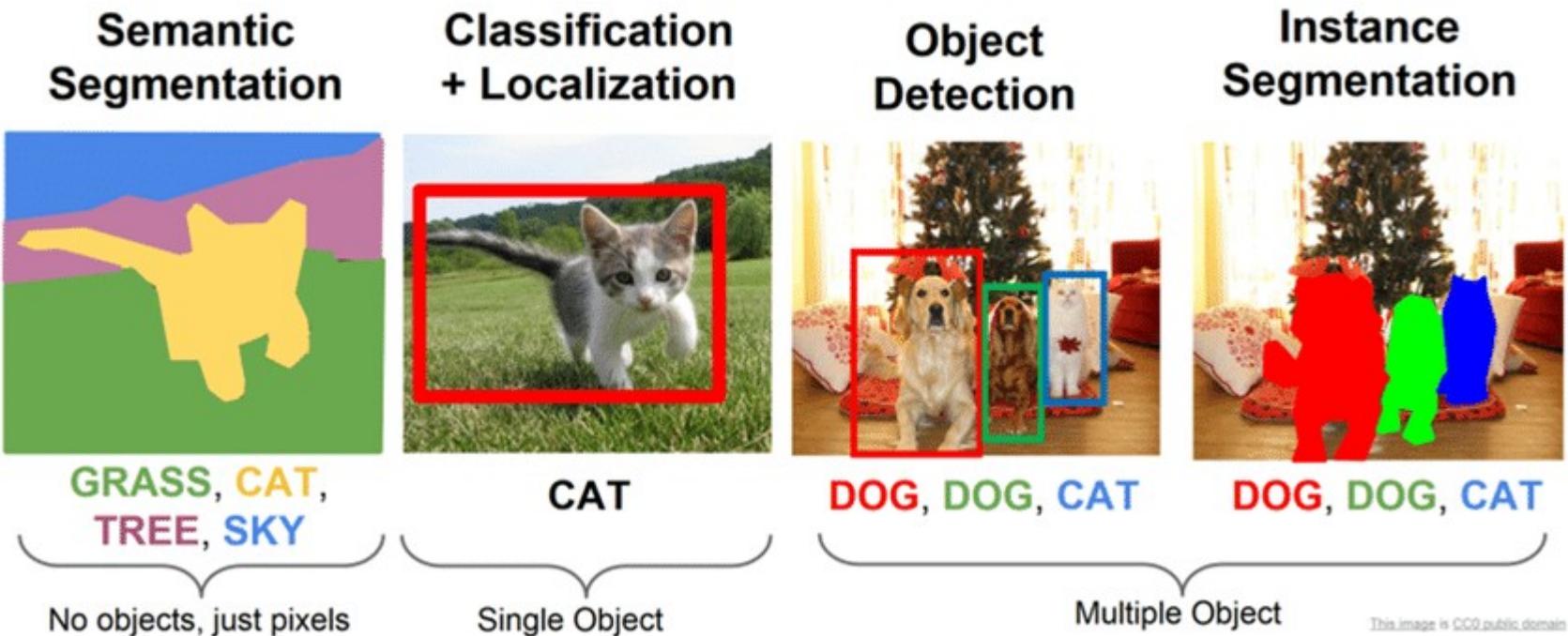
Знаковые архитектуры в мире сверточных сетей для задач классификации изображений:

1. [LeNet - 1998](#)
2. [AlexNet - 2012](#)
3. [Network in network - 2013](#)
4. [VGG - 2014](#)
5. [GoogLeNet \(inception\) - 2014](#)
6. [ResNet - 2015](#)
7. [MobileNet - 2017](#)
8. [EfficientNet - 2019](#)

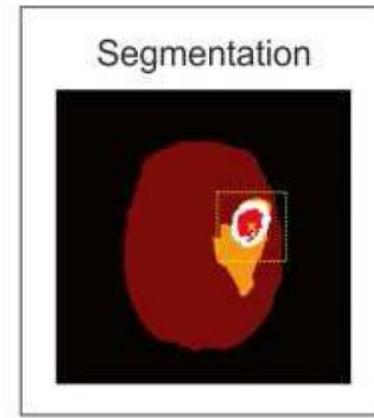
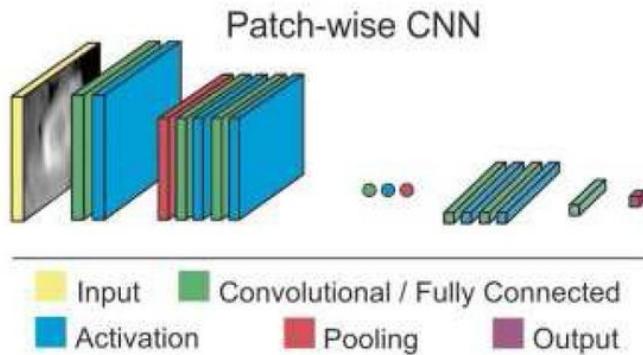
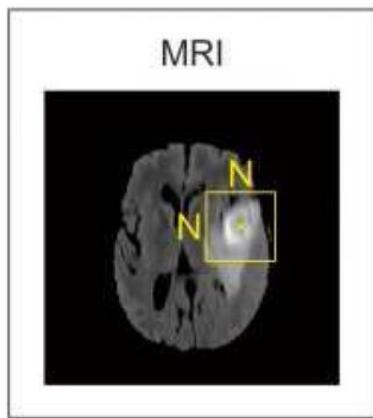
Что учитываем при обучении CNN:

- Функции активации без горизонтальных асимптот (ReLU)
- Адаптивные градиентные методы
- Dropout
- Batch normalization
- Residual NN
- Подбираем число слоев и размеры слоев
- Используем аугментацию

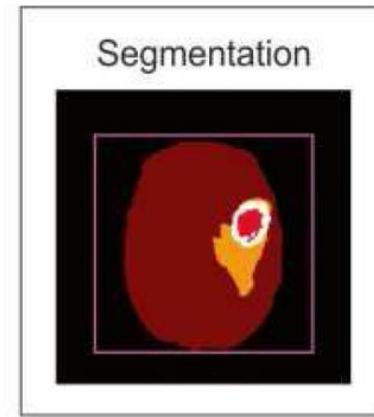
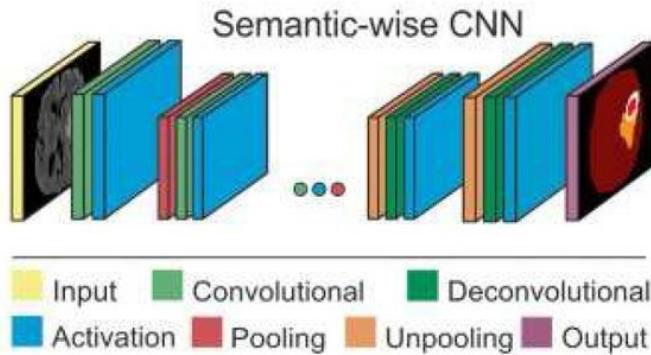
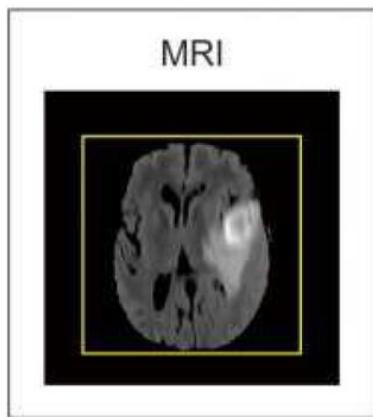
Практическое применение CNN



Применение в медицине

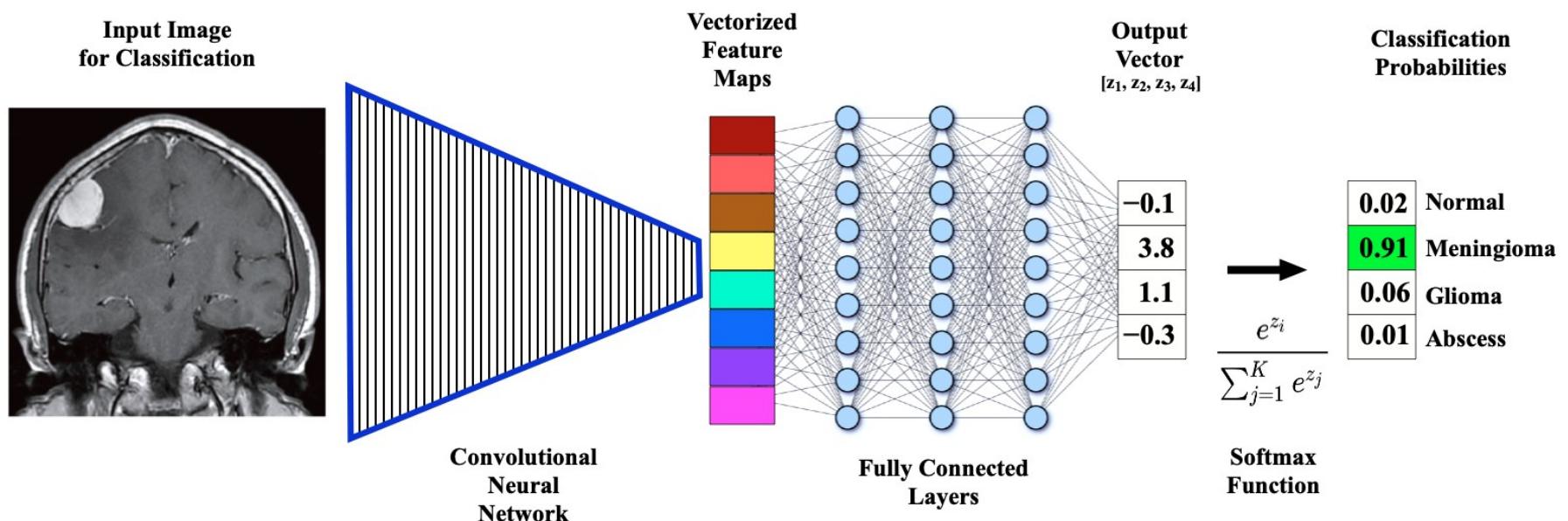


(a) Voxel-wise CNN architecture

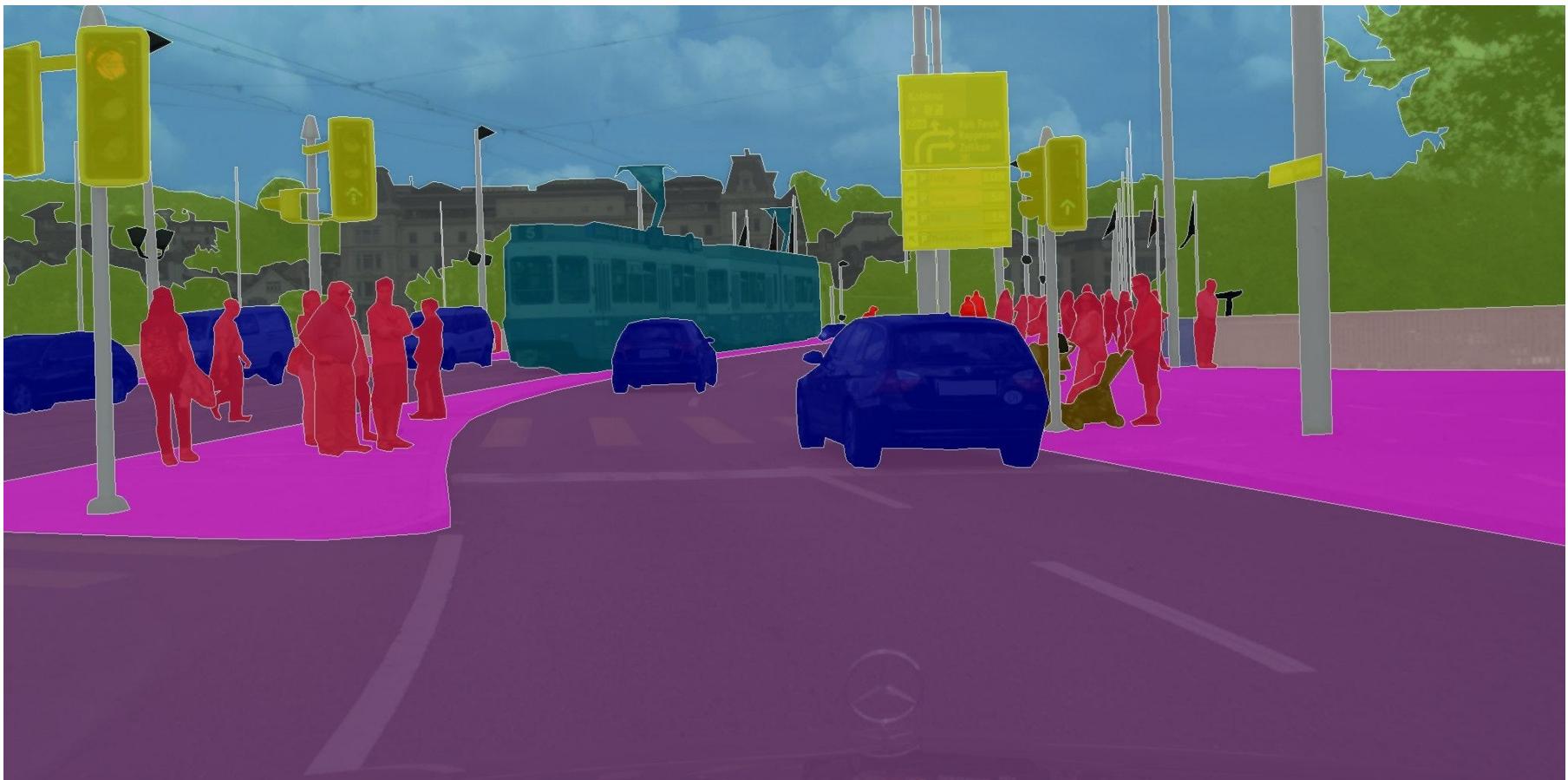


(b) Semantic-wise CNN architecture

Применение в медицине



Применение в беспилотном транспорте



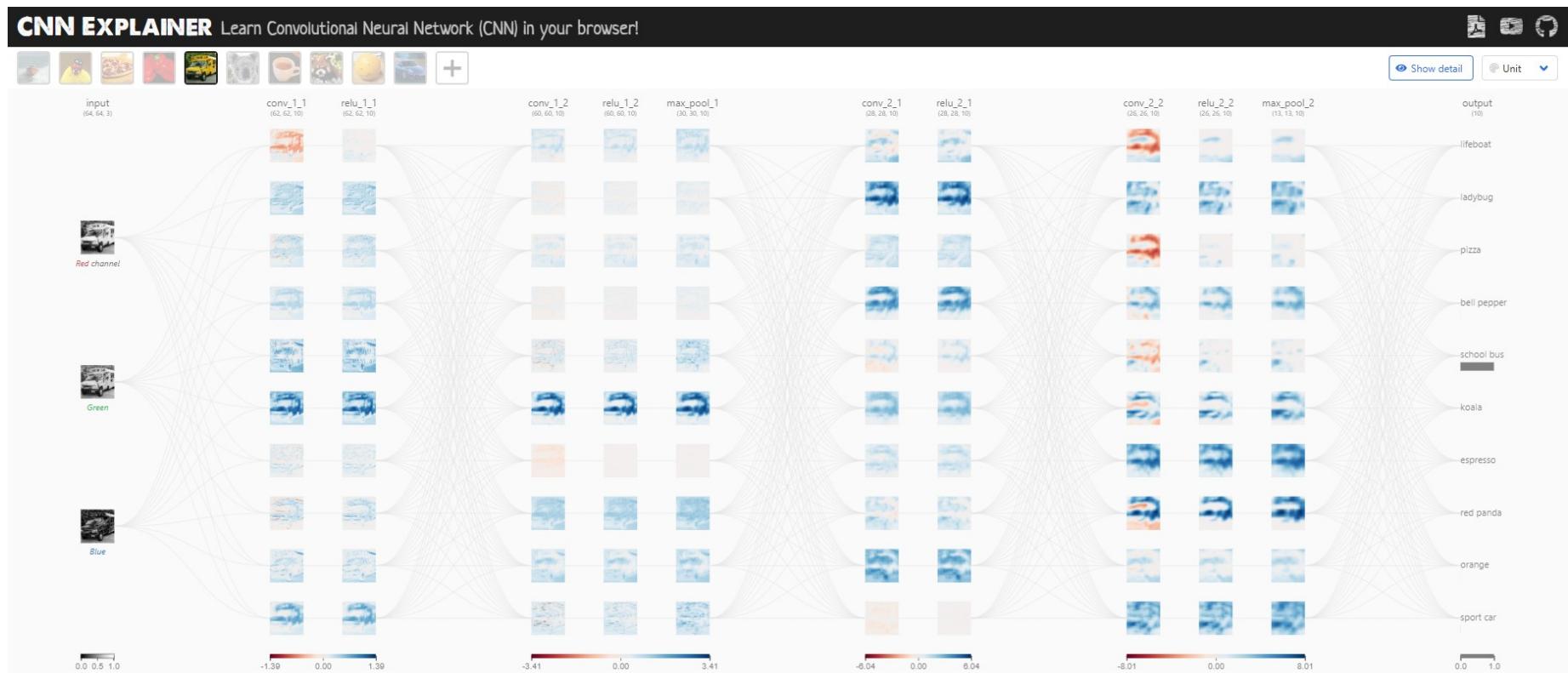
Применение в беспилотном транспорте



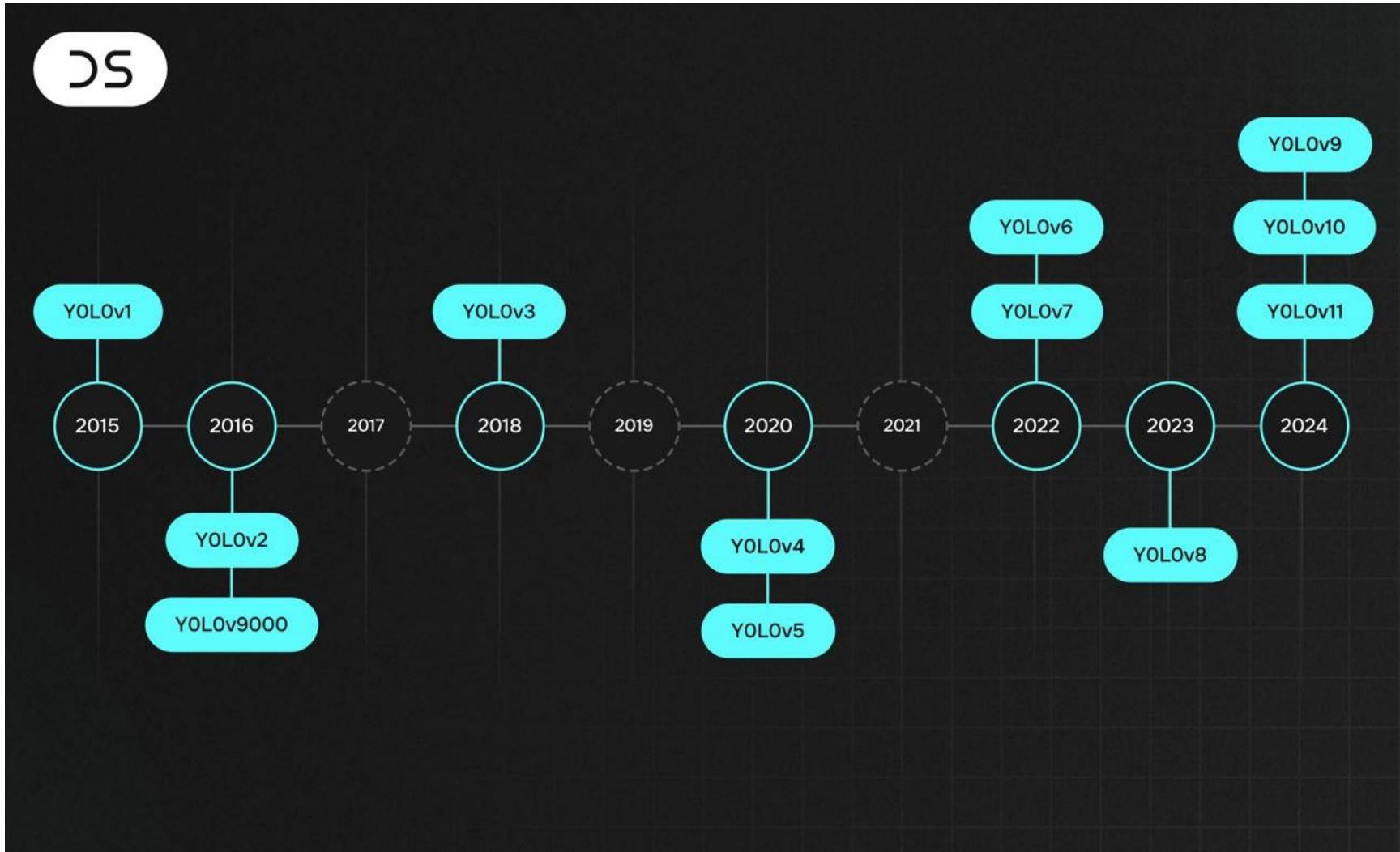
Применение в промышленности (поиск дефектов)



CNN explainer – наглядное средство визуализации CNN



You Only Look Once - YOLO



YOLOv1

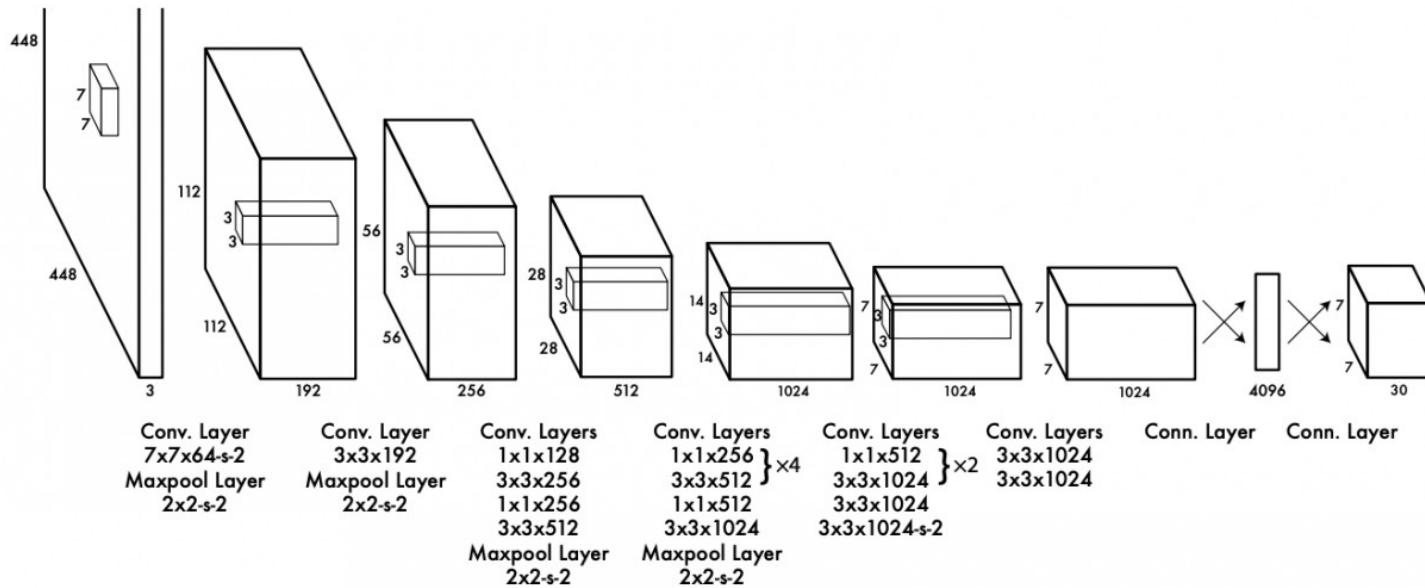
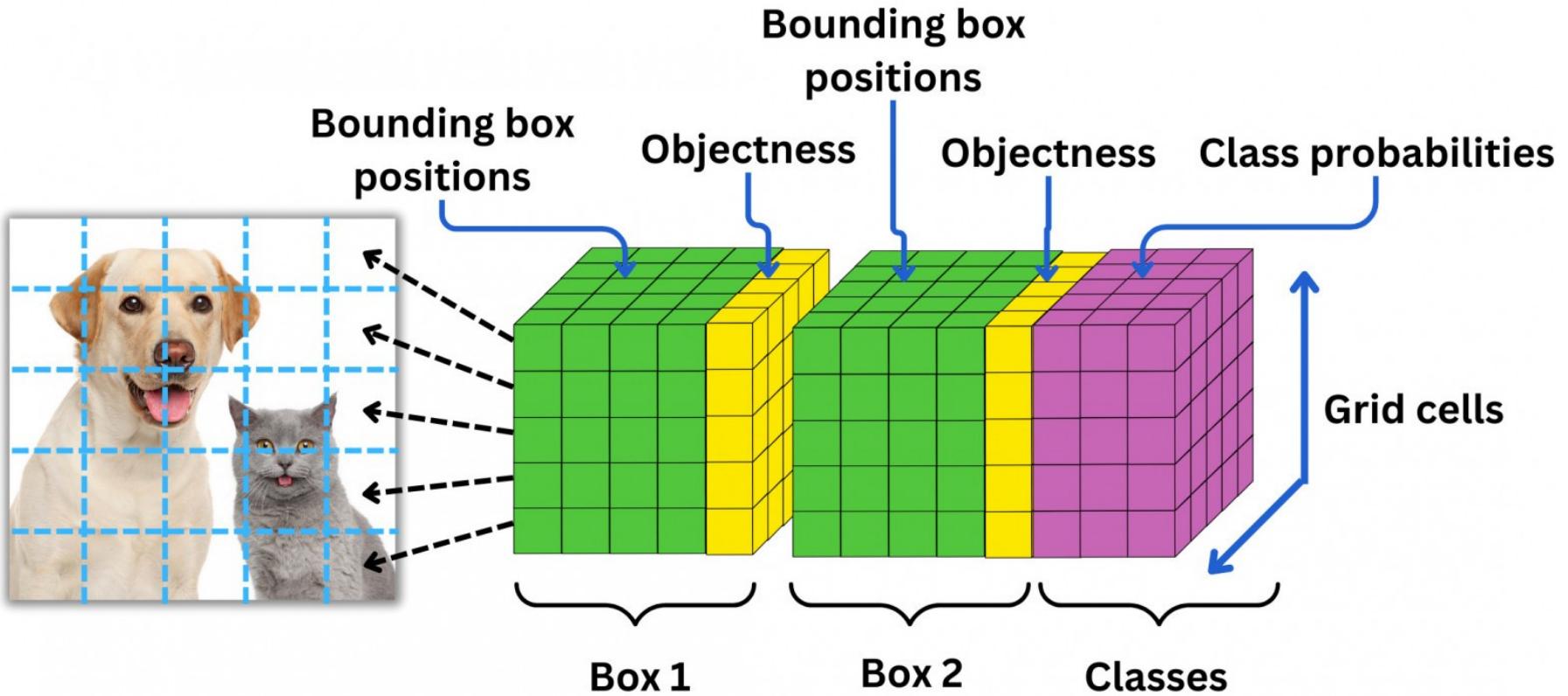


Figure 3: The Architecture. Our detection network has 24 convolutional layers followed by 2 fully connected layers. Alternating 1×1 convolutional layers reduce the features space from preceding layers. We pretrain the convolutional layers on the ImageNet classification task at half the resolution (224×224 input image) and then double the resolution for detection.

YOLO представляет из себя единую сеть, которая сразу предсказывает координаты некоторого количества б-боксов вместе с их характеристиками, такими, как вероятность класса

YOLOv1



Выходной тензор сети имеет размер $7 \times 7 \times 30$. То есть для каждой из 7×7 ячеек нашего изображения модель предсказывает вектор из 30 чисел. Внутри этого вектора и скрывается описание б-боксов и меток классов. Первые 10 значений отвечают за координаты двух б-боксов-кандидатов: координаты центра + ширина + высота + confidence score, то есть уверенность модели в том, что внутри б-бокса находится центр объекта. Оставшиеся 20 (по количеству классов) значений вектора ответственны за метки классов, то есть оценку вероятности того, что объект определенного класса присутствует в ячейке.

YOLOv1

Regression
loss

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

Confidence
loss

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

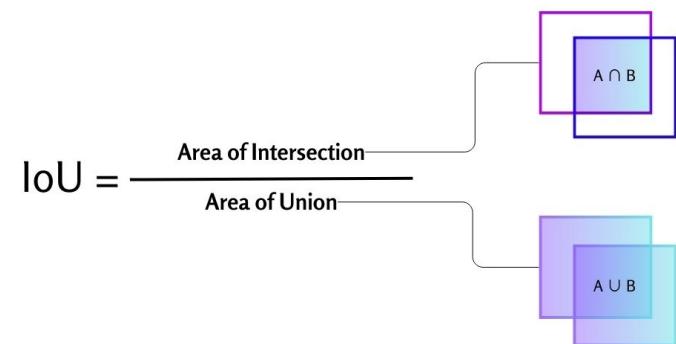
Classification
loss

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

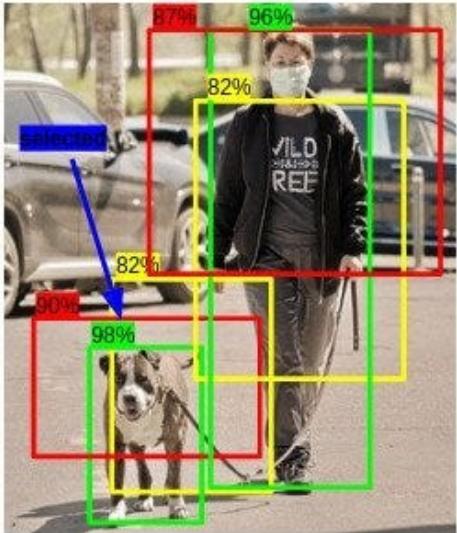
Regression loss – ошибка в предсказании координат центра (x, y), высоты (h), ширины (w) б-боксов для объектов i с боксами j .

Classification loss – ошибка на всех метках классов.

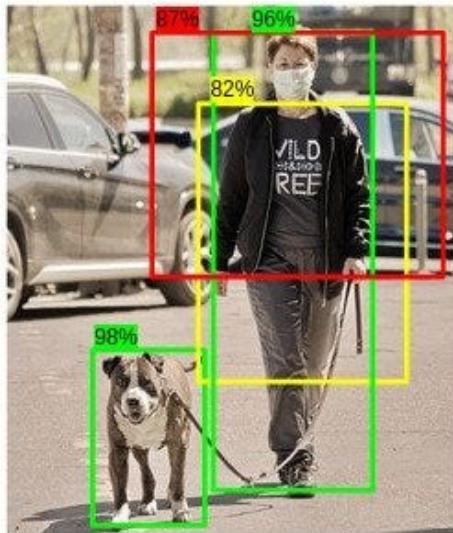
Confidence loss – оценка уверенности модели в том, что внутри б-бокса находится центр объекта. Это предсказанное значение функции IoU – Intersection over Union. IoU – перекрытие между прогнозируемой рамкой и истинным значением из обучающей выборки



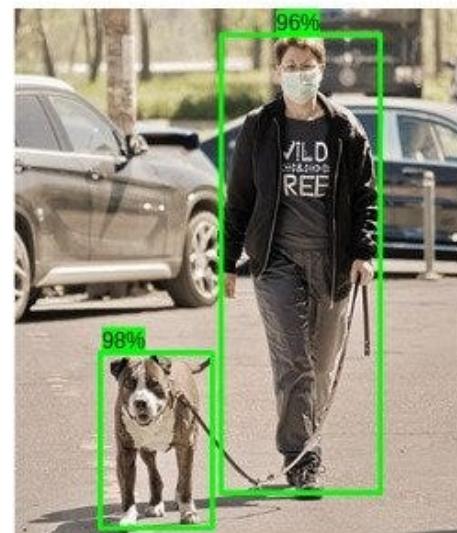
YOLOv1



Step 1: Selecting Bounding box with highest score



Step 3: Delete Bounding box with high overlap



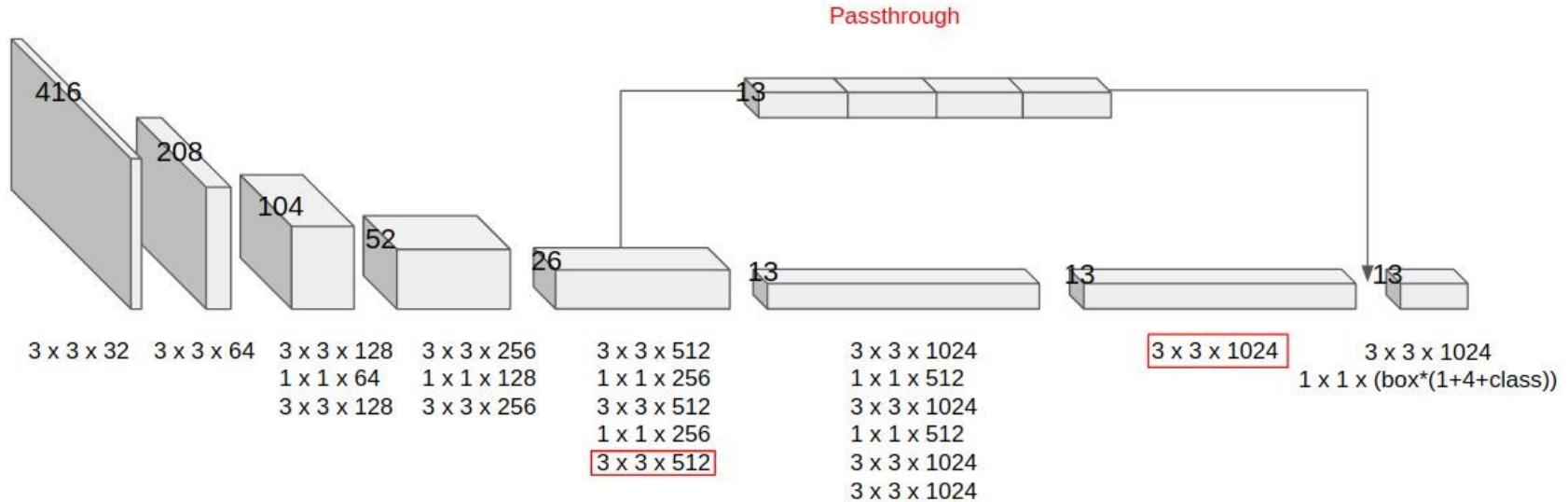
Step 5: Final Output

- Выбираем б-боксы с наивысшим скором.
- Удаляем все рамки, для которых $\text{IoU} < 0.5$.

Алгоритм Non-maximum Suppression:

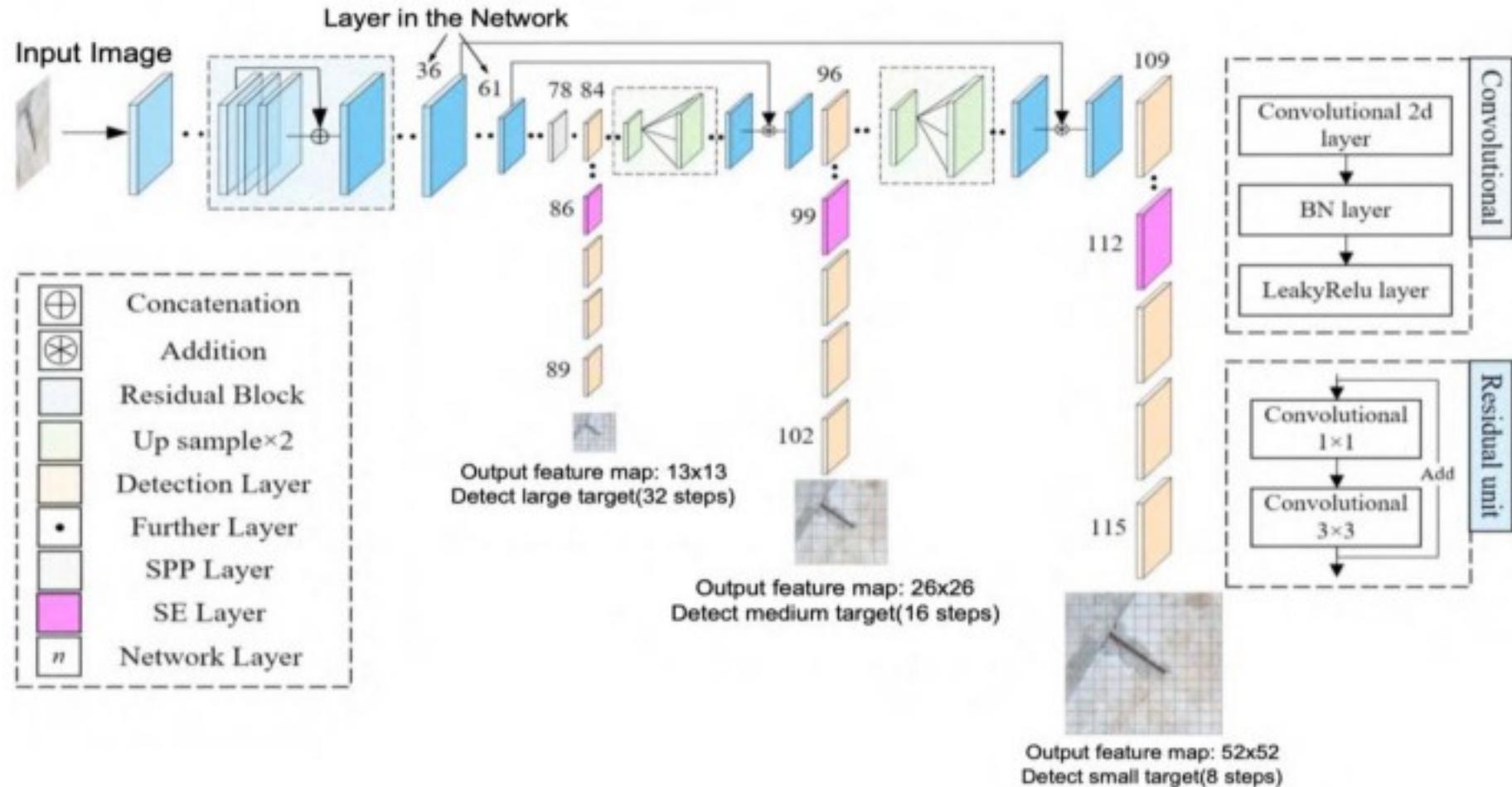
- Сначала мы берем список б-боксов, которые остались после пункта 2 и сортируем его по убыванию IoU, так, чтобы в начале были те б-боксы, в которых, как думает модель, расположены центры объектов.
- Затем будем по очереди брать самых "вероятных" кандидатов и находить все б-боксы, которые пересекаются с ними настолько, что IoU этого пересечения больше некоторого порога. Все такие б-боксы мы будем удалять и вычеркивать из исходного списка, и так до тех пор, пока список не опустеет, а у нас не появится набор финальных отфильтрованных б-боксов.

YOLOv2

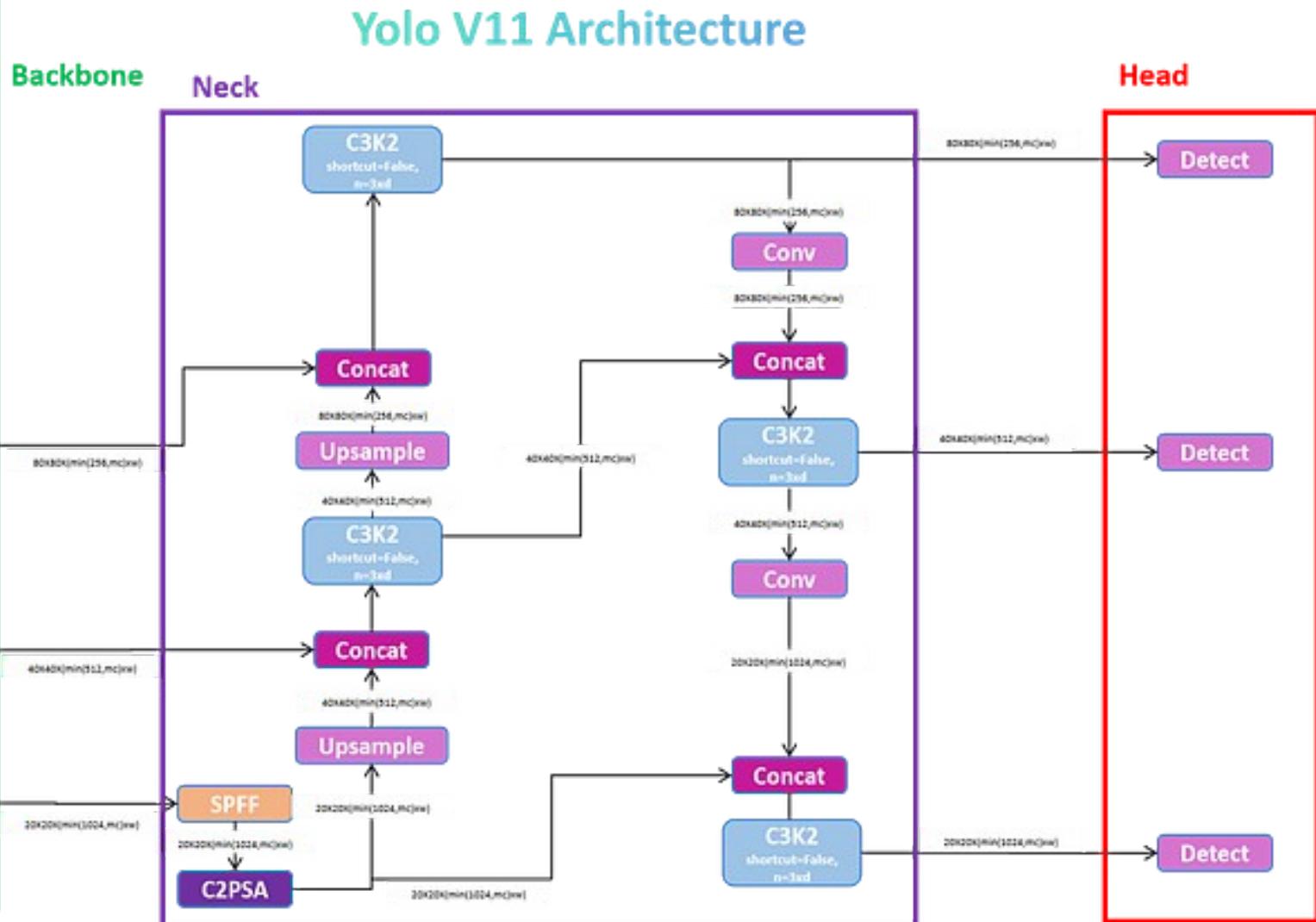


1. Изменение базовой архитектуры с GoogLeNet на Darknet-19.
2. Изменение структуры сети, добавление skip connection.
3. Вместо предсказания б-боксов используется предсказание anchor boxes.

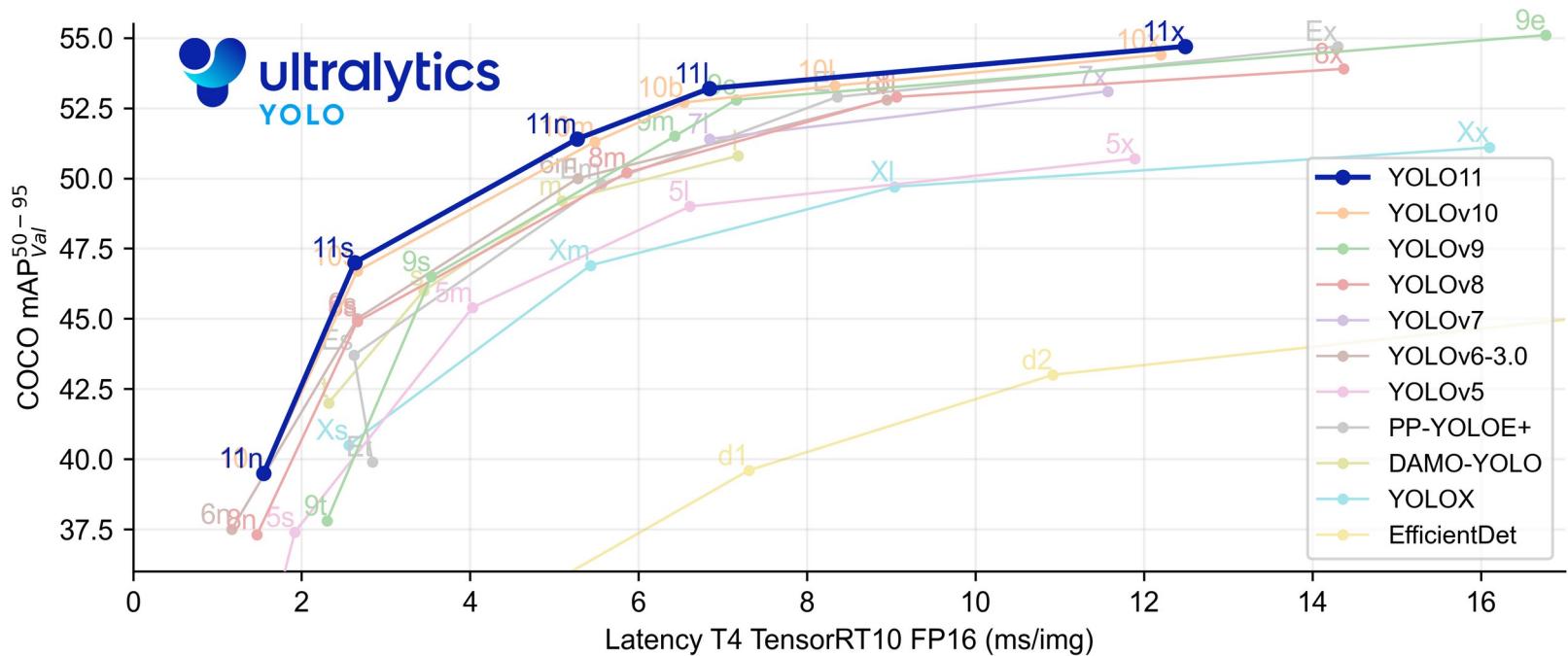
YOLOv3



YOLOv11



Сравнение моделей



YOLOv12

Key Features:

- **Area Attention Mechanism:** A new self-attention approach that processes large receptive fields efficiently. It divides [feature maps](#) into $/$ equal-sized regions (defaulting to 4), either horizontally or vertically, avoiding complex operations and maintaining a large effective receptive field. This significantly reduces computational cost compared to standard self-attention.
- **Residual Efficient Layer Aggregation Networks (R-ELAN):** An improved feature aggregation module based on ELAN, designed to address optimization challenges, especially in larger-scale attention-centric models. R-ELAN introduces:
 - Block-level residual connections with scaling (similar to layer scaling).
 - A redesigned feature aggregation method creating a bottleneck-like structure.
- **Optimized Attention Architecture:** YOLOv12 streamlines the standard attention mechanism for greater efficiency and compatibility with the YOLO framework. This includes:
 - Using FlashAttention to minimize memory access overhead.
 - Removing positional encoding for a cleaner and faster model.
 - Adjusting the MLP ratio (from the typical 4 to 1.2 or 2) to better balance computation between attention and feed-forward layers.
 - Reducing the depth of stacked blocks for improved optimization.
 - Leveraging convolution operations (where appropriate) for their computational efficiency.
 - Adding a 7×7 separable convolution (the "position perceiver") to the attention mechanism to implicitly encode positional information.
- **Comprehensive Task Support:** YOLOv12 supports a range of core computer vision tasks: object detection, [instance segmentation](#), [image classification](#), pose estimation, and oriented object detection (OBB).
- **Enhanced Efficiency:** Achieves higher accuracy with fewer parameters compared to many prior models, demonstrating an improved balance between speed and accuracy.
- **Flexible Deployment:** Designed for deployment across diverse platforms, from edge devices to cloud infrastructure.

YOLOv12

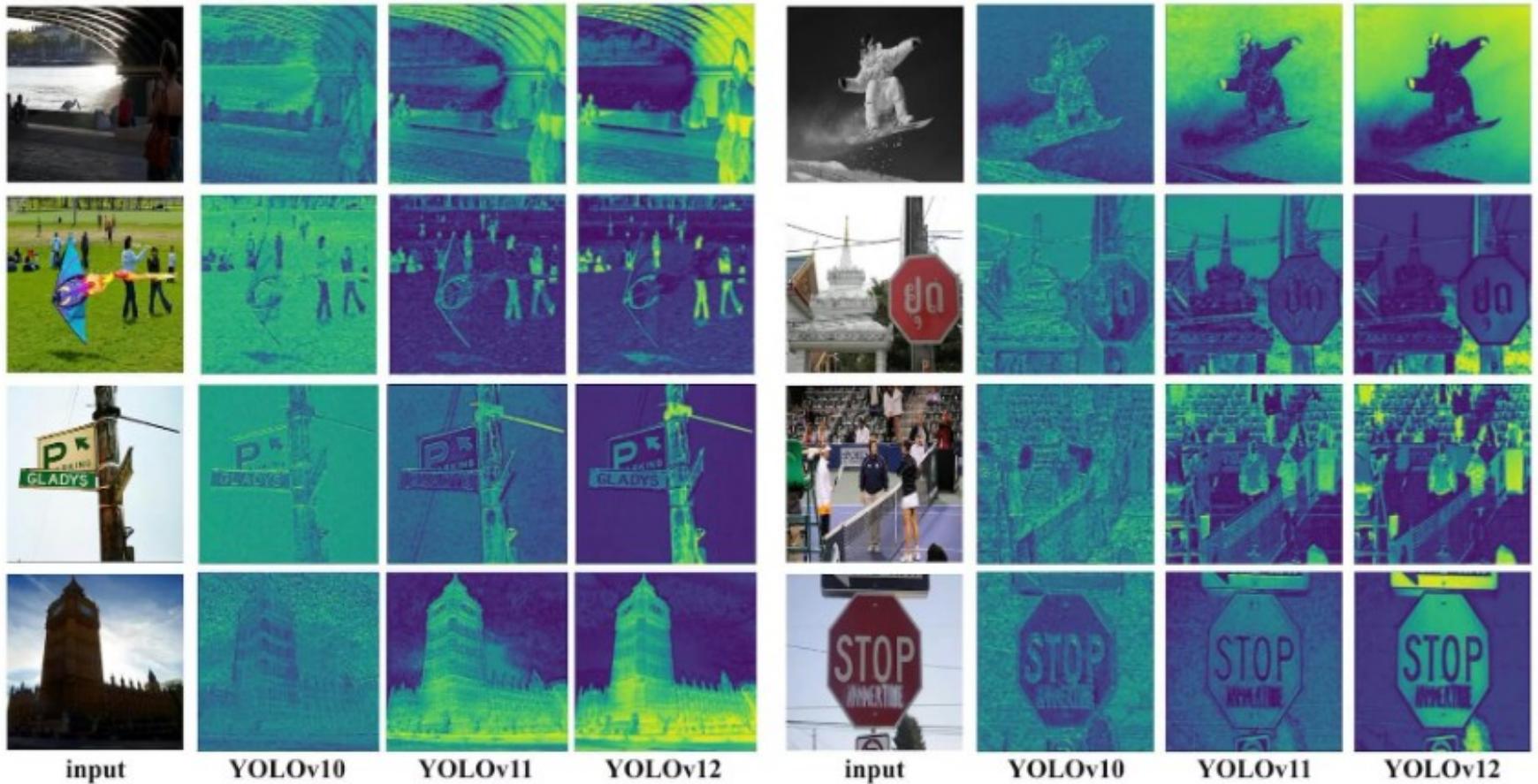
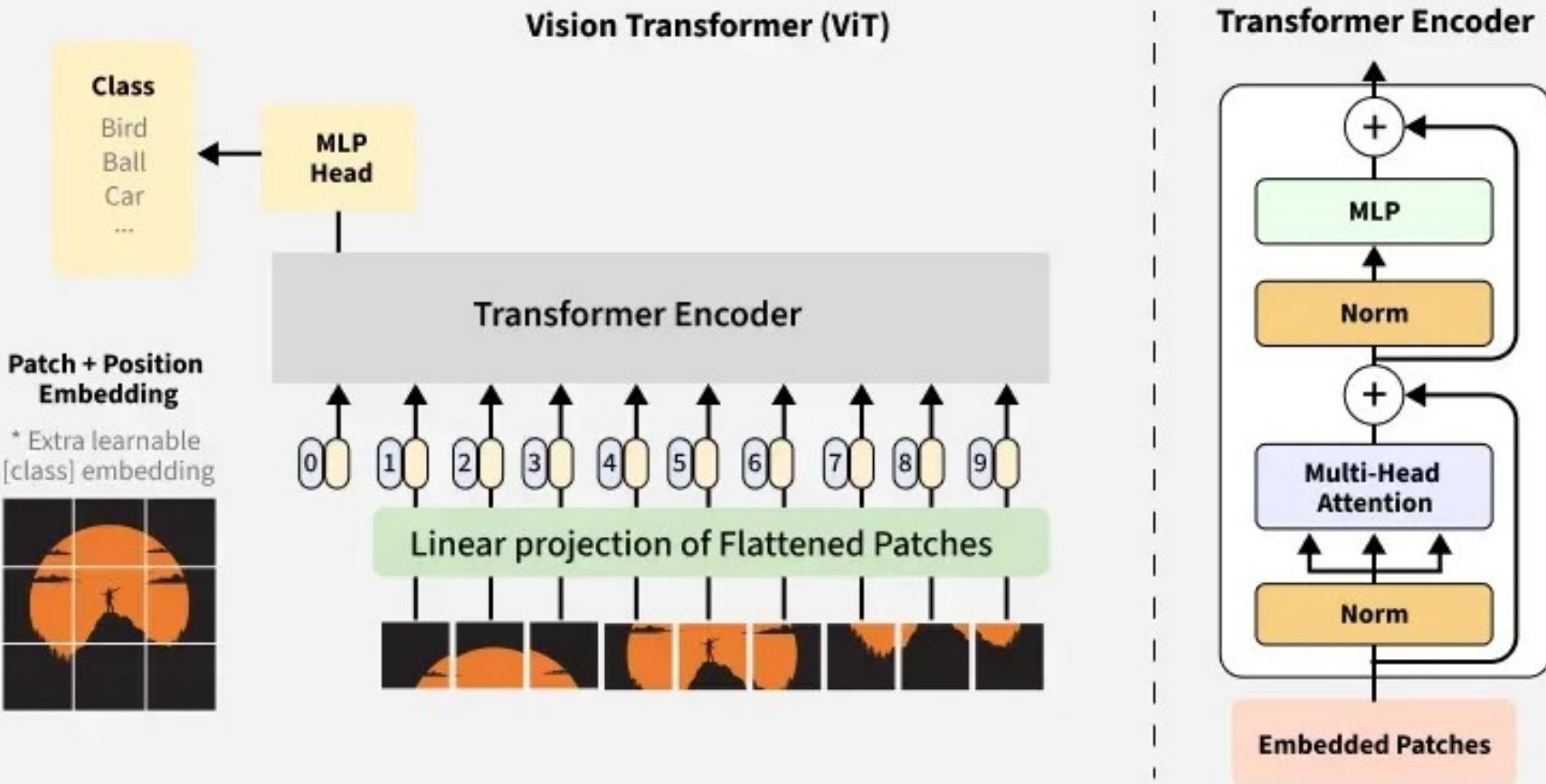
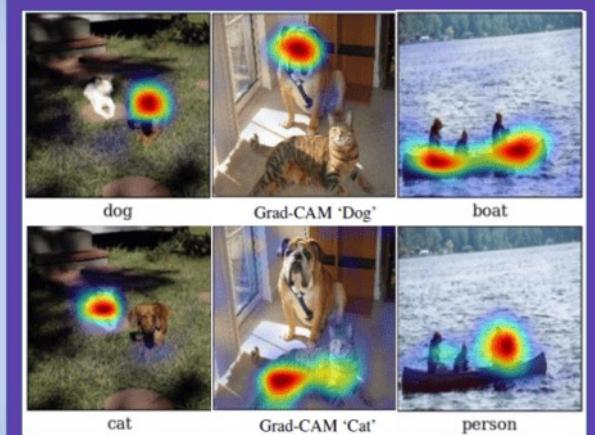


Figure 5. Comparison of heat maps between YOLOv10 [53], YOLOv11 [28], and the proposed YOLOv12. Compared to the advanced YOLOv10 and YOLOv11, YOLOv12 demonstrates a clearer perception of objects in the image. All the results are obtained using the X scale models. *Zoom in to compare the details.*

Visual Transformers



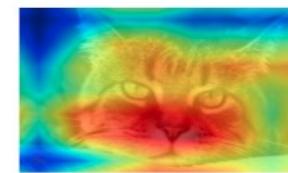
Интерпретация моделей: Gradient-weighted Class Activation Maps (Grad-CAM)



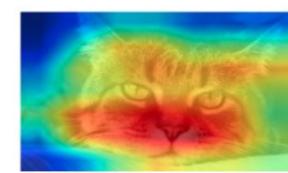
Input image



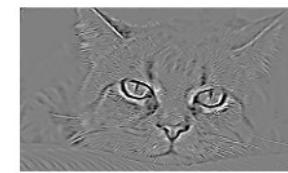
Class activation map



Grad-CAM



Guided Backpropagation



Guided Grad-CAM

