Нейронные сети в компоютерном зрении

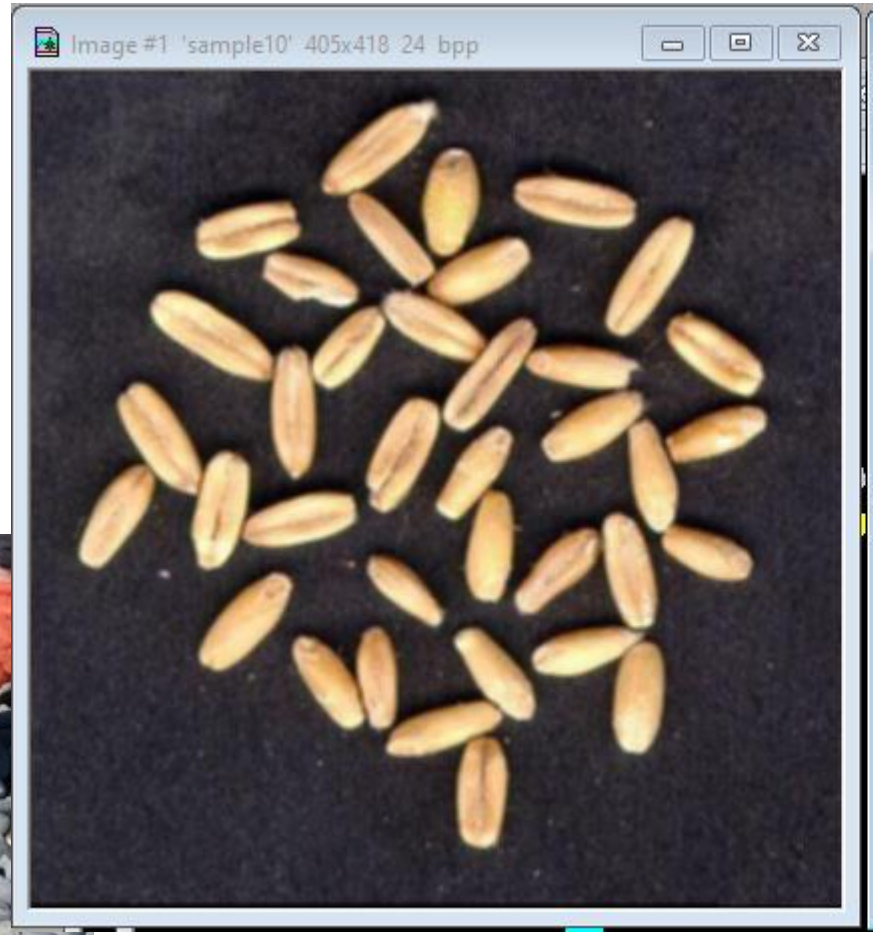
Особенности

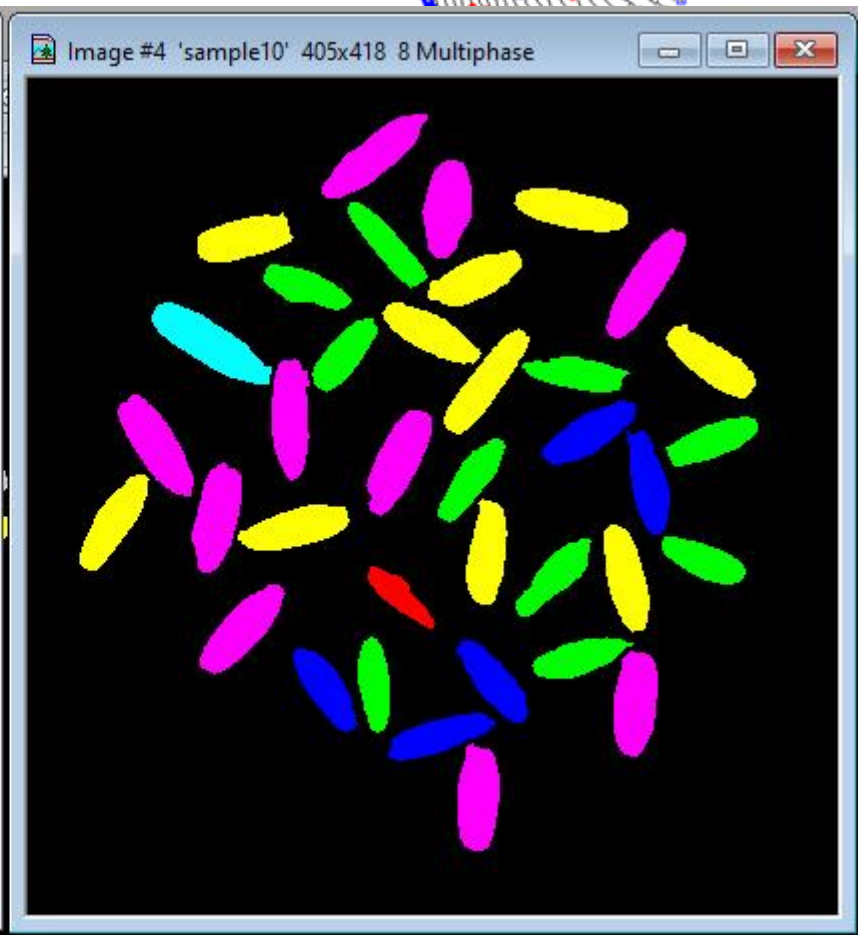
А.М.Недзьведь

Задачи, которые могут решать НС

- **идентификация**

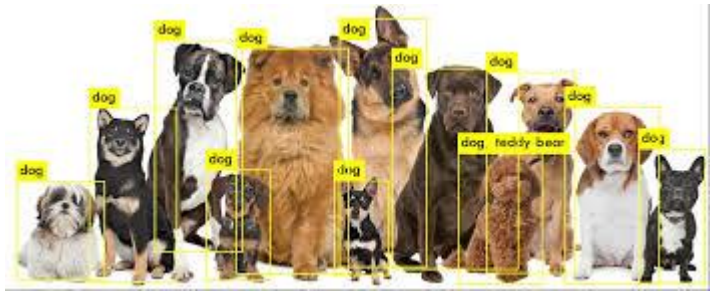
Идентификация — установление тождественности неизвестного объекта известному на основании совпадения признаков





Задачи, которые могут решать НС

- **Распознавание объектов**

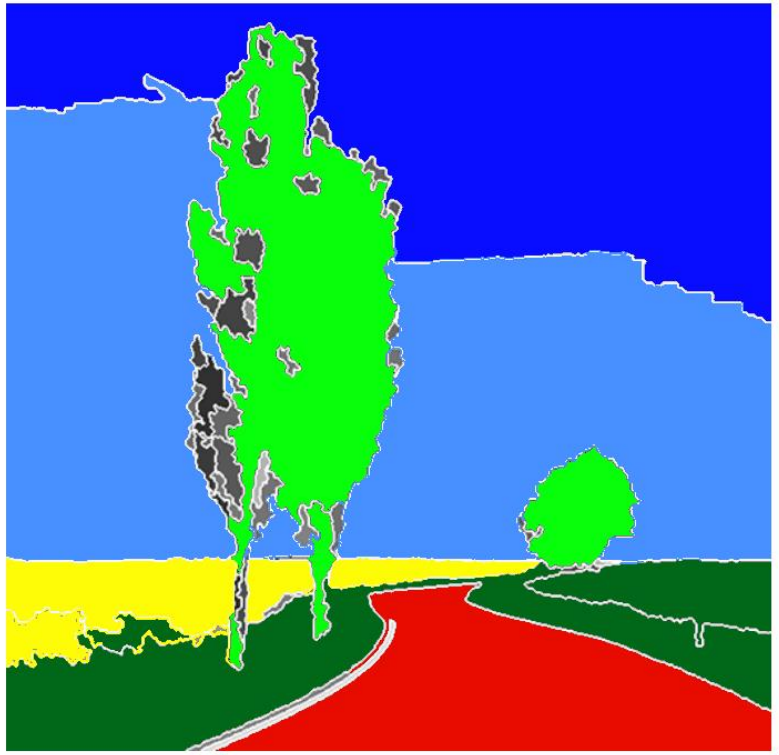


Задача состоит в том, чтобы по изображению суметь выделить на нем некоторый набор объектов. Возможно указать их расположение

Задачи, которые могут решать НС

- **Сегментация изображений**

Сегментация — это процесс разделения цифрового изображения на несколько сегментов (множество пикселей, также называемых суперпикселями). Сегментация изображений обычно используется для того, чтобы выделить объекты и границы на изображениях. Более точно, сегментация изображений — это процесс присвоения таких меток **<u>каждому</u>** пикселю изображения.



Задачи, которые могут решать НС

- **Оценка положения**

Заключается в выделении некоторого каркаса объекта (например скелета, если речь идет о людях) и определении положения этого каркаса на изображении. Этот скелет может быть использован в последствии например для предсказания направления движения.

Задачи, которые могут решать НС

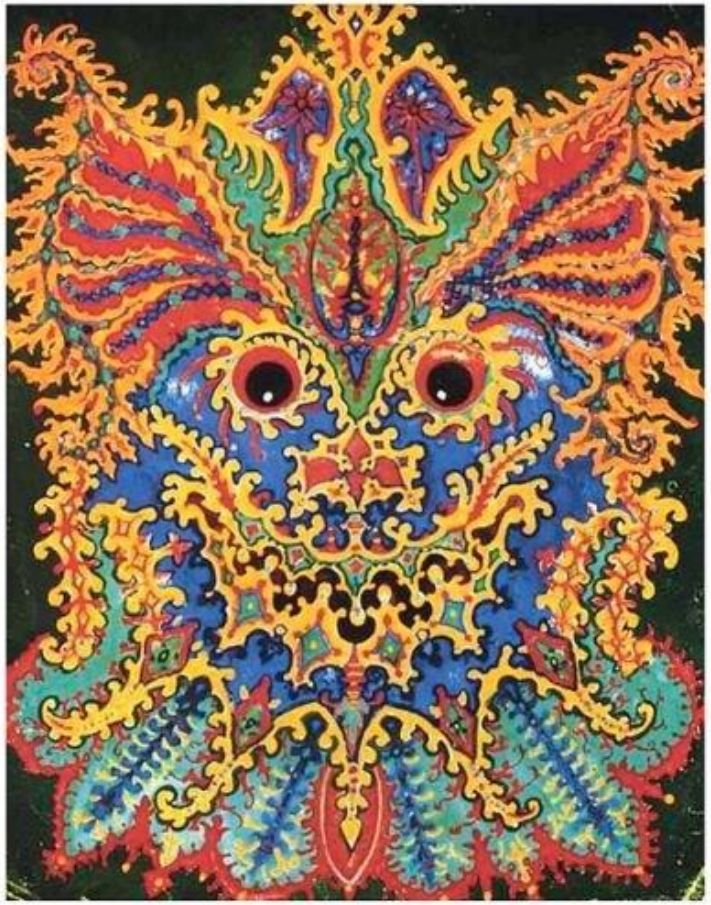
- Распознавание текста

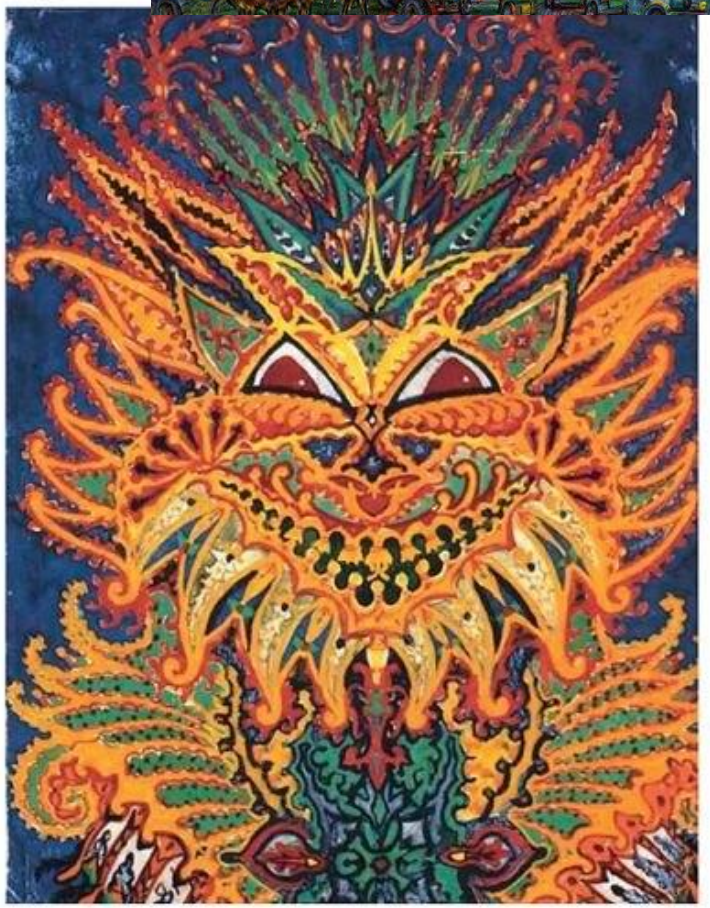
производится непосредственно распознавание текста например с помощью алгоритмов сегментации. При этом задачи распознавания текста написанного на листе бумаги, и распознавания текста написанного где-то на изображении, например текст на дорожном знаке, номер машины и т. д., сильно различаются, в силу наличия в последнем случае помех, которые мешают выделить конкретные буквы. В этом случае может помочь, например обучение предсказания буквы по остальным буквам в слове.

Задачи, которые могут решать НС

- Генерация объектов

Задача состоит в том, чтобы по известному набору объектов научится создавать похожие объекты, но при этом не совпадающие ни с одним из тестовых.





Постановка Задачи

На базе программно-аппаратного комплекса построить нейронную сеть – классификатор.

Сеть распознает представленные ей образы, например графические файлы с изображением цифр.

1 2 3

Реализация

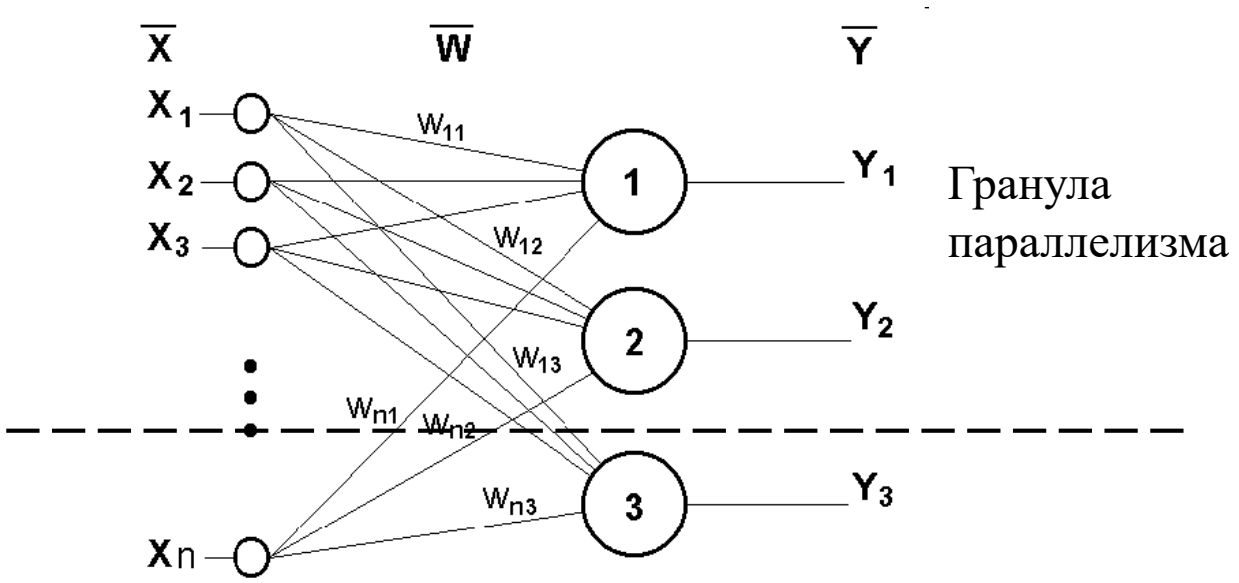


Рисунок 3. Однослойный персептрон

Гранула параллелизма группа из Np нейронов

Np=N/Nproc, где

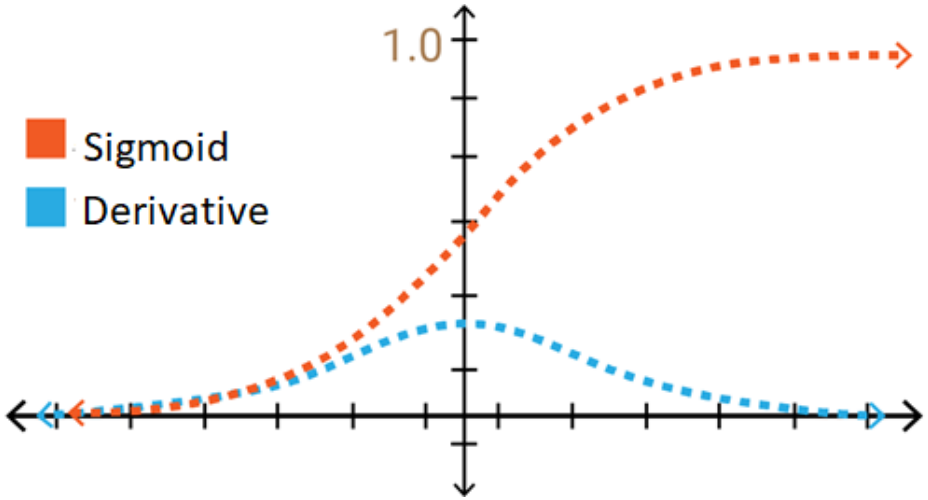
N-общее количество нейронов

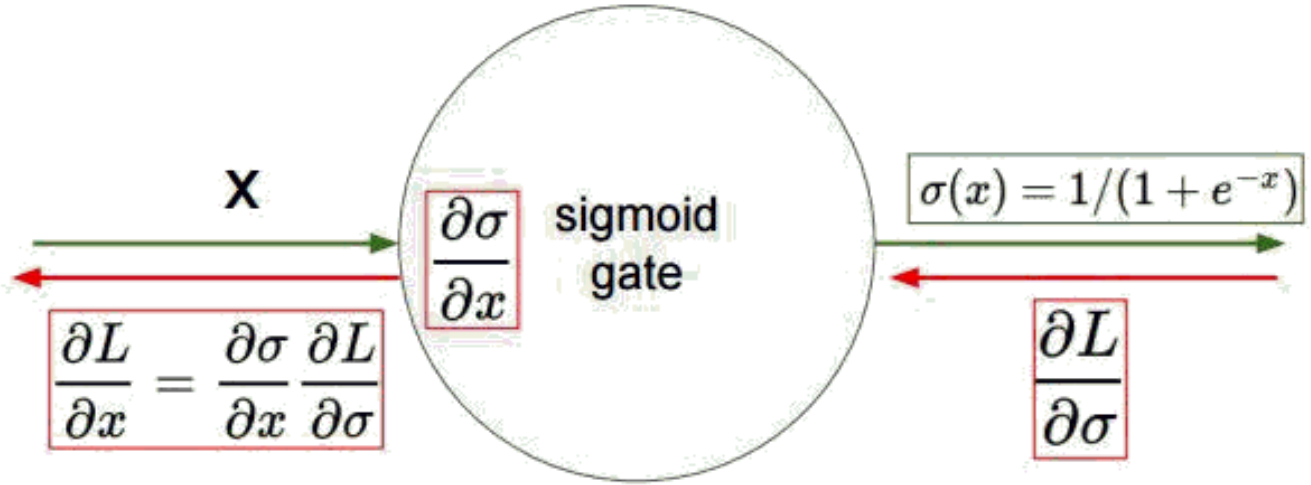
Nproc – количество процессоров в системе

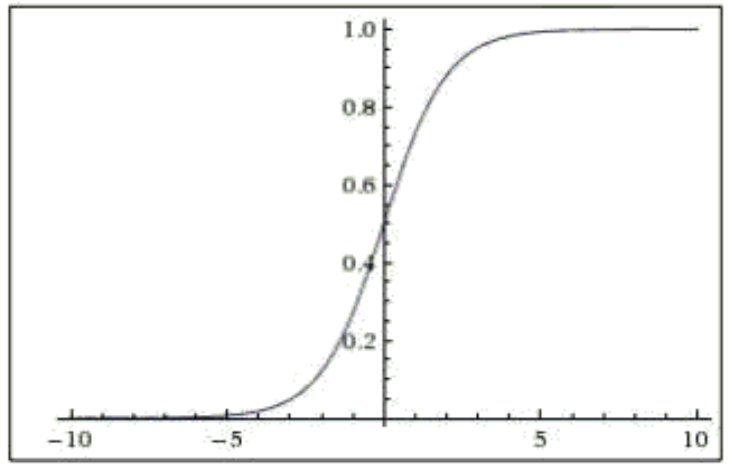
Функции Активации

Сигмоида $\sigma(x) = \frac{1}{1+e^{-x}}$.

Сигмоиды насыщают и убивают градиенты



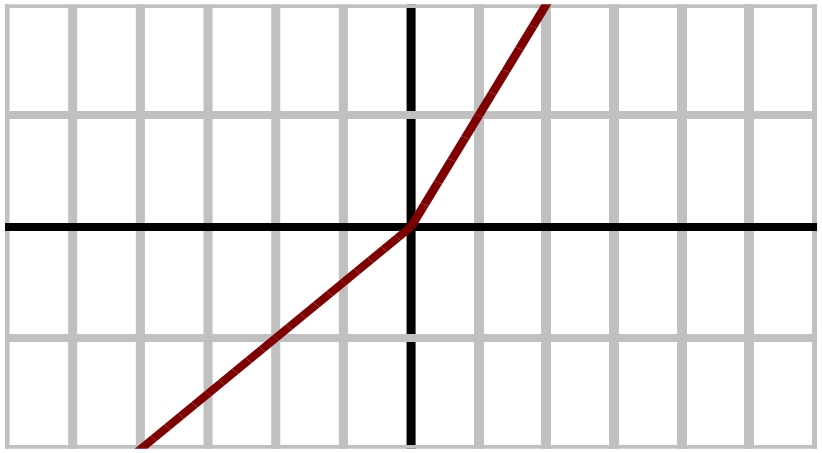




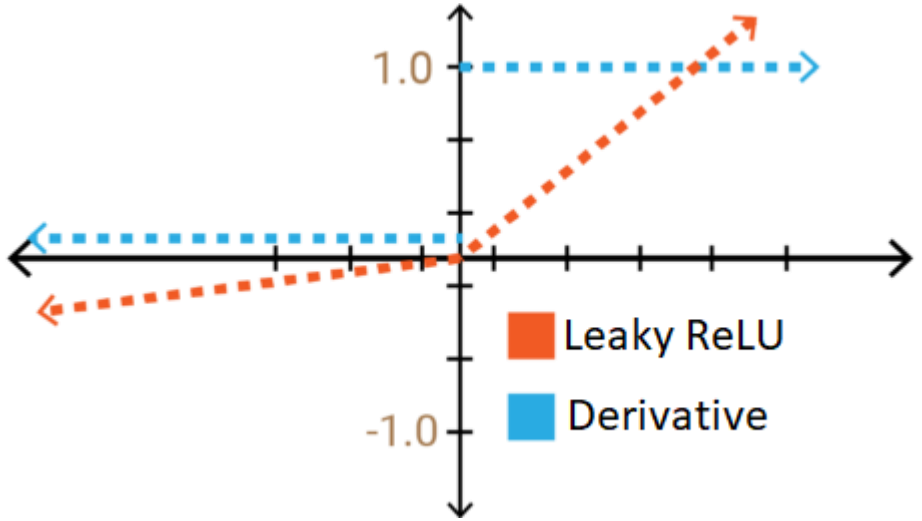
Функции Активации

Линейная функция на положительной оси но, прежде всего, ReLU не является линейным по своей природе.

Обучение не проходит в области нулевого значения



ReLU (выпрямленный линейный блок)



$$f(x) = \begin{cases} 0,01x & x < 0 \\ x & x \geqslant 0 \end{cases}$$

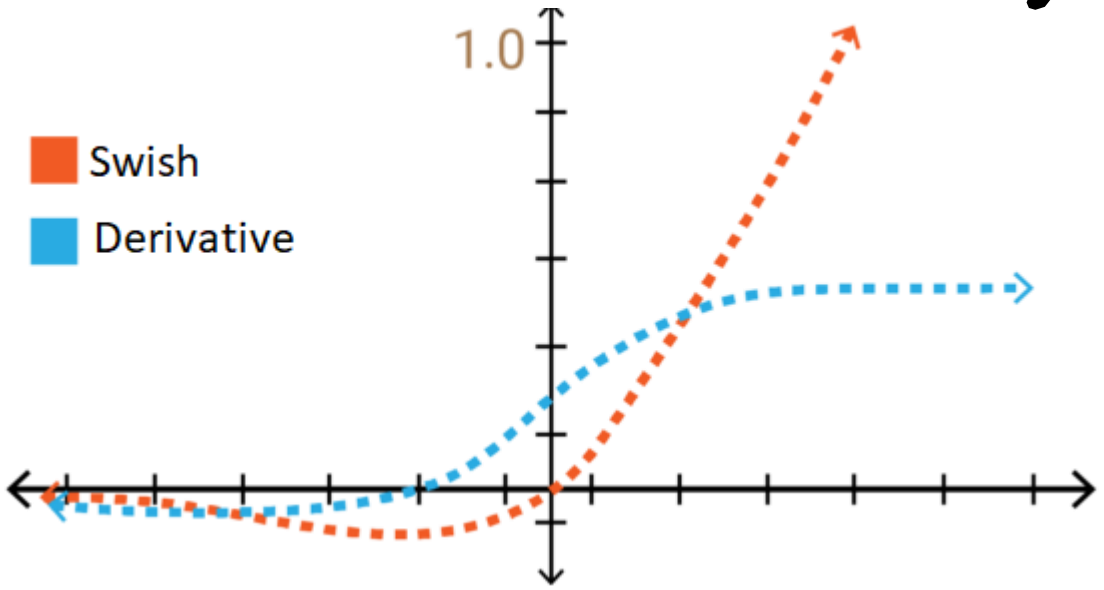
$$f'(x) = \begin{cases} 0,01 & x < 0 \\ 1 & x \geqslant 0 \end{cases}$$

Функции Активации

SOFTMAX $$f_i(\vec{x}) = \frac{e^{x_i}}{\sum_{j=1}^{J} e^{x_j}}$$

предпочтителен в выходном слое моделей глубокого обучения, особенно когда необходимо классифицировать.

Swish (самоограниченная) функция



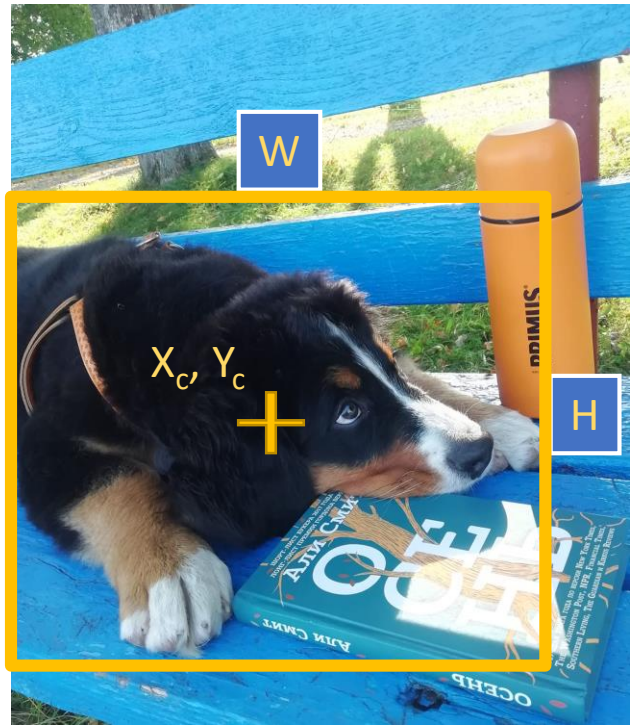
$$\frac{\partial f_i(\vec{x})}{\partial x_j} = f_i(\vec{x})(\delta_{ij} - f_j(\vec{x}))$$

Функции Активации

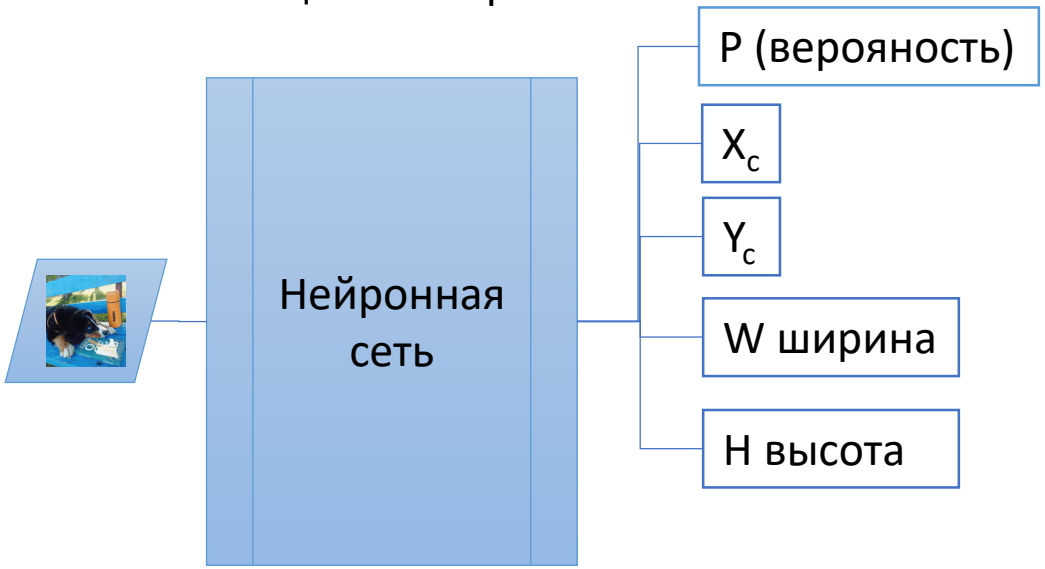
Простая и быстрая конвергенция сети может быть первым критерием.
ReLU будет выгоден с точки зрения скорости. Вы должны позволить градиентам умереть / исчезнуть. Обычно используется в промежуточных слоях, а не на выходе.
Leaky ReLU может стать первым решением проблемы исчезновения градиентов.
Для моделей глубокого обучения желательно начать эксперименты с ReLU.
Softmax обычно используется в выходных слоях.

ACTIVATION FUNCTION	TEST ACCURACY	TEST LOSS
Sigmoid Function	0,9866	0,0567
Hyperbolic Tanjant Function	0,9905	0,0474
ReLU	0,9929	0,0401
Leaky ReLU	0,9925	0,0403

Распознавание изображений



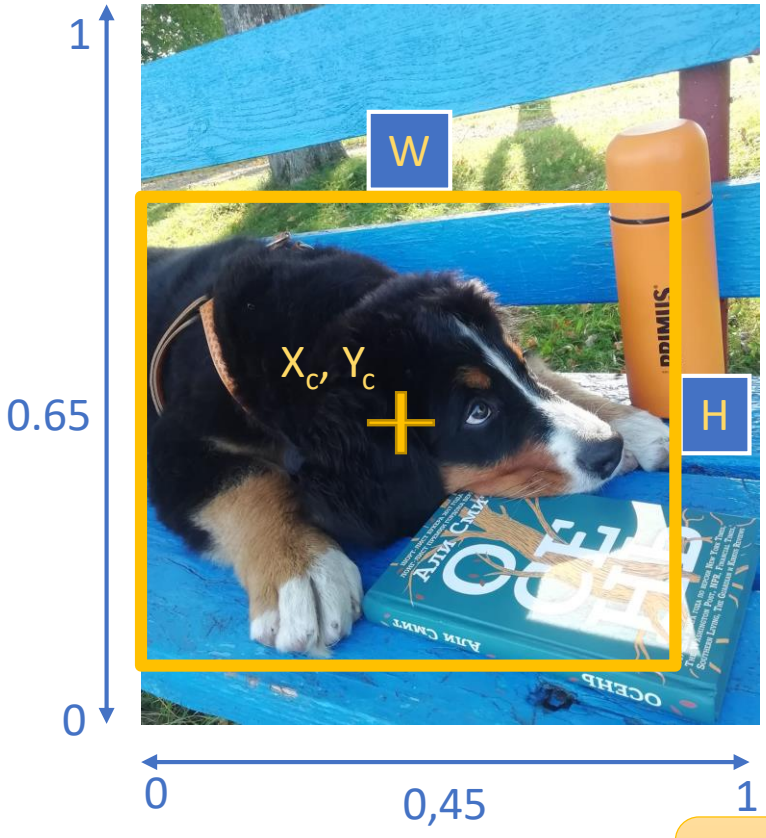
Есть ли щенок на фото?



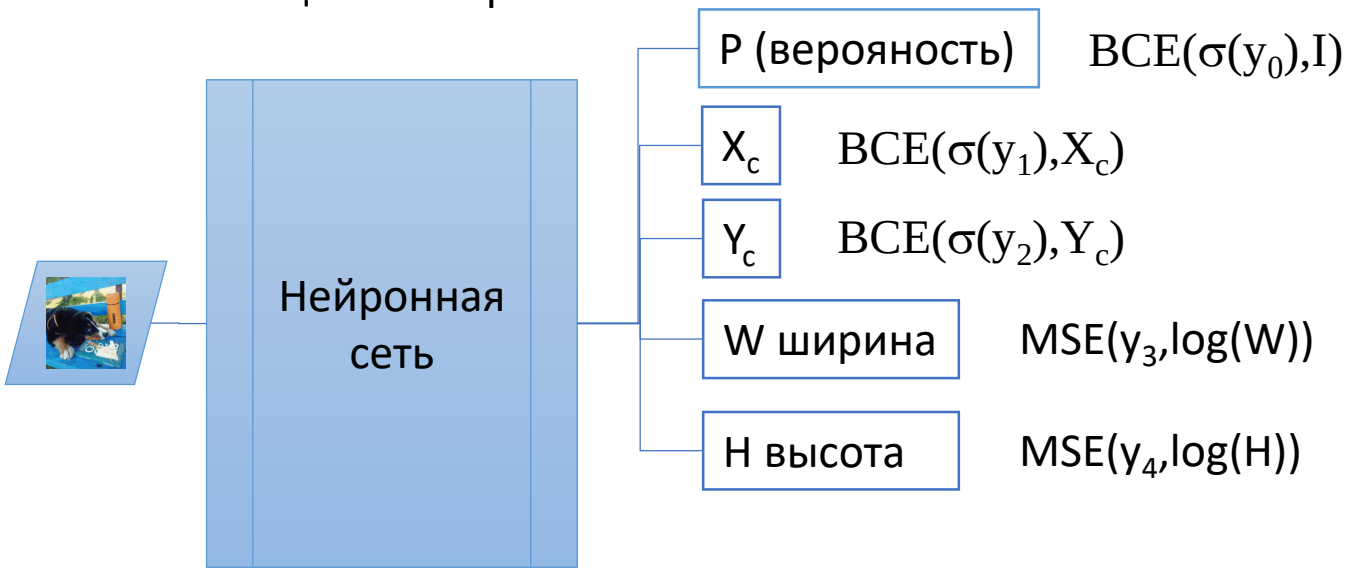
1. На изображении может быть один объект или ни одного
2. Центр в пределах изображения
3. Объект может выходить за пределы изображения

Распознавание изображений

Функция активации?
Функция потерь?



Есть ли щенок на фото?



$P=\sigma(y_0)$
$X_c=\sigma(y_1)$
$Y_c=\sigma(y_2)$
$W=\sigma(y_0)$
$H=\sigma(y_1)$

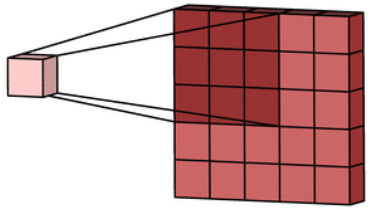
Для оценки в пределах картинки (объекты больше картинки):

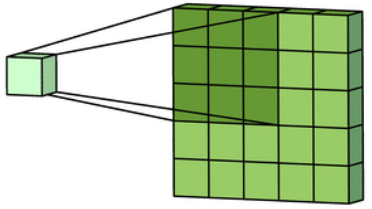
$$W = e^{y_3}$$
$$H = e^{y_4}$$

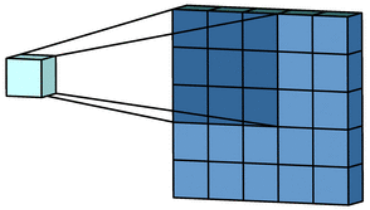
$$F = BCE(\sigma(y_0),I)+I(BCE(\sigma(y_1),X_c)+BCE(\sigma(y_2),Y_c)+MSE(y_3,\log(W))+MSE(y_4,\log(H)))$$

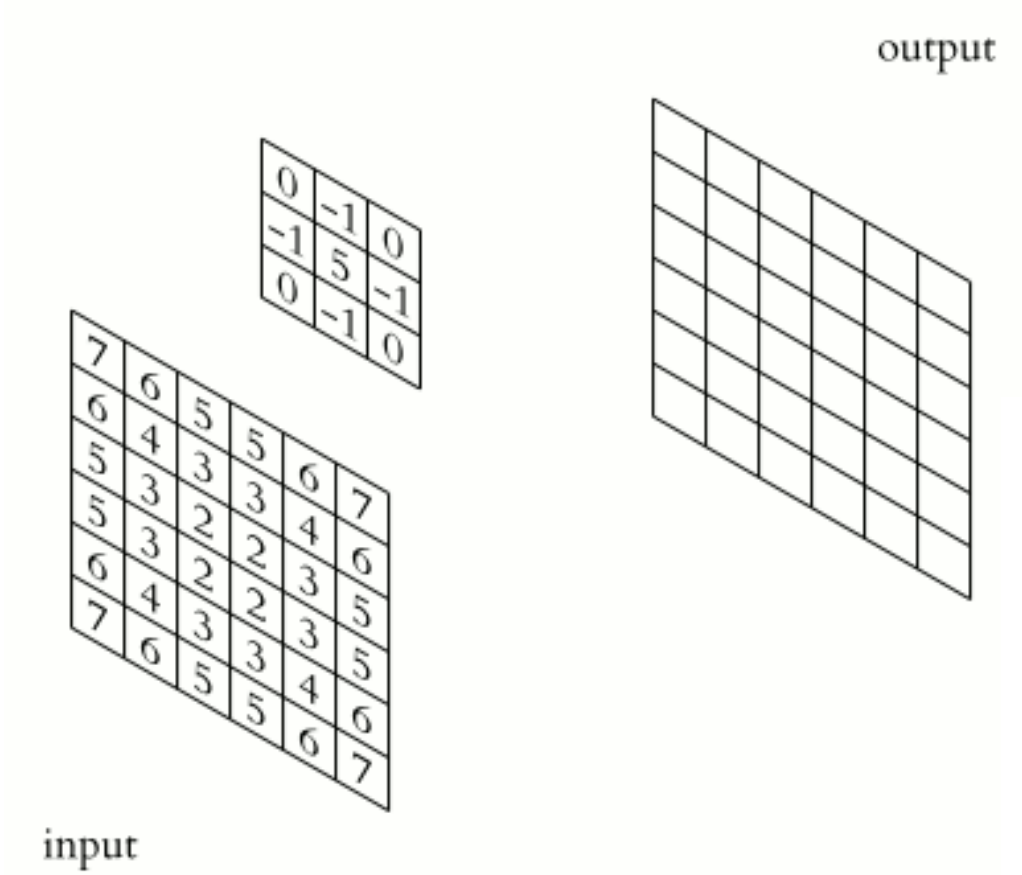
* BCE- бинарная кросс-энтропия, MSE- средний квадрат ошибки

Сверточная сеть

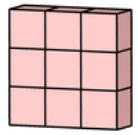


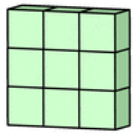


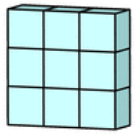




Striding. при работе со сверточным слоем, нужно получить выходные данные меньшего размера, чем входные.



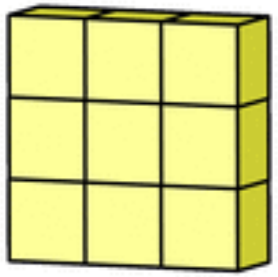




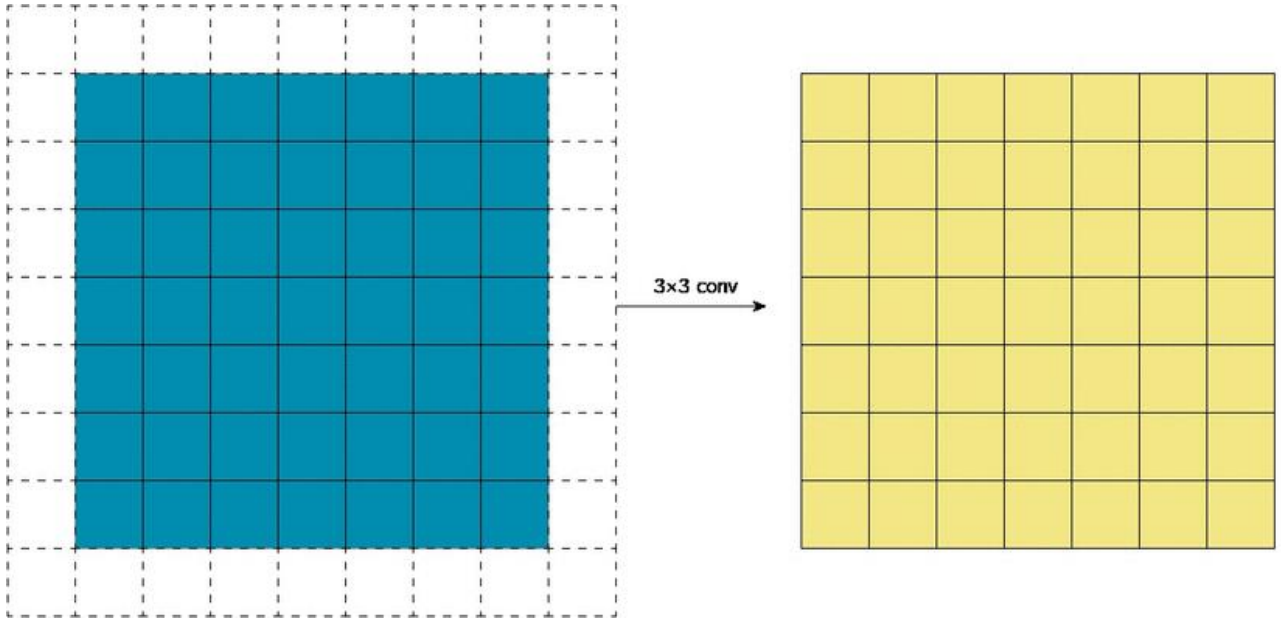
Padding.

каждая из обработанных в канале версий суммируется вместе для формирования одного канала. Ядра каждого фильтра генерируют одну версию каждого канала, а фильтр в целом создает один общий выходной канал

Сверточная сеть



каждый фильтр обрабатывает вход со своим отличающимся от других набором ядер и скалярным, создавая один выходной канал. Затем они объединяются вместе для получения общего выхода, причем количество выходных каналов равно числу фильтров. При этом обычно применяется нелинейность перед передачей входа другому слою свертки, который затем повторяет этот процесс.

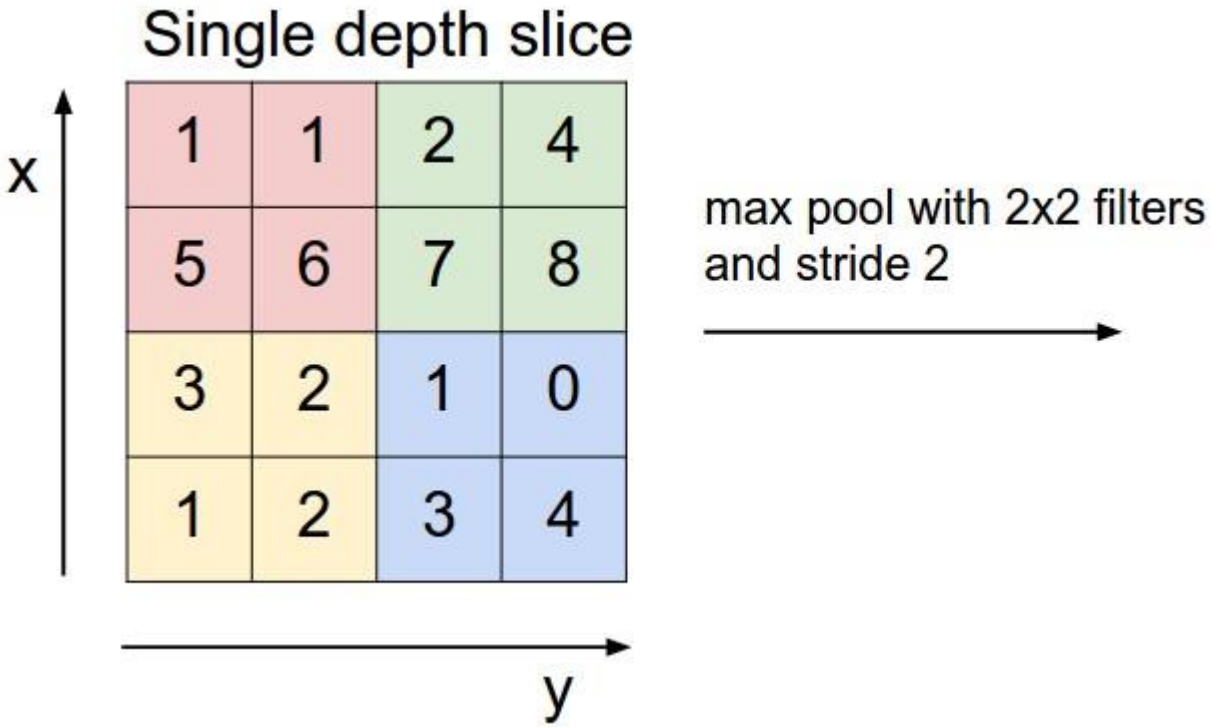


strided слой

каждый выходной слой имеет свое смещение. Смещение добавляется к выходному каналу для создания конечного выходного канала

Сверточная сеть

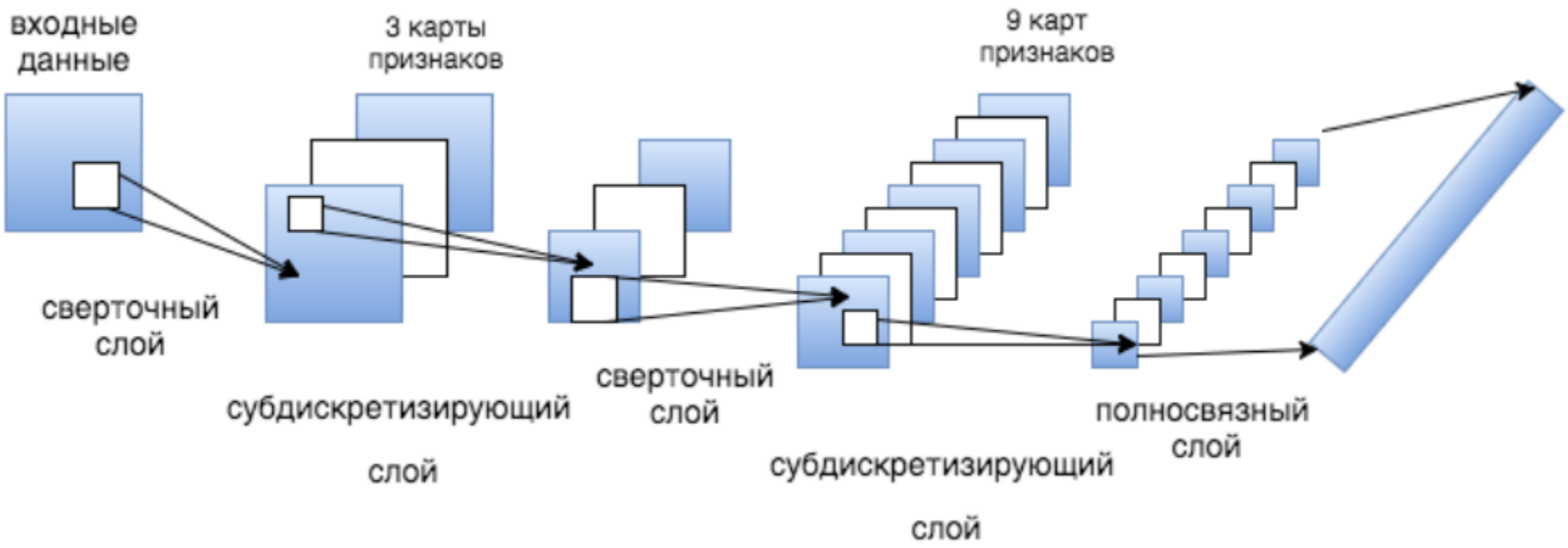
Пулинговый слой или слой субдискретизации



Основные цели пулингового слоя:

- уменьшение изображения, чтобы последующие свертки оперировали над большей областью исходного изображения – инвариант масштаба;
- увеличение инвариантности выхода сети по отношению к малому переносу входа;
- ускорение вычислений.

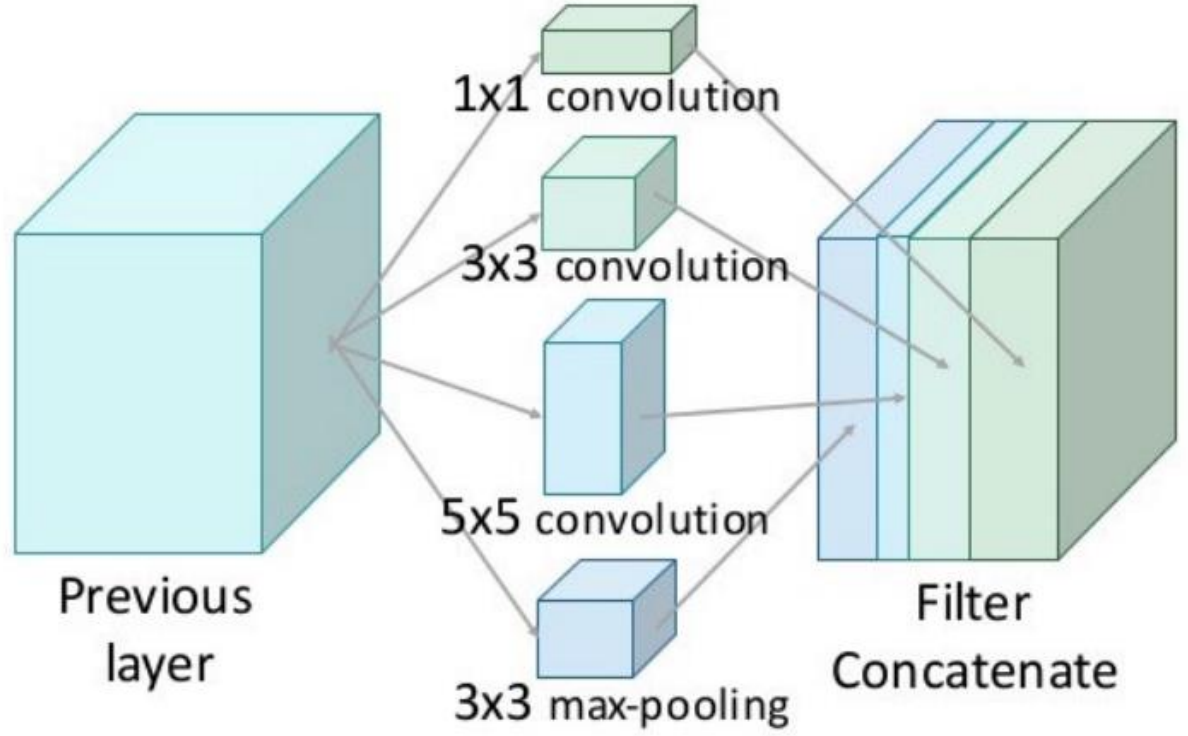
Сверточная сеть

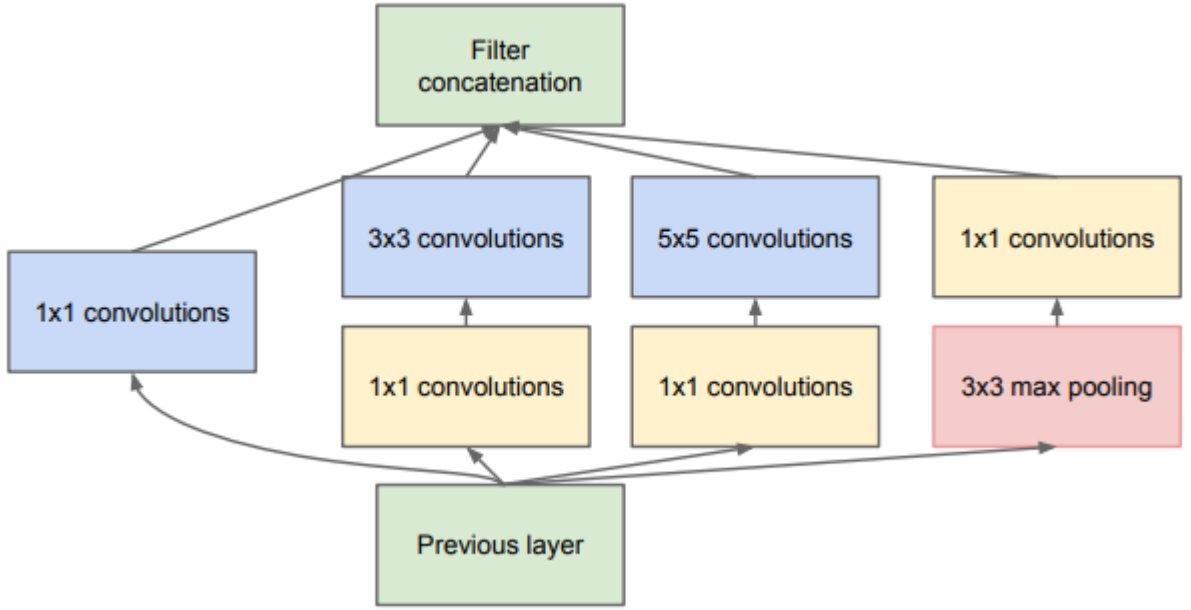


Сверточная сеть

Inception module

специальный слой нейронной сети, в котором каждый элемент предыдущего слоя соответствует определенной области исходного изображения и каждая свертка по таким элементам будет увеличивать область исходного изображения, пока элементы на последних слоях не будут соответствовать всему изображению целиком.

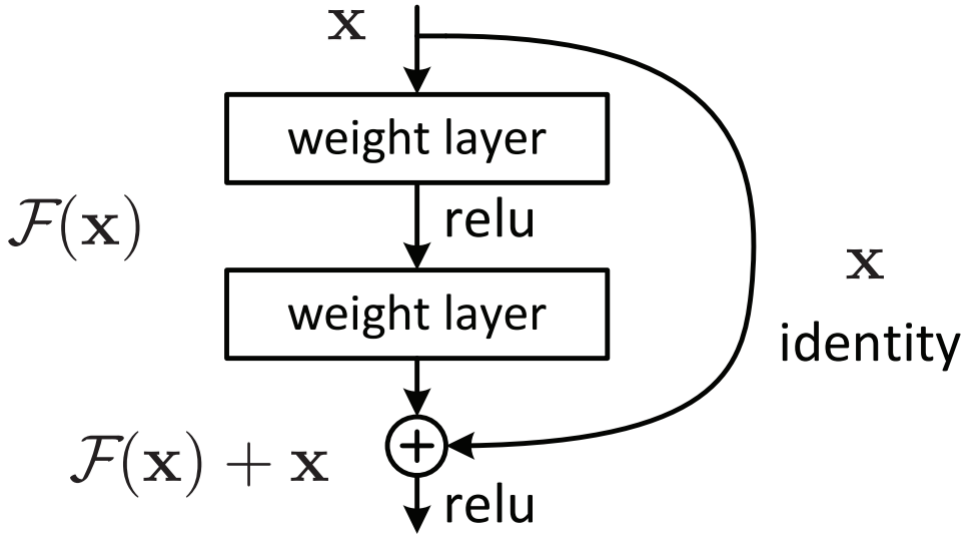




Сверточная сеть

Residual block (остаточный)

Решает проблему деградации градиента (*vanishing или exploding gradient*). Обучается предсказывать функцию активации ***F(x)*** Для компенсации этой разницы и вводится это замыкающее, которое добавляет недостающий ***x*** к функции.

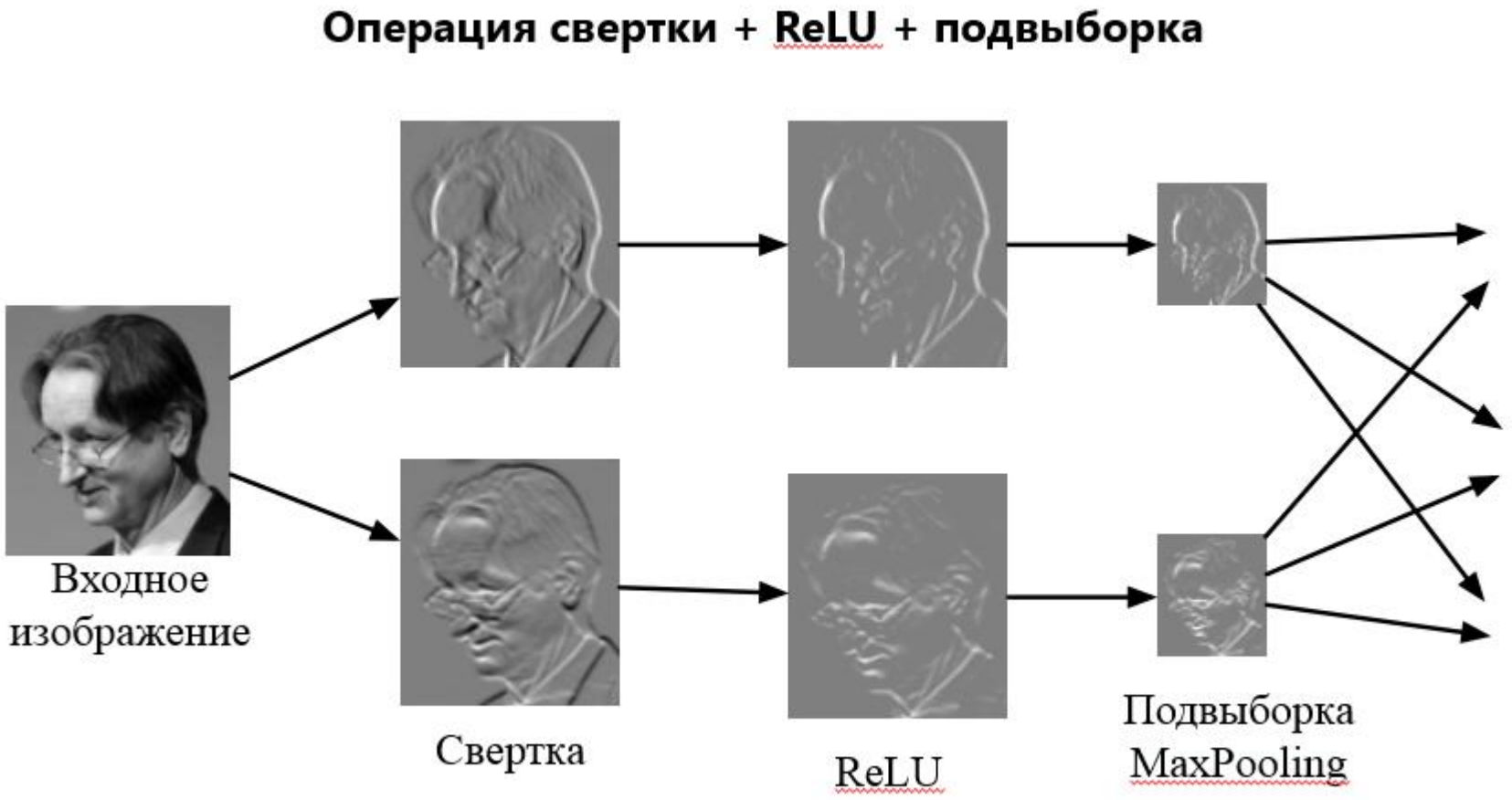


Соединения быстрого доступа (shortcut connections) пропускают один или несколько слоев и выполняют сопоставление идентификаторов. Их выходы добавляются к выходам stacked layers. Используя ResNet, можно решить множество проблем, таких как:

•ResNet относительно легко оптимизировать: «простые» сети (которые просто складывают слои) показывают большую ошибку обучения, когда глубина увеличивается.

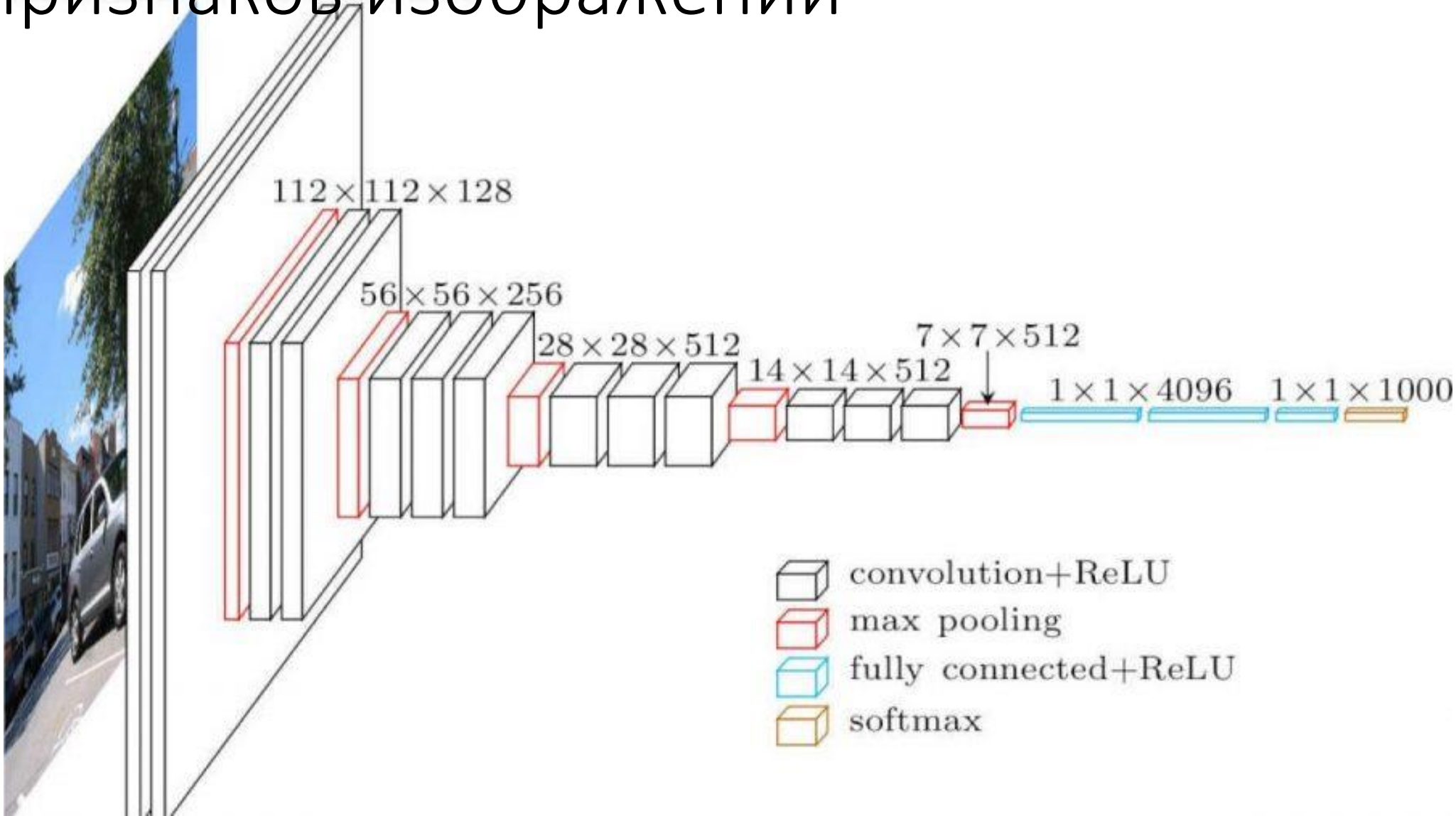
•ResNet позволяет относительно легко увеличить точность благодаря увеличению глубины, чего с другими сетями добиться сложнее.

Сверточная сеть

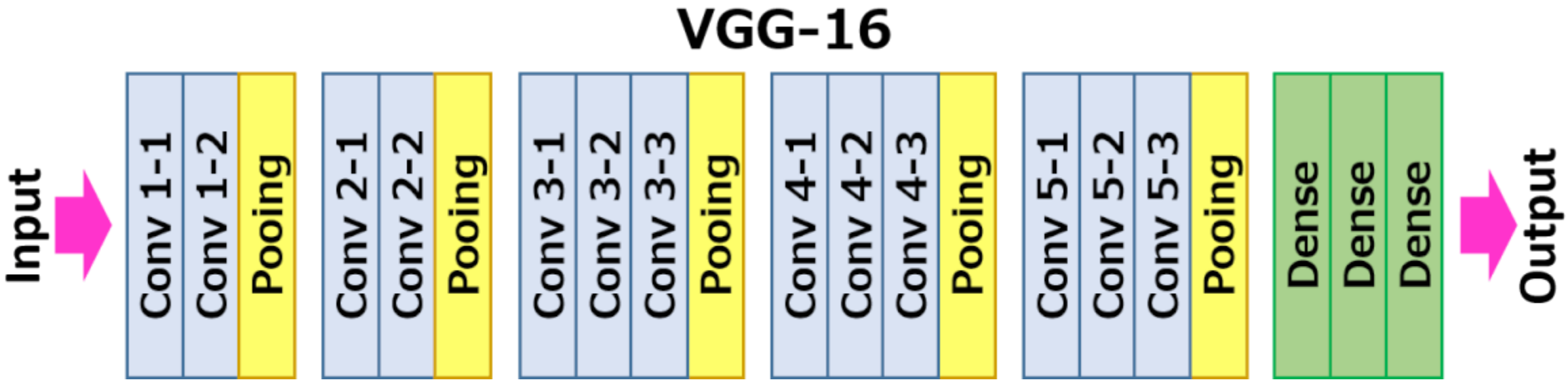


Основной причиной успеха СНС стало концепция общих весов. Несмотря на большой размер, эти сети имеют небольшое количество настраиваемых параметров по сравнению с их предком – неокогнитроном. Имеются варианты СНС (Tiled Convolutional Neural Network), похожие на неокогнитрон, в таких сетях происходит, частичный отказ от связанных весов, но алгоритм обучения остается тем же и основывается на обратном распространении ошибки. СНС могут быстро работать на последовательной машине и быстро обучаться за счет чистого распараллеливания процесса свертки по каждой карте, а также обратной свертки при распространении ошибки по сети.

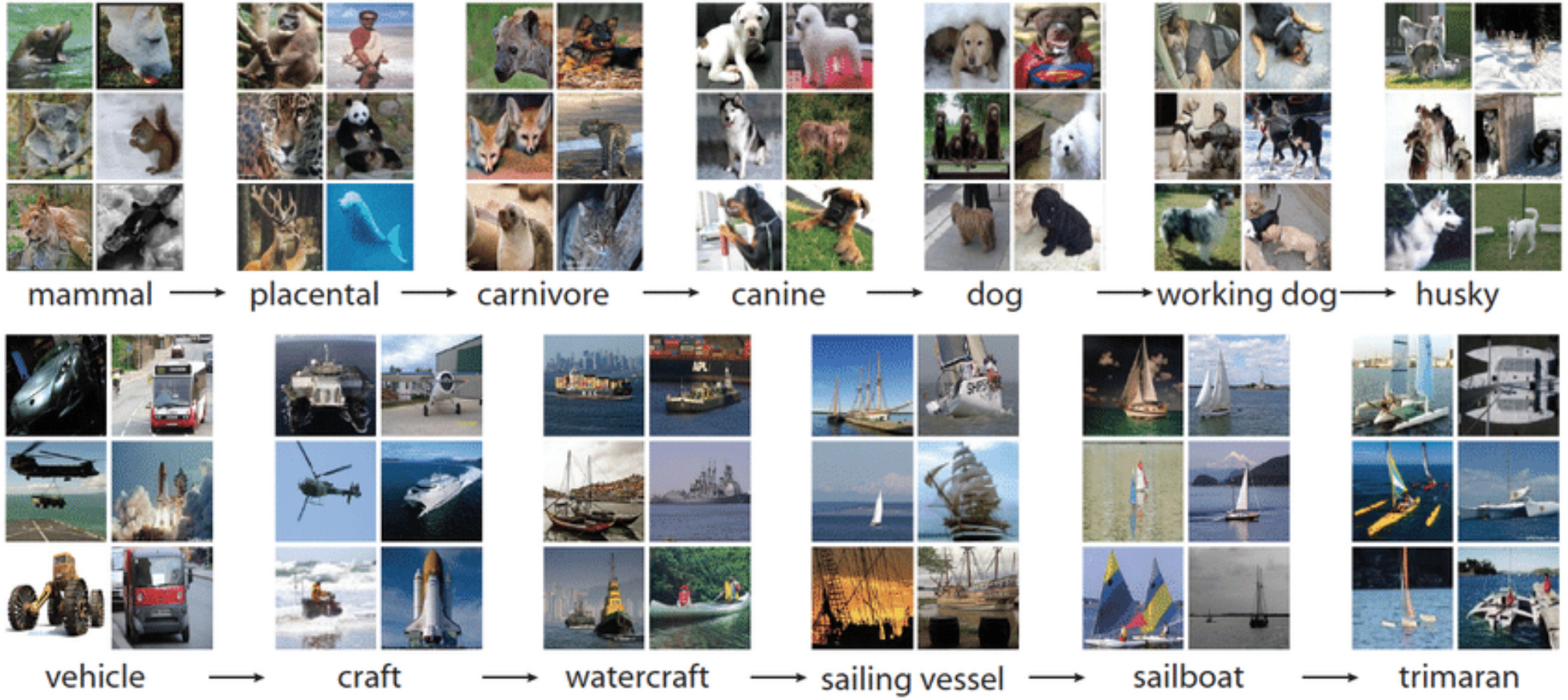
VGG16 — сверточная сеть для выделения признаков изображений



VGG16 — сверточная сеть для выделения признаков изображений

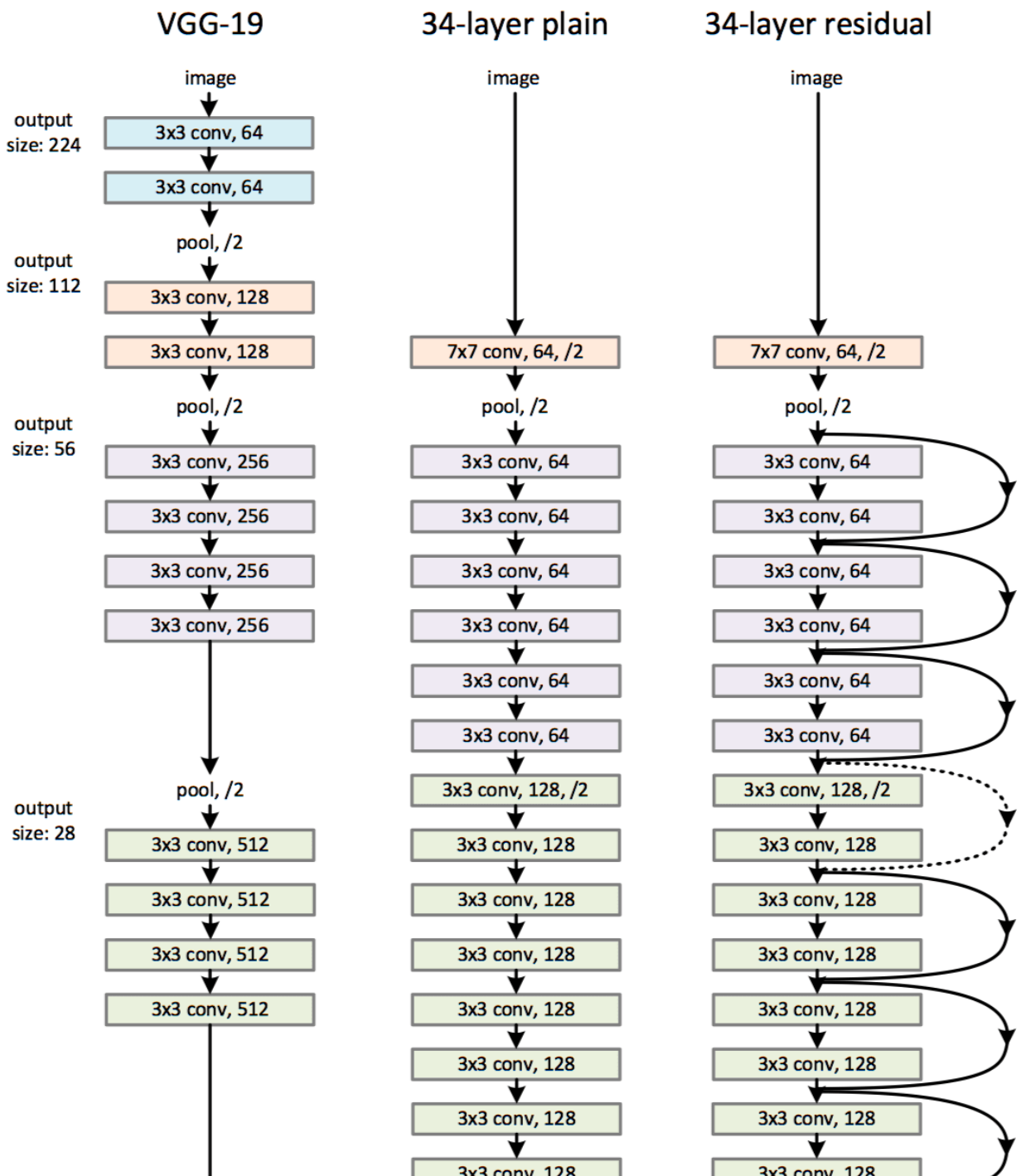


VGG16 (2014 год) — одна из самых знаменитых моделей, отправленных на соревнование ILSVRC-2014. Она является улучшенной версией AlexNet, в которой заменены большие фильтры (размера 11 и 5 в первом и втором сверточном слое, соответственно) на несколько фильтров размера 3x3, следующих один за другим. Сеть VGG16 обучалась на протяжении нескольких недель при использовании видеокарт NVIDIA TITAN BLACK.



- ImageNet использует краудсорсинг для аннотирования изображений.

- Аннотации на уровне самих изображений показывают наличие или отсутствие объекта данного класса (например, «на картинке имеется тигр» или «на картинке нет тигров»). На уровне объекта в аннотацию включается прямоугольник с координатами видимой части объекта. ImageNet использует вариант семантической сети WordNet для категоризации объектов, которая достаточно детализирована, например, породы собак представлены 120 классами. Каждому узлу сети WordNet сопоставлены сотни или тысячи изображений, но в среднем на 2016 год — около 500 изображений.



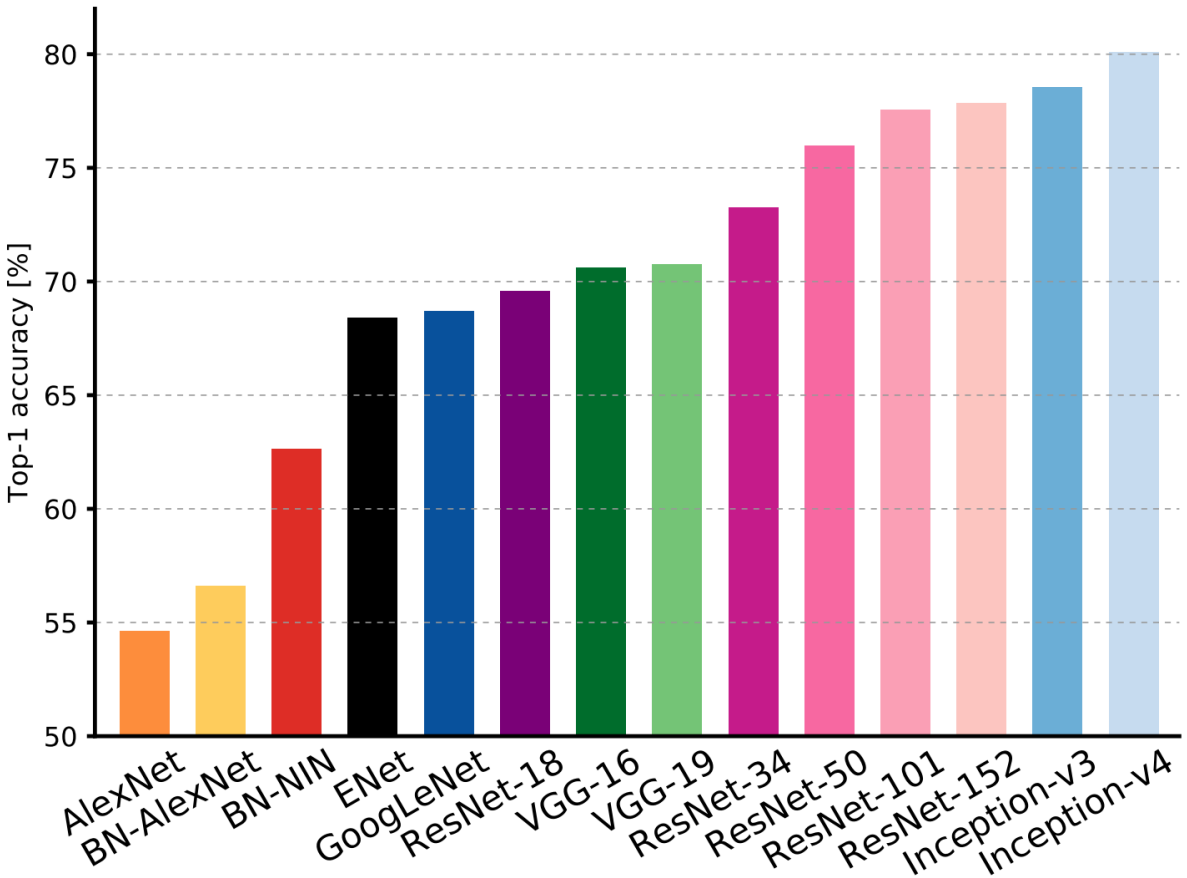
Слева: модель VGG-19 (19,6 млрд. FLOP) в качестве эталона. Посередине: простая сеть с 34 слоями (3,6 млрд. FLOP). Справа: ResNet с 34 слоями (3,6 миллиарда FLOP). Пунктирные быстрые соединения увеличивают размерность.

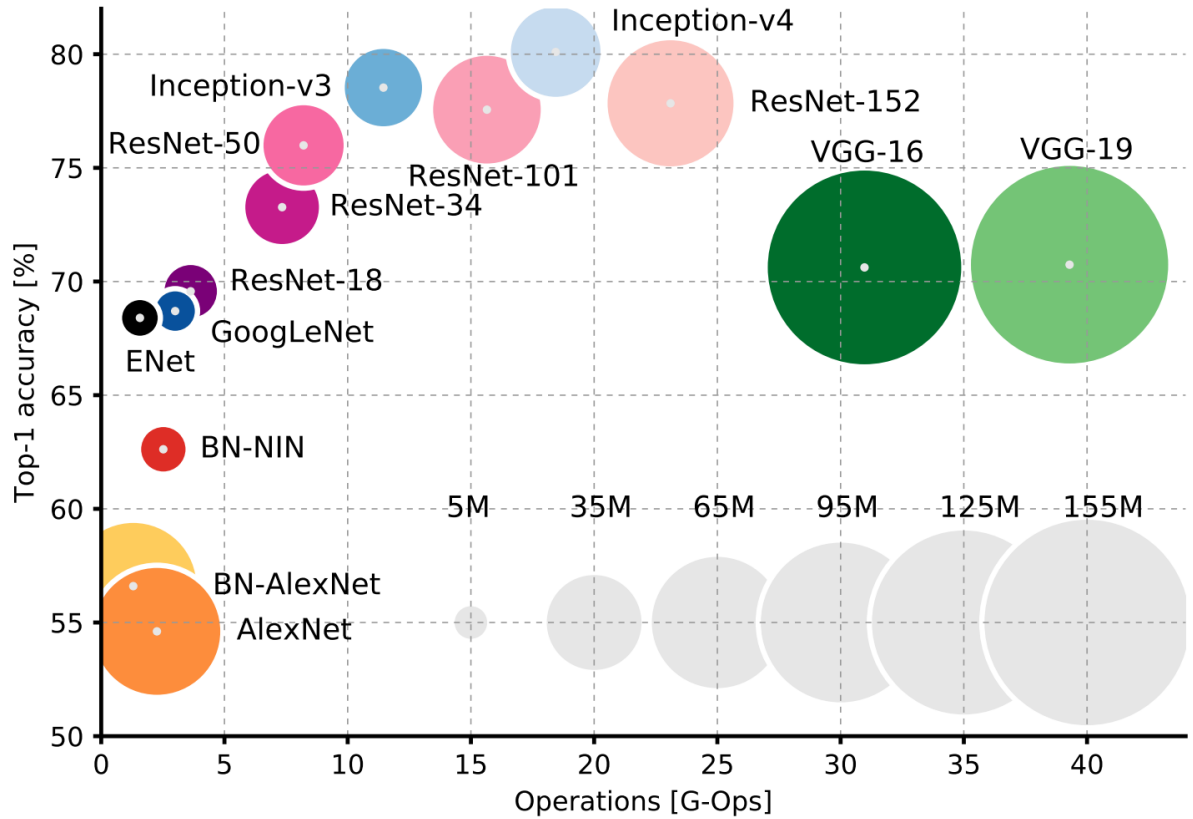
Простые базовые линии как VGG. Сверточные слои в основном имеют фильтры 3×3 и следуют двум простым правилам:

1. Для одной и той же выходной карты объектов слои имеют одинаковое количество фильтров;
2. Если размер карты объектов уменьшается вдвое, число фильтров удваивается, чтобы сохранить временную сложность каждого слоя.

ResNet: когда размерности увеличиваются (пунктирные лини) рассматривает два варианта:

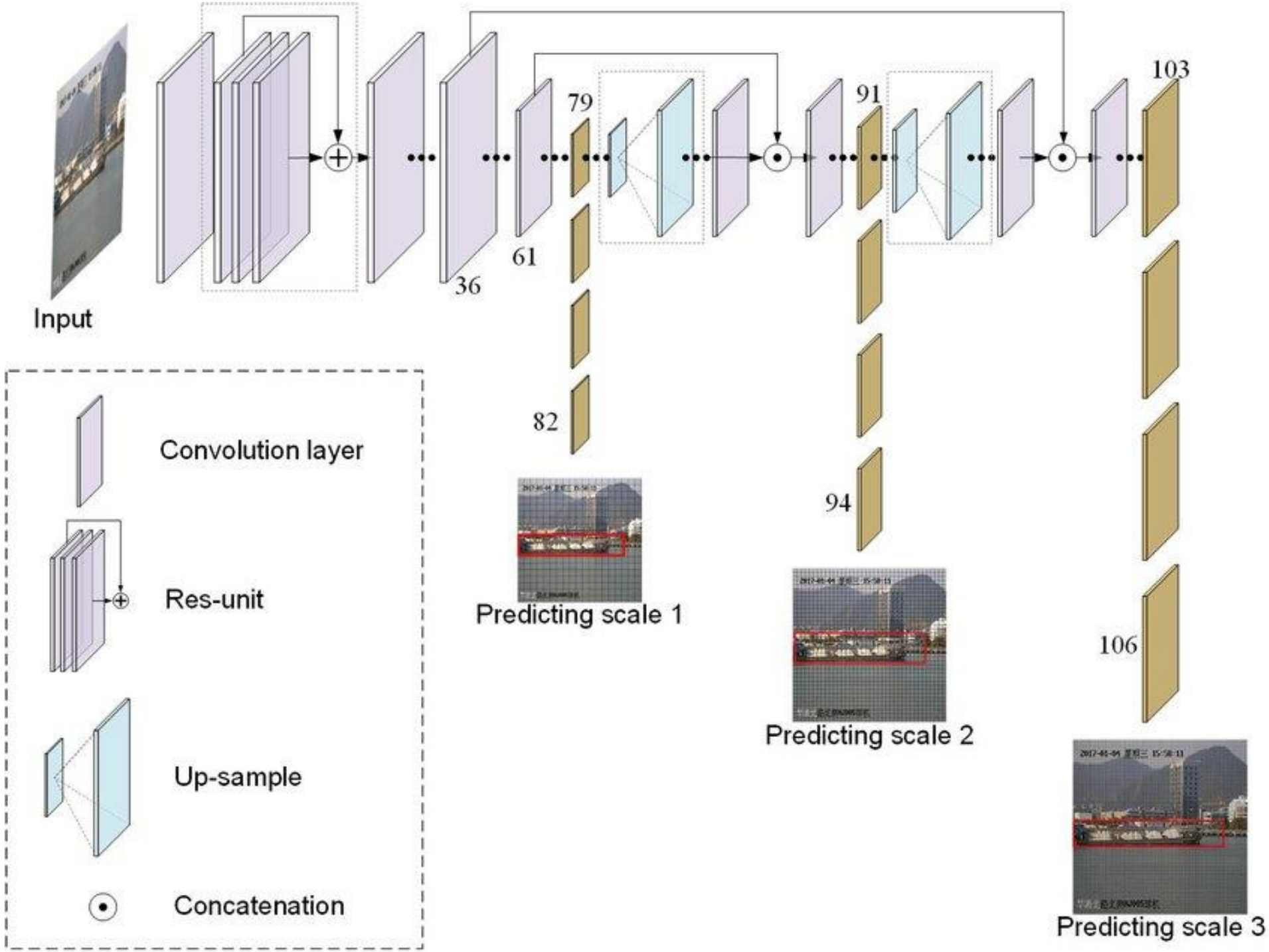
1. Быстрое соединение выполняет сопоставление идентификаторов с дополнительными нулями, добавленными для увеличения размерности. Эта опция не вводит никаких дополнительных параметров.
2. Проекция быстрого соединения в F (x {W} + x) используется для сопоставления размерностей (выполнен с помощью 1×1 сверток).


Top-1 accuracy [%]
80
75
70
65
60
55
50
AlexNet
BN-AlexNet
BN-NIN
ENet
GoogLeNet
ResNet-18
VGG-16
VGG-19
ResNet-34
ResNet-50
ResNet-101
ResNet-152
Inception-v3
Inception-v4


Top-1 accuracy [%]
80
75
70
65
60
55
50
0
5
10
15
20
25
30
35
40
Operations [G-Ops]
Inception-v4
Inception-v3
ResNet-152
ResNet-50
ResNet-101
VGG-16
VGG-19
ResNet-34
ResNet-18
GoogLeNet
ENet
BN-NIN
5M
35M
65M
95M
125M
155M
BN-AlexNet
AlexNet

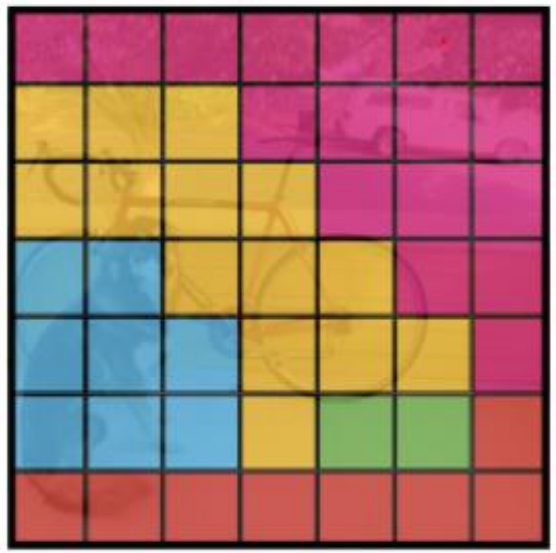
YOLOv3

YOLOv3 — это усовершенствованная версия архитектуры YOLO. Она состоит из 106-ти свёрточных слоев и лучше детектирует небольшие объекты по сравнению с её предшествиницей YOLOv2. Основная особенность YOLOv3 состоит в том, что на выходе есть три слоя каждый из которых расчитан на обнаружения объектов разного размера.

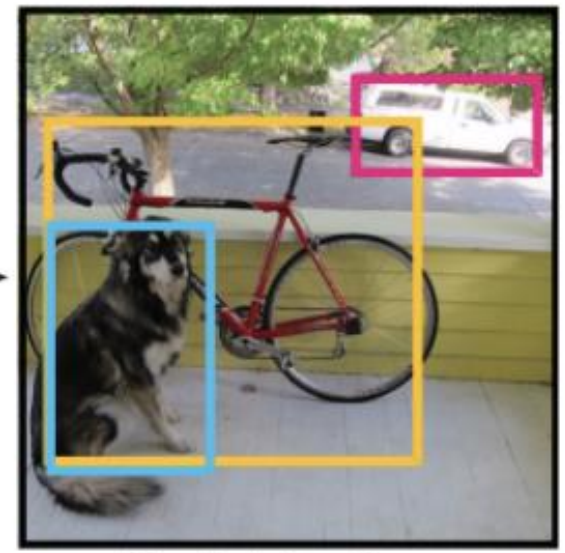


S × S grid on input

Bounding boxes + confidence



Class probability map



Final detections

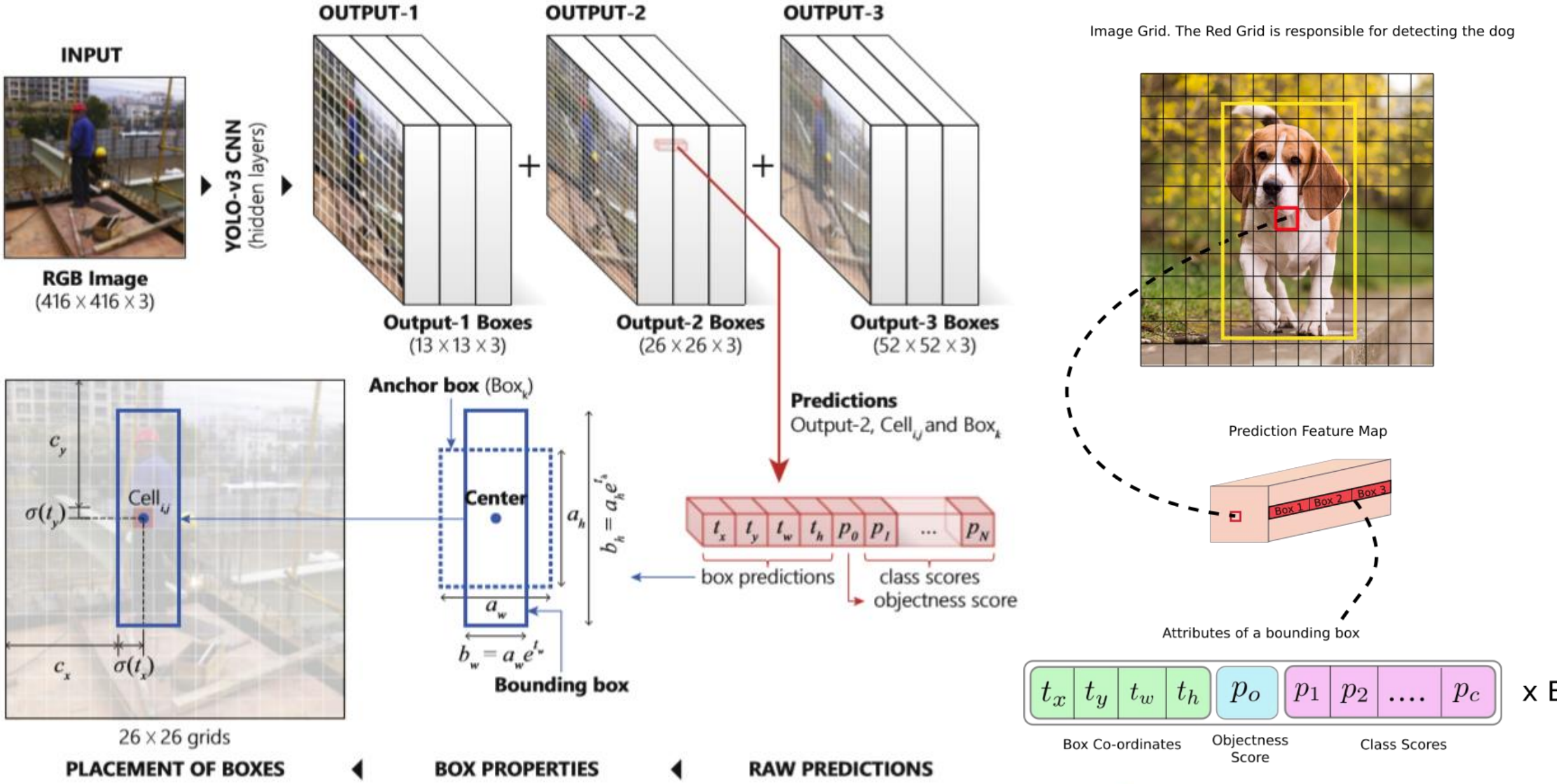
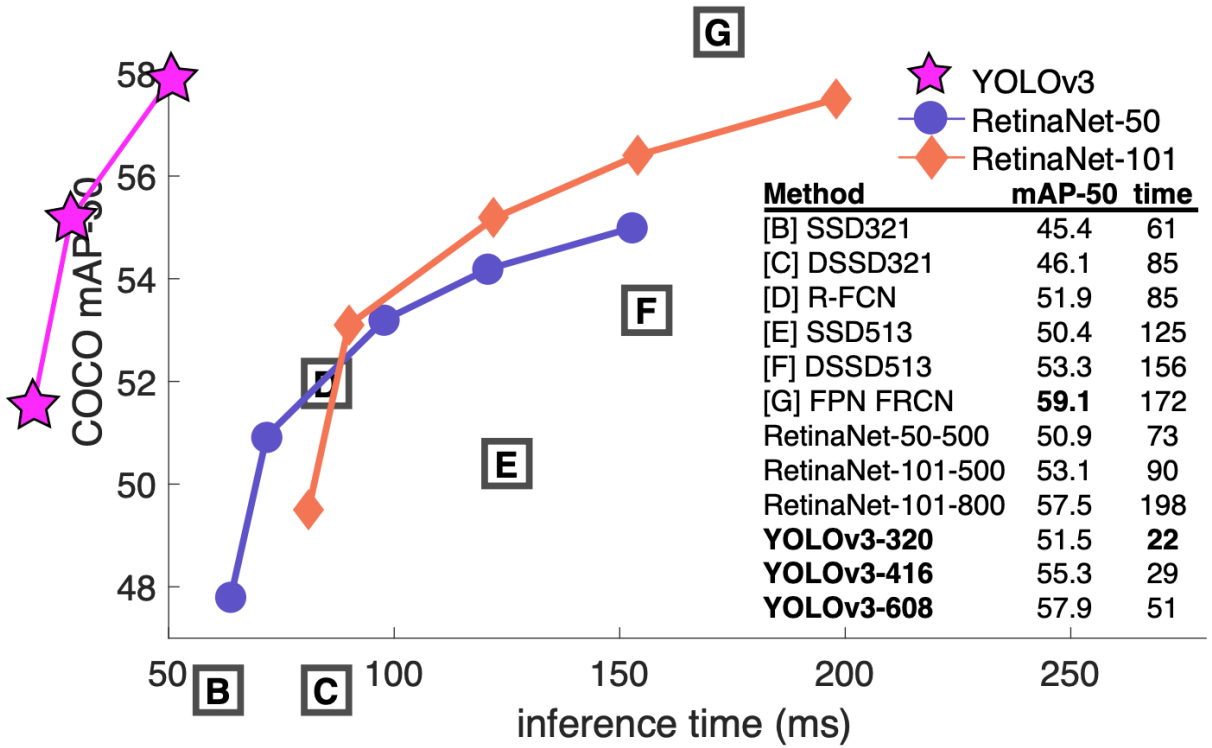
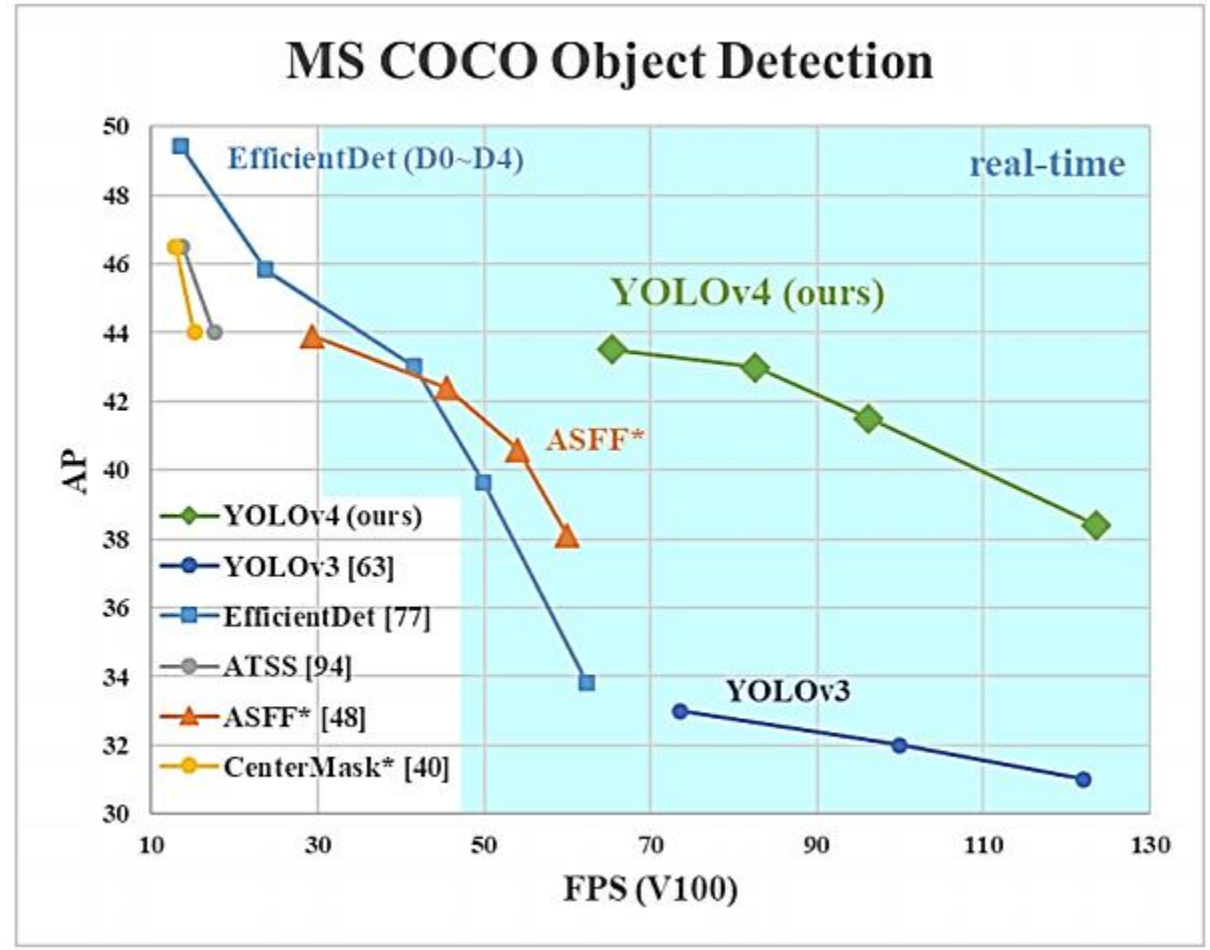


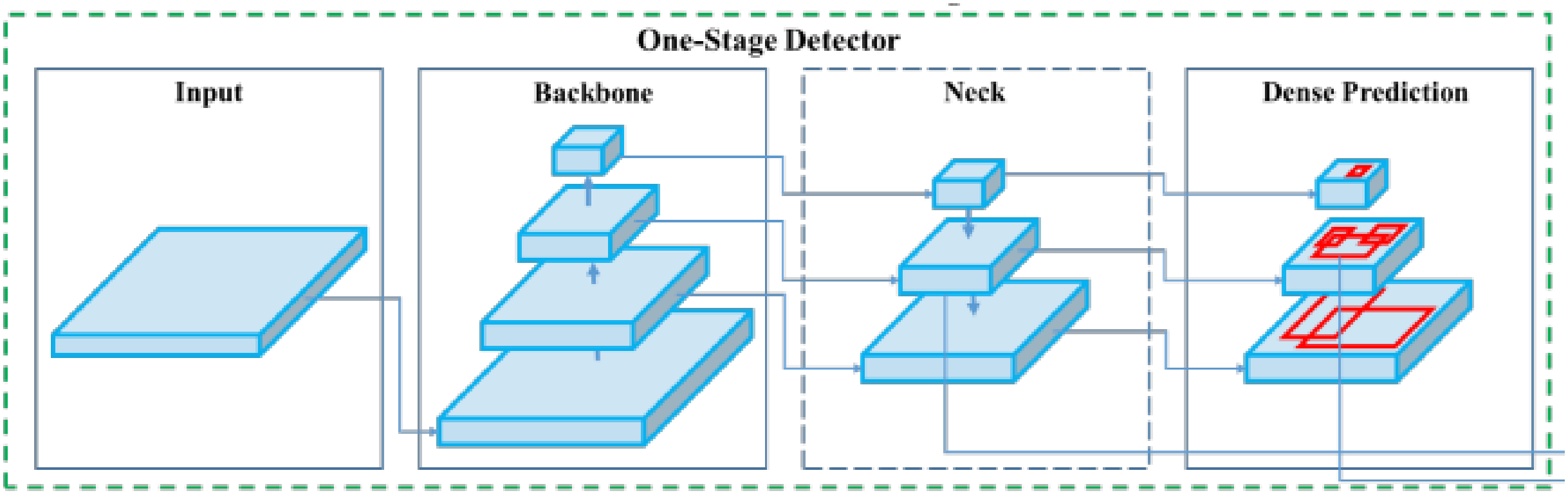
Fig. 2. Schematic diagram of the YOLO-v3 algorithm.

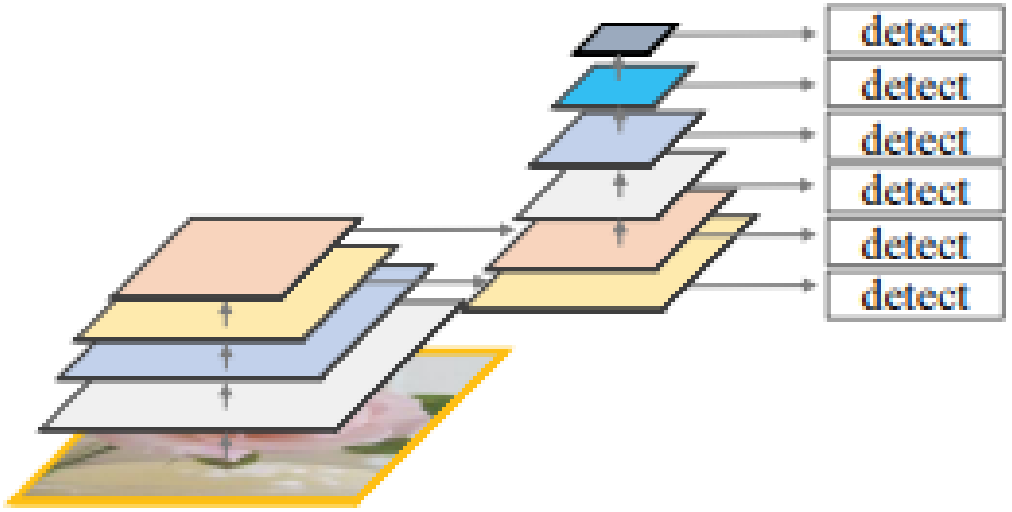


Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

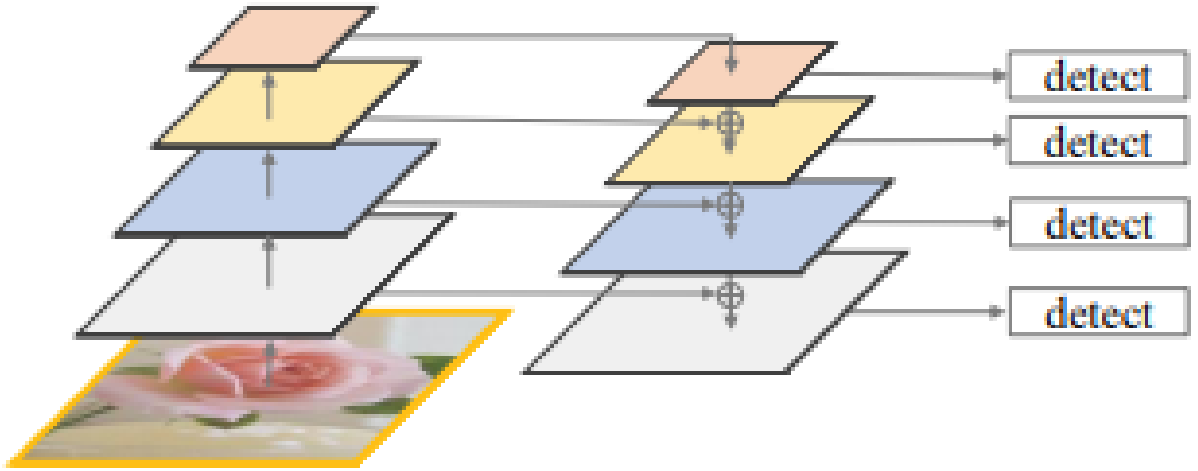


YOLO v4

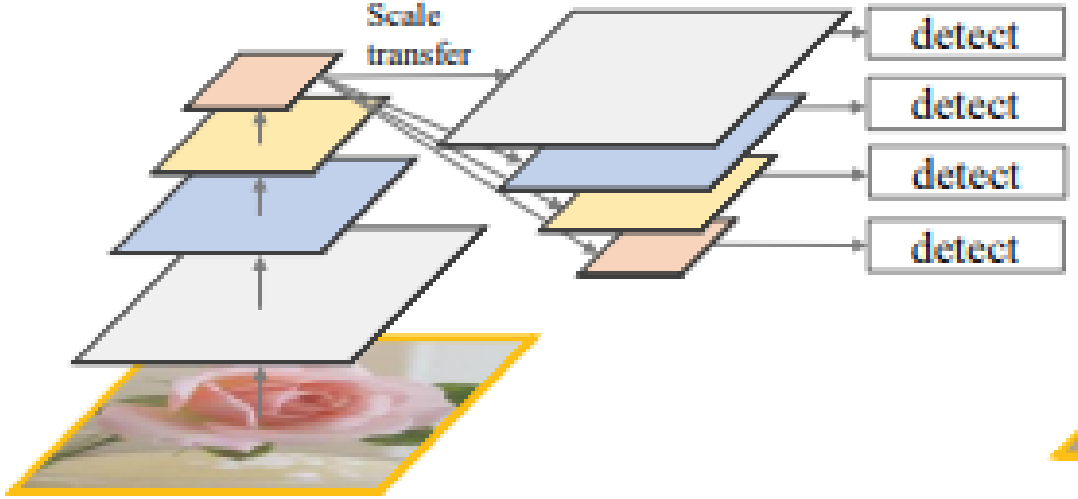




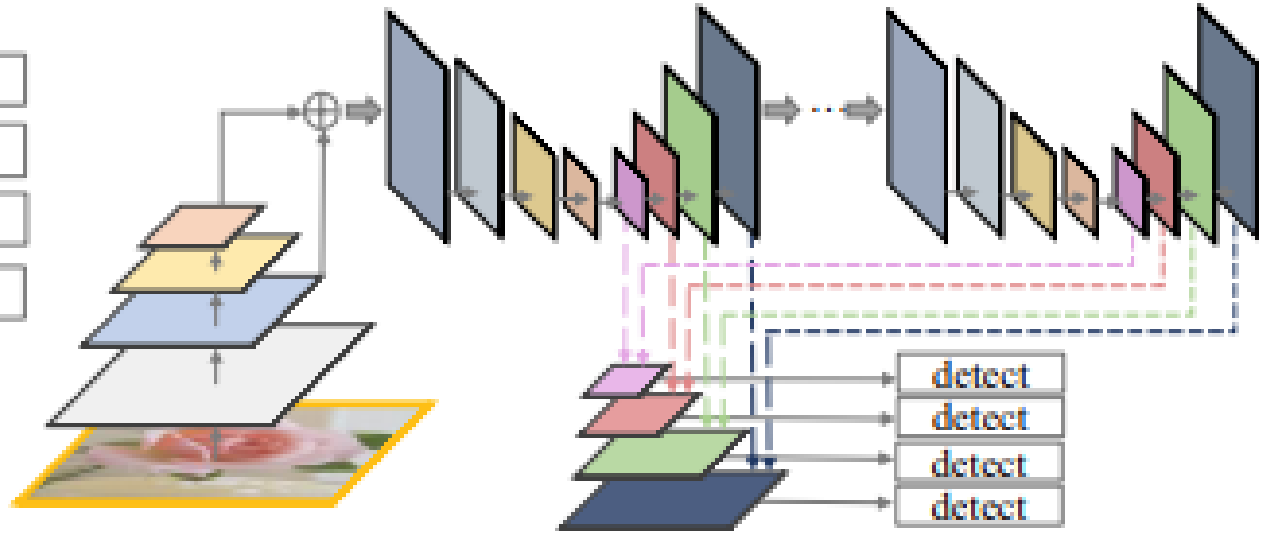
(a) SSD-style feature pyramid



(b) FPN-style feature pyramid

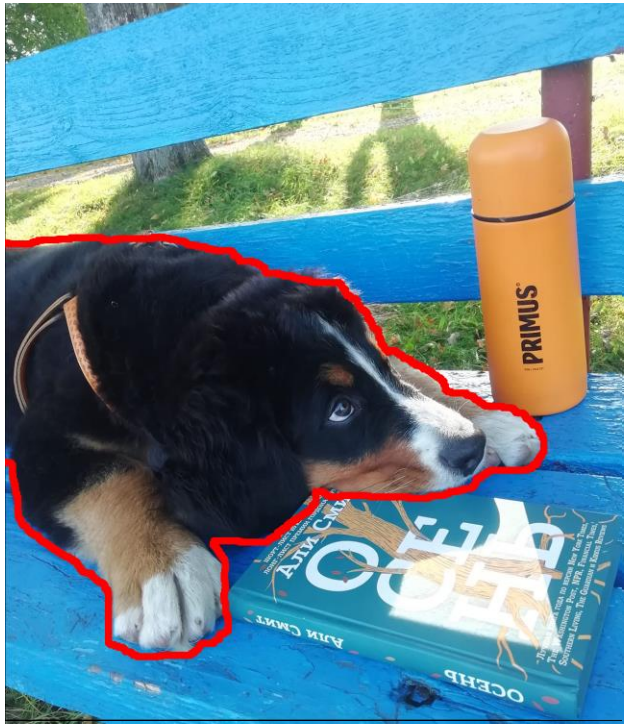


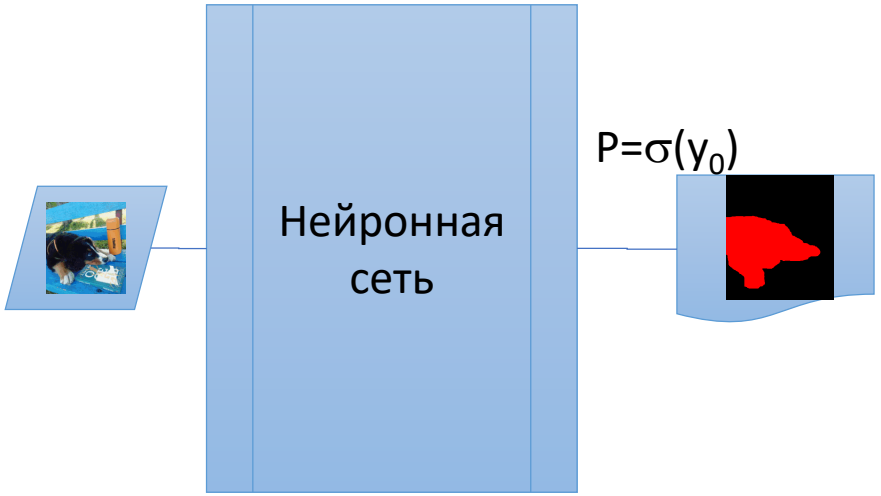
(c) STDN-style feature pyramid



(d) Our multi-level feature pyramid

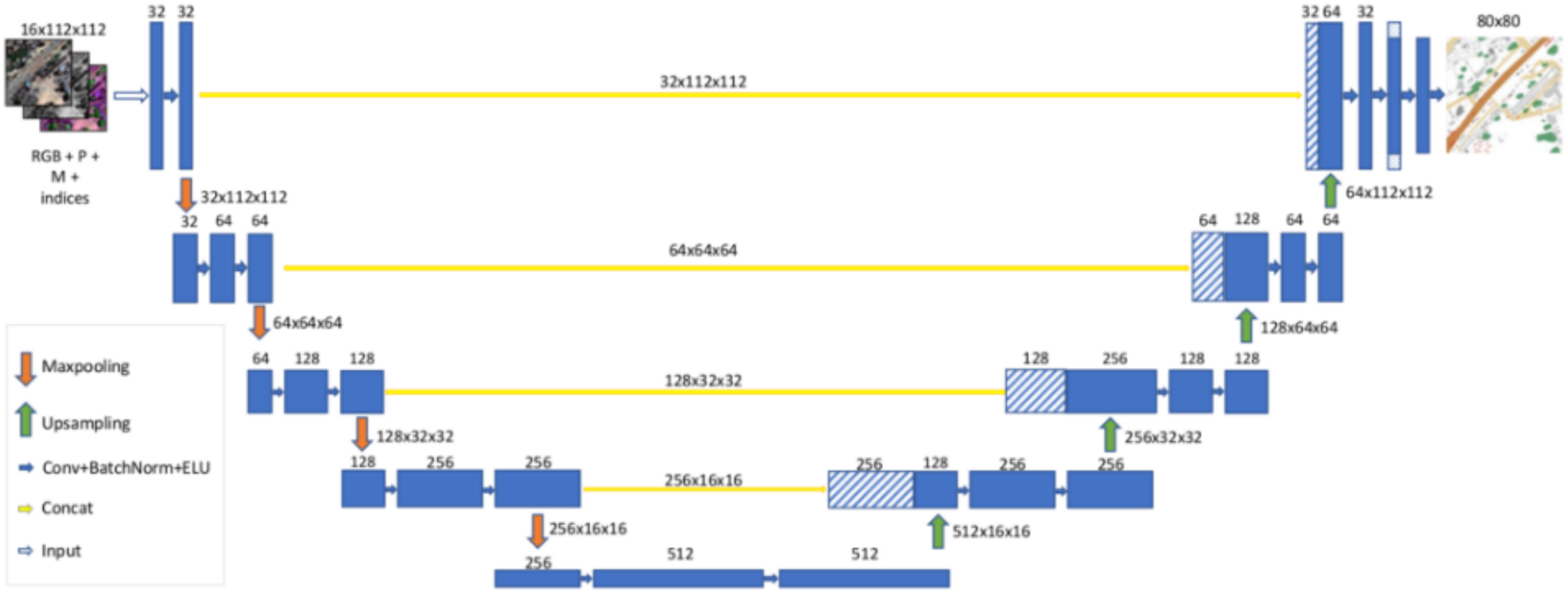
Сегментация изображений



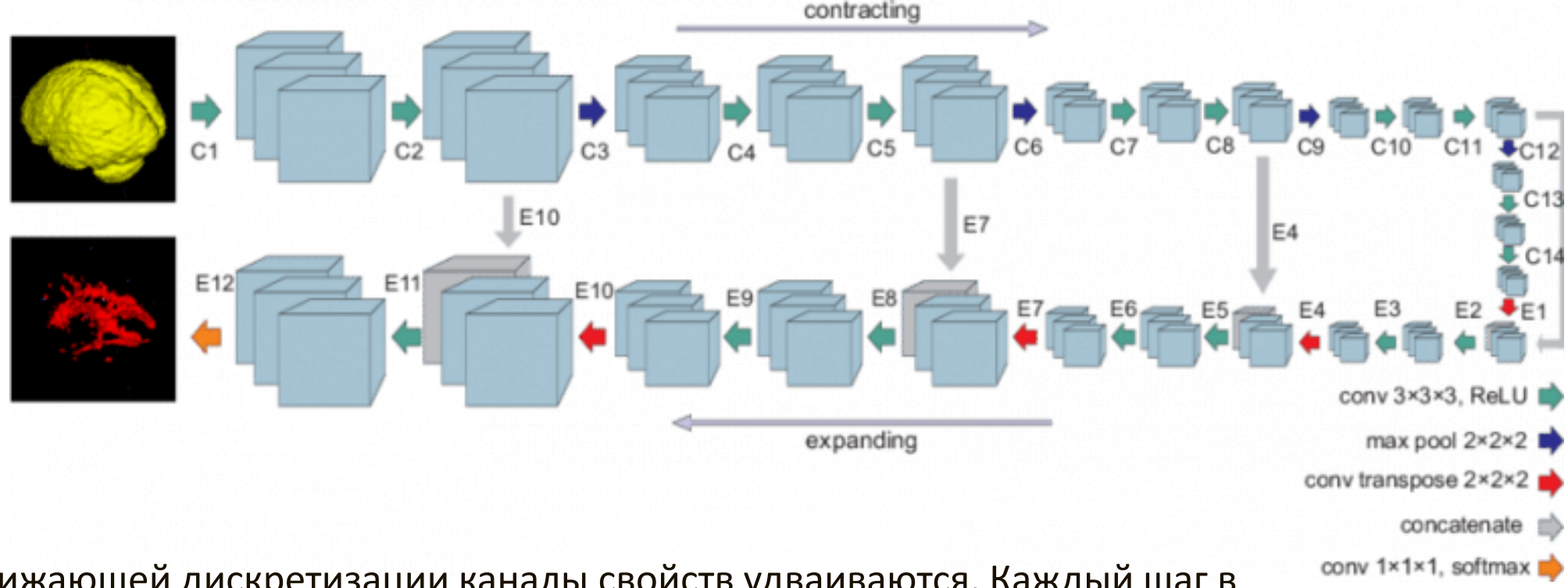


$$F = \sum_j BCE(\sigma(yj), t_j)$$

U-Net


16x112x112
RGB + P + M + indices
32 32
32x112x112
32 64 64
64x64x64
64 128 128
128x32x32
128 256 256
256x16x16
256 512 512
512x16x16
256x32x32
128 256 128 128
128x64x64
64 128 64 64
64x112x112
32 64 32
32x112x112
80x80
Maxpooling
Upsampling
Conv+BatchNorm+ELU
Concat
Input

U-Net



На каждом этапе понижающей дискретизации каналы свойств удваиваются. Каждый шаг в расширяющемся пути состоит из операции повышающей дискретизации карты свойств, за которой следуют:

•свертка 2×2, которая уменьшает количество каналов свойств;

•объединение с соответствующим образом обрезанной картой свойств из стягивающегося пути;

•две 3×3 свертки, за которыми следует ReLU.

Обрезка необходима из-за потери граничных пикселей при каждой свертке.

На последнем слое используется свертка 1×1 для сопоставления каждого 64-компонентного вектора свойств с желаемым количеством классов. Всего сеть содержит 23 сверточных слоя.

GitHub-сегментация

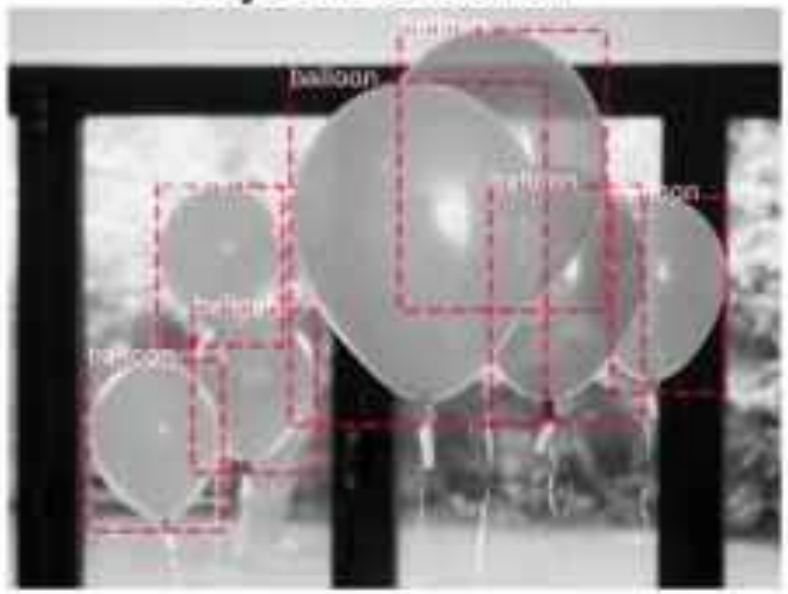
- divamgupta/**image-segmentation-keras**
- LeeJunHyun/**Image_Segmentation**
- AKSHAYUBHAT/**ImageSegmentation**
-

model_name	Base Model	Segmentation Model
fcn_8	Vanilla CNN	FCN8
fcn_32	Vanilla CNN	FCN8
fcn_8_vgg	VGG 16	FCN8
fcn_32_vgg	VGG 16	FCN32
fcn_8_resnet50	Resnet-50	FCN32
fcn_32_resnet50	Resnet-50	FCN32
fcn_8_mobilenet	MobileNet	FCN32
fcn_32_mobilenet	MobileNet	FCN32
pspnet	Vanilla CNN	PSPNet
vgg_pspnet	VGG 16	PSPNet
resnet50_pspnet	Resnet-50	PSPNet
unet_mini	Vanilla Mini CNN	U-Net
unet	Vanilla CNN	U-Net
vgg_unet	VGG 16	U-Net
resnet50_unet	Resnet-50	U-Net
mobilenet_unet	MobileNet	U-Net
segnet	Vanilla CNN	Segnet
vgg_segnet	VGG 16	Segnet
resnet50_segnet	Resnet-50	Segnet
mobilenet_segnet	MobileNet	Segnet

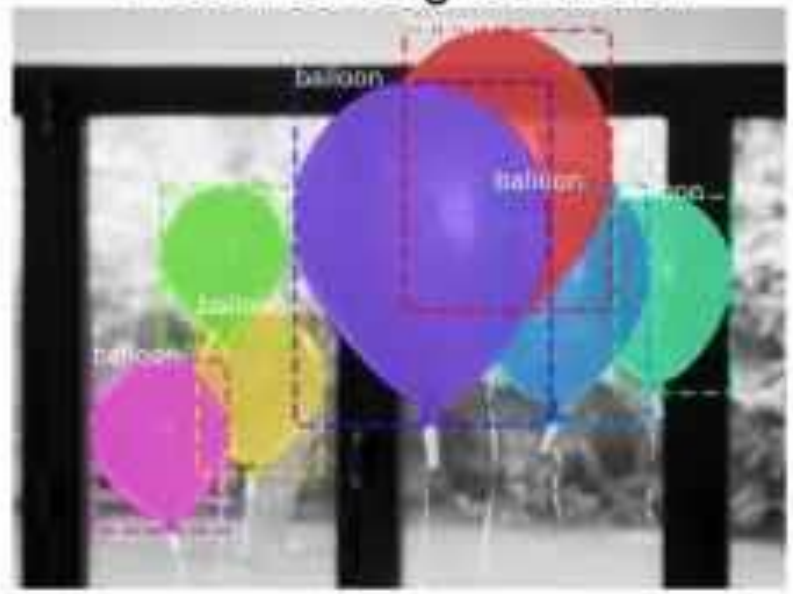
Classification

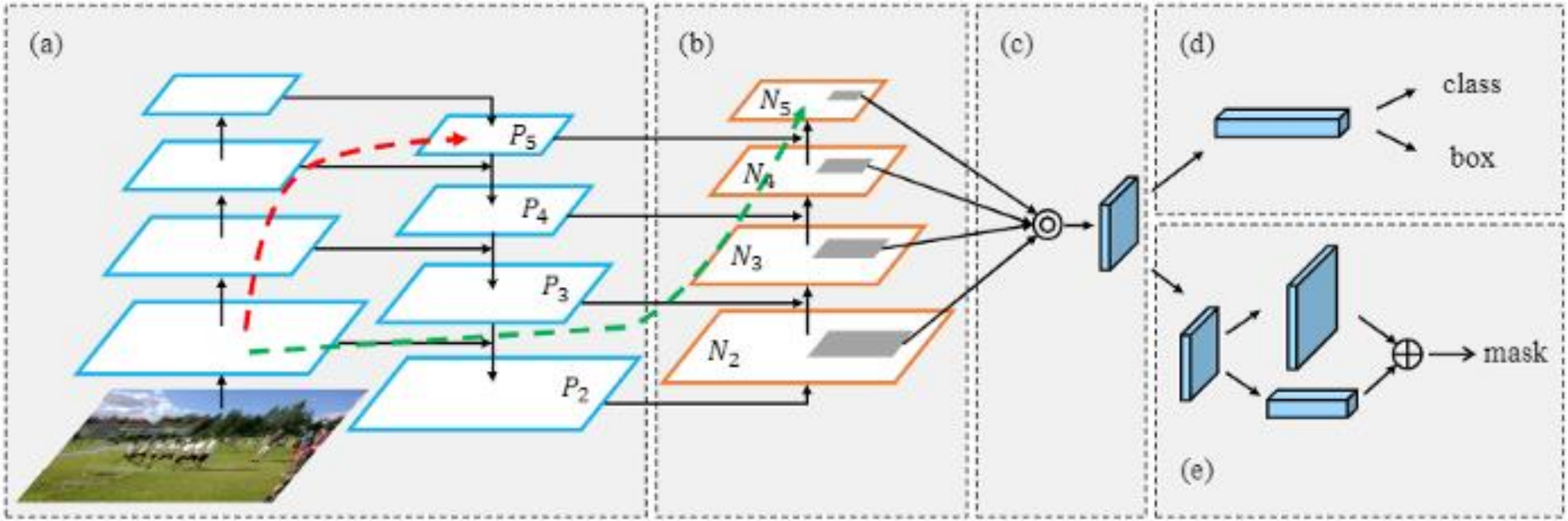
Semantic Segmentation

Object Detection

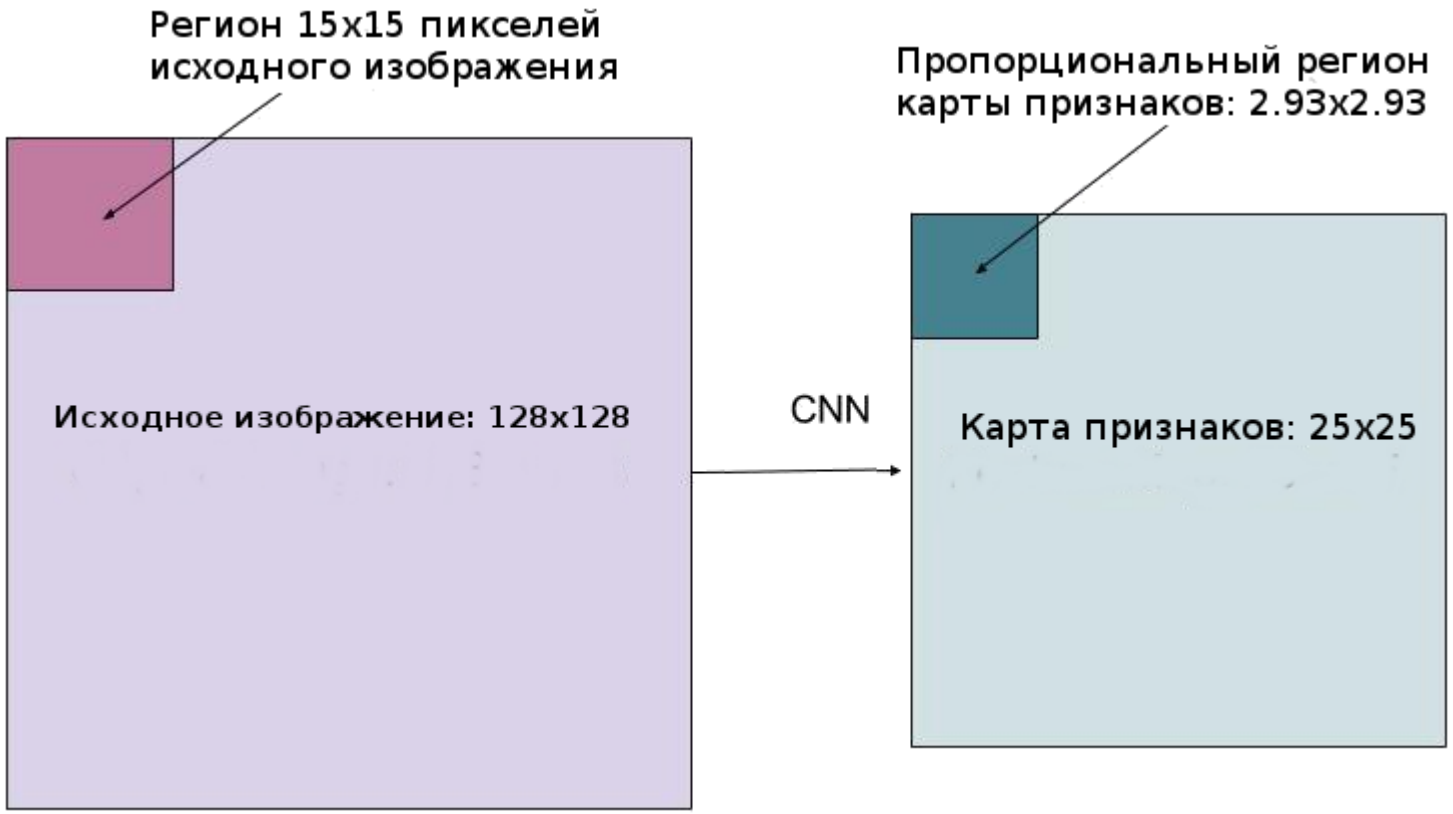


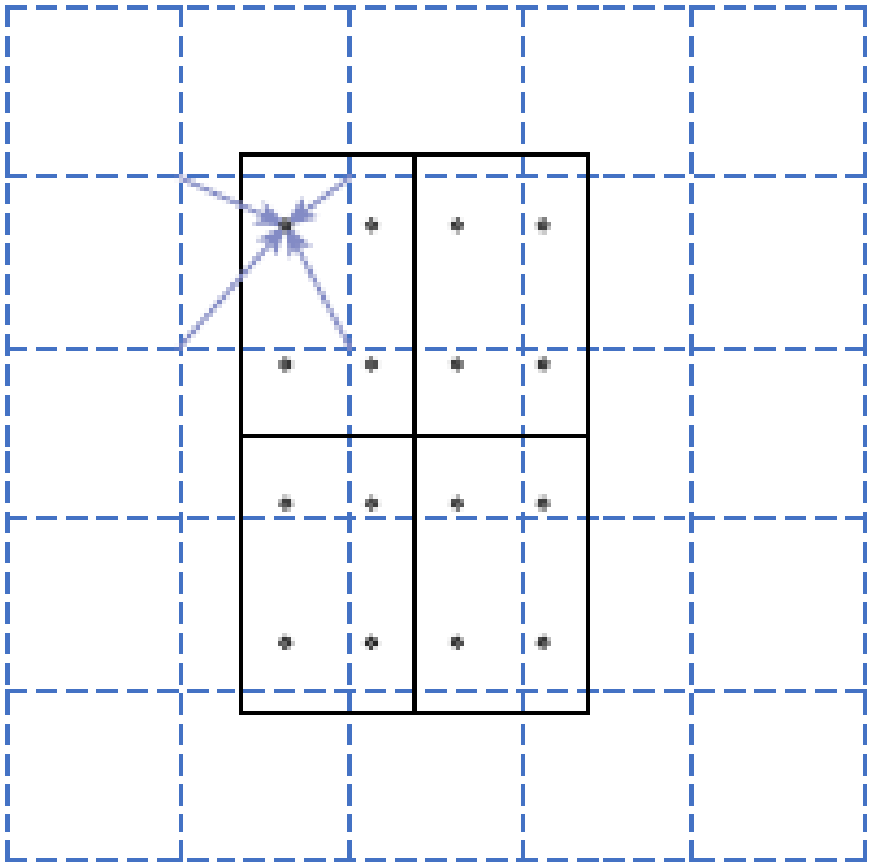
Instance Segmentation

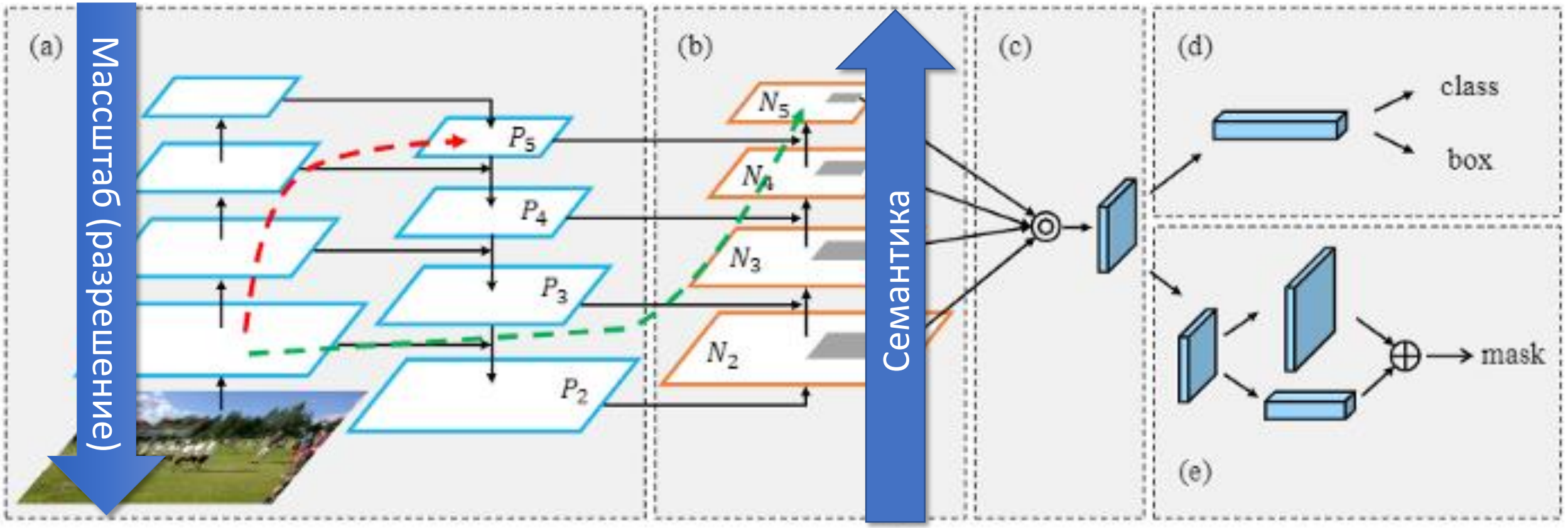



(a)
(b)
(c)
(d)
(e)
P_5
P_4
P_3
P_2
N_5
N_4
N_3
N_2
class
box
mask

Одна из основных модификаций, возникших из-за необходимости предсказывать маску — изменение процедуры **RoIPool**(вычисляющей матрицу признаков для региона-кандидата) на так называемую **RoIAlign**. Дело в том, что карта признаков, полученная из CNN, имеет меньший размер, чем исходное изображение, и регион, охватывающий на изображении целочисленное количество пикселей, не получается отобразить в пропорциональный регион карты с целочисленным количеством признаков

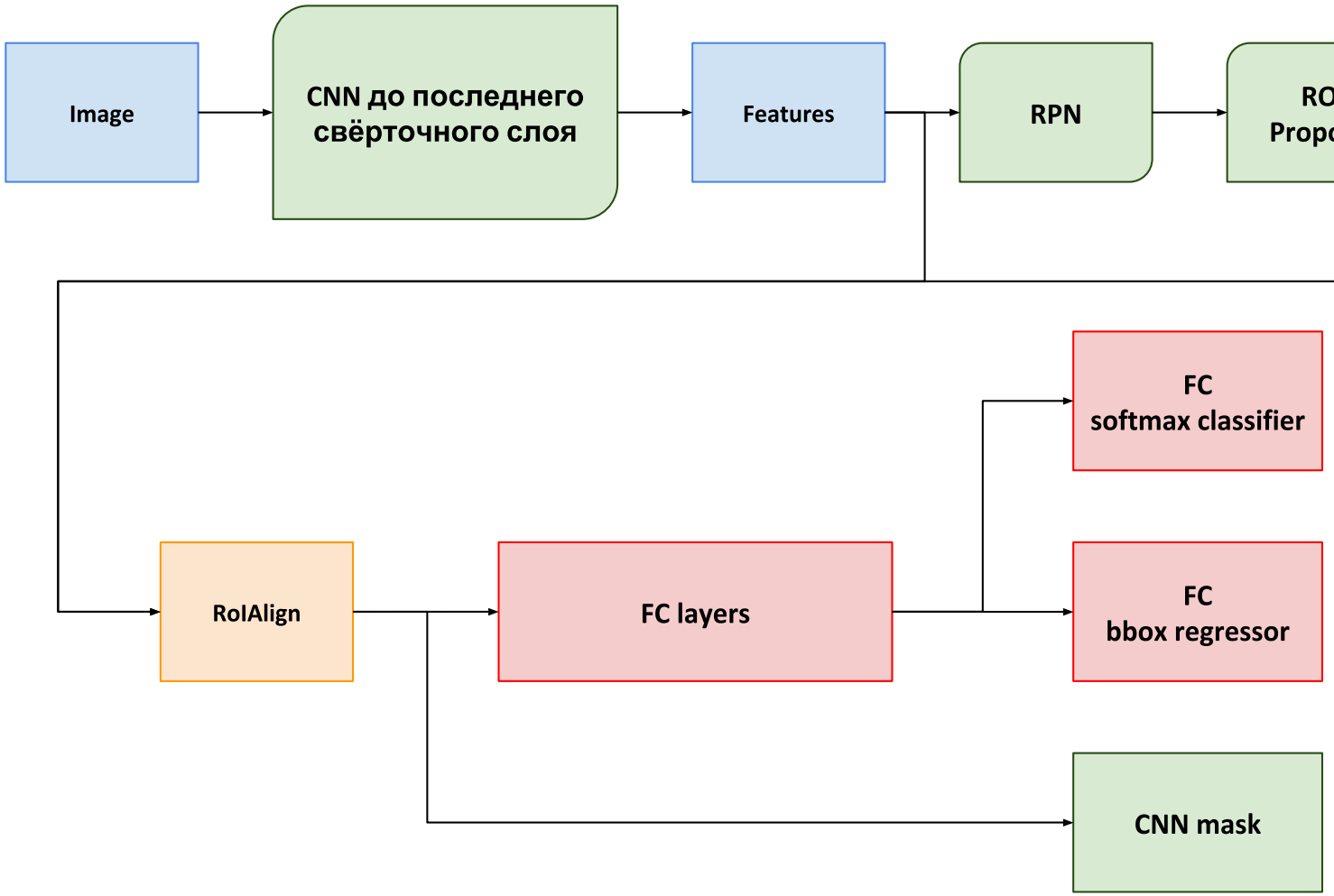






В карты признаков, извлечённые последовательными слоями CNN с уменьшающейся размерностью, рассматриваются как некая иерархическая «пирамида», называемая bottom-up pathway. При этом карты признаков и нижних, и верхних уровней пирамиды обладают своими преимуществами и недостатками: первые имеют высокое разрешение, но низкую семантическую, обобщающую, способность; вторые — наоборот

Mask-RCNN



1.Генератор особенностей (*features extractor*), который, получив на вход изображение, выдаёт 3х-мерный тензор особенностей. При этом каждому вектору особенностей из этого тензора соответствует некоторый прямоугольник исходного изображения.

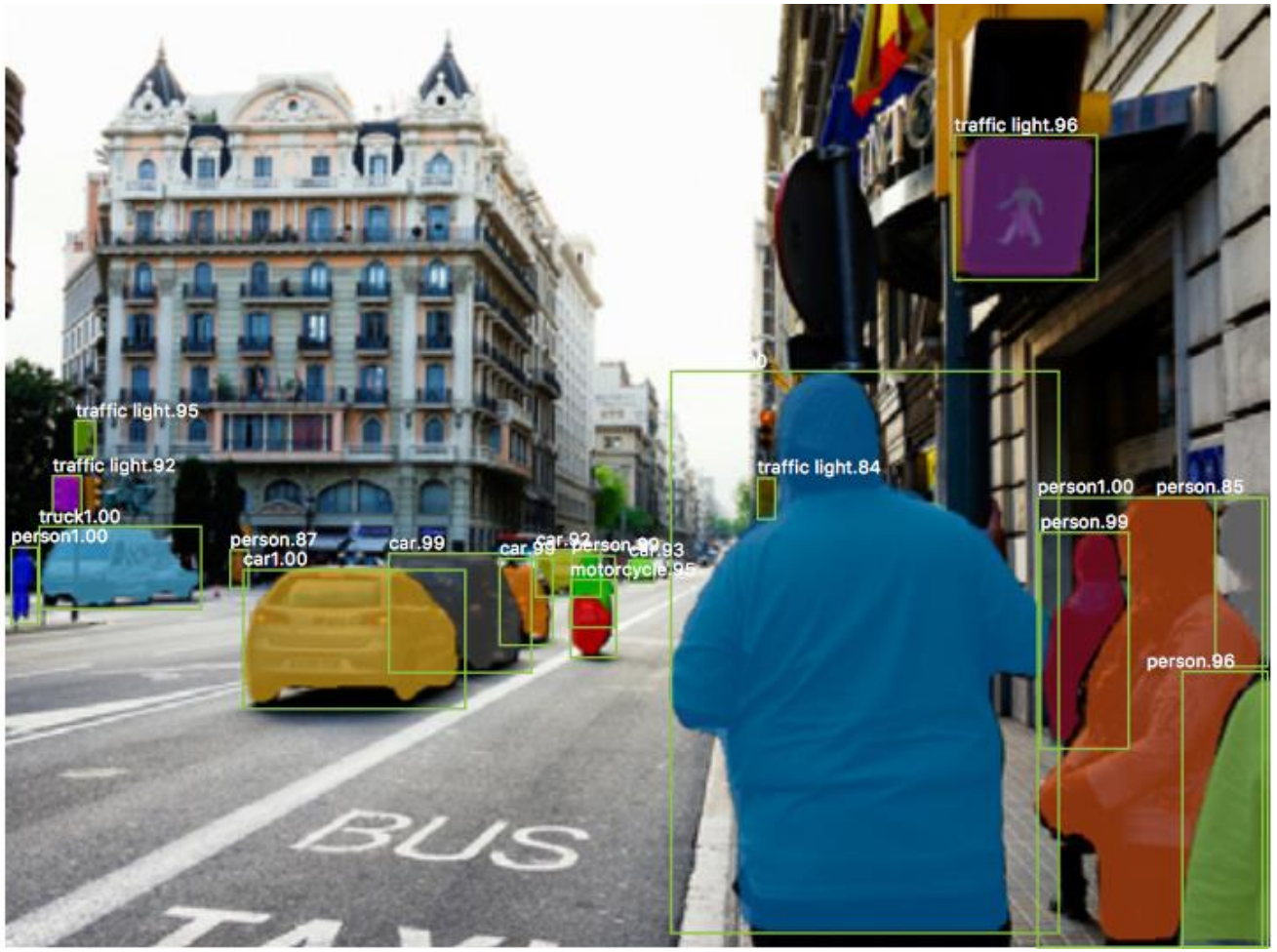
2.Region Proposal Network (RPN). Сеть, которая получив на вход тензор особенностей, генерирует некоторое количество регионов, в которых предположительно есть объекты

3.Полносвязные слои - это сеть, которая для каждого региона, вырезает, соответствующую этому региону, часть из тензора особенностей, и выдаёт

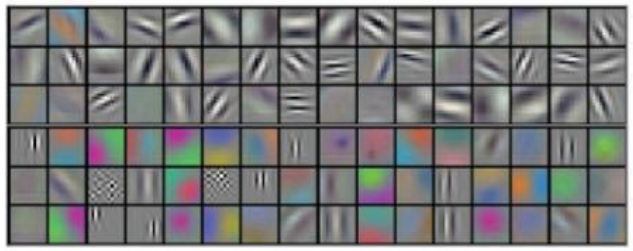
4.3.1. класс объекта (возможно, что класс будет *background*)

5.3.2. уточнённый прямоугольник объекта

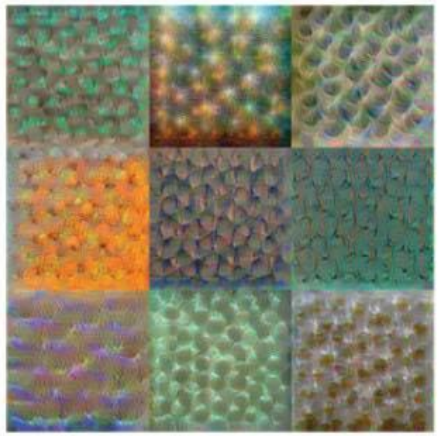
Mask RCNN


traffic light.96
traffic light.95
traffic light.92
truck1.00
person1.00
person.87
car1.00
car.99
car.92
motorcycle.95
traffic light.84
person1.00
person.85
person.99
person.96
BUS

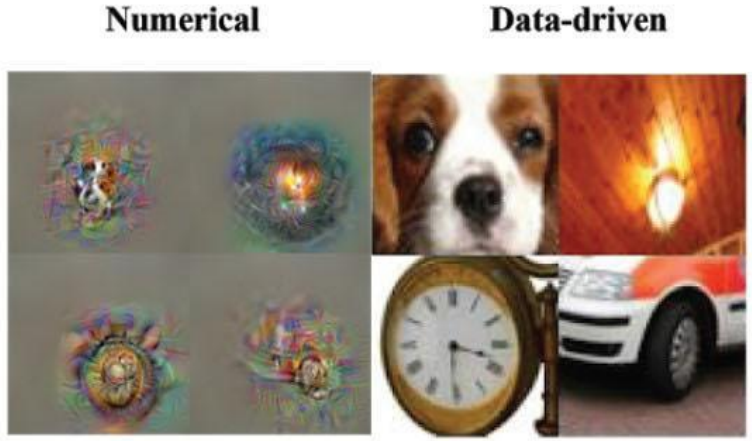
Что внутри?



Edge+Blob

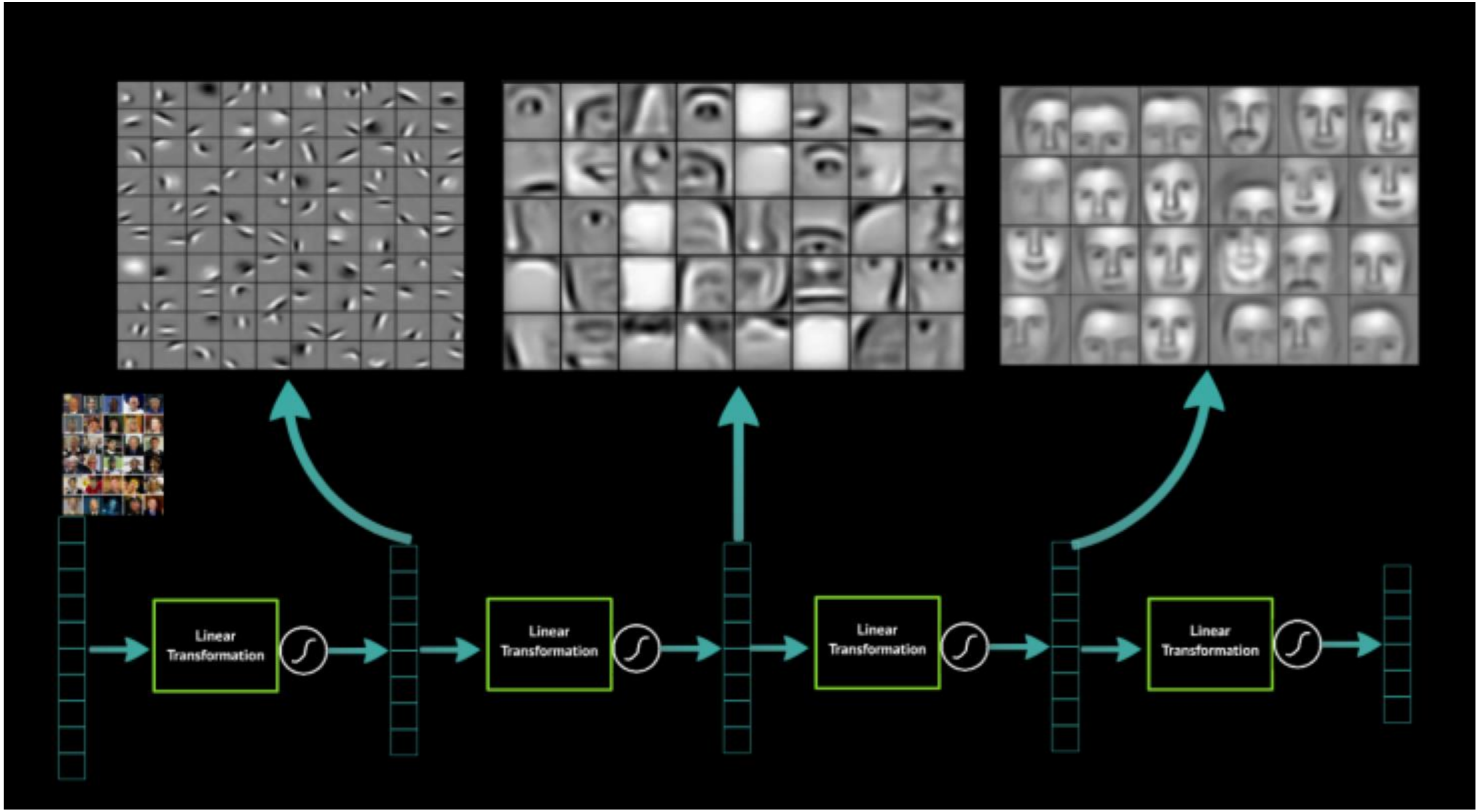


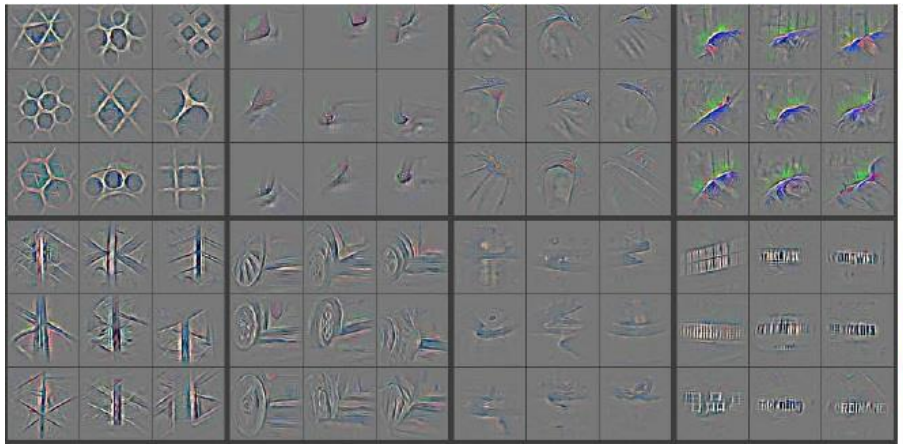
Texture

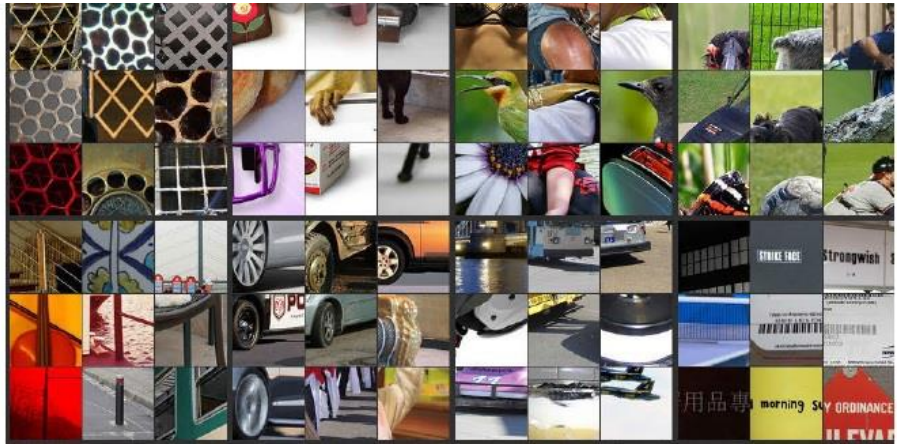


Object Parts

Object Classes


Linear Transformation
Linear Transformation
Linear Transformation
Linear Transformation





После того, как такая сеть обучилась — мы можем её использовать для классификации. Подав на вход какое-то изображение, группы нейронов первого слоя пробегаются по изображению, активируясь в тех местах, где есть соответствующий конкретному элементу элемент картинки. Т.е. эта сеть разбирает картинку на части — сначала на черточки, штрихи, углы наклона, потом более сложные части и в конце она приходит к выводу, что картинка из такого рода комбинации базовых Элементов

ОСОБЕННОСТИ?