

СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ В ОБРАБОТКЕ ИЗОБРАЖЕНИЙ

ЛЕКЦИЯ ДЛЯ ВАС, МОИ ПИРОЖОЧКИ

ЧАСТЬ I

ИЗОБРАЖЕНИЕ КАК МАТЕМАТИЧЕСКИЙ ОБЪЕКТ

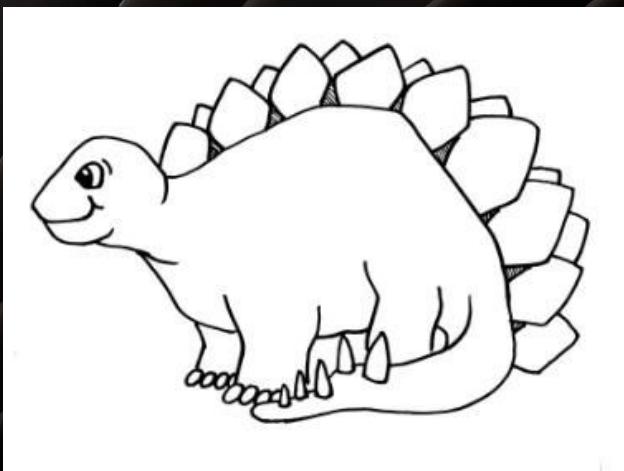
КАК ВИДИМАЯ ИНФОРМАЦИЯ СТАНОВИТСЯ НАБОРОМ ЦИФР, А ЗАДАЧА
ПРИОБРЕТАЕТ ВЫЧИСЛИМОСТЬ

ЦИФРОВОЕ ПРЕДСТАВЛЕНИЕ КАРТИНКИ

Естественное решение – bitmap (битовая карта)

- Это матрица, каждый элемент которой – яркость пикселя, выраженная числом
- Индексы элементов – положение пикселя на картинке
- Каждому цветовому слою – своя матрица
- В наших картинках бывают 1 (черно-белая), 3 (RGB) или даже 4 (RGBA) цветовых слоя

Исходная картинка



Как радуется глаз!

Монохромное изображение
(один цвет и его нет)

0	0	0	1	1	1	1	0	0	0
0	0	0	1	0	0	1	0	0	0
0	0	0	1	0	0	1	0	0	0
1	1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1
0	1	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	1	1	1
1	0	1	0	0	0	0	1	0	1
1	0	1	0	0	0	0	1	0	1
1	0	1	0	0	0	0	1	0	1

Каждый пиксель или окрашен,
или нет

Изображение в оттенках серого
(grayscale)

157	159	174	168	160	152	129	151	172	161	155	156
155	182	163	74	75	62	83	17	110	210	180	154
180	180	50	14	34	6	10	33	48	105	159	181
206	159	6	124	181	111	120	204	166	15	56	180
194	68	197	251	237	299	299	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	87	165	84	10	160	134	11	31	63	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	234	147	108	227	210	127	103	36	101	255	224
190	214	173	66	103	143	96	59	2	109	249	215
187	196	235	75	1	81	47	0	6	217	258	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	297	177	121	123	200	175	19	96	218

Каждый пиксель имеет значение
яркости в пределах шкалы

ЧТО МОЖНО СДЕЛАТЬ БЕЗ НЕЙРОСЕТЕЙ?

Да уйму всего.

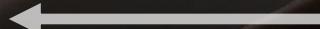
Нейросети небыстры и ресурсоемки

Если есть возможность сделать проще, легче и быстрее – надо делать

- Выделение геометрических примитивов или их комбинаций
 - Шумоподавление и очистка
 - Распознавание: OCR, QR-коды и штрих-коды
- Сегментация изображения на отдельные области или объекты
 - Сжатие изображений
 - Изменение с помощью фильтров и многое другое

МОРФОЛОГИЧЕСКИЕ ПРЕОБРАЗОВАНИЯ

Определяются ближним окружением
самих пикселей и по принципу похожи
на свертки (см. далее!)



Вот такую возьмем
базовую картинку:

Эрозия



Дилатация



Размыкание
(эррозия → дилатация)



Морфологический
градиент



Замыкание
(дилатация → эрозия)



ГЕОМЕТРИЧЕСКИЕ ПРЕОБРАЗОВАНИЯ

Определяют уже векторами, прямыми и областями картинки

Масштабирование

- Умножаем все векторы на константу
- Важен выбор типа интерполяции

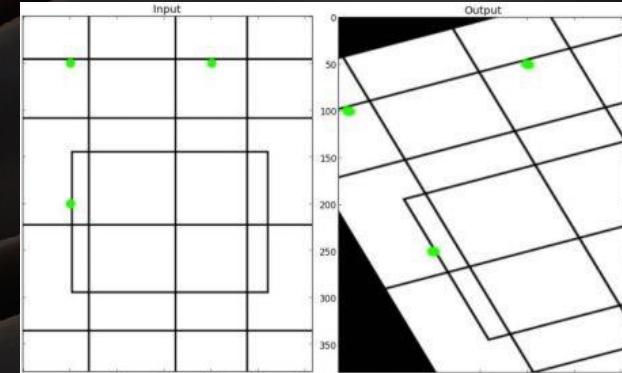
Поворот

- Умножаем все векторы на матрицу поворота 2×2



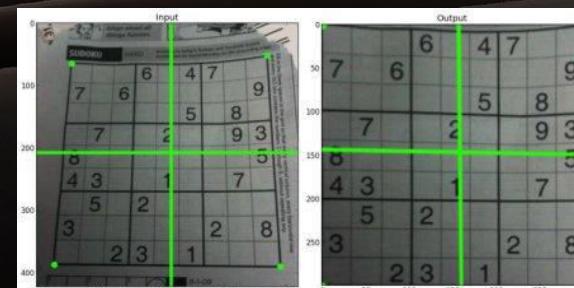
Аффинные преобразования

- Сохраняют параллельность линий



Преобразования перспективы

- Умножаем все векторы на матрицу 3×3
- Прямые остаются прямыми



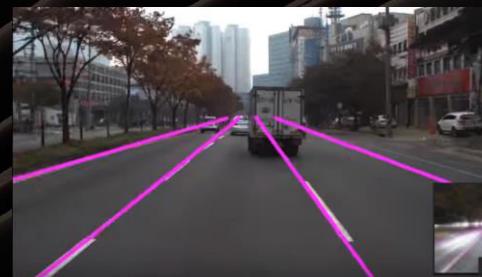
ДРУГИЕ ВОЗМОЖНОСТИ

Тысячи их!

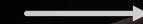
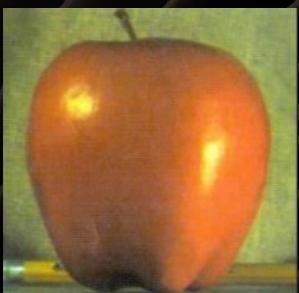
Адаптивная бинаризация



Преобразование Хафа

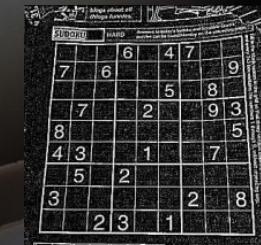
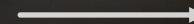


Пирамидальный блендинг

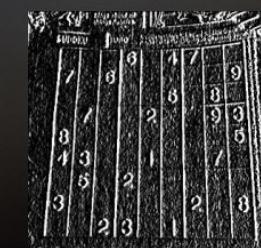
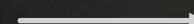


Выделение границ

Оператор Лапласа



Оператор Собеля (вертикальный)



Алгоритм Кэнни



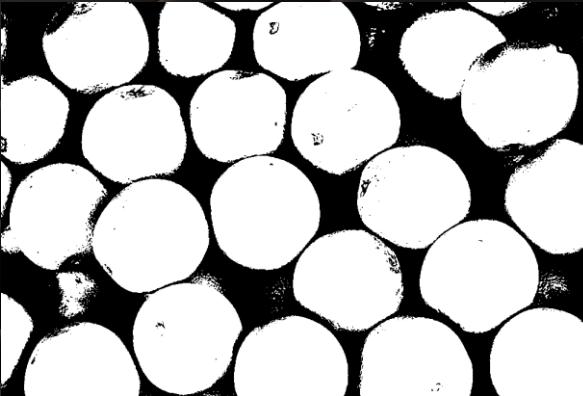
WATERSHED-АЛГОРИТМ

Решает задачу сегментации изображения

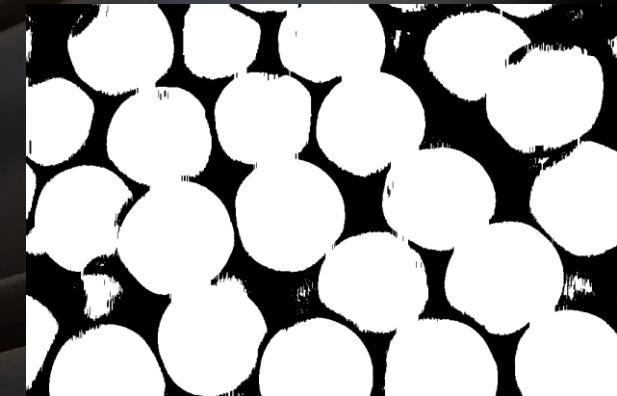
Исходное изображение



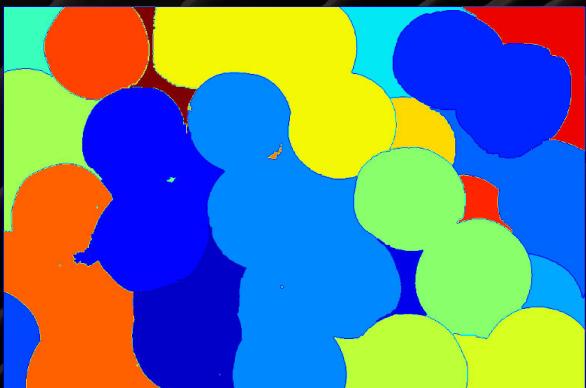
Бинаризация



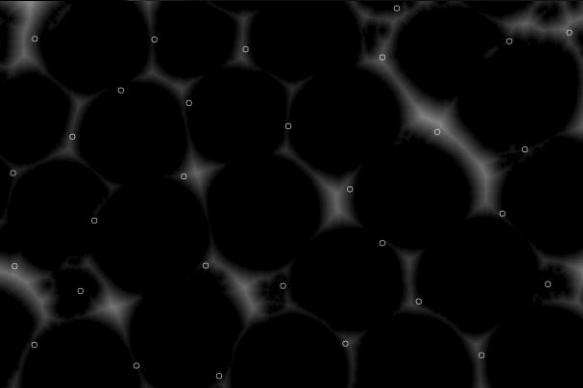
Размытие + Замыкание



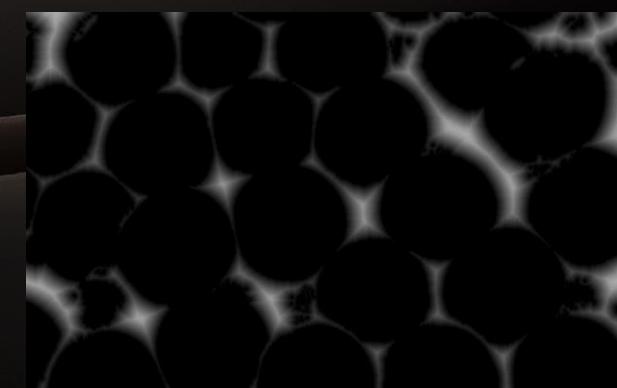
Разлив



Локальные максимумы



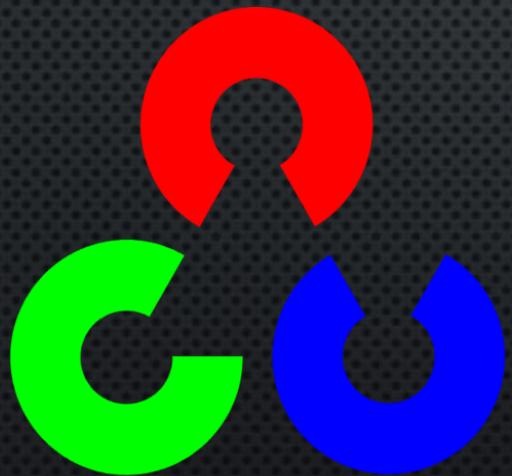
Distance transform



ИНСТРУМЕНТАРИЙ

OpenCV

Легендарный продукт.
Очень быстрая
библиотека на C++.



scikit-image

Аналог и расширение
scikit-learn. Много редких
функций и фильтров.

scipy.ndimage

Модуль `scipy` для работы
с изображениями.



ЧАСТЬ II

СВЕРТКИ И ТЕНЗОРЫ

СВЕРТКА КАК ОПЕРАЦИЯ И КАК ОБЪЕКТ, А ТАКЖЕ ЕЩЕ БОЛЬШЕ ВЕСЕЛОЙ
МАТЕМАТИКИ ДЛЯ ЮНЫХ ЛИНГВИСТОВ

ТЕНЗОРЫ

Обобщение понятия «вектор» на все случаи жизни

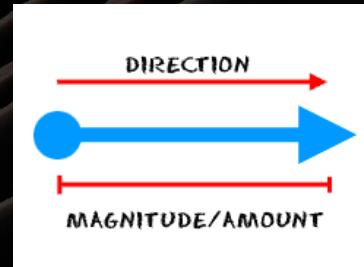
С точки зрения физики, тензорная величина характеризуется **анизотропией** – неравнозначным распределением в пространстве

Рангом тензора мы будем называть размерность его представления

Ранг 0 Скаляр

Электрический заряд

Ранг 1 Вектор (ковектор)



$$[F_x, F_y, F_z]$$

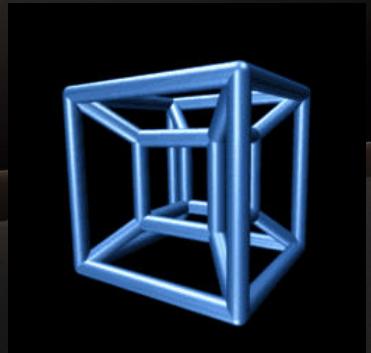
Сила, скорость

Ранг 2 Представляется матрицей

$$\bar{\epsilon} = \begin{bmatrix} \epsilon_{xx} & \epsilon_{xy} & \epsilon_{xz} \\ \epsilon_{yx} & \epsilon_{yy} & \epsilon_{yz} \\ \epsilon_{zx} & \epsilon_{zy} & \epsilon_{zz} \end{bmatrix}$$

Диэлектрическая проницаемость

Ранг >2 Многомерные массивы

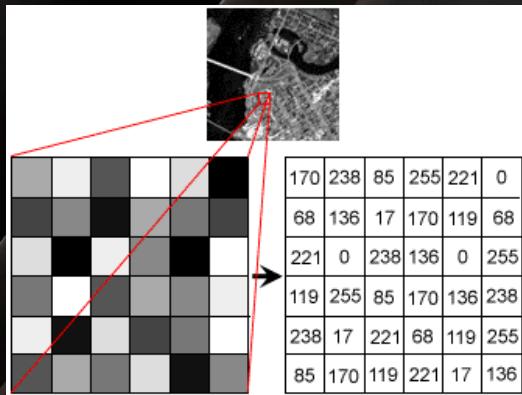


Тензор упругих постоянных (ранг 4)

ИЗОБРАЖЕНИЕ КАК ТЕНЗОР

Для нас это лишь **многомерный массив**, остальное в сторону

Черно-белое изображение



Без альфа-канала

A 4x4 matrix of floating-point numbers representing a grayscale image. The values range from 0.170 to 0.576. The matrix is color-coded by value: red for higher values, green for intermediate values, and blue for lower values.

.392	.482	.576					
.478	.63	.169	.263	.376			
.580	.79	.263	.44	.306	.376	.451	
.373	.60	.376	.478	.561			
		.443	.569	.569	.674		

RGB-изображение

С альфа-каналом

A 4x4 matrix of floating-point numbers representing an RGB image with an alpha channel. The matrix is identical to the one above, but includes an additional column and row for the alpha channel, which is represented by red values. The matrix is color-coded by value, with red being the highest and blue being the lowest.

.306	.482	.576							
.263	.478	.392	.482	.576					
.443	.569	.478	.63	.169	.263	.376	.451		
.580	.79	.263	.44	.306	.376	.478	.561		
.373	.60	.376	.478	.561	.443	.569	.674		

Почему это вообще стало важно?

В pytorch размерности
стекаются

Входящий в нейросеть батч изображений является тензором ранга 4 с размерностями (B, C, H, W), где:

- B – число изображений в батче
- C – число цветовых слоев
- H – высота изображения
- W – ширина изображения

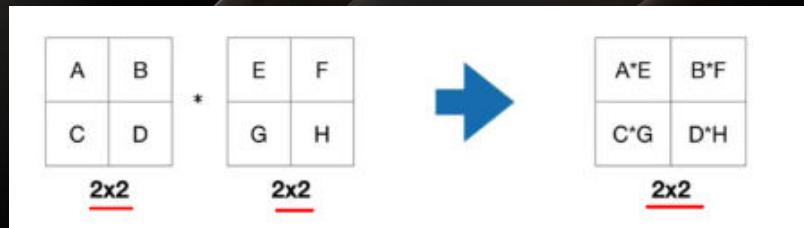
Внутри или на выходе, например,
при сегментации, ранг может
быть 5 или выше

ОПЕРАЦИИ НАД ТЕНЗОРАМИ

Во многом все как с матрицами, но есть нюансы

1. Поэлементные операции:

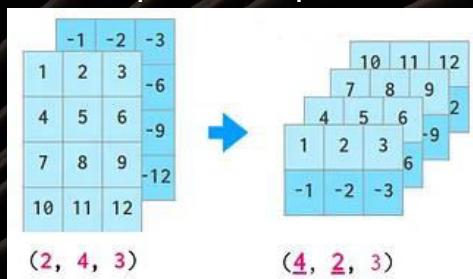
- Сложение/вычитание
- Умножение тензора на число
- Поэлементное умножение/деление



Над тензорами только одинаковой размерности

3. Транспонирование

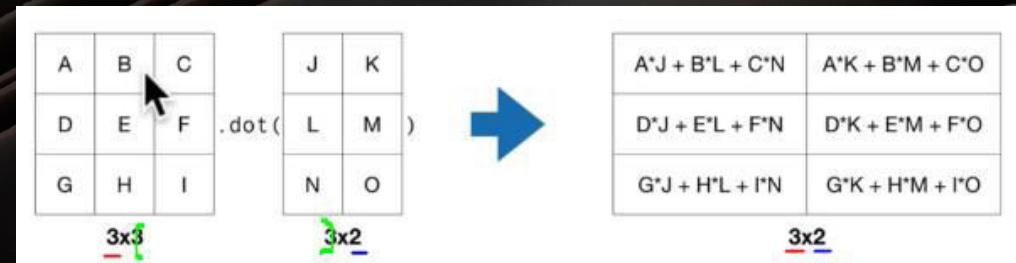
Перестановка размерностей местами



Частный случай – `squeeze` (сжатие одной размерности) или `flatten` (всех)

2. Тензорное произведение

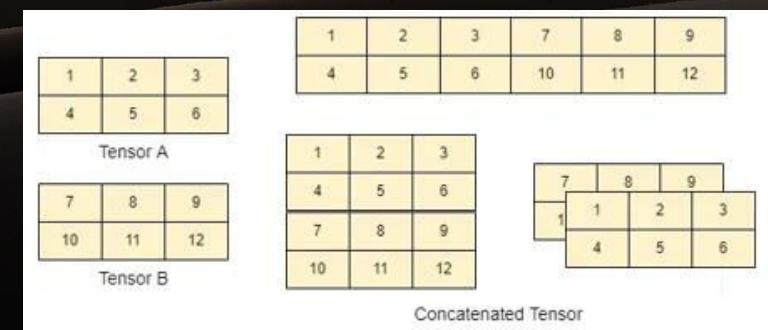
Аналог скалярного произведения векторов и произведения матриц



Над тензорами только определенных размерностей

4. Конкатенация

Приклеивание одного тензора к другому вдоль одной из осей размерностей

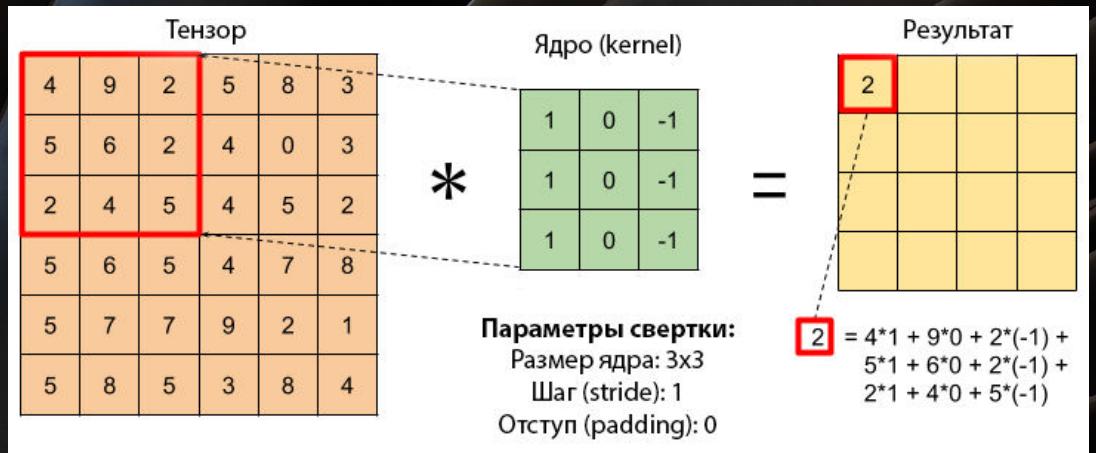


СВЕРТКА КАК ОПЕРАЦИЯ

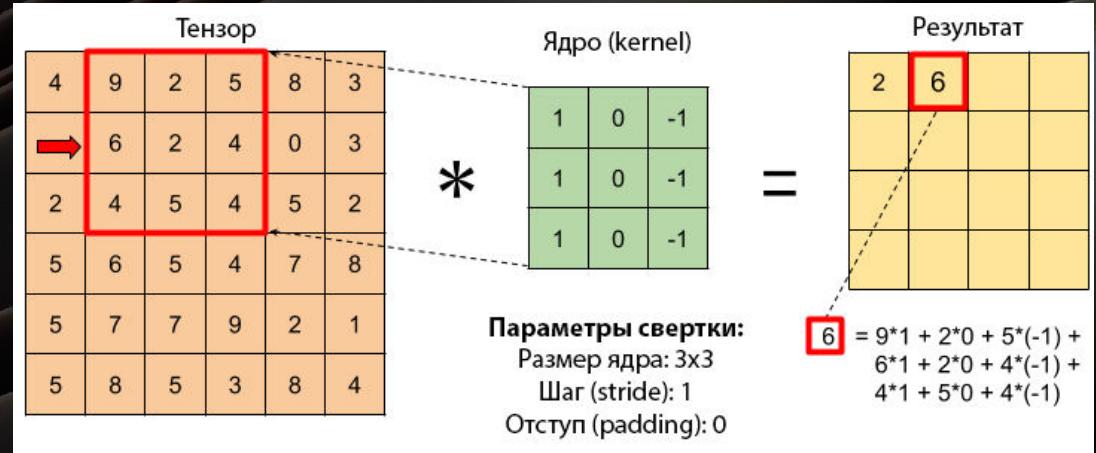
Математически, свертка (convolution) – комбинация из произведения Адамара (поэлементного) и суммирования полученного результата

На примере простого тензора 2 ранга

Шаг 1



Шаг 2



и так далее...

Параметры свертки

Размер ядра (kernel size)

Чем больше ядро, тем меньше получается размерность результата

Шаг (stride)

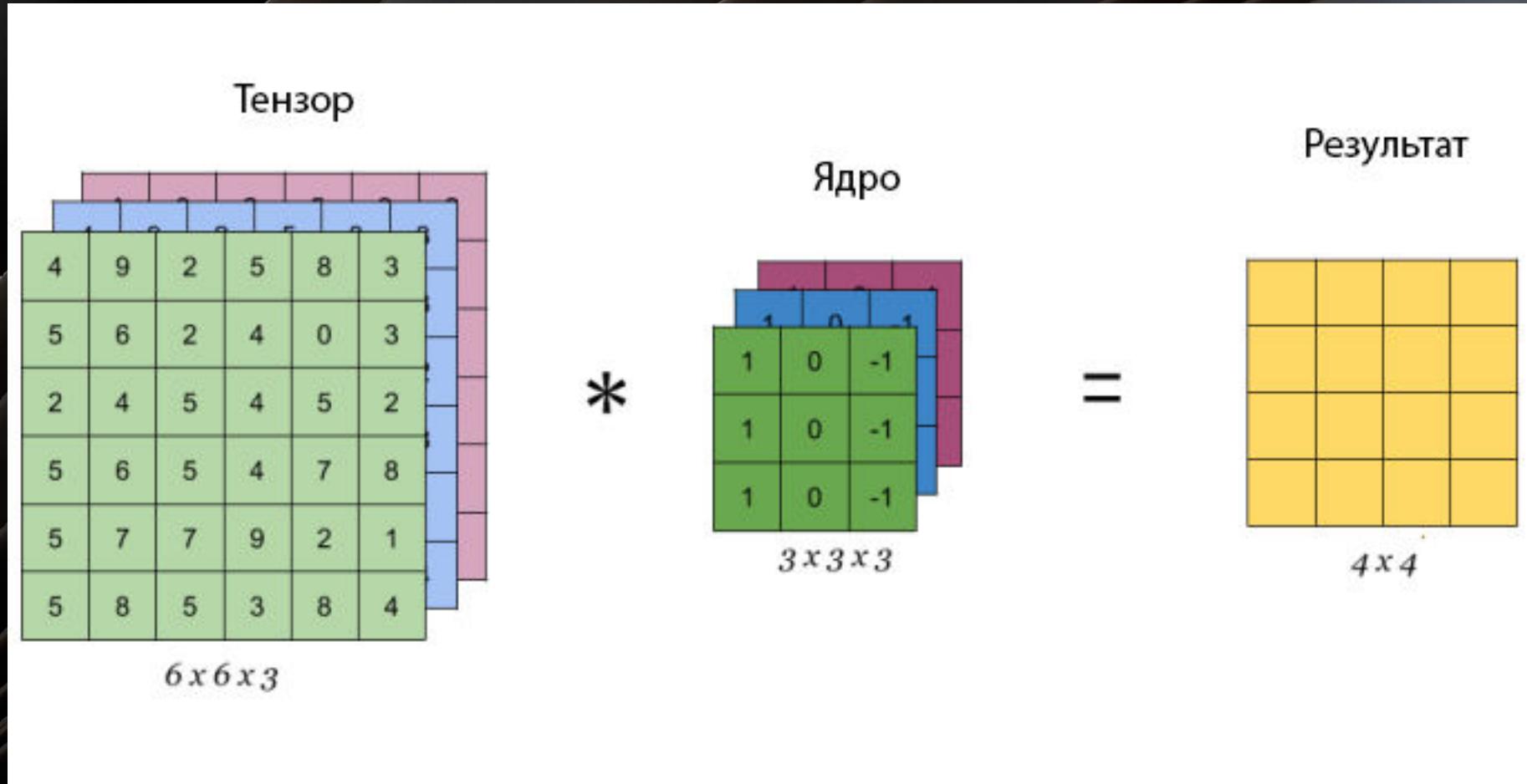
Чем больше шаг, тем меньше получается размерность результата

Отступ (padding)

Окружение тензора внешними рядами заданных чисел,
Чтобы сохранить размер при свертке

СВЕРТКА В МНОГОМЕРНОМ СЛУЧАЕ

Ничуть не сложнее



СВЕРТКА КАК ФИЛЬТР

Ваш любимый Фотошоп

Если сохранять размерность исходного тензора, например, с помощью отступа, то свертку можно использовать для преобразования изображений

Горизонтальный градиент (ядро Собеля)

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix}$$

Вертикальный градиент (ядро Собеля)

$$\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Увеличение резкости (sharpen)

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Разные ядра

Гауссово размытие

1	4	7	4	1
4	16	26	16	4
7	26	41	26	7
4	16	26	16	4
1	4	7	4	1

1/273

Размытие по рамке (box blur)

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Шестиугольный эффект боке*

255	255	255	255	255	255	255	255	0	255	255	255	255	255	255	255
255	255	255	255	255	255	0	0	0	0	0	0	0	0	0	255
255	255	0	0	0	0	0	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
255	255	255	255	255	255	0	0	0	0	0	0	0	0	0	255
255	255	255	255	255	255	0	0	0	0	0	0	0	0	0	255
255	255	255	255	255	255	0	0	0	0	0	0	0	0	0	255

*применяется с нормировкой и max вместо суммирования по свертке

ИНСТРУМЕНТАРИЙ



scikit-image

Много самых разных
фильтров

scipy.ndimage

Есть как встроенные фильтры,
так и функция generic_filter с
любым кастомным ядром,
какое душа пожелает



ЧАСТЬ III

СВЕРТОЧНЫЕ НЕЙРОННЫЕ СЕТИ

Ну наконец-то

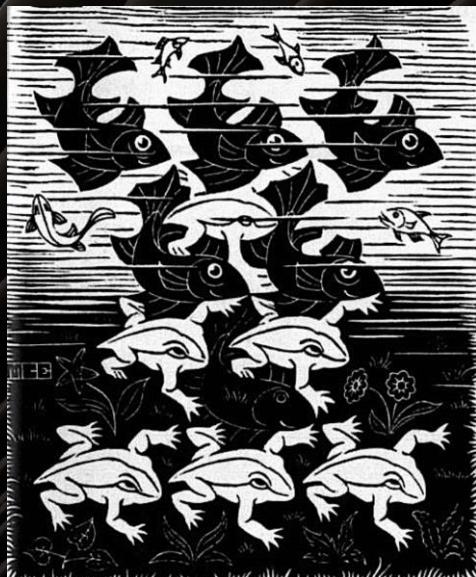
ИЗОБРАЖЕНИЯ И ОБЫЧНЫЕ FFNN-НЕЙРОСЕТИ

Гвоздь и микроскоп: уроки и итоги

Вопрос к вам, дорогие!

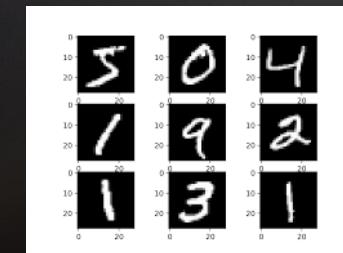
Как засунуть изображение в многослойный перцепtron и почему этого лучше не делать?

Как мы понимаем, что изображено на картинке?



ЕМКОСТЬ

Датасет MNIST

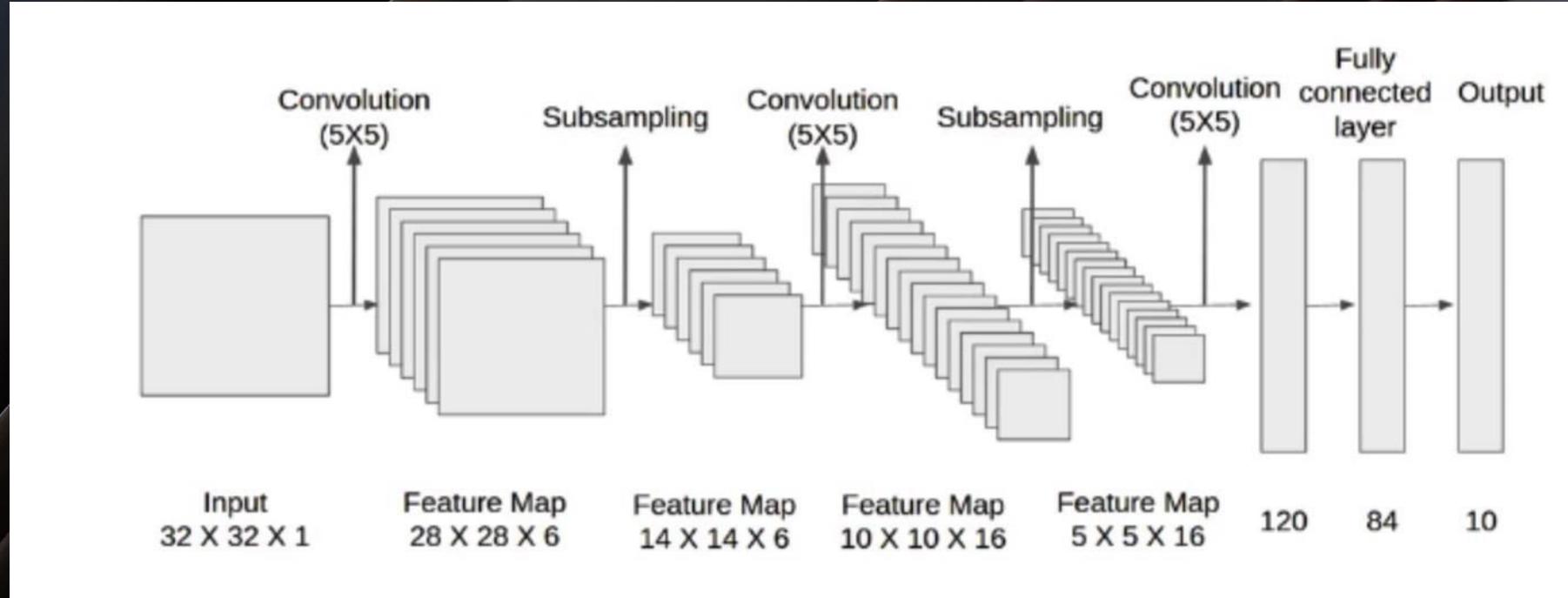


70000 рукописных цифр
размера 28x28 – прадедушка
современных CV-датасетов

С ним справится и SVM-
классификатор, но

CNN.НАЧАЛО.АРХИТЕКТУРА СЕТИ LENET

Родом из девяностых и забыта больше, чем на 10 лет



Изобретена Яном Лекуном в 1990
Использовалась для
классификации рукописных
цифр и букв

Три типа слоев:

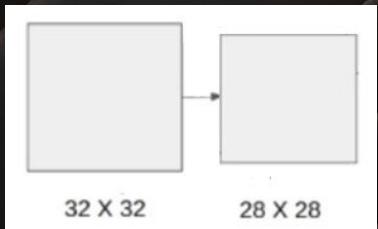
- сверточные
- сжатия
- полносвязные

Использовала обычные
функции активации
типа сигмоиды и $tanh$

СВЕРТКА КАК СЛОЙ. ПРИНЦИП

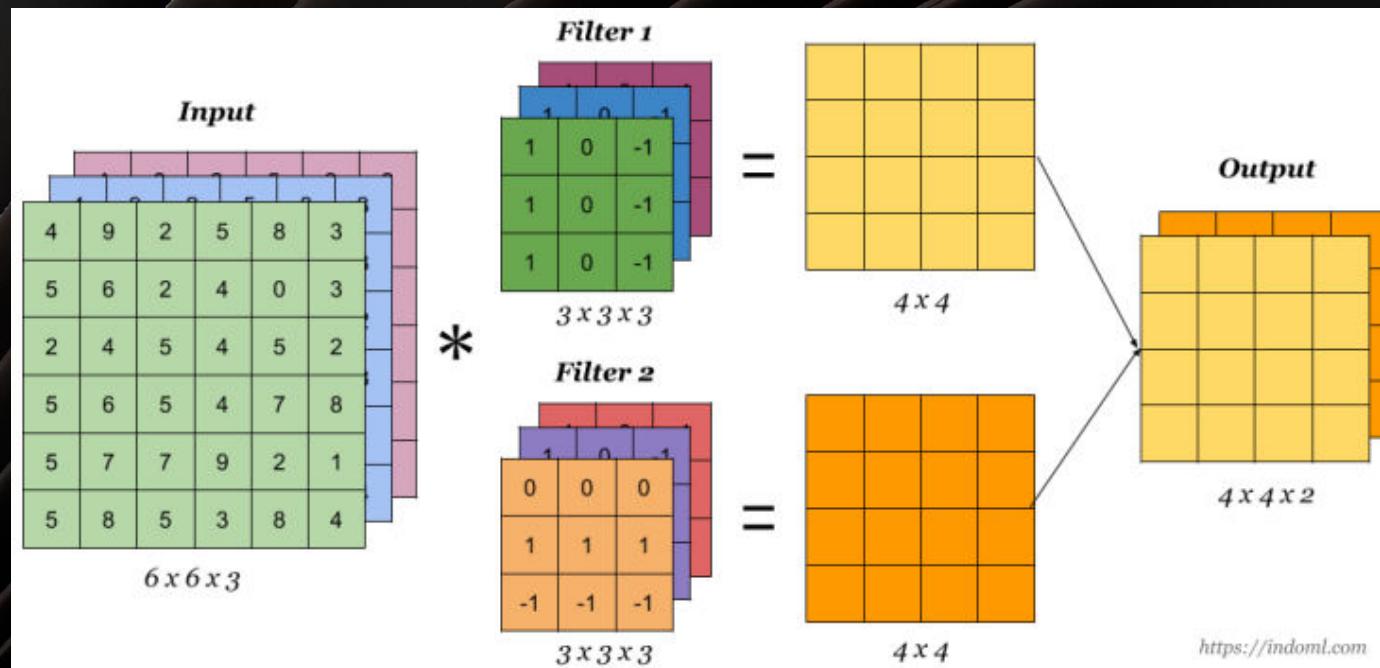
Изящная идея превращения с малым числом параметров

Представим ситуацию с прошлой картинки – первый переход



Сколько параметров нужно было бы для полносвязного слоя?

Для свертки с ядром 5x5 – всего 25 параметров



Идея – взять несколько **сверток-фильтров** и сложить результаты в тензор-стопку

Идея – сделать эти **фильтры** обучающимися при бэкпропе

Функция активации применяется на результат поэлементно

СЛОИ СЖАТИЯ. POOLING

Способ избавиться от лишних данных и разгрузить сеть

Свертка без отступа сама уменьшает изображение, но этого недостаточно

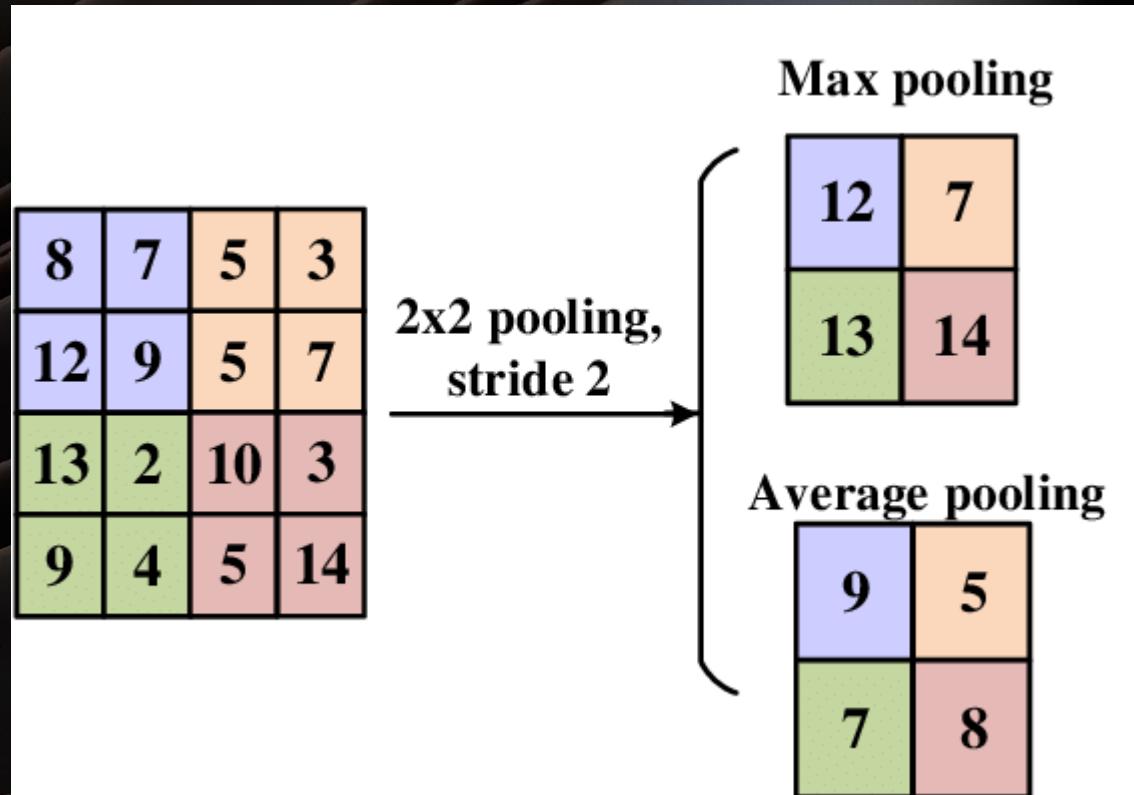
Max Pooling

- Изображение разбивается на квадраты $n \times n$
- В каждом квадрате берется максимум
- Он ставится в конечное изображение

Average Pooling

- Ровно то же самое, но вместо максимума берется среднее

Эмпирически выяснило,
что Max лучше Avg

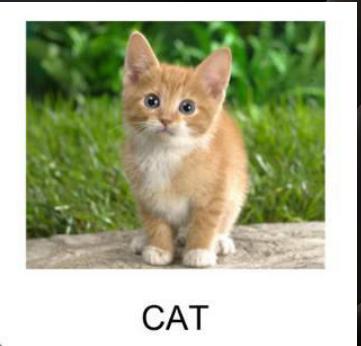


Как и в свертках, здесь есть параметр **stride**.
Он **должен быть равен размеру pooling!**

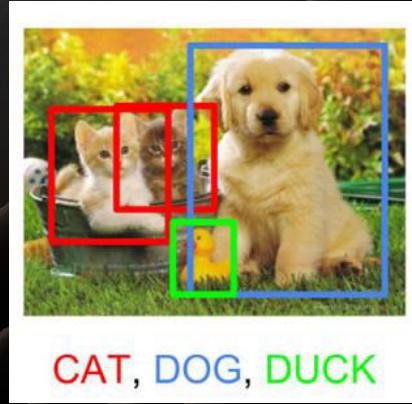
ЗАДАЧИ, РЕШАЕМЫЕ CNN

Аналитические

Классификация



Детекция/локализация



Семантическая сегментация



Instance-сегментация



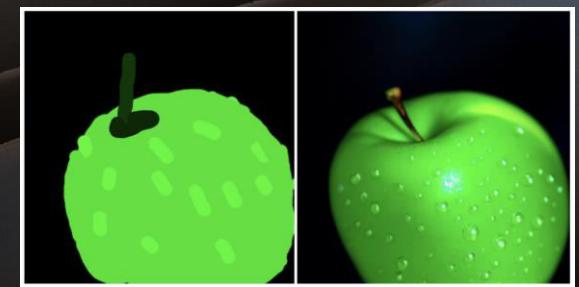
Генеративные

txt2img

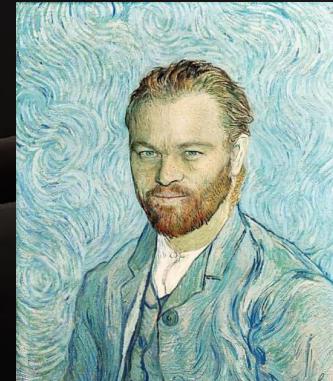


PaperCut R2-D2

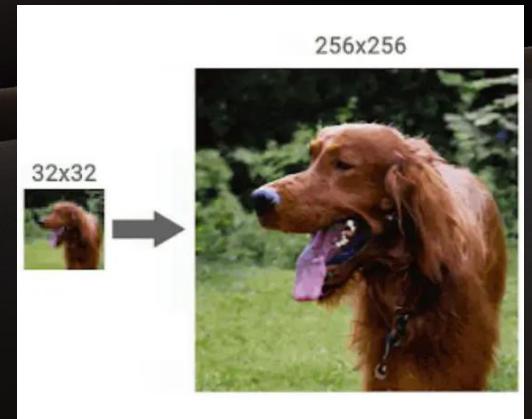
img2img



Style transfer



Upscaling



CNN ДЛЯ КЛАССИФИКАЦИИ. IMAGENET

Огромное поле для соревнований и обкатки



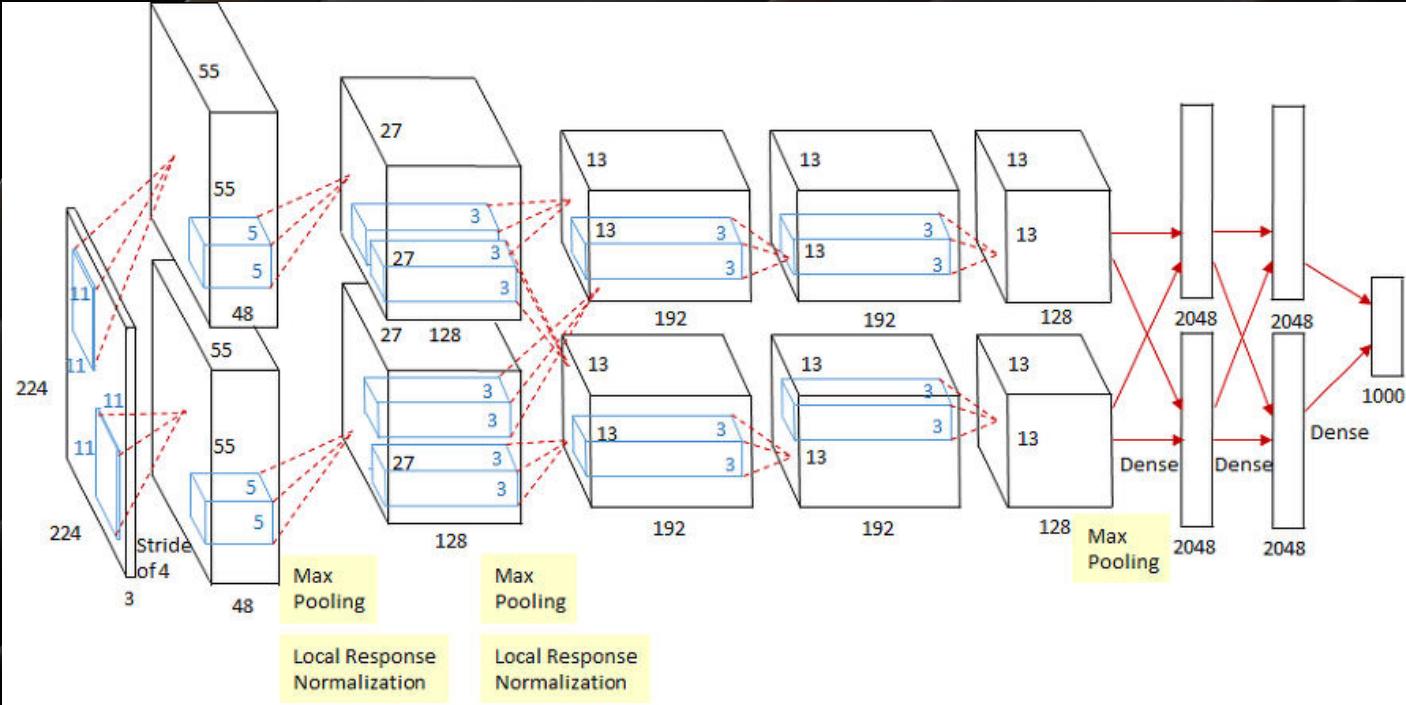
ImageNET – огромный масштабируемый датасет размеченных изображений, используемый для развития распознавания образов посредством машинного обучения

Подсет ImageNET-21k имеет более **14 млн** изображений, отнесенных к более чем **21 тысяче** классов

С 2010 года используется для **ILSVRC** – соревнований моделей по распознаванию
Как бои роботов, только круче!

CNN ДЛЯ КЛАССИФИКАЦИИ. АРХИТЕКТУРА ALEXNET

2012 принес революцию



62,3M
параметров

В 2012 Алекс Крижевский представил AlexNet на ILSVRC и порвал всех конкурентов и коллег с отрывом

Архитектура AlexNet кажется сложной, но на самом деле очень похожа на LeNet, взятую два раза

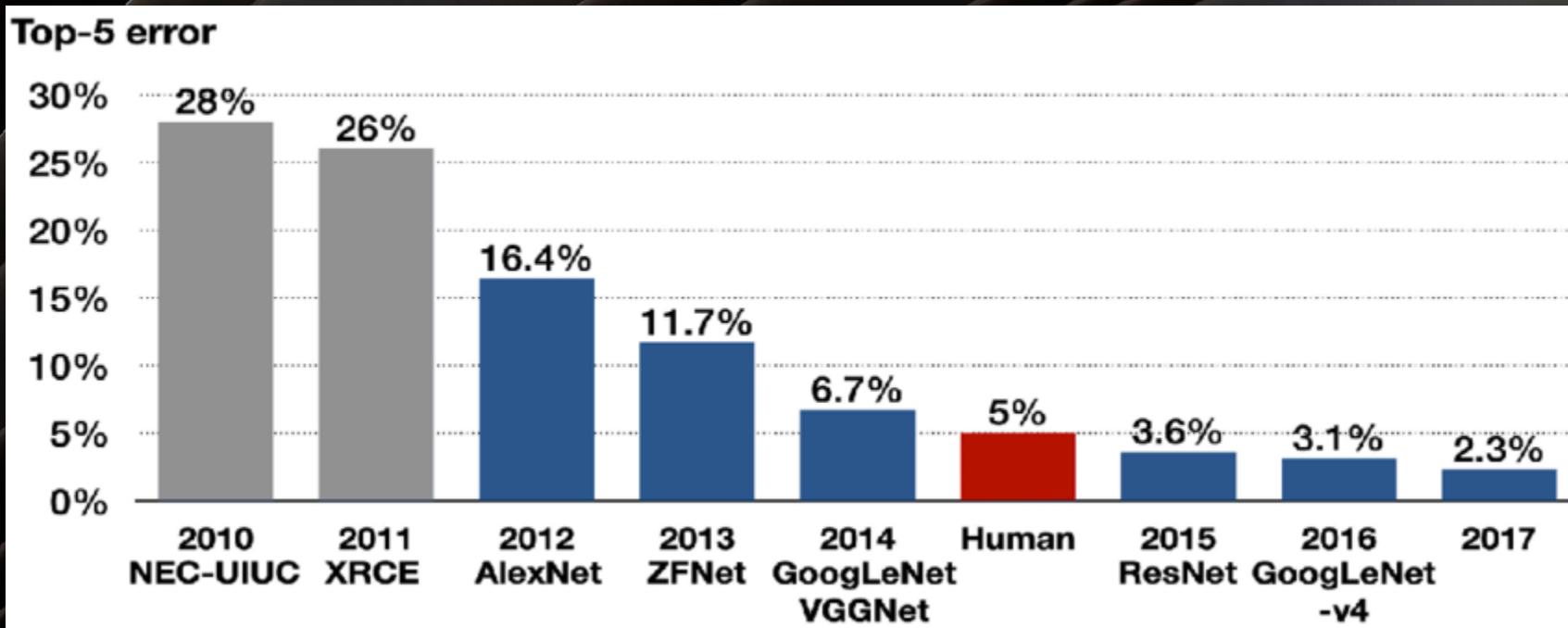
Вместо average pooling взят максимум, и в качестве функции активации – **ReLU**

Это улучшило работу сети!

CNN ДЛЯ КЛАССИФИКАЦИИ. ЭВОЛЮЦИЯ

За революцией

Важность ILSVRC



Обратите внимание – уже в 2015 ResNet в классификации нагибала кожаных мешков!

CNN ДЛЯ КЛАССИФИКАЦИИ. ДАЛЕЕ

Взрывообразный рост и развитие CNN

Проблемы:

1. Разрастание числа параметров
2. Увеличение глубины сетей
3. Выключение нейронов ReLU

Решения:

1. Работа с архитектурой: skip-connections, свертки 1x1, bottleneck-слои
2. Масштабируемые сети (depth scaling)
3. Использование более продвинутых функций активации: ELU, Softplus, LReLU

EfficientNet B4 имеет всего **19М параметров**, а результат дает **96,3%** в Top5 ILSVRC

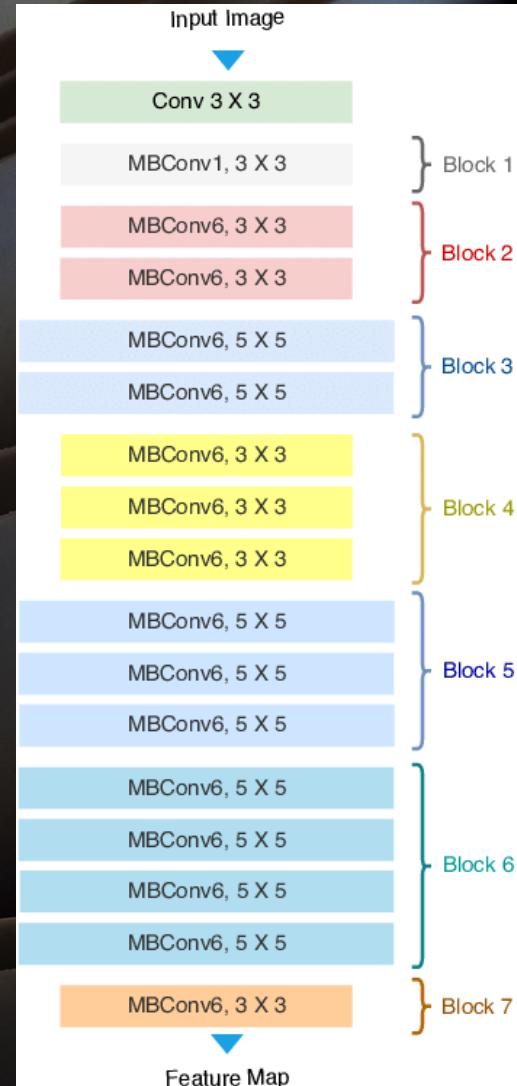
Состоит из масштабируемых MBConv-блоков

Обычные свертки становятся комбинацией послойных сверток (depthwise separable) и поточечных сверток 1x1 (pointwise)

Позволяет очень сильно сократить число параметров

На выходе каждого блока ставится тонкий bottleneck-слой БЕЗ функции активации – как будто новый input, но поменьше

Позволяет не терять нейроны на лишнем слое ReLU



11M параметров!

CNN В СЕМАНТИЧЕСКОЙ СЕГМЕНТАЦИИ

Принципиально иная задача



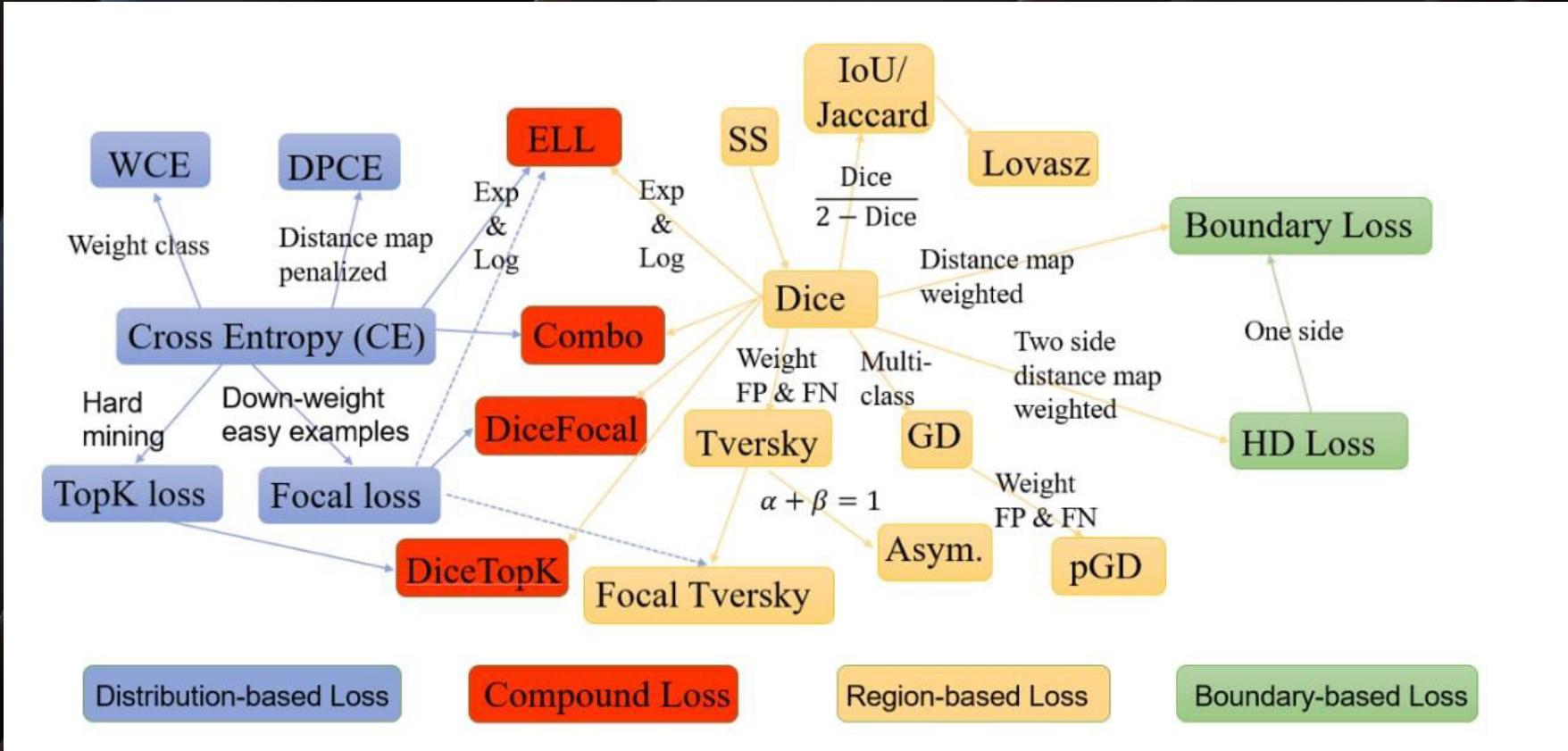
Сегментация не определяет объекты, исключительно области изображения

Применяется, например, при программировании беспилотных автомобилей, чтобы разделять зоны внимания

На выходе такой задачи сеть возвращает **маску сегментации**, т.е. **изображение**

CNN В СЕМАНТИЧЕСКОЙ СЕГМЕНТАЦИИ. LOSS

Как определить эффективность сегментации?



При обучении сравниваются две маски – заданная и полученная

Точнее, сравнивается каждый регион сегментации в этих масках

Глобальный принцип – найти баланс между поточечной и региональной функциями потерь

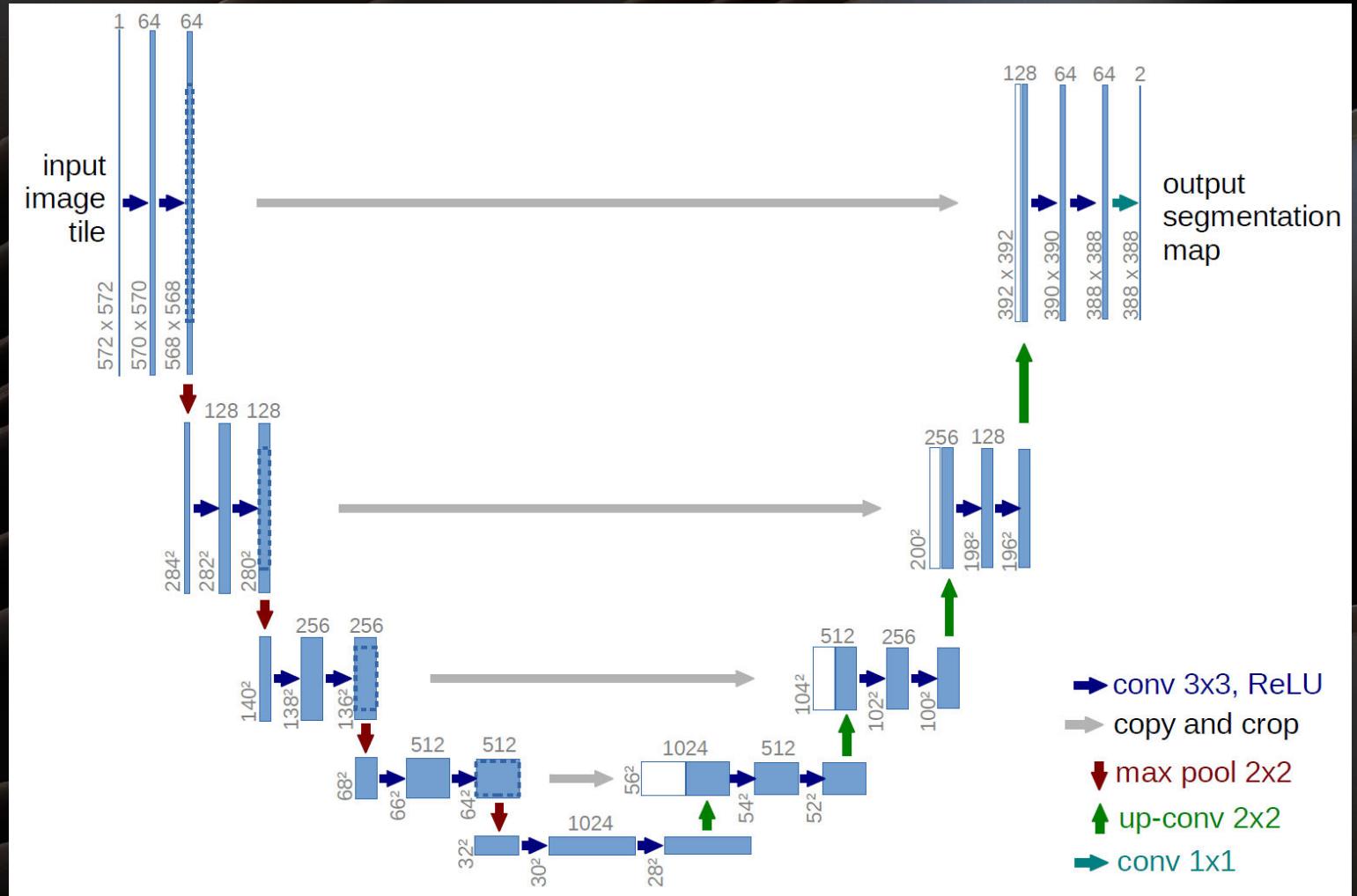
CNN В СЕМАНТИЧЕСКОЙ СЕГМЕНТАЦИИ. UNET

Изображение на выходе сети
порождает
Две принципиально новых вещи
в архитектуре

Зеленые стрелки – upsampling-слои
Действуют обратно
сверточным, также называются
deconvolutional (разверточные)

Серые стрелки – skip connections
Прицепляют копию свернутых
данных на этапе развертки,
чтобы сеть не забывала
привязывать маску к
исходному изображению

Основная архитектура



CNN В ДЕТЕКЦИИ

Задача порядком сложнее предыдущих



Разделяется на несколько подзадач:

1. Классификация объектов на изображении
2. Определение границ (bounding box) объектов требуемых классов
3. Дискриминация перекрывающихся объектов

Производная задача – трекинг: детекция объекта в кадре и отслеживание **его** же на следующих кадрах

CNN В ДЕТЕКЦИИ. МОДЕЛИРОВАНИЕ ЗАДАЧИ

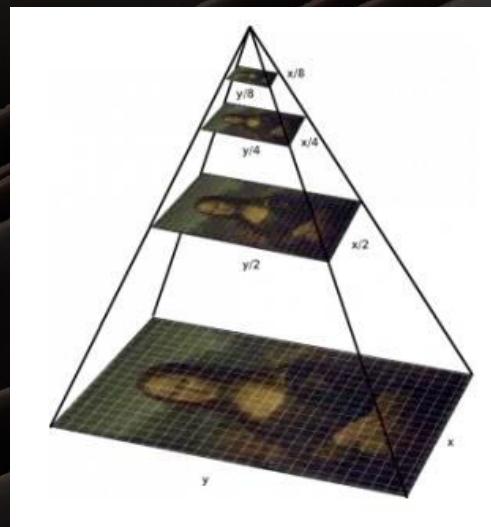
Основной принцип

В базе детекция сводится к классификации объекта в рамках не всего изображения, но некоего **заданного кусочка**

По изображению запускали **скользящее окно**, в каждой итерации которого проводили классификацию

Первая проблема – как выбрать размер этого окна? На одних фото собака маленькая, на других - большая

Для этого придумали предварительно выделять некоторые признаки и по ним ограничивать размеры окна и потенциальные области поиска



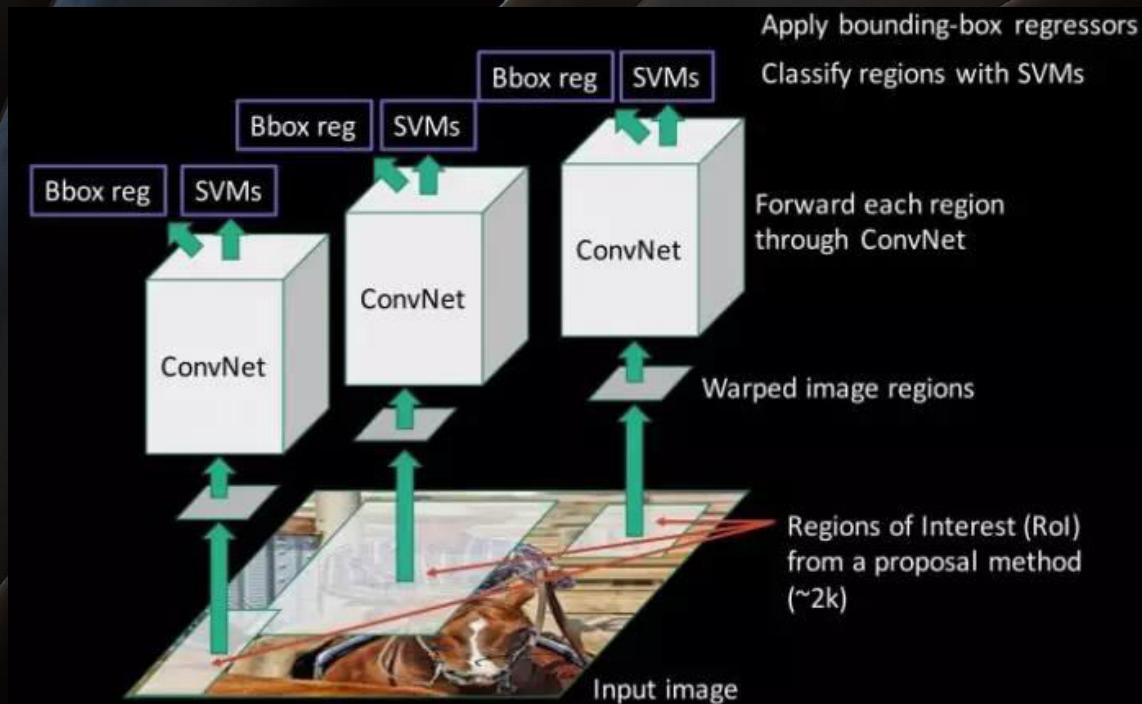
Пирамиды + HOG



Гистограммы ориентированных градиентов выделяют признаки в скользящих окнах на пирамидах, и их можно уже пихать в классификаторы

CNN В ДЕТЕКЦИИ. R-CNN

Region-based сверточные нейросети



В случае CNN подход со скользящим окном не годится – слишком много расчетов
Как быть? Как выделять области детекции?

Selective Search – специальный метод выделения ROI (regions of interest) на основе **семантической сегментации**.
Их небольшое число, < 2000.

На каждом ROI строится своя CNN, выход которой отправляется на **классификатор** – для определения класса объекта; и на **регрессор** – для определения **границ**.

Недостатки:

1. Все равно медленно и ресурсоемко – 2000 CNN
2. Selective Search - отдельный алгоритм

CNN В ДЕТЕКЦИИ. FAST RCNN, FASTER RCNN

Давайте быстрее

FastRCNN

Радикальное развитие идей RCNN

Выделение ROI запихнули внутрь нейросети, введя специальный слой SPP: spatial pyramid pooling, основанный на концепции пирамиды изображений

Регрессор bounding box и классификатор также встроили внутрь сети, так что теперь можно было работать end2end

Результат – намного более быстрая работа!

FasterRCNN

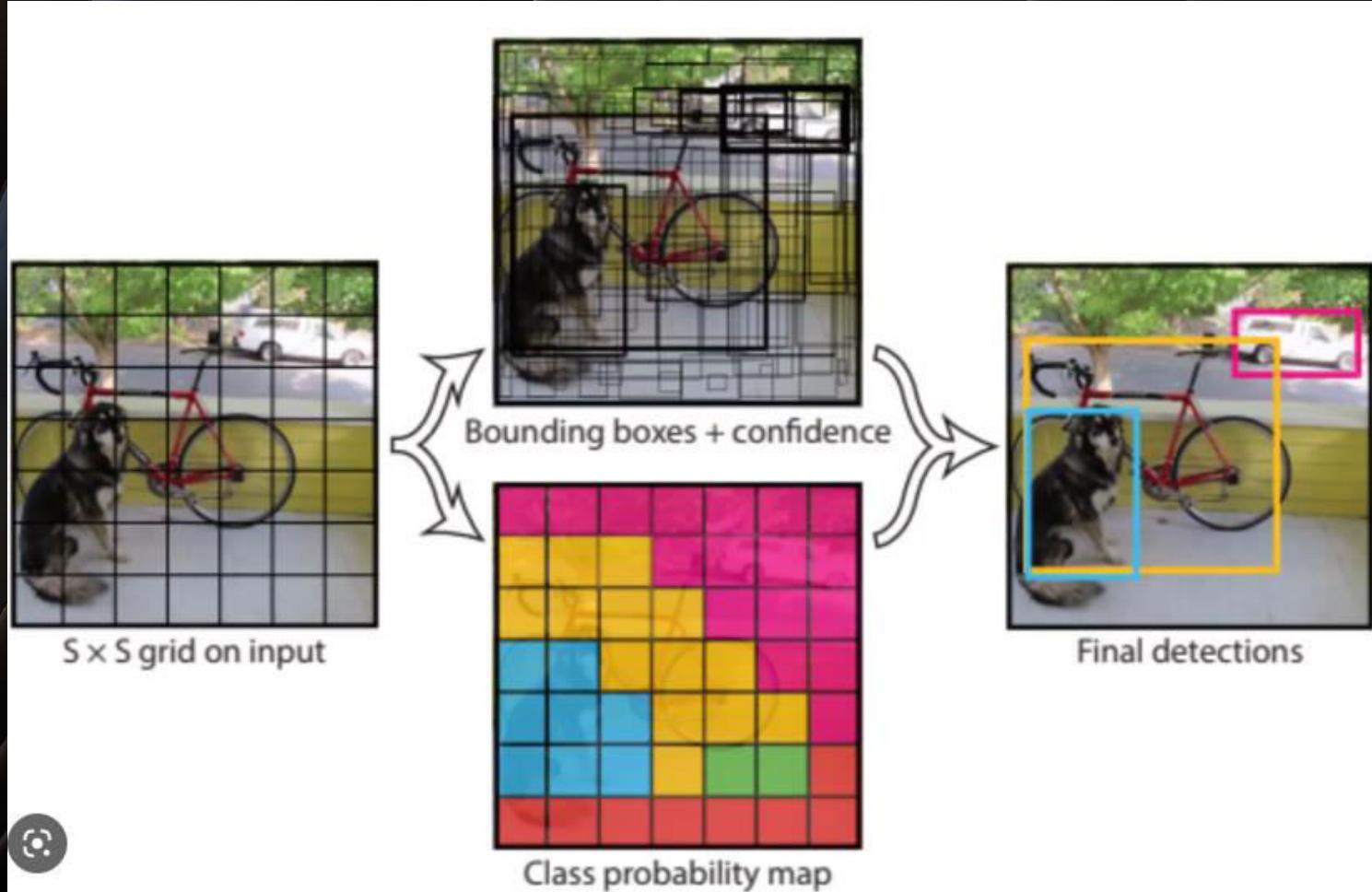
Замена алгоритма Selective Search на отдельную небольшую быструю нейросеть RPN – Region Proposal Network

Введение принципа anchor boxes – набора тестовых bounding box определенного размера, которые затем гоняют по изображению

Конечный регрессор **подгоняет** размеры итогового bounding box с anchor под реальные

CNN В ДЕТЕКЦИИ. YOLO

You Only Look Once – достоинства FasterRCNN + разбиение на сетки



Действительно, один раз сеть видит картинку

Принцип довольно сложный – основан на разбиении изображения на сетку размером $S \times S$ пикселей

Для каждой ячейки вычисляются кандидаты в bounding box и вероятности найти здесь объекты соответствующих классов

Результаты объединяются и обобщаются с помощью специального механизма NMS – Non-Maximum Suppression

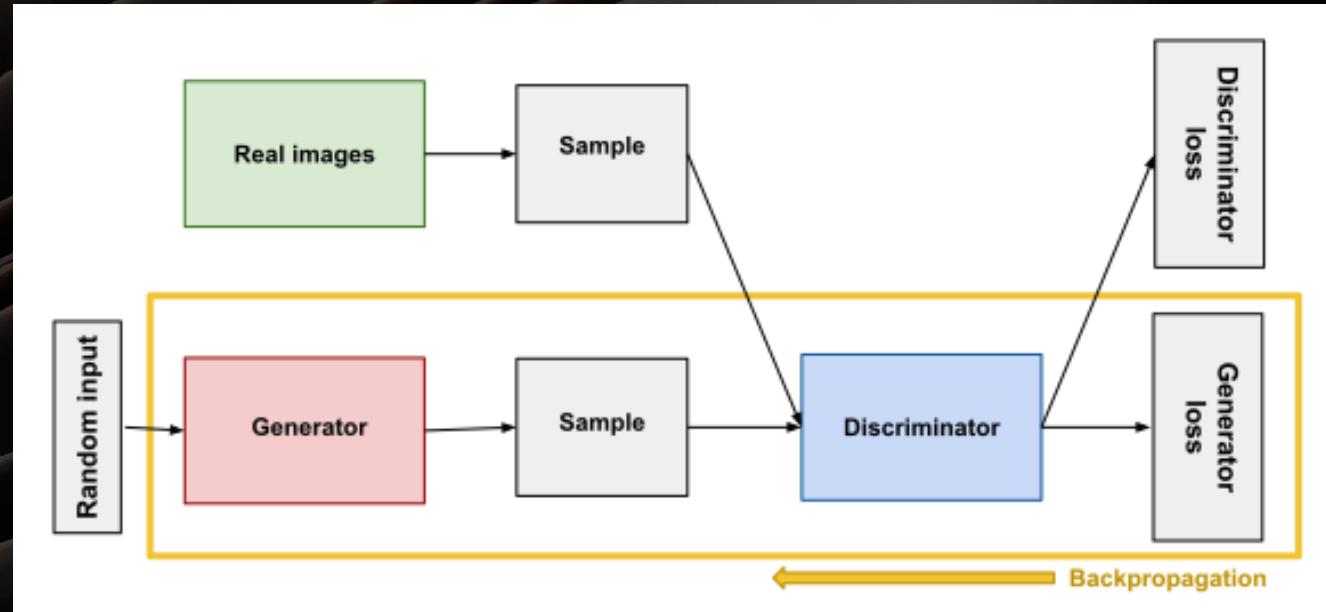
ГЕНЕРАТИВНЫЕ CNN. GAN

Принцип соревновательного обучения

GAN – генеративная
соревновательная сеть

Называется так, потому что состоит
из двух «борющихся» частей:
генератора и дискриминатора

Генератор создает пробные
объекты **из случайного шума**, а
дискриминатор сравнивает их с
образцами и штрафует генератор



К **латентному вектору** исходного случайного шума можно подцеплять условную
или дополнительную информацию для создания объекта нужного класса или с
заданными свойствами – ConditionalGAN, InfoGAN

ГЕНЕРАТИВНЫЕ CNN. ДИФФУЗИОННЫЕ МОДЕЛИ

Обучение восстановлению из гауссова шума



Irish Setter

Весь передний край AI-генерации: Imagen, DALL-E 2, Midjourney, Stable Diffusion, dreambooth – весь основан именно на диффузионных моделях

Диффузионные модели **поэтапно** «растворяют» данные в случайном шуме, взятом из нормального распределения

А затем обучаются «восстанавливать» информацию из этого шума
согласно прикрепленным
дополнительным данным

Данные могут быть текстовыми, позволяя получать txt2img – картинку по описанию или просьбе

Или графическими, заряжая img2img

Или и то, и другое!

ИНСТРУМЕНТАРИЙ



pytorch-lightning

Лучшее из pytorch теперь
автоматизировано без лишнего
кода

Полезные ресурсы

https://pytorch.org/serve/model_zoo.html

<https://paperswithcode.com/>

https://github.com/qubvel/segmentation_models.pytorch

<https://github.com/zhangming8/yolox-pytorch>

<https://github.com/WongKinYiu/yolov7>

СПАСИБО!