



УНИВЕРСИТЕТ ИТМО

Обработка и генерация изображений

Введение

Ефимова Валерия Александровна

vefimova@itmo.ru

05.10.2022

Ефимова Валерия

- Основное направление исследований – генерация изображений.
- Закончила аспирантуру в Университете ИТМО.
- Научный сотрудник лаборатории машинного обучения – более 6 лет.
- Руководитель отдела компьютерного зрения в Статанли Технолоджис – более 2 лет.
- Научный руководитель дипломов и практик.



- 12 лекций
- 5 лабораторных – 60 баллов
- Экзамен 40 баллов

Обработка изображений

- 
- 1. Компьютерное зрение и глубокие нейронные сети
 - 2. Наборы данных, задача классификации и известные архитектуры
 - Классификация (8 баллов)
 - 3. Задача детекции
 - Детекция (15 баллов)
 - 4. Сегментация. Обработка видео
 - Сегментация (15 баллов)
 - 5. Обработка изображений (Карты глубины. Поиск похожих изображений)

Генерация изображений

- 
- 6. Порождающие модели. Оценка качества генерации. VAE
 - 7. GAN, их проблемы, StyleGAN
 - 8. Условный GAN, нейронный перенос стиля
 - 9. Трансляция из изображения в изображение. Генерация изображения по сегментации
 - Генерация изображения по сегментации (15 баллов)
 - 10. Современные методы генерации изображений (авторегрессионные и диффузные)
 - 11. Генерация изображения по тексту
 - Генерация изображения по тексту (7 баллов)
 - 12. Генеративная обработка изображений



УНИВЕРСИТЕТ ИТМО

Основы компьютерного зрения

Компьютерное зрение и его задачи.

Классические подходы к компьютерному зрению

Ефимова Валерия Александровна

vefimova@itmo.ru

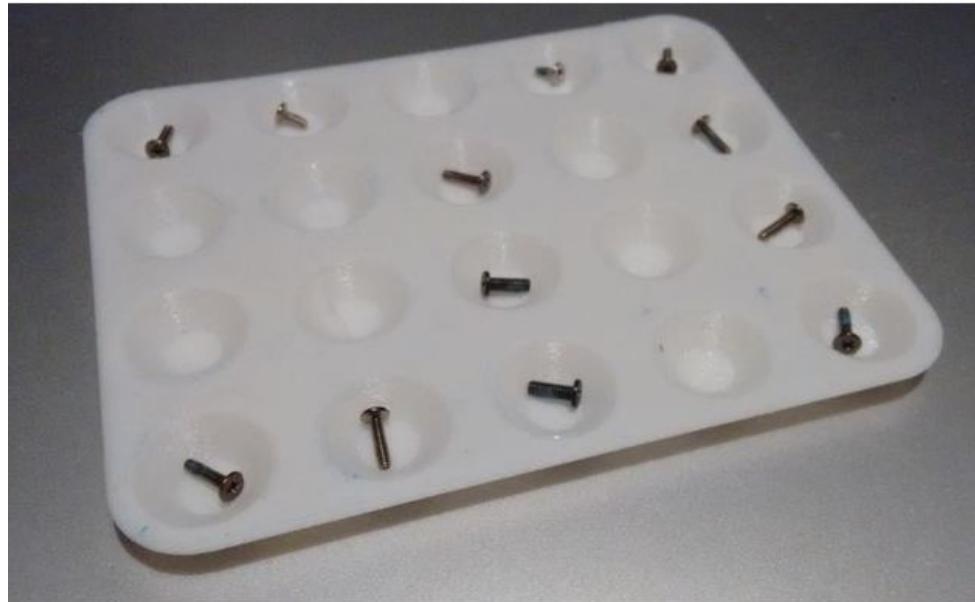
05.10.2022

ОБРАЗОВАТЕЛЬНЫЕ ПРОГРАММЫ В ОБЛАСТИ
ТЕХНОЛОГИЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

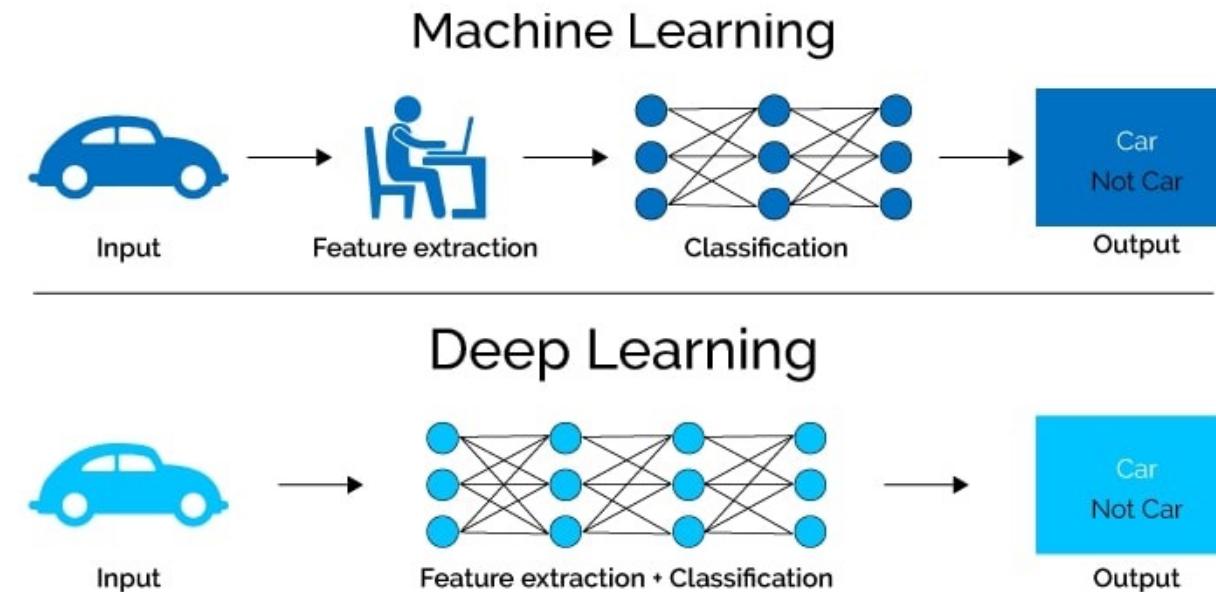
- Обзор компьютерного зрения
- Классические подходы к компьютерному зрению
- Глубокие нейронные сети
- Реализация и запуск глубоких нейронных сетей

Компьютерное зрение (Computer Vision, CV), в том числе машинное зрение (Machine Vision, MV) – это автоматическая фиксация и обработка изображений.

Представим рабочего на заводе, который ищет дефективные болты...

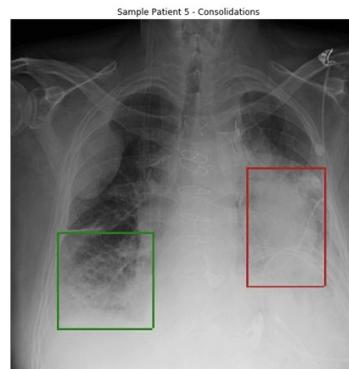
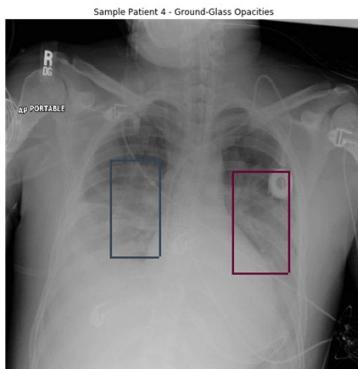
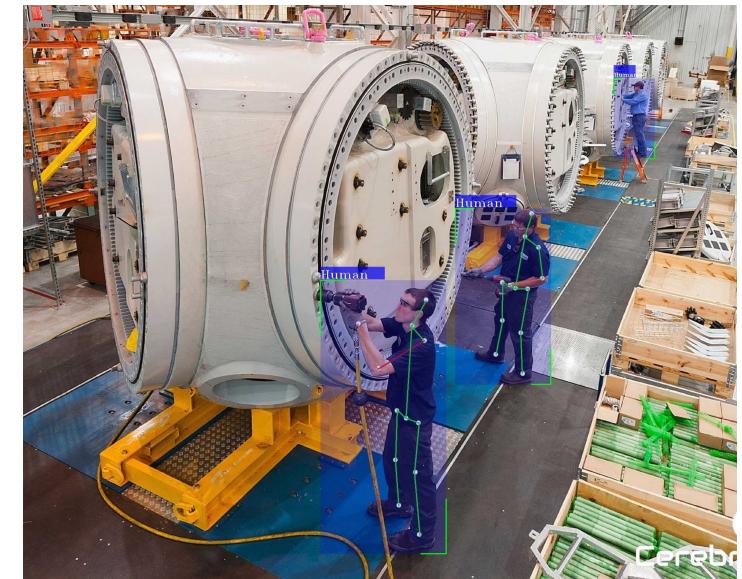


- Считается, что первая идея о том, чтобы научить компьютер видеть, возникла у профессора MIT (Массачусетский Институт Технологий) Лоуренса Робертса (Lawrence Roberts) в 1963 году.
- При первых попытках обработки изображений признаки конструировались вручную.
- Начиная с 2010-2011 годов глубокие нейронные сети в большинстве задач оставили позади классические методы обработки изображений.
- Глубокое обучение позволяет обрабатывать не сконструированный вектор чисел, а само изображение.



Сегодня компьютерное зрение широко применяется:

- Промышленность (автоматизация производственных процессов, их контроль, дефектоскопия).
- Ритейл (считывание информации о товаре, подсчет посетителей, определение внештатных ситуаций).
- Логистика (автоматизация погрузки, перемещение грузов, их сортировка).
- Медицина (диагностика, поиск лекарств).
- Робототехника (распознавание объектов).



Сегодня компьютерное зрение применяется:

- Интернет вещей (Internet of Things, IoT)
- Интеллектуальные транспортные системы ITS (Intelligent Transportation System),
- Автономные автомобили (Driverless Car) и системы помощи водителю ADAS (Advanced driver-assistance systems),
- Беспилотные летательные аппараты,
- Сельское хозяйство,
- Системы военного применения,
- Аддитивное производство (3D-printing) и многих других.



Viewpoint variation



Scale variation



Deformation



Occlusion



Illumination conditions



Background clutter



Intra-class variation



Изображение может быть:

- повернуто;
- изменено освещение;
- изменен размер/масштаб;
- деформировано;
- добавлен фон.

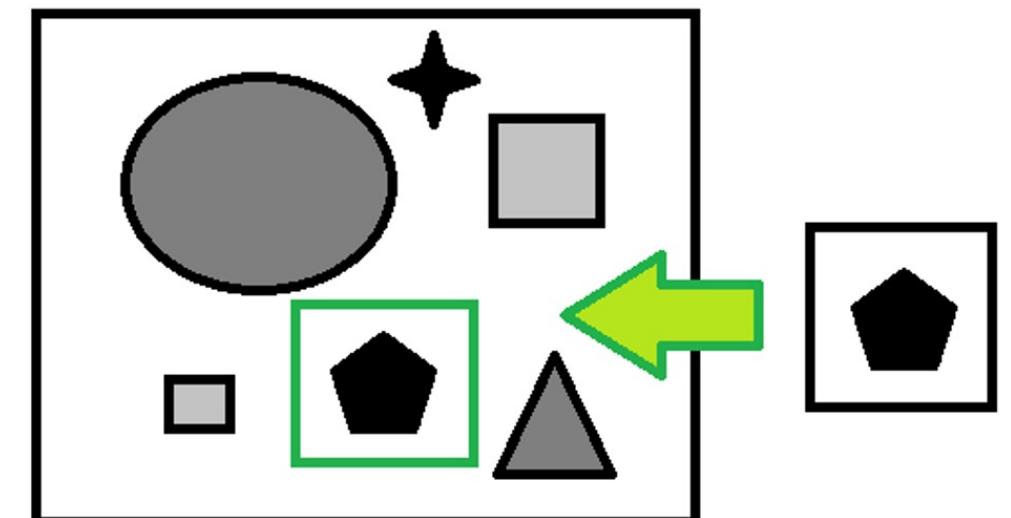
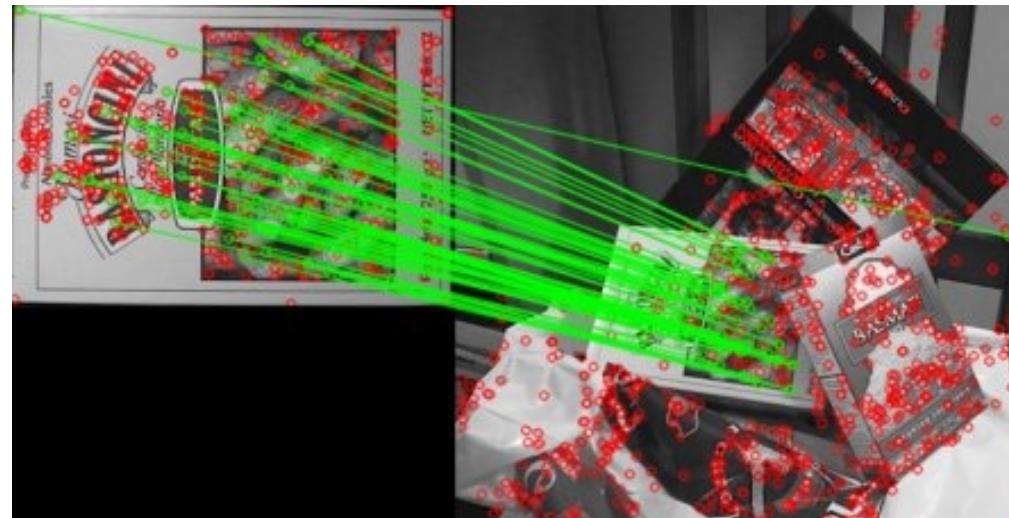
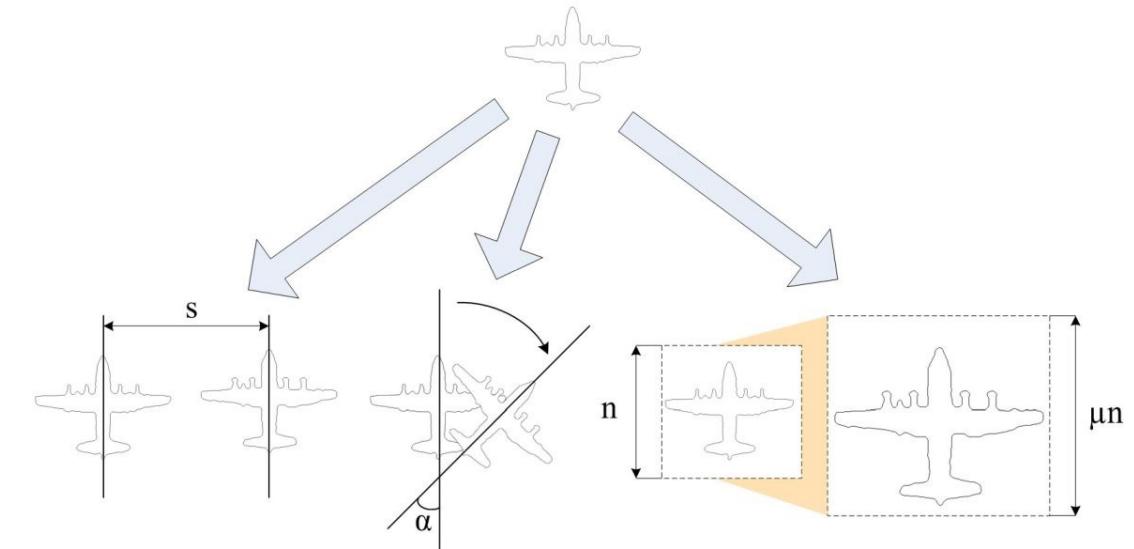
Объект на изображении может быть частично закрыт.

Объекты одного класса могут выглядеть совершенно по-разному.

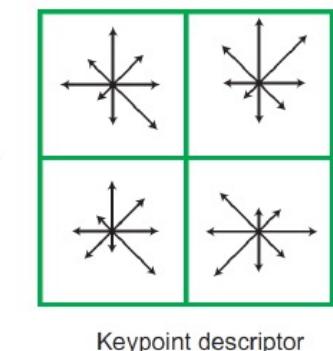
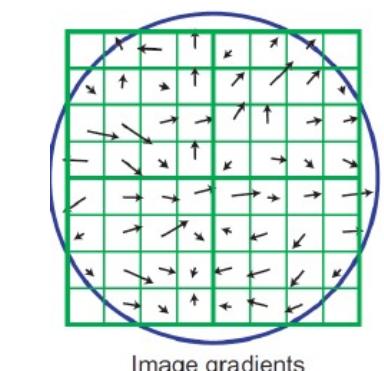
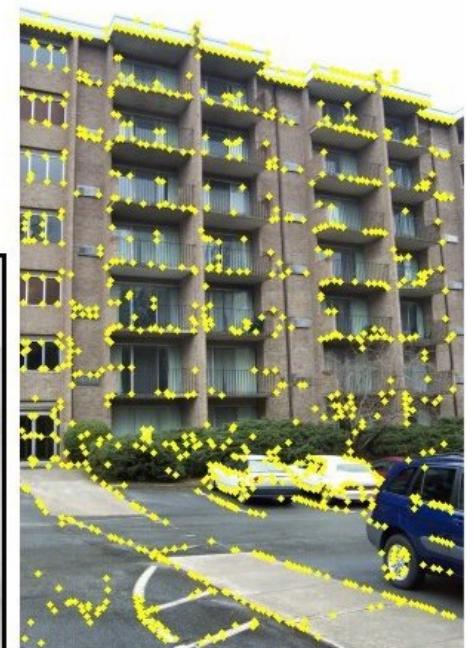
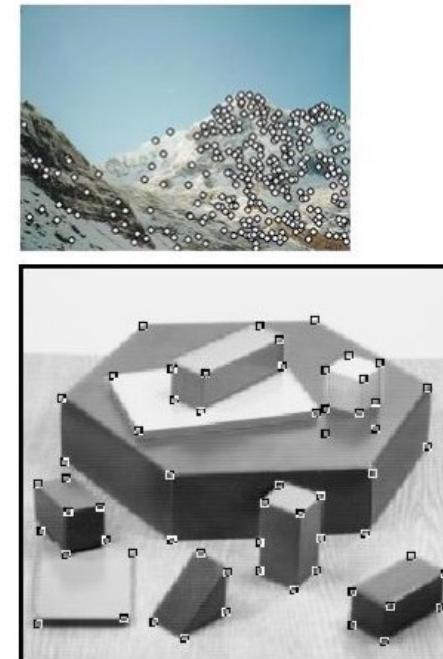
Объекты разных классов могут иметь еле заметные отличия.

Основные подходы к решению задач CV:

- Контурный анализ
- Поиск по шаблону (template matching)
- Поиск вне шаблонов, сопоставление по ключевым точкам (feature detection, description matching)
- Совмещение данных (Data Fusion)

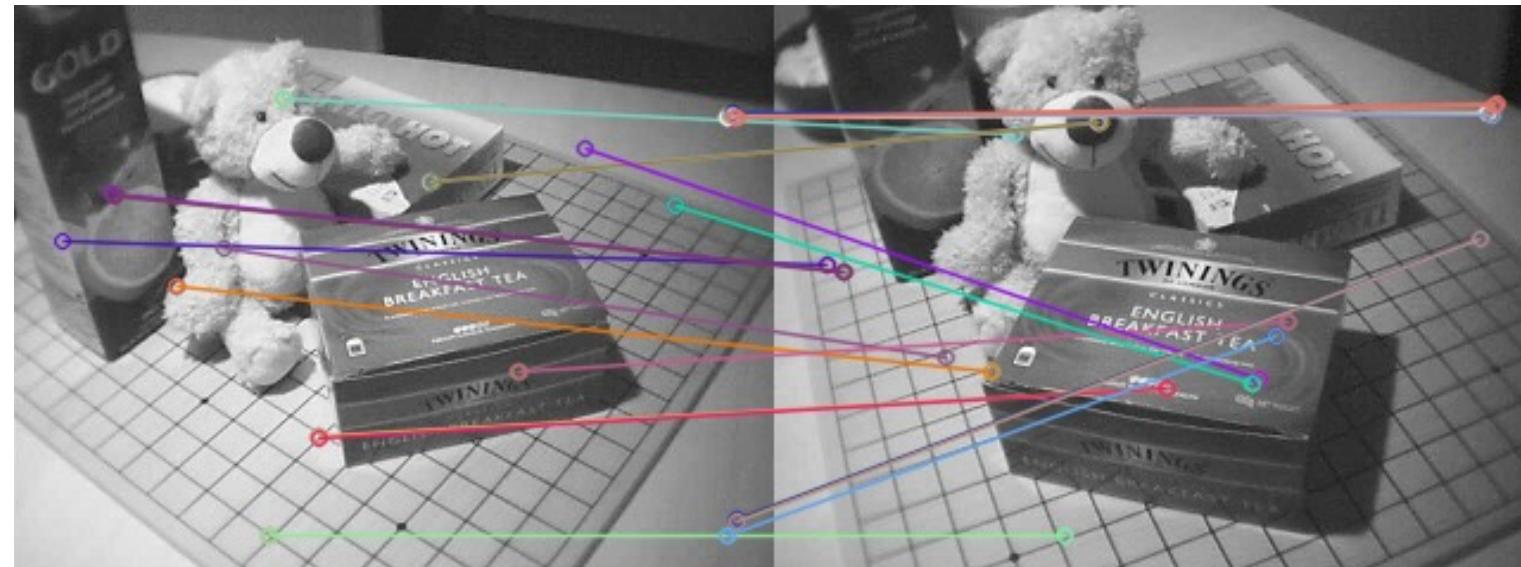


- Ключевые точки (keypoints) – это точки, по которым можно классифицировать изображение, распознать его, некая особенность изображения.
- Обладают следующими свойствами:
 - определенность (distinctness);
 - устойчивость (repeatability);
 - инвариантность (invariance);
 - стабильность (stability).
- Дескриптор – описание особой точки, определяющее особенности её окрестности, представляет собой вектор определенных параметров.
- Множество алгоритмов для нахождения ключевых точек и дескрипторов: SIFT, SURF, Brisk, ORB, BRIEF.



Для сопоставления дескрипторов: порог, ближайшие соседи, Nearest Neighbor Distance Ratio (NNDR).

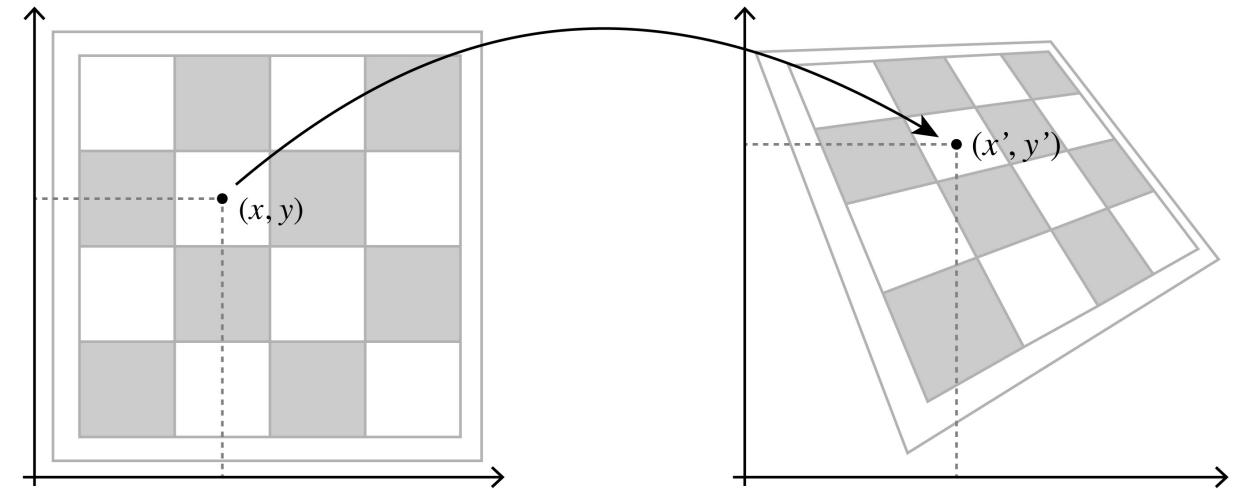
1. Посчитать евклидово расстояние между всеми парами ключевых точек.
2. Отбросить пары точек, находящихся дальше порога.
3. Отсортировать точки по уверенности (сравнить расстояния между первым и вторым ближайшим соседом).



Гомография – проективное преобразование плоскости.

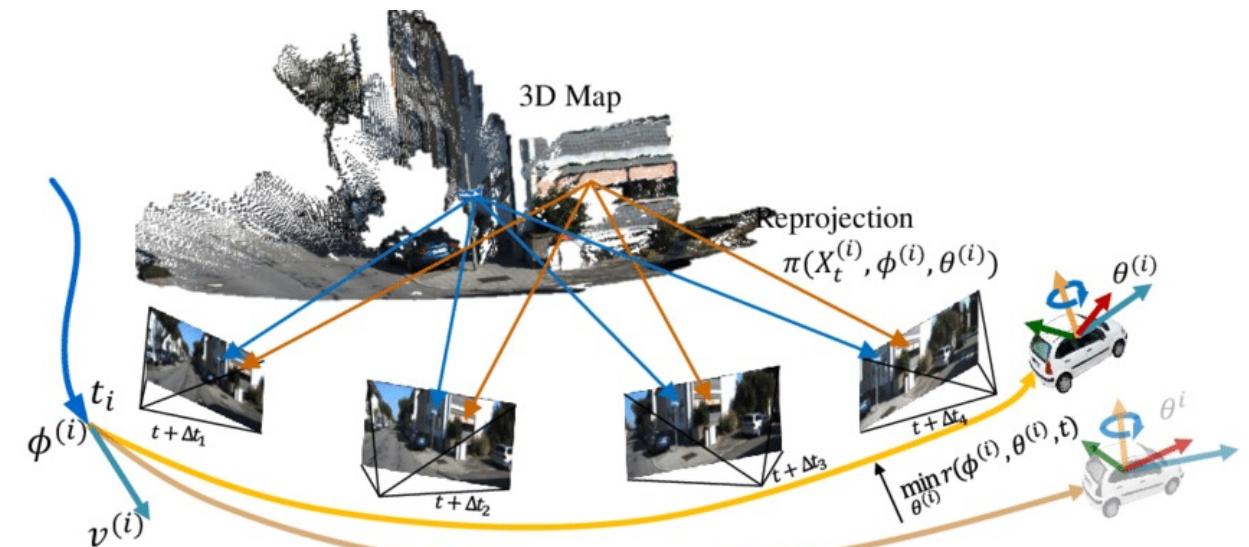
$$H = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}$$

$$x' = \frac{ax + by + c}{gx + hy + 1} \quad y' = \frac{dx + ey + f}{gx + hy + 1}$$



- a, e – масштабирование по x и y ;
- b, d – сдвиг по осям (вместе с a, e влияют на поворот);
- c, f – смещение по осям;
- g, h – изменение перспективы.
- Алгоритмы для вычисления: **RANSAC**, PROSAC, LMS.

- Визуальная одометрия (определение положения робота)
- Склейка карт и панорам
- Определение небольших изменений в кадрах видео
- Определение позы





УНИВЕРСИТЕТ ИТМО

Глубокие нейронные сети

Ефимова Валерия Александровна

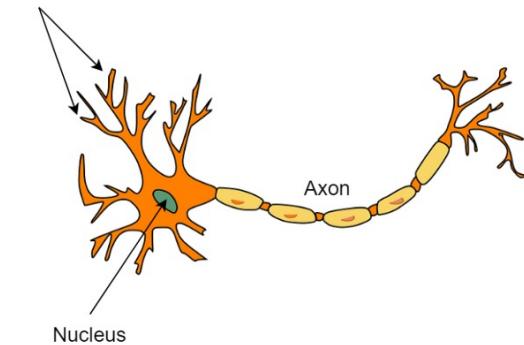
vefimova@itmo.ru

05.10.2022

ОБРАЗОВАТЕЛЬНЫЕ ПРОГРАММЫ В ОБЛАСТИ
ТЕХНОЛОГИЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

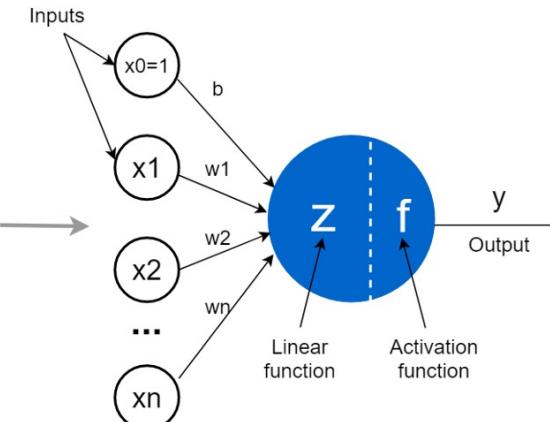
- В мозге много одинаковых нейронов (примерно 10^{11}), каждый из них получает и передает сигналы.
- Нейронные сети появились давно (современная модель нейрона предложена в 1943 году), но до 2009 года их не умели эффективно обучать.
- Один нейрон:

Dendrites



Nucleus

Axon



Inputs

$x_0=1$

x_1

x_2

...

x_n

w_1

w_2

w_n

b

x_n

w_n

- Сигмоида не сохраняет 0, но ее просто вычислять.
- Функция ReLU затирает производную на половине своей области определения, поэтому используют ReLU с протечкой (leaky ReLU), она очень популярна в компьютерном зрении.

Шумный (noisy) ReLU:

$$h(x) = \max(0, x + \epsilon), \epsilon \sim N(0, \sigma(x))$$

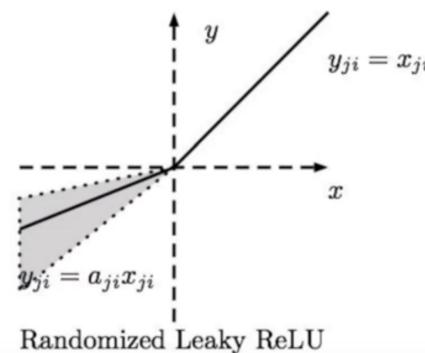
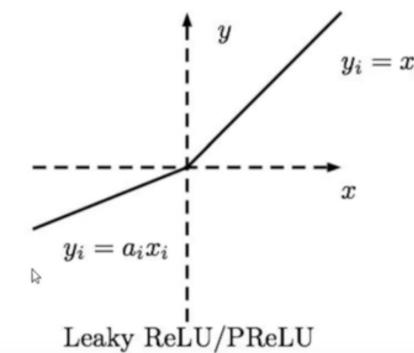
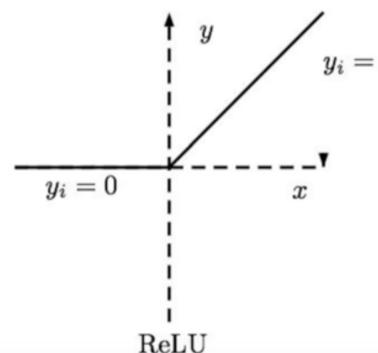
ReLU с утечкой (leaky ReLU):

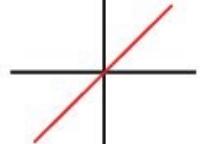
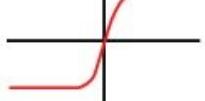
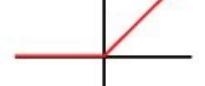
$$h(x) = \begin{cases} x & \text{если } x > 0, \\ 0,01x & \text{иначе} \end{cases}$$

Параметрический ReLU

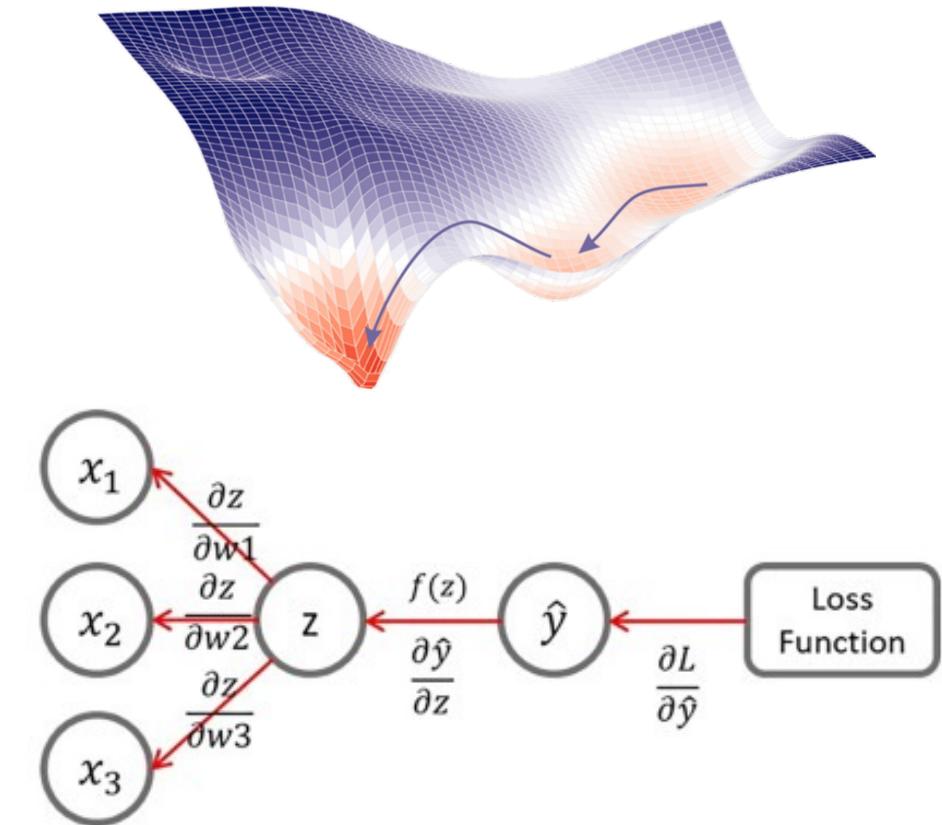
(параметр β настраиваемый):

$$h(x) = \begin{cases} x & \text{если } x > 0, \\ \beta x & \text{иначе} \end{cases}$$

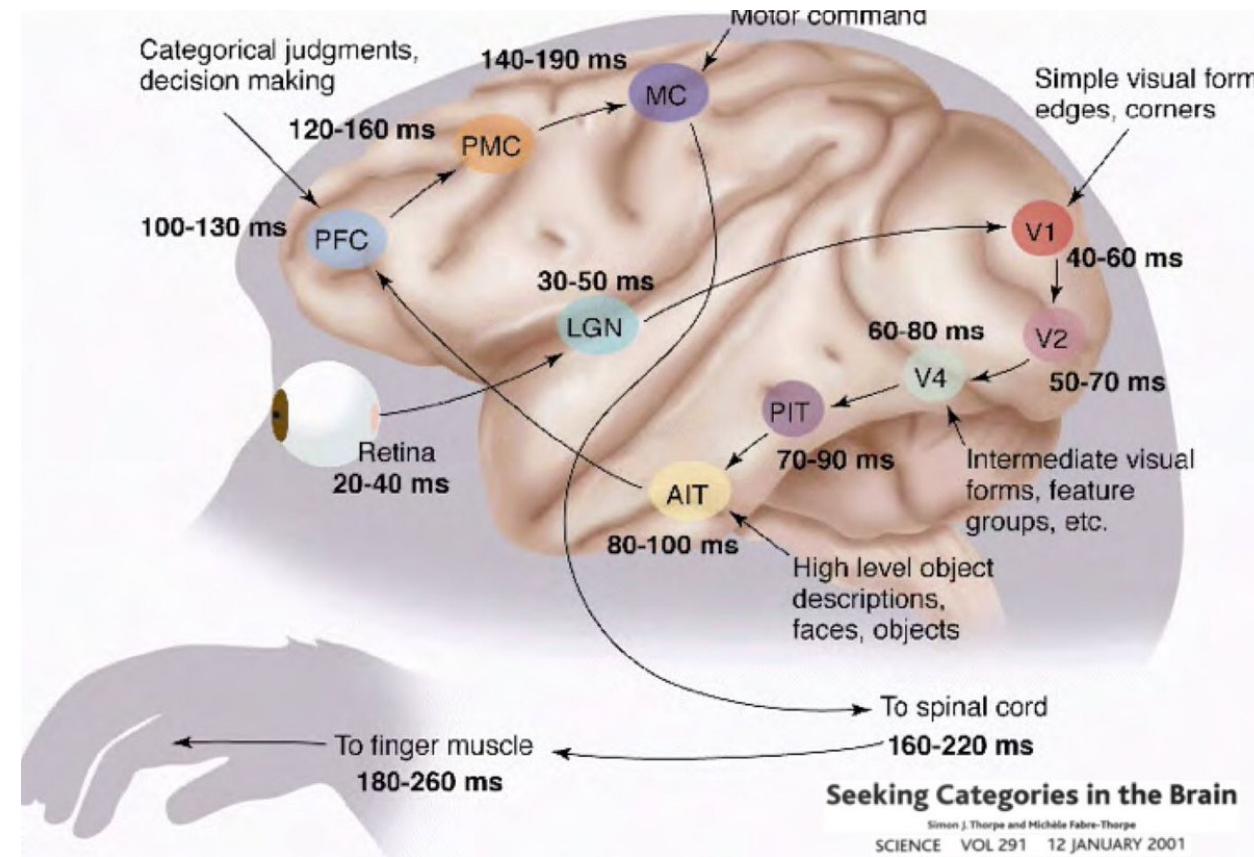


Activation function	Mathematical representation	Figure
Identity	$f(x) = x$	
Binary step	$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	
Sigmoid	$f(x) = \frac{1}{1 + e^{-x}}$	
TanH	$f(x) = \frac{2}{1 + e^{-2x}} - 1$	
ReLU	$f(x) = \begin{cases} 0, & x < 0 \\ 1, & x \geq 0 \end{cases}$	

- Градиентный спуск – способ оптимизации очень сложных функций (таких, как нейронные сети).
- Нейронная сеть задает граф вычислений.
- Обратное распространение ошибки (backpropagation) – алгоритм подсчета производных в больших композициях функций.
- Для подсчета градиента нужно посчитать частные производные по каждому входу в каждом промежуточном узле.
- Существует много видов улучшенного градиентного спуска: Momentum, метод Нестерова, Adagrad, RMSProp, Adadelta, Adam (Adaptive Model Estimation).

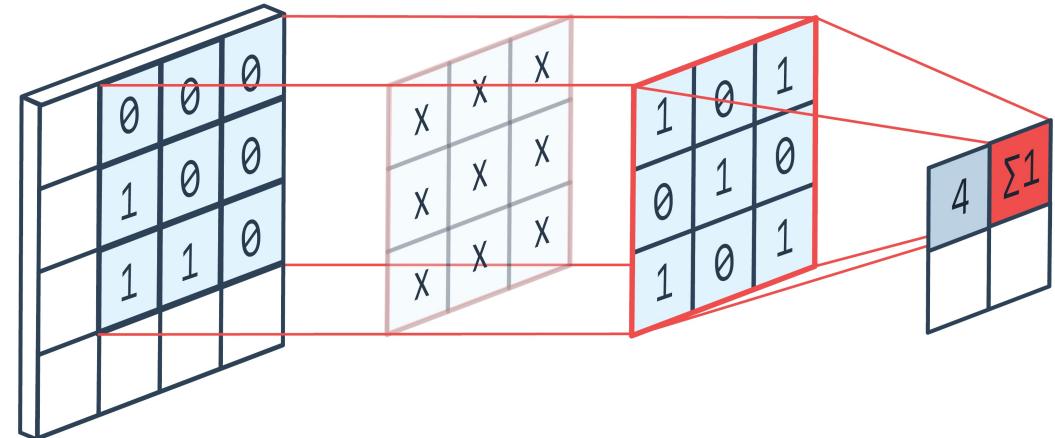
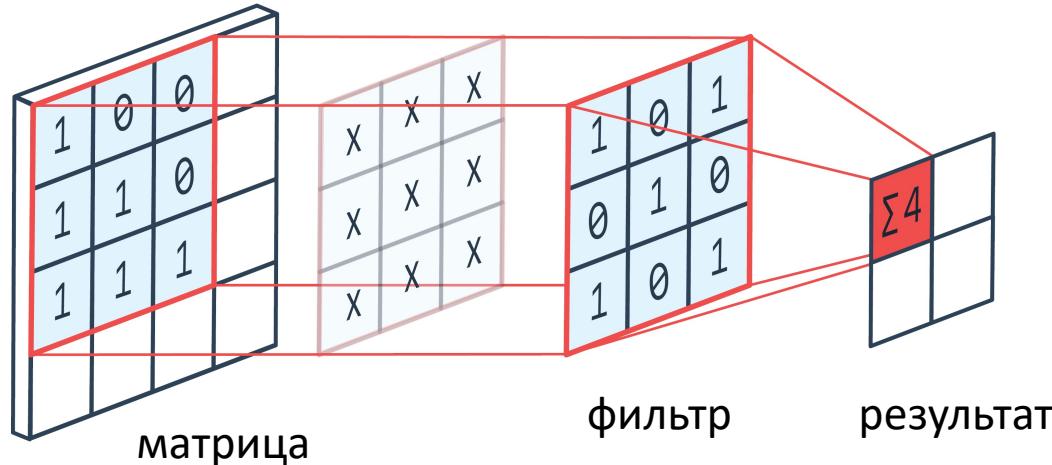


- V1 – локальные признаки, топографические карты;
- V2 – больше локальных признаков, бинокулярное зрение;
- V3 – цвет, текстура, первые результаты сегментации;
- V4 – геометрические формы, силуэты; это самый важный уровень для внимания;
- V5 – распознаёт движения объектов, сегментированных на V4;
- V6 – обобщает данные со всей картинки, wide-field stimulation;
- V7 – распознаёт сложные объекты, например человеческие лица.



Уровней немного!
Нет длинной
цепочки
вычислений.

Мозг хорошо умеет обучаться на очень маленьких выборках.



Операция свёртки: $X * F$.

Свёртки: 1D, 2D, 3D.

Свойства свёрток:

- Ассоциативность
- Коммутативность
- Линейность

- Определение линий (вертикальные, горизонтальные, наклонные $\pm 45^\circ$)

$$\begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix}, \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix}, \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix}, \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix}$$



- Определение границ (оператор Собеля)

$$G_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & 2 \\ 1 & 0 & -1 \end{bmatrix} \text{ — горизонтальная компонента}$$



$$G_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & -1 \end{bmatrix} \text{ — вертикальная компонента}$$

- Сглаживающий фильтр (размытие)

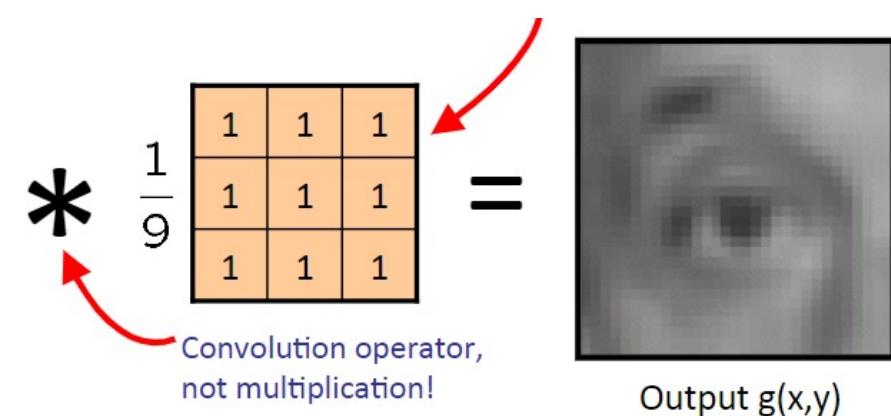
https://en.wikipedia.org/wiki/Sobel_operator

<https://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm>

<https://homepages.inf.ed.ac.uk/rbf/HIPR2/linedet.htm>

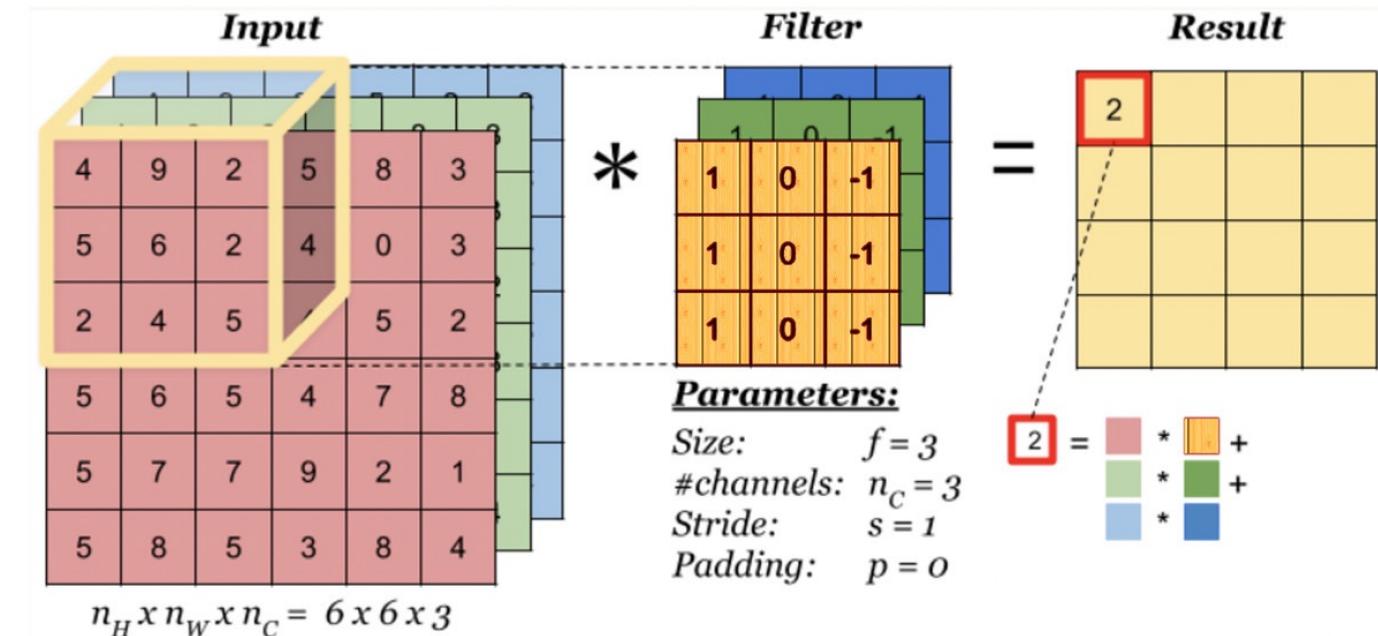


Input $f(x,y)$

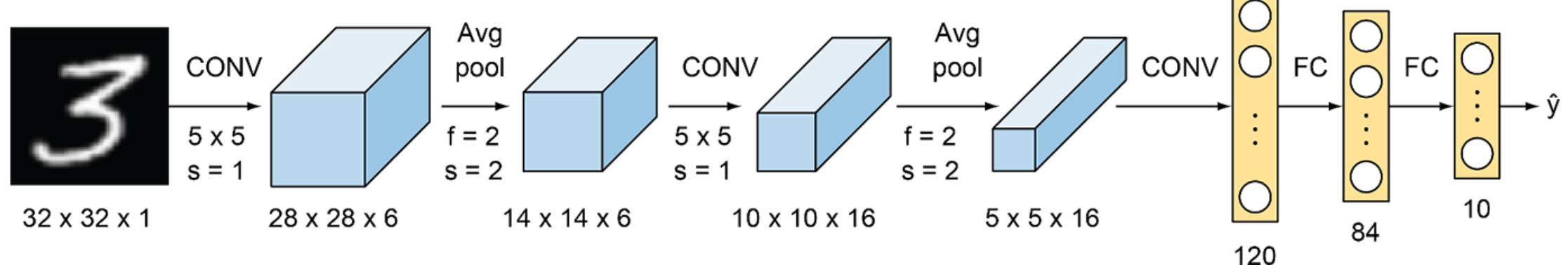


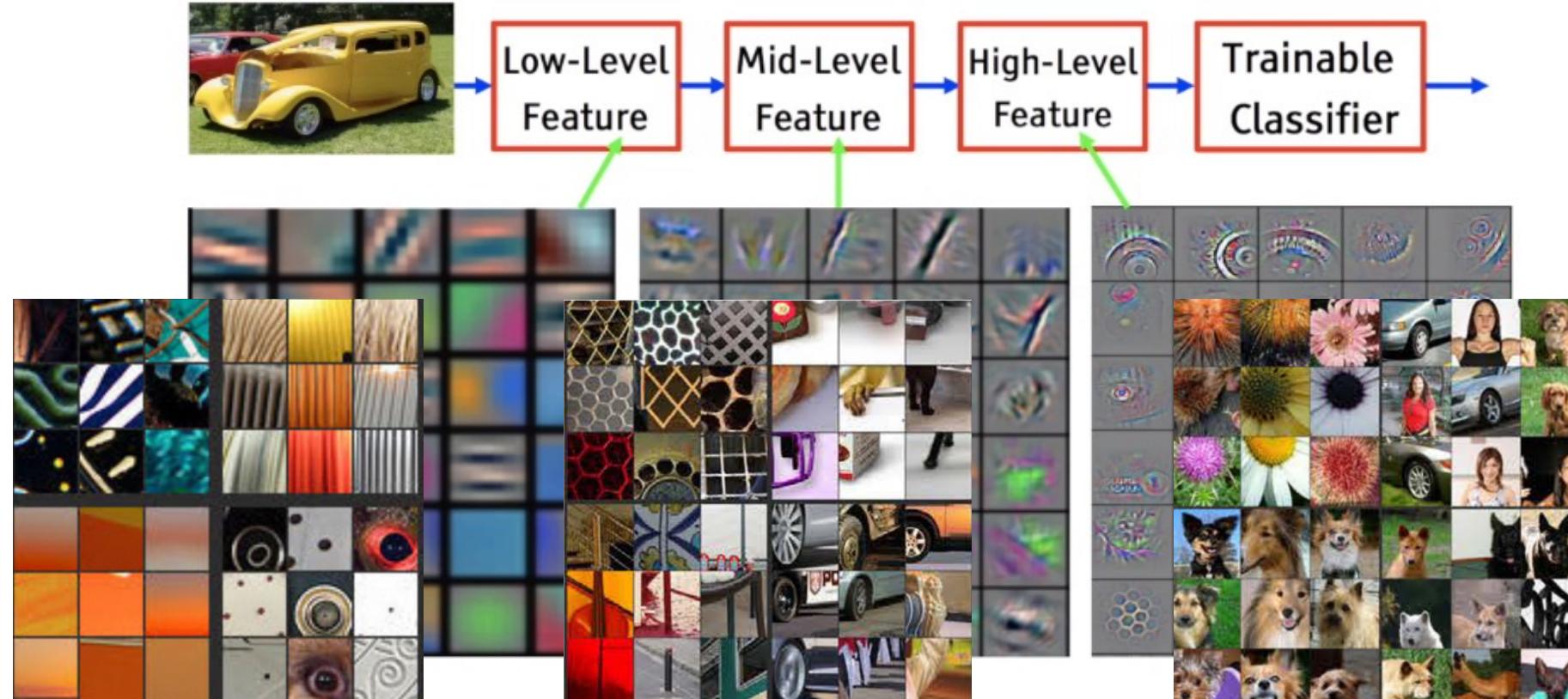
Output $g(x,y)$

Цветное изображение как правило кодируется трёхмерной матрицей (тензором) с размерностью: ширина, высота и глубина (w, h, c).



Пример свёрточной сети:





- При обычной свёртке в центре изображения многие регионы перекрываются, но пиксели на краях используются значительно реже.
- Дополнение изображения по краям разными способами:

Нулевой сдвиг

0	0	A	B	C	0	0
---	---	---	---	---	---	---

Расширение границ

A	A	A	B	C	C	C
---	---	---	---	---	---	---

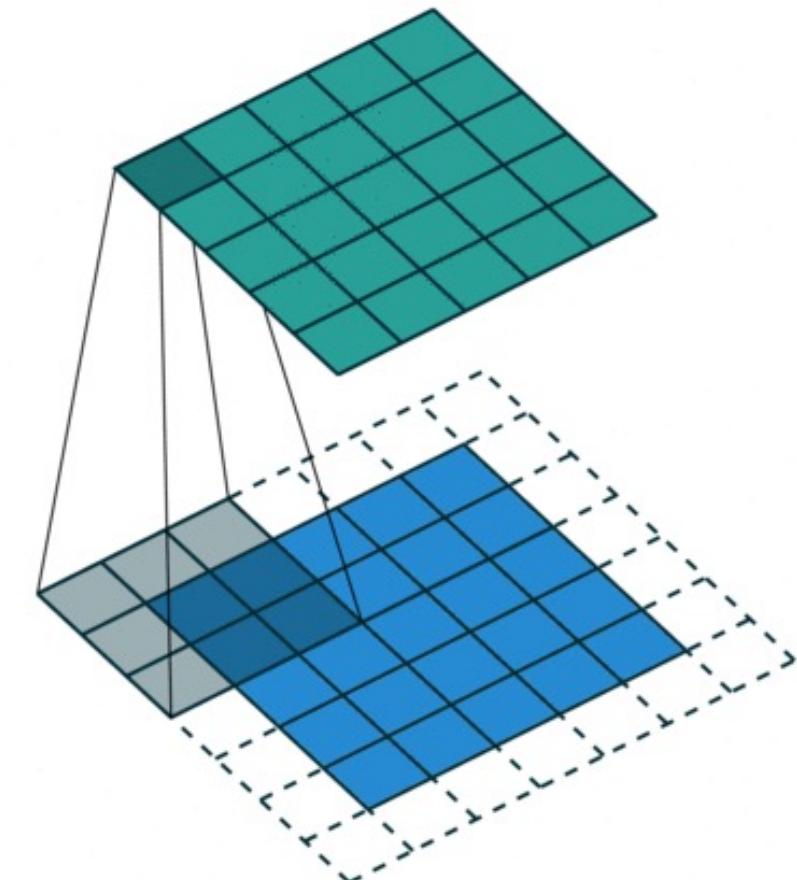
Зеркальный сдвиг

B	A	A	B	C	C	B
---	---	---	---	---	---	---

C	B	A	B	C	B	A
---	---	---	---	---	---	---

Циклический сдвиг

B	C	A	B	C	A	B
---	---	---	---	---	---	---



Сдвигаем фильтр перед умножением.

Input

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

Dimension: 6×6

Filter

$$\begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix}$$

*

Result

$$\begin{matrix} 2 & 1 \\ \hline & \end{matrix}$$

=

Parameters:
 Size: $f = 3$
Stride: $s = 2$
Padding: $p = 0$

$$1 = 2*1 + 5*0 + 3*(-1) + \\ 2*1 + 4*0 + 3*(-1) + \\ 5*1 + 4*0 + 2*(-1)$$

<https://indoml.com>

O – размер (ширина) выходного изображения.

I – размер (ширина) входного изображения.

K – размер (ширина) ядер, используемых в свёрточном слое.

S – сдвиг (stride).

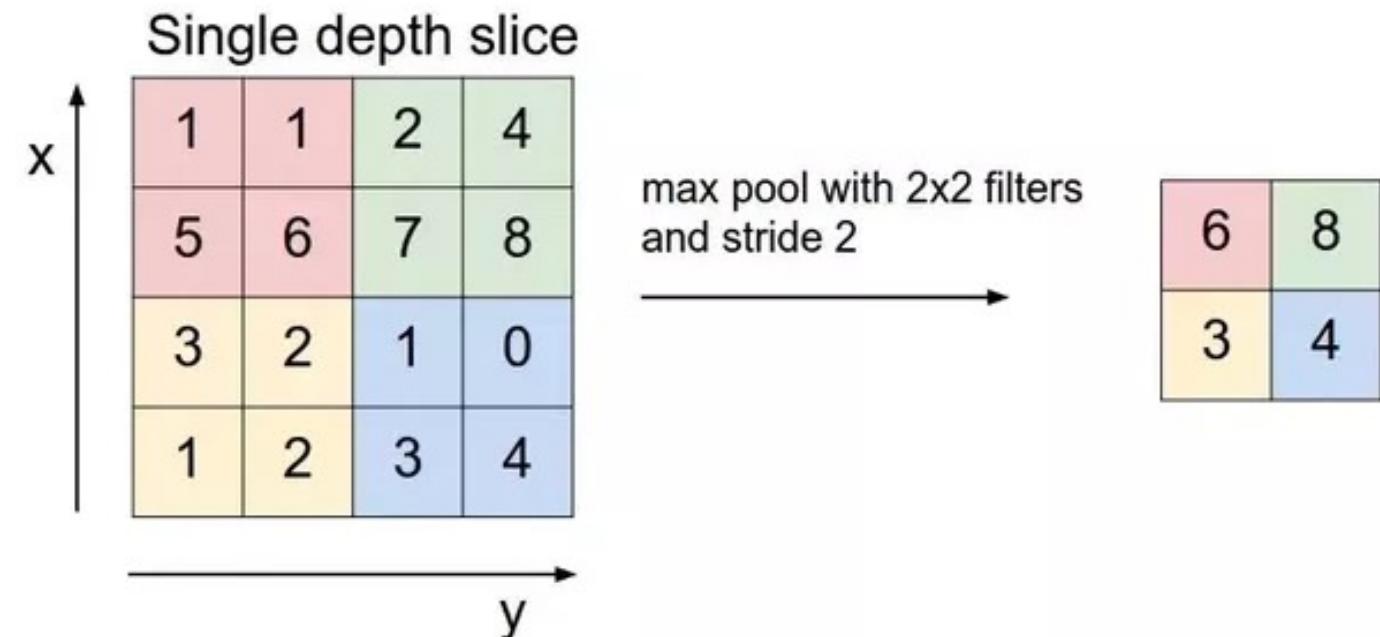
P – число элементов заполнения (padding).

$$O = \frac{I - K + 2P}{S} + 1$$

- Снижение размеров изображения (и увеличение скорости вычислений).
- Экспериментально хорошо работает.
- Нет обучаемых параметров.

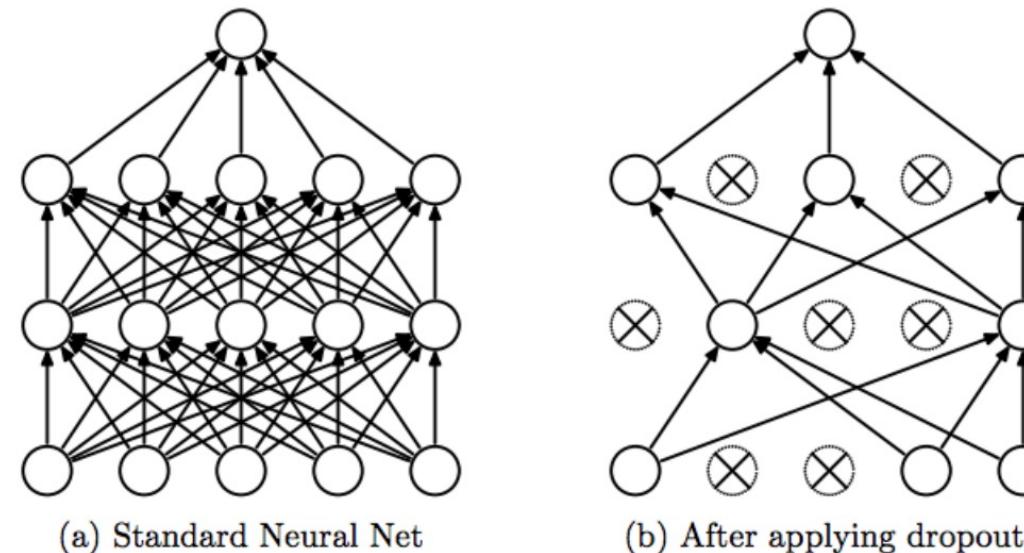
Пулинг:

- максимум (max pooling);
- среднее (avg pooling).



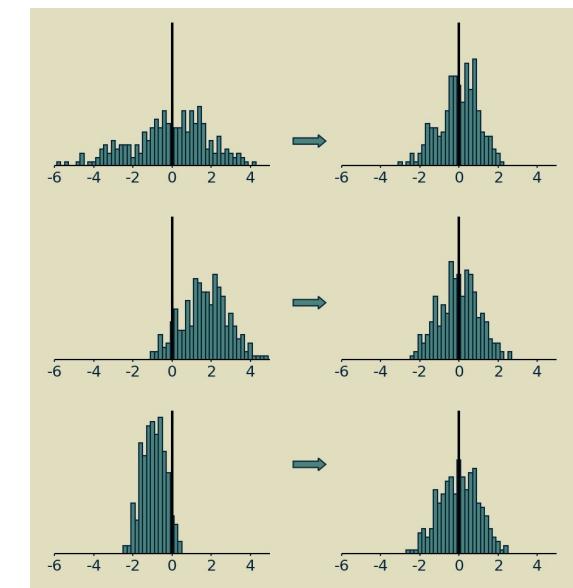
- **Локальность восприятия:** каждое преобразование действует на небольшую часть изображения. Считаем, что признаки в соседних ячейках матрицы зависимы.
- **Общие параметры:** используем небольшие и одинаковые наборы фильтров для всех пикселей (мало настраиваемых параметров).
- **Уменьшение размерности:** пулинг и subsampling.
- **Разреженность связей:** выходное значение зависит от малого числа входов.

- На разных шагах тренировки выкидываем случайные нейроны с каждого слоя.
- На каждом шаге обучаются немногие разные нейросети.
- Позволяет сети обучиться на выделение более информативных признаков.
- Как бы усредняет 2^N сетей, что улучшает результат.

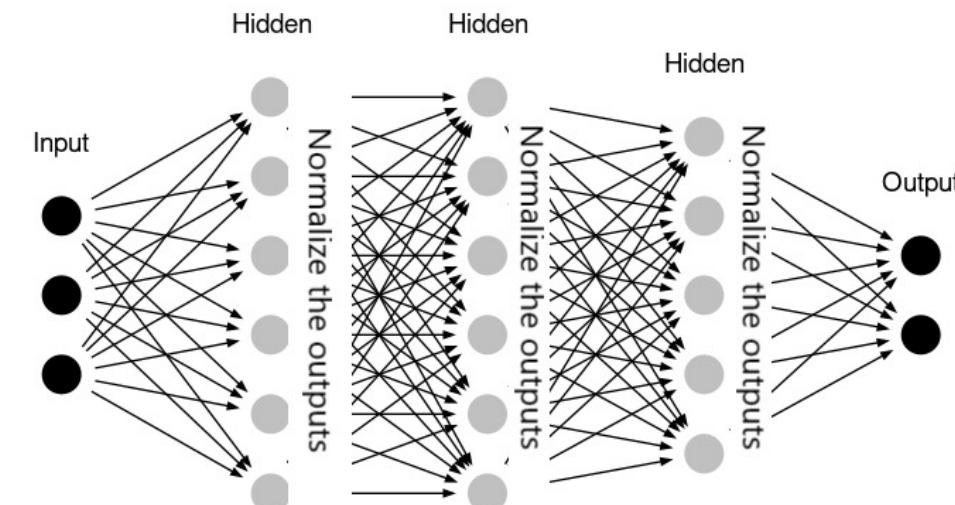


Пакетная нормализация (Batch Normalization)

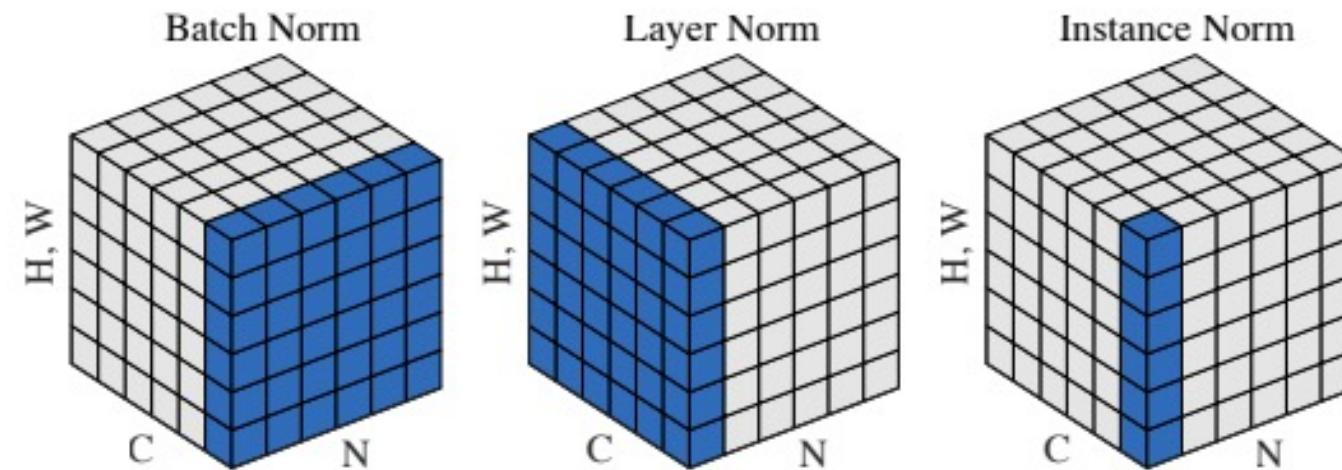
- Пакет (батч, batch) – небольшая порция данных, в глубоком обучении его размер выбирают как степень 2, но так, чтобы график вычислений помещался в память GPU.
- На практике используют стохастический градиентный спуск, который обновляет веса после обработки каждого батча.
- Пакетная нормализация – обработка данных, чтобы они имели нулевое математическое ожидание и единичную дисперсию. Позволяет повысить производительность и стабилизировать работу NN.



$$\begin{aligned}\mu_B &\leftarrow \frac{1}{m} \sum_{i=1}^m x_i && // \text{mini-batch mean} \\ \sigma_B^2 &\leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_B)^2 && // \text{mini-batch variance} \\ \hat{x}_i &\leftarrow \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} && // \text{normalize} \\ y_i &\leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) && // \text{scale and shift}\end{aligned}$$

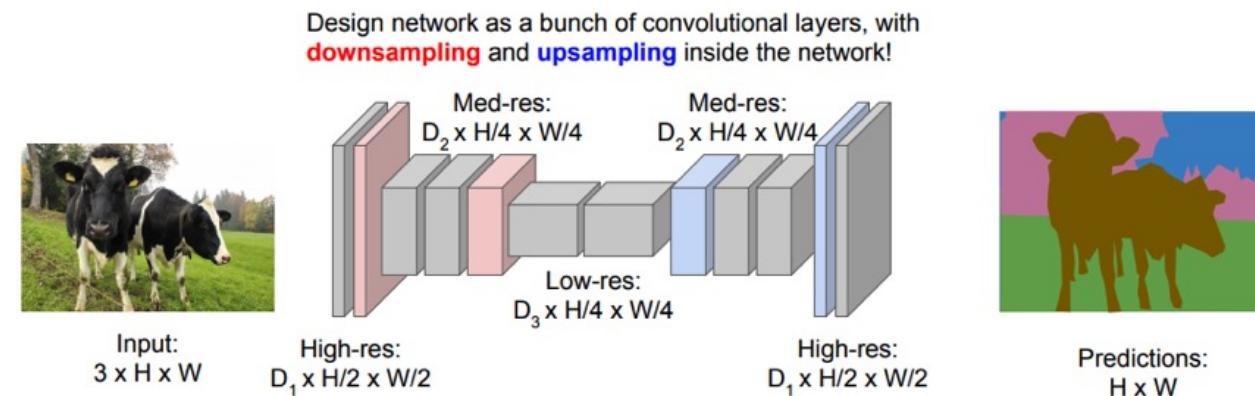


- При усреднении по батчу теряются индивидуальные характеристики изображений.
- Уровневая нормализация (Layer normalization) – для RNN.
- Индивидуальная нормализация (Instance normalization) – нормализация происходит по каждому отдельному объекту, а не по всему пакету.



1. Ba J. L., Kiros J. R., Hinton G. E. Layer normalization //arXiv preprint arXiv:1607.06450. – 2016.

2. Nam H., Kim H. E. Batch-instance normalization for adaptively style-invariant neural networks //Advances in Neural Information Processing Systems. – 2018. – Т. 31.



1. Ближайшие соседи (не обучается)

Nearest Neighbor

1	2
1	2
3	4



1	1	2	2
1	1	2	2
3	3	4	4
3	3	4	4

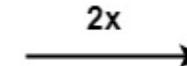
Input: 2×2

Output: 4×4

2. Би-линейная интерполяция (не обучается)

10	20
30	40

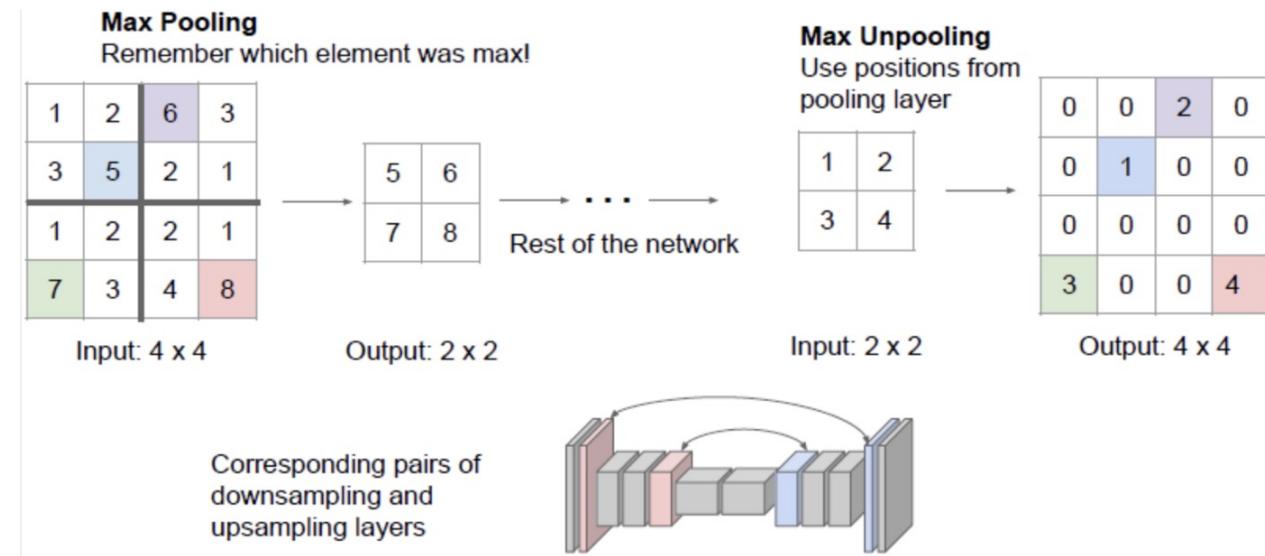
2x2



10	12	17	20
15	17	22	25
25	27	32	35
30	32	37	40

4x4

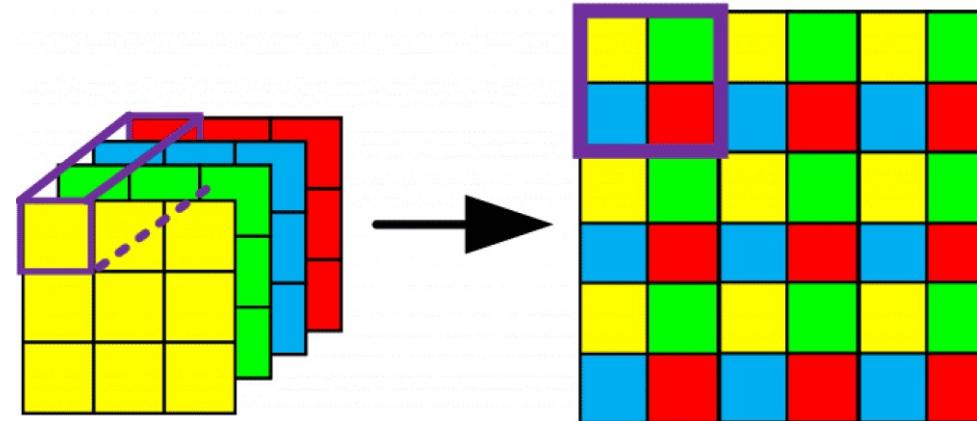
3. Max Unpooling (не обучается)



4. Transposed convolution

5. Pixel shuffle

Объединение четырех слоев размером $n \times n$ в слой размером $2n \times 2n$.



Обратная свёртка (transposed convolution/deconvolution)

Input	Kernel					Output																																															
<table border="1"><tr><td>0</td><td>1</td></tr><tr><td>2</td><td>3</td></tr></table>	0	1	2	3	<table border="1"><tr><td>0</td><td>1</td></tr><tr><td>2</td><td>3</td></tr></table>	0	1	2	3	=	<table border="1"><tr><td>0</td><td>0</td><td></td></tr><tr><td>0</td><td>0</td><td></td></tr><tr><td></td><td></td><td></td></tr></table>	0	0		0	0					+	<table border="1"><tr><td>0</td><td>1</td></tr><tr><td>2</td><td>3</td></tr></table>	0	1	2	3	+	<table border="1"><tr><td>0</td><td>2</td><td></td></tr><tr><td>4</td><td>6</td><td></td></tr></table>	0	2		4	6		+	<table border="1"><tr><td>0</td><td>3</td><td></td></tr><tr><td>6</td><td>9</td><td></td></tr></table>	0	3		6	9		=	<table border="1"><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>4</td><td>6</td></tr><tr><td>4</td><td>12</td><td>9</td></tr></table>	0	0	1	0	4	6	4	12	9
0	1																																																				
2	3																																																				
0	1																																																				
2	3																																																				
0	0																																																				
0	0																																																				
0	1																																																				
2	3																																																				
0	2																																																				
4	6																																																				
0	3																																																				
6	9																																																				
0	0	1																																																			
0	4	6																																																			
4	12	9																																																			

Возникает эффект сетки



Ground Truth



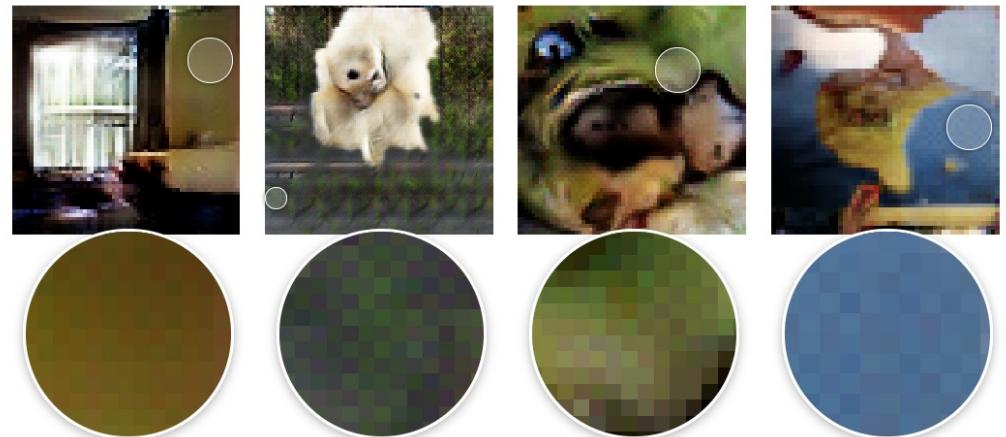
1/4 Sized
Input



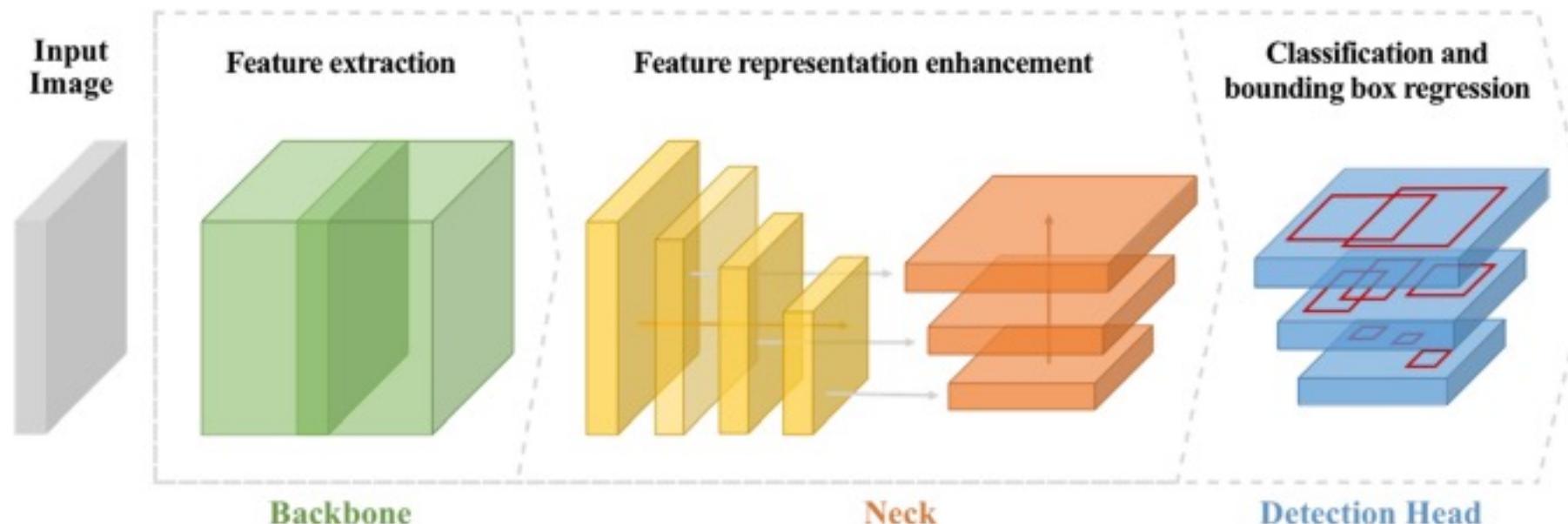
Bicubic



Super Resolution
Network



- Backbone – обычно одна из классических сетей без конечных слоев, для извлечения признаков изображения.
- Шея (neck) – обработка признаков.
- Голова (head) – получение результата для конкретной задачи.

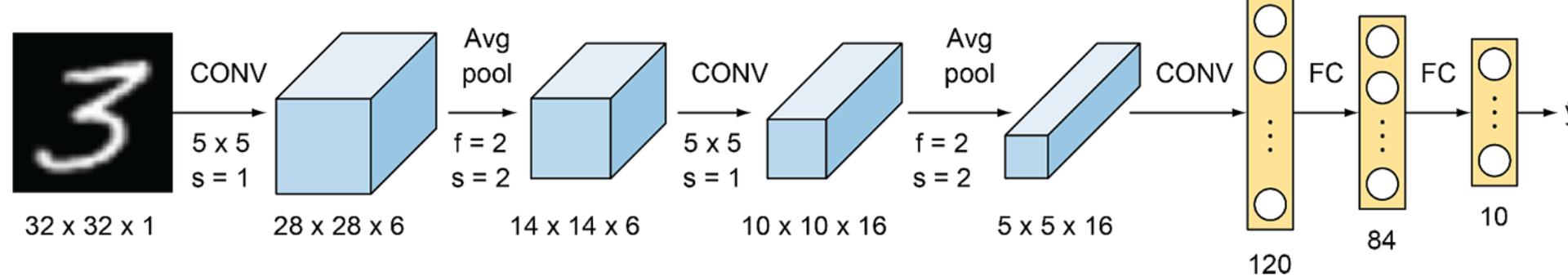
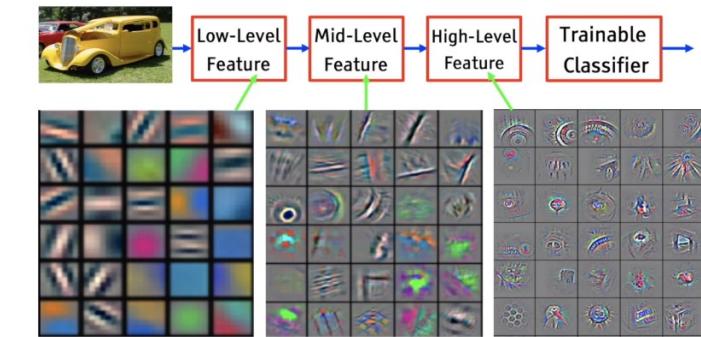


Python	PyTorch и PyTorch Lightning	 PyTorch  Lightning ^{AI}
	Tensorflow 1 и 2	 TensorFlow
	Keras	 Keras
C++	Caffe	 Caffe
Java	Deeplearning4j	 DL4J

1. Google Colaboratory – запуск кода на облачных серверах Google (в том числе на GPU).
 1. ~12 часов на запуск модели.
 2. Отключение после 30 минут бездействия.
2. Kaggle Kernels
3. Yandex DataSphere
4. Azure Notebooks



- Компьютерное зрение позволяет многое автоматизировать.
- Компьютерное зрение появилось раньше нейронных сетей, классические подходы основаны на других принципах.
- Для решения задач компьютерного зрения используют глубокие нейронные сети со сверточными слоями, они анализируют изображение от мелких деталей к крупным.
- Существует несколько фреймворков для глубокого обучения, в этом курсе рассмотрим PyTorch.
- Удобно запускать глубокие модели в Google Colaboratory (колаб).



1. Shinde P. P., Shah S. A review of machine learning and deep learning applications //2018 Fourth international conference on computing communication control and automation (ICCUBEA). – IEEE, 2018. – C. 1-6.
2. Paul S. K. et al. Performance analysis of keypoint detectors and binary descriptors under varying degrees of photometric and geometric transformations //arXiv preprint arXiv:2012.04135. – 2020.
3. Ioffe S., Szegedy C. Batch normalization: Accelerating deep network training by reducing internal covariate shift //International conference on machine learning. – PMLR, 2015. – C. 448-456.
4. Zeiler M. D., Fergus R. Visualizing and understanding convolutional networks //European conference on computer vision. – Springer, Cham, 2014. – C. 818-833.
5. Глубокое обучение — Николенко С., Кадурин А., Архангельская Е., 2018.

Полезные лекции:

1. Курс Николенко "Введение в коммуникационную сложность" <https://www.youtube.com/watch?v=-s3kucJlawo&list=PL- cKNuVAYAUhvlUfW7P2cdhWCRDWs0pG>
2. Специализация Coursera: Deep Learning <https://www.coursera.org/specializations/deep-learning?=>
3. Специализация Coursera: Generative Adversarial Networks (GANs) <https://www.coursera.org/specializations/generative-adversarial-networks-gans?=>
4. Серия лекций Samsung Innovation Campus - AI Lectorium https://www.youtube.com/watch?v=RS_U6qodpsc&list=PLJEYfuHbcEIB-DdeoWaQ6Bzt0903kbmWK

Колаб

https://colab.research.google.com/drive/1mmrpvoxhYYDuvKjihZhR_t0VjjQKlam?usp=sharing



УНИВЕРСИТЕТ ИТМО

Спасибо за внимание!

ОБРАЗОВАТЕЛЬНЫЕ ПРОГРАММЫ В ОБЛАСТИ
ТЕХНОЛОГИЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА