

# Лекция 2

## Сверточные нейронные сети

Разработка нейросетевых систем

Канев Антон Игоревич

# Cifar100

- Набор данных, состоящий из цветных изображений 100 классов
- Размер 32 на 32 пикселя
- 3 цвета

**airplane**



**automobile**



**bird**



**cat**



**deer**



**dog**



**frog**



**horse**



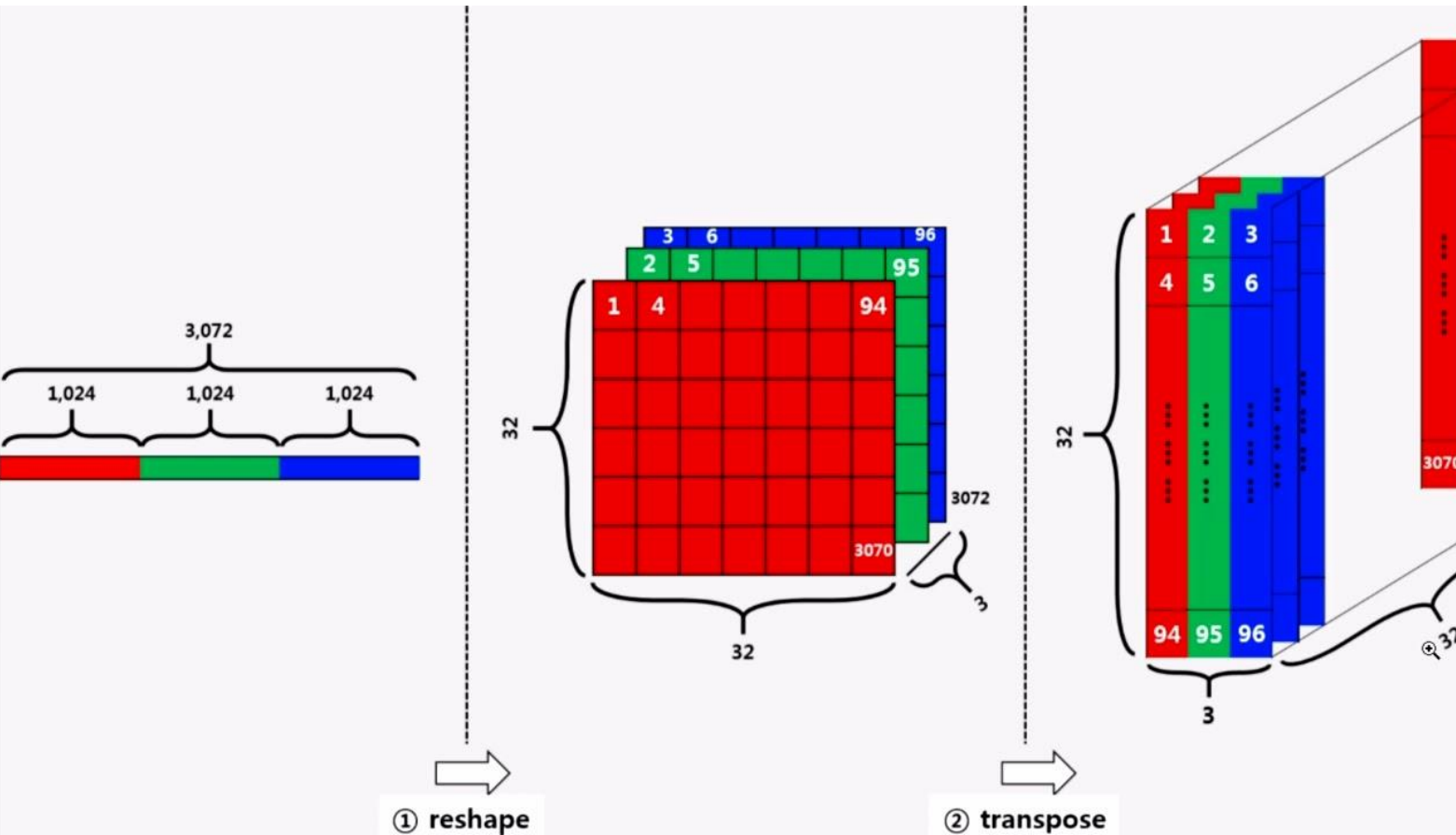
**ship**



**truck**



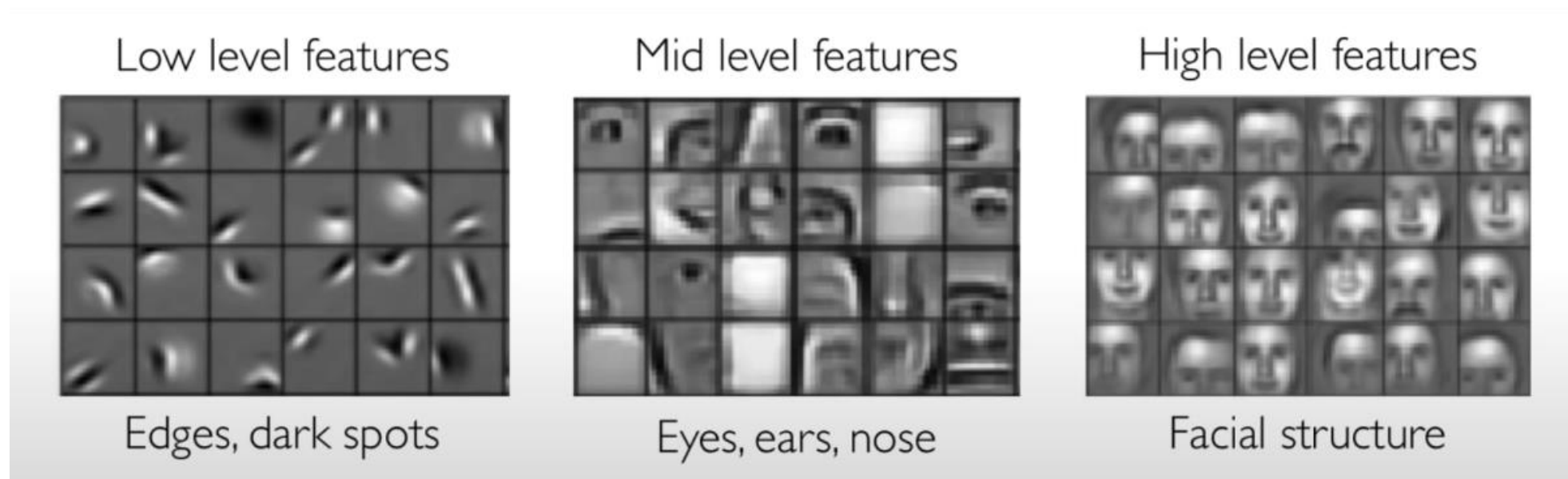
# Reshape, transpose



- Reshape – изменение размерности матрицы
- Transpose – транспонирование
- Количество элементов в матрице остается прежним - 3072

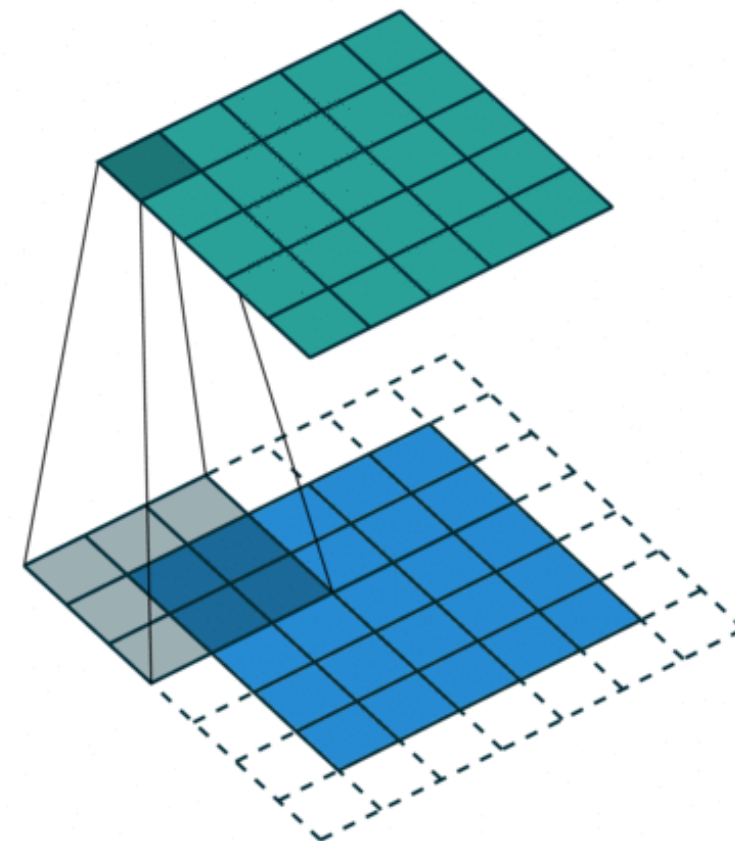
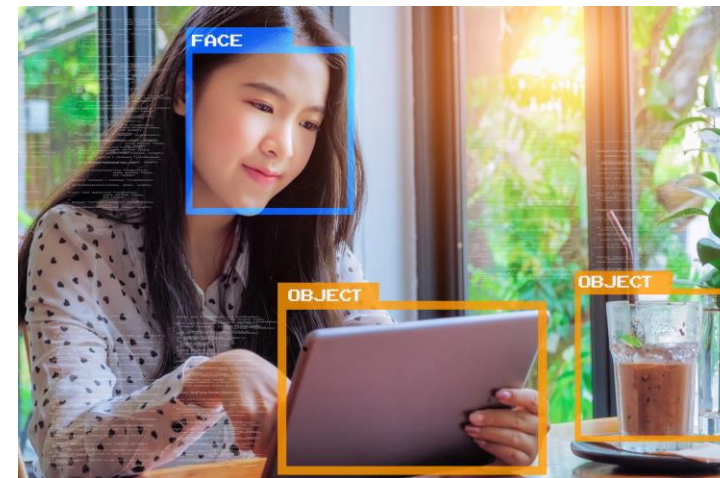
# Из биологии

- Идея сверточных нейронных сетей навеяна нам из реального мира
- Зрение человека ищет нужный набор признаков вокруг, воспринимает их и из простых признаков собирает сложные



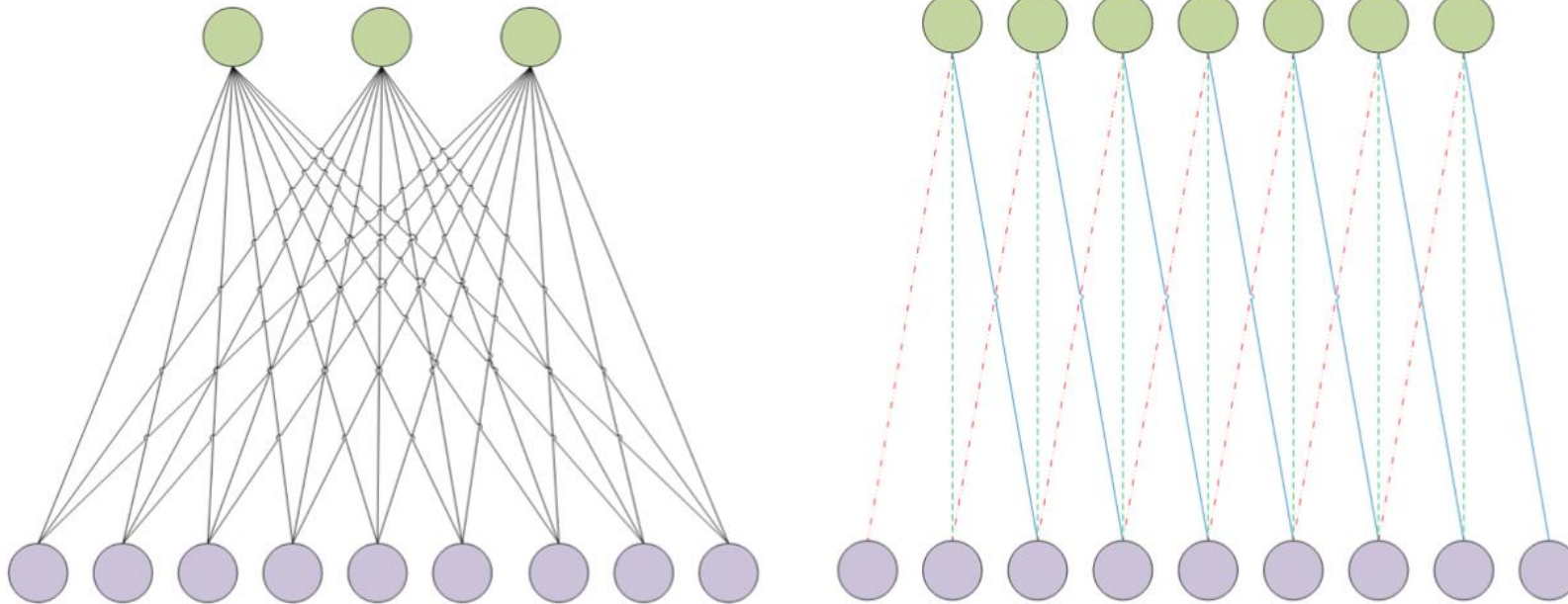
# Свертка

- Свертка – это применение одного и того же фильтра (нейрона) к разным частям исходного изображения
- В результате на разных частях исходных данных идет поиск одинаковых признаков
- Выходными данными свертки являются карты признаков
- Несколько фильтров позволяют сформировать несколько карт признаков одного сверточного слоя





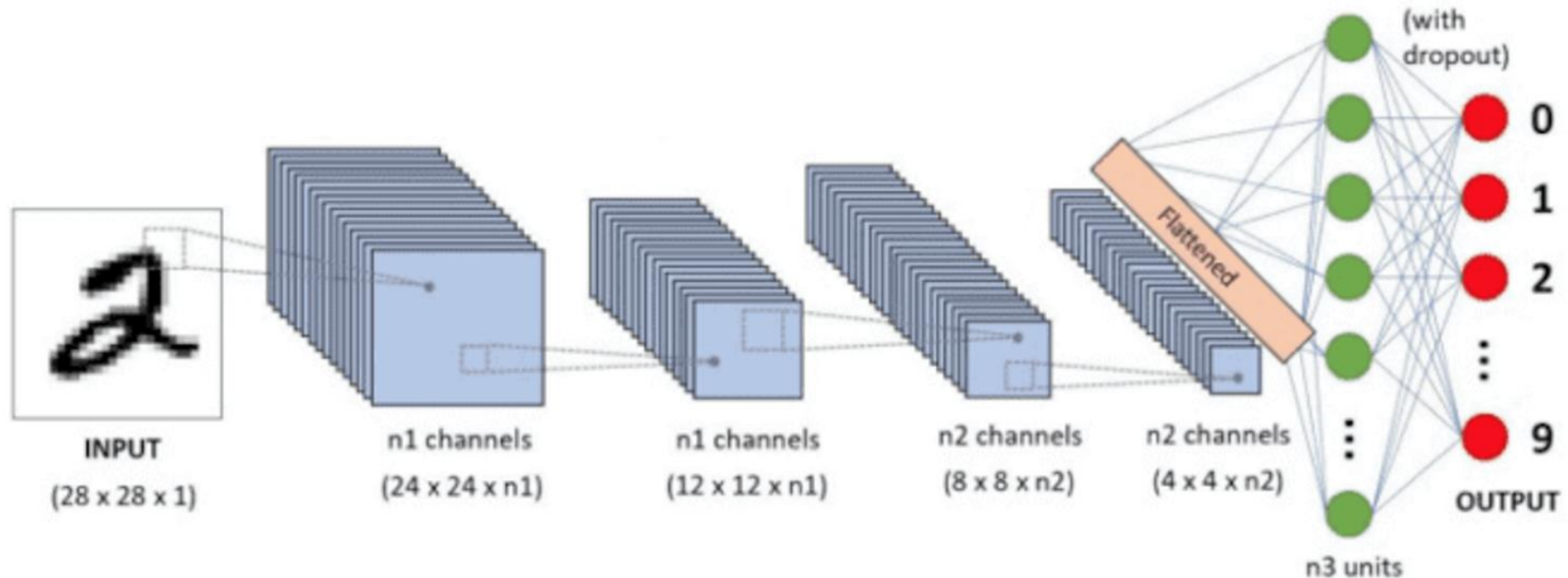
# Свойства сверточного слоя



- Разреженные взаимодействия – каждый нейрон связан с ограниченным числом входных нейронов
- Разделение параметров – в карте признаков все нейроны имеют одинаковый набор параметров
- Инвариантность – сдвиг исходных данных вызывает аналогичный сдвиг значений выходного слоя

# Сверточная нейросеть

- Сверточная нейросеть состоит из нескольких слоев: свертки, пуллинга, полносвязного
- Слои свертки и пуллинга чередуются друг за другом
- Слои свертки применяют набор  $n_1$ ,  $n_2$  фильтров к исходному изображению. Каждый фильтр ищет определенные признаки в исходных данных и формируется карту признаков
- Слои свертки обучаются, меняют количество каналов. Вход для свертки трехмерный:  $\text{ядро} \times \text{ядро} \times \text{каналы}$
- Слои пуллинга только уменьшают размерность карты признаков, количество каналов сохраняется.
- Данные последнего слоя пуллинга преобразуются в вектор для использования в полносвязном слое



# Вычисления в сверточном слое

0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

- Трехмерный случай для трех цветов
- Аналогично несколько каналов слоя свертки
- На входе нейрона 27 значений

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

Kernel Channel #1



308

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

Kernel Channel #2



-498

0	1	1
0	1	0
1	-1	1

Kernel Channel #3



164

+

+

↑  
Bias = 1

+ 1 = -25

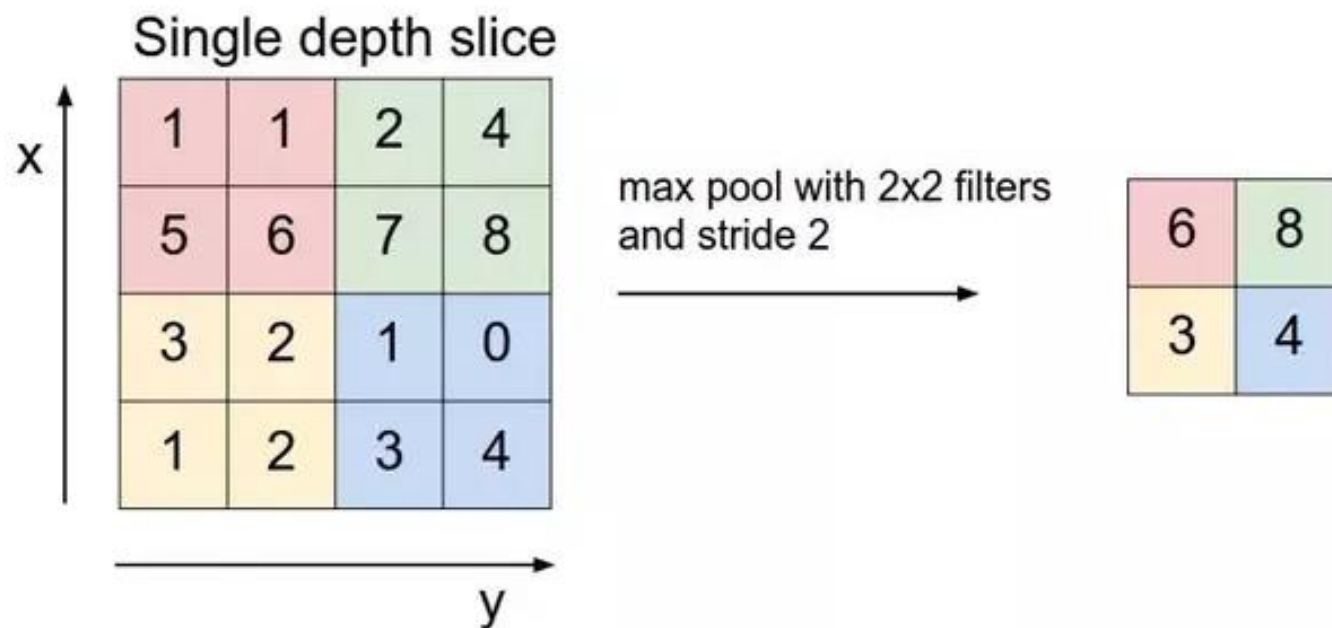
Output

-25				...
				...
				...
				...
...	...	...	...	...



# Пуллинг

- Слой пуллинга позволяет сократить размерность карты признаков
- Первый вариант пуллинга - максимальное значение из нескольких соседних



# Average Pooling

- Второй вариант вычисления – среднее значение из нескольких соседних

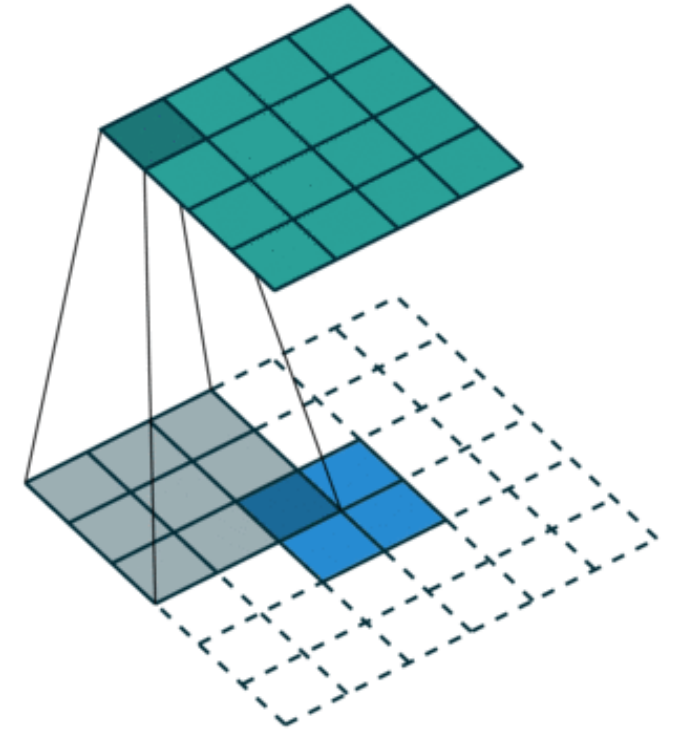
2	2	7	3
9	4	6	1
8	5	2	4
3	1	2	6

Average Pool  
→  
Filter - (2 x 2)  
Stride - (2, 2)

4.25	4.25
4.25	3.5

# Padding

- Padding – заполнение исходных данных для свертки
- Либо нулями, либо повторение соседних ячеек
- Padding – возможность сохранить размерность карты признаков или даже ее увеличить
- Для сохранения размера, padding равен  $(\text{размеру ядра свертки}-1)/2$



0 <sub>2</sub>	0 <sub>0</sub>	0 <sub>1</sub>	0	0	0	0
0 <sub>1</sub>	2 <sub>0</sub>	2 <sub>0</sub>	3	3	3	0
0 <sub>0</sub>	0 <sub>1</sub>	1 <sub>1</sub>	3	0	3	0
0	2	3	0	1	3	0
0	3	3	2	1	2	0
0	3	3	0	2	3	0
0	0	0	0	0	0	0

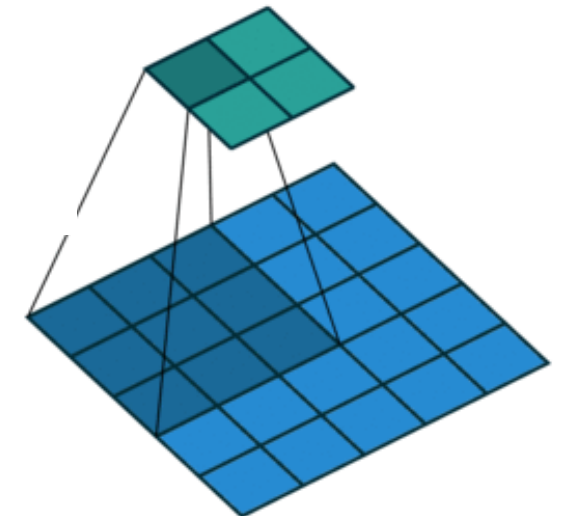
1	6	5
7	10	9
7	10	8

# Шаг свертки - stride

- Stride - шаг свертки
- Это регулируемый параметр, который определяет размерность карты признаков

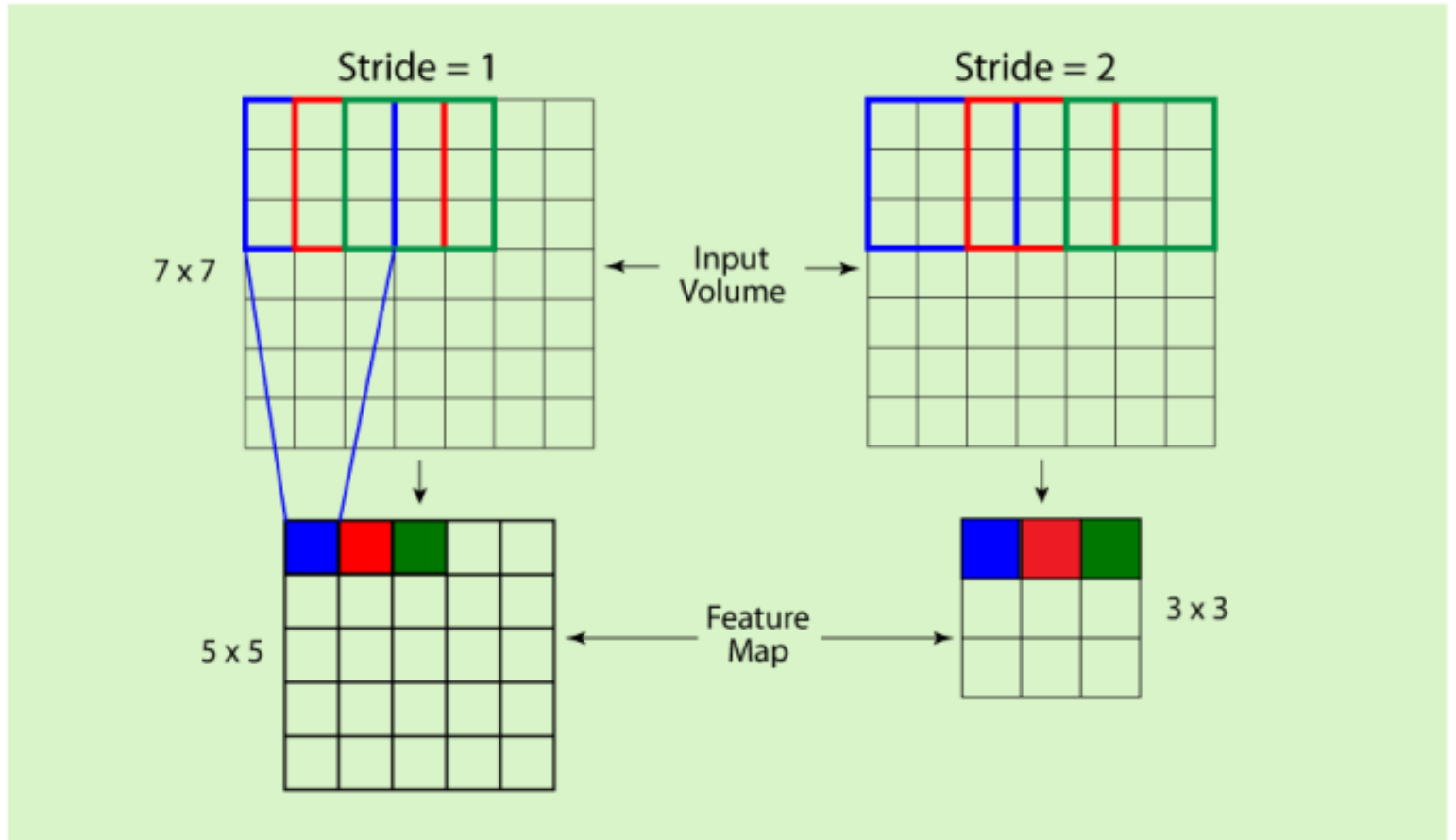
0 <sub>2</sub>	0 <sub>0</sub>	0 <sub>1</sub>	0	0	0	0
0 <sub>1</sub>	2 <sub>0</sub>	2 <sub>0</sub>	3	3	3	0
0 <sub>0</sub>	0 <sub>1</sub>	1 <sub>1</sub>	3	0	3	0
0	2	3	0	1	3	0
0	3	3	2	1	2	0
0	3	3	0	2	3	0
0	0	0	0	0	0	0

1	6	5
7	10	9
7	10	8



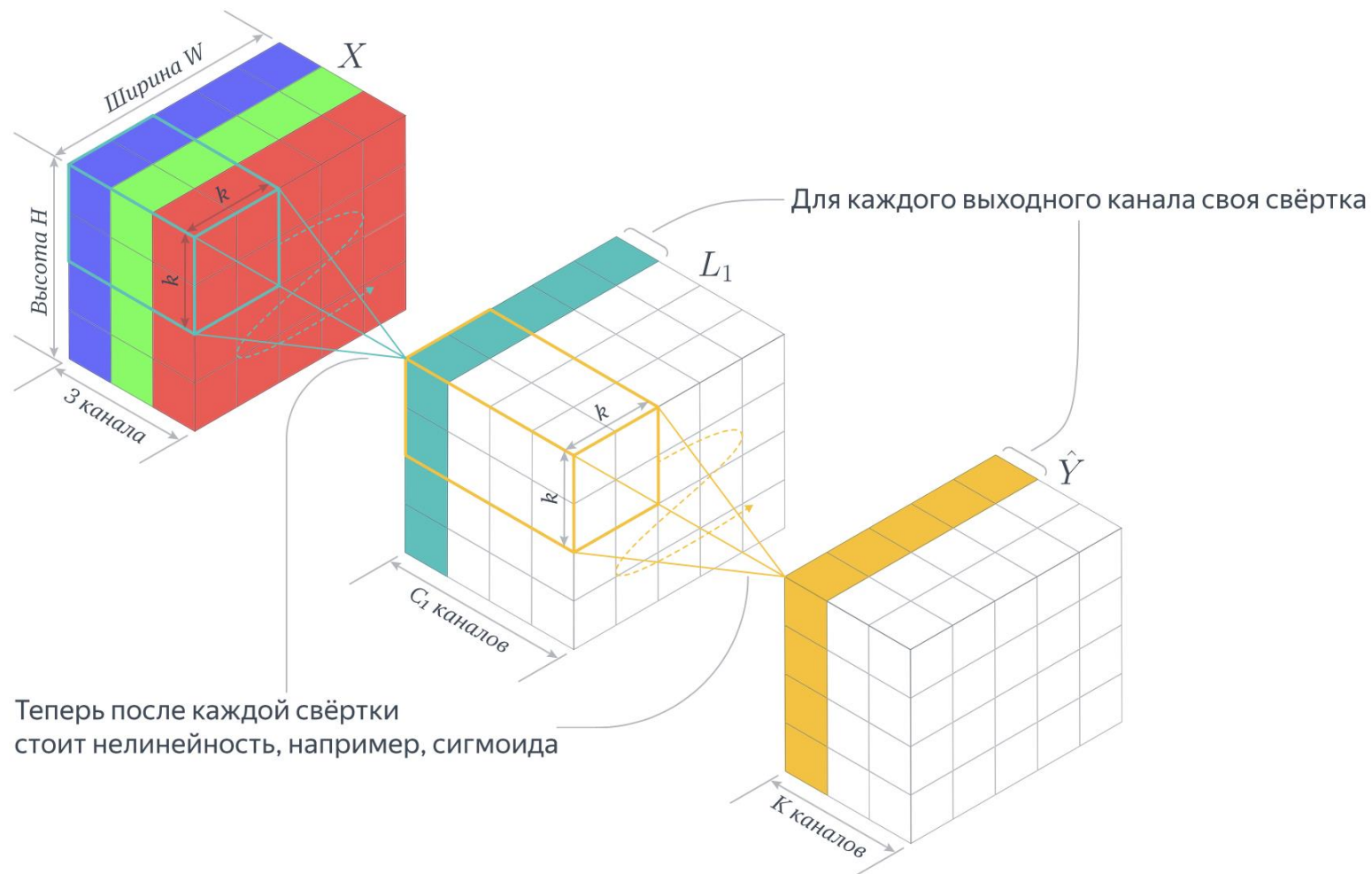
# Stride

- Stride - это еще один способ сократить размерность карты признаков
- Чем больше шаг, тем меньше становится итоговая карта признаков



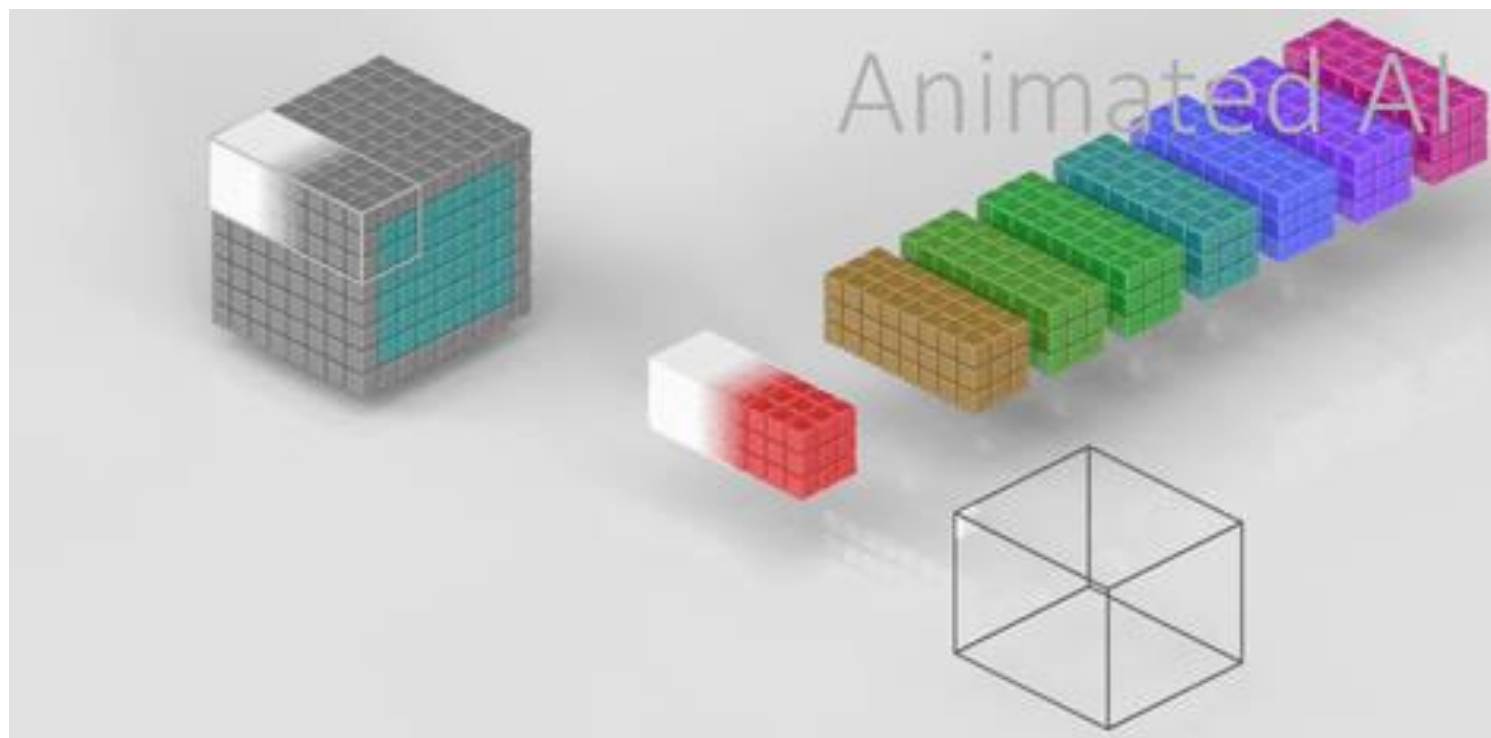


# Хорошая статья по CNN



<https://education.yandex.ru/handbook/ml/article/svyortochnye-nejroseti>

# Визуализация CNN



<https://animatedai.github.io>

# Пример

```
def __init__(self, hidden_size=32, classes=100):
    super(Cifar100_MLP, self).__init__()
    # https://blog.jovian.ai/image-classification-of-cifar100-dataset-using-pytorch-8b7145242df1
    self.seq = nn.Sequential(
        Normalize([0.5074,0.4867,0.4411],[0.2011,0.1987,0.2025]),
        # первый способ уменьшения размерности картинки - через stride
        nn.Conv2d(3, HIDDEN_SIZE, 5, stride=4, padding=2),
        nn.ReLU(),
        # второй способ уменьшения размерности картинки - через слой пуллинг
        nn.Conv2d(HIDDEN_SIZE, HIDDEN_SIZE*2, 3, stride=1, padding=1),
        nn.ReLU(),
        nn.AvgPool2d(4), #nn.MaxPool2d(4),
        nn.Flatten(),
        nn.Linear(HIDDEN_SIZE*8, classes),
    )
```

Layer (type)	Output Shape	Param #
Normalize-1	[-1, 3, 32, 32]	0
Conv2d-2	[-1, 32, 8, 8]	2,432
ReLU-3	[-1, 32, 8, 8]	0
Conv2d-4	[-1, 64, 8, 8]	18,496
ReLU-5	[-1, 64, 8, 8]	0
AvgPool2d-6	[-1, 64, 2, 2]	0
Flatten-7	[-1, 256]	0
Linear-8	[-1, 3]	771

# Результаты обучения

- Точность классификации удалось значительно повысить
- Однако точность на тестовой выборке пока значительно уступает точности на обучающей
- Нейросеть “запомнила” примеры с обучающей выборки

train					
	precision	recall	f1-score	support	
0	1.0000	1.0000	1.0000	500	
55	1.0000	1.0000	1.0000	500	
58	1.0000	1.0000	1.0000	500	
accuracy			1.0000	1500	
macro avg	1.0000	1.0000	1.0000	1500	
weighted avg	1.0000	1.0000	1.0000	1500	
-----					
test					
	precision	recall	f1-score	support	
0	0.9238	0.9700	0.9463	100	
55	0.8515	0.8600	0.8557	100	
58	0.9043	0.8500	0.8763	100	
accuracy			0.8933	300	
macro avg	0.8932	0.8933	0.8928	300	
weighted avg	0.8932	0.8933	0.8928	300	
-----					

# Стохастический градиентный спуск

$L(f(\mathbf{x}(i); \boldsymbol{\theta}), \mathbf{y}(i))$  – значение функции потерь

$f(\mathbf{x}(i); \boldsymbol{\theta})$  – результат вычисления нейронной сети от входа  $\mathbf{x}(i)$  и параметров (весов)  $\boldsymbol{\theta}$

Обновление на  $k$ -ой итерации стохастического градиентного спуска (СГС)

**Require:** скорость обучения  $\epsilon_k$

**Require:** Начальные значения параметров  $\boldsymbol{\theta}$

**while** критерий остановки не выполнен **do**

Выбрать из обучающего набора мини-батч  $m$  примеров  $\{\mathbf{x}(1), \dots, \mathbf{x}(m)\}$  и соответствующие им метки  $\mathbf{y}(i)$ .

Вычислить оценку градиента:  $\mathbf{g} \leftarrow + (1/m) \nabla_{\boldsymbol{\theta}} \sum_i L(f(\mathbf{x}(i); \boldsymbol{\theta}), \mathbf{y}(i))$ .

Применить обновление:  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \epsilon \mathbf{g}$ .

**end while**



# Стохастический градиентный спуск

- Основной параметр алгоритма СГС – скорость обучения  $\epsilon$
- На практике же необходимо постепенно уменьшать скорость обучения со временем
- $\epsilon_k = (1 - \alpha) \epsilon_k + \alpha \epsilon_k$ , где  $\alpha = k / \tau$ . После  $\tau$ -й итерации  $\epsilon$  остается постоянным.
- Если скорость изменяется линейно, то нужно задать параметры  $\epsilon_0, \epsilon_\tau$  ит.

# Импульсный метод

Стохастический градиентный спуск (СГС) с учетом импульса

**Require:** скорость обучения  $\varepsilon$ , параметр импульса  $\alpha$

**Require:** начальные значения параметров  $\theta$ , начальная скорость  $v$

**while** критерий остановки не выполнен **do**

Выбрать из обучающего набора мини-пакет  $m$  примеров  $\{x(1), \dots, x(m)\}$  и соответствующие им метки  $y(i)$ .

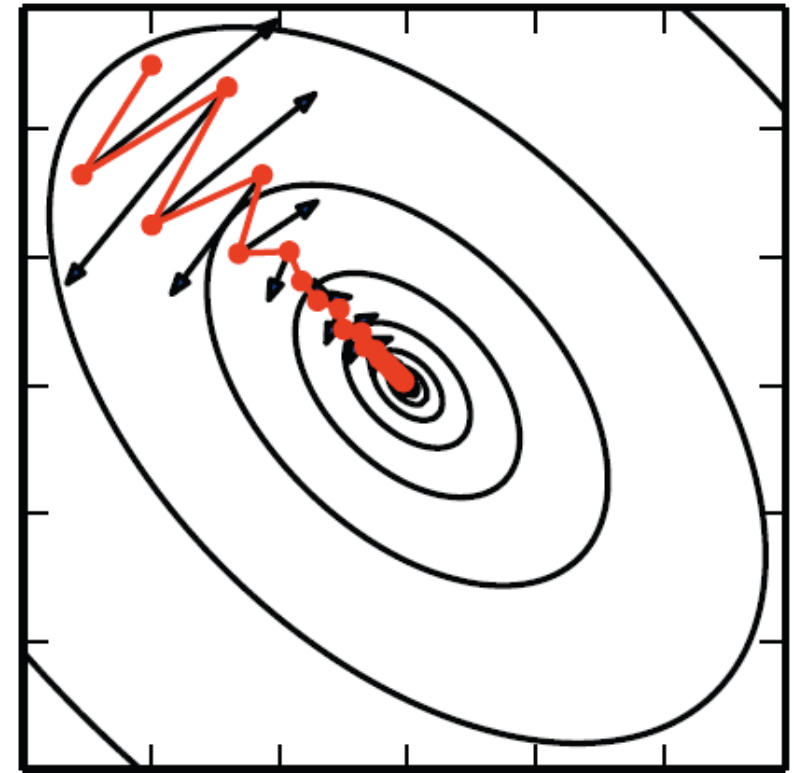
Вычислить оценку градиента:

$$g \leftarrow (1/m) \nabla_{\theta} \sum_i L(f(x(i); \theta), y(i)).$$

Вычислить обновление скорости:  $v \leftarrow \alpha v - \varepsilon g$ .

Применить обновление:  $\theta \leftarrow \theta + v$ .

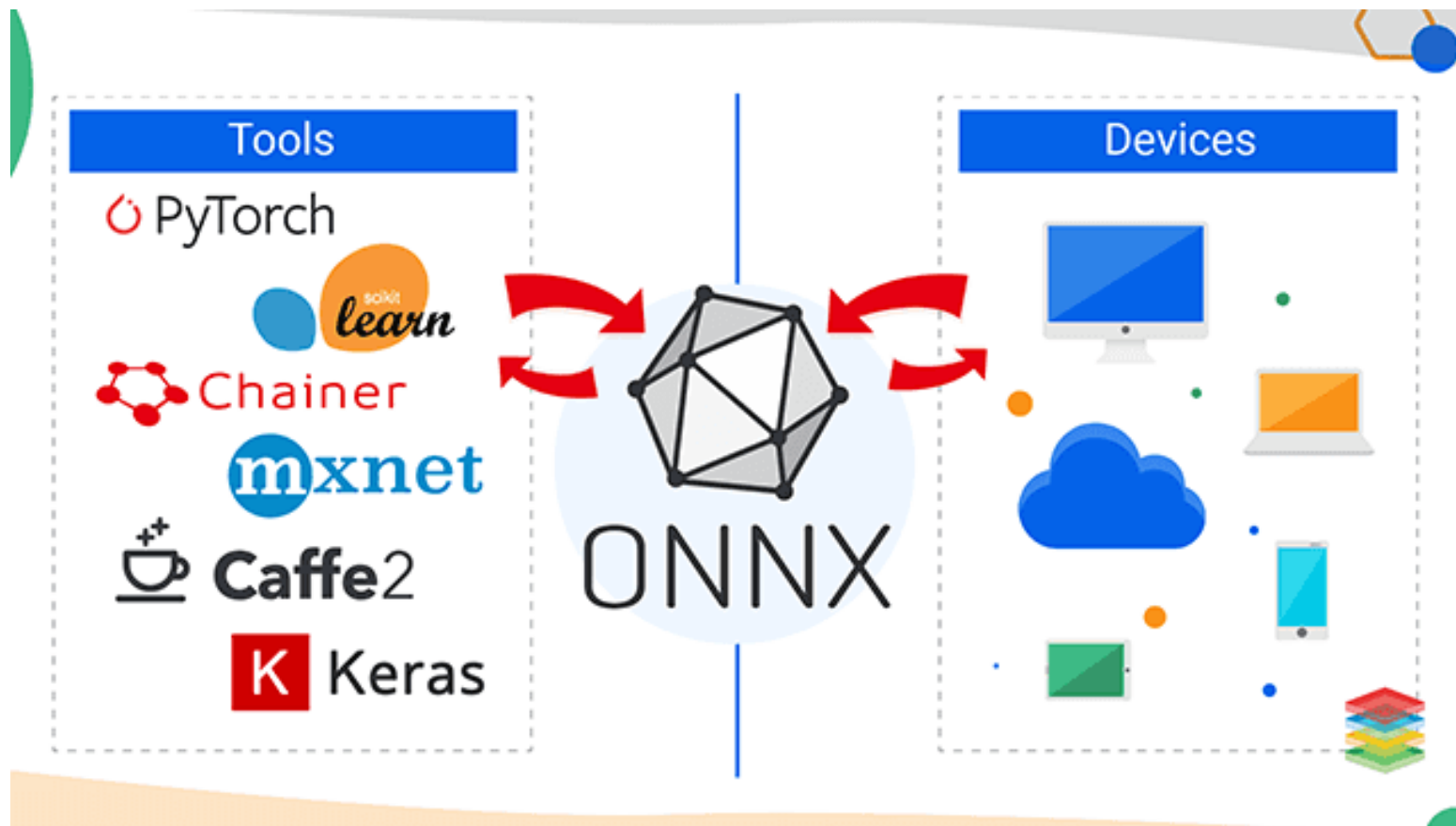
**end while**



Импульсный алгоритм можно рассматривать как имитацию движения частицы, подчиняющейся динамике Ньютона.

# ONNX

- ONNX - библиотека для конвертации моделей между разными технологиями
- ONNX дает возможность исследователям и разработчикам выбрать нужную комбинацию инструментов для решения задачи
- ONNX.js позволяет запускать модели в браузере, то есть на стороне пользователя



# ONNX

## Step 3. Select class labels and get predictions

Выбрать файл cifar100\_CNN.onnx Select ONNX file

Выбрать файл c25c94fe96\_1000.jpg

Class label 54

0,50,54

