Свёрточные нейронные сети

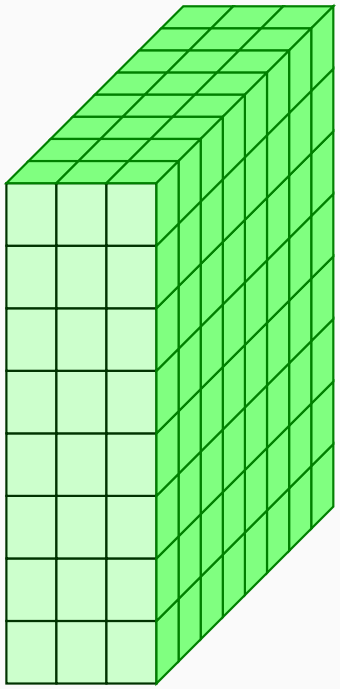
Алексей Артёмов

16 сентября 2016 г.

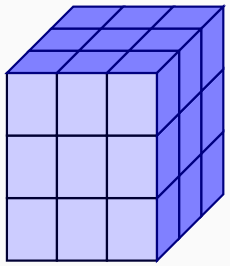
Сверточный слой (convolution layer)

$$H \times W \times C$$

Изображение: $8 \times 8 \times 3$

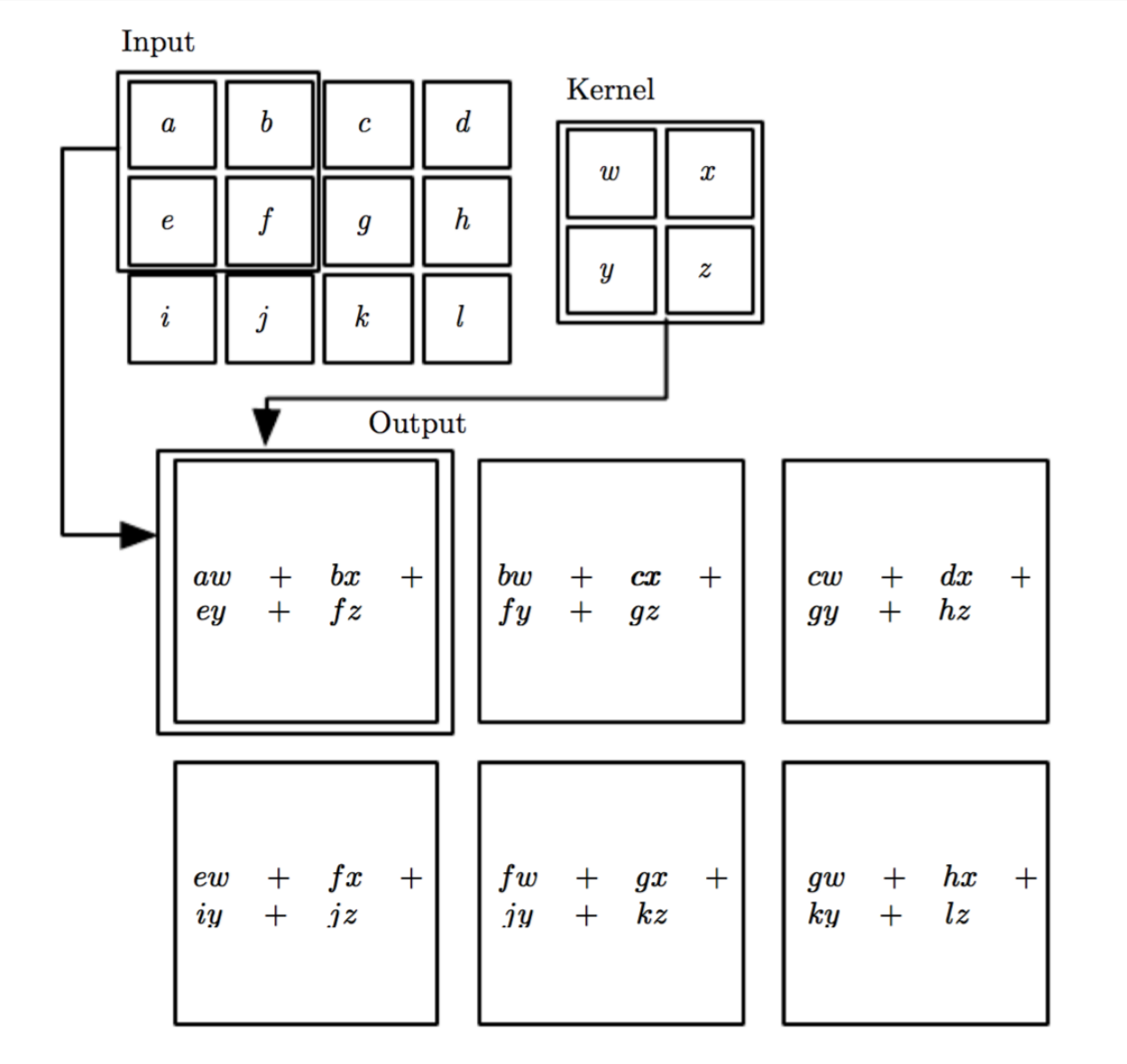


Фильтр: $3 \times 3 \times 3$



Свернуть изображение с фильтром:
пробежать по изображению (пространственно),
вычисляя скалярные произведения

Свертка изображения с ядром

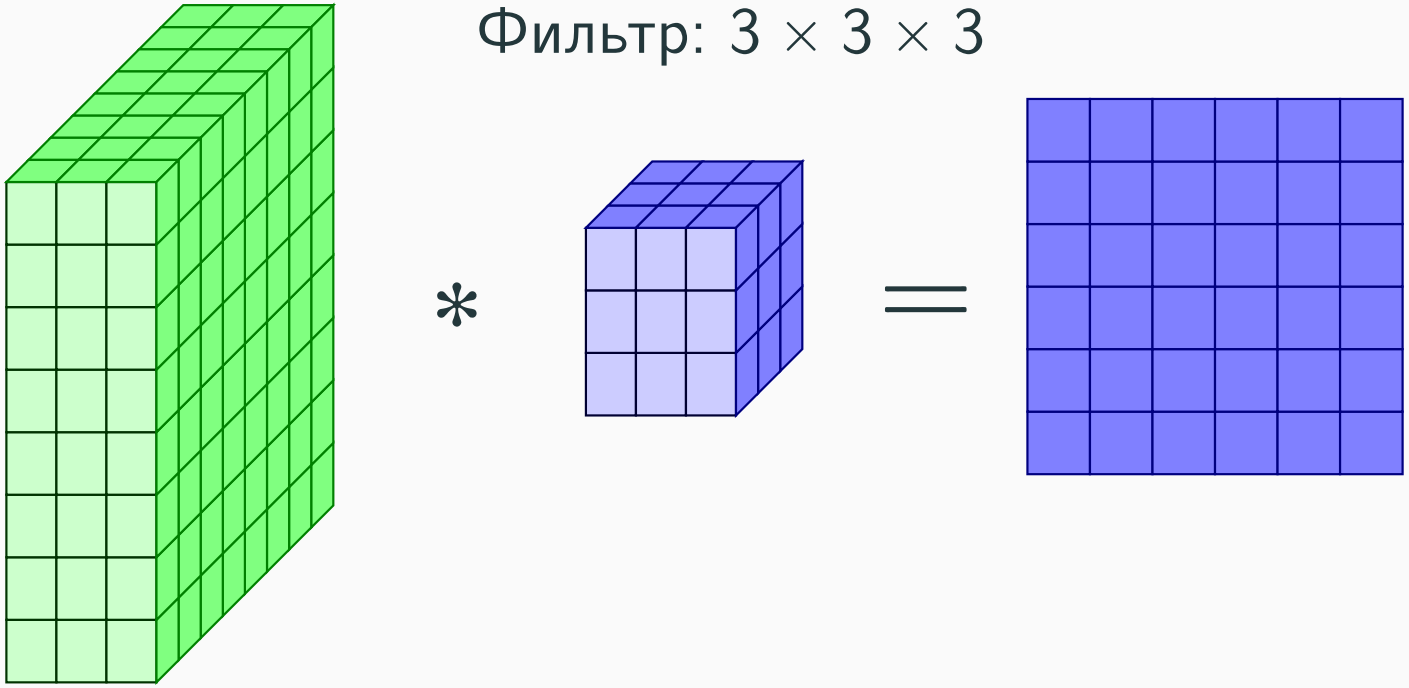


Сверточный слой (convolution layer)

Изображение: $8 \times 8 \times 3$

Карта: 6×6

Фильтр: $3 \times 3 \times 3$



$$y = w^T x + b$$

Сколько параметров?

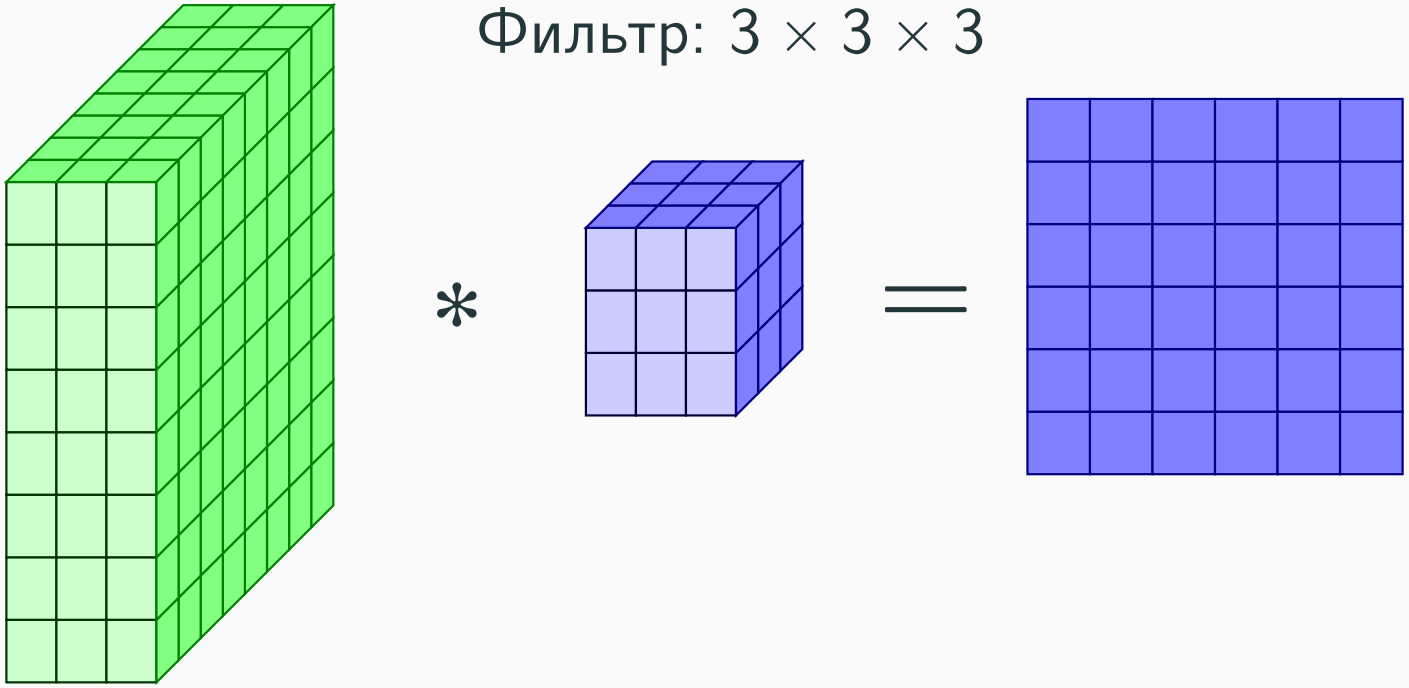
Сколько параметров у полносвязного слоя?

Сверточный слой (convolution layer)

Изображение: $8 \times 8 \times 3$

Карта: 6×6

Фильтр: $3 \times 3 \times 3$



$$y = w^T x + b$$

Сколько параметров? $28 = 3 \times 3 \times 3 + 1$

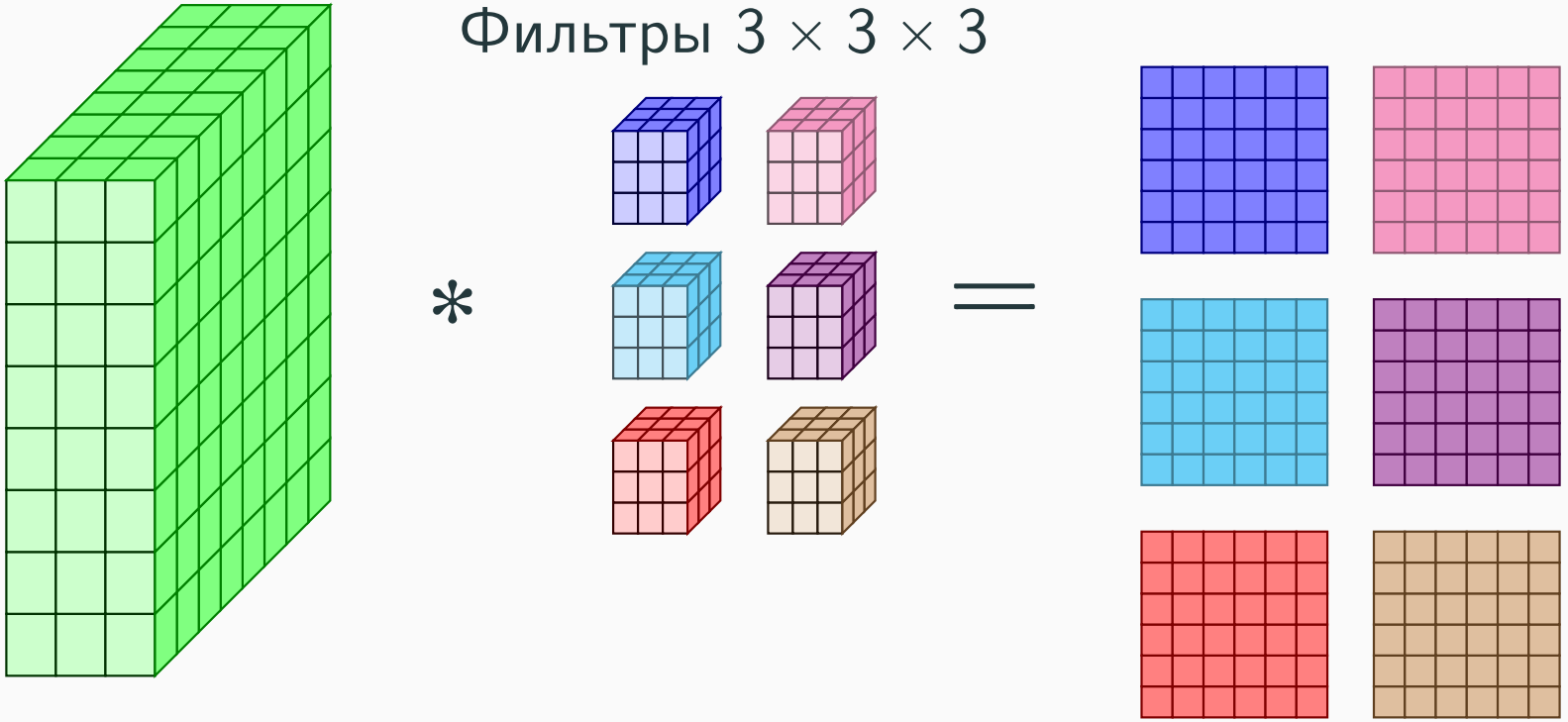
Сколько параметров у полносвязного слоя? $6913 = 192 \times 36 + 1$

Сверточный слой (convolution layer)

Изображение: $8 \times 8 \times 3$

Фильтры $3 \times 3 \times 3$

Карты 6×6

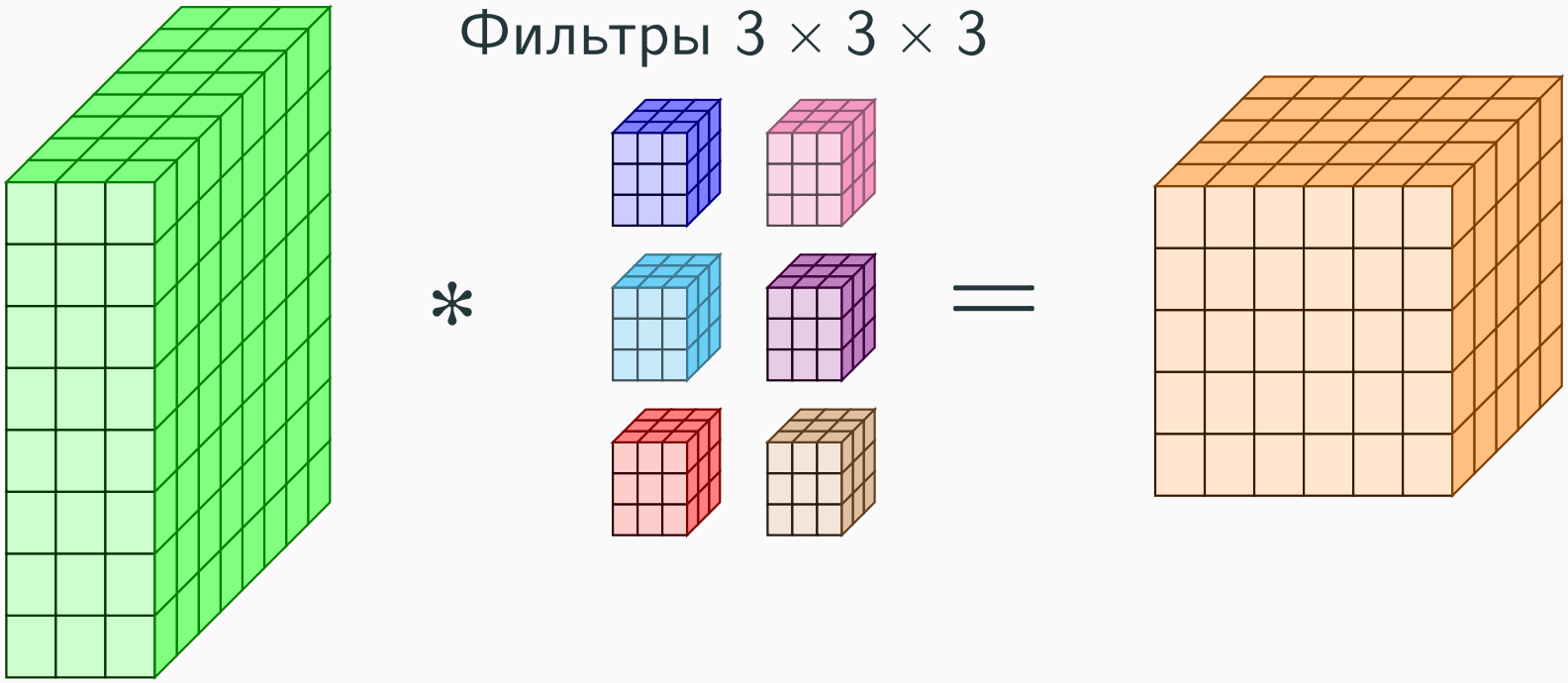


Число фильтров F: гиперпараметр

Сверточный слой (convolution layer)

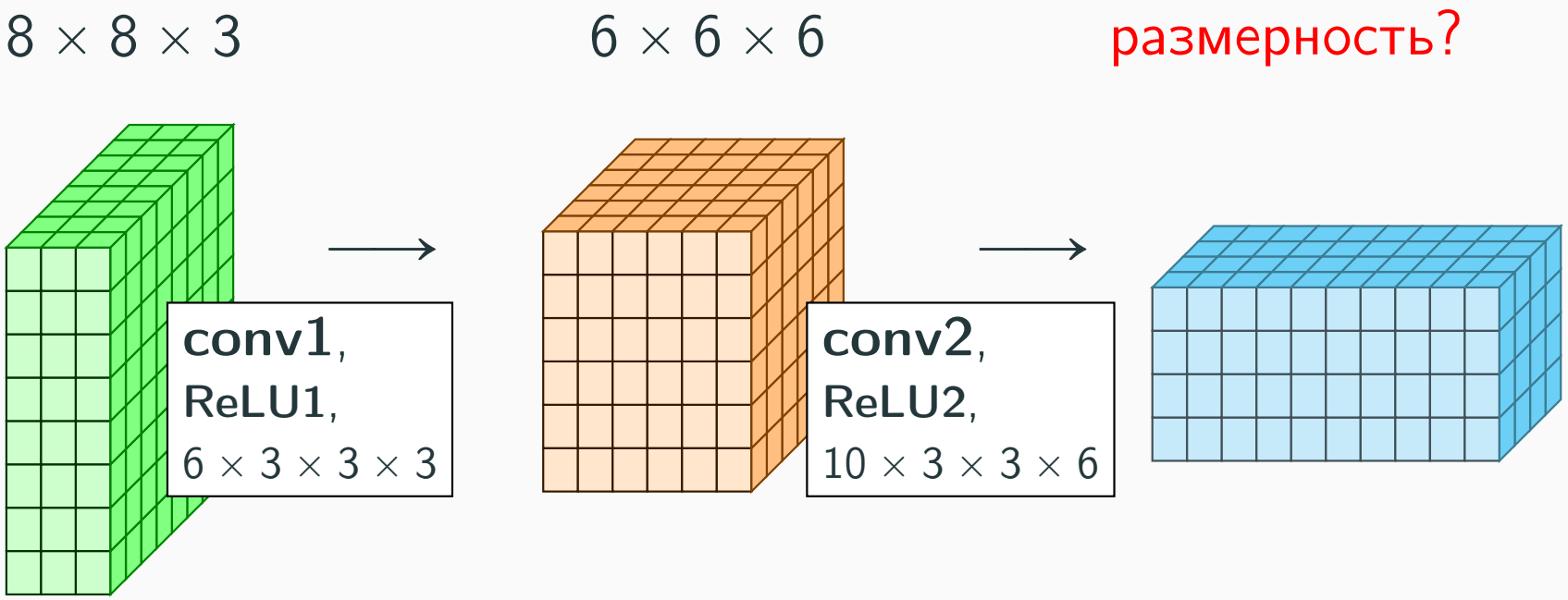
Изображение: $8 \times 8 \times 3$

Карты $6 \times 6 \times 6$

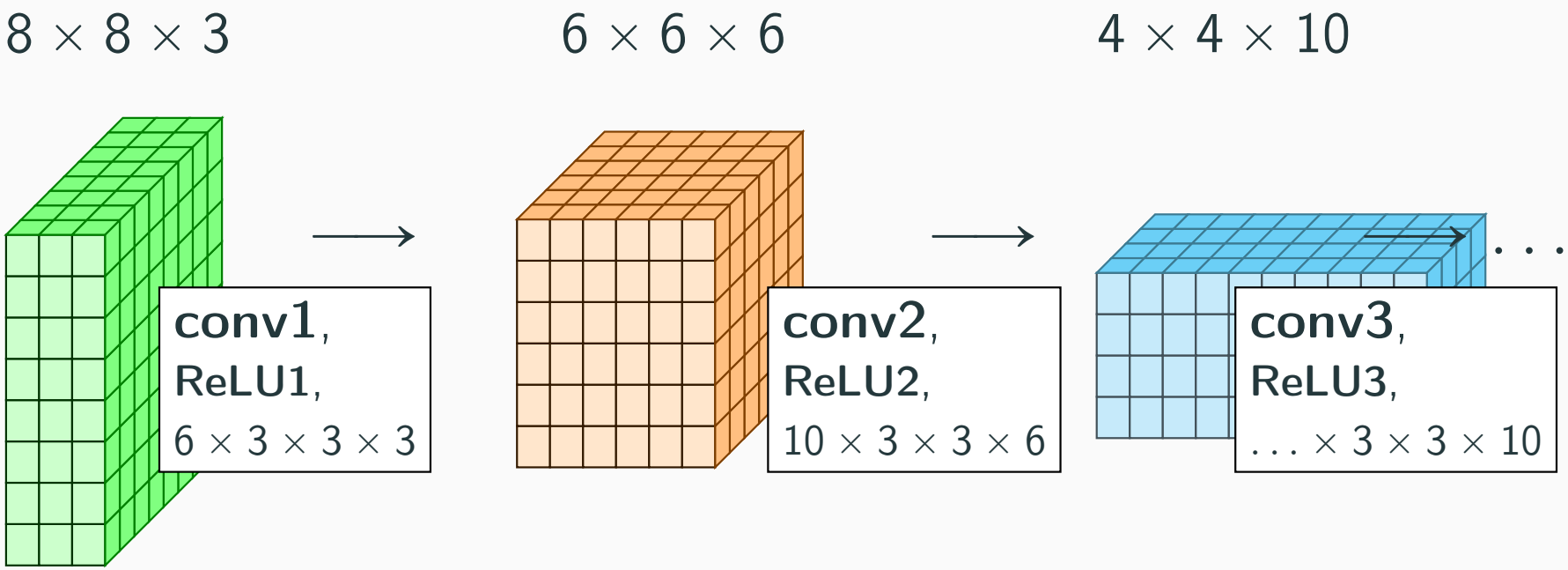


Тензор $6 \times 6 \times 6$: входное изображение следующего сверточного слоя

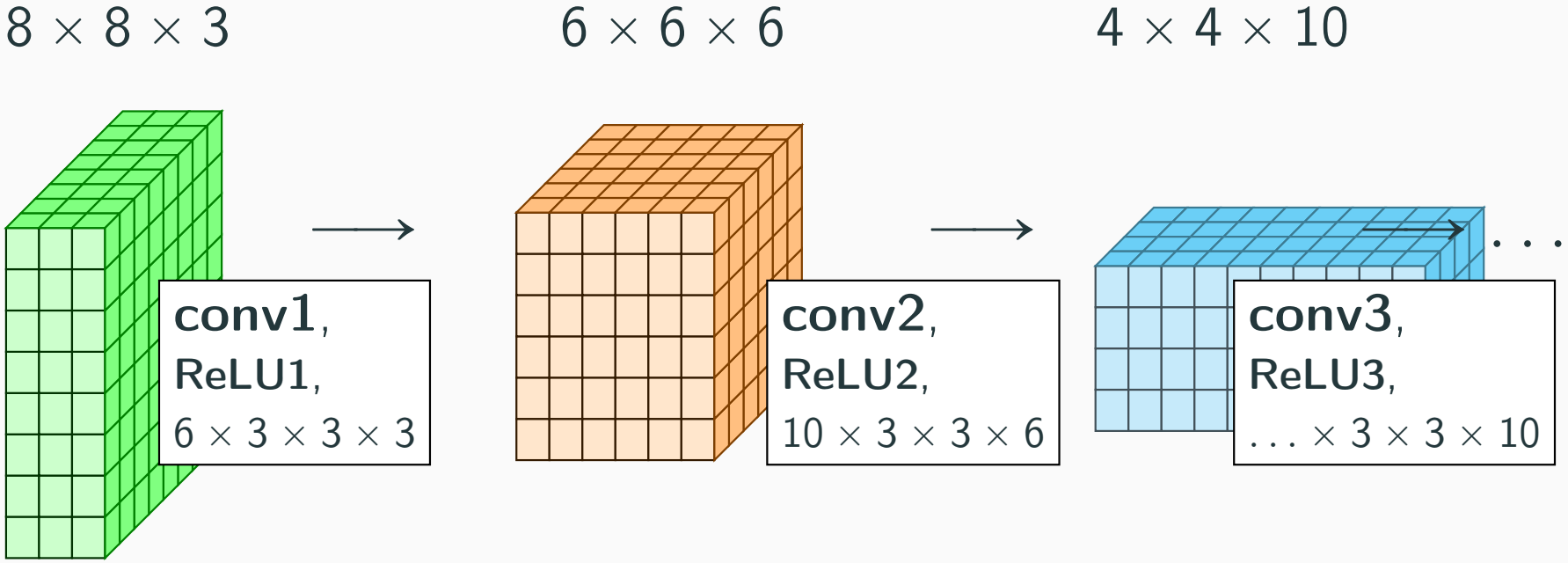
Сверточная сеть: последовательность сверток



Сверточная сеть: последовательность сверток

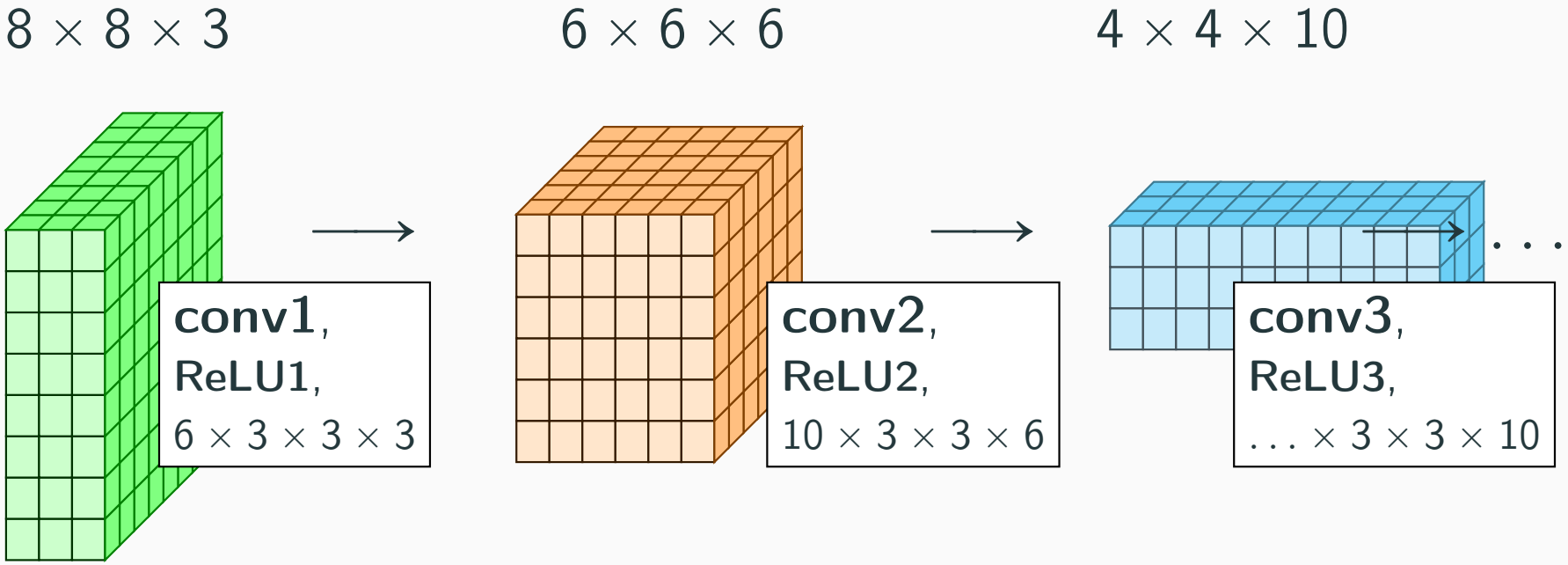


Сверточная сеть: последовательность сверток



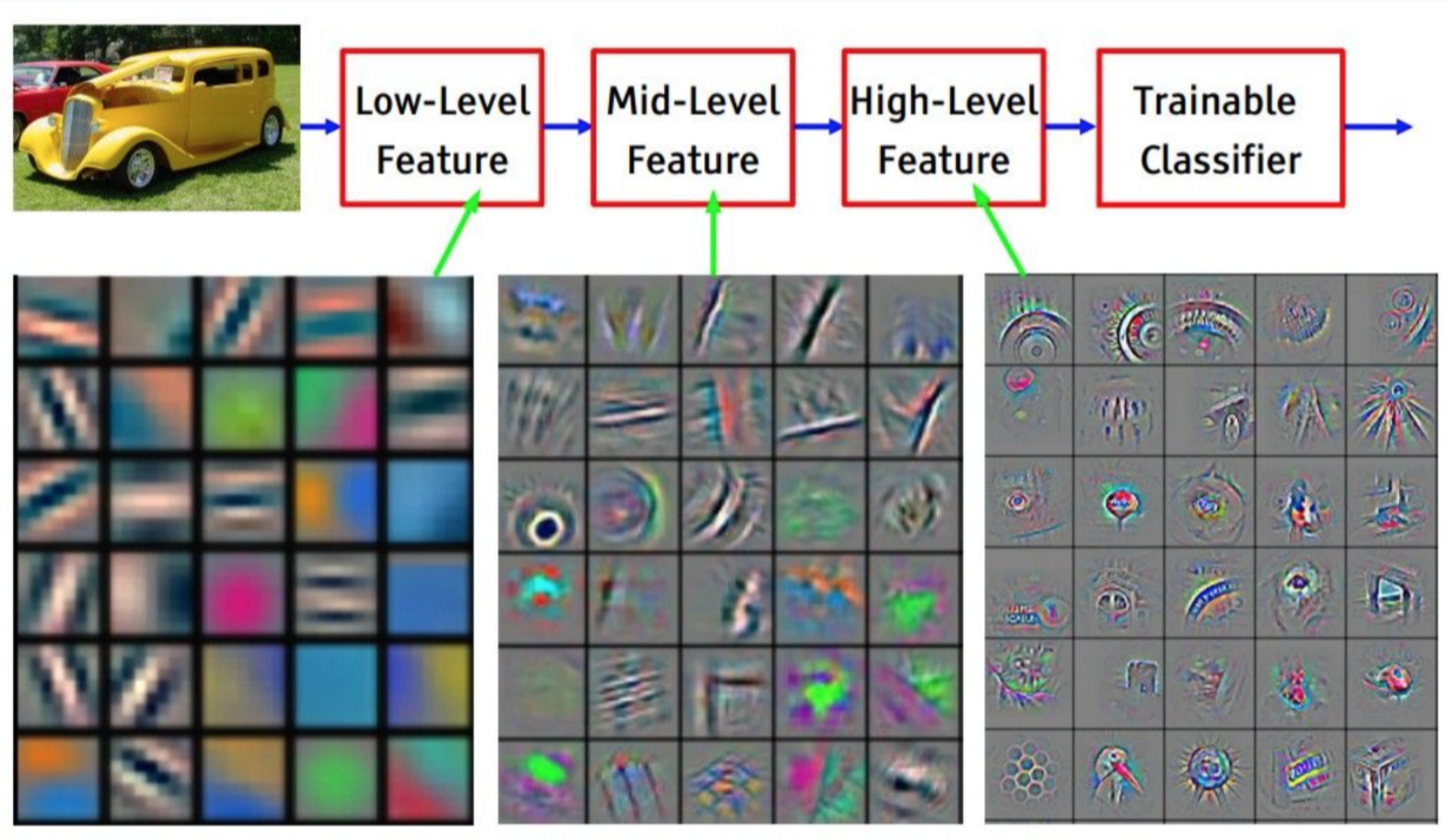
- Размер фильтра conv2?
- Глубина входа conv2?
- Глубина выхода conv2?
- Число параметров conv2?
- Число нейронов conv2?

Сверточная сеть: последовательность сверток



- $F_{\text{OUT}} \times H_K \times W_K \times F_{\text{IN}}$ 10 фильтров $3 \times 3 \times 6$
- Размер фильтра conv2? 3×3
- Глубина входа conv2? 6
- Глубина выхода conv2? 10
- Число параметров conv2? $550 = 10 \times 3 \times 3 \times 6 + 10$
- Число нейронов conv2? 270

Иерархия представлений



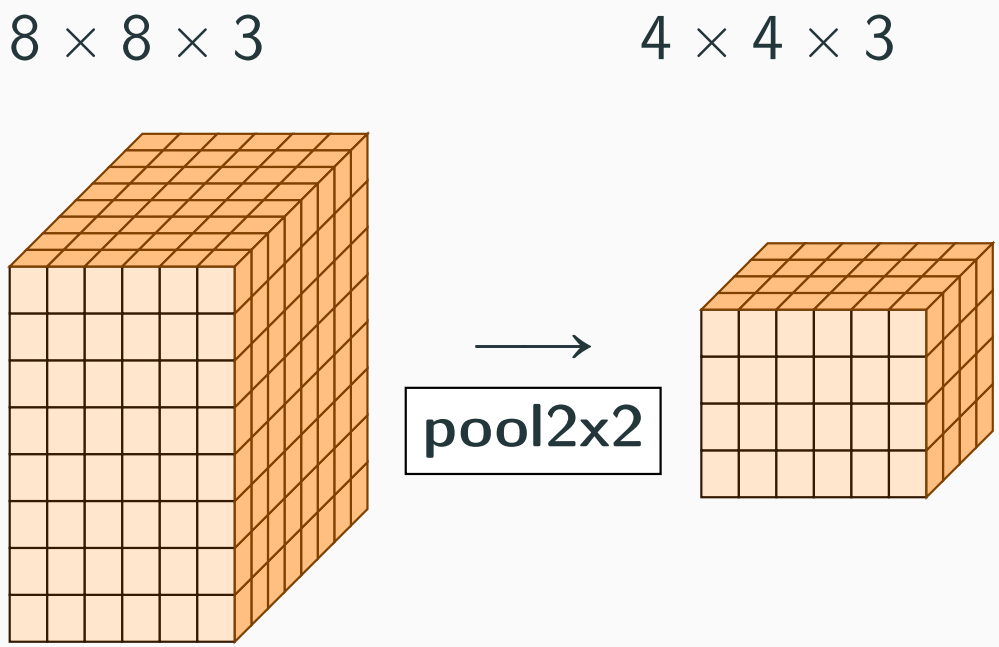
(Слайд Yann LeCun)

Сверточный слой: итог

- **Вход:** тензор $H_{\text{IN}} \times W_{\text{IN}} \times F_{\text{IN}}$
- **Выход:** тензор $H_{\text{OUT}} \times W_{\text{OUT}} \times F_{\text{OUT}}$
- 4 гиперпараметра:
  - F: число фильтров
  - H_K, W_K: пространственный размер фильтров
  - S: шаг
  - P: количество заполнения нулями
- Соотношение размеров входа и выхода:

$$\begin{aligned} H_{\text{OUT}} &= (H_{\text{IN}} - H_K + 2P)/S + 1, \\ W_{\text{OUT}} &= (W_{\text{IN}} - W_K + 2P)/S + 1, \\ F_{\text{OUT}} &= F \end{aligned}$$

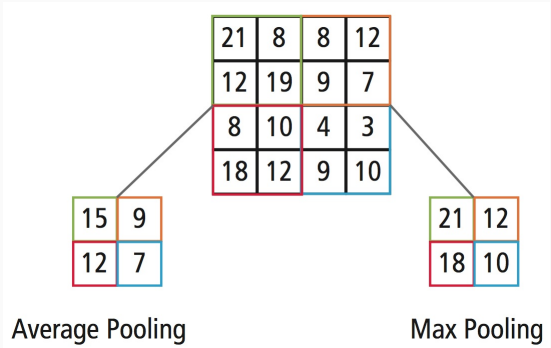
Pooling-слой



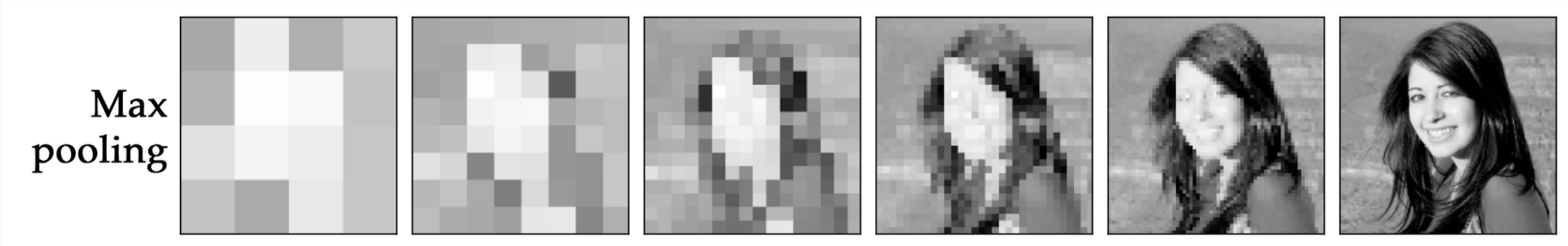
- Снижение пространственной размерности
- Применяется к каждой выходной карте независимо

Pooling-слой

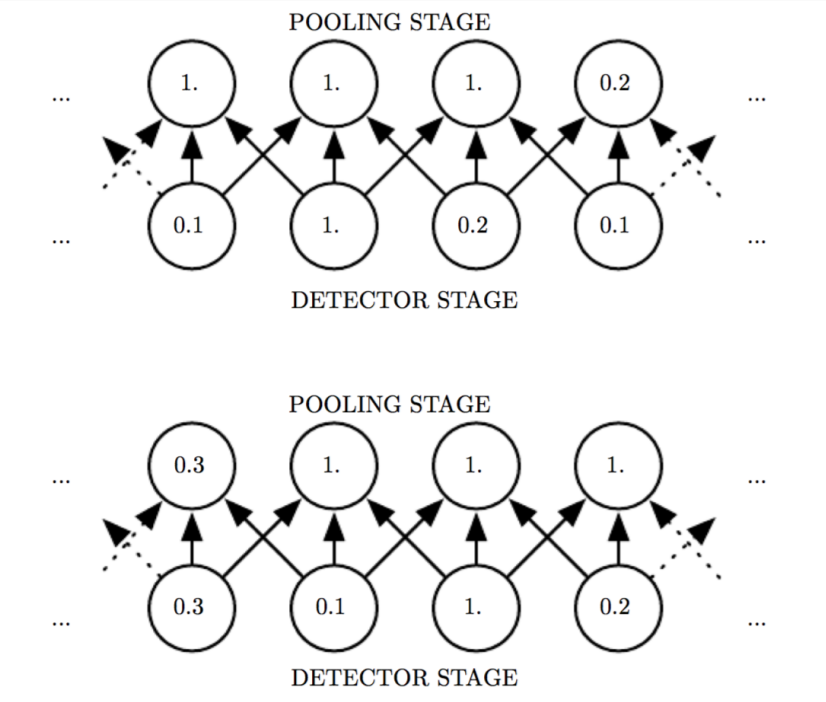
- Замена выхода нейрона статистикой, подсчитанной по его соседям
  - max pooling
  - (weighted) average pooling

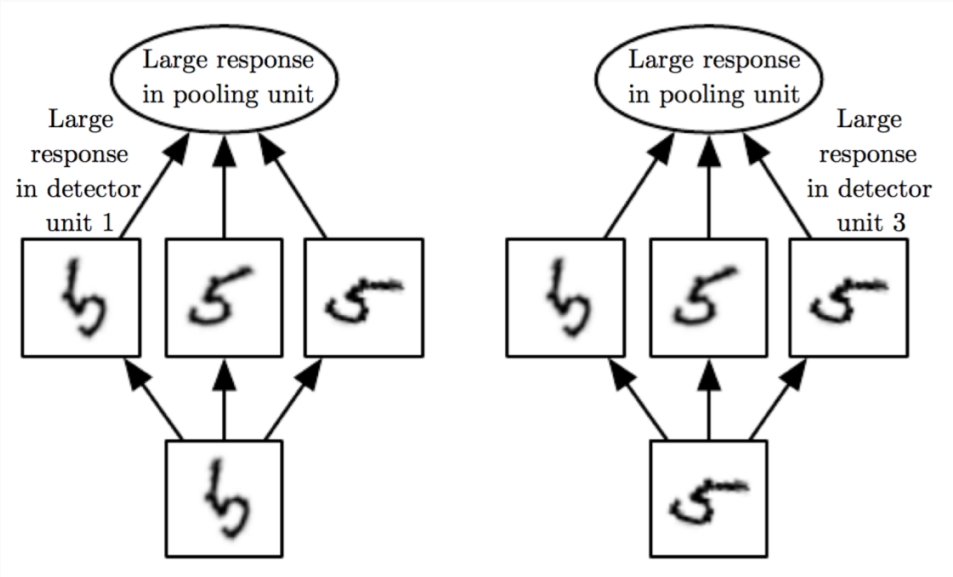


- Обеспечивает приблизительную инвариантность выхода к малому переносу входа
- Наличие признака важнее, чем его точная позиция!



Pooling-слой и инвариантные представления





- Изменение входных данных может достигать 100%, но изменение выхода остается небольшим
- Инвариантность к более сложным преобразованиям входных данных (поворот, ...)

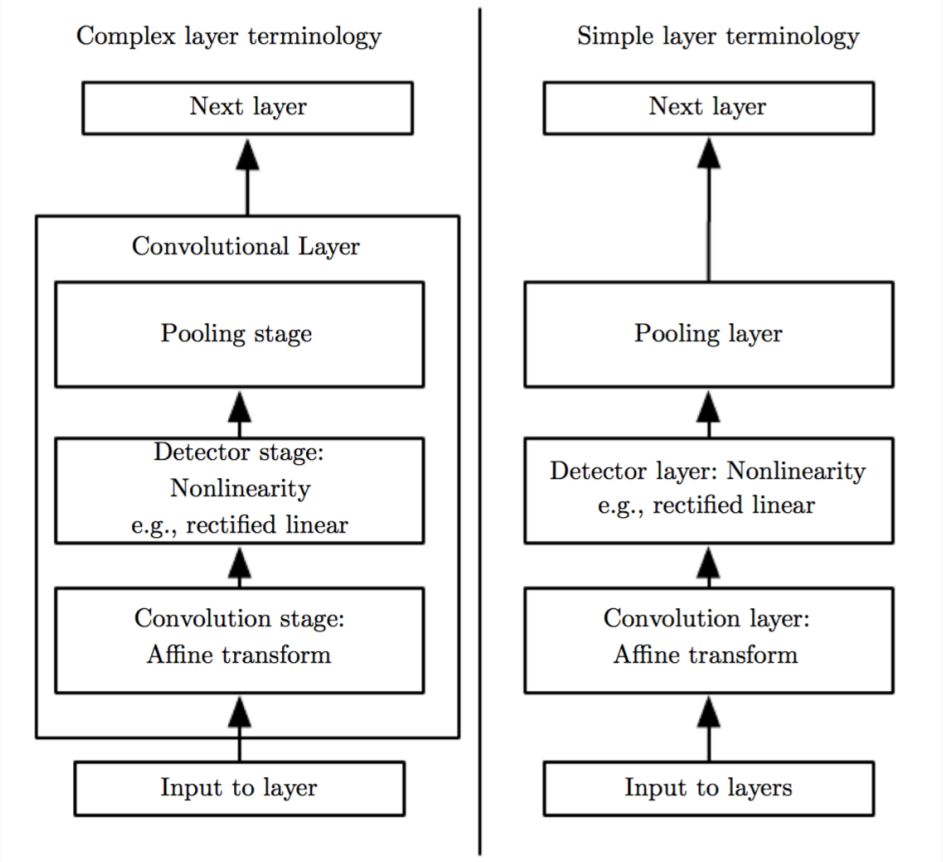
Pooling-слой: итог

- **Вход:** тензор $H_{\text{IN}} \times W_{\text{IN}} \times F_{\text{IN}}$
- **Выход:** тензор $H_{\text{OUT}} \times W_{\text{OUT}} \times F_{\text{OUT}}$
- 3 гиперпараметра:
  - F: пространственный размер объединения
  - S: шаг объединения
- Соотношение размеров входа и выхода:

$$\begin{aligned} H_{\text{OUT}} &= (H_{\text{IN}} - F)/S + 1, \\ W_{\text{OUT}} &= (W_{\text{IN}} - F)/S + 1, \\ F_{\text{OUT}} &= F_{\text{IN}} \end{aligned}$$

Архитектура сверточного слоя

1. Свертка входа
   с множеством ядер
2. Пропускание откликов
   через нелинейную
   активацию (детектор)
3. Объединение соседних
   активаций (pooling)



Немного математики

Операция свертки

- $x(t)$ — некоторая функция $t \in \mathbb{R}$
- Свертка $x(t)$ с ядром $k(t)$ — это функция

$$s(t) = (x * k)(t) \equiv \int_{-\infty}^{\infty} x(\tau)k(t-\tau)d\tau$$

- Если $t \in \mathbb{Z}$ (дискретна)

$$s(t) = (x * k)(t) \equiv \sum_{\tau=-\infty}^{\infty} x(\tau)k(t-\tau)$$

- Образ свертки $s(t)$ — карта откликов (feature map)

Свертка изображения с ядром

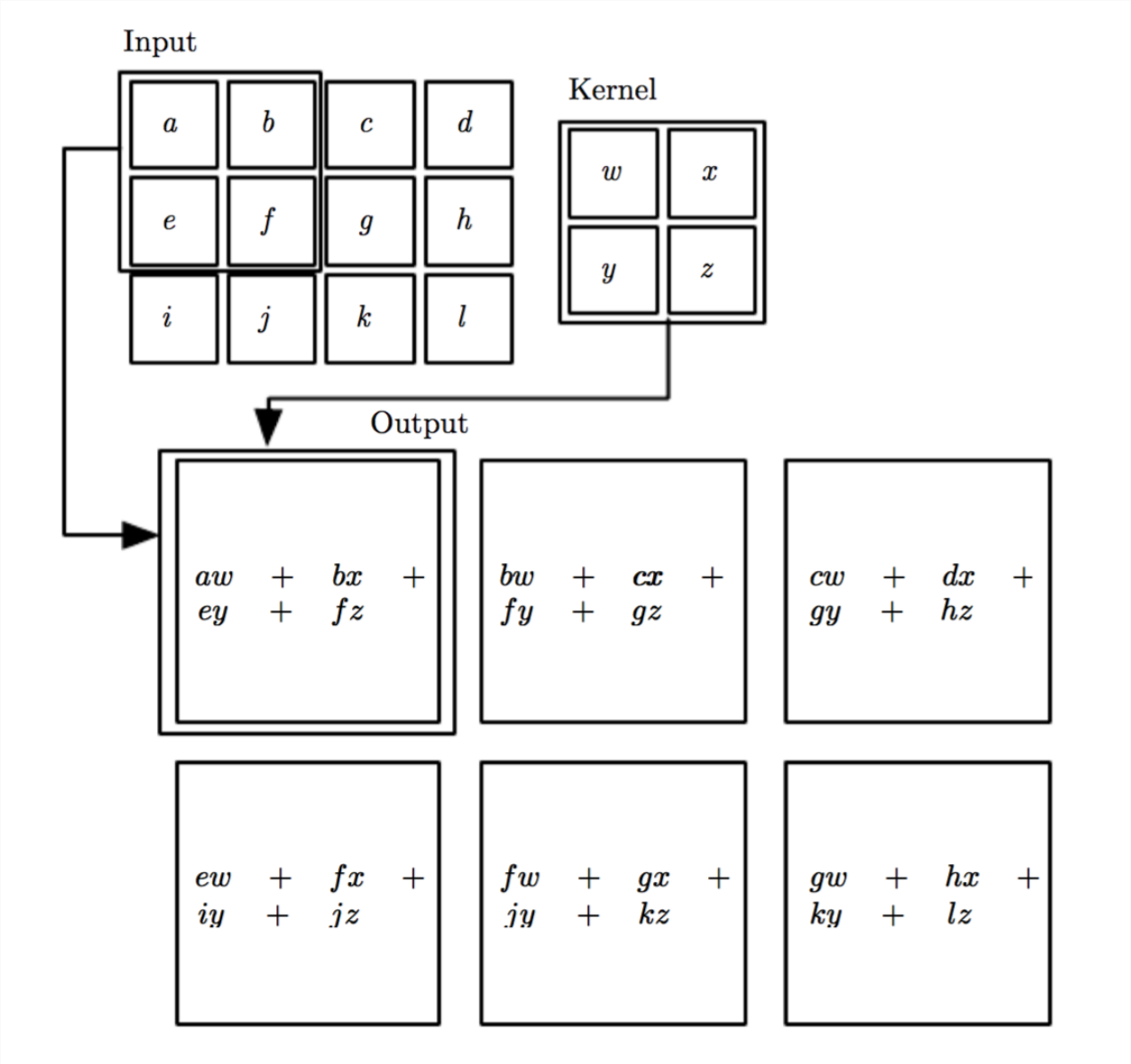
- $I(i,j)$ — изображение
- Свертка изображения $I(i,j)$ с ядром $K(t,s)$

$$S(i,j) = (I * K)(i,j) \equiv \sum_m \sum_n I(m,n)K(i-m, j-n)$$

- Эквивалентно (коммутативность):

$$S(i,j) = (I * K)(i,j) \equiv \sum_m \sum_n I(i-m, j-n)K(m,n)$$

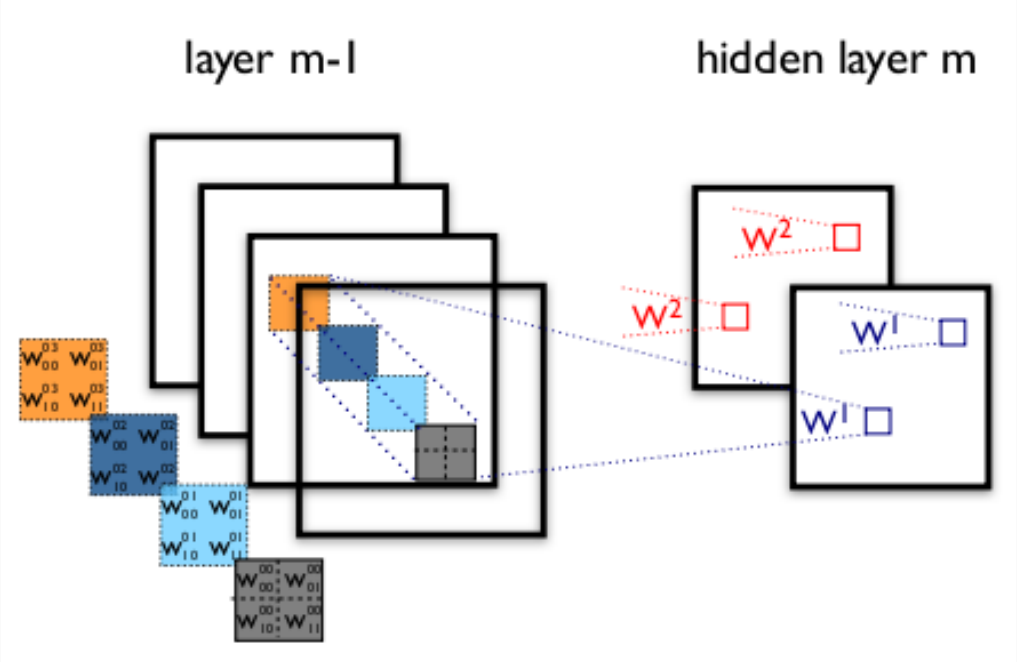
Свертка изображения с ядром



Стандартная реализация сверточного слоя

- $V_{i,j,k}$ — пиксель входа (канал i, строка j, столбец k)
- $K_{i,j,k,l}$ — элемент ядра (входной канал i, выходной канал j, строка k, столбец l)
- Базовое соотношение для свертки

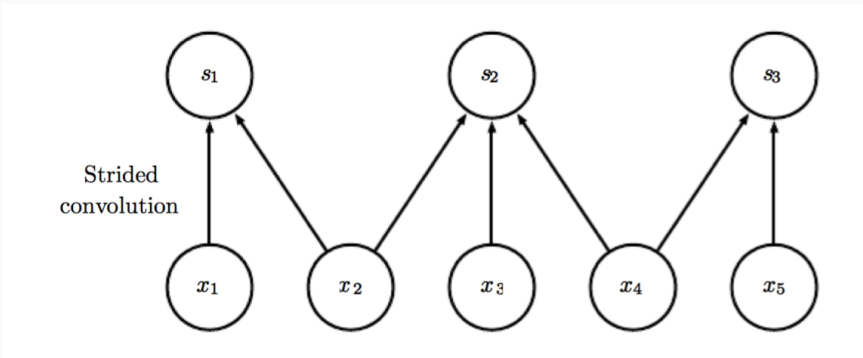
$$Z_{i,j,k} = \sum_{l,m,n} V_{l,j+m-1,k+n-1} K_{i,l,m,n}$$

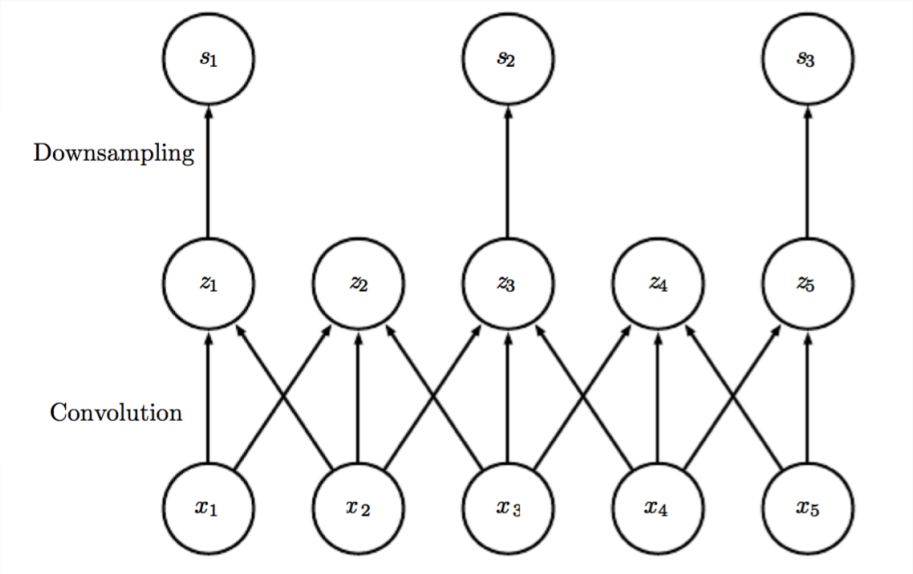


Сверточный слой с шагом s

- Уменьшение вычислительных затрат в s раз
- Шагающая (strided) свертка

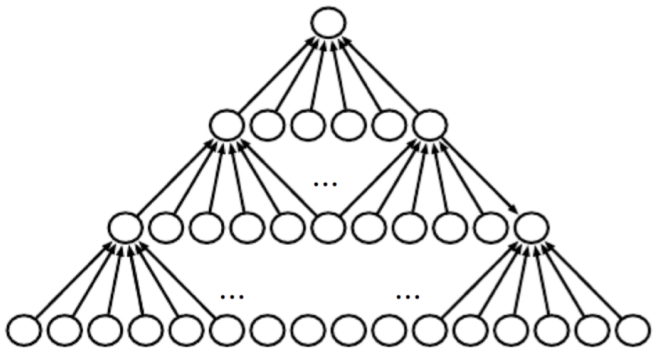
$$Z_{i,j,k} = \sum_{l,m,n} V_{l,(j-1)\times s+m,(k-1)\times s+n} K_{i,l,m,n}$$

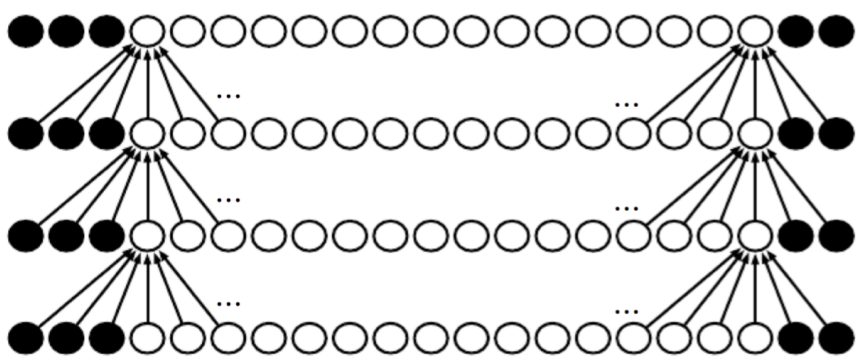




Неявное заполнение нулями

- Выход каждого слоя уменьшается на $k - 1$ пиксель после каждой свертки (k - размер ядра)
- Неявное заполнение нулями (zero-padding) — способ управлять шириной ядра и размером выхода независимо:
  - нет заполнения
  - заполнение с сохранением размера выхода
  - заполнение с равным посещением пикселей



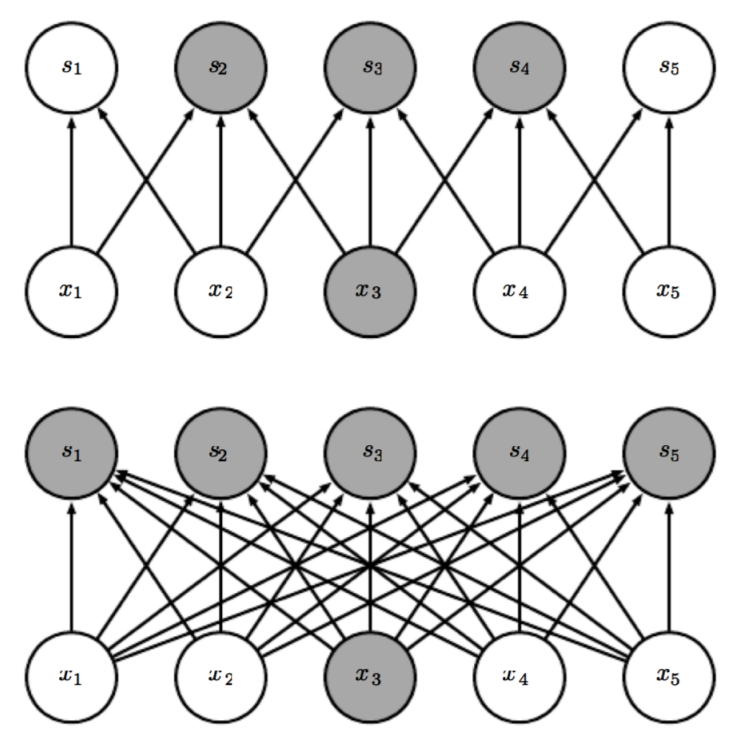


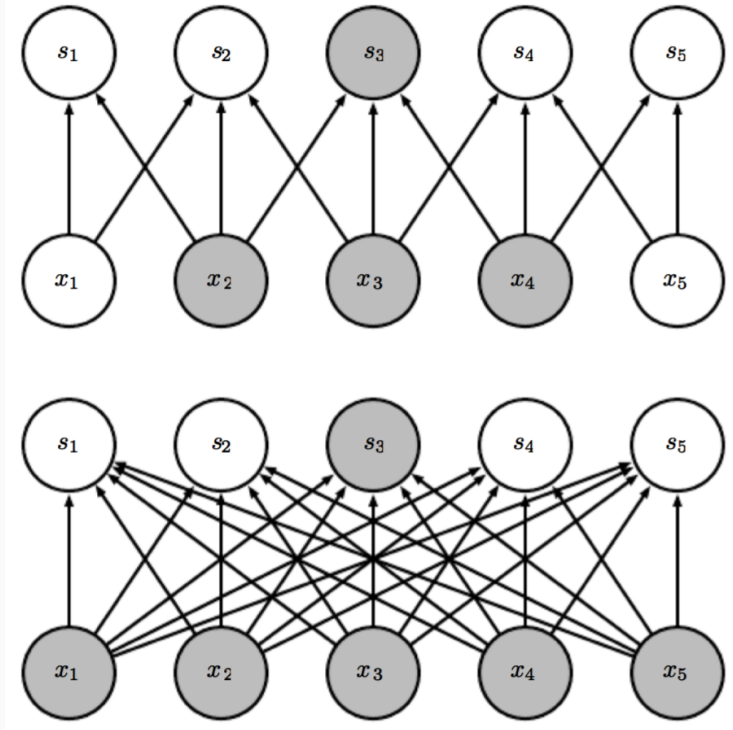
Зачем нужны свертки?

- Разреженность взаимодействия нейронов
- Разделяемые (общие) параметры
- Эквивариантность представления
- + Работа со входом различного размера

Разреженные веса между нейронами

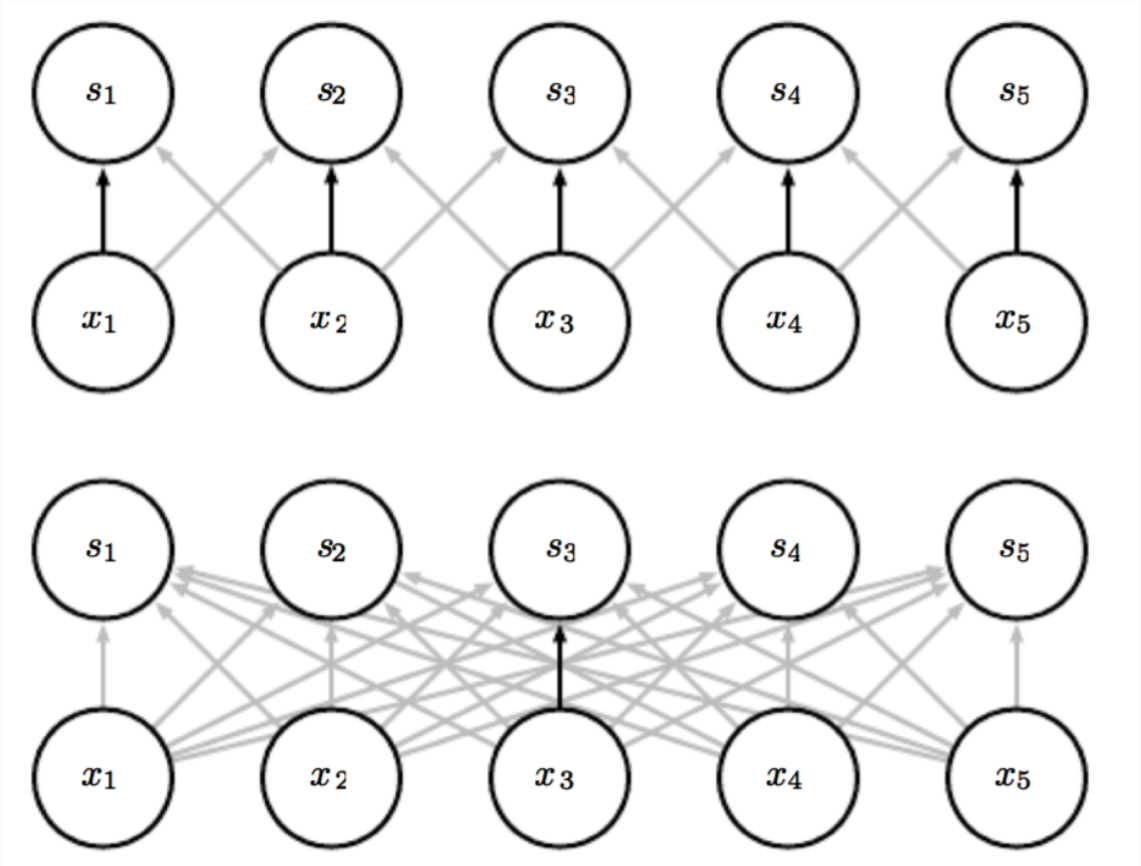
- Ядро имеет меньший размер, чем изображение
- Извлечение локальных признаков
- Храним меньше параметров, статистическое поведение модели лучше, вычисляем меньше





Разделяемые (совместные) параметры

- Один набор параметров (ядро) используется в различных частях модели
- Каждый параметр используется многократно



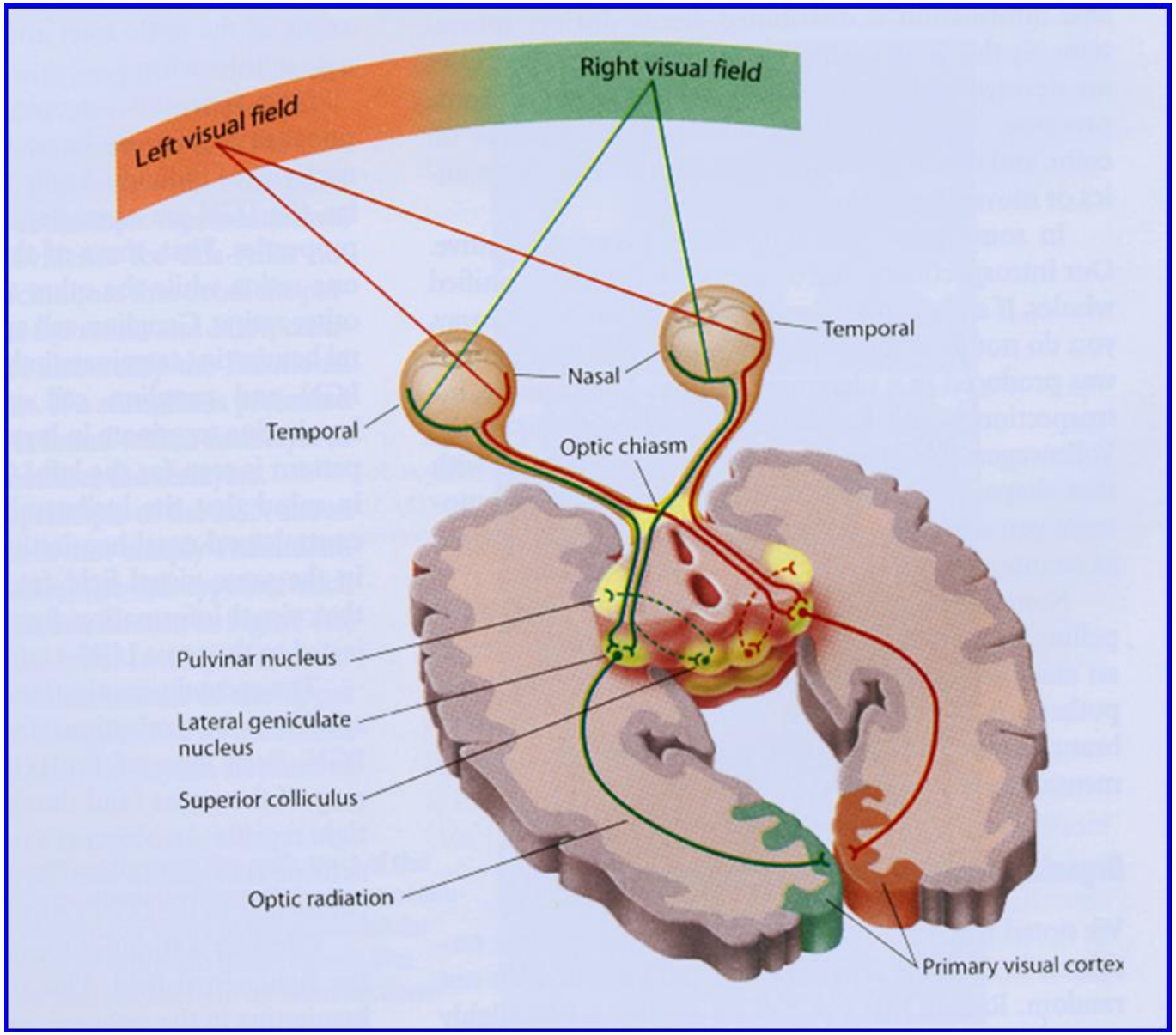
Эквивариантность к переносам

- Эквивариантность: если изменяется вход, то выход изменяется так же
- Функция $f(x)$ эквивариантна к функции $g(y)$, если $f(g(y)) = g(f(x))$
- Свертка $f(I)$ эквивариантна к параллельному переносу:
  - $I' = g(I) \implies I'(x, y) = I(x - 1, y)$
  - Тогда результат свертки $f(I') = g(f(I))$

[ConvNetJS demo: training on CIFAR-10]

http://cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html

Зрительная система млекопитающих



Зрительная система млекопитающих

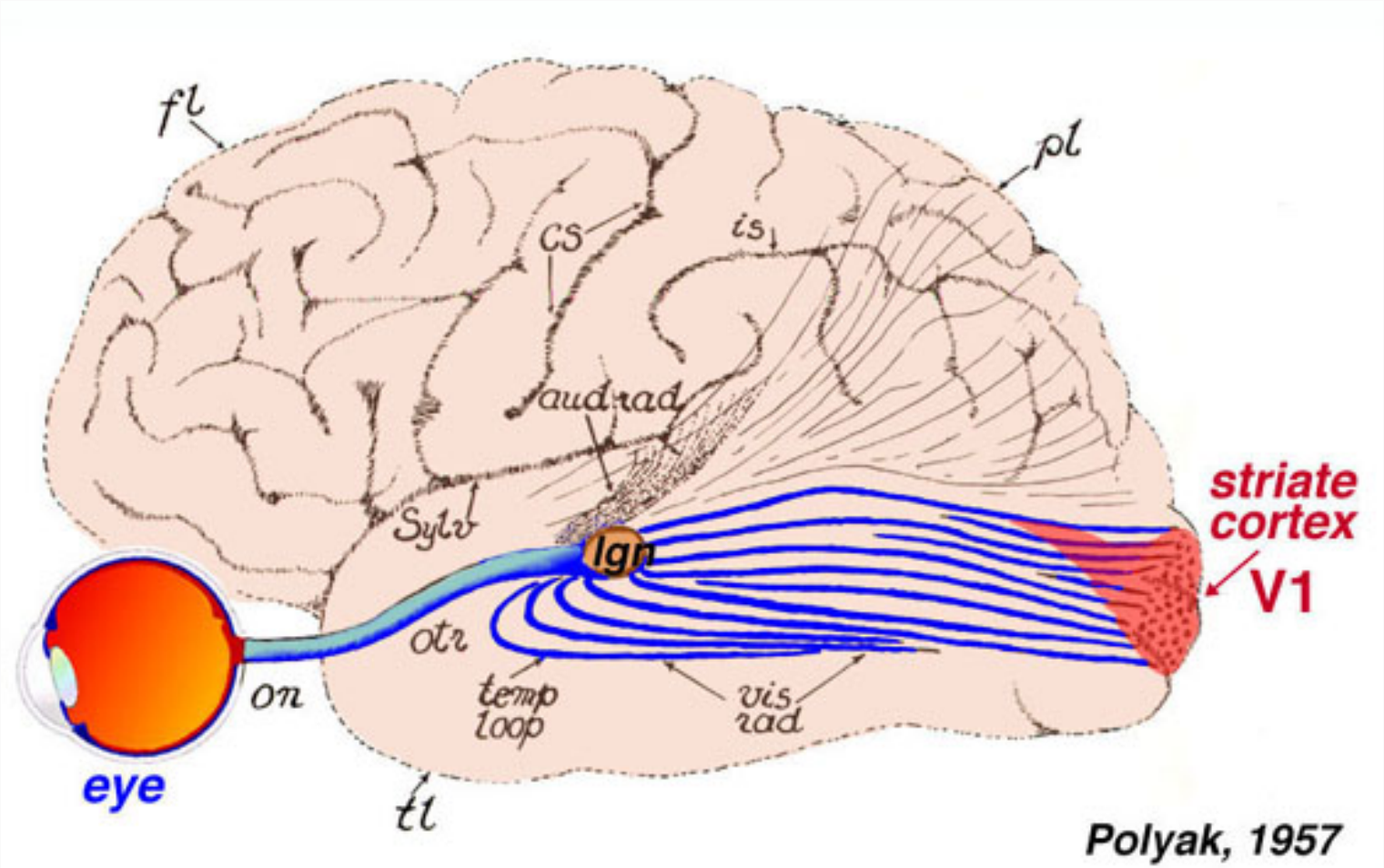
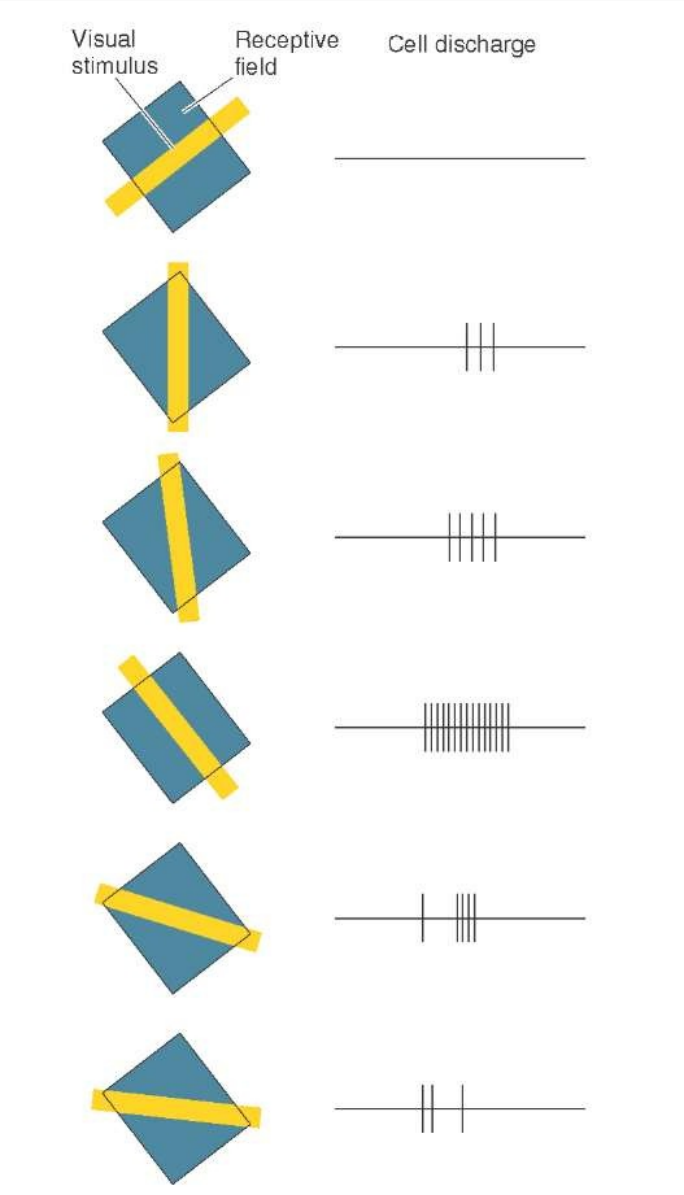
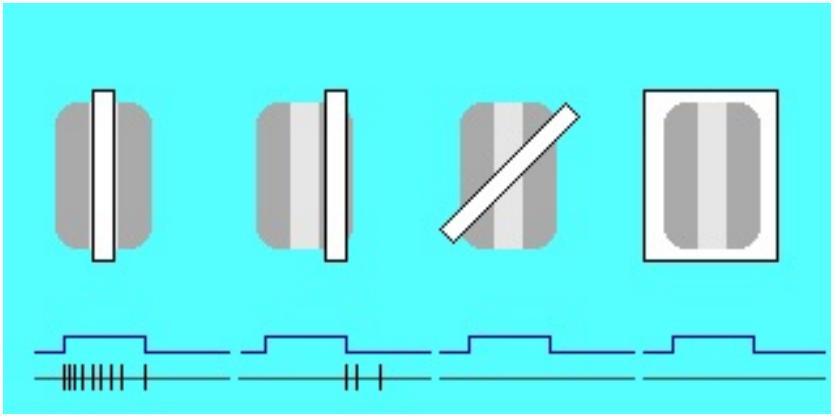


Figure 8. Visual input to the brain goes from eye to LGN and then to primary visual cortex, or area V1, which is located in the posterior of the occipital lobe.
Adapted from Polyak (1957).

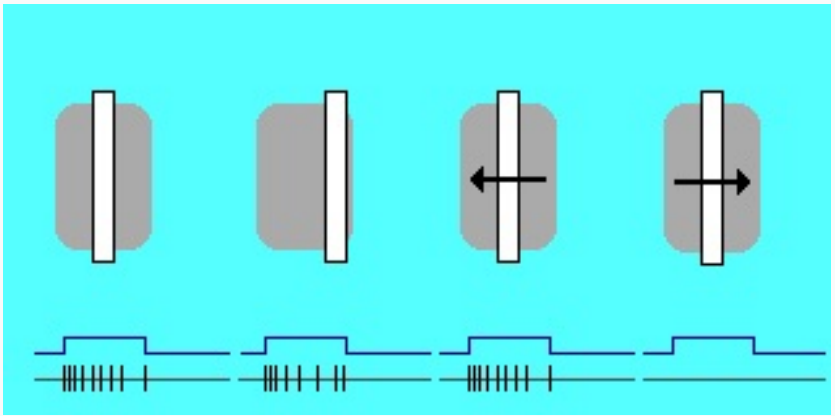
Клетки зрительной коры — детекторы признаков!



Простые клетки:

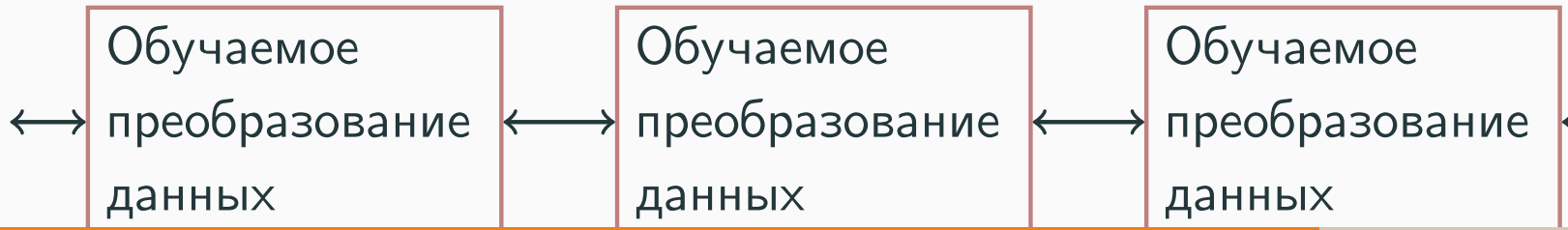


Сложные клетки:



Иерархия представлений

- Сетчатка - LGN - V1 - V2 - V4 - IT - ...
- Иерархия представлений с увеличением уровня абстракции
- Каждый этап — это обучаемое преобразование признаков
- Распознавание изображений
  - Пиксель → ребро → банк фильтров → шаблон → часть → объект
- Текст
  - Символ → слово → группа слов → фраза → предложение → текст
- Речь
  - Отсчет → полоса спектра → звук → фонема → слово



Архитектура LeNet [LeCun et al., 1998]

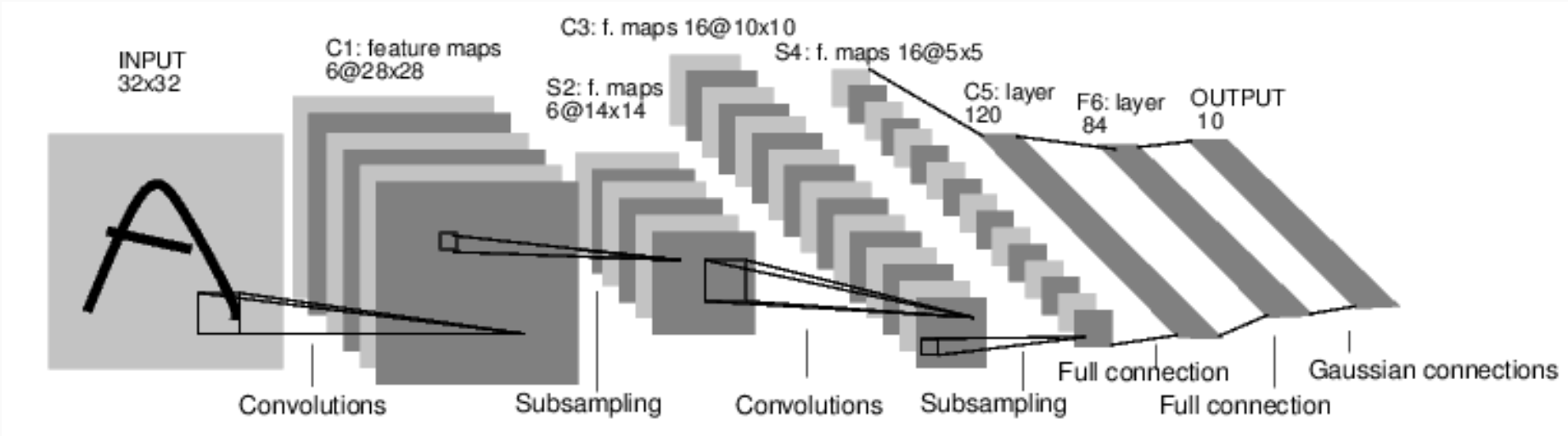
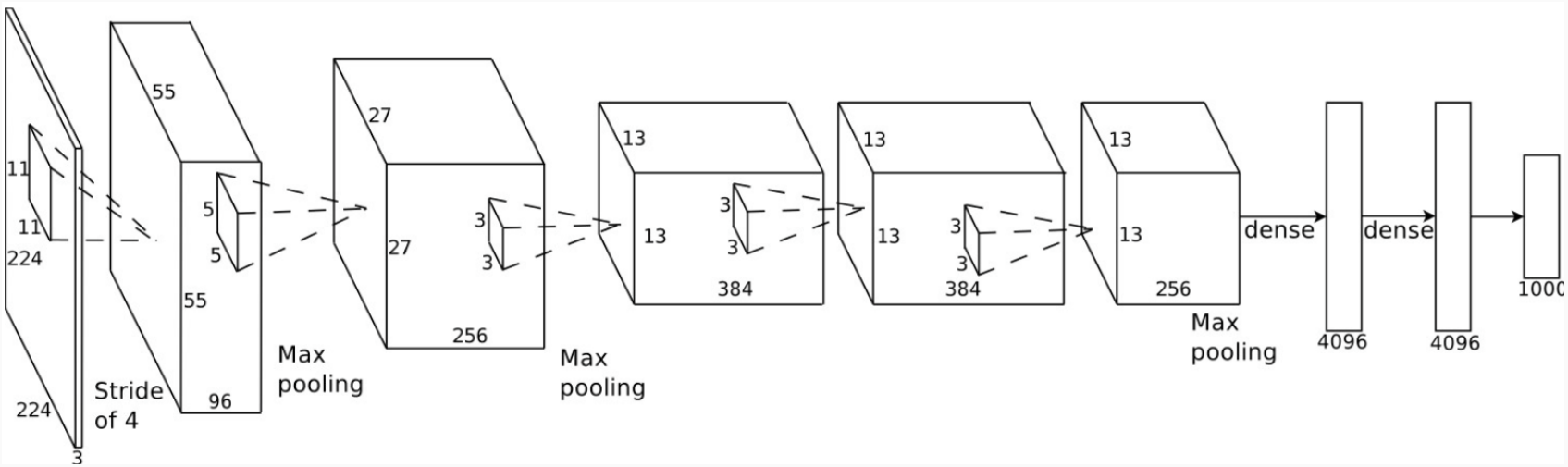


Рис. 1: LeNet-5 [LeCun et al., 1998]

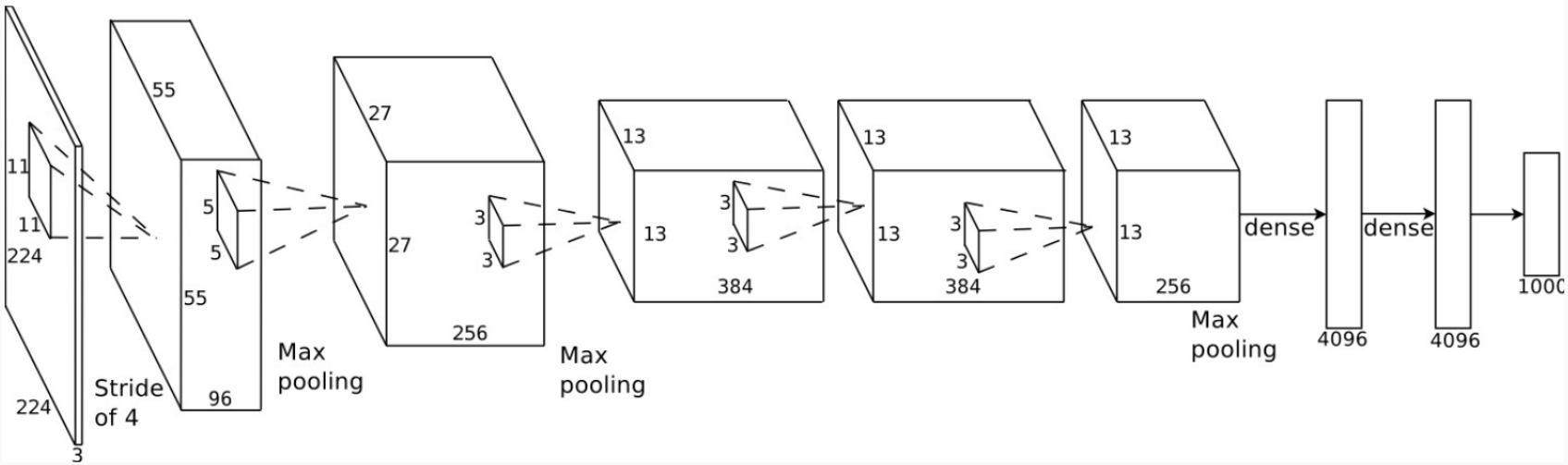
- CONV-POOL-CONV-POOL-CONV-FC
- Вход: 32×32
- Фильтры: 5×5, шаг 1

Архитектура сети AlexNet [Krizhevsky et al., 2012]



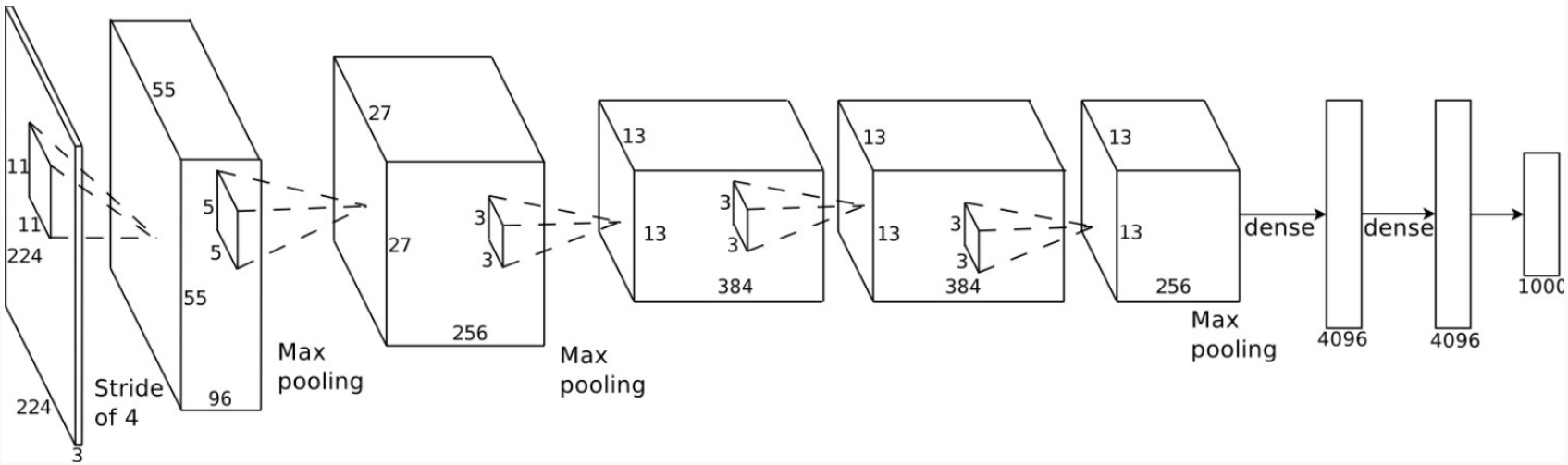
- CONV1: 96 фильтров 11×11, шаг 4
  - размер выхода?
  - число параметров?

Архитектура сети AlexNet [Krizhevsky et al., 2012]



- CONV1: 96 фильтров 11×11, шаг 4
  - размер выхода? $[55 \times 55 \times 96]$
  - число параметров? $96 \times 11 \times 11 \times 3 = 35K$
- POOL1: фильтры 3×3, шаг 2
  - размер выхода?
  - число параметров?

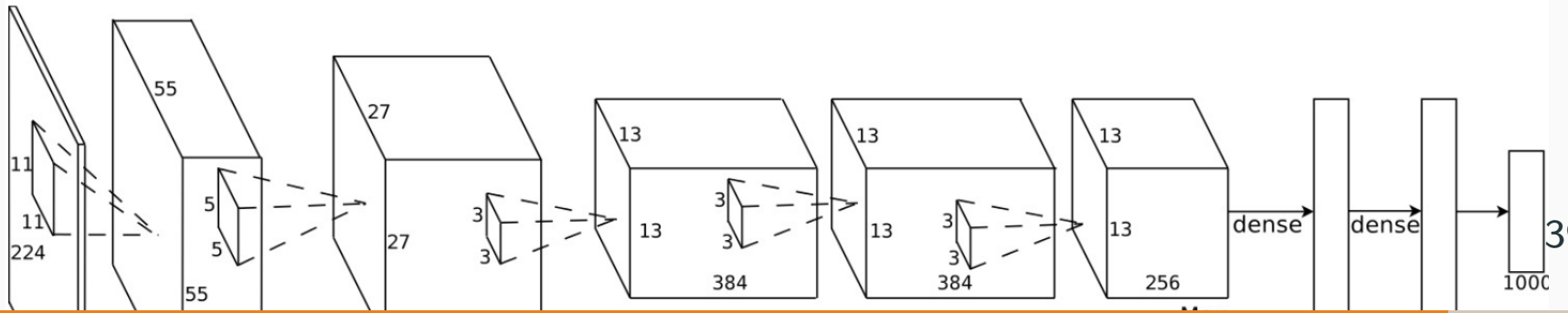
Архитектура сети AlexNet [Krizhevsky et al., 2012]



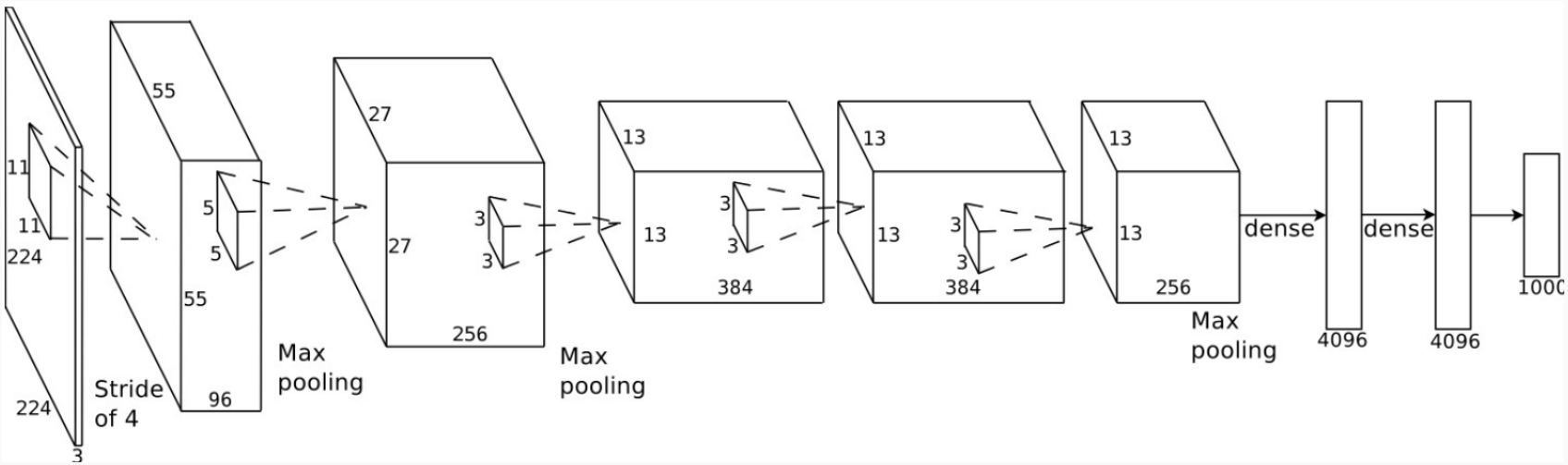
- CONV1: 96 фильтров 11×11, шаг 4
  - размер выхода? $[55 \times 55 \times 96]$
  - число параметров? $96 \times 11 \times 11 \times 3 = 35K$
- POOL1: фильтры 3×3, шаг 2
  - размер выхода? $[27 \times 27 \times 96]$
  - число параметров? 0!

Архитектура сети AlexNet [Krizhevsky et al., 2012]

```
[227x227x3] INPUT
[55x55x96]  CONV1: 96 11x11 filters at stride 4, pad 0
[27x27x96]  MAX POOL1: 3x3 filters at stride 2
[27x27x96]  NORM1: Normalization layer
[27x27x256] CONV2: 256 5x5 filters at stride 1, pad 2
[13x13x256] MAX POOL2: 3x3 filters at stride 2
[13x13x256] NORM2: Normalization layer
[13x13x384] CONV3: 384 3x3 filters at stride 1, pad 1
[13x13x384] CONV4: 384 3x3 filters at stride 1, pad 1
[13x13x256] CONV5: 256 3x3 filters at stride 1, pad 1
[6x6x256]   MAX POOL3: 3x3 filters at stride 2
[4096]      FC6: 4096 neurons
[4096]      FC7: 4096 neurons
[1000]      FC8: 1000 neurons (class scores)
```

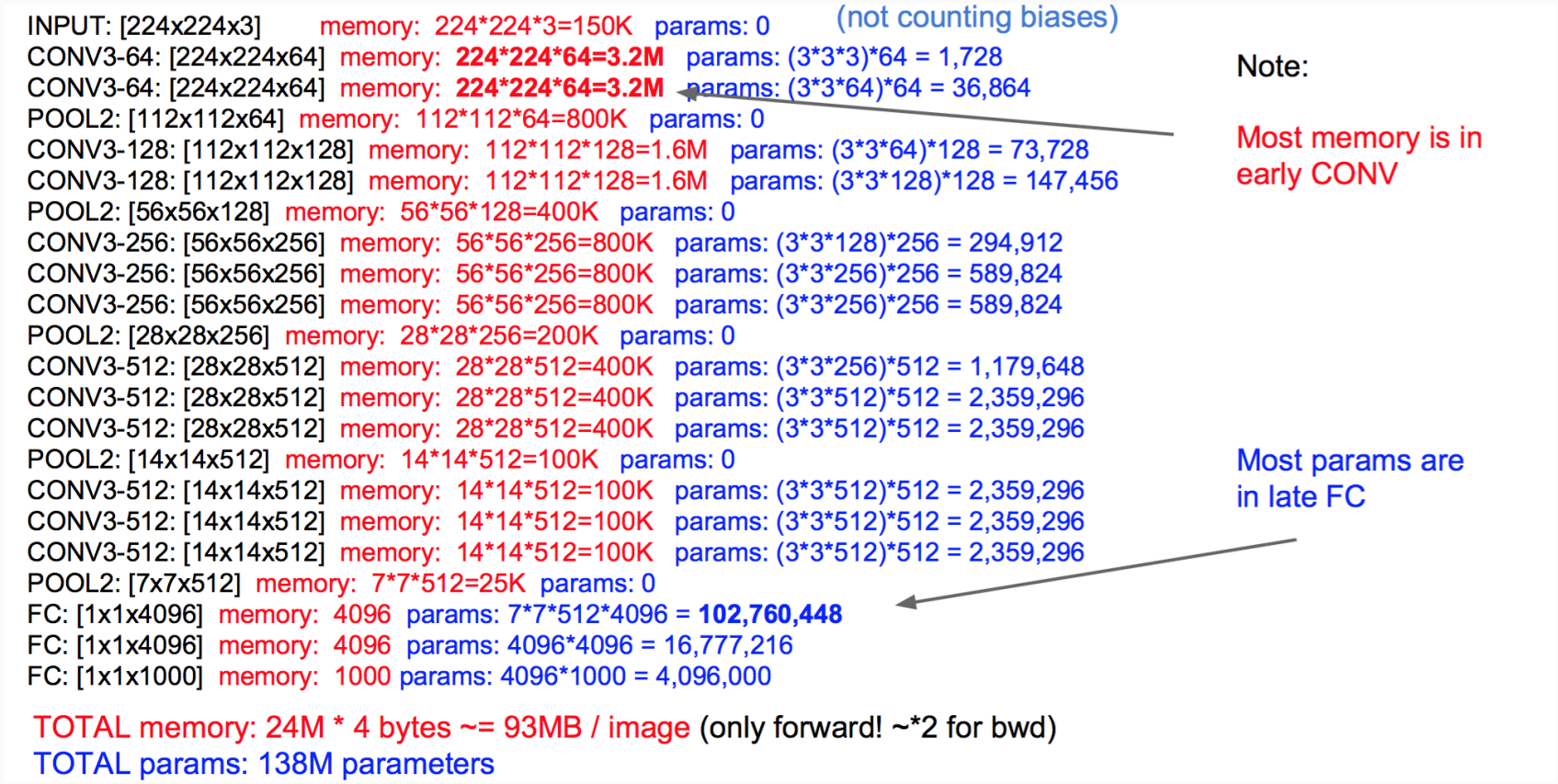


Архитектура сети AlexNet [Krizhevsky et al., 2012]



- 60 млн параметров
- 27 Гб данных на диске
- Обучается 1 неделю
- 2 Гб памяти на каждом GPU
- 5 Гб системной памяти
- ILSVRC 2012: 40.7% (Top-1), 18.2% (Top-5)

VGGNet [Simonyan and Zisserman, 2014]



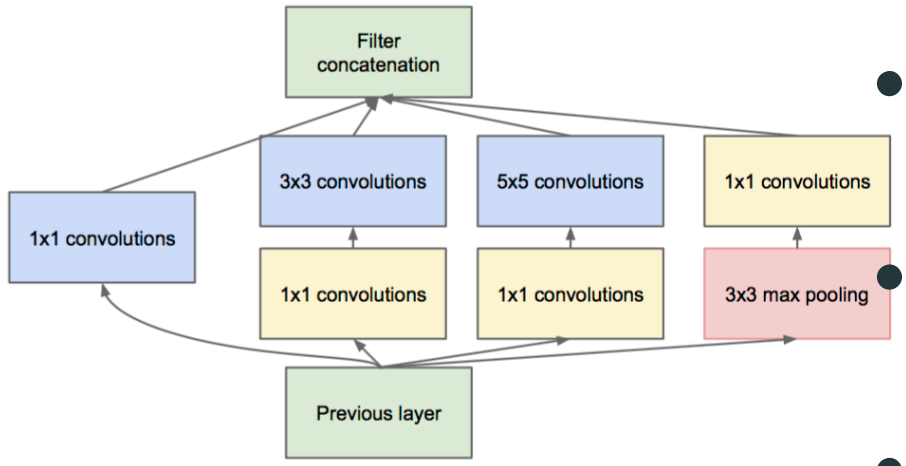
(Слайд Andrej Karpathy)

- Свертки только размера 3×3, шаг 1
- Пулинг только 2×2, шаг 2
- ILSVRC 2012: 25.5% (Top-1), 8.0% (Top-5)

Архитектура Inception [Szegedy et al., 2014]

- Два способа увеличения качества сети — увеличение глубины и увеличение ширины
- Опасность №1: повышение вероятности переобучения
- Опасность №2: неэффективное использование вычислительных ресурсов
- Разреженная структура сверток неэффективна в вычислительном смысле
- Идея: использование корреляционной структуры активаций предыдущих слоев

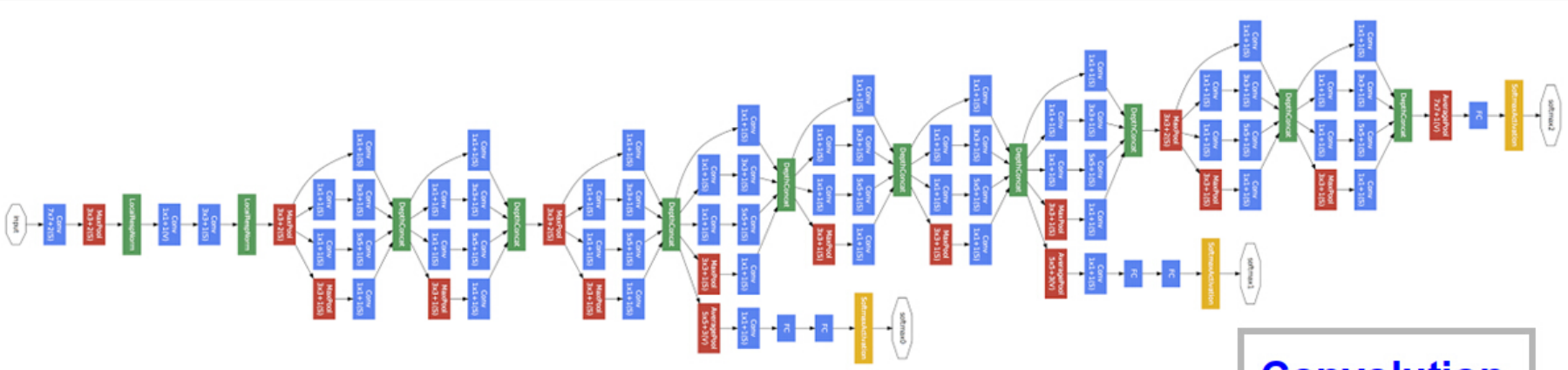
Архитектура Inception [Szegedy et al., 2014]



(b) Inception module with dimensionality reduction

По сравнению с AlexNet:

- в 12 раз меньше параметров
- в 2 раза больше вычислений
- ILSVRC 2014: 6.67% (vs. 16.4%)



Convolution
Pooling
Softmax

Архитектура ResNet [He et al., 2015]

[Оригинальные слайды]

Сверточные сети: итог

- Использование слоев conv, pool, relu, fc
- Современный тренд: меньший размер фильтра, большая глубина архитектуры
- Тренд к исчезновению полносвязных слоев
- Типичная архитектура:
  [(CONV-RELU)*N-POOL?]*M-(FC-RELU)*K,SOFTMAX
- Однако более новые архитектуры (GoogLeNet, ResNet) меняют парадигму