

# Свертка.

Блуменау Марк, магистратура ИИ

# Почему полносвязные слои не решают всё?

Возьмем MNIST, 28 x 28 пикселей -> на вход 784 числа

На выходе 10 чисел (по кол-ву цифр)

Пусть в скрытом слое 1000 параметров, тогда итого:

$(784+1)*1000 + (1000 + 1) * 10 = 785,000 + 10,100 = \text{очень много}$

А что значит очень много? Переобучиться не так уж сложно...

# Но ведь есть дропаут!

Да в целом и без него можно уговорить

Table 1: Error rates on MNIST test set.

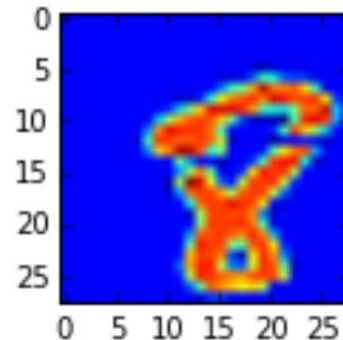
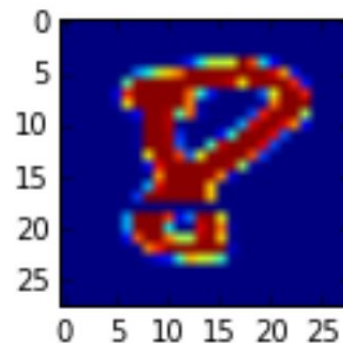
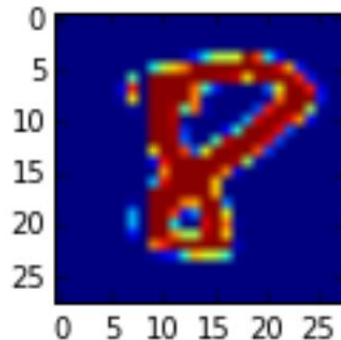
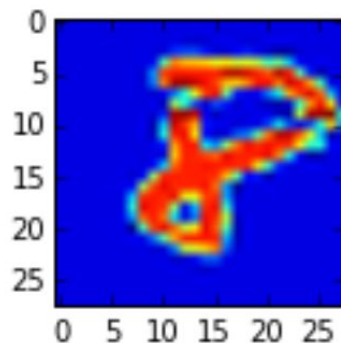
| ID | architecture<br>(number of neurons in each layer) | test error for<br>best validation [%] | best test<br>error [%] | simulation<br>time [h] | weights<br>[milions] |
|----|---|---------------------------------------|------------------------|------------------------|----------------------|
| 1  | 1000, 500, 10                                     | <b>0.49</b>                           | 0.44                   | 23.4                   | 1.34                 |
| 2  | 1500, 1000, 500, 10                               | <b>0.46</b>                           | 0.40                   | 44.2                   | 3.26                 |
| 3  | 2000, 1500, 1000, 500, 10                         | <b>0.41</b>                           | 0.39                   | 66.7                   | 6.69                 |
| 4  | 2500, 2000, 1500, 1000, 500, 10                   | <b>0.35</b>                           | 0.32                   | 114.5                  | 12.11                |
| 5  | $9 \times 1000$ , 10                              | <b>0.44</b>                           | 0.43                   | 107.7                  | 8.86                 |

# А если не получится, то будет дропаут, да?

Подумать на досуге: возьмем картинку 224 на 224. А дальше слои 1000, 500 и 100, 10.

Нет. Лучшее лекарство от проблем – убить параметры!

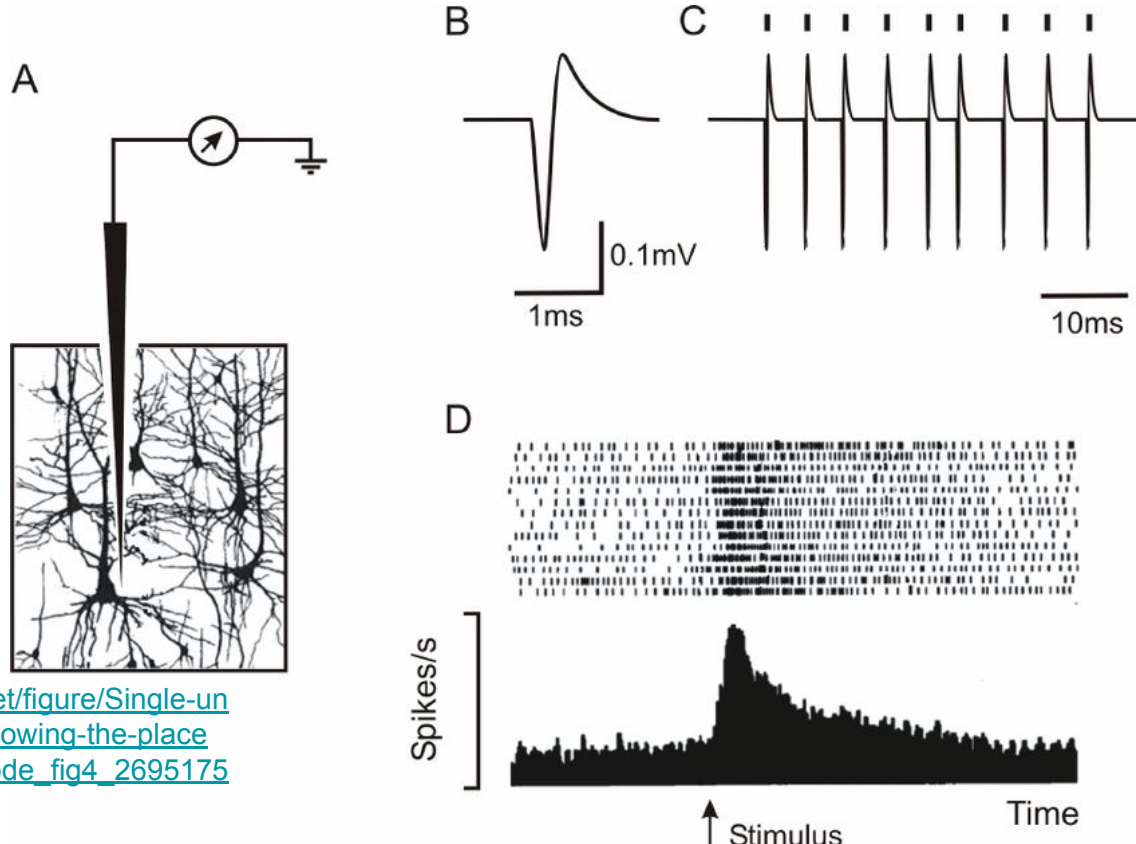
# Специфика картинок



Страшилка на ночь (впечатлительным просьба  
уйти до сообщения в чате, без шуток)

Или нобелевка Hubel и Wiesel по медицине

# Дали людям иголочку...



[https://www.researchgate.net/figure/Single-unit-recording-A-Schematic-showing-the-place-ment-of-the-tip-of-an-electrode\\_fig4\\_269517554](https://www.researchgate.net/figure/Single-unit-recording-A-Schematic-showing-the-place-ment-of-the-tip-of-an-electrode_fig4_269517554)

Вот это “нейронки” якобы пытаются повторить

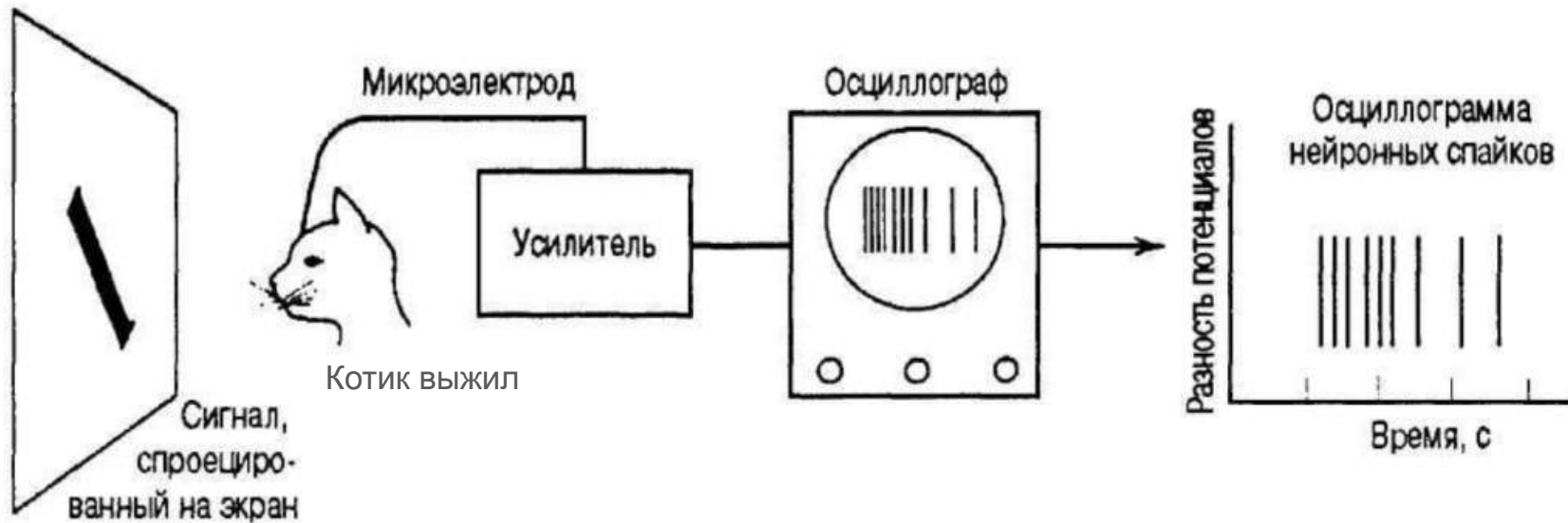
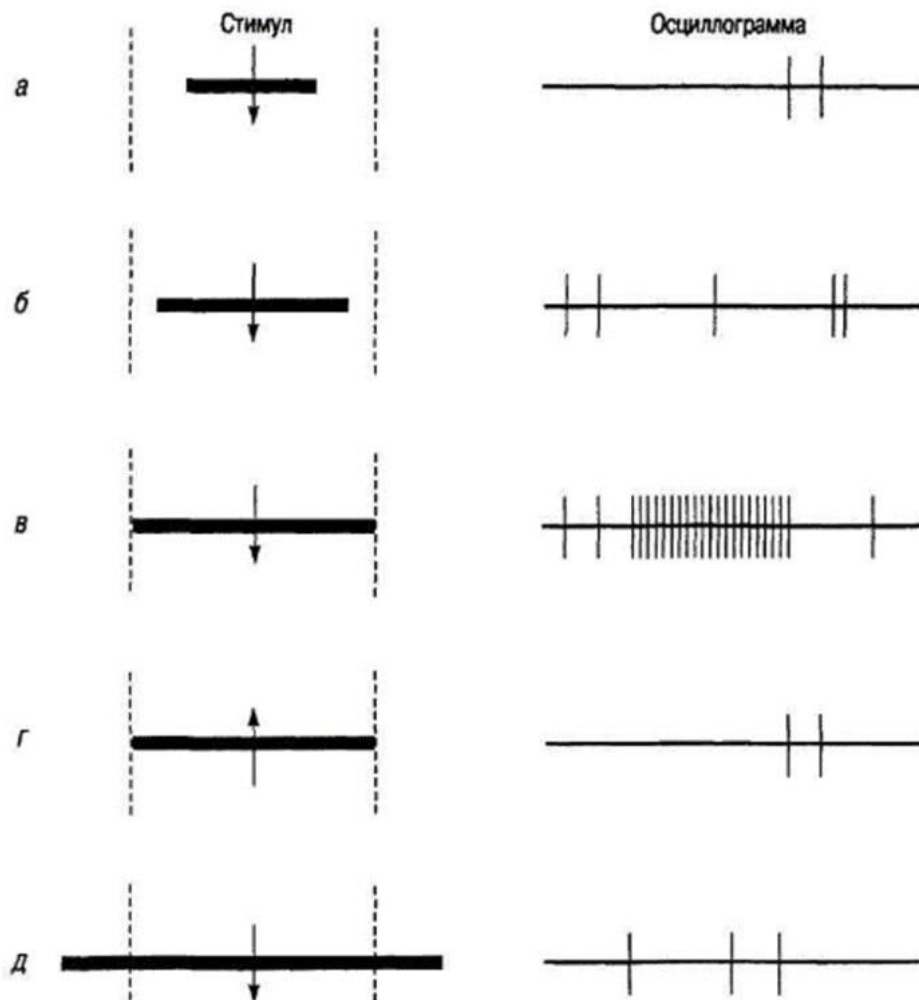
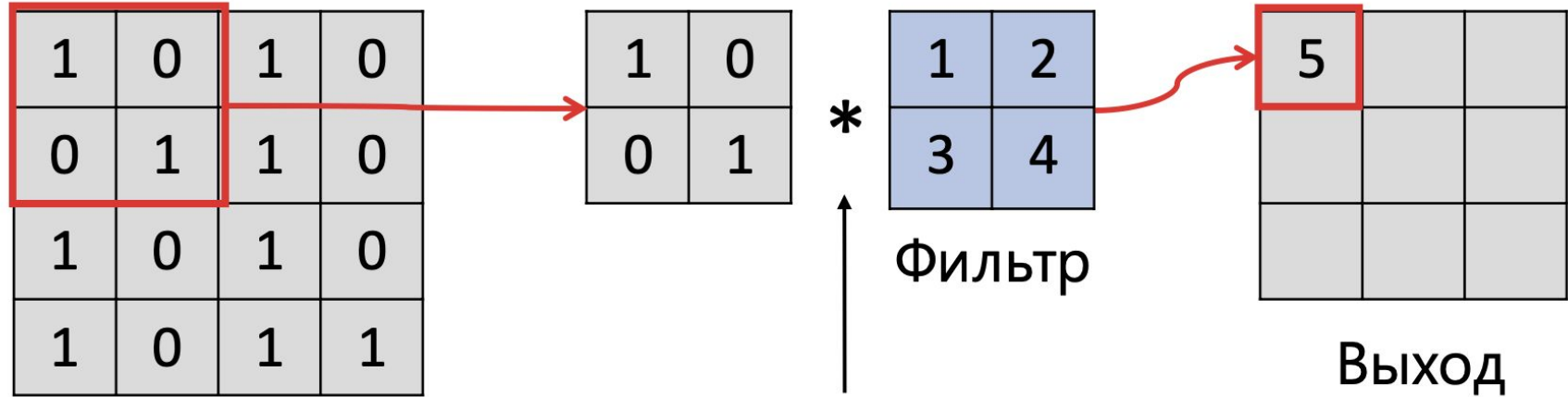


Рис. 3.26. Оборудование для экспериментального обнаружения и изучения рецептивных полей



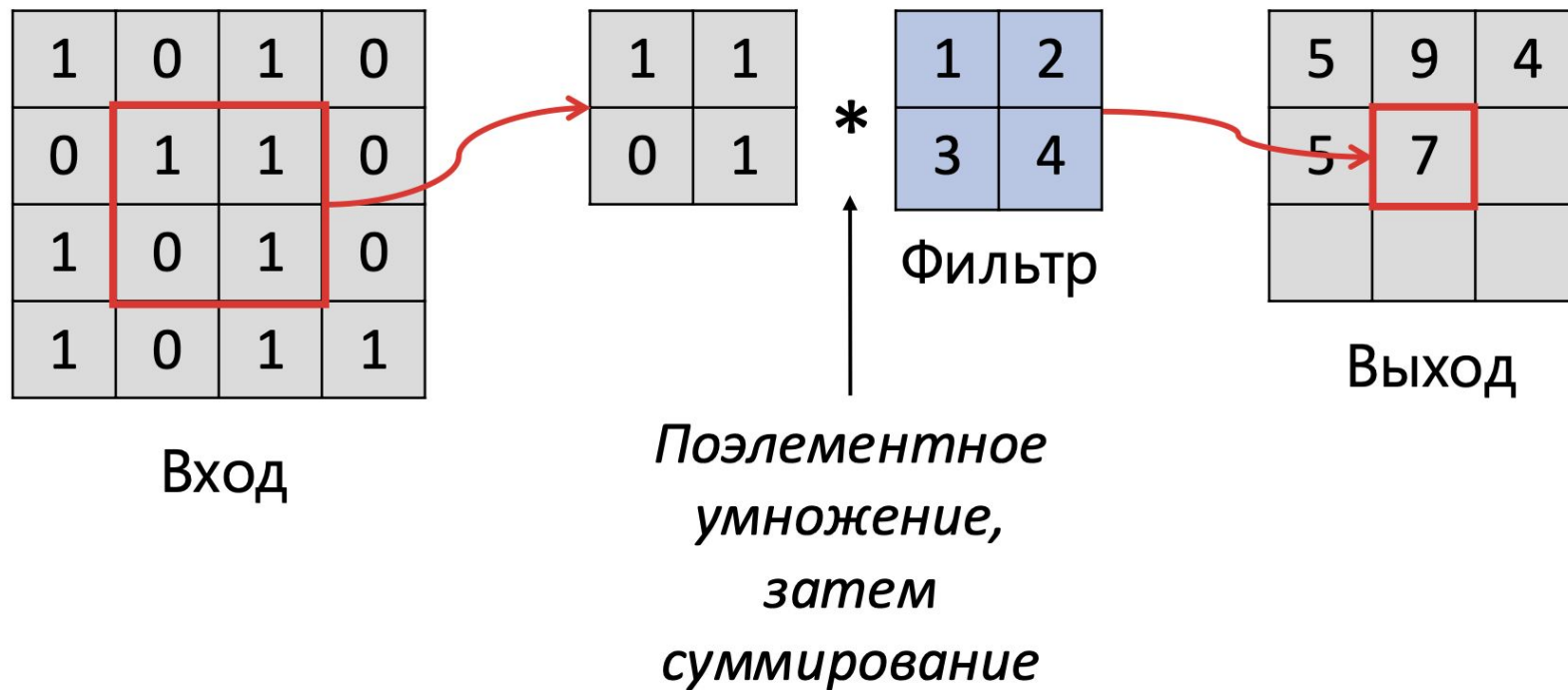


Давайте теперь соберем “похожее”

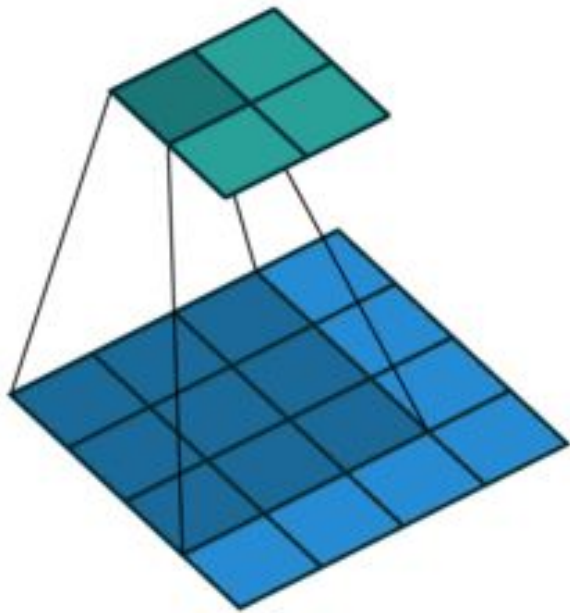


*Поэлементное  
умножение,  
затем  
суммирование*

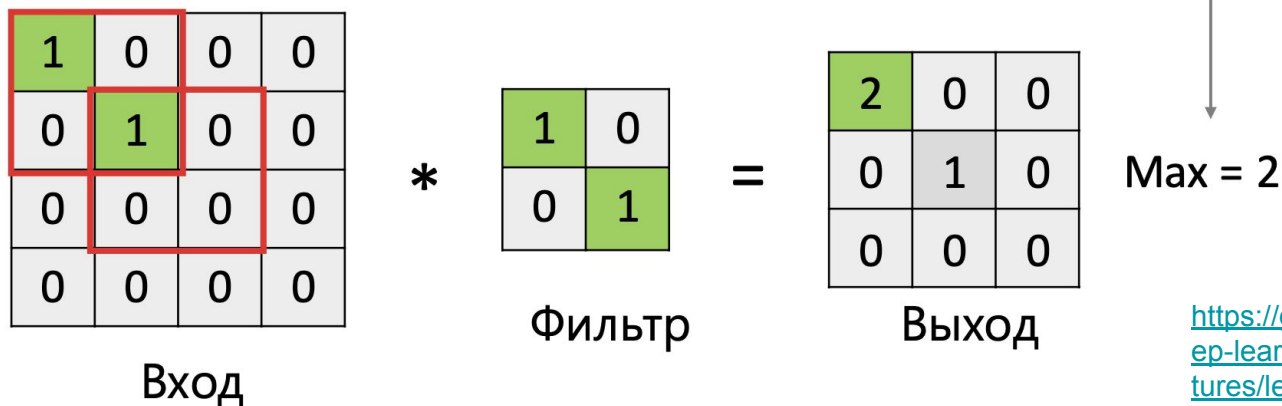
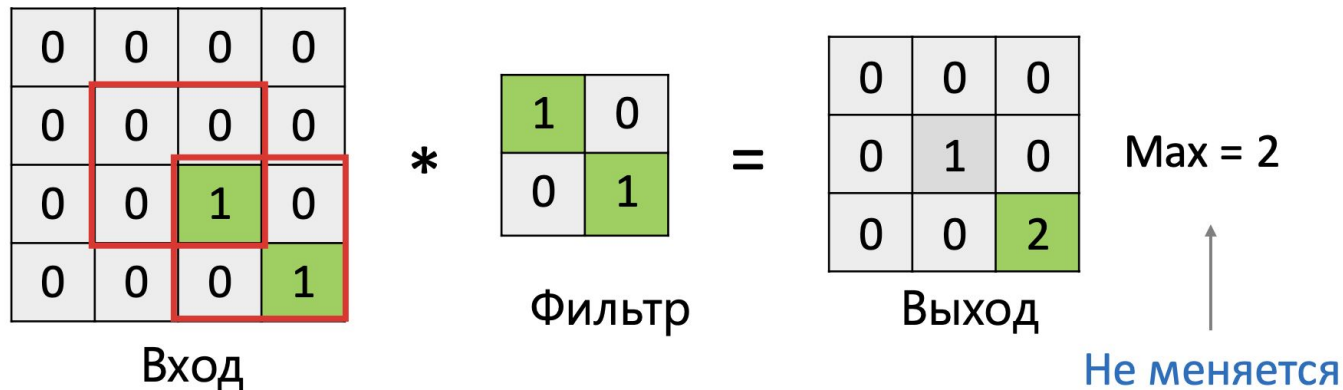
# Повторим операцию много раз



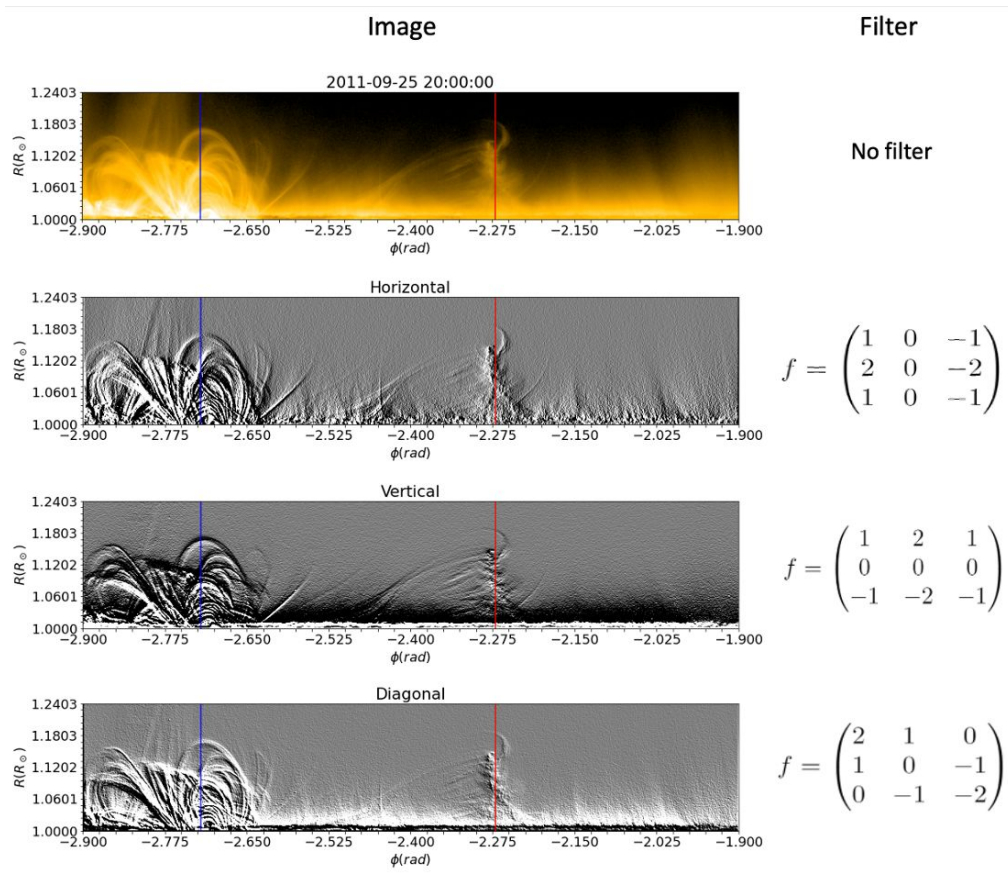
# Conv2d



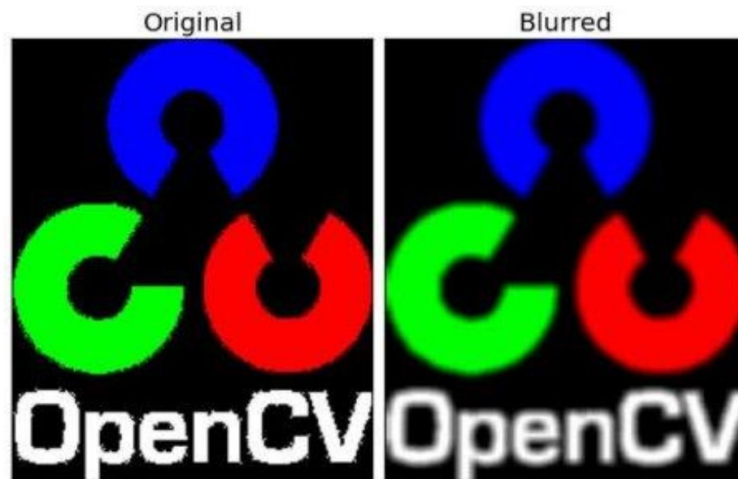
# Максимум свертки инвариантен к сдвигам



# Вы только что придумали велосипед



# OpenCV для классики



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

Делай раз

$$\text{Im}^{out}(x, y) = \sum_{i=-d}^d \sum_{j=-d}^d (K(i, j) \text{Im}^{in}(x + i, y + j) + b)$$



Но у нас обычно хотя бы RedGreenBlue

$$\text{Im}^{out}(x, y) = \sum_{i=-d}^d \sum_{j=-d}^d \sum_{c=1}^C (K(i, j, c) \text{Im}^{in}(x + i, y + j, c) + b)$$

Хотим искать много паттернов

$$\text{Im}^{out}(x, y, t) = \sum_{i=-d}^d \sum_{j=-d}^d \sum_{c=1}^C (K_t(i, j, c) \text{Im}^{in}(x + i, y + j, c) + b_t)$$

# А сколько параметров?

В терминах формулы:

$$\text{Im}^{out}(x, y, t) = \sum_{i=-d}^d \sum_{j=-d}^d \sum_{c=1}^C (K_t(i, j, c) \text{Im}^{in}(x + i, y + j, c) + b_t)$$

$$((2d + 1)^2 * C + 1) * T$$

В других терминах: K - размер ядра, как в торче, C - каналы

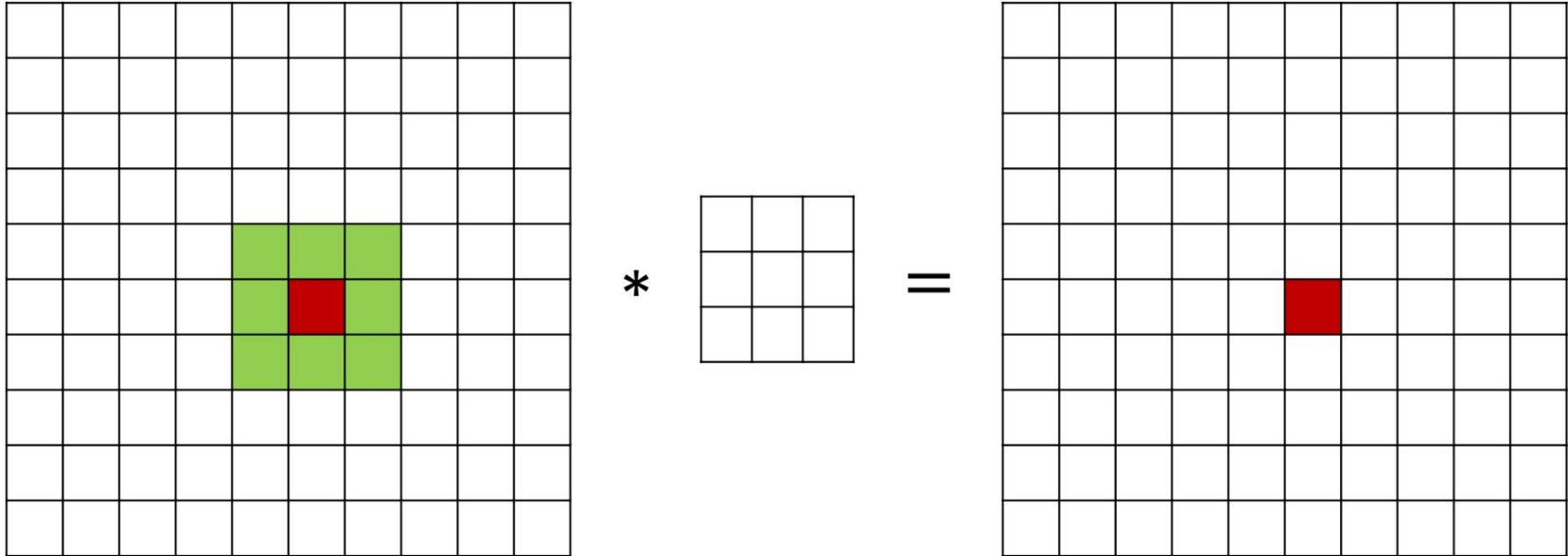
$$K \cdot K \cdot C_{in} \cdot C_{out} + C_{out}$$

В чем подвох? Умножений не сильно мало

$H$ ,  $W$  – выходного тензора

$$H \cdot W \cdot C_{out} \cdot (K \cdot K \cdot C_{in})$$

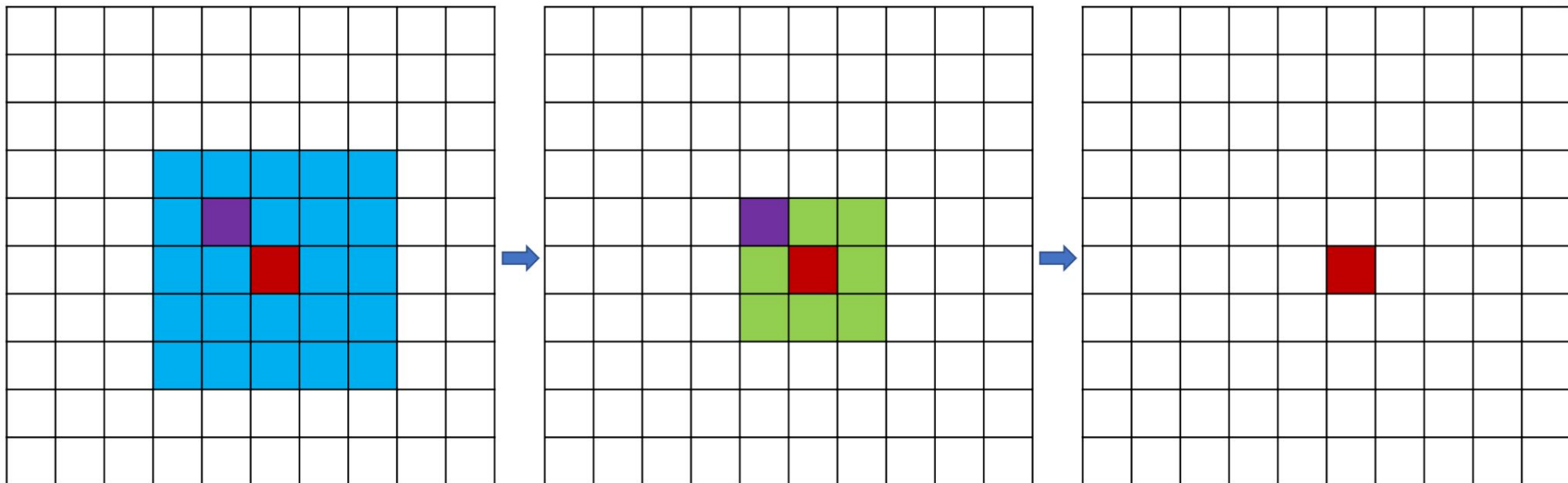
# Поле восприятия (у глаза не так)



Поле восприятия: 3 x 3

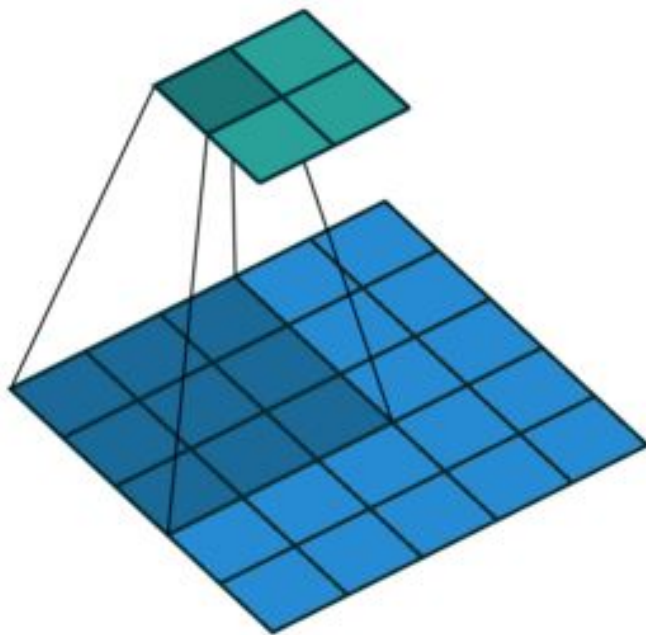
<https://github.com/hse-ds/iad-dep-learning/blob/master/2024/lectures/lecture02-convnets.pdf>

# Receptive field

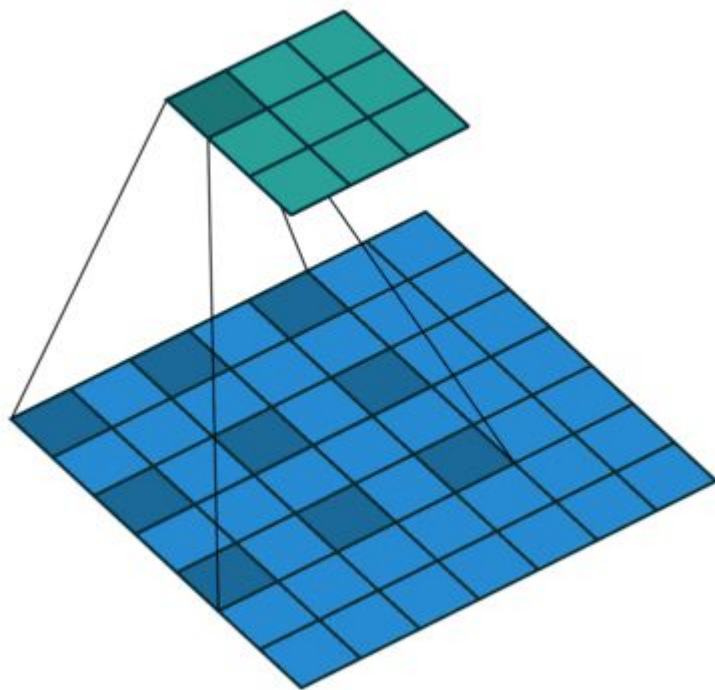


Поле восприятия: 5 x 5

# Stride



# Dilation





## Pooling (max, etc.)

|   |   |   |   |   |   |
|---|---|---|---|---|---|
| 1 | 0 | 2 | 1 | 0 | 0 |
| 0 | 1 | 3 | 2 | 1 | 2 |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |
|   |   |   |   |   |   |



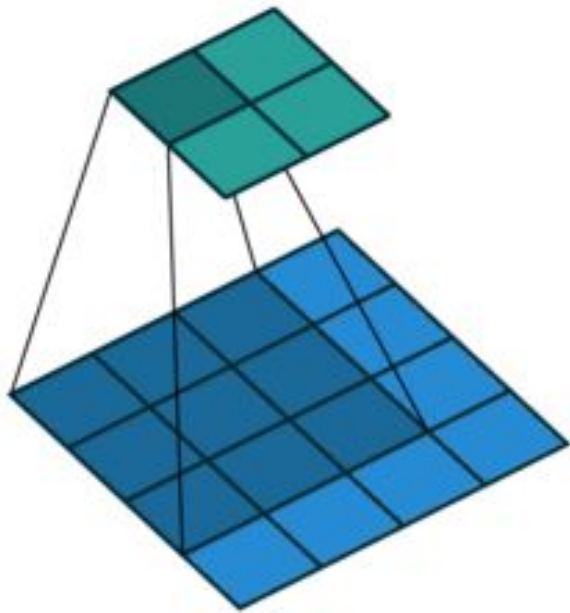
|   |   |   |
|---|---|---|
| 1 | 3 | 2 |
|   |   |   |
|   |   |   |

# В чем профит от всего этого?

Выигрываем в поле восприятия (а как будто хочется на последних слоях поле восприятия полное)

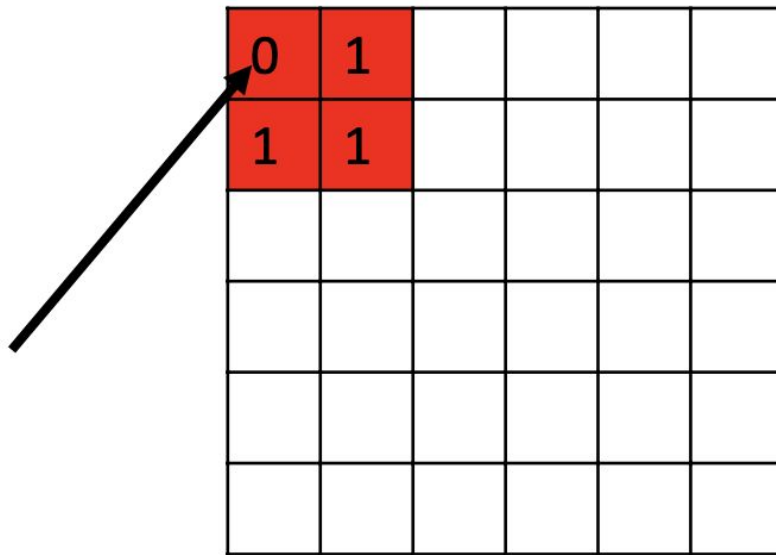
Уменьшаем размер картинки, количество слоев и параметров

А вдруг края важны?



# Обычная свертка не работает при

Не увидим, что фильтр имеет хороший отклик при помещении центра в этот пиксель

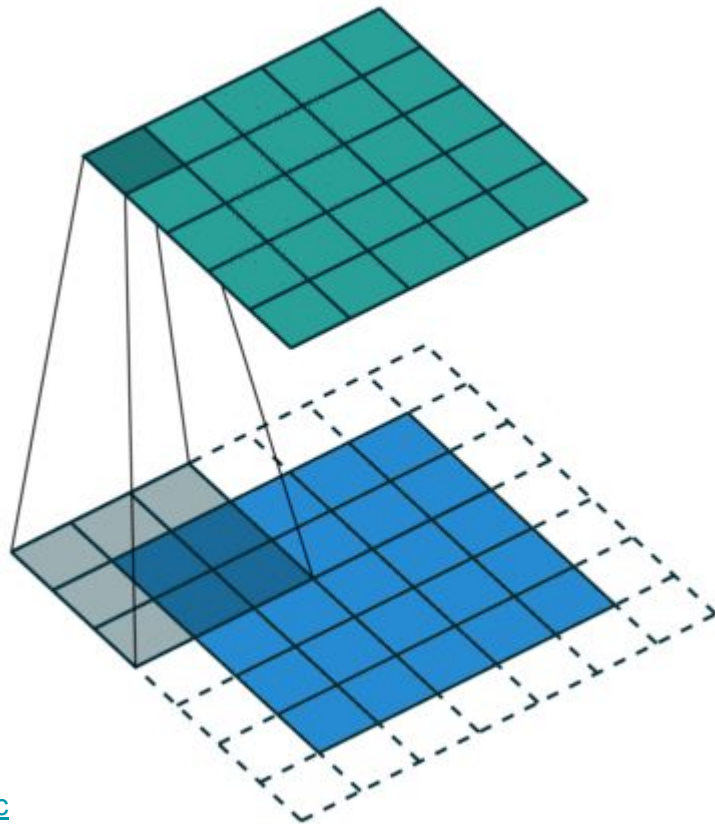


|   |   |  |  |  |  |
|---|---|--|--|--|--|
| 0 | 1 |  |  |  |  |
| 1 | 1 |  |  |  |  |
|   |   |  |  |  |  |
|   |   |  |  |  |  |
|   |   |  |  |  |  |
|   |   |  |  |  |  |

\*

|   |   |   |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 0 | 1 |
| 1 | 1 | 1 |

# Padding



# Zero padding

|   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 |   |   |   |   |   |   | 0 |
| 0 |   |   |   |   |   |   | 0 |
| 0 |   |   |   |   |   |   | 0 |
| 0 |   |   |   |   |   |   | 0 |
| 0 |   |   |   |   |   |   | 0 |
| 0 |   |   |   |   |   |   | 0 |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

\*

|  |  |  |
|--|--|--|
|  |  |  |
|  |  |  |
|  |  |  |

=

|  |  |  |  |  |  |
|--|--|--|--|--|--|
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Reflection

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

\*

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

=

|  |  |  |  |  |  |
|--|--|--|--|--|--|
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

# Replication

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

\*

|  |  |  |  |  |
|--|--|--|--|--|
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

=

|  |  |  |  |  |  |
|--|--|--|--|--|--|
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |



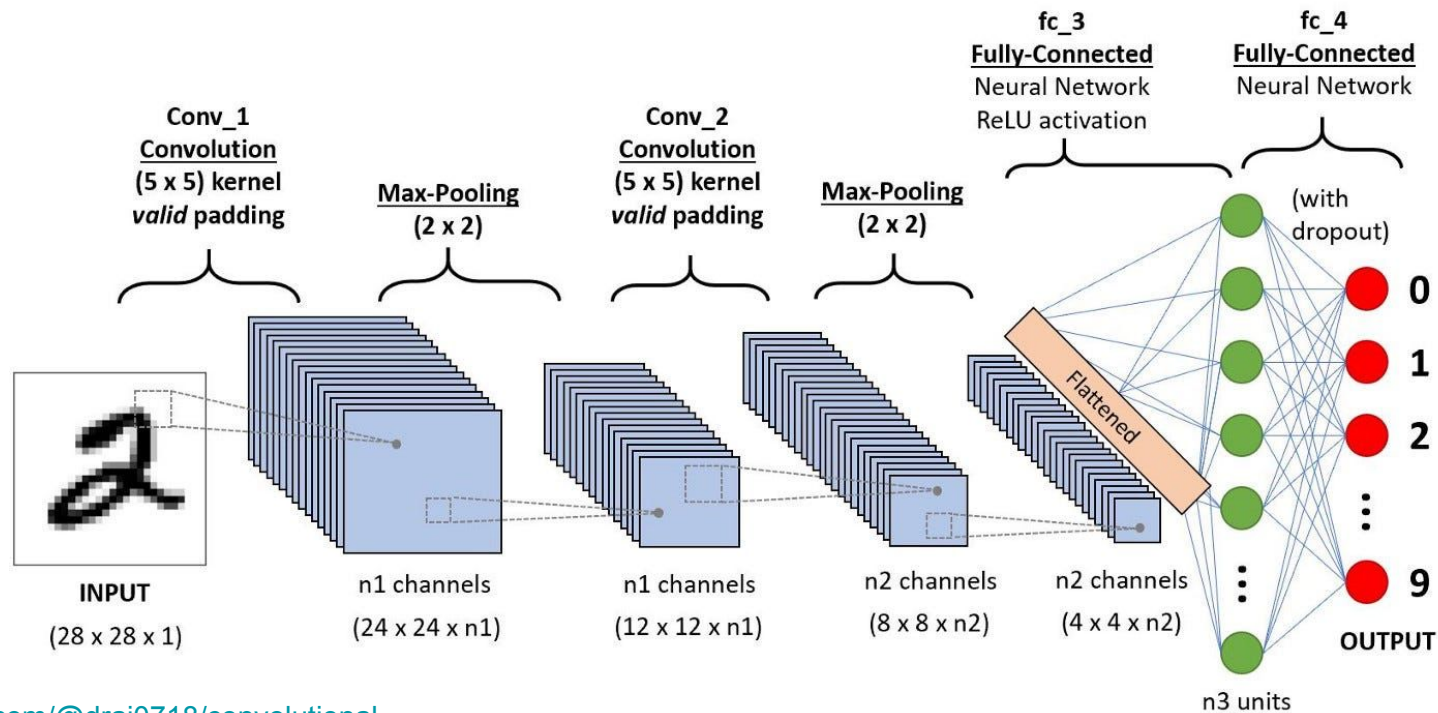
А как посчитать размер выходного “изображения”?

P. S.: Pooling тоже на самом деле так меняет, просто используйте формулу ниже :)

$$O = \frac{W - F + 2P}{S} + 1$$

W — размер входа, F — размер ядра,  
P — паддинг, S — шаг

# Типичная архитектура в CV N лет назад



# Flatten



Где почитать-посмотреть?

<https://cs231n.github.io/>