

Основы глубинного обучения

Лекция 3

Свёрточные сети

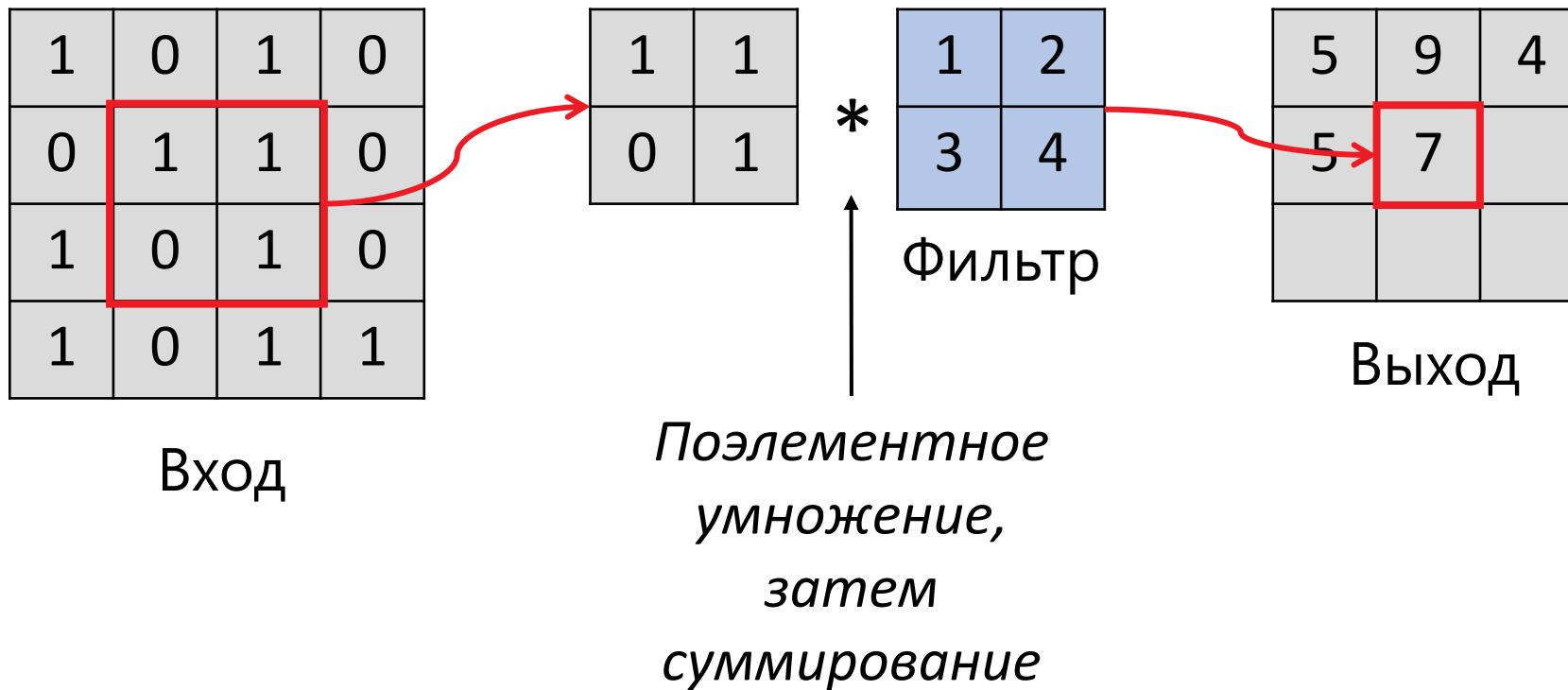
Евгений Соколов

esokolov@hse.ru

НИУ ВШЭ, 2020

Свёртка

Поле восприятия
(receptive field)



Свёртка инвариантна к сдвигам

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

Вход

*

1	0
0	1

=

0	0	0
0	1	0
0	0	2

Фильтр

Выход

Max = 2

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

Вход

*

1	0
0	1

=

2	0	0
0	1	0
0	0	0

Фильтр

Выход

Не меняется

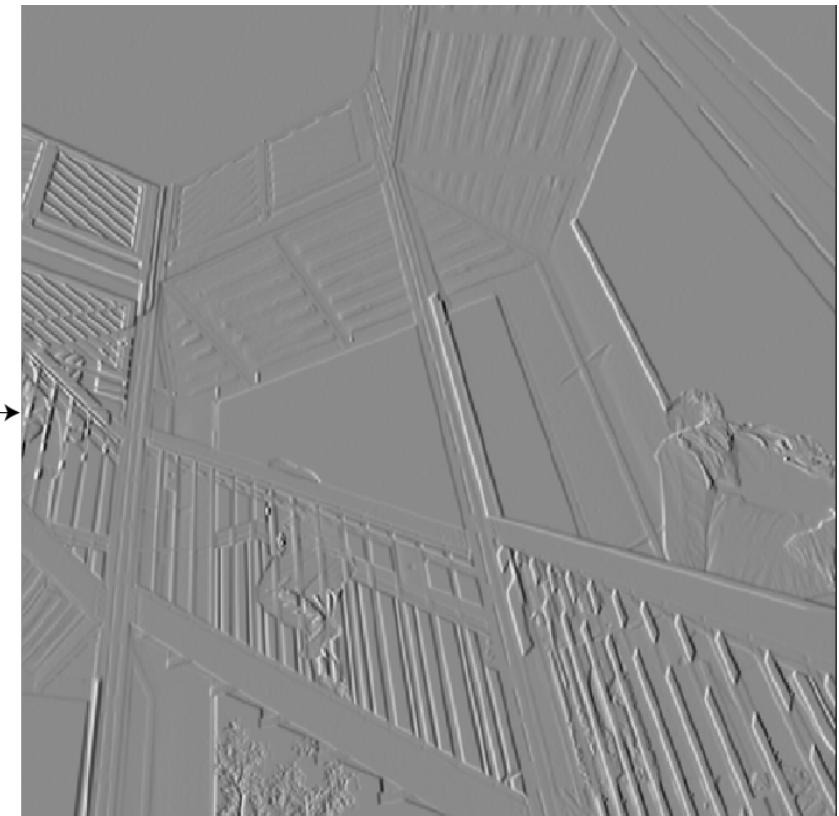
Max = 2

Свёртки в компьютерном зрении



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

Horizontal Sobel kernel



Свёртка

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

Свёртка

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

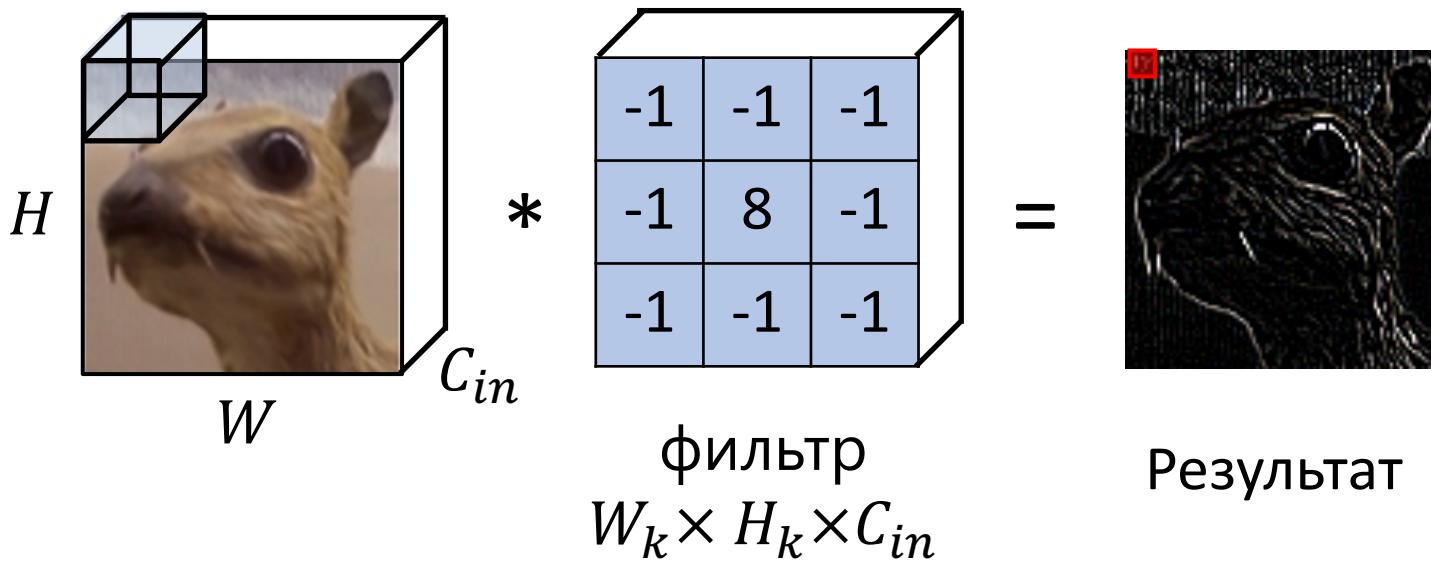
- Пиксель в результирующем изображении зависит только от небольшого участка исходного изображения (local connectivity)
- Веса одни и те же для всех пикселей результирующего изображения (shared weights)

Свёртка

- Обычно исходное изображение цветное!
- Это означает, что в нём несколько каналов (R, G, B)
- Учтём в формуле:

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

Свёртка



Свёртка

- Одна свёртка выделяет конкретный паттерн на изображении
- Нам интересно искать много паттернов
- Сделаем результат трёхмерным:

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

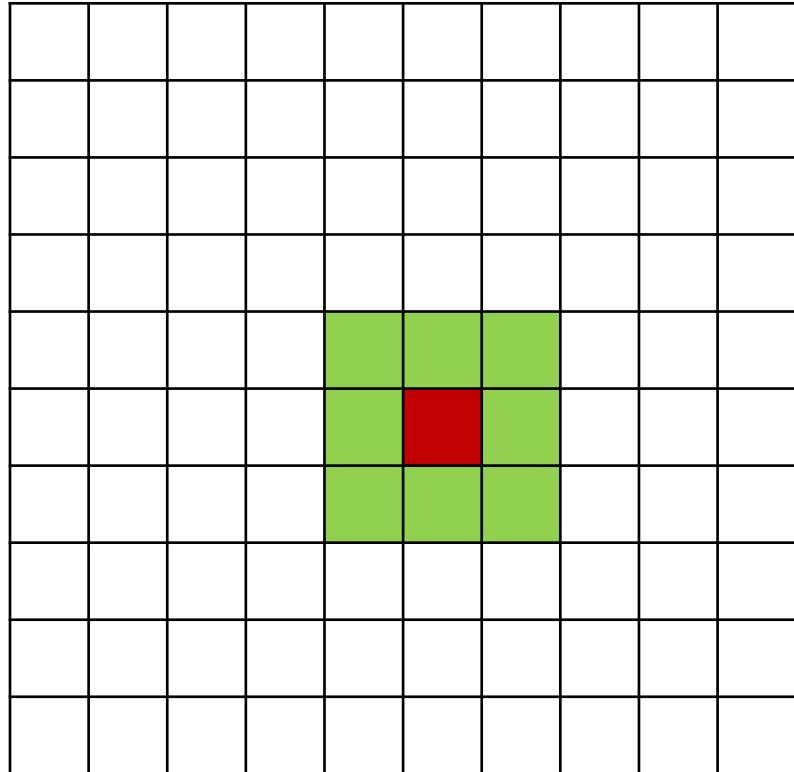
Число параметров

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

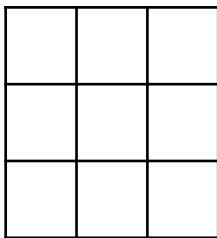
- Обучается только фильтр
- $(2d + 1) * C * T$ параметров

Receptive field

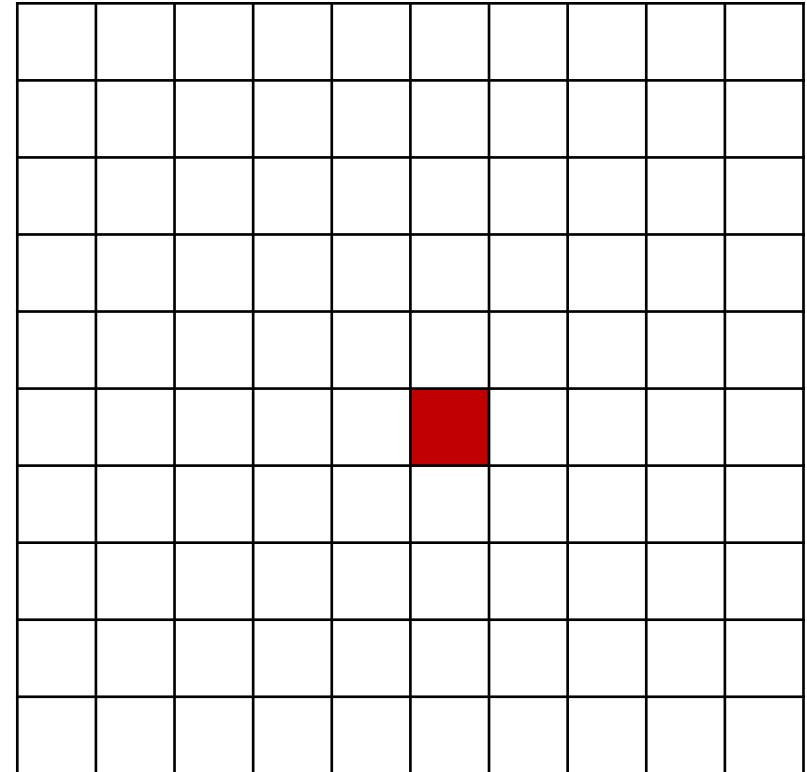
Receptive field



*

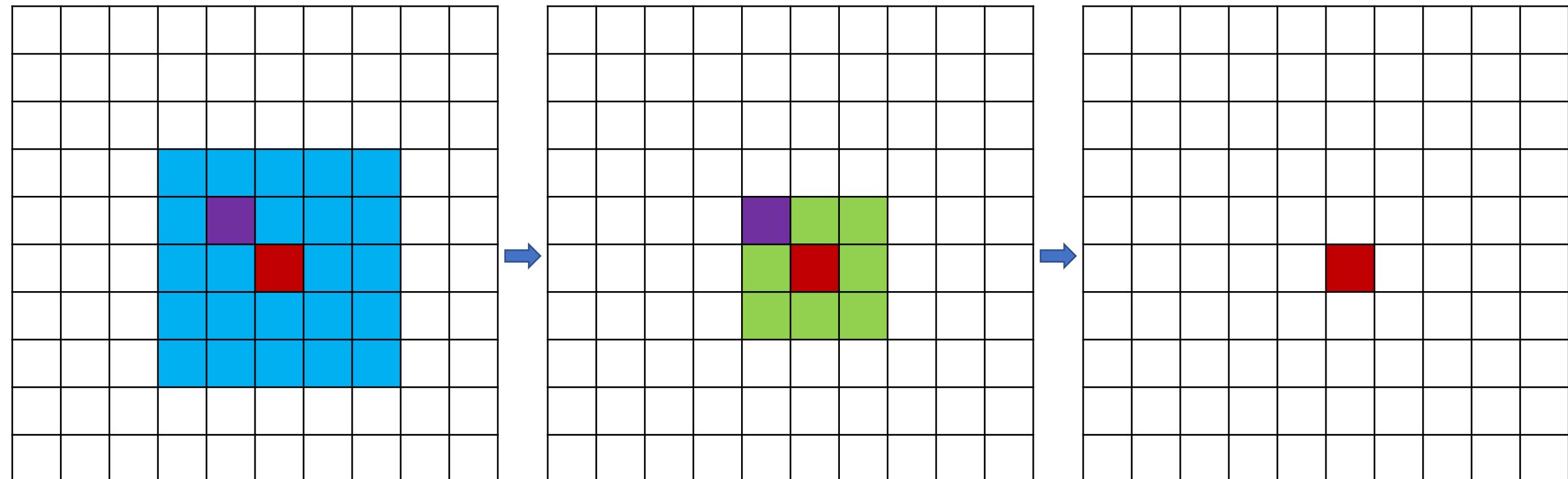


=



Поле восприятия: 3×3

Receptive field



Поле восприятия: 5×5

Receptive field

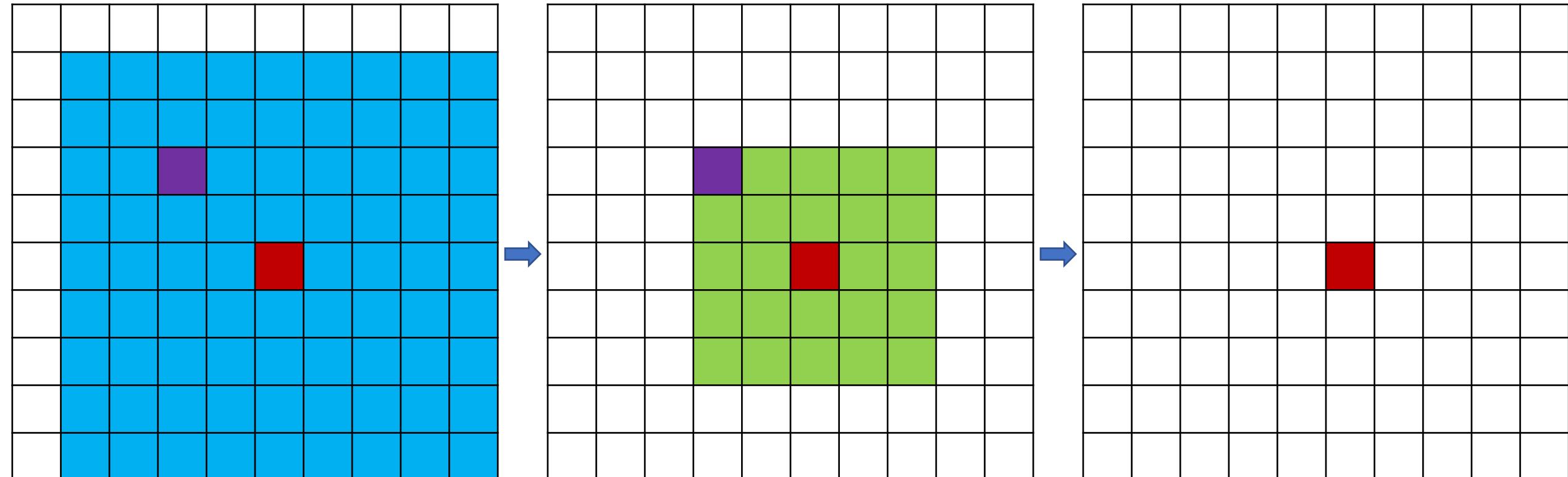
Поле восприятия для свёртки 3×3 :

- После 1 свёрточного слоя: 3×3
- После 2 свёрточных слоев: 5×5
- После 3 свёрточных слоёв: 7×7

Receptive field

Поле восприятия для свёртки 5×5 :

Receptive field



Поле восприятия: 5×5

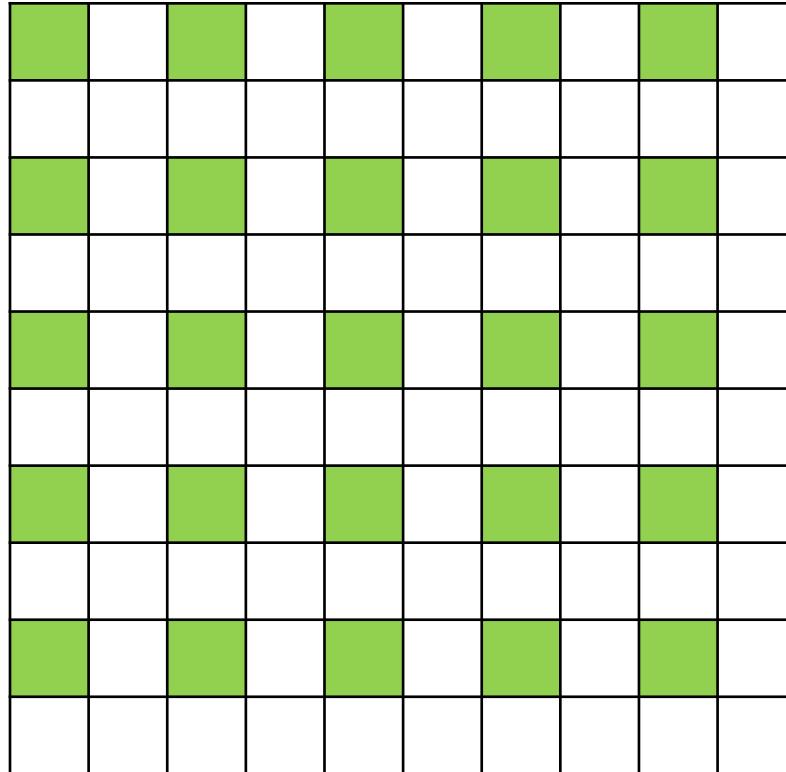
Receptive field

Поле восприятия для свёртки 5×5 :

- После 1 свёрточного слоя: 5×5
- После 2 свёрточных слоев: 9×9
- После 3 свёрточных слоёв: 13×13

Нужно очень много слоёв, если изображение размера 512×512

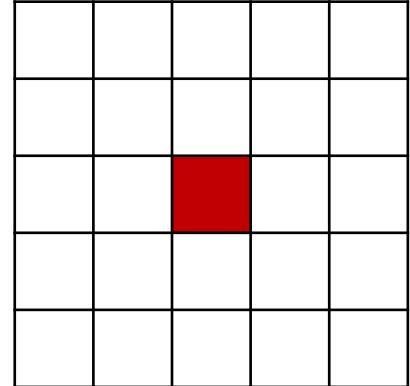
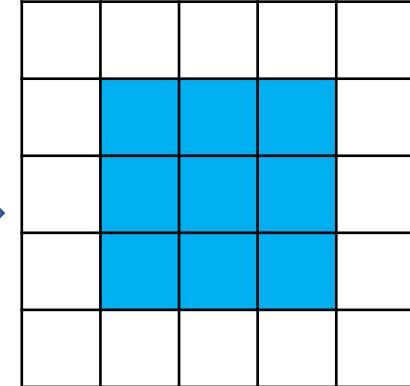
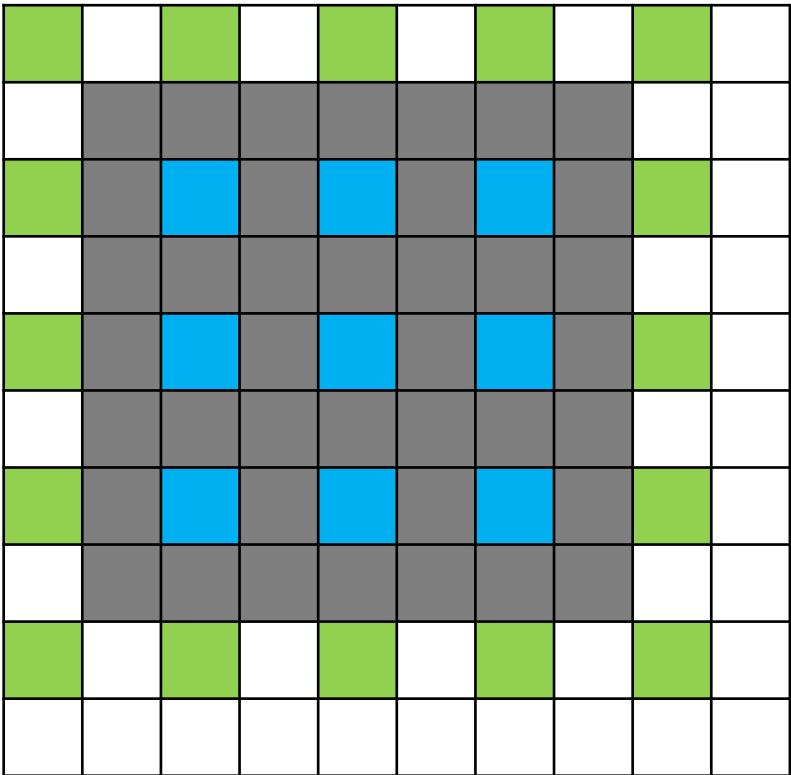
Свёртки с пропусками (strides)



$$\begin{matrix} * & \begin{matrix} \square & \square & \square \\ \square & \square & \square \end{matrix} & = & \begin{matrix} \square & \square & \square & \square \\ \square & \square & \square & \square \end{matrix} \end{matrix}$$

$s = 2$

Свёртки с пропусками (strides)



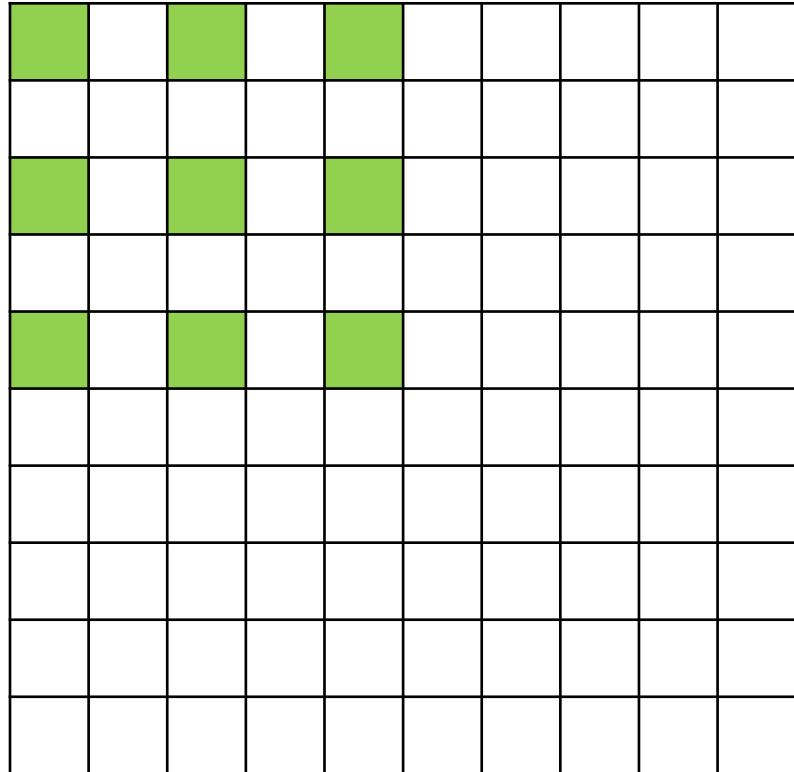
Поле восприятия: 7×7

Свёртки с пропусками (strides)

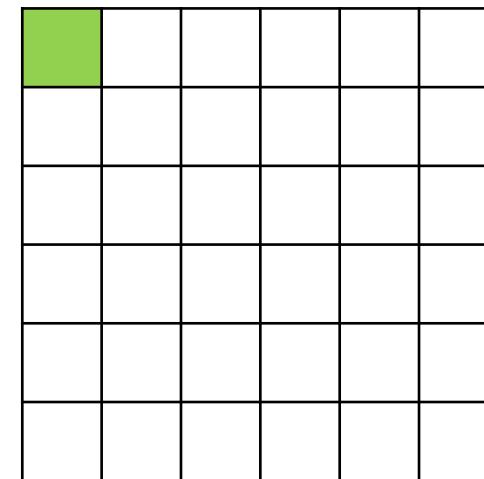
Подробности про подсчёт размера поля:

<https://distill.pub/2019/computing-receptive-fields/>

Dilated convolutions («раздутые» свёртки)

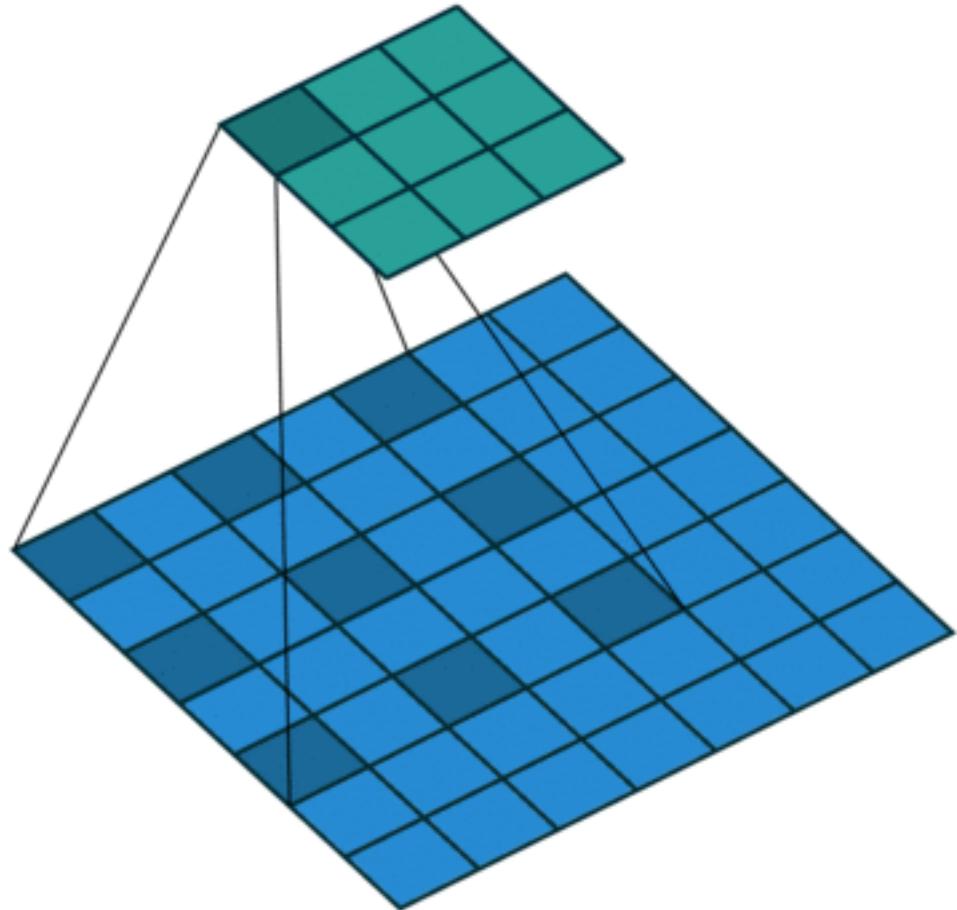


$$\begin{matrix} * & \begin{matrix} \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \\ \square & \square & \square \end{matrix} & = \end{matrix}$$

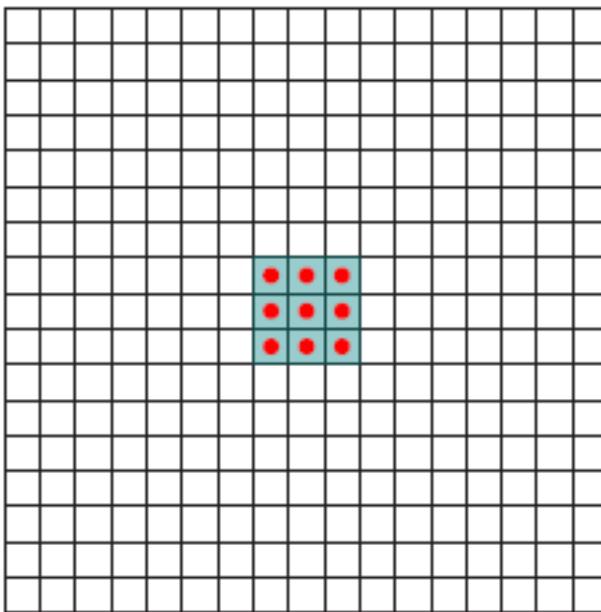


$$l = 2$$

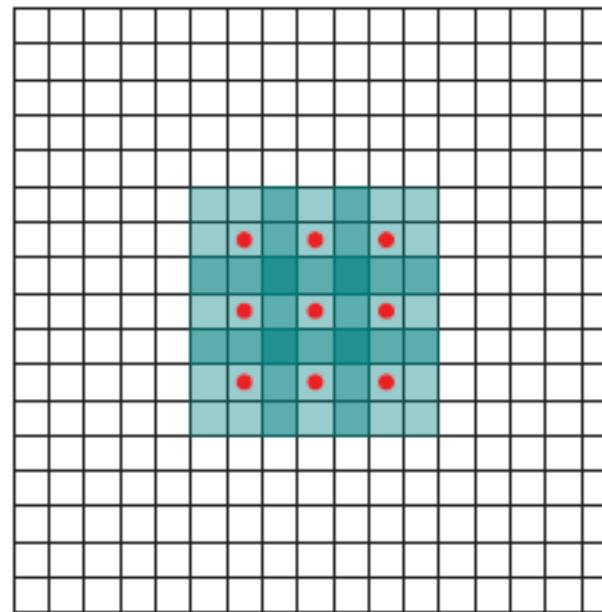
Dilated convolutions («раздутые» свёртки)



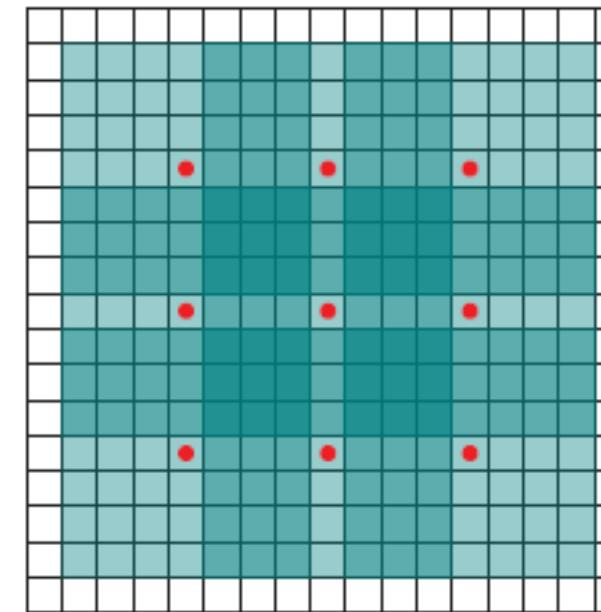
Dilated convolutions («раздутые» свёртки)



$$l = 1$$



$$l = 2$$



$$l = 4$$

Pooling

1	0	2	1	0	0
0	1	3	2	1	2



1	3	2

Max-pooling с фильтром 2x2

Pooling

- Разбивает изображение на участки $n \times m$ и считает некоторую статистику в каждом участке (обычно максимум)
- Существенно сокращает размер изображения (значит, увеличивает поле восприятия следующих слоёв)
- Не имеет параметров

Зачем это всё?

- Важно следить за тем, чтобы последние свёрточные слои имели размер поля восприятия, сравнимый со всей картинкой

Padding

Свёртки

- Если применять свёртку по формуле, то выходное изображение будет меньше входного

Свёртки

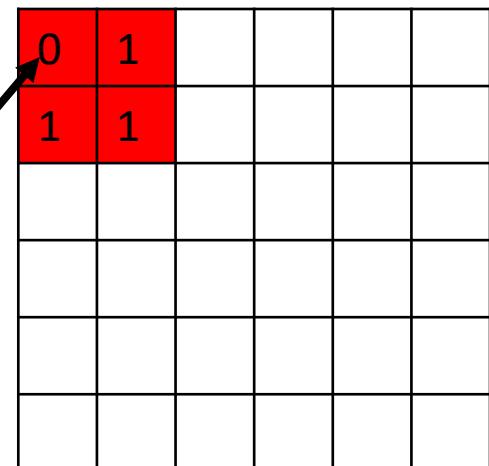
$$\begin{matrix} & \begin{matrix} & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \\ & & & & & \end{matrix} & * & \begin{matrix} & & \\ & & \\ & & \\ & & \\ & & \end{matrix} & = & \begin{matrix} & & & & \\ & & & & \\ & & & & \\ & & & & \\ & & & & \end{matrix} \end{matrix}$$

The diagram illustrates a convolution operation. On the left is a 7x7 input matrix with alternating light green and dark green cells. In the center is a 3x3 kernel matrix with all cells empty (white). To the right of the multiplication symbol (*) is an equals sign (=) followed by a 5x5 output matrix where every cell is filled with dark green.

Valid mode

- При честном подсчёте свёрток пиксели на краях не оказывают большого влияния на результат

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



A 5x5 grid of squares. The top-left square contains the value 1. An arrow points from the text "Не увидим, что фильтр имеет хороший отклик при помещении центра в этот пиксель" to this square. The square containing 1 is highlighted with a red border.

0	1			
1	1			

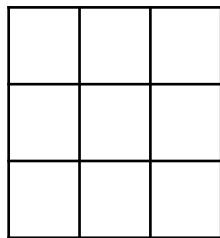
*

0	0	1
0	0	1
1	1	1

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

*



=

Zero padding

- Добавляем по границам нули так, чтобы посчитанная после этого свёртка в valid mode давала изображение такого же размера, как исходное
- Есть риск, что модель научится понимать, где на изображении края — можем потерять инвариантность

Reflection padding

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					

*

=

Reflection padding

- Не получится легко находить края изображения
- Но теперь модель может начать находить зеркальные отражения и подбирать фильтры под них

Replication padding

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				

*

=

Replication padding

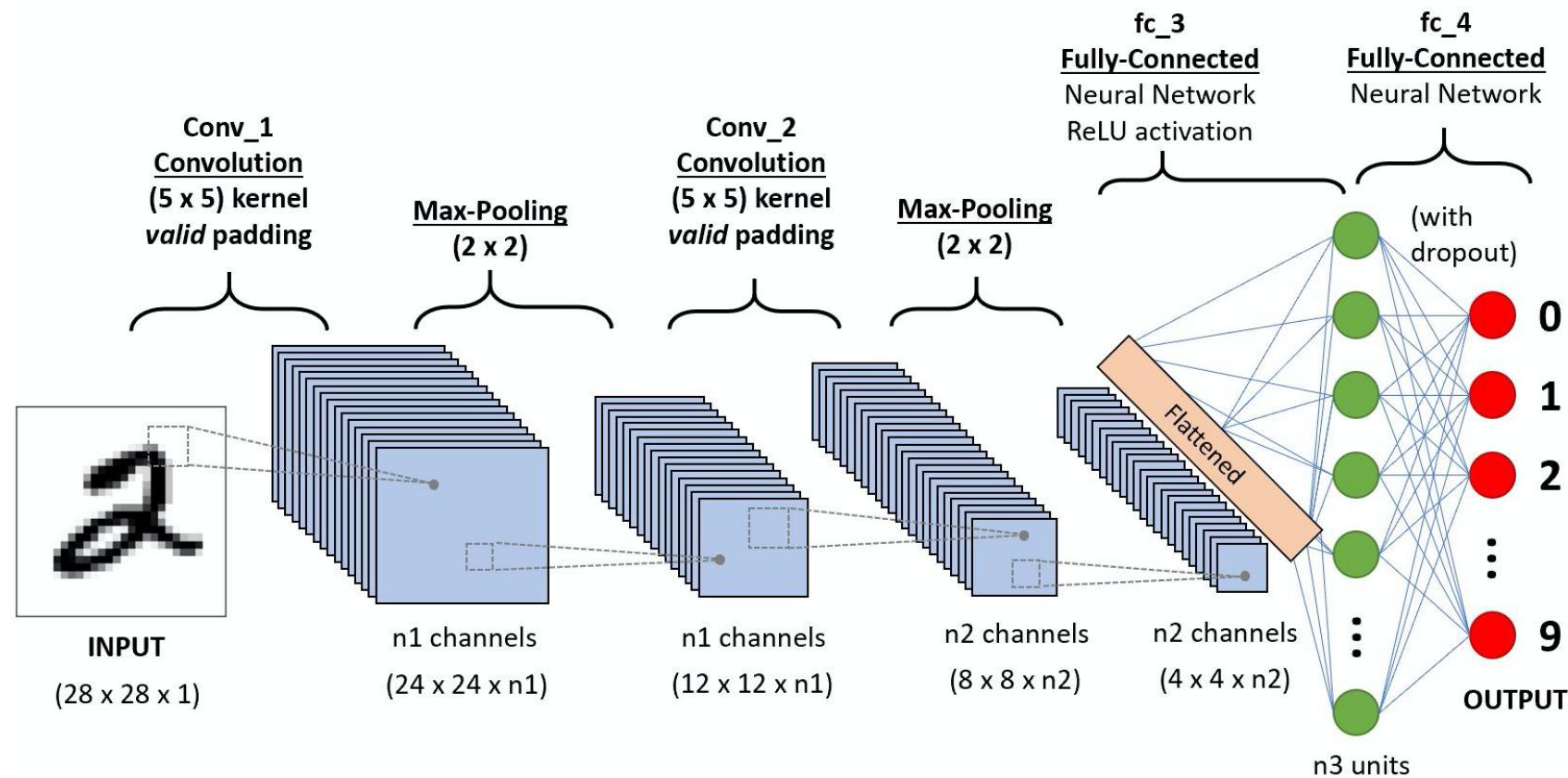
- Пиксель на границе равен ближайшему пикслю из изображения
- Модель всё ещё может настроиться под паттерны, которые возникают из-за такого паддинга

Резюме

- Паддинг позволяет контролировать размер выходных изображений
- Паддинг позволяет учитывать даже объекты на краях
- Разные типа паддингов допускают разные способы переобучения под края

Структура свёрточных сетей

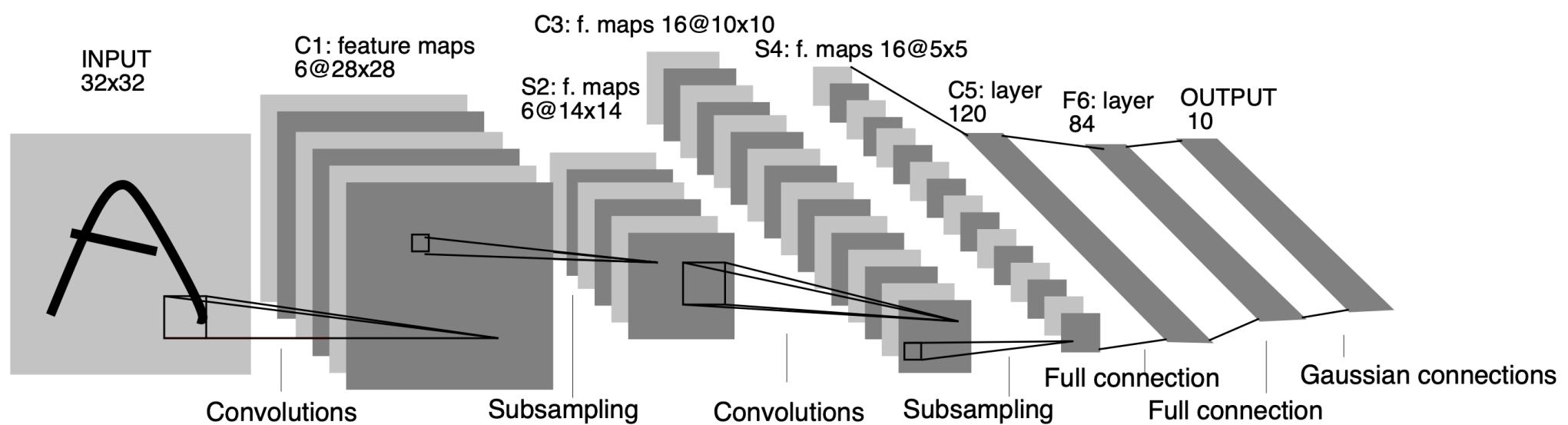
Типичная архитектура



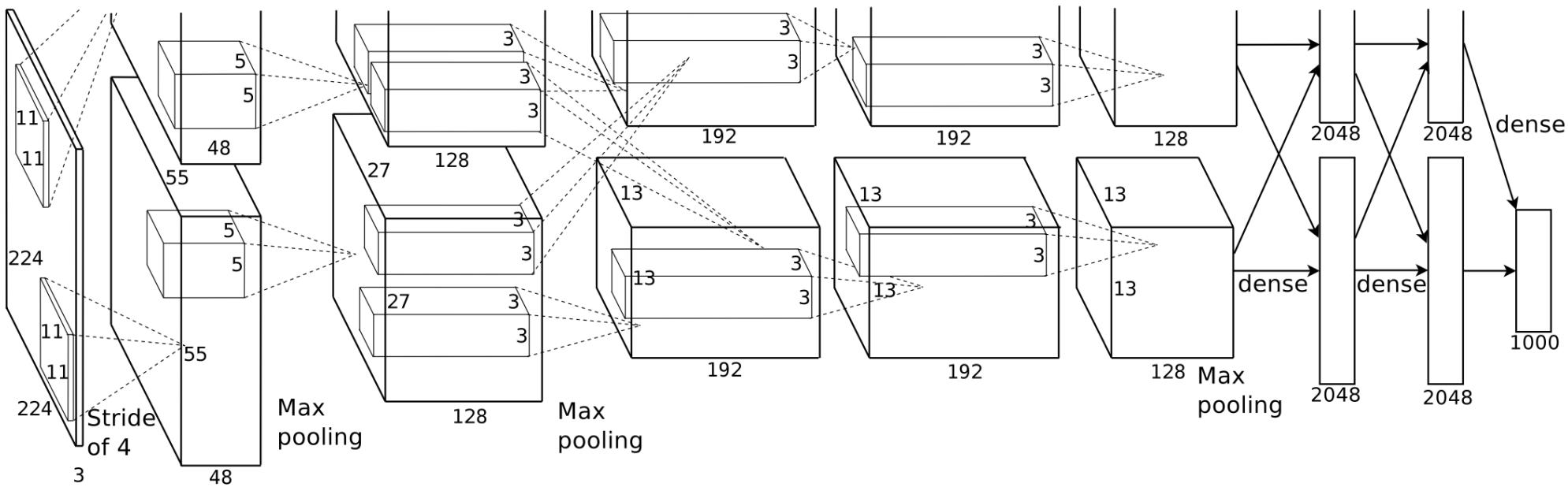
Типичная архитектура

- Последовательное применение комбинаций вида «свёрточный слой -> нелинейность -> pooling» или «свёрточный слой -> нелинейность»
- Выпрямление (flattening) выхода очередного слоя
- Серия полносвязных слоёв

LeNet



AlexNet

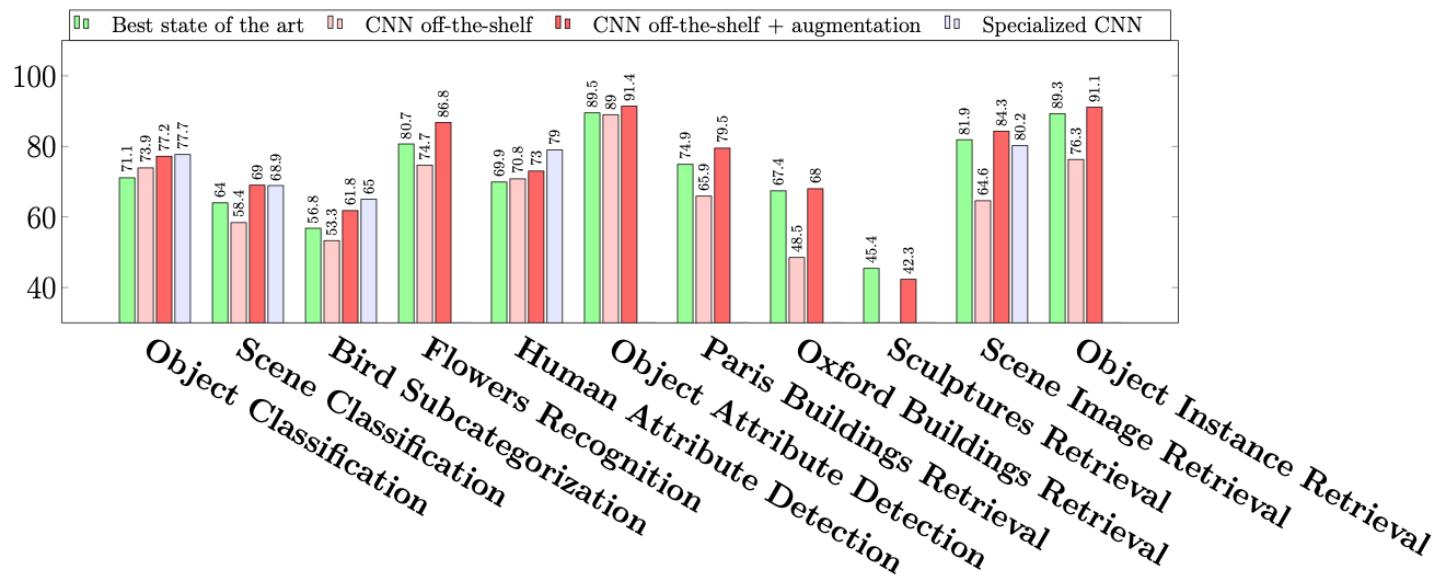
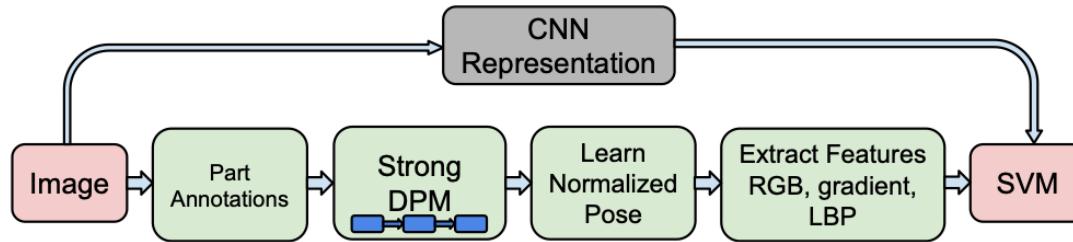


<http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf>

Представления с последних слоёв

- Важное наблюдение: выходы полносвязных слоёв являются хорошими признаковыми описаниями изображений
- Полезны во многих задачах
- Например, поиск похожих изображений

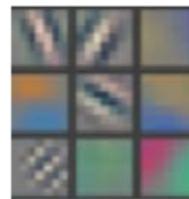
Представления с последних слоёв



Представления с последних слоёв

- Не интерпретируется (в отличие от классического компьютерного зрения)
- По смыслу — «индикаторы» наличия каких-то паттернов

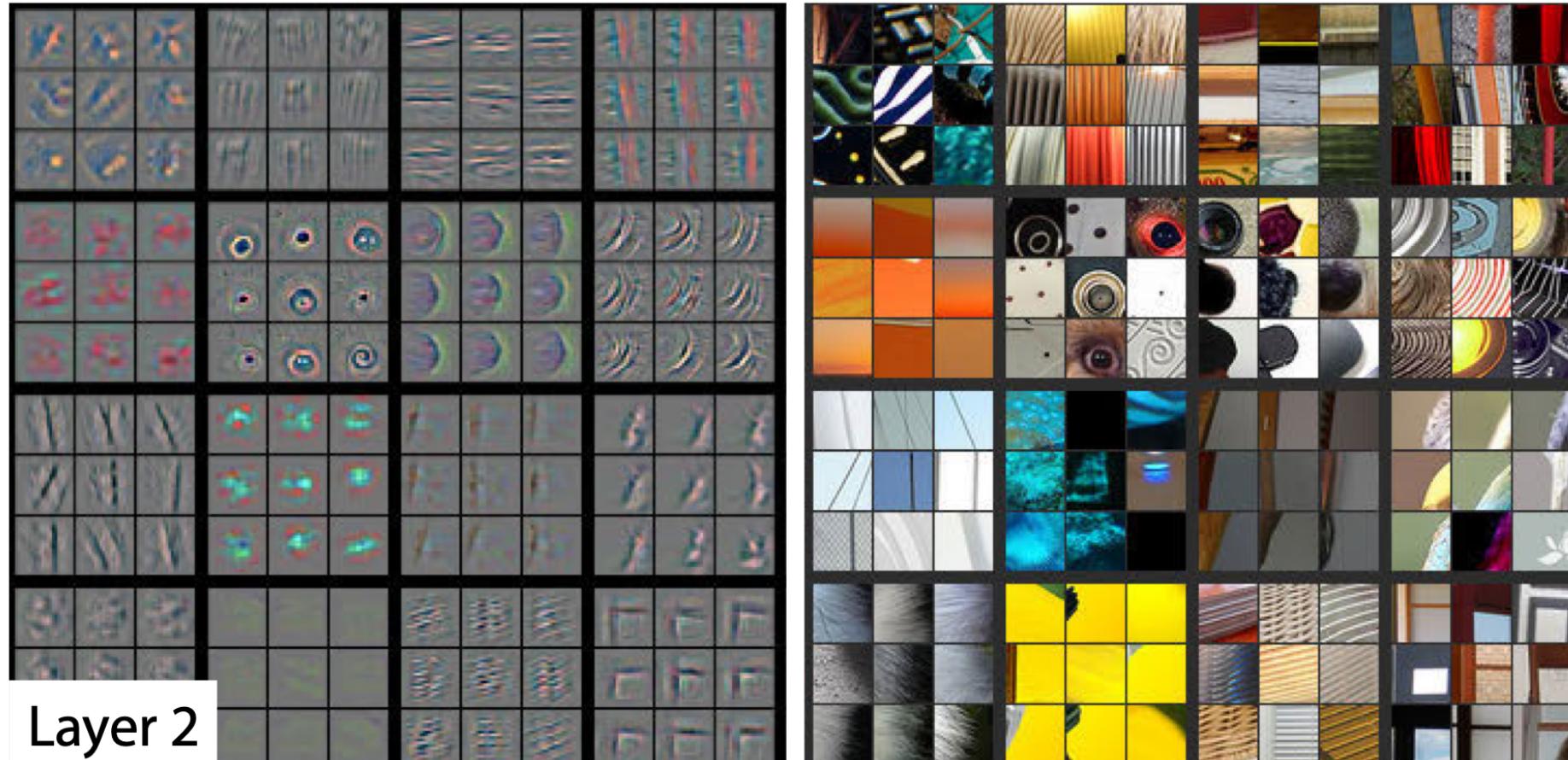
Представления с последних слоёв



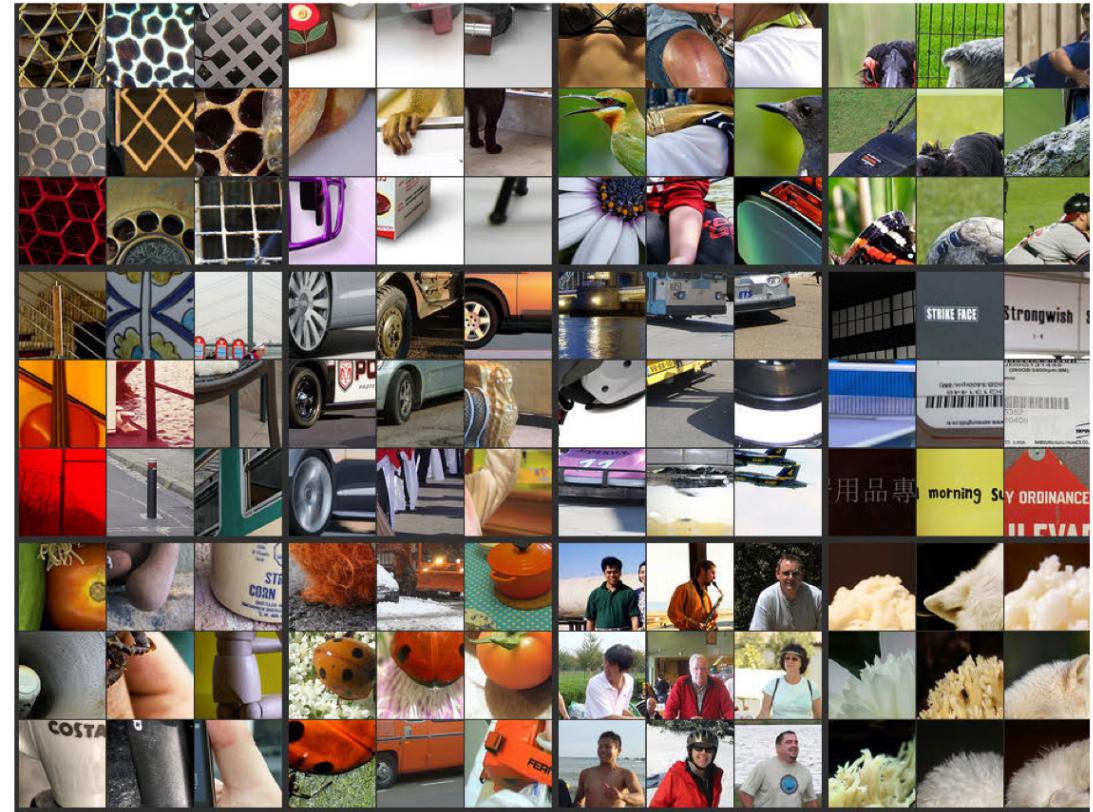
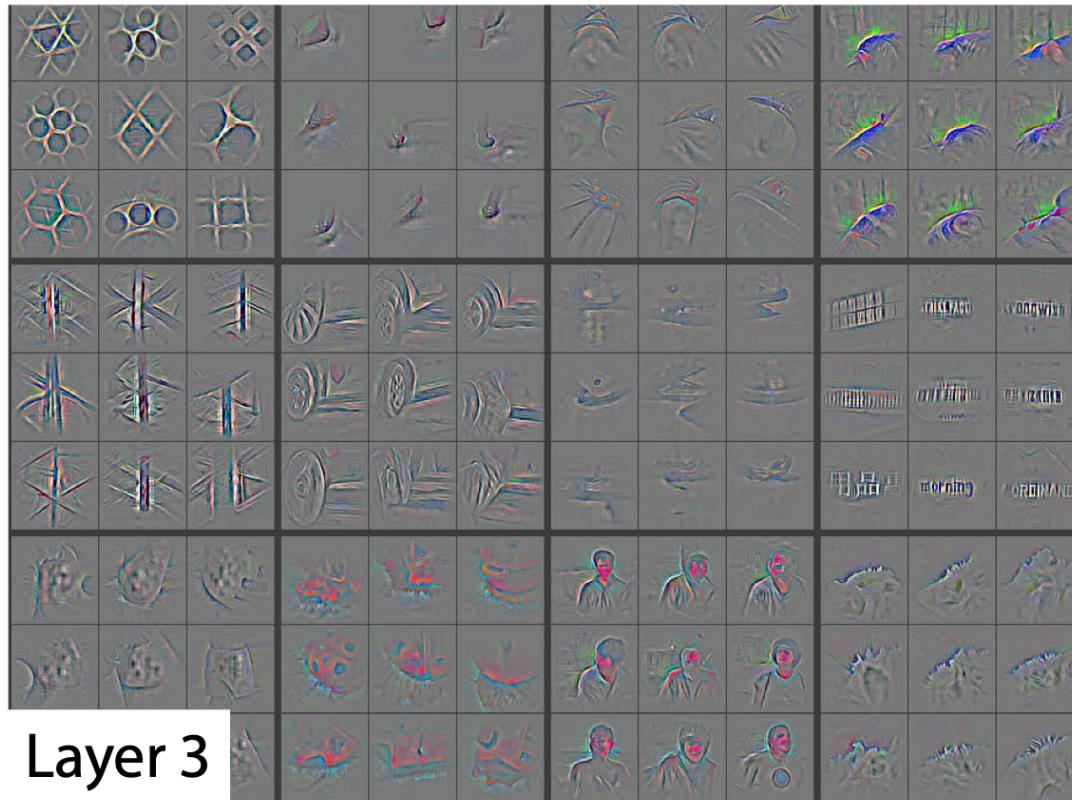
Layer 1



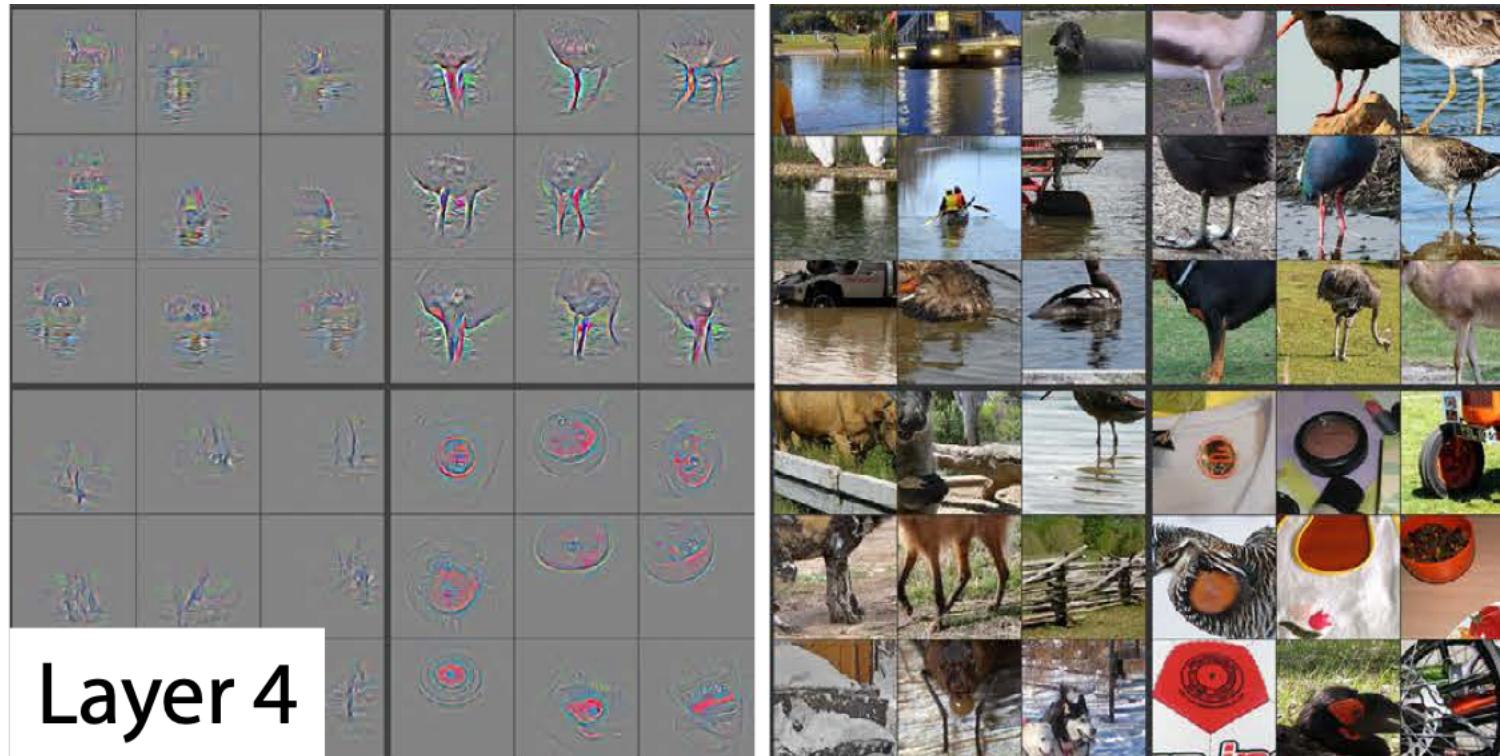
Представления с последних слоёв



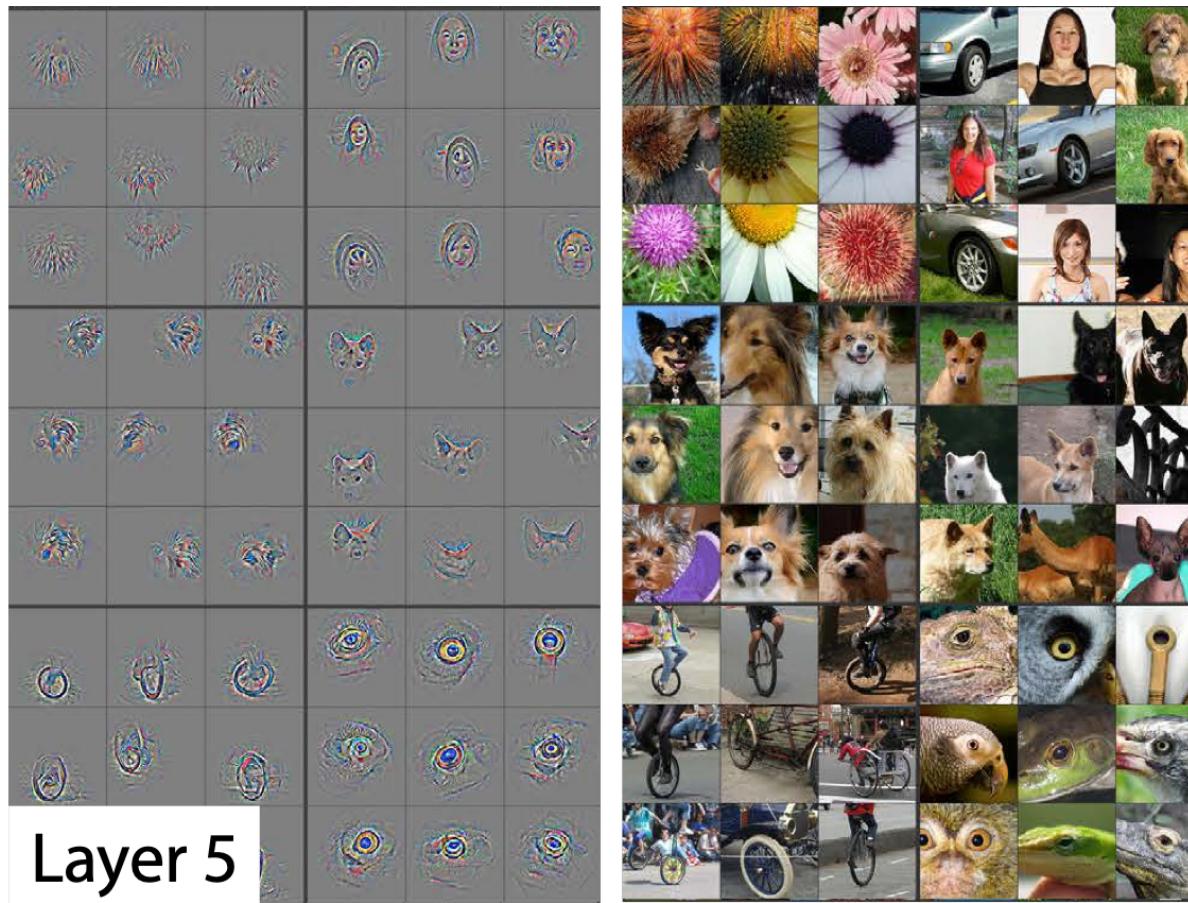
Представления с последних слоёв



Представления с последних слоёв



Представления с последних слоёв

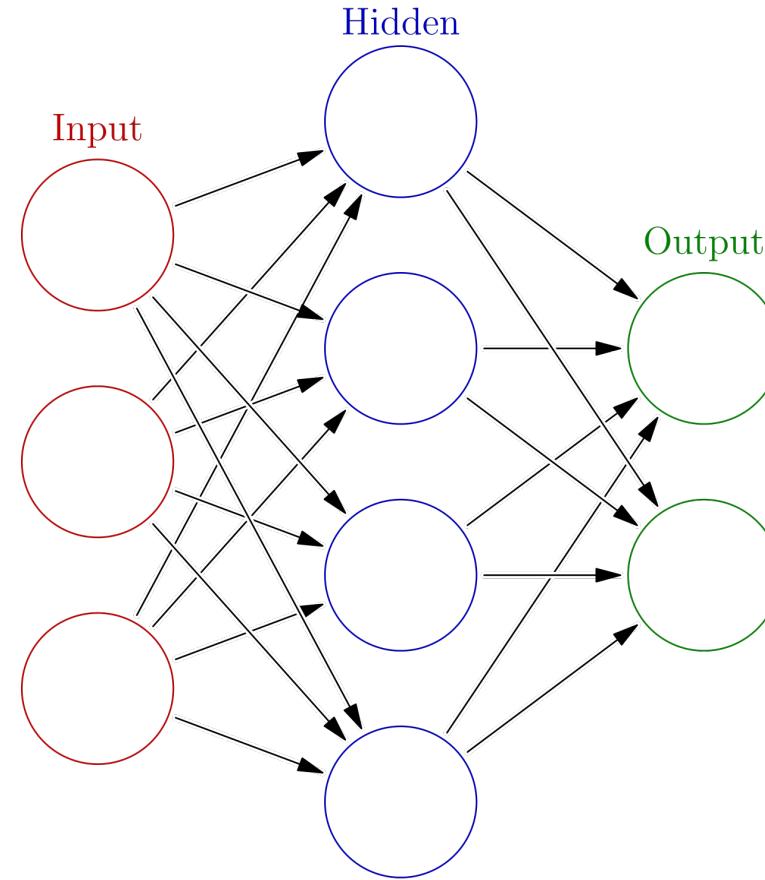


Dropout

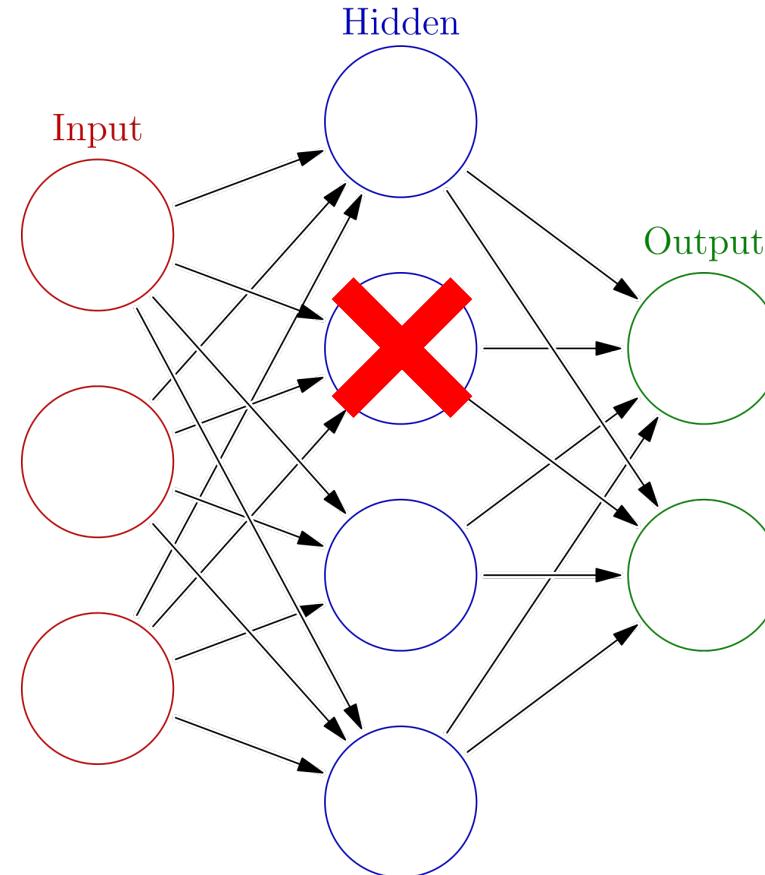
Борьба с переобучением

- Сокращение числа параметров (свёрточные слои помогают с этим)
- Регуляризация
- Можно как-то ещё мешать модели подгоняться под обучающую выборку

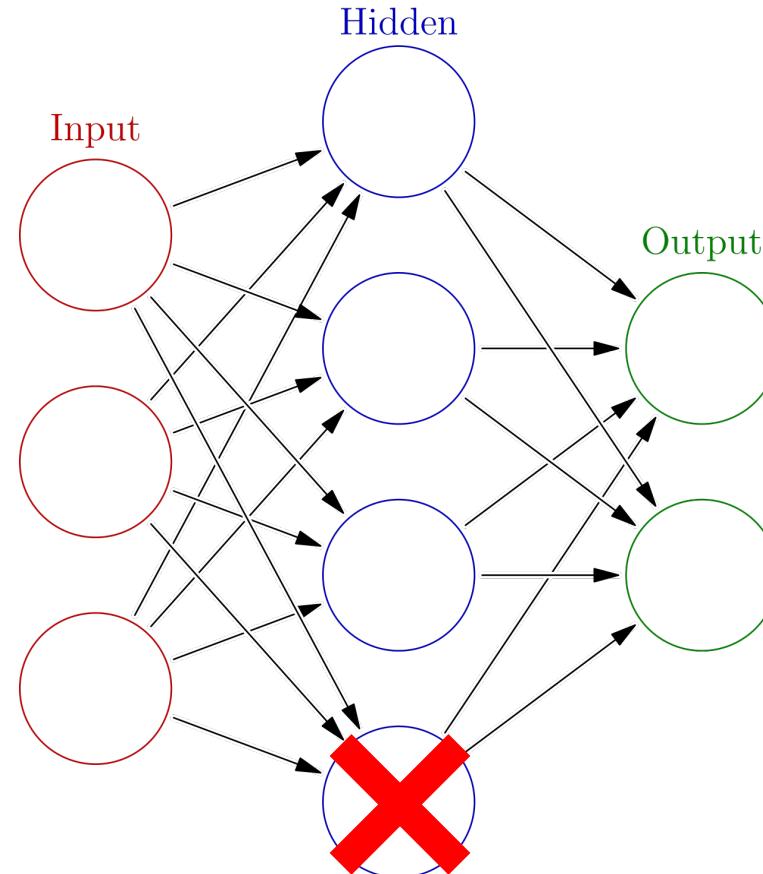
Dropout



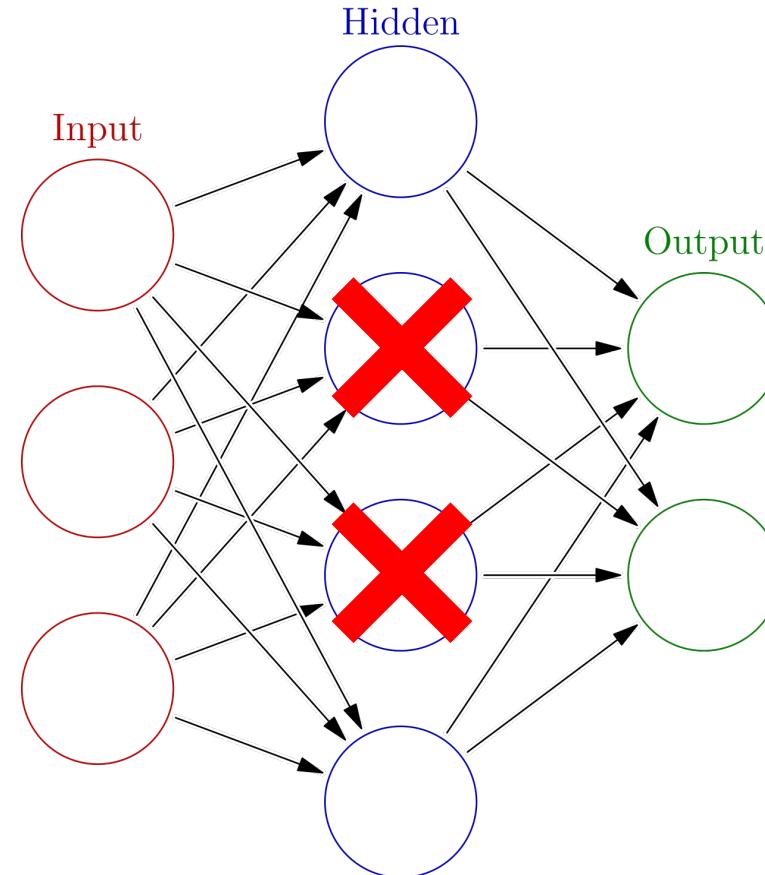
Dropout



Dropout



Dropout



Dropout

- Можно определить как слой $d(x)$
- Параметров нет, единственный гиперпараметр — p (вероятность удаления нейрона)
- На этапе обучения:

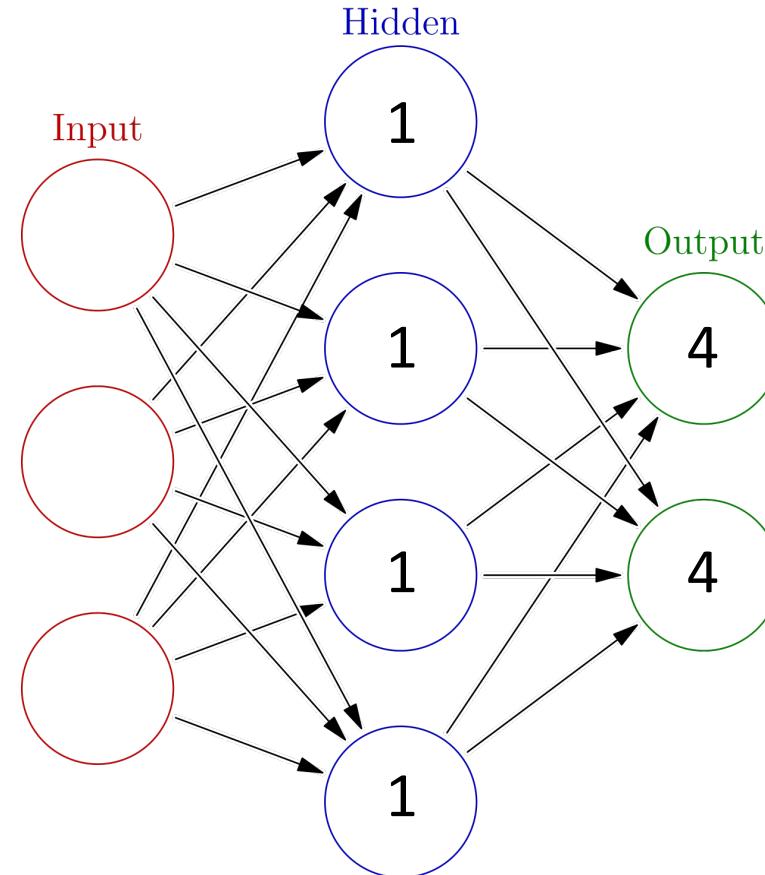
$$d(x) = \frac{1}{p} m \circ x$$

(m — вектор того же размера, что и x , элемент берутся из распределения $\text{Ber}(p)$)

- Деление на p — для сохранения суммарного масштаба выходов

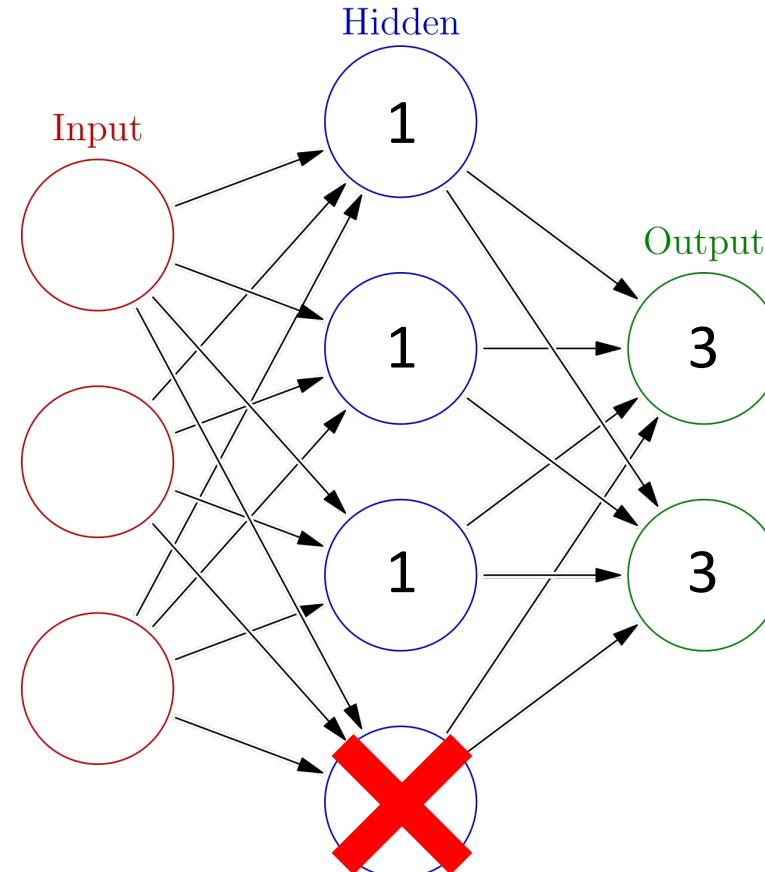
Dropout

Пусть все веса единичные



Dropout

Пусть все веса единичные



Надо компенсировать снижение
масштаба суммы выходов!

Dropout

- На этапе обучения:

$$d(x) = \frac{1}{p} m \circ x$$

- На этапе применения:

$$d(x) = x$$

В оригинальной статье нет нормировки на этапе обучения, но есть домножение на p на этапе применения

Вариант на слайде — inverted dropout

Dropout

- Интерпретация: мы обучаем все возможные архитектуры нейросетей, которые получаются из исходной выбрасыванием отдельных нейронов
- У всех этих архитектур общие веса
- На этапе применения (почти) усредняем прогнозы всех этих архитектур