

# Машинное обучение (Machine Learning)

## Глубокое обучение: Сверточные сети (Deep Learning: Convolutional nets)

Уткин Л.В.

Санкт-Петербургский политехнический университет Петра Великого



# Содержание

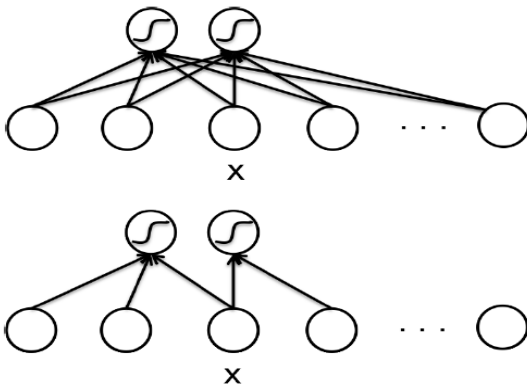
- 1 Причины создания сверточных сетей и предварительные замечания
- 2 Сверточный слой
- 3 Max-pooling

# Причины создания сверточных сетей и предварительные замечания

# Причины создания сверточных сетей

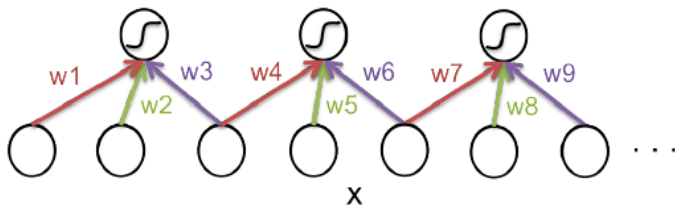
- Вход - большой размерности: каждый нейрон имеет огромное число соединений. Малая картинка  $100 \times 100$  пикселей (размерность входа - 10000), каждый нейрон имеет 10000 параметров. Если скрытый слой - 2000 нейронов, то всего  $2 \times 10^7$  соединений.
- А что, если часть соединений убрать?

# Две сети



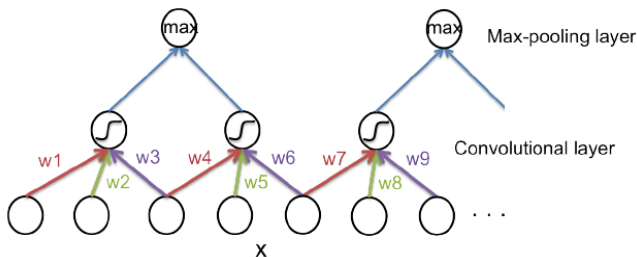
# Как уменьшить число соединений?

- Сделать часть весов одинаковыми ("weight sharing" или **свертка**)
- $w_1 = w_4 = w_7$ ,  $w_2 = w_5 = w_8$ ,  $w_3 = w_6 = w_9$  ("фильтр")
- Вместо хранения всех весов, храним  $w_1$ ,  $w_2$ ,  $w_3$



# Сверточные сети

К свертке добавляется другой слой, известный как “**max-pooling layer**”, который вычисляет максимальное значение из выбранного подмножества выходных нейронов сверточного слоя и использует его как входное значение следующего слоя



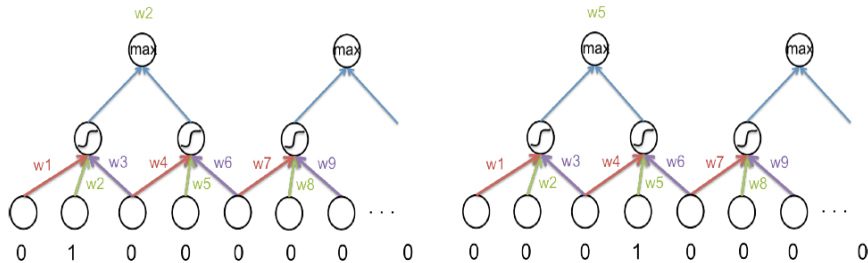
# Свойство инвариантности

- Интересное свойство сети: выход **max-pooling** нейронов инвариантен к “смещению” входа.
- $x_1 = (0, 1, 0, 0, 0, 0, 0, \dots)$  и  $x_2 = (0, 0, 0, 1, 0, 0, 0, \dots)$  - два 1D-изображения с одним белым пикселем.
- Оба изображения одинаковы, исключая то, что в  $x_2$  белый пиксель смещен на 2 пиксела вправо.
- При смещении на 2 пиксела вправо, значение первого **max-pooling** нейрона изменяется от  $w_2$  к  $w_5$ . Но  $w_2 = w_5$ , т.е. значение нейрона не изменилось.



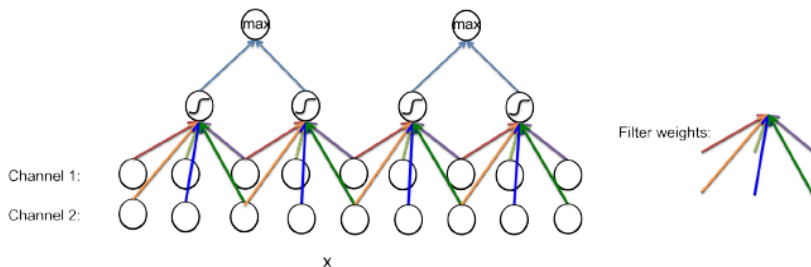
## Свойство инвариантности

- При смещении на 2 пиксела вправо, значение первого **max-pooling** нейрона изменяется от  $w_2$  к  $w_5$ . Но  $w_2 = w_5$ , т.е. значение нейрона не изменилось.



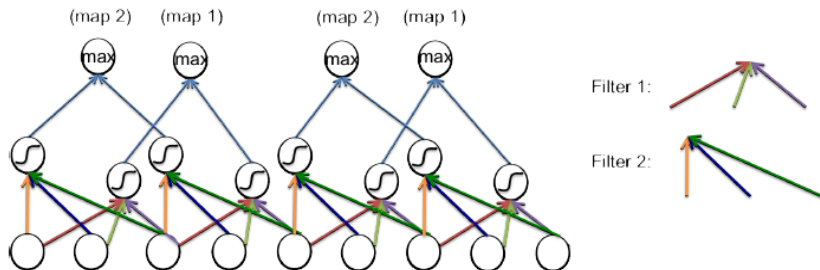
# Многоканальная сеть

- Изображения имеют несколько каналов (красный, зеленый, голубой)
- Сверточная архитектура с двумя каналами



# Сеть с многими фильтрами

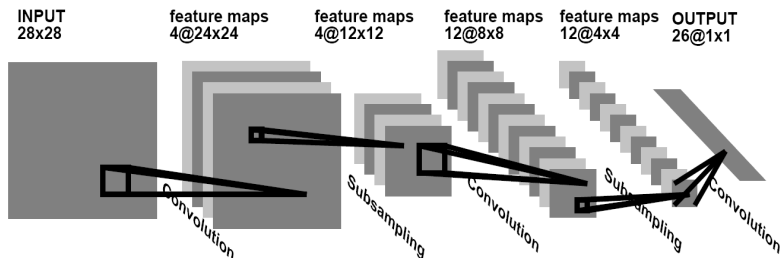
- Пример с двумя фильтрами



- Выход одного фильтра называется “карта”.
- Карта 1 (2) создана фильтром 1 (2).

# Иллюстрация сверточной сети

LeCun Y., Bengio Y. Convolutional Networks for Images, Speech, and Time-Series // The Handbook of Brain Theory and Neural Networks, MIT Press, 1995

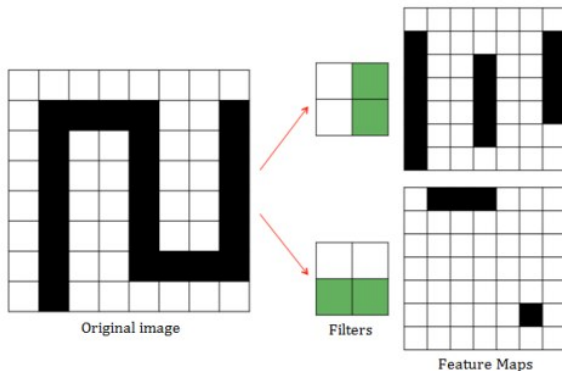


# Слои сверточной сети

- **Сверточный слой:** реализует обычную операцию свертки - двигаясь по изображению скользящим окном, перемножаем значения в окне с заданными весами (ядром), а затем все складываем. Наборов весов может быть несколько.
- **Pooling Layers:** Для уменьшения числа данных и выбора наиболее интересных признаков. Входной двумерный массив делится на сектора, в зависимости от параметров, и в каждом из них происходит максимизация (MAX) или усреднение (AVE) (два самых распространенных вида pooling'a).

# Сверточный слой

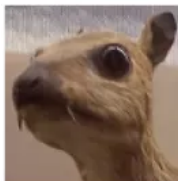
# Сверточный слой (фильтр)



Цель - определение горизонтальных и вертикальных границ. Используем фильтры (зеленый цвет) – это небольшая матрица, представляющая признак, который хотим найти на исходном изображении.

# Сверточный слой (фильтр)

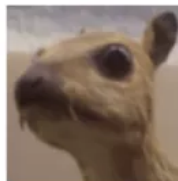
Input image



Kernel

$$\begin{pmatrix} \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \end{pmatrix}$$

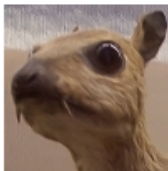
Feature map





# Сверточный слой (фильтр)

Input image



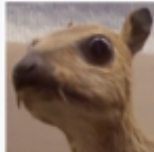
Convolution  
Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$




Feature map






# Сверточный слой (фильтр)

Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	

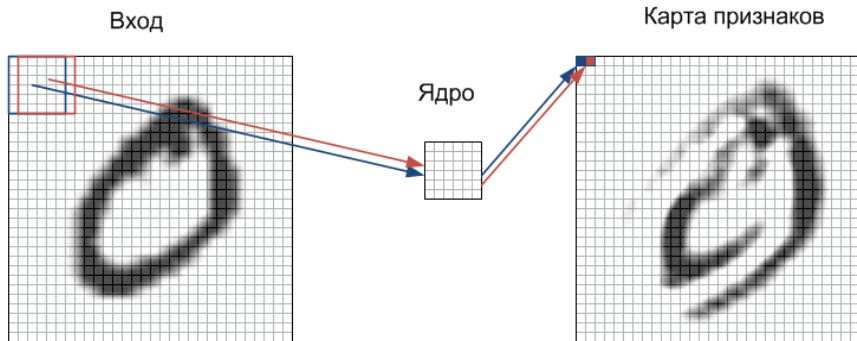
# Сверточный слой (фильтр)

Operation	Filter	Convolved Image
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

# Сверточный слой (фильтр)

Operation	Filter	Convolved Image
<b>Sharpen</b>	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
<b>Box blur</b> (normalized)	$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
<b>Gaussian blur</b> (approximation)	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

# Сверточный слой (фильтр)



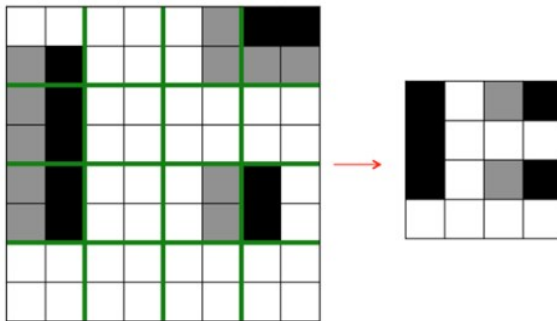
<https://geektimes.ru/post/74326/>

# Max-pooling

# Max-pooling

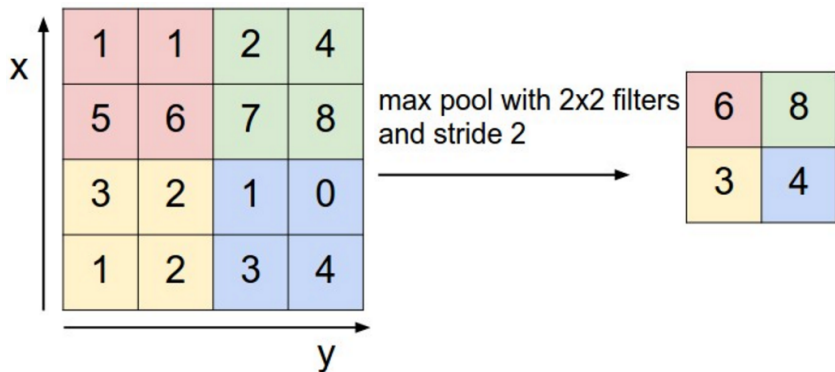
- По мере продвижения вглубь сверточной нейронной сети, уточняем информацию на картах признаков. Если признак присутствует на предыдущем слое, полученная карта содержит ключевые пиксели, окруженные нечетким «ореолом», в рамках которого признак может быть определен не до конца.
- Решение - метод - max-объединение (max-pooling). Это - разделение карты признаков на непересекающиеся участки и выделения на этих участках нейронов с максимальной активностью. Max-pooling карты признаков делает распознавание более точным, избавляясь от ненужных «ореолов».

# Max-pooling





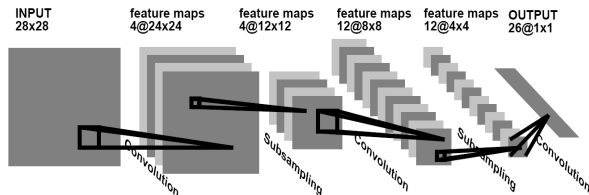
# Max-pooling



# Субдискретизирующий слой

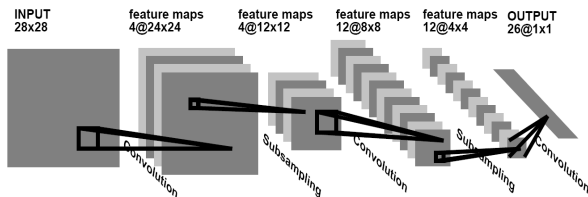
- Выполняет уменьшение размера входной карты признаков, например в 2 раза.
- Разные методы, например, метод выбора максимального элемента (max-pooling) - вся карта признаков разделяется на ячейки 2x2 элемента, из которых выбираются максимальные по значению.
- Формально  $\mathbf{x}^{(k)} = f(W^{(k)} \cdot \text{subsample}(\mathbf{x}^{(k-1)}) + b^{(k)})$ ,  $f$  - функция активации,  $\text{subsample}$  - операция выборки локальных максимальных значений.
- Использование этого слоя позволяет улучшить распознавание примеров с измененным масштабом (уменьшенных или увеличенных).

# Сверточная сеть в целом (1)



- 1 входной слой - матрица картинки  $28 \times 28$
- 2 сверточный слой - набор однотипных матриц (карт признаков) по числу ядер свертки
- 3 субдискретизирующий слой - уменьшенный в 2 раза предыдущий набор матриц

# Сверточная сеть в целом (2)



- 4 сверточный слой - предыдущий набор матриц суммируется в соответствии со схемой соединения слоев и генерируется новый набор по числу ядер свертки
- 5 субдискретизирующий слой - уменьшенный в 2 раза предыдущий набор матриц
- 6 выходной слой - предыдущий набор матриц разворачивается в вектор и обрабатывается

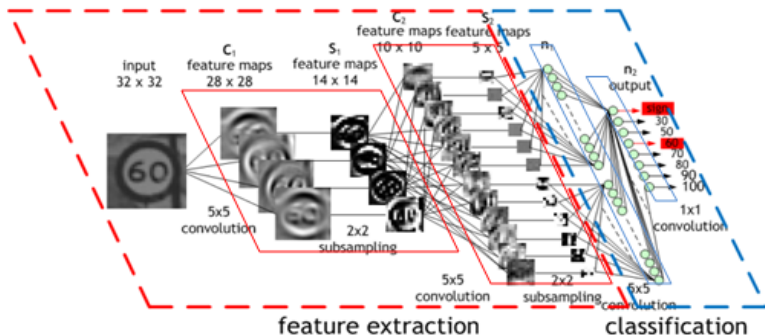
# Особенность соединения слоев

Нейроны 2-го субдискретизирующего и 3-го сверточного слоев соединяются **выборочно** в соответствии с **матрицей смежности**, которая задается как параметр сети, так для сети с количеством карт признаков во втором 7 и 9 в третьем слое:

	1	2	3	4	5	6	7	8	9
1	1	0	0	0	0	1	1	1	0
2	1	1	0	0	0	0	1	1	1
3	1	1	1	0	0	0	0	1	1
4	0	1	1	1	0	0	0	1	1
5	0	0	1	1	1	0	0	0	1
6	0	0	0	1	1	1	0	0	0
7	0	0	0	0	1	1	1	0	0

# Сверточная сеть

- Фильтрация 4-мя 5x5 сверточными ядрами, создающими 4 карты признаков.
- Max-pooling
- Фильтрация 10-ю 5x5 сверточными ядрами, max-pooling



# Сверточные сети (формально)

**Дано:** большие изображения  $x_{\text{large}}$  размера  $r \times c$

- 1 обучаем прореженный автокодер на малых “кусочках”  $x_{\text{small}}$  размера  $a \times b$
- 2 обучаем  $k$  признаков  $f = \sigma(W^{(1)}x_{\text{small}} + b^{(1)})$  ( $\sigma$  - сигмоид), при наличии весов  $W^{(1)}$  и  $b^{(1)}$  из входа в нейроны скрытого слоя
- 3 для каждого “кусочка”  $x_s$  размера  $a \times b$  в большом изображении вычисляем  $f_s = \sigma(W^{(1)}x_s + b^{(1)})$
- 4 откуда получаем  $f_{\text{convolved}}$  - матрицу “свернутых” признаков размера  $k \times (r - a + 1) \times (c - b + 1)$

# Преимущества сверточных сетей

- Один из лучших алгоритмов по распознаванию и классификации изображений.
- По сравнению с обычной нейронной сетью гораздо меньшее количество настраиваемых весов, так как одно ядро весов используется целиком для всего изображения, вместо того, чтобы делать для каждого пикселя входного изображения свои весовые коэффициенты.
- Нейросеть при обучении обобщает информацию, а не попиксельно запоминает каждую картинку в весовых коэффициентах, как перцептрон.



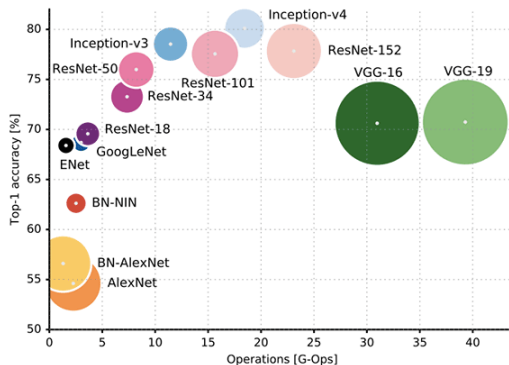
# Преимущества сверточных сетей

- Удобное распараллеливание вычислений, возможность реализации на графических процессорах.
- Относительная устойчивость к повороту и сдвигу распознаваемого изображения.
- Обучение при помощи классического метода обратного распространения ошибки.

# Недостатки сверточных сетей

- 1 Архитектура сверточной сети по большей части для распознавания изображений.
- 2 Слишком много варьируемых параметров сети: количество слоев, размерность ядра свертки для каждого из слоев, количество ядер для каждого из слоев, шаг сдвига ядра при обработке слоя, необходимость слоев субдискретизации, степень уменьшения ими размерности, функция по уменьшению размерности (выбор максимума или среднего) и т.д. Выбираются эмпирически.

# Типовые сверточные сети



<http://www.topbots.com/14-design-patterns-improve-convolutional-neural-network-cnn-architecture>

# Программное обеспечение Deep Learning: Torch

- Torch основан на библиотеке Lua
- Обработка естественного языка с помощью глубоких нейронных сетей
- Используется в Facebook и Twitter Research для исследований и разработки систем глубокого обучения

# Программное обеспечение Deep Learning: MxNet

- MxNet - мощная библиотека, поддерживающая различные языки программирования: Python, Scala, R
- Одна из самых эффективных по быстродействию и по использованию памяти библиотек
- Простота использования нескольких графических процессоров (GPU)

# Программное обеспечение Deep Learning: Theano

- Theano - Python-библиотека
- Объединяет Keras и Lasagne
- Охватывает не только глубокое обучение, но и различные методы машинного обучения: рекуррентная нейронная сеть, ограниченная машина Больцмана, глубокие сети доверия, сверточные нейронные сети
- Проста для разработчиков
- Имеются скрипты для конвертации моделей Caffe

# Программное обеспечение Deep Learning: Lasagne

- Lasagne - библиотека для построения и обучения нейронных сетей в Theano
- Проста в использовании, понимании и расширении
- Для установки требует сначала установить Python и Theano

# Программное обеспечение Deep Learning: Keras

- Keras - модульная библиотека для построения нейронных сетей для Python
- Запускается “поверх” либо TensorFlow, либо Theano



# Программное обеспечение Deep Learning: Caffe

- Caffe - флагман глубокого обучения
- Первая успешная открытая реализация с мощной, но простой базой: нет необходимости знать код для использования Caffe, используются простые файлы описаний сети
- Не поддерживает GPU, отличные от Nvidia

# Программное обеспечение Deep Learning: TensorFlow

- TensorFlow - открытая библиотека для анализа представления данных в виде графа
- Вершины графа - математические операции, крайние вершины - матрицы данных большой размерности (тензоры)
- TensorFlow разработана в Google Brain Team в целях проведения исследования в области машинного обучения и глубоких нейронных сетей

# Программное обеспечение Deep Learning: Deeplearning4j

- Deeplearning4j (DL4j) - JVM-фреймворк (Java Virtual Machine) для решения задач, связанных с большими данными

# Вопросы

?