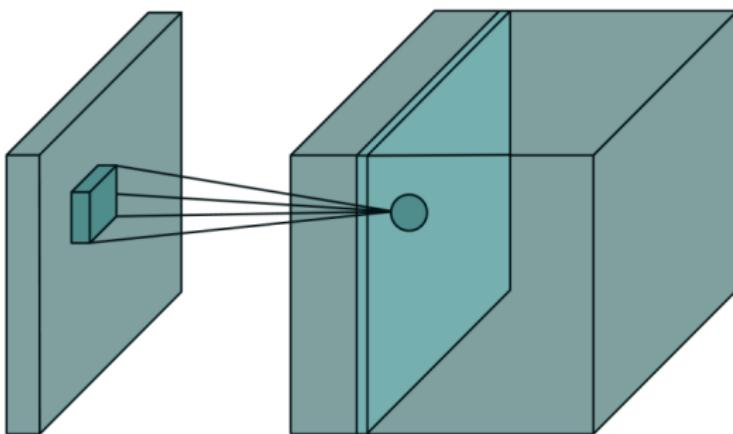


# Свёрточные нейросети

Виктор Китов  
[victorkitov.github.io](http://victorkitov.github.io)



# Содержание

- 1 Численное представление изображений
- 2 Одномерная свёртка
- 3 Двумерная свертка
- 4 Пулинг
- 5 Особые виды свёрток
- 6 Прореживание и нормализация

## Черно-белое (grayscale) изображение

Черно-белое изображение (в оттенках серого, grayscale image) - матрица интенсивностей каждого пикселя:

- размера  $H \times W$
- целочисленная (значения 0-255 для 8 bit)
- элемент  $i, j$ : яркость в позиции



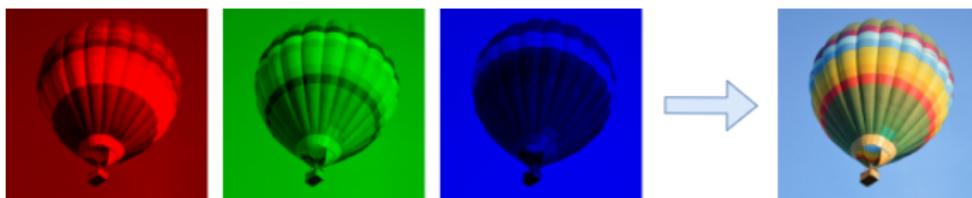
150	90	80	150
110	75	85	110
120	40	35	150
150	100	130	150

150	90	80	150
110	75	85	110
120	40	35	150
150	100	130	150

## Цветное (color) изображение

Цветное (color) изображение - тензор размера  $CxHxW$ ,  $C = 3$   
- число каналов (RGB)

- тензор - обобщение матриц
- в pytorch используются тензоры  $BxCxHxW$



## Фото Прокудина-Горского

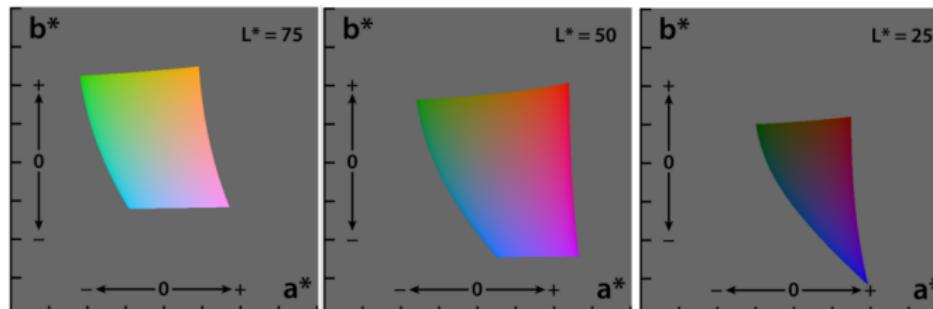
В начале 20 века Прокудин-Горский делал цветные фото Российской империи

- фотографирование: 3 раза через красный, синий и зеленый фильтр
- отображение: на проекторе через 3 фильтра



## Цветовая схема CIELab

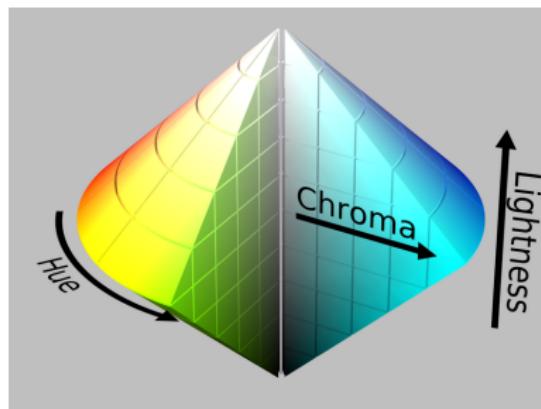
- CIELab (Lab, Luv) разделяет L-luminance (яркость), и (a,b)  
- цвет
- Разделение на яркость и цвет позволяет моделям работать инвариантно
  - к степени освещенности (L): по (a,b)
  - к цветовым различиям (a,b): по L
- Масштабирование осей: одинаковым численным изменениям в Lab => одинаковая воспринимаемая разница человеком (в RGB не так)



## Цветовая схема HCL

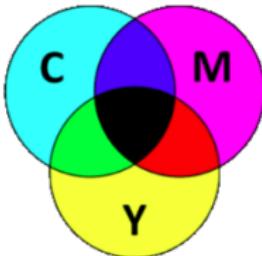
HCL (hue-chroma-lightness) кодирует цвет интерпретируемыми характеристиками:

- цвет (hue), насыщенность цвета (chroma), яркость (lightness)
- последняя компонента кодирует черно-белую версию изображения
- более интерпретируемые компоненты

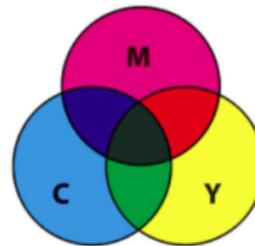


## Цветовая схема CYMK

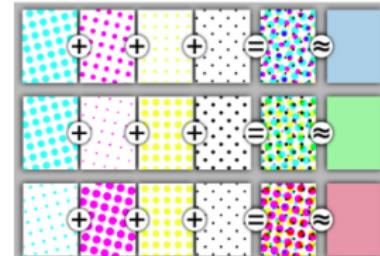
- CMYK (cyan-magenta-yellow-key): используется в типографии
  - базис из голубого (cyan), пурпурного (magenta) и желтого (yellow)
  - в отличие от предыдущих цвета вычитаются (из-за смеси красок), а не складываются (за счет совместного свечения)
  - CMY краски при смещивании дают грязно-коричневый, а не черный, поэтому отдельно добавляют черный (key color)
    - также черная краска сама по себе дешевле и контролирует яркость



Смешивание красок в теории



Смешивание реальных красок

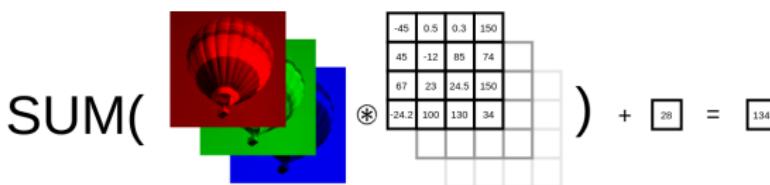


Изображение цвета в типографии

## Линейный подход к классификации

- Линейный классификатор изображений

$$\langle \text{Linearize}(image), \mathbf{w} \rangle + b = \text{class score}$$



- $\mathbf{w}, b$  - свое для каждого класса
- Какие недостатки подхода?

## Недостатки подхода

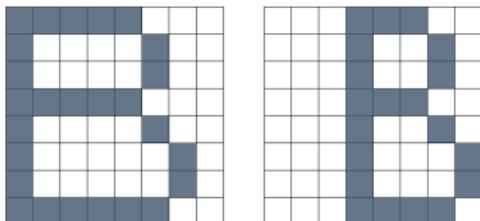
- Много параметров:  $w_c \in \mathbb{R}^{3xHxW}$ ,  $c = 1, 2, \dots, C$ . Метод переобучается на каждый пиксель изображения.
  - при этом модель линейна (недостаточно выразительна)
- Детектируем объекты только в фиксированной области.
  - можно отчасти решить аугментацией

Визуализации  $w$  как изображения:



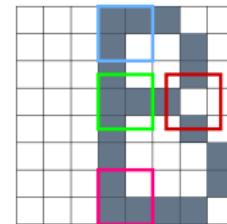
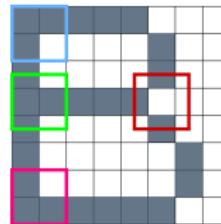
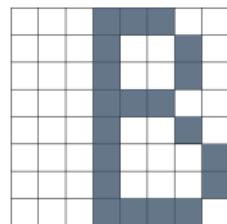
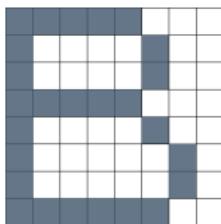
## Инвариантность к позициям и трансформациям

- Интересуемый объект может
  - быть сдвинут
  - по-разному изображен
- Хотим решение, инвариантное к трансформациям.



## Инвариантность к позициям и трансформациям

- Интересуемый объект может
  - быть сдвинут
  - по-разному изображен
- Хотим решение, инвариантное к трансформациям.



## Содержание

- 1 Численное представление изображений
- 2 Одномерная свёртка
- 3 Двумерная свертка
- 4 Пулинг
- 5 Особые виды свёрток
- 6 Прореживание и нормализация

## Понятие свертки

свертка (convolution) в математике:

непр. случай:  $(f * k)(x) = \int_{\mathbb{R}} f(x - y)k(y)dy$

дискр. случай:  $(f * k)(x) = \sum_{y \in \mathbb{Z}} f(x - y)k(y)$

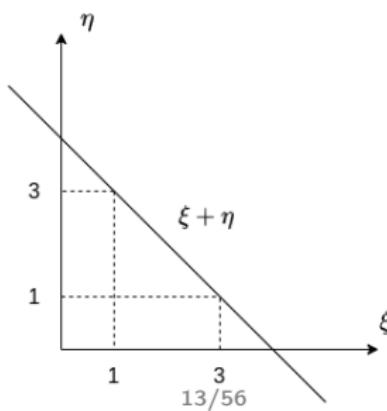
## Понятие свертки

свертка (convolution) в математике:

непр. случай:  $(f * k)(x) = \int_{\mathbb{R}} f(x - y)k(y)dy$

дискр. случай:  $(f * k)(x) = \sum_{y \in \mathbb{Z}} f(x - y)k(y)$

Пример: плотность суммы сл. вел.



## Понятие свёртки

Дискр. случай, когда  $k(j) = 0$  при  $|j| > n$ :

$$\begin{aligned} (f * k)(x) &= \sum_{j \in \mathbb{Z}} f(x - j)k(j) = f(x)k(0) + \\ &+ f(x + 1)k(-1) + \dots + f(x + n)k(-n) \\ &+ f(x - 1)k(+1) + \dots + f(x - n)k(+n) \end{aligned}$$

В нейросетях используют индексацию  $K(n + i) = k(-i)$ , добавляют смещение  $b$ ,  $K(\cdot)$ -"ядро" свертки:

$$out(x, y) = \sum_{i=-n}^n K(i + n)in(x + i) + b, \quad K \in \mathbb{R}^{2n+1}, \quad b \in \mathbb{R}$$

Пример: stride=1

временной ряд:

1	2	3	4	3	2	1
---	---	---	---	---	---	---

ядро свёртки:

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

смещение:

+10
-----

1	2	3	4	3	2	1
1	0	-1				
+10						
8						

Пример: stride=1

временной ряд:

1	2	3	4	3	2	1
---	---	---	---	---	---	---

ядро свёртки:

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

смещение:

+10
-----



Пример: stride=1

временной ряд:

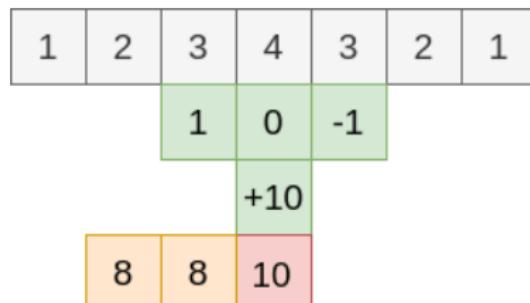
1	2	3	4	3	2	1
---	---	---	---	---	---	---

ядро свёртки:

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

смещение:

+10



Пример: stride=1

временной ряд:

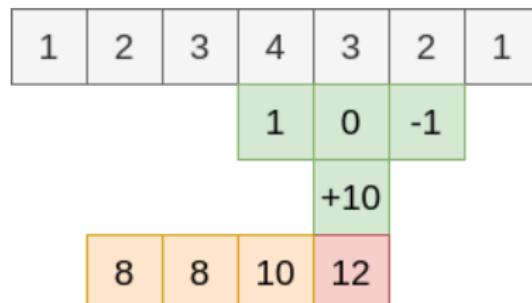
1	2	3	4	3	2	1
---	---	---	---	---	---	---

ядро свёртки:

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

смещение:

+10
-----



Пример: stride=1

временной ряд:

1	2	3	4	3	2	1
---	---	---	---	---	---	---

ядро свёртки:

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

смещение:

+10
-----



Пример: stride=2

временной ряд:

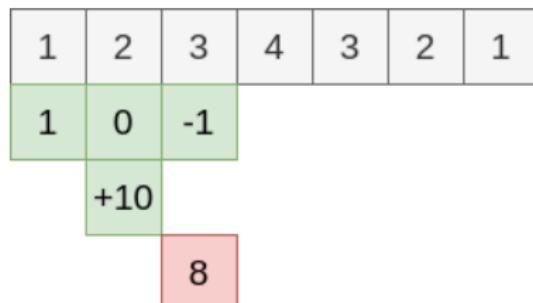
1	2	3	4	3	2	1
---	---	---	---	---	---	---

ядро свёртки:

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

смещение:

+10
-----



Пример: stride=2

временной ряд:

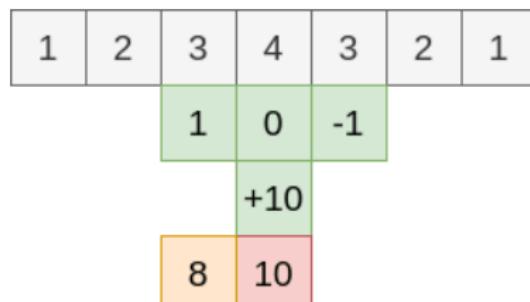
1	2	3	4	3	2	1
---	---	---	---	---	---	---

ядро свёртки:

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

смещение:

+10



Пример: stride=2

временной ряд:

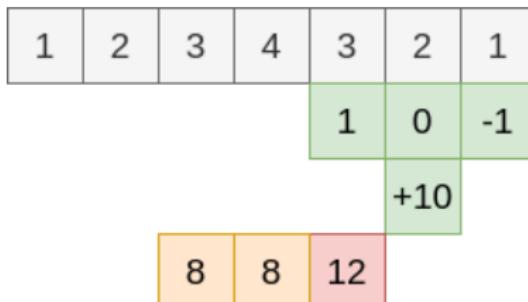
1	2	3	4	3	2	1
---	---	---	---	---	---	---

ядро свёртки:

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

смещение:

+10
-----



# Расширение временного ряда (отступ, padding)

- Свертка уменьшает длину выхода.
- Контроль размера выхода - расширение входа (padding).

исходный ряд [valid padding]

1	2	3	4	3	2	1
---	---	---	---	---	---	---

zero padding [нули, можно др. константой]

0	0	0	1	2	3	4	3	2	1	0	0	0
---	---	---	---	---	---	---	---	---	---	---	---	---

same padding [прологнация]

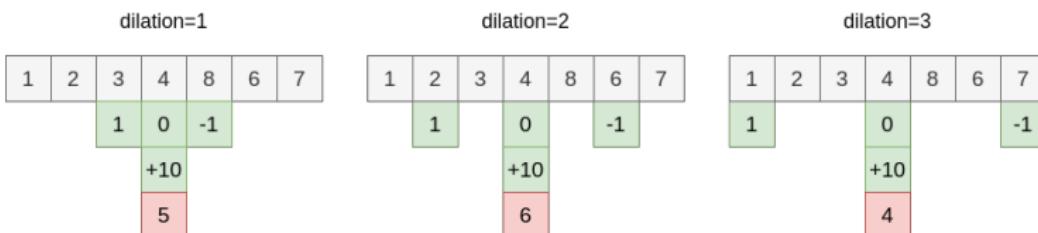
1	1	1	1	2	3	4	3	2	1	1	1	1
↑	↑	↑								↑	↑	↑

mirror padding [отражение]

3	2	1	1	2	3	4	3	2	1	1	2	3
↑	↑	↑								↑	↑	↑

# Расширенная свёртка (dilated convolution)

- Параметр расширения (dilation) задает смещение во входном ряде при применении свёртки.
- Позволяет увеличить область видимости свёрткой (receptive field)
  - при том же #параметров



## Задача

Параметры:

- $W$  - длина входного ряда;
- $2n + 1$  - размер ядра (м. быть и чётным)
- $P$  - расширение (padding) для увеличения размера выхода
- $S$  - шаг при применении (stride);  $D$  - смещение внутри применения (dilation)

Какой будет размер выхода свертки при данных параметрах?

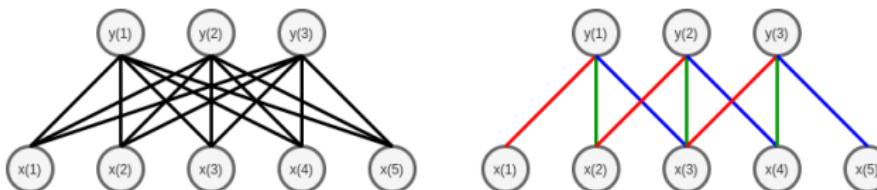
## Примеры сверток

- Примеры свёрток:
  - $K = (\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5})$  - равномерное усреднение.
  - $K = (0.1, 0.2, 0.4, 0.2, 0.1)$  - усреднение с убывающими весами.
  - $K = (-1, 0, +1)$  - величина изменения ( $f' \approx z_{t+1} - z_{t-1}$ )
  - $K = (+1, -2, +1)$  - динамика величины изменения ( $f'' \approx (z_{t+1} - z_t) - (z_t - z_{t-1})$ )
- Свёртка извлекает локальный линейный признак.
  - делает это независимо от размера входа.
- Если настраивать извлекаемый признак автоматически - лучше работает.

# Преимущества свёрток

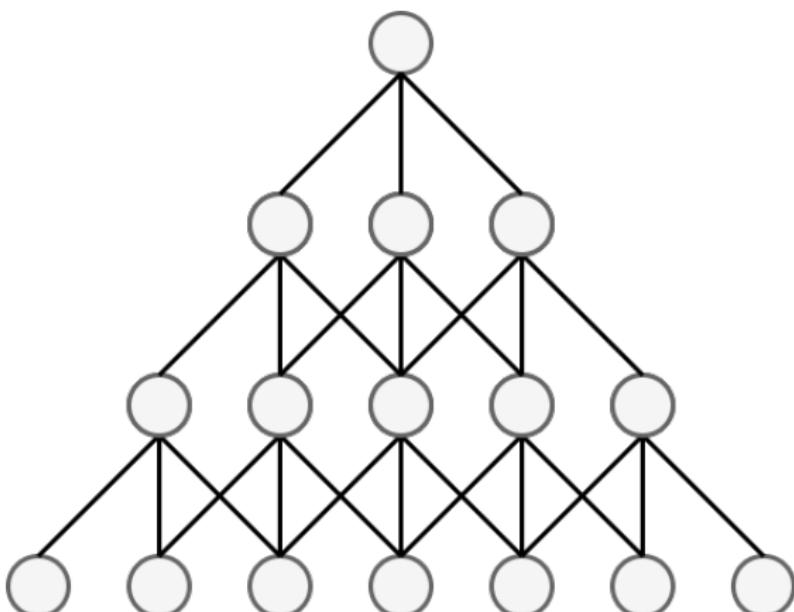
Преимущества свёрток:

- признак извлекается локально (sparse connections)
- общие параметры для разных участков (parameter sharing)

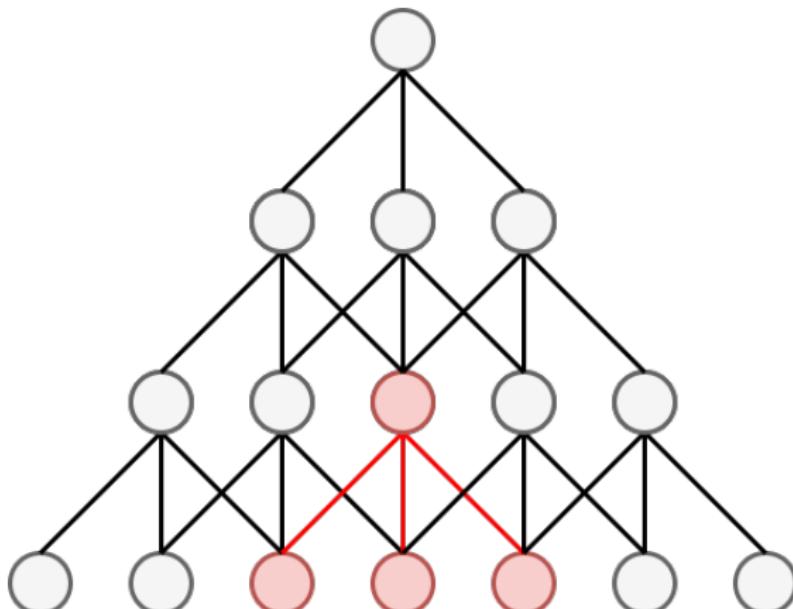


- извлекает линейный признак (быстро)
- применима ко входам произвольного размера
  - большего, чем размер ядра свёртки
- наславивая свёртки друг на друга с нелинейностью, можем извлекать более сложные признаки с расширенной областью видимости (receptive field)

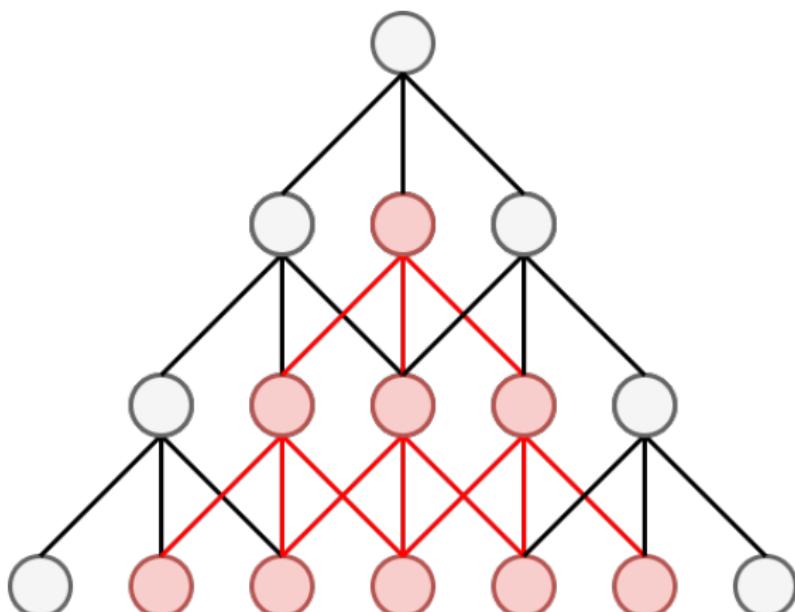
## Наслаивание свёрток, область видимости



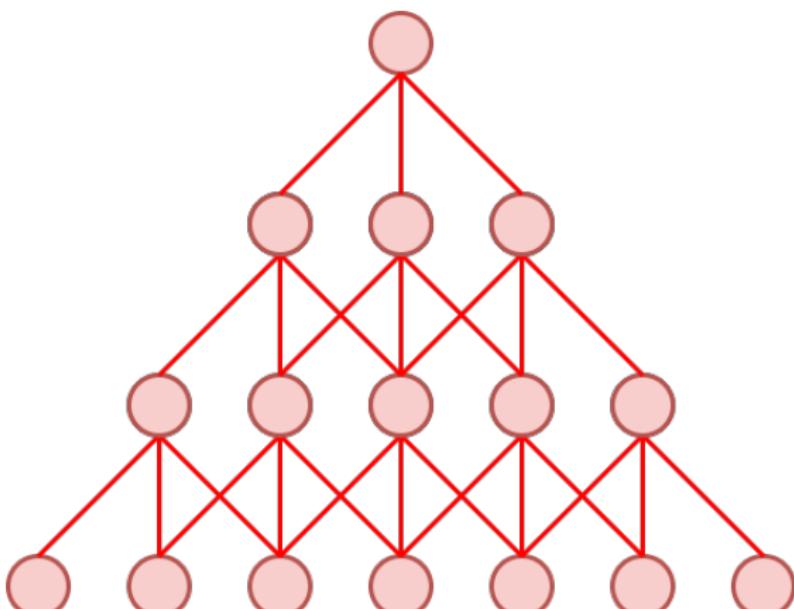
## Наслаивание свёрток, область видимости



## Наслаивание свёрток, область видимости



## Наслаивание свёрток, область видимости



## Свёрточные нейросети для анализа текстов

- Рассмотрим последовательность слов  $w_{1:n} = w_1, \dots, w_n$ , векторн. представление  $w_i$  -  $\mathbf{w}_i \in \mathbb{R}^{d_{emb}}$
- Определим конкатенацию  $\oplus(\mathbf{w}_{i:i+k-1}) = [\mathbf{w}_i; \dots; \mathbf{w}_{i+k-1}] \in \mathbb{R}^{k \cdot d_{emb}}$  ( $k$  - размер ядра свёртки).

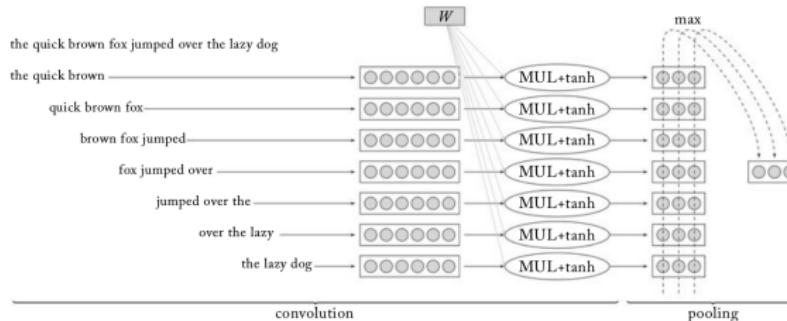
Свёртка ( $g(\cdot)$ -нелинейность,  $b \in \mathbb{R}$ ):

- для  $i = 1, \dots, n - k + 1$ :
  - $\mathbf{x}_i = \oplus(\mathbf{w}_{i:i+k-1})$ ;  $p_i = g(x_i^T \cdot u + b)$

На практике извлекаются  $l$  фильтров ( $U \in \mathbb{R}^{k \cdot d_{emb} \times l}$ ,  $b \in \mathbb{R}^l$ ):

- для  $i = 1, \dots, n - k + 1$ :
  - $\mathbf{x}_i = \oplus(\mathbf{w}_{i:i+k-1})$ ;  $\mathbf{p}_i = g(x_i^T U + \vec{b})$

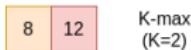
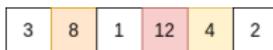
# Свёрточные нейросети для анализа текстов



- Варианты пулинга:
  - MaxPooling:  $c_j = \max_{1 \leq i \leq n-k+1} p_i[j], \quad j = \overline{1, l}$ .
  - AvgPooling:  $c_j = \frac{1}{n-k+1} \sum_{i=1}^{n-k+1} p_i[j], \quad j = \overline{1, l}$ .
- Пулинг преобразует  $[p_1, \dots, p_{n-k+1}] \in \mathbb{R}^{n-k+1 \times l}$  в вектор фикс. длины.
  - можем подать на вход MLP или др. модели
- Свёрточные нейросети для текстов хорошо учитывают фразы ("очень понравилось", "не понравилось"), но не учитывают более долгосрочные зависимости.

# Усложнения пулинга для последовательностей

Pooling для последовательностей:



- K-max: К максимальных элементов возвращаются в исходной последовательности.
- dynamic K-max<sup>1</sup>: К определяется автоматически.

<sup>1</sup><https://arxiv.org/pdf/1404.2188.pdf>

## Содержание

- 1 Численное представление изображений
- 2 Одномерная свёртка
- 3 Двумерная свертка
- 4 Пулинг
- 5 Особые виды свёрток
- 6 Прореживание и нормализация

## Идеология свёртки: извлечение локального признака



## Идеология свёртки: извлечение локального признака



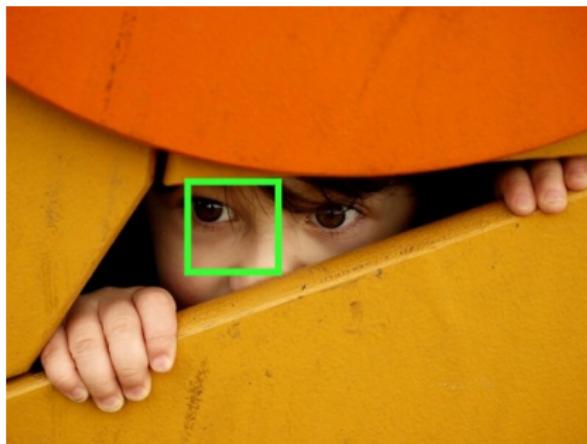
## Идеология свёртки: извлечение локального признака



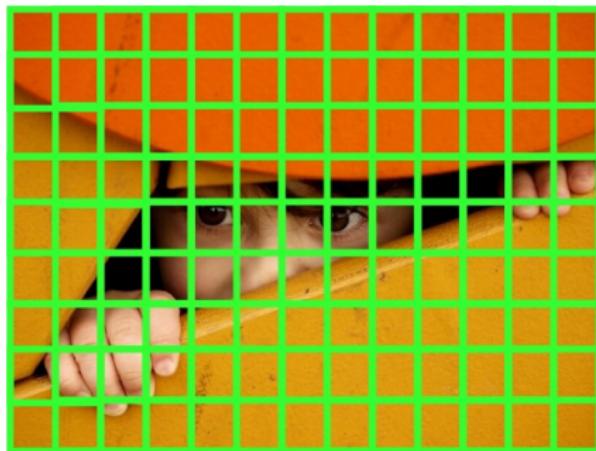
## Идеология свёртки: извлечение локального признака



## Идеология свёртки: извлечение локального признака



## Идеология свёртки: извлечение локального признака



# Примеры применения стандартных свёрток<sup>2</sup>

$$\frac{1}{9} \cdot \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{matrix}$$

Unweighted 3x3 smoothing kernel

$$\frac{1}{8} \cdot \begin{matrix} 0 & 1 & 0 \\ 1 & 4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

Weighted 3x3 smoothing kernel with Gaussian blur

$$\begin{matrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{matrix}$$

Kernel to make image sharper

$$\begin{matrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{matrix}$$

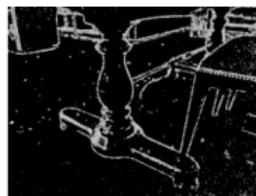
Intensified sharper image



Gaussian Blur



Sharpened image



X – Direction Kernel

$$\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$$

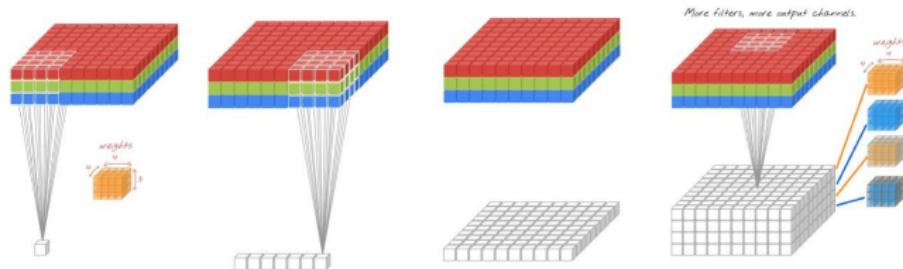
Y – Direction Kernel

$$\begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix}$$

- В нейросетях параметры свёрток настраиваются - лучше работает.

<sup>2</sup>Представим ли медианный фильтр в виде свёртки?

# Двумерная свертка<sup>3</sup>



$$\mathbf{out}(x, y, s) =$$

$$\sum_{i=-n}^n \sum_{j=-n}^n \sum_{c=1}^{C_{in}} K_s(i+n, j+n, c) \mathbf{in}(x+i, y+j, c) + b_s,$$

$$K_s \in \mathbb{R}^{(2n+1)x(2n+1) \times C}, \quad b_s \in \mathbb{R}, \quad s = 1, \dots C_{out}$$

Глубина свёртки=глубине входного тензора.

Выходное число каналов = числу применяемых свёрток.

<sup>3</sup>Сколько параметров у одного свёрточного слоя? (с учетом смещений)

# Применение двумерной свертки (1 канал)<sup>4</sup>

## Применение двумерной свертки:

3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1 <sub>3</sub>	0 <sub>4</sub>
0 <sub>2</sub>	0 <sub>2</sub>	1 <sub>0</sub>	3 <sub>1</sub>	1
3 <sub>0</sub>	1 <sub>1</sub>	2 <sub>2</sub>	2 <sub>3</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>2</sub>	2 <sub>2</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>0</sub>	1 <sub>1</sub>	

12.0 12.0 17.0  
10.0 17.0 19.0  
9.0 6.0 14.0

3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1 <sub>3</sub>	0 <sub>4</sub>
0 <sub>0</sub>	0 <sub>2</sub>	1 <sub>1</sub>	3 <sub>0</sub>	1
3 <sub>2</sub>	1 <sub>2</sub>	2 <sub>0</sub>	2 <sub>3</sub>	
2 <sub>0</sub>	0 <sub>1</sub>	0 <sub>2</sub>	2 <sub>2</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>1</sub>	0 <sub>1</sub>	

12.0 12.0 17.0  
10.0 17.0 19.0  
9.0 6.0 14.0

3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1 <sub>3</sub>	0 <sub>4</sub>
0 <sub>0</sub>	0 <sub>2</sub>	1 <sub>1</sub>	3 <sub>0</sub>	1 <sub>0</sub>
3 <sub>1</sub>	2 <sub>0</sub>	2 <sub>1</sub>	3 <sub>2</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>2</sub>	2 <sub>2</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>0</sub>	0 <sub>1</sub>	

12.0 12.0 17.0  
10.0 17.0 19.0  
9.0 6.0 14.0

3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1 <sub>3</sub>	0 <sub>4</sub>
0 <sub>0</sub>	0 <sub>1</sub>	1 <sub>3</sub>	3 <sub>1</sub>	1
3 <sub>2</sub>	1 <sub>2</sub>	2 <sub>0</sub>	2 <sub>3</sub>	
2 <sub>0</sub>	0 <sub>1</sub>	0 <sub>2</sub>	2 <sub>2</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>1</sub>	0 <sub>1</sub>	

12.0 12.0 17.0  
10.0 17.0 19.0  
9.0 6.0 14.0

3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1 <sub>3</sub>	0 <sub>4</sub>
0 <sub>0</sub>	0 <sub>2</sub>	1 <sub>1</sub>	3 <sub>0</sub>	1
3 <sub>2</sub>	1 <sub>2</sub>	2 <sub>0</sub>	2 <sub>3</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>2</sub>	2 <sub>2</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>0</sub>	0 <sub>1</sub>	

12.0 12.0 17.0  
10.0 17.0 19.0  
9.0 6.0 14.0

3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1 <sub>3</sub>	0 <sub>4</sub>
0 <sub>0</sub>	0 <sub>2</sub>	1 <sub>1</sub>	3 <sub>1</sub>	1 <sub>2</sub>
3 <sub>1</sub>	2 <sub>0</sub>	2 <sub>2</sub>	3 <sub>0</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>2</sub>	2 <sub>2</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>0</sub>	0 <sub>1</sub>	

12.0 12.0 17.0  
10.0 17.0 19.0  
9.0 6.0 14.0

3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1 <sub>3</sub>	0 <sub>4</sub>
0 <sub>0</sub>	0 <sub>1</sub>	1 <sub>3</sub>	3 <sub>1</sub>	1
3 <sub>0</sub>	1 <sub>1</sub>	2 <sub>2</sub>	2 <sub>3</sub>	
2 <sub>2</sub>	0 <sub>2</sub>	0 <sub>0</sub>	2 <sub>2</sub>	
2 <sub>0</sub>	0 <sub>1</sub>	0 <sub>2</sub>	1 <sub>1</sub>	

12.0 12.0 17.0  
10.0 17.0 19.0  
9.0 6.0 14.0

3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1 <sub>3</sub>	0 <sub>4</sub>
0 <sub>0</sub>	0 <sub>1</sub>	1 <sub>3</sub>	3 <sub>1</sub>	1
3 <sub>2</sub>	1 <sub>2</sub>	2 <sub>0</sub>	2 <sub>3</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>2</sub>	2 <sub>2</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>1</sub>	0 <sub>2</sub>	

12.0 12.0 17.0  
10.0 17.0 19.0  
9.0 6.0 14.0

3 <sub>0</sub>	3 <sub>1</sub>	2 <sub>2</sub>	1 <sub>3</sub>	0 <sub>4</sub>
0 <sub>0</sub>	0 <sub>1</sub>	1 <sub>3</sub>	3 <sub>1</sub>	1
3 <sub>1</sub>	2 <sub>0</sub>	2 <sub>1</sub>	3 <sub>2</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>2</sub>	2 <sub>0</sub>	
2 <sub>0</sub>	0 <sub>0</sub>	0 <sub>1</sub>	0 <sub>1</sub>	

12.0 12.0 17.0  
10.0 17.0 19.0  
9.0 6.0 14.0

<sup>4</sup>Иллюстрации: Dumoulin et al. 2018.



## Содержание

- 1 Численное представление изображений
- 2 Одномерная свёртка
- 3 Двумерная свертка
- 4 Пулинг
- 5 Особые виды свёрток
- 6 Прореживание и нормализация

# Агрегация

- Max pooling: максимум по квадратной области
  - "признак присутствует где-то в области"
  - при дифференцировании градиент идет в тах элемент
- Average pooling: среднее по квадратной области
  - "средняя представленность признака"
  - можно усреднять с весами
- Независимо по каналам (#каналов не меняется)
- Обычно со  $\text{stride}=\text{kernel size}>1$  ( $\text{dilation}=1$ )
  - уменьшение пространственного разрешения.
- Добавляет инвариантность к небольшим сдвигам.

INPUT

1	2	5	6
3	4	7	8
9	10	13	14
11	12	15	16

MAX  
pooling

4	8
12	16

AVERAGE  
pooling

2.5	6.5
10.5	14.5

# Max pooling, stride=1

Max pooling 3x3, stride=1:

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

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

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

3.0	3.0	3.0
3.0	3.0	3.0
3.0	2.0	3.0

# Стохастический пулинг (stochastic pooling<sup>5</sup>)

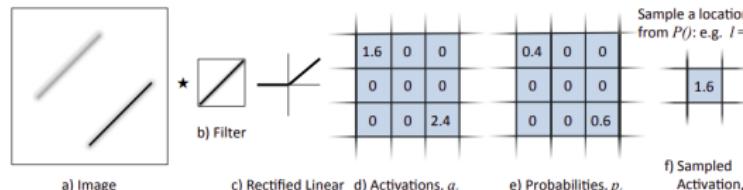
**ОБУЧЕНИЕ** (активации  $a_k \geq 0$  - после ReLU):

- для активаций  $\{a_k\}_k$  окрестности считаем их  $p_i = \frac{a_i}{\sum_k a_k}$

$$StochasticPooling(\{a_k\}_k) = \begin{cases} a_1, & \text{с вероятностью } p_1 \\ a_2 & \text{с вероятностью } p_2 \\ \dots & \dots \end{cases}$$

**ПРИМЕНЕНИЕ:**

$$StochasticPooling(\{a_k\}_k) = \sum_k p_k a_k$$



<sup>5</sup><https://arxiv.org/abs/1301.3557>

# Стохастический пулинг (stochastic pooling<sup>6</sup>)

Комментарии:

- в обучении более высокие значения выбираются с большей вероятностью
- мотивация: аналог dropout для изображений:
  - обучение: параллельно учим ансамбль моделей
  - применение: усредняем по ансамблю
  - нет переобучения по одному выходу фильтра, важна вся совокупность значений

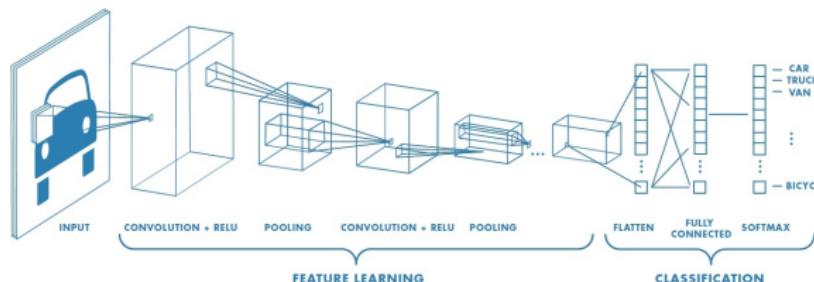
---

<sup>6</sup><https://arxiv.org/abs/1301.3557>

# Свёрточные нейросети (CNN, ConvNet)<sup>7</sup>

Свёрточная нейросеть для анализа изображений:  
 [свёртка->нелинейность->пулинг] $\times N$  -> MLP.

- Последующие свёртки извлекают более сложные и глобальные признаки:
  - пример: цвета->границы->линии, углы, изгибы->геометрические фигуры->глаза, нос, рот, брови
- Постепенно  $\downarrow$  разрешение и  $\uparrow$  число каналов. Conv: большая часть вычислений, MLP: большая часть пар-ров.

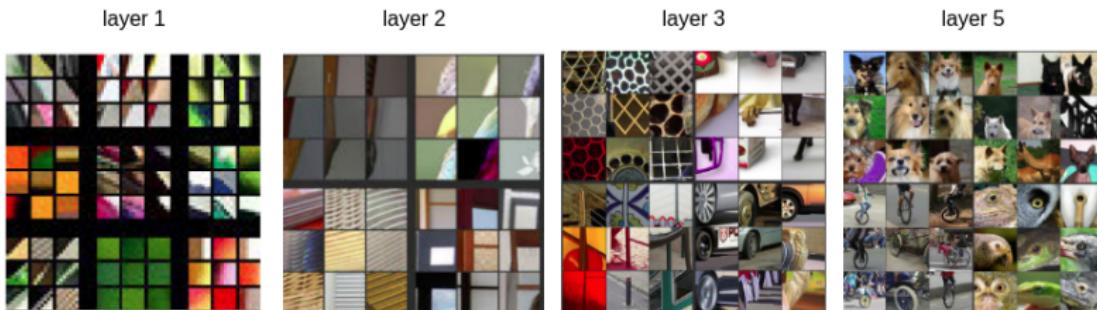


<sup>7</sup>  $\text{Pool}(\text{NonLinearity}(\text{Conv}(x))) = \text{NonLinearity}(\text{Pool}(\text{Conv}(x)))$  для max и avg pooling?

# Интерпретация свёрточных признаков<sup>8</sup>

Фрагменты, максимально активирующие нейроны разных слоев на валидационном множестве.

- для каждого нейрона - 9 максимально активирующих его фрагментов изображений
- архитектура ~AlexNet



<sup>8</sup><https://arxiv.org/pdf/1311.2901.pdf>

# Альтернативные способы интерпретации

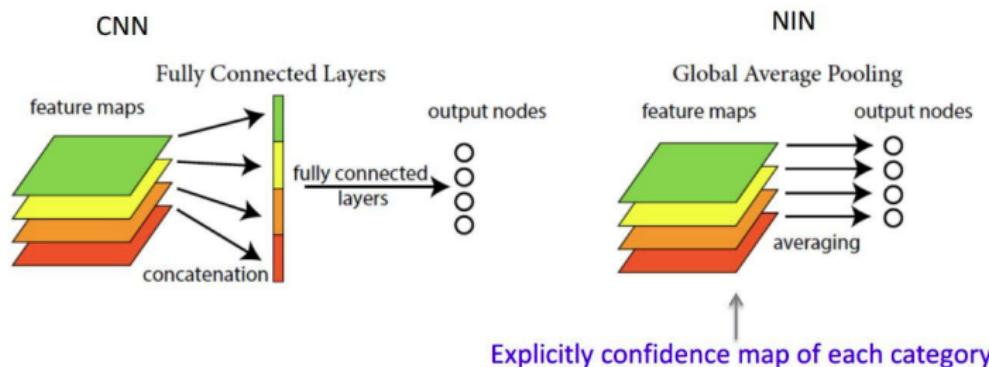
Визуализация сверток 1го слоя сети AlexNet



- Свертки 1го слоя используют ядра  $\in \mathbb{R}^{(2n+1) \times (2n+1) \times 3}$ , поэтому м. быть визуализированы как изображения.
  - соответствует результатам в нейрофизиологии
- Оптимизация участка изображения, максимизирующего активацию.
  - backprop по пикселям при фикс. весах сети

# Глобальный пулинг

Глобальный пулинг преобразует изображение любого размера в эмбеддинг фикс. размера:

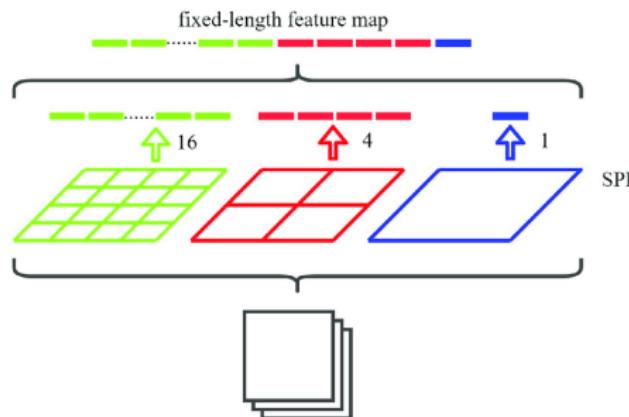


- Интерпретация: можем отследить расположение интересующего объекта
- Также можно подать выход на вход MLP.

# Пространственный пирамидальный пулинг<sup>9</sup>

Пространственный пирамидальный пулинг (spatial pyramid pooling)

- Выдаёт вектор фикс. размера.
- В нём частично сохранена пространственная информация:



<sup>9</sup><https://arxiv.org/pdf/1406.4729.pdf>

## Содержание

- 1 Численное представление изображений
- 2 Одномерная свёртка
- 3 Двумерная свертка
- 4 Пулинг
- 5 Особые виды свёрток
- 6 Прореживание и нормализация

## Сепарабельные свёртки

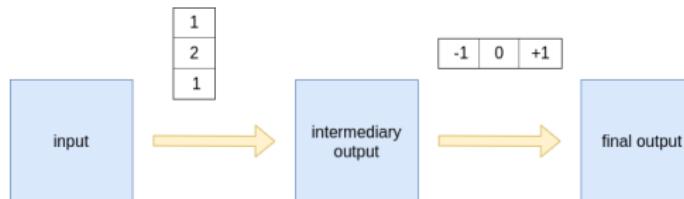
Сепарабельные свёртки (spatially separable convolutions) допускают представление в виде произведения 2x векторов<sup>10</sup>.

пример для одноканального входа:

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

Сепарабельные свёртки (с ядром  $K \in \mathbb{R}^{NxN}$ )

- меньше параметров и вычислений:  $O(N^2) \rightarrow O(2N)$



- применимо:

<sup>10</sup>Что будет в случае входа с несколькими каналами?

## Сепарабельные свёртки

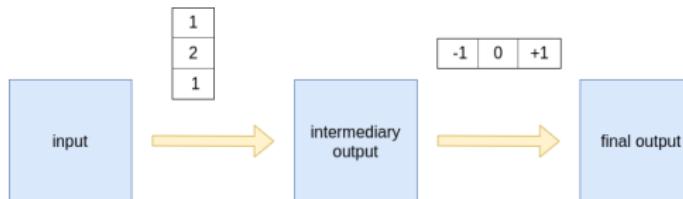
Сепарабельные свёртки (spatially separable convolutions) допускают представление в виде произведения 2x векторов<sup>10</sup>.

пример для одноканального входа:

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

Сепарабельные свёртки (с ядром  $K \in \mathbb{R}^{N \times N}$ )

- меньше параметров и вычислений:  $O(N^2) \rightarrow O(2N)$



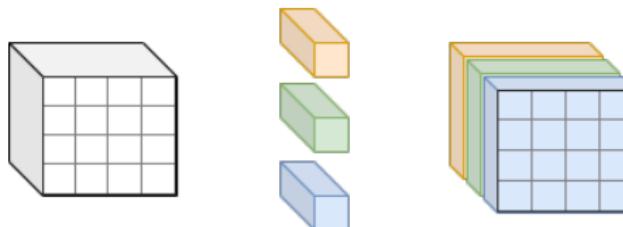
- применимо: только для свёртки с  $\text{rg } K \leq 1$
- обобщение:  $K = A^T B$  для низкоранговых матриц  $A, B$ .

<sup>10</sup>Что будет в случае входа с несколькими каналами?

## Свёртка 1x1 (pointwise convolution)

Свёртка 1x1 (pointwise convolution):  $K \in \mathbb{R}^{C \times 1 \times 1}$ , каждый пиксель зависит от предыдущих каналов такого же пикселя.

- векторная линейная регрессия в каждом пикселе (преобразование признаков), не изменяются HxW
- вычислительно эффективная, можем варьировать C перед применением более сложных свёрток
- можем делать бутылочное горлышко автокодировщика



## Упрощенные виды свёрток

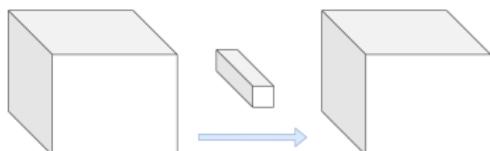
- Групповые (grouped) свёртки: линейный фильтр в рамках группы признаков.
- Поканальные (depthwise) свёртки - это групповые, где  $\#\text{групп} = \#\text{входных каналов}$ .
- Depthwise separable - к выходу depthwise свёрток применяем свёртки  $1 \times 1^{11}$ .
  - получается, выход зависит от всех входных каналов, но не все полные свёрток можем так моделировать.

---

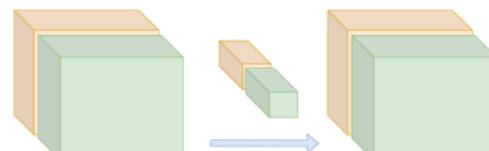
<sup>11</sup>Сравните  $\#\text{параметров}$  и  $\#\text{вычислений}$  у обычного и depthwise separable слоя (depthwise-общий). Различается ли выразительная способность?

# Групповые и поканальные свёртки

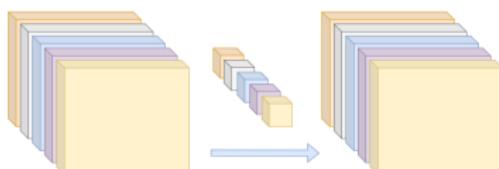
standard convolution



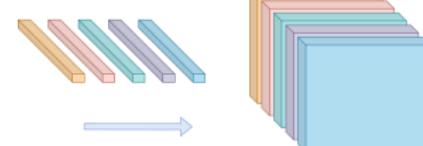
grouped convolution



depthwise convolution



depthwise separable convolution: depthwise conv+свёртки 1x1 в конце:



# Снижение и повышение #признаков

## Bottleneck:

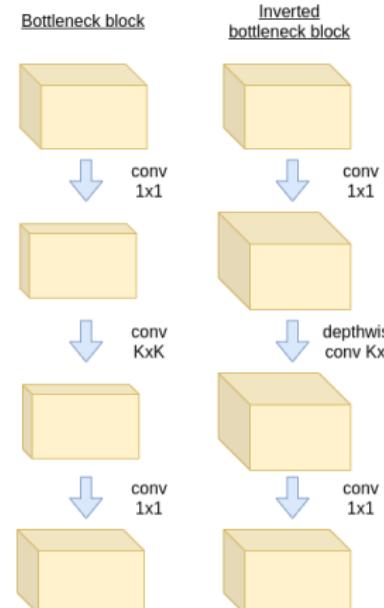
$\downarrow$  #признаков  $\rightarrow$  сложная операция  $\rightarrow$   $\uparrow$  #признаков

- обоснование:  $\downarrow$  #операций и #параметров
- пример: Inception

## Inverted bottleneck:

$\uparrow$  #признаков  $\rightarrow$  depthwise conv  $\rightarrow$   $\downarrow$  # признаков

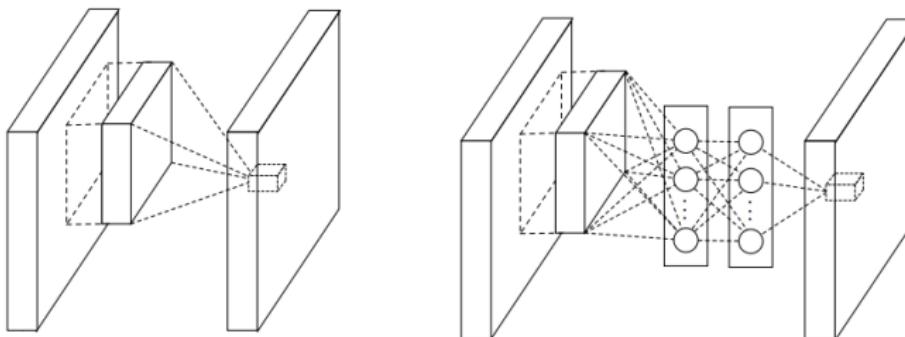
- обоснование:  $\uparrow$  взаимозависимость существующих каналов при depthwise операции, расширяя их линейными комбинациями каналов
- пример: MobileNet V2



## Network in Network<sup>12</sup>

Network in Network: вместо свёртки (линейной операции) использовать MLP к окрестности точек.

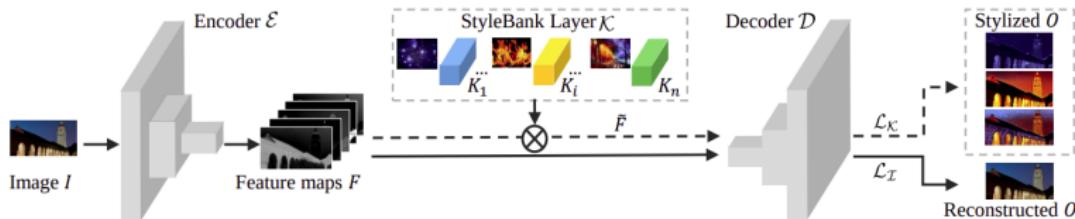
- MLP применяется с одинаковыми весами ко всем участкам изображения.



<sup>12</sup><https://arxiv.org/pdf/1312.4400.pdf>

# Изменяемые параметры свёртки

- Вся архитектура настраивается на класс задач.
- А для каждого подкласса выбирается отдельная свёртка<sup>13</sup>:



- Альтернативно может применяться лин. комбинация фильтров  $w_1K_1 + \dots w_nK_n$ , где веса зависят от входа<sup>14</sup>:

$$[w_1, \dots, w_n] = F(\text{input})$$

- либо предсказывать параметры свёртки напрямую, используя гиперсеть<sup>15</sup>.

<sup>13</sup> StyleBank

<sup>14</sup> Dynamic Convolution: Attention over Convolution Kernels

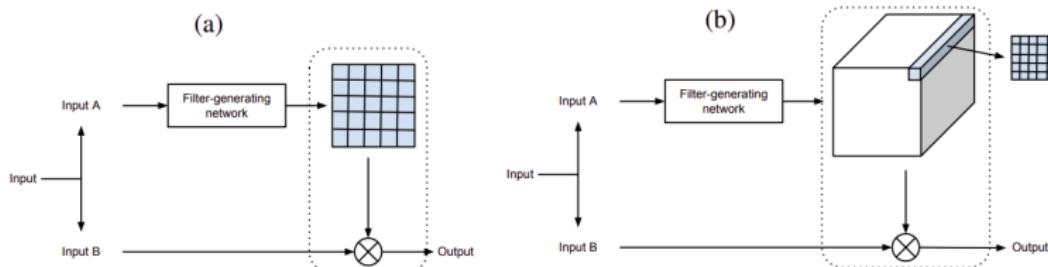
<sup>15</sup> Neural Style Transfer via Meta Networks

## Изменяемые параметры свёртки<sup>16</sup>

Можно применять свёртку, веса которой предсказываются отдельной сетью:

- (a) веса свёртки не зависят от  $(x,y)$  [Dynamic Convolution]
- (b) веса свёртки ещё зависят от  $(x,y)$  [Dynamic local filtering]

Пример: определение человека осуществляется разными фильтрами, если если он сфотографирован в фас и в профиль. По-разному идентифицируем волосы и подбородок.



<sup>16</sup><https://arxiv.org/pdf/1605.09673.pdf>

# Содержание

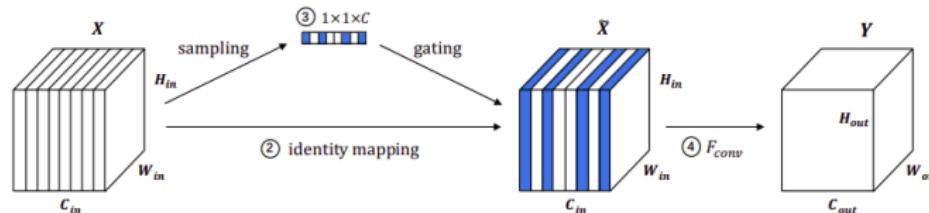
- 1 Численное представление изображений
- 2 Одномерная свёртка
- 3 Двумерная свертка
- 4 Пулинг
- 5 Особые виды свёрток
- 6 Прореживание и нормализация

# Пространственное прореживание (Spatial DropOut)

Стандартный DropOut на уровне нейронов неэффективен: соседние активации сильно коррелированы, и сеть все равно восстанавливает значения по соседям.

Пространственное прореживание<sup>17</sup> (Spatial DropOut, DropChannel):

- training: независимо с вероятностью  $(1 - p)$  зануляются целые каналы.
- test: используются все каналы, умноженные на  $p$
- pytorch: ф-ция `torch.nn.Dropout2d()`

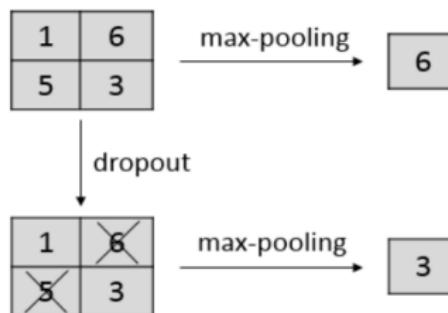


<sup>17</sup> <https://arxiv.org/pdf/1904.03392.pdf>

# Прореживание в агрегации

Max-pooling DropOut<sup>18</sup>:

- training:



- test: отмасштабированный обычный pooling.

<sup>18</sup><https://arxiv.org/pdf/1512.01400.pdf>

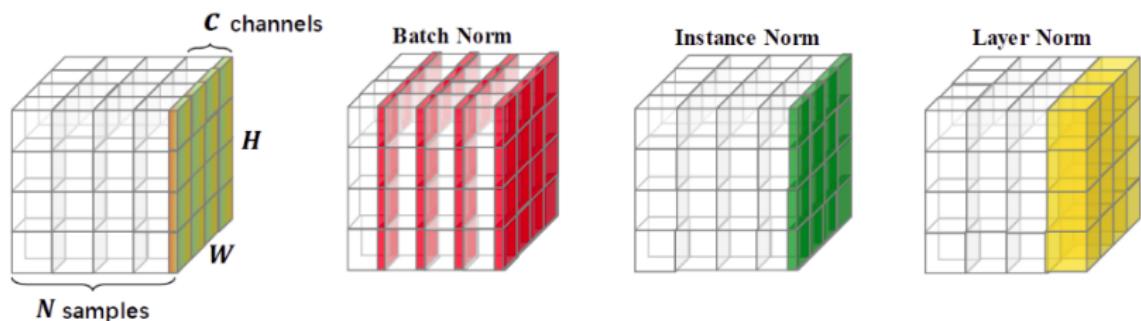
## Нормализация в свёртках

- При применении BatchNorm (nn.BatchNorm2d) предшествующую свёртку нужно запускать с bias=False
  - смещение и так возникнет после нормализации
- Признак=канал, поэтому нормализуется каждый канал
  - nn.BatchNorm2d: усреднение по позициям и объектам батча
  - nn.InstanceNorm2d: усреднение по позициям 1 объекта
    - каждый объект и так даёт HW реализаций признака
    - train и test не различаются (перенормировка per object)
  - nn.LayerNorm<sup>19</sup>: усреднение по всем каналам 1 объекта
    - применимо и к изображениям, и др. объектам
    - train и test не различаются (перенормировка per object)
    - можно разбить каналы на группы и усреднять в рамках группы (GroupNorm).

<sup>19</sup> <https://arxiv.org/abs/1607.06450>

# Нормализация в свёртках

Виды нормализации в свёртках:



## Заключение

- Свёртка - линейный локальный фильтр.
  - раньше подбирали вручную, сейчас обучаются
- Свёрточный слой содержит мало параметров:
  - разреженные связи и общность параметров в разных позициях
- Average и max pooling:
  - снижение пространственного разрешения
  - инвариантность к небольшим сдвигам
- Последующие свертки извлекают более глобальные и сложные признаки.
- Свёрточный слой применим к изображениям любого размера.
  - в конце: MLP после reshape, GlobalAvgPooling или PyramidPooling.
- DropOut - по слоям, нормализация возможна в рамках объекта (InstanceNorm, LayerNorm)