Лекция 4. Свёртки

Денис Деркач, Дмитрий Тарасов

Слайды от А. Маевского, М. Гущина, А. Кленицкого, М Борисяка,

А. Устюжанина, А. Зимовнова

9 февраля 2026 года

LAMBDA • HSE

Картинки

Распознавание картинок

Banksy

- "Ребёнок"
- "Медсестра"
- "Сверхгерой"
- "Банкси"
- "Игра"
- "Картинка"
- ?

Распознавание картинок

Banksy

Чёрно-белое изображение представляет собой матрицу пикселей [HxW]

Pixel = **pic**ture **el**ements

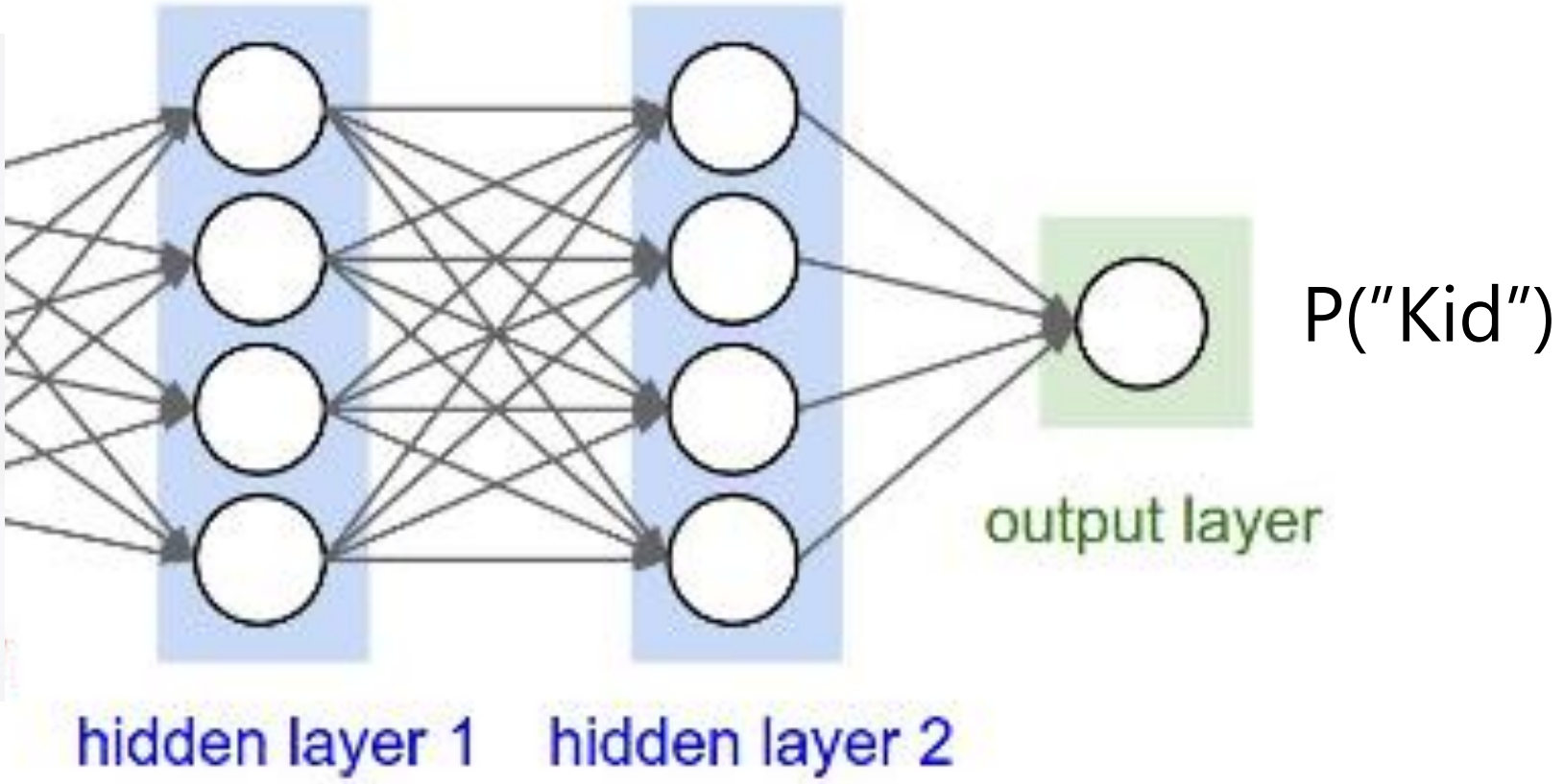
Каждый пиксель хранит яркость [0,255].

Распознавание картинок

Banksy

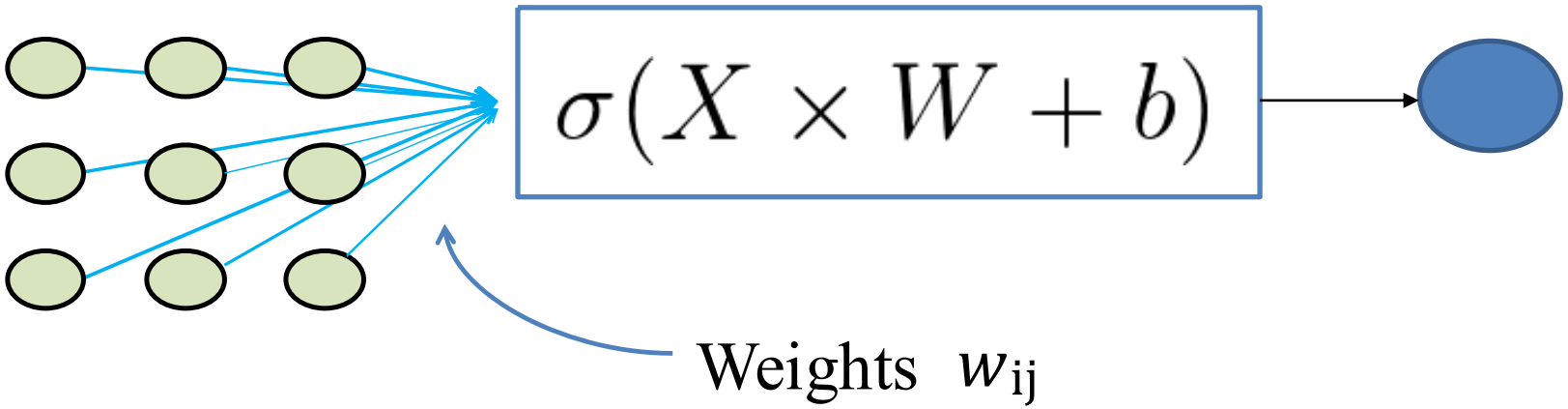
Изображение RGB представляет собой 3D-массив
[HxWx3] или [3xHxW]

Общая идея

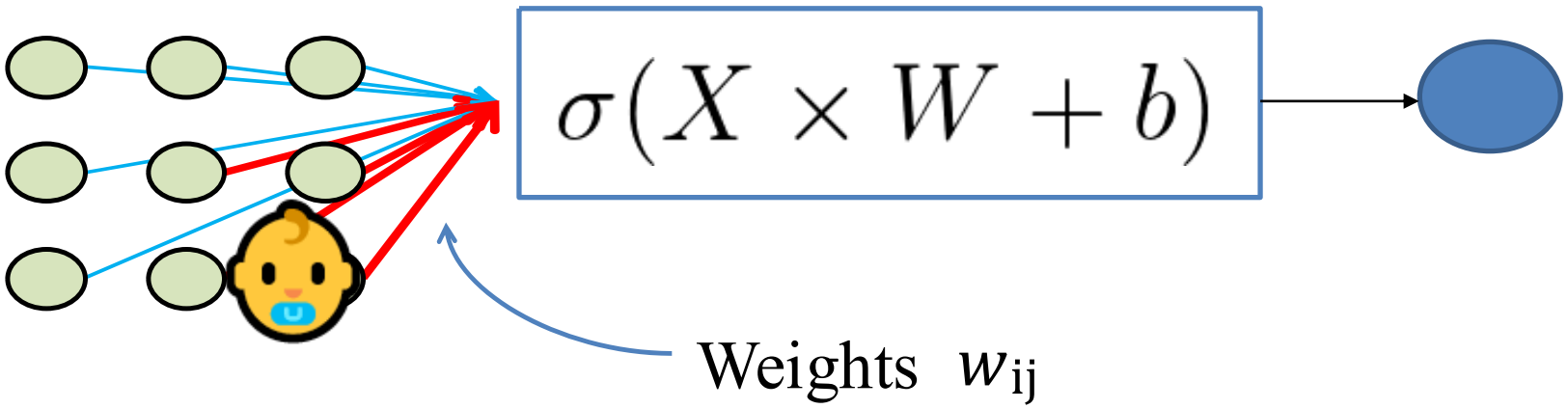


Использовать нейронную сеть как классификатор картинок?

Проблемы картинок

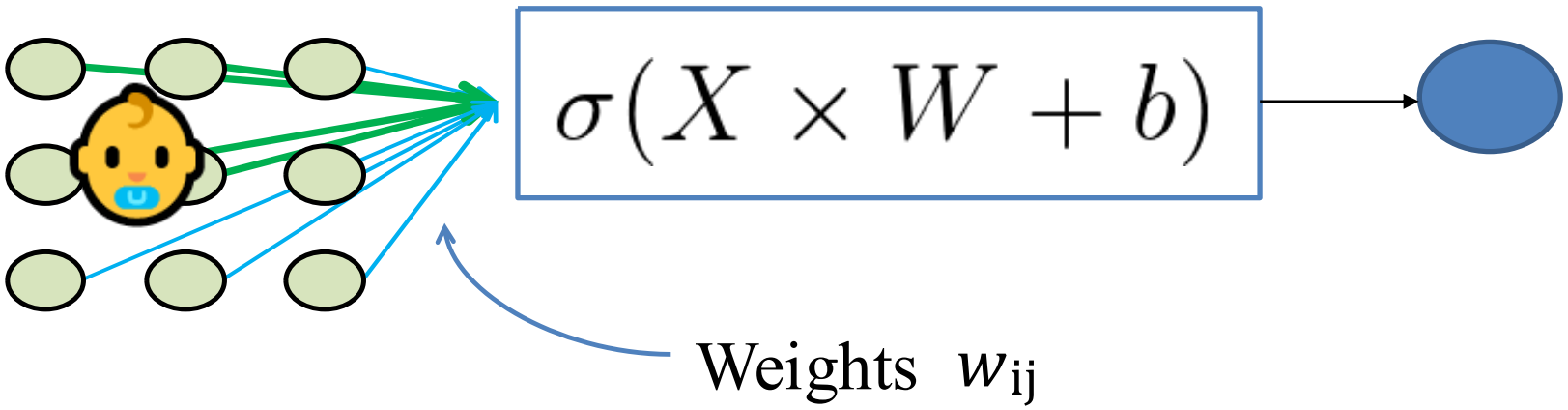


Проблемы картинок



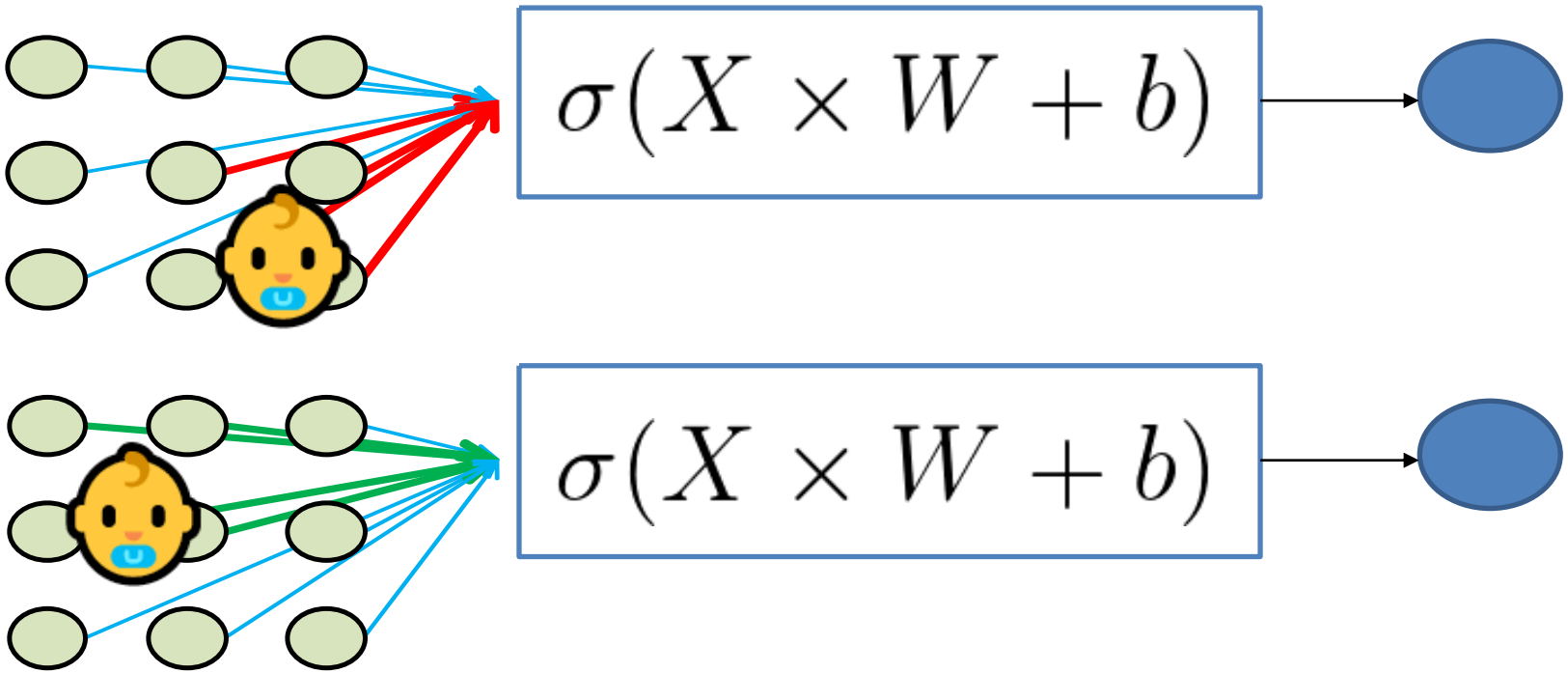
Если лицо находится снизу справа, можно тренировать нейроны, которые ведут снизу справа

Проблемы картинок



Если лицо находится сверху слева, можно тренировать нейроны, которые ведут сверху слева

Проблемы картинок



Нейронной сети придется изучать эти два случая отдельно!
Худший случай: один нейрон на позицию.

MLP и картинки

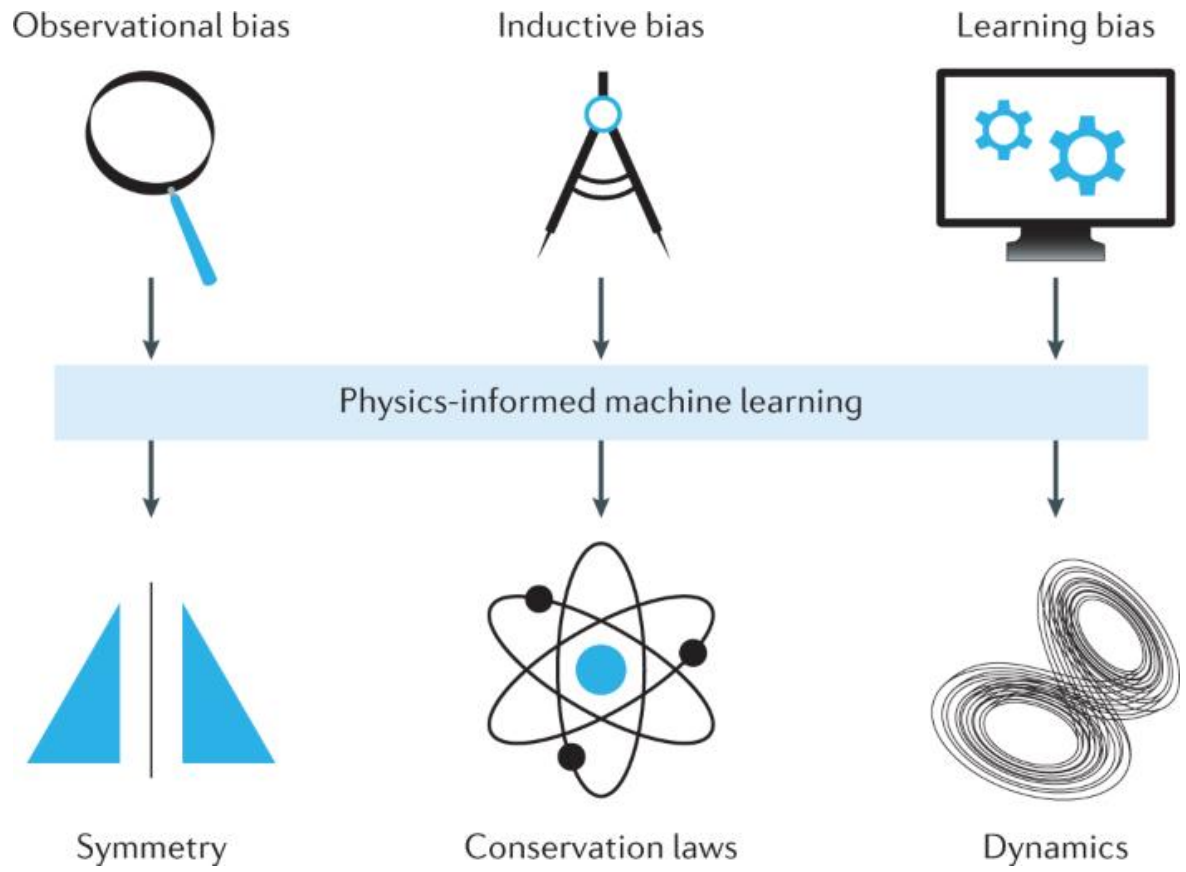
Проблемы с обычными нейросетями (MLP):

- Слишком много параметров – переобучение $224 \times 224 \times 3$ изображение ~ 150к измерений на входе.
- Не учитывает связь соседних пикселей.
- Не учитывает устойчивости по отношению к сдвигам.

Свёртки

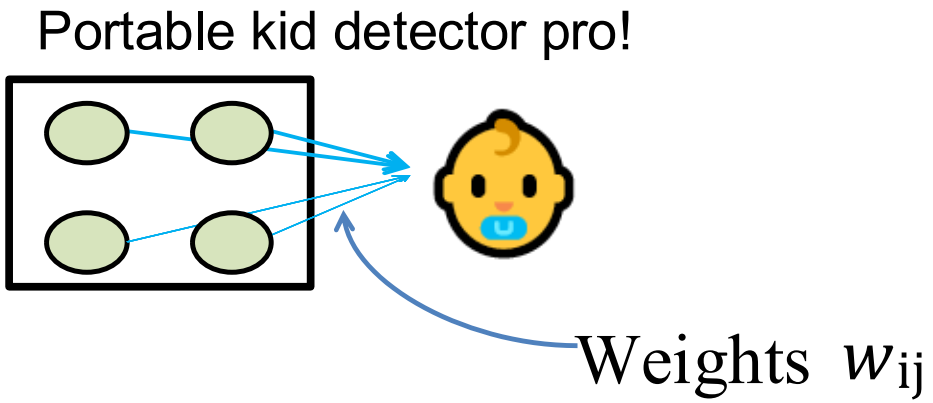
Лирическое отступление: современные тенденции

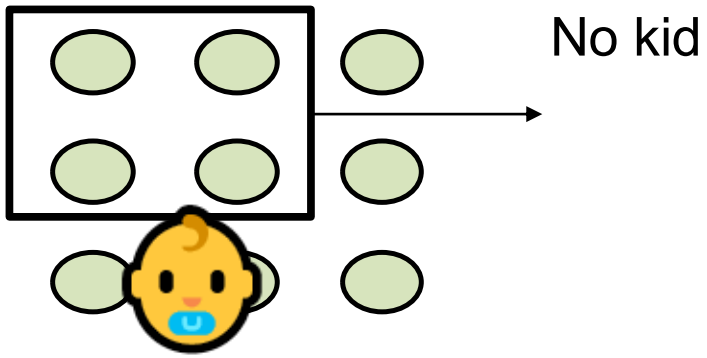
- Theoretically motivated model design.
- Theoretically motivated model learning.
- Theoretically motivated inference refinement.
- Theoretical model augmentation.
- Hybrid learning



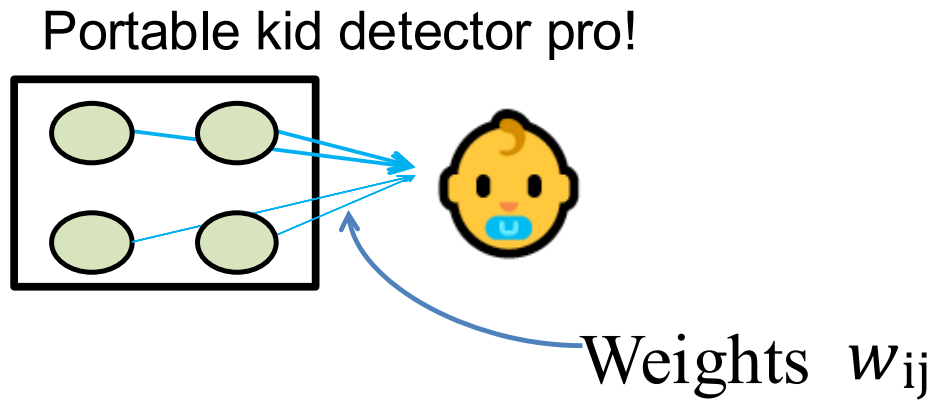
https://www.nature.com/articles/s42254-021-00314-5

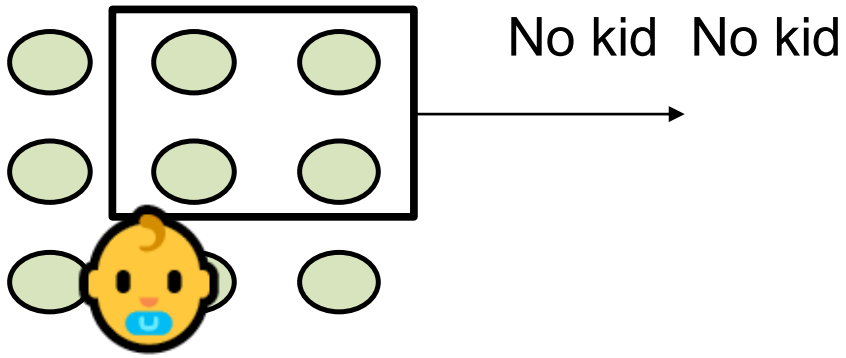
Детектор лица



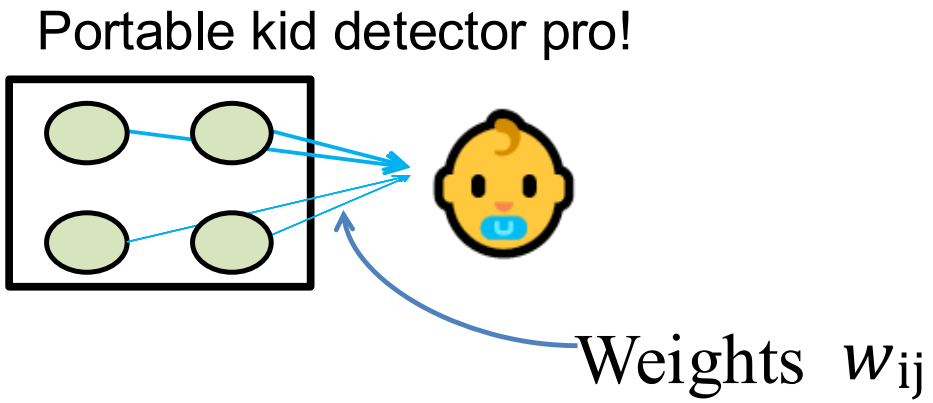


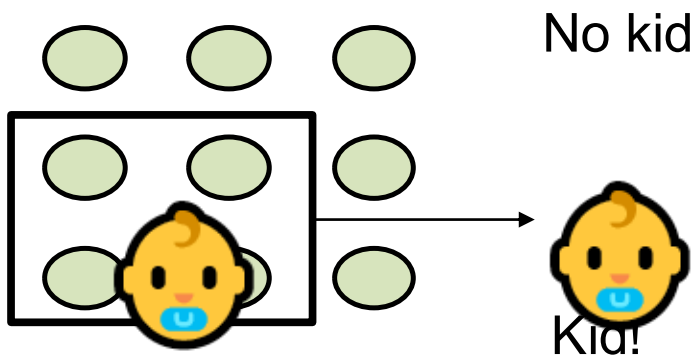
Детектор лица



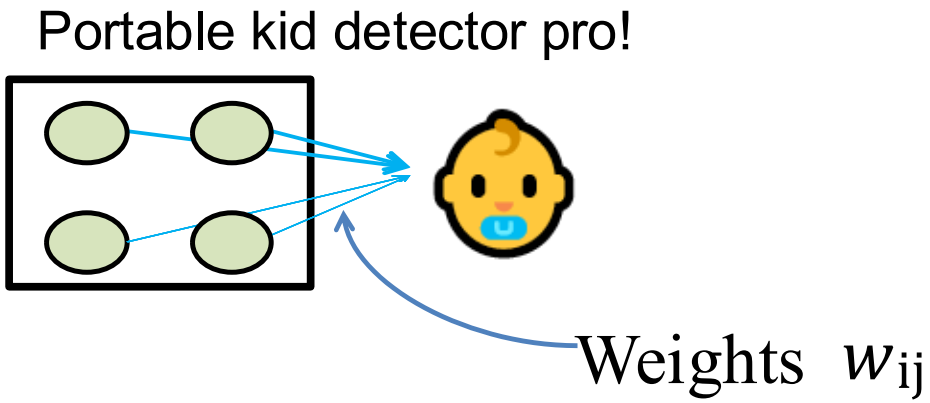


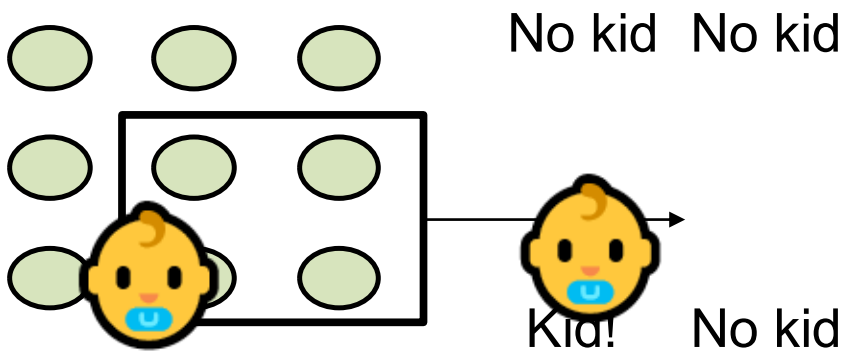
Детектор лица



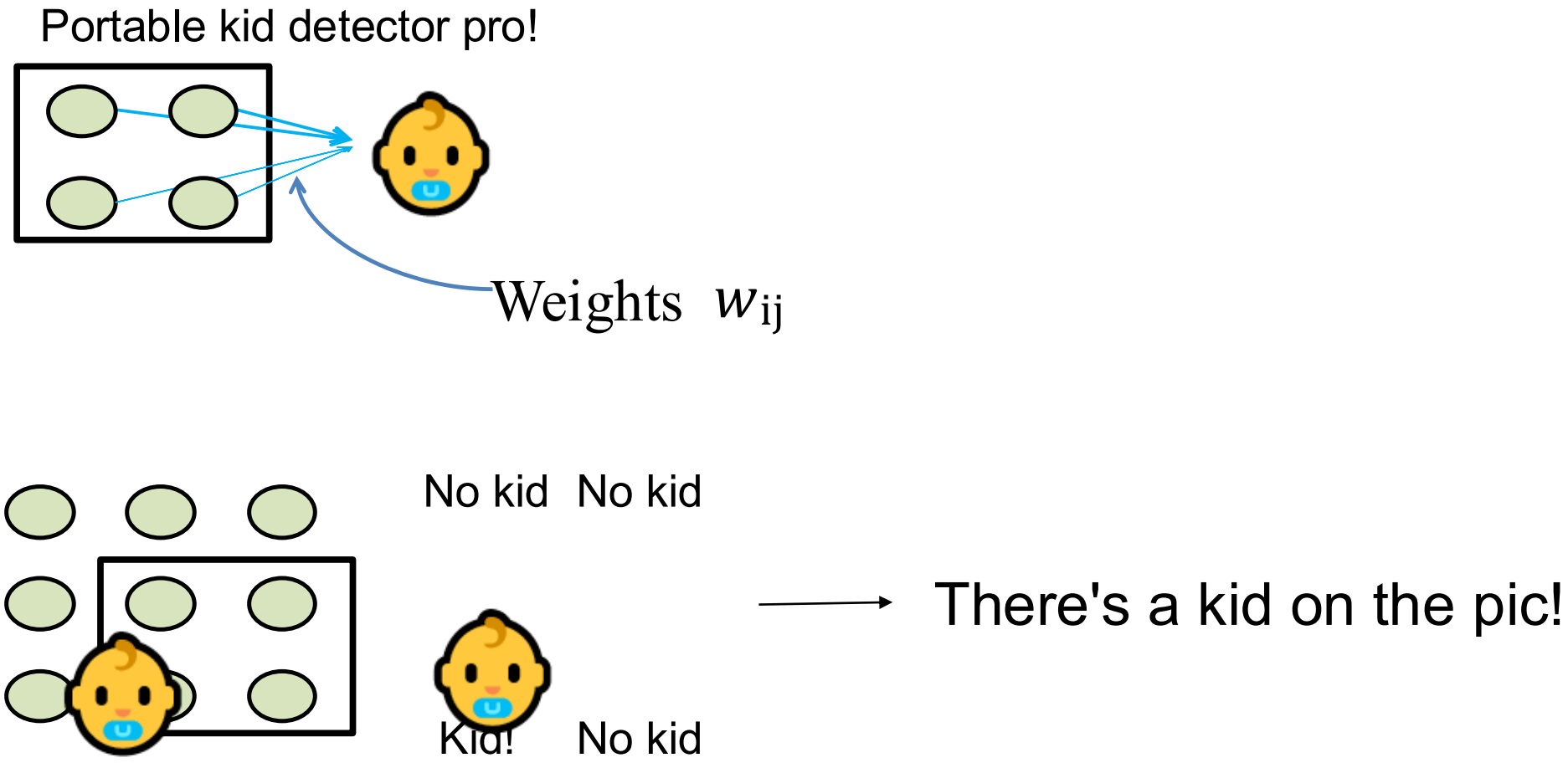


Детектор лица



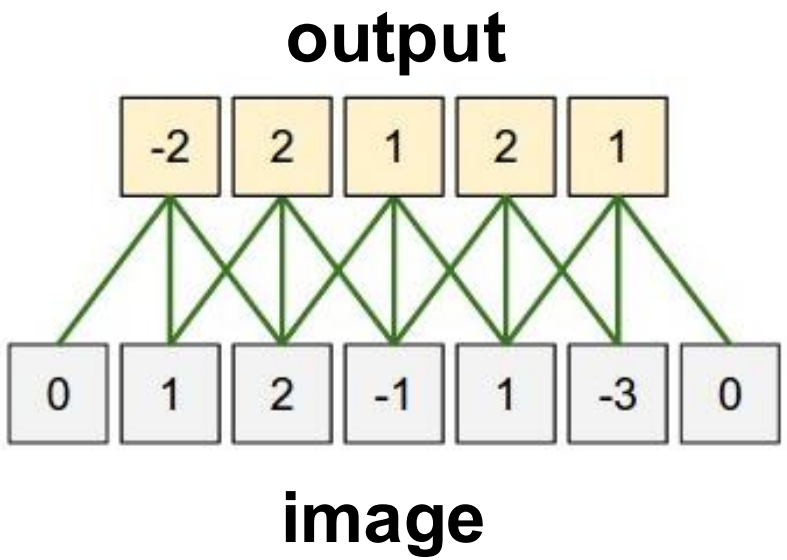


Детектор лица



Свёртки

- Используем одни веса для всех кусочков



- Автоматически даёт устойчивость к переносам и т.п.
- Радикально сокращает число весов
- Свёрточная структура – это радикальная форма регуляризации

Свёрточная структура – начало классического моделирования для машинного обучения

1D свёртки

Идем скользящим окном по последовательности, применяем одно и то же преобразование

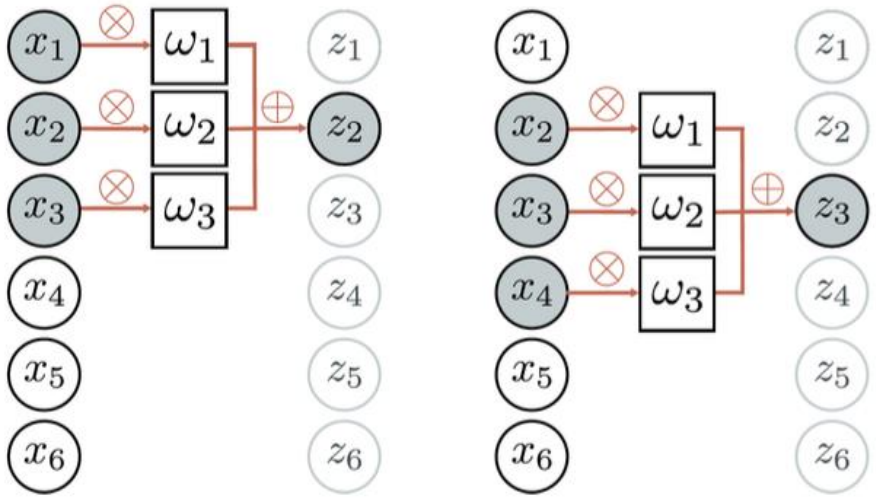


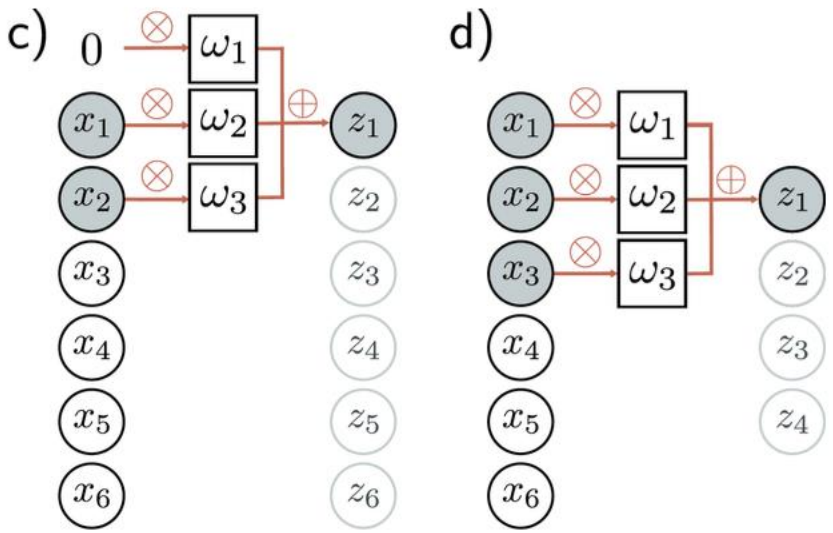
Image credit

$$z_i = w_1 x_{i-1} + w_2 x_i + w_3 x_{i+1}$$

(w_1, w_2, w_3) - ядро свертки (convolutional kernel / filter)

Отступ (padding)

Zero padding - добавляем с краю нули, чтобы на выходе получился вектор той же длины

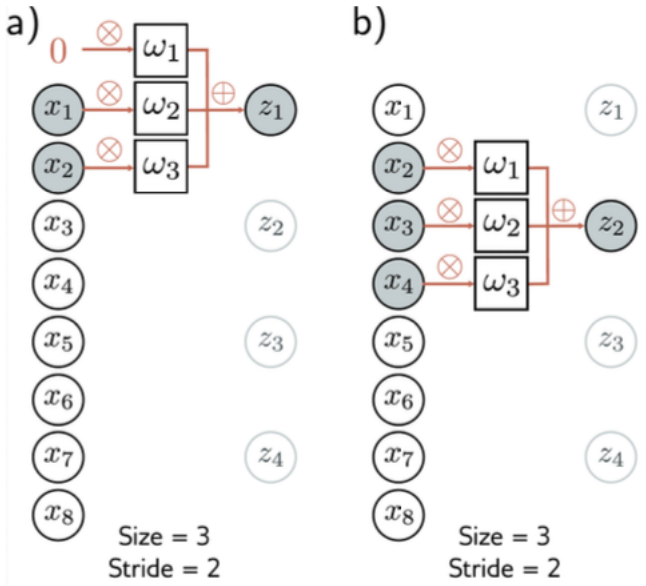


https://udlbook.github.io/udlbook/

Шаг перемещения (stride)

Stride (шаг) - можно смещаться не на один, а на несколько шагов перед следующим применением фильтра

- Получаем на выходе вектор меньшей длины
- Элементы выходного вектора менее скоррелированы



https://udlbook.github.io/udlbook/

1D-convolution vs fully connected

Сверточный слой - частный случай обычного полносвязного слоя

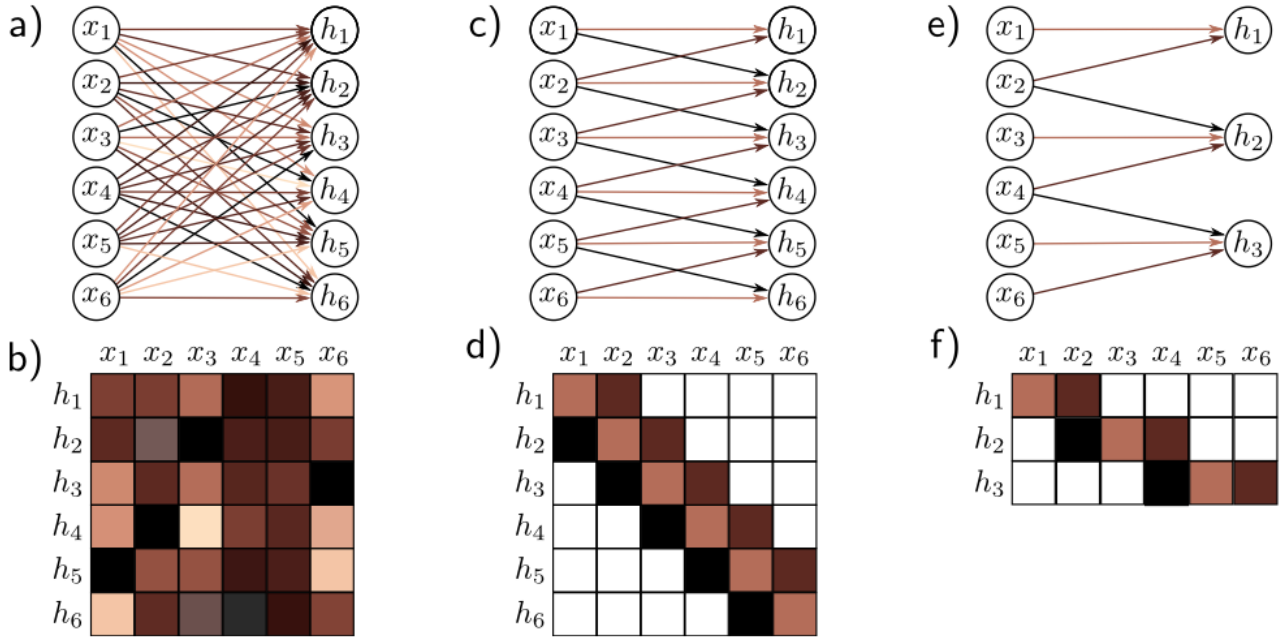


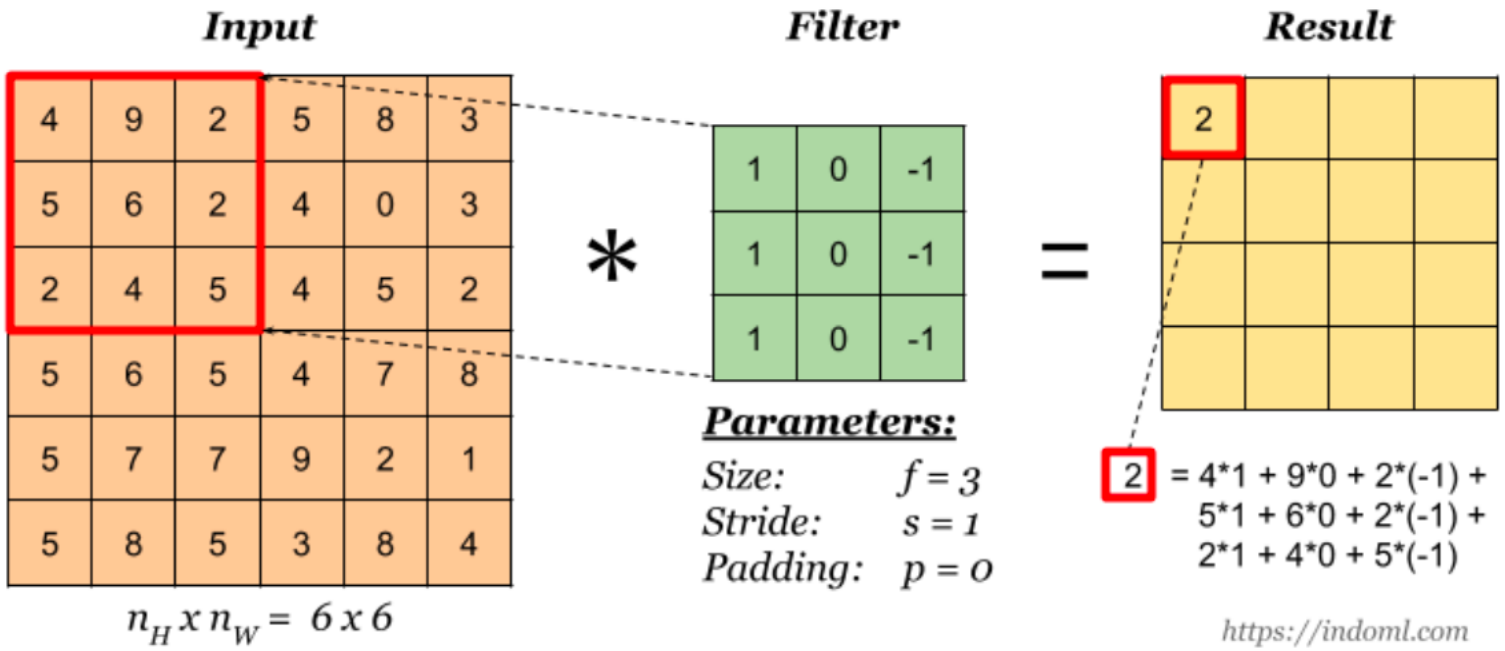
Image credit

https://udlbook.github.io/udlbook/

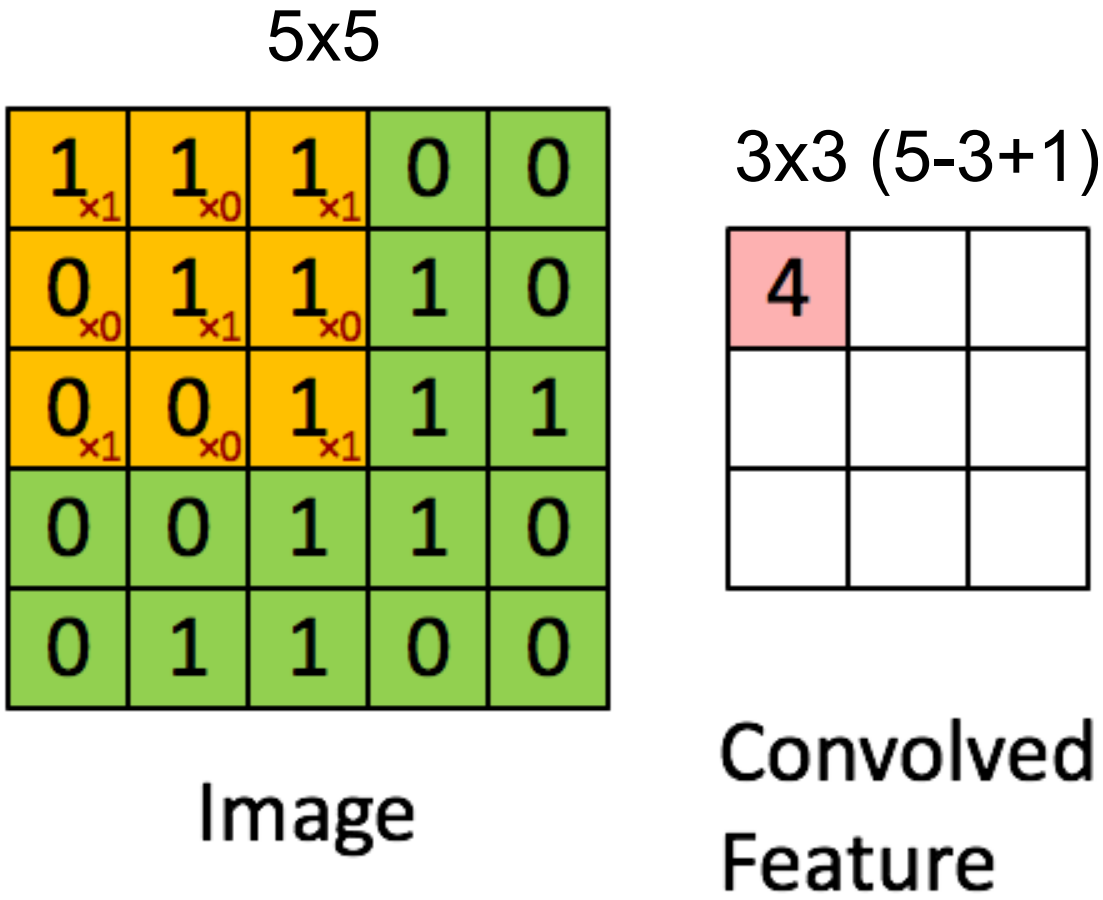
2D-свёртки

2D-convolution

$$h_{ij} = f\left(\beta + \sum_{m=1}^{3}\sum_{n=1}^{3} w_{mn} x_{i+m-2,j+n-2}\right)$$



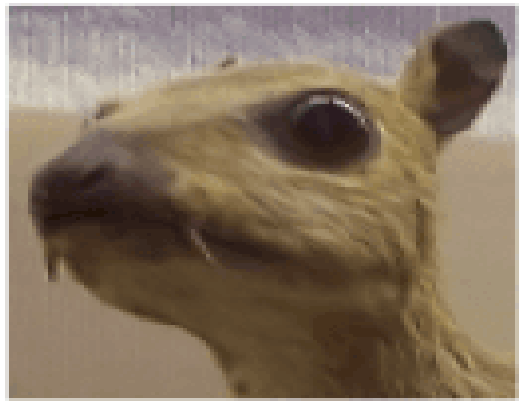
2D-свёртка



Интуиция: есть ли ребёнок в квадрате?

Пример свёртки

Input image



Convolution Kernel

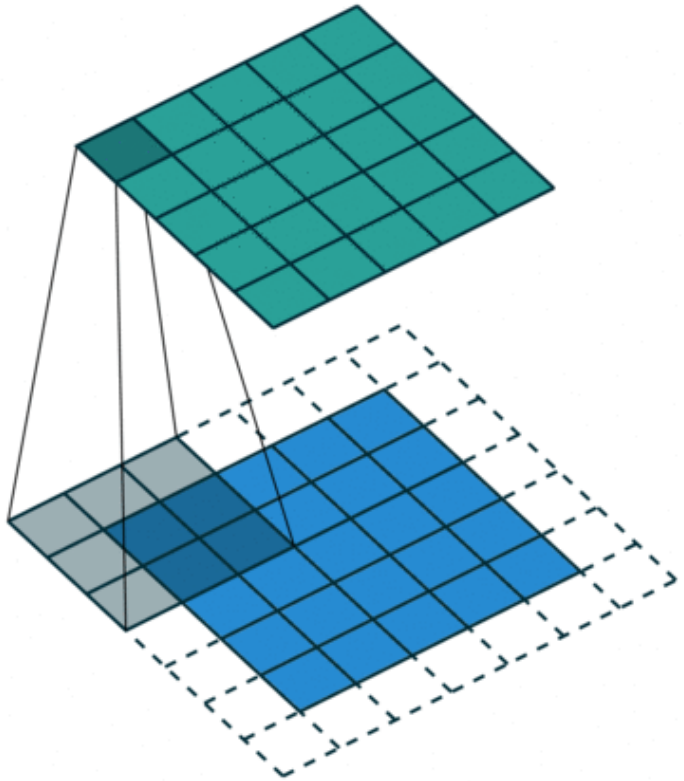
$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Feature map

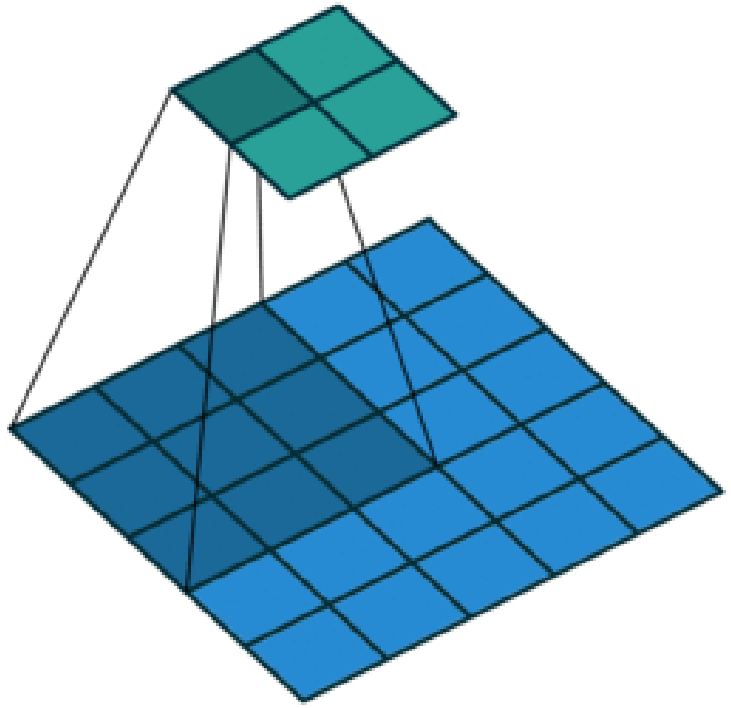
Интуиция: есть ли граница в квадрате?

Отступ, шаг

Padding (отступ) - добавляем с краю нули, чтобы на выходе получилась та же размерность, что и на входе

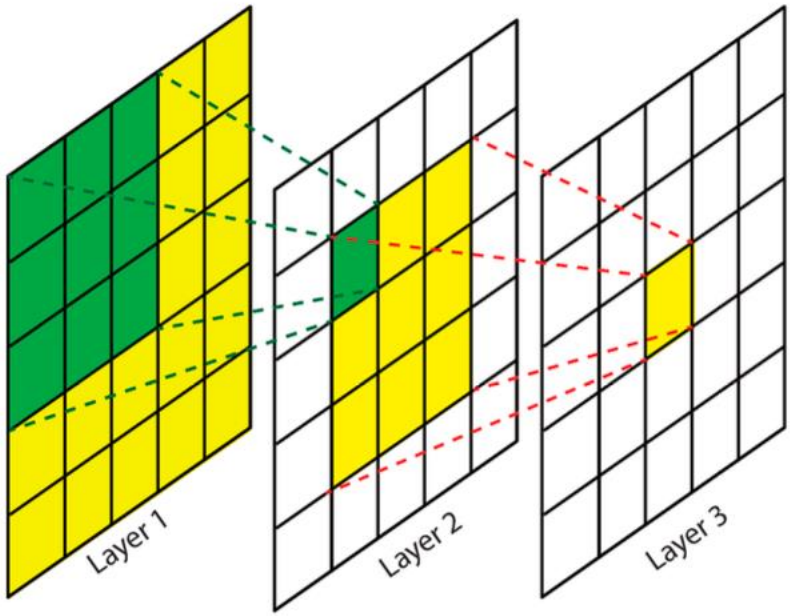


Stride (шаг) - количество шагов, на которые смещаемся перед следующим применением фильтра



Отступ, шаг

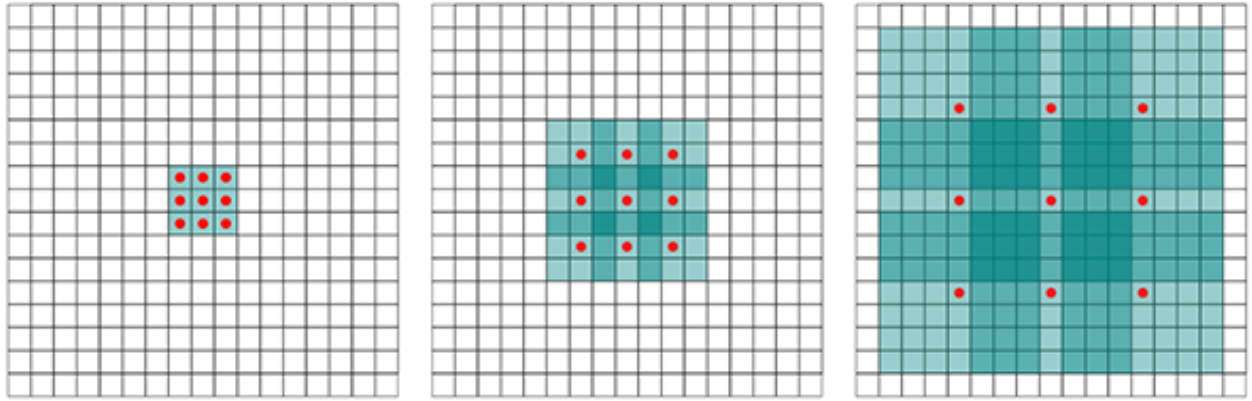
Область исходного изображения, которая влияет на конкретный признак



Расширенная свёртка (dilated convolution)

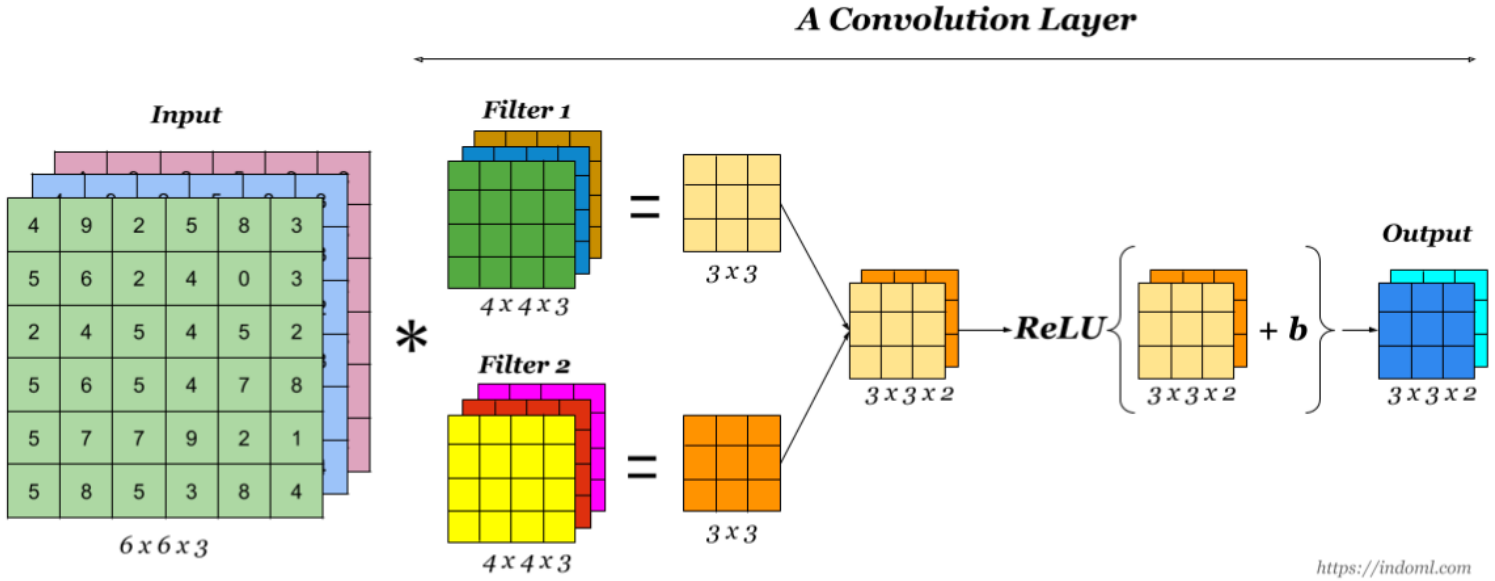
- Расширенная свертка — это базовая свертка, применяемая только к входному объему с определенными зазорами (см. ниже). Вы можете использовать расширенную свертку, когда:

Примеры
- работа с изображениями более высокого разрешения, но мелкие детали все ещё важны
- сеть с меньшим количеством параметров



Каналы

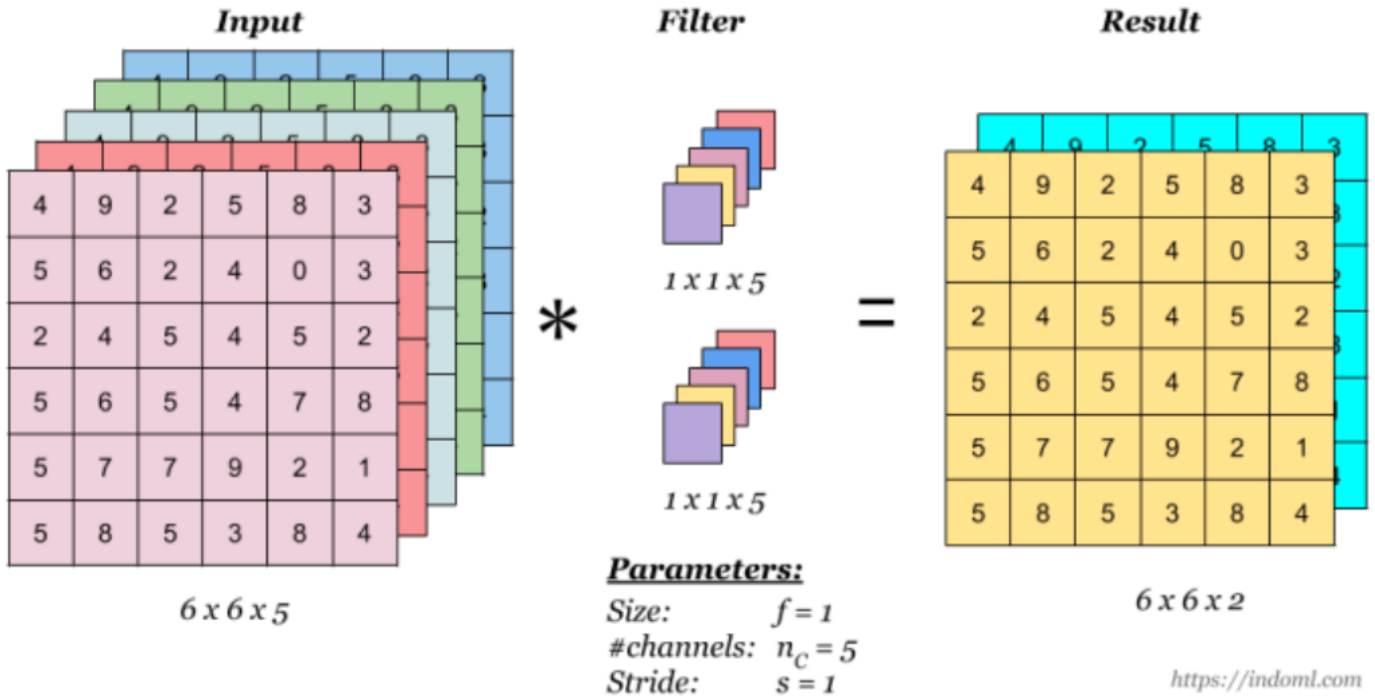
На самом деле на входе и на выходе много каналов (channels / feature maps)



1X1 свёртка

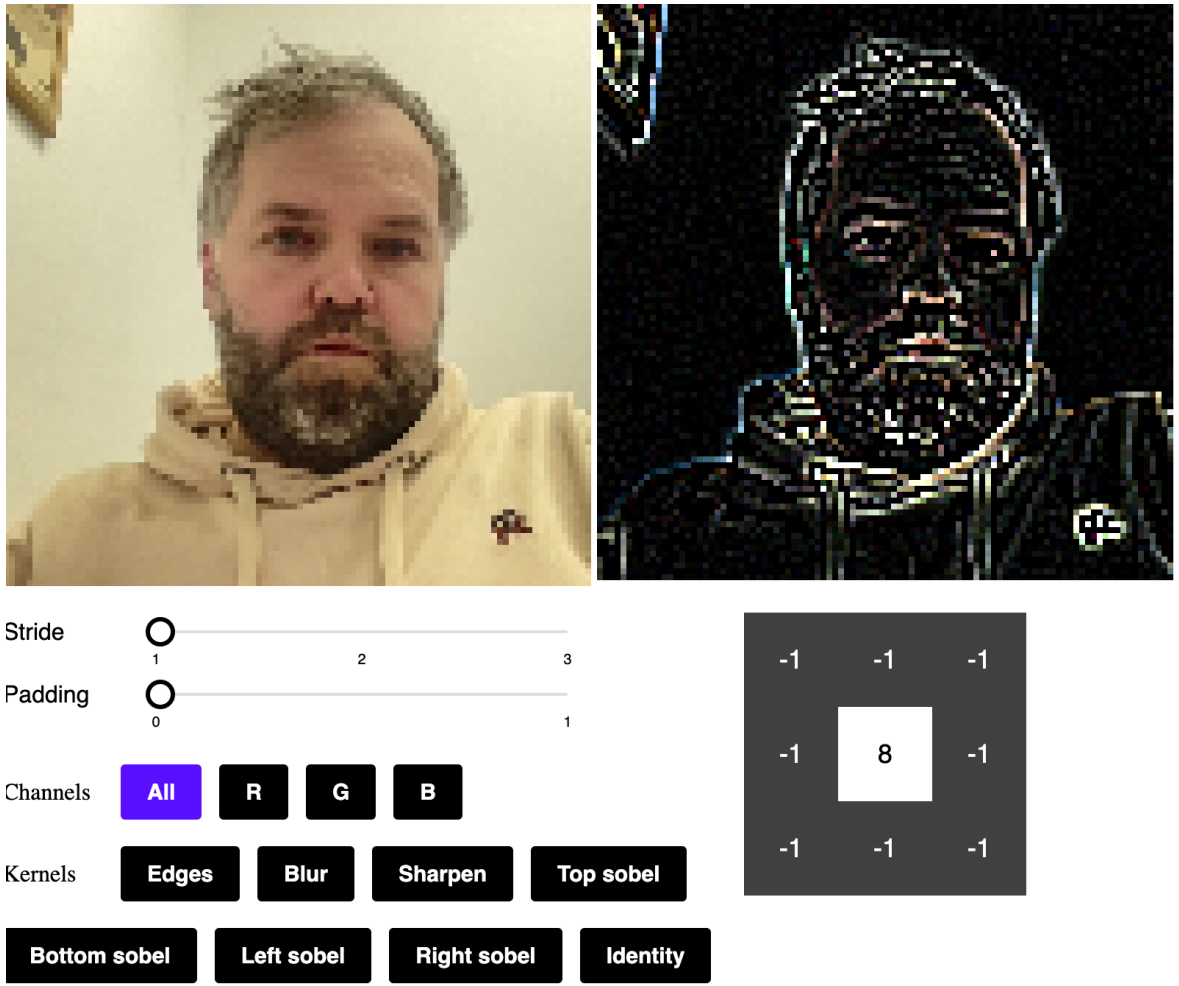
Для каждого пикселя применяем одно и то же преобразование «вдоль каналов»

- Преобразование признаков
- Изменение числа каналов



Convolution Demo

▶ <Demo>



Convolution Layer

Набор сверток (фильтров) + нелинейность

- Количество каналов (channels) на входе и выходе
- Kernel size - размеры ядра
- Stride (шаг) - большие значения понижают разрешение
- Padding (valid, same)
- Dilation - увеличивает receptive field

Размеры тензоров в свёрточных слоях

Размер выхода (высота/ширина):

$$H_{out} = \frac{H_{in} + 2 \times \text{padding}_h - \text{dilation}_h \times (\text{kernel\_size}_h - 1) - 1}{\text{stride}_h} + 1$$

$$W_{out} = \frac{W_{in} + 2 \times \text{padding}_w - \text{dilation}_w \times (\text{kernel\_size}_w - 1) - 1}{\text{stride}_w} + 1$$

Количество параметров:

$$\text{params} = (K_h \times K_w \times C_{in} + 1) \times C_{out}$$

где K_h, K_w — высота и ширина ядра, C_{in} — количество входных каналов, C_{out} — количество выходных каналов, $+1$ учитывает bias.

Количество операций (FLOPs):

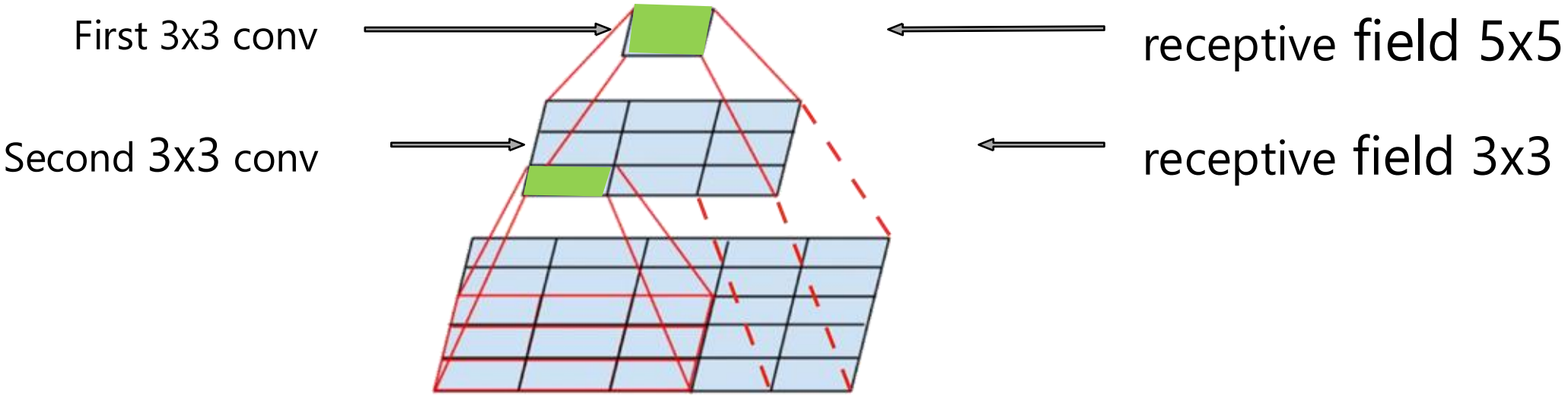
$$\text{FLOPs} = H_{out} \times W_{out} \times C_{out} \times (K_h \times K_w \times C_{in} + 1)$$

Обычно умножают на 2 (сложение + умножение).

Практические правила:

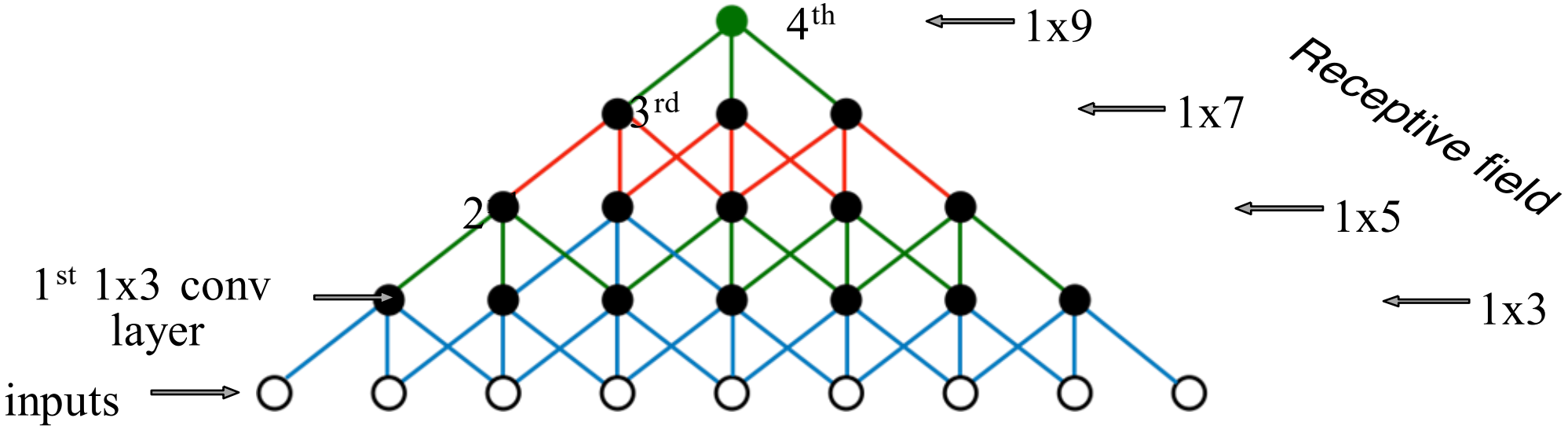
- Для сохранения размера при stride=1: padding $= \lfloor K/2 \rfloor$
- Для уменьшения вдвое при $K = 3$: stride $= 2$, padding $= 1$

Receptive Field



Большие объекты можно распознавать,
используя несколько маленьких свёрток

Receptive Field



Q:сколько 3x3 сверток нам нужно использовать, чтобы распознать ребенка размером 100x100 пикселей

A: около 50... нам нужно быстрее увеличивать RF быстрее!

Receptive Field – оценка параметров

Формула для receptive field (RF):

$$RF_l = RF_{l-1} + (K_l - 1) \times \prod_{i=1}^{l-1} S_i$$

где:

- RF_l — receptive field на слое l
- K_l — размер ядра на слое l
- S_i — stride на слое i

Пример для 3 слоёв 3x3, stride=1:

- Слой 1: $RF = 3$
- Слой 2: $RF = 3 + (3 - 1) \times 1 = 5$
- Слой 3: $RF = 5 + (3 - 1) \times 1 = 7$

Сравнение:

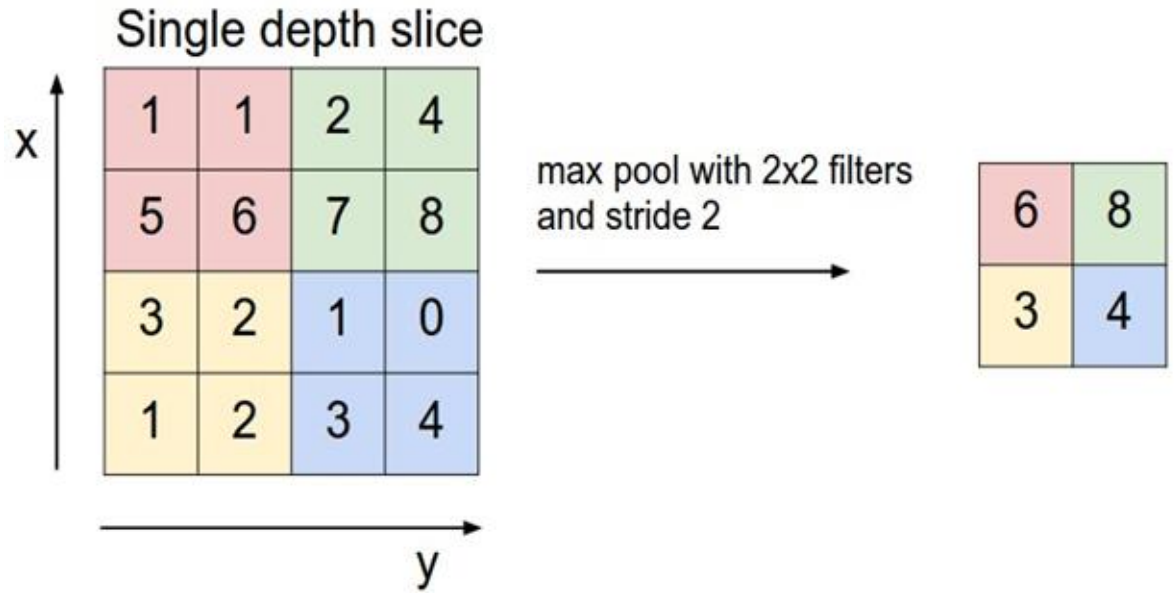
- 3 слоя 3x3: 27 параметров, $RF = 7$
- 1 слой 7x7: 49 параметров, $RF = 7$
- Вывод: несколько малых свёрток эффективнее!

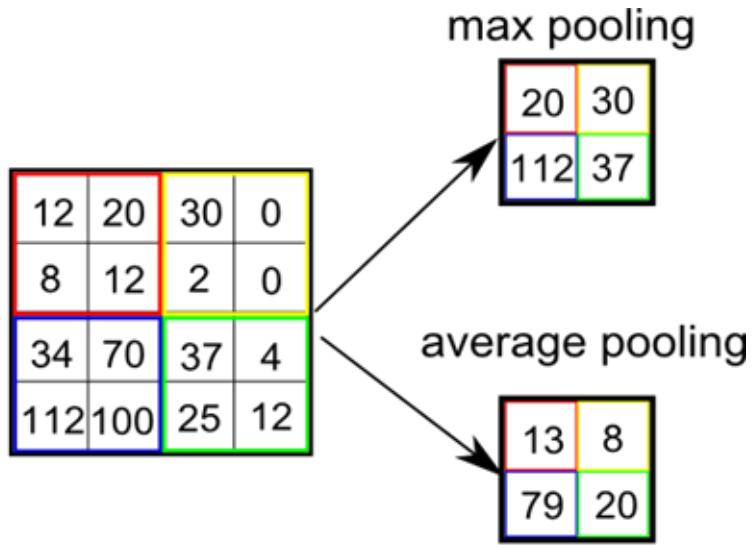
Эвристика выбора размера ядра

- Нечетное число на измерение

- Если входное изображение больше 128x128:
  - Используйте 5x5 или 7x7
  - и затем быстро уменьшите пространственные измерения — затем начните работать с ядрами 3×3
  - В противном случае рассмотрите возможность использования фильтров 1×1 и 3×3.

Пулинг



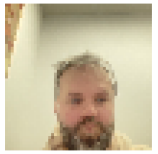


Pooling (субдискретизация)

- Агрегация (max, mean) соседних признаков
- Делается независимо по каналам
- Уменьшает размер представления
- Обеспечивает инвариантность к небольшим сдвигам

Pooling Demo

▶ <Demo>



Pooling size

1 2 3 4

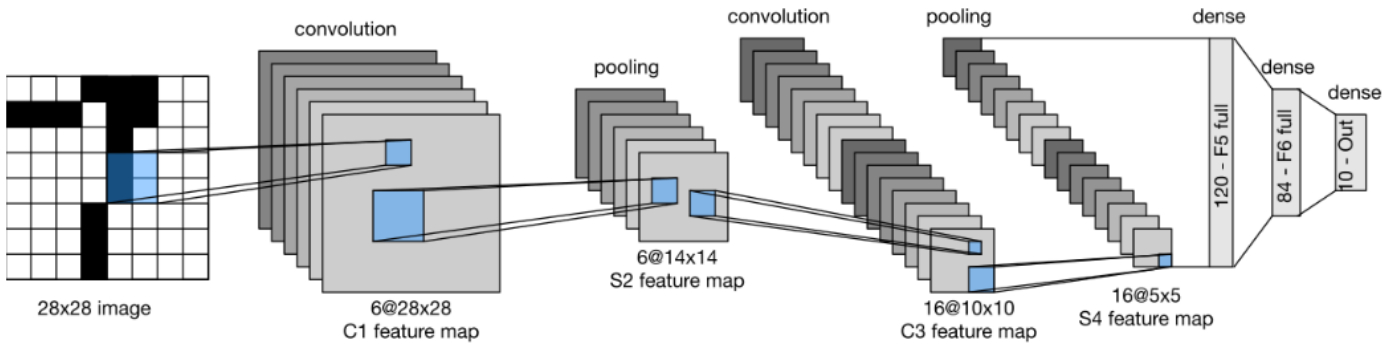
Function Max Average

Channels All R G B

LeNet

Стандартная архитектура для классификации:

- Чередование сверточных и pooling слоев
- Постепенное понижение размера
- Полносвязные слои на выходе

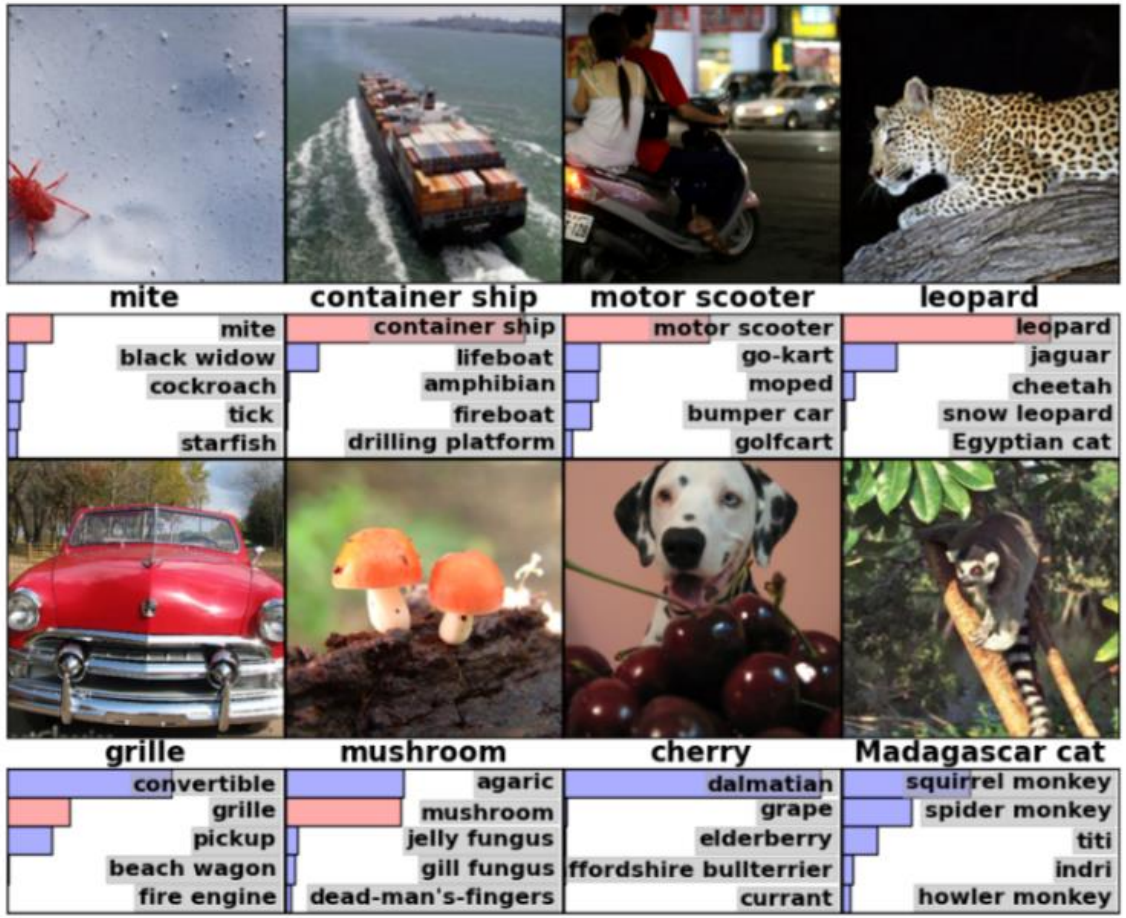


LeCun et al., 1998 (Image credit)

https://arxiv.org/pdf/1409.4842.pdf

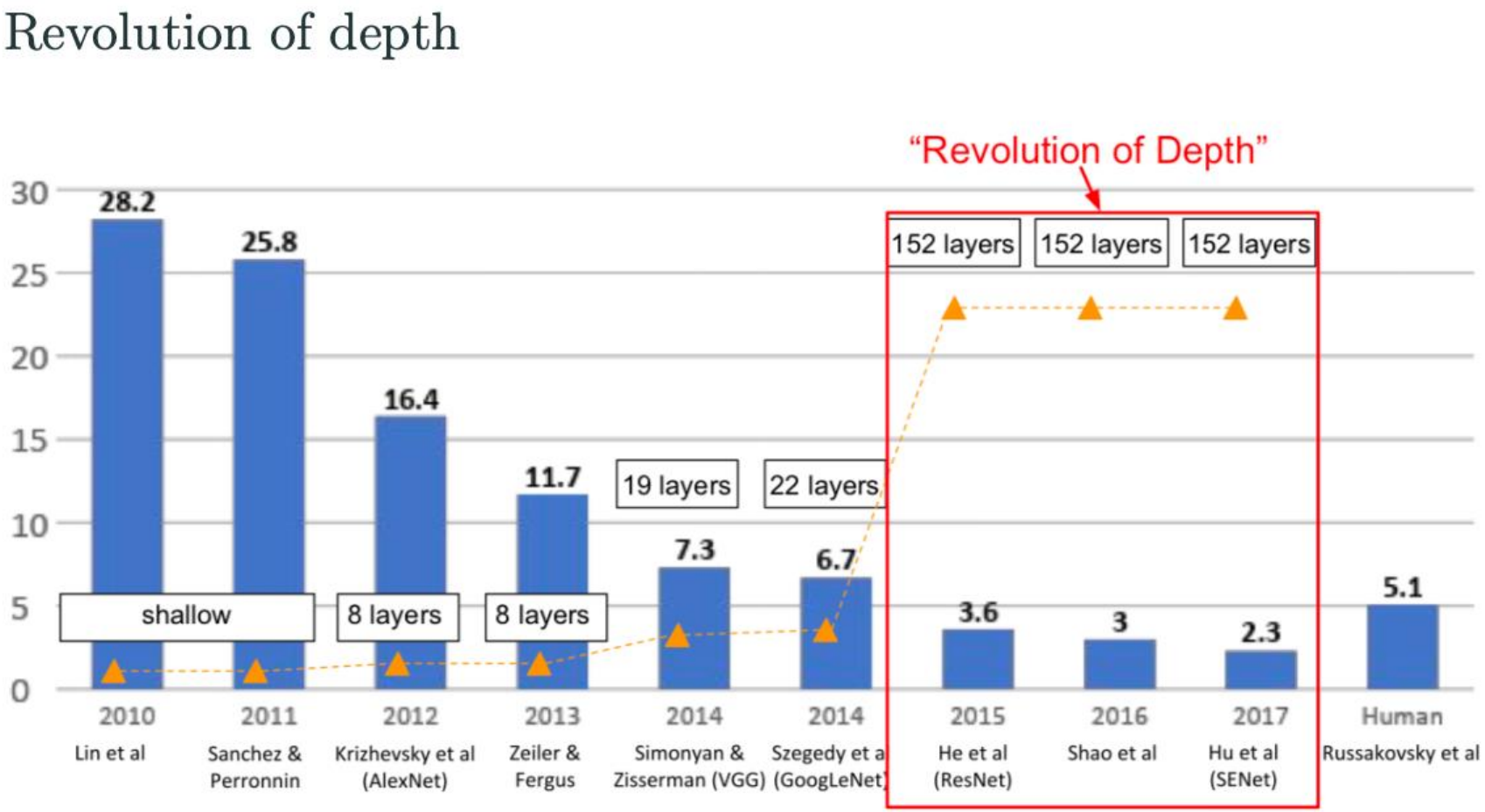
ImageNet

ImageNet Large Scale Visual Recognition Challenge



- Классификация изображений
- 1M объектов
- 1000 классов
- Данные из интернета
- Аннотация при помощи Amazon MTurk

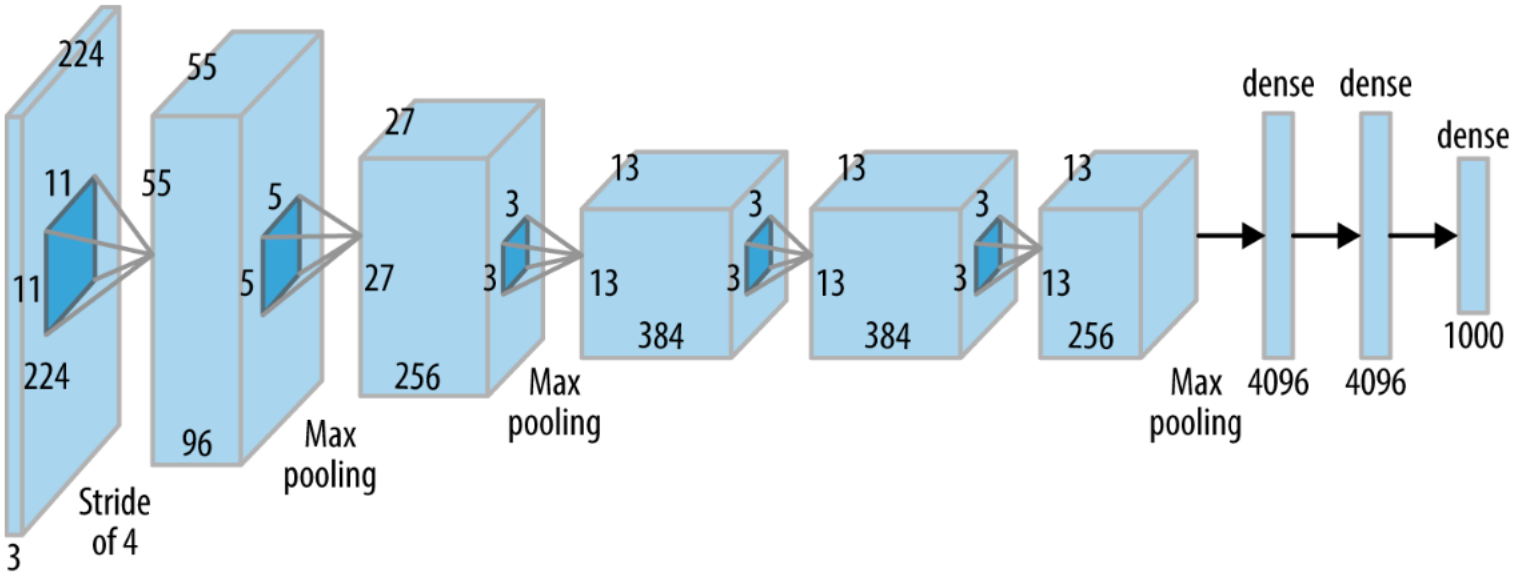
ImageNet



https://iphysresearch.github.io/blog/post/dl_notes/cs231n/cs231n_9/

Свёрточные сети

AlexNet



Krizhevsky et al. 2012 (Image credit)

- ReLU
- 8 слоев
- 60M параметров
- Много аугментаций
- 1 неделя обучения на 2 GPU

VCG

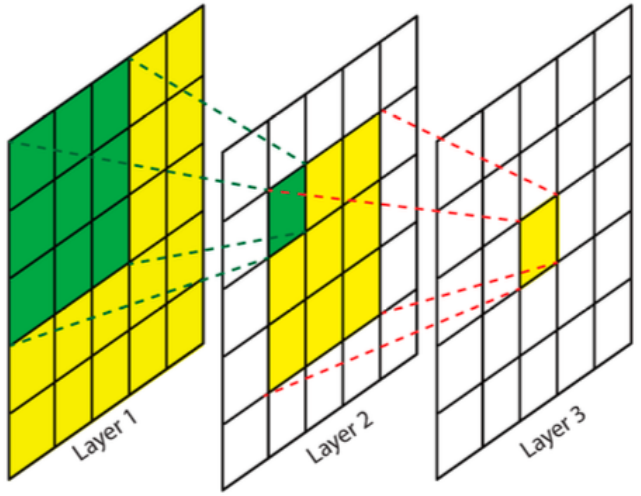


Image credit

Идея - давайте представлять большие свёртки как комбинации свёрток 3×3.

- Меньше параметров и быстрее, сеть глубже
- 2 свертки 3x3 имеют receptive field как одна 5x5
- 3 свертки 3x3 имеют receptive field как одна 7x7

VCG

Simonyan et al. Very Deep Convolutional Networks for Large-Scale Image Recognition (2014)

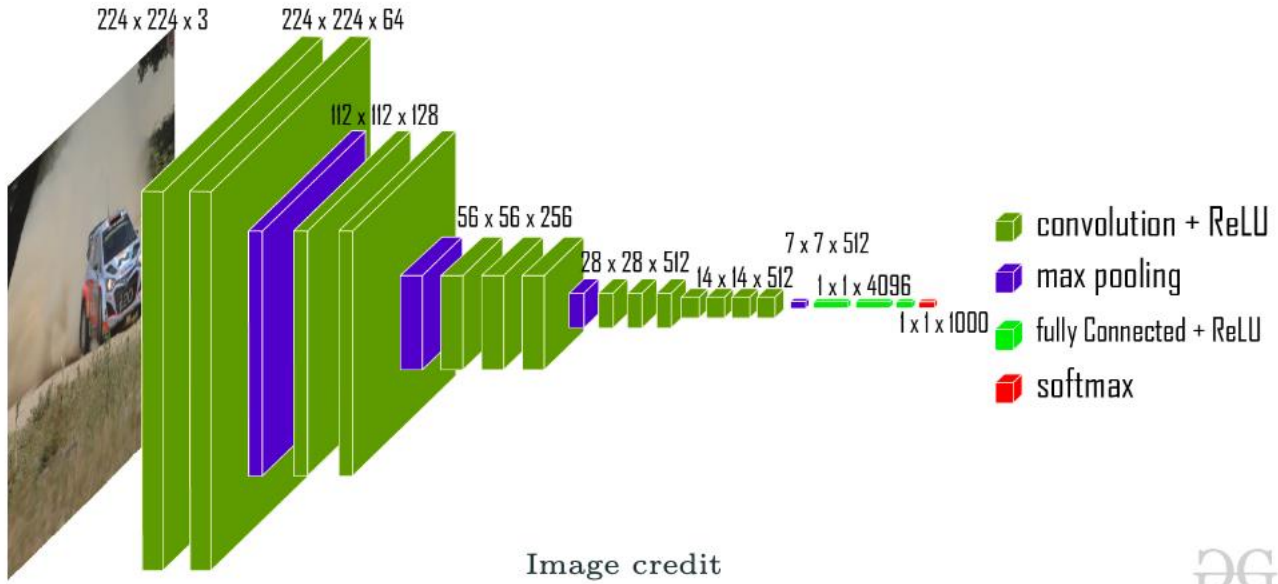
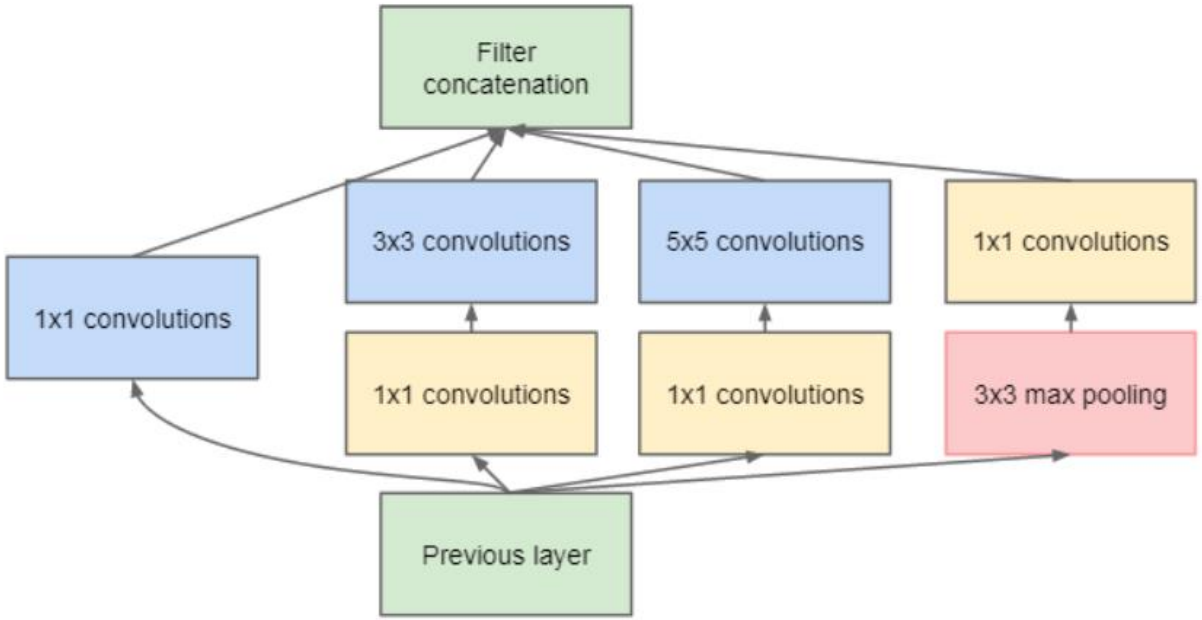


Image credit

- Свертки 3x3
- 16 / 19 слоев
- 140M параметров

Inception block

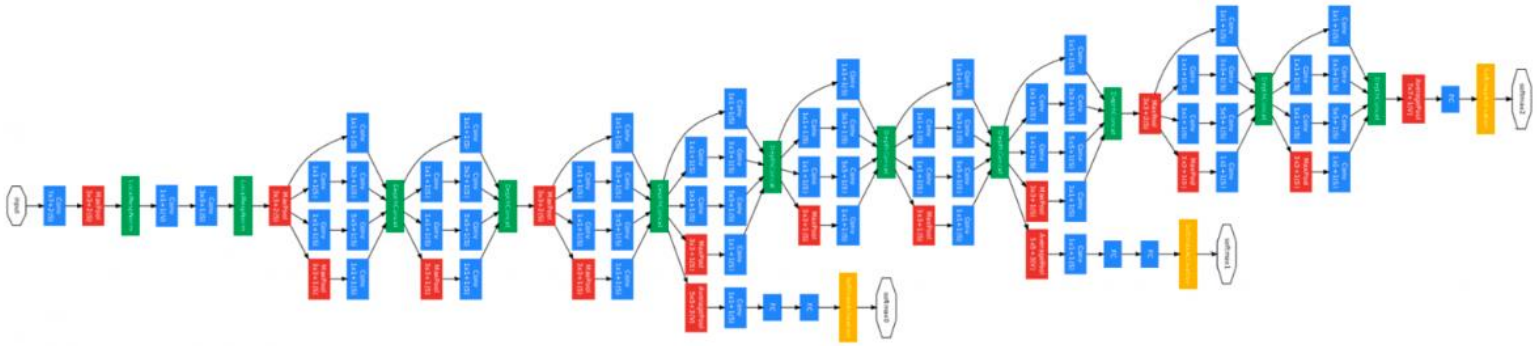
Szegedy et al. Going Deeper with Convolutions (2014)



1×1 свертки - понижение размерности

Во одной из версий свертки $n \times n$ заменили на комбинации свёрток $n \times 1$ и $1 \times n$.

GoogLeNet

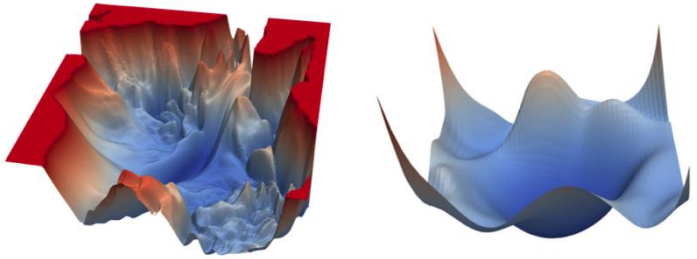


- 22 слоя
- Inception блоки
- Всего 5M параметров
- GlobalAvgPool в конце
- Вспомогательные выходы для обучения

Residual connections

Как обучить еще более глубокую сеть? Просто добавление слоев не работает

Loss landscape without and with skip connections

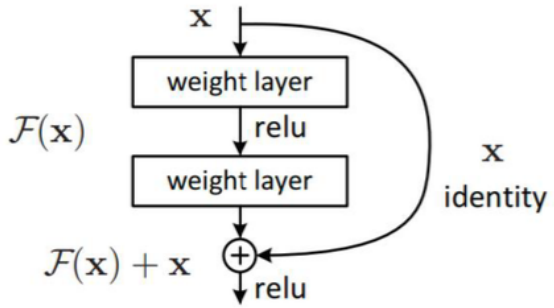


Обучаем разность между очередным уровнем и предыдущим - residual learning

$$y = F(x) + x$$

Тогда градиенты беспрепятственно проходят через такой блок и не затухают

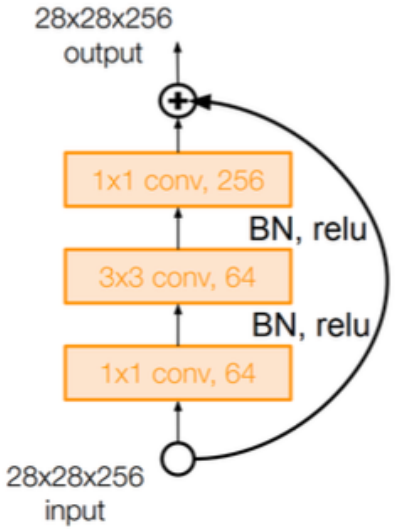
$$\frac{\partial y}{\partial x} = \frac{\partial F(x)}{\partial x} + 1$$



Residual block

ResNet

He et al. Deep Residual Learning for Image Recognition (2015)

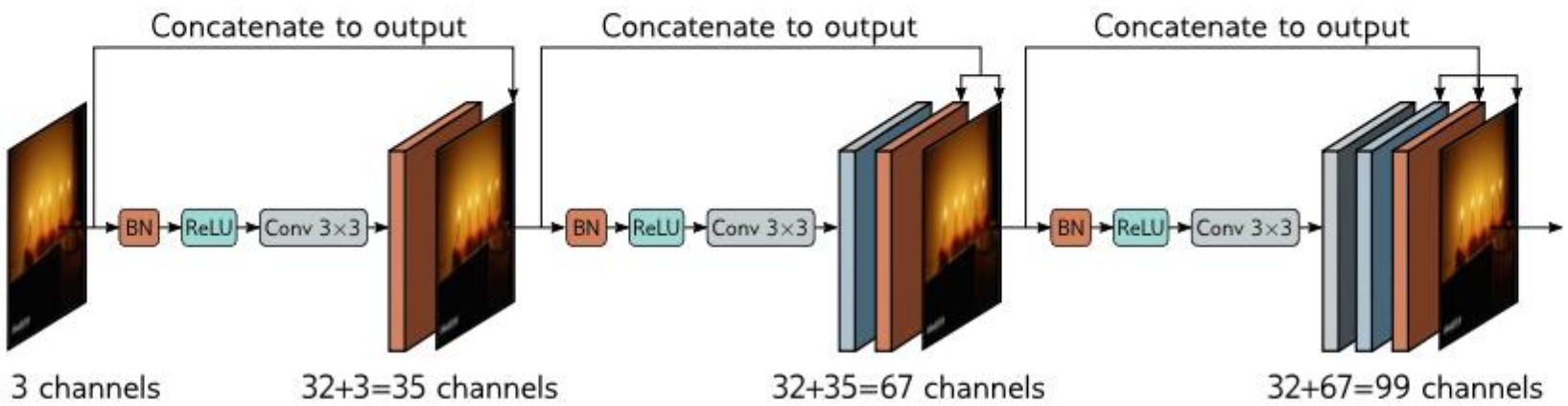


- 152 слоя
- Batch normalization
- Есть версии с 18, 34, 50, 101 слоями

DenseNet

Huang et al. Densely Connected Convolutional Networks (2016)

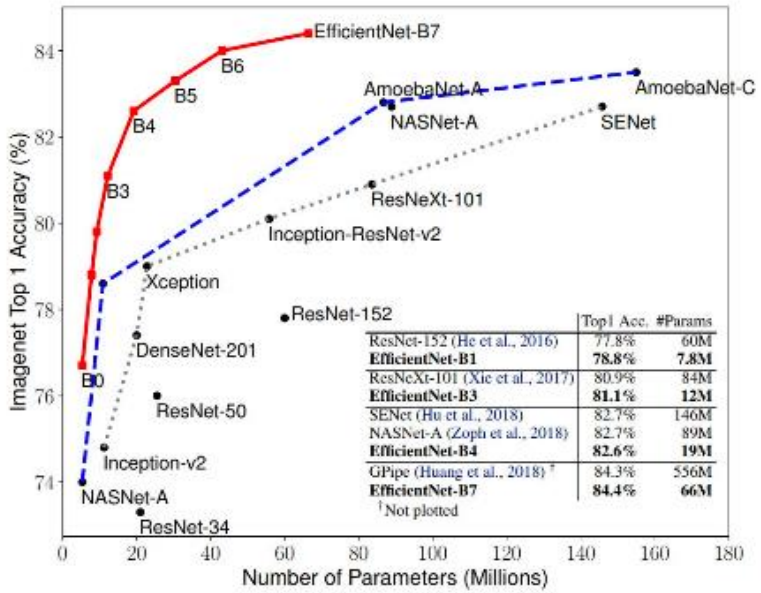
Конкатенируем, а не складываем выходы предыдущих слоев



EfficientNet

EfficientNet: Improving Accuracy and Efficiency through AutoML and Model Scaling

- Класс моделей с равномерным масштабированием глубины/ширины/разрешения
- Поиск архитектуры - Neural Architecture Search (NAS)



	Top1 Acc.	#Params
ResNet-152 (He et al., 2016)	77.8%	60M
EfficientNet-B1	**78.8%**	**7.8M**
ResNeXt-101 (Xie et al., 2017)	80.9%	84M
EfficientNet-B3	**81.1%**	**12M**
SENet (Hu et al., 2018)	82.7%	146M
NASNet-A (Zoph et al., 2018)	82.7%	89M
EfficientNet-B4	**82.6%**	**19M**
GPipe (Huang et al., 2018) †	84.3%	556M
EfficientNet-B7	**84.4%**	**66M**

† Not plotted

Сравнение архитектур

Архитектура	Год	Параметры	Топ-1 на ImageNet
LeNet-5	1998	60K	-
AlexNet	2012	60M	63.3%
VGG-16	2014	138M	74.4%
ResNet-50	2015	25.6M	79.3%
Inception v3	2015	23.8M	78.8%
DenseNet-121	2016	8.0M	75.0%
EfficientNet-B0	2019	5.3M	77.1%

Тренды:

- Уменьшение параметров при росте точности
- Увеличение глубины сети
- Использование residual/dense connections
- AutoML для поиска архитектур

Вывод

- Распознавание изображений — важный компонент глубокого обучения.
- Изображения — это многомерные объекты, для эффективной обработки которых требуются мощные методы:
  - свёртка;
  - макс-пулинг.
- Большую помощь оказывают предварительно обученные модели и библиотеки.