

# **Лекция 5. Сверточные нейронные сети.**

Глубинное обучение

---

Антон Кленицкий

19 февраля занятия не будет!

# Recap

---

# Learning rate

Важно правильно настроить learning rate

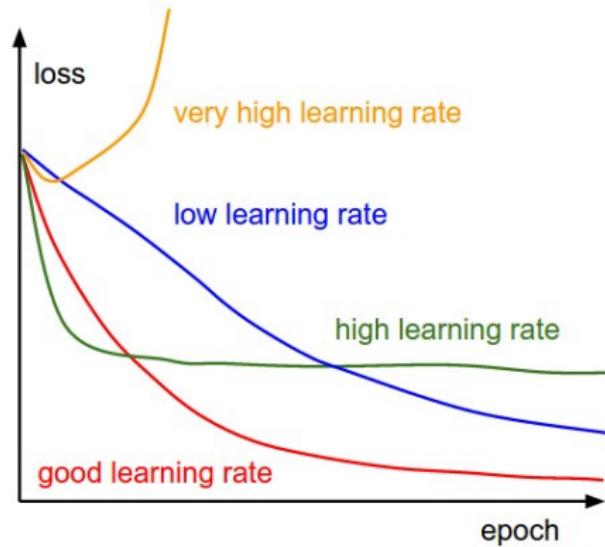


Image credit

# Learning rate schedulers

Изменение learning rate со временем по расписанию

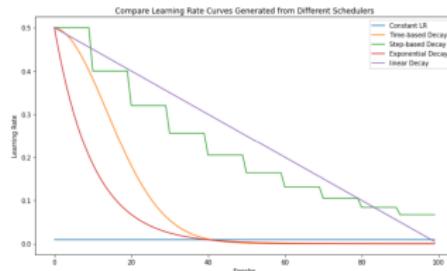


Image credit

- Decay (exponential, linear, step, ..)
- ReduceLROnPlateau
- Cyclic learning rate
- Cosine annealing with warm restarts

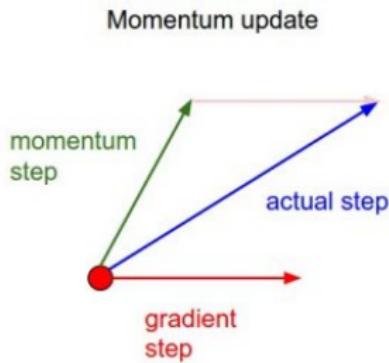
# Momentum

Сохраняем часть «скорости» с предыдущего шага

Momentum

$$m_t = \beta m_{t-1} + (1 - \beta) \nabla_{\theta} L(\theta_t)$$

$$\theta_{t+1} = \theta_t - \alpha m_t$$



Nesterov momentum

$$m_t = \beta m_{t-1} + (1 - \beta) \nabla_{\theta} L(\theta_t - \beta m_{t-1})$$

$$\theta_{t+1} = \theta_t - \alpha m_t$$

Nesterov momentum update

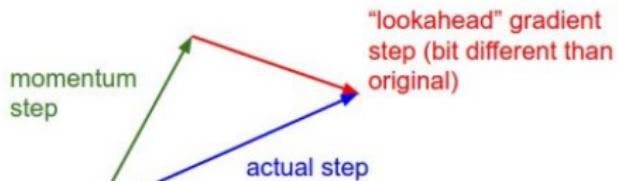


Image credit

## Идея адаптивных методов градиентного спуска

- Использовать разные learning rate для разных параметров
- Быстрее двигаться по тем параметрам, которые не сильно меняются, и медленнее по быстро меняющимся параметрам

$$\begin{aligned}m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \\\theta_{t+1} &= \theta_t - \frac{\alpha}{\sqrt{v_t + \epsilon}} m_t\end{aligned}$$

# Many faces of regularization

---

- L1/L2 regularization (weight decay)
- Early stopping
- Dropout
- Data augmentation
- Noise injection
- Label smoothing
- Batch / Layer Normalization
- SGD - Implicit regularization

# Сверточные нейронные сети

---

# Нейросети для изображений

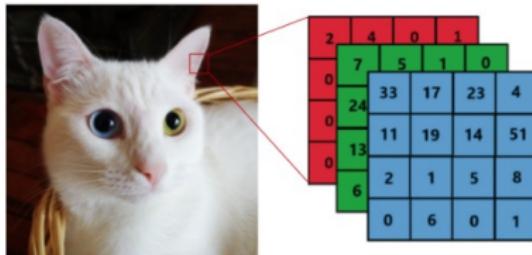


Image credit

## Проблемы с обычными нейросетями (MLP)

- Слишком много параметров - переобучение  
 $224 \times 224 \times 3$  изображение - 150к измерений на входе
- Соседние пиксели связаны друг с другом
- Хочется устойчивости по отношению к преобразованиям (сдвиги и т.п.)

# Нейросети для изображений

---

Сверточные сети (Convolutional neural networks, CNN) - архитектура для данных с пространственной структурой

Основная идея: применяем одну и ту же операцию (одни и те же веса) к разным частям изображения

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

# 1D Convolutions

---

# 1D Convolution

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

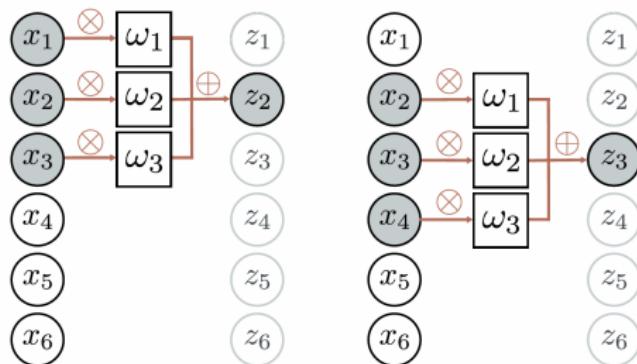


Image credit

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

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

# Padding

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

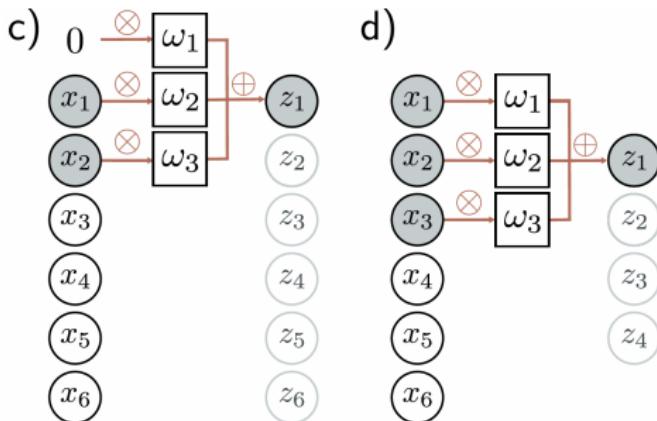
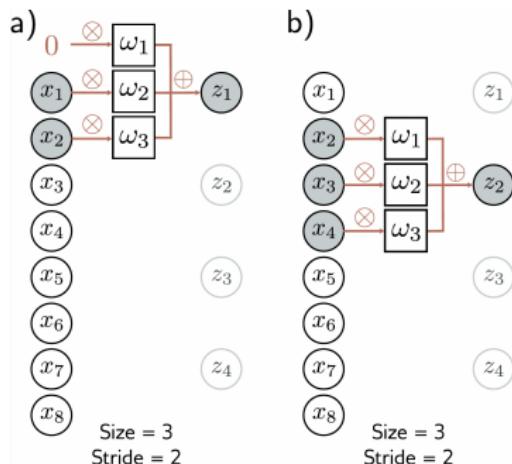


Image credit

# Stride

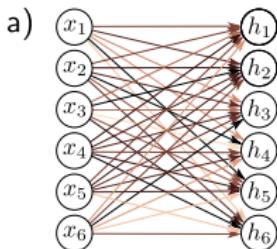
Stride (шаг) - можно смещаться не на один, а на несколько шагов перед следующим применением фильтра

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



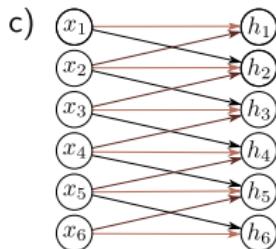
# 1D Convolution vs Fully-connected layer

Сверточный слой - частный случай обычного полносвязанного слоя



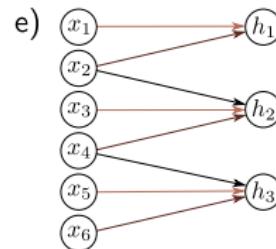
b)

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
$h_1$	dark brown	light brown	dark brown	dark brown	dark brown	light brown
$h_2$	dark brown	grey	dark brown	dark brown	dark brown	dark brown
$h_3$	light brown	dark brown	light brown	dark brown	dark brown	dark brown
$h_4$	light brown	black	light brown	dark brown	dark brown	light brown
$h_5$	black	dark brown	light brown	light brown	dark brown	dark brown
$h_6$	light brown	dark brown	dark brown	dark brown	dark brown	dark brown



d)

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
$h_1$	light brown	dark brown	light brown	white	white	white
$h_2$	dark brown	black	light brown	dark brown	white	white
$h_3$	white	black	light brown	dark brown	white	white
$h_4$	white	black	white	light brown	dark brown	white
$h_5$	white	white	white	white	light brown	dark brown
$h_6$	white	white	white	white	white	light brown



f)

	$x_1$	$x_2$	$x_3$	$x_4$	$x_5$	$x_6$
$h_1$	light brown	dark brown	light brown	white	white	white
$h_2$	white	black	light brown	dark brown	white	white
$h_3$	white	white	white	white	light brown	dark brown

Image credit

- Parameter sharing

# 2D Convolutions

---

# 2D Convolution

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

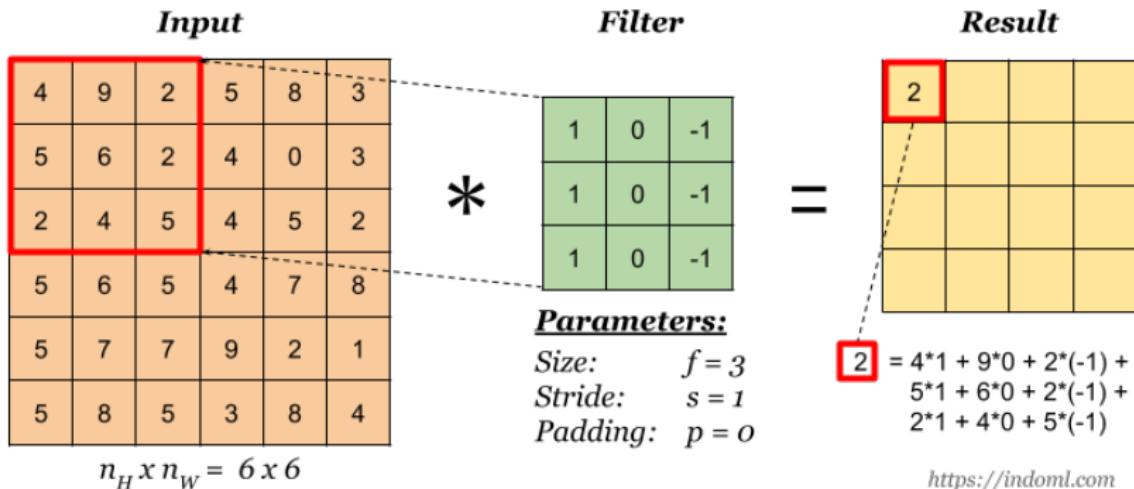
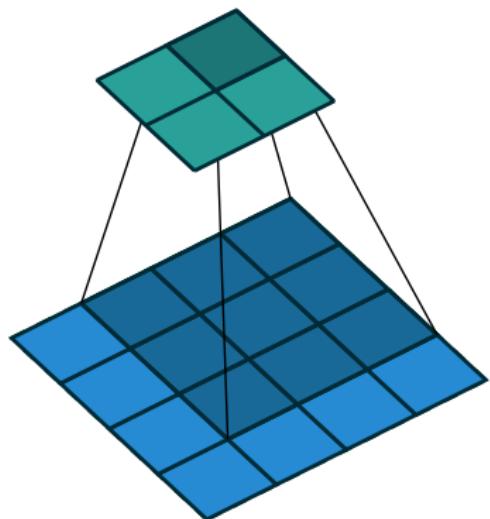
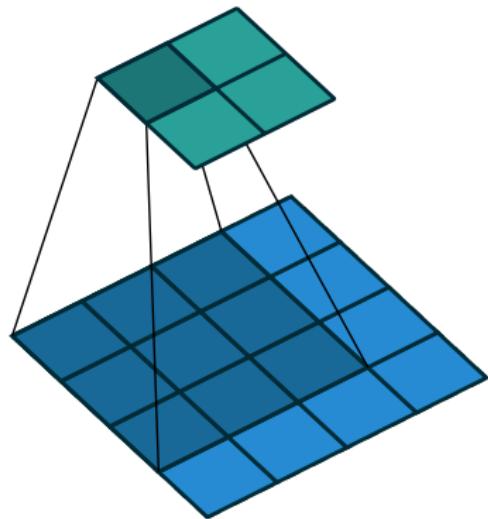


Image credit

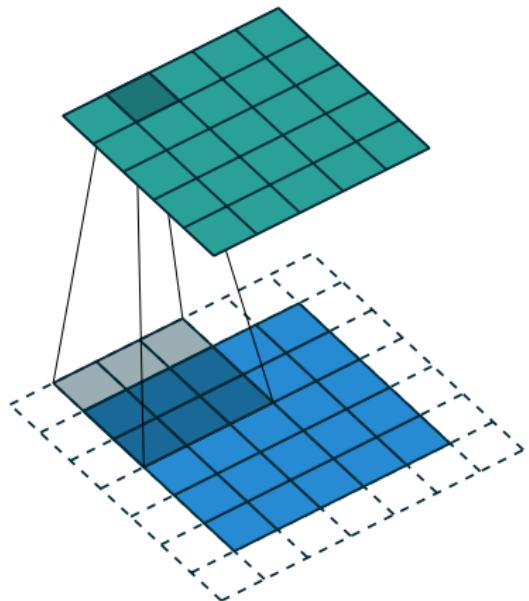
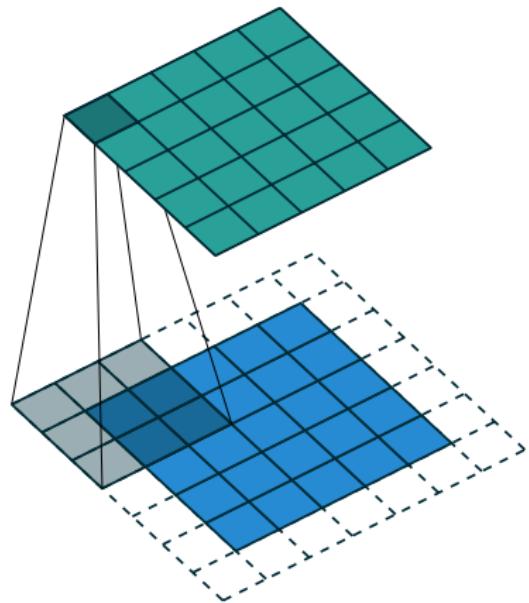
## 2D Convolution

Стандартная свертка - сдвигаемся на один шаг перед каждым следующим применением фильтра



# Padding

Zero padding - добавляем с краю нули, чтобы на выходе получилась та же размерность, что и на входе



# Stride

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

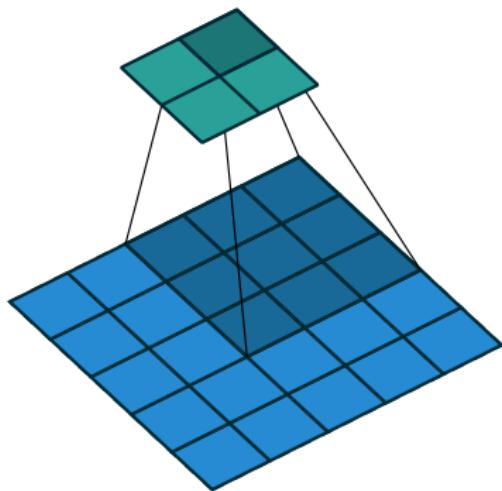
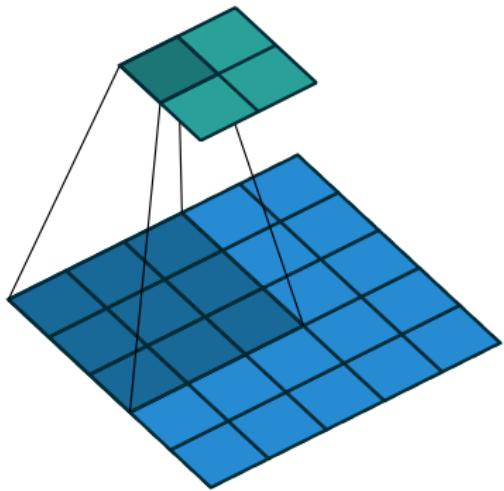


Image credit

# Receptive field

Область исходного изображения, которая влияет на конкретный признак

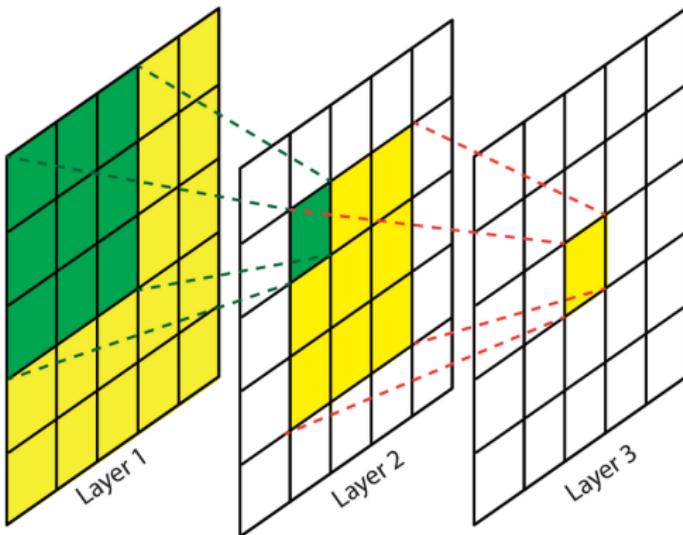


Image credit

# Dilation

Dilated convolution - позволяет увеличить область действия свертки (receptive field)

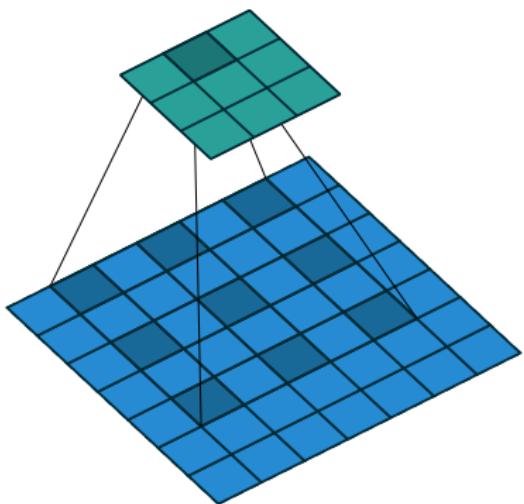
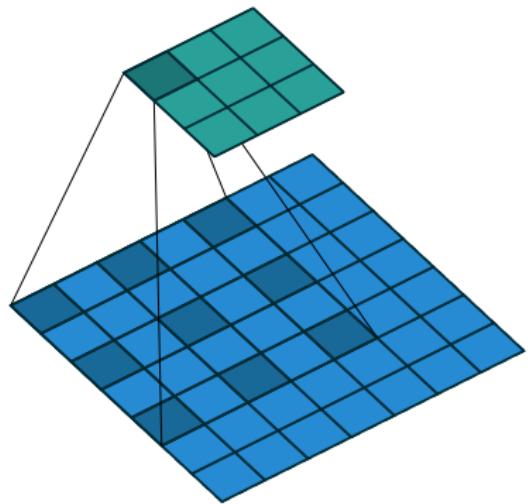


Image credit

# Channels

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

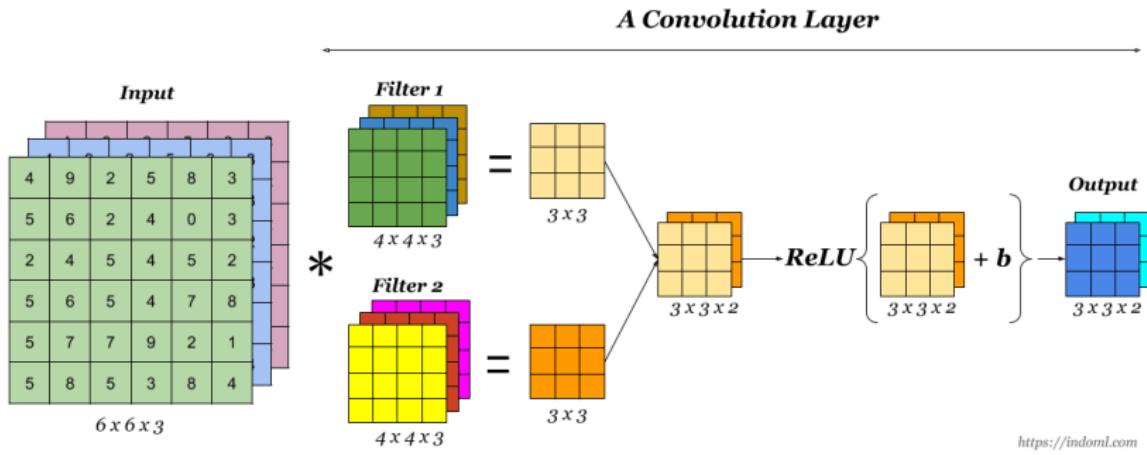
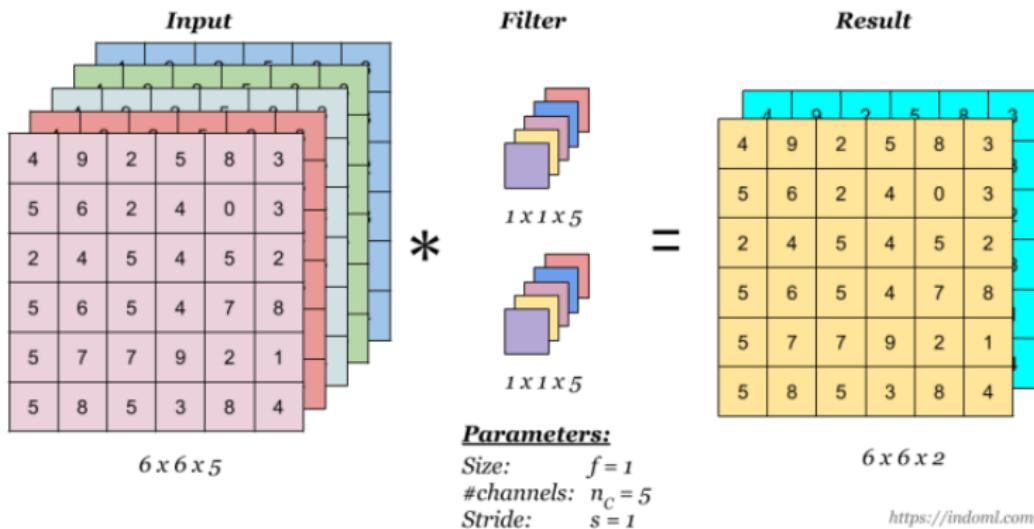


Image credit

# 1x1 Convolution

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

- Преобразование признаков
- Изменение числа каналов



# Convolutional 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

Количество параметров  $K \times K \times C_{in} \times C_{out} + C_{out}$

# Pooling

## Pooling (субдискретизация)

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

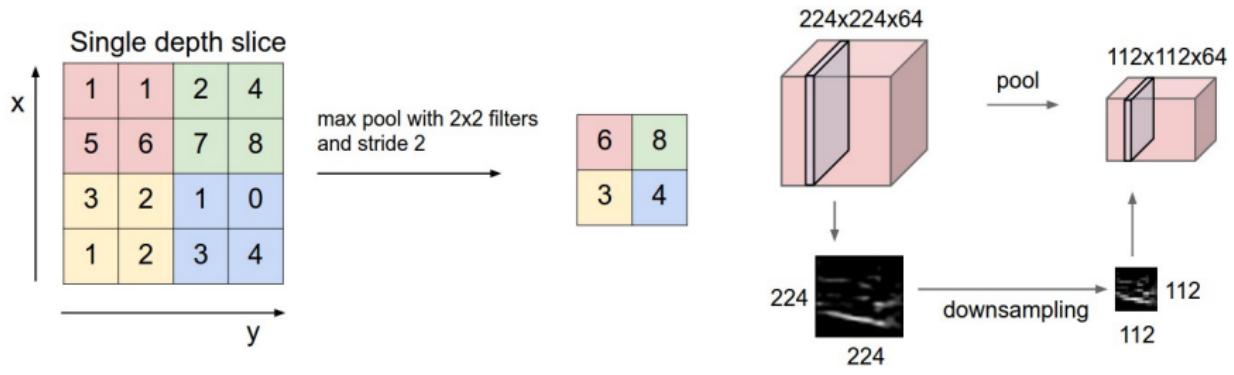
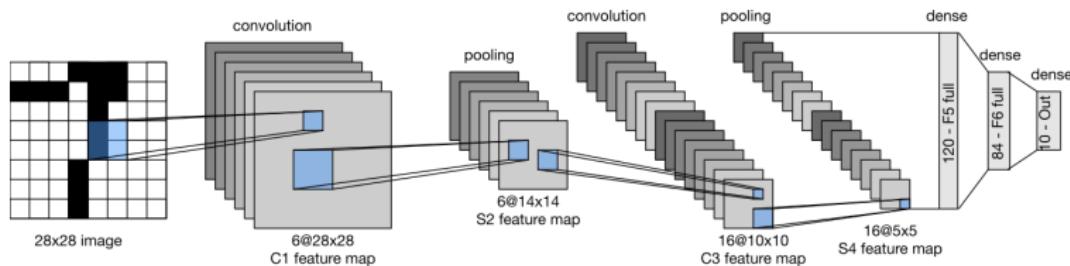


Image credit

# LeNet

Стандартная архитектура для классификации:

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



LeCun et al., 1998 (Image credit)

## ImageNet Large Scale Visual Recognition Challenge



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

## Revolution of depth

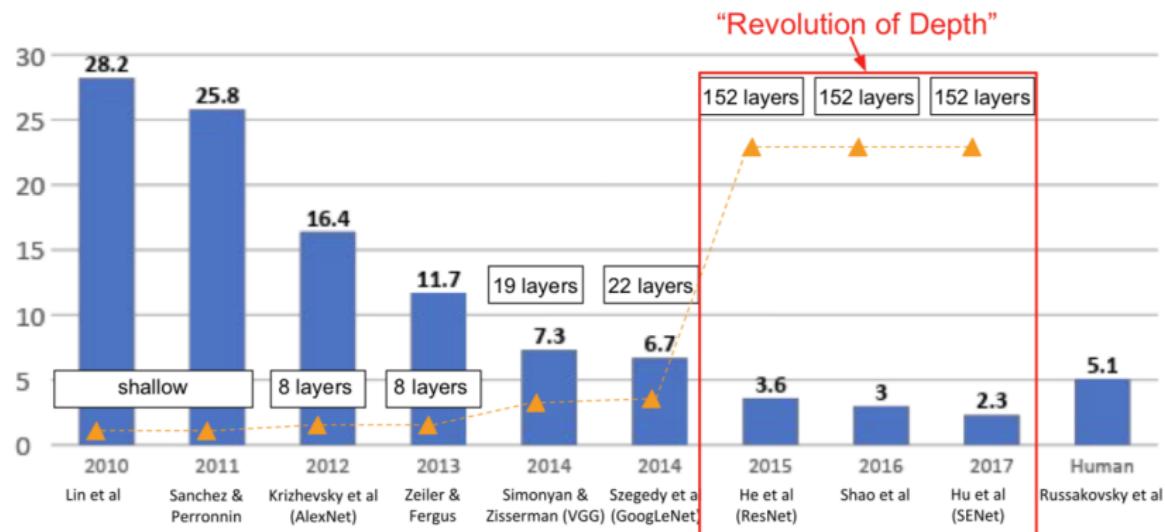
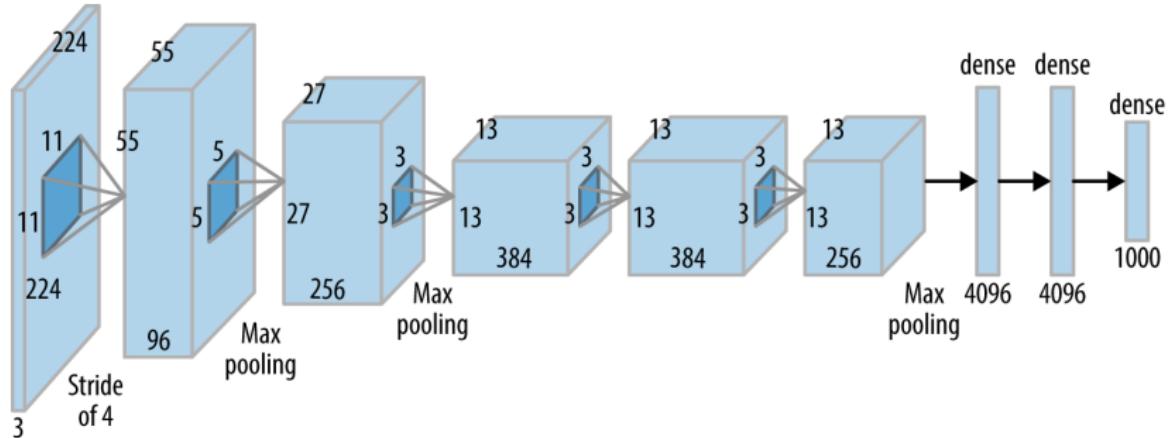


Image credit

# Эволюция сверточных сетей

---

# AlexNet



Krizhevsky et al. 2012 (Image credit)

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

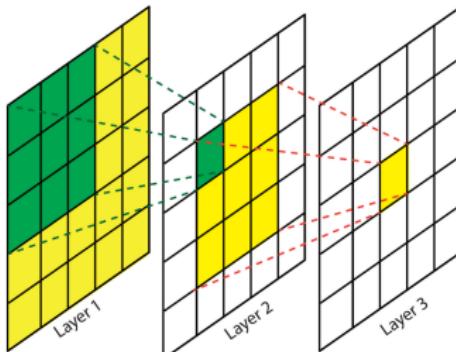


Image credit

Идея - давайте представлять большие свёртки как комбинации свёрток  $3 \times 3$ .

- Меньше параметров и быстрее, сеть глубже
- 2 свёртки  $3 \times 3$  имеют receptive field как одна  $5 \times 5$
- 3 свёртки  $3 \times 3$  имеют receptive field как одна  $7 \times 7$

# VGG

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

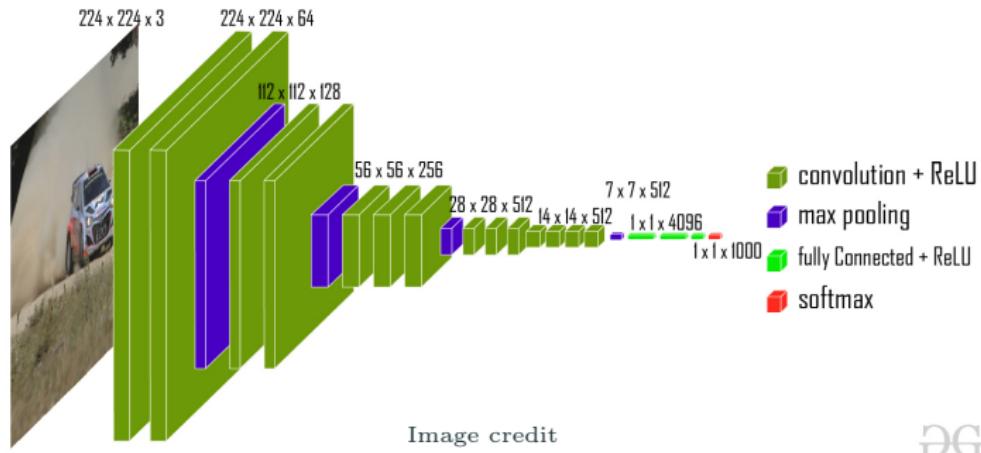


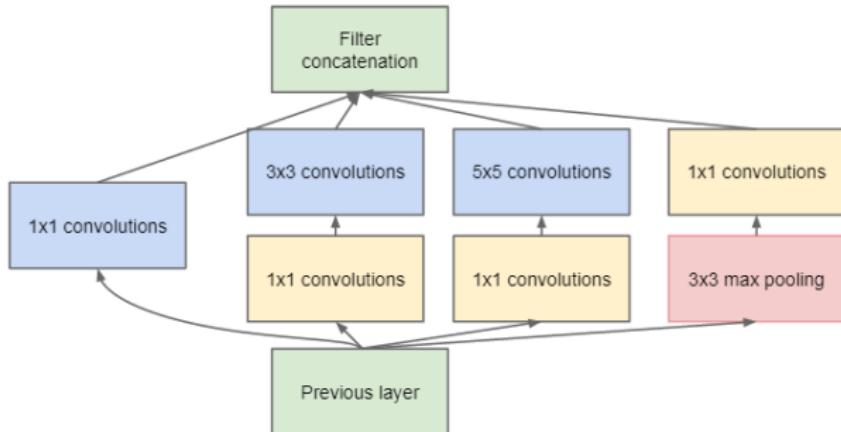
Image credit



- Свертки  $3 \times 3$
- 16 / 19 слоев
- 140M параметров

# Inception block

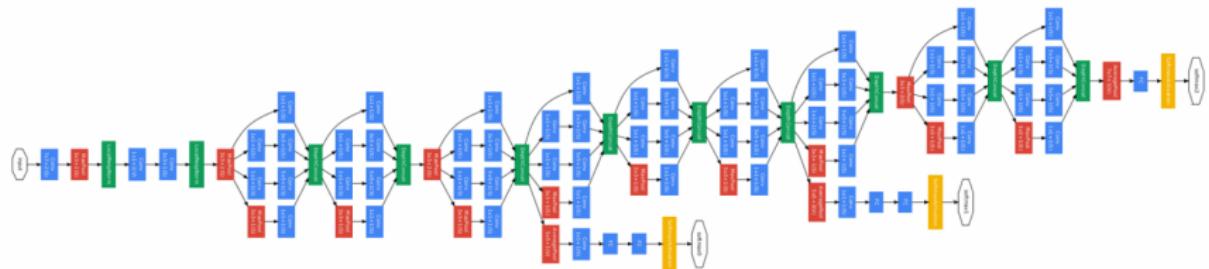
Szegedy et al. Going Deeper with Convolutions (2014)



$1 \times 1$  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

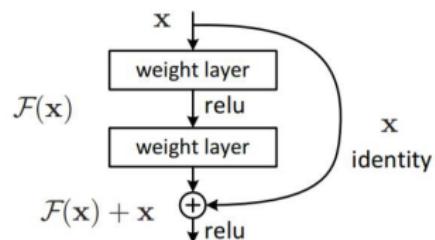
# Residual connections

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

Обучаем разность между  
очередным уровнем и предыдущим -  
residual learning

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

Тогда градиенты беспрепятственно  
проходят через такой блок и не  
затухают



Residual block

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

# Residual connections

---

Loss landscape without and with skip connections

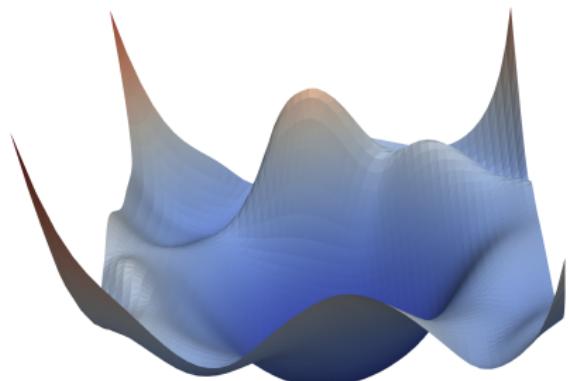
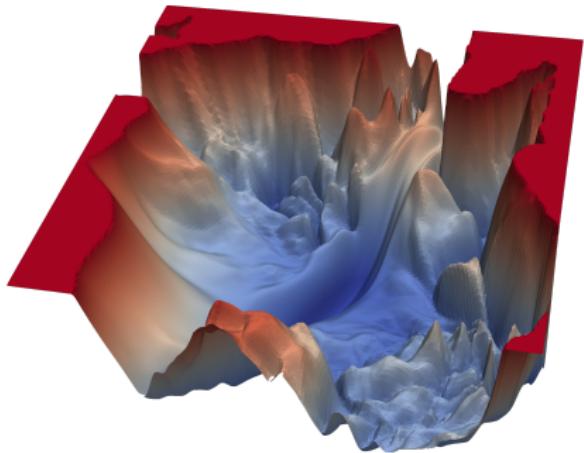
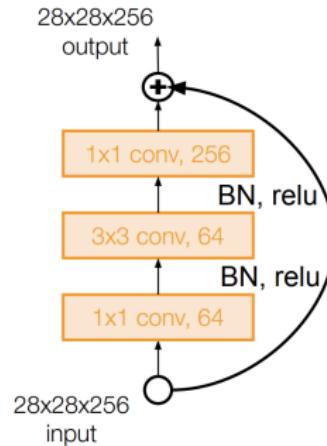


Image credit

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)

Конкatenируем, а не складываем выходы предыдущих слоев

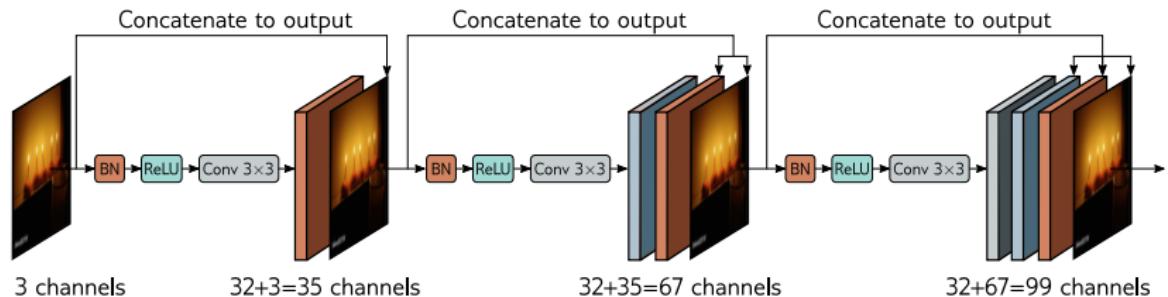
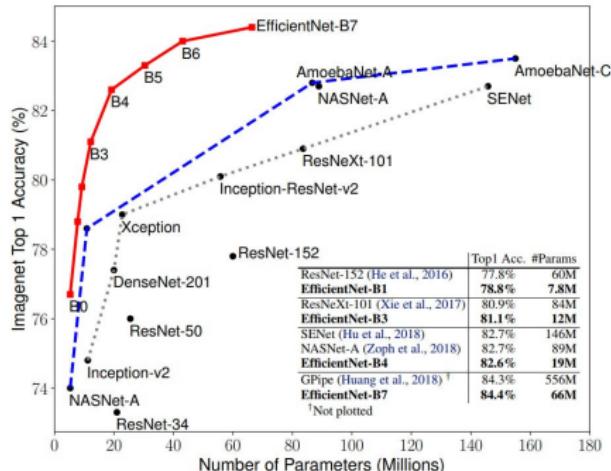


Image credit

# 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)



## В следующий раз

Следующее занятие будет 26 февраля, 19 февраля занятия не будет!

- Задачи Computer Vision (гостевая лекция)