

Цифровая обработка изображения

6. Сверточные нейронные сети: практическое
применение

План занятия

- Данные по задачам компьютерного зрения
- Архитектуры сверточных сетей
- Обучение сверточной сети на практике

ImageNet

ImageNet

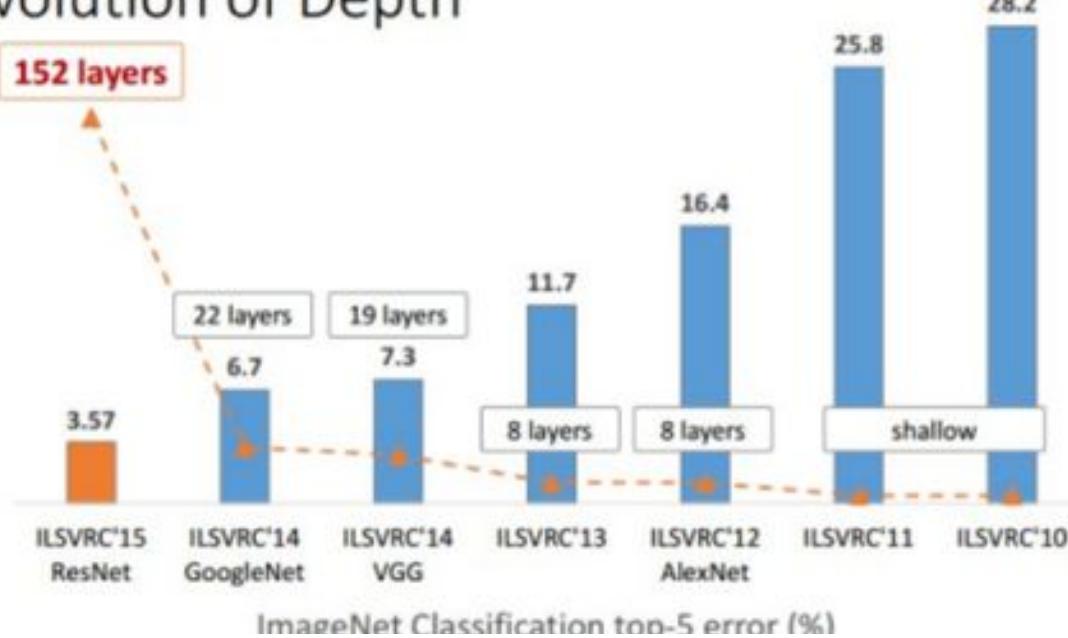
IMAGENET

- 1,000 object classes (categories).
- Images:
 - 1.2 M train
 - 100k test.



ImageNet

Revolution of Depth



Pascal VOC

Pascal VOC (Visual Object Classes)



Visual Object Classes Challenge 2009 (VOC2009)



[click on an image to see the annotation]

<http://host.robots.ox.ac.uk/pascal/VOC/>

Pascal VOC

Bicycle



Bus



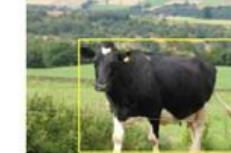
Car



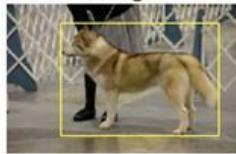
Cat



Cow



Dog



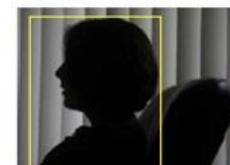
Horse



Motorbike



Person



Sheep



Pascal VOC



Image

Class map

Instance map

Part map

Part map (high level)

COCO Common Object in Context

COCO Dataset



<http://cocodataset.org/dataset.htm>

Списки открытых датасетов

- [Are we there yet?](#)
- [Computer Vision Dataset on the web](#)
- [Yet Another Computer Vision Index To Datasets](#)
- [Computer Vision Online Datasets](#)
- [CVOnline Dataset](#)
- [CV datasets](#)
- [Visionbib](#)

Данные из поисковых систем

Яндекс
Поиск №1 в России*

ПОИСК КАРТИНКИ ВИДЕО КАРТЫ МАРКЕТ НОВОСТИ ПЕРЕВОДЧИК ЕЩЁ

Найти

большой барьерный риф

переулки Москвы фото

криперсы

варенье

весенние этюды

реалистичные рисунки животных

транспорт будущего

редкие птицы

лукс одёжды

космические корабли будущего

пейзажи сибири фото

первые самолёты

Инструмент разметки - LabelMe



Welcome to LabelMe, the open annotation tool.

The goal of LabelMe is to provide an online annotation tool to build image databases for computer vision research. You can contribute to the database by visiting the annotation tool.

Log In

You need to log in.

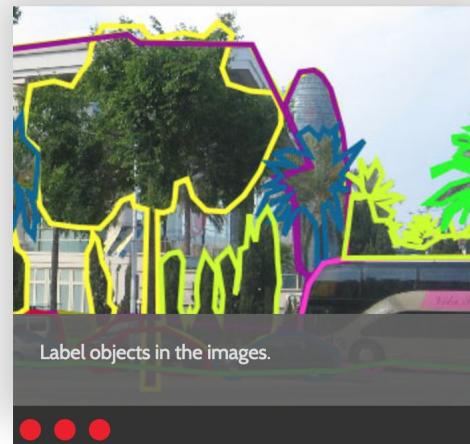
Username

Password
[Forgot your password?](#)

or [Sign Up](#) [Why?](#)

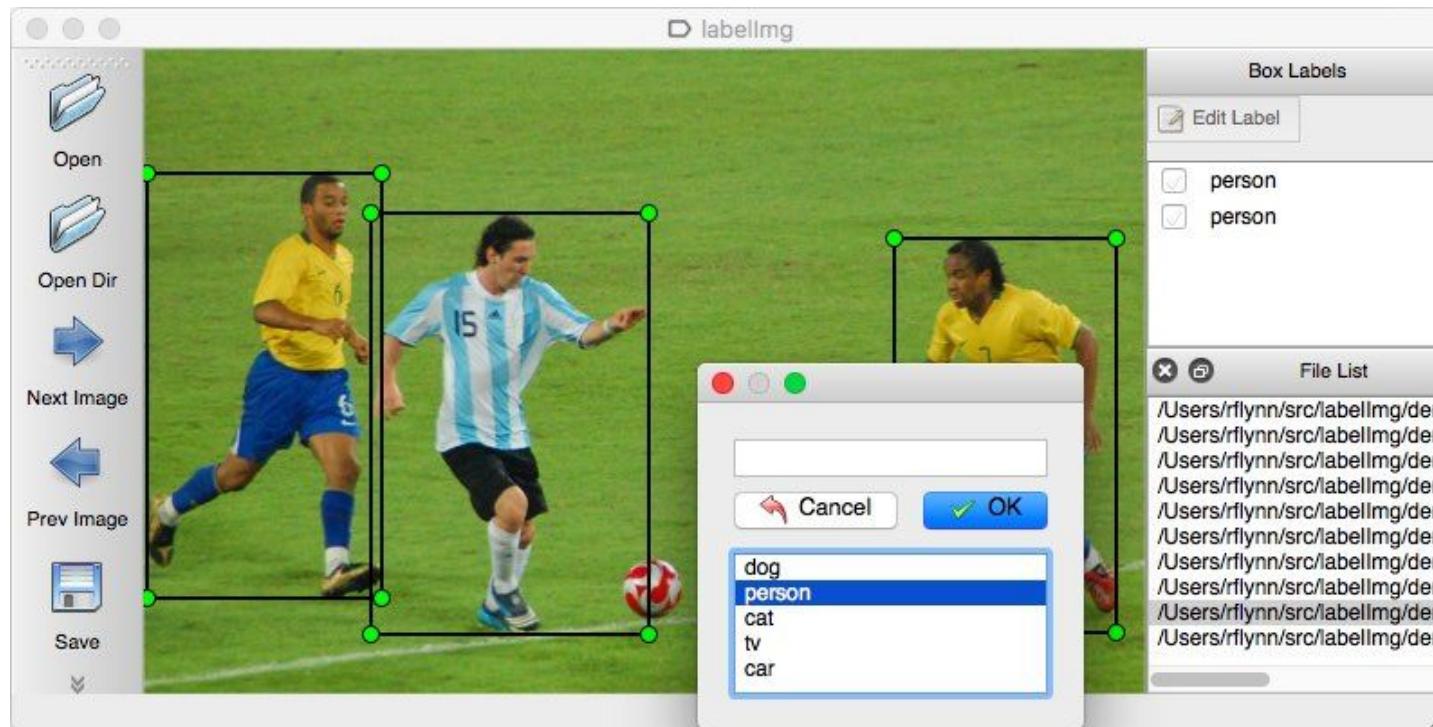
Name

Institution



A photograph of a street scene with buildings and trees. Various objects are outlined with thick yellow and purple lines, demonstrating the tool's ability to segment different elements in an image. At the bottom of the image, the text "Label objects in the images." is displayed, followed by three red dots.

Инструмент разметки - LabelImg



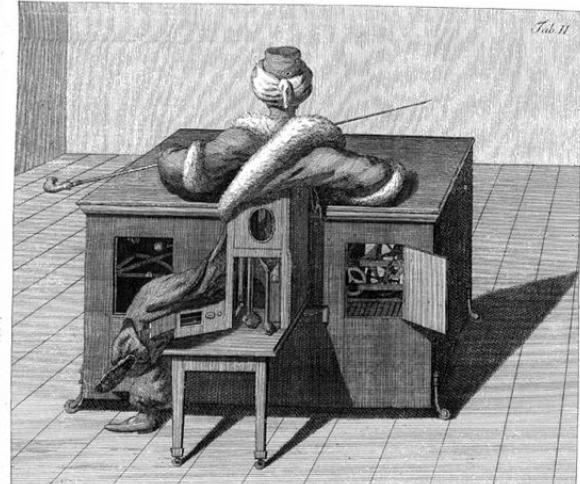
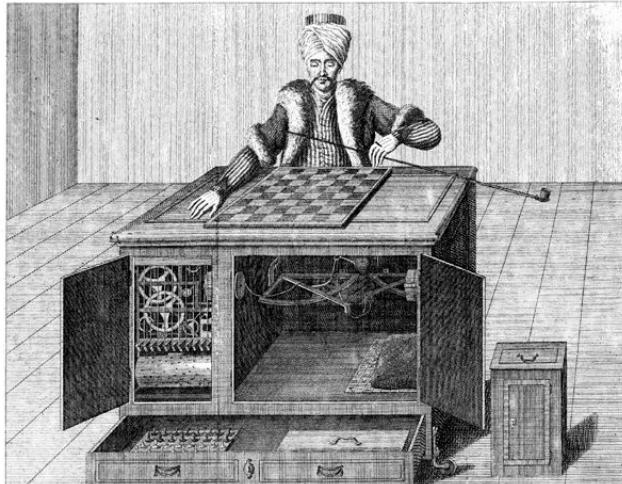
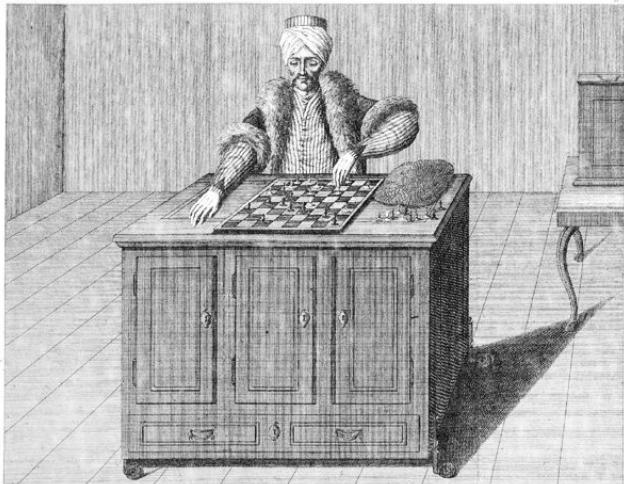
<https://github.com/tzutalin/labelImg>

Сервис разметки - Mechanical Turk



<https://www.mturk.com>

Сервис разметки - Mechanical Turk



<https://www.mturk.com>

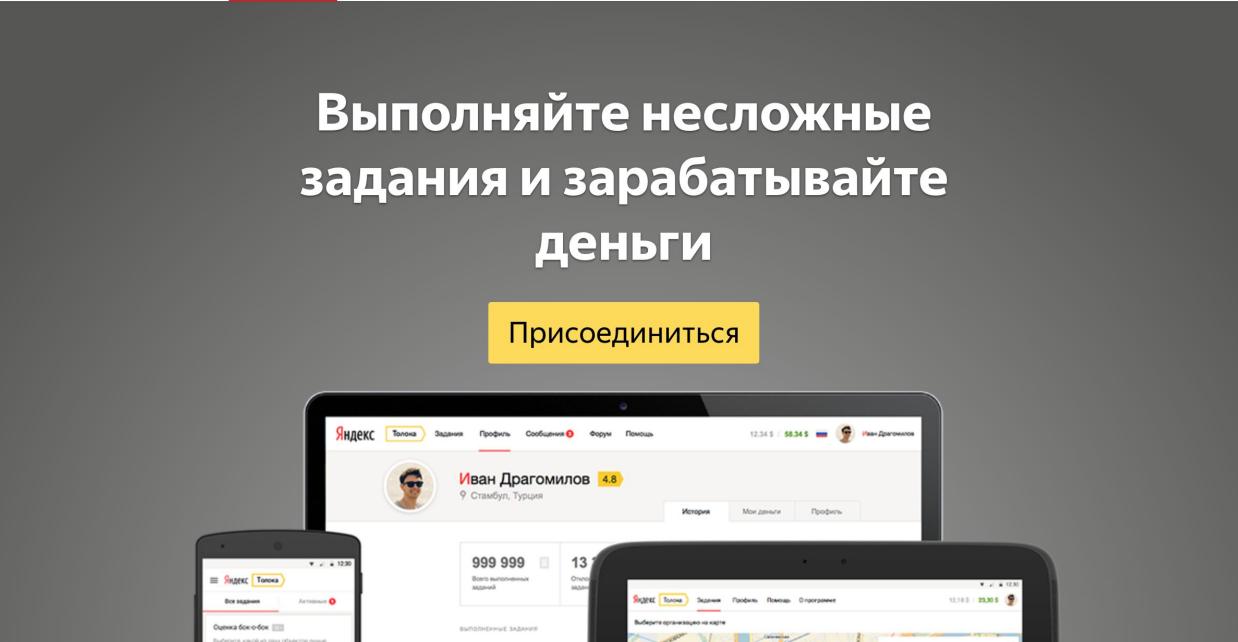
Сервис разметки - Yandex.Toloka

Яндекс Толока Исполнителям Заказчикам

mvjacheslav 

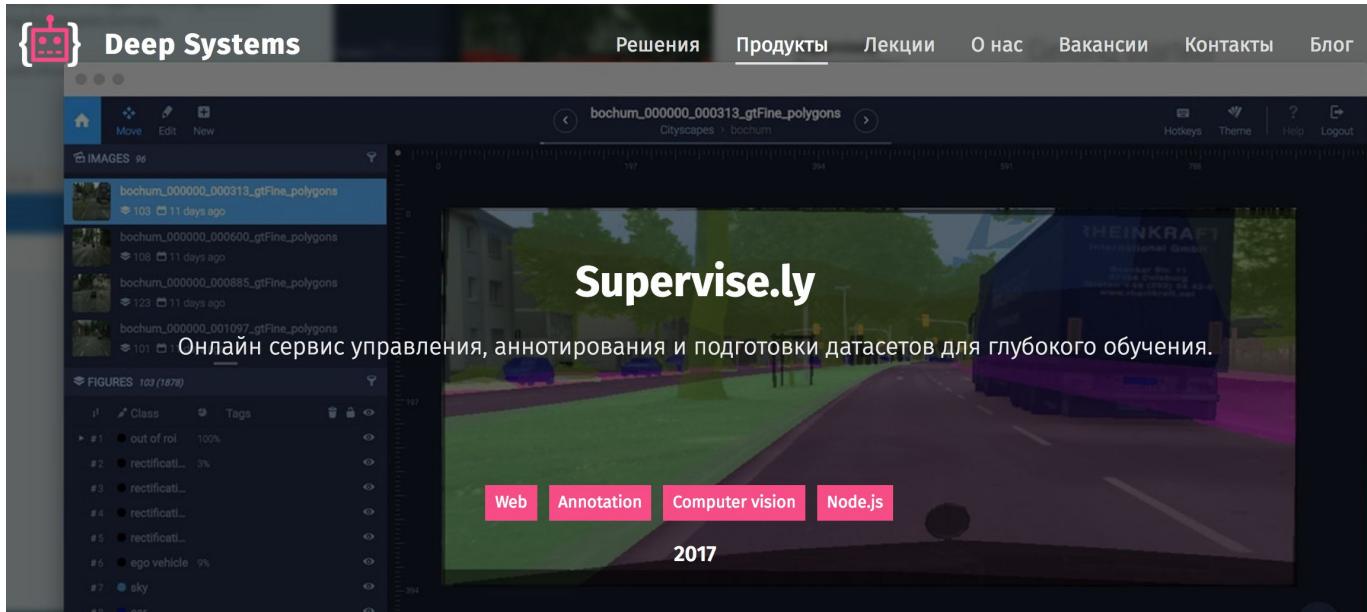
Выполняйте несложные
задания и зарабатывайте
деньги

Присоединиться



<https://toloka.yandex.ru/>

Сервис разметки - Supervise.ly



Перейти на
[Supervise.ly](https://supervise.ly)

Читать на
[Medium](#)

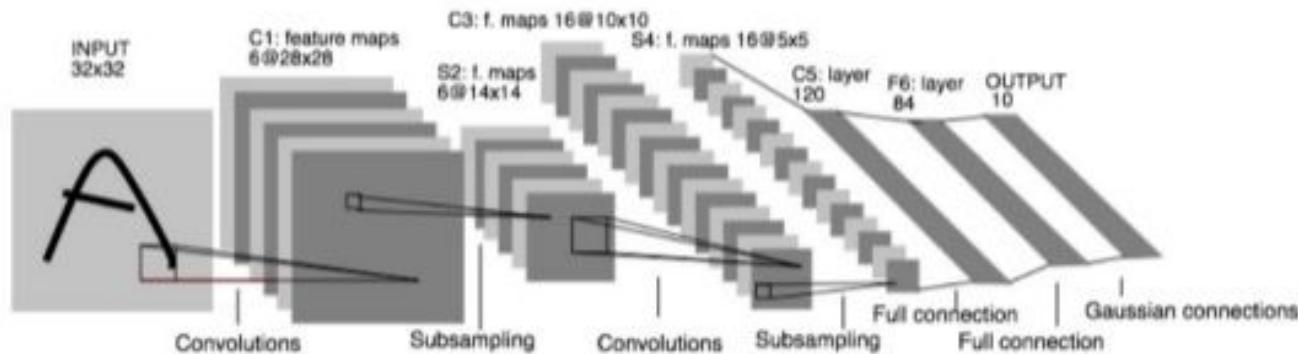
<https://supervise.ly>

Архитектуры сверточных сетей

LeNet - 1989

LeNet - 1989

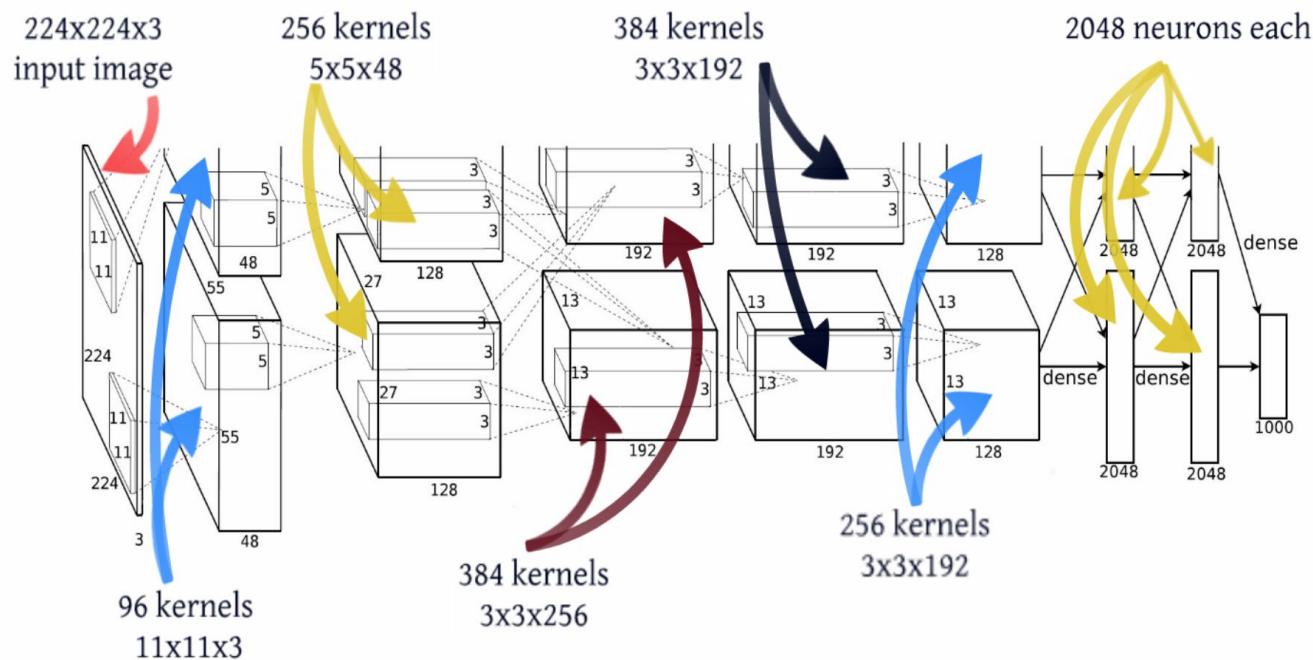
Convolutional Neural Nets (CNNs): 1989



LeNet: a layered model composed of convolution and subsampling operations followed by a holistic representation and ultimately a classifier for handwritten digits. [LeNet]

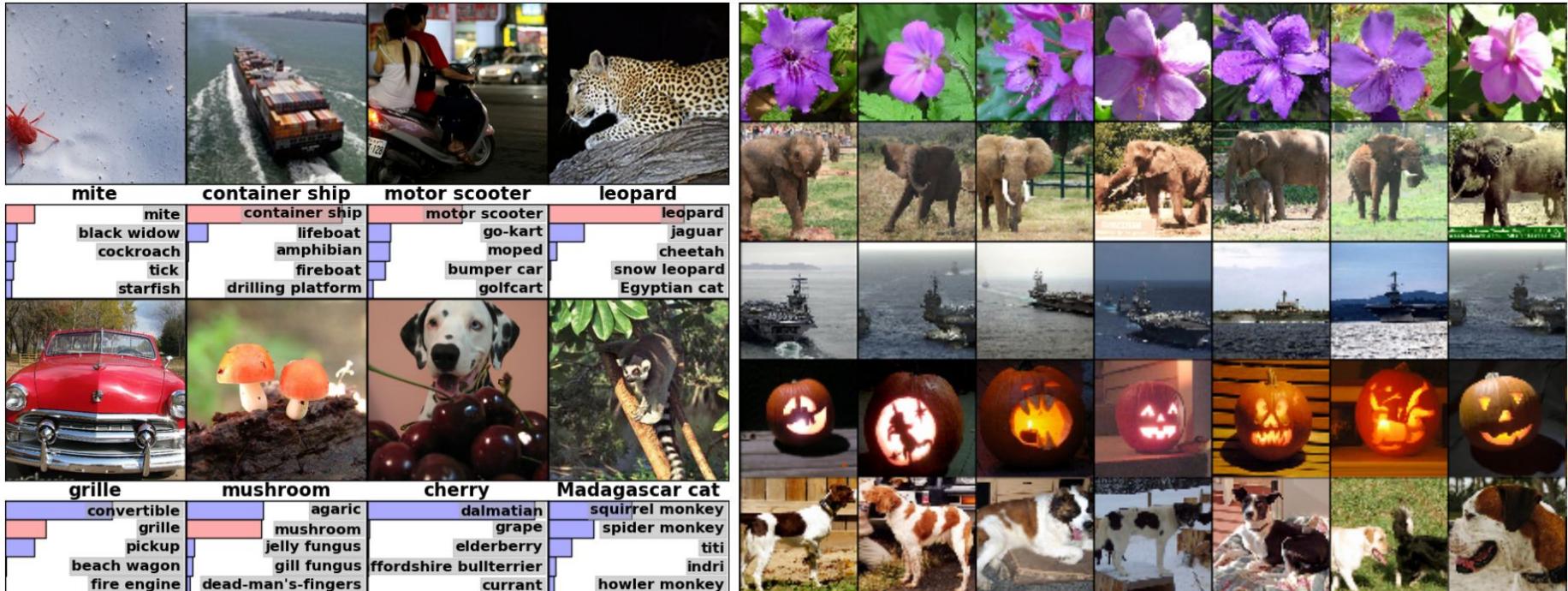
AlexNet - 2012

AlexNet - 2012



[ImageNet Classification with Deep Convolutional Neural Networks](#)

AlexNet - 2012

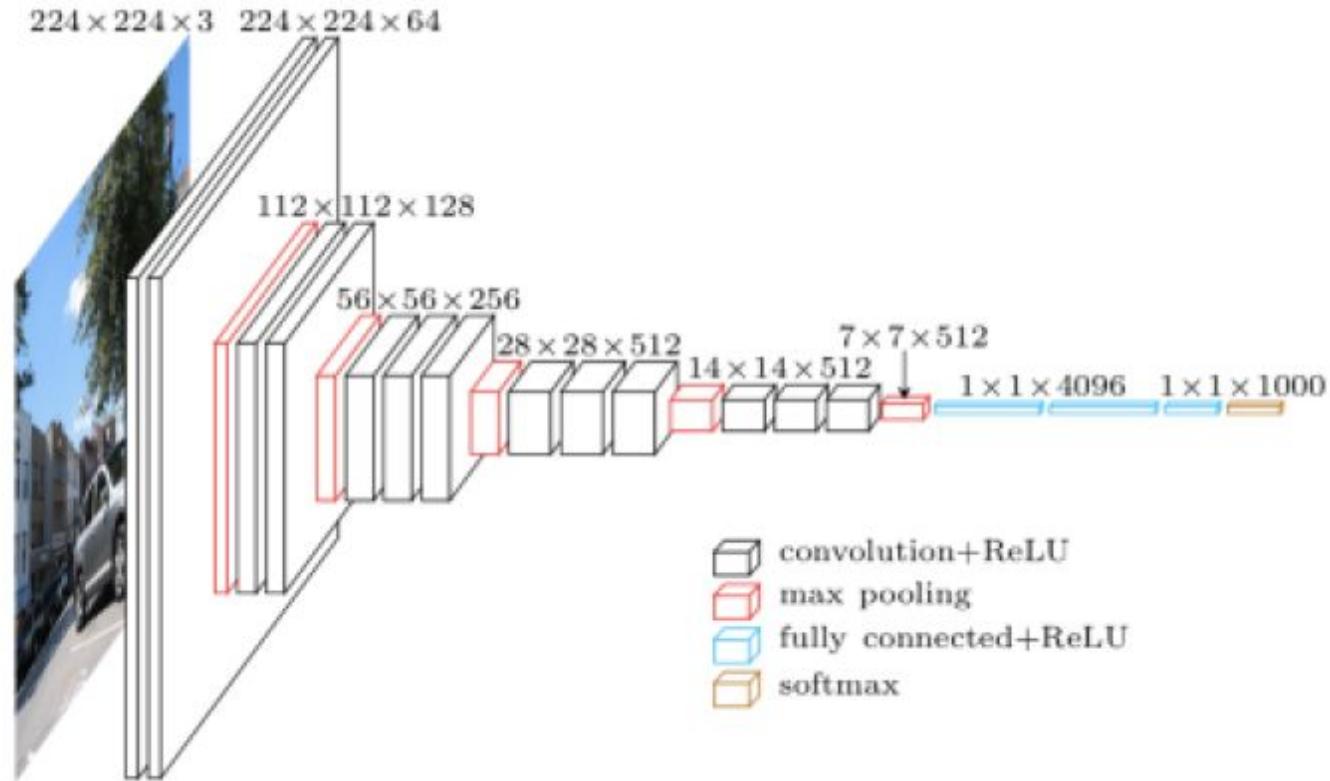


AlexNet - 2012

- параллельная архитектура
- использование ReLU в качестве функций активации
- для регуляризации использование Dropout перед полносвязными слоями
- число параметров ~60M

VGG - 2014

VGG - 2014

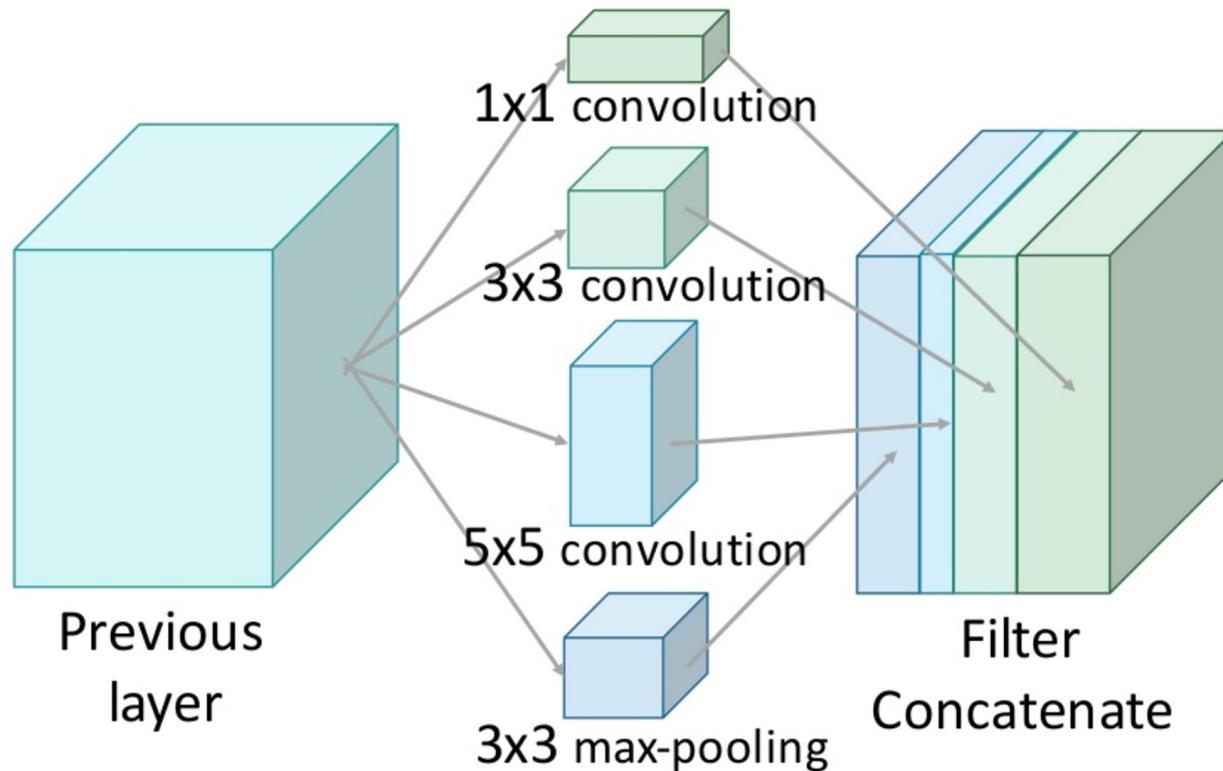


VGG - 2014

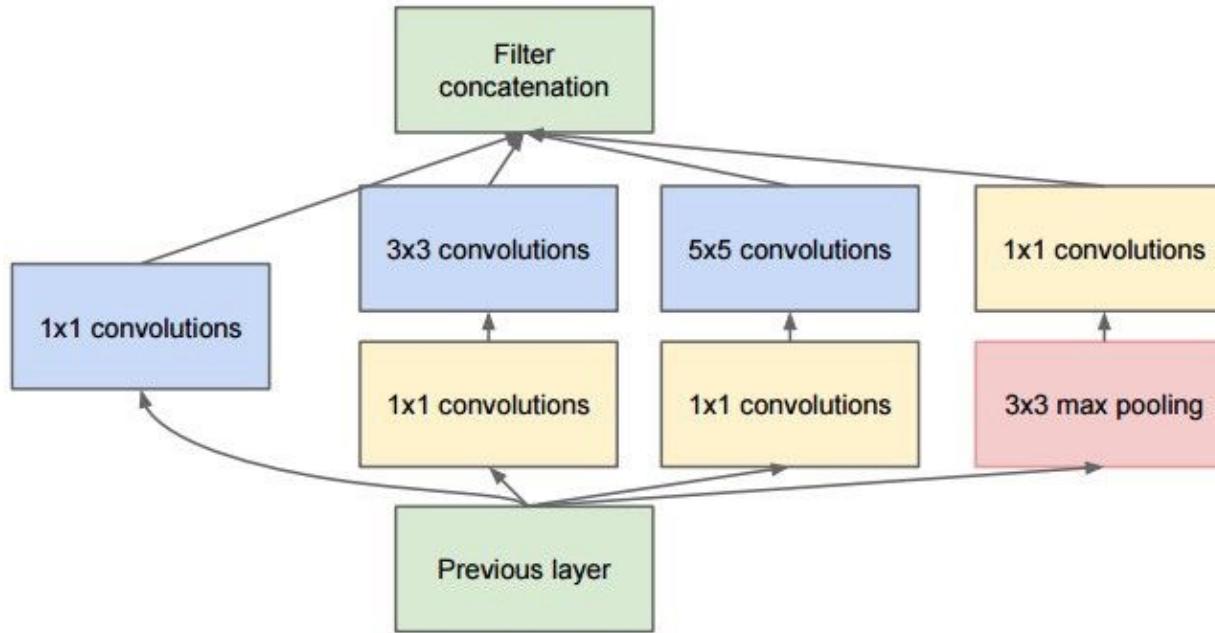
- последовательно применяются свертки фильтров с небольшим размером ядра
- большой объем данных на выходе каждого слоя требует большого количества памяти
- число параметров ~130M

GoogleNet (Inception) - 2014

GoogleNet (Inception) - 2014

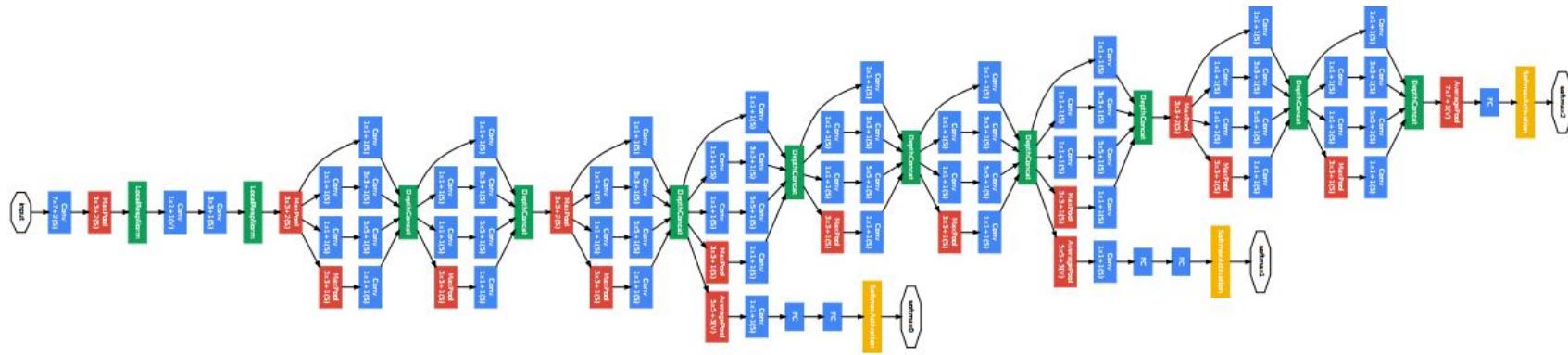


GoogleNet (Inception) - 2014



[Going deeper with convolutions](#)

GoogleNet (Inception) - 2014

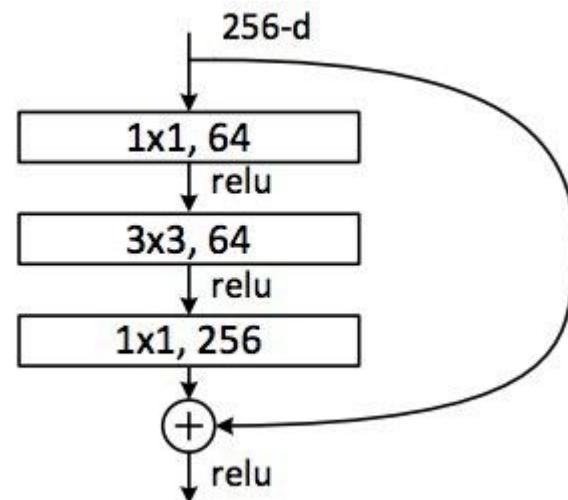
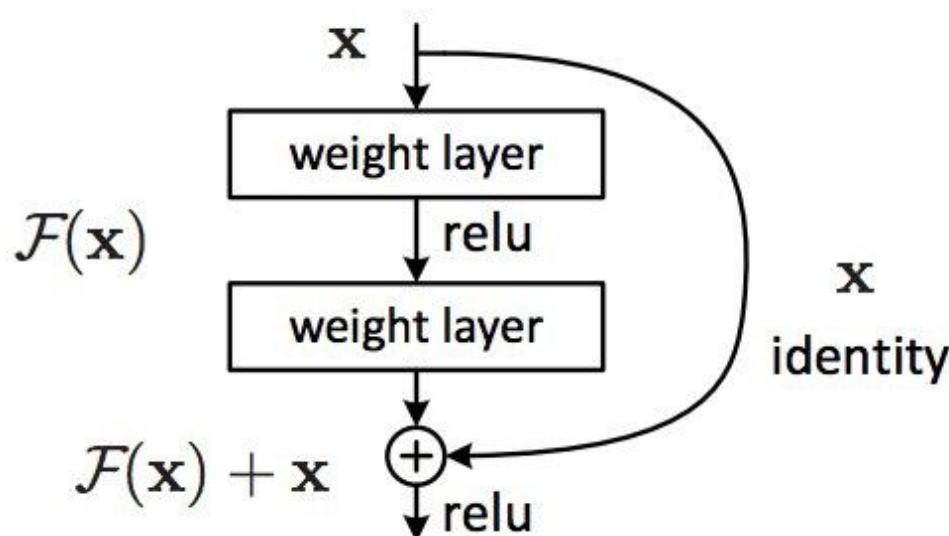


GoogleNet (Inception) - 2014

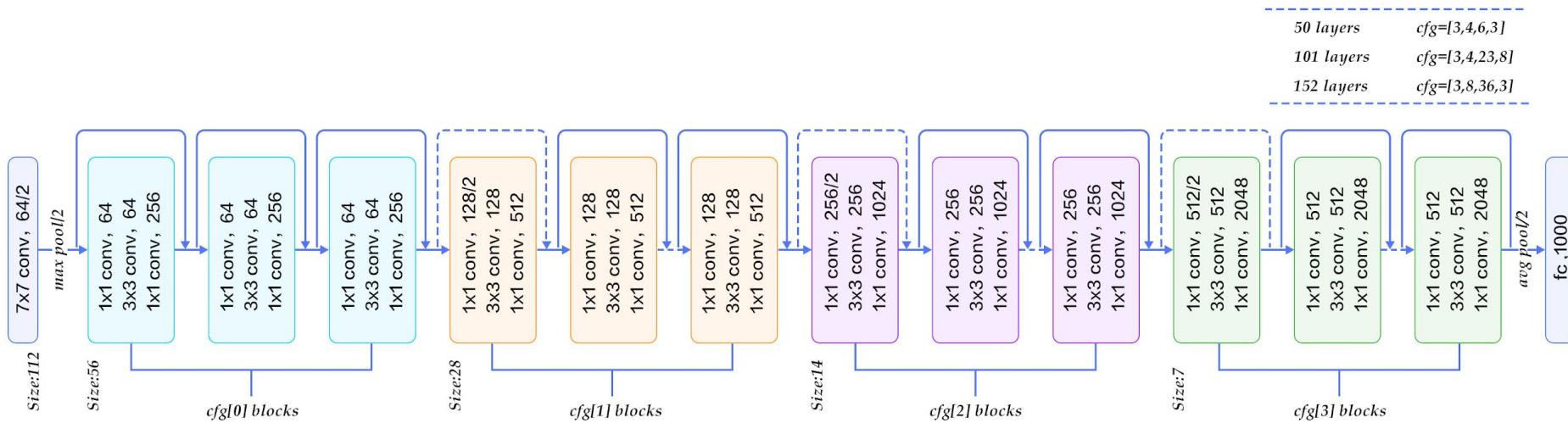
- применяются фильтры разного размера к одним и тем же данным
- уменьшение размерности выхода (глубины) за счет свертки 1×1
- в результате уменьшения глубины получаем ускорение сверток 3×3 и 5×5
- число параметров $\sim 7M$

ResNet - 2015

ResNet - 2015



ResNet - 2015

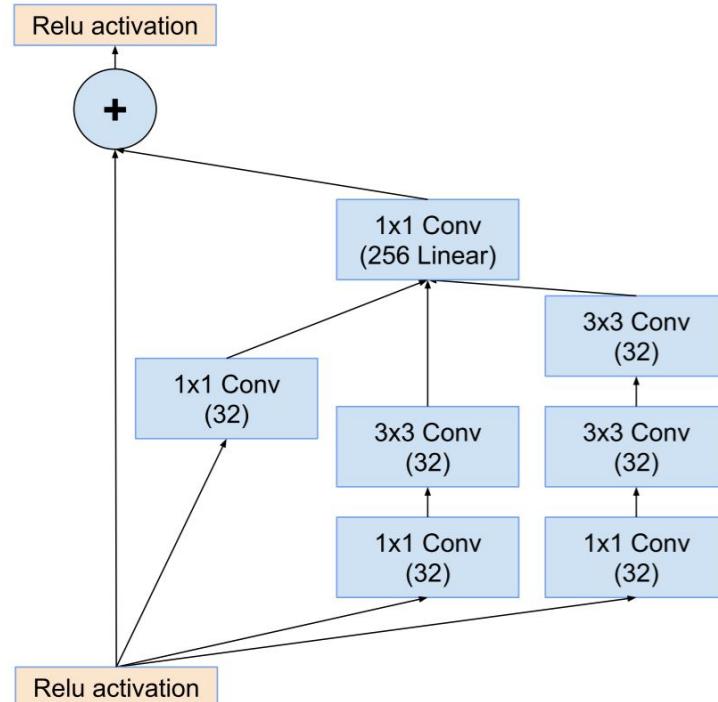


ResNet - 2015

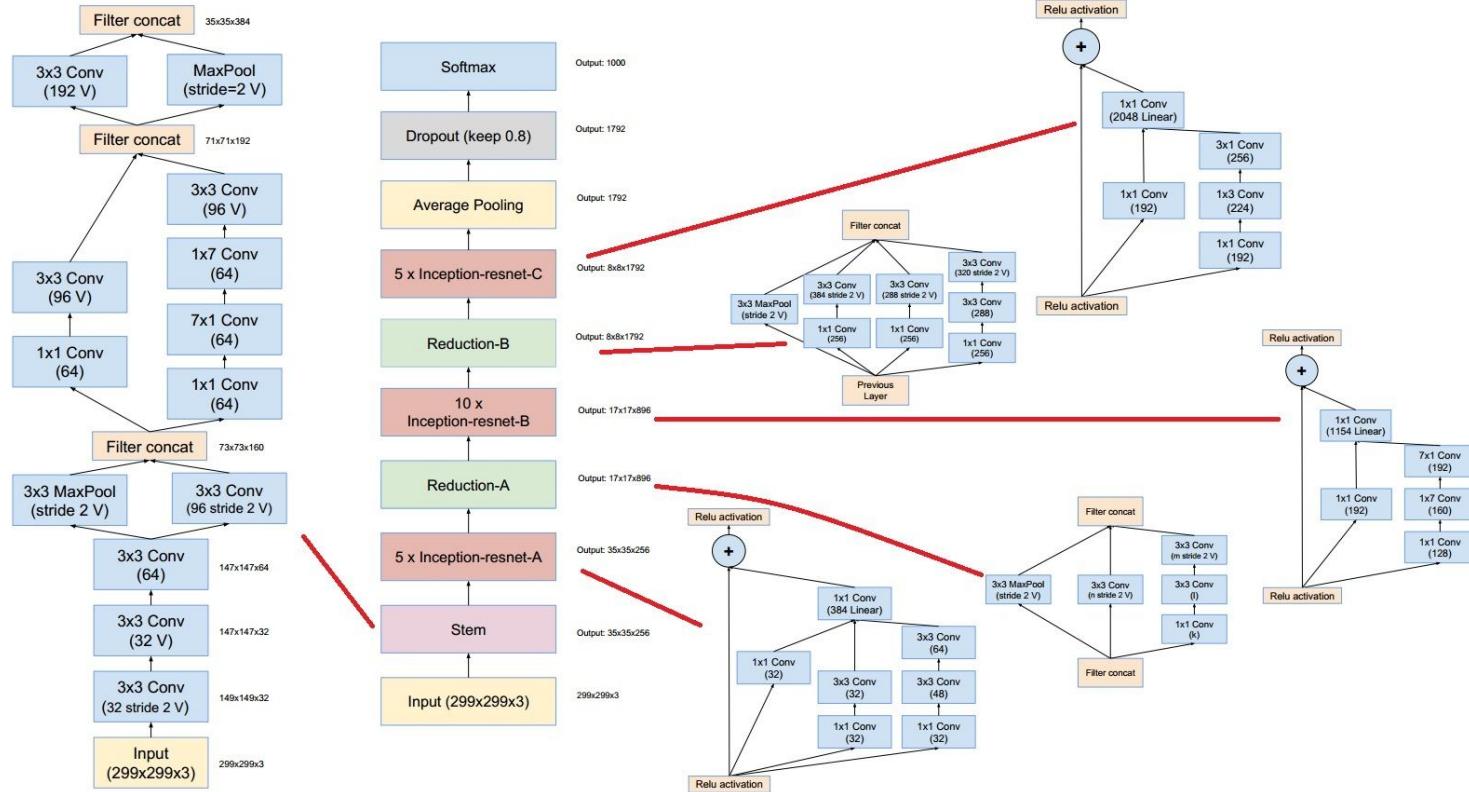
- добавлена параллельная ветка с исходными данными
- первая архитектура, у которой более 100 слоев
- число параметров ResNet 50 ~25M

Inception ResNet

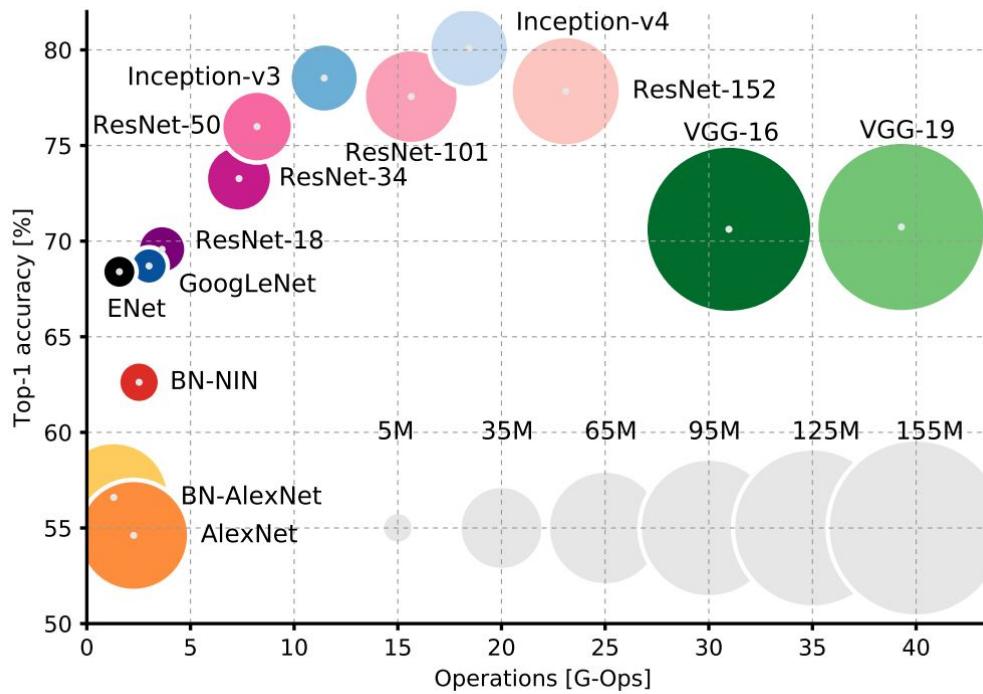
Inception ResNet



Inception ResNet



Сравнение: качество vs скорость



Предобученные модели

- [CAFFE Model-Zoo](#)
- [Keras Applications](#)
- [Tensorflow/Research/Slim](#)
- [Deep Learning Model Convertors](#)

Обучение сверточной сети на практике. Transfer Learning

Transfer Learning

- на практике очень редко обучаются сверточные сети с нуля
- это связано как правило с ограниченным объемом доступных данных и ограниченными вычислительными ресурсами
- как правило, существующие предобученные сети адаптируют под конкретную задачу

Transfer Learning

- как правило выходной слой предобученной сети требует изменения для каждой задачи (разное число классов)
- копируем архитектуру и веса предобученной сети и заменяем последний слой
- дообучаем новый выходной слой на данных задачи

Transfer Learning

	Задача похожа	Задача сильно отличается
Данных мало	обучаем линейный классификатор, на признаках последнего внутреннего слоя	у нас проблемы :) обучаем классификатор на признаках с разных слоев
Данных много	дообучаем несколько последних внутренних слоев	фиксируем первые слои, остальные слои дообучаем

Обучение сверточной сети на практике.
Баланс классов

Баланс классов

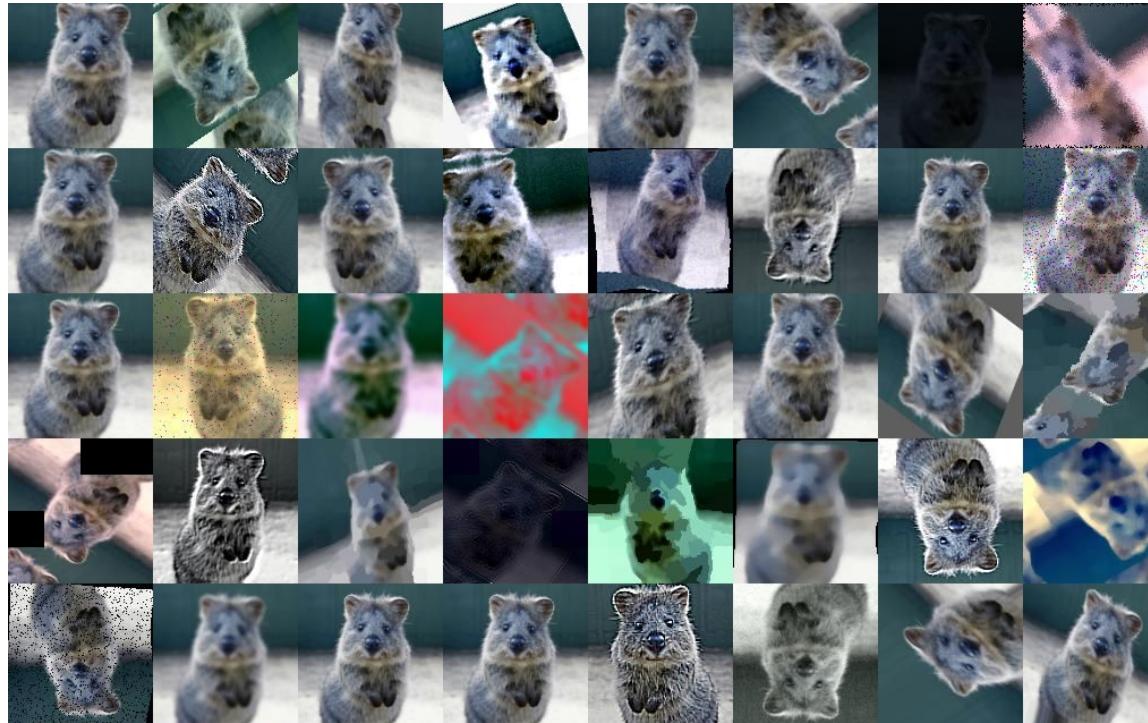
- часто на практике выборка не сбалансирована
- это приводит к тому, что модель переобучается на классы с большим числом примеров и дает смещеннное предсказание

Баланс классов

- выравниваем градиенты, взвешивая ошибку обратно пропорционально числу классов
- выравниваем баланс классов на уровне батча
- добавляем число примеров редких классов за счет аугментации

Обучение сверточной сети на практике. Аугментация данных

Аугментация данных



<https://github.com/aleju/imgaug>

Аугментация данных

- зеркальное отражение по горизонтали
- вырезаем случайную часть из изображения (сгор) и масштабируем до исходного
- аугментация освещенности (в пространстве HSV)
- аугментация цвета (случайный шум по каналам)

Реализация

- на практике аугментированные изображения сильно увеличивают размер выборки
- хранить на диске аугментированные копии нецелесообразно
- процесс аугментации запускают на лету параллельно с обучением
- аугментация выполняется на CPU и существенно не влияет на скорость при обучении на GPU

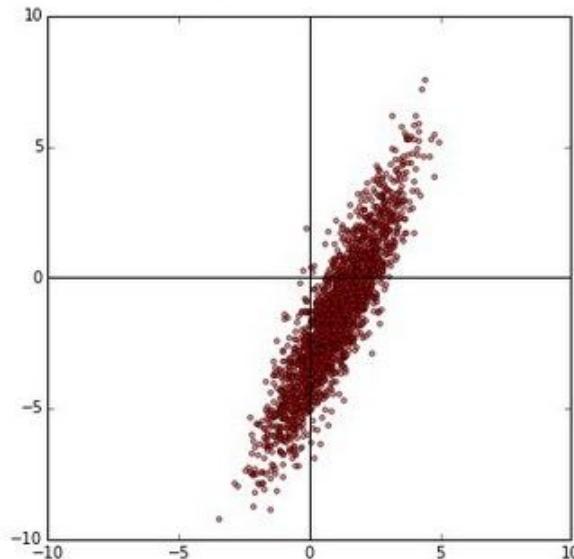
Библиотеки аугментации изображений

- [aleju/imgaug](#)
- [Augmentor](#)
- [keras.preprocessing.image.ImageDataGenerator](#)
- [tf.image](#)

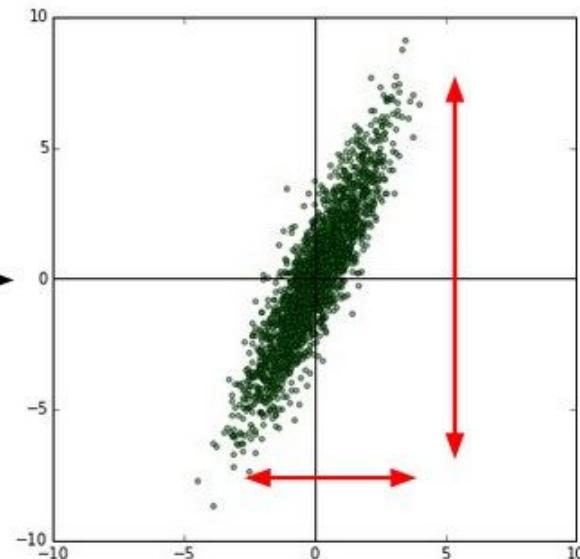
Обучение сверточной сети на практике. Предобработка данных

Центрирование и нормализация

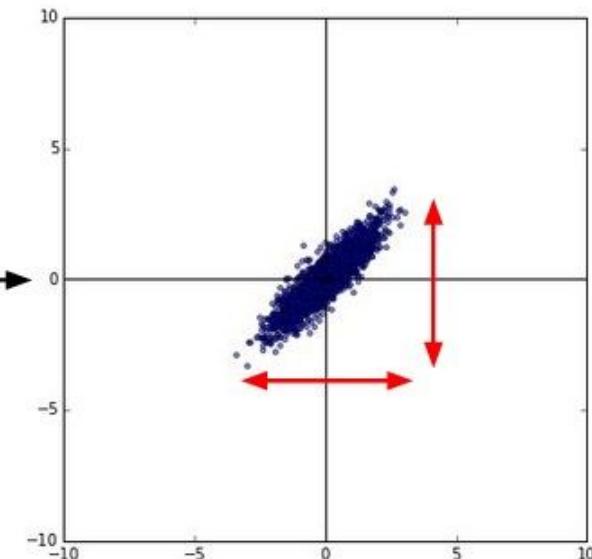
original data



zero-centered data

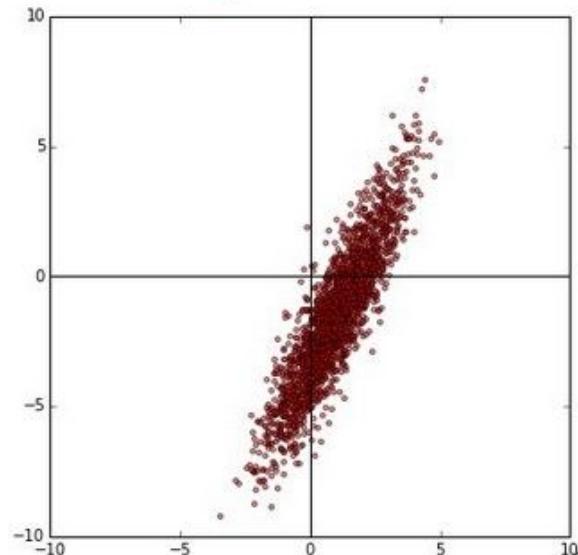


normalized data

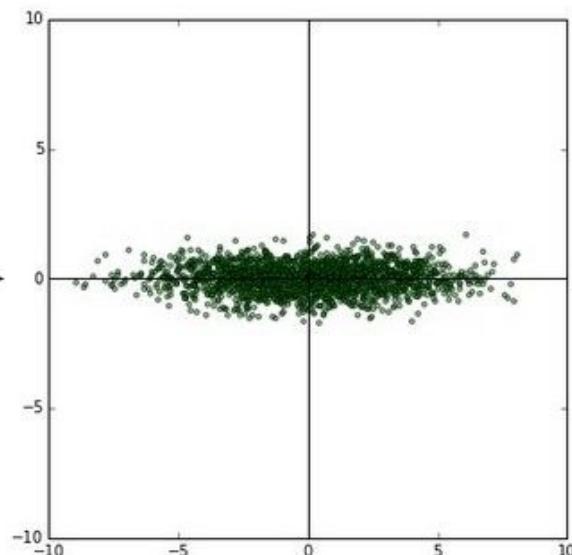


Обеление (PCA whitening)

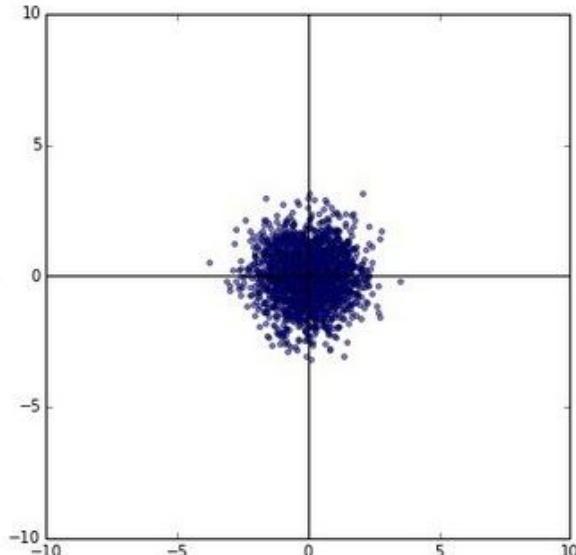
original data



decorrelated data



whitened data

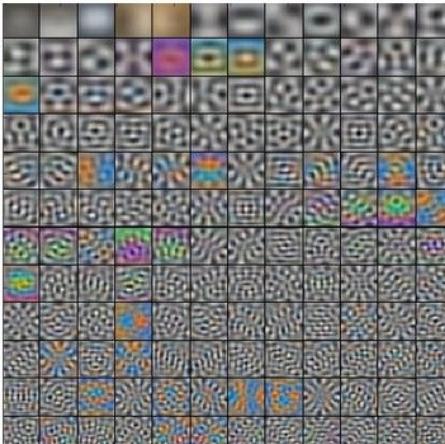


Обеление (PCA whitening)

original images



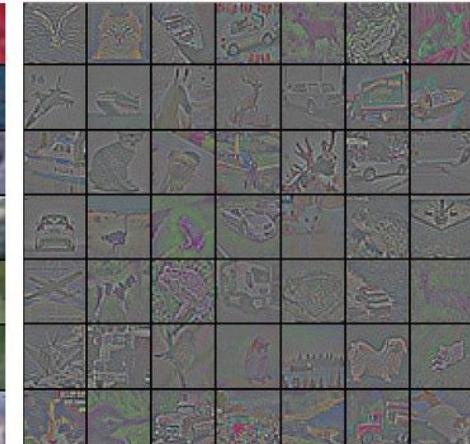
top 144 eigenvectors



reduced images



whitened images



Batch Norm

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_1 \dots m\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \text{BN}_{\gamma, \beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m} \sum_{i=1}^m x_i \quad // \text{mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m} \sum_{i=1}^m (x_i - \mu_{\mathcal{B}})^2 \quad // \text{mini-batch variance}$$

$$\hat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \quad // \text{normalize}$$

$$y_i \leftarrow \gamma \hat{x}_i + \beta \equiv \text{BN}_{\gamma, \beta}(x_i) \quad // \text{scale and shift}$$

Обучение сверточной сети на практике. Инициализация

Инициализация весов

- инициализация различными весами (если нейроны возвращают одинаковые значения, значит и градиенты для них будут одинаковыми)
- инициализация случайными значениями с небольшой дисперсией (гаусс или нормальное - не существенно важно)
- дисперсия растет с увеличением числа входов - нормируем дисперсию на число входов ([lecun_normal](#))

<https://keras.io/initializers/>

Обучение сверточной сети на практике. Оптимизация вычислений

Параметры фильтров

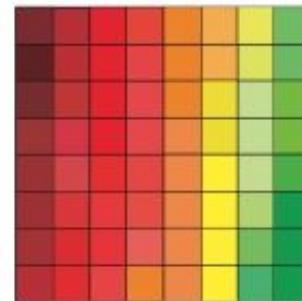
- размеры изображения кратны числу сэмплирующих слоев (Pooling)
- чем меньше размер фильтра, тем меньше операций необходимо выполнить для вычисления свертки
- отступы (padding) необходимо добавлять при свертке для сохранения информации на краях картинки (актуально с увеличением глубины)

Формат данных и производительность

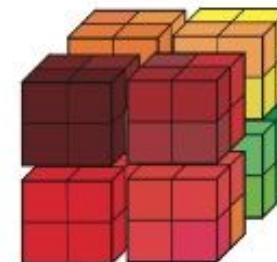
vector



matrix



tensor

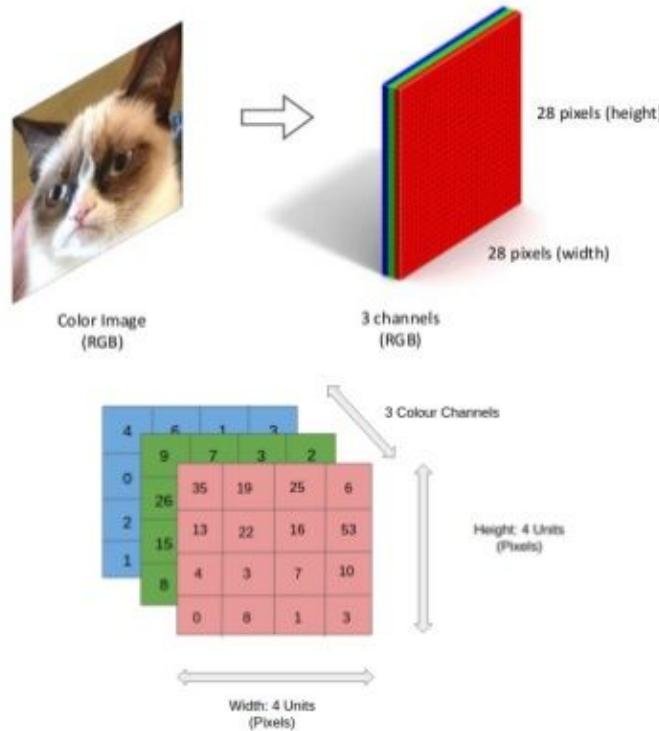


$$\mathbf{v} \in \mathbb{R}^{64}$$

$$\mathbf{X} \in \mathbb{R}^{8 \times 8}$$

$$\mathcal{X} \in \mathbb{R}^{4 \times 4 \times 4}$$

Формат данных и производительность



Формат данных и производительность

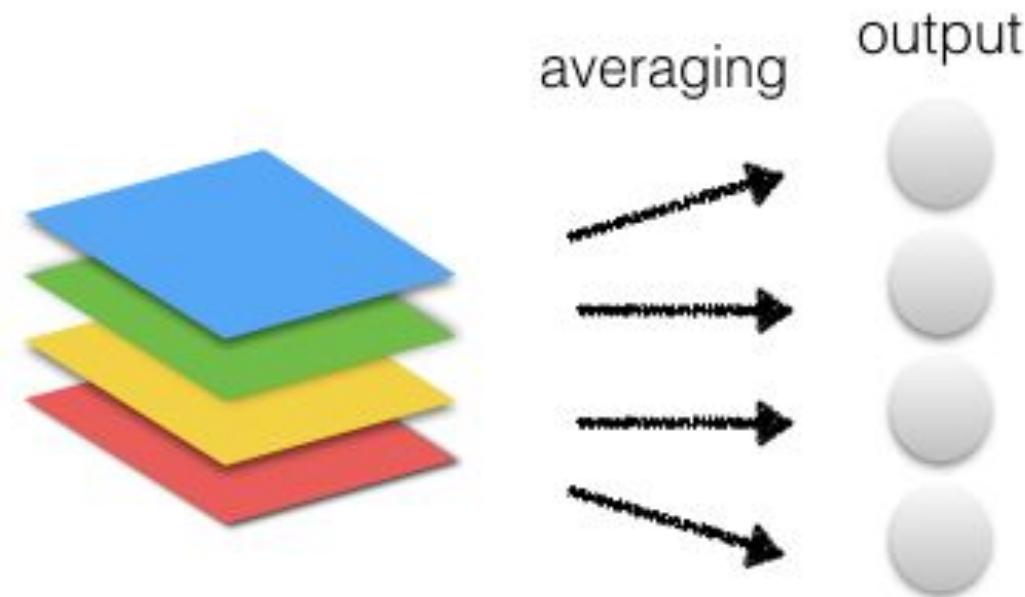
- производительность зависит от порядка измерений
- изменение порядка с NHWC (channel_last) на NCHW (channel_first) дает прирост в скорости вычислений на GPU до 10%
- это связано с особенностью библиотеки cuDNN, которая поддерживает только NCHW формат
- если вы используете NHWC Tensorflow автоматически приводит данные в формат NCHW перед обработкой на GPU
- при этом, в формате NCHW некоторые операции Tensorflow могут не поддерживаться на CPU

Обучение сверточной сети на практике.
Ограничение на размер входного изображения

Размер исходного изображения и FC слой

- сверточные сети с полносвязными слоями имеют ограничение на размер входного изображения
- один из способов снять это ограничение - добавить слой Global Average Pooling перед FC слоем

Global Average Pooling

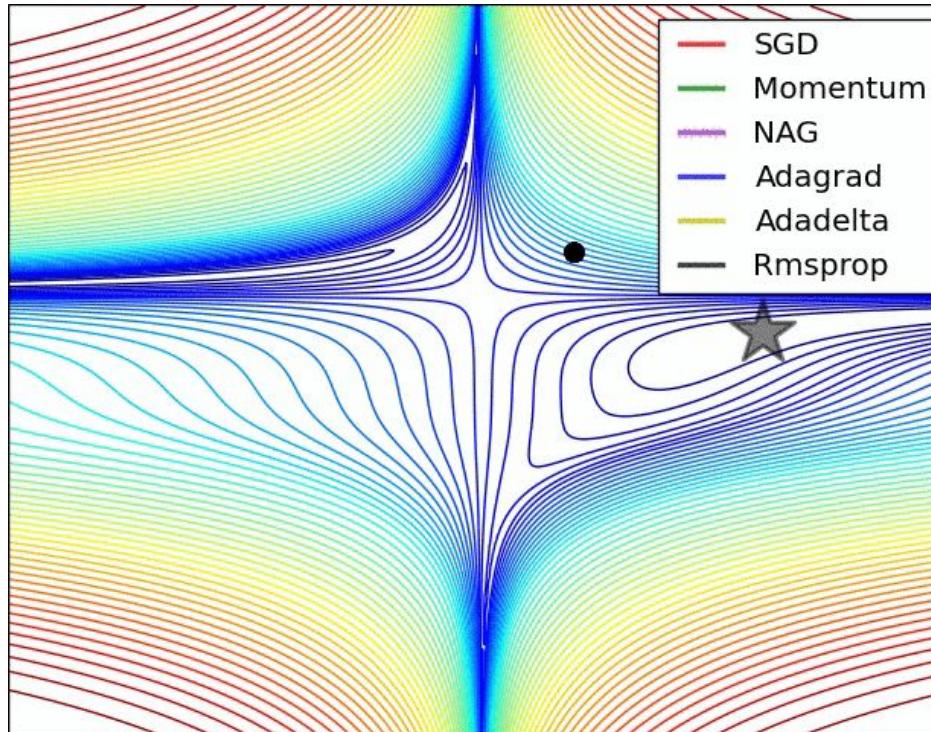


Обучение сверточной сети на практике.
Градиентный спуск

Оптимизация (градиентный спуск)

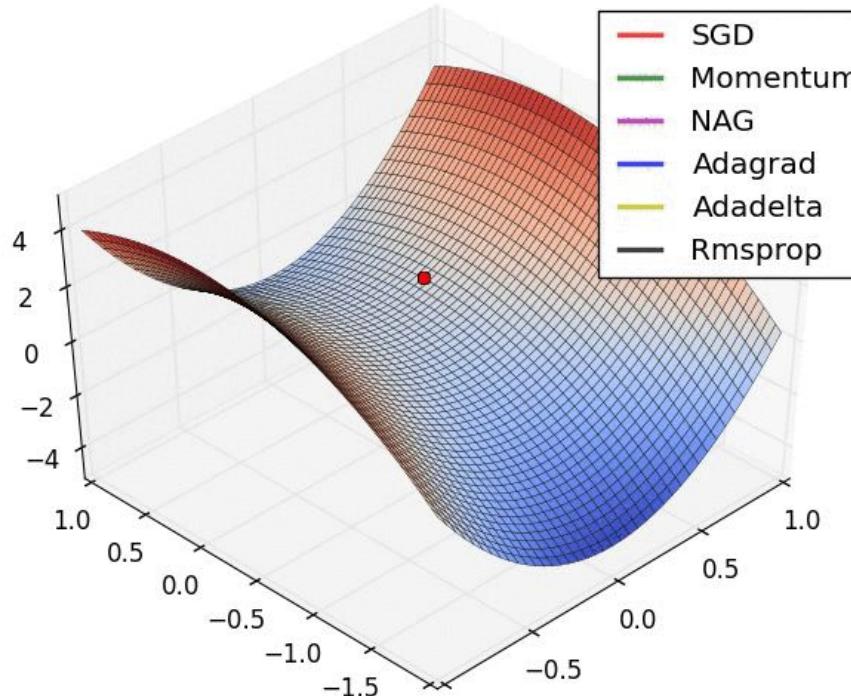
- при увеличении размера батча оценка градиента получается стабильнее, что позволяет увеличить learning rate
- в случае, если не происходит изменение метрики на валидации, стоит уменьшить значение learning rate [ReduceLROnPlateau](#)
- наиболее популярные оптимизаторы: [sgd](#), [adam](#)

Оптимизация (градиентный спуск)



<http://ruder.io/optimizing-gradient-descent/>

Оптимизация - седловая точка

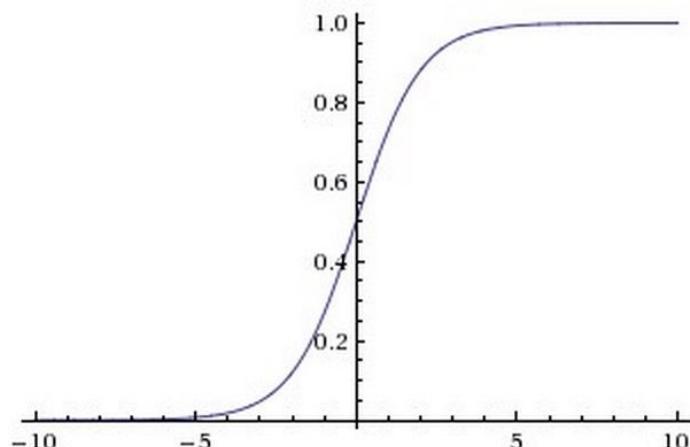


Обучение сверточной сети на практике. Функции активации

<https://keras.io/layers/advanced-activations/>

Функции активации - sigmoid

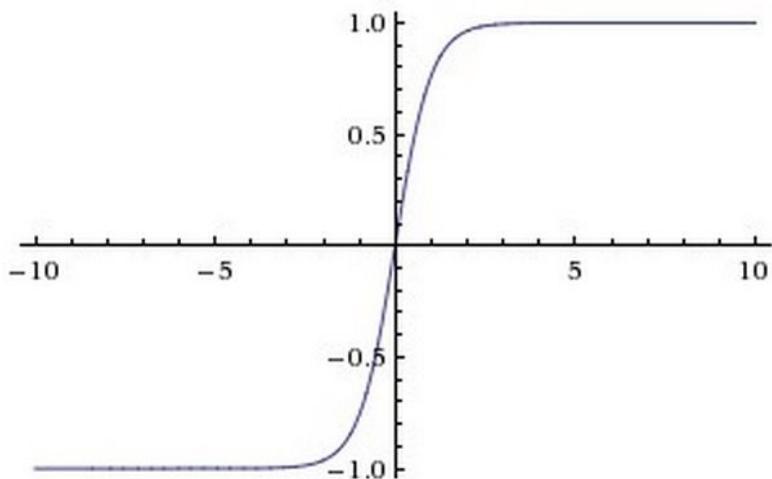
$$\sigma(x) = 1/(1 + e^{-x})$$



- убивает градиент
- смещена относительно нуля - проблема для обучения
- используется на выходном слое

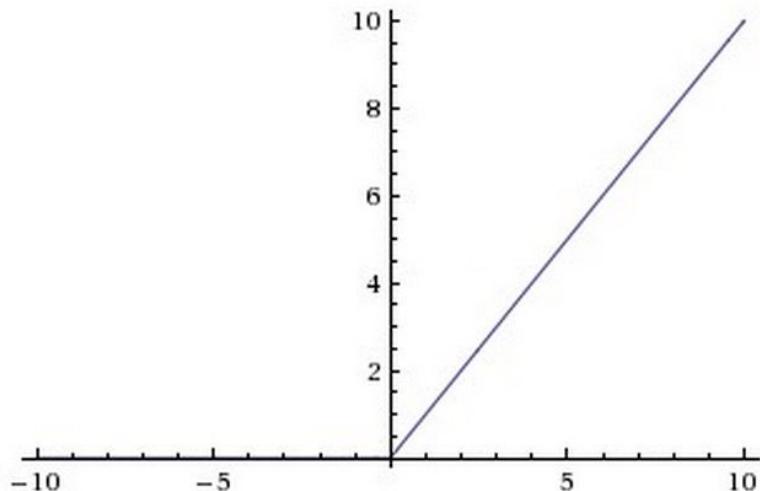
Функция активации - \tanh

- остается проблема с градиентами
- решена проблема с центрированием



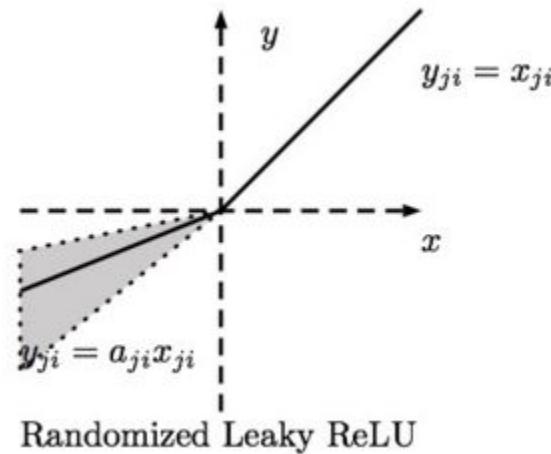
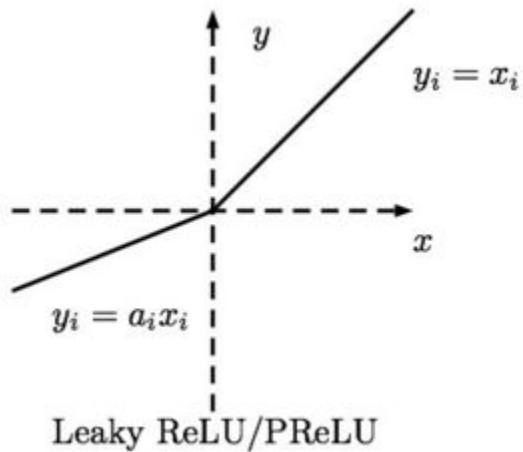
Функция активации - ReLU

$$f(x) = \max(0, x)$$



- простая в реализации
- отсутствие насыщения ускоряет процесс сходимости
- при большом значении градиента значение может уйти в минус и не вернуться dying ReLU

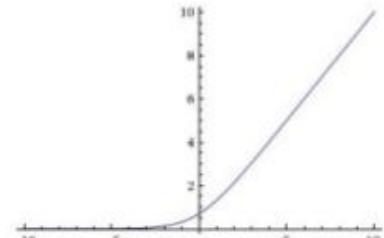
Функция активации - Parametric LU



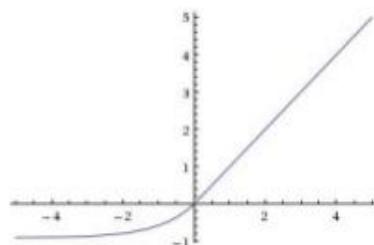
- угол наклона является обучаемым параметром
- угол наклона изменяется случайным образом

Функция активации - Exponential LU

Softplus and Exponential Linear Unit (ELU)



$$\text{softplus}(x) = \log(1 + e^x)$$



$$f(x) = \begin{cases} x & \text{if } x > 0 \\ \alpha (\exp(x) - 1) & \text{if } x \leq 0 \end{cases}$$

- решена неоднозначность с градиентом в области нуля

Функции активации ReLU

Activation	Training Error	Test Error
ReLU	0.1356	0.429
Leaky ReLU, $a = 100$	0.11552	0.4205
Leaky ReLU, $a = 5.5$	0.08536	0.4042
PReLU	0.0633	0.4163
RReLU	0.1141	0.4025

Table 4. Error rate of CIFAR-100 Network in Network with different activation function

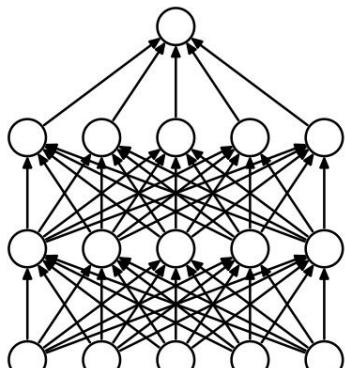
Обучение сверточной сети на практике. Регуляризация

Регуляризация

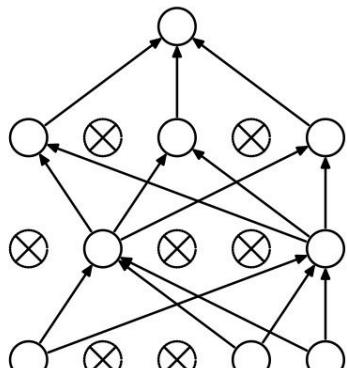
- L1, L2 - добавка в функцию потерь модуль, квадрат, при добавлении обоих - называется elastic net
- регуляризация заключается в добавлении соответствующего слагаемого в функцию потерь

<https://keras.io/regularizers/>

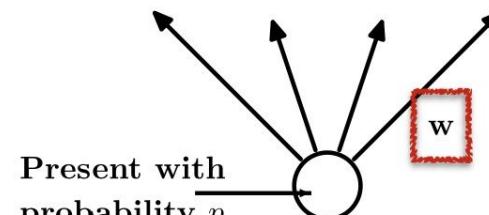
Регуляризация - Dropout



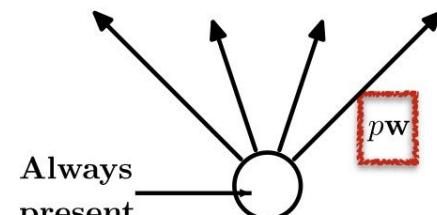
(a) Standard Neural Net



(b) After applying dropout.



(c) At training time



(d) At test time

Dropout

```
keras.layers.core.Dropout(rate, noise_shape=None, seed=None)
```

- **rate**: диапазон 0..1
- **noise_shape**: позволяет задать одинаковую маску по каналам
- **seed**: инициализация генератора случайных чисел

Обучение сверточной сети на практике.
Ансамбль нейронных сетей

Ансамбли нейросетей

- выбираем топ лучших моделей на кросс-валидации
- одна модель - разные инициализации (разные seed)
- сохранение весов с разных эпох
- дообучить предобученные на разных датасетах модели
- объединить результат линейной моделью

[How can I obtain reproducible results using Keras during development?](#)

Резюме

- качество различных нейросетевых архитектур сравнивают на открытых датасетах
- предобученные на открытых датасетах модели доступны для скачивания
- для решения практической задачи как правило адаптируют готовую архитектуру, предобученную на открытом датасете
- существует набор подходов, позволяющий повысить качество модели: аугментация, предобработка и нормализация входных данных, регуляризация модели, использование ReLU в качестве функции активации

Полезные материалы

- [AN ANALYSIS OF DEEP NEURAL NETWORK MODELS FOR PRACTICAL APPLICATIONS](#)
- [The 9 Deep Learning Papers You Need To Know About](#)
- [Must Know Tips/Tricks in Deep Neural Networks](#)
- [Practical Recommendations for Gradient-Based Training of Deep Architectures](#)
- [Building powerful image classification models using very little data](#)
- [An overview of gradient descent optimization algorithms](#)