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

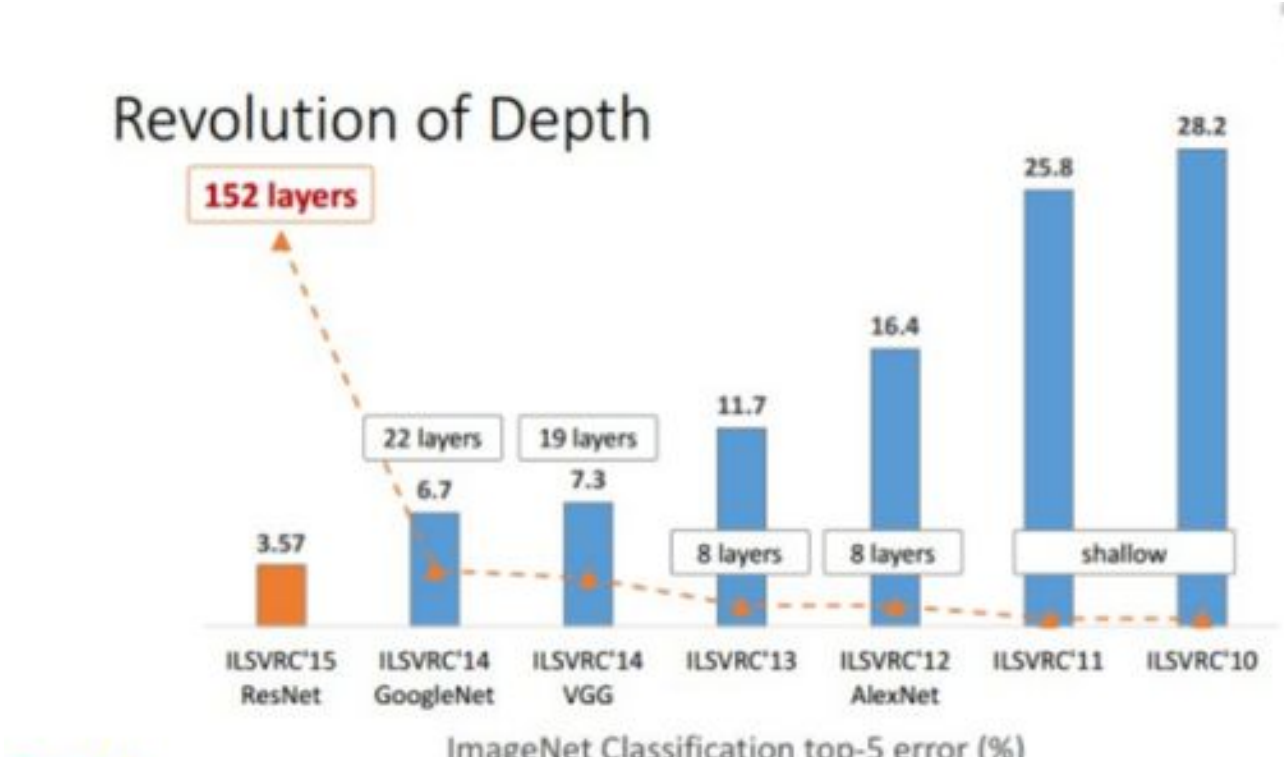
ImageNet

IMAGENET

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

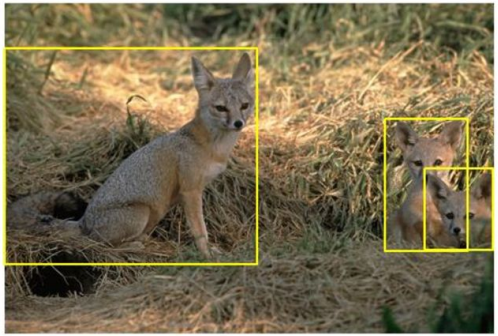
http://image-net.org

ImageNet

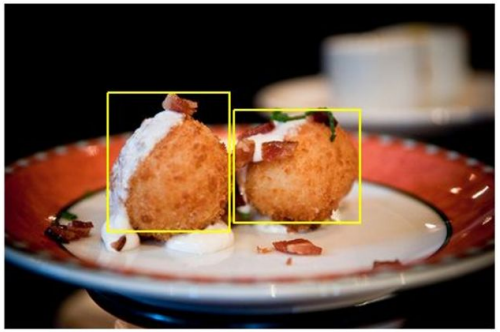

Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15 ResNet
ILSVRC'14 GoogleNet
ILSVRC'14 VGG
ILSVRC'13
ILSVRC'12 AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)

ImageNet

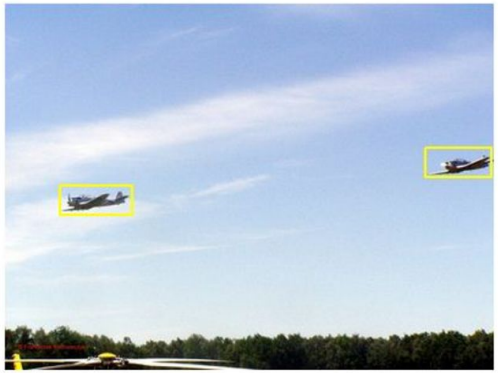
- разметка для 3000 категорий
- в среднем 150 изображений на категорию



kit fox



croquette



airplane

frog

ImageNet

- разметка 25 атрибутов для ~400 категорий
- в среднем 25 изображений на категорию

Pascal VOC

Pascal VOC (Visual Object Classes)

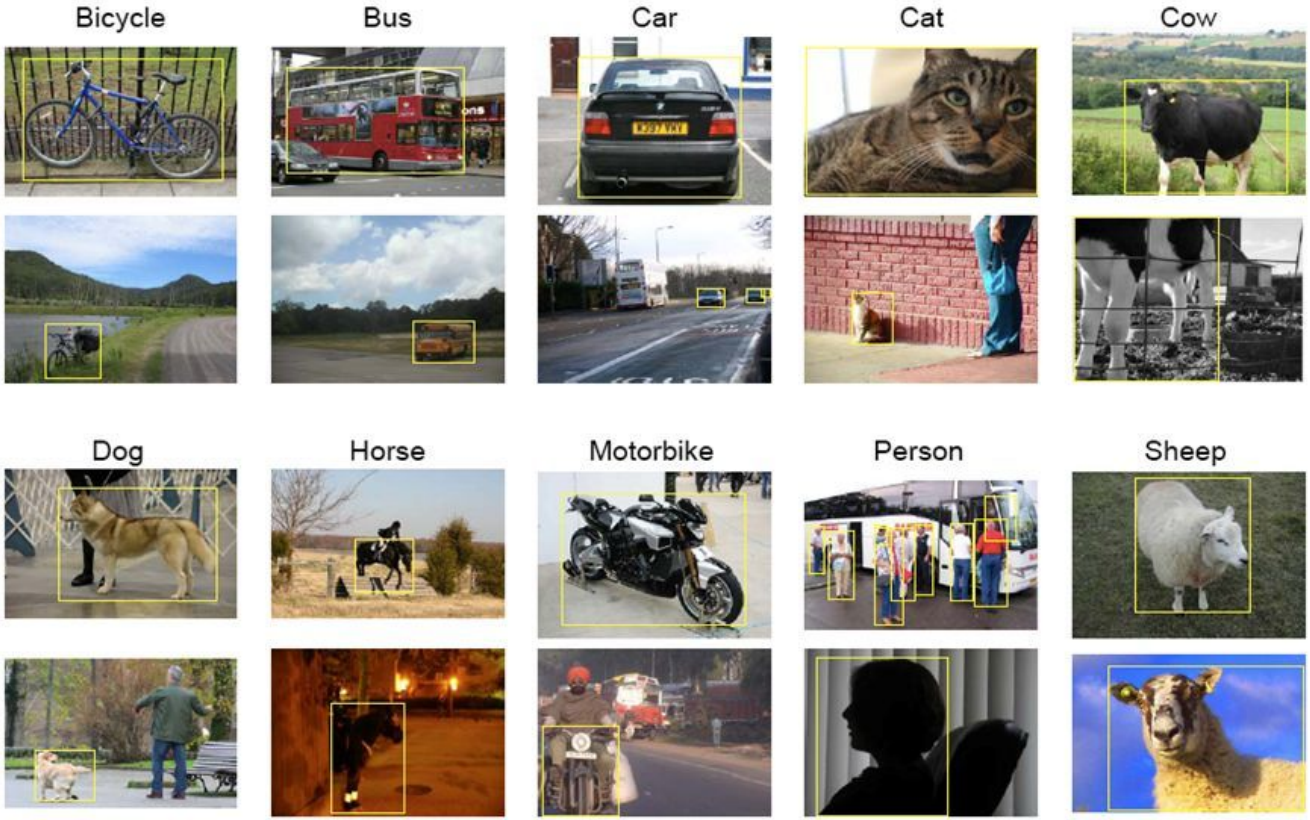
Visual Object Classes Challenge 2009 (VOC2009)

PASCAL2
Pattern Analysis, Statistical Modelling and Computational Learning

[click on an image to see the annotation]

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

Pascal VOC



Pascal VOC

Image | Class map | Instance map | Part map | Part map (high level)

COCO Common Object in Context

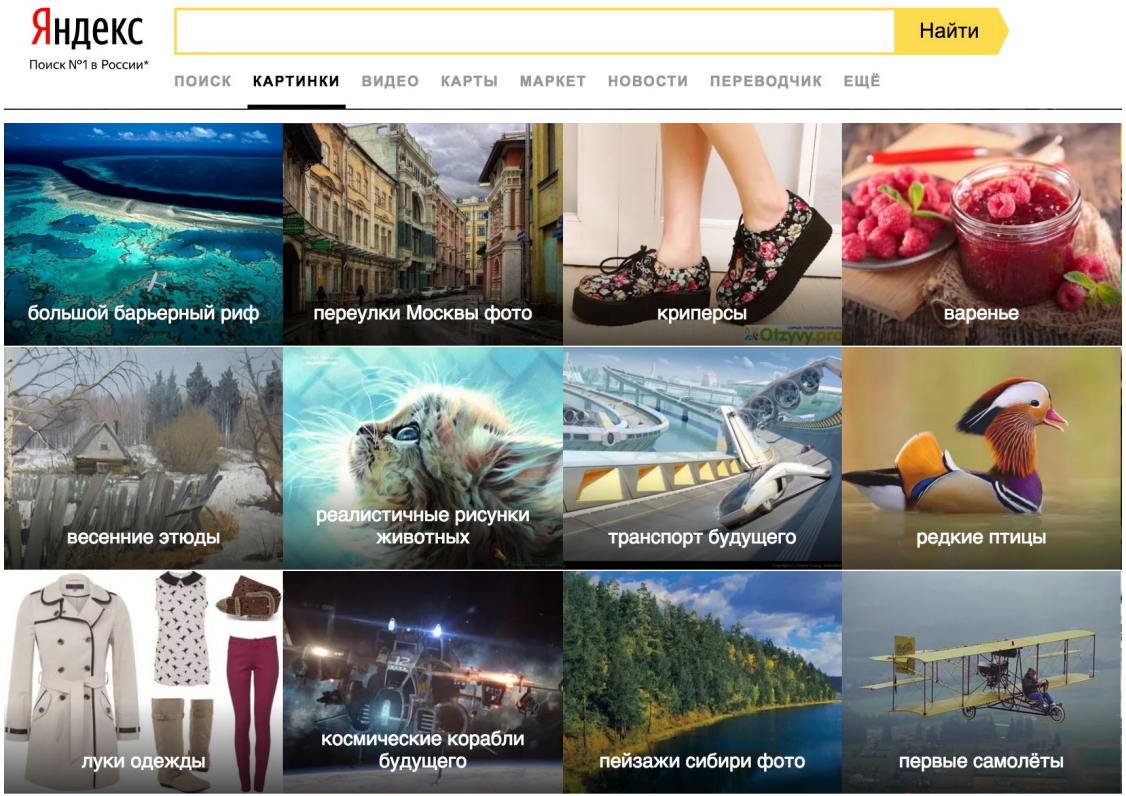
COCO Dataset

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

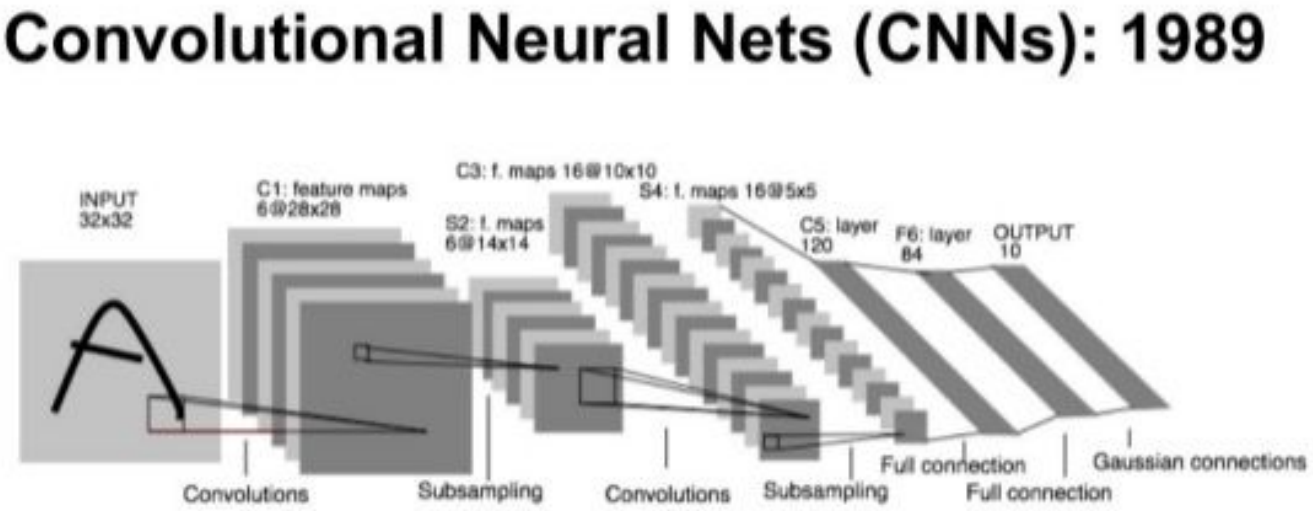
Данные из поисковых систем



Архитектуры сверточных сетей

LeNet - 1989

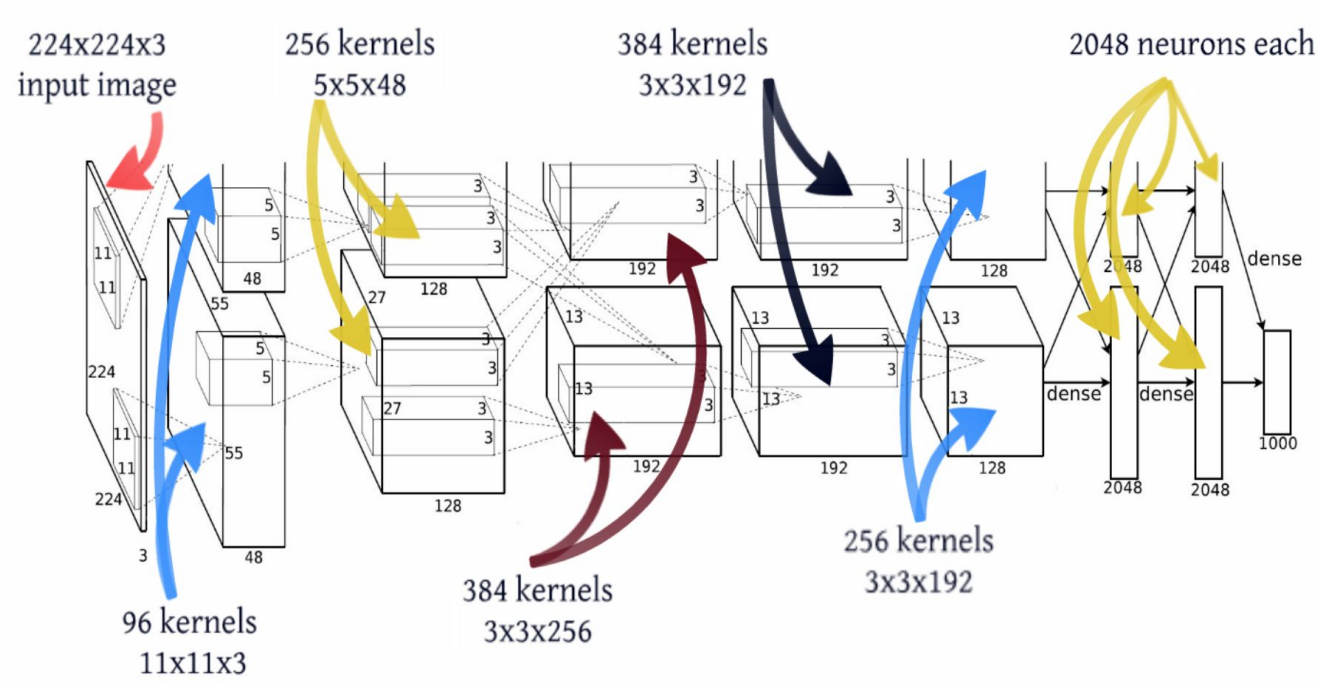
LeNet - 1989



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

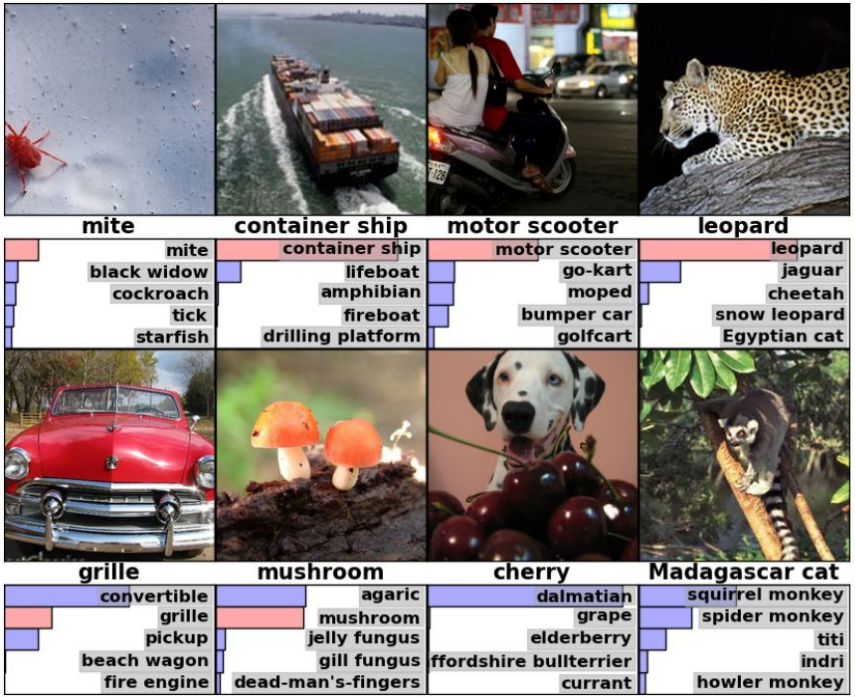
AlexNet - 2012

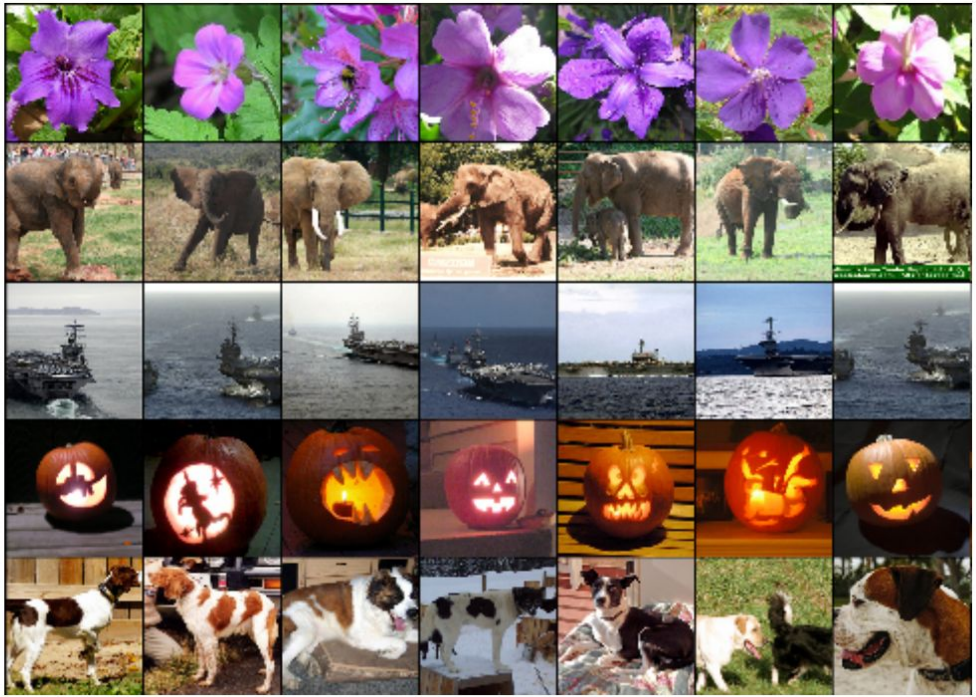
AlexNet - 2012



ImageNet Classification with Deep Convolutional Neural Networks

AlexNet - 2012



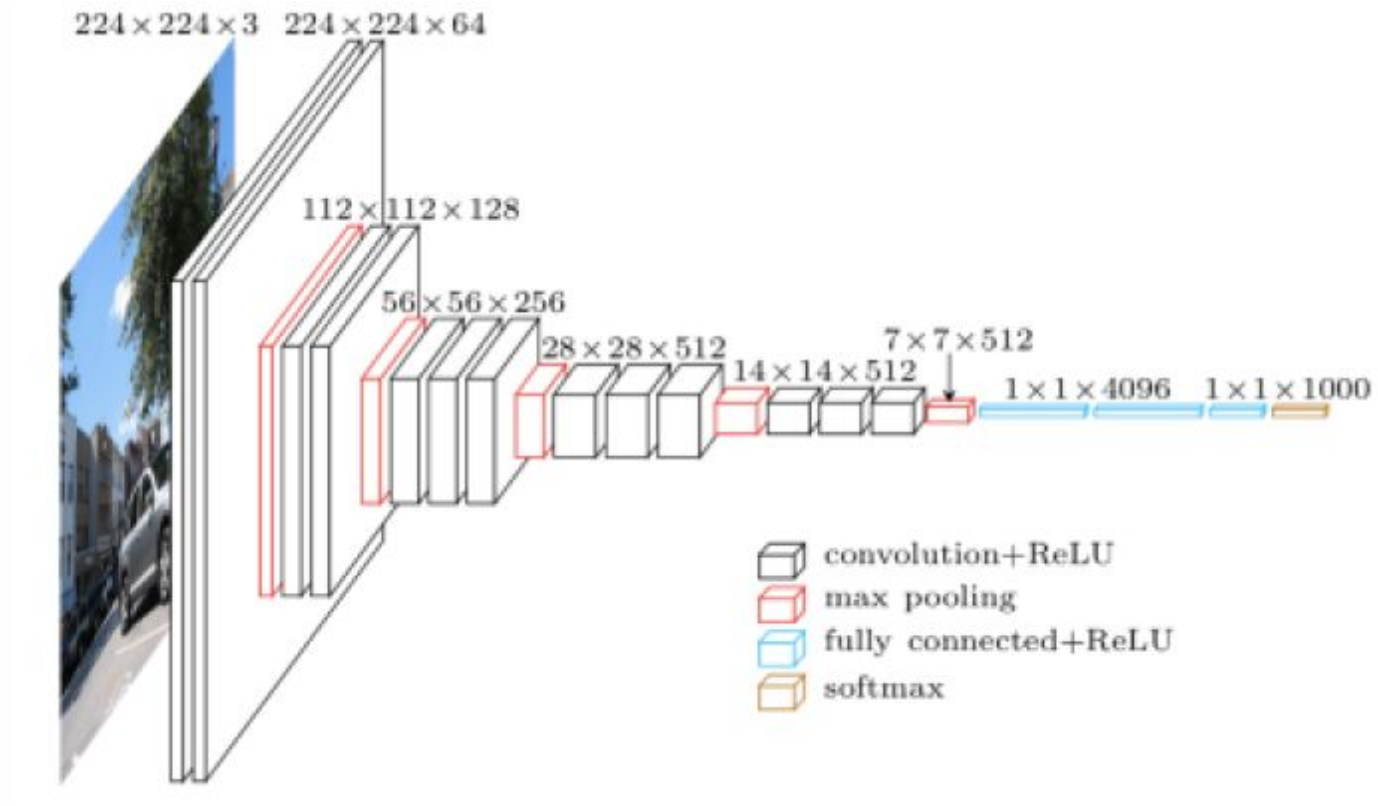


AlexNet - 2012

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

VGG - 2014

VGG - 2014

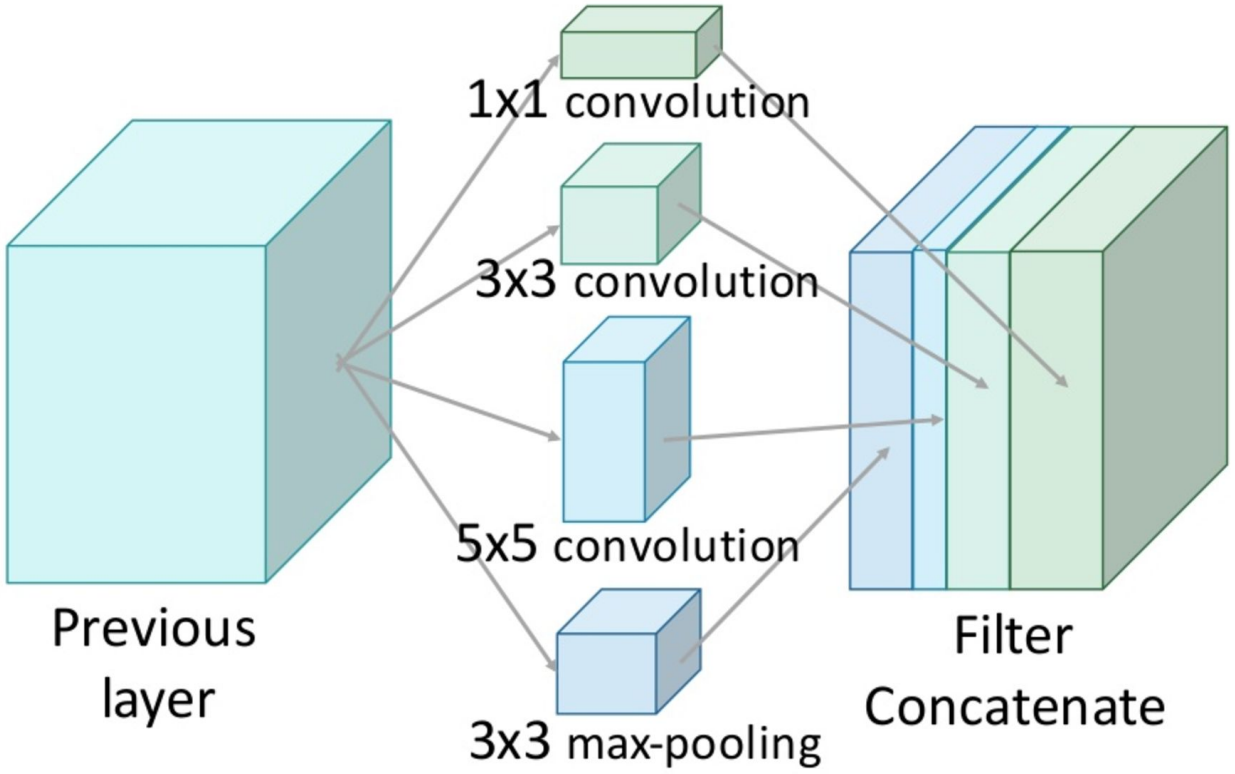

224 × 224 × 3
224 × 224 × 64
112 × 112 × 128
56 × 56 × 256
28 × 28 × 512
14 × 14 × 512
7 × 7 × 512
1 × 1 × 4096
1 × 1 × 1000
convolution+ReLU
max pooling
fully connected+ReLU
softmax

VGG - 2014

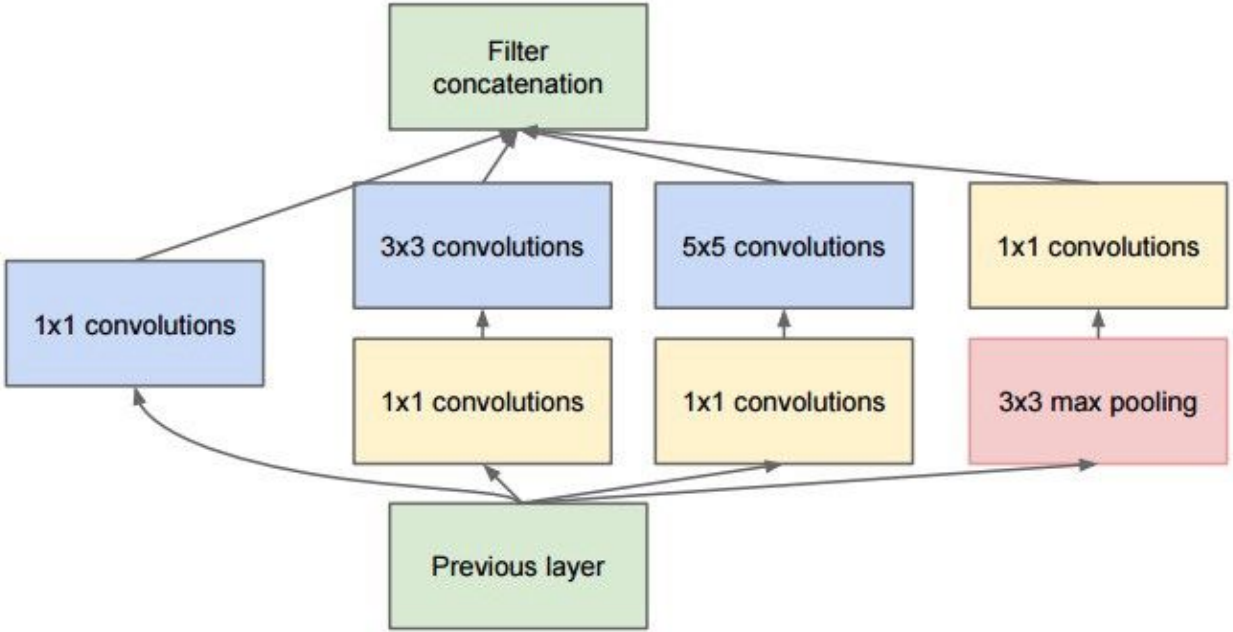
- последовательно применяются свертки фильтров с небольшим размером ядра
- большой объем данных на выходе каждого слоя требует большого количества памяти
- число параметров ~130M

GoogleNet (Inception) - 2014

GoogleNet (Inception) - 2014

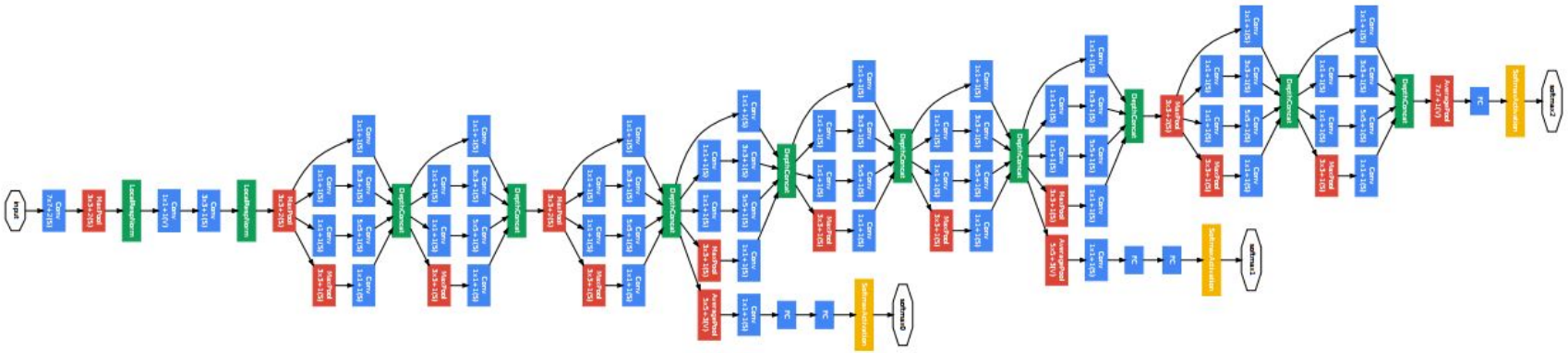


GoogleNet (Inception) - 2014



Going deeper with convolutions

GoogleNet (Inception) - 2014

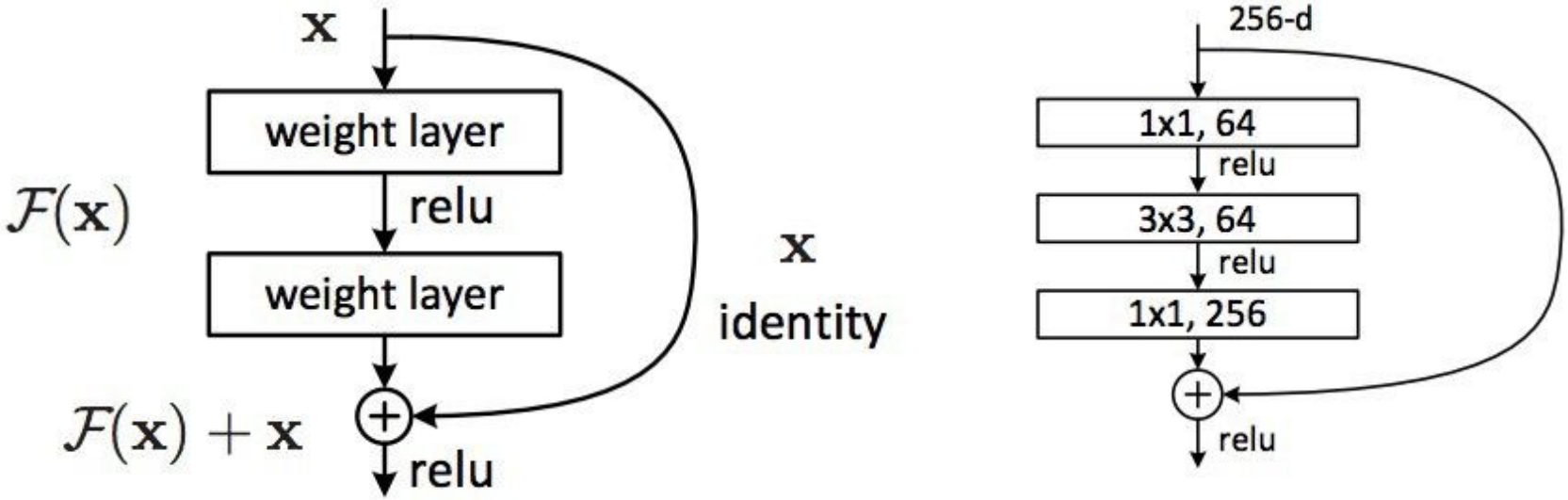


GoogleNet (Inception) - 2014

- применяются фильтры разного размера к одним и тем же данным
- уменьшение размерности выхода (глубины) за счет свертки 1x1
- в результате уменьшения глубины получаем ускорение сверток 3x3 и 5x5
- число параметров ~7M

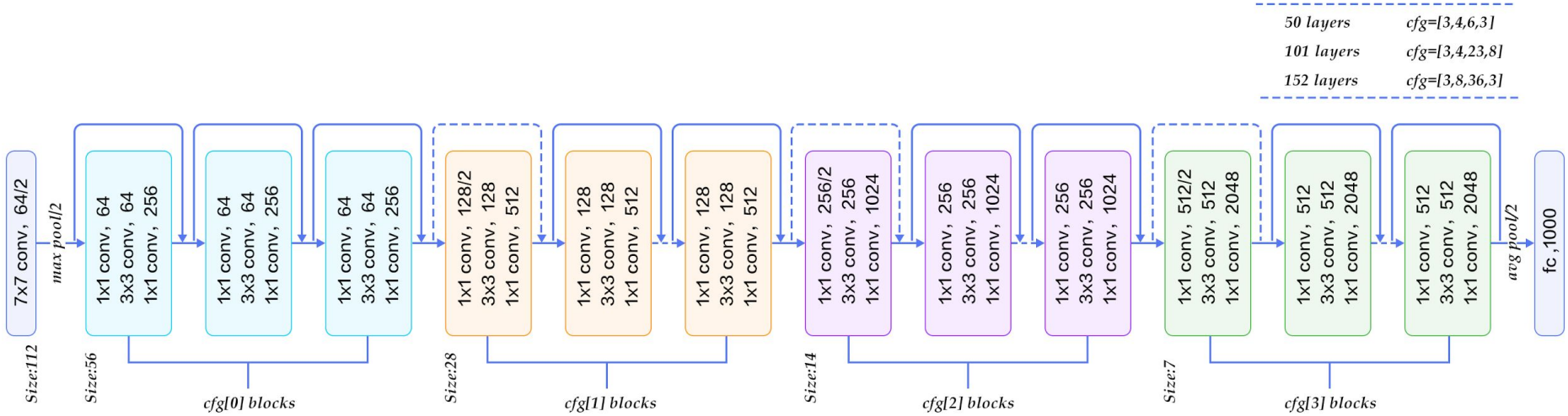
ResNet - 2015

ResNet - 2015



Deep Residual Learning for Image Recognition

ResNet - 2015

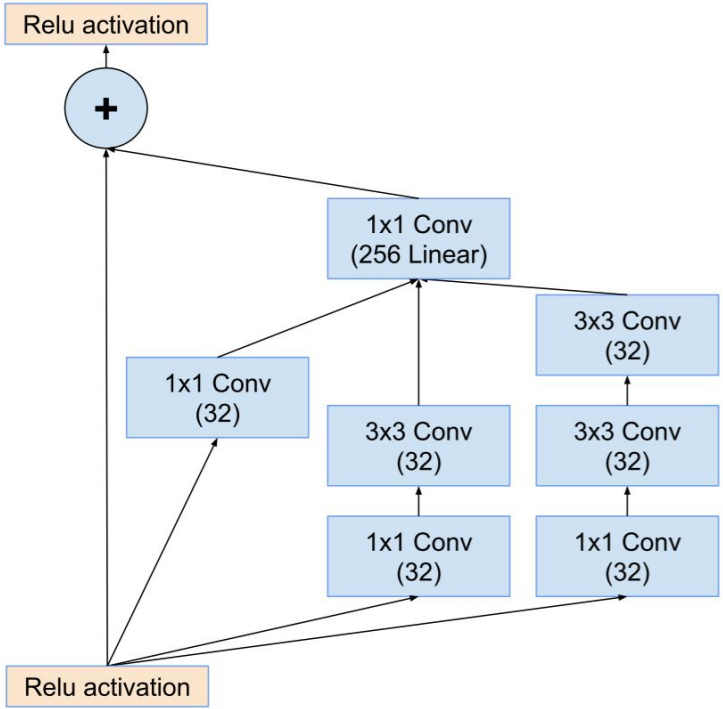

50 layers cfg=[3,4,6,3]
101 layers cfg=[3,4,23,8]
152 layers cfg=[3,8,36,3]
7x7 conv, 64/2
max pool/2
1x1 conv, 64
3x3 conv, 64
1x1 conv, 256
1x1 conv, 128/2
3x3 conv, 128
1x1 conv, 512
1x1 conv, 256/2
3x3 conv, 256
1x1 conv, 1024
1x1 conv, 512/2
3x3 conv, 512
1x1 conv, 2048
avg pool/2
fc ,1000
Size:112
Size:56
Size:28
Size:14
Size:7
cfg[0] blocks
cfg[1] blocks
cfg[2] blocks
cfg[3] blocks

ResNet - 2015

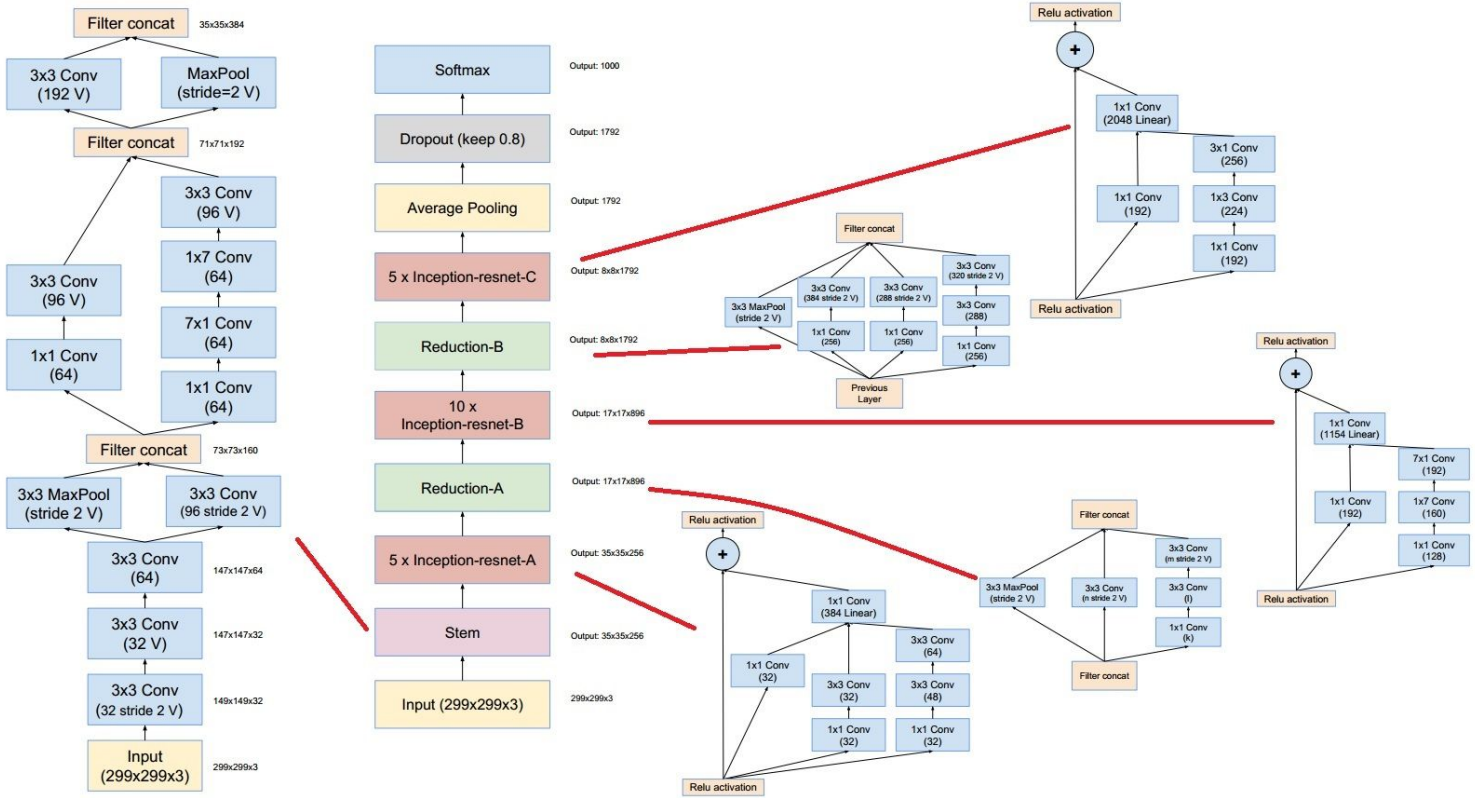
- добавлена параллельная ветка с исходными данными
- первая архитектура, у которой более 100 слоев
- число параметров ResNet 50 ~25M

Inception ResNet

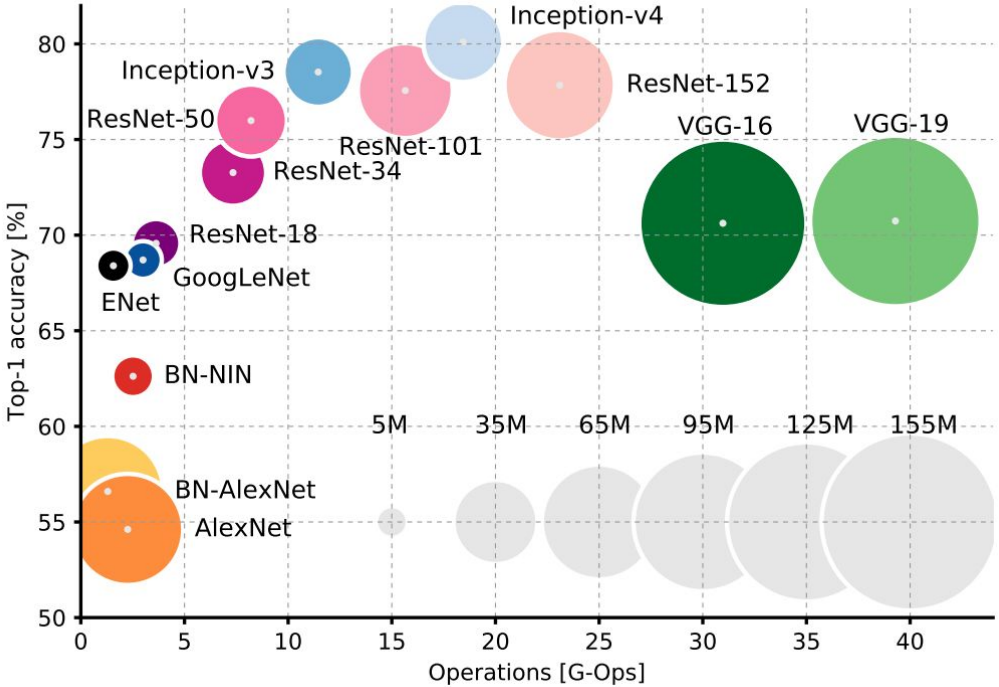
Inception ResNet



Inception ResNet



Сравнение: качество vs скорость



AN ANALYSIS OF DEEP NEURAL NETWORK MODELS FOR PRACTICAL APPLICATIONS

Предобученные модели

- CAFFE Model-Zoo
- Keras Applications
- Deep Learning Model Convertors

Обучение сверточной сети на практике. Transfer Learning

Transfer Learning

- на практике очень редко обучают 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. Разметка данных

Обучение с частичной разметкой

- на практике часто встречаются задачи с неполной разметкой данных
- обучаем последнии слои на доступной разметке
- размечаем датасет полученной моделью
- дообучаем на всех данных
- важно контролировать процесс на корректно размеченной валидационной выборке

Обучение сверточной сети на практике. Баланс классов

Баланс классов

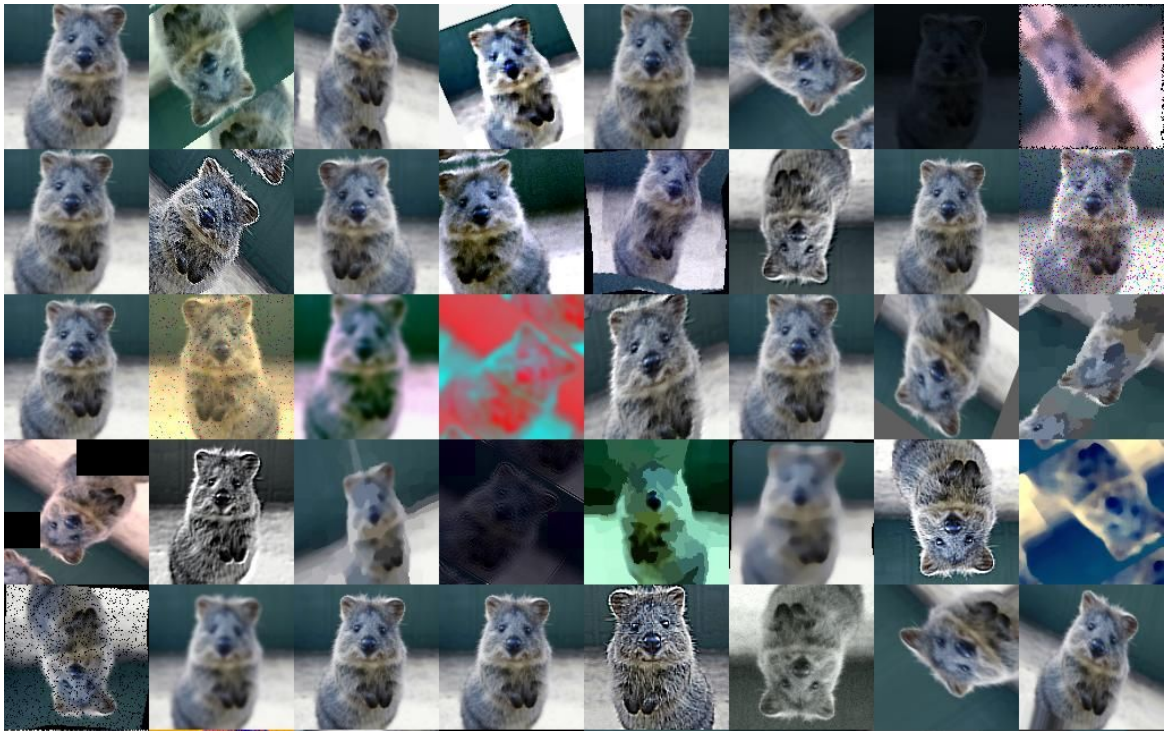
- часто на практике выборка не сбалансирована
- это приводит к тому, что модель переобучается на классы с большим числом примеров и дает смещенное предсказание

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 (crop) и масштабируем до исходного
- аугментация освещенности (в пространстве HSV)
- аугментация цвета (случаный шум по каналам)

Реализация

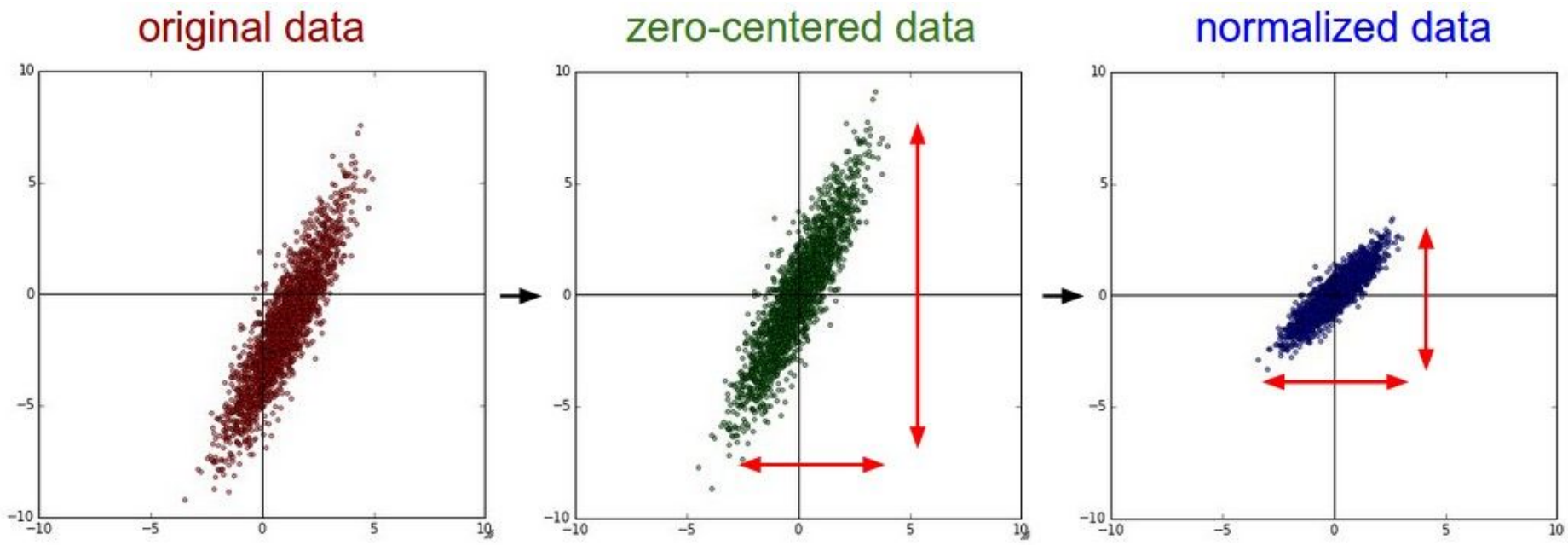
- на практике аугментированные изображения сильно увеличивают размер выборки
- хранить на диске аугментиованные копии нецелесообразно
- процесс аугментации запускают на лету параллельно с обучением
- аугментация выполняется на CPU и существенно не влияет на скорость при обучении на GPU

Библиотеки аугментации изображений

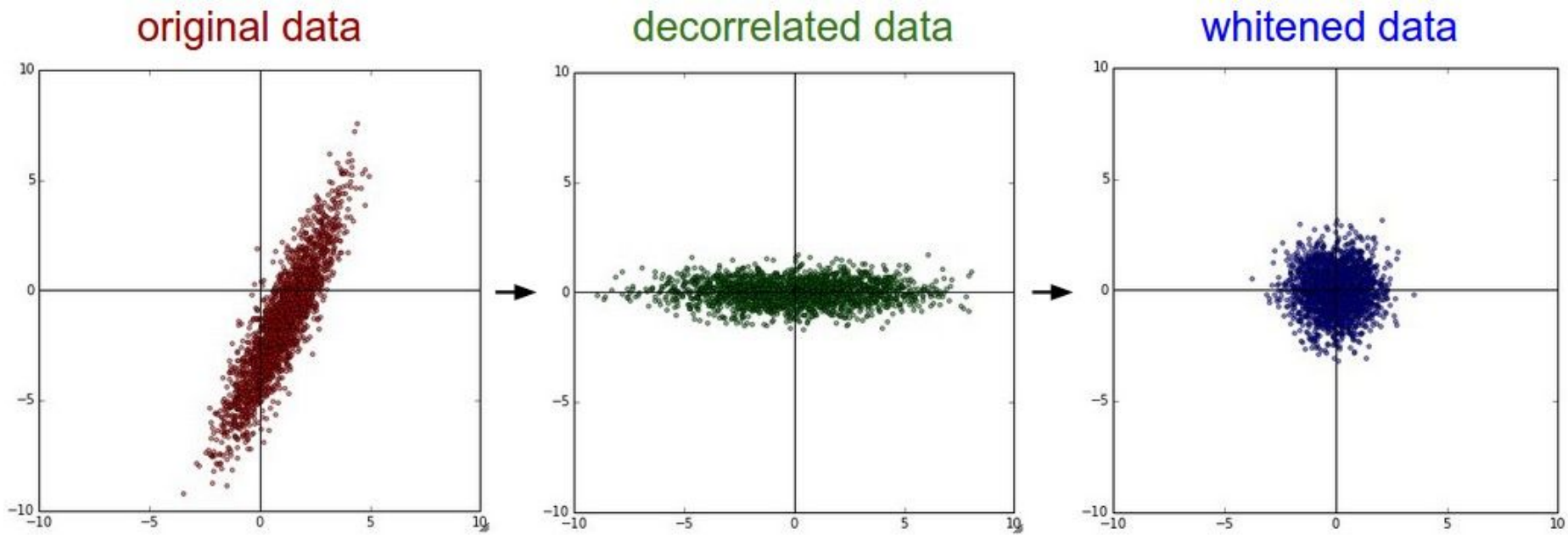
- aleju/imgaug
- keras.preprocessing.image.ImageDataGenerator
- tf.image

Обучение сверточной сети на практике. Предобработка данных

Центрирование и нормализация

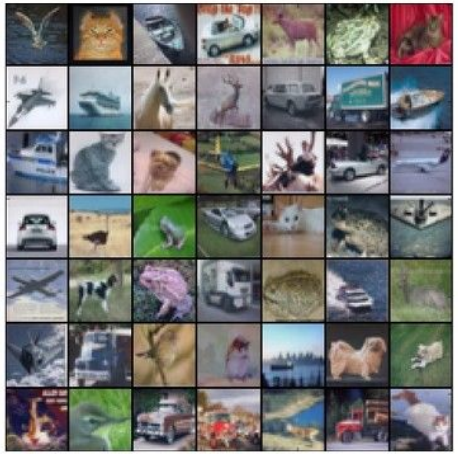


Обеление (PCA whitening)

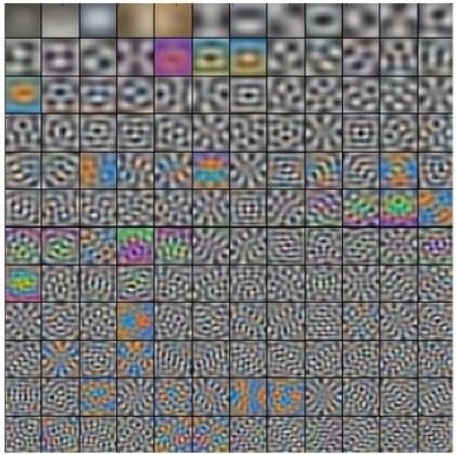


Обеление (PCA whitening)

original images

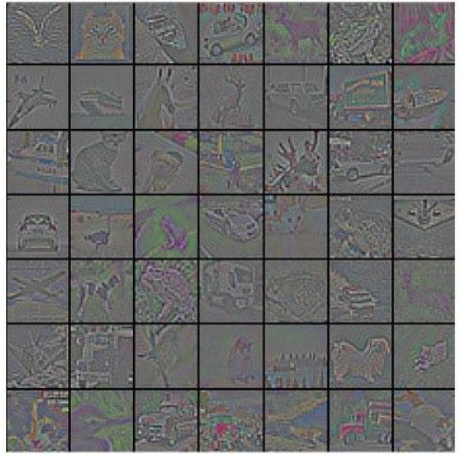


top 144 eigenvectors



reduced images

whitened images



Локальная нормализация (AlexNet)

$$b^i_{x,y} = a^i_{x,y} / (k + \alpha \sum_{j=max(0,i-n/2)}^{j=min(N-1,i+n/2)} {a^j_{x,y}}^2)^\beta$$

where

$b^i_{x,y}$ – regularized output for kernel i at position x, y

$a^i_{x,y}$ – source ouput of kernel i applied at position x, y

N – total number of kernels

n – size of the normalization neigbourhood

$\alpha, \beta, k, (n)$ – hyperparameters

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

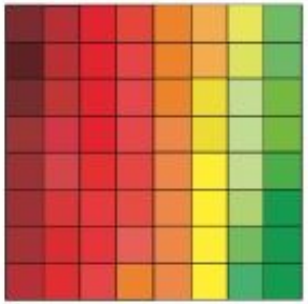
- размеры изображения кратны числу семплирующих слоев (Pooling)
- чем меньше размер фильтра, тем меньше операций необходимо выполнить для вычисления свертки
- падинги необходимо добавлять при свертке для сохранения информации на краях картинки (актуально с увеличением глубины)

Тензор

vector

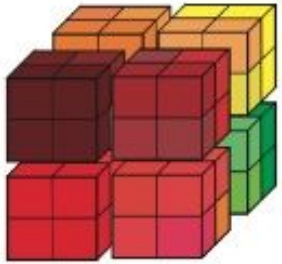
$\mathbf{v} \in \mathbb{R}^{64}$

matrix



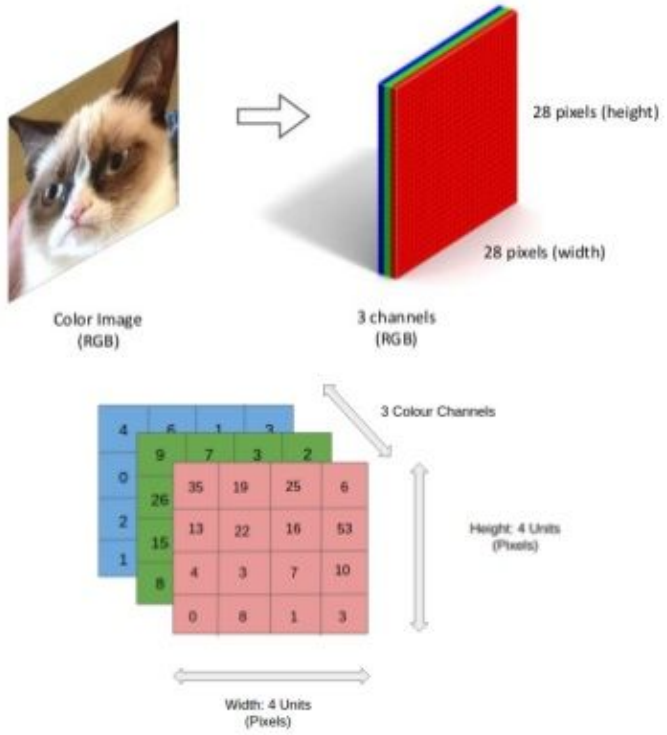
$X \in \mathbb{R}^{8x8}$

tensor



$\mathcal{X} \in \mathbb{R}^{4x4x4}$

RGB Тензор


28 pixels (height)
28 pixels (width)
Color Image
(RGB)
3 channels
(RGB)
3 Colour Channels
Height: 4 Units
(Pixels)
Width: 4 Units
(Pixels)

Формат данных и производительность

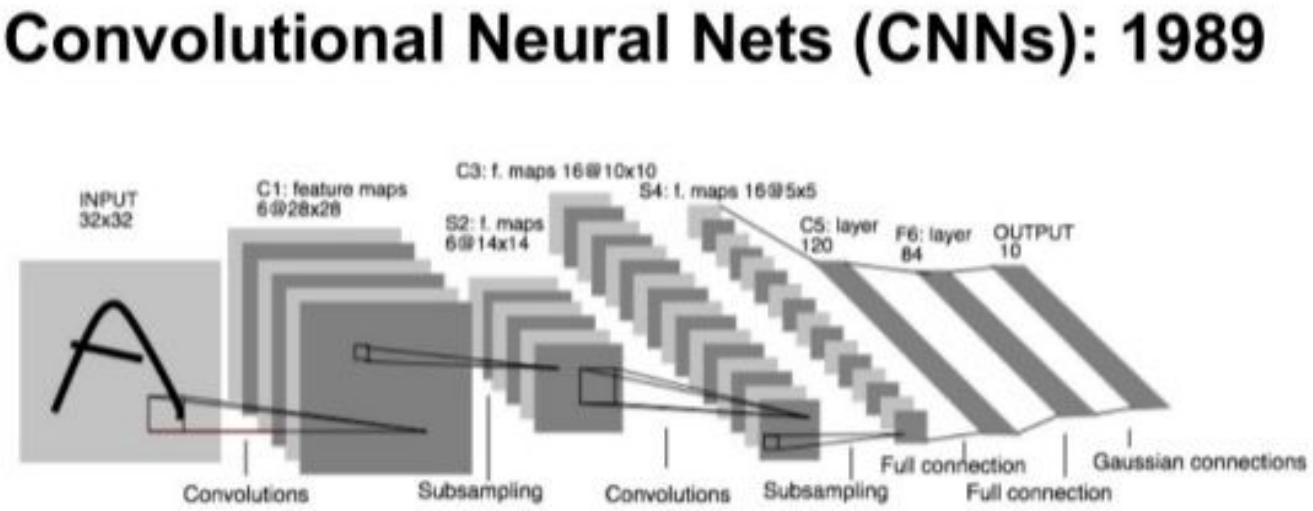
- производительность зависит от порядка измерений
- изменение порядка с NHWC (channel_last) на NCHW (channel_first) дает прирост в скорости вычислений на GPU до 10%
- это связано с особенностью библиотеки cuDNN, которая поддерживает только NCHW формат
- если вы используете NHWC Tensorflow автоматически 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

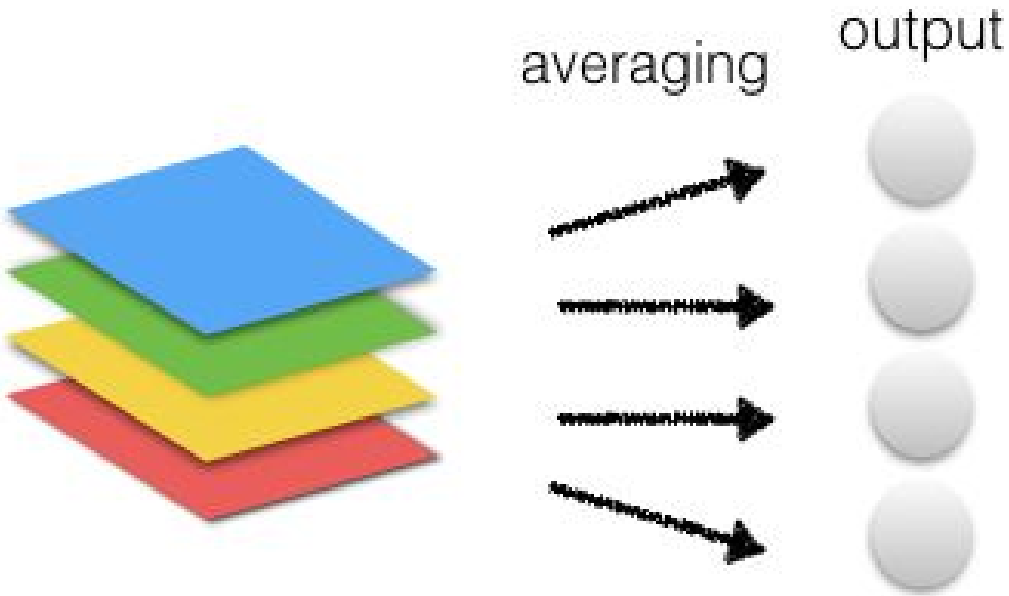
- сверточные сети с полносвязными слоями имеют ограничение на размер входного изображения
- один из способов снять это ограничение - добавить слой Global Average Pooling перед FC слоем

Размер исходного изображения и FC слой



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

Global Average Pooling



Обучение сверточной сети на практике. Градиентный спуск

Градиентный спуск

- при увеличении размера батча, необходимо уменьшить learning rate
- в случае, если не происходит изменение метрики на валидации, стоит уменьшить значение learning rate
- наиболее популярные оптимизаторы: adam, adadelta

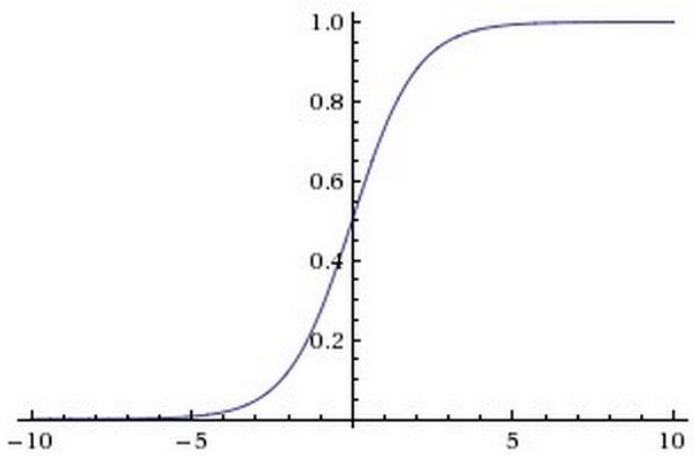
https://keras.io/optimizers/

Обучение сверточной сети на практике. Функции активации

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

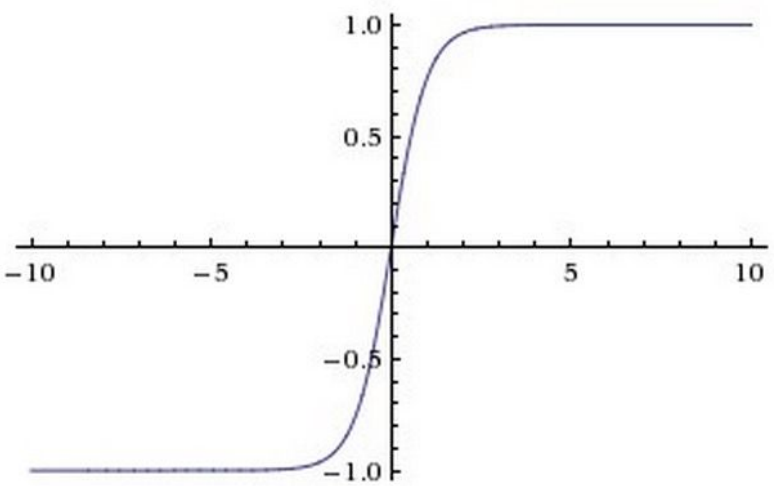
Функции активации - sigmoid

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



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

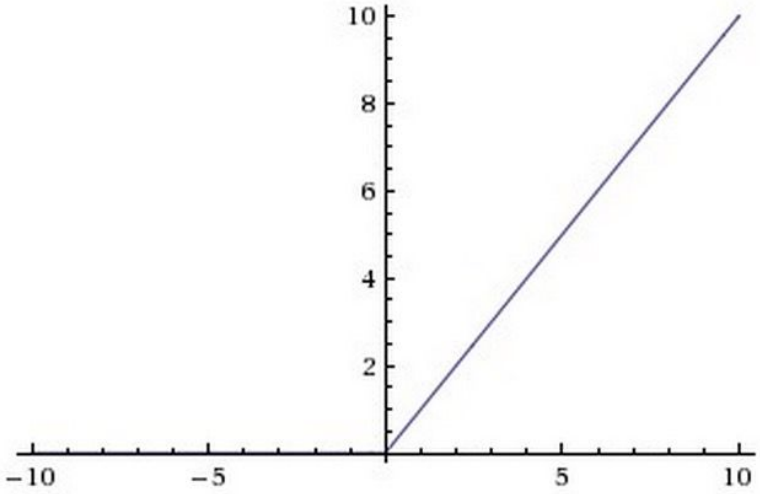
Функция активации - tanh



- остается проблема с градиентами
- решена проблема с центрированием

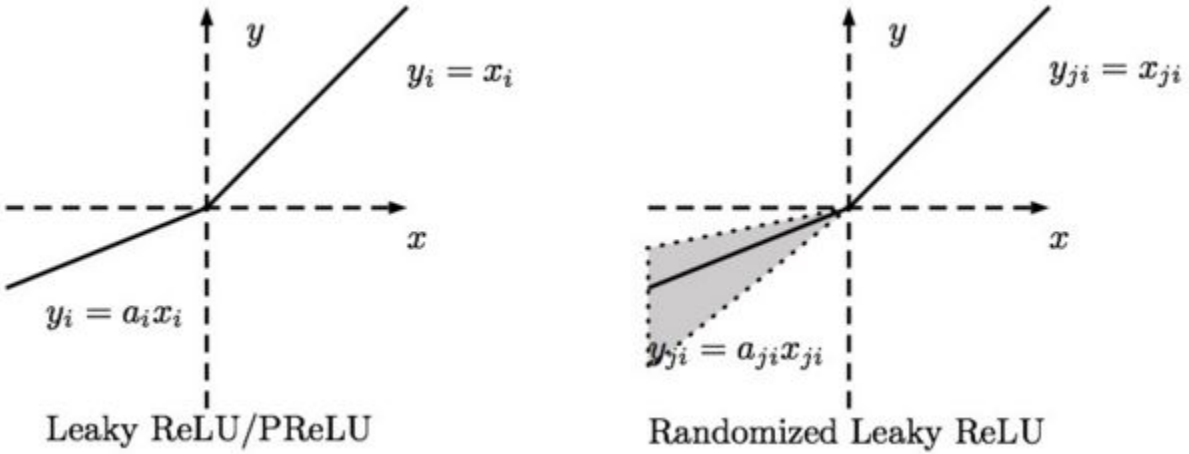
Функция активации - ReLU

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



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

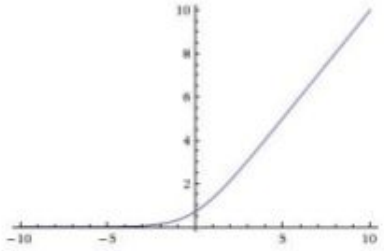
Функция активации - Parametric LU



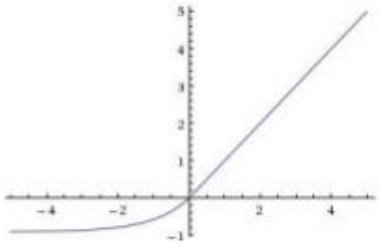
- угол наклона является обучаемым параметром
- угол наклона изменяется случайным образом

Функция активации - Exponential LU

Softplus and Exponential Linear Unit (ELU)



$$softplus(x) = log(1 + e^x)$$



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

- 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

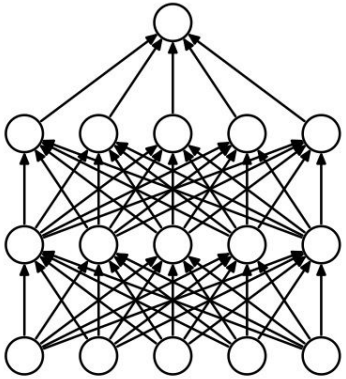
Обучение сверточной сети на практике. Регуляризация

Регуляризация

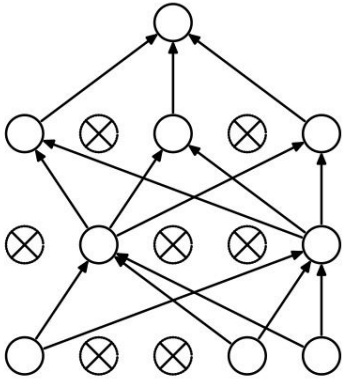
- L1, L2 - добавка в функцию потерь модуль, квадрат, при добавлении обоих - называется elastic net
- регуляризация заключается в добавлении соответствующего слагаемого в функцию потерь

https://keras.io/regularizers/

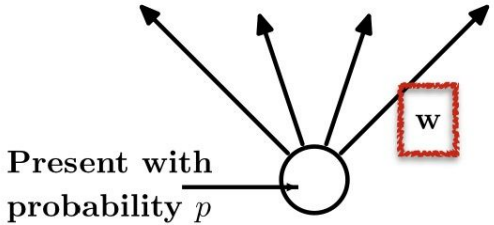
Регуляризация - Dropout



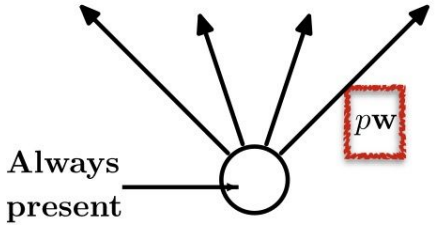
(a) Standard Neural Net



(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 нормалицзация 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