



Национальный исследовательский  
Нижегородский государственный университет им. Н.И. Лобачевского

# **Библиотека OpenCV. Базовые возможности модулей core, highgui, imgproc, dnn**

Кустикова В.Д.,  
к.т.н., старший преподаватель  
кафедры МОСТ ИИТММ

# Содержание

---

- ❑ Краткая справочная информация
- ❑ С чего начать изучение?
- ❑ Архитектура и разработка OpenCV
- ❑ Общая схема типичного приложения компьютерного зрения
- ❑ Модуль core. Хранение изображений
- ❑ Модуль highgui. Чтение/запись/отображение изображений
- ❑ Модуль imgproc. Базовые операции обработки изображений
- ❑ Модуль dnn. Состав и основные возможности



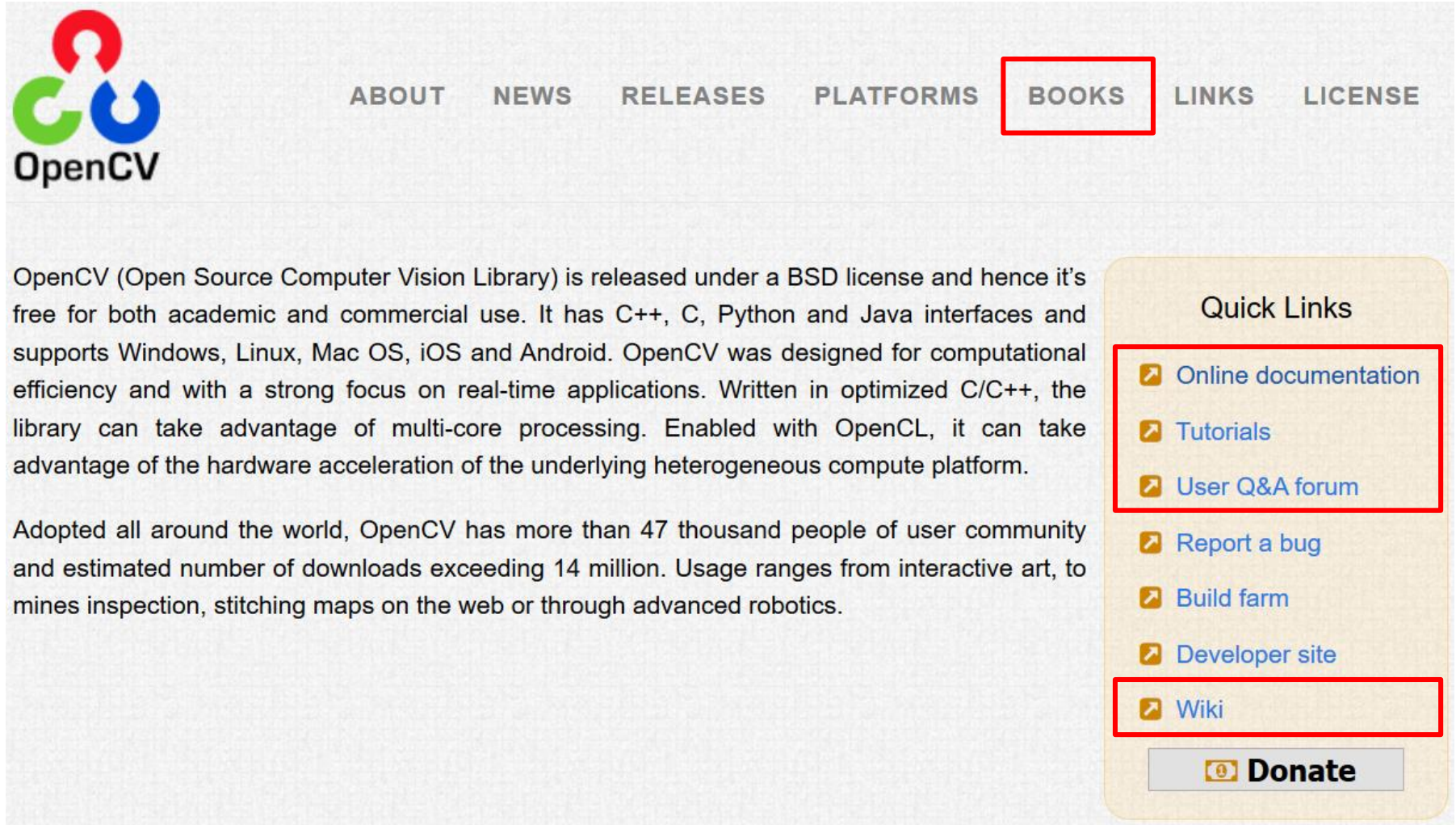
# Краткая справочная информация

---

- ❑ Официальная страница: <http://opencv.org>
- ❑ Библиотека с открытым исходным кодом
- ❑ Распространяется под лицензией BSD
- ❑ Разрабатывается с 1998 г. (Intel, Itseez при активном участии сообщества)
- ❑ Используется многими компаниями, организациями, ВУЗами (NVIDIA, Willow Garage, Intel, Google, Stanford и другие)



# С чего начать изучение?



The screenshot shows the OpenCV website homepage. The OpenCV logo is in the top left. A navigation bar at the top contains links: ABOUT, NEWS, RELEASES, PLATFORMS, BOOKS, LINKS, and LICENSE. The 'BOOKS' link is highlighted with a red rectangle. Below the navigation bar, a paragraph describes OpenCV as an open-source computer vision library released under a BSD license, supporting various operating systems and programming languages. To the right, a 'Quick Links' sidebar is shown, containing several links: Online documentation, Tutorials, User Q&A forum, Report a bug, Build farm, Developer site, and Wiki. The first three links in this sidebar are grouped by a red rectangle. At the bottom of the sidebar is a 'Donate' button.

**OpenCV**

ABOUT NEWS RELEASES PLATFORMS **BOOKS** LINKS LICENSE

OpenCV (Open Source Computer Vision Library) is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform.

Adopted all around the world, OpenCV has more than 47 thousand people of user community and estimated number of downloads exceeding 14 million. Usage ranges from interactive art, to mines inspection, stitching maps on the web or through advanced robotics.

**Quick Links**

- Online documentation
- Tutorials
- User Q&A forum
- Report a bug
- Build farm
- Developer site
- Wiki

**Donate**

# Архитектура и разработка OpenCV

## Языки

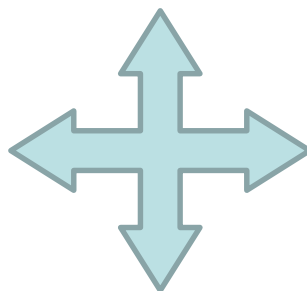
C/C++  
Python  
CUDA  
Java

## Сторонние библиотеки

Eigen  
IPP  
Jasper  
JPEG, PNG  
OpenNI  
Qt  
TBB  
VideoInput

## Разработка

Maintainers  
Contributors



## Контроль качества

Buildbot  
Google Tests

## Модули

**core**  
**imgproc**  
**highgui**  
gpu  
ml  
**dnn**  
objdetect  
video  
calib3d  
features2d  
flann

## Архитектуры

x86  
x64  
ARM  
CUDA

## ОС

Windows  
Linux  
Mac OS  
Android

## Технологии

CUDA  
SSE  
TBB



# Общая схема приложения компьютерного зрения



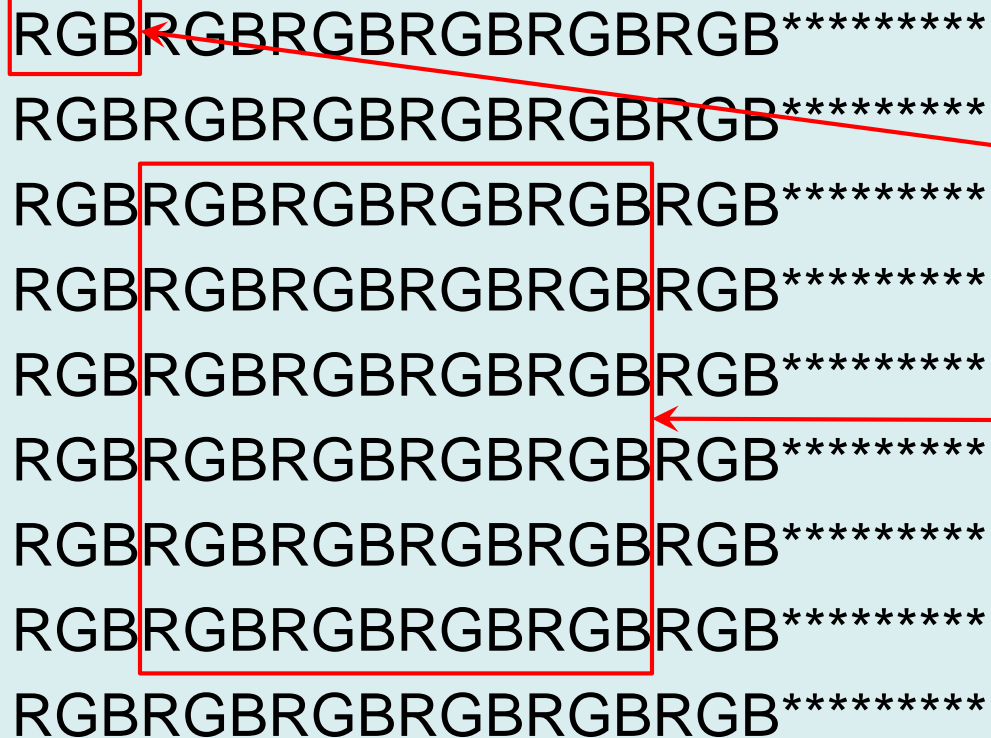


Библиотека OpenCV

# МОДУЛЬ CORE



# Модуль core. Хранение изображений (1)



```
RGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGB*****
RGBRGBRGBRGBRGBRGB*****
```

**cv::Mat**

Элемент изображения – пиксель

Область интереса

## Расположение в памяти

```
RGBRGBRGBRGBRGBRGB*****RGBRGBRGBRGBRGBRGB*****...
```





# Модуль core. Хранение изображений (2)

- ❑ `cv::Mat` – многомерный многоканальный массив (матрица)
- ❑ Создание:
  - `cv::Mat A(h, w, type)`, `h` – высота, `w` – ширина, `type` – тип элементов (общий вид названия типа `CV_<битность><знаковость><канальность>`, например, `CV_8UC3`)
  - `cv::Mat B = A`
  - `cv::Mat C = A(roi)`, где `roi` – прямоугольная область интереса `cv::Rect`
- ❑ Структура хранения:
  - Размер, шаг выравнивания
  - Счетчик ссылок
  - Указатель на данные



# Модуль core. Хранение изображений (3)

---

- ❑ `cv::Mat` – многомерный многоканальный массив (матрица)
- ❑ Каналы изображения в формате RGB загружаются и хранятся в порядке BGR





Библиотека OpenCV

# МОДУЛЬ HIGHGUI



# Модуль highgui. Чтение/запись изображений

```
Mat imread(const string& filename, int flags=1)
```

- ❑ Чтение изображения `filename`

```
bool imwrite(const string& filename,  
             InputArray img,  
             const vector<int>& params=vector<int>())
```

- ❑ Запись изображения `img` в файл `filename` с параметрами качества `params`
- ❑ `InputArray = const cv::Mat&`
- ❑ `OutputArray = cv::Mat&`



# Модуль highgui. Отображение изображений

```
void namedWindow(const string& winname,  
                 int flags=WINDOW_AUTOSIZE)
```

- ❑ Создание окна с названием `winname`.

```
void imshow(const string& winname,  
            InputArray mat)
```

- ❑ Отображение изображения `mat` в окне с названием `winname`.

```
int waitKey(int delay=0)
```

- ❑ Ожидание ввода символа в течение времени `delay`.



# Модуль highgui. Отображение геометрических примитивов и текста

---

```
void rectangle(Mat& img, Point pt1, Point pt2,  
    const Scalar& color, int thickness=1,  
    int lineType=8, int shift=0)  
void rectangle(Mat& img, Rect rec,  
    const Scalar& color, int thickness=1,  
    int lineType=8, int shift=0)  
void putText(Mat& img, const string& text,  
    Point org, int fontFace, double fontScale,  
    Scalar color, int thickness=1,  
    int lineType=8,  
    bool bottomLeftOrigin=false)
```





Библиотека OpenCV

# МОДУЛЬ IMGPROC



# Модуль `imgproc`. Базовые операции обработки изображений

---

- ❑ **Линейные фильтры**
- ❑ **Морфологические операции**
  
- ❑ Операторы Собеля (Sobel) и Лапласа (Laplace)
- ❑ Детектор ребер Канни (Canny)
- ❑ Оператор Шарра
  
- ❑ Построение Гауссовой пирамиды изображений
  
- ❑ Методы вычисления гистограмм
- ❑ Методы повышения контраста изображения
- ❑ ...





# Свертка и линейные фильтры

- ❑  $I$  – полутоновое изображение.
- ❑ Линейный фильтр определяется вещественнозначной функцией  $F$ , заданной на растре. Данная функция называется **ядром фильтра**, а операция фильтрации выполняется посредством вычисления **дискретной свертки**:

$$I'(x, y) = \sum_i \sum_j F(i, j) \cdot I(x + i, y + j)$$

- ❑ Окрестность называется **шаблоном** или **апертурой**.
- ❑ Шаблон накладывается на каждый текущий пиксель посредством совмещения пикселя с конкретной точкой шаблона – **ведущей позицией шаблона**.



# Свертка и линейные фильтры. Проблемы

- ❑ Текущий пиксель находится на границе изображения?
- ❑ Возможные решения:
  - Обрезать края.
  - Не учитывать в процессе суммирования пиксель, который реально не существует.
  - Доопределить окрестности граничных пикселей посредством экстраполяции (например, простым дублированием граничных пикселей).
  - Доопределить окрестности граничных пикселей посредством зеркального отражения – завернуть изображение в тор.



# Линейные фильтры. Функции OpenCV

```
void filter2D(InputArray src, OutputArray dst,  
             int ddepth, InputArray kernel,  
             Point anchor=Point(-1, -1),  
             double delta=0,  
             int borderType=BORDER_DEFAULT)
```

- ❑ Новое значение интенсивности пикселя вычисляется по формуле:

$$dst(x, y) = \sum_{\substack{0 \leq x' < anchor.x \\ 0 \leq y' < anchor.y}} kernel(x', y') \cdot src(x + x' - anchor.x, y + y' - anchor.y)$$

- ❑ В случае многоканального изображения ядро применяется к каждому каналу в отдельности.



# Сглаживание изображений

---

- ❑ **Сглаживание** – свертка с ядрами специального вида.
- ❑ Сглаживание играет важную роль при необходимости уменьшить разрешение изображения и получить пирамиду изображений разного масштаба (image pyramids).



# Сглаживание изображений. Функции OpenCV

```
void blur(InputArray src, OutputArray dst,  
         Size ksize, Point anchor=Point(-1, -1),  
         int borderType=BORDER_DEFAULT);  
void boxFilter(InputArray src, OutputArray dst,  
              int ddepth, Size ksize,  
              Point anchor=Point(-1, -1),  
              bool normalize=true,  
              int borderType=BORDER_DEFAULT);  
void GaussianBlur(InputArray src,  
                 OutputArray dst, Size ksize,  
                 double sigmaX, double sigmaY=0,  
                 int borderType=BORDER_DEFAULT);
```



# Морфологические преобразования. Дилатация

- ❑ **Дилатация** (морфологическое расширение) – «свертка» изображения или выделенной области изображения с некоторым ядром.
- ❑ Ядро может иметь произвольную форму и размер (во квадрат или круг).
- ❑ При этом в ядре выделяется единственная **ведущая позиция** (anchor), которая совмещается с текущим пикселем при вычислении свертки.
- ❑ Проход шаблоном по изображению и применение **оператора локального максимума к интенсивностям** пикселей изображения, которые накрываются шаблоном => **рост светлых областей**.

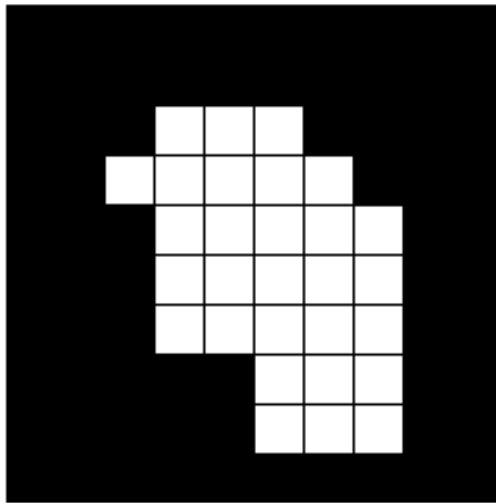


# Морфологические преобразования. Эрозия

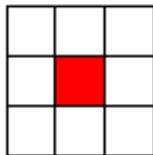
- ❑ **Эрозия** (морфологическое сужение) – операция, обратная к дилатации.
- ❑ Действие эрозии подобно дилатации, разница лишь в том, что используется **оператор поиска локального минимума** => **рост темных областей**.



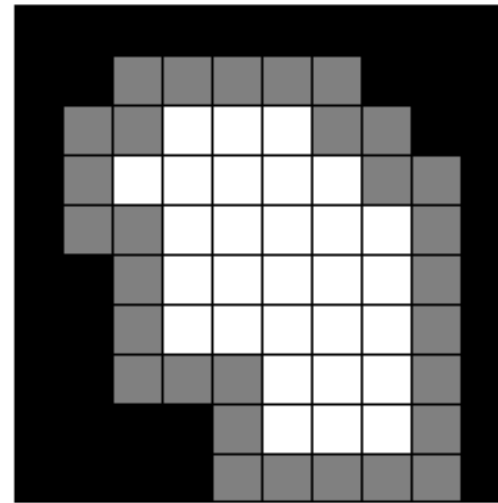
# Пример действия дилатации и эрозии



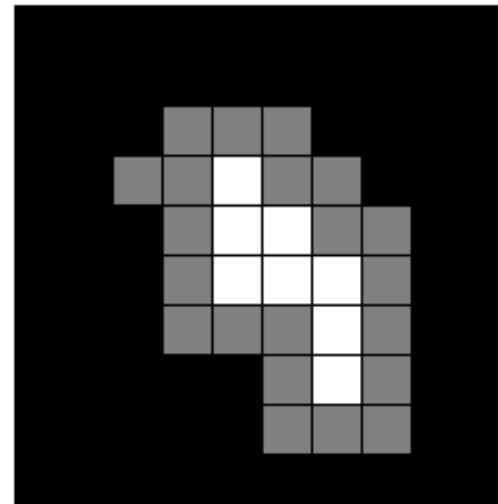
а) исходное изображение



б) шаблон (центр – ведущий элемент)



с) результат дилатации



д) результат эрозии



# Дилатация и эрозия. Функции OpenCV

```
void dilate(InputArray src, OutputArray dst,  
            InputArray element,  
            Point anchor=Point(-1, -1),  
            int iterations=1,  
            int borderType=BORDER_CONSTANT,  
            const Scalar& borderValue =  
                morphologyDefaultBorderValue())  
void erode(.)
```

- ❑ `element` – шаблон, который используется в процессе дилатации. Если `element=Mat()`, то применяется квадратный шаблон размером 3x3.





Библиотека OpenCV

# МОДУЛЬ DNN



# Модуль dnn. Состав модуля

- ❑ API для создания новых слоев, которые по сути являются строительными единицами нейронных сетей
  - ❑ Множество встроенных наиболее полезных слоев
  - ❑ API для построения и изменения нейронных сетей из слоев
  - ❑ Функционал, необходимый для загрузки моделей нейронных сетей в форматах, которые используются библиотеками Caffe, Torch, TensorFlow
- 
- ❑ ***Функциональность этого модуля направлена на поддержку прямого прохода вычислений по сети, т.е. для тестирования сетей***



# Общая схема тестирования нейронной сети



# Функции чтения модели нейронной сети

```
cv::dnn::Net cv::dnn::readNetFromCaffe(  
    const String &prototxt,  
    const String &caffeModel=String())  
cv::dnn::Net cv::dnn::readNetFromTensorflow(  
    const String &model)  
cv::dnn::Net cv::dnn::readNetFromTorch(  
    const String &model, bool isBinary=true)
```

- ❑ Чтение натренированных моделей в форматах, с которыми работают библиотеки Caffe, TensorFlow, Torch
- ❑ Возвращают объект нейронной сети



# Формат prototxt на примере сети GoogLeNet

```
name: "GoogLeNet"
input: "data"
input_dim: 10
input_dim: 3
input_dim: 224
input_dim: 224
layer {
  name: "conv1/7x7_s2"
  type: "Convolution"
  bottom: "data"
  top: "conv1/7x7_s2"
  convolution_param {
    num_output: 64
    pad: 3
    kernel_size: 7
    stride: 2
    weight_filler {
      type: "xavier"
      std: 0.1
    }
  }
  ...
}
```



# Функции подготовки изображения (1)

```
Mat cv::dnn::blobFromImage(const Mat &image,  
    double scalefactor=1.0, const Size &size=Size(),  
    const Scalar &mean=Scalar(), bool swapRB=true)  
Mat cv::dnn::blobFromImages(  
    const std::vector<Mat> &images,  
    double scalefactor=1.0, Size size=Size(),  
    const Scalar &mean=Scalar(), bool swapRB=true)
```

- ❑ Создание четырехмерного блока из изображения/набора изображений размерности  $[W \times H \times C \times N]$ ,  $W$  – ширина изображения,  $H$  – высота изображения,  $C$  – число каналов,  $N$  – количество изображений



## Функции подготовки изображения (2)

```
Mat cv::dnn::blobFromImage(const Mat &image,  
    double scalefactor=1.0, const Size &size=Size(),  
    const Scalar &mean=Scalar(), bool swapRB=true)  
Mat cv::dnn::blobFromImages(  
    const std::vector<Mat> &images,  
    double scalefactor=1.0, Size size=Size(),  
    const Scalar &mean=Scalar(), bool swapRB=true)
```

- Дополнительно обеспечивается масштабирование с коэффициентом **scalefactor**, обрезка изображения для размеров входа сети **size**, вычитание среднего значения интенсивности **mean** из всех пикселей изображения





# Класс представления нейронной сети Net

- ❑ Класс `cv::dnn::Net` предоставляет возможности для создания и манипулирования нейронными сетями
- ❑ Нейронная сеть представляется направленным ациклическим графом
- ❑ Вершина графа – слой нейронной сети. Каждый слой сети имеет уникальный целочисленный идентификатор или уникальное в рамках сети название
- ❑ Ребра графа – отношения между слоями
- ❑ Для представления слоев сети определен класс `cv::dnn::Layer`. В [документации](#) представлен полный список поддерживаемых слоев



# Прямой проход сети. Методы класса Net

```
void setInput(const Mat &blob,  
              const String &name = "")
```

- ❑ Установка данных blob в качестве входа name сети

```
Mat forward(const String &outputName=String())  
void forward(std::vector<Mat> &outputBlobs,  
             const String &outputName=String())  
void forward(std::vector<Mat> &outputBlobs,  
             const std::vector<String> &outBlobNames)
```

- ❑ Вычисление значений выходного слоя нейронной сети с названием outputName / outBlobNames



# Названия входного/выходного слоев сети GoogLeNet

```
name: "GoogLeNet"
```

```
input: "data"
```

```
input_dim: 10
```

```
input_dim: 3
```

```
input_dim: 224
```

```
input_dim: 224
```

```
layer {
```

```
  name: "conv1/7x7_s2"
```

```
  type: "Convolution"
```

```
  bottom: "data"
```

```
  ...
```

```
}
```

Название входного слоя

Название выходного слоя

```
...
```

```
layer {
```

```
  name: "prob"
```

```
  type: "Softmax"
```

```
  bottom: "loss3/classifier"
```

```
  top: "prob"
```

```
}
```

# Анализ выхода сети

- ❑ **Метод анализа выхода сети зависит от задачи!**
- ❑ **Задача классификации изображений**
  - Выход сети – вектор достоверностей принадлежности изображения каждому из возможных классов
  - Анализ предполагает выбор класса с максимальной достоверностью
- ❑ **Задача семантической сегментации**
  - Выход сети – набор матриц, каждая из которых содержит достоверность принадлежности пикселей определенному классу
  - Анализ предполагает выбор класса с максимальной достоверностью для каждого пикселя



# Вопросы

---

□ ???

