**Национальный исследовательский**
**Нижегородский государственный университет им. Н.И. Лобачевского**
**Институт информационных технологий, математики и механики**

# Обзор инструмента Intel® Distribution of OpenVINO™ toolkit

Евгений Васильев,
преподаватель кафедры МОСТ ИИТММ,
ННГУ им. Н.И. Лобачевского

# Содержание

- Общая информация об инструменте Intel® Distribution of OpenVINO™ toolkit
- Состав инструмента Intel® Distribution of OpenVINO™ toolkit
  - Model Optimizer
  - Inference Engine
  - Open Model Zoo
  - OpenCV
- Возможности модуля Inference Engine для реализации вывода глубоких моделей

# ОБЩАЯ ИНФОРМАЦИЯ ОБ ИНСТРУМЕНТЕ INTEL® DISTRIBUTION OF OPENVINO™ TOOLKIT

# Общая информация об инструменте Intel® Distribution of OpenVINO™ toolkit (1)

- Intel® Distribution of OpenVINO™ toolkit – инструмент для решения задач компьютерного зрения глубокого обучения, разрабатываемый компанией Intel
- Цель разработки – простота использования различных алгоритмов компьютерного зрения и глубокого обучения на различном оборудовании
- Основное преимущество продукта – производительность, минимальный размер и небольшое количество зависимостей
- Важным преимуществом является возможность работать с глубокими нейронными сетями, разработанными в разных библиотеках глубокого обучения.

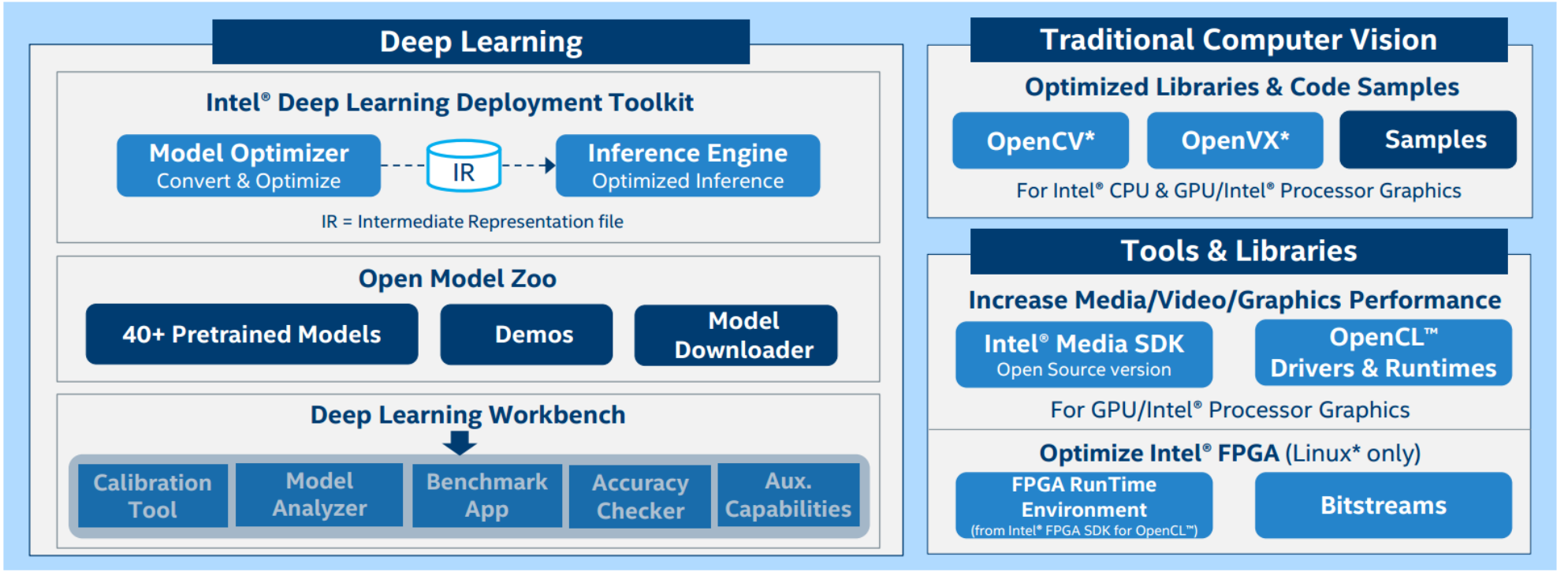
# Общая информация об инструменте Intel® Distribution of OpenVINO™ toolkit (2)

- Лицензия: EULA (также существует открытая версия OpenVINO™ toolkit под лицензией Apache 2.0, https://01.org/openvinotoolkit)

- Документация [https://docs.openvinotoolkit.org]

- Официальная страница Intel® Distribution of OpenVINO™ toolkit [https://software.intel.com/en-us/openvino-toolkit]

# Состав Intel® Distribution of OpenVINO™ toolkit (1)




**OS Support:** CentOS* 7.4 (64 bit), Ubuntu* 16.04.3 LTS (64 bit), Microsoft Windows* 10 (64 bit), Yocto Project* version Poky Jethro v2.0.3 (64 bit), macOS* 10.13 & 10.14 (64 bit)


Источник: *Anna Belova. Introduction to the Intel® Distribution of OpenVINO™ toolkit. Tutorial "Object detection with deep learning: Performance optimization of neural network inference using the Intel OpenVINO™ toolkit " on PPAM 2019*

# Состав Intel® Distribution of OpenVINO™ toolkit (2)

- **Intel® Deep Learning Deployment Toolkit**
  - **Model Optimizer**
  - **Inference Engine**
  - GNA library
  - Intel® Vision Accelerator Design with Intel® Movidius™ VPUs
  - **Open Model Zoo**
  - **Deep Learning Workbench**
- **OpenCV**
- OpenVX
- Intel® Graphics Compute Runtime for OpenCL™
- Intel® Media SDK for CV

# Model Optimizer

- OpenVINO для исполнения нейронных сетей конвертирует их в промежуточное представление (intermediate representation, IR)

- Model Optimizer – инструмент для конвертирования моделей из различных форматов в промежуточное представление
- Допустимые форматы: ONNX, TensorFlow, Caffe, MXNet, Kaldi

- Документация по Model Optimizer [https://docs.openvinotoolkit.org/latest/_docs_MO_DG_Deep_Learning_Model_Optimizer_DevGuide.html]

# Inference Engine

- Inference Engine предоставляет API для выполнения эффективного вывода глубоких нейронных сетей на следующих платформах:
  - Intel® CPUs
  - Intel® Processor Graphics
  - Intel® FPGAs
  - Intel® Movidius™ Neural Compute Stick и т.д.
- Inference Engine поддерживает гетерогенный вывода нейронных сетей, например, Intel® CPU + Intel® Processor Graphics

# Open Model Zoo

- Open Model Zoo – набор примеров и демо-приложений для знакомства с возможностями библиотеки OpenVINO™
  - Модели глубоких сетей в разных форматах (публичные модели + модели, обученные инженерами компании Intel)
  - Модели можно использовать для решения классических задач (классификация изображений, детектирование объектов, семантическая сегментация и др.)
  - Набор примеров и демо-приложений на C++ и Python
  - Модуль для автоматического скачивания моделей и их конвертации в промежуточное представление
- Репозиторий Open Model Zoo [https://github.com/opencv/open_model_zoo]

# Модели, обученные в Intel (1)

- Задачи детектирования объектов (object detection)
  - Детектирование и распознавание лиц
  - Детектирование людей, пешеходов
  - Детектирование автомобилей и их типа, цвета
  - Детектирование автомобильных номеров
- Задачи распознавания объектов (object recognition)
  - Распознавание пола и возраста
  - Распознавание позы человека
  - Распознавание эмоций
- Задачи сегментации изображений (segmentation)
  - Семантическая сегментация (semantic segmentation)
  - Объектная сегментация (instance segmentation)

# Модели, обученные в Intel (2)

- Задачи отслеживания объектов (tracking objects)
  - Задача сопровождения людей
  - Задача распознавания человека
- Задача обработки изображений (image processing)
  - Увеличение разрешения изображений
- Работа с текстом
  - Детектирование слов
  - Распознавание слов
- Распознавание движений (action recognition)
  - Распознание движений
  - Распознавание движений водителя

# Примеры и демо-приложения

- Каждую из готовых моделей можно попробовать в демо-приложении
- Демо-приложения решают конкретную небольшую или крупную задачу, например, приложение «умная классная комната» или «дорожная камера»
- В демо-приложении можно использовать несколько разных взаимозаменяемых глубоких нейронных сетей, поскольку разные архитектуры дают разную точность и скорость работы
- Демо-приложения доступны по ссылке [https://github.com/opencv/open_model_zoo/tree/master/demos]

# Deep Learning Workbench

- Deep Learning Workbench – набор инструментов, необходимых для решения задач исследования качества работы модели и повышения производительности модели за счет использования грубого типа данных INT8.
  - Accuracy Checker tool – инструмент для подсчета точности работы модели на тренировочном наборе данных.
  - Calibration tool – инструмент для конвертации моделей в INT8 представление весов.
  - Deep Learning Workbench – инструмент, предоставляющий графическую оболочку для работы с Accuracy checker tool и Calibration tool.

# OpenCV

- OpenCV (библиотека компьютерного зрения с открытым исходным кодом) — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом
- Реализована на C/C++, имеет программный интерфейс для Python, Java, и других языков
- Лицензия BSD 3-Clause License, может свободно использоваться в коммерческих проектах
- OpenCV [https://opencv.org]

# Состав библиотеки OpenCV

- Основные модули библиотеки
  - **core**. Ядро библиотеки, основные типы данных и математические функции.
  - **imgproc**. Модуль обработки изображений (фильтрация изображений, функции рисования, цветовые пространства)
  - **video**. Модуль видеоаналитики классическими методами
  - **features2d**. Модуль содержит детекторы и дескрипторы ключевых точек изображения
  - **objdetect**. Модуль детектирования объектов с помощью каскадных классификаторов
  - **ml**. Модуль классических алгоритмов машинного обучения (кластеризация, регрессия, статистическая классификация)
  - **dnn**. Модуль для вывода глубоких сетей

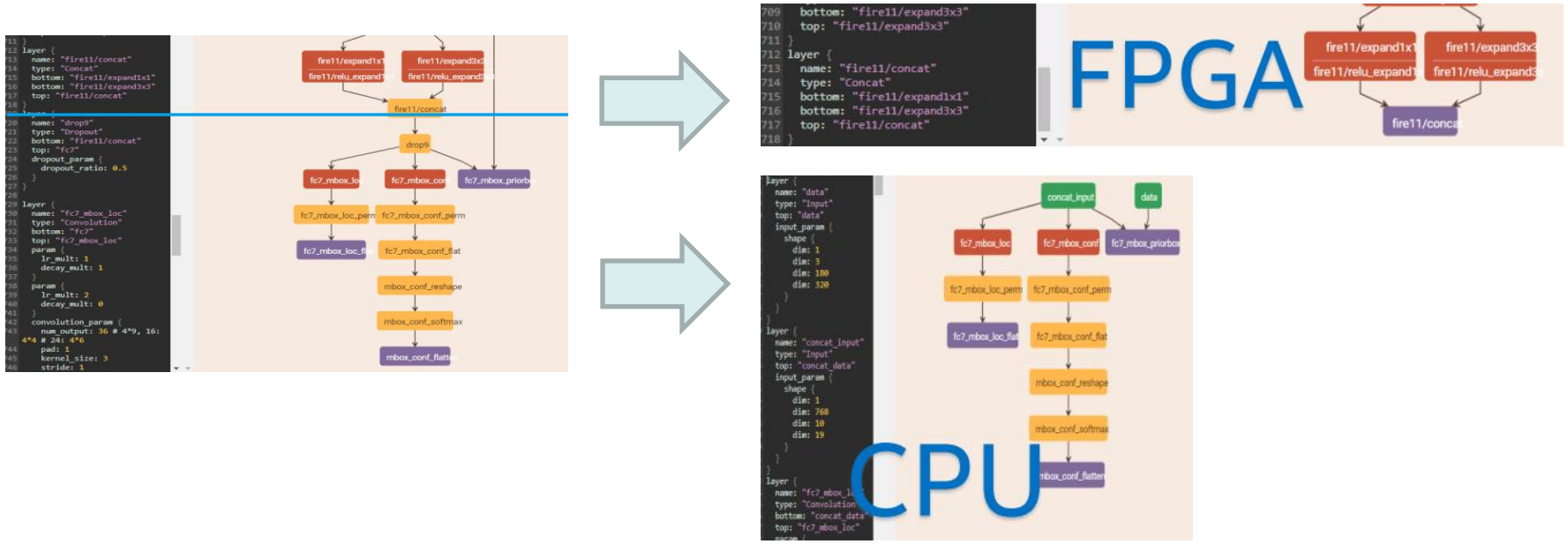
# ВОЗМОЖНОСТИ INFERENCE ENGINE ДЛЯ РЕАЛИЗАЦИИ ВЫВОДА ГЛУБОКИХ МОДЕЛЕЙ

# Inference Engine

- Inference Engine – высокоуровневый API (C++, Python) для запуска глубоких нейронных сетей в промежуточном представлении на различном аппаратном обеспечении за счет подключаемых плагинов
  - Плагин MKL-DNN (CPU на основе x86 архитектуры)
  - Плагин clDNN (Intel GPUs на основе OpenCL)
  - Плагин для FPGA
  - Плагин MYRIAD для Neural Compute Stick (OpenCL)
  - Плагин GNA (для маломощных мобильных процессоров)

# Гетерогенный вывод

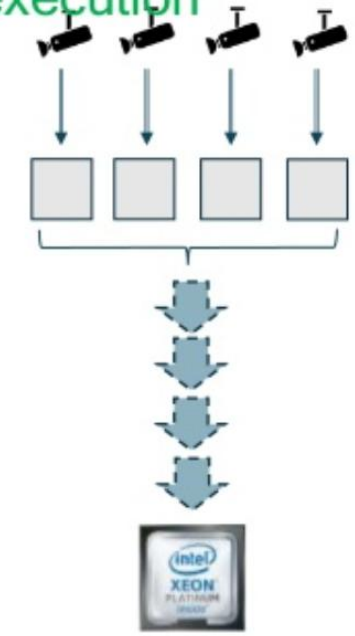
- Inference Engine поддерживает разбиение нейронных сетей на отдельные части (слои) и выполнение их на разных устройствах, например, CPU+GPU, CPU+FPGA



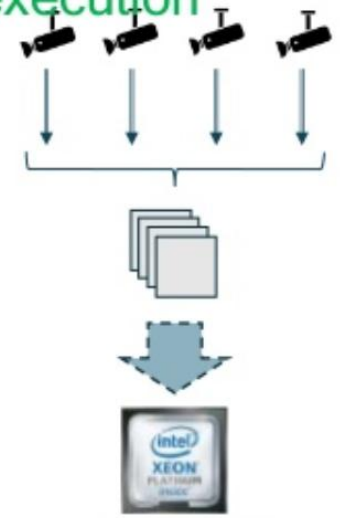

Источник: *Anna Belova. Introduction to the Intel® Distribution of OpenVINO™ toolkit. Tutorial “Object detection with deep learning: Performance optimization of neural network inference using the Intel OpenVINO™ toolkit ” on PPAM 2019*

# Варианты запуска обработки изображений

Serialized execution



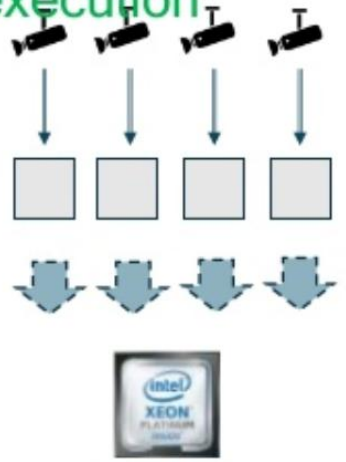
Batched execution



+56%

Throughput Performance on XEON 8180

Parallel execution



+83%

Throughput Performance on XEON 8180

Better latency and easier programming

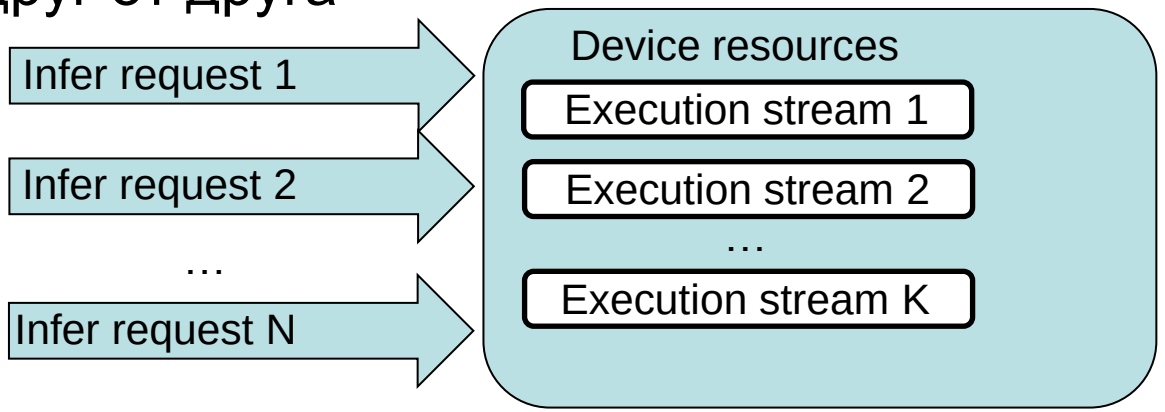
Источник: *https://www.slideshare.net/YuryGorbachev1/how-to-get-the-best-deep-learning-performance-with-openvino-toolkit/8*

# Синхронное и асинхронное выполнение вывода

- Поддерживается два режима вывода:
  - ***Синхронный режим (synchronous or latency).*** Очередной запрос на исполнение выполняется по завершении предыдущего
  - ***Асинхронный режим (asynchronous or throughput).*** Предполагает формирование очереди запросов на исполнение, очередной запрос не дожидается исполнения предыдущего
    - Объявляется функция обратного вызова (callback)
    - Проверяется состояние выполнения функцией wait()

# Асинхронный режим

- Асинхронный режим предполагает, что повышение производительности происходит не за счет уменьшения времени обработки одного изображения, а за счет параллельной обработки нескольких запросов, поставленных на исполнение одновременно
- Позволяет увеличить общую пропускную способность
- В данном режиме ресурсы системы делятся на потоки (streams), в которых вычисления происходят одновременно и независимо друг от друга



# Асинхронный режим на гетерогенных устройствах

- Плагин Multi-Device автоматически назначает запросы на вывод доступным вычислительным устройствам для параллельного выполнения запросов

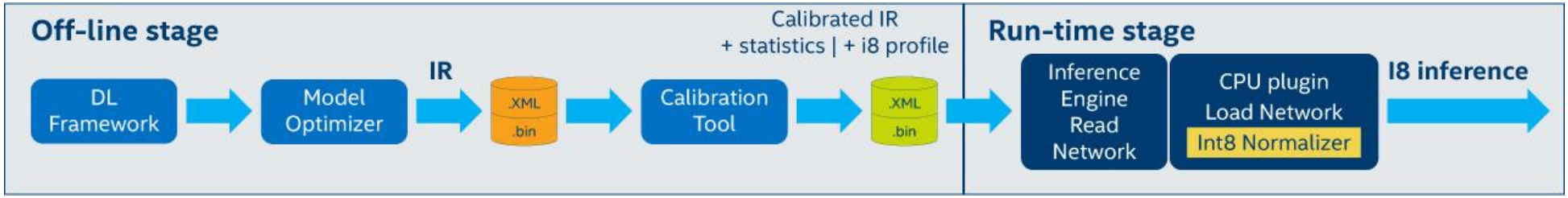
- Различные комбинации оборудования:
  - Intel® CPU + Intel® Processor Graphics
  - Intel® CPU + Intel® Movidius™ Neural Compute Stick
  - Несколько Intel® Movidius™ Neural Compute Stick
  - и т.д.

# Ускорение с помощью типа данных INT8

- Снижение точности модели – эффективный способ ускорить исполнение вывода сетей на процессорах, которые поддерживают тип данных INT8 на уровне железа
- При переходе от FP32 или FP16 к INT8 значительно ускоряется вывод, при этом на большей части задач потеря точности незначительна
- Inference Engine поддерживает работу с INT8 моделями, а процессоры Intel® архитектуры Cascade Lake поддерживают специальные инструкции для ускорения вывода INT8 моделей

# Создание INT8 моделей (1)

- Два варианта создания INT8 моделей
  - Конструирование и обучение сети в формате INT8
  - Квантизация весов и калибровка сети при помощи инструмента Calibration Tool




Источник: https://www.intel.ai/introducing-int8-quantization-for-fast-cpu-inference-using-openvino/

# Создание INT8 моделей (2)

- Если после квантизации весов и калибровки сети происходит сильное ухудшение качества работы сети, то те слои, чья точность упала ниже пороговой, не квантуются в INT8, а остаются в FP32
- Таким образом модель становится смешанной, состоящей из INT8 и FP32 слоев
- Это компромисс между поддержанием высокой точности и максимальным ускорением модели

# ПРИМЕНЕНИЕ INFERENCE ENGINE ДЛЯ РЕАЛИЗАЦИИ ВЫВОДА ГЛУБОКИХ МОДЕЛЕЙ

# Общая процедура разработки программы с использованием Inference Engine

***Разработка конвейера, содержащего следующие этапы***

1. Загрузка модели в
2. Загрузка изображения
3. Конвертация изображения в формат входа нейронной сети
4. Исполнение нейронной сети
5. Обработка результата

# Общая процедура разработки программы с использованием Inference Engine

***Разработка конвейера, содержащего следующие этапы***

1. Загрузка модели в Inference Engine
2. Конвертация изображения в формат входа нейронной сети
3. Исполнение нейронной сети
4. Обработка результата

# 1. Загрузка модели в Inference Engine

- Инициализировать Inference Engine
- Загрузить модель в Inference Engine
- Загрузить модель в устройство

```
from openvino.inference_engine import IENetwork, IECore

ie = IECore()
if cpu_extension:
    ie.add_extension(cpu_extension, 'CPU')

net = IENetwork(model=configPath, weights=weightsPath)

exec_net = self.ie.load_network(network=self.net,
                                device_name='CPU‘)
```

# 2. Конвертация изображения в формат входа нейронной сети (1)

- Как правило, на вход нейронной сети подается тензор размера [BxCxHxW]
  - B – количество изображений
  - C – количество каналов в изображении
  - H – высота изображения
  - W – ширина изображения
- Например, вход сети MobileNet-SSD: [1, 300, 300]
- Необходимо перевести изображение из формата OpenCV {BGRBGRBGR} в формат тензора {RRRGGGBBB}, и изменить его размер

# 2. Конвертация изображения в формат входа нейронной сети (2)

- Алгоритм подготовки изображения
  - Изменяем размер изображения
  - Меняем последовательность BGR -> RGB (если необходимо)
  - Меняем чередование каналов
  - Добавляем 4 размерность тензору изображения

```
image = cv2.resize(image, (w, h))

image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

image = image.transpose((2, 0, 1))

image = np.expand_dims(image, axis=0)
```

# 3. Запуск нейронной сети на исполнение

- Для запуска нейронной сети на исполнение необходимо установить входной тензор как вход нейронной сети, и выполнить функцию `infer()`.

```
input_blob = next(iter(self.net.inputs))
out_blob = next(iter(self.net.outputs))
n, c, h, w = self.net.inputs[input_blob].shape

blob = self.prepare_image(image, h, w)

output = self.exec_net.infer(inputs={input_blob: blob})
    output = output[out_blob]
```

# 4. Обработка результата (1)

- Для обработки результата нужно знать, какой формат выходных данных.
- Выход модели MobileNet-SSD: тензор [1, 1, N, 7], где N – количество прямоугольников
- Каждый прямоугольник описывается вектором:

  [image_id, label, conf, xmin, ymin, xmax, ymax]
  - image_id – номер изображения в пачке
  - label – предсказанный класс объекта
  - conf – вероятность класса
  - (xmin, ymin) – координаты верхнего левого угла (от 0 до 1)
  - (xmax, ymax) – координаты нижнего правого угла (от 0 до 1)

# 4. Обработка результата (2)

- Функция рисования прямоугольника на изображении:
  - cv2.rectangle(image, p1x, p1y, p2x, p2y, (0, 255, 0), line_width)
- Функция рисования текста на изображении:
  - cv2.putText(image, ‘text’, p1x, p1y, cv2.FONT_HERSHEY_SIMPLEX, 0.8, (0, 255, 255), 2, cv2.LINE_AA)
- https://docs.opencv.org/master/dc/da5/tutorial_py_drawing_functions.html

# Производительность вывода модели

- Под производительностью вывода понимается задержка - время, необходимое на загрузку изображения в сеть и обработку изображения при помощи сети (время чтения изображения из файла и конвертации в формат сети не учитывается)
- Для измерения производительности повторяем загрузку данных и исполнение сети большое количество раз, исключаем выбросы, находим медианное значение времени исполнения

# Измерение производительности вывода

```
from time import time

times = []
t0_total = time()
for i in range(number_iter):
    t0 = time()
    output = exec_net.infer(...)
    t1 = time()
    time.append(t1 - t0);
}
t1_total = time()

latency = np.median(times)
fps = number_iternp.median / (t1_total-t0_total)
```

# Основная литература

- OpenVINO documentation website [https://docs.openvinotoolkit.org]
- OpenVINO - Open Sourced version [01.org/openvinotoolkit]
- OpenVINO performance topics [https://docs.openvinotoolkit.org/latest/_docs_IE_DG_Intro_to_Performance.html]
- CPU Inference Performance Boost with “Throughput” Mode in the Intel® Distribution of OpenVINO™ Toolkit [https://www.intel.ai/cpu-inference-performance-boost-openvino]
- Introducing int8 quantization using OpenVINO [https://www.intel.ai/introducing-int8-quantization-for-fast-cpu-inference-using-openvino]
- OpenCV [https://opencv.org]
- Open Model Zoo [https://github.com/opencv/open_model_zoo]