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

Образовательный курс
«Современные методы и технологии
глубокого обучения в компьютерном зрении»

Обзор инструмента Intel Distribution of OpenVINO Toolkit

При поддержке компании Intel

Васильев Евгений

Содержание

- Цель лекции
- Общая информация об инструменте Intel Distribution of OpenVINO Toolkit
- Состав инструмента Intel Distribution of OpenVINO Toolkit
- Компонент Inference Engine
- Модуль DNN библиотеки OpenCV
- Заключение

Цель лекции

- ***Цель*** – изучить возможности инструмента Intel Distribution of OpenVINO Toolkit для работы с глубокими нейросетевыми моделями, рассмотреть основные компоненты для реализации эффективного вывода нейронных сетей

ОБЩАЯ ИНФОРМАЦИЯ ОБ ИНСТРУМЕНТЕ INTEL DISTRIBUTION OF OPENVINO TOOLKIT

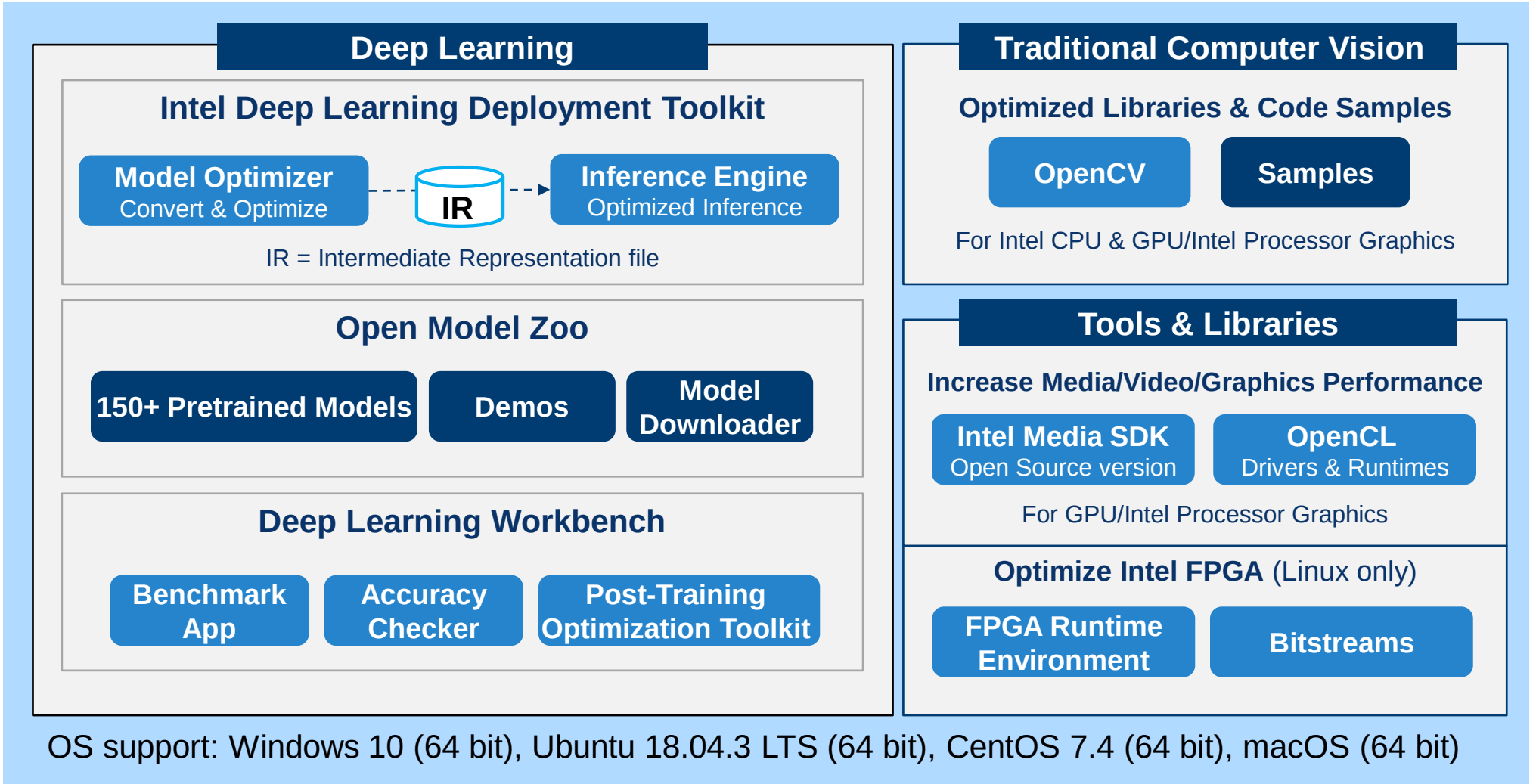
Общая информация об инструменте Intel Distribution of OpenVINO Toolkit (1)

- Intel Distribution of OpenVINO Toolkit – инструмент , разрабатываемый компанией Intel, для решения задач компьютерного зрения и глубокого обучения
- Цель разработки – простота и эффективность использования различных алгоритмов компьютерного зрения и глубокого обучения на различных аппаратных платформах Intel
- Основные преимущества:
  - Производительность, минимальный размер и небольшое количество зависимостей
  - Эффективный вывод глубоких нейронных сетей, разработанных и обученных с использованием разных библиотек 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)



Состав Intel Distribution of OpenVINO Toolkit (2)

- **Глубокое обучение для компьютерного зрения (Deep Learning for Computer Vision)**
  - Intel Deep Learning Deployment Toolkit (DLDT)
    - Model Optimizer – компонент для конвертации предварительно обученных моделей из формата какого-либо обучающего фреймворка в промежуточное представление (intermediate representation, IR) OpenVINO
    - Inference Engine – компонент для реализации эффективного вывода глубоких моделей на различных аппаратных платформах Intel
  - Open Model Zoo – открытый репозиторий, содержащий обученные модели для решения различных задач и примеры их использования
  - Deep Learning Workbench – набор инструментов для калибровки моделей, вычисления точности работы моделей и бенчмаркинга

Состав Intel Distribution of OpenVINO Toolkit (3)

- **Классическое компьютерное зрение (Traditional Computer Vision)**
  - OpenCV – широко известная библиотека компьютерного зрения
- **Вспомогательные пакеты (Tools & Packages)**
  - Набор пакетов для повышения производительности работы с графикой и видео

Model Optimizer

- Для эффективного вывода нейронных сетей OpenVINO обеспечивает их конвертацию в ***промежуточное представление*** (intermediate representation, IR)

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

- Документация к Model Optimizer [https://docs.openvino.ai/latest/openvino_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 вывод нейронных сетей, который предполагает распределение слоев сети между вычислительными устройствами
- Inference Engine также поддерживает вывод на нескольких устройствах (multi-device), при котором множество запросов на вывод распределяется между доступными устройствами

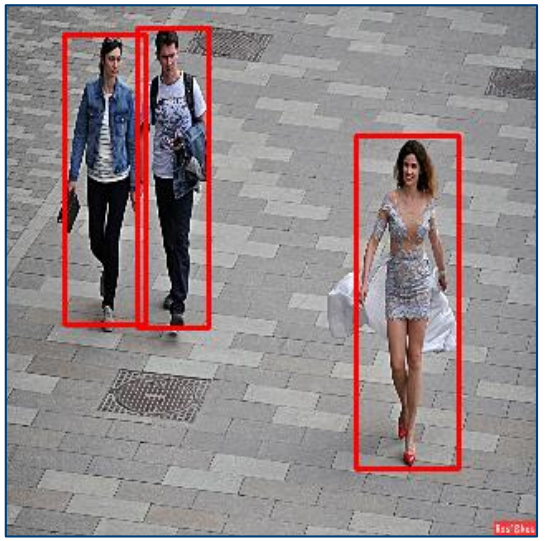
Open Model Zoo

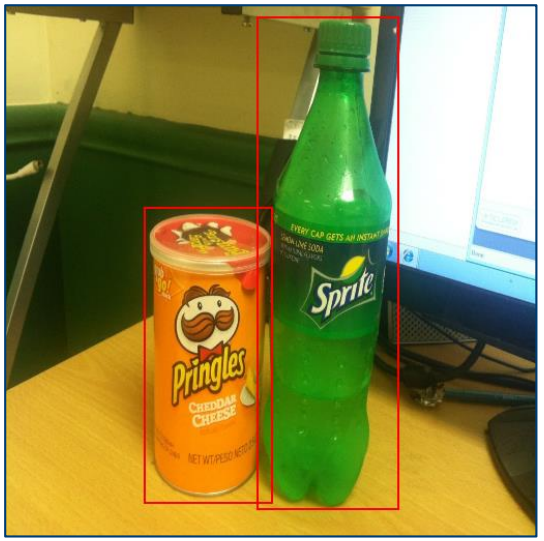
- ***Open Model Zoo***
  - Модели глубоких сетей в разных форматах (публичные модели и модели, обученные инженерами компании Intel)
  - Набор примеров и демо-приложений на C++ и Python
  - Модуль Model Downloader для автоматического скачивания моделей и их конвертации в промежуточное представление

- 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

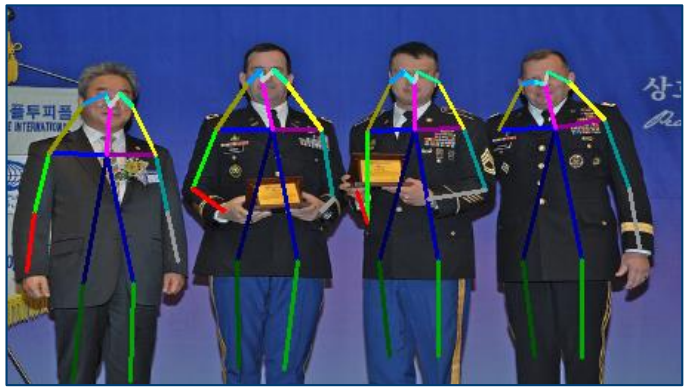




* Validation results for the selected Intel models [https://github.com/itlab-vision/openvino-dl-benchmark/blob/master/results/validation_results_intel_models.md].

Модели, обученные в Intel (2)

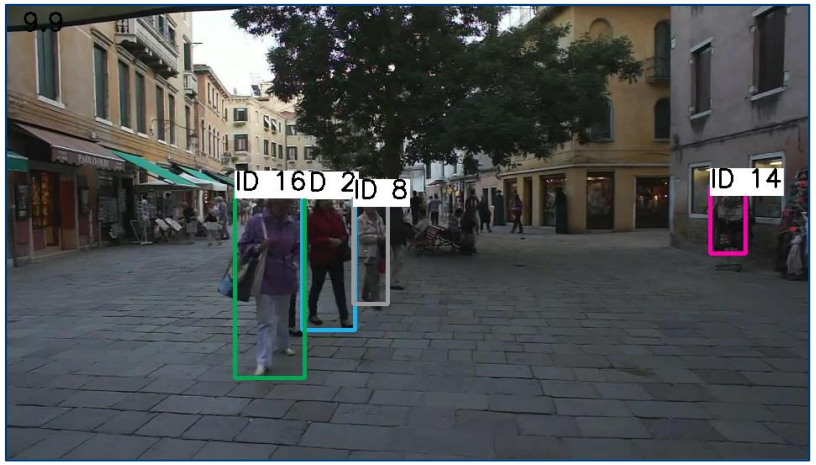
- Задачи распознавания объектов (object recognition)
  - Распознавание пола и возраста
  - Распознавание эмоций
- Задача определения позы человека
- Задачи сегментации изображений (segmentation)
  - Семантическая сегментация (semantic segmentation)
  - Сегментация объектов (instance segmentation)



* Validation results for the selected Intel models [https://github.com/itlab-vision/openvino-dl-benchmark/blob/master/results/validation_results_intel_models.md].

Модели, обученные в Intel (3)

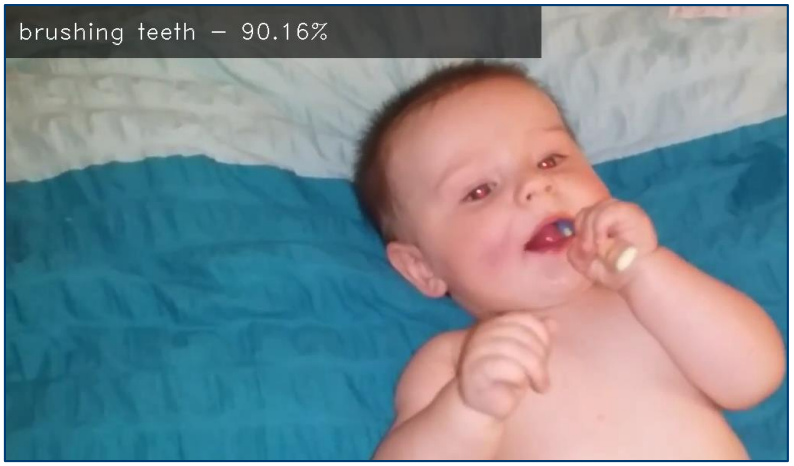
- Задачи отслеживания объектов (object tracking)
  - Задача сопровождения людей
  - Задача идентификации человека
- Задача обработки изображений (image processing)
  - Увеличение разрешения изображений



* Validation results for the selected Intel models [https://github.com/itlab-vision/openvino-dl-benchmark/blob/master/results/validation_results_intel_models.md].

Модели, обученные в Intel (4)

- Работа с текстом
  - Детектирование слов
  - Распознавание слов
- Распознавание движений (action recognition)
  - Распознание движений
  - Распознавание движений водителя



* Validation results for the selected Intel models [https://github.com/itlab-vision/openvino-dl-benchmark/blob/master/results/validation_results_intel_models.md].

Примеры и демо-приложения

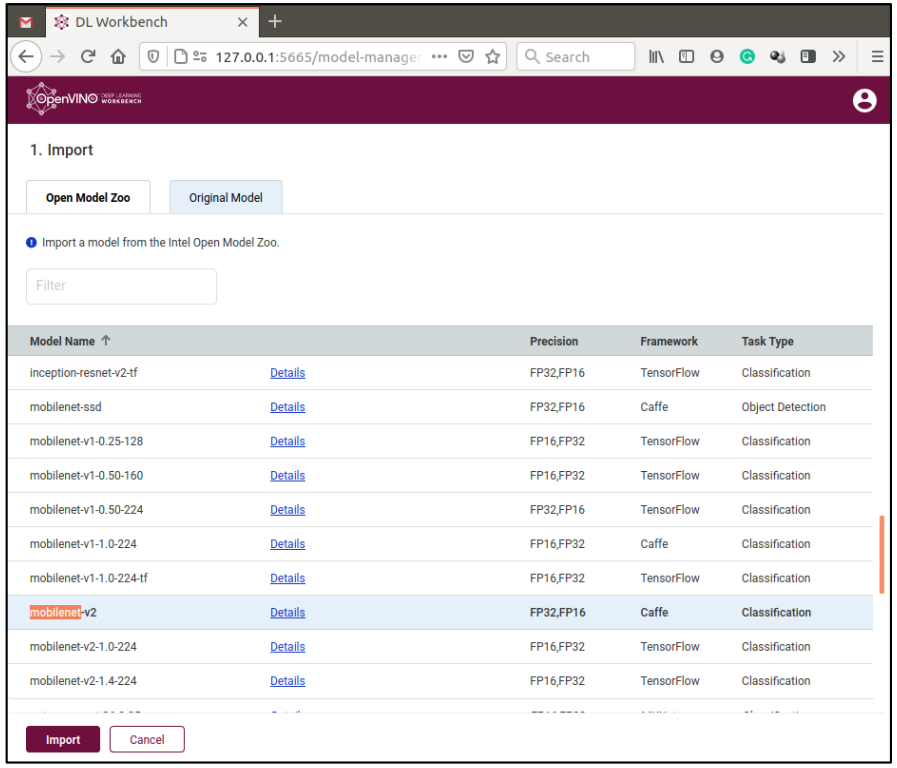
- Обученные модели можно использовать в демо-приложениях
- Демо-приложения решают конкретную небольшую или крупную задачу
  - Приложение «умная классная комната» позволяет обнаруживать учеников и распознавать их движения
  - Приложение «дорожная камера» позволяет детектировать автомобили и распознавать их регистрационные номера
- В демо-приложении можно использовать разные глубокие нейронные сети, обеспечивающие решение одной и той же задачи. Разные архитектуры могут демонстрировать разную точность и скорость работы
- Демо-приложения доступны в репозитории [https://github.com/openvinotoolkit/open_model_zoo]

Deep Learning Workbench (1)

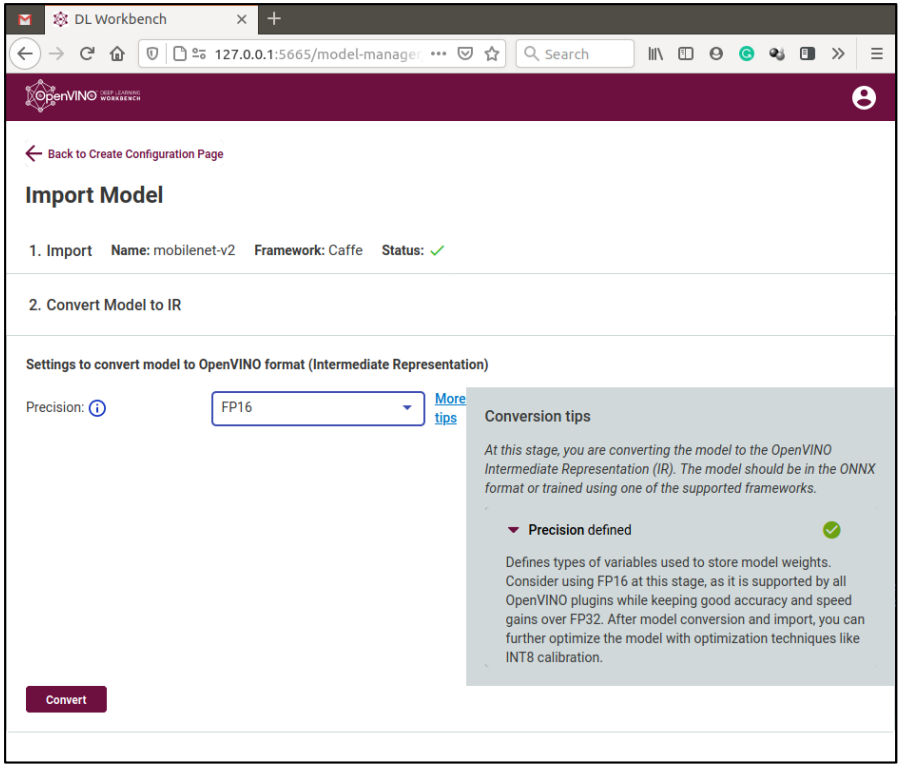
- ***Deep Learning Workbench*** – набор инструментов, необходимых для исследования качества работы модели и повышения производительности модели
  - Deep Learning Workbench – инструмент, предоставляющий графическую оболочку для инструментов OpenVINO. Также позволяет собрать и визуализировать статистику исполнения модели
  - Benchmark App – инструмент для оценки производительности глубоких моделей на различных устройствах
  - Accuracy Checker Tool – инструмент для определения качества работы модели на предоставленном наборе данных
  - Post-Training Optimization Toolkit – инструмент для оптимизации моделей посредством их преобразования в дружественное оборудованию представление

Deep Learning Workbench (2)

- Deep Learning Workbench объединяет инструменты OpenVINO в едином приложении с графическим интерфейсом для удобства работы с OpenVINO



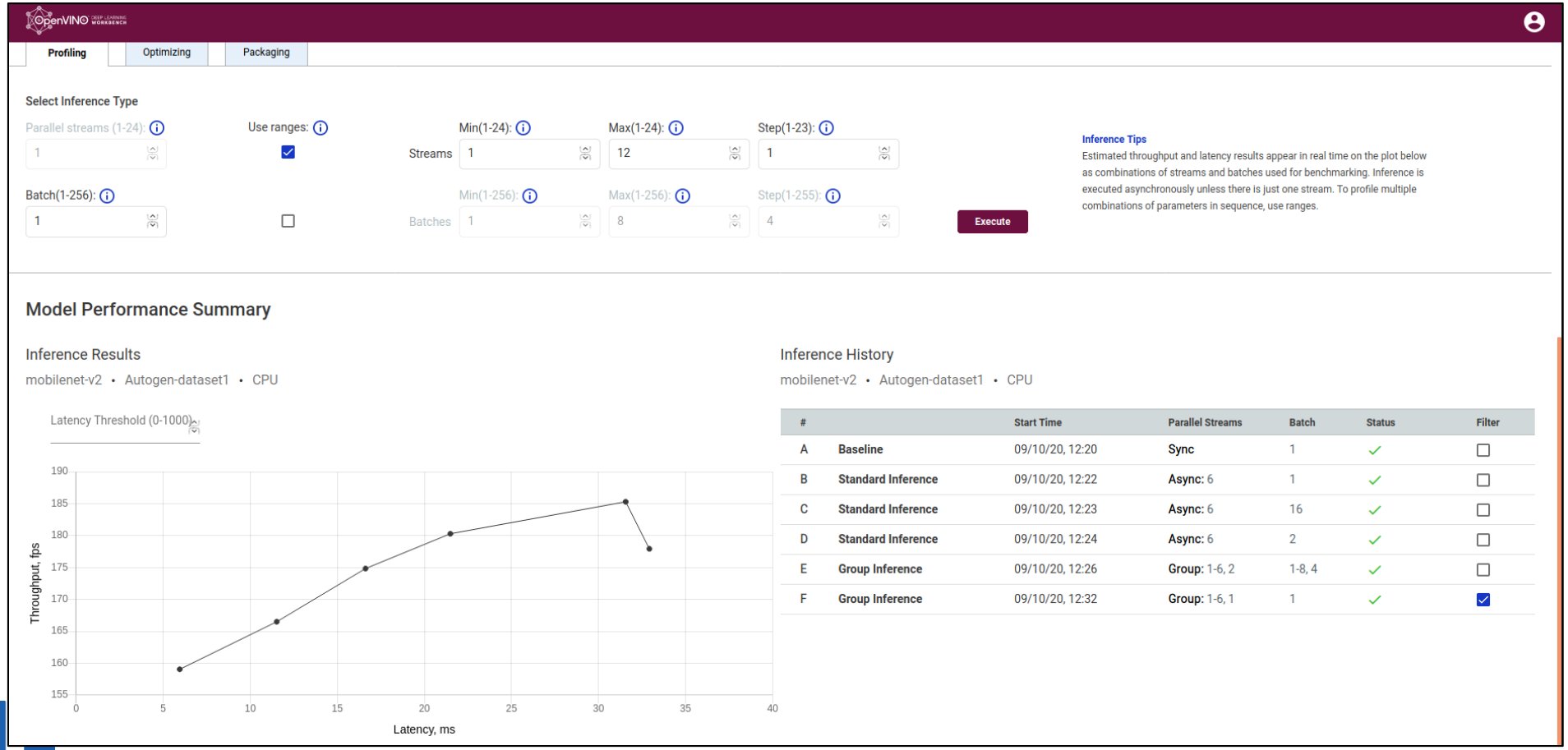
Скачивание модели



Конвертация
и оптимизация модели

Deep Learning Workbench (3)

- Deep Learning Workbench позволяет провести эксперименты чтобы выбрать параметры для лучшей производительности



OpenCV

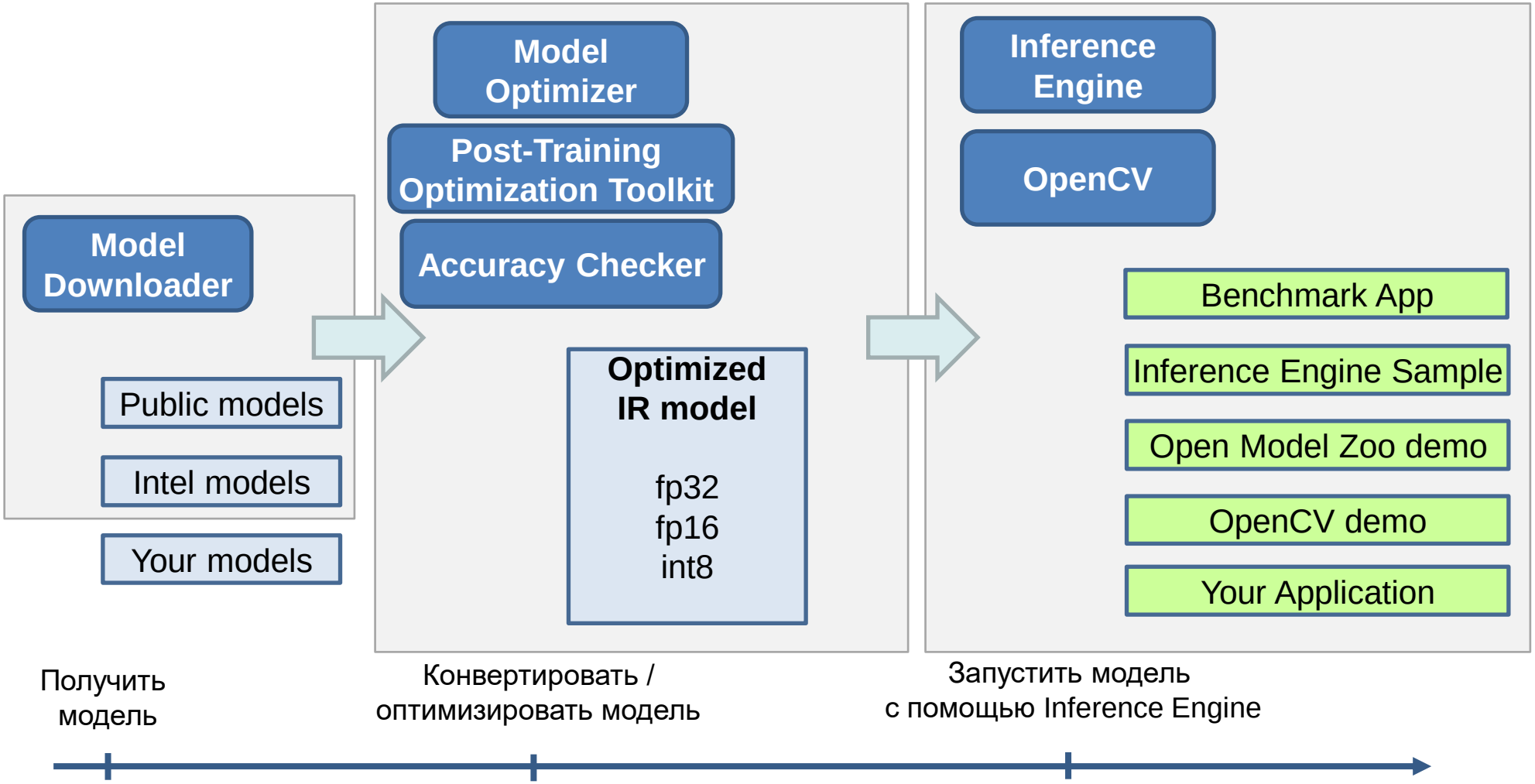
- ***OpenCV*** – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения
- Лицензия BSD 3-Clause License (может использоваться в коммерческих проектах)
- Реализована на C/C++, имеет программные интерфейсы для Python, Java, и других языков программирования

- Официальная страница OpenCV [https://opencv.org]

Состав библиотеки OpenCV

- Основные модули библиотеки:
  - **core** – ядро библиотеки, содержащее основные типы данных и математические функции
  - **imgproc** – модуль обработки изображений (фильтрация изображений, функции рисования, цветовые пространства)
  - **video** – модуль видеоаналитики
  - **features2d** – модуль, содержащий реализацию детекторов и дескрипторов 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 нейронных сетей

Связь компонентов Intel Distribution of OpenVINO Toolkit



Рассматриваемые компоненты

- Далее в лекции рассматриваются компоненты Intel Distribution of OpenVINO Toolkit, обеспечивающие вывод глубоких нейросетевых моделей
  - Компонент Inference Engine
  - Модуль DNN библиотеки OpenCV
- Схема изучения:
  - Назначение и возможности компонента
  - Программный интерфейс
  - Пример использования

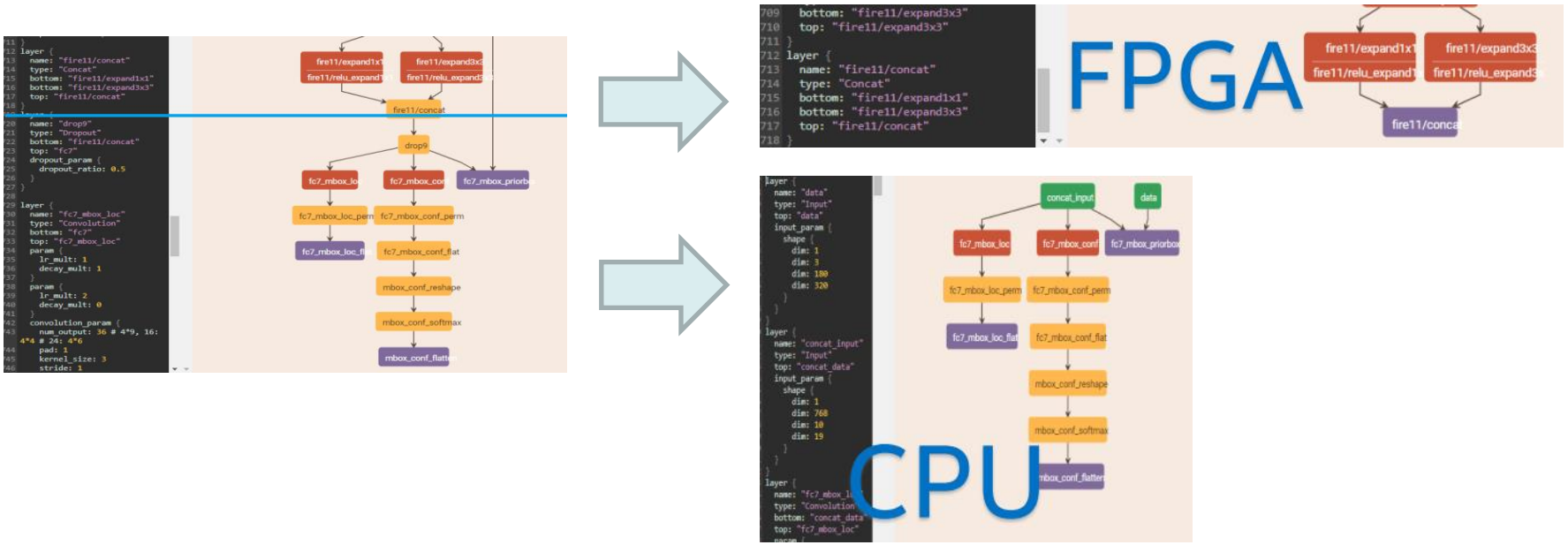
КОМПОНЕНТ INFERENCE ENGINE ДЛЯ РЕАЛИЗАЦИИ ВЫВОДА ГЛУБОКИХ МОДЕЛЕЙ

Inference Engine

- ***Inference Engine*** – компонент, предоставляющий высокоуровневый программный интерфейс (C++, C, Python) для вывода глубоких нейронных сетей в промежуточном представлении на различном аппаратном обеспечении Intel за счет подключаемых плагинов
  - Плагин CPU (для Intel Xeon, Intel Core Processors и Intel Atom Processors, реализован на базе библиотеки MKL-DNN)
  - Плагин GPU (для Intel Processor Graphics, clDNN (OpenCL))
  - Плагин для FPGA (для Intel Programmable Acceleration Card)
  - Плагин MYRIAD (для Intel Movidius Neural Compute Stick, OpenCL)
  - Гетерогенный плагин (heterogeneous plugin)
  - Плагин для нескольких устройств (multi-device plugin)

Гетерогенный вывод

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



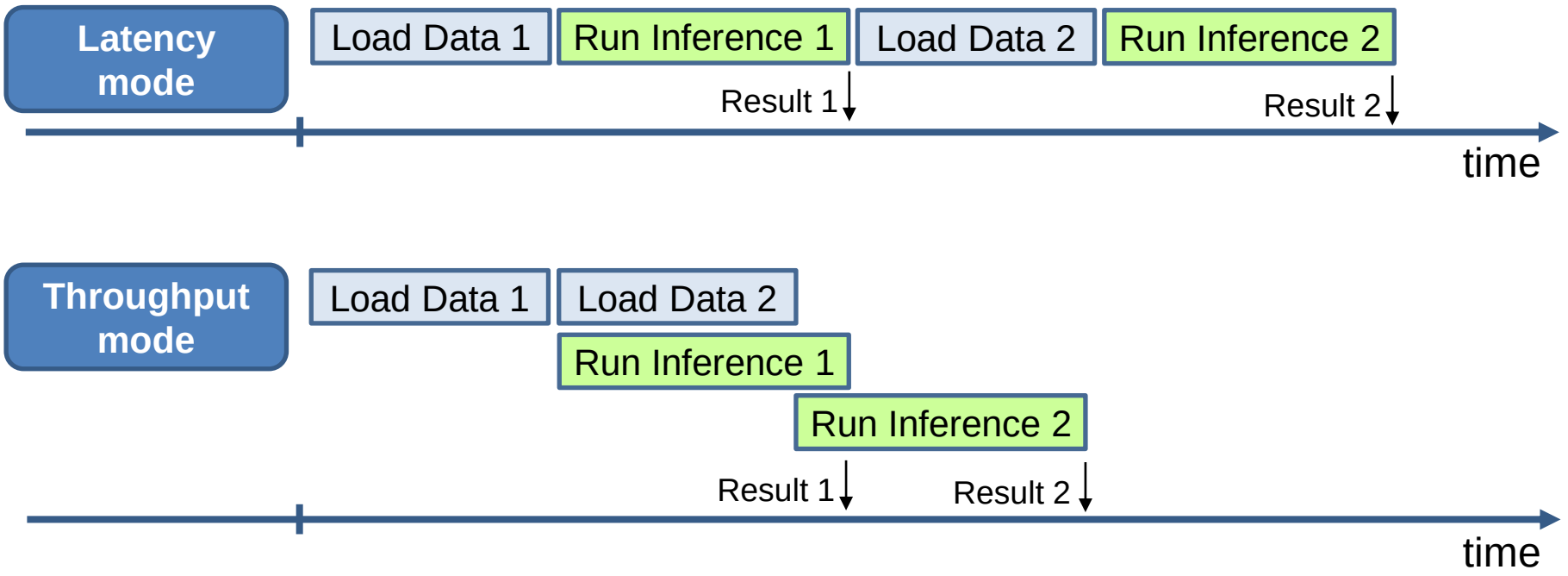
* Belova A. Introduction to the Intel Distribution of OpenVINO Toolkit. Tutorial “Object detection with deep learning: Performance optimization of neural network inference using the Intel OpenVINO toolkit” on PPAM 2019.

Режимы выполнения вывода (1)

- Поддерживается два режима вывода:
  - ***Режим минимизации времени исполнения запроса (latency mode).*** Очередной запрос на исполнение выполняется по завершении предыдущего, минимизируется время выполнения одного запроса за счет распараллеливания вычислений на прямом проходе сети
  - ***Режим повышения пропускной способности (throughput mode).*** Предполагает формирование очереди запросов на исполнение, очередной запрос не дожидается завершения предыдущего, максимизируется количество выполненных запросов за счет параллельного выполнения запросов

Режимы выполнения вывода (2)

- Схематическое представление работы разных режимов вывода:

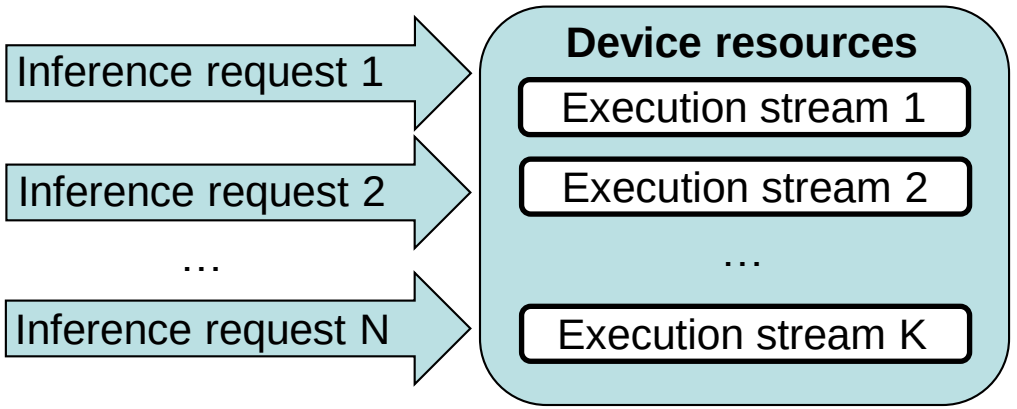


Режим минимизации времени исполнения запроса

- Режим минимизации времени исполнения запроса (latency mode) используется с целью минимизации времени вывода глубокой модели на одной входной пачке примеров
- Ускорение вывода на CPU обеспечивается за счет параллелизма вычислений на системах с общей памятью
- Параллелизм на CPU реализуется с использованием ***потоков*** (threads)
- Количество потоков на CPU является параметром и может быть установлено вручную, по умолчанию устанавливается оптимальное количество, равное количеству физических ядер в системе

Режим повышения пропускной способности

- Режим повышения пропускной способности (throughput mode) предполагает повышение производительности за счет параллельной обработки нескольких запросов, одновременно поставленных на исполнение
- Позволяет увеличить общую пропускную способность
- Режим позволяет разделить физические ***потоки*** (threads) системы на логические группы – ***нити*** (streams), в которых вычисления могут выполняться одновременно и независимо друг от друга. Каждая нить обрабатывает один ***запрос*** (request) на вывод



Программный интерфейс Inference Engine (1)

- При реализации вывода используются следующие классы модуля **`openvino.inference_engine`**:
  - **`IECore`** – класс для представления сущности Inference Engine, обеспечивает возможности для манипуляции с плагинами
  - **`IENetwork`** – класс для представления сущности загруженной модели в промежуточном представлении, позволяет манипулировать некоторыми параметрами модели (например, выходными слоями)
  - **`ExecutableNetwork`** – класс для представления модели, загруженной в плагин для последующего вывода
  - **`InferRequest`** – класс для представлении сущности запроса на вывод глубокой модели

* Inference Engine Python API Overview [https://docs.openvinotoolkit.org/latest/_inference_engine_ie_bridges_python_docs_api_overview.html].

Программный интерфейс Inference Engine (2)

Метод	Назначение
`read_network(model_path)`	Загружает глубокую модель в ОЗУ
`compile_model(model=read_model, device_name = 'CPU')`	Оптимизирует модель под конкретное устройство исполнения
`infer(inputs = {input_layer: image})`	Устанавливает на вход модели изображение и запускает вывод в синхронном режиме
`start_async(request_id = request_id, inputs = blobs[request_id])`	Устанавливает на вход модели изображение и запускает вывод в асинхронном режиме
`wait(-1)`	Заставляет дождаться завершения запроса на вывод в асинхронном режиме

Общая схема использования Inference Engine для вывода глубоких моделей

1. Загрузка модели
2. Загрузка изображений и конвертация в формат входа нейронной сети
3. Вывод нейронной сети
4. Обработка результата

1. Загрузка модели в Inference Engine

- Инициализировать Inference Engine средствами **`Core`**
- Прочитать модель с помощью функции **`read_network`** у объекта **`Core`**
- Загрузить модель в плагин и создать объект для исполнения вывода на устройстве с использованием функции **`compile_model`** у объекта **`Core`**

```
from openvino.runtime import Core
config_path = 'path_to_model_config.xml‘

ie = Core()
net = ie.read_model(model = config_path)
exec_net = ie.compile_model(model = net, device_name = 'CPU')
```

2. Загрузка изображений и конвертация в формат входа нейронной сети (1)

- Как правило, на вход нейронной сети подается тензор размера $[B \times C \times H \times W]$
  - B – количество изображений
  - C – количество каналов в изображении
  - H – высота изображения
  - W – ширина изображения
- Если изображения загружены средствами библиотеки OpenCV, то необходимо перевести тензор из формата **{BGRBGR…}** в формат **{RRR…GGG…BBB…}**, и изменить его размер в соответствии с размером входа модели

2. Загрузка изображений и конвертация в формат входа нейронной сети (2)

- Прочитать одно или несколько изображений средствами функции **`imread`**
- Изменить размер изображений (функция **`resize`**)
- Изменить последовательность каналов BGR -> RGB (если необходимо) в изображениях (функция **`cvtColor`**)
- Изменить порядок размерностей (функция **`transpose`**)
- Добавить четвертую размерность тензора, если загружено только одно изображение (**`expand_dims`**)

```
def prepare_image(imagePath, h, w):
    image = cv2.imread(imagePath)
    image = cv2.resize(image, (w, h))
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
    image = image.transpose((2, 0, 1))
    blob = np.expand_dims(image, axis = 0)
    return blob
```

3. Запуск вывода нейронной сети (1)

- Для запуска вывода нейронной сети в OpenVINO могут использоваться синхронный и асинхронный вызовы:
  - ***Синхронный вызов*** (Sync API) блокирует пользовательское приложение на время выполнения запроса на вывод, нет необходимости отслеживать завершение обработки запроса. Используется для реализации режима минимизации времени выполнения одного запроса (latency mode)
  - ***Асинхронный вызов*** (Async API) не блокирует пользовательское приложение на время выполнения запроса на вывод, необходимо самостоятельно отслеживать завершение обработки запроса. Может быть использован как для реализации режима минимизации времени выполнения одного запроса (latency mode), так и для режима повышения пропускной способности (throughput mode)

3. Запуск вывода нейронной сети (2)

- ***Синхронный интерфейс вывода нейронной сети***
  - Для запуска синхронного вывода нейронной сети необходимо установить входной тензор как вход нейронной сети и вызвать метод **`infer()`** у объекта нейронной сети, загруженной в плагин на исполнение

```
input_layer = exec_model.input(0)
output_layer = exec_model.output(0)

n, c, h, w = input_layer.shape

# Load, transpose, expand operations
blob = prepare_image(imagePath, h, w)

# Execute
request = self.exec_model.create_infer_request()
request.infer(inputs={input_layer.any_name: blob})
result = request.get_output_tensor(output_layer.index).data
```

3. Запуск вывода нейронной сети (3)

- ***Асинхронный интерфейс вывода нейронной сети***
  - Для асинхронного запуска вывода нейронной сети необходимо установить входной тензор как вход нейронной сети, вызвать метод **`start_async()`** у объекта нейронной сети, загруженной в плагин на исполнение, и дождаться завершения запроса, чтобы получить выход модели
  - Отследить завершение запроса можно двумя способами:
    - Использование функции **`wait()`** для проверки статуса завершения запроса или ожидания завершения запроса
    - Создание функции обратного вызова (callback), которая будет вызвана после завершения запроса

3. Запуск вывода нейронной сети (4)

- ***Асинхронный интерфейс вывода нейронной сети***
  - Пример кода для запуска трех запросов и отслеживания их завершения с помощью функции **`wait_all()`**

```
from openvino.runtime import AsyncInferQueue, Tensor

infer_queue = AsyncInferQueue(compiled_model, num_request)
blobs = [blob1, blob2, blob3] # Images for independent requests
# Start async requests
for request_id in range(len(blobs)):
    idle_id = infer_queue.get_idle_request_id()
    infer_queue[idle_id].set_tensor(
        model_input.get_any_name(), Tensor(blobs[request_id])))
    infer_queue.start_async()
# Wait for completing requests
infer_queue.wait_all()

# Copy results
list = [copy(exec_net.requests[request_id].outputs)
    for request_id in range(len(blobs))]
```

3. Запуск вывода нейронной сети (5)

- Для одновременного запуска нескольких запросов на CPU необходимо при загрузке сети указать количество запросов, которые могут одновременно выполняться

```
cpu_throughput = {'CPU_THROUGHPUT_STREAMS': 'CPU_THROUGHPUT_AUTO'}
core.set_property('CPU', cpu_throughput)
```

- Получить доступное количество запросов можно с использованием следующей команды:

```
requests_number = len(exec_net.requests)
```

- Если количество входных пачек изображений для обработки больше, чем доступное количество запросов, то необходимо реализовать очередь, и направлять необработанные пачки из очереди в освобождающиеся запросы

4. Обработка результата

- Для обработки результата необходимо знать и понимать смысл и формат выходных тензоров запускаемой модели

- Выходные тензора варьируются в зависимости от задачи и модели

- Для публичных моделей, входящих в состав Open Model Zoo и решающих задачу классификации изображений на наборе данных ImageNet, как правило, выходной тензор имеет размерность $[B \times 1000]$, где B – размер пачки обрабатываемых изображений, а 1000 соответствует количеству категорий изображений

Запуск вывода на различных устройствах

- Для вывода моделей на Intel GPU измените параметр ***device_name*** на устройство, которое хотите использовать

```
exec_net = core.compile_model(model = model, device_name = 'GPU')
```

- Для вывода моделей с помощью мультидевайсного плагина установите приоритет использования устройств:

```
core.set_property({{"MULTI_DEVICE_PRIORITIES", "GPU,CPU"}},
                  "MULTI");
exec_net = core.compile_model(model = model,
                              device_name = 'MULTI:CPU,GPU')
```

- Для гетерогенного режима:

```
exec_net = core.compile_model(network = net,
                              device_name = 'HETERO:FPGA,CPU')
```

МОДУЛЬ DNN БИБЛИОТЕКИ OPENCV

Модуль DNN библиотеки OpenCV

- Модуль DNN библиотеки OpenCV поддерживает исполнение глубоких нейросетевых моделей на различном оборудовании, включая процессоры ARM-архитектуры

- Модуль DNN появился в OpenCV, начиная с версии 3.3

- OpenCV поддерживает загрузку моделей в формате Caffe, TensorFlow, Darknet, ONNX
- Модели в форматах MXNet, Pytorch и CNTK поддерживаются посредством конвертации в формат ONNX

- Документация к модулю DNN [https://docs.opencv.org/master/d2/d58/tutorial_table_of_content_dnn.html]

Бэкенды модуля DNN

- OpenCV поддерживает несколько ***бэкендов*** для вывода нейронных сетей:
  - OpenCV (самый простой)
  - Inference Engine (самый производительный, с богатыми возможностями)
  - Halide [https://halide-lang.org]. Halide – язык программирования, который разработан для создания производительных программ обработки изображений и массивов

Устройства исполнения для модуля DNN

- Параметр, описывающий тип устройства, на котором будет запускаться сеть, и формат запуска, называется ***целевым устройством*** (target)
- Модуль DNN поддерживает следующие целевые устройства с разными бэкендами:
  - Устройства CPU – OpenCV, Inference Engine, Halide
  - Устройства OpenCL – OpenCV, Inference Engine, Halide
  - Устройства OpenCL с половинной точностью весов – OpenCV, Inference Engine
  - Intel Movidius Neural Compute Stick – Inference Engine
  - Устройства Intel FPGA – Inference Engine

Программный интерфейс OpenCV

Метод	Назначение
`readNet(model)` или `readNet(model, config)`	Читает глубокую модель из файла (файлов)
`blobFromImage(image, scalefactor = 1.0, size, mean, swapRB = True)`	Выполняет предобработку изображения и конвертирует изображение в формат, используемый модулем DNN OpenCV (необходимо, так как интерфейс OpenCV не поддерживает матрицы с dimension>2)
`setInput(blob)`	Установка изображения на вход модели
`forward()`	Запуск инференса

Общая схема реализации вывода с использованием модуля DNN

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

1. Загрузка модели (1)

- Модель, как правило, состоит из одного или двух файлов
- Чтение модели происходит при помощи функции **`readNet`**, параметры которой – путь или два пути до модели, в любой последовательности
- Пример загрузки модели в формате библиотеки Caffe и установки бэкенда и целевого устройства посредством вызова методов **`setPreferableBackend`** и **`setPreferableTarget`** у объекта модели:

```
model = "deploy.prototxt"
weights = "bvlc_alexnet.caffemodel"

net = cv2.dnn.readNet(model, config)
net.setPreferableBackend(backend)
net.setPreferableTarget(target)
```

1. Загрузка модели (2)

- Возможные значения параметра бэкенда:

```
backend = cv2.dnn.DNN_BACKEND_DEFAULT
backend = cv2.dnn.DNN_BACKEND_HALIDE
backend = cv2.dnn.DNN_BACKEND_INFERENCE_ENGINE
backend = cv2.dnn.DNN_BACKEND_OPENCV
```

- Возможные значения параметра устройства исполнения:

```
target = cv2.dnn.DNN_TARGET_CPU
target = cv2.dnn.DNN_TARGET_OPENCL
target = cv2.dnn.DNN_TARGET_OPENCL_FP16
target = cv2.dnn.DNN_TARGET_MYRIAD
```

2. Загрузка изображения

- Загрузка изображения происходит при помощи функции **`imread`**, параметр – путь до изображения

```
image = cv2.imread(imagePath)
```

3. Конвертация изображения в формат входа нейронной сети (1)

- Как правило, на вход нейронной сети подается тензор размера $[B \times C \times H \times W]$
  - B – количество изображений
  - C – количество каналов в изображении
  - H – высота изображения
  - W – ширина изображения

- Если изображения загружены средствами библиотеки OpenCV, то необходимо перевести тензор из формата **{BGRBGR…}** в формат **{RRR…GGG…BBB…}**, и изменить его размер в соответствии с размером входа модели

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

- Для конвертации одного изображения используется функция **`blobFromImage`**
- Пример конвертации изображения в формат входа нейронной сети приведен ниже

```
scalefactor = 1.0
# mean intensity
mean = (104, 117, 123)
# input size
size = (224, 224)

blob = cv2.dnn.blobFromImage(image, scalefactor = 1.0, size,
    mean, swapRB = True)
```

4. Запуск вывода нейронной сети

- Для запуска вывода нейронной сети необходимо установить входной тензор как вход нейронной сети, и вызвать метод **`forward()`** у объекта загруженной модели

```
net.setInput(blob)
preds = net.forward()
```

5. Обработка результата

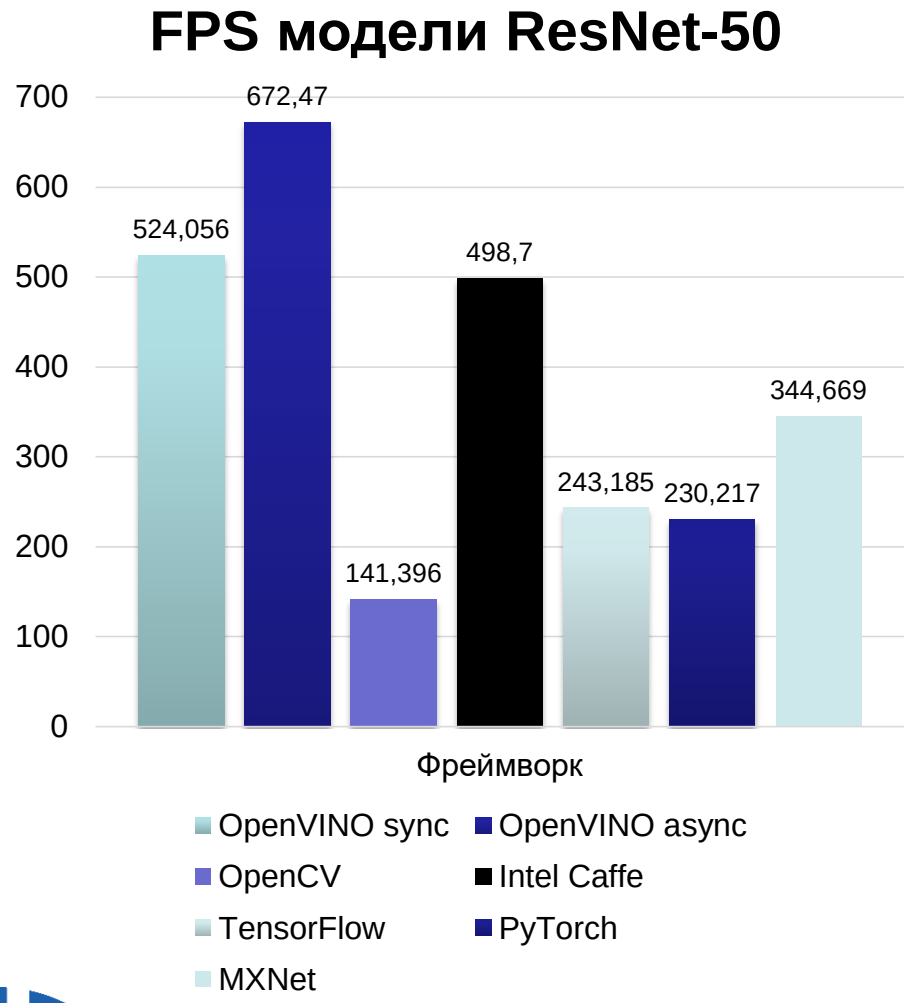
- Для обработки результата необходимо знать и понимать смысл и формат выходных тензоров запускаемой модели
- Для публичных моделей, входящих в состав Open Model Zoo и решающих задачу классификации изображений на наборе ImageNet, как правило, выходной тензор имеет размерность $[B \times 1000]$, где B – размер пачки обрабатываемых изображений, а 1000 соответствует количеству категорий изображений

```
# output shape [1, 1000] for one input image
prob = preds[0]
classid = np.argmax(prob)
classprob = np.max(prob)
print('Class {}, probability {}'.format(classid, classprob))
```

Сравнение производительности вывода (1)

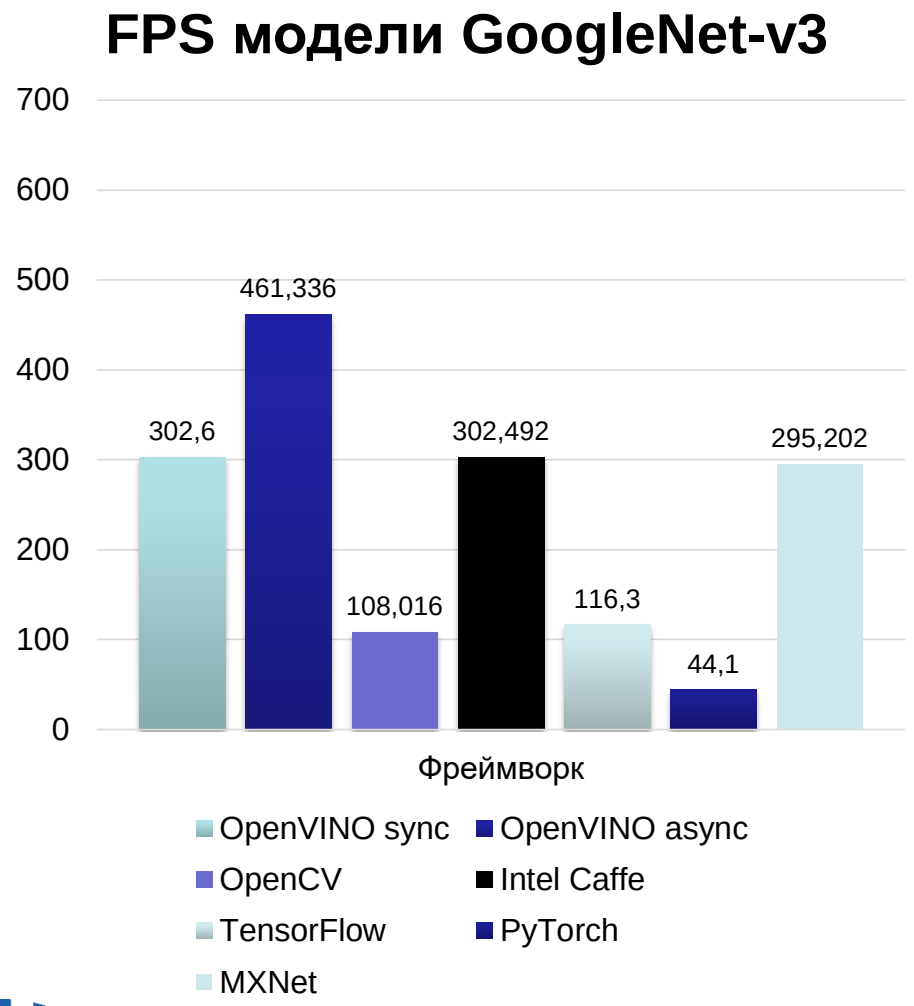
- Входные данные эксперимента:
  - Оборудование: 2x Intel Xeon Platinum 8260L 2.4GHz (2 процессора по 24 ядра – 48 ядер 96 потоков)
  - Обрабатывается 12228 изображений
  - Вычисляется среднее количество кадров, обрабатываемых за секунду (FPS)
  - Выбирается лучшее сочетание количества потоков и размера пачки для получения наибольшей производительности
  - Для режима повышения пропускной способности в OpenVINO используется 24 нити (streams) и 24 запроса (requests)

Сравнение производительности вывода (2)



Фреймворк	Количество потоков	Размер пачки
OpenVINO sync	48	128
OpenVINO async	48	16
OpenCV	96	512
Intel Caffe	48	512
TensorFlow	96	512
PyTorch	48	96
MXNet	48	128

Сравнение производительности вывода (3)



Фреймворк	Количество потоков	Размер пачки
OpenVINO sync	48	96
OpenVINO async	48	48
OpenCV	48	512
Intel Caffe	48	96
TensorFlow	96	384
PyTorch	48	192
MXNet	48	192

Измерение качества модели с помощью Accuracy Checker

- Модуль Accuracy Checker позволяет проверить качество работы модели после конвертации в формат OpenVINO

Модель	Опубликованные метрики качества		Фреймворк	Качество работы натренированных моделей	
	Top-1	Top-5		Top-1	Top-5
ResNet-50	75.30%	92.20%	Caffe	75.17%	92.20%
			OpenVINO	75.17%	92.20%
			OpenVINO (int8)	75.20%	92.16%
GoogleNet-v3	76.60%	-	TensorFlow	77.89%	93.80%
			OpenVINO	77.89%	93.80%
			OpenVINO (int8)	77.85%	93.75%

Заключение

- Рассмотрен компонентный состав инструмента Intel Distribution of OpenVINO Toolkit
- Описаны возможные способы реализации вывода глубоких моделей средствами компонента Inference Engine и библиотеки OpenCV
- В дальнейшем при решении практических заданий курса предлагается использовать один из предложенных способов
- Описание последовательности решения практических заданий авторами курса осуществляется на базе компонента Inference Engine

Основная литература

- Intel Distribution of OpenVINO Toolkit [https://software.intel.com/en-us/openvino-toolkit].
- 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].
- OpenCV [https://opencv.org].
- Open Model Zoo [https://github.com/opencv/open_model_zoo].

Авторский коллектив (1)

- **Турлапов Вадим Евгеньевич**
  д.т.н., профессор кафедры МОСТ ИИТММ ННГУ
  vadim.turlapov@itmm.unn.ru
- **Васильев Евгений Павлович**
  преподаватель кафедры МОСТ ИИТММ ННГУ
  evgeny.vasiliev@itmm.unn.ru
- **Гетманская Александра Александровна**
  преподаватель кафедры МОСТ ИИТММ ННГУ
  getmanskaya.alexandra@gmail.com
- **Кустикова Валентина Дмитриевна**
  к.т.н., доцент каф. МОСТ ИИТММ ННГУ
  valentina.kustikova@itmm.unn.ru

Авторский коллектив (2)

- **Золотых Николай Юрьевич**
  д.т.н., доцент кафедры АГДМ ИИТММ ННГУ
  nikolai.zolotykh@gmail.com
- **Носова Светлана Александровна**
  преподаватель кафедры МОСТ ИИТММ ННГУ
  nosova.sv.a@gmail.com
- **Тужилкина Анастасия Андреевна**
  магистрант ИИТММ ННГУ
  tan98-52@yandex.ru