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

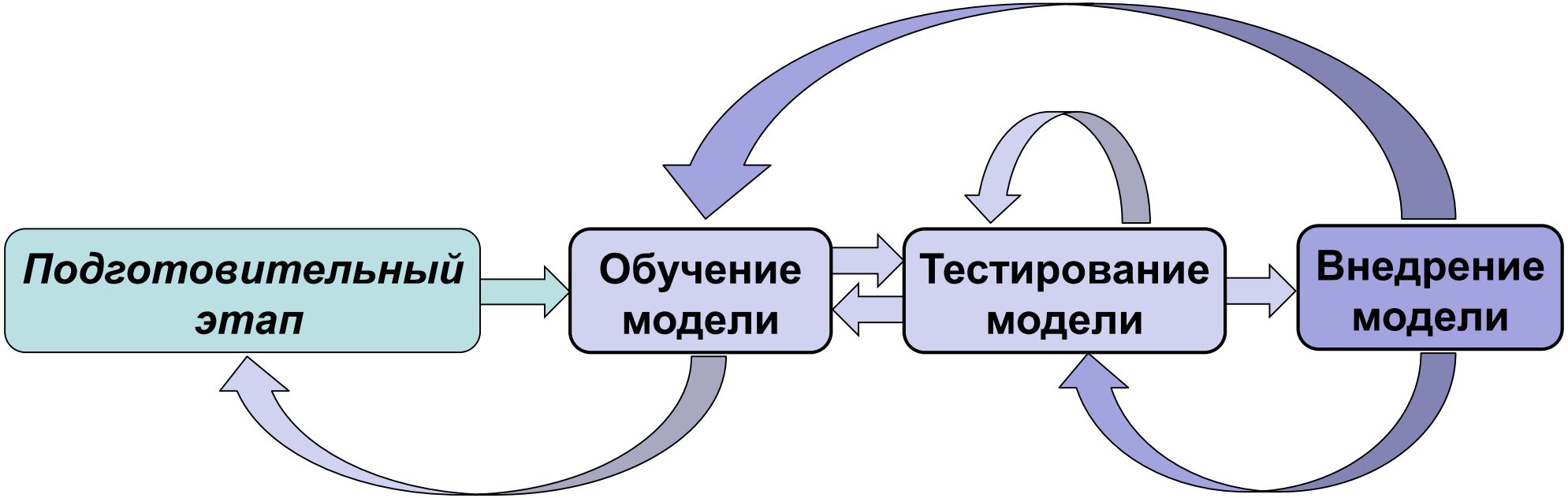
# Методика анализа производительности вывода глубоких нейронных сетей на примере задачи классификации изображений

М.Р. Алибеков[1], Н.Е. Березина[2], И.Б. Вихрев[2], Е.П. Васильев[1], Ю.Д. Камелина[2], В.Д. Кустикова[1], З.А. Маслова[2], И.С. Мухин[1], А.К. Сидорова[1], В.Н. Сучков[1]

[1] Нижегородский государственный университет им. Н.И. Лобачевского,
[2] ООО «Ядро Центр Исследований и Разработки»

# Введение

- Цикл решения задач с использованием глубоких моделей:




**Обзор литературы**

**Подготовка данных**

**Подготовка тестовой инфраструктуры**

**Разработка топологии**

**Обучение модели**

**Верификация модели**

**Определение качества решения задачи**

**Анализ производительности**

**Оптимизация модели**

**Многократный вывод**

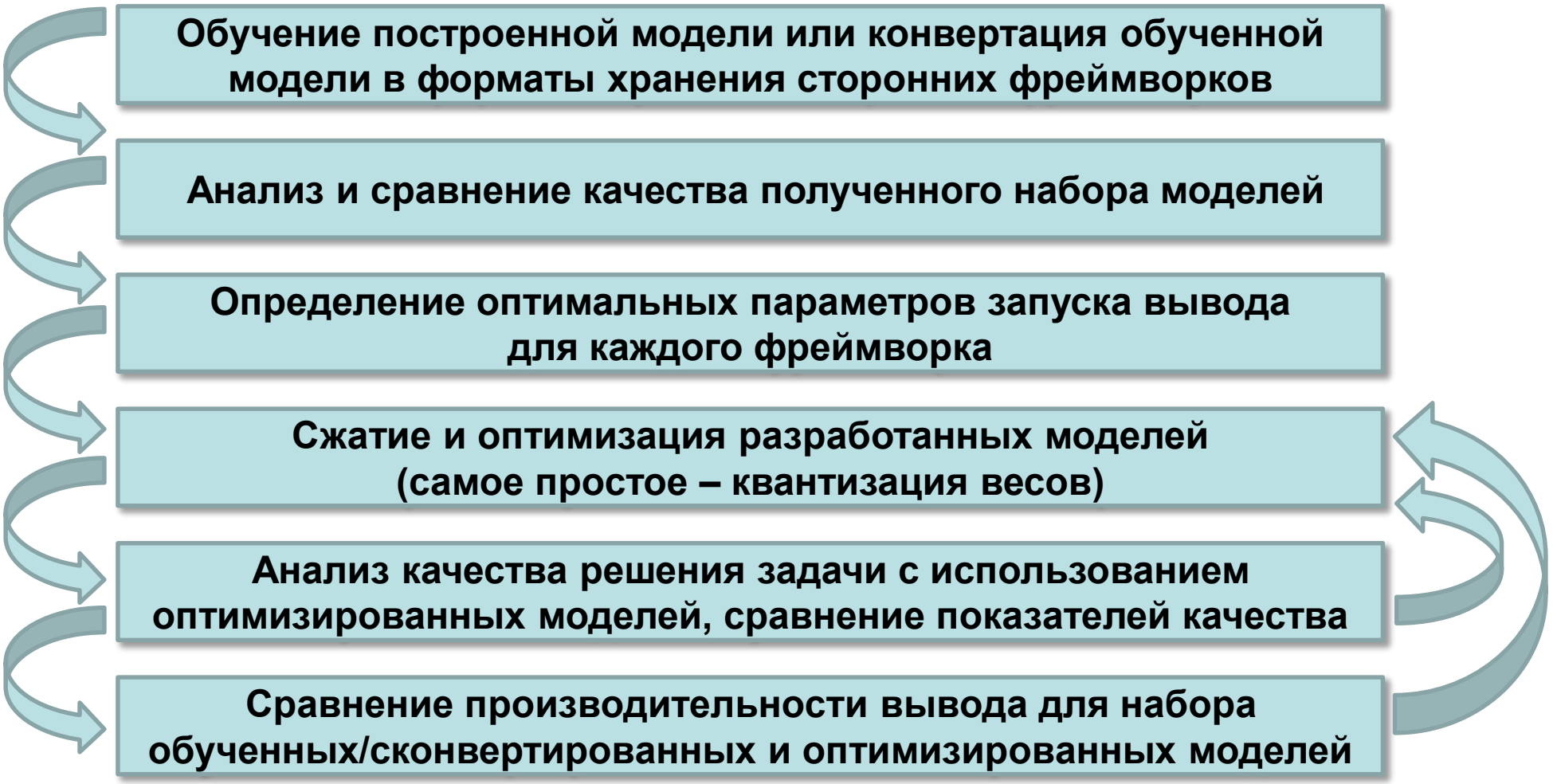
# Цели и задачи

- ***Цель*** – обобщить и автоматизировать схему анализа производительности вывода глубоких моделей
- ***Задачи:***
  - Разработать методику анализа и сравнения производительности вывода
  - Разработать программную систему для поддержки процесса анализа производительности вывода
- ***Отличия от существующих решений:***
  - Отсутствие ориентированности на определенное аппаратное обеспечение
  - Поддержка большого числа известных нейронных сетей (!возможность запуска вывода собственных моделей)
  - Регулярная публикация результатов производительности вывода на доступных аппаратных конфигурациях

# Постановка тестовой задачи

- Задача классификации изображений состоит в том, чтобы поставить в соответствие изображению класс объектов, содержащихся на этом изображении
- ***Входные данные:***
  - Трехканальное изображение $I$ в формате RGB, которое представляется трехмерной матрицей с пространственными размерами $w$ и $h$ (ширина и высота)
  - Каждый элемент – интенсивность пикселя в диапазоне от 0 до 255 (или от 0 до 1, если выполнена нормировка)
- ***Выходные данные:***
  - Вектор вещественных значений
  - Длина вектора = число категорий изображений
  - Каждый элемент вектора – достоверность принадлежности изображения определенному классу

# Схема анализа и сравнения производительности вывода



# Программная система Deep Learning Inference Benchmark

- Открытый код [https://github.com/itlab-vision/dl-benchmark]
- Поддерживаемые фреймворки:
  - Intel Distribution of OpenVINO Toolkit (C++ и Python APIs)
  - Intel Optimization for Caffe (Python API)
  - Intel Optimizations for TensorFlow (Python API)
  - TensorFlow Lite (C++ и Python APIs)
  - MXNet (Python Gluon API)
  - OpenCV DNN (C++ и Python APIs)
  - *ONNX Runtime (C++ и Python APIs)*
  - *PyTorch (C++ и Python APIs)*
- Возможность одновременного проведения экспериментов на нескольких узлах с разным составом
- Запуск возможен на узле или внутри docker-контейнера

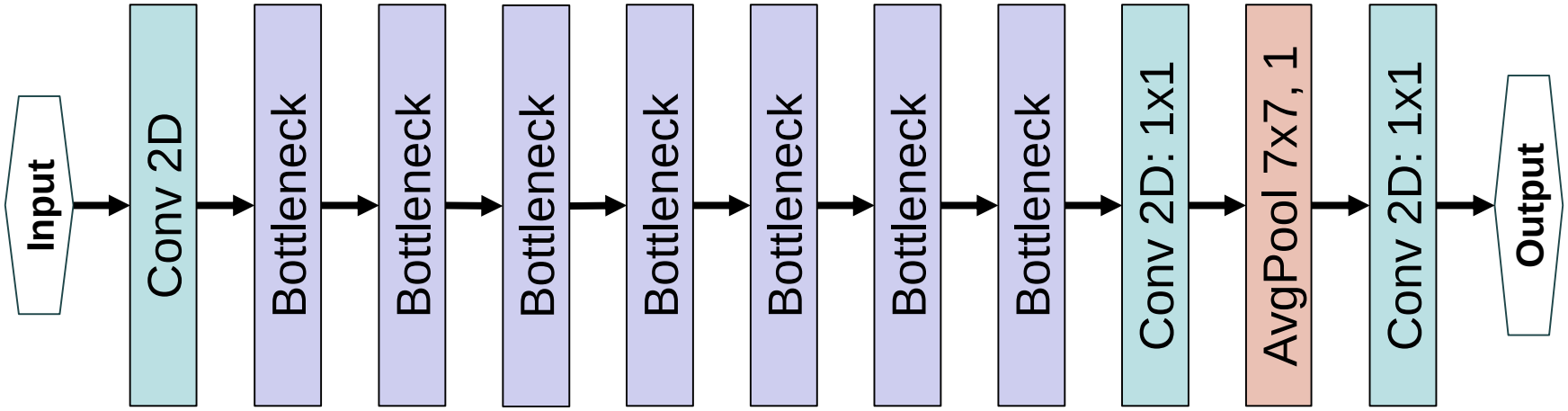
# Фреймворки для вывода глубоких моделей

- Фреймворки для демонстрации методики анализа производительности вывода:
  - Intel Distribution of OpenVINO Toolkit (OpenVINO)
  - TensorFlow
  - TensorFlow Lite
  - MXNet
  - OpenCV DNN

- ***Примечание:*** используется Python API

# Тестовые модели

- MobileNetV2:
  - Классификационная модель (1 000 категорий, ImageNet)
  - Сверточная нейронная сеть (53 слоя, включая сверточные слои, функции активации и слои нормализации по пачке)



# Тестовые данные

- Анализ качества:
  - Валидационная выборка из набора данных ImageNet (50 000 изображений) [https://www.image-net.org]

- Анализ производительности:
  - Подмножество валидационной выборки, состоящее из 320 изображений

# Тестовая инфраструктура

| | |
|---|---|
| **CPU** | Intel® Core™ i7-8700 3.20GHz (6 ядер и 12 потоков) |
| **GPU** | Intel® Gen9 HD Graphics (iGPU) |
| **ОП** | 64 ГБ |
| **ОС** | Ubuntu 20.04.4 LTS |
| **Библиотеки** | Intel Distribution of OpenVINO Toolkit 2022.3<br>TensorFlow 2.9.3<br>TensorFlow Lite 2.9.3 (part of TensorFlow)<br>MXNet 1.9.1<br>OpenCV 4.7 |

# Показатели качества классификации изображений

- $N$ – количество категорий изображений
- Метод строит вектор достоверностей $p^j = \left(p_1^j, p_2^j, \dots, p_N^j\right)$ для каждого изображения $I_j, j = \overline{1, S}$ в выборке, где $p_i^j$ – достоверность того, что изображение $I_j$ принадлежит классу $i$
- ***Точность top-K*** (top-K accuracy) определяется по формуле:

$$topK = \frac{\sum_{j=1}^{S} 1_{\left\{i_1^j, i_2^j, \dots, i_K^j\right\}}\left(l_j\right)}{S},$$

где $\left\{i_1^j, i_2^j, \dots, i_K^j\right\} \subseteq \{1,2, \dots, N\}$, а $p_{i_1^j}^j, p_{i_2^j}^j, \dots, p_{i_K^j}^j$ – $K$ наибольших достоверностей, $l_j$ – класс, которому принадлежит изображение $I_j$ согласно разметке, $1_{\left\{i_1^j, i_2^j, \dots, i_K^j\right\}}\left(l_j\right)$ – индикаторная функция

# Показатели производительности вывода

- Эксперимент:
  - Множество входных примеров разбивается на пачки
  - Прямой проход = решение задачи для пачки
  - Запросы на прямой проход по сети выполняются последовательно, следующий запрос выполняется после завершения предыдущего
  - Число запросов (итераций) = 1 000
  - Для каждого запроса измеряется продолжительность его выполнения
- ***Латентность*** (Latency) – медиана времен выполнения запросов
- ***Среднее количество кадров, обрабатываемых за секунду*** (Frames per Second, FPS) – отношение размера пачки изображений к латентности

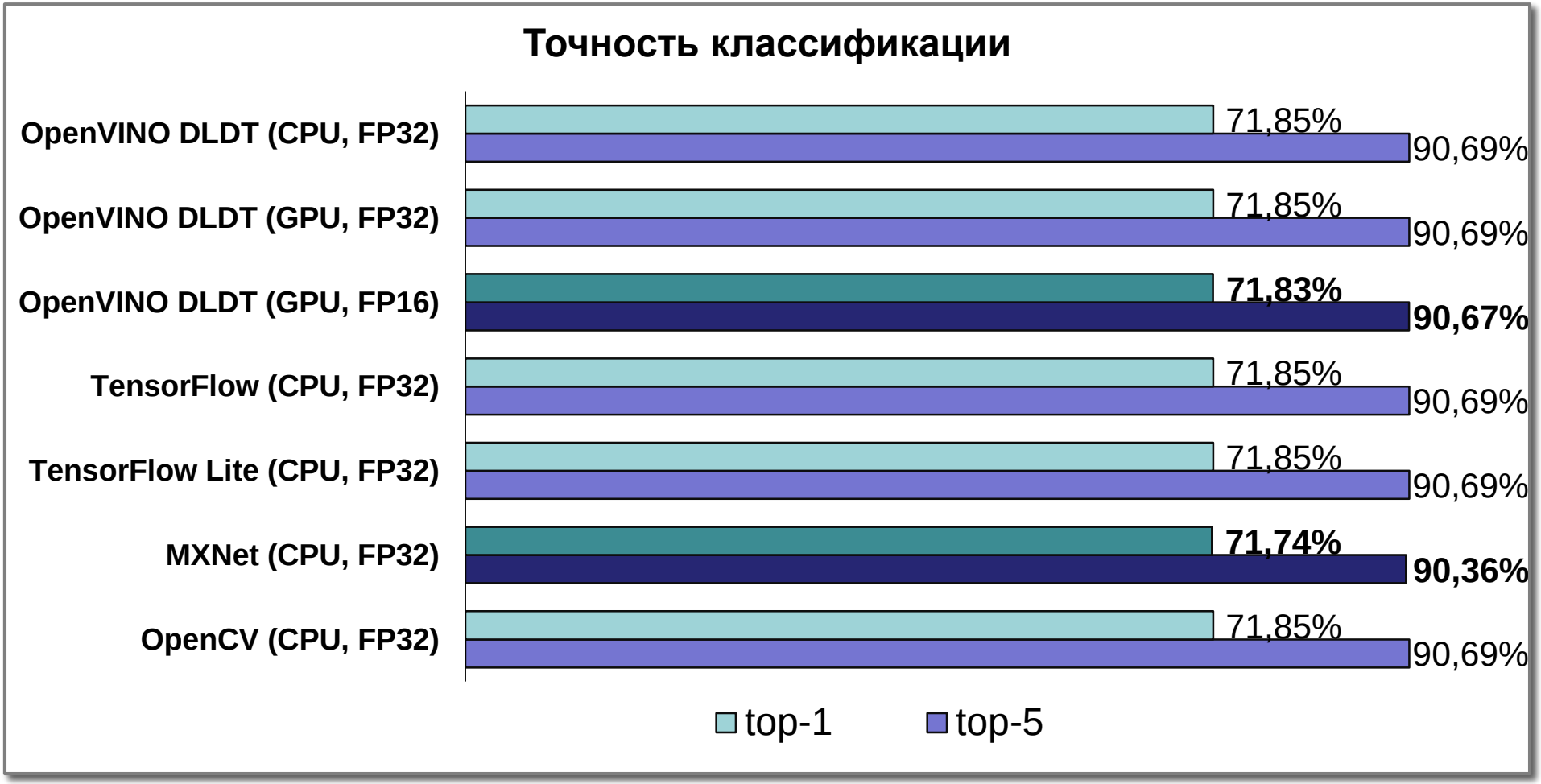
# Результаты экспериментов

## *1. Обучение и/или конвертирование модели*

- Исходная модель: MobileNetV2 в формате TensorFlow
- Intel Distribution of OpenVINO Toolkit:
  - Конвертируется в промежуточное представление с весами FP32 и FP16 с помощью omz_converter
- TensorFlow и OpenCV: модель подается в исходном виде
- TensorFlow Lite:
  - Конвертируется с помощью внутреннего инструмента в системе DLI
- MXNet:
  - Используется модель MobileNetV2 из GluonCV Model Zoo
- ***Примечание:*** модель в форматах TensorFlow и MXNet обучена и провалидирована на наборе данных ImageNET сторонними исследователями и выложена в открытый доступ

# Результаты экспериментов

## *2. Анализ и сравнение качества*




- Наибольшие отличия: top-1 – 0.11%, top-5 – 0.33%

# Результаты экспериментов

## *2. Анализ и сравнение качества*

*Более низкая точность весов*


*Модель обучена средствами MXNet (выбор начального приближения весов)*

ие отличия: top-1 – 0.11%, top-5 – 0.33%

# Результаты экспериментов

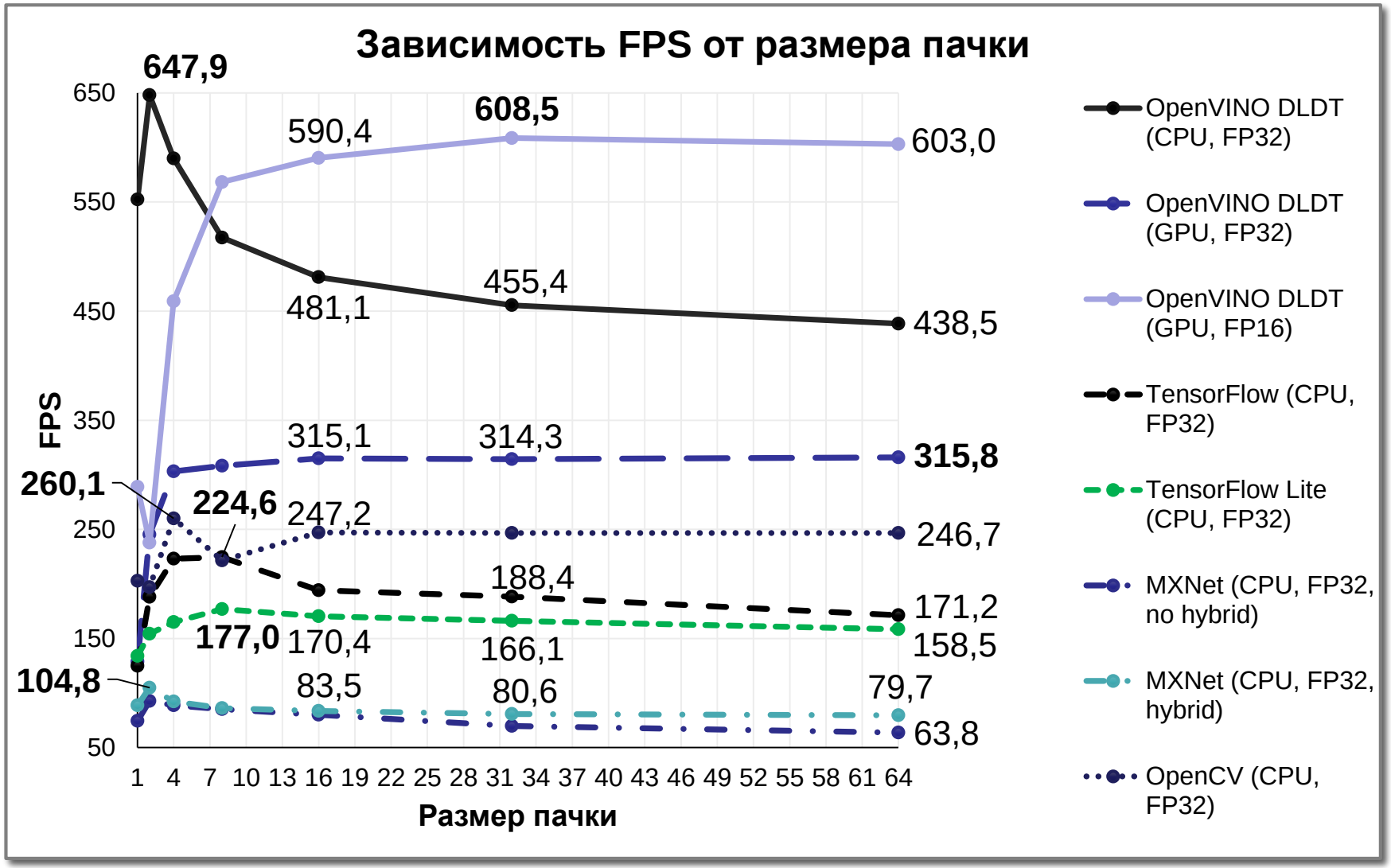
## *3. Определение оптимальных параметров запуска вывода, поиск оптимальной реализации…*

- Эксперименты запускаются с параметрами исполнения вывода по умолчанию (число потоков и прочие), чтобы выровнять сложность этого этапа для всех фреймворков

- ***Примечания:***
  - Для OpenVINO и TensorFlow приведена процедура подбора оптимальных параметров в ранее опубликованных работах
  - MXNet запускается без подключения библиотеки oneDNN (ранее MKL-DNN) без (no hybrid) и с (hybrid) использованием символических вычислений
  - Оптимальные параметры для запуска вывода зависят от архитектуры нейронной сети

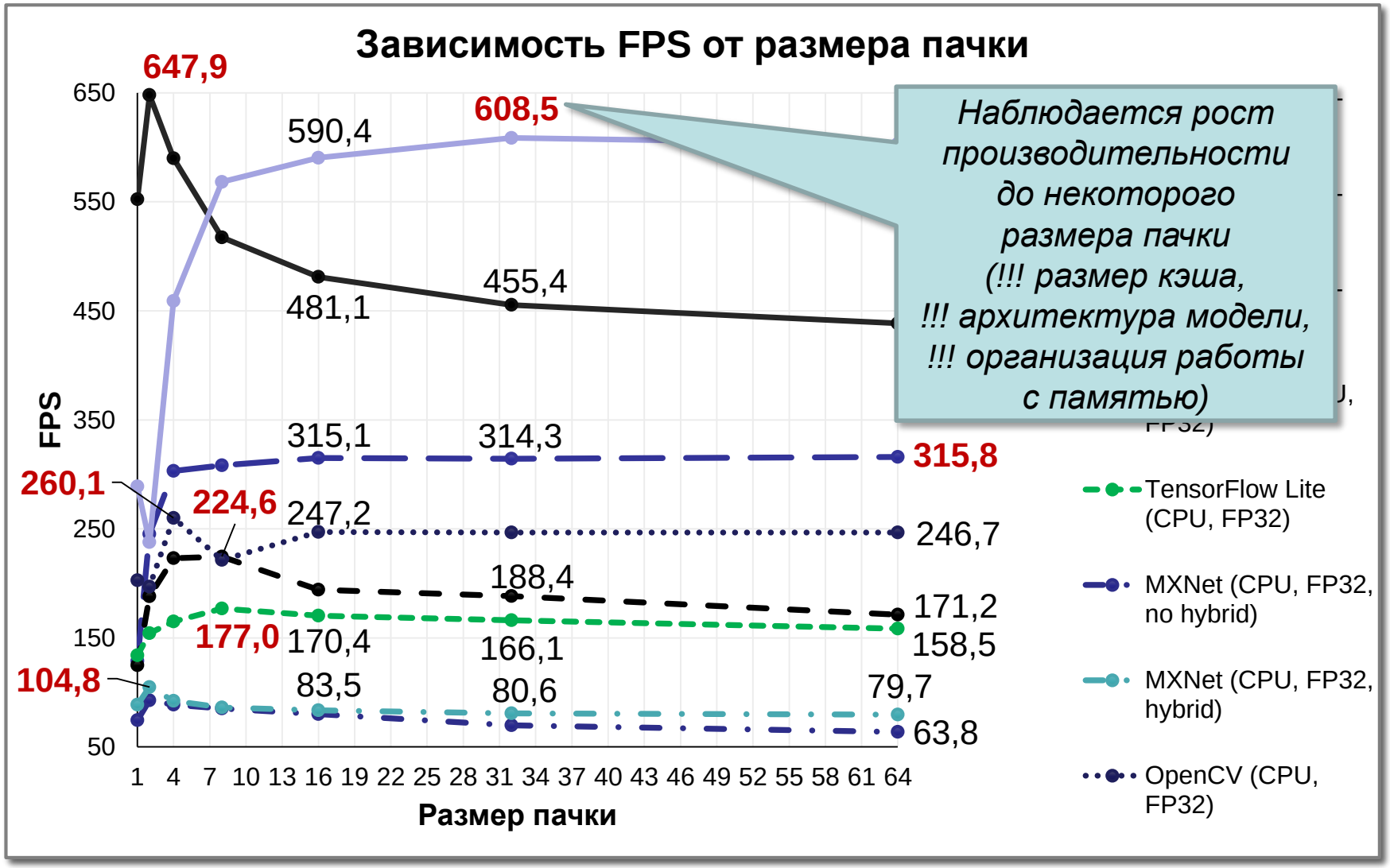
# Результаты экспериментов

## *3. Определение оптимальных параметров запуска вывода, поиск оптимальной реализации…*



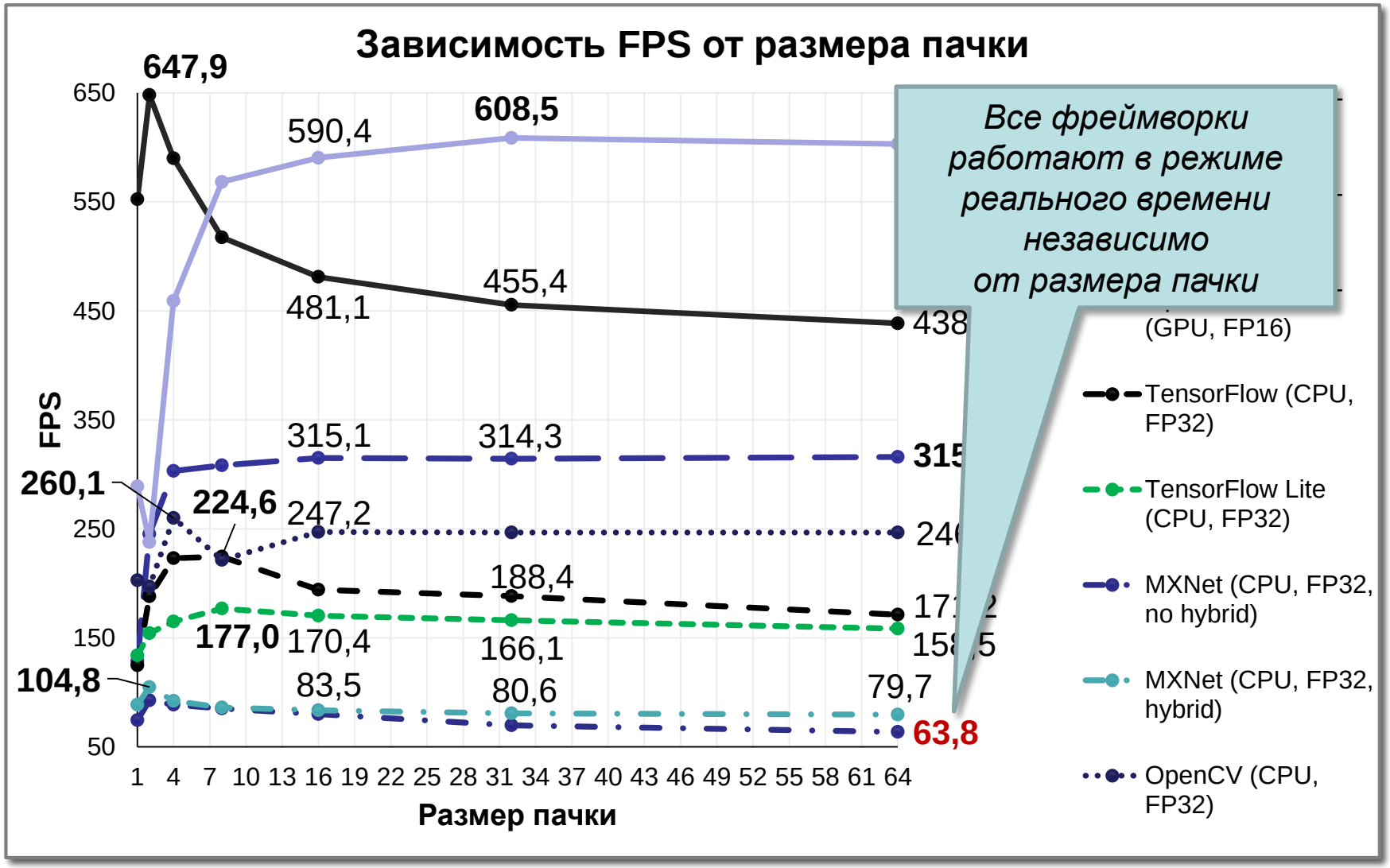
# Результаты экспериментов

## *3. Определение оптимальных параметров запуска вывода, поиск оптимальной реализации…*



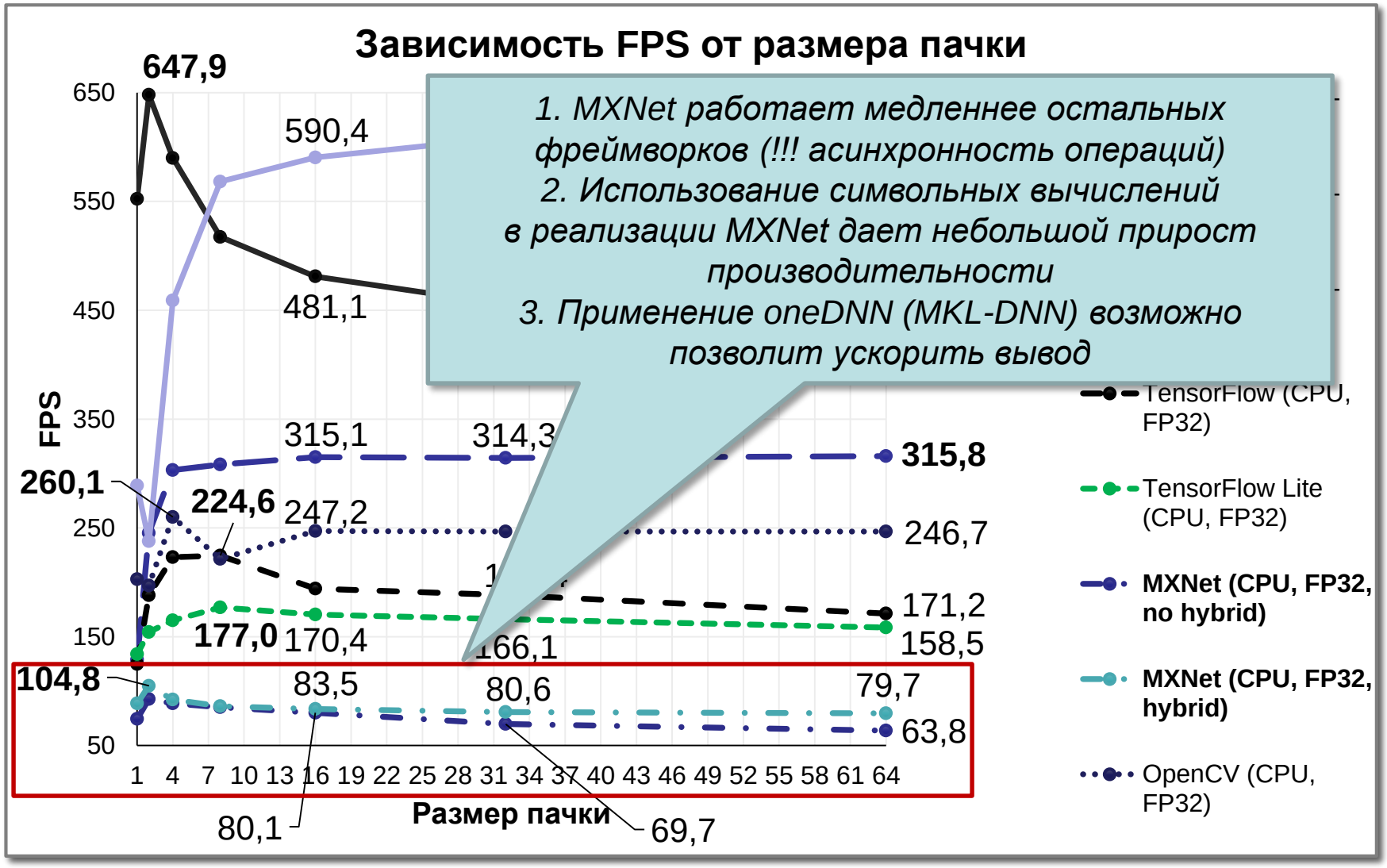
# Результаты экспериментов

## *3. Определение оптимальных параметров запуска вывода, поиск оптимальной реализации…*



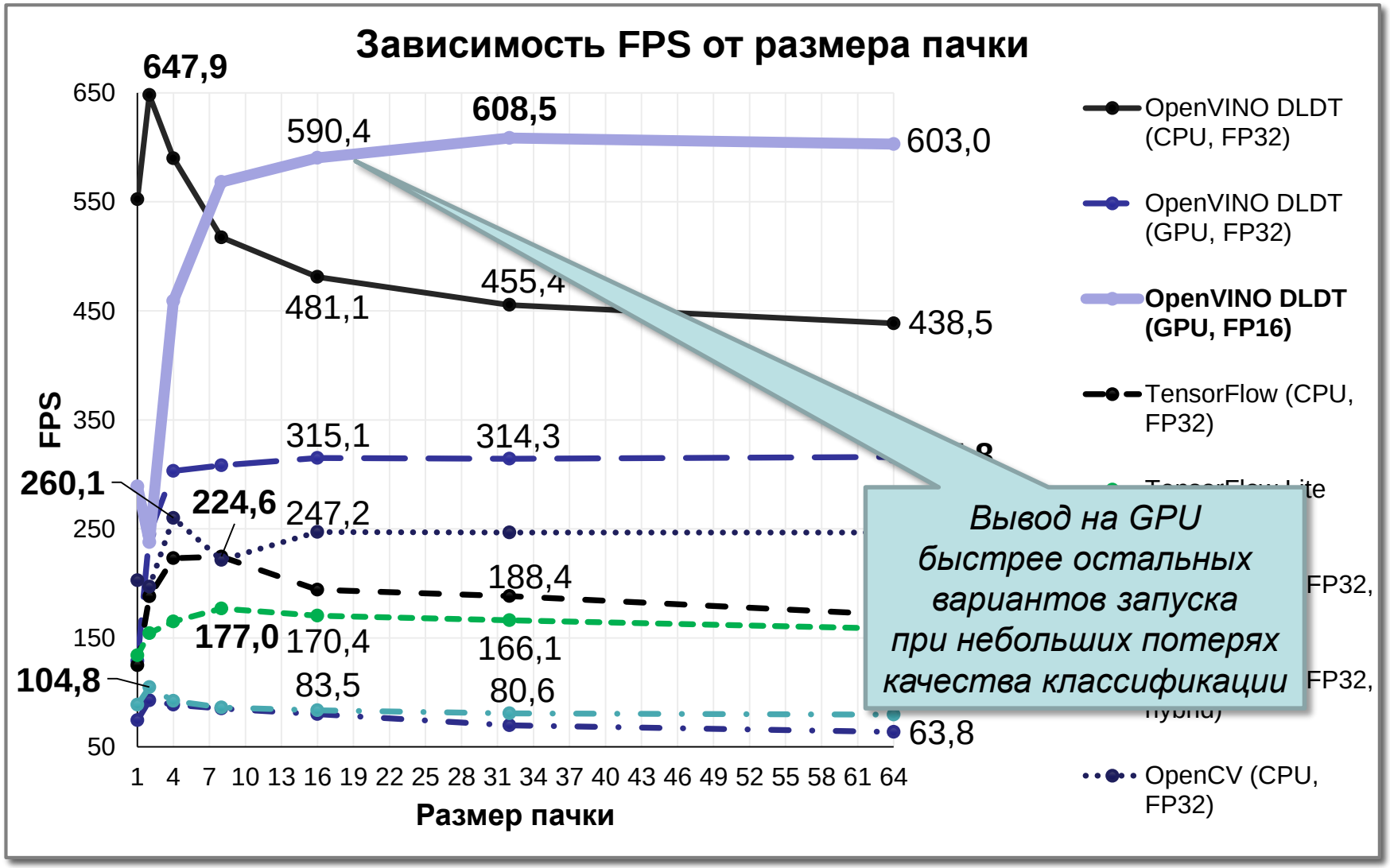
# Результаты экспериментов

## *3. Определение оптимальных параметров запуска вывода, поиск оптимальной реализации…*



# Результаты экспериментов

## *3. Определение оптимальных параметров запуска вывода, поиск оптимальной реализации…*



# Результаты экспериментов

## *3. Определение оптимальных параметров запуска вывода, поиск оптимальной реализации*

- ***Замечания:***
  - В процессе внедрения выбор фреймворка зависит от скорости вывода для фиксированного размера пачки данных на целевой аппаратуре
  - Размер пачки определяется тем, с какой скоростью поступают данные с устройства

# Результаты экспериментов

## *4. Сжатие и оптимизация моделей*

- ***Квантизация моделей*** – переход от формата весов FP32 или FP16 к INT8 или UINT8

- Intel Distribution of OpenVINO Toolkit
  - Модели с весами INT8 получены средствами Post-Training Optimization Tool в составе инструмента
- TensorFlow Lite
  - Модель с весами UINT8 имеется в открытом доступе

- ***Примечание:*** на практике имеет смысл выполнять квантизацию и запуск экспериментов для всех доступных фреймворков

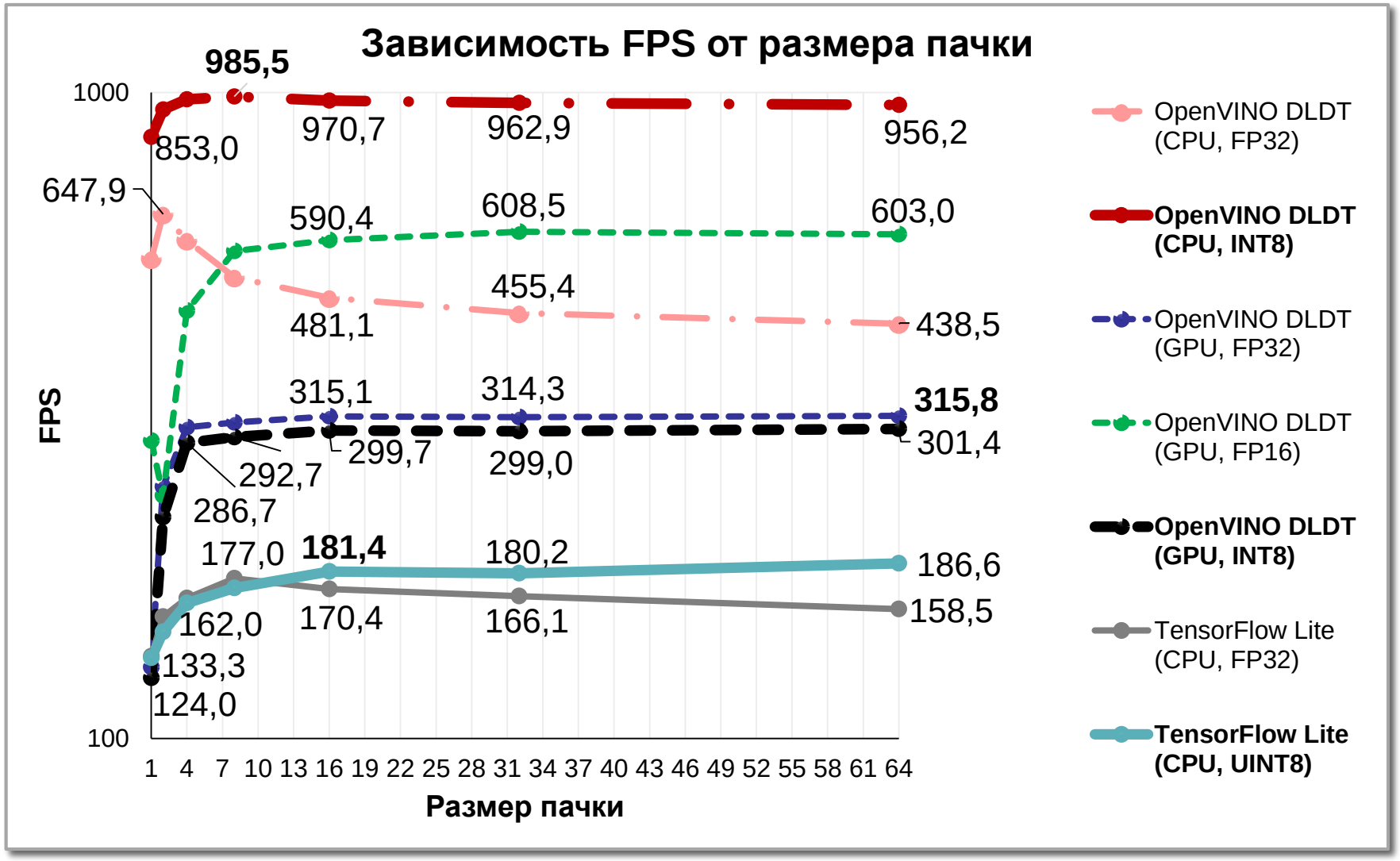
# Результаты экспериментов

## *5. Анализ качества оптимизированных моделей*


- Квантизация приводит к снижению качества менее, чем на 1%

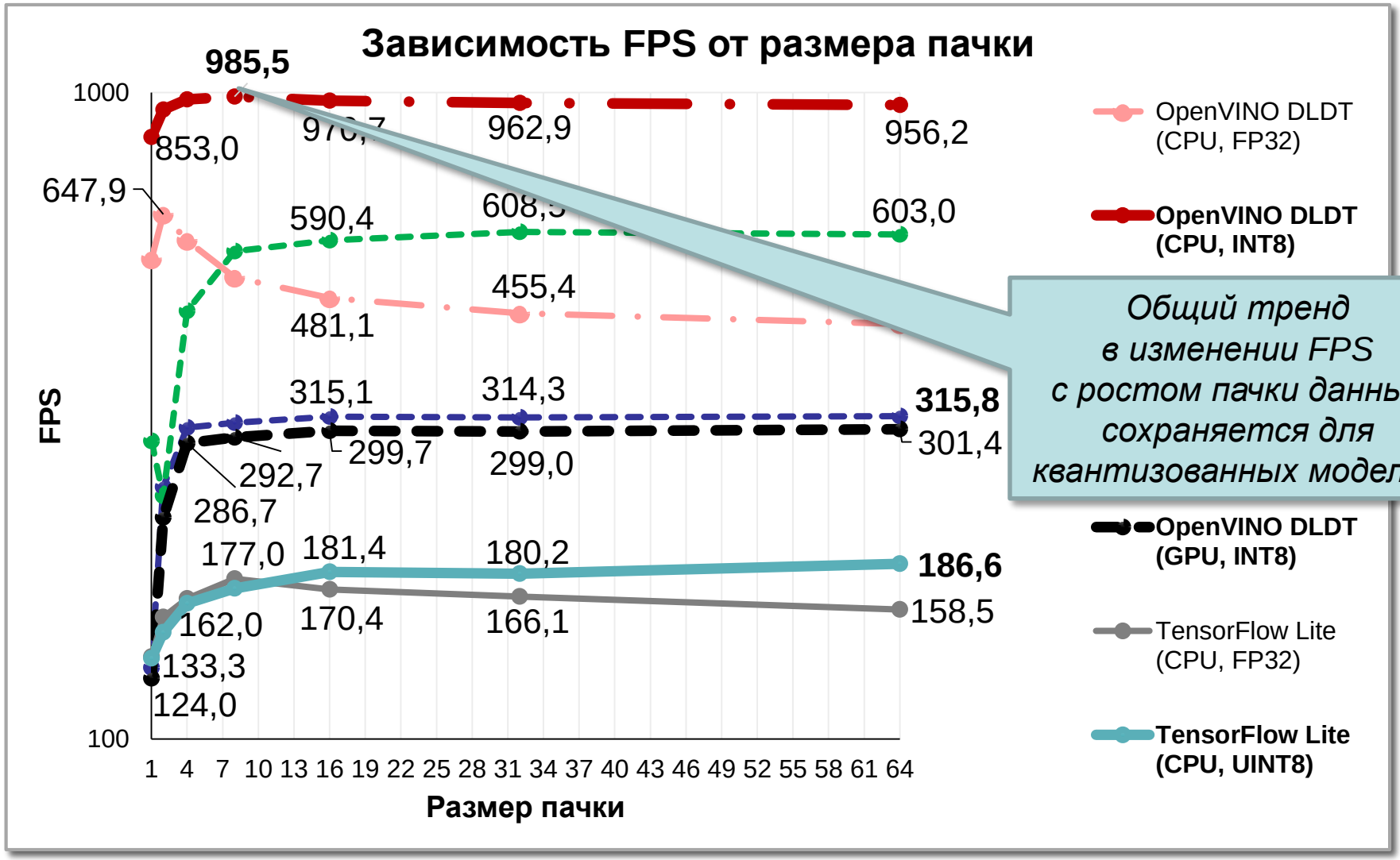
# Результаты экспериментов

## *6. Сравнение производительности вывода для обученных / сконвертированных и оптимизированных моделей…*



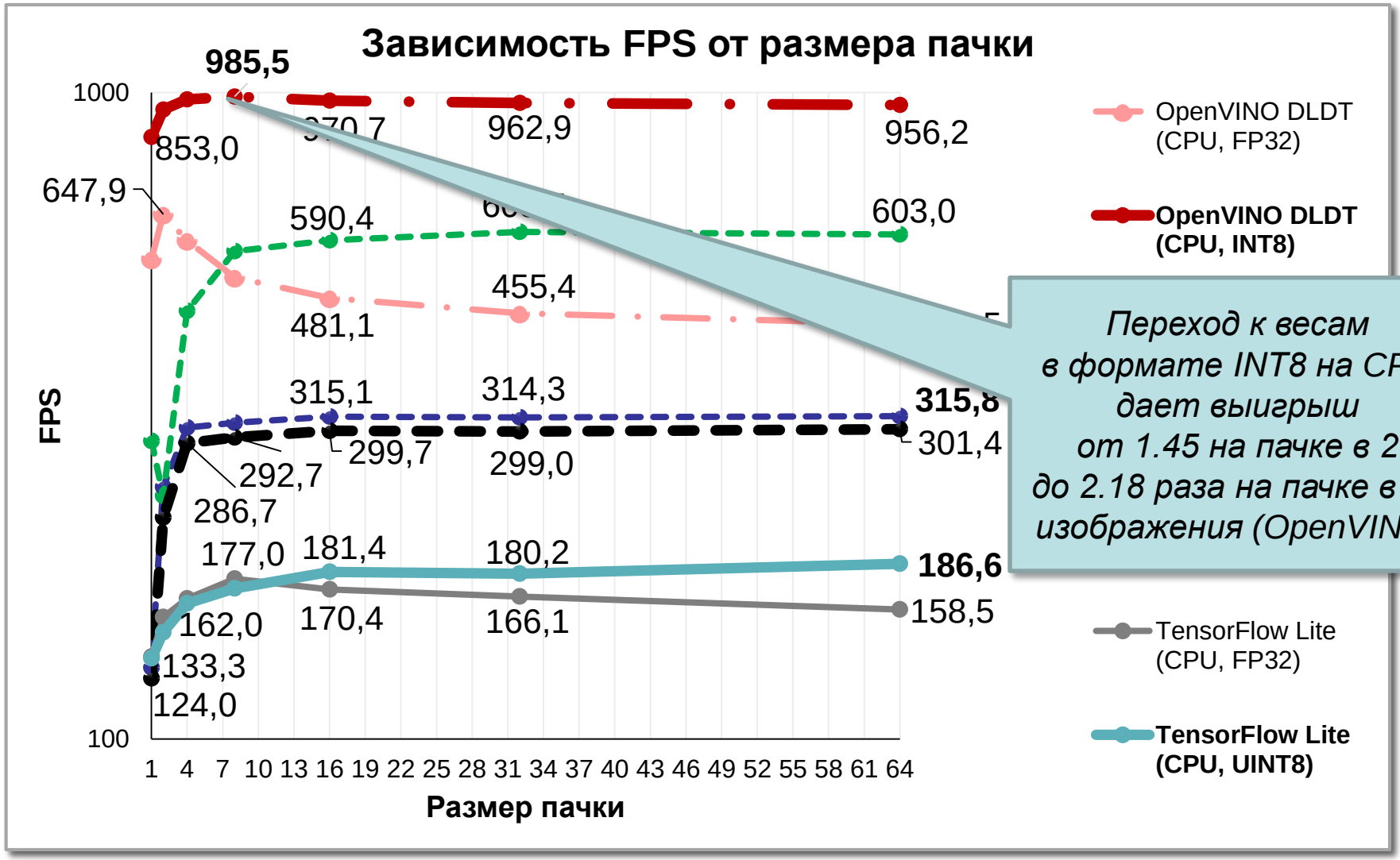
# Результаты экспериментов

## *6. Сравнение производительности вывода для обученных / сконвертированных и оптимизированных моделей...*



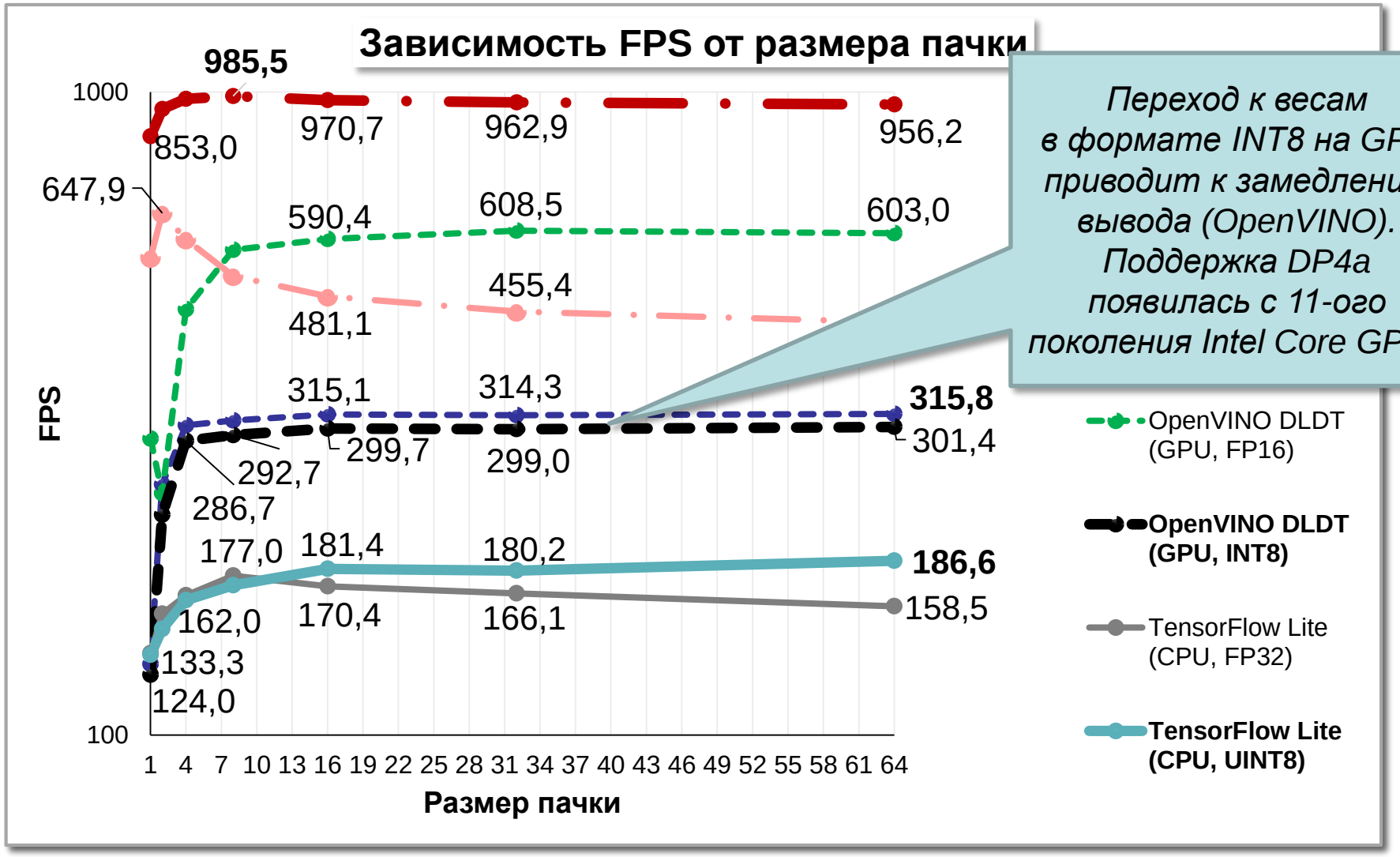
# Результаты экспериментов

## *6. Сравнение производительности вывода для обученных / сконвертированных и оптимизированных моделей...*



# Результаты экспериментов

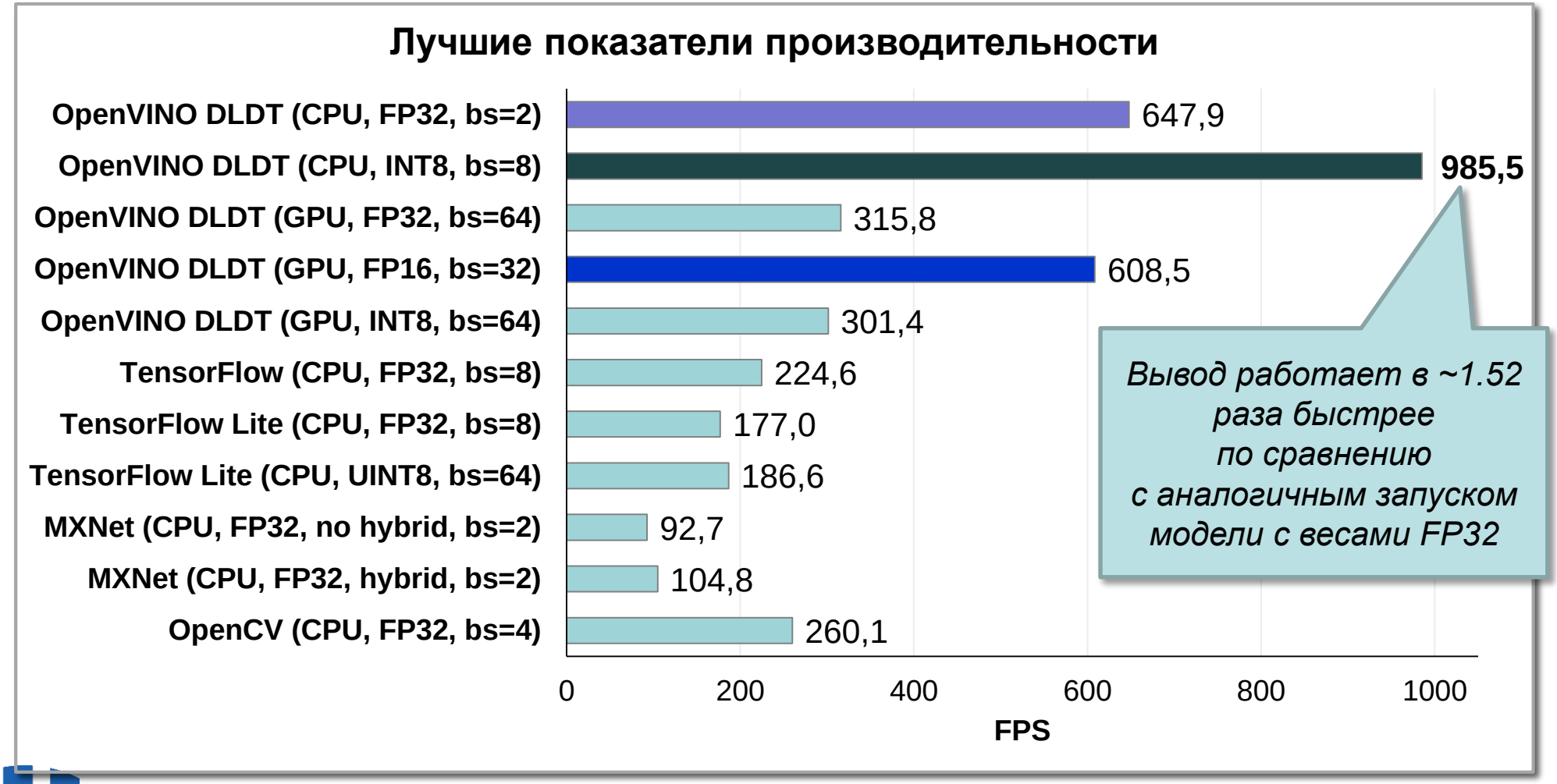
## *6. Сравнение производительности вывода для обученных / сконвертированных и оптимизированных моделей...*




*Переход к весам в формате INT8 на GPU приводит к замедлению вывода (OpenVINO). Поддержка DP4a появилась с 11-ого поколения Intel Core GPUs*

# Результаты экспериментов

## *6. Сравнение производительности вывода для обученных / сконвертированных и оптимизированных моделей…*



# Результаты экспериментов

## *6. Сравнение производительности вывода для обученных / сконвертированных и оптимизированных моделей…*

- ***Выводы:***
  - Если требуется достичь максимальной производительности, то следует запускать квантизованную модель средствами OpenVINO DLDT на CPU
  - Если необходимо освободить CPU для решения других задач, то вполне возможен запуск FP16-модели на Intel GPU

- ***Примечание:***
  - Полученные показатели производительности можно улучшить (тонкая настройка параметров исполнения вывода, оптимизация фреймворков под конкретную аппаратуру)

# Заключение

- В исследовании проработана методика анализа и сравнения производительности вывода
- Разработана программная система Deep Learning Inference Benchmark для поддержки сбора результатов и автоматизации процесса
- Практическое применение методики продемонстрировано на примере решения задачи классификации изображений с использованием модели MobileNetV2