2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

doi 10.26089/NumMet.v25r211 УДК 004.032.26; 004.048; 004.021

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

М. Р. Алибеков
Нижегородский государственный университет имени Н. И. Лобачевского, Нижний Новгород, Российская Федерация
ORCID: 0000-0002-9201-8878, e-mail: muradalibekov2000@gmail.com

Н. Е. Березина
ООО "Ядро Центр Исследований и Разработки", Нижний Новгород, Российская Федерация
ORCID: 0009-0003-3111-8829, e-mail: n.berezina@yadro.com

Е. П. Васильев
Нижегородский государственный университет имени Н. И. Лобачевского, Нижний Новгород, Российская Федерация
ORCID: 0000-0002-7949-1919, e-mail: evgeny.vasiliev@itmm.unn.ru

И. Б. Вихрев
ООО "Ядро Центр Исследований и Разработки", Нижний Новгород, Российская Федерация
ORCID: 0009-0007-8500-1561, e-mail: i.vikhrev@yadro.com

Ю. Д. Камелина
ООО "Ядро Центр Исследований и Разработки", Нижний Новгород, Российская Федерация
ORCID: 0009-0008-7660-3066, e-mail: y.kamelina@yadro.com

В. Д. Кустикова
Нижегородский государственный университет имени Н. И. Лобачевского, Нижний Новгород, Российская Федерация
ORCID: 0000-0002-6159-1145, e-mail: valentina.kustikova@itmm.unn.ru

З. А. Маслова
ООО "Ядро Центр Исследований и Разработки", Нижний Новгород, Российская Федерация
ORCID: 0009-0001-2587-0438, e-mail: z.maslova@yadro.com

И. С. Мухин
Нижегородский государственный университет имени Н. И. Лобачевского, Нижний Новгород, Российская Федерация
ORCID: 0009-0006-2558-688X, e-mail: ismukhin03@gmail.com

А. К. Сидорова
Нижегородский государственный университет имени Н. И. Лобачевского, Нижний Новгород, Российская Федерация
ORCID: 0009-0002-3081-9482, e-mail: alexa.sanechka@ya.ru

В. Н. Сучков
Нижегородский государственный университет имени Н. И. Лобачевского, Нижний Новгород, Российская Федерация
ORCID: 0009-0009-8246-7686, e-mail: suchkov.vladislavn@yandex.ru

Аннотация: Внедрение глубоких нейронных сетей требует анализа производительности этапа вывода на целевом аппаратном обеспечении. Результаты производительности позволяют принимать решение о возможности внедрения построенных моделей и/или необходимости их

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

предварительной оптимизации. В работе описана методика анализа и сравнения производительности вывода на примере решения задачи классификации изображений: конвертация обученной модели под разные фреймворки, анализ качества, определение оптимальных параметров запуска вывода, оптимизация модели и повторный анализ качества, анализ и сравнение производительности. Разработана система Deep Learning Inference Benchmark для поддержки цикла анализа производительности. Методика продемонстрирована на примере открытой модели MobileNetV2.

Ключевые слова: глубокое обучение, нейронные сети, вывод, производительность, MobileNetV2, Deep Learning Inference Benchmark.

Для цитирования: Алибеков М.Р., Березина Н.Е., Васильев Е.П., Вихрев И.Б., Камелина Ю.Д., Кустикова В.Д., Маслова З.А., Мухин И.С., Сидорова А.К., Сучков В.Н. Методика анализа производительности вывода глубоких нейронных сетей на примере задачи классификации изображений // Вычислительные методы и программирование. 2024. **25**, № 2. 127–141. doi 10.26089/NumMet.v25r211.

Performance analysis methodology of deep neural networks inference on the example of an image classification problem

Murad R. Alibekov
National Research Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia
ORCID: 0000-0002-9201-8878, e-mail: muradalibekov2000@gmail.com

Natalia E. Berezina
YADRO, Nizhny Novgorod, Russia
ORCID: 0009-0003-3111-8829, e-mail: n.berezina@yadro.com

Evgenii P. Vasiliev
National Research Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia
ORCID: 0000-0002-7949-1919, e-mail: evgeny.vasiliev@itmm.unn.ru

Ivan B. Vikhrev
YADRO, Nizhny Novgorod, Russia
ORCID: 0009-0007-8500-1561, e-mail: i.vikhrev@yadro.com

Yulia D. Kamelina
YADRO, Nizhny Novgorod, Russia
ORCID: 0009-0008-7660-3066, e-mail: y.kamelina@yadro.com

Valentina D. Kustikova
National Research Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia
ORCID: 0000-0002-6159-1145, e-mail: valentina.kustikova@itmm.unn.ru

Zoya A. Maslova
YADRO, Nizhny Novgorod, Russia
ORCID: 0009-0001-2587-0438, e-mail: z.maslova@yadro.com

Ivan S. Mukhin
National Research Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia
ORCID: 0009-0006-2558-688X, e-mail: ismukhin03@gmail.com

Alexandra K. Sidorova
National Research Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia
ORCID: 0009-0002-3081-9482, e-mail: alexa.sanechka@ya.ru

Vladislav N. Suchkov
National Research Lobachevsky State University of Nizhny Novgorod, Nizhny Novgorod, Russia
ORCID: 0009-0009-8246-7686, e-mail: suchkov.vladislavn@yandex.ru

Abstract: Deploying of deep neural networks requires inference performance analysis on the target hardware. Performance results are aimed to be used as motivation to evaluate a decision for

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

deployment, find the best performing hardware and software configurations, decide is there's a need for optimization of DL model and DL inference software. The paper describes a technique for analyzing and comparing inference performance using an example of image classification problem: converting a trained model to the formats of different frameworks, quality analysis, determining optimal inference execution parameters, model optimization and quality reanalysis, analyzing and comparing inference performance for the considered frameworks. Deep Learning Inference Benchmark Tool is aimed to support the performance analysis cycle. The technique is implemented on the example of the MobileNetV2 model.

Keywords: deep learning, neural networks, inference, performance, MobileNetV2, Deep Learning Inference Benchmark.

For citation: M. R. Alibekov, N. E. Berezina, E. P. Vasiliev, I. B. Vikhrev, Yu. D. Kamelina, V. D. Kustikova, Z. A. Maslova, I. S. Mukhin, A. K. Sidorova, V. N. Suchkov, "Performance analysis methodology of deep neural networks inference on the example of an image classification problem," Numerical Methods and Programming. **25** (2), 127–141 (2024). doi 10.26089/NumMet.v25r211.

1. Введение. В настоящее время применение методов *глубокого обучения (deep learning)* для решения прикладных задач является актуальным направлением исследований во многих областях. Разработано значительное количество архитектур глубоких нейронных сетей, которые позволяют с высокой точностью решать классические задачи компьютерного зрения и обработки естественного языка. Обученные модели находятся в открытом доступе в сети Интернет, они обучены или сконвертированы под разные фреймворки глубокого обучения [1, 2].

Типовой подход к применению накопленных знаний — *перенос обучения (transfer learning)*. Перенос обучения предполагает использование части архитектуры открытой нейронной сети, относящейся к извлечению признаков (как правило, часть сети без последних полносвязных слоев), дополнение отобранной последовательности слоев необходимыми полносвязными слоями и последующее обучение модифицированной модели на данных, специфичных для рассматриваемой прикладной задачи. Цикл "модификация сети → обучение" повторяется до момента достижения удовлетворительного качества решения задачи. Далее разработанная модель внедряется в реальное приложение. *Внедрение (deployment)* предполагает многократный прямой проход по обученной модели — *вывод (inference)* — на вновь поступающих данных. На практике требуется, чтобы вывод выполнялся в режиме реального времени на целевом аппаратном обеспечении. Поэтому на этапе внедрения анализ и сравнение производительности вывода глубоких моделей является одной из наиболее важных задач.

Существует множество факторов, влияющих на производительность вывода. Сложность глубокой модели определяется количеством вычислительных операций, выполняемых в процессе прямого прохода. Когда получены низкие показатели производительности на целевом оборудовании, реализуется цикл "оптимизация/модификация модели → проверка качества → анализ производительности". Проверка качества модели — неотъемлемый этап, поскольку в процессе оптимизации качество может сильно изменяться. Выбор фреймворка для реализации вывода также оказывает значительное влияние на производительность. Наличие программных оптимизаций для некоторых вычислительных операций под конкретную аппаратную архитектуру позволяет ускорить вывод и достичь необходимых для внедрения модели показателей производительности.

В настоящей работе предлагается методика анализа и сравнения производительности вывода глубоких нейросетевых моделей, применяемая в ходе их внедрения. Для поддержки процесса разрабатывается программный инструмент Deep Learning Inference Benchmark (DLI) с целью автоматической верификации корректности решения задачи и сбора показателей производительности вывода [3–6]. Отличительная особенность системы — расширяемость поддерживаемых фреймворков для вывода глубоких моделей, аппаратных архитектур и множества тестируемых моделей. Результаты производительности вывода для большого числа моделей на имеющемся аппаратном обеспечении публикуются в открытом доступе [7]. Опубликованные результаты производительности позволяют оценить скорость вывода моделей на целевом оборудовании, если нейронная сеть спроектирована с использованием переноса обучения на базе какой-либо открытой модели. Инструмент DLI может использоваться сторонними разработчиками на этапе внедрения собственных моделей или в процессе их оптимизации.

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

2. Обзор литературы. Проблема анализа и повышения производительности глубоких нейронных сетей на этапе их внедрения является актуальной. Процедура оптимизации производительности вывода, как правило, сложная и трудоемкая, поскольку включает не только программную оптимизацию операций, но и сжатие сети, повторное обучение, анализ качества решения задачи и последующий анализ производительности вывода. Такой цикл может повторяться многократно до тех пор, пока на практике не будет найден компромисс между качеством и производительностью вывода модели. Существуют различные подходы к анализу качества и производительности, разработаны программные инструменты, которые обеспечивают автоматизацию процесса на разных этапах внедрения.

Разработчики компании Intel в [8] предлагают инструмент Deep Learning Workbench [9], входящий в состав пакета Intel Distribution of OpenVINO Toolkit (далее сокр. OpenVINO) [10]. Инструмент обеспечивает поддержку всего цикла внедрения глубоких моделей. Вместе с множеством открытых моделей, обученных для решения различных задач компьютерного зрения и обработки естественного языка, в его состав входят инструменты для конвертации обученных моделей во внутреннее представление OpenVINO, которое эффективно исполняется на аппаратных платформах Intel, инструменты для оптимизации моделей посредством понижения точности весов с FP32 и FP16 до INT8 — *квантизации*, инструменты для определения качества и производительности вывода моделей.

В [11] приводится анализ качества для широко известных моделей глубокого обучения, развернутых на аналоговых устройствах. Аналоговое аппаратное обеспечение является перспективным для систем с ограниченным энергопотреблением. Аналоговый характер устройства и множество источников шума приводят к изменению весов в обученных моделях, которые развернуты в таких системах. Поэтому в данной работе исследуется устойчивость широко известных моделей при добавлении к весам сети аддитивного белого гауссовского шума.

Авторы [12] разрабатывают бенчмарк MLMark [13] для сравнения качества и производительности вывода на граничных устройствах (edge devices), анализируют результаты на нескольких ускорителях. Бенчмарк поставляется в открытых исходных кодах. Обеспечивается вывод для трех нейросетевых моделей средствами библиотек TensorFlow, TensorRT, ArmNN/TFL, Google TPU и OpenVINO. Сторонние исследователи могут самостоятельно запустить MLMark на собственных ускорителях для трех поддерживаемых моделей и опубликовать результаты на официальной странице проекта.

MLPerf Inference Benchmark [14, 15] — один из наиболее известных бенчмарков. Авторы пытаются эмулировать сценарии вывода, возникающие в реальных приложениях [14]. При этом в зависимости от сценария вводятся различные показатели производительности. Определен набор эталонных моделей, для которых публикуются результаты производительности в сети Интернет [15]. По аналогии с MLMark разработчики могут опубликовать полученные результаты на собственной аппаратуре на странице бенчмарка.

В [16] приводится анализ и сравнение производительности вывода, который реализован с использованием различных фреймворков, на Intel Cascade Lake CPUs. Поэтапно демонстрируется процедура для Intel Optimization of Caffe, TensorFlow, PyTorch, MXNet, Intel Distribution of OpenVINO Toolkit и OpenCV на примере двух глубоких моделей: подбор оптимальных параметров запуска вывода, анализ и сравнение масштабируемости, сравнение качества работы моделей, сравнение лучших показателей производительности вывода.

Цель настоящей статьи — обобщить и автоматизировать методику анализа и сравнения производительности вывода глубоких нейросетевых моделей, отдельные этапы которой описаны в [8] и [16]. Основное отличие от приведенных работ — наличие формализованной схемы анализа, которая может быть использована исследователями и инженерами IT-компаний в процессе внедрения обученных моделей. Описанная методика четко определяет множество параметров, влияющих на производительность вывода и порядок проведения экспериментов, а разработанная программная система DLI автоматизирует сбор результатов качества работы моделей и производительности их вывода на целевом устройстве. Использование программной системы DLI позволяет избежать большого объема технической работы и полностью сосредоточиться на анализе результатов производительности и возможных путях оптимизации моделей. DLI является расширяемой в плане набора фреймворков, множества моделей и аппаратных платформ для запуска вывода. Система уже сейчас поддерживает вывод средствами наиболее известных фреймворков. Архитектура DLI при необходимости позволяет быстро интегрировать новые фреймворки. В отличие от [12, 14, 15] поддерживается вывод для значительного количества широко известных нейросетевых моделей. Более того, в рамках системы можно запускать вывод собственных

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

моделей. Разрабатываемая система не ориентирована на аппаратное обеспечение Intel, как в [9] или NVIDIA.

3. Постановка задачи классификации изображений. Задача классификации изображений с большим числом категорий состоит в том, чтобы поставить в соответствие изображению класс объектов, содержащихся на этом изображении. Как правило, на входе имеется трехканальное изображение в формате RGB, которое представляется трехмерным тензором I с пространственными размерами w и h, отвечающими ширине и высоте изображения, и глубиной 3. Каждый элемент тензора содержит значение интенсивности соответствующего пикселя в диапазоне от 0 до 255 (или от 0 до 1, если выполнена нормировка). На выходе классификационной нейронной сети формируется вектор вещественных значений. Длина вектора соответствует количеству категорий изображений с учетом или без учета фона, а каждый элемент вектора отвечает достоверности принадлежности изображения определенному классу.

4. Общая схема анализа и сравнения производительности вывода. При выполнении анализа производительности вывода предполагается, что имеется глубокая нейросетевая модель, которая обучена средствами какого-либо фреймворка. Для данной модели достигнуто качество решения прикладной задачи, приемлемое для ее дальнейшего внедрения.

Общая схема анализа и сравнения производительности вывода в процессе внедрения глубоких нейронных сетей состоит из нескольких этапов.

1. Обучение разработанной архитектуры с использованием других фреймворков или конвертация обученной модели в форматы хранения сторонних фреймворков. Отметим, что при обучении необходимо воспроизвести условия и параметры тренировки исходной модели, чтобы получить близкие значения обученных параметров сети.
2. Анализ и сравнение качества полученного набора моделей. Необходимо гарантировать, что вновь обученная и/или сконвертированная модели обеспечивают качество решения прикладной задачи, сравнимое с исходной моделью.
3. Определение оптимальных параметров запуска вывода для каждого фреймворка. Анализ масштабируемости программных реализаций вывода в различных фреймворках, поиск оптимальной реализации вывода.
4. Сжатие и оптимизация разработанных моделей. В простейшем случае можно выполнить квантизацию весов модели с использованием инструментов глубокого обучения. Более сложная оптимизация предполагает модификацию архитектуры сети и повторное обучение.
5. Анализ качества решения задачи с использованием оптимизированных моделей, сравнение показателей качества. Если достигнутые показатели качества не являются достаточными для внедрения, то переход к шагу 4.
6. Сравнение производительности вывода для набора обученных/сконвертированных и оптимизированных моделей. Если полученная производительность не является достаточной для внедрения, то возврат на шаг 4.

Далее демонстрируется реализация представленной схемы на примере задачи классификации изображений с большим числом категорий.

5. Вычислительные эксперименты.

5.1. Программная система Deep Learning Inference Benchmark. DLI — программная система, разрабатываемая в ННГУ [3, 5]. Инструмент позволяет собирать и анализировать показатели производительности вывода нейросетевых моделей, который реализован с использованием разных фреймворков, на доступном аппаратном обеспечении. В настоящее время имеются реализации вывода на Python для Intel Distribution of OpenVINO Toolkit [10], Intel Optimization for Caffe [17], TensorFlow [18], TensorFlow Lite [19], MXNet [20], OpenCV [21] и реализации вывода на C++ для ONNX Runtime [22] и OpenCV. Архитектура системы предусматривает возможность одновременного проведения экспериментов на нескольких узлах с разным составом, находящихся в одной сети. При этом запуск возможен непосредственно на узле или внутри docker-контейнера, что упрощает подготовку окружения.

5.2. Фреймворки для вывода глубоких моделей. В процессе анализа и сравнения производительности вывода используются следующие программные инструменты и библиотеки: Intel Distribution of OpenVINO Toolkit [10], TensorFlow [18], TensorFlow Lite [19], MXNet [20] и OpenCV [21]. Приведенный набор фреймворков выбран по следующим причинам.

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

1. OpenVINO — широко известный открытый инструмент для вывода глубоких сетей, оптимизированный под аппаратные архитектуры Intel (CPU, GPU, VPU или GNA). При этом вывод может выполняться как на отдельных устройствах, так и одновременно на нескольких устройствах, установленных на узле. Более того, OpenVINO содержит два режима вывода: режим максимизации времени выполнения одного запроса (latency mode) и режим максимизации пропускной способности (throughput mode), которые могут быть реализованы средствами синхронного (Sync API) и асинхронного (Async API) программных интерфейсов. Далее в работе используется latency-режим, реализованный с использованием синхронного интерфейса, что соответствует реализациям вывода во всех остальных фреймворках. Подробнее режимы вывода рассматриваются и анализируется в [5, 16].
2. TensorFlow — одна из популярных библиотек глубокого обучения, которая широко используется сообществом. Она обеспечивает обучение и тестирование глубоких моделей. Реализована поддержка символьных вычислений и автоматического дифференцирования.
3. TensorFlow Lite представляет собой реализацию TensorFlow, в которой вывод оптимизирован под мобильные архитектуры, что имеет особое значение на этапе внедрения глубоких нейросетевых моделей.
4. MXNet — менее известный фреймворк глубокого обучения. Разработчики обеспечивают 8 программных интерфейсов для обучения и тестирования нейронных сетей на разном аппаратном обеспечении. Эффективность вычислений в процессе вывода обеспечивается за счет применения символьных вычислений.
5. OpenCV — открытая библиотека алгоритмов компьютерного зрения, которая, в частности, содержит модуль для вывода нейронных сетей, обученных с помощью известных фреймворков. Значительное количество систем видеонаблюдения разрабатывается на базе указанной библиотеки, а доступный функционал позволяет быстро создавать прототипы программных решений с использованием классических алгоритмов, а также глубоких нейросетевых моделей.

5.3. Тестовые модели. Из набора моделей, вывод которых поддерживается во всех перечисленных фреймворках, выбрана известная сверточная сеть MobileNetV2 [23]. На базе указанной классификационной модели строится большое количество глубоких сетей для детектирования объектов на изображениях, а также семантической сегментации изображений.

5.4. Тестовые данные. Определение качества решения задачи с использованием тестовой модели и измерение показателей производительности — два отдельных эксперимента. Соответственно для каждого эксперимента используются свои тестовые данные.

Для определения качества работы моделей используется реальная валидационная выборка из набора данных ImageNET (50 000 изображений) [24]. При этом показатели качества вычисляются на основании результатов классификации, полученных на выходе сети для каждого изображения.

Для анализа производительности моделей используется часть валидационной выборки ImageNET, состоящая из 320 изображений. Следует отметить, что при измерении показателей производительности также допустимо использовать одинаковые изображения или синтетические данные, поскольку вычислительная сложность прямого прохода по сети и время вывода не зависят от конкретных входных данных и значений интенсивности отдельных пикселей.

5.5. Тестовая инфраструктура. Эксперименты выполняются на тестовой инфраструктуре, параметры которой приведены в табл. 1. В общем случае состав инфраструктуры обусловливается тем, на каком оборудовании предполагается запускать вывод глубоких моделей на этапе внедрения.

5.6. Показатели качества классификации изображений. В эксперименте для оценивания качества классификации изображений используются метрики top-1 и top-5. *Точность top-k (top-k accuracy)* — отношение числа правильно проклассифицированных изображений к общему их количеству [5]. На выходе классификационной нейронной сети имеется вектор достоверности принадлежности изображения каждому из допустимых классов. В случае top-1 изображение считается проклассифицированным правильно, если искомому классу соответствует максимальное значение достоверности, а в случае top-5 — если искомому классу соответствует одно из пяти наибольших значений достоверностей.

5.7. Показатели производительности вывода. Эксперимент по измерению показателей производительности вывода предполагает, что запросы на прямой проход по сети выполняются последовательно и многократно. *Число повторений (итераций)* — параметр теста производительности. Далее указанный

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

Таблица 1. Вычислительная инфраструктура

Table 1. Computational infrastructure

CPU	Intel® Core™ i7-8700 3.20GHz (6 ядер и 12 потоков)
GPU	Intel® Gen9 HD Graphics (iGPU)
RAM	64 ГБ
Operating system	Ubuntu 20.04.4 LTS
Software libraries	Intel Distribution of OpenVINO Toolkit 2022.3 TensorFlow 2.9.3 TensorFlow Lite 2.9.3 (part of TensorFlow) MXNet 1.9.1 OpenCV 4.7

параметр во всех экспериментах устанавливается равным 1000. Один прямой проход предполагает решение задачи для подмножества входных данных — *пачки*. В ходе эксперимента выборка из 320 тестовых изображений делится на пачки, для каждой пачки выполняется прямой проход. Если общего числа пачек недостаточно, чтобы выполнить 1000 итераций, то реализуется кольцевой алгоритм обхода входных данных. Из множества пачек организуется кольцевой буфер и последовательно подаются на вывод ранее обработанные пачки данных. Каждый следующий запрос выполняется после завершения предыдущего. Для отдельного запроса измеряется продолжительность его выполнения. Рассчитывается стандартное среднеквадратическое отклонение на основании набора полученных длительностей выполнения запросов. Длительности, которые выходят за пределы трех стандартных отклонений относительно среднего времени вывода, отбрасываются. Результирующий набор времен используется для вычисления двух метрик производительности [6].

1. *Латентность (Latency)* — медиана времен выполнения запросов. Меньшее значение времени соответствует более эффективной реализации.
2. *Среднее количество кадров, обрабатываемых за секунду (Frames per Second, FPS)* — отношение размера пачки изображений к латентности. Большее значение FPS соответствует более эффективной реализации.

Далее рассматриваются только показатели FPS, поскольку для каждого фиксированного размера пачки латентность можно вычислить на основании данной метрики. Отметим, что приведенный перечень показателей справедлив для всех фреймворков, включая latency-режим инструментария OpenVINO.

5.8. Перебираемые при анализе производительности параметры. В процессе анализа производительности выделяется несколько уровней перебираемых параметров.

1. *Фреймворки.* Перечень фреймворков для анализа производительности вывода нейросетевых моделей зависит от двух факторов. Первый фактор — выбранное целевое устройство, поскольку фреймворки можно собрать и запустить только на ограниченном наборе аппаратных платформ; второй — формат тестовой модели и наличие конвертеров в форматы выбранных фреймворков.
2. *Формат весов модели.* Большинство фреймворков работает с моделями, веса которых представляют собой вещественные числа в формате FP32. Тем не менее фреймворки могут содержать внутренние инструменты для конвертации весов в формат FP16, а также инструменты для квантизации моделей (формат весов INT8). Использование форматов, отличных от FP32, позволяет уменьшить объем необходимой для хранения модели памяти и ускорить вычисления в процессе прямого прохода по сети при наличии аппаратной поддержки операций для чисел в используемом формате. Поэтому для каждого выбранного фреймворка имеет смысл рассматривать все допустимые веса моделей.
3. *Параметры запуска вывода.* Основной параметр данной группы — размер пачки данных, обрабатываемой за один проход по сети. Наряду с размером пачки, каждый фреймворк имеет набор специфичных параметров, например количество потоков, которые параллельно выполняют операции. Поэтому один из этапов описанной методики предполагает, что для каждого фиксированного размера пачки данных необходимо определить оптимальные значения специфичных параметров. Для некоторых фреймворков разработчики гарантируют, что оптимальными являются значения по умолчанию.

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

5.9. Результаты экспериментов.

§ 5.9.1. Обучение и/или конвертирование обученной модели. Согласно описанной схеме анализа производительности вывода первый шаг — обучение и/или конвертация обученной модели в форматы, доступные для чтения в различных фреймворках глубокого обучения. Перечень фреймворков, устройство (CPU, GPU, другие) и формат хранения весов являются параметрами процедуры анализа. В качестве исходной модели используется MobileNetV2 в формате библиотеки TensorFlow, загруженная из Open Model Zoo [25]. Обученная модель конвертируется в *промежуточное представление (intermediate representation, IR)* с весами FP32 и FP16 для вывода средствами OpenVINO с помощью инструмента omz_converter, входящего в его состав. Модель с весами в формате FP16 получается посредством понижения точности обученных весов без дополнительной тонкой настройки модели на каких-либо сторонних данных. В процессе вывода MobileNetV2 через TensorFlow и OpenCV модель подается в исходном виде. Для преобразования модели в формат библиотеки TensorFlow Lite используется конвертер, доступный в разрабатываемой системе DLI. Для вывода средствами MXNet используется соответствующая модель, входящая в состав GluonCV Model Zoo [2]. Модель MobileNetV2 в форматах библиотек TensorFlow и MXNet обучена и провалидирована на наборе данных ImageNET [24] сторонними исследователями и выложена в открытый доступ. При этом условия и параметры обучения в обоих фреймворках соответствуют описанным в работе авторов модели [23].

В рамках имеющейся тестовой инфраструктуры вывод модели MobileNetV2 с весами FP32 для всех фреймворков доступен для запуска на CPU, для OpenVINO также имеется возможность запуска с весами FP32 и FP16 на интегрированной графической карте. Отметим, что для некоторых фреймворков имеется возможность конвертации модели в формат весов FP16, но в процессе вывода моделей с весами FP16 на CPU все операции выполняются с числами в формате FP32, поскольку отсутствует аппаратная поддержка операций в FP16, т.е. такой переход позволит получить модель меньшего размера с точки зрения объема памяти, необходимой для ее хранения, но не обеспечит выигрыша в производительности вывода.

§ 5.9.2. Анализ и сравнение качества. После того как сформирован набор обученных и/или сконвертированных моделей для запуска под разные фреймворки, необходимо провалидировать качество решения задачи классификации с помощью этих моделей. На рис. 1 приведены полученные значения показателей точности top-1 и top-5 на полной валидационной выборке набора ImageNET. Из построенной гистограммы можно видеть, что результаты качества работы моделей в разных фреймворках практически совпадают: наибольшее отличие в значении top-1 — 0.11%, top-5 — 0.33%. Незначительное отличие для моделей с

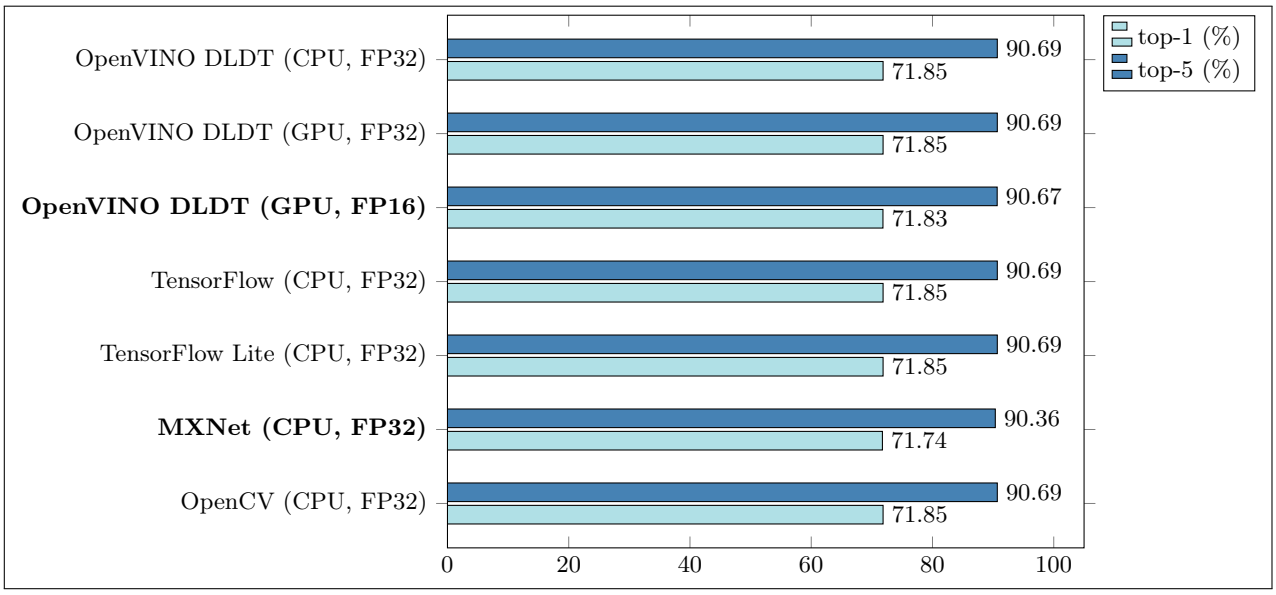

Рис. 1. Точность классификации изображений
с использованием модели MobileNetV2 с весами в форматах FP32 и FP16

Fig. 1. Image classification accuracy using MobileNetV2 model with weights in FP32 and FP16 formats

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

весами в форматах FP32 и FP16 (см. результат в третьей строке сверху на рис. 1) объясняется более низкой точностью хранения весов модели и исполнения. Разница в качестве для модели в формате MXNet и других фреймворков является следствием того, что исследуемая модель в составе GluonCV Model Zoo обучена с использованием MXNet, а не сконвертирована напрямую из формата TensorFlow, как для всех остальных запускаемых фреймворков. Вероятнее всего, отличие обусловлено выбором начального приближения весов в процессе обучения модели в разных фреймворках с использованием метода обратного распространения ошибки.

§ 5.9.3. Определение оптимальных параметров запуска вывода, анализ масштабируемости, поиск оптимальной реализации вывода. Следующий шаг — подбор оптимальных параметров запуска вывода. В [16] описывается процедура подбора параметров запуска вывода (количество потоков и др.) для инструмента OpenVINO. Показано, что лучшие результаты производительности достигаются при установленных значениях по умолчанию. В [5] описывается подбор параметров запуска для Intel Optimizations for TensorFlow. Общая схема подбора параметров для всех фреймворков идентична, отличие состоит только в назначении и количестве допустимых параметров. Эксперименты, проводимые в настоящей работе, предполагают запуск вывода с параметрами по умолчанию, чтобы выровнять сложность этого этапа для всех фреймворков. Отметим, что вывод средствами MXNet запускается без подключения библиотеки oneDNN (ранее MKL-DNN) без (no hybrid) и с (hybrid) использованием символических вычислений. Запуск с oneDNN также требует подбора параметров, обеспечивающих эффективное исполнение операций, распараллеленных с помощью OpenMP. Оптимальные параметры для запуска вывода в значительной степени зависят от архитектуры нейронной сети.

К настоящему моменту можно гарантировать, что обученные и сконвертированные модели одинаково работают в разных фреймворках, а также определены параметры окружения для запуска вывода и дальнейшего анализа масштабируемости. Далее необходимо собрать результаты производительности при изменении допустимого размера пачки данных для каждого фреймворка. На рис. 2 показаны графики изменения FPS при переборе размеров пачки, являющихся степенями двойки {1, 2, 4, 8, 16, 32, 64}. Для всех фреймворков наблюдается рост производительности до некоторого размера пачки (2 — для OpenVINO (CPU, FP32) и MXNet (CPU, FP32), 4 — для TensorFlow и OpenCV, 32 — для OpenVINO (GPU, FP32 и

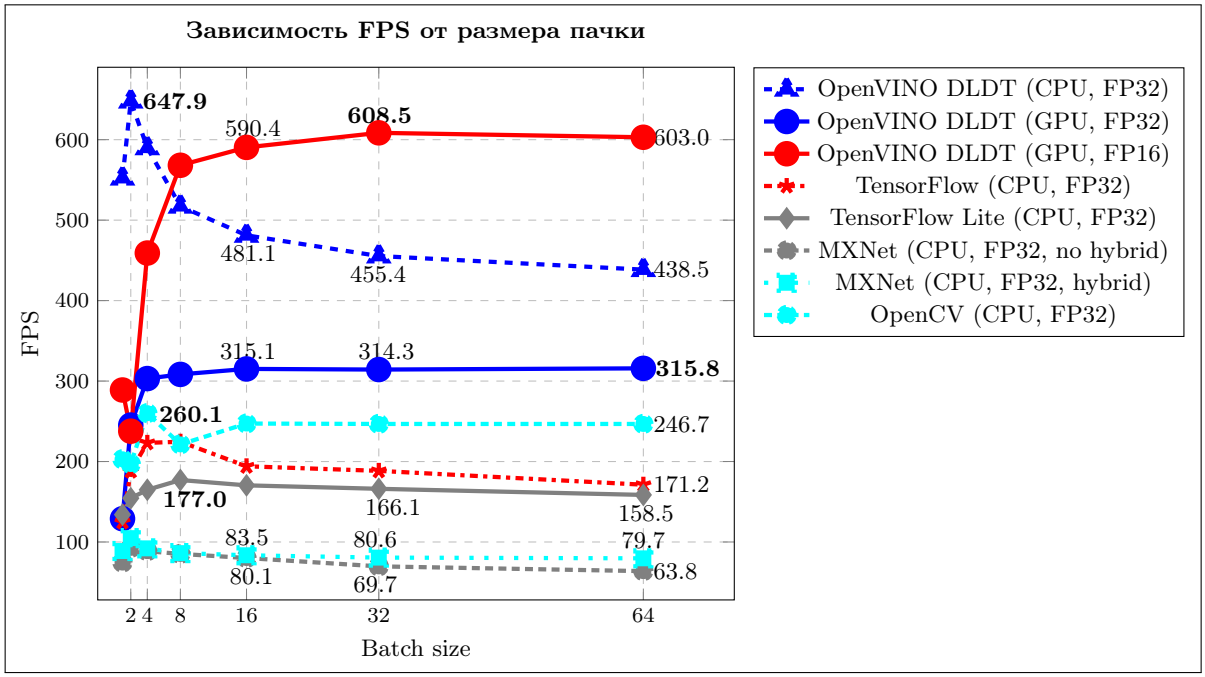

Рис. 2. Изменение показателя FPS при разных размерах входной пачки изображений для модели MobileNetV2 с весами в форматах FP32 и FP16

Fig. 2. Changes in FPS at different sizes of the batch size for the MobileNetV2 model with weights in the FP32 and FP16 formats

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

FP16)), далее происходит постепенное снижение показателя. Размер пачки, на котором достигается максимальная производительность вывода, во многом определяется размером доступного кэша. Нерегулярное поведение FPS при небольших размерах пачки для OpenVINO DLDT (CPU, FP32) (пик на пачке в 2 изображения) вероятнее всего обусловлено эффективностью векторизации вычислений и организации доступа к памяти при таких объемах данных. Использование символьных вычислений в реализации MXNet (MXNet (CPU, FP32, hybrid)) дает небольшой прирост производительности, но в целом показатели производительности MXNet ниже, чем для других фреймворков. Следует отметить, что в MXNet все операции выполняются асинхронно, поэтому после каждого прямого прохода осуществляется ожидание завершения вывода. Вероятнее всего, асинхронность позволит получить лучшие результаты в других сценариях исполнения, например когда необходимо минимизировать общее время выполнения набора запросов. Тем не менее, все рассматриваемые фреймворки работают в режиме реального времени независимо от размера пачки входных данных, т.е. способны обрабатывать более 25 кадров в секунду (FPS>25 fps). Отдельно следует выделить результаты, полученные при запуске FP16-модели с использованием OpenVINO на Intel GPU (OpenVINO DLDT (GPU, FP16)). Вывод на GPU при размерах пачки, превышающих 8 изображений, работает быстрее реализации на TensorFlow, TensorFlow Lite, MXNet, OpenCV и других вариантов запуска OpenVINO при очень небольших потерях качества классификации. Приведенный факт говорит о перспективности использования интегрированных карт, поскольку это позволяет освободить CPU для решения более трудоемких задач.

В процессе внедрения выбор фреймворка во многом зависит от скорости вывода для фиксированного размера пачки данных на целевой аппаратуре (срез построенного графика по одному значению оси абсцисс). Размер пачки, как правило, определяется тем, с какой скоростью поступают данные с устройства, в частности изображения с камеры для задач компьютерного зрения.

§ 5.9.4. Сжатие и оптимизация моделей. На следующем этапе внедрения выполняется оптимизация построенных моделей. В простейшем случае — квантизация весов, т.е. переход от формата весов FP32 или FP16 к INT8 или UINT8. Квантизованные модели с весами INT8 для OpenVINO получены средствами Post-Training Optimization Tool, входящего в состав самого инструментария. Для фреймворка TensorFlow Lite соответствующая квантизованная модель с весами UINT8 находится в открытом доступе [26]. Отметим, что некоторые рассматриваемые фреймворки (TensorFlow и MXNet) также обеспечивают квантизацию моделей. На практике имеет смысл выполнять квантизацию и запуск экспериментов для всех доступных фреймворков, но цель настоящей работы — продемонстрировать методику анализа производительности, поэтому перебор всех возможных вариантов выходит за рамки исследования.

§ 5.9.5. Анализ качества оптимизированных моделей. Для полученного набора оптимизированных моделей осуществляется анализ качества классификации. Результаты экспериментов показывают, что значения точностей top-1 и top-5 для всего набора моделей сопоставимы. Квантизация приводит к снижению качества классификации менее чем на 1% (top-1 снижается не более чем на 0.91%, и top-5 — не более чем на 0.71%).

§ 5.9.6. Сравнение производительности вывода для набора обученных/сконвертированных и оптимизированных моделей. Сравним производительность вывода исходных и квантизованных моделей для OpenVINO и TensorFlow Lite. Заметим, что общий тренд в изменении показателей FPS с ростом размера пачки данных сохраняется и для квантизованных моделей (рис. 3).

При запуске вывода средствами OpenVINO на CPU переход к весам в формате INT8 дает выигрыш от 1.45 на пачке в 2 до 2.18 раза на пачке в 64 изображения, на GPU такой переход приводит к замедлению. Разработчики фреймворка отмечают, что исполнение квантизованных моделей на GPU реализовано через DP4a, а поддержка DP4a появилась с 11-го поколения Intel Core GPUs [27]. При запуске UINT8-модели через TensorFlow Lite для размера пачки, меньшей 16, наблюдается уменьшение показателя FPS, далее увеличение на 6–18% относительно модели с весами в формате FP32 в зависимости от размера пачки, выигрыш производительности менее заметный, чем для OpenVINO.

Обсудим лучшие показатели производительности, полученные в ходе экспериментов. На рис. 4 приведена гистограмма с максимальными значениями FPS, достигнутыми для каждого фреймворка при разных вариантах запуска. Лучшие результаты наблюдаются при выводе MobileNetV2 с весами в формате INT8 с использованием библиотеки OpenVINO на размере тестовой пачки в 8 изображений. При незначительных потерях в качестве классификации вывод работает в ∼1.52 раза быстрее по сравнению с аналогичным запуском модели с весами FP32 (рис. 4, строка OpenVINO DLDT (CPU, FP32, bs=2)). Вто-

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

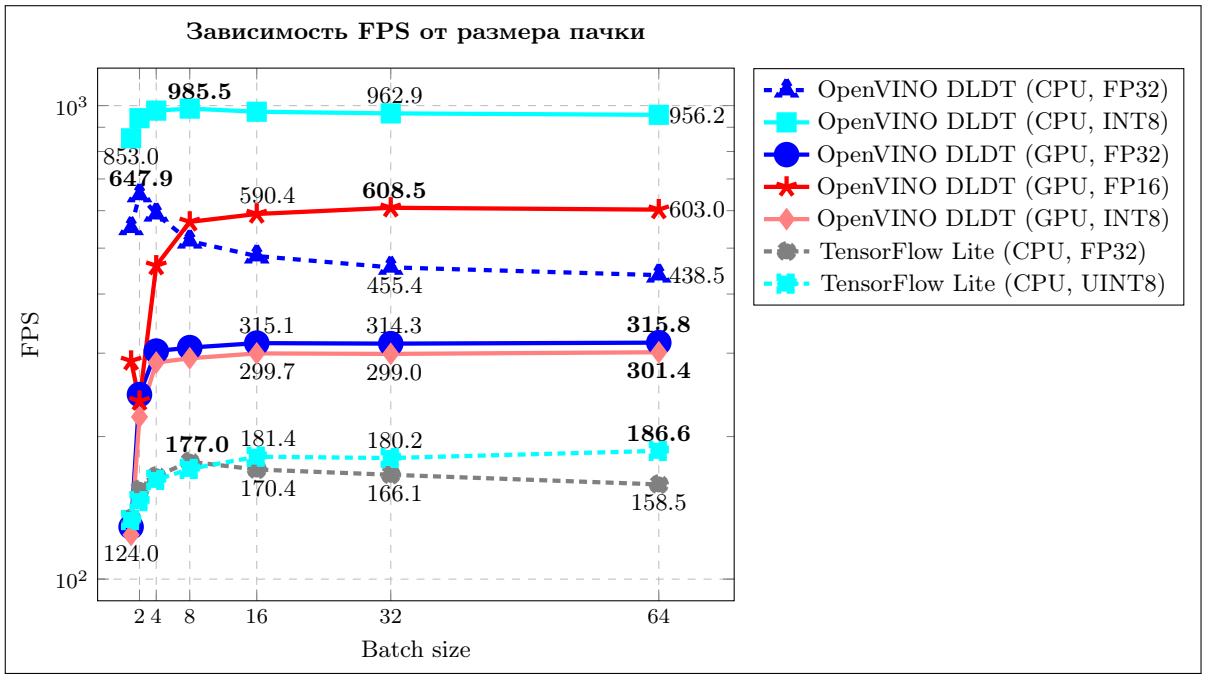

Рис. 3. Изменение показателя FPS при разных размерах входной пачки изображений для модели MobileNetV2 при переходе от весов в форматах FP32 и FP16 к INT8 и UINT8 (логарифмическая ось Oy)

Fig. 3. Changes in FPS at different sizes of the batch size for the MobileNetV2 model when moving from weights in the FP32 and FP16 formats to INT8 and UINT8 (logarithmic axis Oy)

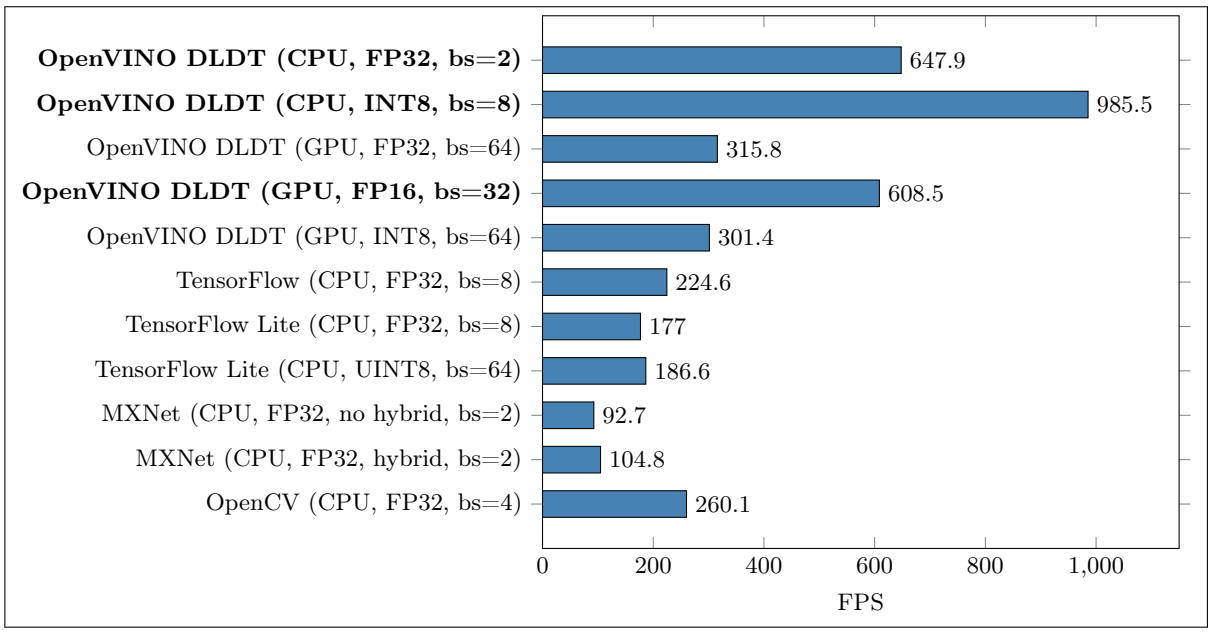

Рис. 4. Наиболее высокие показатели производительности вывода модели MobileNetV2, полученные при проведении экспериментов (в скобках указано устройство, формат весов модели и размер пачки входных данных bs, на которой получена наибольшая производительность)

Fig. 4. The highest inference performance of the MobileNetV2 model obtained during experiments (the device, the format of the model weights and the batch size bs, on which the highest performance was obtained, are indicated in parentheses)

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

рой лучший результат получен при запуске FP32-модели средствами OpenVINO на CPU (рис. 4, строка OpenVINO DLDT (CPU, FP32, bs=2)). Третий результат достигается при выводе FP16-модели на GPU (рис. 4, строка OpenVINO DLDT (GPU, FP16, bs=32)). Остальные реализации вывода работают значительно медленнее, тем не менее все они решают задачу классификации в режиме реального времени. На основании лучших значений показателя FPS, полученных для каждого фреймворка, можно выбрать конфигурацию параметров в процессе внедрения модели. Если требуется достичь максимальной производительности, то следует запускать квантизованную модель (INT8) средствами OpenVINO DLDT на CPU. Если необходимо освободить CPU для решения других задач, то вполне возможен запуск FP16-модели на Intel GPU.

6. Заключение. Анализ производительности вывода глубоких нейронных сетей — важная проблема, возникающая на этапе их внедрения. В исследовании проработана методика анализа и сравнения производительности вывода, которая определяет перечень параметров, влияющих на скорость вывода и последовательность проведения экспериментов. Следует отметить, что для каждой конкретной глубокой модели необходимо отдельно выполнять анализ и сравнение производительности вывода, поскольку архитектура модели в значительной степени влияет на время вывода. Анализ, как правило, требует не меньше времени, чем построение и исследование новых архитектур нейросетей для решения прикладной задачи.

Разработана программная система Deep Learning Inference Benchmark для поддержки сбора результатов и автоматизации некоторых этапов методики. Система является расширяемой в отношении набора поддерживаемых фреймворков для вывода, аппаратных архитектур и множества тестируемых моделей, что существенно отличает ее от аналогов. Программная система DLI выложена в открытый доступ, поэтому может быть использована разработчиками на этапе внедрения собственных глубоких нейросетевых моделей. Система открыта для модификации и расширения.

Практическое применение методики продемонстрировано на примере решения задачи классификации изображений с использованием модели MobileNetV2, вывод которой запускается через широко известные фреймворки и библиотеки (Intel Distribution of OpenVINO Toolkit, TensorFlow, TensorFlow Lite, MXNet, OpenCV). Проведенные эксперименты на доступной тестовой инфраструктуре (Intel CPU, Intel GPU) показывают, что вывод MobileNetV2 средствами всех сравниваемых фреймворков работает в режиме реального времени. Лучшие показатели производительности получены при выводе модели с весами INT8 средствами OpenVINO. Следует отметить эффективность квантизации модели при исполнении на Intel CPU через OpenVINO (дает выигрыш почти в 1.5 раза по сравнению с соответствующим запуском FP32-модели), а также перспективность использования Intel GPU для MobileNetV2, если необходимо освободить центральный процессор для других вычислений (модель с весами FP16 на GPU отстает от FP32 на CPU всего лишь на 6% на лучшем размере пачки данных). Дополнительно необходимо исследовать оптимальные параметры запуска вывода для фреймворков TensorFlow, TensorFlow Lite, MXNet, OpenCV по аналогии с тем, как это сделано в [16] для Intel Distribution of OpenVINO Toolkit, в [5] для Intel Optimization for Caffe и Intel Optimizations for TensorFlow. Такой подбор вероятнее всего позволит улучшить показатели производительности вывода.

Список литературы

1. Open Model Zoo for OpenVINO toolkit. https://docs.openvino.ai/latest/model_zoo.html. (Дата обращения: 24 марта 2024).
2. GluonCV Model Zoo. https://cv.gluon.ai/model_zoo/index.html. (Дата обращения: 24 марта 2024).
3. Deep Learning Inference Benchmark. https://github.com/itlab-vision/dl-benchmark. (Дата обращения: 24 марта 2024).
4. *Kustikova V., Vasiliev E., Khvatov A., et al.* DLI: deep learning inference benchmark // Communications in Computer and Information Science. Vol. 1129. Cham: Springer, 2019. 542–553. doi 10.1007/978-3-030-36592-9_44.
5. *Сидорова А.К., Алибеков М.Р., Макаров А.А., Васильев Е.П., Кустикова В.Д.* Автоматизация сбора показателей производительности вывода глубоких нейронных сетей в системе Deep Learning Inference Benchmark // Труды XXI Международной конференции “Математическое моделирование и суперкомпьютерные технологии”, 22–26 ноября 2021 г., Нижний Новгород. Н. Новгород: Изд-во Нижегородского госуниверситета, 2021. 318–325. https://hpc-education.unn.ru/ru/files/conference_hpc/2021/MMST2021_Proceedings.pdf. (Дата обращения: 24 марта 2024).
6. DLI: Deep Learning Inference Benchmark. https://hpc-education.unn.ru/dli-ru. (Дата обращения: 24 марта 2024).

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

7. Deep Learning Inference Benchmark Wiki. https://github.com/itlab-vision/dl-benchmark/wiki. (Дата обращения: 24 марта 2024).
8. *Demidovskij A., Gorbachev Yu., Fedorov M., et al.* OpenVINO Deep Learning Workbench: Comprehensive Analysis and Tuning of Neural Networks Inference. https://openaccess.thecvf.com/content_ICCVW_2019/papers/SDL-CV/Gorbachev_OpenVINO_Deep_Learning_Workbench_Comprehensive_Analysis_and_Tuning_of_Neural_ICCVW_2019_paper.pdf. (Дата обращения: 24 марта 2024).
9. OpenVINO Deep Learning Workbench Overview. https://docs.openvino.ai/latest/workbench_docs_Workbench_DG_Introduction.html. (Дата обращения: 24 марта 2024).
10. Intel Distribution of OpenVINO Toolkit. https://docs.openvino.ai/latest/home.html. (Дата обращения: 24 марта 2024).
11. *Fagbohungbe O., Qian L.* Benchmarking inference performance of deep learning models on analog devices // Proc. 2021 Int. Joint Conf. on Neural Networks, Shenzhen, China, July 18–22, 2021. Piscataway: IEEE Press, 2021. 1–9. doi 10.1109/IJCNN52387.2021.9534143.
12. *Torelli P., Bangale M.* Measuring inference performance of machine-learning frameworks on edge-class devices with the MLMark Benchmark. https://www.eembc.org/techlit/articles/MLMARK-WHITEPAPER-FINAL-1.pdf. (Дата обращения: 24 марта 2024).
13. EEMBC's Machine-Learning Inference Benchmark targeted at edge devices. https://github.com/eembc/mlmark. (Дата обращения: 24 марта 2024).
14. *Reddi V.J., Cheng C., Kanter D., et al.* MLPerf Inference Benchmark. https://arxiv.org/abs/1911.02549. (Дата обращения: 24 марта 2024).
15. MLPerf Inference Benchmarks for Image Classification and Object Detection Tasks. https://github.com/mlcommons/inference. (Дата обращения: 24 марта 2024).
16. *Vasiliev E.P., Kustikova V.D., Volokitin V.D., et al.* Performance analysis of deep learning inference in convolutional neural networks on Intel Cascade Lake CPUs // Communications in Computer and Information Science. Vol. 1413. Cham: Springer, 2021. 346–360. doi 10.1007/978-3-030-78759-2_29.
17. Intel Optimization for Caffe. https://github.com/intel/caffe. (Дата обращения: 24 марта 2024).
18. TensorFlow. https://www.tensorflow.org. (Дата обращения: 24 марта 2024).
19. TensorFlow Lite. https://www.tensorflow.org/lite. (Дата обращения: 24 марта 2024).
20. MXNet. https://mxnet.apache.org. (Дата обращения: 24 марта 2024).
21. OpenCV. https://opencv.org. (Дата обращения: 24 марта 2024).
22. ONNX Runtime. https://onnxruntime.ai. (Дата обращения: 24 марта 2024).
23. *Sandler M., Howard A., Zhu M., et al.* MobileNetV2: inverted residuals and linear bottlenecks. https://arxiv.org/abs/1801.04381. (Дата обращения: 24 марта 2024).
24. ImageNet. https://www.image-net.org. (Дата обращения: 24 марта 2024).
25. OpenVINO Toolkit — Open Model Zoo repository. https://github.com/openvinotoolkit/open_model_zoo. (Дата обращения: 24 марта 2024).
26. TensorFlow Hub. Mobilenet V2 trained on Imagenet. https://tfhub.dev/iree/lite-model/mobilenet_v2_100_224/uint8/1. (Дата обращения: 24 марта 2024).
27. Intel Launches World's Best Processor for Thin-and-Light Laptops: 11th Gen Intel Core. https://www.intc.com/news-events/press-releases/detail/1411/intel-launches-worlds-best-processor-for-thin-and-light. (Дата обращения: 24 марта 2024).

Поступила в редакцию
10 декабря 2023 г.

Принята к публикации
5 марта 2024 г.

Информация об авторах

Мурад Рамазанович Алибеков — студент; Нижегородский государственный университет имени Н. И. Лобачевского, пр-кт Гагарина, д. 23, 603022, Нижний Новгород, Российская Федерация.

Наталья Евгеньевна Березина — руководитель группы измерения производительности компонентов программного обеспечения искусственного интеллекта; ООО "Ядро Центр Исследований и Разработки", ул. Алексеевская, д. 6/16, 603005, Нижний Новгород, Российская Федерация.

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

Евгений Павлович Васильев — мл. науч. сотр.; Нижегородский государственный университет имени Н. И. Лобачевского, пр-кт Гагарина, д. 23, 603022, Нижний Новгород, Российская Федерация.

Иван Борисович Вихрев — инженер по разработке программного обеспечения искусственного интеллекта; ООО "Ядро Центр Исследований и Разработки", ул. Алексеевская, д. 6/16, 603005, Нижний Новгород, Российская Федерация.

Юлия Дмитриевна Камелина — инженер по разработке программного обеспечения искусственного интеллекта; ООО "Ядро Центр Исследований и Разработки", ул. Алексеевская, д. 6/16, 603005, Нижний Новгород, Российская Федерация.

Валентина Дмитриевна Кустикова — к.т.н., ст. науч. сотр.; Нижегородский государственный университет имени Н. И. Лобачевского, пр-кт Гагарина, д. 23, 603022, Нижний Новгород, Российская Федерация.

Зоя Александровна Маслова — старший инженер по разработке инфраструктуры; ООО "Ядро Центр Исследований и Разработки", ул. Алексеевская, д. 6/16, 603005, Нижний Новгород, Российская Федерация.

Иван Сергеевич Мухин — студент; Нижегородский государственный университет имени Н. И. Лобачевского, пр-кт Гагарина, д. 23, 603022, Нижний Новгород, Российская Федерация.

Александра Константиновна Сидорова — студент; Нижегородский государственный университет имени Н. И. Лобачевского, пр-кт Гагарина, д. 23, 603022, Нижний Новгород, Российская Федерация.

Владислав Николаевич Сучков — студент; Нижегородский государственный университет имени Н. И. Лобачевского, пр-кт Гагарина, д. 23, 603022, Нижний Новгород, Российская Федерация.

References

1. Open Model Zoo for OpenVINO Toolkit. https://docs.openvino.ai/latest/model_zoo.html. Cited March 22, 2024.
2. GluonCV Model Zoo. https://cv.gluon.ai/model_zoo/index.html. Cited March 22, 2024.
3. Deep Learning Inference Benchmark. https://github.com/itlab-vision/dl-benchmark. Cited March 22, 2024.
4. V. Kustikova, E. Vasiliev, A. Khvatov, et al., "DLI: Deep Learning Inference Benchmark," in *Communications in Computer and Information Science* (Springer, Cham, 2019), Vol. 1129, pp. 542–553. doi 10.1007/978-3-030-36592-9_44.
5. A. K. Sidorova, M. R. Alibekov, A. A. Makarov, et al., "Automating the Collection of Deep Neural Network Inference Performance Metrics in the Deep Learning Inference Benchmark System," in *Proc. XXI Int. Conf. on Mathematical Modeling and Supercomputer Technologies, Nizhny Novgorod, Russia, November 22–26, 2021* (Nizhny Novgorod University Press, Nizhny Novgorod, 2021), pp. 318–325 [in Russian].
6. DLI: Deep Learning Inference Benchmark. https://hpc-education.unn.ru/dli-ru. Cited March 22, 2024.
7. Deep Learning Inference Benchmark Wiki. https://github.com/itlab-vision/dl-benchmark/wiki. Cited March 22, 2024.
8. A. Demidovskij, Yu. Gorbachev, M. Fedorov, et al., "OpenVINO Deep Learning Workbench: Comprehensive Analysis and Tuning of Neural Networks Inference," https://openaccess.thecvf.com/content_ICCVW_2019/papers/SDL-CV/Gorbachev_OpenVINO_Deep_Learning_Workbench_Comprehensive_Analysis_and_Tuning_of_Neural_ICCVW_2019_paper.pdf. Cited March 22, 2024.
9. OpenVINO Deep Learning Workbench Overview. https://docs.openvino.ai/latest/workbench_docs_Workbench_DG_Introduction.html. Cited March 22, 2024.
10. Intel Distribution of OpenVINO Toolkit. https://docs.openvino.ai/latest/home.html. Cited March 22, 2024.
11. O. Fagbohungbe and L. Qian, "Benchmarking Inference Performance of Deep Learning Models on Analog Devices," in *Proc. 2021 Int. Joint Conf. on Neural Networks, Shenzhen, China, July 18–22, 2021* (IEEE Press, Piscataway, 2021), pp. 1–9. doi 10.1109/IJCNN52387.2021.9534143.
12. P. Torelli and M. Bangale, "Measuring Inference Performance of Machine-Learning Frameworks on Edge-class Devices with the MLMark Benchmark," https://www.eembc.org/techlit/articles/MLMARK-WHITEPAPER-FINAL-1.pdf. Cited March 22, 2024.

2024, **25** (2), 127–141. doi 10.26089/NumMet.v25r211

13. EEMBC's Machine-Learning Inference Benchmark targeted at edge devices. https://github.com/eembc/mlmark. Cited March 22, 2024.
14. V. J. Reddi, C. Cheng, D. Kanter, et al., "MLPerf Inference Benchmark," https://arxiv.org/abs/1911.02549. Cited March 22, 2024.
15. MLPerf Inference Benchmarks for Image Classification and Object Detection Tasks. https://github.com/mlcommons/inference. Cited March 22, 2024.
16. E. P. Vasiliev, V. D. Kustikova, V. D. Volokitin, et al., "Performance Analysis of Deep Learning Inference in Convolutional Neural Networks on Intel Cascade Lake CPUs," in *Communications in Computer and Information Science* (Springer, Cham, 2021), Vol. 1413, pp. 346–360. doi 10.1007/978-3-030-78759-2_29.
17. Intel Optimization for Caffe. https://github.com/intel/caffe. Cited March 22, 2024.
18. TensorFlow. https://pypi.org/project/intel-tensorflow. Cited March 22, 2024.
19. TensorFlow Lite. https://www.tensorflow.org/lite. Cited March 22, 2024.
20. MXNet. https://mxnet.apache.org. Cited March 22, 2024.
21. OpenCV. https://opencv.org. Cited March 22, 2024.
22. ONNX Runtime. https://onnxruntime.ai. Cited March 22, 2024.
23. M. Sandler, A. Howard, M. Zhu, et al., "MobileNetV2: Inverted Residuals and Linear Bottlenecks," https://arxiv.org/abs/1801.04381. Cited March 22, 2024.
24. ImageNet. https://www.image-net.org. Cited March 22, 2024.
25. OpenVINO Toolkit – Open Model Zoo repository. https://github.com/openvinotoolkit/open_model_zoo. Cited March 22, 2024.
26. TensorFlow Hub. Mobilenet V2 trained on Imagenet. https://tfhub.dev/iree/lite-model/mobilenet_v2_100_224/uint8/1. Cited March 22, 2024.
27. Intel Launches World's Best Processor for Thin-and-Light Laptops: 11th Gen Intel Core. https://www.intc.com/news-events/press-releases/detail/1411/intel-launches-worlds-best-processor-for-thin-and-light. Cited March 22, 2024.

Received
December 10, 2023

Accepted for publication
March 5, 2024

Information about the authors

Murad R. Alibekov — Student; National Research Lobachevsky State University of Nizhny Novgorod, Gagarina prospekt, 23, 603022, Nizhny Novgorod, Russia.

Natalia E. Berezina — AI benchmarking lead; YADRO, Alexeevskaya ulitsa, 6/16, 603005, Nizhny Novgorod, Russia.

Evgenii P. Vasiliev — Junior researcher; National Research Lobachevsky State University of Nizhny Novgorod, Gagarina prospekt, 23, 603022, Nizhny Novgorod, Russia.

Ivan B. Vikhrev — AI software engineer; YADRO, Alexeevskaya ulitsa, 6/16, 603005, Nizhny Novgorod, Russia.

Yulia D. Kamelina — AI software engineer; YADRO, Alexeevskaya ulitsa, 6/16, 603005, Nizhny Novgorod, Russia.

Valentina D. Kustikova — Ph.D., Senior researcher; National Research Lobachevsky State University of Nizhny Novgorod, Gagarina prospekt, 23, 603022, Nizhny Novgorod, Russia.

Zoya A. Maslova — AI framework engineer; YADRO, Alexeevskaya ulitsa, 6/16, 603005, Nizhny Novgorod, Russia.

Ivan S. Mukhin — Student; National Research Lobachevsky State University of Nizhny Novgorod, Gagarina prospekt, 23, 603022, Nizhny Novgorod, Russia.

Alexandra K. Sidorova — Student; National Research Lobachevsky State University of Nizhny Novgorod, Gagarina prospekt, 23, 603022, Nizhny Novgorod, Russia.

Vladislav N. Suchkov — Student; National Research Lobachevsky State University of Nizhny Novgorod, Gagarina prospekt, 23, 603022, Nizhny Novgorod, Russia.