1 слайд - Введение

Киреев К.А. 8383

Использование OpenVINO Toolkit для распределенной обработки информации

2 слайд - Актуальность

В современном мире большую популярность приобрел искусственный интеллект, и в частности компьютерное зрение. Если оглянуться на 10 лет назад, то это было не так, поскольку область компьютерного зрения была скорее темой академического интереса. Главный плюс ИИ в том, что он позволяет решить множество рутинных и сложных задач, но самое главное его можно применить на большом объёме данных, который человек не в состоянии обработать. По этой причине бурное развитие ИИ, в частности компьютерного зрения, привело к появлению больших и сложных алгоритмов обработки данных. На данный самой популярной и эффективной технологией компьютерного зрения является глубокое обучение.

Основной проблемой данного подхода является большое количество требуемых ресурсов при использовании моделей глубокого обучения. Производительность глубокого обучения, как и любого другого алгоритма, зависит от платформы, на которой его запускают, и насколько эффективно эта платформа работает с вычислительными устройствами (CPU, GPU и др.). Улучшить производительность можно за счёт запуска оптимизированного алгоритма непосредственно на вычислительном устройстве или же использования ресурсы вычислительного устройства эффективнее, чем это возможно сейчас.

В попытках решить данную проблему компания Intel создала программное обеспечение OpenVINO Toolkit, позволяющее

оптимизировать уже созданные ранее модели и выполнять их на вычислительных устройствах компании Intel с гораздо большой эффективностью.

В частности, данный инструмент предоставляет возможность использования нескольких устройств для выполнения нейронных сетей, что позволяет:

- Эффективнее использовать все доступные аппаратные срЕдства в течение одного инференса (запуск готовой натренированной сети как готовой программы), что дает более стабильную производительность, так как устройства разделяют нагрузку на выводы
- Повышение пропускной способности за счет использования нескольких устройств (по сравнению с выполнением на одном устройстве).
- Использовать мощность ускорителей для обработки наиболее тяжелых частей сети и выполнять неподдерживаемые уровни на резервных устройствах.

Цель: проанализировать эффективность оптимизации OpenVINO в случае распределенной обработки данных. Задачи: изучить функционал OpenVINO; разработать демо-приложение; проанализировать эффективность Multi-device плагина; проанализировать эффективность гетерогенного плагина

3 слайд – 1. Функционал

Если выделить главную идею инструмента, то можно сказать, что *OpenVINO Toolkit* – это набор инструментов, который призван максимально эффективно запустить нейронную сеть на вашем железе, то есть выполнить какие-то различные оптимизации, чтобы все работало максимально быстро на предоставляемых устройствах.

Плюсы OpenVINO это:

1. Облегченная среда исполнения, которая отвечает только за инференс модели и не привносит за собой какие-либо модули, отвечающие за обучение и тестирование модели, в отличие от фреймворков

2. Software оптимизации – например оптимизация топологии сети

3. Оптимизации под железо Intel

Архитектуру OpenVINO глобально можно разделить на две части:

- Model Optimizer – набор скриптов, написанных на Python, которые позволяют перевести сеть, обученную в каком-то фреймворке в IR-формат. IR (Intermediate Representation) – специальный формат, разработанный Intel (промежуточное представление), с которым дальше работает Inference Engine и все инструменты, которые есть в OpenVINO Toolkit. Он состоит из xml-файла, в котором находится описание топологии сети (какие слои, какого типа, как они соединены) и bin-файла, где в бинарном формате описаны веса модели какой-либо точности (float32, float16, int8)
- Inference engine (дословно механизм инференса) – отвечает как раз таки за исполнение сетей на каком-либо железе, целевом устройстве (подгружаются плагины, которые отвечают за каждый девайс и затем сеть исполняется).

Он позволяет:

1. Прочитать модель из файлов (IR)
2. Загрузить модель в плагин, работающий с конкретным устройством
3. Отправить данные для обработки
4. Получить результаты обработки

Главная идея – это единый API для разных устройств, выпускаемых Intel, позволяющий запускать нейронные сети, оставляя при этом

возможность на каждом устройстве выполнить какую-либо тонкую настройку,

4 слайд – 1. Функционал

За счет чего OpenVINO Toolkit ускоряет работу нейронной сети:

- Оптимизация топологии сети

  Например, отдельные примитивы, такие как линейные операции (BatchNorm, ScaleShift), автоматически соединяются в свертки. Также происходит слияние других слоев.

- Понижение точности

  Точность инференса прямо влияет на производительность. Использование вычислений с меньшей разрядностью – позволяет, например, при небольшой потере точности выполнить гораздо больше операций за такт, что увеличит производительность.

- Также OpenVINO ускоряет работу нейронной сети за счет распределённой обработки данных. Данная оптимизация представлена в инструменте двумя плагинами: Multi-device, то есть выполнение на нескольких устройствах и гетерогенный.

За счет всех оптимизаций Inference Engine как раз и достигает высокой производительности.

5 слайд – 2. Демо-приложение

Чтобы исследовать оптимизации, предоставляемые OpenVINO Toolkit, было разработано демо-приложение. Первой целью разработки приложения была оценка сложности интеграции инструмента, так как Intel позиционирует данный инструмент, как о потенциально легкий в интеграции в готовое приложение. По результатам разработки были сделаны выводы о относительно легкой интеграции OpenVINO без каких-

либо трудностей. Вторая цель – это как раз-таки оценка производительности при использовании нескольких устройств для исполнения модели.

Первым этапом использования OpenVINO Toolkit является выбор модели. На данном этапе предполагается использование уже обученной каким-либо фреймворком модели, которая и будет использоваться для оптимизации.

Была использована модель, взятая из открытого источника MobileNet, обученная на фреймворке Caffe (датасет COCO), в качестве экстрактора признаков на кадре. Для использования в Inference Engine модель была преобразована с помощью Model Optimizer.

Для сравнения были выбраны следующие метрики,

- Пропускная способность – количество данных, обработанных за определенный период времени. В данном случае в качестве пропускной способности был выбран FPS. FPS – это средняя скорость обработки видеокадров (кадров в секунду)
- Задержка инференса – среднее время, необходимое для обработки одного кадра (время от начала инференса входных данных до получения результатов) в миллисекундах.

Также стоит заметить, что инструментарий OpenVINO Toolkit почти не влияет на веса нейронной сети или ее топологию, поэтому точность моделей не сравнивалась.

В качестве входных данных использовался видеопоток (1 мин.) с камеры в тренажерном зале.

6 слайд – 2. Демо-приложение

Для использования в демо-приложении была выбрана задача оценки позы нескольких людей на видео, так как данная задача является актуальной на данный момент и использование которой требует

задействования большого количества вычислительных ресурсов. Приложение также позволяет переключить устройство инференса, доступны CPU, GPU, режим нескольких устройств и гетерогенный режим. Приложение собирает заданные метрики (среднее и макс/мин).

Как уже было сказано ранее, интеграция OpenVINO в приложение достаточно проста. На слайде представлены этапы работы приложения:

1. Загрузка уже преобразованной модели в формате IR
2. Загрузка входного видео
3. Предобработка: изменение размера кадра
4. Запуск нейронной сети на целевом устройстве: по умолчанию это CPU и получение результата обработки
5. Отрисовка поз на изображении
6. Вывод конечного видео

7 слайд — 3. Анализ эффективности Multi-device

Первый график

Для анализа эффективности оптимизации инструмента в случае распределенной обработки данных было проведено сравнение исполнения сети на различных целевых устройствах. Сравнение проводилось между оригинальной моделью и моделью OpenVINO, которая работает на CPU, GPU, на CPU+GPU.

Метриками как уже было сказано ранее являются задержка инференса и FPS.

Количество экспериментов составляло 10 раз для оригинальной модели и 20 раз для модели OpenVINO. (запускал само приложение, оно само собирает метрики). Исследование показало, что использование OpenVINO позволяет получить значительный рост производительности в 7 раз по сравнению с оригинальной моделью даже просто при выполнении на CPU. (FP32)

8 слайд — 3. Анализ эффективности Multi-device

Второй график

Плагин Multi-device позволяет автоматически распределять запросы на инференс по доступным вычислительным устройствам для параллельного выполнения запросов. Доступными устройствами были CPU и встроенный GPU.

Видно, что для данной точности плагин позволяет достичь производительности минимум в полтора раза большей, чем при использовании просто CPU или GPU. Значение пропускной способности увеличилось с 18 и 26 FPS до 37 FPS.

9 слайд — 3. Анализ эффективности Multi-device

Третий график

Как уже было сказано точность модели также напрямую влияет на производительность, поэтому также было произведено сравнение модели с весами различной точности: float32, float16 и int8.

Можно сказать, что использование данного режима значительно ускоряет инференс нейронной сети по сравнению с оригинальной моделью, так как прирост производительности составил 9, 13 и 17 раз для точностей FP32, FP16 и INT8 соответственно.

10 слайд — 4. Анализ эффективности гетерогенного режима

Гетерогенный плагин, в свою очередь, может запускать разные слои на различных устройствах, но не параллельно. Данный плагин позволяет запустить наиболее требовательные к вычислениям части сети, используя мощности ускорителей. В топологии данной модели присутствует значительное количество слоев свертки, поэтому для данного эксперимента они были связаны для выполнения на GPU. Остальные слои выполнялись на CPU.

На графике представлено изменение метрик для гетерогенного режима. Естественно можно увидеть, что также замечается большой рост производительности по сравнению с оригинальной моделью. Но сравнивая использование гетерогенного режима и выполнение на единственном устройстве, можно наблюдать не такой большой рост производительности, как у Multi-device режима. Прирост производительности составил 18%. (29 FPS)

Это связано с тем, что некоторые топологии плохо поддерживаются для гетерогенного выполнения на некоторых устройства. Если передача данных из одной части сети в другую в гетерогенном режиме занимает больше времени, чем в обычном режиме, возможно, не имеет смысла выполнять их в гетерогенном режиме.

Можно сделать вывод, что применение Multi-device режима предпочтительнее, из-за намного большего прироста производительности. А гетерогенный же режим можно использовать скорее для выполнения неподдерживаемых слоев в различных топологиях.

11 слайд – Апробация работы

Код приложения размещен на github. Там же расположено руководство по использованию и видео-пример.

12 слайд – Заключение

В ходе данной исследовательской работы был изучен функционал инструмента; было написано приложение для демонстрации работы и оценки оптимизации при использовании OpenVINO; выяснено, что использование Intel OpenVINO Toolkit для распределенной обработки информации позволило сильно улучшить производительность работы

модели, как по сравнению с оригинальной, так и с выполнением на единственном устройстве.

Дальнейшие направления исследований включают в себя эксперименты для иных вычислительных устройств Intel, например, VPU или GNA, а также дальнейшее исследование гетерогенного режима для различных топологий для поиска большего прироста производительности.

Вопросы

Сравниваются модели с точностью FP32, FP16 и INT8. То есть 32-битный формат IEEE 754 с плавающей запятой одинарной точности FP32, 16-битный FP16 и модель с 8-битными целыми значениями

Дискретный GPU NVIDIA GeForce GTX 1650 не используется из-за того, что не принадлежит продуктам компании Intel.

Из графиков, можно заметить, что при точности FP32 CPU показывает лучшую производительность, а при FP16 GPU. Это связано с ограничениями поддерживаемых форматов моделей и устройств. Например, IR FP16 изначально предназначен для устройств GPU, тогда как для CPU IR FP16 обычно масштабируется до FP32 автоматически при загрузке. В настоящее время только ограниченный набор топологий может выиграть от включения модели INT8 на GPU, и в данном случае так и происходит.

Для Multi-device и гетерогенного исполнения поддерживаемые форматы моделей зависят от фактических базовых устройств. Как правило, предпочтительным является FP16, так как он наиболее

распространен и производителен. Но в данном эксперименте формат INT8 показал наилучшую производительность.

В данном случае попробуем выполнить наиболее требовательную к вычислениям часть сети на GPU, то есть слои свертки. Для этого определим какие слои относятся к данной части и установим сродство вручную, так как по умолчанию только неподдерживаемые слои выполняются на резервном устройстве:

1. Сначала получим карту связывания для конфигурации по умолчанию, где все слои выполняются на CPU.
2. Получим сеть в виде nGraph.
3. Найдем все сверточные слои в сети.
4. Присвоим каждому сверточному слою выполнение на GPU.
5. Создадим новое связывание

Ошибочные результаты могут быть вызваны какой-либо занятостью конкретного устройства в определённое время.

Плагин Multi-Device автоматически распределяет запросы на инференс по доступным вычислительным устройствам для параллельного выполнения запросов. Потенциальными преимуществами плагина Multi-Device являются:

1. Увеличение пропускной способности за счет использования нескольких устройств (по сравнению с выполнением на одном устройстве).
2. Более стабильная производительность, так как устройства разделяют нагрузку на выводы (если одно устройство слишком занято, другое может взять на себя больше нагрузки).

Гетерогенный плагин позволяет производить инференс одной сети на многих устройствах, то есть отправлять не поддерживаемые слои на другое устройство (fallback). Целями инференса сетей в гетерогенном режиме являются:

1. Использование мощности ускорителей для обработки особо тяжелых частей сети и выполнять неподдерживаемые уровни на резервных устройствах, таких как CPU.
2. Эффективнее утилизировать все доступные аппаратные средства в течение одного инференса.

Отдельные топологии плохо поддерживаются для гетерогенного выполнения на определенных устройствах или вообще не могут быть выполнены в этом режиме.

Дополнительные утилиты

1. Post-training Optimization – позволяет оптимизировать нейронную сеть с использованием целочисленной арифметики, то есть, например, перейти из FP16 в INT8.

2. Model Analyzer – позволяет узнать теоретический предел производительности, который можно достичь с вашим железом и сравнить с реальным результатом.

3. Benchmark App – позволяет с помощью специфичных методологий, разработанных Intel, измерить скорость работы сети.

4. Deployment Manager – позволяет минимизировать набор компонентов, который требуется для разработки, то есть позволяет устанавливать не весь пакет OpenVINO, а какие-то его определенные инструменты.

5. Accuracy Checker – позволяет измерить точность модели

6. Model Downloader – позволяет загружать модели из открытого зоопарка OpenVINO

Аналоги

PuzzleLib

Вообще говоря, существует еще несколько способов оптимизации сети, позволяющие при тех же ресурсах добиваться выигрыша в скорости работы сети как с минимальной потерей качества, так и без оной. Но оптимизировать с их помощью можно только инференс сети:

1. Оптимизация графа нейронной сети (fusion) - в большинстве популярных нейросетевых архитектурах стандартизирован набор используемых модулей (Conv, MaxPool, Activation, BatchNorm и т.д.). Благодаря тому что мы знаем архитектуру сети после ее обучения, мы можем оптимизировать различные комбинации модулей из архитектуры с точки зрения вычисления инференса сети;
2. Конвертация данных для вычислений в нейронных сетях в числа половинной точности (half precision) - обычно данные для вычислений в нейронных сетях в Python имеют тип float32. То есть каждое число представляется в виде дробного с определенным количеством знаков после запятой и занимает в оперативной памяти 4 байта. Зачастую, такая точность после запятой не нужна. При этом, перемножение больших матриц, состоящих из float32 элементов, является достаточно трудоемкой операцией. Перевод данных из float32 в float16 сокращает вычисления без потерь в точности работы инференса нейронной сети. Каждое число с типом float16 занимает в оперативной памяти 2 байта. В итоге, при таком подходе в лучшем случае получается ускорение примерно в 2 раза;
3. Квантизация данных для вычислений в нейронных сетях - по аналогии с предыдущим пунктом происходит конвертация данных из float32 в int8. При этом есть, в основном, незначительные потери в точности работы инференса сети. Каждое число с типом int8 занимает в оперативной памяти всего 1 байт. В итоге, при таком

подходе в лучшем случае получается ускорение инференса примерно в 4 раза.

TensorRT (Nvidia GPU)

TensorRT выполняет несколько важных преобразований и оптимизаций на графиках нейронных сетей (рисунок 2). Во-первых, устраните наличие неиспользуемых выходных слоев, чтобы избежать ненужных вычислений. Затем, где это возможно, слои свертки, смещения и ReLU объединяются, образуя единый слой. Другим типом преобразования является объединение горизонтальных слоев или агрегация слоев и разделение агрегированных слоев на их соответствующие выходные данные. Горизонтальное слияние слоев повышает производительность, комбинируя слои, используя один и тот же тензор источника и применяя одну и ту же операцию с аналогичными параметрами. Обратите внимание, чтоЭти операции оптимизации графа не изменяют базовые вычисления в графе вычислений: вместо этого они восстанавливают граф вычислений, чтобы сделать его более быстрым и более эффективным для вывода

Процессоры (CPU) – так как с большой вероятностью поддержка слоев и операций присутствует на процессоре; поддерживаются все процессоры Intel, начиная с 4-го поколения (насчет процессоров AMD: так архитектура Intel и AMD X86 – схожая, то процессоры AMD по крайней мере года полтора назад работали, то есть они позволяли на себе запускать OpenVINO и работать с сетями, однако никто естественно не гарантирует то, что все будет максимально оптимизировано, так как все таки архитектура у AMD немного другая, поэтому возможность запустить OpenVINO скорее всего будет, но насколько эффективно неизвестно)

Видеокарты (GPU) – поддерживаются встроенные видеоядра в процессоре и прирост от использования графического ускорителя (даже встроенного) значительный

FPGA (программируемая логическая интегральная схема)

VPU (процессоры машинного зрения)

GNA (процессор для обработки звука)