Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафеда электронных вычислительных машин

РЕФЕРАТ
на тему
«Архитектура ускорителя вычисления свёрточных нейронных сетей»

Выполнила: магистрант группы 355841
А.В. Деркач

Проверил: к.т.н., доцент каф. ЭВС
Н.А. Петровский

Минск 2023

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Сегодня свёрточные нейронные сети повсеместно применяются в задачах автоматизации. Нейросетевые модели позволяют решать широкий спектр задач по анализу изображений, что позволяет широко применять их на практике. К наиболее часто решаемым свёрточными нейронными сетями задачам относятся: классификация изображений, детектирование объектов на изображениях, сегментация изображений и генеративные модели.

Современные вычислительные системы исчисляются в термине производительности и эффективности. В контексте глубокого обучения и искусственного интеллекта, свёрточные нейронные сети стали незаменимым инструментом, но их вычисление требует огромных вычислительных ресурсов. Для оптимизации и ускорения обработки изображений и данных с использованием свёрточных нейронных сетей были разработаны специализированные ускорители. Архитектура ускорителей для свёрточных нейронных сетей представляет собой важную область исследований, ведь она существенно влияет на эффективность и применимость свёрточных нейронных сетей в решении ряда задач.

Технологии ускорителей для нейронных сетей включают в себя не только аппаратные компоненты, но и методы оптимизации алгоритмов, что позволяет более эффективно выполнять вычисления. Программные решения, такие как оптимизированные библиотеки для работы с глубокими нейронными сетями, также играют значительную роль в повышении производительности нейросетевых моделей.

Большое внимание уделяется разработке ускорителей, способных обрабатывать данные параллельно и использовать специализированные вычислительные блоки, такие как тензорные ядра, для эффективного выполнения операций свёртки. Такие ускорители часто интегрируются в облачные вычислительные платформы, что демократизирует доступ к высокопроизводительным вычислениям и делает их более доступными для различных отраслей, включая медицину, робототехнику, автомобильную промышленность и другие.

Кроме того, эволюция ускорителей направлена не только на увеличение скорости обработки данных, но и на снижение энергопотребления, что важно в условиях постоянного роста объемов информации и стремительного развития технологий. Это позволяет сокращать затраты на электроэнергию и снижать негативное воздействие на окружающую среду.

В данном реферате будет рассмотрена архитектура ускорителей, спроектированных для эффективного выполнения операций свёртки в нейронных сетях, и их роль в развитии современных вычислительных технологий. Также будут рассмотрены фреймворки и библиотеки, позволяющие взаимодействовать с ускорителями и оптимизировать вычисления свёрточных нейронных сетей.

1 ОСНОВЫ СВЁРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

1.1 Описание свёрточной нейронной сети

Свёрточные нейронные сети (Convolutional Neural Networks, CNN) – особые типы нейронных сетей, которые помогают компьютерам видеть и понимать изображения и видео. Такие сети имеют несколько слоев, называемых сверточными. Они позволяют CNN изучать сложные особенности и делать более точные предсказания о содержимом визуальных материалов.

CNN работают, имитируя человеческий мозг, и используют наборы правил, которые помогают компьютеру находить особенности в изображениях, понимать и интерпретировать информацию. Применяются в основном в областях компьютерного зрения и обработки изображений, таких как распознавание образов, детекция объектов, сегментация изображений, а также в различных приложениях, связанных с анализом и обработкой визуальных данных, включая медицинскую диагностику, автомобильную безопасность, распознавание лиц и другие сферы.

Каждый слой такой сети обрабатывает данные и направляет выявленные особенности следующему слою для дальнейшей обработки. В них используются фильтры, которые помогают выделить важные особенности, например края или формы объектов на изображении. Когда к визуальному материалу применяются фильтры, мы получаем свернутое изображение. Затем CNN его анализирует и выявляет важные особенности. Этот процесс называется извлечением признаков. Помимо сверточных слоев, CNN включают [1]:

- слои пулинга, которые уменьшают размер изображения, чтобы сеть могла работать быстрее и лучше обобщать данные;
- слои нормализации, которые помогают предотвратить переобучение и улучшить производительность сети;
- солносвязные слои, которые используются для классификации.

Сверточные нейронные сети работают следующим образом [1]:

- входные данные, такие как изображения или видео, поступают на входной слой;
- сверточные слои извлекают различные признаки из входных данных (используются фильтры для обнаружения границ, форм, текстур и других характеристик);
- после каждого сверточного слоя применяется функция активации ReLU, которая добавляет нелинейность и помогает улучшить производительность сети;
- далее следует слой пулинга, который уменьшает размерность карт признаков, выбирая наиболее важные значения из каждой области;
- полносвязные слои принимают выходные данные из слоя пулинга и используют набор весов для классификации или предсказания – они объединяют выделенные признаки и принимают окончательное решение.

Сверточная нейросеть учится на примерах с изображениями, которые уже имеют подписи, указывающие, что на них изображено. В процессе обучения вес фильтров и полносвязных слоев изменяется, чтобы снизить вероятность ошибок между предсказаниями сети и правильными ответами.

Когда обучение закончено, CNN может точно определить, что изображено на новых, еще не знакомых, изображениях. Она использует полученные знания о признаках и шаблонах, чтобы принять правильное решение о классификации.

Существуют 4 типа сверточных нейронных сетей [2]:

1. Традиционные CNN, также известные как «обычные», состоят из серии сверточных и субдискретизирующих слоев, за которыми следуют один или несколько полносвязных слоев. Каждый сверточный слой в такой сети выполняет свертки с использованием обучаемых фильтров для извлечения признаков из входного изображения. Примером традиционной CNN является архитектура Lenet-5, которая была одной из первых успешных сверточных нейронных сетей для распознавания рукописных цифр. Она состоит из двух наборов сверточных и субдискретизирующих слоев, за которыми следуют два полносвязных слоя. Архитектура Lenet-5 продемонстрировала эффективность CNN в идентификации изображений, и они стали широко применяться в области компьютерного зрения.

2. Рекуррентные нейронные сети (Recurrent Neural Networks, RNN) – могут обрабатывать последовательные данные, учитывая контекст предыдущих значений. В отличие от обычных нейронных сетей, которые обрабатывают данные в фиксированном порядке, RNN могут работать с входами переменной длины и делать выводы, зависящие от предыдущих входов. Рекуррентные нейросети широко используются в обработке естественного языка. При работе с текстами они могут не только генерировать текст, но и выполнять перевод. Для этого рекуррентная нейросеть обучается на парных предложениях, составленных на двух разных языках. RNN обрабатывает предложения по одному, создавая выходное предложения, которое на каждом шаге зависит от входного. Благодаря этому, рекуррентная нейросеть может правильно переводить даже сложные тексты, так как она учитывает предыдущие входы и выходы, что позволяет ей понимать контекст.

3. Полностью сверточные сети (Fully Convolutional Networks, FCN) – широко используются в задачах компьютерного зрения, таких как сегментация изображений, обнаружение объектов и классификация изображений. Они обучаются от начала до конца с использованием метода обратного распространения ошибки (backpropagation) для категоризации или сегментации изображений. Backpropagation помогает нейронной сети вычислить градиенты функции потерь по весам. Функция потерь используется для измерения того, насколько хорошо модель машинного обучения предсказывает ожидаемый результат для заданного входа. В отличие от традиционных сверточных нейронных сетей, FCN не имеют полносвязных

слоев и полностью базируются на сверточных слоях. Это делает их более гибкими и эффективными для вычислений.

4. Сеть пространственных трансформаций (Spatial Transformer Network, STN) – применяется в задачах компьютерного зрения для улучшения способности нейронной сети распознавать объекты или узоры на изображении независимо от их местоположения, ориентации или масштаба. Это называется пространственной инвариантностью. Примером использования STN является сеть, которая применяет преобразование к входному изображению перед его обработкой. Преобразование может включать выравнивание объектов на изображении, исправление перспективных искажений или другие изменения, улучшающие работу сети в конкретной задаче. STN помогает сети обрабатывать изображения, учитывая их пространственные особенности, и улучшает ее способность распознавать объекты в разных условиях.

Одно из главных преимуществ сверточных нейронных сетей – инвариантность к сдвигу. Это означает, как уже было сказано выше, что CNN может распознавать объекты на изображении независимо от их местоположения.

Еще одно преимущество – общее использование параметров. Это означает, что один и тот же набор параметров применяется для всех частей входного изображения. Такой подход позволяет сети быть более компактной и эффективной, поскольку она не должна запоминать отдельные параметры для каждой области изучаемого материала. Вместо этого она обобщает знания о признаках на всем изображении, что особенно полезно при работе с большими объемами данных.

Другие преимущества CNN включают иерархические представления, которые позволяют моделировать сложные структуры данных, и устойчивость к изменениям, что делает их надежными для разных условий изображений. Кроме того, сверточные сети могут быть обучены end-to-end, то есть обучение модели происходит на всем пути от входных данных до вывода, что ускоряет процесс обучения и повышает общую производительность сети.

CNN могут изучать разные уровни характеристик входного изображения. Верхние слои сети изучают более сложные характеристики, такие как части и формы объектов, а нижние слои — более простые элементы, например границы и текстуры. Эта иерархическая модель позволяет распознавать объекты на разных уровнях абстракции, что особенно полезно для сложных задач, таких как обнаружение объектов и сегментация.

Кроме того, CNN могут быть обучены на всей сети сразу. Это означает, что градиентный спуск (алгоритм оптимизации) может одновременно оптимизировать все параметры сети для улучшения ее производительности и быстрой сходимости. Градиентный спуск позволяет модели корректировать параметры на основе информации об ошибке, чтобы минимизировать потери в процессе обучения.

Для обучения CNN требуется большой объем размеченных данных, и оно часто занимает много времени. Это связано с высокими требованиями к

вычислительной мощности. Архитектура CNN, включающая количество и тип слоев, может влиять на производительность сети. Например, добавление большего числа слоев позволяет повысить точность модели, но и увеличивает сложность сети, а с ней и требования к вычислительным ресурсам. Глубокие архитектуры CNN также страдают от переобучения, когда сеть сосредотачивается на тренировочных данных и плохо применяет полученные знания к новым, неизвестным данным.

В задачах, где требуется контекстное понимание, например, в обработке естественного языка, сверточные сети могут иметь ограничения. Для таких задач предпочтительны другие типы нейронных сетей, которые специализируются на анализе последовательностей и учитывают контекстуальные зависимости между элементами.

Несмотря на эти недостатки, сверточные нейронные сети все еще широко применяются и демонстрируют высокую эффективность в глубоком обучении. Они являются ключевым инструментом в области искусственных нейросетей, особенно в задачах компьютерного зрения.

1.2 Структура свёрточной нейронной сети

Идея, заключённая в свёрточных нейронных сетях, состоит в попеременном использовании субдискретизирующих и свёрточных слоёв. Неокогнитрон, предшественник сверточных сетей, был представлен в работе 1980 года. В 1988 году свёрточные нейронные сети были представлены отдельно, с явными параллельными слоями и способными к обучению свертками. Их конструкция была усовершенствована 1998 году, обобщена в 2003 и того же года упрощенна. Нейронная сеть LeNet-5 успешно классифицирует цифры, и применяется для распознавания чисел в чеках. Однако при увеличении объёма распознавания выявляются проблемы в производительности из-за ограничения в вычислительных ресурсах. В 1988 году была предложена отличная конструкция CHP для разложения одномерных сигналов. В 1989 году данная сеть была видоизменена для других схем на основе свертки [3].

Сверточная нейронная сеть не имеет обратных связей и является многослойной. В большинстве случаев, в её обучении применяется метод обучения с учителем. Данная структура получила название из-за наличия так называемой операции свёртки над парой матриц A (размера $nx \times ny$) и B (размера $mx \times my$), результатом которой является матрица $C = A \times B$ размера $(nx - mx + 1) \times (ny - my + 1)$.

Работа данной модели обычно описывается как переход от определённых особенностей изображения к их абстрактным представлениям, а затем к ещё более абстрактным деталям вплоть до выявления представлений высокого уровня. При этом нейронная сеть изменяет свои настройки и

вырабатывает необходимую ротацию абстрактных признаков, отбрасывая маловажные детали и выделяя существенное [4].

Двумерная свертка – это простая операция, начинающаяся с ядра, представляющего из себя матрицу весов. Ядро производит действия над двумерным изображением, поэлементно умножая ту часть входных данных, над которой оно сейчас находится, и затем суммирует все полученные значения в один выходной пиксель. Ядро повторяет эту процедуру, преобразуя двумерную матрицу в матрицу признаков.

Признаки на выходе являются взвешенными суммами признаков на входе, расположенных примерно в том же месте, что и выходной пиксель на входном слое. Независимо от того куда попадает входной признак, он определяется в зависимости от нахождения в зоне ядра, создающего выходные данные. Следовательно, размер ядра сверточной нейронной сети определяется количеством признаков, которые будут объединены для получения нового признака на выходе [5].

Чаще всего в сверточных нейронных сетях используется техники Padding и Striding.

Padding добавляет к краям пиксели нулевого значения. Таким образом, ядро позволяет ненулевым пикселям оказываться в своем центре, а затем распространяется на нулевые пиксели за пределами края, создавая выходную матрицу того же размера, что и входная.

Striding же пропускает некоторые области, над которыми скользит ядро, в зависимости от заданного шага. Шаг означает, что берутся пролеты через пиксель, то есть по факту каждый пролет является стандартной сверткой, и чем выше шаг, тем выше сжатие.

Свёрточная нейронная сеть состоит из большого количества слоёв. Сначала идёт слой входного изображения, затем сигнал передаётся через серию свёрточных слоёв. Чередование таких слоёв даёт возможность группировать карты признаков, а на каждом следующем слое карта уменьшается в размере, однако количество каналов увеличивается. На практике это повышает способности к распознаванию сложных критериев признаков. На выходе свёрточных слоёв сети дополнительно ставят слои полносвязной нейронной сети, на вход которому подаётся результат работы свёрточных слоёв (конечные карты признаков) (см. рисунок 1.1).

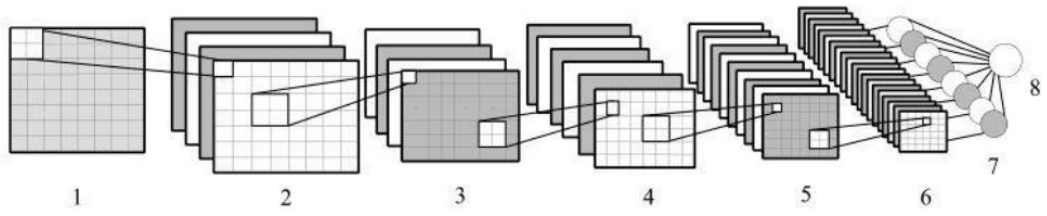

Рисунок 1.1 – Архитектура свёрточной нейронной сети: *1* – вход; *2,4,6* – свёрточные слои; *3,5* – подвыборочные слои; *7* – слои из обычных нейронов; *8* – выход

1.3 Роль ускорителей в вычислениях свёрточных нейронных сетей

Ускорители играют важную роль в вычислениях свёрточных нейронных сетей из-за их способности значительно увеличить скорость обучения и выполнения операций в этих сетях. Сверточные нейронные сети имеют сложную структуру, требующую большого количества матричных операций, включая свёртки, пулинг, активации и многие другие. Такие операции являются вычислительно интенсивными и могут занимать большое количество времени, если выполнять их на обычных процессорах.

Ускорители, такие как графические процессоры (GPU), специальные процессоры (ASIC) или даже специализированные интегральные схемы (FPGA), обладают параллельной архитектурой, которая позволяет эффективно выполнять множество вычислительных операций одновременно. Это идеально подходит для задач глубокого обучения, особенно в контексте сверточных нейронных сетей, где операции свертки и пулинга могут быть легко параллелизованы.

Глубокие нейронные сети требуют большого количества вычислительных ресурсов для обучения. Ускорители могут значительно уменьшить время обучения за счет распределения вычислительных задач между большим количеством ядер или блоков. Это позволяет исследователям и инженерам быстрее проводить эксперименты с различными моделями и гиперпараметрами.

Некоторые ускорители, такие как специализированные ASIC или FPGA, спроектированы для оптимизации энергопотребления при выполнении специфических вычислительных задач. Это может быть особенно важно в областях, где энергопотребление играет ключевую роль, таких как автономные устройства, мобильные приложения и облачные вычисления.

Существует множество программных инструментов, библиотек и фреймворков, которые позволяют эффективно использовать ускорители для глубокого обучения. Некоторые из них включают в себя TensorFlow, PyTorch, и CUDA, которые предоставляют оптимизированные функции для работы с ускорителями.

Ускорители могут быть настроены для выполнения специфических операций, часто встречающихся в сверточных нейронных сетях, таких как операции свертки, активации и пулинга. Это позволяет им выполнять эти операции более эффективно и быстрее, чем общепроцессорные решения.

Благодаря постоянному развитию технологий ускорителей, исследователи постоянно ищут новые способы улучшения производительности глубокого обучения. Это включает в себя разработку новых архитектур, алгоритмов и методов оптимизации, способствуя появлению более эффективных и мощных ускорителей.

2 АППАРАТНАЯ АРХИТЕКТУРА УСКОРИТЕЛЕЙ СВЁРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

2.1 Общая структура аппаратных ускорителей вычисления свёрточных нейронных сетей

Аппаратные ускорители вычислений для сверточных нейронных сетей могут иметь различные архитектуры, но общая структура обычно включает следующие основные компоненты:

Ядра вычислений. Ускорители обычно включают специализированные вычислительные блоки для выполнения операций свертки и пулинга. Они могут быть оптимизированы для выполнения матричных операций и других вычислительно интенсивных операций, связанных со сверточными слоями. Ядра вычислений включают в себя матричные умножители (используются для выполнения операций свертки и других матричных операций, таких как умножение и суммирование.) или специализированные вычислительные блоки (тензорные ядра или блоки, спроектированные для выполнения конкретных операций нейронных сетей.).

Память и кэш. Ускорители обычно имеют специализированную память для хранения весов нейронной сети, промежуточных данных и промежуточных результатов вычислений. Оптимизированные кэши могут быть использованы для уменьшения задержек при доступе к данным. Как правило, этот компонент состоит из встроенной памяти (для быстрого доступа к весам, активациям и промежуточным данным) и буфера для промежуточных данных (хранят промежуточные результаты вычислений).

Контроллеры и блоки управления – осуществляют управление данными и выполнение инструкций.

Оптимизированные связи и шины – используются для передачи данных между компонентами, чтобы обеспечить эффективную работу ускорителя.

Оптимизированные вычисления. Ускорители могут включать в себя специализированные инструкции или архитектуры для улучшения производительности выполнения операций, характерных для нейронных сетей.

Поддержка параллелизма. Ускорители обычно используют параллельные вычисления для увеличения скорости обработки. Это может включать в себя параллельные вычисления на нескольких вычислительных ядрах, использование многопоточности или даже распределенные вычислений.

Оптимизация энергопотребления. Некоторые ускорители могут быть спроектированы для оптимизации энергопотребления, используя эффективные алгоритмы и архитектуры, чтобы снизить энергозатраты при выполнении вычислений нейронных сетей.

Программируемость и гибкость. Часть ускорителей обладают гибкостью настройки и программирования, позволяя пользователям оптимизировать и адаптировать их под конкретные задачи и архитектуры нейронных сетей.

Драйверы и программные интерфейсы – используются для взаимодействия с вычислительным устройством.

2.2 Архитектура вычисления свёрточных нейронных сетей на CPU

Центральный процессор (CPU) – электронный блок либо интегральная схема, исполняющая машинные инструкции (код программ), главная часть аппаратного обеспечения компьютера или программируемого логического контроллера.

В 1980-е годы для обучения и выполнения первых нейронных сетей чаще всего применялся центральный процессор (CPU), подходящий для общих вычислений. Однако в пользу общности жертвовалось производительностью специфичных вычислений, например, векторных. Общая схема работы нейронной сети на CPU представлена на рисунке 2.1.

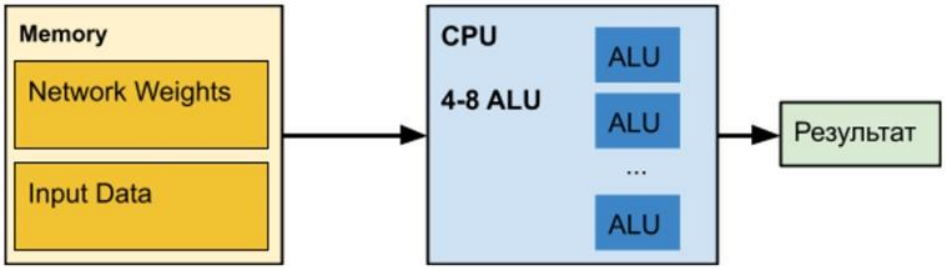

Рисунок 2.1 – Схема выполнения вычислений CNN на CPU

Когда мы работаем с нейросетями, особенно глубокими, у нас непосредственно сеть может занимать сотни мегабайт. К примеру, требования к памяти сетей обнаружения объектов приведены в таблице 2.1.

Таблица 2.1 — Требование к памяти нейронных сетей обнаоужения объектов

Модель	**Входной размер**	**Память параметров**	**Используемая память**
rfcn-res50-pascal	600 x 850	122 Мб	1 Гб
rfcn-res101-pascal	600 x 850	194 Мб	2 Гб
ssd-pascal-vggvd-300	300 x 300	100 Мб	116 Мб
ssd-pascal-vggvd-512	512 x 512	104 Мб	337 Мб
ssd-pascal-mobilenet-ft	300 x 300	22 Мб	37 Мб
faster-rcnn-vggvd-pascal	600 x 850	523 Мб	600 Мб

При этом график задержки при обращении к памяти в зависимости от размера данных представлен на рисунке 2.2.

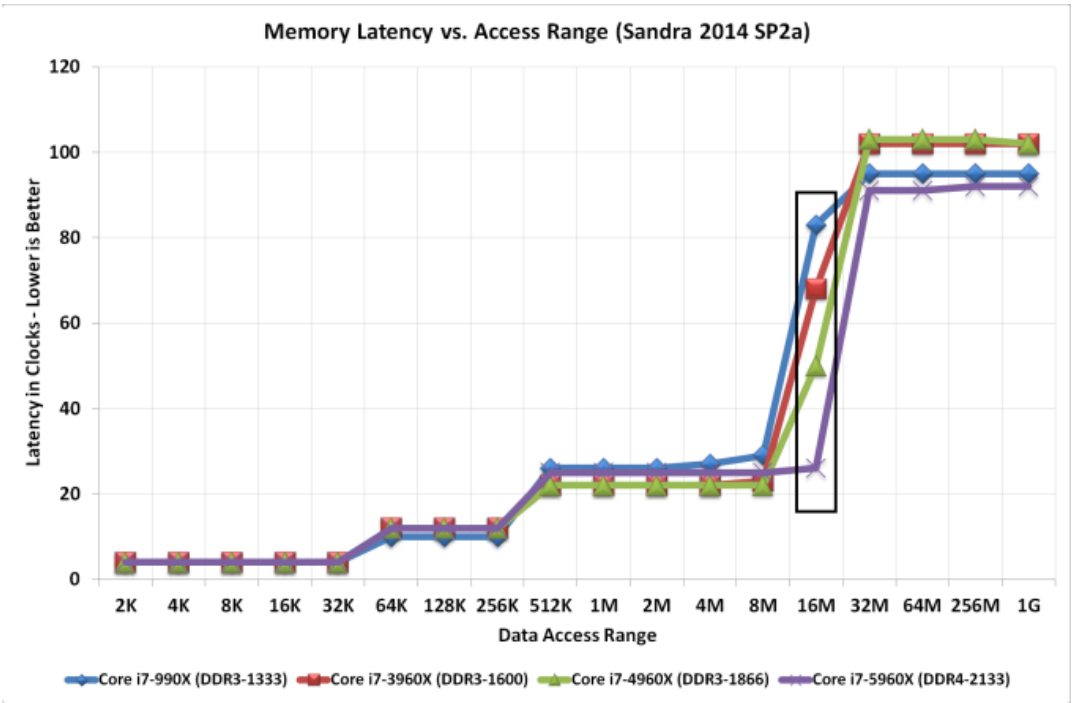

Рисунок 2.2 – График задержек при обращении к памяти на CPU

Как мы видим, при увеличении объема данных больше 16 Мб задержка возрастает в 50 и более раз, что фатально сказывается на производительности. Фактически большую часть времени CPU при работе с глубокими нейросетями ждет данных. Интересны данные Intel по ускорению разных сетей, где, фактически, ускорение идет, только когда сеть становится маленькой (например, в результате квантования весов), чтобы начать хотя бы частично входить в кэш вместе с обрабатываемыми данными. Заметим, что кэш современного CPU потребляет до половины энергии процессора. В случае тяжелых нейросетей он оказывается малоэффективен.

Таким образом, к плюсам использования CPU при работе с нейронными сетями можно отнести то, что он есть в каждом устройстве, т.е. относительно низкая входная цена расчетов и внедрения. Также есть отдельные не CV сети, которые хорошо работают на CPU, например, Wide&Deep и GNMT.

Главным минусом является то, что CPU неэффективен при работе с глубокими нейросетями (когда число слоев сети и размер входных данных велики).

2.3 Архитектура вычисления свёрточных нейронных сетей на GPU

Графический процессор (GPU) – отдельное устройство персонального компьютера или игровой приставки, выполняющее графический рендеринг; в

начале 2010-х годов графические процессоры стали массово применяться и в других устройствах: планшетные компьютеры, встраиваемые системы, цифровые телевизоры. GPU получает на вход полигоны, а после проведения над ними необходимых математических и логических операций выдаёт координаты пикселей. По сути, работа GPU сводится к оперированию над огромным количеством независимых между собой задач. Поэтому он содержит огромное количество исполнительных блоков – в современных GPU их 2048 и более. Внешне он похож на центральный процессор (CPU), но его архитектура отлична от архитектуры CPU:

– В GPU содержатся несколько сотен вычислительных ядер, с помощью которых сложные расчеты выполняются очень быстро. Энергии при таких вычислениях потребляется значительно меньше, чем при работе CPU.

– В GPU обработка графики в видеороликах, графических программах, играх выполняется быстрее и эффективнее благодаря разделению процессов. Это позволяет разгрузить CPU для других задач.

– GPU может взять на себя некоторые вычисления вместо процессора. Он позволяет выполнять расчеты с плавающей точкой или расчеты с одинаковой или сходной формулой.

– CPU разделяет потоки и выполняет сразу несколько процессов, когда пользователь работает с несколькими программами одновременно.

Как было описано в разделе 2.2, CPU неэффективен при работе с глубокими нейросетями. Ситуация изменилась с появлением GPU, аппаратная архитектура которых нацелена на работу с 3D объектами и одновременной работой с множеством данных. Известным примером является AlexNet [7], являющийся одной из первых сверточных нейронных сетей, пригодной к применению на GPU и позволившей выиграть соревнование по распознанию изображений ImageNet. Это дало толчок к развитию как самих GPU, так и к поиску других аппаратных ускорителей. Общая схема работы нейронной сети на GPU представлена на рисунке 2.3.

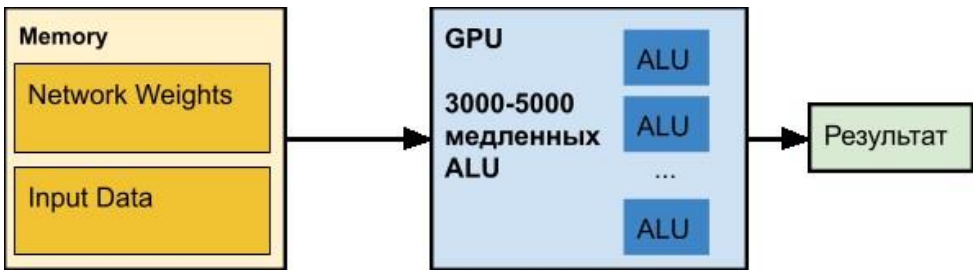

Рисунок 2.3 – Схема выполнения вычислений CNN на GPU

Важнее то, что память у GPU изначально рассчитана под примерно в 5 раз более высокую производительность чем у CPU. В нейросетях вычисления с данными крайне простые. Несколько элементарных действий, и нужны новые данные. Как следствие, скорость доступа к данным является критичной

для эффективной работы нейросети. Высокоскоростная память GPU и более гибкая система управления кэш-памятью, чем на CPU, позволяет решить проблему с задержками в доступе к памяти. График скорости при обращении к памяти представлен на рисунке 2.4.

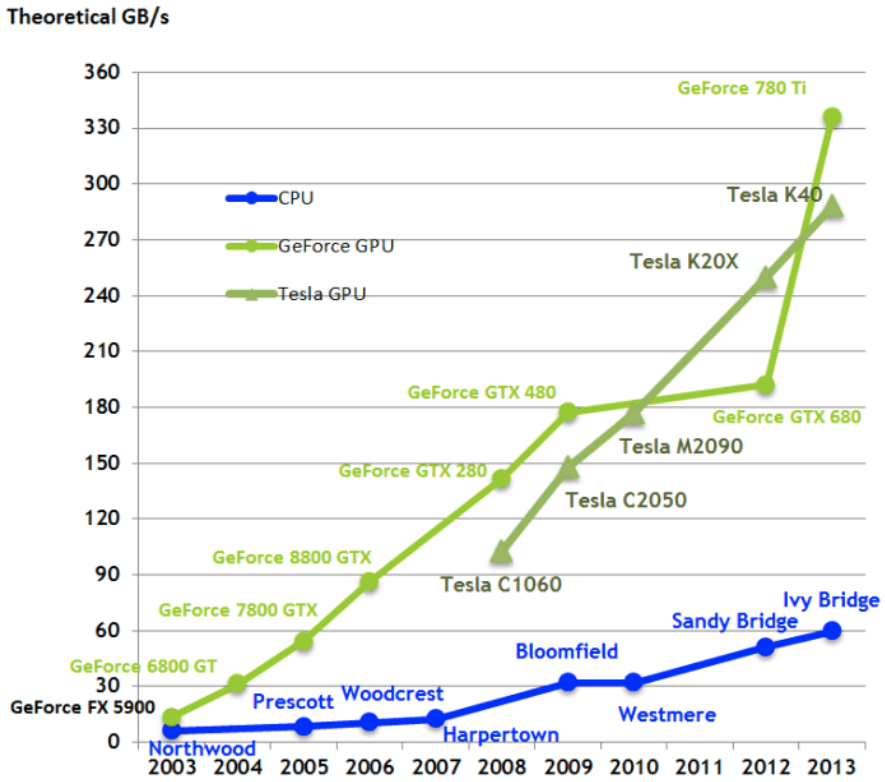

Рисунок 2.4 – График сравнения скоростей обращения к памяти на GPU и CPU

Успех сверточных нейронных сетей, накопление большого объема данных создавало запрос на улучшение аппаратных возможностей. Большинство нейронных сетей оперируют сверточными слоями, полносвязными слоями, а также различными функциями активации между ними. Поскольку и сверточные слои, и полносвязные слои можно представить как операцию матричного или тензорного умножения, одно из направлений ускорения была работа именно над оптимизацией этой операции. Так, в 2018 году NVIDIA презентовала новое поколение графических ускорителей серии RTX 2xxx, особенностью которой являлось наличие так называемых тензорных ядер [8], выполняющих операцию сплавленного сложения и умножения в матричном виде.

Таким образом, к плюсам использования GPU при работе с нейронными сетями можно отнести кардинальное (в 10–100 раз) ускорение работы по сравнению с CPU. А также то, что GPU эффективны для обучения (и несколько менее эффективны для применения).

Главным минусом является то, что хорошие GPU (памяти на которых достаточно для обучения сетей большого размера) относятся к высокому ценовому сегменту.

2.4 Архитектура тензорных вычислительных модулей

Поскольку основной специализацией графического процессора является графика, а не вычисления, большего ускорения можно достичь за счет дальнейшей специализации. Это привело к созданию и использованию тензорных вычислительных модулей (TPU) [9]. Этот тип устройств отличаются характерным высоким соотношением производительности к единице потребляемой энергии.

Тензорные процессоры – устройства, как правило, являющиеся сопроцессорами, управляемыми центральным процессором. Тензорные процессоры оперируют тензорами – объектами, при помощи которых удобно выполнять преобразование элементов одного векторного пространства в другое, и которые могут быть представлены как многомерные массивы чисел, обработка которых осуществляется с помощью таких программных библиотек, как, например TensorFlow и Caffe 2. В угоду производительности они, обычно, выполняют операции над числами малой разрядности (8 или 16 бит) и специализированы для быстрого выполнения таких операций, как матричное умножение и свёртка, используемых для эмуляции свёрточных нейронных сетей, которые используются для задач машинного обучения. График сравнения производительности выполнения сети на CPU, GPU и TPU представлен на рисунке 2.5. Как видно, TPU производительней по сравнению с GPU в 10-30 раз.

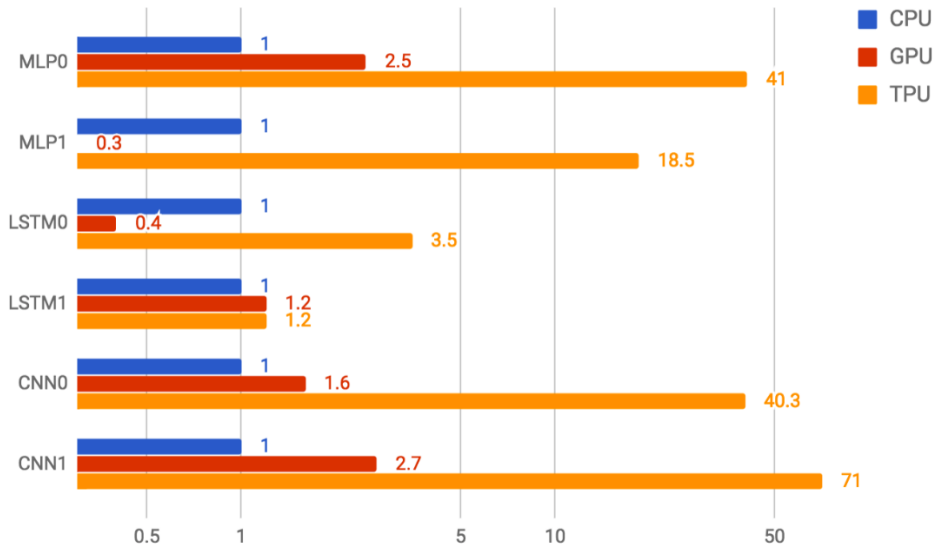

Рисунок 2.5 – График сравнения скоростей обращения к памяти на GPU и CPU

Одно TPU ядро состоит из множества матричных вычислительных блоков, число которых зависит от версии, а также одного векторного и скалярного блоков. Ядра оперируют числами с пониженной точностью, в частности 8 или 16 бит, что позволяет упростить оборудование. Вместе с

использованием квантования параметров и вычислений это позволяет значительно ускорить вычисления с небольшой потерей точности. Также, TPU ядра могут масштабироваться и объединяться в кластер для повышения эффективности.

В сравнении с обычными CPU и GPU архитектура Google в машинном превосходит их в десятки раз. Для примера, процессор Haswell Xeon E5-2699 v3 с 18 ядрами на тактовой частоте 2,3 ГГц с 64-битной плавающей точкой выполняет 1,3 тера-операций в секунду (TOPS) и показывает скорость обмена с памятью 51 ГБ/с. При этом сам чип потребляет 145 Вт, а вся система на нём с 256 ГБ памяти — 455 Вт.

Для сравнения, TPU на 8-битных операциях с 256 ГБ внешней памяти и 32 ГБ собственной памяти демонстрирует скорость обмена с памятью 34 ГБ/с, но при этом карта выполняет 92 TOPS, то есть примерно в 71 раз больше, чем процессор Haswell. Энергопотребление сервера на TPU составляет 384 Вт.На рисунке 2.6 представлена схема TPU Google 2-й и 3-й версии.

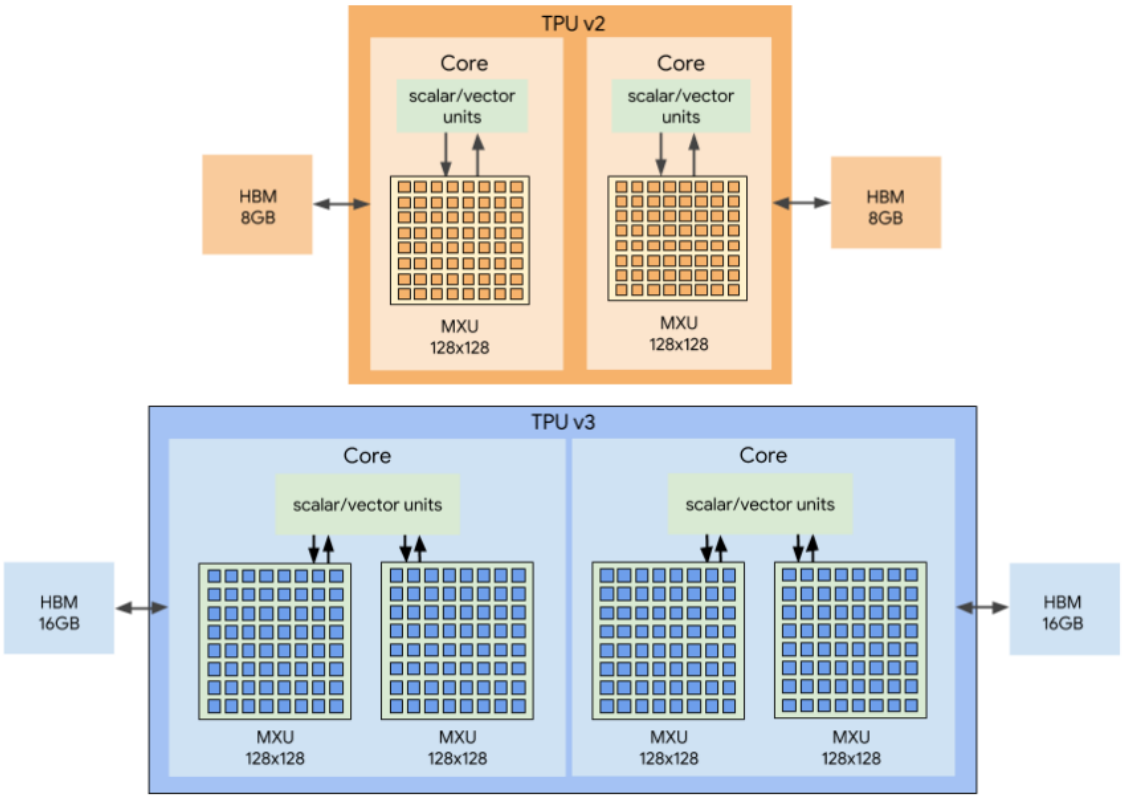

Рисунок 2.6 – Схема TPU Google 2-й и 3-й версии

TPU – это либо аппаратное ускорение обучения, либо относительно универсальное ускорение работы произвольной сети. Пока используются тензоры ранга 2 с Mixed Multiply Unit (MXU) соединенных с высокоскоростной памятью (High-Bandwidth Memory – HBM).

На рисунке 2.7 представлена полная архитектура TPU за исключением памяти DDR3.

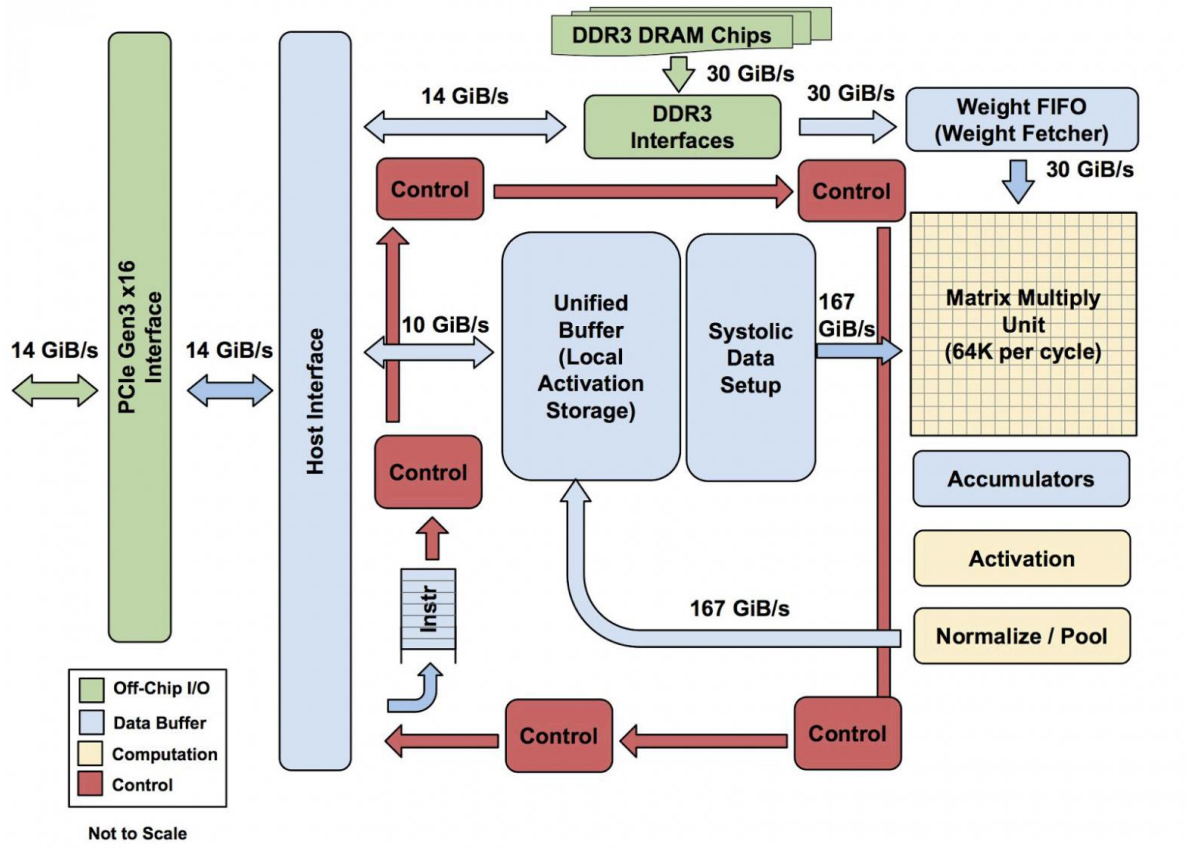

Рисунок 2.7 – Схема архитектуры TPU за исключением памяти DDR3

2.5 Архитектура нейронных процессоров

Нейронный процессор (Neural Processing Unit, NPU) – это специализированный класс микропроцессоров и сопроцессоров (часто являющихся специализированной интегральной схемой), используемый для аппаратного ускорения работы алгоритмов искусственных нейронных сетей, компьютерного зрения, распознавания по голосу, машинного обучения и других методов искусственного интеллекта [6].

Нейронные процессоры относятся к вычислительной технике и используются для аппаратного ускорения эмуляции работы нейронных сетей и цифровой обработки сигналов в режиме реального времени. Как правило, нейропроцессор содержит регистры, блоки памяти магазинного типа, коммутатор и вычислительное устройство, содержащее матрицу умножения, дешифраторы, триггеры и мультиплексоры.

NPU используются для выполнения нейронных сетей на маленьких устройствах, таких как смартфоны или Edge устройства. Для NPU также характерно использование вычислений с пониженной точностью в 8, 16 бит или даже под-байтовой точностью, а также использование SIMD/векторных блоков. На рисунке 2.8 представлена архитектура NPU от Sumsung.

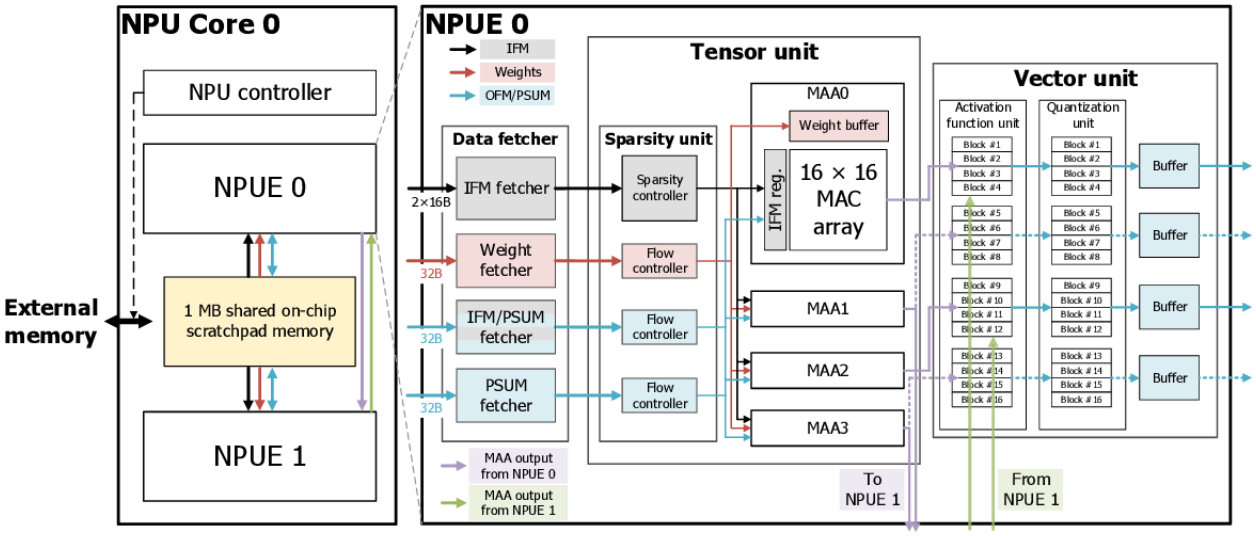

Рисунок 2.8 – Архитектура NPU Sumsung

2.6 Архитектура ускорителей FPGA

Программируемая вентильная матрица (Field Programmable Gate Array, FPGA) – полупроводниковое устройство, которое может быть сконфигурировано производителем или разработчиком после изготовления; наиболее сложная по организации разновидность программируемых логических интегральных схем. Общая схема работы нейронной сети на FPGA представлена на рисунке 2.9.

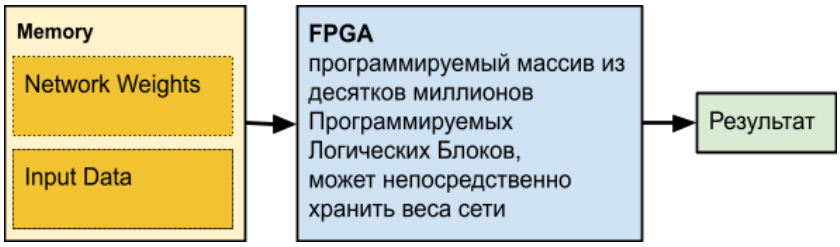

Рисунок 2.9 – Общая схема выполнения вычислений CNN на FPGA

FPGA – это сеть из нескольких миллионов программируемых блоков, которые мы можем также программно соединять между собой. Ключевой плюс FPGA в том, что мы можем хранить сеть непосредственно в ячейках, тоесть проблема в виде перегоняемых 25 раз в секунду (для видео) в одном и том же направлении сотен мегабайт одних и тех же данных пропадает. Это позволяет при более низкой тактовой частоте и отсутствии кэшей вместо понижения производительности получить заметное повышение и кардинально снизить энергопотребление на единицу вычислений.

Поскольку FPGA используются для создания конфигурируемых цифровых электронных схем любой архитектуры, в том числе параллельных, то они отлично подходят в качестве ускорителя для создания нейронных сетей.

К преимуществам данного подхода можно отнести низкое потребление энергии, высокие вычислительные мощности и гибкость [10].

1. FPGA могут использоваться для высокоскоростной обработки сигналов, в особенности для датчиков с высокой частотой дискретизации, а также фильтровать данные и снижать скорость передачи данных, что упрощает обработку, передачу и хранение сложного сигнала.

2. FPGA в высокой степени подходят для обработки данных в режиме реального времени, таких как изображения, радиолокационные и медицинские сигналы, более эффективно, чем центральные процессоры CPU.

3. Важными особенностями FPGA являются параллелизм и конвейеризация. Благодаря параллелизму можно многократно распределять и вычислять ресурсы, когда несколько модулей могут работать независимо, одновременно. Конвейеризация делает аппаратные ресурсы многоразовыми. Эти два фактора вместе могут значительно улучшить параллельную производительность.

Структура блоков ускорителя CNN на основе FPGA представлена на рисунке 2.10.

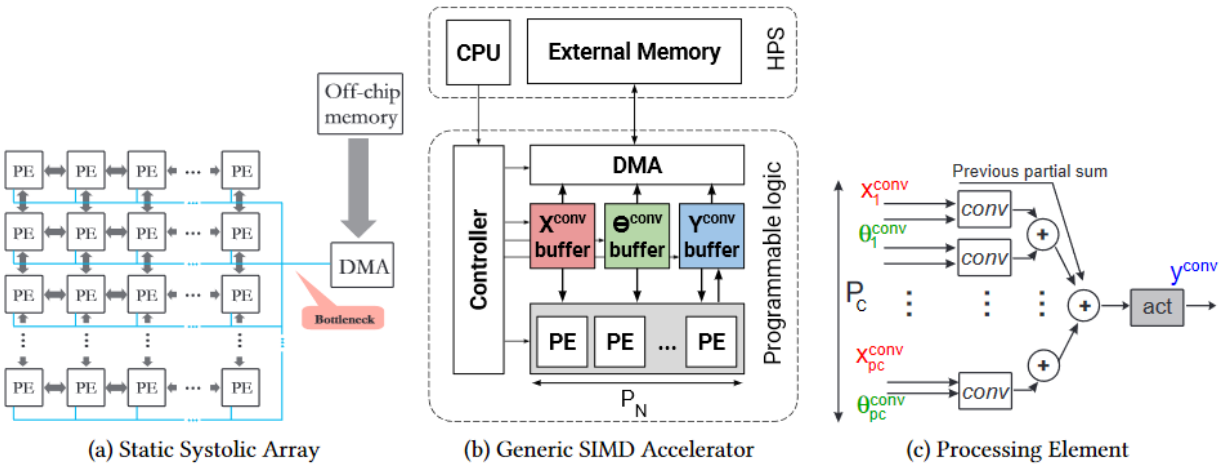

Рисунок 2.10 – Структура блоков ускорителя CNN на основе FPGA

Поскольку большинство FPGA реконфигурируемы, это позволяет быстро усовершенствовать и оптимизировать аппаратную реализацию без дополнительных затрат на её производство. Следует отметить, однако, что такая универсализация накладывает ограничения на максимальную производительность, и в ряде задач FPGA не способны на равных конкурировать с процессорами специального назначения. Так, Nurvitadhi и др. (2017) показали, что графические процессоры выигрывают по скорости у FPGA в операциях над числами с плавающей точкой, хотя при этом проигрывают по эффективности относительно энергопотребления. Авторам удалось достичь сопоставимой скорости обработки чисел с плавающей точкой на Titan X Pascal и Intel Stratix 10. По энергоэффективности FPGA оказалась

эффективнее примерно на 40%, что, впрочем, с учётом стоимости микросхемы преимуществом является очень незначительным.

Естественно, применять FPGA имеет смысл уже на этапе применения нейросети (для обучения в большинстве случаев маловато памяти). Причем тема выполнения на FPGA сейчас начала активно развиваться. Например, вот фреймворк fpgaConvNet, помогающий заметно ускорить применение CNN на FPGA и снизить при этом энергопотребление.

К плюсам использования ускорителя CNN на основе FPGA можно отнести:

– Потенциально возможно более быстрое выполнение сети.

– Заметно ниже энергопотребление по сравнению с CPU и GPU (особенно это важно для мобильных решений).

К минусам относится:

– В основном помогают с ускорением выполнения, обучать на них, в отличие от GPU, заметно менее удобно.

– Более сложное программирование по сравнению с другими вариантами.

2.7 Архитектура ускорителей ASIC

ASIC (Application-Specific Integrated Circuit, «интегральная схема для конкретного применения») – интегральная схема, специализированная для решения конкретной задачи. В отличие от обычных интегральных схем для общего назначения, специализированные интегральные схемы применяются в конкретном устройстве и выполняют строго ограниченные функции, характерные только для данного устройства; вследствие этого выполнение функций происходит более эффективно и, в конечном счёте, дешевле. В контексте нейронных сетей создаются специализированные ASIC для оптимизации вычислений в них. Общая схема работы нейронной сети на ASIC представлена на рисунке 2.11.

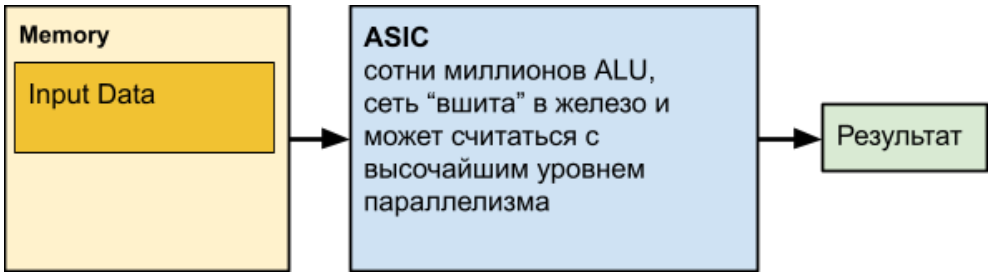

Рисунок 2.11 – Общая схема выполнения вычислений CNN на ASIC

К плюсам использования ускорителя CNN на основе ASIC можно отнести:

– Самая низкая стоимость чипа по сравнению со всеми другими решениями.

– Самое низкое энергопотребление на единицу операций.

– Высокая скорость работы (в том числе, при желании, рекордная).

Из минусов можно выделить следующее:

– Очень ограничены возможности обновления сети и логики.

– Самая высокая стоимость разработки по сравнению со всеми другими решениями.

Таким образом, использование ASIC рентабельно в основном при больших тиражах.

2.8 Аналоговые ускорители

Аналоговые ускорители вычисления для сверточных нейронных сетей представляют собой специализированные аппаратные устройства, разработанные для оптимизации выполнения операций над данными в нейронных сетях, в частности, в CNN. Они отличаются от традиционных цифровых ускорителей за счет использования аналоговых вычислений вместо цифровых. Это позволяет им решать некоторые проблемы, связанные с энергоэффективностью и скоростью вычислений.

Сейчас аналоговые ускорители не применяются на практике, но когда-то операции умножения и сложения выполнялись электронными лампами и транзисторами. Так как нейронная сеть относительно устойчива к неточным вычислениям внутри, то есть возможность перевести эти вычисления в аналоговый вид. Из-за этого мы получим заметное ускорение вычислений и потенциально кардинальное снижение расхода энергии на выполнение одной операции, как показано на рисунке 2.12.

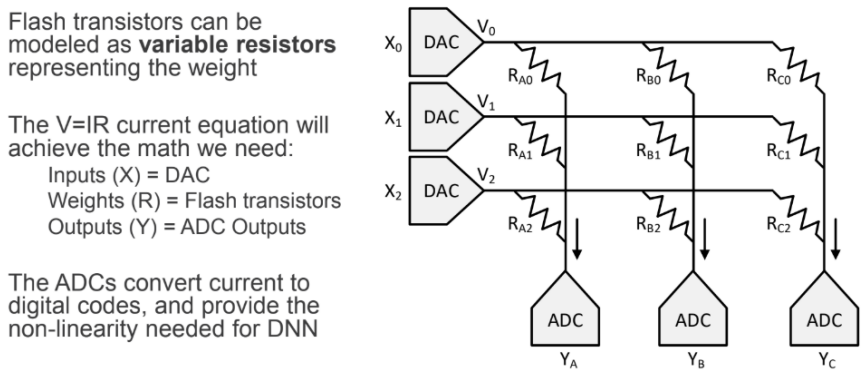

Рисунок 2.12 – Аналоговый ускоритель CNN

При таком подходе CNN вычисляются быстро и энергоэффективно. Но есть проблема – это ЦАП/АЦП (DAC/ADC) – преобразователи из цифрового представления в аналоговое и обратно, которые уменьшают и энергоэффективность, и точность процесса.

3 ПРОГРАММНАЯ АРХИТЕКТУРА УСКОРИТЕЛЕЙ СВЁРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ

3.1 Технология CUDA

Особенностью оборудования, поддерживающего технологию CUDA (Compute Unified Device Architecture – унифицированная архитектура вы числительного устройства), является возможность обеспечивать на порядок большую (по сравнению с кластерами) пропускную способность при работе с памятью [11].

В графических ускорителях NVIDIA, начиная с восьмой серии, реализована архитектура параллельных вычислений CUDA, которая предоставляет специализированный программный интерфейс для не графических вычислений.

Логически графический процессор с поддержкой CUDA можно рассматривать как набор многоядерных процессоров. Основными вычислительными блоками таких видеочипов являются мультипроцессоры, которые состоят из восьми ядер, нескольких тысяч 32-битных регистров, 16 Кбайт общей памяти, текстурного и константного кэшей.

До официального появления технологии CUDA проводились эксперименты по использованию графических карт настольных систем для реализации потоковых вычислений. Так, с помощью графических программно-аппаратных интерфейсов и представления данных в качестве массивов текстур, удалось добиться троекратного увеличения производительности в экспериментах. Стоит отметить также и одно из ключевых достоинств технологии CUDA – отсутствие необходимости в разработке программ следовать графическим «метафорам» – типам данных и принципам построения вычислений, характерным исключительно для обработки вершин и пикселей при построении кадра.

Реализация нейронной сети с сочетанием CUDA, OpenMP и параллелизма центральных процессоров на уровне языка программирования и программно-аппаратного интерфейса создает еще один уровень прироста производительности.

Запуск ядра CUDA порождает процессы на GPU в соответствии с заданным параметрами (см. рисунок 3.1). Всё множество процессов, порождаемых запуском ядра, в терминологии CUDA называться грид (grid). Грид состоит из блоков (block), блок из процессов (thread). Thread – это элементарный параллельный процесс. Процессы в блоках и блоки в гриде могут быть представлены в виде одномерной, двухмерной или трёхмерной решетки.

Для FX1700, например, максимальный размер грида 65535 × 65535 блоков. Максимально возможные значения индексов номера процесса в блоке 512 × 512 × 64, но при этом количество процессов в блоке не должно превышать 512.

Ядра можно запускать асинхронно, тогда несколько гридов будут выполняться на GPU параллельно.

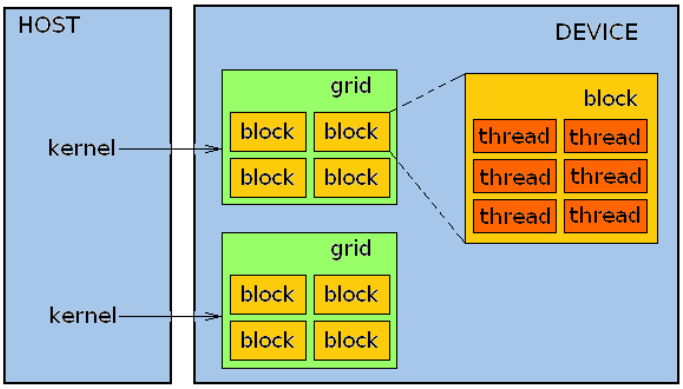

Рисунок 3.1 – Схема организации параллельных процессов CUDA

При конфигурировании топологии процессов необходимо учитывать аппаратные особенности. Здесь следует ввести понятие варпа. Варп (warp) это группа тредов. Например, размер варпа для FX1700 – 32 процесса. Полуварп (half-warp) – половина тредов варпа. Все процессы одного варпа выполняются одновременно и синхронно (SIMD) на своём мультипроцессоре. При доступе треда к основной памяти GPU её части могут кэшироваться в локальной памяти данного мультипроцессора. Если все данные, которые требуются процессам полуварпа будут находиться в этом кэше то это может повысить производительность.

GPU имеет сложно организованную память, помимо основной (или глобальной) памяти каждый мультипроцессор обладает своей памятью. Программная модель памяти CUDA представлена на рисунке 3.2.

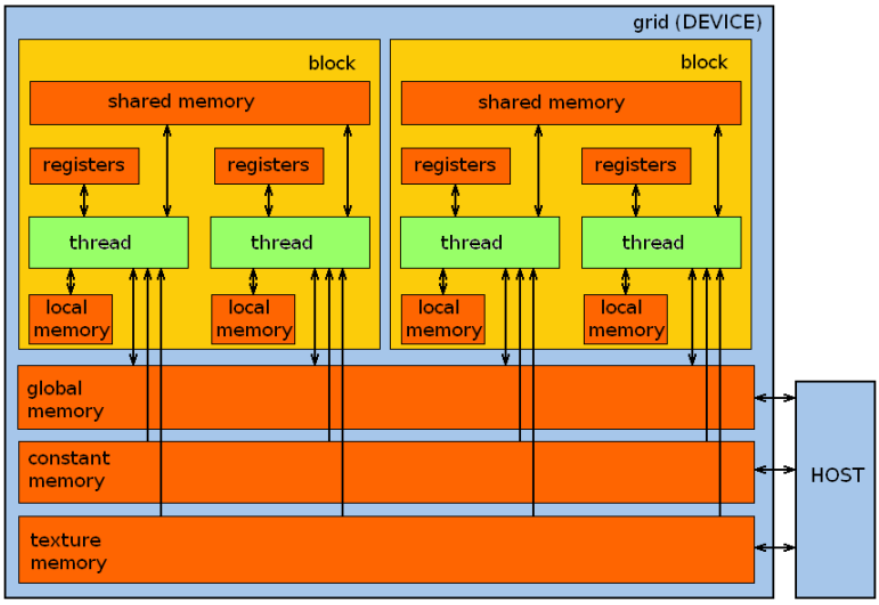

Рисунок 3.2 – Схема организации памяти CUDA-программы

На сегодня тема CUDA является перспективной и востребованной. CUDA широко применяется для решения задач обработки изображений, машинного обучения и инженерных расчётов, часто позволяя относительно недорого и без громоздкой аппаратуры обеспечить удовлетворительную производительность.

3.2 Технология OpenCL

OpenCL (Open Computing Language) – фреймворк для написания компьютерных программ, связанных с параллельными вычислениями на различных графических и центральных процессорах, а также FPGA. В OpenCL входят язык программирования, который основан на стандарте языка программирования Си C99, и API. OpenCL обеспечивает параллелизм на уровне инструкций и на уровне данных и является осуществлением техники GPGPU. OpenCL является полностью открытым стандартом, его использование не облагается лицензионными отчислениями.

Цель OpenCL состоит в том, чтобы дополнить открытые отраслевые стандарты для трёхмерной компьютерной графики и звука – OpenGL и OpenAL, соответственно, – возможностями GPU для высокопроизводительных вычислений. OpenCL разрабатывается и поддерживается некоммерческим консорциумом Khronos Group, в который входят много крупных компаний, включая AMD, Apple, ARM, Intel, Nvidia, Sony Computer Entertainment и другие.

Центральным элементом платформы OpenCL выступает понятие хоста (host) – первичного устройства, которое управляет OpenCL-вычислениями и осуществляет все взаимодействия с пользователем. Хост всегда представлен в единственном экземпляре, в то время как OpenCL-устройства (devices), на которых выполняются OpenCL-инструкции могут быть представлены во множественном числе. OpenCL-устройством может быть CPU, GPU, DSP или любой другой процессор в системе, поддерживающийся установленными в системе OpenCL-драйверами.

OpenCL-устройства логически делятся на вычислительные модули (compute units), которые в свою очередь делятся на обрабатывающие элементы (processing elements). Вычисления на OpenCL-устройствах в действительности происходят на обрабатывающих элементах [13]. На рисунке 3.3 схематически изображена OpenCL-платформа из 3-х устройств.

С хостом неразрывно связано понятие хостовой программы (host program) – программного кода, выполняющегося только на хосте. OpenCL не указывает как именно должна работать хостовая программа, а лишь определяет интерфейс взаимодействия с OpenCL-объектами.

Взаимодействие между хостом и OpenCL-устройством происходит посредством команд, помещенных в командную очередь (command-queue). Данные команды ожидают в командной очереди своего выполнения на OpenCL-устройстве. Командная очередь создается хостом и сопоставляется

одному OpenCL-устройству после того, как будет определен контекст. Команды делятся на те, что отвечают за: выполнение ядер, управление памятью и синхронизацию выполнения команд в очереди. Команды могут выполняться последовательно (in-order execution) или внеочередно (out-of-order execution). Второй вариант организации очередей поддерживается не всеми платформами, о чем необходимо помнить.

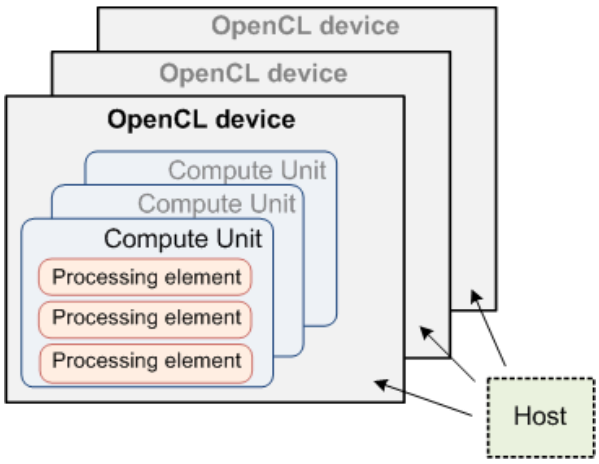

Рисунок 3.3 – Схематическое представление OpenCL-платформы

Экземпляр ядра носит название work-item. Множество всех work-item разбивается на группы. Такая группа носит название work-group. OpenCL-объекты, инкапсулирующие регионы памяти, носят название объектов памяти (memory objects). Объекты памяти бывают двух типов: буферные объекты (buffer objects) и объекты изображения (image objects).

Буферные объекты памяти инкапсулируют непрерывные участки памяти, доступные ядрам во время выполнения. Программист обычно производит отображение структур данных на данные объекты, а в коде ядра получает доступ к данным структурам посредством указателей.

Объекты изображений ограничены хранением изображений. При этом как именно изображение хранится не определяется стандартом и скрыто от программиста. Обычно хранение и доступ к изображению оптимизирован под конкретную аппаратную платформу.

Стандарт описывает следующие пять различных регионов памяти [13].

1. Память хоста (host memory) – доступна лишь с хоста.

2. Глобальная память (global memory) – определяется в памяти, доступной на чтение и запись для всех work-item во всех work-group. Чтение и запись в глобальную память может кешироваться, если OpenCL-устройство поддерживает данную возможность. В случае CPU глобальной является оперативная память. В подавляющем большинстве случаев глобальная (и константная) память самая медленная, так что использовать без необходимости ее не стоит.

3. Константная память (constant memory) – глобальный регион памяти, который инициализируемые хостом и из которого work-item может лишь читать данные.

4. Локальная память (local memory) – доступна лишь в пределах одной work-group. Все work-item в данной work-group могут как читать, так и писать в данный регион памяти одновремено.

5. Приватная память (private memory) – доступна лишь одному work-item.

На рисунке 3.4 показаны отношения между перечисленными регионами памяти согласно стандарту OpenCL.

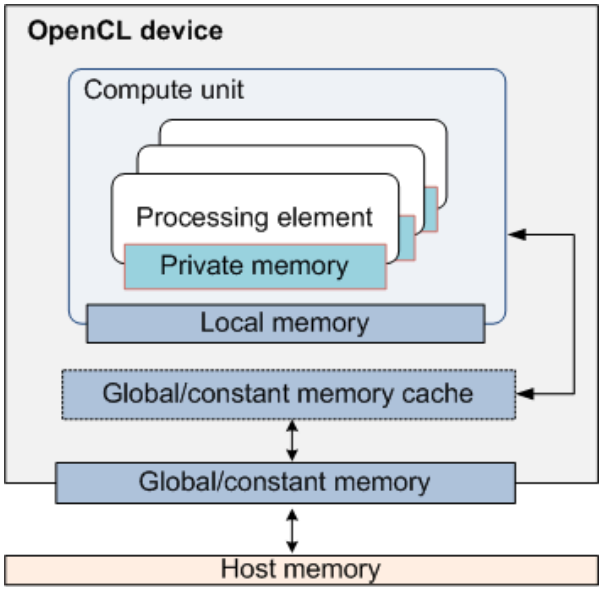

Рисунок 3.4 – Схематическое представление уровней памяти в OpenCL

OpenCL играет важную роль в ускорении сверточных нейронных сетей, позволяя эффективно использовать вычислительные ресурсы различных устройств. Это открывает возможности для быстрой обработки больших объемов данных, улучшая производительность и сокращая время обучения и выполнения моделей глубокого обучения. Однако, при использовании OpenCL необходимо учитывать особенности конкретных устройств и оптимизировать код для максимальной эффективности вычислений.

3.3 Фреймворки и библиотеки

Существует несколько фреймворков и библиотек, специально разработанных для использования ускорителей в вычислениях свёрточных нейронных сетей. Вот 5 наиболее распространенных из них:

1. CNTK – это мощный фреймворк глубокого обучения, разработанный Microsoft. Он предоставляет разработчикам и исследователям инструменты для создания, обучения и применения разнообразных нейронных сетей.

Основное преимущество CNTK заключается в его высокой производительности и эффективности в вычислениях глубокого обучения. Фреймворк оптимизирован для использования как на CPU, так и на GPU, а также поддерживает распределенные вычисления, что позволяет значительно ускорить процесс обучения моделей на больших объемах данных. Он предоставляет API на различных языках программирования, что обеспечивает гибкость и удобство в работе с ним. CNTK также известен своей способностью обрабатывать последовательности данных, что делает его удобным инструментом для задач обработки текста, речи, анализа последовательностей и других задач, связанных с последовательными данными.

2. TensorFlow – это один из ведущих и наиболее популярных фреймворков глубокого обучения, разработанный компанией Google. Этот мощный инструмент предоставляет разработчикам и исследователям возможность создавать, обучать и развертывать разнообразные нейронные сети. Одним из ключевых преимуществ TensorFlow является его поддержка для различных устройств, включая CPU, GPU и TPU. Это обеспечивает высокую производительность и эффективность при работе с различными моделями глубокого обучения.

Фреймворк TensorFlow обладает обширным набором инструментов и библиотек, что делает его универсальным инструментом для разработки различных видов и сложности нейронных сетей. Он предлагает как высокоуровневые API для удобства использования начинающими пользователями, так и низкоуровневые API для опытных разработчиков, позволяя более гибко управлять и оптимизировать модели.

TensorFlow также активно используется в исследованиях и промышленности благодаря своей широкой экосистеме и большому сообществу разработчиков. Этот фреймворк является популярным выбором для создания и обучения различных типов нейронных сетей, от сверточных и рекуррентных нейронных сетей до моделей глубокого обучения для обработки изображений, текста, звука и других типов данных. Скорость обучения на центральном процессоре можно повысить при помощи установки пакета Python, созданного TinyMind.

3. PyTorch – это фреймворк глубокого обучения, разработанный компанией Facebook, который завоевал популярность благодаря своей гибкости, простоте использования и динамическому подходу к построению вычислительных графов. Одной из ключевых особенностей PyTorch является динамический вычислительный граф, который позволяет пользователям оптимизировать и менять структуру моделей непосредственно во время выполнения, что упрощает отладку и экспериментирование.

PyTorch предоставляет интуитивный и простой интерфейс для разработки нейронных сетей, позволяя использовать язык Python для описания моделей и обработки данных. Он также предоставляет богатую библиотеку функций для глубокого обучения, что делает его подходящим для различных

задач, начиная от обработки изображений и текста до решения задач генерации контента или работы с последовательными данными.

Благодаря активному сообществу разработчиков, PyTorch постоянно развивается и предлагает последние инновации в области искусственного интеллекта и глубокого обучения. Он также часто используется в научных исследованиях благодаря своей гибкости и возможности реализовывать новейшие методики глубокого обучения с относительной легкостью.

4. Neon – был разработан компанией Nervana Systems (позднее приобретенной Intel). Этот фреймворк был ориентирован на оптимизацию производительности и использование для работы с высокоскоростными вычислениями в области нейронных сетей.

Neon изначально был известен своей специализацией на использование аппаратного обеспечения, такого как Intel Xeon и процессоры Nervana NNP (Nervana Neural Processor), чтобы улучшить производительность обучения нейронных сетей. Фреймворк предлагает удобный интерфейс программирования и призван облегчить разработку и эксперименты с глубоким обучением.

Скорость работы фреймворка на центральном процессоре перед можно оптимизировать путем установки определенного количества физических ядер локальной машины для OMP_NUM_THREADS, а также установкой значений compact: 1, 0 – для KMP_AFFINITY.

Однако к настоящему времени активная разработка и поддержка Neon существенно снизились, и у пользователей возникают определенные ограничения при использовании этого фреймворка. Вместо Neon многие разработчики предпочитают использовать другие популярные фреймворки, такие как TensorFlow, PyTorch или другие, которые широко поддерживаются и имеют более активные сообщества и развитие.

5. MXNet – разработан Apache Software Foundation, представляет собой гибкий фреймворк для глубокого обучения, который предоставляет разработчикам возможности для создания, обучения и развертывания нейронных сетей. Он обладает способностью работать на различных устройствах, включая CPU, GPU и FPGA, что делает его универсальным инструментом для разработки моделей глубокого обучения. Скорость работы фреймворка на центральном процессоре можно повысить путем установки флагов USE_MKL_EXPERIMENTAL=1 и USE_MKL=1, которые повышают скорость его работы на процессоре до 114.4 секунд на эпоху (в сравнении с первоначальной скоростью 219.9 секунд).

MXNet известен своим простым и интуитивным интерфейсом на языке Python, что делает его привлекательным для новичков в области глубокого обучения. Однако он также предоставляет API на других языках, таких как Scala, Julia и R, что делает его более доступным для широкого круга разработчиков с разным опытом.

Фреймворк поддерживает распределенное обучение нейронных сетей, что является важным аспектом при работе с большими объемами данных.

MXNet предлагает богатую библиотеку функций для глубокого обучения, включая возможности работы с последовательными данными, изображениями, текстом и другими типами данных.

MXNet также широко используется в индустрии и академических исследованиях благодаря своей гибкости, производительности и масштабируемости. Он остается одним из важных фреймворков глубокого обучения, предоставляющим широкие возможности для разработчиков и исследователей в области искусственного интеллекта.

На рисунках 3.5-3.6 представлены сравнительные диаграммы вышеописанных фреймворков по времени обучения трех свёрточных нейросетевых моделей на CPU и GPU соответственно в случае размера входных данных 48 × 48 [14].

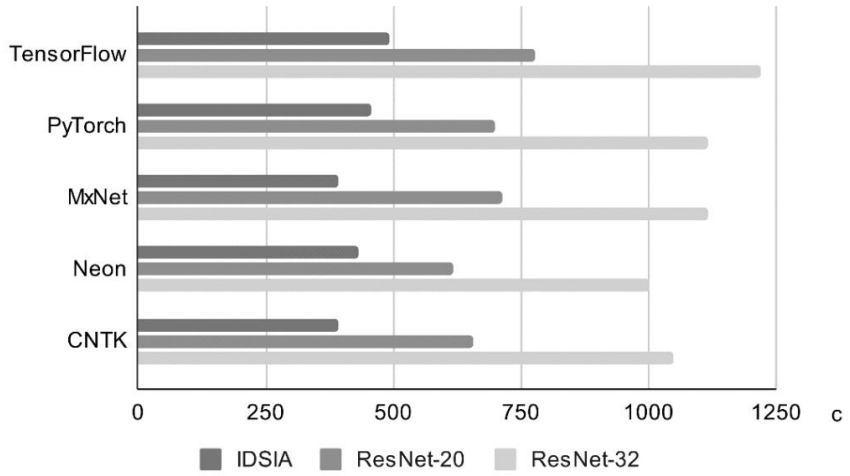

Рисунок 3.5 – Время обучения трех нейросетевых моделей на CPU

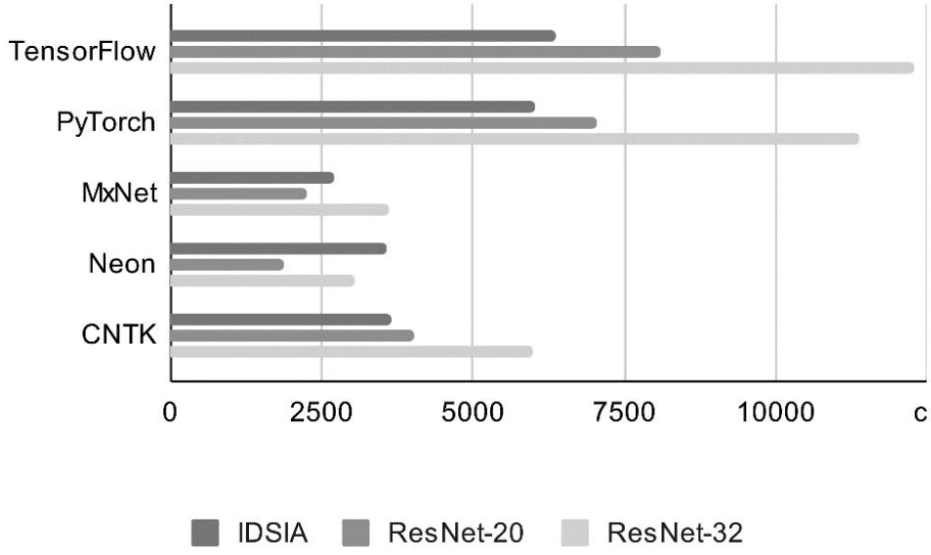

Рисунок 3.6 – Время обучения трех нейросетевых моделей на GPU

ЗАКЛЮЧЕНИЕ

В данном реферате были рассмотрены основы свёрточных нейронных сетей, их структура, роль ускорителей в вычислениях, а также подробно изучена аппаратная и программная архитектура ускорителей для обработки данных свёрточных нейронных сетей.

Изучение аппаратной архитектуры, начиная от общей структуры до конкретных типов процессоров и ускорителей, позволяет понять многообразие технологий, которые используются для оптимизации вычислений нейронных сетей.

Программная архитектура, включая технологии CUDA, OpenCL, фреймворки и библиотеки, играет ключевую роль в управлении и оптимизации процессов обучения и выполнения свёрточных нейронных сетей, а также в повышении эффективности и скорости работы алгоритмов глубокого обучения.

Это исследование позволяет лучше понять значимость и важность ускорителей в области нейронных сетей, расширяя возможности для создания более эффективных и быстрых моделей машинного обучения.

В целом, данная работа помогает осознать, насколько существенными являются разработка и оптимизация ускорителей для работы с нейронными сетями, их влияние на развитие технологий глубокого обучения и перспективы дальнейших исследований в данной области, а также позволяет прийти к выводу о необходимости постоянного совершенствования аппаратной и программной базы для развития и оптимизации свёрточных нейронных сетей в сфере искусственного интеллекта.

Работа над аппаратной и программной архитектурой ускорителей для свёрточных нейронных сетей остаётся важным направлением исследований, и её результаты имеют большое значение для дальнейшего прогресса в области машинного обучения и искусственного интеллекта в целом.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1]. Сверточные нейросети: что это и для чего они нужны? [Электронный ресурс]. – Режим доступа: https://forklog.com/cryptorium/ai/svertochnye-nejroseti-chto-eto-i-dlya-chego-oni-nuzhny – Дата доступа: 16.10.2023.

[2]. Сикорский, О.С. Обзор свёрточных нейронных сетей для задачи классификации изображений / О. С. Сикорский // Новые информационные технологии в автоматизированных системах – Москва, 2017. – № 20. – С. 37–42.

[3]. АРХИТЕКТУРА СВЁРТОЧНЫХ НЕЙРОННЫХ СЕТЕЙ [Электронный ресурс]. – Режим доступа: https://cyberleninka.ru/article/n/arhitektura-svyortochnyh-neyronnyh-setey – Дата доступа: 20.10.2023.

[4]. Таганов А.И. Нейросетевые системы искусственного интеллекта в задачах обработки изображений – М.: Телеком, 2016. – 148 с.

[5]. Хайкин С. Нейронные сети. Полный курс – М.: Вильямс, 2016. – 973 с.

[6]. Нейронный процессор [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Нейронный_процессор – Дата доступа: 22.10.2023.

[7]. Классификация ImageNet с помощью глубоких сверточных нейронных сетей [Электронный ресурс] – Режим доступа: https://papers.nips.cc/paper_files/paper/2012-Abstract.html – Дата доступа: 29.10.2023.

[8]. Тензорные ядра NVIDIA [Электронный ресурс] – Режим доступа: https://www.nvidia.com/en-us/data-center/tensorcores – Дата доступа: 03.11.2023.

[9]. Тензорные вычислительные модули, используемые для обучения [Электронный ресурс] – Режим доступа: https://cloud.google.com/tpu/docs/system-architecture-tpu-vm – Дата доступа: 03.11.2023.

[10]. Развитие аппаратно-ориентированных нейронных сетей на fpga и ASIC [Электронный ресурс] – Режим доступа: https://cyberleninka.ru/article/n/razvitie-apparatno-orientirovannyh-neyronnyh-setey-na-fpga-i-asic – Дата доступа: 05.11.2023.

[11]. Harris M. Mapping Computational Concepts to GPUs // GPU Gems 2. – 2006. – № 2. – P. 493–508.

[12]. Технология параллельного программирования CUDA [Электронный ресурс] – Режим доступа: http://mechanoid.su/parallel-cuda.html – Дата доступа: 07.11.2023.

[13]. OpenCL [Электронный ресурс] – Режим доступа: http://opencl.ru/design – Дата доступа: 08.11.2023.

[14]. ТЕСТИРОВАНИЕ И АНАЛИЗ ФРЕЙМВОРКОВ, ПРЕДНАЗНАЧЕННЫХ ДЛЯ ГЛУБОКОГО ОБУЧЕНИЯ НЕЙРОСЕТЕВЫХ МОДЕЛЕЙ [Электронный ресурс] – Режим доступа: https://cyberleninka.ru/article/n/testirovanie-i-analiz-freymvorkov-prednaznachennyh-dlya-glubokogo-obucheniya-neyrosetevyh-modeley – Дата доступа: 10.11.2023.