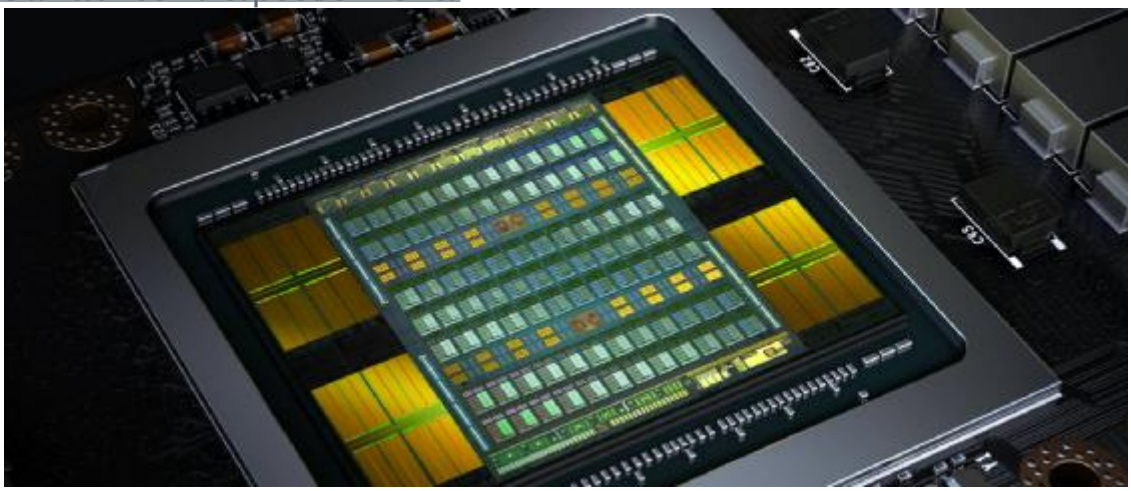
Что такое тензорные ядра: вычисления со смешанной точностью

https://habr.com/ru/post/512816/



В течение последних трёх лет Nvidia создавала графические чипы, в которых помимо обычных ядер, используемых для шейдеров, устанавливались дополнительные. Эти ядра, называемые тензорными, уже есть в тысячах настольных PC, ноутбуков, рабочих станций и дата-центров по всему миру. Но что же они делают и для чего применяются? Нужны ли они вообще в графических картах?

Сегодня мы объясним, что такое тензор, и как тензорные ядра используются в мире графики и глубокого обучения.

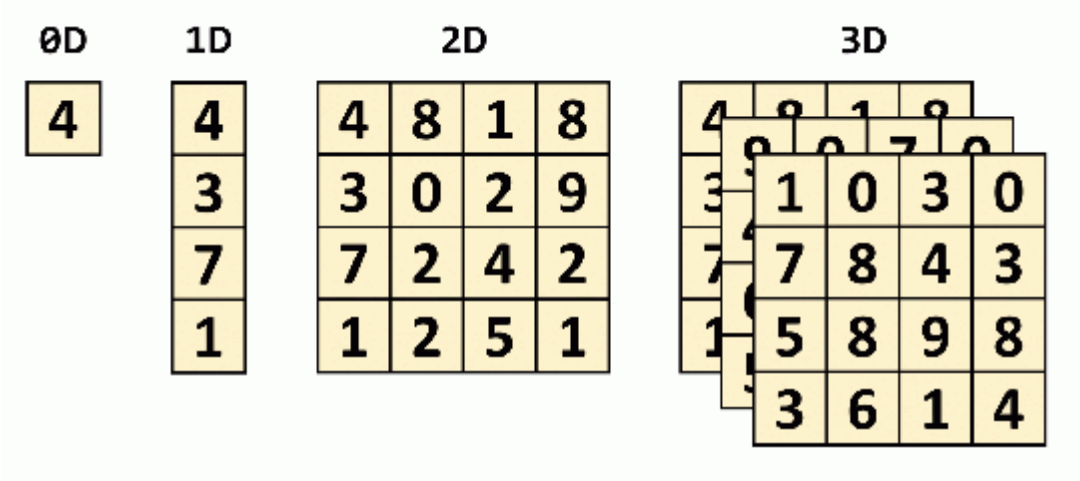
Краткий урок математики

Чтобы понять, чем же заняты тензорные ядра и для чего их можно использовать, нам сначала разобраться, что такое тензоры. Все микропроцессоры, какую бы задачу они ни выполняли, производят математические операции над числами (сложение, умножение и т.д.).

Иногда эти числа необходимо группировать, потому что они обладают определённым значением друг для друга. Например, когда чип обрабатывает данные для рендеринга графики, он может иметь дело с отдельными целочисленными значениями (допустим, +2 или +115) в качестве коэффициента масштабирования или с группой чисел с плавающей точкой (+0.1, -0.5, +0.6) в качестве координат точки в 3D-пространстве. Во втором случае для позиции точки требуются все три элемента данных.

Тензор — это математический объект, описывающий соотношения между другими математическими объектами, связанными друг с другом.

Обычно они отображаются в виде *массива* чисел, размерность которого показана ниже.



Простейший тип тензора имеет нулевую размерность и состоит из единственного значения; иначе он называется *скалярной* величиной. При увеличении количества размерностей мы сталкиваемся с другими распространёнными математическими структурами:

- 1 измерение = вектор
- 2 измерения = матрица

Строго говоря, скаляр — это тензор 0 x 0, вектор — 1 x 0, а матрица — 1 x 1, но ради простоты и привязки к тензорным ядрам графического процессора мы будем рассматривать тензоры только в виде матриц.

Одна из самых важных математических операций, выполняемых над матрицами — это умножение (или произведение). Давайте взглянем на то, как перемножаются друг на друга две матрицы, имеющие по четыре строки и столбца данных:

$$\mathbf{C} = \mathbf{A} \times \mathbf{B}$$

$$\mathbf{C} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \\ m & n & o & p \end{bmatrix} \times \begin{bmatrix} \alpha & \beta & \gamma & \delta \\ \varepsilon & \zeta & \eta & \theta \\ \kappa & \lambda & \mu & \nu \\ \pi & \tau & \psi & \omega \end{bmatrix}$$

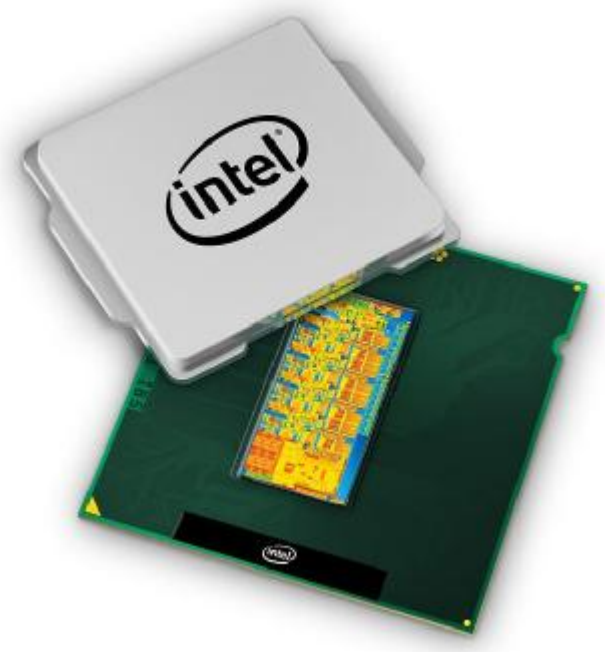
Окончательным результатом умножения всегда будет то же количество строк, что и в первой матрице, и то же количество столбцов, что и во второй. Как же перемножить эти два массива? Вот так:

$$\mathbf{C} = \begin{bmatrix} (a \times \alpha) - (b \times \varepsilon) + (c \times \kappa) + (d \times \pi) & (a\beta) + (b\zeta) + (c\lambda) + (d\tau) & (a\gamma) + (b\eta) + (c\mu) - (d\psi) & (a\delta) + (b\theta) + (c\nu) + (d\omega) \\ (e\alpha) - (f\varepsilon) + (g\kappa) - (h\pi) & (e\beta) + (f\zeta) + (g\lambda) + (h\tau) & (e\gamma) + (f\eta) + (g\mu) + (h\psi) & (e\delta) + (f\theta) + (g\nu) + (h\omega) \\ (i\alpha) + (j\varepsilon) + (k\kappa) + (l\pi) & (i\beta) - (j\zeta) + (k\lambda) + (l\tau) & (i\gamma) + (j\eta) - (k\mu) + (l\psi) & (i\delta) + (j\theta) + (k\nu) + (l\omega) \\ (m\alpha) + (n\varepsilon) + (o\kappa) + (p\pi) & (m\beta) + (n\zeta) + (o\lambda) + (p\tau) & (m\gamma) + (n\eta) + (o\mu) + (p\psi) & (m\delta) + (n\theta) + (o\nu) + (p\omega) \end{bmatrix}$$

На пальцах это посчитать не удастся

Как вы видите, вычисление «простого» произведения матриц состоит из целой кучи небольших умножений и сложений. Так как любой современный центральных процессор может выполнять обе эти операции, простейшие тензоры способен выполнять каждый настольный компьютер, ноутбук или планшет.

Однако показанный выше пример содержит 64 умножений и 48 сложений; каждое небольшое произведение даёт значение, которое нужно где то хранить, прежде чем его можно будет сложить с другими тремя небольшими произведениями, чтобы позже можно было сохранить окончательное значение тензора. Поэтому, несмотря на математическую простоту умножений матриц, они затратны *вычислительно* — необходимо использовать множество регистров, а кэш должен уметь справляться с кучей операций считывания и записи.



Архитектура Intel Sandy Bridge, в которой впервые появились

расширения AVX

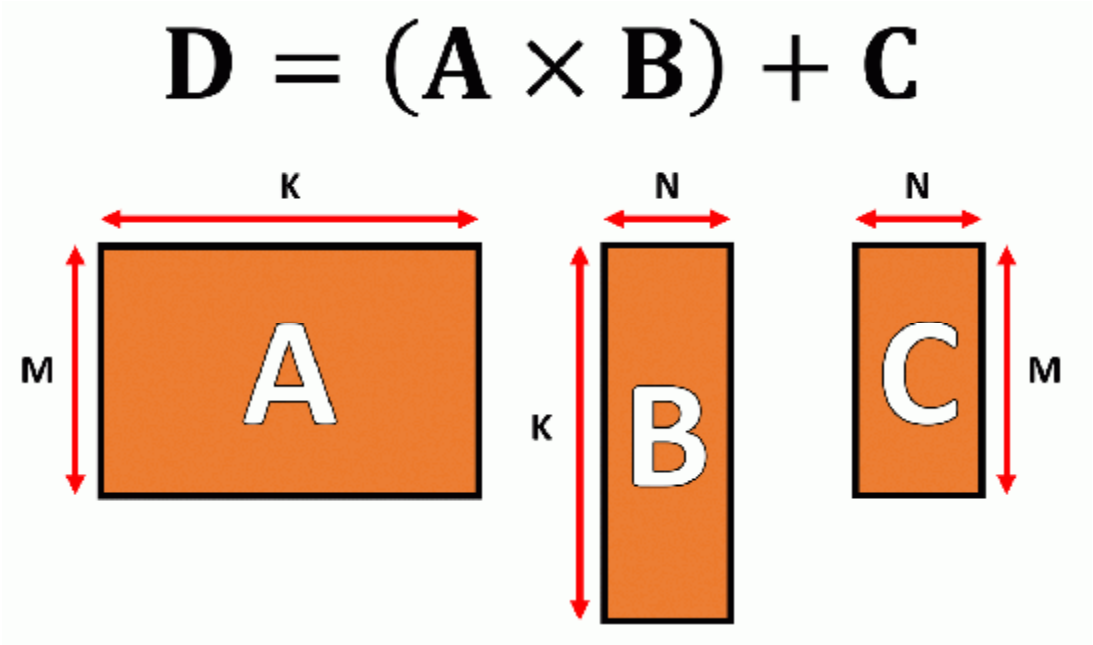
На протяжении многих лет в процессорах AMD и Intel появлялись различные расширения (MMX, SSE, а теперь и AVX — все они являются SIMD, *single instruction multiple data*), позволяющие процессору одновременно обрабатывать множество чисел с плавающей запятой; это как раз то, что требуется для перемножения матриц.

Но существует особый тип процессоров,
который *специально* спроектирован для обработки операций SIMD: графические процессоры (graphics processing unit, GPU).

Умнее, чем обычный калькулятор?

В мире графики одновременно необходимо передавать и обрабатывать огромные объёмы информации в виде векторов. Благодаря своей способности параллельной обработки GPU идеально подходят для обработки тензоров; все современные графические процессоры поддерживают функциональность под названием GEMM (*General Matrix Multiplication*).

Это «склеенная» операция, при которой перемножаются две матрицы, а результат затем накапливается с другой матрицей. Существуют важные ограничения на формат матриц и все они связаны с количеством строк и столбцов каждой матрицы.



Требования GEMM к строкам и столбцам: матрица A(m x k), матрица B(k x n), матрица C(m x n)

Алгоритмы, используемые для выполнения операций с матрицами, обычно лучше всего работают, когда матрицы квадратные (например, массив 10 x 10 будет работать лучше, чем 50 x 2) и довольно небольшие по размеру. Но они всё равно будут работать лучше, если обрабатываются на оборудовании, которое предназначено исключительно для таких операций.

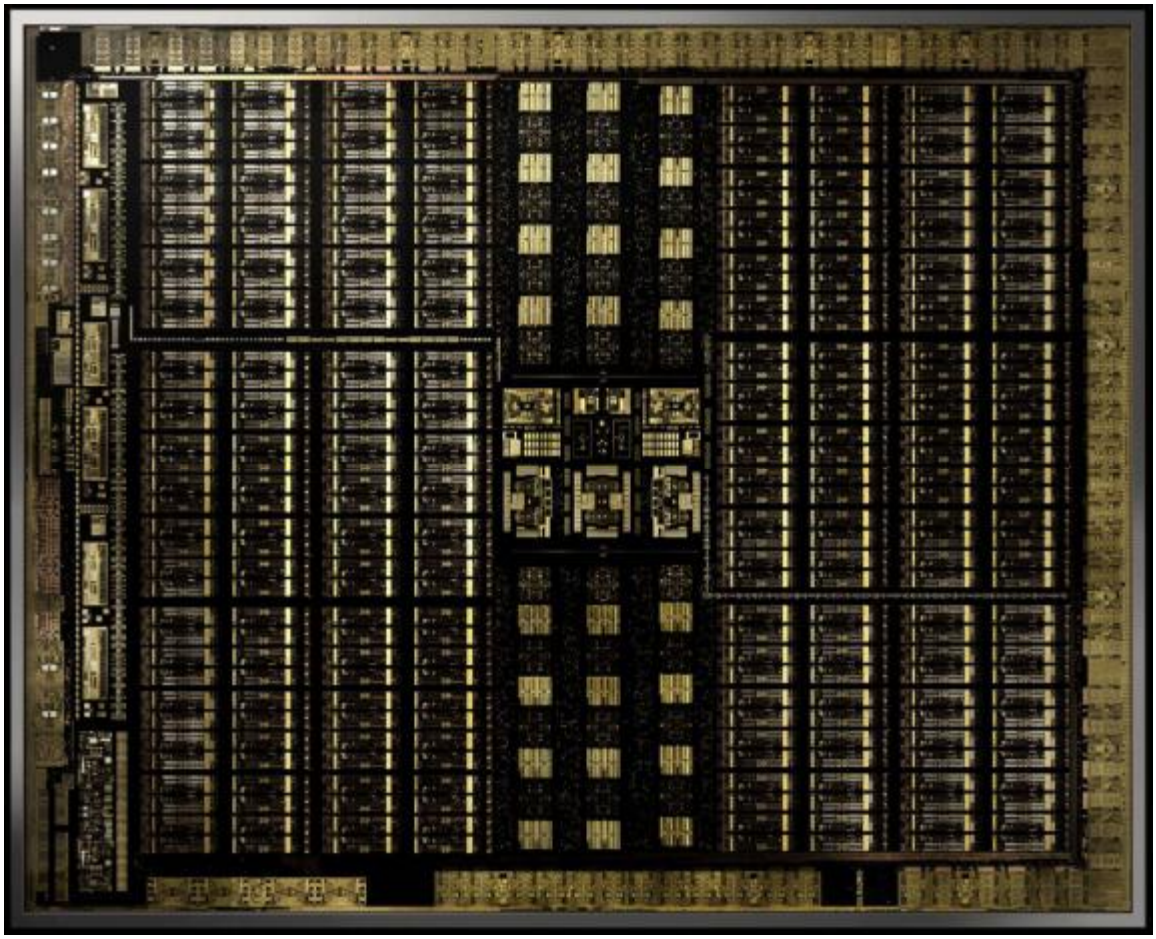
В декабре 2017 года Nvidia выпустила графическую карту с GPU, имеющим новую архитектуру *Volta*. Она была нацелена на профессиональные рынки, поэтому этот чип не использовался в моделях GeForce. Уникальным он был потому, что стал первым графическим процессором, имеющим ядра только для выполнения тензорных вычислений.

Графическая карта Nvidia Titan V, на которой установлен чип GV100 Volta. Да на ней можно запустить Crysis

Тензорные ядра Nvidia были предназначены для выполнения по 64 GEMM за тактовый цикл с матрицами 4 x 4, содержащими значения FP16 (числа с плавающей запятой размером 16 бит) или умножение FP16 со сложением FP32. Такие тензоры очень малы по размеру, поэтому при обработке настоящих множеств данных ядра обрабатывают небольшие части больших матриц, выстраивая окончательный ответ.

Менее года спустя Nvidia выпустила архитектуру Turing. На этот раз тензорные ядра были установлены и в модели

GeForce потребительского уровня. Система была улучшена для поддержки других форматов данных, например, INT8 (8-битное целочисленное значение), но во всём остальном они работали так же, как в Volta.



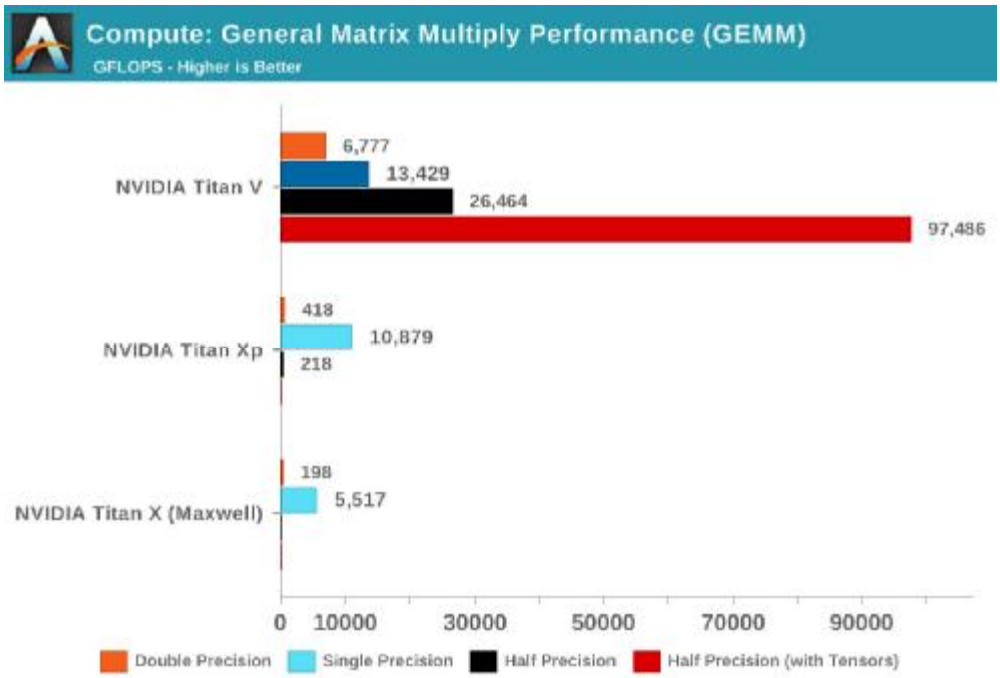
В начале этого года архитектура Ampere дебютировала в графическом процессоре дата-центра A100, и на этот раз Nvidia повысила производительность (256 GEMM за цикл вместо 64), добавила новые форматы данных и возможность очень быстрой обработки *разреженных тензоров (sparse tensor)* (матриц со множеством нулей).

Программисты могут получить доступ к тензорным ядрам чипов Volta, Turing и Ampere очень просто: код всего лишь должен использовать флаг, сообщающий API и драйверам, что нужно применять тензорные ядра, тип данных должен поддерживаться ядрами, а размерности матриц должны быть кратными 8. При выполнении всех этих условий всем остальным займётся оборудование.

Всё это здорово, но насколько тензорные ядра лучше в обработке GEMM, чем обычные ядра GPU?

Когда появилась Volta, сайт Anandtech провёл математические тесты трёх карт Nvidia: новой Volta, самой мощной из линейки Pascal и старой

карты Maxwell.



Понятие *точности (precision)* относится к количеству бит, использованных для чисел с плавающей запятой в матрицах: двойная (double) обозначает 64, одиночная (single) — 32, и так далее. По горизонтальной оси отложено максимальное количество операций с плавающей запятой, выполняемое за секунду, или сокращённо FLOPs (помните, что одна GEMM — это 3 FLOP).

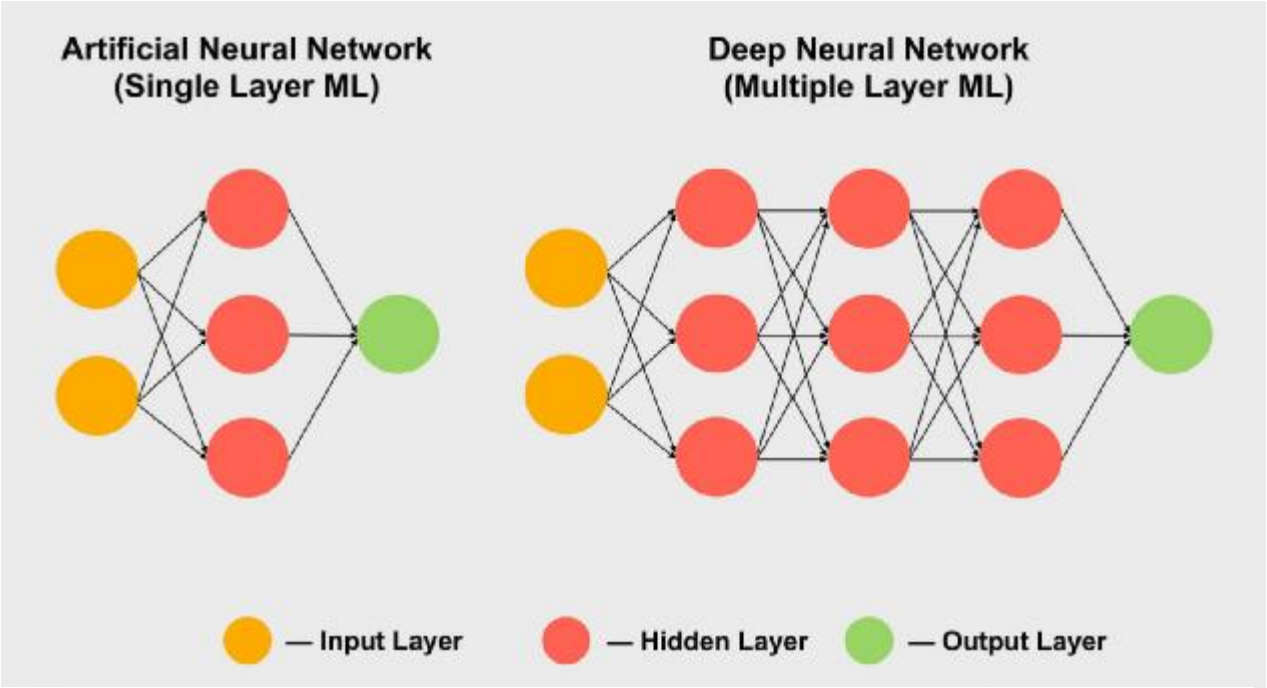
Просто взгляните на результаты при использовании тензорных ядер вместо так называемых ядер CUDA! Очевидно, что они потрясающе справляются с подобной работой, но что же мы можем делать при помощи тензорных ядер?

Математика, делающая всё лучше

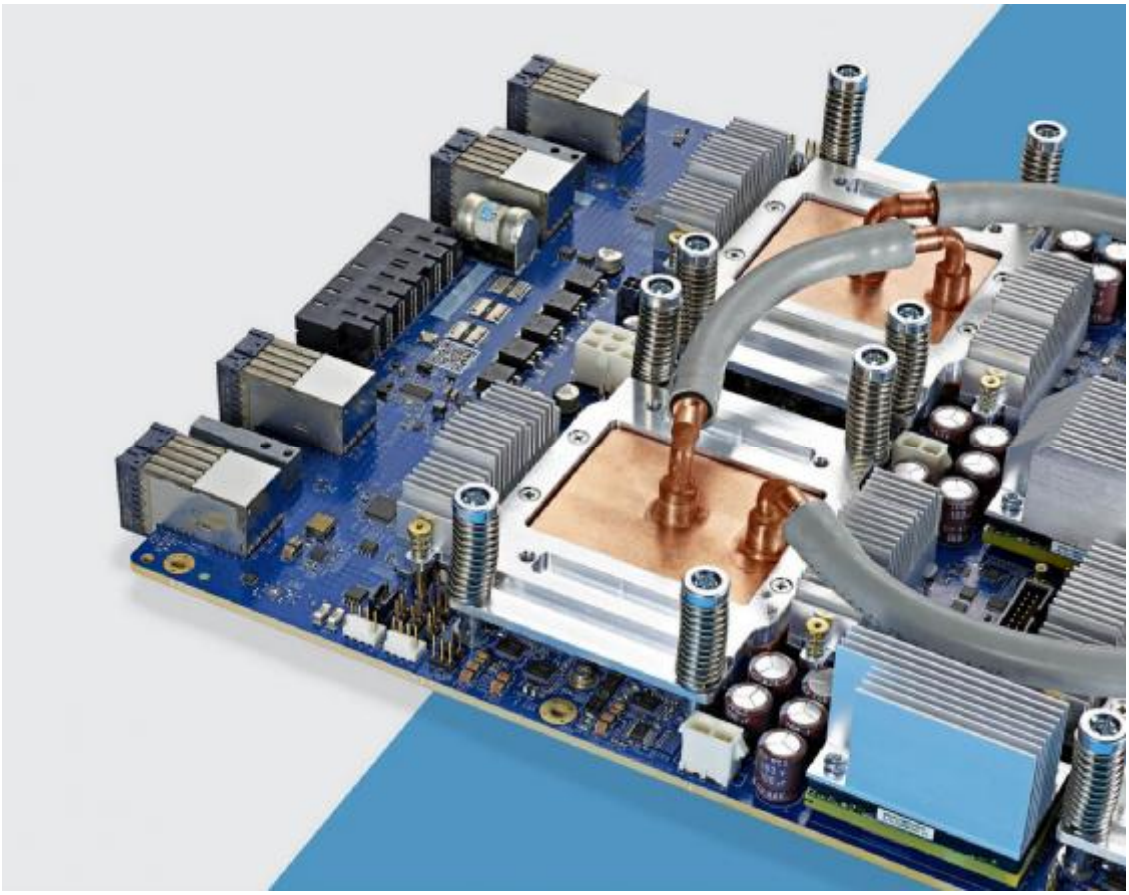
Тензорные вычисления чрезвычайно полезны в физике и проектировании, они используются для решения всевозможных сложных задач в механике жидкостей, электромагнетизме и астрофизике, однако компьютеры, которые использовались для обработки подобных чисел, обычно выполняли операции с матрицами в больших кластерах из центральных процессоров.

Ещё одна область, в которой любят применять тензоры — это машинное обучение, особенно её подраздел «глубокое обучение». Его смысл сводится к обработке огромных наборов данных в гигантских массивах, называемых нейронными сетями. Соединениям между различными значениями данных задаётся определённый вес — число, выражающее

важность конкретного соединения.



Поэтому когда нам нужно разобраться, как взаимодействуют все эти сотни, если не тысячи соединений, нужно умножить каждый элемент данных в сети на все возможные веса соединений. Другими словами, перемножить две матрицы, а это классическая тензорная математика!



Чипы Google TPU 3.0, закрытые системой водяного охлаждения

Именно поэтому во всех суперкомпьютерах глубокого обучения используются GPU, и почти всегда это Nvidia. Однако некоторые компании даже разработали собственные процессоры из тензорных ядер. Google, например, в 2016 году объявила о разработке своего первого TPU (*tensor processing unit*), но эти чипы настолько специализированные, что не могут выполнять ничего, кроме операций с матрицами.

Тензорные ядра в потребительских GPU (GeForce RTX)

Но что если я куплю графическую карту Nvidia GeForce RTX, не являясь ни астрофизиком, решающим задачи римановых многообразий, ни специалистом, экспериментирующим с глубинами свёрточных нейронных сетей...? Как я могу использовать тензорные ядра?

Чаще всего они не применяются для обычного рендеринга, кодирования или декодирования видео, поэтому может показаться, что вы потратили деньги на бесполезную функцию. Однако Nvidia встроила тензорные ядра в свои потребительские продукты в 2018 году (Turing GeForce RTX), внедрив при этом DLSS — *Deep Learning Super Sampling*.

Принцип прост: рендерим кадр в довольно низком разрешении, а после завершения повышаем разрешение конечного результата так, чтобы он совпадал с «родными» размерами экрана монитора (например, рендерим в 1080p, а затем изменяем размер до 1400p). Благодаря этому повышается производительность, ведь обрабатывается меньшее количество пикселей, а на экране всё равно получается красивое изображение.

Консоли имели такую функцию уже многие годы, и многие современные

игры для PC тоже обеспечивают эту возможность. В Assassin's Creed: Odyssey компании Ubisoft можно уменьшить разрешение рендеринга до всего 50% от разрешения монитора. К сожалению, результаты выглядят не так красиво. Вот как игра выглядит в 4K с максимальными настройками графики:

В высоких разрешениях текстуры выглядят красивее, потому что сохраняют в себе больше деталей. Однако для вывода этих пикселей на экран требуется много обработки. Теперь взгляните на то, что происходит при установке рендеринга на 1080p (25% от предыдущего количества пикселей), с использованием шейдеров в конце для растягивания картинки до 4K.

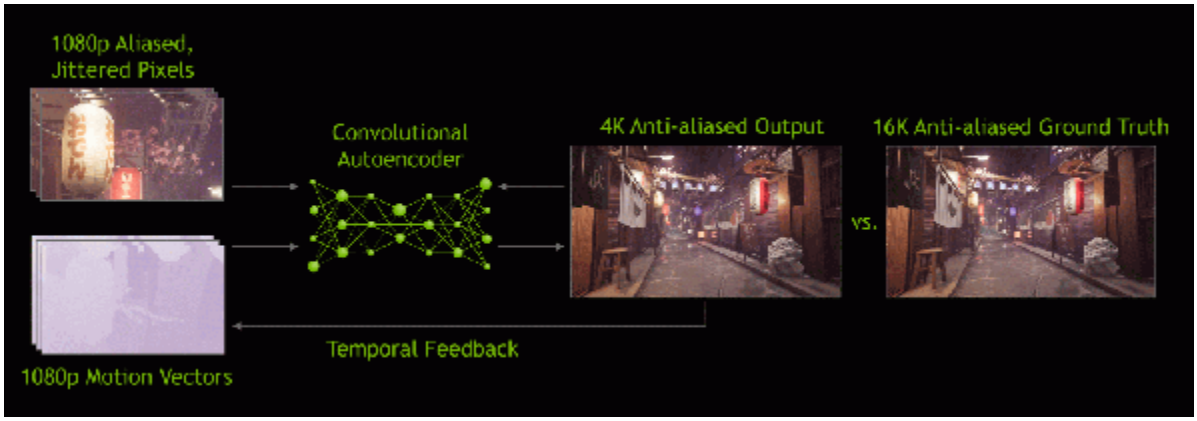
Из-за сжатия jpeg разница может быть заметной не сразу, но видно, что броня персонажа и скала вдали выглядят размытыми. Давайте приблизим часть изображения для более детального изучения:

Изображение слева отрендерено в 4K; изображение справа — это 1080p, растянутые до 4K. Разница гораздо заметнее в движении, потому что смягчение всех деталей быстро превращается в размытую кашу. Частично чёткость можно восстановить благодаря эффекту резкости драйверов графической карты, но лучше бы нам вообще не приходилось этим не заниматься.

Именно здесь в ход идёт DLSS — в первой версии этой технологии Nvidia анализировались несколько выбранных игр; они запускались в высоких разрешениях, низких разрешениях, со сглаживанием и без него. Во всех этих режимах был сгенерирован набор изображений, загруженный затем в суперкомпьютеры компании, которые использовали нейронную сеть, чтобы определить, каким образом лучше всего превратить изображение в разрешении 1080p в идеальную картинку в

более высоком разрешении.

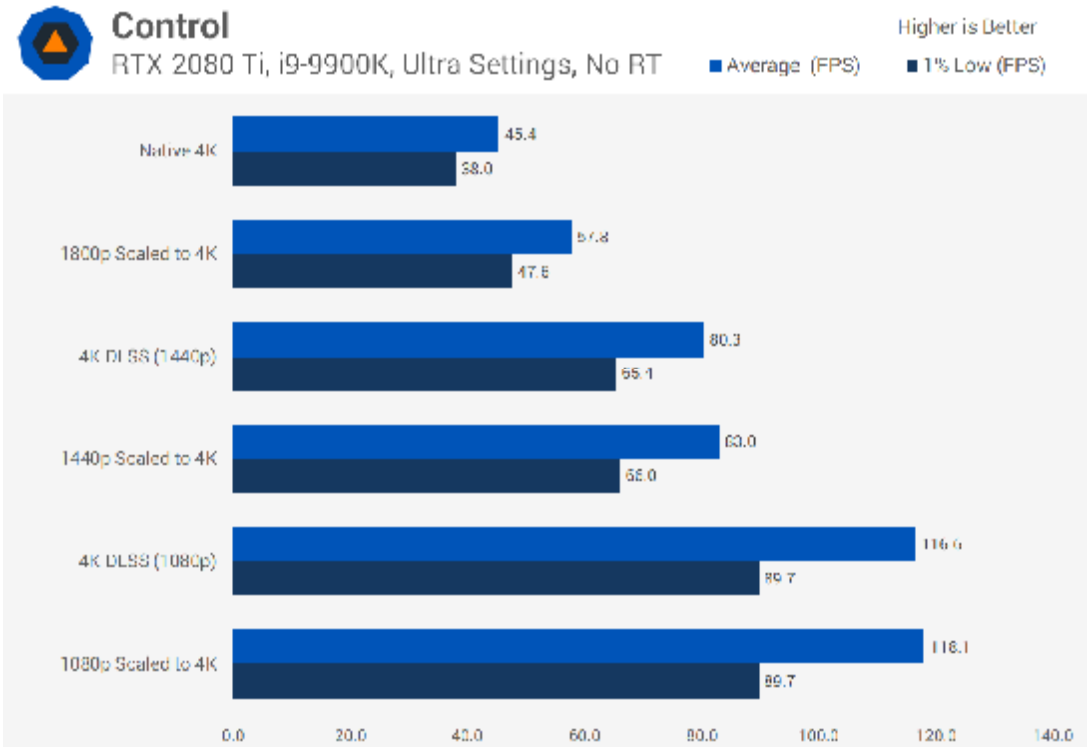
Нужно сказать, что DLSS 1.0 не был идеальным: детали часто терялись и в некоторых местах возникало странное мерцание. К тому же он не использовал сами тензорные ядра графической карты (он выполнялся в сети Nvidia) и каждой игре с поддержкой DLSS для генерации алгоритма повышения масштаба требовалось отдельное исследование компанией Nvidia.



Когда в начале 2020 года вышла версия 2.0, в неё были внесены серьёзные улучшения. Самым важным стало то, что суперкомпьютеры Nvidia теперь использовались только для создания общего алгоритма увеличения масштаба — в новой версии DLSS для обработки пикселей с помощью нейронной модели (тензорными ядрами GPU) используются данные из отрендеренного кадра.

Нас впечатляют возможности DLSS 2.0, но пока его поддерживает очень мало игр — на момент написания статьи их было всего 12. Всё больше разработчиков хочет реализовать его в своих будущих играх, и на то есть причины.

Благодаря любому увеличению масштаба можно добиться серьёзного роста производительности, поэтому можно быть уверенными, что DLSS продолжит эволюционировать.



Хотя визуальные результаты работы DLSS не всегда идеальны, освободив занятые рендерингом ресурсы, разработчики смогут добавить больше визуальных эффектов или обеспечить один уровень графики на более широком диапазоне платформ.

Например, DLSS часто рекламируют вместе с трассировкой лучей (ray tracing) в играх с «поддержкой RTX». Карты GeForce RTX содержат дополнительные вычислительные блоки, называемые RT-ядрами, это специализированные логические блоки для ускорения вычислений пересечения луча с треугольником и обхода иерархии ограничивающих объёмов (bounding volume hierarchy, BVH). Эти два процесса являются очень длительными процедурами, определяющими способ взаимодействия света с другими объектами сцены.

Как мы выяснили, ray tracing — очень трудоёмкий процесс, поэтому чтобы обеспечить в играх приемлемый уровень частоты кадров, разработчики должны ограничить количество лучей и выполняемых в сцене отражений. При выполнении этого процесса могут создаваться зернистые изображения, поэтому необходимо применять алгоритм устранения шумов, что повышает сложность обработки. Ожидается, что тензорные ядра повысят производительность этого процесса благодаря устранению шумов с использованием ИИ, однако это ещё предстоит реализовать: большинство современных приложений по-прежнему использует для этой задачи ядра CUDA. С другой стороны, благодаря тому, что DLSS 2.0 становится вполне практичной техникой повышения размера, тензорные ядра можно будет эффективно использовать для повышения частоты кадров после применения в сцене трассировки лучей.

Существуют и другие планы по использованию тензорных ядер карт GeForce RTX, например, улучшение анимаций персонажей или симуляция тканей. Но как и в случае с DLSS 1.0, пройдёт ещё немало времени, прежде чем появятся сотни игр, использующие специализированные матричные вычисления на GPU.

Многообещающее начало

Итак, ситуация такова — тензорные ядра, отличные аппаратные блоки, которые, однако, встречаются только в некоторых картах потребительского уровня. Изменится ли что-то в будущем? Так как Nvidia уже значительно улучшила производительность каждого тензорного ядра в своей архитектуре Ampere, есть большая вероятность того, что они будут устанавливаться и в модели нижнего и среднего ценового уровня.

Хотя таких ядер пока нет в GPU компаний AMD и Intel, возможно, в будущем мы их увидим. У AMD есть система повышения резкости или улучшения деталей в готовых кадрах ценой небольшого снижения производительности, поэтому компания, возможно, будет придерживаться этой системы, особенно учитывая то, что её не нужно интегрировать разработчикам, достаточно включить её в драйверах.

Существует также мнение, что пространство на кристаллах в графических чипах лучше было бы потратить на дополнительные шейдерные ядра — так поступила Nvidia при создании бюджетных версий своих чипов Turing. В таких продуктах, как GeForce GTX 1650, компания полностью отказалась от тензорных ядер и заменила их дополнительными FP16-шейдерами.

Но пока, если вы хотите обеспечить сверхбыструю обработку GEMM и воспользоваться всеми её преимуществами, то у вас есть два варианта: купить кучу огромных многоядерных CPU или просто один GPU с тензорными ядрами.