ТЕХНОЛОГИЯ РЕАЛИЗАЦИИ НЕЙРОСЕТЕВОГО АЛГОРИТМА В СРЕДЕ CUDA НА ПРИМЕРЕ РАСПОЗНАВАНИЯ РУКОПИСНЫХ ЦИФР

Изотов П.Ю. [1,2], *Суханов С.В.* [1], *Головашкин Д.Л.* [2]
[1] *Самарский государственный аэрокосмический университет им. С.П. Королева,*
[2] *Институт систем обработки изображений РАН*

Аннотация

На примере сверточной нейронной сети продемонстрированы особенности реализации нейросетевого алгоритма распознавания образов на видеокарте (GPU) в среде NVIDIA CUDA. Длительность обучения нейронной сети на видеоадаптере уменьшена в 5,96, а распознавания набора тестовых образцов – в 8,76 раза по сравнению с оптимизированным алгоритмом, выполняющим вычисления только на центральном процессоре (CPU). Показана перспективность реализации нейросетевых алгоритмов на графических процессорах.

Ключевые слова: сверточная нейронная сеть, распознавание образов, обучение нейронных сетей, метод обратного распространения ошибки, параллельные вычисления, GPGPU, NVIDIA, CUDA, умножение матриц, CUBLAS.

Введение

Нейросетевые алгоритмы находят широкое применение в вычислительной практике. В частности, особое внимание уделяется классификации и кластеризации изображений, распознаванию речи и изображений, прогнозу финансовых показателей [1], искусственному синтезу речи, аппроксимации функционалов, совершенствованию методов электроимпедансной и магнитоиндукционной томографии, анализу данных (data mining).

Известным ограничением развития нейросетевых алгоритмов следует признать высокие вычислительные затраты на реализацию таких методов [2]. К традиционным способам решения данной проблемы относят организацию параллельных и распределенных вычислений на специализированном аппаратном обеспечении, таком как нейронные чипы, систолические нейропроцессоры, ПЛИС, распределенные кластерные системы [3], GRID-технологии.

Появившийся недавно аппаратно-программный комплекс CUDA (Compute Unified Device Architecture) позволяет использовать процессоры видеокарт (GPU) как ускорители научных и инженерных расчетов и проводить вычисления, по эффективности сравнимые с современными кластерными системами [4], [5]. Принципиальное отличие архитектур состоит в следующем: исполнение команд на кластере происходит в стиле MIMD (много потоков команд, много потоков данных – когда набор процессоров независимо выполняет различные наборы команд, обрабатывающих различные наборы данных), а в среде CUDA характеризуется стилем SIMD (один поток команд, много потоков данных – когда несколько процессоров исполняют одну и ту же команду над разными данными, обычно элементами массива).

Особенностью оборудования, поддерживающего технологию CUDA, является возможность обеспечивать на порядок большую (по сравнению с кластерами) пропускную способность при работе с памятью. В частности, при вычислении кулоновской ионизации сверхбольших структур, таких как вирусы, может потребоваться несколько дней работы кластера средних размеров, в то время как решение той же задачи на персональном компьютере, оснащенном видеокартой с поддержкой CUDA, можно провести менее чем за час [6].

Еще одной немаловажной характеристикой решений на CUDA является стоимость. Так, персональный суперкомпьютер NVIDIA Tesla C2050 имеет производительность 520GFLOPS (миллиардов операций плавающей арифметики в секунду) и стоит в 10 раз меньше традиционного кластера той же производительности, построенного на основе только центральных процессоров (CPU) [7], и, что более важно, имеет в 20 раз большую эффективность вычислений на ватт мощности.

Целью настоящей работы было исследование возможностей и особенностей среды CUDA по реализации эффективного нейросетевого алгоритма распознавания рукописных цифр.

Для распознавания образов обычно используются следующие виды нейронных сетей: многослойный персептрон; сверточная нейронная сеть, нейронная сеть высокого порядка, нейронная сеть Хопфилда, самоорганизующиеся карты Кохонена, когнитрон и неокогнитрон Фукушимы.

В работе [8] проведен анализ решения задачи распознавания рукописных символов с использованием многослойных персептронов, машин опорных векторов (SVM) и сверточных нейронных сетей. Показано, что последние дают ошибку распознавания образцов из проверочной выборки в 0,4%, в то время как другие перечисленные методы – от 0,6% до 1,6%. В силу этого авторы сочли целесообразным обратить внимание на алгоритм функционирования и обучения сверточной нейронной сети.

В данной работе рассмотрен алгоритм, основанный на идеях публикации [9]. Реализация, представленная в ней, связана с технологиями (объектно-ориентированным программированием и использованием стандартной библиотеки шаблонов STL), которые увеличивают длительность вычислений; этот факт отмечен в [10]. Поэтому авторы настоящей

работы сочли необходимым переписать алгоритм без использования указанных технологий, провести его векторизацию и реализовать в среде CUDA.

Особенности организации вычислений в среде CUDA

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

Логически GPU с поддержкой CUDA можно рассматривать как набор многоядерных процессоров. Основными вычислительными блоками таких видеочипов являются мультипроцессоры, которые, как показано на рис. 1, состоят из восьми ядер, нескольких тысяч 32-битных регистров, 16 кБайт общей памяти, текстурного и константного кэшей.

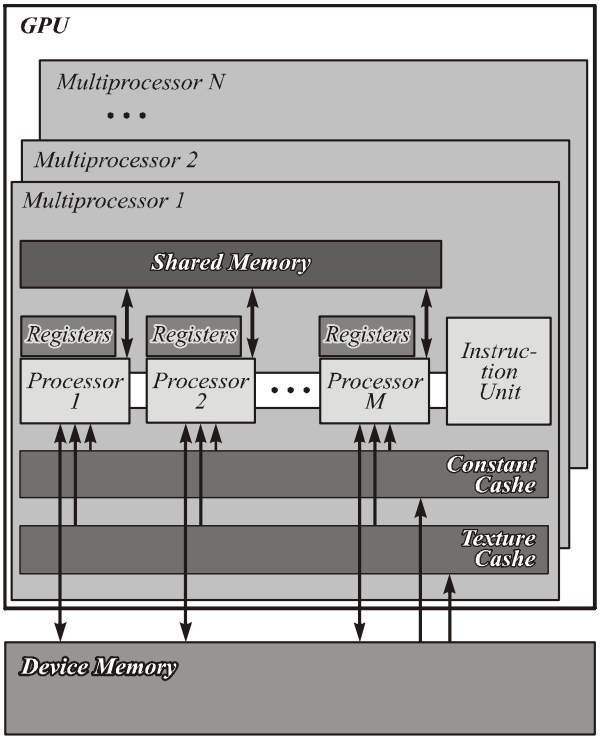

Рис. 1. Набор SIMD-мультипроцессоров и виды памяти видеокарты

Для взаимодействия CPU и GPU используется глобальная память, размеры которой могут варьироваться от нескольких сотен МБайт до нескольких ГБайт. Скорость передачи данных с CPU на GPU ограничивается пропускной способностью шины PCI Express.

Ключевой абстракцией CUDA с программной точки зрения является иерархия сгруппированных вычислительных потоков, параллельно выполняющих один и тот же набор инструкций, называемый ядром (kernel). Потоки, объединенные в так называемый вычислительный блок, исполняются на одном мультипроцессоре, могут взаимодействовать посредством общей памяти и проводить барьерную синхронизацию.

Сверточные нейронные сети в задаче распознавания рукописных символов

Сверточная нейронная сеть была впервые упомянута в работе Фукушимы в 1980 г. и называлась неокогнитроном. В 1989 г. Ян ЛеКун [11] и его сотрудники разработали несколько сетей этого типа, объединенных под общим названием LeNet. Важным отличием этих сетей от неокогнитрона было использование метода обучения с учителем.

Сверточные нейронные сети могут эффективно использоваться для распознавания образов, лиц и речи. Они объединяют три архитектурных идеи для обеспечения частичной устойчивости к изменению масштаба, повороту, сдвигу и пространственным искажениям:

- локальные рецепторные поля (позволяют учитывать двумерную топологию входных данных);
- общие веса (обеспечивают детектирование общих черт в любом месте изображения и уменьшают количество настраиваемых параметров);
- иерархическая организация с пространственными подвыборками (позволяет строить иерархии признаков).

Для обучения сверточных нейронных сетей может применяться как стандартный метод обратного распространения погрешности, так и его различные модификации [12].

В цели данной работы не входило исследование сверточных нейронных сетей общего вида, поэтому использовалась сеть со вполне определенными характеристиками, приведенными ниже.

В настоящем исследовании рассматривалась сеть, состоящая из входного, двух сверточных и двух полносвязных слоев. На входной слой поступало изображение размерностью 28×28 рукописной цифры в 256 градациях серого из базы MNIST – наборов изображений рукописных цифр, представленных в свободном доступе. Значения яркости отдельных пикселей изображения после нормализации в интервале от минус единицы до единицы и обработки активационной функцией поступали на вход первого сверточного слоя, представляющего собой двумерный массив нейронов размерностью 29×29. Такая размерность позволяет с минимальными издержками на размещение данных в памяти реализовывать структуру перекрывающихся рецепторных полей.

Выход начального слоя, который можно представить в виде матрицы вещественных чисел X^0, является входом для нейронов первого сверточного слоя, состоящего из шести карт нейронов, выходы которых размещаются в матрицах X^{1i}, $i=\overline{1,6}$ размерностью 13×13 и поступают на вход второго сверточного слоя. Последний, в свою очередь, состоит из 48 карт X^{2i}, $i=\overline{1,48}$ размерностью 5×5. Для выходов третьего и четвертого введем обозначения X^3 и X^4 соответственно. Общая структура сети представлена на рис. 2.

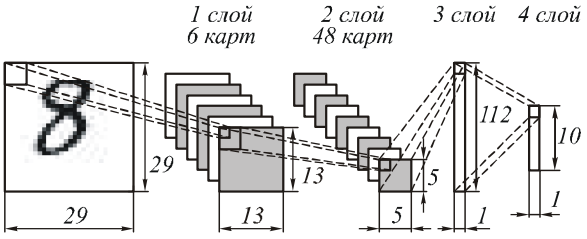

Рис. 2. Структура сверточной нейронной сети для распознавания рукописных символов, использованная в данной работе

Реализация последовательного алгоритма на CPU

В качестве базовой версии алгоритма функционирования и обучения сверточной нейронной сети была использована реализация, описанная в [9].

Алгоритм 1 – обучение сверточной нейронной сети

Шаг 1. Создание структуры нейронной сети:

Шаг 1.1. Создание массивов нейронов, по одному массиву на слой.

Шаг 1.2. Создание массивов весовых коэффициентов, по одному массиву на слой.

Шаг 1.3. Создание массивов соединений между нейронами соседних слоев. Соединение характеризуется тройкой чисел $(i; w_{ij}^n; j)$, что означает, что между j-м нейроном n-го слоя и i-м нейроном $(n-1)$-го слоя есть связь, вес которой равен w_{ij}^n.

Шаг 2. Распознавание образов нейронной сетью:

```
Для n от 1 до 4
  Для i от 0 до размер_n-го_слоя
  k := номер нейрона (n-1)-го слоя j-й связи
  f_net := 0
    Для j от 0 до количество_соединений_i-го_нейрона (*)
    f_net := f_net +
      (выход_k-го_нейрона_(n-1)-го_слоя) ×
      × (вес_j-й_связи).
  выход_i-го_нейрона_n-го_слоя := активац._функция(f_net).
```

Шаг 3. Обучение нейронной сети обратным распространением погрешности:

Шаг 3.1. Найти погрешность распознавания как разность между полученным выходом четвертого слоя и требуемым значением вектора, соответствующим распознаваемому образцу: $\frac{\partial E_n^P}{\partial x_i^n} = x_i^n - T_i$.

Шаг 3.2. Для n от 4 до 1:

Шаг 3.2.1. Найти $\frac{\partial E_n^P}{\partial y_i^n} = \frac{\partial E_n^P}{\partial x_i^n} \cdot G(x_i^n)$.

Шаг 3.2.2. Положить $\partial E_n^P / \partial w_{ij}^n$ равным нулю для всех i, j.

Шаг 3.2.3. Для каждого соединения, связывающего нейрон i текущего слоя с нейроном j предыдущего слоя, обновить его вес: $\frac{\partial E_n^P}{\partial w_{ij}^n} = x_j^{n-1} \cdot \frac{\partial E_n^P}{\partial y_i^n}$.

Шаг 3.2.4. Вычислить ошибку предыдущего слоя: $\frac{\partial E_n^P}{\partial x_k^{n-1}} = \sum_{i=1}^{C_n} w_{ik}^n \cdot \frac{\partial E_n^P}{\partial y_i^n}$.

Шаг 3.2.5. Произвести обновление весовых коэффициентов w_{ij}^n в соответствии с методом второго порядка [13].

Шаг 4. Ввод данных следующего образа и переход к шагу 2.

У использованного автором [9] подхода к реализации алгоритма работы сверточной нейронной сети есть следующие преимущества:

- В результате сокрытия структуры нейронной сети в данных уменьшается размер исходного кода. Структура сети описывается лишь однажды – при ее инициализации.
- Достигается унификация процедур, описывающих функционирование и обучение нейронной сети.

Основным недостатком данной реализации является низкая скорость работы. В [10] упоминается возможность повышения производительности на порядок при отказе от использования объектно-ориентированного подхода и стандартной библиотеки шаблонов, а также при использовании специальных приемов программирования. К сожалению, автор [10] не опубликовал свой вариант алгоритма. Ниже представлена оптимизация рассмотренного алгоритма, предложенная авторами настоящей работы.

Оптимизация последовательного алгоритма

Поскольку использование контейнеров стандартной библиотеки шаблонов, равно как и объектно-ориентированного кода, при написании процедур на CUDA невозможно, возникла необходимость написать эффективную реализацию сверточной нейронной сети на языке программирования Си, которую в дальнейшем проще модифицировать для работы в среде CUDA.

Оптимизация заключается в следующем. Действия, описанные в шаге 1.3 алгоритма 1, перемещаются внутрь цикла (*), за счет чего получается четыре различных шага для каждого слоя. Такое явное описание структуры нейронной сети позволяет компилятору проводить более тонкую оптимизацию операций свертки и умножения матриц, что благоприятно сказывается на производительности.

В [10] показано, что более глубокая оптимизация, основанная на разворачивании циклов, позволяет еще более повысить производительность.

Приемы построения эффективных алгоритмов в среде CUDA

Основным приемом перевода последовательного алгоритма в параллельный для реализации в среде CUDA является замена циклов параллельно выполняющимися командами вида SIMD. При этом, в частности, ускоряется выполнение операций умножения векторов и матриц. Вначале нужно разбить задачу на вычислительные блоки (blocks) и потоки (threads) [14], после чего, если необходимо, внести изменения в способ их взаимодействия и записать алгоритм в среде CUDA.

Этих преобразований при реализации алгоритмов, рассмотренных в настоящей работе, оказалось недостаточно для получения эффективного алгоритма в среде CUDA, поэтому пришлось прибегнуть к специальным приемам работы с различными видами видеопамяти, перечисленным ниже:

- использование общей [14] памяти видеокарты для хранения часто используемых значений;
- планирование порядка исполнения вычислительных потоков, чтобы избежать невыровненного (misaligned [14]) и непоследовательного (некогерентного, uncoalesced [14]) доступа к глобальной [14] памяти видеокарты;
- использование текстур для осуществления невыровненного доступа к глобальной памяти;
- изменение порядка обработки данных во избежание конфликтов доступа к отдельным участкам памяти.

Распознавание рукописных символов в среде CUDA с использованием сверточной нейросети

Ниже приведен алгоритм обучения нейронной сети на видеокарте.

Алгоритм 2 – параллельный алгоритм обучения сверточной нейронной сети

Шаг 1. Вычисление выходов X^0 начального слоя.

Шаг 2. Распознавание образов нейронной сетью:

Шаг 2.1. Для n от 1 до 4:

Шаг 2.1.1. Вычислить $\tilde{X}^{n-1}$.

Шаг 2.1.2. Вычислить выход n-го слоя:

$$X^n := F\left(\left(\tilde{X}^{n-1}\right)^T \times W^n\right) = F(Y^n).$$

Шаг 3. Обучение нейронной сети обратным распространением погрешности:

Шаг 3.1. Вычислить погрешность 4-го слоя dX^4.

Шаг 3.2. Для n от 4 до 1:

Шаг 3.2.1. Вычислить $Y_\varepsilon^n = X^n \otimes G(dX^n)$, где знаком $\otimes$ обозначено поэлементное умножение матриц, а $G(x)$ – первая производная активационной функции.

Шаг 3.2.2. Вычислить $d\tilde{X}^{n-1} := W^n \times Y_\varepsilon^n$.

Шаг 3.2.3. Преобразовать $d\tilde{X}^{n-1}$ к dX^{n-1}.

Шаг 3.2.4. Найти изменения весовых коэффициентов $dW^n := Y_\varepsilon^n \times X^n$.

Шаг 3.2.5. Произвести обновление весовых коэффициентов w_{ij}^n в соответствии с методом второго порядка [13].

Шаг 4. Ввод данных следующего образа и переход к шагу 1.

Матрицы $\tilde{X}^2$ и $\tilde{X}^3$ совпадают с X^2 и X^3, а $d\tilde{X}^3$ и $d\tilde{X}^2$ – с dX^3 и dX^2 соответственно. Ниже рассмотрена процедура формирования $\tilde{X}^0$, $\tilde{X}^1$ и $d\tilde{X}^1$.

Значения выхода начального слоя X^0 с использованием текстурной памяти записываются в матрицу $\tilde{X}^0$ размерностью 26×169, которая для оптимизации доступа к памяти дополнена до 32×169. Ее i-й столбец содержит выходы нейронов начального слоя, значения которых используются для вычисления выхода i-го нейрона каждой из шести карт первого слоя.

Весовые матрицы W^{1k}, $k=\overline{1,6}$ размерностью 5×5 и веса w_b^{1k} вспомогательных (bias) нейронов, соответствующие этим картам, по столбцам записаны в двумерный массив W^1 размерностью 26×6, который дополнен до 32×6. Выход первого слоя размещается в матрице X^1, имеющей размерность 169×6, и вычисляется, согласно шагу 2.1.2 алгоритма 2, следующим образом:

$$X^1 = F\left(\left(\tilde{X}^0\right)^T \times W^1\right) = F(Y^1). \qquad (1)$$

Выход первого слоя X^1 так же записывается в матрицу $\tilde{X}^1$ размерностью 156×25, которая дополнена до 192×25. Ее столбцы заполняются сходным с $\tilde{X}^0$ образом. Весовые коэффициенты 48 карт второго слоя W^{2kl}, $k=\overline{1,48}$, $l=\overline{1,6}$ размерностью 5×5 и шесть весов w_b^{2kl} вспомогательных нейронов записываются по столбцам в весовую матрицу W^2 размерностью 156×48, дополненную до 192×48. Выход второго слоя вычисляется также по формуле из шага 2.1.2:

$$X^2 = F\left(\left(\tilde{X}^1\right)^T \times W^{12}\right) = F(Y^2).$$

Процесс дублирования данных проиллюстрирован на рис. 3. Номерами в картах первого слоя X^{11}, X^{12} и т.д. отмечены рецепторные поля размерностью 5×5.

Обучение сверточной нейросети в среде CUDA

Обучение описанной сети методом обратного распространения погрешности проводилось в соответствии с формулами, использованными в шагах 3.2.3 и 3.2.4 алгоритма 1. При обучении второго слоя вводилась матрица $d\tilde{X}^1$ такой же размерности, как и $\tilde{X}^1$. Ее значения записывались в матрицу dX^1, имеющую такую же размерность, как и X^1, которая в дальнейшем использовалась для обучения первого слоя.

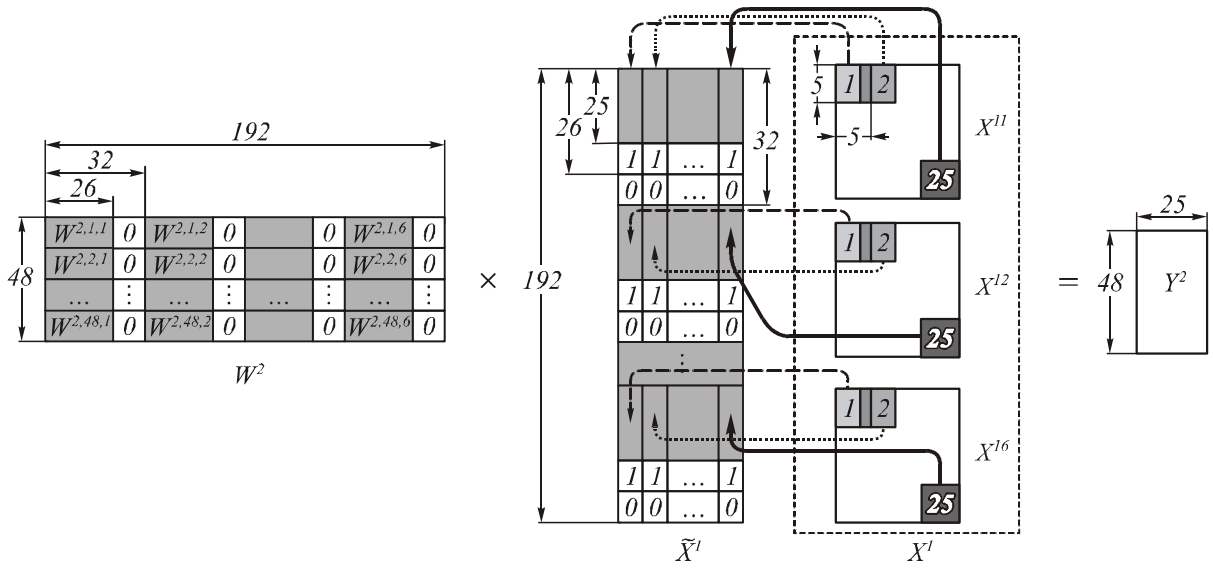

Рис. 3. Схема расположения данных в видеопамяти в процессе вычисления выхода второго сверточного слоя

Операции над матрицами (умножение, модификация матрицы внешним произведением) проводились с использованием функций специализированной библиотеки CUBLAS, содержащей реализацию BLAS (Basic Linear Algebra Subprograms, базового пакета подпрограмм линейной алгебры) над уровнем драйвера CUDA.

В процессе формирования матрицы dX^1 значения, соответствующие одному и тому же нейрону, суммировались. На рис. 4 области, соответствующие таким нейронам, заштрихованы.

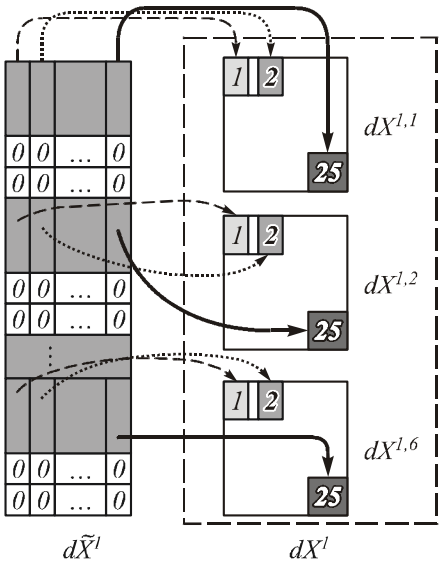

Рис. 4. Схема преобразования данных в видеопамяти в процессе вычисления погрешности нейронов первого сверточного слоя

Результаты вычислительных экспериментов

Все измерения длительности исполнения программ были проведены при помощи функции QueryPerformanceCounter из Windows API (Application Programming Interface) на следующей программно-аппаратной конфигурации: CPU Intel Pentium D 925 (3000 МГц), 2 ГБ DDR2 (PC2-5300), видеокарта на основе NVIDIA 9600GT; Microsoft Windows XP Service Pack 3; NVIDIA Forceware 191.07 (дата выпуска 05.10.2009); NVIDIA CUDA 2.3 (Toolkit + SDK); Microsoft C/C++ compiler 14.0 (в составе Microsoft Visual Studio 2005 Service Pack 1). Настройки оптимизации: Maximize Speed (/O2), Inline Function Expansion (/Ob1), Enable Intrinsic Functions (/Oi), Favor Fast Code (/Ot).

Обучение проводилось следующим образом. Перед началом очередной эпохи коэффициенты скорости обучения каждого веса корректировались на некоторую величину, зависящую от значения конкретного веса и от результата распознавания нейронной сетью 500 случайных образцов из тренировочного набора. Затем 60000 раз выбирался случайный образец из этого набора, и на нем проводилось обучение нейронной сети методом обратного распространения ошибки. В один момент времени нейронная сеть обучалась одному образцу. В конце эпохи проводилось тестирование качества полученной сети, которое более подробно описано ниже.

На этапе распознавания нейронной сети предъявлялось по одному из 10000 образцов тестового набора, результаты распознавания сравнивались с эталонными и делался вывод о ее качестве.

Длительности работы нейронной сети в режимах обучения и распознавания для трех описанных реализаций представлены в таблице 1. Длительность тестирования качества сети, проводившегося в конце эпохи, не включалась в результаты замеров длительности обучения.

Как видно из таблицы 1, длительность обучения нейронной сети на CUDA в 5,96 раз меньше таковой при обучении на CPU с использованием оптимизированного алгоритма, а длительность распознавания набора тестовых образцов меньше в 8,76 раза. Больший прирост производительности может быть

получен за счет одновременного обучения нескольких нейронных сетей (в режиме обучения) либо одновременного распознавания нескольких образов (в режиме распознавания). Повышение эффективности вычислений достигается за счет более полной загрузки мощностей GPU, что снижает вклад в длительность работы накладных расходов, связанных с передачей данных из оперативной памяти в видеопамять и с задержкой при запуске вычислений на GPU.

Таблица 1. Длительность работы нейронной сети в режимах обучения и распознавания

	Время обучения нейронной сети в течение 20 эпох, с	Время распознавания набора тестовых образцов, с
Алгоритм из [9]	39819	227,03
Оптимизированный алгоритм на CPU	10529	42,76
Алгоритм на CUDA	1767	4,88

Чтобы приблизительно оценить прирост производительности в условиях возможно более полной загрузки процессоров (как CPU, так и GPU), было смоделировано одновременное распознавание N образцов на CUDA. Для этого была исполнена процедура распознавания одного образца, принимающая в N раз большее количество данных и запускающая в N раз большее количество вычислительных потоков по сравнению с распознаванием одного образа.

В [9] показано, что использование двуядерного процессора, на котором запущено распознавание тестовой выборки в двухпоточном режиме, уменьшает длительность распознавания на 92%, а при использовании четырехъядерного в четырехпоточном режиме – на 270%.

Полученная зависимость ускорения расчетов на GPU по сравнению с оптимизированным алгоритмом, выполняющим вычисления только на одно-, дву- и четырехъядерном процессоре, от количества образцов, распознаваемых одновременно, отражена на рис. 5.

Как видно из рис. 5, прирост ускорения вычислений снижается при достижении состояния полной загрузки видеокарты. Более того, при одновременном распознавании 512 образов наблюдается снижение эффективности вычислений, которое можно объяснить усилением влияния издержек на запуск большого числа вычислительных потоков на видеокарте.

Важным различием между вычислениями, проводимыми на центральном и графическом процессорах, является порядок операций с вещественными числами, что выражается в различных ошибках округления. Поскольку в процессе обучения нейронной сети производится большое количество итераций, ошибка округления может существенно влиять на результат. В виду того, что поддержка чисел с плавающей точкой двойной точности в оборудовании, использованном в настоящем исследовании, отсутствует, все вычисления в данной работе (как на CPU, так и на GPU) велись с использованием чисел с плавающей точкой одинарной точности.

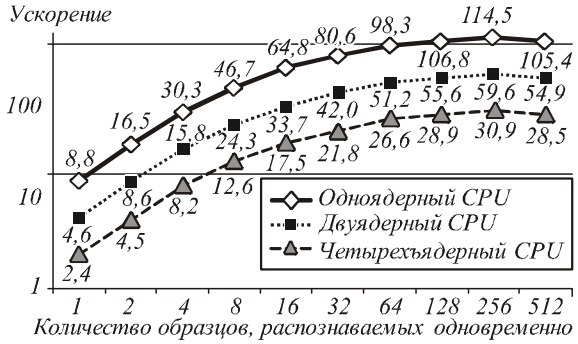

Рис. 5. Зависимость ускорения распознавания тестового набора от количества одновременно распознаваемых образов

Эти различия проявились в более высоком развитии способности генерализации у нейронной сети, обучавшейся на центральном процессоре, что выразилось в меньшем количестве ошибок, делаемых ею на тестовой выборке, по сравнению с сетью, обучавшейся на GPU (рис. 6).

Из рис. 6 также видно, что качество нейронной сети, обученной на GPU, ограничено 1,6% (около 160 из 10000) ошибок распознавания тестового набора образцов, в то время как сеть, обученная на CPU, делает около 1,3% ошибок. Эти результаты сравнимы с результатами обучения сверточных нейронных сетей различных конфигураций, представленными в [15]. В частности, сверточные нейронные сети LeNet-4 и LeNet-5, обучение которых велось на том же тренировочном наборе, что и в настоящей работе, на тестовом наборе MNIST делают около 1,1% и 0,95% ошибок соответственно.

Дальнейшее повышение качества нейронной сети может быть достигнуто изменением ее конфигурации либо, при сохранении прежней структуры, путем расширения тренировочного набора. В [15] показано, что для этого можно использовать искажение исходных образцов тренировочного набора применением деформации вида сдвиг, растяжение, поворот.

Заключение

Рассмотренные в данной работе приемы повышения эффективности CUDA-программ (планирование порядка обращения к видеопамяти во избежание конфликтов доступа, использование текстурной и константной памяти при невозможности избежать невыравненного доступа к памяти) показали свою действенность при реализации параллельных алгоритмов.

Проведенные вычислительные эксперименты продемонстрировали эффективность реализации нейросетевых алгоритмов распознавания рукопис-

ных цифр на GPU. Программа, реализующая вычисления в среде CUDA, обеспечивает ускорение обучения сверточной нейронной сети в 5,96, а скорости распознавания – в 8,76 раз (по сравнению с оптимизированным алгоритмом, проводящим вычисления только на центральном процессоре). Если же сравнивать процессоры в условиях максимально полной загрузки их вычислительных ресурсов (максимальная загрузка и CPU, и GPU), то длительность распознавания на GPU может быть меньше по сравнению с одноядерным CPU приблизительно в 100 раз.

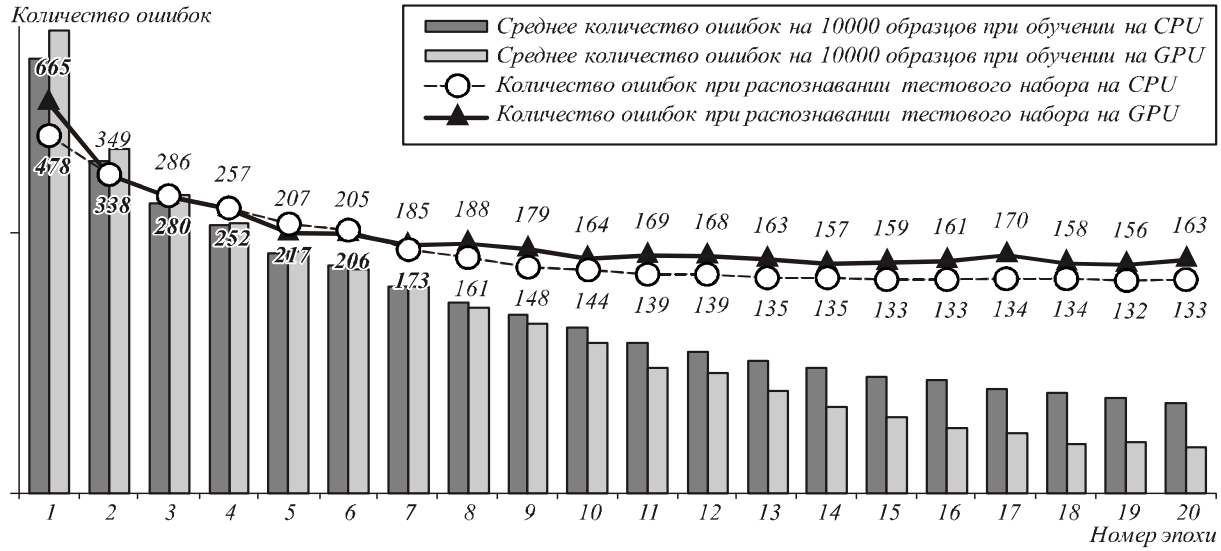

Рис. 6. Зависимость количества ошибок, сделанных нейронной сетью, от длительности обучения

Среда CUDA позволяет реализовывать вычислительно более сложные нейросетевые алгоритмы классификации и кластеризации изображений, финансового прогнозирования и добычи данных по сравнению с теми, при реализации которых графические ускорители не используются.

Благодарности

Работа выполнена при поддержке российско-американской программы «Фундаментальные исследования и высшее образование» ("BRHE") и грантов РФФИ №№ 10-07-00553-а и 10-04-00723-а.

Литература

1. **Акулов, П.В.** Решение задач прогнозирования с помощью нейронных сетей [Электронный ресурс] // Портал магистров Донецкого национального технического университета [сайт]. [2006]. URL: http://masters.donntu.edu.ua/2006/fvti/akulov/diss/index.htm (дата обращения: 26.10.2009).
2. **Кульчин, Ю.Н.** Нейро-итерационный алгоритм томографической реконструкции распределенных физических полей в волоконно-оптических измерительных системах / Ю.Н. Кульчин, Б.С. Ноткин, В.А. Седов // Компьютерная оптика. – 2009. – Т. 33, № 4. – С. 446-455.
3. **Довженко, А.Ю.** Параллельная нейронная сеть с удаленным доступом на базе распределенного кластера ЭВМ: [Текст] / А.Ю. Довженко, С.А. Крашаков // Тезисы докл. II международн. симп. «Компьютерное обеспечение химических исследований» (Москва, 22-23 мая 2001 г.) и III Всерос. школы-конф. по квантовой и вычисл. химии им. В.А. Фока. – С. 52-53.
4. **Fatica, M.** CUDA for High Performance Computing – Materials of HPC-NA Workshop 3, January 2009.
5. Belgian researchers develop desktop supercomputer [Электронный ресурс] // FASTRA [сайт]. [2008]. URL: http://fastra.ua.ac.be/en/index.html (дата обращения: 28.10.2009).
6. **Stone, J.E.** Accelerating molecular modeling applications with graphics processors / John E. Stone, James C. Phillips, Peter L. Freddolino, David J. Hardy, Leonardo G. Trabuco, and Klaus Schulten // Journal of Computational Chemistry, 28:2618-2640, 2007.
7. New NVIDIA Tesla GPUs Reduce Cost Of Supercomputing By A Factor Of 10 [Электронный ресурс] // NVIDIA – World Leader in Visual Computing Technologies [сайт]. [2009]. URL: http://www.nvidia.com/object/io_1258360868914.html (дата обращения: 20.10.2009).
8. **Simard, P.** Best practices for convolutional neural networks applied to visual document analysis / Patrice Simard, Dave Steinkraus, and John C. Platt // In Proceedings of ICDAR 2003. – 2003. – P. 958–962.
9. **O'Neill, M.** Neural Network for Recognition of Handwritten Digits [Электронный ресурс] / Mike O'Neill // Codeproject. Free source code and programming help [сайт]. [2006]. URL: http://www.codeproject.com/KB/library/NeuralNetRecognition.aspx (дата обращения: 20.09.2009).
10. **Царегородцев, В.Г.** Неисчерпаемы, как атом (о нейронных сетях и нейропрограммах) [Электронный ресурс] // NeuroPro – нейросети, анализ данных, прогнозирование и классификация [сайт]. [2008]. URL: http://www.neuropro.ru/memo311.shtml (дата обращения: 20.10.2009).
11. **LeCun, Y.** Convolutional networks for images, speech, and time-series / Y. LeCun and Y. Bengio. In M. A. Arbib, editor // The Handbook of Brain Theory and Neural Networks, MIT Press, 1995.
12. **Rumelhart, D.E.** Learning Internal Representations by Error Propagation / David E. Rumelhart, G.E. Hinton,

R.J. Williams // In Parallel Distributed Processing, Cambridge: M.I.T. Press, 1986. – V. 1. – P. 318-362.
13. **LeCun, Y.** Efficient BackProp / Y. LeCun, L. Bottou, G. Orr and K. Muller – Neural Networks: Tricks of the trade, Springer, 1998.
14. NVIDIA CUDA Programming Guide Version 2.3.1 [Электронный ресурс] // NVIDIA – World Leader in Visual Computing Technologies [сайт]. [2009]. URL: http://www.nvidia.com/object/cuda_develop.html (дата обращения: 20.10.2009).
15. **LeCun, Y.** The MNIST database of handwritten digits [Электронный ресурс] // MNIST handwritten digit database, Yann LeCun and Corinna Cortes [сайт]. [2009]. URL: http://yann.lecun.com/exdb/mnist (дата обращения: 20.10.2009).

References

1. **Akulov, P.V.** Solving problems of forecasting using neural networks // Master's portal Donetsk National Technical University. [2006]. URL: http://masters.donntu.edu.ua/2006/fvti/akulov/diss/index.htm (verified at 10/26/2009). – (in Russian).
2. **Kulchin, Yu.N.** Neuro-iterative algorithm of tomographic reconstruction of the distributed physical fields in the fibre-optic measuring systems / Yu.N. Kulchin, B.S. Notkin, V.A. Sedov // Computer Optics. – 2009. – T. 33, № 4. – P. 446-455. – (in Russian).
3. **Dovzhenko, A.Yu.** Parallel neural network with remote access based on a distributed cluster of computers / A.Yu. Dovzhenko, S.A. Krashakov // Abstracts of II International conference on new techniques and applications of modern physical chemical methods for environmental studies. – P. 52-53. – (in Russian).
4. Fatica, M. CUDA for High Performance Computing: materials of HPC-NA Workshop 3 (January 2009).
5. Belgian researchers develop desktop supercomputer // FASTRA. URL: http://fastra.ua.ac.be/en/index.html (verified at 10/28/2009).
6. John E. Stone, James C. Phillips, Peter L. Freddolino, David J. Hardy, Leonardo G. Trabuco, and Klaus Schulten. Accelerating molecular modeling applications with graphics processors. Journal of Computational Chemistry, 28:2618-2640, 2007.
7. New NVIDIA Tesla GPUs Reduce Cost Of Supercomputing By A Factor Of 10 // NVIDIA – World Leader in Visual Computing Technologies. [2009]. URL: http://www.nvidia.com/object/io_1258360868914.html (verified at 10/20/2009).
8. Patrice Simard, Dave Steinkraus, and John C. Platt. Best practices for convolutional neural networks applied to visual document analysis. In Proceedings of ICDAR 2003, pages 958–962, 2003.
9. Mike O'Neill. Neural Network for Recognition of Handwritten Digits // Codeproject. Free source code and programming help. [2006]. URL: http://www.codeproject.com/KB/library/NeuralNetRecognition.aspx (verified at 09/20/2009).
10. **Tsaregorodtsev, V.G.** Inexhaustible as the atom (about neural networks and neural applications). [2008]. URL: http://www.neuropro.ru/memo311.shtml (verified at 10/20/2009). – (in Russian).
11. LeCun, Y., Y. Bengio. Convolutional networks for images, speech, and time-series. In M. A. Arbib, editor, The Handbook of Brain Theory and Neural Networks. MIT Press, 1995.
12. Rumelhart, David E.; Hinton, G.E.; Williams, R.J. Learning Internal Representations by Error Propagation. In Parallel Distributed Processing, Cambridge: M.I.T. Press, v. 1, p. 318-362 (1986).
13. Y. LeCun, L. Bottou, G. Orr and K. Muller. Efficient BackProp – Neural Networks: Tricks of the trade, Springer, 1998.
14. NVIDIA CUDA Programming Guide Version 2.3.1 // NVIDIA – World Leader in Visual Computing Technologies. [2009]. URL: http://www.nvidia.com/object/cuda_develop.html (verified at 20.10.2009).
15. Y. LeCun. The MNIST database of handwritten digits // MNIST handwritten digit database, Yann LeCun and Corinna Cortes. [2009]. URL: http://yann.lecun.com/exdb/mnist (verified at 10/20/2009).

TECHNOLOGY OF IMPLEMENTATION OF NEURAL NETWORK ALGORITHM IN CUDA ENVIRONMENT AT THE EXAMPLE OF HANDWRITTEN DIGITS RECOGNITION

P.Y. Izotov [1], S.V. Sukhanov [1], D.L. Golovashkin [2],
[1] S. P. Korolyov Samara State Aerospace University,
[2] Image Processing Systems Institute of the Russian Academy of Sciences

Abstract

On a convolution neural network example features of implementation of pattern recognition algorithm on Graphic Processing Unit (GPU) on NVIDIA CUDA are shown. Duration of training of a network on the video adapter is reduced in 5.96, and recognition of test samples set in 8.76 times in comparison with the optimised algorithm which uses only central processor (CPU) for calculations. Perspective of implementation of such neural network algorithms on graphic processors is shown.

Key words: convolutional neural network, pattern recognition, neural network training, back-propagation of error, parallel computing, GPGPU, NVIDIA, CUDA, matrices multiplication, CUBLAS.

Сведения об авторах

Изотов Павел Юрьевич, математик-системный программист, стажер-исследователь Института систем обработки изображений РАН. E-mail: *izogfif@rambler.ru.* Область научных интересов: параллельные вычисления на гетерогенных системах.

P. Y. Izotov, a mathematician-system programmer, Trainee Researcher at the Image Processing Systems Institute of the Russian Academy of Sciences. E-mail: *izogfif@rambler.ru*. Research Interests: parallel computing on heterogeneous systems.

Суханов Сергей Васильевич, кандидат технических наук, доцент кафедры технической кибернетики Самарского государственного аэрокосмического университета им. С.П. Королева . E-mail*: sukhanov@smr.ru.*

Область научных интересов: компьютерные телекоммуникации, параллельные вычисления на гетерогенных системах.

S. V. Sukhanov, Candidate of Technical Sciense, Associate Professor at the Department of Technical Cybernetics of the Samara State Aerospace University. E-mail: *sukhanov@smr.ru*. Research interests: computer tellecomunications, parallel computing on heterogeneous systems.

Головашкин Димитрий Львович, доктор физико-математических наук, доцент кафедры технической кибернетики Самарского государственного аэрокосмического университета им. С.П. Королева, старший научный сотрудник Института систем обработки изображений РАН. E-mail: *dimitriy@smr.ru*. Область научных интересов: разностное решение уравнений Максвелла (FDTD метод); дифракционная оптика; векторные и параллельные матричные вычисления.

D. L. Golovashkin, Doctor of Physical and Mathematical Sciences, Associate Professor at the Department of Technical Cybernetics of the Samara State Aerospace University, a Senior Researcher at the Image Processing Systems Institute of the Russian Academy of Sciences. E-mail: *dimitriy@smr.ru*. Research interests: FDTD method; diffractive optics; vector and parallel matrix computations.

Поступила в редакцию 9 апреля 2010 г.