DOI 10.37882/2223–2966.2021.09.11

РАЗРАБОТКА САМООБУЧАЮЩЕЙСЯ МОДЕЛИ СБОРА И ОБРАБОТКИ ИНФОРМАЦИИ НА ГРАФИЧЕСКОМ ПРОЦЕССОРЕ ДЛЯ РАСПОЗНАВАНИЯ МНОГОПАРАМЕТРИЧЕСКИХ ОБЪЕКТОВ

Игнатьев Денис Алексеевич
Аспирант, Институт Вычислительной математики и Математической Геофизики (ИВМиМГ СО РАН) (г. Новосибирск)
den.ignatev@gmail.com

Аннотация. Автоматическое распознавание объектов является востребованной задачей для многих отраслей промышленности, начиная от военной разведки и заканчивая производством. Возможность визуального мониторинга окружающей среды без наблюдателя потенциально приводит к повышению производительности. Целью данной работы является разработка самообучающейся модели сбора и обработки информации на графическом процессоре для распознавания многопараметрических объектов. Полученные результаты показали, что реализация глубокой сети с помощью графического процессора обеспечивает быстрое решение для общего распознавания объектов. Отсутствие врожденных свойств системы делает ее непрактичной для большинства промышленных применений. Однако следует также отметить, что никакие исходные свойства системы не делают ее особенно удобной для поиска шаблонов в изображениях, в отличие от других входных пространств. В частности, ничто в системе не использует структуру входного пространства, которая идет с использованием изображений. Возможные улучшения могут быть использованы путем интеграции других методов, используемых для распознавания изображений, в ограниченную машину Больцмана или глубокую сеть. Таким образом, предложенная система отлично справляется с поставленными задачами и может быть полезной в полезной областях.

Ключевые слова: автоматическое распознавание объектов, нейронная сеть, самообучающаяся модель, графический процессор, многопараметрические объекты.

DEVELOPMENT OF A SELF-LEARNING MODEL OF COLLECTION AND PROCESSING OF INFORMATION ON A GRAPHIC PROCESSOR FOR RECOGNIZING MULTI-PARAMETER OBJECTS

D. Ignatiev

Summary. Automatic object recognition is a demanded task for many industries, from military intelligence to manufacturing. The ability to visually monitor the environment without an observer has the potential to increase productivity. The purpose of this work is to develop a self-learning model for collecting and processing information on a graphics processor for recognizing multiparameter objects. The results showed that the GPU-assisted deep network implementation provides a fast solution for general object recognition. The lack of inherent properties of the system makes it impractical for most industrial applications. However, it should also be noted that none of the initial properties of the system make it particularly convenient for finding patterns in images, unlike other input spaces. In particular, nothing in the system uses the input space structure that comes with the use of images. Potential improvements can be exploited by integrating other techniques used for image recognition into a constrained Boltzmann machine or deep network. Thus, the proposed system does an excellent job with the assigned tasks and can be useful in useful areas.

Keywords: automatic object recognition, neural network, self-learning model, graphics processor, multi-parameter objects.

Введение

Автоматическое распознавание объектов является востребованной задачей для многих отраслей промышленности, начиная от военной разведки и заканчивая производством [1, 2].

Возможность визуального мониторинга окружающей среды без наблюдателя потенциально приводит к повышению производительности. Например, изображения с беспилотных летательных аппаратов, могут быть отсканированы для раннего обнаружения угроз и обеспечения надлежащих мер предосторожности. Если задача сформулирована и имеются данные предметной области, могут быть созданы сложные программируемые системы. Однако, из-за сложности визуального ввода общая мультиклассовая система требует высокой вычислительной мощности [3, 4, 5].

Доступ к памяти в GPU является связанным — если считывается минимальная единица текстуры трёхмерного объекта, пиксел текстуры или тексель, то через несколько тактов считывается соседний тексель; если записывается пиксель, то через несколько тактов будет

записываться соседний [6]. Разумно организуя память, можно получить производительность, близкую к теоретической пропускной способности [7]. Это означает, что GPU, в отличие от CPU, не требуется огромного кэша, поскольку его роль заключается в ускорении операций текстурирования. В этом случае необходимы килобайты, содержащие несколько текселей, используемых в билинейных и трилинейных фильтрах [8].

Благодаря методам вычислений на графических процессорах (GPU) возможно улучшить качество систем распознавания образов на графических изображениях. Использование графических процессоров является наиболее целесообразным. GPU предназначен для быстрой визуализации изображений, поэтому эффективно использовать те же аппаратные комплектующие для распознавания образов. GPU также имеют ряд недостатков по сравнению с CPU: слабое ветвление и строгие ограничения памяти. Однако реализация такой системы на GPU возможна, она способна обеспечить оперативное и точное решение проблем [9, 10].

Цель данной работы

Разработка самообучающейся модели сбора и обработки информации на графическом процессоре для распознавания многопараметрических объектов.

Материалы и методы

Описываемая в рамках данной работы система разработана в программно-аппаратной архитектуре параллельных вычислений CUDA. Одним из ограничивающих факторов CUDA является то, что ядра не могут быть определены или выполнены изнутри класса, что является технически выполнение может происходить, но не чисто объектно-ориентированным способом. Это обусловлено необходимостью определений ядер с помощью директивы __global__. В результате вызовы должны выполняться из статических функций внутри основного файла.cu, а не внутри класса. При этом классы имеют полный доступ к памяти устройства. Классы могут хранить указатели на расположение устройств, выделять память на устройстве и копировать между устройством и хостом [11, 12, 13].

Учитывая перечисленное выше, система была спроектирована следующим образом. Классы созданы для подкомпонентов, используемых всеми системами. Эти субъединицы содержат всю системную информацию и могут представлять содержащиеся в ней данные. Созданы классы более высокого уровня, которые упорядочивают эти подкомпоненты таким образом, чтобы определить предполагаемую систему обучения. Эти классы высокого уровня связаны с файлами.cu, которые определяют ядра и основные процессы, происходящие во время обучения. Наконец, класс «тренера» используется для управления увиденными примерами и процессом обучения.

В настоящее время внедрены три системы обучения. Ограниченная Машина Больцмана содержит простейшую структуру, содержащую всего два слоя и связь между ними. Сеть глубоких убеждений содержит 3 слоя и соответствующие связи. Нейронная сеть содержит дополнительный верхний уровень с k двоичными единицами, где k — количество классов в данных [13, 14, 15].

Каждая система содержит ряд функций передачи к подкомпонентам для получения системных фактов и местоположения памяти устройства. Вызов функций действует как оболочка, которая придает служебному уровню больше смысла в отношении его использования в системе. Кроме того, каждый системный класс позволяет загружать и сохранять отдельные подкомпоненты одним вызовом [16, 17].

Файлы *.cu, связанные с машиной, определяют ядра и режимы обучающих систем [18]. Программа использует #defines для создания конфигурации системы, которая будет использоваться для обучения. Это, конечно, требует перекомпиляции при каждом изменении системы. Может показаться, что этого можно избежать с помощью файла конфигурации, который можно редактировать в виде обычного текста и читать скомпилированной программой. Однако во время тестирования это вызвало некоторые проблемы с замедлением работы ряда ядер CUDA. Замедление происходит, когда условие цикла зависит от переданной переменной. Это происходит независимо от того, хранится ли эта переменная на устройстве или хосте. Если переменная хранится в #define, компилятор знает точное значение во время компиляции и может оптимизировать. Скорее всего, это происходит в форме разворачивания цикла, но это также может обеспечить некоторую гарантию итераций, необходимых для реального объединения обращений к памяти [19].

Как правило, при работе с CUDA рекомендуется хранить все данные, к которым необходимо получить доступ в ядре, в памяти устройства. Время передачи между хостом и устройством является самым медленным из всех обращений к памяти, поэтому его следует избегать любым способом [20, 21]. Однако в случае обновления веса несколько переменных (импульс и скорость обучения) передаются по значению без какого-либо измеримого замедления. Скрытие задержки, которое выполняется, позволяя готовым потокам запускаться в ожидании медленного доступа к памяти, объясняют

аналогичные сроки. Однако это не относится к более быстрым вычислениям, таким как проход вверх.

Помимо этого, остальная часть реализации была простой, а объектно-ориентированная структура компонентов позволяла создавать чистый и простой код. Как и в случае со всем кодом, есть возможности для улучшения и, безусловно, еще более краткий способ проектирования частей системы. Были использованы методы гибкой разработки, и система была разработана в течение нескольких итераций.

База данных MNIST представляет собой большую коллекцию рукописных цифр от 0 до 9. Набор данных представляет собой подмножество из более крупной базы данных NIST с некоторыми изменениями, чтобы сделать его более стандартизированным в18]. Набор содержит 60 000 обучающих примеров и 10 000 тестовых изображений. Все цифры были нормализованы по размеру и центрированы. Центрированы относительно вычисленного центра масс пикселей Изображения 8-битные, серого оттенка, в отличии от исходных двоичных данных, в результате нормализации.

Самая низкая частота ошибок в тестовом наборе составляет 0,23% при использовании из 35 сверточных сетей и дополнительных методов. Этот результат также использовал эластичное искажение данных, чтобы, по сути, позволить увидеть больше обучающих примеров. 3-слойный NN с 500 и 150 скрытыми блоками, использующими данные MNIST, привел к частоте ошибок 2,95%.

Этот набор данных использовался для проверки системы и для того, чтобы увидеть преимущества, полученные от различных методов обучения. Использовались только исходные данные, и системы обучались с использованием постоянного числа эпох и последовательных темпов обучения по всем методам. Данные были представлены системе как «вероятность пикселя», что означает, что значение оттенков серого каждого пикселя было преобразовано в диапазон от 0 до 1. Система содержала 784 визуальных слоя, 512 скрытых слоев, 512 скрытых слоев и верхний слой из 10 блоков softmax.

Первый запуск системы был обучен с использованием CD_1. Для каждой части обучения было выбрано стандартное количество эпох, чтобы можно было сравнить с более поздними версиями. Производительность системы могла бы быть улучшена за счет более длительных циклов, поскольку, похоже, не происходило переобучения, что очевидно при сравнении свободной энергии на данных обучения и проверки. Изучались только сравнительные графики свободной энергии из прогона PCD. Использованная в ходе исследования конфигурация системы для MNIST выглядит следующим образом: 10 единиц Softmax, 512 скрытых единиц двоичного 0 или 1, 784 визуального блока 28*28 плавающих изображений 0–1. Использованная в ходе исследования конфигурация системы для NORB выглядит следующим образом: 5 единиц Softmax, 4096 скрытых единиц двоичного 0 или 1, 9216 визуального блока 96*96 плавающих изображений 0–1.

Идентичная система использовалась для тренировки с использованием CD_5. Тренировка прошла примерно в четыре раза медленнее, чем CD_1. Результаты классификации показали улучшение примерно на 0,2% по всем наборам. Считается, что система не ухудшилась от переобучения.

Эта же система была обучена с использованием непрерывной цепи Маркова по мини-пакетам и эпохам. Это было единственное отличие от тренировочных прогонов CD_n. Было выявлено, что свободная энергия данных проверки не будет резко расти по сравнению с данными обучения. При этом фактическое значение свободной энергии произвольно, и RBM второго уровня имеет более высокие значения. Ошибка нейронной сети быстро улучшается с самого начала из-за того, что веса верхнего уровня примерно совпадают с правильными конфигурациями предпоследнего слоя. Прогресс становится более постепенным, поскольку веса начинают точно настраиваться для классификации.

Слои, которые были предварительно обучены с помощью PCD, затем были настроены с помощью методов, описанных ранее. Точно настроенные слои затем использовались для инициализации нейронной сети. Частота исходных ошибок была немного выше, скорее всего, из-за случайной инициализации веса верхнего уровня. Глобально настроенная сеть стабильно опережала комплексные RBM по всем наборам данных.

Набор данных NORB был разработан для проверки общего распознавания 3D-объектов. Было сфотографировано 50 игрушек из 5 классификационных групп в различных условиях. Категории: четвероногие животные, человеческие фигуры, самолеты, грузовики и автомобили. Две камеры использовались для получения стереоизображений при различных условиях освещения, возвышениях и углах.

Данные представлялись системе в виде «вероятностей» пикселей с плавающей запятой между 0 и 1. Для ускорения обучения второе изображение каждого стереовхода было выброшено. Это вдвое уменьшает размер входных данных и значительно сокращает время обучения. Предварительное обучение RBM

Таблица 1. Параметры обучения

	Эпохи	Размер мини-пакета	Распределение Гиббса	Скорость обучения	Снижение скорости обучения
RBM уровень 1 (CD_1)	50	100	1	0.001	0
RBM уровень 2 (CD_1)	50	100	1	0.001	0
Нейронная сеть	200	20	–	0.01	0.0001
RBM уровня 1 (PCD)	50	100	1	0.001	0
RBM уровня 2 (PCD)	50	100	1	0.001	0
Глобальная тонкая настройка	100	100	1	0.001	0.0001
RBM NORB уровня 1	500	100	1	0.001	0.0001
RBM NORB уровня 2	500	100	1	0.001	0.0001
FPCD	100	100	1	0.001	0.0001

Таблица 2. Параметры тренировки

	Эпохи	Размер мини-пакета	Распределение Гиббса	Скорость обучения	Снижение скорости обучения
RBM уровень 1 (CD_5)	50	100	5	0.001	0
RBM уровень 2 (CD_5)	50	100	5	0.001	0

Таблица 3. Результаты классификации

	Обучающая последовательность	Набор валидации	Набор тестов
с использованием CD1			
Источник	1193	151	273
Ошибочная классификация	2.2%	2.5%	2.7%
с использованием CD5			
Источник	1054	132	259
Ошибочная классификация	1.95%	2.2%	2.6%
PCD			
Источник	1088	127	257
Ошибочная классификация	2.01%	2.1%	2.6%
Глобально настроенная DBN			
Источник	933	117	242
Ошибочная классификация	1.7%	1.95%	2.4%
Набор данных NORB			
Источник	13683	1527	15367
Ошибочная классификация	23.7%	24%	29.8%

казалось неэффективным, поскольку ошибка валидации начиналась относительно выше после первой эпохи и постепенно снижалась. Возможно, данные были слишком сложными или ввод необработанных пикселей был неправильным методом. Успех был достигнут при использовании сырых пиксельных данных на аналогичном обучающем устройстве, известном как Глубокая машина Больцмана. Это обобщение машины Больцмана, к которой добавлено несколько «невидимых» уровней. В отличие от «видимых» уровней, которые моделируют наблюдаемую конфигурацию спинов, по «невидимым» уровням производится суммирование. Более высокая скорость обучения, использованная в этом исследовании, также могла объяснить улучшение предварительной подготовки и лучшие результаты классификации.

Результаты

Полученные параметры обучения для различных систем представлены в таблице 1. Полученные параметры тренировки представлены в таблице 2.

Система четко научилась различать 5 классов. Один графический процессор для настольных ПК способен находить различия между 5 довольно похожими типами объектов. Более глубокий анализ результатов дает некоторое представление о реальных приложениях такой системы и исходной постановке проблемы.

Окончательная нейронная сеть способна выполнить 50 классификаций за 192 мс. Это означает, что при разрешении 96x96 текущая система способна анализировать более 250 изображений в секунду. Эти изображения, конечно, также могут быть образцами изображения. Это позволяет анализировать изображения гораздо большего размера, если система обучена на таких данных. Например, если система была обучена путем нарезки изображений с беспилотника на равные части, она могла бы анализировать одно большое изображение, используя нарезки таких же размеров. Более того, поскольку используется один графический процессор для настольных ПК, было бы возможно и, скорее всего, рентабельно иметь массив таких устройств, использующих одни и те же предварительно обученные параметры, но анализирующих разные сегменты системы.

Вывод

Результаты показывают, что реализация глубокой сети с помощью графического процессора обеспечивает быстрое решение для общего распознавания объектов. Отсутствие врожденных свойств системы делает ее непрактичной для большинства промышленных применений. Однако следует также отметить, что никакие исходные свойства системы не делают ее особенно удобной для поиска шаблонов в изображениях, в отличие от других входных пространств. В частности, ничто в системе не использует структуру входного пространства, которая идет с использованием изображений. Возможные улучшения могут быть использованы путем интеграции других методов, используемых для распознавания изображений, в ограниченную машину Больцмана или глубокую сеть. Таким образом, предложенная система отлично справляется с поставленными задачами и может быть полезной в полезной областях.

ЛИТЕРАТУРА

1. Агеев, А.Д. Нейроматематика. Книга 6: учебное пособие для вузов [Текст] / А.Д. Агеев, А.Н. Балухто, А.В. Бычков, С.А. Верещагин и др.— М.: ИПРЖР, 2002.— 448 с.
2. Wang, M.Y. Regionlets for generic object detection [Text] / M.Y. Wang, S. Zhu, Y. Lin // International Conference on Computer Vision.— Sydney, 2013.— pp. 17–24.
3. Осовский, С. Нейронные сети для обработки информации [Текст] / С. Оссовский.— М.: Финансы и статистика, 2002.— 344 с.
4. Тархов, Д.А. Нейронные сети. Модели и алгоритмы. Книга 18 [Текст] / Д.А. Тархов.— М.: Радиотехника, 2005.— 256 с.
5. Царегородцев, В.Г. Свёрточные нейронные сети с полиномиальными (high-order) сумматорами нейронов [Электронный ресурс] // URL: http://neuropro.ru/memo334.shtml (дата обращения 10.06.2021).
6. Козадаев, А.С. Принципы реализаций искусственной нейронной сети [Текст] / А.С. Козадаев // Вестник Тамбовского университета.—2010.— No 1.— Т. 15.— С. 108–110.
7. Васильев, В.И. Распознающие системы. Справочник. 2-е издание [Текст] / В.И. Васильев.— Киев: Наукова думка, 1983.— 424 с.
8. Галушкин, А.И. Нейрокомпьютеры и их применение: учебное пособие для ВУЗов. Книга 1 — Теория нейронных сетей [Текст] / А.И. Галушкин.— М.: ИПРЖР, 2000.— 416 с.
9. Cadieu, C.F. Deep Neural Networks Rival the Representation of Primate IT Cortex for Core Visual Object Recognition [Text] / C.F. Cadieu, H. Hong, D.L.K. Yamins, N. Pinto // Computational Biology.— 2014.— Vol. 10.— pp. 1–18.
10. Everitt, B.S. Cambridge Dictionary of Statistics, 4th edition [Text] / B.S. Everitt.— Cambridge University Press, 2010.— 480 p.
11. Ciresan, D.C. Handwritten Digit Recognition with a Committee of Deep Neural Nets on GPUs [Text] / D.C. Ciresan, U. Meier, L.M. Gambardella, J. Schmidhuber // International Conference on Computer Vision.— Portugal, 2011.— pp. 240–254.
12. Reed, S. Training Deep Neural Networks on Noisy Labels with Bootstrapping [Text] / S. Reed, D. Anguelov, C. Szegedy, D. Erhan, A. Rabinovich // Neural and Evolutionary Computing.— 2014.— Vol. 4.— pp. 412–428.
13. Желтов, С.Ю. Обработка и анализ изображений в задачах машинного зрения [Текст] / С.Ю. Желтов.— М.: Физматкнига, 2010.— 672 с.
14. Забалуев, М. Описание реализации программы. Оценка производительности эмуляции нейронной сети [Текст] / М. Забалуев.— М.: Модуль, 1999.— 22 с.
15. Shaika, K.B. Comparative Study of Skin Color Detection and Segmentation in HSV and YCbCr Color Space [Text] / K.B. Shaika, P. Ganesana, V. Kalista, B.S. Sathisha // Computer Science.— 2015.— Vol. 57.— pp. 41–48.
16. Царегородцев, В.Г. Оптимизация предобработки признаков выборки данных: критерии оптимальности [Текст] / В.Г. Царегородцев // Нейрокомпьютеры.— 2005.— No4.— С. 32–40.

17. Agostinelli, F. Learning Activation Functions to Improve Deep Neural Networks [Text] / F. Agostinelli, M. Hoffman, P. Sadowski, P. Baldi // International Conference on Learning Representations. — Puerto Rico, 2015. — pp. 1024–1032.
18. Goyal, S. Object Recognition Using Deep Neural Networks: A Survey [Text] / S. Goyal, P. Benjamin // Neural and Evolutionary Computing. —2014. — Vol. 2. — pp. 1–16.
19. Carreira, S. Constrained parametric min-cuts for automatic object segmentation [Text] / S Carreira, C. Sminchisescu // Transactions on pattern analysis machine intelligence. — 2012. — Vol. 34. — pp.1312–1328.
20. Fink, G.A. Models for Pattern Recognition From Theory to Applications [Text] / G.A. Fink, B.M. Markov. — Berlin Heidelberg: Springer- Verlag, 2008. — 424 p.
21. Girshick, R. Rich feature hierarchies for accurate object detection and semantic segmentation [Text] / R. Girshick, J. Donahue, T. Darrell, J. Malik // Computer Vision and Pattern Recognition. — Columbus, 2014. — pp. 580–587.

Журнал «Современная наука: актуальные проблемы теории и практики»

г. Новосибирск