Т. 1, № 1, 2019

https://doi.org/10.23939/csn2019.01.097

УДК 004.8

ПОРІВНЯЛЬНИЙ АНАЛІЗ ПРОГРАМНО-АПАРАТНОГО ЗАБЕЗПЕЧЕННЯ АЛГОРИТМІВ ГЛИБОКОГО НАВЧАННЯ

Ю. В. Хома[1], А. Я. Бенч[2]

Національний університет "Львівська політехніка",
[1] кафедра інформаційно-вимірювальних технологій,
[2] кафедра теоретичної радіотехніки та радіовимірювань

Автоматичний переклад, розпізнавання мови та її синтез, розпізнавання об'єктів та навіть людських емоцій – надзвичайно складні завдання, із якими легко справляються сучасні смартфони. Їх ефективна реалізація стала можливою завдяки широкому застосуванню алгоритмів штучного інтелекту та машинного навчання, серед яких надзвичайно популярними є штучні нейронні мережі та алгоритми глибокого навчання. Ці алгоритми проникли в усі галузі індустрії, а їх стрімкий розвиток неможливий без застосування апаратної акселерації та чіткої взаємодії між апаратними складовими та програмним забезпеченням. Особливо актуальним це завдання стає, коли програмне забезпечення, призначене для застосування в хмарах, адаптується для невеликих за розміром та обчислювальними потужностями вбудованих систем. Статтю присвячено трьом пунктам, що, відповідно, пов'язані з програмним забезпеченням глибокого навчання, спеціалізованою апаратурою на основі GPU та перспективами побудови акселераторів для алгоритмів глибокого навчання на основі програмованих логічних матриць. У роботі проведено порівняльний аналіз найпопулярніших програмних фреймворків, таких як Caffe, Theano, Torch, MXNet, Tensorflow, Neon, CNTK. Описано переваги GPU-рішень на основі CUDA і cuDNN. Розглянуто перспективи FPGA як високошвидкісних та енергоефективних рішень для розроблення алгоритмів глибокого навчання, особливо у поєднанні з мовою OpenCL.

Ключові слова: штучний інтелект, алгоритми глибокого навчання, штучні нейронні мережі, програмні рішення.

Вступ

Широке застосування штучного інтелекту (ШІ) та машинного навчання (МН) для вирішення різноманітних прикладних завдань є одним із основних сучасних трендів у царині інформаційних технологій та комп'ютерних систем [1, 2]. До того ж варто зазначити, що прогрес останніх років значною мірою забезпечували поєднанням технологій глибокого навчання (ТГН) і алгоритмів на основі штучних нейронних мереж (ШНМ).

Штучна нейронна мережа – це обчислювальна система, утворена зі з'єднаних і взаємодіючих між собою простих процесорів (штучних нейронів), які під дією вхідних зважених (масштабованих) сигналів змінюють свій внутрішній стан згідно з функцією активації (збудження) і передають результат на вихід. Ваги, як і функції активації, можуть змінюватися у так званому процесі навчання за певним правилом. Але, будучи з'єднаними у доволі велику мережу із керованою взаємодією, такі окремо прості процесори разом здатні виконувати складні завдання у сфері ШІ [1, 2].

Через складність навчання впродовж тривалого часу застосовували лише прості архітектури нейронних мереж із одним прихованим шаром. Це, звісно, обмежувало можливості ШНМ – мережа потребувала нормалізації вхідних даних, а також була позбавлена гнучкості – модифікація завдання потребувала перенавчання. Ситуація докорінно змінилася у зв'язку із досягненнями у сфері ШІ, що відбулися за останнє десятиліття, і які називають терміном глибоке навчання (англ. DL – Deep Learning).

Глибоке навчання можна окреслити як статистичну техніку класифікації закономірностей із використанням багатошарових нейромереж, які наявні у великих масивах даних. Відмінною особливістю глибокого навчання є те, що комп'ютер (машина) самостійно знаходить ознаки, тобто ключові характерні риси чого-небудь, за якими найпростіше виокремити один клас об'єктів від іншого, причому ці ознаки (features) є структуровані ієрархічно: від простіших складаються все складніші та більш абстрактні форми репрезентації знань [2, 3].

Глибокі нейронні мережі (ГНМ) останніми роками набули широкого поширення, істотно витіснивши більшість інших методів у таких галузях: машинне навчання, комп'ютерний зір і опрацювання сигналів. Підставами для такого стрімкого успіху глибоких нейронних мереж є два чинники: наявність великих об'ємів даних (англ. Big Data) і здешевлення обчислювальних потужностей [4, 5]. Проте крім очевидних переваг у вигляді поліпшення і розширення функціоналу комп'ютерних систем, реалізація алгоритмів глибокого навчання на стандартних ПК спричиняє очевидні труднощі, пов'язані із нестачею ресурсів, насамперед, пам'яті й обчислювальних потужностей процесора.

Постановка завдання

Нагромадження великої кількості даних та доступність обчислювальних потужностей зумовили стрімкий розвиток у сфері програмно-апаратного забезпечення алгоритмів глибокого навчання. Варто розуміти, що оптимізація процесів чи вирішення проблем у різних випадках потребують застосування різних методик та реалізацій ШНМ. Крім цього, необхідно обирати програмні рішення, що будуть сумісними з апаратним забезпеченням, в іншому разі належно компенсовувати нестачу потужностей.

Мета цієї роботи – порівняльний аналіз наявних на ринку спеціалізованих програмних і апаратних рішень для глибокого навчання та формування рекомендацій для їх застосування під час побудови систем на основі глибоких нейронних мереж.

У роботі розглянуто такі завдання:

1. Порівняльний аналіз програмних фреймворків для глибокого навчання.
2. Аналіз особливостей імплементації алгоритмів глибокого навчання на основі графічних процесорів.
3. Огляд перспективи реалізації алгоритмів глибокого навчання на основі програмованих логічних матриць.

Порівняльний аналіз програмних фреймворків для глибокого навчання

Як вище зазначено, ГНМ потребують великої кількості даних, над якими ітеративно виконують обчислювально складні матричні операції. В класичних алгоритмах комп'ютерного зору чи опрацювання сигналів функціонал розроблявся та удосконалювався протягом тривалого часу, відтак, створено найоптимальніші рішення, зокрема на апаратному рівні, що дало змогу відчутно поліпшити швидкодію систем і мінімізувати енергоспоживання. Прикладом може бути алгоритм "butterfly" для швидкого перетворення Фур'є і відповідні апаратні прискорювачі в сигнальних (DSP) процесорах. Відтак, побудова спеціалізованих програмних імплементацій, сумісних з різними апаратними платформами – одне із першочергових завдань для впровадження ГНМ у прикладні комп'ютеризовані системи.

Сьогодні практично всі великі IT-корпорації пропонують свої програмні набори або фреймворки (від англ. – framework) для розроблення алгоритмів машинного інтелекту на основі технології глибокого навчання. Найпопулярнішими серед них є Caffe, Theano, Torch, MXNet,

Tensorflow, Neon, CNTK. Зазвичай ядро цих фреймворків складається з коду, написаного мовою C/C++, що дає змогу досягти найкращих показників щодо швидкості виконання, споживання пам'яті і використання інших апаратних ресурсів комп'ютера. Водночас також вдається максимально оптимізувати і розпаралелити матричні обчислення, що становлять основу глибокого навчання. З метою зручності інтерфейс до C/C++ ядра зазвичай обгортають однією або кількома мовами вищого рівня, як от Python, R, Matlab або Lua, хоча, загалом, найпоширенішою стала саме мова Python. Загальний огляд фреймворків наведено в табл. 1 [6].

Таблиця 1

Порівняльний аналіз програмних фреймворків глибокого навчання

Назва	Мова інтерфейсу	Розробник/партнер	Апаратна підтримка
Tensorflow	Python, Java, Go	Google LLC	CPU/GPU/кластер
Theano	Python	Université de Montréal	CPU/GPU
PyTorch	Python	Facebook, Ink.	CPU/GPU/кластер
Neon	Python	Intel Corporation	CPU/GPU
Caffe	Python	UC Berkeley	CPU/GPU
MXNet	Python, Scala, Julia, Perl. R	Amazon.com, Ink	CPU/GPU/кластер
CNTK	Python, C#, C++	Microsoft Corporation	CPU/GPU/кластер

Програмна імплементація алгоритмів глибокого навчання як звичайних універсальних процесорів (англ. Central Processing Unit – CPU) не надає бажаних результатів через великі обсяги даних. Тому всі без винятку фреймворки мають вбудовану підтримку графічних процесорів (англ. Graphical Processing Unit – GPU), що, завдяки розпаралеленню, дає значно прискорити процес навчання моделей [5].

Інколи навіть цих потужностей недостатньо і виникає потреба в розподілених обчисленнях на спеціалізованих комп'ютерних кластерах. До того ж варто зазначити, що кластери теж формують на основі графічних процесорів, поєднуючи із універсальними процесорами. Такий підхід називають HPC (англ. HPC – High-Performance Computing). Як видно з таблиці, цей функціонал підтримують лише фреймворки, розроблені великими корпораціями, оскільки він достатньо складний в імплементації. Також цей підхід має значно вищу собівартість через необхідність побудови і технічної підтримки обчислювального кластера. Тому найпопулярнішими і в академічному середовищі, і в індустрії залишаються апаратні реалізації саме на GPU. Далі розглянуто детальніше переваги, недоліки та особливості апаратних платформ.

Аналіз особливостей імплементації алгоритмів глибокого навчання на основі графічних процесорів

Як зазначено вище, типовим рішенням для імплементації алгоритмів глибокого навчання є універсальні процесори, графічні процесори і обчислювальні кластери на основі різноманітних хмарних платформ.

Графічні процесори виникли як спеціалізоване відгалуження (співпроцесори) для опрацювання відео з метою розвантажити центральний процесор. У разі опрацювання зображення алгоритмічно просто реалізувати розпаралелення обчислень на рівні даних (для кожного окремого пікселя або групи пікселів). Спершу поширення і розвитку вони набули завдяки індустрії відеоігор, 3D-дизайну і моделювання. Згодом графічні процесори почали використовувати і для інших прикладних завдань, насамперед, наукових обчислень, зокрема, для глибокого навчання. Такий підхід названо "обчисленням загального призначення на графічних процесорах" (англ. GPGPU – General-purpose computing on graphics processing units) і наразі є звичною практикою в галузі машинного інтелекту і аналітики даних [5, 6].

Основна ідея, на якій ґрунтується принцип роботи GPU, полягає в наборі великої кількості обчислювальних ядер, але значно простіших і дешевших, ніж в універсальних процесорах. Вони ділять спільну оперативну пам'ять і кеш, а також модулі управління і синхронізації. Крім того,

обчислювальні ядра об'єднані в спеціальні блоки (англ. wrap), що дає змогу порівняно легко запускати обчислення в паралельних потоках. Така архітектура забезпечує ефективне розпаралелення обчислення [5].

Ринок GPU представлений широким асортиментом продукції від різних виробників, проте найпоширенішими стали GPU від компанії Nvidia Corporation. Причиною такого успіху є їхня сумісність із спеціальною платформою CUDA, яка забезпечує зручний інтерфейс для розроблення прикладних програм. Іншим плюсом GPU від Nvidia є наявність спеціалізованого драйвера cuDNN, який містить набір інструментів для прискорення алгоритмів глибокого навчання [7, 8, 9].

Отже, до переваг графічних процесорів потрібно зарахувати високу обчислювальну продуктивність, відносну простоту написання коду і сумісність із персональними комп'ютерами. Недоліками є доволі велика вартість самої апаратури, значне енергоспоживання і відносна складність мініатюризації. Ці чинники є критичними в багатьох випадках, що зумовлює існування таких специфічних метрик як точність/ват, точність/розмір або точність/долар. Тобто під час побудови комп'ютерних систем на базі ГНМ розробникам доводиться здійснювати оптимізацію архітектури не лише з позиції точності роботи алгоритму, але й враховувати, як саме покращення точності впливає на швидкодію, споживану потужність чи масогабаритні показники системи. Необхідність оптимізації одночасно кількох параметрів істотно ускладнює процес проектування, відтак породжує необхідність пошуку інших, відмінних від графічних процесорів, апаратних рішень.

Перспективи реалізації алгоритмів глибокого навчання на основі програмованих логічних матриць

Сьогодні найпоширенішою альтернативою графічним процесорам є програмовані логічні матриці (англ. FPGA – field programmable gate array). Спрощено FPGA можна трактувати як програмно конфігуровану інтегральну мікросхему. Особливістю FPGA є те, що в їх основу покладено технологію, аналогічну оперативній пам'яті, тому сконфігурована матриця щоразу очищується після вимкнення живлення. З огляду на це FPGA зазвичай використовують у поєднанні з флеш-пам'яттю, в якій зберігається скомпільований набір інструкцій, так звана "прошивка". У разі ввімкнення живлення прошивка завантажується в пам'ять FPGA і конфігурує гнучкі логічні блоки відповідно до заданих інструкцій. Загалом FPGA складається з трьох базових наборів примітивів: програмованих логічних блоків (англ. configurable Logic Block – CLB), блоків введення-виведення (англ. Input/Output Block – IOB) та програмованих зв'язків (трасувальних ліній) (англ. programmable switch matrix – SM). Логічні блоки, своєю чергою, складаються з простіших примітивів різного рівня, наприклад, логічні комірки, елементи пам'яті, мультиплексори, логічні вентилі, таблиці відповідності (англ. Look-up table – LUT). У межах логічних блоків можна конфігурувати логіку різного рівня абстракції і складності за допомогою апаратного забезпечення. Також трасувальні лінії дають змогу сполучити будь-які логічні блоки між собою або з портами введення-виведення, а також змінити ці з'єднання у будь-який момент часу. На відміну від інших інтегральних мікросхем, таку гнучкість створюють за допомогою істотного збільшення транзисторних вентилів на кристалі.

Для значної оптимізації алгоритмів потрібна належна обізнаність в архітектурі FPGA, зокрема, у специфічних мовах програмування, таких як VHDL та Verilog, що є основним недоліком FPGA з позиції розробників ГНМ, які зазвичай не мають відповідного досвіду. Крім цього, VHDL та Verilog мають доволі довгий цикл розроблення [11–13].

Відомі альтернативи ще не досягли рівня, аби повноцінно задовольнити потреби індустрії, хоча показують значне зростання, і, як очікується, складуть повноцінну конкуренцію GPU через 3–5 років [6]. Найбільші очікування щодо цього покладають на фреймворк OpenCL, який ґрунтується на стандарті мови C (C99), що сумісна з CPU, GPU, DSP, FPGA, і який підтримують провідні компанії в галузі процесорної і напівпровідникової техніки, як от IBM, Intel, Altera, Nvidia, Xilinx, Qualcomm, Samsung та інші. Фактично, OpenCL – це безкоштовний стандартизований набір бібліотек із відкритим вихідним кодом, який ставить за мету забезпечити кросплатформеність між

різними апаратними ресурсами. Серед інших інструментів, OpenCL також пропонує бібліотеки для паралельних обчислень, що найцікавіше для розроблення алгоритмів глибокого навчання. В перспективі програмний інтерфейс мовою С, що не потребує знання апаратної архітектури, істотно спрощує можливість портування наявних алгоритмів і бібліотек на FPGA. Водночас зазначена гнучкість і високі темпи розвитку OpenCL призводять до випадків, коли значна частина функціоналу, що підтримується для однієї з платформ, не сумісна з іншою, і навпаки.

До інших недоліків FPGA потрібно зарахувати обмеження, пов'язані зі швидкістю обміну даними з ПК, а також значно меншу пам'ять порівняно з GPU. Цю проблему вдається вирішити, створюючи FPGA кластери. Зокрема, віднедавна існують хмарні сервіси від компаній Amazon та Google, які дають змогу виконувати обчислення на віддалених FPGA кластерах [11, 12]. Також час компіляції й перекомпіляції становить десятки хвилин, що значно довше, ніж для CPU/GPU. Це може бути критичним у експериментальному підборі оптимальної архітектури ГНМ, проте на практиці під час цих експериментів вдається використовувати попередньо скомпільовані базові блоки ГНМ, що істотно пришвидшує процес.

Основними перевагами FPGA є значно менше споживання, а також розміри, порівняно з GPU. Іншою перевагою є можливість будувати гнучку архітектуру під конкретну модель ГНМ, на відміну від звичайних процесорів, чия архітектура фіксована і потребує адаптації моделі. Крім того, FPGA дають змогу легко переносити алгоритми з кластерів на рівень вбудованих систем, або ж робити тверді копії у разі масового виробництва [13, 14].

Також потрібно зазначити, що окрім FPGA існує ще низка інших спеціалізованих апаратних рішень, таких як Intel Movidius Neural Compute Stick, Intel Nervana NNP, Intel Loihi, Orange Pi AI Stick 2801 (на базі Lightspeeur 2801S), проте вони також поки не є настільки поширені, як графічні процесори.

Висновки

1. Впродовж останніх років спостерігається стрімкий прогрес у галузі штучного інтелекту і машинного навчання, зумовлений виникненням і розвитком технологій глибокого навчання. Завдяки цьому до основних драйверів цього процесу, насамперед потрібно зарахувати здешевлення і доступність комп'ютерних ресурсів, а саме: зростання обчислювальних потужностей завдяки графічним процесорам і хмарним сервісам, спеціалізованим файловим системам для роботи з великими даними (типу HDFS). Важливу роль, також, відіграла поява великої кількості даних завдяки оцифруванню наявної інформації, інформатизації робочих процесів у приватному і державному секторі, появі Інтернету речей, смартфонів, портативних пристроїв. До того ж власне алгоритми глибокого навчання і надалі залишаються надзвичайно ресурсозатратними і з боку кількості обчислень, і з боку споживання пам'яті, що породжує потребу в побудові спеціалізованих програмно-апаратних рішень.

2. Спеціалізоване програмне забезпечення для глибокого навчання переважно подане у вигляді окремих програмних наборів або фреймворків, ядро яких, зазвичай, написане мовами C/C++, а високорівнений інтерфес представлений кількома скриптовими мовами (Python, R, Matlab, або Lua). Сьогодні більшість апаратних платформ для глибокого навчання побудовано на основі графічних процесорів (GPU). Беззаперечним лідером на ринку GPU є компанія Nvidia. Її провідні позиції зумовлені, значною мірою, наявністю спеціалізованого програмного забезпечення і драйверів, зокрема CUDA та cuDNN.

3. Серед альтернатив графічним процесорам найперспективнішими є рішення на основі програмованих логічних матриць (FPGA). Основними їх перевагами, порівняно з графічними процесорами, є менша споживана потужність і розміри. Іншою перевагою є можливість будувати гнучку архітектуру для конкретного алгоритму глибокого навчання. До недоліків варто зарахувати необхідність використання мов програмування VHDL та Verilog, які характеризуються доволі

довгим циклом розроблення та потребують додаткових знань, пов'язаних із внутрішньою будовою та особливостями організації обчислень на FPGA.

Список літератури

1. Christopher M. Bishop. Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag Berlin, Heidelberg, 2006.

2. Ian Goodfellow, Yoshua Bengio, Aaron Courville, Deep Learning, The MIT Press, 2016.

3. L. Deng and D. Yu. Deep Learning: Methods and Applications. Foundations and Trends in Signal Processing, 2013, vol. 7, nos. 3–4, pp. 197–387.

4. Mostapha Zbakh, Mohammed Essaaidi, Pierre Manneback, Chunming Rong, Cloud Computing and Big Data: Technologies, Applications and Security, Springer International Publishing, 2019.

5. Gerassimos Barlas, Multicore and GPU Programming: An Integrated Approach, Morgan Kaufmann Publishers Inc., San Francisco, CA, 2014.

6. Seonwoo Min, Byunghan Lee, Sungroh Yoon; Deep learning in bioinformatics, Briefings in Bioinformatics, Volume 18, Issue 5, 1 September 2017, pp. 851–869.

7. NVIDIA GPU Computing. https://www.nvidia.com/object/doc_gpu_compute.html

8. CUDA Toolkit Documentation. https://docs.nvidia.com/cuda/

9. cuDNN Developer Guide. https://docs.nvidia.com/deeplearning/ sdk/cudnn-developer-guide/index.html

10. Amazon EC2 F1 Instances. https://aws.amazon.com/ec2/ instance-types/f1/

11. Cloud TPU documentation. https://cloud.google.com/tpu/docs/

12. Accelerating DNNs with Xilinx Alveo Accelerator Cards. https://www.xilinx.com/support/documentation/ white_papers/wp504-accel-dnns.pdf

13. An OpenCLTM Deep Learning Accelerator on Arria 10. https://arxiv.org/pdf/1701.03534.pdf.

COMPARATIVE ANALYSIS OF THE SPECIALIZED SOFTWARE AND HARDWARE FOR DEEP LEARNING ALGORITHMS

Y. Khoma[1], A. Bench[2]

Lviv Polytechnic National University,
[1] Department of Information and Measurement Technologies,
[2] Department of Theoretical Radio Engineering and Radio Measurement

The automated translation, speech recognition and synthesis, object detection as well as emotion recognition are well known complex tasks that modern smartphone can solve. It became possible with intensive usage of algorithms of Artificial Intelligence and Machine Learning. Most popular now are implementations of deep neural networks and deep learning algorithms. Such algorithms are widely used in all verticals and need hardware accelerators as well as deep cooperation between both software and hardware parts. The mentioned task became very actual during embedding of cloud-based algorithms into systems with limited computing capabilities, small physical size, and extremely low power consumption. The aim of this paper is to compare existing software and hardware solutions dedicated to the development of artificial neural networks and deep learning applications. The paper is focused on three topics related to deep learning software frameworks, specialized GPU-based hardware, and prospects of deep learning acceleration using FPGA. The most popular software frameworks, such as Caffe, Theano, Torch, MXNet, Tensorflow, Neon, CNTK have been compared and analyzed in the paper. Advantages of GPU solutions based on CUDA and cuDNN frameworks have been described. Prospects of FPGA as high-speed and power-efficient solutions for deep learning algorithm design, especially in terms of combination with OpenCL language have been discussed in the paper.

Key words: artificial intelligence, deep learning algorithms, artificial neural networks, software solutions.