УДК 004.3

С.Д. Погорілий, д-р техн. наук, проф.,
М.В. Чечула, студент магістратури,
Київський національний університет імені Тараса Шевченка, м.Київ, Україна
sdp77@i.ua
nikofomi@gmail.com

Застосування технології GPGPU при роботі з BigData

Досліджено методи роботи гетерогенних програмно-апаратних систем з бібліотекою TensorFlow. Для аналізу ефективності роботи бібліотеки зі стандартною моделлю об'єктів було обрано згорточну нейромережу і виконано дослідження її роботи з використанням технологій OpenCV та Nvidia CUDA. Побудовано модель об'єкту шляхом її поступового навчання. Виконано порівняння ефективності роботи бібліотеки на гетерогенних обчислювальних платформах з графічними відеоадаптерами.

Ключові слова: технологія GPGPU, технологія CPGPU, бібліотека TensorFlow, мова програмування Python, проблема BigData, графічний процесор GPU, згорточна нейронна мережа, фреймворк Hadoop, функції API TensorFlow.

DOI: 10.31474/1996-1588-2017-2-25-98-102

Вступ

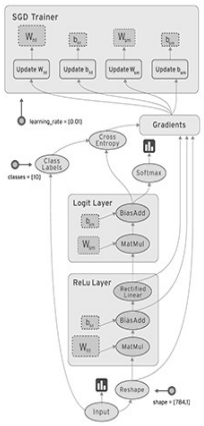
Тема BigData та відкритості даних набуває все більшого поширення у світі та, зокрема, в Україні. Однак проблема обробки та систематизації цих даних досі є актуальною. Наразі існує декілька проектів, які надають можливості використання BigData у наукових цілях та комерційній діяльності. Серед них Hadoop, Bluemix, Spark, TensorFlow. Всі вони мають різні ступені інтеграції з розподіленими системами та в більшості випадків орієнтовані на великі обчислювальні потужності. Звідси постає проблема використання у досліджені алгоритмів з використанням невеликих обсягів обчислювальної потужності. Саме тому в цій статті буде розглянуто та проаналізовано інструмент Google – TensorFlow.

Метою роботи є: дослідження роботи бібліотеки TensorFlow з малопотужними гетерогенними обчислювальними потужностями; створення навченої нейромережі для обробки графічних даних та окреслення основних нюансів роботи з бібліотекою; порівняння ефективності роботи з TensorFlow з використанням GPU та CPU як під час навчання нової нейромережі так і під час використання готової [1].

Призначення бібліотеки TensorFlow

TensorFlow є відкритим проектом, метою якого є надання зручних інструментів обробки BigData на більшості існуючих платформ. Проект було обрано у зв'язку зі значною гнучкістю до обчислювальних потужностей, можливістю інтегрування у інші системи обробки BigData, значними темпами розвитку, підтримкою зі сторони розробників та прозорістю в плані використання і досліджень незважаючи на початкову стадію розробки. Бібліотека також має можливість інтегрування з Hadoop, що значно розширює межі використання [1 - 3].

Основою бібліотеки TensorFlow є граф обчислень, де кожна вершина це або математичний оператор або стала величина, а кожний перехід являє собою потік даних у вигляді багатовимірного тензору. Звідси і назва продукту. Таким чином утворюється направлений ациклічний граф [4].



(а)

(б)
Define – ініціалізація графу з вхідними параметрами нейромережі.
Compilation – компіляція графу (на цьому кроці йде навчання)
Execution – виконання розпізнавання пристрою

Рисунок 1 — Детальна (а) та загальна (б) блок схеми роботи TensorFlow

На вхід програми для побудови графу подається модель яка зазвичай являє собою один з видів нейронних мереж, однак на вхід може подаватися будь яка модель яку можна представити у вигляді графу обчислень. На цей час стандартним пакетом підтримуються такі класи нейронних мереж як глибинна нейромережа переконань, радіально базисна нейромережа, згорточна нейромережа, рекурсивна нейромережа [5].

Основною перевагою бібліотеки TensorFlow серед інших подібних бібліотек є гнучкість у виборі обчислювальної потужності. TensorFlow надає доступ до API на таких мовах програмування як Python, C++, Java та Go. Але, в зв'язку з тим, що велика частина ядра бібліотеки TensorFlow написана саме на Python, ефективність використання API саме для цієї мови програмування буде найбільшою [1,3].

Апаратна платформа та методика дослідження

Далі розглянуто використання потужностей Nvidia GeForce 970M покоління Maxwell з 1280 обчислювальними ядрами та оперативною пам'яттю DDR5 розміром 3Гб та IntelCore i7 відповідно з 8 ядрами в режимі Hyper Threading та DDR4 на 16 Гб з використанням мови програмування Python 3 [6].

При роботі TensorFlow з GPU існують досить суттєві обмеження на CUDA CC, відтак серія GeForce повністю підтримує бібліотеку тільки з 7-8 покоління, тому для обчислень на GPU треба заздалегідь обирати останні моделі Nvidia. Це обумовлено експоненційним зростанням продуктивності нових GPU та швидким старінням попередніх [7-9].

Одним з останніх нововведень TensorFlow було додання API для детектування декількох об'єктів одночасно, ця функція може бути використана в реальному часі з вже підготовленими навченими моделями від Google. Оскільки BigData являє собою нескінченно зростаючий обсяг даних в реальному часі та найбільш вагомою частиною є саме графічна інформація різного роду, то використання саме цього API представляє найбільший інтерес. Проте, треба зазначити, що кількість моделей та точність доволі обмежена на даний момент, що, можливо, буде враховано в майбутніх версіях TensorFlow [3].

Для дослідження роботи алгоритму був задіяний потік відео з веб-камери в розширенні HD з допомогою бібліотеки OpenCV для Python, з допомогою якої відео трансформувалося в послідовність зображень, які подавалися до згорточної нейронної мережі в реальному часі. Джерелом даних може бути відео чи архів зображень.

Нейронну мережу було створено з трьома прихованими шарами з метою більшої узагальнюючої здатності для обробки відео з розширенням 1280x768 з частотою 30 кадрів за секунду.

Далі, як ілюстрацію, наведено приклад роботи програми під час розпізнавання декількох предметів одночасно в реальному часі.

Як бачимо, якість розпізнавання по заздалегідь навченим об'єктам є доволі високою, вірогідність розпізнавання об'єктів залишалась більше 50 відсотків навіть при поганій якості освітлення та під гострим кутом нахилу, що свідчить про високу точність нейронної мережі, а саме: дуже довге навчання та велику вибірку навчальних матеріалів.

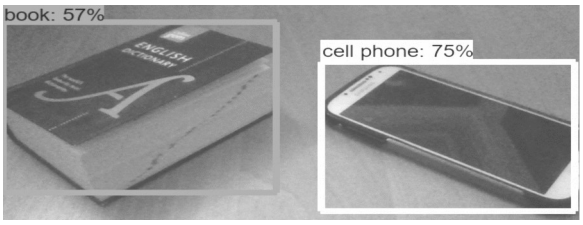


Рисунок 2 — Розпізнавання відео з веб-камери за допомогою стандартної моделі

Додатково було зроблено порівняння завантаження обчислювального пристрою в залежності від його типу. В даному випадку в якості CPU використовувався 8 потоковий IntelCore i7 6 покоління Skylake.

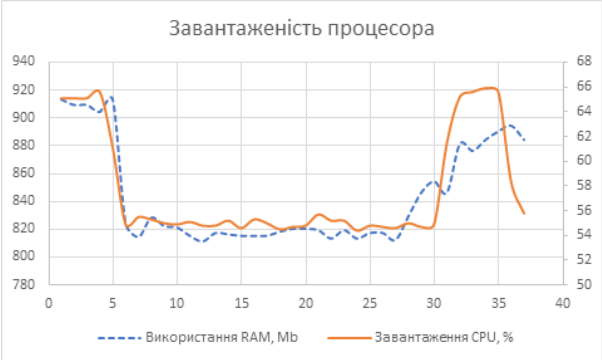


Рисунок 3 — Завантаженість процесора під час розпізнавання з використанням CPU

Можна помітити, що графік завантаженості CPU негладкий, зміна розподілу ресурсів відбувається в моменти різкого збільшення об'єктів в полі зору камери. Система TensorFlow в більшості випадків намагається автоматично оптимізувати завантаженість під потреби користувача, щоб не завдавати шкоду системним процесам. Однак незважаючи на те, що в цілому завантаження процесора не було повним, подекуди відбувалися затримки в обробці зображення.

Натомість графічний адаптер демонструє вельми ефективну роботу — використовуючи лише 40 відсотків потужності дозволяє досягти високої продуктивності розпізнавання та набагато більшої стійкості до зміни кількості об'єктів у полі зору. Однак, слід зазначити, що у зв'язку з тим, що система гетерогенна, збільшується кількість інформації, яку потрібно постійно передавати між пристроями для їх комунікації, тому завантаження CPU теж присутнє, а кількість оперативної пам'яті, необхідної для розрахунків зросла в 2 рази.

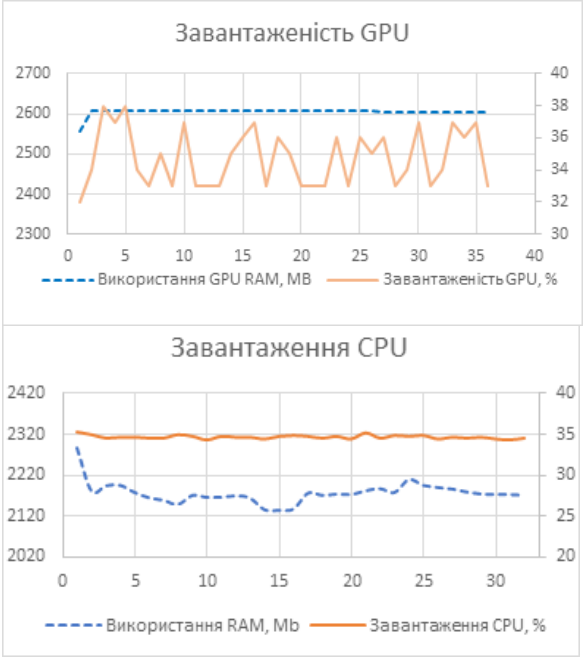


Рисунок 4 — Завантаженість CPU та GPU під час розпізнавання з використанням графічного адаптеру

Наступним кроком у досліджені роботи TensorFlow було навчання мережі для нової моделі з необроблених даних та вивчення можливості мережі до узагальнення. Було обрано довільний об'єкт, якого немає в стандартній моделі.

Слід зазначити, що наразі процес створення власної моделі потребує попередньої обробки всіх даних, тож для цього було створено декілька окремих програм на Python 3.

Процес створення моделі TensorFlow поділяється на такі основні кроки:

1. Обрання зображень об'єкту якомога різного характеру від 100 одиниць.
2. Автоматичне чи напівавтоматичне виділення потрібного фрагменту —об'єкту на зображенні для навчання та збереження цих даних у форматі XML. Чим більша точність виділення потрібної ділянки на зображенні – тим краще подальше узагальнення мережі.
3. Створення так званого TensorFlow Record як вхідних даних для подальшого навчання у форматі CSV.
4. Ініціалізація графу з допомогою бібліотеки TensorFlow.
5. Безпосередньо навчання моделі з заданими умовами зупинки.
6. Інтеграція моделі в алгоритми аналізу даних.
7. Для кожного етапу потрібно створювати сценарій, наприклад, на мові Python з параметрами вхідних даних та бажаного результату.

Процес навчання триває багато кроків, поки не буде досягнуто необхідної точності. На кожному кроці виконується тестування та обраховуються втрати (кількість помилок) з використанням методу зворотного розповсюдження помилки (чим менше втрати — тим більша точність). Збільшення точності йде логарифмічно, тож для зменшення часових витрат в більшості випадків доцільно припинити дію програми після проходження декількох годин. Ідеальна величина втрат становить менше одиниці.

Рисунок 5 — Кількість неправильно розпізнаних об'єктів по відношенню до кількості кроків тренування

На рисунку 6 можна побачити, що завантаженість GPU нерівномірна у зв'язку з тим, що також йде постійний обмін між GPU, CPU та SSD, також можна помітити що завантаженість пам'яті графічного адаптеру знаходиться на своєму піковому значенні, що означає, що NVIDIA вимушена постійно використовувати збірник сміття для вивільнення нових ресурсів що часто призводило до передвчасної зупинки алгоритму. Ця проблема може бути вирішена шляхом збільшення обчислювальних потужностей.

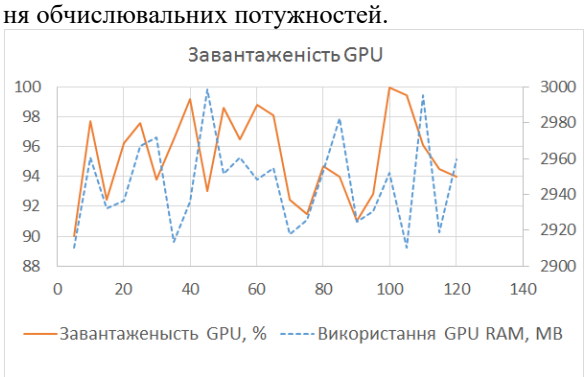


Рисунок 6 — Завантаження GPU під час навчання

При порівнянні аналогічних показників для CPU також маємо повну завантаженість системи, проте швидкість навчання в цьому випадку була у 8 разів меншою. Однак треба зазначити, що результат навіть повільного навчання буде еквівалентний, а зазначена в описі проекту мультиплатформеність виконується в повному обсязі.

Після навчання мережі було проведено серію експериментів з тестування на об'єктах, які

раніше мережа не бачила ані під час тренування ані під час попередніх тестувань. В цілому модель показала приблизно такі самі результати, що й стандартна. Але варто зазначити, що на побудову моделі для 1000 об'єктів та використання великої кількості зображень потрібно задіяти в декілька разів більше ресурсів.

Рисунок 7 — Порівняння часу тренування для GPU та CPU у годинах

Здатність до узагальнення нової нейромережі залишилася доволі високою, однак при більш детальному дослідженні поведінки мережі можна було побачити, що деколи мережа починає неправильно оцінювати об'єкти, особливо якщо шуканих об'єктів взагалі немає. Це виходить з того факту, що кожного разу шукається найбільш достовірний об'єкт з відомих. То ж зі збільшенням кількості відомих об'єктів проблема поступово нівелюється.

Висновки

Бібліотека TensorFlow побудована таким чином, щоб зберігати відносно високу продуктивність обчислень навіть при використанні малопродуктивних апаратних платформ, тому вона є основним фаворитом для роботи з BigData, бо дозволяє розподілити навантаження на гетерогенні обчислювальні потужності. Незважаючи на доволі ранній період розробки можливість інтеграції інструменту з Hadoop дозволяє створювати масштабовані обчислювальні мережі доволі швидко та зручно.

Показано, що процес навчання потребує підготовки даних за допомогою інших бібліотек та процес навчання може займати завелику кількість часу

В результаті проведеного аналізу роботи бібліотеки TensorFlow було доведено доцільність та ефективність роботи з використанням сучасного графічного адаптеру. Тренування нейромереж виконувалися в 7-8 разів швидше на GPU але з врахуванням затримок та операцій на комунікацію обрахунки лише на CPU використовували в 2-3 рази менше оперативної пам'яті.

Список літератури

1. Офіційний сайт TensorFlow [Електронний ресурс] : (опис бібліотеки). — Режим доступу: https://www.tensorflow.org/
2. Офіційний сайт Hadoop [Електронний ресурс] : (опис фреймворку) — Режим доступу: http://hadoop.apache.org/
3. Офіційний сайт TensorFlow [Електронний ресурс] : (опис API) — Режим доступу: https://www.tensorflow.org/api_docs/
4. Погорілий С.Д. Програмне конструювання. — Київ: Київський національний університет імені Тараса Шевченка. — 2007. — 438 с.
5. Офіційний сайт Інституту Азімова [Електронний ресурс] : (типи нейронних мереж) — Режим доступу: http://www.asimovinstitute.org/neural-network-zoo/
6. R.I. Levchenko, O.O. Sudakov, S.D. Pogorilyy. DDCI: Simple Dynamic Semiautomatic Parallelizing for Heterogeneous Multicomputer Systems. Proceedings of the 5th IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications. — Італія : Ренд, 2009. — 21 с.
7. Офіційний сайт NVidia [Електронний ресурс] : (опис CUDA neuron network) — Режим доступу: https://developer.nvidia.com/cudnn
8. Офіційний сайт NVidia [Електронний ресурс] : (глибинне навчання на CUDA) — Режим доступу: https://developer.nvidia.com/deep-learning
9. Офіційний сайт NVidia [Електронний ресурс] : (перелік продуктів, які підтримують CUDA Compute Compatibility) — Режим доступу: https://developer.nvidia.com/cuda-gpus

References

1. TensorFlow official site (library description), available at: Режим доступу: https://www.tensorflow.org/
2. Hadoop official site, available at: http://hadoop.apache.org/
3. TensorFlow official site (API description), available at: https://www.tensorflow.org/api_docs/

4. Pogorilyy, S.D. (2007), Software design [Prohramne konstruyuvannya], Kyiv, Kyiv National Taras Shevchenko University, 2007, 438 p.
5. The official site of the Azimov Institute, available at: http://www.asimovinstitute.org/neural-network-zoo/
6. Levchenko R.I., Sudakov O.O. and Pogorilyy S.D. (2009), DDCI: Simple Dynamic Semiautomatic Parallelizing for Heterogeneous Multicomputer Systems, Proceedings of the 5th IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, Italy, Rend, 2009, 21 p.
7. NVidia official site(CUDA neuron network), available at: https://developer.nvidia.com/cudnn.
8. NVidia official site (CUDA deep learning), available at: https://developer.nvidia.com/deep-learning
9. NVidia official site (CUDA Compute Compatibility), available at: https://developer.nvidia.com/cuda-gpus

Надійшла до редакції 19.09.2017

S.D. POGORILYY, M.V. CHECHULA
[1]Taras Shevchenko National University of Kyiv (Ukraine)

GPGPU TECHNOLOGY APPLICATION IN WORKING WITH BIGDATA

The article is devoted to the discovering of the TensorFlow library as well as to analyze its work on various platforms. As a research platform, a system equipped with 8 threads InteCore i7 CPU of the SkyLake generation and NVidia GeForce-series GPU of Maxwell generation with 1280 cores and 3 Gb GDDR5 memory was chosen. An analysis was made on the TensorFlow work with pattern recognition based on a convolutional neural network with three hidden layers of neurons. Two alternative approaches were chosen for the analysis. First one was about using the completed and trained neural network provided by Google. The second was about the training of a custom neural network based on a new raw data. In the first case to enhance the analysis a 30 FPS HD camera was used. This camera in a real time transmitted data to the new neural network. As a result of performance testing in the recognition mode, it was shown that the efficiency of work on different types of processing power was almost the same and the hardware and software platforms perfectly performed. In the second case, an analysis was made on the method of constructing a new neural network based on the object photos. It was shown that for the custom neuron system construction it is necessary to use a series of transformations of the input data before sending them to train the network. These series of transformations were presented in the form of sequential actions to convert raw images to so-called TensorFlow Record in CSV format. As a result of discovering the performance of the platform in the training mode, it was noted that the speed of training in the case of GPGPU technology was 8 times better than in the CPU case. In general, as a result of the TensorFlow library discovering the high effectiveness of its use in a combination with the modern graphics adapter was noted. Besides, a number of nuances were observed while exploring it such as the inconvenience of training with raw data and sudden malfunctioning that happen during training on the GPU cased by the lack of RAM.

Key words. GPGPU technology, CPGPU technology, TensorFlow library, Python programming language, BigData problem, GPU, convolutional neural network, Hadoop framework, TensorFlow API.