

УДК 004.021 (004.89)

Леощенко С.Д.¹, Олійник А.О.², Субботін С.О.³

¹асп. НУ «Запорізька політехніка»

²канд. техн. наук, доц. НУ «Запорізька політехніка»

³д-р техн. наук, проф. НУ «Запорізька політехніка»

ВИКОРИСТАННЯ ХЕШУВАННЯ ДЛЯ ПРИСКОРЕННЯ НЕЙРОННИХ МЕРЕЖ

В останні роки величезна кількість сил було спрямоване на прискорення штучних нейронних мереж (ШНМ) – як для навчання, так і для отримання висновків її роботи – так були представлені апаратні платформи, на яких можна виконувати обчислення: Google розробили Tensor Processing Unit; пізніше Nvidia представила V100 Graphics Processing Unit, описуючи його, як чіп, спеціально розроблений для навчання і використання ШНМ, а також для інших високопродуктивних обчислювальних потреб.

Однак, вже в 2019 році автори Бейді Чен, Тарун Медіні і Аншумалі Шрівастава з Університету Райса, у своїй роботі довели, що, можливо, спеціальне обладнання, що розробляється для роботи ШНМ, оптимізується не під той алгоритм. Дослідники зрозуміли, що активація багатьох нейронів в певному шарі ШНМ виявляється занадто малою, і не впливає на вихідне значення, що підраховується наступними шарами. Тому, якщо знати, що це за нейрони, їх можна просто ігнорувати.

Так найпростішим способом дізнатися, які нейрони в шарі не активуються, буде спочатку зробити всі операції матричного перемноження для цього шару. Але дослідники зрозуміли, що насправді можна визначитися з цим більш ефективним способом, якщо подивитися на проблему під іншим кутом і запропонували підходити до цього питання, як до вирішення завдання пошуку. Тобто, замість того, щоб обчислювати множення матриць і дивитися, які нейрони активувалися для заданих вхідних даних, можна просто подивитися, які це нейрони в базі даних. Перевага такого підходу в задачі полягає в тому, що можна використовувати узагальнену стратегію, давно вже вдосконалену фахівцями з інформатики з метою прискорення пошуку даних в базі: хешування.

Хешування дозволяє швидко перевірити, чи є потрібне значення в таблиці бази даних, без необхідності послідовно проходити кожну її рядок. Використовуючи хеш, легко розраховується через застосування хеш-функції до потрібної величини, який вказує, де це значення має зберігатися в базі. Потім можна перевірити тільки одне це місце, щоб дізнатися, чи зберігається там це значення.

Але що якщо просунутися у використанні хешування від використання ШНМ до її синтезу. Так, з огляду на широке використання хеш-функцій при

кодуванні інформації в імовірнісних структурах даних звернемося до них. Bloom Filter – це бітовий масив з M бітів, ініціалізованих в 0. Щоб додати елемент, він обробляється k хеш-функціям, щоб отримати k позицій масиву, і встановить біти в цих позиціях рівними 1. Щоб запросити елемент, він передається в k хеш-функцій, щоб отримати k позицію масиву. Тоді логічним буде використання такої структури даних при пересилці інформації про ШНМ між ядрами паралельної системи. MinHash – це метод швидкої оцінки того, наскільки схожі два набори. Ця схема була винайдена Andrei Broder (1997) і спочатку використовувалася в пошуковій системі AltaVista для виявлення дублікатів веб-сторінок і виключення їх з результатів пошуку. Він також був застосований у великомасштабних завданнях кластеризації, таких як кластеризація документів за подібністю їх наборів слів. Тоді, MinHash використаємо при послідовному виконанні нейроеволюційного синтезу для кодування генетичної інформації про особини, для оцінки прискорення на етапі селекції. Нарешті, Count–min sketch – це імовірнісна структура даних, яка представляє собою таблицю частот подій в потоці даних. Вона використовує хеш-функції для зіставлення подій з частотами, але на відміну від хеш-таблиці використовує тільки сублінійний простір, за рахунок перерахунку деяких подій через колізій. Саме тому, Count-Min Sketch також будемо використовувати для кодування генетичної інформації про особини під час нейроеволюційного синтезу.

Безумовно, такий підхід має бути випробуваний і лише на основі результатів тестування можна буде говорити про його імплементацію. Але вже можна сказати про те, що це безумовно підкреслює небезпеку захоплення певного типу апаратним забезпеченням у випадках, коли існує можливість появи нового і кращого алгоритму для роботи ШНМ.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SLIDE: In Defense of Smart Algorithms over Hardware Acceleration for Large-Scale Deep Learning Systems / B. Chen, T. Medini, J. Farwell, S. Gobriel, C. Tai, A. Shrivastava // arXiv [Electronic resource]. – Access mode: <https://arxiv.org/abs/1903.03129>
2. Gakhov A. Probabilistic Data Structures and Algorithms for Big Data Applications / A. Gakhov. – Madison : Books on Demand, 2019. – 220 p.