DOI: 10.36910/6775-2524-0560-2019-37-7
УДК: 004.852
М. М. Поліщук, С. М. Костючко, М. О. Христинець
Луцький національний технічний університет

ПОРІВНЯННЯ МЕТОДІВ ОПТИМІЗАЦІЇ НЕЙРОННИХ МЕРЕЖ НА ПРИКЛАДІ ЗАДАЧІ КЛАСИФІКАЦІЇ ЗОБРАЖЕНЬ

М. М. Поліщук, С. М. Костючко, М. О. Христинець. Порівняння методів оптимізації нейронних мереж на прикладі задачі класифікації зображень. У статті проаналізовано існуючі методи оптимізації та типи розподілених обчислень для тренування нейронних мереж. На основі проведених експериментів досліджено доцільність використання даних методів для різних типів даних та архітектури нейронних мереж.
Ключові слова: нейронні мережі, стохастичний градієнтний спуск, методи оптимізації, тренування нейронних мереж, розподілені обчислення, асинхронний сервер

Н. Н. Полищук, С. М. Костючко, М. А. Христинец. Сравнение методов оптимизации нейронных сетей на примере задачи классификации изображений. В статье проанализированы существующие методы оптимизации и типы распределенных вычислений для тренировки нейронных сетей. На основе проведенных экспериментов исследованы целесообразность использования данных методов для различных типов данных и архитектуры нейронных сетей.
Ключевые слова: нейронные сети, стохастический градиентный спуск, методы оптимизации, тренировки нейронных сетей, распределенные вычисления, асинхронный сервер

M. Polishchuk, S. Kostyuchko, M. Khrystynets Comparison of neural network optimization methods using the image classification problem. The article analyzes existing optimization methods and types of distributed computing for neural network training. On the basis of the conducted experiments, it was investigated the feasibility of using these methods for different types of data and architecture of neural networks.
Keywords: neural networks, stochastic gradient descent, optimization methods, training of neural networks, distributed computing, asynchronous server

Вступ. Нові алгоритми, механізм або винаходи мають характерні риси, що відрізняють їх від застарілих або неперспективних аналогів. Для нейронних мереж також можна виділити основні особливості, які відмінні від традиційних алгоритмів розв'язування практичних і теоретичних завдань. Однією з головних особливостей нейромереж є те, що вони навчаються. Існують різні методи навчання (з учителем, без учителя, змішані), але всі вони засновані на вивченні прикладів з заданої бази даних. Здатністю до навчання нейронні мережі і відрізняються від традиційних алгоритмів, у яких є чіткий порядок обчислень, наявність формул та інше.

Ще однією особливістю нейронних мереж є можливість роботи з різними джерелами даних. Мережа може аналізувати дані різного походження в ході вирішення однієї задачі і на їх основі робити відповідні висновки. Крім того, якщо вхідні дані мають сторонню складову (так звані шуми), то в процесі навчання нейронна мережа вчиться відсіювати ці шуми і залишає тільки необхідне. Порівнюючи з традиційними алгоритмами розв'язання деяких завдань, можна сказати, що у випадку останніх, виникнення сторонніх даних в обчисленнях може привести до помилки всього алгоритму і як наслідок – до невірного кінцевого результате. При правильному навчанні нейромережі можна виявити ще одну важливу особливість, що при роботі з великими обсягами різноманітної інформації нейронна мережа може одночасно розв'язувати кілька завдань.

Постановка проблеми. Частота застосування технології штучних нейронних мереж в різних сферах життя суспільства і в науці, безсумнівно, зростає. Про це свідчать нововведення, які постійно впроваджуються в побут людей. Звичайно, не можна поки сказати, що нейронні мережі оточують нас всюди, але і зворотного також стверджувати не можна. Технології з застосуванням нейронних мереж активно використовуються в області інформаційних технологій. Нейронні мережі також активно використовуються для реалізації технологій розумного транспорту (Tesla, Uber, та інші). Обробкою отриманих даних займається спеціальний алгоритм, який з кожним днем удосконалюється, і, можливо, незабаром дана технологія буде повністю впроваджена в життя [1]. У сфері економіки нейронні мережі найчастіше використовуються для прогнозування цін, курсів валют, а також оптимізації торгівлі на ринку. Найбільш популярна бібліотека для нейромережевого аналізу даних Brain Marker [2], основна мета якої – розв'язання нетрадиційних задач, таких як біржові передбачення, моделювання різних ринкових ситуацій. В його основі лежить нейронна мережа (мережа Хопфілда), яка навчається на багатьох прикладах.

Популярність даної технології, безсумнівно, росте, як і кількість даних які потрібно обробляти, тому оптимізація процесу тренування глибоких нейронних мереж є дуже важливою для зменшення витрат і економії часу.

Мета і завдання дослідження. Проаналізувати та порівняти існуючі методи оптимізації нейронних мереж на різних типах нейромереж та даних, провести експерименти з розподіленим тренуванням нейромереж для дослідження ефективності методів тренування для одного вузла та для тренування на кластерах.

Аналіз попередніх досліджень. В алгоритмах машинного навчання зазвичай доводиться виконувати багато чисельних розрахунків. Як правило, мова йде про застосування методів, які ітеративно уточнюють розв'язки систем рівнянь, а не шукають його аналітично по формулі. Типові операції для таких розрахунків - оптимізація (знаходження мінімума або максимума деякої функції) і розв'язання системи лінійних рівнянь. Але навіть саме обчислення математичної функції за допомогою комп'ютера може виявитися важким завданням, якщо у вираз функції містить дійсні числа, які не можна точно задати в пам'яті скінченного розміру.

Фундаментальна складність виконання безперервних математичних операцій на цифровому комп'ютері полягає в тому, як зберегти нескінченно багато дійсних чисел за допомогою скінченного числа комбінацій бітів. Це означає, що майже для всіх дійсних чисел проводиться деяка апроксимація і, отже, виникає помилка округлення. Це може привести до того, що теоретично правильний алгоритм, при проектуванні якого не була передбачена мінімізація накопичення помилок округлення, на практиці не працює, тому важливо обрати оптимальний метод оптимізації, який найкраще підходить для даної задачі машинного навчання.

Алгоритми оптимізації, які використовуються для навчання глибоких нейронних мереж, відрізняються від традиційних алгоритмів оптимізації в декількох аспектах. Машинне навчання зазвичай працює не напряму. У більшості ситуацій оцінюється деяка міра якості процесу навчання *P*, яка розраховується на основі тестового набору. Тому оптимізація параметру *P* відбувається не на пряму. Під час тренування оцінюється інша функція вартості J(θ) і процес відслідковує при цьому покращення міри *P*.

Це дуже відрізняється від чистої оптимізації, де мінімізація функції вартості *J* і є кінцевою метою. Крім того, алгоритми оптимізації для навчання глибоких моделей зазвичай включають спеціалізації для конкретної структури цільових функцій.

Типову функцію вартості можна представити у вигляді середнього по тренувальному набору:

$$J(\theta) = E_{(x,y)\sim P_{data}} L(f(x;\theta), y) \quad (1)$$

де *L* – функція втрат розрахована на одному прикладі, $f(x;\theta)$ - передбачений результат для входу *x*, а P_{data} - емпіричний розподіл. У разі навчання з учителем *y* - асоційована з входом мітка.

Рівняння (1) визначає цільову функцію щодо навчального набору.

Найбільш розповсюджені методи оптимізації нейромереж для пришвидшення процесу тренування:

- оптимізація градієнтним методом;
- мінімізація емпіричного ризику;
- пакетні та міні-пакетні алгоритми оптимізації;
- стохастичний градієнтний спуск;
- імпульсний метод оптимізації;
- метод Нестерова;
- стратегії ініціалізації параметрів нейронної мережі;
- алгоритми оптимізації з адаптивною швидкістю навчання;
- наближені методи оптимізації другого порядку.

Виклад основного матеріалу.

Порівняння ефективності алгоритмів оптимізації

Для перевірки ефективності оптимізаторів було використано згорткову нейронну мережу для класифікації зображень на наборі даних CIFAR-10. Перевірено ефективність семи популярних алгоритми оптимізації на основі градієнтного спуску, що використовуються для навчання глибоких нейронних мереж:

– SGD;
– SGD з методом моментів;
– SGD з використанням методу Нестерова;
– RMSProp;
– Adam;
– Adagrad;
– Nadam.

Для тренування нейромереж було використано "Cifar-10" - датасет з 60 тис. зображень розміром 32 х 32 пікселя розподілених на 10 категорій. 50 тис. зображень використовувались для тренування і 10 тис. для тестування моделей.

Всі, описані вище, алгоритми оптимізації були використані під час тренування згорткової нейронної мережі, яка навчались з одинаковим набором гіперпараметрів. Тренування відбувалось впродовж 150 епох.

Нейронна мережа для перевірки ефективності оптимізаторів складається з трьох згорткових шарів з розміром фільтра 3х3 та активаціями ReLu, також після кожного згорткового шару додано MaxPooling для зменшення розмірності після операції згортки. Перед вихідним шаром нейромережі було додано Dropout для уникнення перенавчання. Загальна схема нейронної мережі представлена на рис. 1.

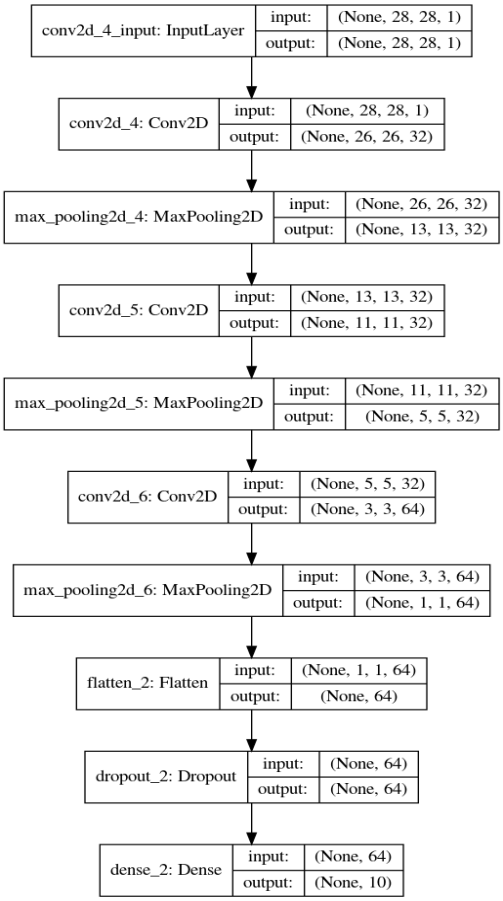


Рис. 1. Загальна структура нейронної мережі

Результати експериментів ефективності методів оптимізації нейронних мереж

На рисунках 2-8 представлені результати, отримані після проведення експериментів. За результатами Adam - найкращий оптимізатор з точки зору точності на тестовому наборі даних та значення функції втрат. Метод Nadam отримав майже аналогічні результати в порівнянні з попереднім методом. RMSProp на тестовому наборі показав точність приблизно 80%. Всі інші методи показали значно гірші результати навчання. Варто відзначити покращення, виявлене для методів AdaGrad та SGD з методом моментів(~40% точності) порівняно з SGD (30%), SGD Nesterov (~25%). Для конвергенції моделей необхідно було 140 епох тренування і в середньому 1,5 години для всіх оптимізаторів.

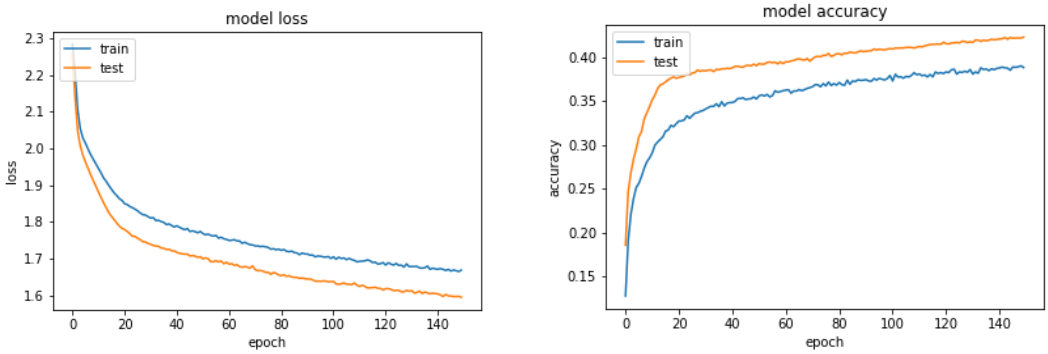


Рис. 2. Графіки результатів тренування нейронної мережі з використанням оптимізатора AdaGrad

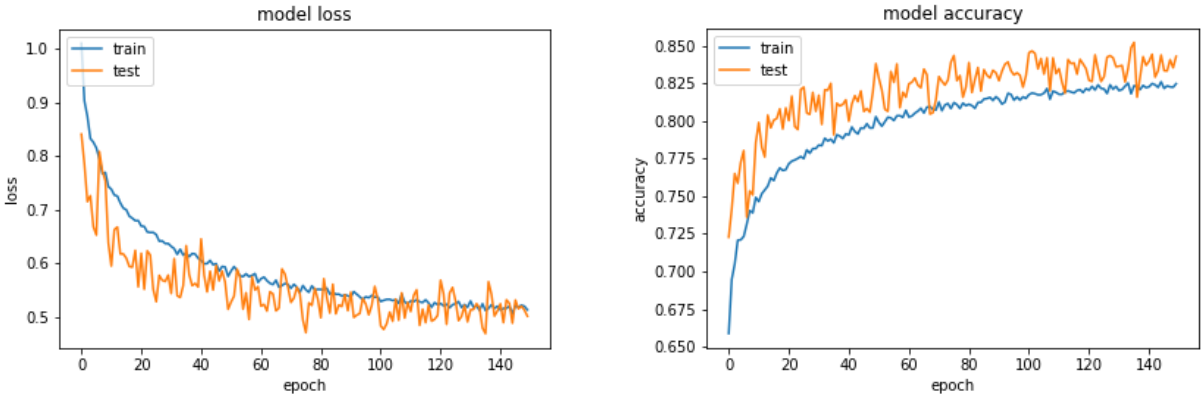


Рис. 3. Графіки результатів тренування нейронної мережі з використанням оптимізатора Adam

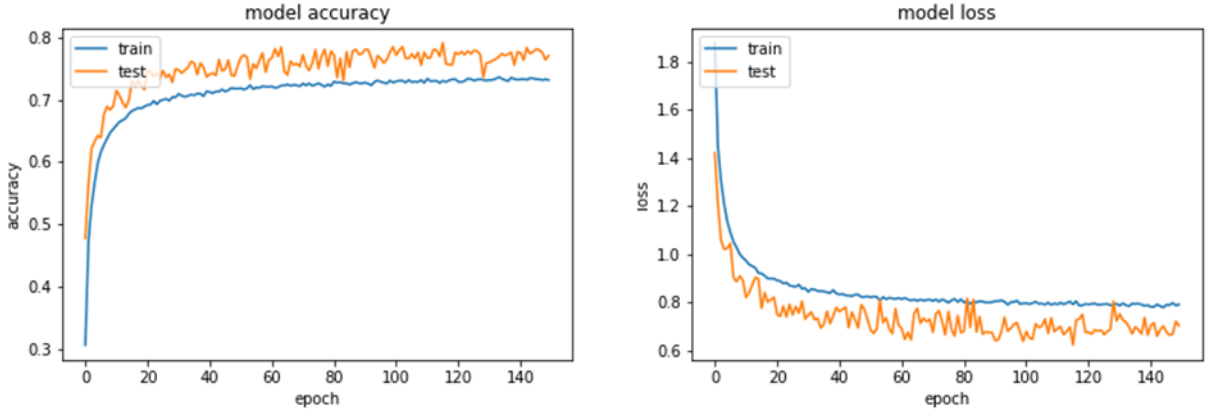


Рис. 4. Графіки результатів тренування нейронної мережі з використанням оптимізатора Nadam

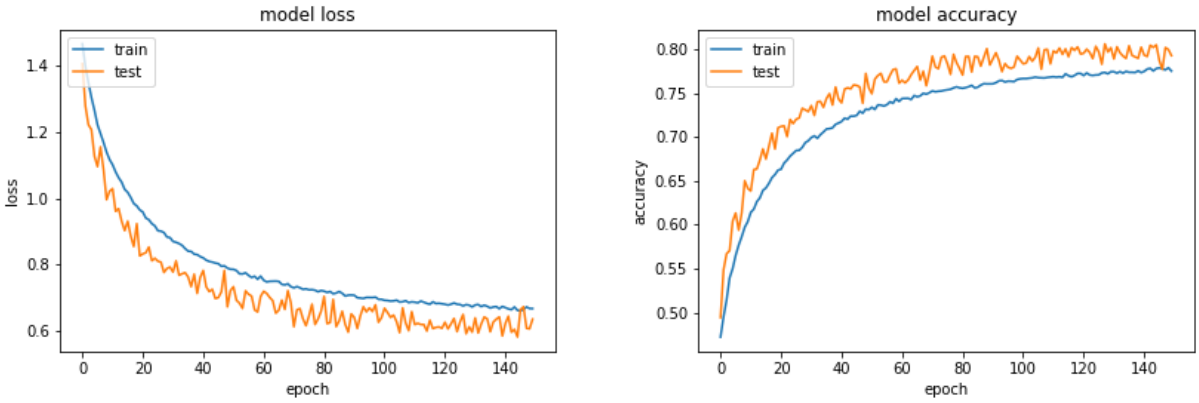


Рис 5. Графіки результатів тренування нейронної мережі з використанням оптимізатора RMSProp

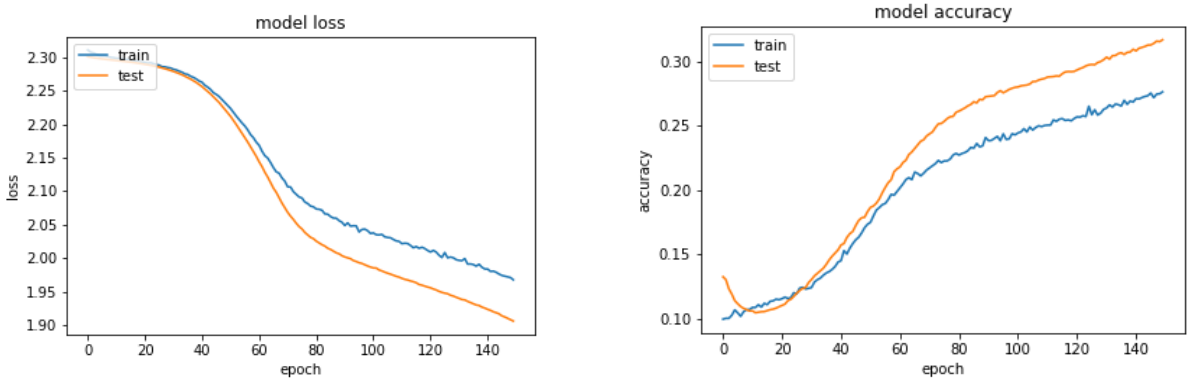
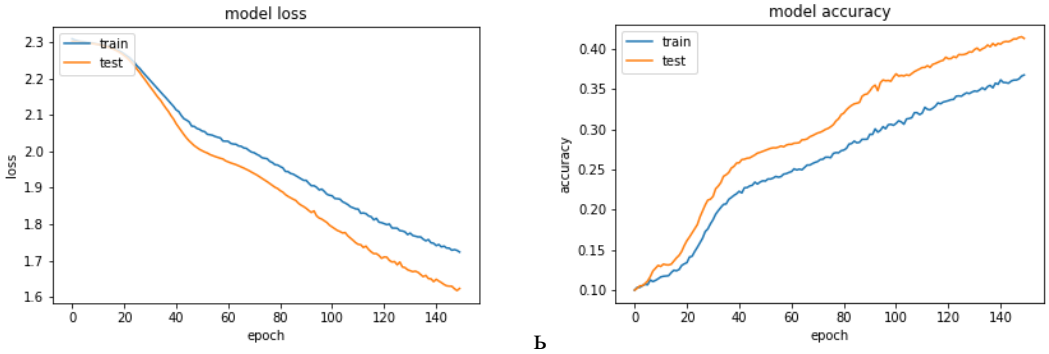


Рис. 6. Графіки результатів тренування нейронної мережі з використанням оптимізатора SGD



ь

Рис. 7. Графіки результатів тренування нейронної мережі з використанням оптимізатора SGD з використанням методу моментів

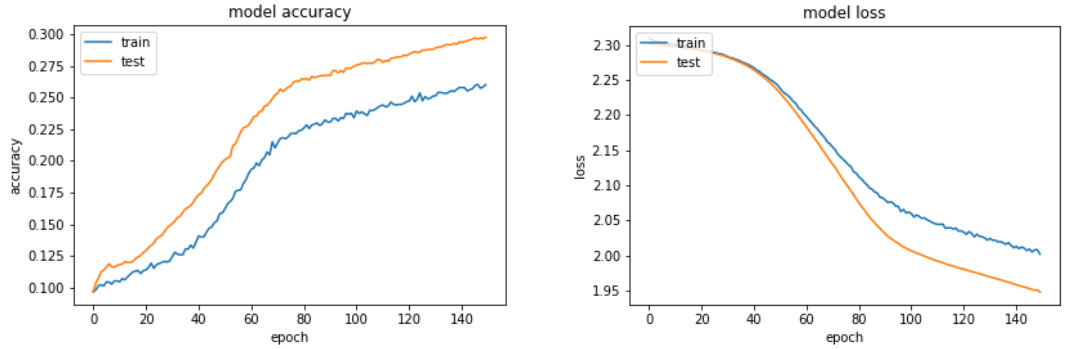


Рис. 8. Графіки результатів тренування нейронної мережі з використанням оптимізатора SGD з використанням методу Нестерова

У даному експерименті було проведено порівняльний аналіз семи алгоритмів оптимізації за допомогою простої архітектури згорткової нейронної мережі для класифікації зображень. Результати показують, що продуктивність кожного оптимізатора відрізнялася підчас тренування, що підтверджує вплив типу та розміру даних на продуктивність різних оптимізаторів. На основі декількох проведених експериментів можна зробити висновок, що оптимізатори Adam та Nadam демонструють більш досконалі та надійні показники точності та значення функції втрат, в порівнянні з іншими методами оптимізації. Це можна пояснити тим, що дані методи поєднують в собі найкращі сторони прискореного градієнта Нестерова (NAG) та адаптивного оцінювання градієнта(Adam).

Для аналізу ефективності різних розподілених методів тренування нейронних мереж було використано декілька популярних архітектур глибоких моделей: InceptionV4 [3], ResNet50 [4], VGG19[5], ResNet152[4] та VGG16 [5]. Під час проведення експериментів було використано синтетичний набір даних ImageNet[6] для оцінки продуктивності методів без впливу затримки зв'язку в мережі. Синтетичні набори даних - це випадкові згенеровані значення пікселів, які відповідають розмірам реальних даних (256 × 256 для ImageNet). Використання синтетичних наборів даних дозволяє перевірити обчислювальну продуктивність та нівелювати комунікаційні витрати \ читання запису з диска.

Моделі, які використовувались для експериментів, є реалізацією оригінальних архітектур за допомогою TensorFlow [7]. Моделі були вибрано для представлення різних архітектур та кількості параметрів. Моделі та їх відповідна кількість параметрів, кількість операцій за секунду, та точність розпізнавання наведена в таблиці 1.

Таблиця 1 – Моделі, використані для проведення експериментів

Архітектура	Кількість параметрів (млн.)	Кількість операцій в секунду (GFLOPS)	Точність розпізнавання для ImageNet (%)
VGG16	138	30.9	90.0
VGG19	143	39	92.7
ResNet50	25	10	92.9
Inception4	65	20	95.0

Для проведення досліджень було використано кластер AWS p2.16xlarge який складається з:
- 16х відеокарт NVIDIA K80 12GB
- Процессор Intel Xeon E5-2686 v4 (2.3 ГГц)
- 732 ГБ ОЗП

Тренування відбувалось на базі операційної системи Ubuntu 16.04 з використанням наступного програмного забезпечення:
- Open MPI: version 3.0.1
- CUDA Toolkit: version 9.0
- NCCL: version 2.1.15
- cuDNN: version 7.1
- python: 3.6
- Horovod: 0.18.0
- TensorFlow: 1.12

Експерименти проводилися за допомогою тестів TensorFlow benchmarks [7]. В таблиці 2 описано параметри тренувального процесу для кожного з типів розподілених обчислень.

Таблиця 2 – Параметри проведення експерименту

Параметри	Tensorflow Distributed (сервер параметрів)		Horovod
	Синхронний	Асинхронний	
К-ть процесів для формування даних	2-10	2-10	2-20
Розмір міні-пакетів (batch size) для кожного з паралельних процесів	32	32	32
Тип даних	синтетичні	синтетичні	синтетичні
К-ть ітерацій навчання	500	500	500

Результати дослідження продуктивності методів розподіленого тренування нейронних мереж. Сервер параметрів з синхронним оновленням параметрів вимагає від усіх процесів оновлення глобальної моделі, перш ніж градієнти можуть бути передані процесам. В експериментах із синхронним оновленням сервера параметрів оцінювався ефект пливу пропускної здатності мережі на загальний процес тренування. На рис. 9 показані результати в зображеннях на секунду, оброблені різними моделями з використання синхронного сервера параметрів.

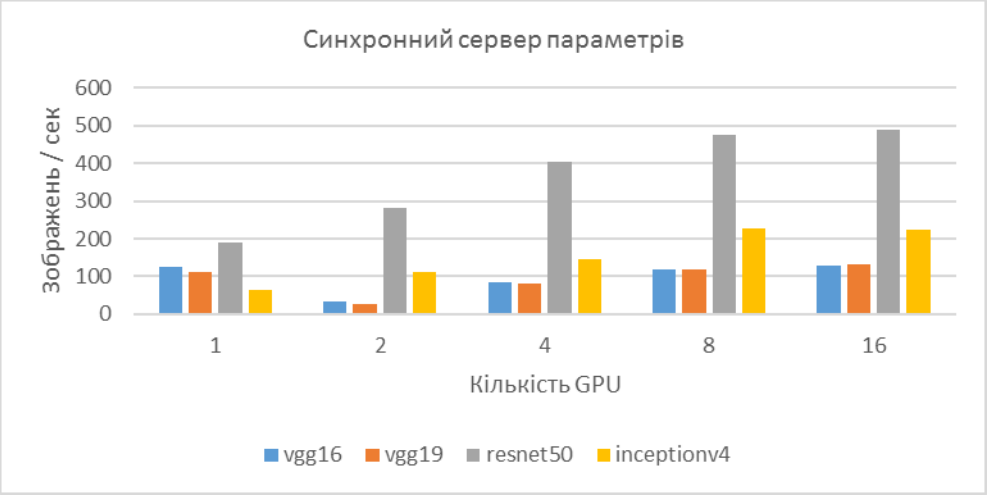


Рис. 9. Результати тренування синхронного сервера параметрів

На рис. 10 представлено результати, оброблені моделями з використанням сервера параметрів в режимі асинхронного оновлення градієнтів. Режим асинхронного оновлення не показав значного покращення продуктивності у порівнянні з синхронним оновленням параметрів.

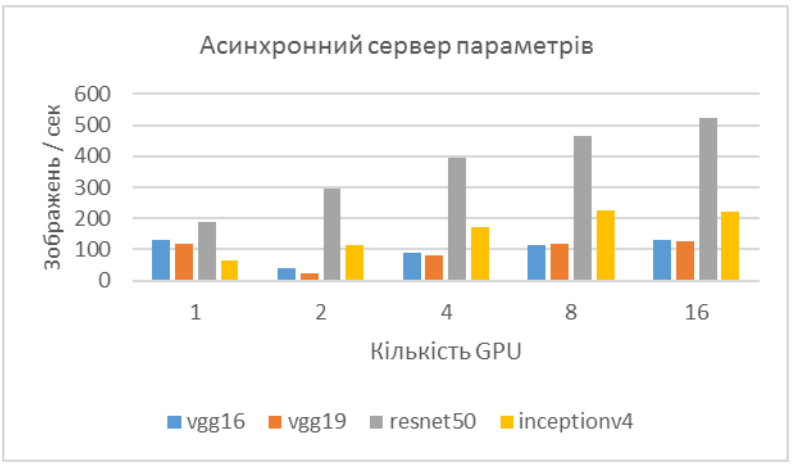


Рис. 10. Результати тренування асинхронного сервера параметрів

Результати експериментів Ring All-reduce (Horovod) показані на рис. 11. Усі моделі, крім vgg16, демонструють вдосконалення з кожним доданим процесом. Хоча, кількість зображень які обробляються за секунду, для такої ж кількості процесів, є майже вдвічі більшою порівняно з сервером параметрів.

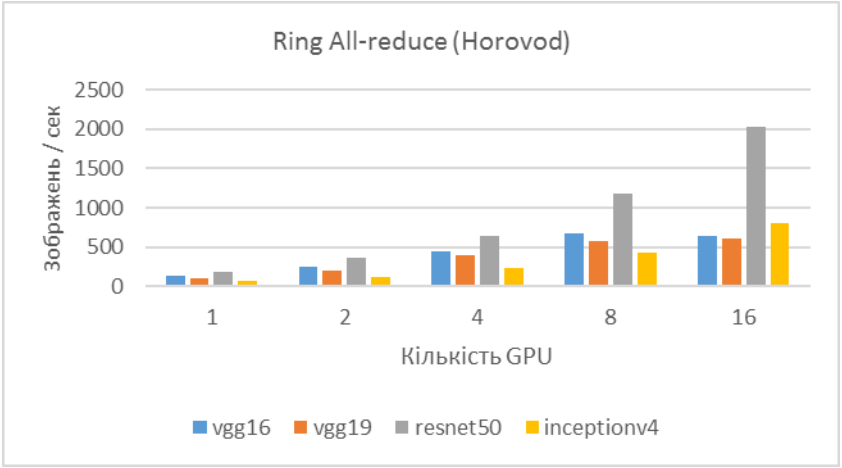


Рис. 11. Результати тренування методом Ring All-reduce

Масштабованість методів розподіленого тренування – це один важливий фактор для розподіленого тренування. Масштабованість – коефіцієнт збільшення кількості зображень оброблених за секунду, кожним доданим обчислювальним ресурсом (GPU). На рис. 12 показано швидкість роботи в режимі синхронного сервера параметрів порівняно з ідеальним масштабуванням (зелений графік). На рис. 13, представлено масштабованість тренування за допомогою сервера параметрів із асинхронним оновленням. Результати експериментів Ring All-reduce (Horovod) представлено на рис. 14.

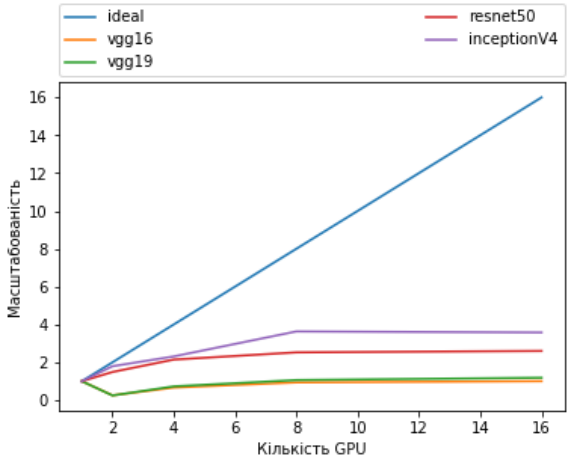


Рис. 12. Результати оцінки масштабованості розподіленого тренування синхронного сервера параметрів.

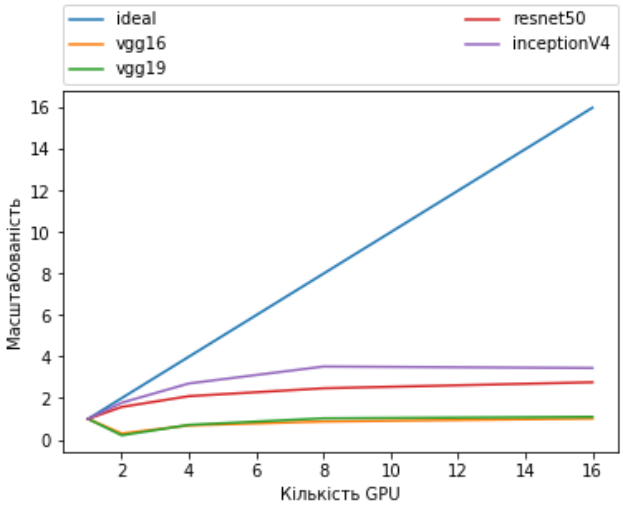


Рис. 13. Результати оцінки масштабованості розподіленого тренування асинхронного сервера параметрів

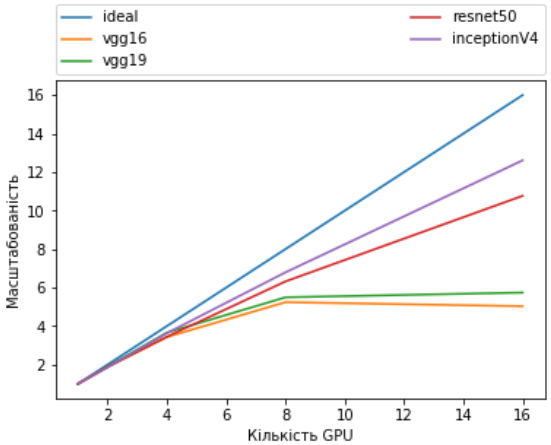


Рис. 14. Результати оцінки масштабованості розподіленого тренування з використанням Ring All-reduce (Horovod)

Ефективність серверів параметрів є не дуже великою, як це можна побачити на рисунках 4 та 5, це пояснюється необхідністю передавати кожному процесу додаткові дані на сервер параметрів, що і уповільнює процес зі збільшенням кількості графічних процесорів.

З результатів, представлених вище, легко зробити висновок що Ring All-reduce (horovod) працює найкраще для всіх моделей. Синхронне та асинхронне оновлення серевера параметрів не показало значного прискорення продуктивності.

Однією з причин того що асинхронний серевер оновлення параметрів показує аналогічні результати що і синхронний, може бути той факт, що всі процеси були розпочаті одночасно і мали однакові обчислювальні ресурси. Таким чином, навіть без жодних обмежень синхронізації всі працівники завершують обрахунки і одночасно надсилаються оновлені дані на сервер параметрів.

Результати тренування Ring All-reduce показують, що даний метод майже лінійно масштабується на деяких моделях, але не має такого ж ефекту для великих моделей, таких як vgg16 та vgg19. Це показує, що для великих моделей навіть висока пропускна спроможність мережі 25Gb/s може бути недостатньою.

Висновки. Популярність технології глибоких нейронних мереж, безсумнівно росте, тому оптимізація процесу тренування глибоких нейронних мереж є дуже важливим завданням для зменшення витрат, економії часу та ресурсів.

Отже, розподілення глибокого навчання в більшості випадків покращило результативність проведених експериментів. Хоча між алгоритмами існує чітка різниця у продуктивності, не існує жодного способу, який би був кращий за інші результати у всіх аспектах. Ring All-reduce масштабується найкраще у всіх тестах, але сервер параметрів легше розгортати в кластерах, а сервер параметрів у режимі асинхронного оновлення є більш стійким до помилок, тому добре підходить для динамічного середовища. Вибір алгоритму розподіленого глибокого навчання повинен враховувати:

1. розмір моделі для навчання та кількість наявних ресурсів (наприклад: для тренування такої великої моделі, як vgg16, слід враховувати пропускну здатність мережі, швидкість шини PCIe та розміщення графічного процесора);
2. компроміс між масштабованістю та можливою відмовою системи.

На основі даних експериментів систематизовано та обґрунтовано оптимальні варіанти використання методів оптимізації під час тренування нейронних мереж. Результати тренування нейронних мереж показують що варіанти розподілення мають різну ефективність в залежності від архітектури нейронної мережі та параметрів системи тренування.

Список бібліографічного опису.

1. Вілсон, Д. Р. та Мартінес, Т. Р. (2003). Загальна неефективність пакетної підготовки для градієнтного спуску. Нейронні мережі, 16 (10), 1429–1451.
2. Рао, С. (1945). Інформація та точність, досяжна при оцінці статистичних параметрів. Вісник Математичного товариства Калькутти, 37, 81–89.
3. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna. "Переосмислення архітектури початків для комп'ютерного зору". В: CoRR abs / 1512.00567 (2015). arXiv: 1512.00567. URL: http: //arxiv.org/abs/1512.00567.
4. К. Він, Х. Чжан, С. Рен та Дж. Сун. «Глибоке залишкове навчання для розпізнавання зображень». В: Ресурс обчислювальних досліджень abs / 1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/ 1512.03385.
5. К. Симонян та А. Зіссерман. "Дуже глибокі згортки для розпізнавання зображень великого масштабу". В: Ресурс обчислювальних досліджень abs / 1409.1556 (2014). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556. К. Симонян та А. Зіссерман. "Дуже глибокі згортки для розпізнавання зображень великого масштабу". В: Ресурс обчислювальних досліджень abs / 1409.1556 (2014). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.
6. ImageNet [Електронний ресурс]. - Режим доступу: http://www.image-net.org/ (Дата звернення 25.10.19 р.).
7. Тезові показники Tensorflow. https://github.com/tensorflow/ орієнтири / дерево / головний / сценарії / tf_cnn_benchmarks. 2018 рік.
8. Метод Коші та градієнта - math.uni-bielefeld.de/documenta/vol-ismp/40_lemarechal-claude.pdf
9. Рассел, С. Дж. І Норвіг, П. (2003). Штучний інтелект: сучасний підхід. Prentice Hall
10. Duchi, J., Hazan, E. and Singer, Y. (2011). Адаптивні субградієнтні методи онлайн-навчання та стохастичної оптимізації. Журнал досліджень машинного навчання.
11. Танг, Ю., Салаххутдінов, Р., і Гінтон, Г. (2012). Глибокі суміші факторних аналізаторів. переддрук arXiv arXiv: 1206,4635.
12. Кінгма, Д. і Ба, Дж. (2014). Адам: Метод стохастичної оптимізації. переддрук arXiv arXiv: 1412.6980

References.

1. Wilson, D. R. and Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. Neural Networks, 16(10), 1429–1451.
2. Rao, C. (1945). Information and the accuracy attainable in the estimation of statistical parameters. Bulletin of the Calcutta Mathematical Society, 37, 81–89.
3. C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna. “Rethinking the Inception Architecture for Computer Vision”. In: CoRR abs/1512.00567 (2015). arXiv: 1512.00567. URL: http: //arxiv.org/abs/1512.00567.

4. K. He, X. Zhang, S. Ren, and J. Sun. "Deep Residual Learning for Image Recognition". In: Computing Research Repository abs/1512.03385 (2015). arXiv: 1512.03385. URL: http://arxiv.org/abs/ 1512.03385.
5. K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: Computing Research Repository abs/1409.1556 (2014). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556. K. Simonyan and A. Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: Computing Research Repository abs/1409.1556 (2014). arXiv: 1409.1556. URL: http://arxiv.org/abs/1409.1556.
6. ImageNet [Електронний ресурс]. – Режим доступу: http://www.image-net.org/ (Дата звернення 25.10.19 р.).
7. Tensorflow benchmarks. https://github.com/tensorflow/ benchmarks/tree/master/scripts/tf_cnn_benchmarks. 2018.
8. Cauchy and the Gradient Method- math.uni-bielefeld.de/documenta/vol-ismp/40_lemarechal-claude.pdf
9. Russel, S. J. and Norvig, P. (2003). Artificial Intelligence: a Modern Approach. Prentice Hall
10. Duchi, J., Hazan, E., and Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. Journal of Machine Learning Research.
11. Tang, Y., Salakhutdinov, R., and Hinton, G. (2012). Deep mixtures of factor analysers. arXiv preprint arXiv:1206.4635.
12. Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980