УДК 004.6 DOI: 10.30748/soi.2020.160.15

С.В. Мінухін

Харківський національний економічний університет ім. С. Кузнеця, Харків

ДОСЛІДЖЕННЯ МОДЕЛІ СЕГМЕНТАЦІЇ ЗОБРАЖЕНЬ З ВИКОРИСТАННЯМ РОЗПОДІЛЕНИХ РЕЖИМІВ TENSORFLOW ТА ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ U-NET

Розглянуто модель сегментації медичних зображень з використанням різних розподілених режимів бібліотеки машинного навчання TensorFlow та згорткові мережі U-Net. Проведено аналіз можливостей розподілених обчислень для їх використання в TensorFlow. Описана побудована архітектура кластера робочих станцій та послідовність кроків для проведення експериментальних досліджень. Отримано оцінки втрат при навчанні у вигляді коефіцієнта Дайса, які свідчать про переваги використання синхронного режиму розподіленого навчання та в умовах масштабованості розгорнутого кластера.

Ключові слова: *модель, сегментація, машинне навчання, TensorFlow, згорткова нейронна мережа, U-Net синхронний та асинхронний режими, розподілені обчислення, медичні зображення.*

Вступ

Натепер різні галузі діяльності з використанням інформаційних технологій містять задачі, які можуть вирішуватися за допомогою сегментації: аналіз медичних зображень, виділення об'єктів на супутникових знімках, розпізнавання облич, автоматичні системи управління дорожнім рухом, тощо. Одним з існуючих методів їх розв'язку є сегментація зображень на основі нейронних мереж, яка має поширення через відносну простоту використання існуючих моделей для різних видів сегментації без необхідності глибинного аналізу предметної області. Однією з досить недавно розроблених моделей сегментації зображень є нейронна мережа U-Net. Вона є підвидом згорткових нейронних мереж (ЗНМ), які призначені для сегментації, зокрема, медичних зображень, та показує досить гарні результати.

Процес навчання нейронних мереж часто потребує великої кількості обчислювальних ресурсів, у разі масштабованості у зв'язку з різким збільшенням розмірностей, задач, що виконуються, що призводить до необхідності об'єднання потужностей окремих комп'ютерів у обчислювальний кластер. Відповідно, на цей певний виклик стала поява програмної бібліотеки для систем машинного навчання TensorFlow. Вона отримала підтримку для використання в розподілених системах навчання, та при цьому може працювати в декількох режимах. Взаємодія цих режимів з різними моделями нейронних мереж та інших методів машинного навчання відома, але для кожного типу моделі мають місце свої особливості.

Метою роботи є дослідження та аналіз результатів сегментації біомедичних зображень з використанням повнозгорткової нейронної мережі U-Net щодо продуктивності розподіленого навчання у порівнянні з локальним режимом на основі бібліотеки машинного навчання Tensorflow.

Виклад основного матеріалу

Огляд бібліотеки машинного навчання TensorFlow та її застосування для розподілених обчислень

Наприкінці 2015 р. Google запропоновано бібліотеку глибокого навчання TensorFlow [1]. У Google TensorFlow можна створювати різноманітні додатки – від пошуку до карт до перекладу, – отже й тому вона була протестована в досить великих масштабах застосувань. Бібліотека з відкритим кодом містить лише реалізацію з однією машиною (комп'ютером), можливо, через повну залежність розподіленої версії від інфраструктури Google (Google File System). TensorFlow має Python API для побудови обчислювальних графів, які потім виконуються в C++, тоді як Spark написаний на Scala і має Java і Python API. Суттєвим питанням для TensorFlow on Spark (TFoS) є те, яким чином розподілити процес навчання нейронних мереж на Spark. Spark в значному ступені підходить для ітеративних завдань зі зменшенням карти (парадигми паралельних обчислень Map/Reduce), але навчання нейронних мереж не є проблемою для зменшення карти. Архітектура DownGourSGD є паралельною передачею даних, тобто кожен вузол-працівник (slave) має всю модель і працює на даних, відмінних від інших працівників (паралелізм даних). Після першої публікації TensorFlow в квітні 2016 р. Google випустив розширений TensorFlow з розподіленими можливостями глибокого навчання, а потім у TensorFlow додано підтримку HDFS (TFoS) (рис. 1) [2–3] та можливості паралелізму даних (рис. 2). Поза межами хмари Google користувачі все ще потребували виді-

леного кластеру для додатків TensorFlow. Програми TensorFlow не можна було розгорнути на існуючих кластерах великих даних, тим самим збільшуючи вартість та затримку для тих користувачів, хто хотів скористатися цією технологією в масштабі.

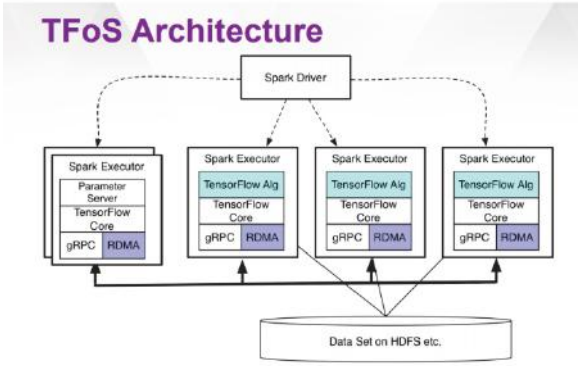

Рис. 1. Архітектура TFoS

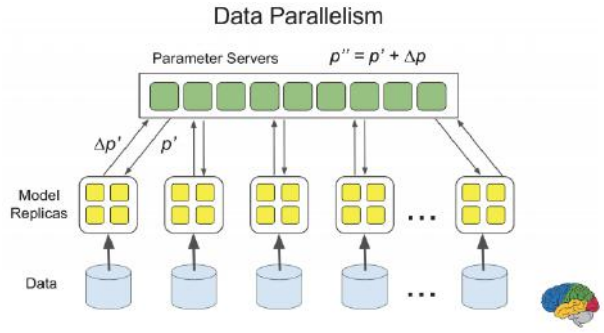

Рис. 2. Модель паралелізму даних у TF

Розподілений TensorFlow з використанням tf.distribute.Strategy. DistributionStrategy API є простим способом поширення розподілу навчальних робочих навантажень між декількома машинами (комп'ютерами). DistributionStrategy API призначений для надання користувачам доступу до існуючим моделям і коду. При цьому користувачам треба тільки внести мінімальні зміни в моделі і код, щоб застосувати розподілене навчання.

Підтримка стратегій розподілу (рис. 3–4). MirroredStrategy: підтримка синхронного розподіленого навчання на декількох графічних процесорах на одному комп'ютері. Це створює одну репліку на пристрій GPU. Кожна змінна в моделі відбивається у всіх репліках. Разом ці змінні утворюють єдину концептуальну змінну, яка називається MirroredVariable. TMirroredVariables синхронізуються один з одним шляхом застосування ідентичних оновлень. MultiWorkerMirroredStrategy – це версія MirroredStrategy для навчання декількох працівників (slaves). Він реалізує синхронне розподілене навчання між декількома працівниками, та при цьому кожен з яких може мати кілька графічних процесорів. Він створює копії всіх змінних в моделі на кожному пристрої для всіх працівників. ParameterServerStrategy: підтримка сервера параметрів. Його можна використовувати для синхронного локального навчання з декількома графічними процесорами або асинхронного навчання на декількох машинах (комп'ютерах). При використанні у разі локального навчання на одній машині змінні не відображаються, замість цього вони будуть розміщені в ЦП, а операції реплікують в усі локальні графічні процесори. MirroredStrategy: підтримка синхронного розподіленого навчання на декількох графічних процесорах на одному комп'ютері. Це створює одну репліку на пристрій GPU. Кожна змінна в моделі відбивається у всіх репліках. Разом ці змінні утворюють єдину концептуальну змінну – MirroredVariable. TMirroredVariables синхронізуються один з одним шляхом застосування ідентичних оновлень. MultiWorkerMirroredStrategy – це версія MirroredStrategy для навчання декількох працівників. Вона реалізує синхронне розподілене навчання між декількома працівниками, кожен з яких може мати кілька графічних процесорів та будує копії всіх змінних в моделі на кожному пристрої для всіх працівників.

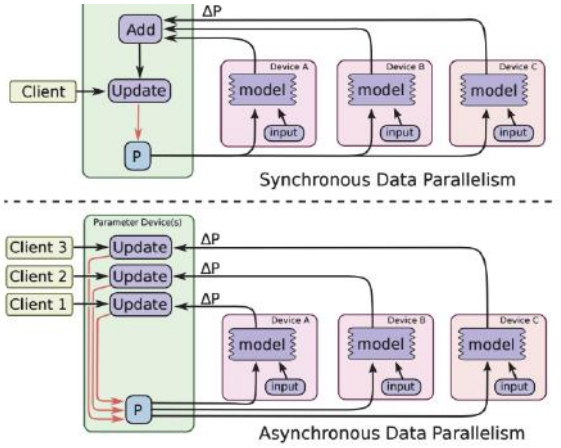

Рис. 3. Синхронний та асинхронний режими паралелізму даних

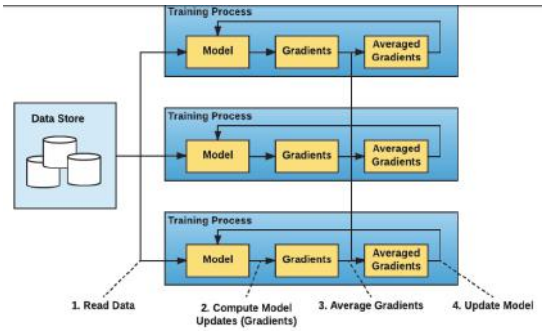

Рис. 4. Розподілене навчання на основі паралельних моделей

Архітектура нейронної мережі U-Net

Нейронна мережа U-Net була розроблена для сегментації біомедичних зображень у відділенні Computer Science Фрайбургського університету. U-Net є підвидом згорткових нейронних мереж, ві-

домим як повнозгорткові мережі [4–6]. Цей вид згорткових мереж відрізняється від традиційних тим, що просторова інформація (тобто розташування кожного фрагменту згортки) враховується на кожному шарі завдяки відсутності повноз'єднаних шарів у мережі. Це також дозволяє мережам даного типу оперувати зображеннями будь-якого розміру, що є неможливим з традиційними мережами, оскільки розмір вхідного зображення визначається кількістю вхідних нейронів у першому повноз'єднаному шарі.

Архітектура мережі наведена на рис. 5 (числа над прямокутниками позначають кількість каналів ознак, числа у нижньому лівому куті – розмір зображення [6]).

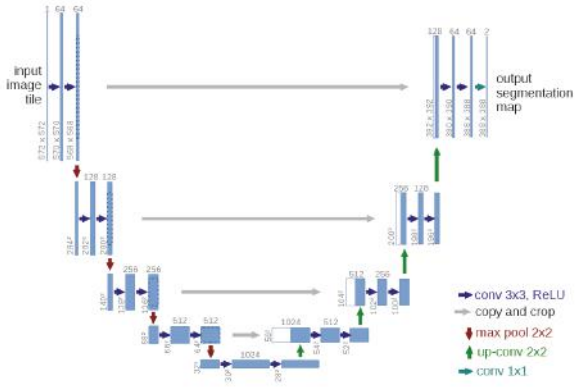

Рис. 5. Архітектура мережі U-Net

Вона складається зі зменшувального шляху (ліва частина) та збільшувального шляху (права частина). Зменшувальний шлях має собою типову згорткову мережу та складається з повторного виконання двох згорток 3×3, за кожною з яких слідує шар ReLU та 2×2 агрегування з функцією максимуму та кроком 2 для зменшення розміру зображення. На кожному кроці зменшення відбувається подвоєння кількості каналів ознак. Кожний крок збільшувального шляху складається зі збільшення карти ознак, за якою слідує згортка 2×2, що ділить навпіл кількість каналів ознак, конкатенація з обрізаною картою ознак з відповідного кроку зменшувального шляху, та двома згортками 3×3 з шарами ReLU. Образка карти ознак необхідна через те, що кожна згортка призводить до втрати кутових пікселів. На останньому кроці згортка 1×1 перетворює 64-елементний вектор ознак у бажану кількість класів. Всього мережа має 23 згорткових шара. Для того, щоб дозволити безшовне покриття вихідної карти сегментації, потрібно вибрати такий розмір вхідного сегмента, щоб усі 2×2 максимум-пулінг операції виконувалися над шаром з парним x- та y-розміром.

Однією з важливих особливостей архітектури U-Net є те, що збільшувальний шлях має велику кількість каналів ознак, що дозволяє передавати контекстну інформацію до шарів більшого розміру зображення. Таким чином, збільшувальний шлях є більш-менш симетричним до зменшувального шляху, через що схема архітектури має U-подібну форму. Мережа не має повноз'єднаних шарів та використовує тільки вагому частину кожної згортки, тобто карта сегментації містить тільки пікселі, для яких присутній повний контекст у вхідному зображенні. Це дозволяє виконувати безшовну сегментацію довільно великих зображень за допомогою стратегії перекриття-покриття. Для передбачення пікселів у прикордонному регіоні зображення, відсутній контекст екстраполюється за допомогою дзеркального відображення вхідного зображення. Ця стратегія покриття необхідна для використання мережі з великими зображеннями, бо в іншому випадку розмір буде обмежений доступним об'ємом пам'яті. Так як часто даних для навчання замало, було запропоновано використовувати надмірне збільшення об'єму даних за допомогою еластичної деформації доступних зображень. Це дозволяє мережі навчитися ігнорувати такий тип деформації без зовнішнього втручання. Це особливо важливо у таких областях, як сегментація біомедичних зображень, де такий тип деформації є найбільш поширеним та може бути ефективно змодельованим.

Вхідні зображення та відповідні карти сегментації використовуються для навчання мережі за допомогою реалізації метода стохастичного градієнтного спуску [7–8]. Через використання згорток без доповнення, вихідне зображення має менший розмір, ніж вхідне на константанту величину. Щоб мінімізувати надмірне використання пам'яті було вирішено віддати пріоритет більшому розміру сегментів покриття та зменшити розмір пакету до одного зображення. Також використовується висока інерція (0.99) оптимізатора, що забезпечує використання великої кількості попередньо оброблених зображень для обчислення зміни до поточного кроку оптимізації. Для оцінки навчання мережі використовується коефіцієнт Дайса (Dice coefficient) [9–10].

Розподілене навчання з використанням TensorFlow

TensorFlow може виконувати розподілене навчання як в синхронному, так і в асинхронному режимі [11]. Різницю між ними можна пояснити та дослідити на прикладі алгоритму стохастичного градієнтного спуску (СГС), який є одним з найбільш популярних алгоритмів для навчання нейронних мереж. Він складається з декількох раундів, де результати кожного з них включаються в модель перед наступним раундом. Кожний раунд СГС виконується над міні-пакетом навчальних зразків. У синхронному режимі навчання кожний з вузлів навчає свою локальну модель на різних частинах даних з одного більшого міні-пакету. Після цього вони від-

силають свої локально обчислені градієнти усім іншим вузлам, і лише після того, як усі вузли закінчили обчислення своїх градієнтів, виконується оновлення моделі. Оновлена модель потім відсилається усім вузлам разом зі зрізами наступного міні-пакету. Таким чином, вузли навчаються на зрізах міні-пакету, що не перетинаються один з одним.

Хоча паралелізм має можливість значно підвищити швидкість навчання, він має певні недоліки. Великий розмір моделі або мала пропускна здатність мережі призведе до збільшення часу навчання. Навчання може призупинитися взагалі, якщо один з вузлів є повільним або має погане підключення до мережі. Також доречним є зменшення кількості ітерацій навчання моделі, бо кожна ітерація потребує відправки оновленої моделі на кожний з вузлів. По суті це значить збільшення розміру міні-пакету як тільки можливо, поки подальше збільшення не почне погіршувати точність моделі. *В асинхронному режимі* вузли не чекають оновлень моделі від інших вузлів. Вони можуть навчатись незалежно та обмінюватися результатами як рівноправні члени або відправляти їх на один або декілька центральних серверів (або "сервери параметрів" в документації TensorFlow). В рівноправній архітектурі кожен вузол виконує цикл, який зчитує навчальні дані, обчислює градієнти та відправляє їх на інші вузли та оновляє модель усіма доступними на даний момент даними. В централізованій архітектурі вузли відправляють обчислені градієнти на сервери параметрів, які збирають та агрегують їх. У синхронному навчанні сервери параметрів обчислюють поточну версію оновленої моделі та відправляють її на робочі вузлі. В асинхронному навчанні сервери параметрів лише відправляють градієнти на робочі вузли та дозволяють їм власноручно обчислити нову модель. В обох архітектурах (рис. 6) цей цикл виконується до тих пір, поки навчання не

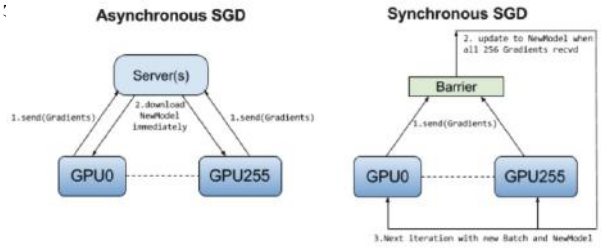

Рис. 6. Асинхронне та синхронне навчання з використанням СГС

При використанні архітектури з сервером параметрів (рис. 7) алгоритм починається з відправки моделі на робочі вузли. На кожному кроці навчання кожний вузол обчислює свої градієнти та відправляє їх на один або декілька серверів параметрів. Сервери параметрів агрегують усі градієнти з вузлів та чекають поки кожен вузол закінчить свої обчислення перед оновленням моделі для наступної ітерації.

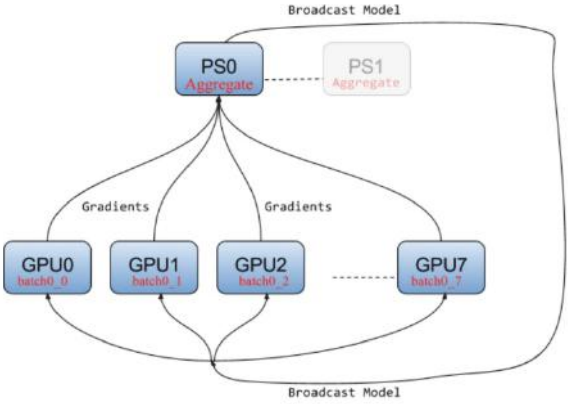

Рис. 7. Архітектура з сервером параметрів

Експериментальні дослідження сегментації медичних зображень з застосуванням розподіленого TensorFlow та U-Net

Проведення досліджень виконувалося на основі наступних кроків.

Крок 1. Налаштування кластеру. Складається з установки TensorFlow та залежностей та розгортання тестового коду. Найбільш простим способом установки TensorFlow є використання менеджеру пакетів Python. Це гарантує автоматичну установку останньої версії та усіх необхідних залежностей. Розгортання тестового коду можна виконати багатьма способами, наприклад, за допомогою засобів автоматичного розгортання (Docker, Kubernetes, тощо). У даному випадку це було зроблене вручну шляхом копіювання необхідних файлів на кожний з вузлів кластеру за допомогою протоколу SSH. Для проведення експериментів використана одна з доступних реалізацій мережі U-Net з використанням TensorFlow [12]. Ця реалізація не призначена для навчання в розподіленому режимі, та отже потребує внесення необхідних налаштувань.

Експеримент здійснювався на кластері з 3-х робочих станцій, об'єднаних у локальну мережу з високою пропускною здатністю (1 Gbit/s); на кластері з 2-х робочих станцій та в локальному режимі на одній з цих робочих станцій. Апаратна конфігурація усіх робочих станцій однакова та включає:

1) CPU: Intel Core i5-7500 CPU @ 3.40GHz;
   а) кількість ядер: 4 (без HyperThreading);
   б) L1 кеш: 256 KB;
   в) L2 кеш: 1 MB;
   г) L3 кеш: 6 MB;
2) RAM: 8 GB DDR4 @ 2400 MHz;
3) HDD: 931 GB (1 TB);
   а) інтерфейс: SATA;
   б) типова швидкість даних: 156 MB/s;
4) мережевий адаптер: Realtek RTL8411 @ 1 Gbit/s;
5) операційна система: Ubuntu Linux 16.0 LTS.

Крок 2. Підготовка даних для навчання.

Для експерименту використано набір даних Kaggle Ultrasound Nerve Segmentation [13]. Вибір цього набору даних пояснюється тим, що нейронна мережа U-Net початково була розроблена для сегментації біомедичних зображень. Використаний набір даних представляє собою ультразвукові знімки людини та попередньо створені бінарні маски, що виділяють групу нервів – плечове нервове сплетіння (рис. 8).

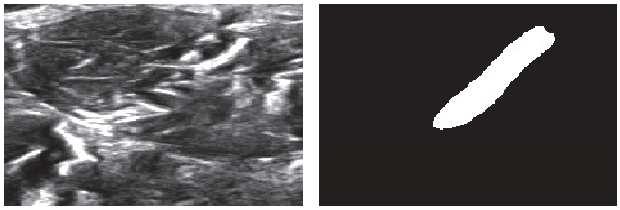

Рис. 8. Зображення (ліва частина) та відповідна маска (права частина) з використаного набору даних

Крок 3. Запуск сервера параметрів.

Режим розподіленого обчислення в TensorFlow використовує 2 типи вузлів: робочі вузли та сервери параметрів. Останні відповідають за зберігання усіх параметрів моделі, тому вони мають бути запущені перед робочими вузлами. Оскільки сервери параметрів не потребують великої кількості обчислювальних ресурсів, їх можна запускати на тих самих фізичних вузлах, що і робочі вузли. Для навчання TensorFlow в локальному режимі цей крок не потрібен.

Крок 4. Запуск робочих вузлів.

Після запуску серверів параметрів треба виконати запуск робочих вузлів. Якщо конфігурація кластеру була задана правильно, то кожний робочий вузол автоматично з'єднається з серверами параметрів та почне навчання. У коді робочий вузол з індексом 0 виводить інформацію про процес навчання у потік стандартного виводу, тому при запуску цій потік перенаправляється у файл для подальшого аналізу результатів після завершення навчання.

Крок 5. Збір результатів.

Окрім файлу з результатами процесу навчання, використана реалізація U-Net на першому робочому вузлі для кожної епохи навчання обчислює передбачення маски для декількох зразків, що вибираються випадково перед початком навчання. Всі ці результати зберігаються поряд з оригінальним зображенням та відомою маскою. Приклад цього формату показаний на рис. 9. Ці зображення допомагають візуально оцінювати ефективність навчання мережі.

Рис. 9. Приклад результату передбачення мережі U-Net

Крок 6. Обробка даних журналу навчання.

Інформація про процес навчання, яку виводить використана реалізація U-Net, включає номер епохи, значення функції втрат (продуктивність навчання моделі) та часову мітку. Ці дані дозволяють виконати порівняльний аналіз для різних режимів та конфігурацій кластера та U-Net. Приклад цих даних для однієї епохи показаний на рис. 10. Для більш зручно-го аналізу цих даних можна скористатися будь-яким зручним засобом обробки тексту для перетворення їх в стандартний формат (наприклад, CSV).

```
2019-05-23 11:22:49,822 Verification error= 5.1%, loss= 0.2276
2019-05-23 11:22:52,230 Iter 1980, Minibatch Loss= 0.2344, Training Accuracy= 0.9309, Minibatch error= 6.9%
2019-05-23 11:22:56,110 Iter 1982, Minibatch Loss= 0.0993, Training Accuracy= 0.9724, Minibatch error= 2.8%
2019-05-23 11:23:00,078 Iter 1984, Minibatch Loss= 0.1510, Training Accuracy= 0.9703, Minibatch error= 3.0%
2019-05-23 11:23:03,912 Iter 1986, Minibatch Loss= 0.2210, Training Accuracy= 0.9488, Minibatch error= 5.1%
2019-05-23 11:23:07,700 Iter 1988, Minibatch Loss= 0.2004, Training Accuracy= 0.9484, Minibatch error= 5.2%
2019-05-23 11:23:11,549 Iter 1990, Minibatch Loss= 0.2580, Training Accuracy= 0.9362, Minibatch error= 6.4%
2019-05-23 11:23:15,387 Iter 1992, Minibatch Loss= 0.1545, Training Accuracy= 0.9643, Minibatch error= 3.6%
2019-05-23 11:23:19,270 Iter 1994, Minibatch Loss= 0.2622, Training Accuracy= 0.9397, Minibatch error= 6.0%
2019-05-23 11:23:23,063 Iter 1996, Minibatch Loss= 0.1497, Training Accuracy= 0.9650, Minibatch error= 3.5%
2019-05-23 11:23:26,938 Iter 1998, Minibatch Loss= 0.1533, Training Accuracy= 0.9544, Minibatch error= 4.6%
2019-05-23 11:23:28,684 Epoch 99, Average loss: 0.1443, learning rate: 0.0010
```

Рис. 10. Приклад даних, що надається про процес навчання мережі U-Net

Крок 7. Після конвертації даних у придатний формат можна будувати порівняльні графіки для усіх режимів навчання, що тестувалися.

Крок 8. Дослідження розподіленого асинхронного режиму.

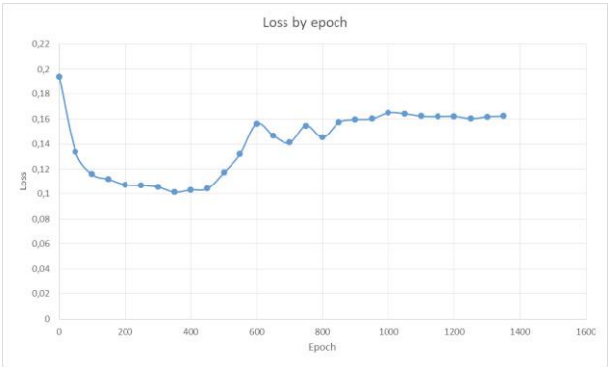

Рис. 11. Залежність функції втрат від кількості епох навчання для розподіленого асинхронного режиму для 3 вузлів

Крок 9. Дослідження розподіленого синхронного режиму.

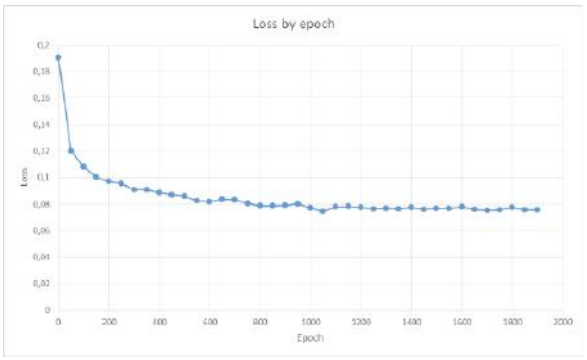

Рис. 12. Залежність функції втрат від кількості епох навчання для розподіленого синхронного режиму для 3 вузлів

На рис. 12 показано, що величина функції втрат становить 8%, що краще ніж результат навчання 12%, отриманий для асинхронного режиму (рис. 11).

Крок 10. Порівняльний аналіз синхронного та асинхронного режимів для різних конфігурацій кластера.

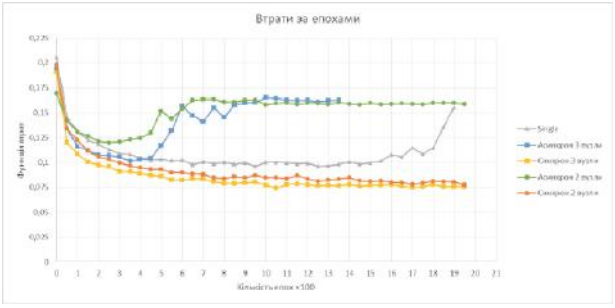

Рис. 13. Залежність функції втрат від кількості епох навчання для розподілених режимів при різних конфігураціях кластера

З рис. 13 слідує, що похибка моделі сегментації зображень для синхронного режиму загалом нижче, ніж для інших двох режимів. Це пояснюється більшою кількістю навчальних даних, що обробляються за раз (тобто "ефективний розмір міні-пакету"): в синхронному режимі воно більше в N разів, де N – кількість вузлів кластеру (дорівнює 2 або 3 у даному випадку).

Висновки

Розглянуто актуальне завдання підвищення продуктивності розв'язку задач сегментації медичних зображень з застосуванням високопродуктивних розподілених обчислень. Для підвищення продуктивності моделі сегментації проаналізовано можливості та використано розподілений режим роботи бібліотеки машинного навчання TensorFlow. Для розв'язку задач сегментації, що потребують значних часових випробувань, пропонується використовувати різні розподілені режими – асинхронні та синхронні, які загалом в результаті проведених експериментальних досліджень підтверджують їх ефективність у порівнянні із локальним режимом. Масштабування кластерної системи призводить загалом до зменшення величини функції втрат та запобігання перенавчання, але потребує додаткових ресурсів в рамках високопродуктивної архітектури кластера, а також:

використання синхронного розподіленого режиму навчання дозволяє досягти бажаної точності сегментації значно швидше, ніж в локальному режимі;

використання асинхронного розподіленого режиму навчання призводить до погіршення точності сегментації та тому є придатним тільки в певних випадках, коли пропускна здатність навчання важливіша за точність навчання;

є необхідність контролювати кількість навчальних епох нейронної мережі для запобігання її перенавчання.

Проблема можливого перенавчання стає важливим фактором для розробки та налаштування програмного забезпечення певної архітектури кластера для заданої (можливо апріорі) архітектури згорткової мережі для застосування розподілених режимів бібліотеки TensorFlow.

В подальшому дослідження будуть спрямовані на дослідження впливу масштабованості кластера на час навчання та тестування розподілених режимів TensorFlow.

Список літератури

1. TensorFlow [Електронний ресурс]. – Режим доступу: https://www.tensorflow.org/. – Назва з титул. екрану.

2. TensorFlowOnSpark [Електронний ресурс]. – Режим доступу : https://github.com/yahoo/TensorFlowOnSpark. – Назва з титул. екрану.

3. Open Sourcing TensorFlowOnSpark: Distributed Deep Learning on Big-Data Clusters [Електронний ресурс] / L. Yang, J. Shi, B. Chern, A. Feng. – Режим доступу: https://developer.yahoo.com/blogs/157196317141/. – Назва з титул. екрану.

4. Ronneberger O. U-Net: Convolutional Networks for Biomedical Image Segmentation [Електронний ресурс] / O. Ronneberger, P. Fischer, T. Brox. – Режим доступу: https://arxiv.org/pdf/1505.04597.pdf/. – Назва з титул. екрану.

5. Long J. Fully Convolutional Networks for Semantic Segmentation [Електронний ресурс]. / J. Long, E. Shelhamer, T. Darrell. – Режим доступу : https://arxiv.org/pdf/1411.4038.pdf. – Назва з титул. екрану.

6. U-Net: Convolutional Networks for Biomedical Image Segmentation. U-Net - Computer Vision Group, Freiburg [Електронний ресурс]. – Режим доступу : https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/. – Назва з титул. екрану.

7. Ruder S. An overview of gradient descent optimization algorithms [Електронний ресурс] / S. Ruder. – Режим доступу: https://arxiv.org/pdf/1609.04747.pdf. – Назва з титул. екрану.

8. Darken C. Learning rate schedules for faster stochastic gradient search / C. Darken, J. Chang, J. Moody // Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop, 1992. September. – P. 1-11. https://doi.org/10.1109/NNSP.1992.253713.

9. Statistical Validation of Image Segmentation Quality Based on a Spatial Overlap Index / H. Zou, S.K. Warfield, A. Bharatha, C.M. Tempany, M.R. Kaus, S.J. Haker, W.M. Wells, F.A. Jolesz, R. Kikinis. https://doi.org/10.1016/S1076-6332(03)00671-8.

10. Sørensen–Dice coefficient [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Dice_coefficient. – Назва з титул. екрану.

11. Distributed Deep Learning Using Synchronous Stochastic Gradient Descent [Електронний ресурс] / D. Das, S. Avancha, D. Mudigere, K. Vaidynathan, S. Sridharan, D. Kalamkar, B. Kaul, P. Dubey. – Режим доступу: https://arxiv.org/pdf/1602.06709.pdf. – Назва з титул. екрану.

12. Akeret J. Tensorflow Unet Documentation [Електронний ресурс] / J. Akeret. – Режим доступу: https://readthedocs.org/projects/tf-unet/downloads/pdf/latest/. – Назва з титул. екрану.

13. Ultrasound Nerve Segmentation [Електронний ресурс]. – Режим доступу: https://www.kaggle.com/c/ultrasound-nerve-segmentation. – Назва з титул. екрану.

References

1. The official site of TensorFlow (2020), *An end-to-end open source machine learning platform*, available at: www.tensorflow.org/.

2. The official site of TensorFlowOnSpark (2020), *Join GitHub today*, available at: www.github.com/yahoo/TensorFlowOnSpark.

3. Yang, L., Shi, J., Chern, B. and Feng, A. (2017), *Open Sourcing TensorFlowOnSpark: Distributed Deep Learning on Big-Data Clusters*, available at: www.developer.yahoo.com/blogs/157196317141/.

4. Ronneberger, O. Fischer, P. and Brox, T. (2015), *U-Net: Convolutional Networks for Biomedical Image Segmentation*, available at: www.arxiv.org/pdf/1505.04597.pdf.

5. Long, J., Shelhamer, E. and Darrell, T. (2015), *Fully Convolutional Networks for Semantic Segmentation*, available at: www.arxiv.org/pdf/1411.4038.pdf.

6. The official site of U-Net - Computer Vision Group Freiburg, (2015), *U-Net: Convolutional Networks for Biomedical Image Segmentation*, (2015), available at: www.lmb.informatik.uni-freiburg.de/people/ronneber/u-net/.

7. Ruder, S. (2016), *An overview of gradient descent optimization algorithms*, available at: www.arxiv.org/pdf/1609.04747.pdf.

8. Darken, C., Chang, J. and Moody, J. (1992), Learning rate schedules for faster stochastic gradient search, *Neural Networks for Signal Processing II Proceedings of the 1992 IEEE Workshop*. https://doi.org/10.1109/NNSP.1992.253713.

9. Zou, K.H., Warfield, S.K., Bharatha, A., Tempany, C.M., Kaus, M.R., Haker, S.J., Wells, W.M., Jolesz, F.A. and Kikinis, R. (2004), Statistical Validation of Image Segmentation Quality Based on a Spatial Overlap Index, *Acad Radiol.*, No. 11(2), pp. 178-189. https://doi.org/10.1016/S1076-6332(03)00671-8.

10. *Sørensen–Dice coefficient*, available at: www.en.wikipedia.org/wiki/Dice_coefficient.

11. Das, D., Avancha, S., Mudigere, D., Vaidynathan, K., Sridharan, S., Kalamkar, D., Kaul, B. and Dubey, P. (2016), *Distributed Deep Learning Using Synchronous Stochastic Gradient Descent*, available at: www.arxiv.org/pdf/1602.06709.pdf.

12. Akeret, J. (2018), *Tensorflow Unet Documentation*, available at: www.readthedocs.org/projects/tf-unet/downloads/pdf/latest/.

13. The official site of Kaggle (2020), *Ultrasound Nerve Segmentation website*, available at: www.kaggle.com/c/ultrasound-nerve-segmentation.

Надійшла до редколегії 29.01.2020
Схвалена до друку 10.03.2020

Відомості про автора:

Мінухін Сергій Володимирович
доктор технічних наук доцент
професор Харківського національного
економічного університету ім. С. Кузнеця,
Харків, Україна
https://orcid.org/0000-0002-9314-3750

Information about the author:

Sergii Minukhin
Doctor of Technical Sciences Associate Professor
Professor of Simon Kuznets Kharkiv National
University of Economics,
Kharkiv, Ukraine
https://orcid.org/0000-0002-9314-3750

ИССЛЕДОВАНИЕ МОДЕЛИ СЕГМЕНТАЦИИ ИЗОБРАЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ РАСПРЕДЕЛЕННЫХ РЕЖИМОВ TENSORFLOW И СВЕРТОЧНОЙ НЕЙРОННОЙ СЕТИ U-NET

С.В. Минухин

Рассмотрена модель сегментации медицинских изображений с использованием различных распределенных режимов библиотеки машинного обучения TensorFlow и сверточной сети U-Net. Проведен анализ возможностей распределенных вычислений для их использования в TensorFlow. Описана построенная архитектура кластера рабочих станций и последовательность шагов для проведения экспериментальных исследований. Получены оценки потерь при обучении в виде коэффициента Дайса, свидетельствующие о преимуществах использования синхронного режима распределенного обучения в условиях масштабируемости развернутого кластера.

Ключевые слова: *модель, сегментация, машинное обучение, TensorFlow, сверточная нейронная сеть, U-Net, синхронный и асинхронный режимы, распределенные вычисления, медицинские изображения.*

PERFORMANCE STUDY FOR IMAGE SEGMENTATION MODEL BASED ON DISTRIBUTED TENSORFLOW AND CNN U-NET

S. Minukhin

An analysis is made of the use of distributed computing capabilities to solve machine learning tasks. In particular, although concurrency has the potential to significantly increase the speed of learning, it has some disadvantages. Large model sizes or low computer network bandwidth can increase learning time. Training may pause altogether if one of the nodes is slow or has poor network connectivity. It is also appropriate to reduce the number of model training iterations, as each iteration requires sending an updated model to each of the cluster nodes. In essence, this determines the increase in the size of the mini-package until further enlargement begins to impair the accuracy of the model. A model using an artificial convolutional neural network for the implementation of a system for solving the problem of segmentation of the medical image (ultrasound image of the human neck to highlight the shoulder plexus) is proposed. The performance of using a convolutional U-Net neural network to decouple segmentation of biomedical images under scalable scalability has been investigated. A cluster system has been deployed as part of work nodes and a parameter server to solve the ultrasound segmentation task and train the U-Net convolutional network to segment human shoulder plexus images using real Ultrasound Nerve Segmentation data. The results of training in the process of distributed computing in asynchronous and synchronous modes are obtained based on the estimation of the Dice loss function. The results of the training make it possible to justify the use of a convolutional neural network with a sufficiently high precision of segmentation of medical images within 8% -12% depending on the number of resources and the mode of synchronization of data processing in the learning process. It is proved that increasing the number of working nodes in case of possible scaling of the cluster reduces the function of learning losses in the range of 3-5%. Using TensorFlow makes it possible to improve image segmentation accuracy by using distributed mode.

Keywords: *model, segmentation, machine learning, TensorFlow, convolutional neural network, U-Net, synchronous and asynchronous modes, distributed computing, medical images.*