DOI 10.31891/2307-5732-2024-337-3-16
УДК 004.8

СЕВЕРИНЕНКО ДАНИЛО
Національний Університет “Львівська Політехніка”
https://orcid.org/0000-0002-1708-316X
e-mail: danylo.y.severynenko@lpnu.ua

РОЗРОБКА ТА НАВЧАННЯ НАЙРОННОЇ МЕРЕЖІ ЗА ДОПОМОГОЮ RAY

Документ описує розробку та навчання нейронної мережі за допомогою фреймворка Ray, зосереджуючись на випадку з набором даних Fashion MNIST. Описано вплив машинного навчання у різних галузях і здатність обробляти великі масиви даних з високою точністю. Ray допомагає упоратися з викликами масштабування ML-застосунків, забезпечуючи простоту розробки складних систем та управління великомасштабними обчисленнями.

Описано технічний процес підготовки, тренування та валідації нейронної мережі для розпізнавання зображень одягу з набору Fashion MNIST за допомогою Ray. Подано архітектуру нейронної мережі з різними шарами, включаючи ReLU і відмову, для нелінійності та регуляції. Тренування включає розподілені обчислення, демонструючи здатність Ray обробляти обчислення на різному обладнанні.

Ray Train, що спрощує масштабування глибокого навчання, дозволяє масштабувати моделі без значних змін коду, підвищуючи продуктивність розробників. Дослідження завершується успішним створенням та тренуванням нейронної мережі з використанням Ray, підкреслюючи ефективність фреймворка у підвищенні продуктивності систем ML.

Ключові слова: машинне навчання, фреймворк Ray, Fashion MNIST, нейронна мережа, розподілені обчислення, масштабування, обробка даних, тренування моделей, глибоке навчання.

SEVERYNENKO DANYLO
Lviv Polytechnic National University

DEVELOPMENT ANS TRAINING OF A NEURAL NETWORK USING RAY

The document discusses the development and training of a neural network using the Ray framework, particularly focusing on a case study involving the Fashion MNIST dataset. Machine Learning (ML) has revolutionized various fields by enabling the processing and analysis of vast data sets with unprecedented accuracy and speed. It highlights ML's impact in healthcare, environmental science, and astrophysics, illustrating its ability to improve disease prediction models, climate change scenarios, and astronomical data analysis.

Ray, an open-source Python framework, addresses several challenges associated with scaling ML applications. It simplifies the development of complex, distributed systems by providing a unified API that supports parallel task execution and large-scale model training. This framework efficiently manages tasks that require high-performance computing resources, such as real-time data processing and extensive simulations. Ray's scalability from single machines to clusters makes it indispensable in environments with fluctuating computational demands.

The document details the technical process of setting up, training, and validating a neural network to recognize clothing images from the Fashion MNIST dataset using Ray. It covers the initial steps, such as importing necessary libraries and preparing the dataset, and outlines the architecture of the neural network built with various layers including ReLU and dropout for non-linearity and regularization to prevent overfitting. The training process involves distributed computing, highlighting Ray's capabilities in handling stateful and stateless computations across diverse hardware setups.

Furthermore, Ray Train, previously known as Ray SGD, is part of Ray's ecosystem designed to simplify the scaling of deep learning and ML across multiple GPUs and nodes. It allows users to scale their ML models from a single computer to a distributed environment without significant code modifications, enhancing the developer's productivity by minimizing setup time for iterative development.

The paper concludes with the successful development and training of a neural network using Ray, which demonstrates the framework's efficiency in enhancing the performance and reliability of ML systems across various industries. This study not only reaffirms Ray's role in advancing ML application scalability but also sets the stage for future explorations in distributed computing for artificial intelligence.

Keywords: Machine Learning, Ray Framework, Fashion MNIST, Neural Network, Distributed Computing, Scalability, Data Processing, Model Training, Deep Learning.

Постановка проблеми у загальному вигляді та її зв’язок із важливими науковими чи практичними завданнями

Машинне навчання (ML) стало основою сучасних наукових досліджень, революціонізуючи різні галузі завдяки своїй здатності обробляти та аналізувати величезні масиви даних із небувалою точністю та швидкістю. У сфері охорони здоров'я алгоритми машинне навчання допомагає передбачати спалахи захворювань, персоналізувати лікування та автоматизувати діагностичні процеси, значно покращуючи результати для пацієнтів. У галузі екології дослідники використовують машинне навчання для моделювання сценаріїв зміни клімату та оптимізації систем відновлюваної енергії. Крім того, у сфері астрофізики машинне навчання допомагає аналізувати величезні набори даних з телескопів та космічних апаратів для ідентифікації небесних тіл та явищ, що перевищують можливості людського зору.

Відкритий фреймворк для Python Ray, розглядає кілька викликів, які виникають при масштабуванні застосунків машинного навчання. Його основна сила полягає в спрощенні розробки складних, розподілених систем шляхом надання уніфікованого програмного інтерфейсу (API) для виконання паралельних завдань та навчання моделей у великому масштабі. Ray ефективно обробляє завдання, які вимагають високопродуктивних обчислювальних ресурсів, такі як обробка даних у реальному часі та масштабні симуляції. Крім того, його здатність безперебійно масштабуватися від одного комп'ютера до кластера

робить його незамінним інструментом у середовищах, де обчислювальні потреби можуть несподівано зростати. Зменшуючи проблеми, пов'язані з масштабованістю та управлінням ресурсами, Ray дозволяє дослідникам та розробникам більше зосереджуватися на розробці моделей, а не на викликах, пов'язаних з інфраструктурою.

Аналіз останніх досліджень і публікацій

Машинне навчання (ML) значно впливає на галузь охорони здоров'я, покращуючи моделі прогнозування захворювань. Кілька останніх досліджень висвітлюють використання різних технік ML для більш точного прогнозування захворювань. Наприклад, методи ансамблю навчання, такі як беггінг, бустинг, стекінг і голосування, показали обіцянки у покращенні точності прогнозів для таких станів, як діабет, шкірні захворювання, хвороби нирок, печінки та серцеві умови. Ці техніки інтегрують кілька моделей, щоб забезпечити кращу продуктивність, ніж окремі класифікатори, вирішуючи проблему помилок узагальнення, які часто асоціюються з прогнозами на основі однієї моделі [1].

Більше того, алгоритми ML, такі як лінійна регресія, логістична регресія, дерева рішень і випадкові ліси, широко використовуються у галузі охорони здоров'я для прогностичної аналітики. Лінійна регресія, наприклад, відмінно підходить для прогнозування безперервних результатів і є простою у впровадженні, що робить її популярним вибором для попереднього аналізу ризику захворювань. Логістична регресія, з іншого боку, ідеально підходить для бінарних результатів і часто використовується для оцінки ймовірності виникнення захворювання. Дерева рішень та випадкові ліси забезпечують більш тонкий підхід, обробляючи як завдання класифікації, так і регресії, що дозволяє детально аналізувати складні набори даних [2].

Останні розробки у фреймворках розподіленого обчислення, подібних до Ray, відзначають значний прогрес у масштабуванні машинного навчання та штучного інтелекту. Зокрема, Ray відіграв важливу роль у підвищенні продуктивності та ефективності витрат на великомасштабні робочі навантаження машинного навчання. Цей фреймворк дозволяє покращити масштабування та знизити затримку, що було корисним при розгортанні складних моделей у різних обчислювальних середовищах, включаючи хмарні платформи, такі як AWS, Azure та GCP [3].

Архітектура Ray підтримує дрібнозернисті, гетерогенні обчислення, які є необхідними для динамічного та гнучкого виконання завдань машинного навчання, починаючи від простої обробки даних до тренування складних політик з використанням різноманітного обладнання, такого як CPU, GPU та TPU. Його здатність обробляти обчислення без стану та зі станом робить його особливо ефективним для різноманітних застосувань машинного навчання, включаючи ті, що вимагають реальної взаємодії та навчання з середовища [4].

Крім того, Ray сприяє розгортанню та управлінню розподіленими застосунками через такі функції, як автоматичне масштабування та кластер Ray, який оптимізує розподіл ресурсів на основі потреб робочого навантаження. Ця адаптивність зробила Ray популярним вибором для розробників та компаній, які працюють над передовими технологіями штучного інтелекту, покращуючи продуктивність та надійність систем машинного навчання у різних галузях [5, 6].

Бібліотека Ray Train, раніше відома як Ray SGD, є частиною екосистеми Ray, призначеної для спрощення масштабування навчання глибокого навчання та машинного навчання для кількох графічних процесорів та вузлів[7]. Вона дозволяє користувачам легко масштабувати свої моделі машинного навчання з одного комп'ютера на розподілене середовище без необхідності значно змінювати їхній існуючий код.

Однією з ключових особливостей Ray Train є її гнучкість у використанні ресурсів. Система підтримує навчання моделей на багатьох графічних процесорах, де кожному процесу робітника призначається окремий графічний процесор, що дозволяє ефективно проводити паралельне навчання. Це доповнюється здатністю проводити розподілене навчання на різноманітному обладнанні, включаючи різні конфігурації центрального процесору та графічного процесору, доступні у хмарних середовищах, таких як AWS, Azure та GCP[8].

Ray Train розроблений з урахуванням продуктивності розробників, акцентуючи увагу на мінімальному часі налаштування для ітеративної розробки. Це дозволяє інженерам з машинного навчання швидко вносити зміни в свої моделі з мінімальним простоєм для налаштування розподілених процесів. Крім того, Ray Train за замовчуванням включає в себе відмовостійкість, зменшуючи перерви, спричинені нестабільними хмарними інстанціями, та забезпечуючи, що навчання може автоматично відновлюватися після перерв.

Для тих, хто працює в середовищах з багатохмарними або гібридними хмарами, Ray Train пропонує значні переваги, уникаючи замкнення вендора. Він надає гнучкість для розгортання робочих процесів навчання на будь-якому хмарному провайдері або навіть на власних установках, використовуючи Kubernetes для оркестрації. Ця адаптивність є важливою для підприємств, які прагнуть підтримувати гнучкість у своїй операційній інфраструктурі.

Формулювання цілей статті

Метою роботи є створення та тренування нейронної мережі за допомогою Ray Train, використовуючи набір даних Fashion MNIST.

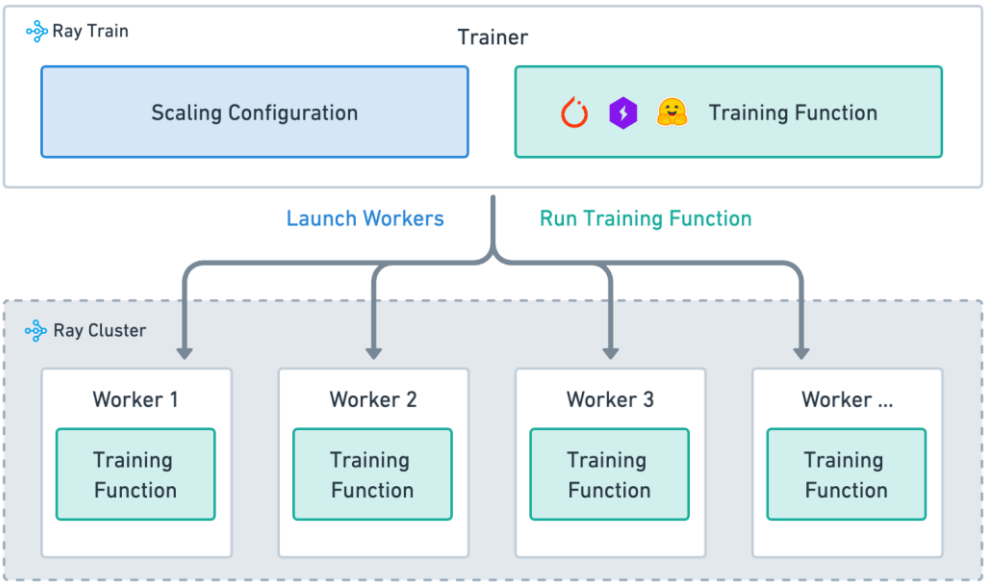

Рис. 1: Процес взаємодії Ray Train з кластером Ray

Розробка програмного забезпечення

Для початку розробки та навчання нейронної мережі необхідно підготувати всі необхідні бібліотеки і інструменти. Використання таких бібліотек як torch та torchvision є критично важливим для створення нейронних мереж та обробки даних зображень. Бібліотека ray.train використовується для розподіленого навчання, що дозволяє ефективніше масштабувати процес тренування моделі.

Першим кроком у процесі навчання є завантаження набору даних FashionMNIST. Цей набір даних є розширенням класичного MNIST і призначений для складнішого бенчмаркінгу алгоритмів машинного навчання. Далі, дані трансформуються та нормалізуються для подальшого використання у тренуванні моделі.

```python
def train_func_per_worker(config: Dict):
    lr = config["lr"]
    epochs = config["epochs"]
    batch_size = config["batch_size_per_worker"]

    # Get dataloaders inside the worker training function
    train_dataloader, test_dataloader = get_dataloaders(batch_size=batch_size)

    # [1] Prepare Dataloader for distributed training
    # Shard the datasets among workers and move batches to the correct device
    # =======================================================================
    train_dataloader = ray.train.torch.prepare_data_loader(train_dataloader)
    test_dataloader = ray.train.torch.prepare_data_loader(test_dataloader)

    model = NeuralNetwork()

    # [2] Prepare and wrap your model with DistributedDataParallel
    # Move the model to the correct GPU/CPU device
    # ============================================================
    model = ray.train.torch.prepare_model(model)

    loss_fn = nn.CrossEntropyLoss()
    optimizer = torch.optim.SGD(model.parameters(), lr=lr, momentum=0.9)

    # Model training loop
    for epoch in range(epochs):
        if ray.train.get_context().get_world_size() > 1:
            # Required for the distributed sampler to shuffle properly across epochs.
            train_dataloader.sampler.set_epoch(epoch)

        model.train()
        for X, y in tqdm(train_dataloader, desc=f"Train Epoch {epoch}"):
            pred = model(X)
            loss = loss_fn(pred, y)

            optimizer.zero_grad()
            loss.backward()
            optimizer.step()

        model.eval()
        test_loss, num_correct, num_total = 0, 0, 0
        with torch.no_grad():
            for X, y in tqdm(test_dataloader, desc=f"Test Epoch {epoch}"):
                pred = model(X)
                loss = loss_fn(pred, y)

                test_loss += loss.item()
                num_total += y.shape[0]
                num_correct += (pred.argmax(1) == y).sum().item()

        test_loss /= len(test_dataloader)
        accuracy = num_correct / num_total
```

Рис. 2. Частина функції train_func_per_worker, що відповідає за треунвання моделі на окремому кластері

Архітектура нейронної мережі включає декілька шарів, кожен з яких виконує певну функцію у процесі обробки вхідних зображень. Початковий шар перетворює двовимірні зображення в одновимірний вектор, що необхідно для обробки даних подальшими лінійними шарами. Ці шари здійснюють лінійну трансформацію даних для виявлення складних взаємозв'язків у вхідних зразках. Між лінійними шарами вставлені шари активації ReLU та шари відмови, які допомагають уникнути проблем зникнення градієнтів та зменшити перенавчання.

Процес навчання нейронної мережі проводиться у розподіленому середовищі, що дозволяє використовувати обчислювальні ресурси ефективніше. Завдяки цьому, можливе паралельне навчання на кількох обчислювальних вузлах, що значно прискорює загальний процес тренування та аналізу даних. Такий підхід дозволяє швидше досягати високої точності розпізнавання зображень та ефективніше оптимізувати параметри моделі.

Аналіз результатів

У результаті цієї роботи було створено та натреновано нейронну мережу для розпізнавання зображення одягу, базуючись на наборі даних Fashion MNIST, за допомогою Ray.

Висновки з даного дослідження

За результатами дослідження, розробка та навчання нейронної мережі за допомогою фреймворка Ray виявилися ефективними для обробки та класифікації зображень одягу з набору даних Fashion MNIST. Використання Ray значно спростило процес масштабування та розподілу обчислень, дозволяючи ефективно використовувати ресурси обчислювальних кластерів. Отримані результати підкреслюють потенціал фреймворка Ray як засобу для підвищення продуктивності та надійності систем машинного навчання, що може бути особливо корисним в галузях, де необхідно швидко обробляти великі масиви даних.

Література

1. Palak Mahajan, Shahadat Uddin, Farshid Hajati, Mohammad Ali Moni Ensemble Learning for Disease Prediction: A Review. Healthcare, 2023 - 11 – 1808. DOI: https://doi.org/10.3390/healthcare11121808
2. Mohammed Badawy, Nagy Ramadan & Hesham Ahmed Hefny. Healthcare predictive analytics using machine learning and deep learning techniques: a survey. Journal of Electrical Systems and Information Technology, 2023 - 10 – 40. DOI: https://doi.org/10.1186/s43067-023-00108-y
1. Productionizing and scaling Python ML workloads simply | Ray. https://www.ray.io/
2. Philipp Moritz, Robert Nishihara, Stephanie Wang. Ray: A Distributed Framework for Emerging AI Applications. arXiv:1712.05889, 2018. https://doi.org/10.48550/arXiv.1712.05889
3. Distributed Processing using Ray framework in Python | DataCamp. https://www.datacamp.com/tutorial/distributed-processing-using-ray-framework-in-python
4. Modern Distributed C++ with Ray Anyscale. https://www.anyscale.com/blog/modern-distributed-c-with-ray
5. Distributed Deep Learning with Ray Train is Now In Beta. https://www.anyscale.com/blog/distributed-deep-learning-with-ray-train-is-now-in-beta
6. Scaling AI and Machine Learning Workloads with Ray on AWS | AWS Open Source Blog. https://aws.amazon.com/blogs/opensource/scaling-ai-and-machine-learning-workloads-with-ray-on-aws/
7. Fashion MNIST. https://www.kaggle.com/datasets/zalando-research/fashionmnist/data
8. Glorot X. Bordes, Bengio A. Deep Sparse Rectifier Neural Networks. Proceedings of Machine Learning Research, 2021 - 15:315-323. https://proceedings.mlr.press/v15/glorot11a.html.