УДК 004.89

ДОРОГИЙ Я.Ю.,
ЦУРКАН В.В.,
ХРЕНОВ О.І.

РОЗРОБКА АРХІТЕКТУРИ СИСТЕМИ ДЛЯ МОДЕЛЮВАННЯ ЗГОРТОЧНИХ НЕЙРОННИХ МЕРЕЖ

Розглядається процес розробки архітектури системи, що допоможе у побудові згорточних нейронних мереж (ЗНМ), їх моделюванні та отриманні оброблених результатів.

The process of developing the system architecture, which will help in building of convolutional neural networks (CNN), their modeling and getting the processed results.

Вступ

У наш час, коли майже у кожному портативному пристрої є цифрова камера з можливістю зйомки фото та відео матеріалів, обсяги інформації зростають з величезною швидкістю, тому виникає проблема у автоматичному розпізнаванні зображень та їх класифікації. Глобально ці проблеми відносяться до терміну «машинне навчання» («Machine Learning»). Постійні публікації у даній сфері підтверджують актуальність даної проблеми.

Як приклад, у жовтні 2012 року відбувся конкурс ImageNet [1], що був присвячений класифікації об'єктів на фотографіях. У конкурсі було потрібно розпізнавання образів в 1000 категорій. Команда переможця Джефрі Хинтона використовувала методи поглибленого вивчення (Deep Learning) та згорточних нейронних мереж (Convolutional Neural Networks) [2].

У березні 2013 року Google інвестував в проект Хінтона, заснований при університеті Торонто. Протягом шести місяців був розроблений сервіс пошуку по фотографіях photos.google.com. Напевно, у кожного є величезний архів фотографій, тепер можна з легкістю їх класифікувати, що допоможе у майбутньому використанні цих матеріалів.

Та повернімось до методів, за допомогою яких ця можливість тепер є досяжною. Зокрема, це згорточні нейронні мережі – окремий клас нейронних мереж, який найкращим чином підходить для інтелектуальної обробки візуальних даних. ЗНМ були розроблені професором Яном Лекуном в кінці 1990-х років. Вже тоді ця технологія дозволяла надійно вирішувати задачі розпізнавання рукописних текстів. З тих пір значно збільшилася потужність комп'ютерів і з'явилися нові алгоритми навчання нейронних мереж.

Також варто зазначити, що Джефрі Хінтон був одним з дослідників, хто запропонував використовувати метод зворотного поширення помилки для тренування нейронних мереж.

Мета дослідження

Об’єктом дослідження даної роботи є процеси розпізнавання та класифікації зображень, а предметом дослідження – використання ЗНМ для розпізнавання та класифікації зображень.

Мета даної роботи – розробка інструментарію для моделювання ЗНМ. Для досягнення поставленої мети виконаний пошук і порівняння існуючих бібліотек для роботи з ЗНМ, наступним кроком був вибір додаткових компонент, а після цього – розробка зручного інтерфейсу, що об’єднує обрані інструменти.

ЗНМ та її застосування

Історія виникнення. Згорточна нейронна мережа представляє особливий клас нейронних мереж, який найкращим чином підходить для інтелектуальної обробки візуальних даних.

У 1981 році нейробіологи Торстен Візел та Девід Хабель досліджували зорову кору головного мозку кішки і виявили, що існують так звані прості клітини, які особливо сильно реагують на прямі лінії під різними кутами і складні клітини, що реагують на рух ліній в одному напрямку.

Пізніше Ян Лекун запропонував використовувати так звані згорточні нейронні мережі, як аналог зорової кори головного мозку для розпізнавання зображень [3,4,5].

Ян Лекун і його співробітники зробили дійсно гарну роботу по розпізнанню почерку, використовуючи ЗНМ, що були застосовані на практиці. Це один з небагатьох на той час прикладів

з застосуванням комп'ютера для тренування нейронних мереж, що мали добрий результат.

LeNet-5. Перший вражаючий приклад використання ЗНМ належить Яну Лекуну та його співробітниками, які створили дійсно добрий розпізнавач написаних від руки цифр. Ця мережа була використана для читання ~10% чеків в Північній Америці [6, 8].

Розроблена ЗНМ мала наступні особливості:

- велика кількість прихованих шарів;
- багато карт ознак у кожному шарі;
- об'єднання виходів близьких виділених ознак;
- виконувала розпізнавання, навіть, якщо символи перекривались;
- навчання виконувалось не окремих символів, а повного напису;
- у якості тренувального алгоритму використовувався алгоритм зворотного поширення помилки.

На рис.1. наведена архітектура LeNet-5.

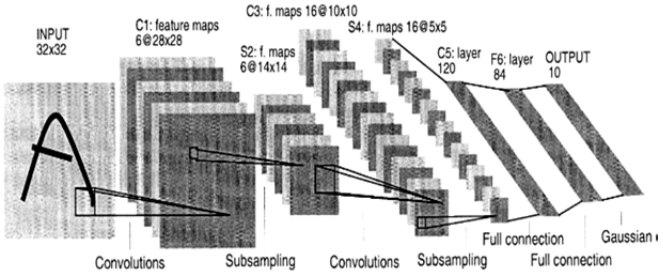

Рис. 1. Архітектура LeNet-5

При проходженні тесту на 10000 зображень помилка виникла лише у 82 випадках, це більш ніж 99% правильних відповідей. Тим не менше, більшість помилок, що робить LeNet-5, люди досить легко розпізнають, отже ці випадки також можна подолати.

Огляд існуючих бібліотек

Розглянемо основні бібліотеки для створення ЗНМ.

CudaCnn. CudaCnn бібліотека для ЗНМ, написана на C++/CUDA з інтерфейсом MATLAB. Підтримує наступні методи навчання: стохастичний градієнтний спуск, стохастичний Левенберга — Марквардта. Опціонально підтримка CUDA. Містить приклад для роботи з датасетом MNIST. Бібліотека не оновлюється з кінця 2012 року.

cuda-convnet. Являє собою швидку (заявлено розробником) C++/CUDA реалізацію ЗНМ. Навчання проводиться з використанням алгоритму зворотного поширення. Бібліотека не оновлюється з липня 2012 року.

nnForge. nnForge – це бібліотека для тренування згорточних та повнозв'язних нейронних мереж. Для обчислення може використовувати CPU і GPU (CUDA). Дана бібліотека написана на мові C++. Це відкрите програмне забезпечення поширюється під ліцензією Apache License v2.0.

Convolupy. Бібліотека написана на мові Python для роботи зі ЗНМ. Підтримка обчислень GPU відсутня. Не оновлюється з 2009 року. Сам розробник порадив використовувати рішення, над яким він зараз працює – Pylearn2 (докладніше у наступному розділі).

Pylearn2. Pylearn2 – бібліотека машинного навчання написана на мові Python [10]. Містить у собі так моделі: ЗНМ, багатошаровий перцептрон, softmax-регресія (як випадок багатошарового перцептрона), обмежена машина Больцмана та інші. Підтримує наступні методи навчання: стохастичний градієнтний спуск, пакетний градієнтний спуск.

Дана бібліотека використовує додаткову інструментарій Theano для оптимізації та обчислення математичних виразів, що включають багатовимірні масиви ефективно. Також ця бібліотека дозволяє використовувати GPU для обчислення складних операцій. Розробник заявляє, що у майбутньому для цього буде використовуватись не лише CUDA, а й OpenCL. Це означає можливу підтримку GPU не тільки компанії Nvidia, а й інших компаній.

Також варто зазначити, що Theano містить у собі реалізацію ЗНМ, та навіть існує бібліотека pynnet, що реалізує інтерфейс для роботи з ними, та вона не підтримується розробником з 2011 року. Сам розробник тепер приймає участь у покращенні та розробці Theano.

Заявленими перевагами Pylearn2 є зрозуміла структура з можливістю створення підкомпонентів та доступність для використання в навчальному процесі (використовується в Монреальському університеті). Бібліотека знаходиться у стадії «бурхливої» розробки.

Мінусом даної бібліотеки є відсутність повної документації, але цей недолік виправляється. Також для основних моделей існують навчальні матеріали у форматі ipynb (IPython Notebook).

Враховуючи основні особливості, було вирішено використовувати саме цю бібліотеку.

Процес розробки архітектури

Вимоги до системи. Необхідно створити архітектуру, яка дозволить виконувати моделювання ЗНМ для користувача без особливих проблем, реалізувати зрозумілий та простий інтерфейс. Так як майбутньою сферою використання даної системи є використання у навчальному класі для початкового знайомства зі ЗНМ, виникає наступна вимога – можливість використовування кількома користувачами одночасно. Але це не виключає використання системи для більш серйозних задач, таких як знаходження оптимальної архітектури ЗНМ для конкретних наборів даних, вдосконалення тренувальних алгоритмів та інше.

Клієнт-серверна архітектура. Система передбачає клієнт-серверну архітектуру (рис. 2). Моделювання ЗНМ покладено на сервер, де виконуються необхідні обчислення. Користувач працює з системою через web-браузер, що встановлений на персональному комп'ютері. Через web-браузер задаються параметри моделювання, та переглядається отриманий результат. Необхідний доступ до мережі Інтернет.

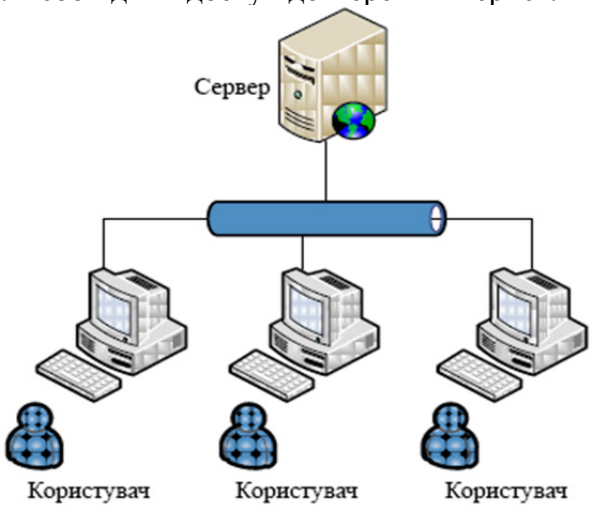

Рис.2. Приклад архітектури

Дана архітектура має такі переваги:

- для роботи користувачу необхідний лише web-браузер;
- відсутність жорстких технічних вимог до комп'ютера користувача;
- модернізація та оновлення програмного продукту відбувається без втручання користувача;
- безпека та цілісність даних покладена на сервер.

Та наступні недоліки:

- повна залежність від підключення до Інтернет;
- обмеженність серверних ресурсів.

Вибір ОС. Враховуючи обрану бібліотеку Pylearn2 для ЗНМ, та компоненти, необхідні для її роботи, операційну систему для сервера було обрано CentOS 6.2 64-bit.

CentOS частіше всього використовують, як серверну операційну систему для веб-хостингу. Багато великих хостингових компаній використовують CentOS для забезпечення стабільної роботи для своїх веб-застосунків. Даний дистрибутив Linux базується на основі комерційного дистрибутиву Red Hat Enterprise Linux компанії Red Hat, що гарантує безпеку і стабільність роботи. Життєвий цикл CentOS складає 10 років, підтримка 6-ої версії буде відбуватись до 2020 року.

Фреймворк для розробки web-системи. Враховуючи, що обрана бібліотека для роботи зі ЗНМ написана на Python, фреймворк був обраний теж написаний на цій мові.

Один з варіантів – Flask, що є мікрофреймворком для створення web-сайтів мовою Python. Основна причина чому Flask називається "мікрофреймворком" - це ідея зберегти ядро простим, але розширюваним. У ньому немає абстрактного рівня бази даних, немає валідації форм або всього такого, що вже є в інших бібліотеках до яких ви можете звертатися. Однак Flask підтримує розширення, які можуть додати необхідну функціональність.

Але вибір був зупинений на Django, що має усі необхідні компоненти «з коробки» та такі основні особливості [11]:

- ORM, API доступу до БД з підтримкою транзакцій;
- вбудований інтерфейс адміністратора, з уже наявними перекладами на більшість мов;
- диспетчер URL на основі регулярних виразів;
- розширювана система шаблонів з тегами та наслідуванням;
- система кешування.

Вибір СКБД. Обрана бібліотека для web-системи Django дозволяє використовувати наступні СКБД: PostgreSQL (8.2 та вище), MySQL (5.0 та вище), SQLite, Oracle Database Server (9i та вище). Оскільки очікуване навантаження на СКБД має бути невеликим та функції покладені на неї не для моделювання ЗНМ, а тільки для забезпечення роботи з web-системою, було обрано найпоширеніше рішення для web-платформ – MySQL 5.5 [12].

Web-сервер. Для обраної ОС розглядалось два найпопулярніші web-сервери: Apache та nginx. Обидва рішення надають надійність та гнучкість у конфігурації. Але головна особливість, в першу чергу в тому, що Apache працює за принципом: новий запит - новий потік; nginx працює з використанням моделі подій, відповідно новий запит не породжує нових потоків. Завдяки цій особливості, web-сервер потребує менше оперативної пам'яті, а швидкість його роботи стає вищою [13].

Gateway Interface HTTP-сервера. Враховуючи обрані вище рішення, в якості Python web-сервера, до якого відбувається звернення з nginx, було вирішено використовувати Gunicorn [14]. Він має наступні особливості:

- підтримка Django з коробки;
- автоматичне управління робочим процесом;
- проста конфігурація Python;
- можливість використовувати кілька конфігурацій;
- різні hooks для розширюваності;
- сумісність з Python 2.x >= 2.5.

Фоновий процес моделювання. Оскільки процес моделювання досить ресурсномісткий, його необхідно виконувати у фоні та асинхронно, бажано навіть на окремому сервері. Тому було обрано бібліотеку ZeroMQ [15].

ZeroMQ (також називають OMQ) – високопродуктивна бібліотека, що призначена для асинхронного обміну повідомленнями. Ця бібліотека призначена для використання в масштабованих розподілених системах чи в паралельних додатках.

Схема отриманої архітектури

На рис. 3. представлена схема отриманої архітектури ПЗ.

Рис.3. Отримана архітектура

Через web-браузер користувач може створювати задачі, що будуть змодельовані. Після створення задачі, вона додається до черги, звідки її на моделювання отримає ZeroMQ. Після моделювання, результати записуються до бази даних, звідки користувач може їх переглядати.

Приклад моделювання

Для перевірки працездатності отриманої архітектури було здійснено наступний експеримент [16].

Тренувальний датасет MNIST – набір написаних від руки цифр, складається з 60 тисяч зображень для тренування, та 10 тисяч для тестування, відповідно це складає 85% та 15% від загального обсягу зображень. Конфігурація шарів: вхідний шар [28, 28], ConvRectifiedLinear (кількість вихідних каналів 64, форма ядра згортки [5, 5], форма вікна для розрідженого max pooling [4, 4], відстань між початком кожного вікна згортки [2, 2], норма ядра згортки 1.9365), ConvRectifiedLinear - дублює параметри попереднього шару, Softmax (норма кожного стовпця матриці ваг 1.9365).

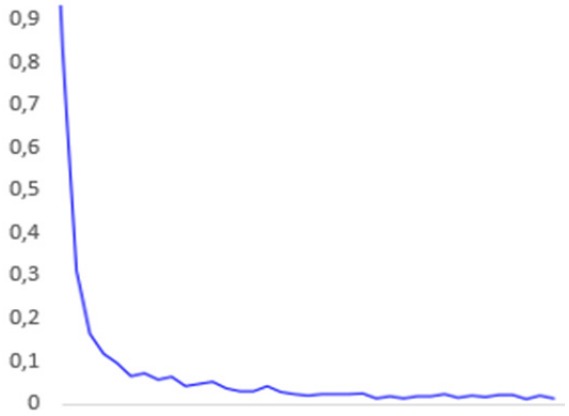

Рис.4. Графік помилки розпізнання

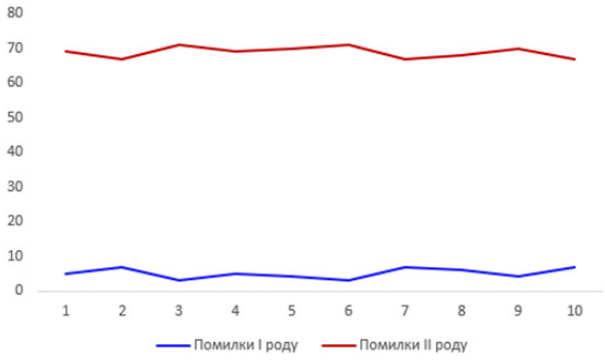

Рис.5. Графік помилки I та II роду

Результат моделювання для тестової вибірки - помилка знизилася до 0.74% (рис. 4). Для 10 тестових вибірок середнє значення помилки I роду складає 6,89% від загальної помилки, а помилки II роду 93,1% (рис. 5).

Висновки

В даній роботі було описано процес розробки архітектури системи для моделювання згорточних нейронних мереж.

Тема даної роботи є актуальною, що підтверджується великою кількістю новин та публікацій за 2012-2013 роки.

На основі виконаного аналізу, було обрано бібліотеку pylearn2, написану на мові Python. Враховуючи поставлену вимоги, обрано допоміжні компоненти для реалізації клієнт-серверної архітектури.

Отримане рішення має переваги у вигляді доступності та простоти у користуванні, але існує недолік, такий як залежність від ресурсів сервера. З іншого боку, сервером може виступати будь-який комп'ютер з ОС Linux, або навіть GPU-кластер.

Список літератури

1. ImageNet Large Scale Visual Recognition Competition 2012 (ILSVRC2012) [Электронний ресурс]. – Режим доступу: http://www.image-net.org/challenges/LSVRC/2012/
2. Команда Джеффри Хинтона победила в конкурсе компьютерного зрения ImageNet с двукратным преимуществом / Хабрахабр [Электронний ресурс]. – Режим доступу: http://habrahabr.ru/post/183380/
3. LeCun Y. A theoretical framework for backpropagation // Proc. of IEEE. - 1998. - P.21-28.
4. LeCun Y., Bottou L., Bengio Y., Haffne P. Gradient-Based Learning Applied to Document Recognition // Proc. IEEE. - 1998. - P.59-67.
5. Дорогий Я.Ю. Модифицированный алгоритм обучения сверточных нейронных сетей. //Сборник материалов. VII Международная научно-практическая конференция "ПЕРСПЕКТИВЫ РАЗВИТИЯ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ". 18 апреля 2012, г. Новосибирск: Издательство НГТУ. – с 34-38.
6. Convolutional nets for digit recognition [16 min] | Neural Networks for Machine Learning [Электронний ресурс]. – Режим доступу: https://class.coursera.org/neuralnets-2012-001/lecture/73
7. Feature extraction using convolution - Ufldl [Электронний ресурс]. – Режим доступу: http://ufldl.stanford.edu/wiki/index.php/Feature_extraction_using_convolution
8. Yann LeCun's Home Page [Электронний ресурс]. – Режим доступу: http://yann.lecun.com
9. Convolutional neural network class - Mikhail Sirotenko's home page [Электронний ресурс]. – Режим доступу: https://sites.google.com/site/mihailsirotenko/projects/convolutional-neural-network-class
10. Welcome - Pylearn2 dev documentation [Электронний ресурс]. – Режим доступу: http://deeplearning.net/software/pylearn2/
11. Using the Django authentication system | Django documentation | Django [Электронний ресурс]. – Режим доступу: https://docs.djangoproject.com/en/dev/topics/auth/default/#topic-authorization
12. MySQL — Вікіпедія [Электронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/MySQL
13. nginx — Вікіпедія [Электронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Nginx
14. Gunicorn (HTTP server) - Wikipedia, the free encyclopedia [Электронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/Gunicorn_%28HTTP_server%29
15. ØMQ - Wikipedia, the free encyclopedia [Электронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/ZeroMQ
16. pylearn2 tutorial: Convolutional network [Электронний ресурс]. – Режим доступу: http://nbviewer.ipython.org/github/lisa-lab/pylearn2/blob/master/pylearn2/scripts/tutorials/convolutional_network/convolutional_network.ipynb