УДК 004.8, 004.032.26, 004.932, 004.054, 004.75 **DOI:** 10.31673/2412-9070.2020.062432

М. Г. КУРОЧКІНА, студентка;
В. В. ЗІЛЬБЕРШТЕЙН, студентка;
І. О. ГАЛУШКО, студент;
Д. С. ТРОЦЕНКО, студент,
Державний університет телекомунікацій, Київ

МЕТОДИ ІНТЕЛЕКТУАЛЬНОГО АНАЛІЗУ ІНФОРМАЦІЇ ЗА ДОПОМОГОЮ ВІРТУАЛІЗАЦІЇ ТА НЕЙРОННИХ МЕРЕЖ

Розглянуто методи, що уможливлюють підвищення ефективності пошуку інформації під час імпорту тексту, а також у процесі аналізу та генерації зображень. Досліджено можливість використання комп'ютерного зору в режимі реального часу та порівняння швидкодії алгоритмів класифікації зображення. Також проаналізовано метод контейнеризації Docker для нейромережі, щоб запобігти перевантаженню пам'яті і ЦП. Запропоновано метод віртуалізації на основі контейнерів, який виконується не повною операційною системою, а в частинних екземплярах операційної системи хоста.

Ключові слова: процес текстового аналізу; методи; комп'ютерний зір; штучні нейронні мережі; машинне навчання; TensorFlow; OpenCV; NAS; GAN; CNN; CTL-10; CIFAR-10; віртуалізація; віртуальна машина; Docker-контейнер; процесор; пам'ять; оцінювання продуктивності.

ВСТУП

Технологія інтелектуального аналізу даних допомагає отримувати корисну інформацію з різних баз даних. Сховища даних зручні для числової інформації, але є неприйнятними для текстової інформації. XXI століття вивело нас за межі лімітованого обсягу інформації в Інтернеті, забезпечивши більшим обсягом інформації, зростанням обізнаності і кращими знаннями. Аналіз текстових даних належить до процесу виокремлення цікавих і нетривіальних шаблонів або знань із текстових документів.

Аналіз тексту — це частина інтелектуального аналізу даних, яка намагається знайти корисну інформацію з великих баз даних. Більшість досліджень із моделювання та реалізації напівструктурованих даних було присвячено розробленню методів пошуку інформації, зокрема методам індексації тексту для оброблення неструктурованих документів.

Окрім цього, саме для аналізу та розпізнавання графічних зображень користуються комп'ютерним зором. Комп'ютерний зір — це робота та результати певних алгоритмів, призначених для отримання, оброблення та аналізування справжніх сцен (зображень, фреймів).

Існують різноманітні методи оброблення та розпізнавання зображень. Ці методи можна реалізувати як програмно, так і апаратно, а результати, здобуті за допомогою комп'ютерного зору, можна застосовувати для прийняття тих чи інших вирішень. Для класифікації зображень було успішно розроблено і оцінено нейронну архітектуру пошуку NAS [1; 2]. Виявлені архітектури перевершують розроблені людиною моделі. Сучасні комп'ютери мають достатню швидкодію для реалізації багатьох алгоритмів із бібліотек цифрових зображень, проте віртуалізація дає можливість керувати такими діями з меншою витратою ресурсів.

Віртуальна машина — це емуляція фізичної системи, на якій користувач може одночасно запускати кілька віртуальних машин із різними гостьовими операційними системами. Але через цю функцію існує проблема перевантаження процесора і пам'яті на віртуальній машині. Щоб подолати зазначену проблему, новий метод під назвою Docker може бути використаний як найкраще вирішення для проблеми з перевантаженням пам'яті і процесора.

Docker забезпечує легку віртуалізацію на системному рівні, розширюючи модель загального контейнера в Linux, звану Linux Containers. Для роботи контейнерів Docker не потрібно окремої операційної системи (ОС), і вона використовує ту саму операційну систему, що і її хост.

У проведеному експерименті створюється один додаток, в якому виконуються процеси завантаження файла входу в систему, завантаження файла та шифрування і дешифрування файла. Це один і той самий додаток, який виконується одночасно на віртуальній машині і в докер-контейнері. Після експерименту адміністратор перевірить продуктивність обох систем.

Аналіз літературних даних та постановка проблеми. Алгоритми пошуку нейронної архітектури NAS спрямовано на відшукання оптимальної архітектури нейронної мережі, а не використовувати створену вручну для конкретного завдання. Попередні роботи на NAS досягли великих успіхів у розв'язанні

задачі класифікації зображень [8], розширили алгоритми NAS до щільного і структурованого прогнозування [9; 10]. Варто зазначити, що NAS також застосовується і для стиснення та прискорення CNN [11]. Однак для генеративних моделей алгоритм NAS ще не розроблено.

Алгоритм NAS складається з трьох ключових компонентів: простір пошуку, алгоритм оптимізації і завдання проксі. Для простору пошуку, як правило, є дві категорії: прямий пошук за всією архітектурою (пошук макросів) та пошук осередків і їх складання заздалегідь визначеним способом (мікропошук). Для алгоритму оптимізації популярними варіантами є навчання з підкріпленням [1; 12; 14; 15], еволюційний алгоритм [16], байєсівська оптимізація [17], випадковий пошук [9] і методи оптимізації на основі градієнта [18; 19]. Для завдання проксі-сервера вона скасовується для ефективного оцінювання продуктивності виявленої архітектури під час навчання. Приклади включають у себе ранню зупинку [9; 15], використання зображень із низьким розрізнянням [8; 21; 22], прогнозування продуктивності за допомогою сурогатної моделі [23], використання невеликої магістралі [20] або використання загальних параметрів [13].

ОСНОВНА ЧАСТИНА

Процес інтелектуального аналізу тексту починається зі збору документів із різних джерел. Інструмент інтелектуального аналізу тексту витягує конкретний документ і попередньо обробляє його, перевіряючи формат і набори символів. Потім документ проходить етап аналізу тексту. Аналіз тексту — це семантичний аналіз для отримання високоякісної інформації з тексту. Сьогодні доступні багато методів аналізу тексту, а залежно від мети організації можуть бути використано комбінації методів. Іноді методи аналізу тексту повторюються доти, поки інформацію не буде видалено (рис. 1).

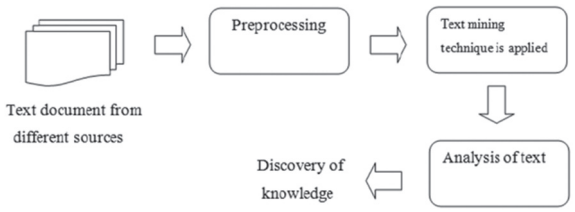

Рис. 1. Процес інтелектуального аналізу тексту

Аналіз тексту та зображень може сильно навантажувати систему, оскільки навантаження — це не що інше, як навантаження процесора, пам'яті, введення-виведення і завантаження смуги пропускання мережі. У домені з різними станами навантаження застосовують іншу стратегію міграції, яка враховує час міграції, потенційну віртуальну машину міграції і місце призначення міграції. Використовуючи цю стратегію, можна зменшити кількість перевантажених хостів, але проблема полягає в тому, що потрібно багато часу, аби визначити, перевантажена машина чи ні [1].

Використання гіпервізора в кластерному середовищі дає змогу консолідувати кілька автономних фізичних машин у віртуалізоване середовище, що потребує менше фізичних ресурсів, ніж будь-коли раніше. Великі хмарні розгортання вимагають тисячі фізичних машин і мегават енергії. Отже, необхідно створити ефективну систему віртуалізації, яка підтримуватиме фізичне навантаження на ЦП із мінімізацією кількості енергії. Таким чином, він розробляє і впроваджує архітектуру з урахуванням енергоспоживання, засновану на зборі інформації про різні ресурси: процесор, оперативну пам'ять, мережу. Але алгоритм планування процесора не настільки сильний і для його виконання потрібен час [4].

Методи інтелектуального аналізу даних

Традиційно існує багато методів, розроблених для вирішення проблеми інтелектуального аналізу тексту, і це не що інше, як пошук відповідної інформації згідно з вимогами користувача. За даними пошуку інформації зазвичай використовують такі методи:

- ♦ метод, заснований на терміні (ТВМ);
- ♦ метод на основі фрази (РВМ);
- ♦ метод таксономічної структури (РТМ).

Метод, заснований на терміні. Термін у документі — це слово, що має семантичне значення. У методі, заснованому на термінах, документ аналізується на основі терміна і має переваги ефективної обчислювальної продуктивності, а також зрілих теорій для зважування терміна. Ці методи з'явилися в останні кілька десятиліть у спільнотах із пошуку інформації та машинного навчання. Методи, засновані на термінах, мають проблеми полісемії та синонімії [4].

Метод на основі фрази. Фраза несе більше семантики як інформація і менш двозначна. У фразеологічному методі документ аналізується за фразою, оскільки фрази менш неоднозначні і більш помітні, ніж окремі терміни [3]. Проте можливі такі причини для низької результативності:

- • фрази мають гірші статистичні властивості щодо термінів;
- • фрази мають низьку частоту виявлення;
- • наявність великої кількості зайвих і галасливих фраз.

Метод таксономії. У шаблонному методі таксономії документи аналізуються на основі шаблонів. Шаблони можуть бути структуровані в таксономії за допомогою відношення *is-a*. Протягом багатьох років патерновий аналіз широко вивчався в спільнотах інтелектуального аналізу даних. Шаблони можуть бути виявлені за допомогою методів інтелектуального аналізу даних, зокрема аналізу правил асоціацій, аналізу наборів частих елементів, послідовного аналізу шаблонів і аналізу закритих шаблонів.

Розпізнавання та оброблення зображень. OpenCV [6] — це бібліотека з набором функцій та алгоритмів загального призначення, необхідних для оброблення зображень. Розпізнавання об'єктів у CV виконується за допомогою колірної сегментації зображення. Будь-яке зображення в сірих відтінках може розглядатися як топографічна поверхня, де висока інтенсивність позначає вершини та пагорби, а низька інтенсивність — долини.

Тому OpenCV є алгоритмом вододілу на основі маркерних позначок, де потрібно вказати, які точки долини підлягають об'єднанню, а які — ні. Це називається інтерактивною сегментацією зображення, де необхідно надати позначки відомим об'єктам. Для застосування інтерактивної сегментації потрібно позначити регіон, який точно є переднім планом або об'єктом одного кольору (або інтенсивності), далі необхідно позначити регіон, який точно є фоном або «не об'єктом» із іншим кольором, і, нарешті, регіон, значення якого невідоме, потрібно позначити нулем.

Алгоритми пошуку нейронної архітектури. Більшість алгоритмів NAS [1] генерують мережну архітектуру (макропошук) або елемент (мікропошук) за один прохід контролера. У [3] було введено багаторівневий пошук у NAS в завданні класифікації зображень із використанням пошуку променя. Відшукування в архітектурі починається з меншого осередка, і найбільш ефективні кандидати будуть збережені. На їх основі продовжується наступний раунд пошуку для більшого осередка.

Генеративна змагальна мережа. Генеративні змагальні мережі GAN є інновацією у машинному навчанні. Вони належать до генеративних моделей і створюють нові дані, які схожі на набір тренувальних даних.

У системі GAN одна з мереж «Generator» генерує зразки, а інша «Discriminator» намагається відрізнити справжні зразки від неправильних. Генеративна мережа намагається згенерувати новий зразок, змішавши кілька вихідних зразків. Дискримінативна мережа навчається розрізняти справжні і згенеровані зразки, а метою мережі є збільшення частоти помилок дискримінатора через генерацію синтезованих зображень, схожих на справжні.

Практичні дослідження

Для аналізу роботи було використано:

- TensorFlow для створення моделі;
- Docker для контейнеризації;
- TensorFlow Serving для хостингу моделі.

TensorFlow — це бібліотека з відкритим вихідним кодом для розроблення моделей машинного навчання та особливо моделей глибокого навчання. Вона створена і підтримується Google і призначена не тільки для наукових цілей, а й для розробки продуктів.

Docker — це дуже популярна технологія контейнеризації, яка надає зручний спосіб перенесення контейнерів для їх локального розгортання або в хмарі.

TensorFlow Serving — це гнучка, високопродуктивна система обслуговування моделей машинного навчання, розроблена для виробничих середовищ. TensorFlow Serving дає можливість легко розгортати нові алгоритми і експерименти, зберігаючи ту саму архітектуру сервера і API-інтерфейси.

TensorFlow Serving реалізує сервер, який обробляє вхідні запити і направляє їх у модель. Цей сервер може бути запущено в хмарі. Нині досить поширено згортати такий сервер і весь його функціонал у пакет, конфігурувати його і розгортати цей пакет.

Зазвичай використовуються контейнери для створення розгорнутих артефактів. Потім можна розгорнути їх всюди — локально, в Інтранеті, у хмарі.

Найпопулярніша контейнерна платформа — Docker. Потрібно створити образ Docker, створити контейнер для цього образу і запустити його.

Отже, далі ми маємо лише перевірити, що контейнер працює, а сервер, наданий TeinsorFlow, успішно запускається та приймає запити до нашої моделі і відповіді на них.

Для тестування нами було використано відеокарту NVIDIA RTX 2080 Ti на операційній системі Ubuntu версії 18.04. tf_cnn_benchmarks містить реалізації кількох популярних згорткових моделей і розроблено так, щоб бути максимально швидким; tf_cnn_benchmarks підтримує запуск на одному комп'ютері або в розподіленому режимі на кількох хостах (рис. 2).

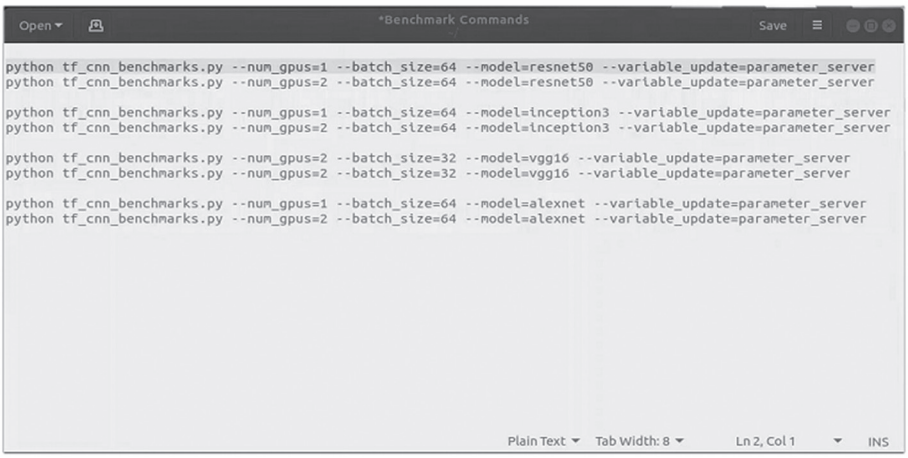

Рис. 2. Деякі використані команди для тестування

Зазначені моделі застосовують багато стратегій щодо керівництва з продуктивності **TensorFlow** (рис. 3, 4).

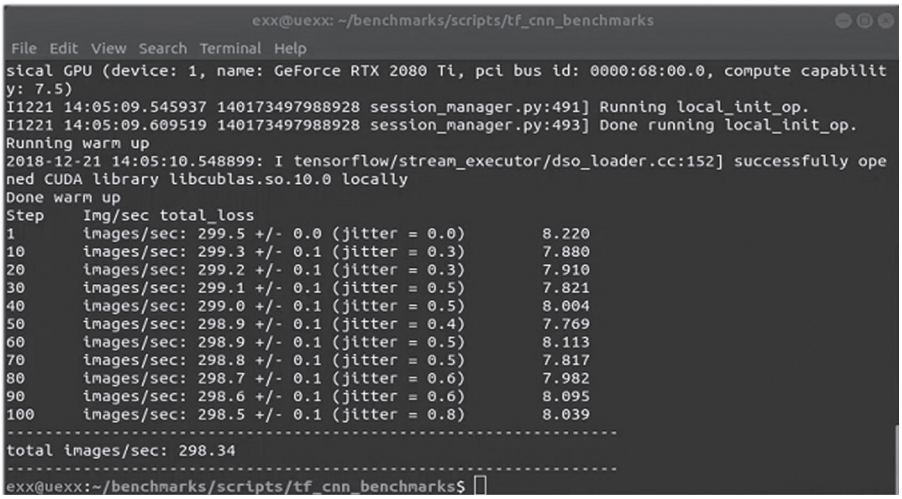

Рис. 3. Генерація зображень тільки з TensorFlow

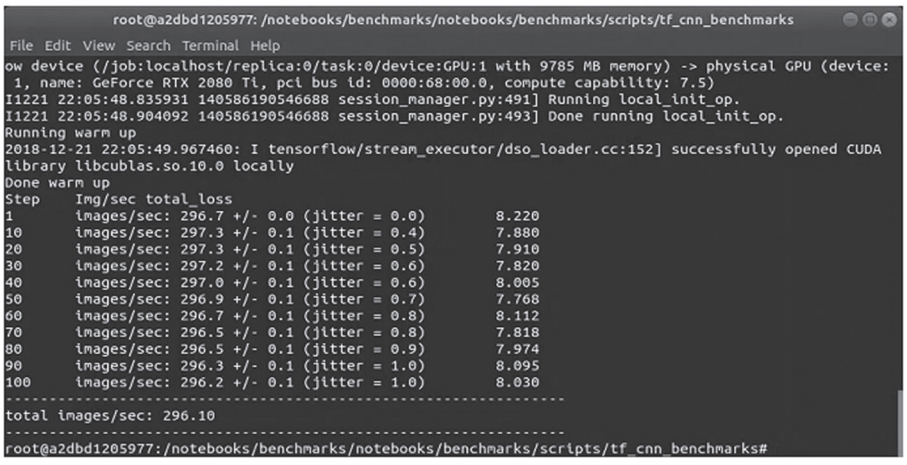

Рис. 4. Образ докера з такою самою версією TensorFlow

Інтерфейс керування системою **NVIDIA** (*nvidia-smi*) — це утиліта командного рядка, заснована на базі бібліотеки керування NVIDIA (NVML), призначена для допомоги в керуванні і моніторингу пристроїв NVIDIA GPU (рис. 5).

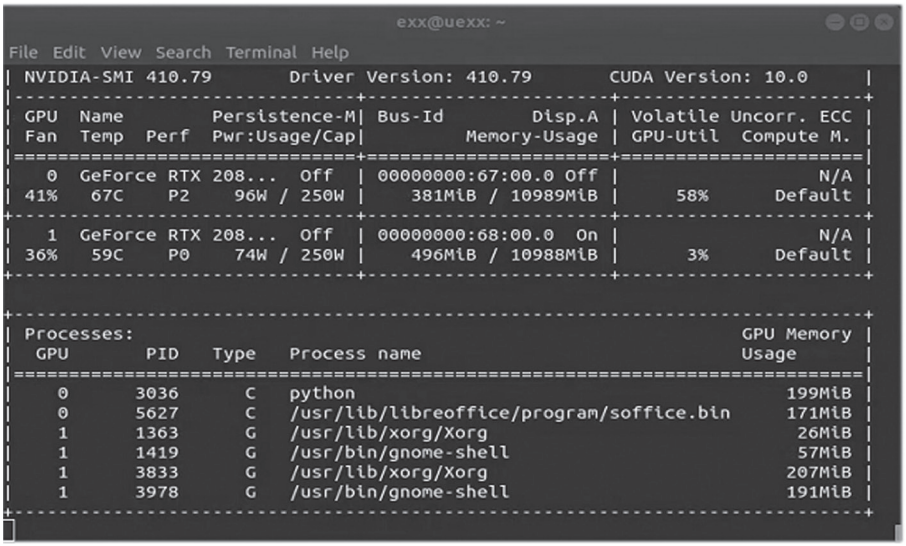

Рис. 5. Вікно NVIDIA-SMI

Після встановлення потрібних бібліотек і ПО було проведено тестування продуктивності запуску алгоритмів машинного навчання (табл. 1).

Тест №1: застосували один графічний процесор і розмір пакета 64, модель нейромережі — ResNet50.

Тест №2: використали два графічних процесора і розмір пакета 64, модель нейромережі — ResNet50.

Тест №3: один графічний процесор і розмір пакета 64, модель нейромережі — Inception3.

Тест №4: два графічних процесора і розмір пакета 64, модель нейромережі — Inception3.

Тест №5: два графічних процесора і розмір пакета 32, модель VGG16.

Тест №6: один графічний процесор і розмір пакета 32, модель VGG16.

Тест №7: один графічний процесор і розмір пакета 64, модель AlexNet.

Тест №8: два графічних процесора і розмір пакета 64, модель AlexNet.

Тест №9: один графічний процесор і розмір пакета 64, модель DCGAN.

Тест №10: два графічних процесора і розмір пакета 64, модель DCGAN.

Тест №11: застосування Ganbreeder із докером і базами даних PostgreSQL; один графічний процесор, модель BigGAN

Тест №12: один графічний процесор і розмір пакета 64, модель AutoGAN.

Таблиця 1

Тестування продуктивності алгоритмів машинного навчання

Номер тесту	Без докера, зображень/с	Із докером, зображень/с
Тест №1	298,33	296,09
Тест №2	552,69	548,21
Тест №3	196,14	195,16
Тест №4	364,86	356,9
Тест №5	228,69	229,34
Тест №6	169,16	168,82
Тест №7	2720,73	2702,69
Тест №8	1511,85	1507,02
Тест №9	2116,64	2101,84
Тест №10	1019,33	989,96
Тест №11	2980,3	1061,69
Тест №12	974,87	966,59

Подамо деякі результати оброблення даних (рис. 6, 7).

Рис. 6. DCGAN із датасетом CIFAR-10

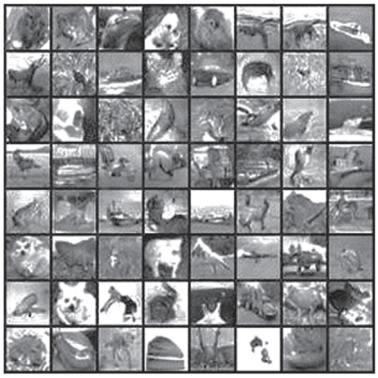

Рис. 7. Результати CIFAR-10 AutoGAN

BigGAN — це підхід, що дає можливість зібрати разом набір кращих новітніх практик зі збільшення розмірів пакетів і кількості параметрів моделі. Результатом є генерація великих і високоякісних зображень (рис. 8).

Рис. 8. BigGAN

CIFAR-10 і **CIFAR-100** є позначеними підмножинами набору даних із 80 млн крихітних зображень. Їх збирали Алекс Крижевський, Вінод Наїр і Джеффрі Хінтон.

Для завдання беззастережної генерації зображень CIFAR-10 (не використовуються позначки класу) можна було б узагальнити низку помітних спостережень:

• виявлена архітектура має три блоки згортки. AutoGAN чітко віддає перевагу блоку згортки перед активацією, ніж блок згортання після активації.

• AutoGAN дуже віддає перевагу деконволюції найближчого сусіда (також білінеарного взірця). Це збігається з попереднім досвідом [5], що деконволюція може спричинити артефакти на борту, і найближчий сусід може бути лояльним.

• AutoGAN явно виступає за (щільніші) пропускні з'єднання. Тим більш, він здатний виявляти середні та довгі пропуски (пропускаючи 2-4 згорткових шари), досягаючи багатомасштабного синтезу.

Також визначено бал Inception та FID бали для найкращих двох та найкращих трьох виявлених архітектур у нижній частині табл. 2. Їх відповідні архітектури ілюструє рис. 4. Отже бачимо, що всі три найкращих виявлених архітектури досягають конкурентоспроможних характеристик за допомогою сучасних моделей.

Таблиця 2

Початковий бал та оцінка FID беззастережного завдання створення зображення на CIFAR-10

Метод	Початковий бал	FID
DCGAN	6,64 ± ,14	–
Improved GAN	6,86 ± ,06	–
LRGAN	7,17 ± ,17	–
DFM	7,72 ± ,13	–
ProbGAN	7,75	24,60
WGAN-GP, ResNet	7,86 ± ,07	–
Splitting GAN	7,90 ± ,09	–
SN-GAN	8,22 ± ,05	21,7 ± ,01
MGAN	8,33 ± ,10	26,7
Dist-GAN	–	17,61 ± ,30
Progressive GAN	8,80 ± ,05	–
Improving MMD GAN	8,29	16,21
AutoGAN-top1	8,55 ± ,10	12,42
AutoGAN-top2	8,42 ± ,07	13,67
AutoGAN-top3	8,41 ± ,11	13,87

Віртуалізація через контейнер. Продуктивність контейнера Docker із зображеним іменем користувача, різними ролями, виконуваними користувачем, часом, процесором і станом пам'яті зображено на рис. 9. Продуктивність процесора зазначено у відсотках, а продуктивність пам'яті — у мегабайтах. Тут користувач запускає один файл, для якого виконує завантаження, шифрування і дешифрування файла на контейнері. Для кожної виконуваної ролі обчислюється відсоток використання процесора і стану пам'яті. Такий самий додаток і файл буде застосовуватися для перевірки продуктивності віртуальної машини.

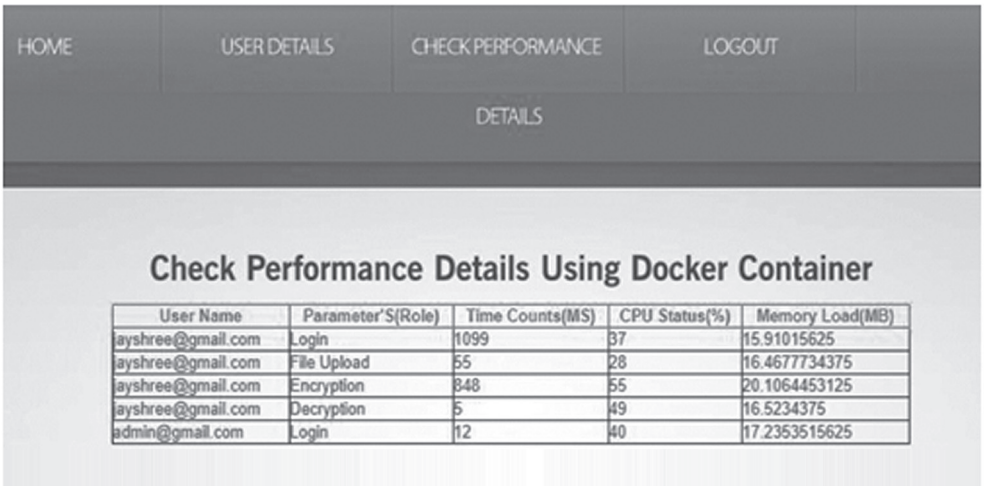

Рис. 9. Продуктивність контейнера Docker

Продуктивність віртуальної машини. Продуктивність віртуальної машини із зображенням імені користувача, різних ролей, що виконуються користувачем, часу, процесора та статусу пам'яті подано на рис. 10. Продуктивність процесора задається у відсотках, а продуктивність пам'яті зазначено в мегабайтах. Тут користувач запускає один файл, для якого здійснює завантаження, шифрування та розшифрування файла на контейнері docker. Для кожної виконаної ролі розраховується відсоток використання процесора та стану пам'яті.

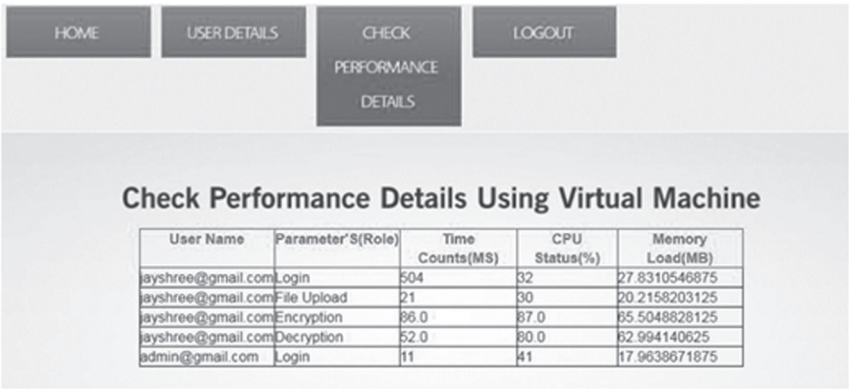

Рис. 10. Продуктивність віртуальної машини

Порівнюючи продуктивність контейнера докера і продуктивність віртуальної машини, доходимо висновку, що максимальний параметр завантаження процесора і пам'яті більший на віртуальній машині, тоді як докер показує менше завантаження процесора і пам'яті.

ВИСНОВКИ

У статті подано методи і складні питання щодо інтелектуального аналізу тексту та зображень. Також розглянуто найбільш складну проблему в розробленні систем інтелектуального аналізу тексту. Обговорюються три методи аналізу тексту на основі термінів, фразеології і модель таксономії. Підхід, заснований на термінах, страждає від багатозначності і синонімії, тоді як підхід, заснований на фразі, працює краще, оскільки фраза несе в собі більше семантики, зокрема інформації, і менш неоднозначна.

В аналізі зображень AutoGAN показала себе більш складною моделлю, ніж NAS, для класифікації зображень через високу нестабільність і гіперпараметричну чутливість самого тренування GAN. На початковому етапі AutoML може проектувати тільки невеликі нейронні мережі, які працюють нарівні з нейронними мережами, розробленими фахівцями-людьми, і ці результати було обмежено невеликими академічними наборами даних, такими як CIFAR-10 і Penn Treebank [1; 7]. Проте, незважаючи на попередні успіхи, у AutoGAN, безсумнівно, є великі можливості для вдосконалення.

Експеримент проводився на двох різних системах віртуалізації. При цьому система, заснована на контейнері Docker, випереджала порівняно з системою віртуальної машини щодо продуктивності процесора та пам'яті. Віртуальна машина має всю свою гостьову операційну систему, включаючи власну пам'ять, завдяки якій вона збільшує накладні витрати на фізичну систему. Оскільки Docker не використовує жодної гостьової операційної системи, а контейнер використовує ядро операційної системи хоста для запуску програми через цю здатність, зменшується завантаження процесора та пам'яті.

Список використаної літератури

*1. **Barret Zoph, Quoc V. Le.** Neural architecture search with reinforcement learning / arXiv preprint arXiv:1611.01578, 2016.*

*2. **Advanced** Memory Reusing Mechanism for Virtual Machines in Cloud Computing. Gursharan Singh' Sunny Behal' Monal Taneja 3rd International Conference on Recent Trends in Computing 2015 (ICRTC-2015).*

*3. **Salton G., Buckley C.** Term-Weighting Approaches in Automatic Text Retrieval, Information Processing and Management // An Int'l J. 2018.*

*4. **Applying** Data Mining Techniques for Descriptive Phrase Extraction in Digital Document Collections / H. Ahonen, O. Heinonen, M. Klemettinen, A. I. Verkamo. 2015. P. 2–11.*

*5. **Wu S.-T., Li Y., Xu Y.** Deploying Approaches for Pattern Refinement in Text Mining // Proc. IEEE Sixth Int'l Conf. Data Mining (ICDM '06). 2016. P. 1157–1161.*

*6. **The** OpenCV Tutorials [Електронний ресурс]. URL: docs.opencv.org/2.4/opencv_tutorials.pdf.*

*7. **Efficiency** analysis of provisioning Microservices / H. Khazaei, C. Barna, N. Beigi-Mohammadi, M. Litoiu // School of Computer Science, York University Toronto, Ontario, Canada.*

*8. **Krizhevsky A., Sutskever I., Hinton G. E.** Imagenet classification with deep convolutional neural networks. 2012. P. 1097–1105.*

*9. **Chen L.-C., Collins M., Zhu Y.** Searching for efficient multiscale architectures for dense image prediction.*

10. ***Liu C., Chen L.-C., Schroff F.*** *Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation.*

11. ***Amc:*** *Automl for model compression and acceleration on mobile devices / Yihui He, Ji Lin, Zhijian Liu [et al.] // In European Conference on Computer Vision, Springer, 2018. P. 815–832.*

12. ***Designing*** *neural network architectures using reinforcement learning / Bowen Baker, Otkrist Gupta, Nikhil Naik, Ramesh Raskar. 2016.*

13. ***Efficient*** *Neural Architecture Search via Parameter Sharing / H. Pham, M. Y. Guan, B. Zoph, Q. V. Le. 2018.*

14. ***Practical*** *blockwise neural network architecture generation / Zhao Zhong, Junjie Yan, Wei Wu [et al.]. 2018. P. 2423–2432.*

15. ***Learning*** *Transferable Architectures for Scalable Image Recognition / Barret Zoph, Vijay Vasudevan, Jonathon Shlens, Quoc V. Le. CVPR, 2018.*

16. ***Xie L., Yuille A.*** *Genetic cnn // IEEE International Conference on Computer Vision (ICCV). 2017. P. 1388–1397.*

17. ***Jin H., Song Q., Hu X.*** *Auto-keras: Efficient neural architecture search with network morphism, 2018.*

18. ***Liu H., Simonyan K., Yang Y.*** *Darts: Differentiable architecture search, 2018.*

19. ***Ahmed K., Torresani L.*** *MaskConnect — Connectivity Learning by Gradient Descent. ECCV, cs. CV, 2018.*

20. ***Chen L.-C., Papandreou G.*** *Rethinking atrous convolution for semantic image segmentation, 2017.*

21. ***Wang Z., Chang S., Yang Y.*** *Studying very low resolution recognition using deep networks. 2016. P. 4792–4800.*

22. ***Chrabaszcz P., Loshchilov I., Hutter F.*** *A downsampled variant of imagenet as an alternative to the cifar datasets: arXiv preprint arXiv: 1707.08819, 2017.*

23. ***Liu C., Zoph B., Neumann M.*** *Progressive Neural Architecture Search. 2018. P. 19–34.*

М. Г. Курочкина, В. В. Зильберштейн, И. О. Галушко, Д. С. Троценко

МЕТОДЫ ИНТЕЛЛЕКТУАЛЬНОГО АНАЛИЗА ИНФОРМАЦИИ С ПОМОЩЬЮ ВИРТУАЛИЗАЦИИ И НЕЙРОННЫХ СЕТЕЙ

Рассмотрены методы, которые позволяют повысить эффективность поиска информации при импорте текста, а также при анализе и генерации изображений. Исследована возможность использования компьютерного зрения в режиме реального времени и сравнения быстродействия алгоритмов классификации изображения. Также проанализирован метод контейнера Docker для уменьшения перегрузки памяти и ЦП. Предложенный метод является виртуализацией на основе контейнеров, который выполняется не в полной операционной системе, а в частичных экземплярах операционной системы хоста.

Ключевые слова: процесс текстового анализа; методы; компьютерное зрение; искусственные нейронные сети; машинное обучение; TensorFlow; OpenCV; NAS; GAN; CNN; CTL-10; CIFAR-10; виртуализация; виртуальная машина; Docker-контейнер; процессор; память; оценка производительности.

M. G. Kurochkina, V. V. Silberstein, I. O. Halushko, D. S. Trocenko

METHODS OF INTELLECTUAL ANALYSIS OF INFORMATION BY VIRTUALIZATION AND NEURAL NETWORKS

The use of information and knowledge extracted from a large amount of data benefits many applications, such as market analysis and business management. In addition, a lot of information is transmitted in graphical form. In this article, we will consider the following successful methods that can improve the efficiency of information retrieval when importing text and also when analyzing and generating images. In addition, the possibility of using computer vision in real time and comparing the performance of image classification algorithms will be investigated. A Docker container method is also proposed to reduce memory and CPU overload. The proposed method is container-based virtualization, which is performed not in the full operating system, but in private instances of the host operating system. Tests were presented to show performance differences of two popular methods of deploying the TensorFlow framework for deep learning. The methods of deploying are TensorFlow GPU from a docker container, and native source version of TensorFlow. Popular neural network models were used to test the performance, such as: ResNet 50, InceptionV3, VGG16, AlexNet, DCGAN, BigGAN and AutoGan. These findings indicate that TensorFlow GPU from a docker method is reliable and there is no lack of performance of Docker for deep learning. Furthermore, we examine the results of different types on neural network architectures, and show how the use of containerization for deep learning can be beneficial for developers and researchers.

Keywords: text analysis process; methods; computer vision; artificial neural networks; machine learning; TensorFlow; OpenCV; NAS; generative adversarial network; AutoGAN; BigGAN; CNN; ResNet 50; InceptionV3; VGG16; AlexNet; CTL-10; CIFAR-10; virtualization; virtual machine; Docker container; processor; memory; performance evaluation.