DOI: 10.18372/2310-5461.45.14572

Т. В. Холявкіна, канд. техн. наук, доц.
Національний Авіаційний Університет
orcid.org/0000-0003-2595-9405
e-mail: holyavkina.t@gmail.com;

Я. О. Резаєв
Національний Авіаційний Університет
orcid.org/0000-0003-0778-2773
e-mail: yar.re.slav@gmail.com;

О. О. Харченко
Національний Авіаційний Університет
orcid.org/0000-0002-8044-428X
e-mail: karasickdev@gmail.com

СИСТЕМА РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ З НЕЙРОМЕРЕЖЕВОЮ АРХІТЕКТУРОЮ НА ОСНОВІ ТЕХНОЛОГІЇ ГЛИБИННОГО НАВЧАННЯ

Вступ

Технології розпізнавання зображень мають значний потенціал майже у будь якій сфері нашого повсякденного і професійного життя.

Метод машинного навчання дозволяє розпізнавати зображення. Суть методу полягає у описі алгоритмів та статистичних моделей, які комп'ютерні системи використовують для виконання конкретного завдання без використання чітких інструкцій, спираючись на закономірності та умовиводи.

Машинне навчання розглядається як підмножина штучного інтелекту (ШІ). Алгоритми машинного навчання будують математичну модель на основі вибіркових даних, відомих як «дані тренінгу», для того, щоб приймати прогнози чи рішення, не будучи явно запрограмованими для виконання завдання [1].

Алгоритми машинного навчання мають широкий спектр різноманітних застосувань, таких як фільтрування електронної пошти та комп'ютерний зір, де складно або неможливо розробити звичайний алгоритм для ефективного виконання завдання [2].

Глибинне навчання — навчання (також відоме як глибоке структуроване навчання або диференційоване програмування) є частиною широкого сімейства методів машинного навчання, заснованого на штучних нейронних мережах з представницьким навчанням. Навчання може здійснюватися під наглядом (контролем), напівконтролем або без нагляду [3; 4].

Розпізнавання зображень — це ідентифікація та виявлення об'єктів чи ознак у цифровому зображенні чи відео.

Для ідентифікації та виявлення зображень використовується технологія машинного зору, що працює з використанням системи штучного інтелекту.

Щоб алгоритм знав, що містить зображення, його потрібно навчити вивчати відмінності між класами. Наприклад, якщо метою системи розпізнавання зображень є виявлення та ідентифікація собак, алгоритм розпізнавання зображень повинен бути навчений тисячам зображень собак та тисячам зображень фонів, які не містять жодної собаки [5].

Аналіз останніх досліджень і публікацій

Аналіз останніх публікацій показав, що інвестиції в ШІ зростають з кожним роком.

Венчурні капіталісти вклали майже 40 млрд доларів США в компанії ШІ та машинного навчання в 2018 р., приблизно вдвічі більше, ніж у 2017 р. (рис. 1) [6].

Рис. 1. Графік інвестицій у галузі штучного інтелекту та машинного навчання

Технології розпізнавання зображень мають значний потенціал у сфері електронної комерції та Інтернет-торгівлі.

Хоча соціальні медіа можуть бути «золотою копальнею» інформації для дистриб'юторів, все ж знадобиться правильний інструмент, щоб скористатися нею. Завдяки мільйонам дописів членів цільової аудиторії, часто можна дізнатися, що цим покупцям подобається, чи що вони хочуть від продуктів, які купують та використовують. Завдяки здатності комп'ютерного зору швидко обробляти велику кількість візуальних даних, це може бути ідеальний інструмент аналізу даних.

Мати ці дані «під рукою» було б неоціненним досягненням для дистриб'юторів, оскільки компанії могли б точніше відрегулювати свій асортимент, гарантуючи, що він відповідає сучасним вимогам їх цільового ринку. Це, в свою чергу, може допомогти залучити нових покупців до свого бренду, які потім будуть підтримувати бренд, оскільки вони бачитимуть, що їхні смаки задовольняються.

У той же час, хоча забезпечення відповідності товарного запасу перевагам цільового ринку і є дуже важливим, підприємства електронної комерції не побачать прибутку, якщо покупці не зможуть знайти свою продукцію. Щоб переконатися в тому, що їх продукція виявлена, компанії знають, що вони повинні використовувати правильні ключові слова у своїх тегах та метаданих. Але це не просте досягнення. Незважаючи на те, що покупці можуть дотримуватися одних і тих же тенденцій, вони не завжди використовують однакові слова для їх опису.

Це ще одна причина, чому візуальне розпізнавання є вагомою інвестицією. Можна швидко навчити моделі комп'ютерного зору бачити та застосовувати до одного об'єкта багато різних концепцій або тегів, охоплюючи всі пошукові терміни, які покупці можуть використовувати, шукаючи товари для придбання (навіть іншими мовами!), Використання комп'ютерного бачення для позначення зображень дає можливість швидко зробити теги продуктів і метадані всеохоплюючими, щоб процес пошуку компанії клієнтом був органічним і легким [7].

Дистриб'ютори, які інвестують у візуальний пошук, можуть очікувати значної віддачі від цієї інвестиції. У той час як *Pinterest* повідомляв [8], що декор та роздрібна торгівля різного типу — це дві найбільші категорії роздрібної торгівлі, для яких їхні клієнти використовували візуальний пошук. Інше дослідження передбачає, що будь-який дистриб'ютор, який переробив свої веб-сайти, щоб забезпечити можливість візуального пошуку, зможе побачити, що його дохід від електронної комерції збільшиться на 30 % до 2021 р. [9].

Переважна більшість досліджень, пов'язаних з розпізнаванням тексту розглядає завдання OCR (*Optical character recognition*). Хоча розпізнавання окремих літер у нашому випадку, як і цифр з набору MNIST насправді не вважається завданням OCR, оскільки воно розглядає лише один символ (літеру або цифру) в один момент часу і лише обмежену кількість символів сумарно, неможливо писати про OCR і не включати цей приклад. Однак саме це може натякати, чому OCR вважається легким. Окрім вищезазначеного, у деяких підходах для вирішення завдання оптичного розпізнавання символів кожна літера виявляється окремо, і тоді класифікаційні моделі типу MNIST стають актуальними.

Розпізнавання тексту — це переважно завдання, що складається з двох кроків. На першому етапі необхідно виявити включення тексту на зображенні, будуть вони щільними (як у друкованому документі) або розрідженими (як текстові елементи у повсякденному житті).

Виявивши рівень рядка/слова, ми можемо знову вибирати із великого набору рішень, які зазвичай виходять із трьох основних підходів:

- класичні методи комп'ютерного зору (CV);
- спеціалізоване глибинне навчання;
- стандартний підхід глибинного навчання (виявлення).

Розглянемо приклади реалізації кожного з методів.

Класичні методи комп'ютерного зору

Як було сказано раніше, комп'ютерний зір вирішує різні проблеми розпізнавання тексту.

Класичний CV-підхід, як правило, базується на таких кроках:

1. Застосування фільтрів для виділення символів з фону.

2. Застосування виявлення контуру для розпізнавання символів один за одним.

3. Застосування класифікації зображень для ідентифікації символів

Зрозуміло, якщо друга частина виконана добре, третя частина буде простою з використанням узгодження зразків або машинного навчання (приклад MNIST).

Однак виявлення контуру є досить складним для узагальнення. Це потребує багато ручної тонкої настройки, тому стає нездійсненним у більшості задач.

В одному простому прикладі з використанням OpenCV-Python для реалізації методів KNN та SVM [10] використовувались цифри одного типу та розміру, тому було досягнуто 100 % точності.

Розглянемо більш складний приклад [11] з використанням скрипту [12] для розпізнавання

деяких зображеннях із набору даних SVHN. При першій спробі (рис. 2) було досягнуто дуже хороших результатів:

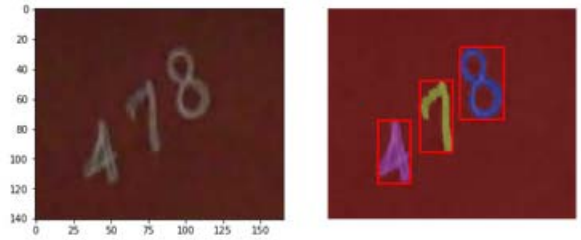

Рис. 2. Перша спроба розпізнавання
зі сприятливими умовами

Але коли символи ближче один до одного (рис. 3), все починає ламатися:

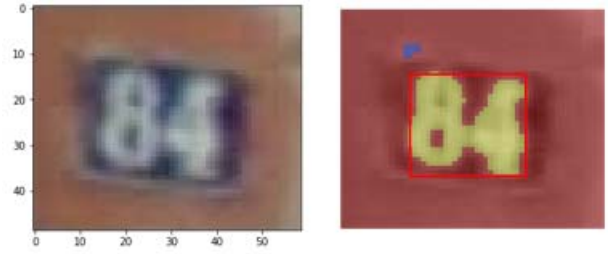

Рис. 3. Спроба розпізнавання
з несприятливими умовами

Автор виявив, що якщо почати налаштовувати параметри, можна зменшити такі помилки, але, будуть виникати інші. Отже, якщо це складне, необхідні більш ефективні методи.

Спеціалізовані підходи глибинного навчання

Більшість успішних підходів глибинного навчання відрізняються своєю загальністю. Однак, ураховуючи описані вище атрибути, спеціалізовані мережі можуть бути дуже корисними.

EAST (ефективний точний детектор тексту сцени) — це простий, але потужний підхід для виявлення тексту. Базується на використанні спеціалізованої мережі [13].

На відміну від інших методів, обмежуються лише виявленням тексту (не фактичним розпізнаванням), однак його експлуатаційна надійність є вагомою підставою для згадки про нього.

Ще одна перевага полягає в тому, що він також був доданий до бібліотеки open-CV (з версії 4), що полегшує його використання.

Мережа — фактично відома U-Net, яка добре підходить для виявлення ознак, які можуть відрізнятися за розміром.

Базовий прямозв'язковий «стовбур» (рис. 4) цієї мережі може відрізнятися — PVANet використовується в дослідженні, однак реалізація open-CV використовує *Resnet*. Очевидно, мережа також може бути попередньо натренованою (наприклад, за допомогою *imagenet*). Як і в U-Net, ознаки екстрагуються з різних рівні у мережі.

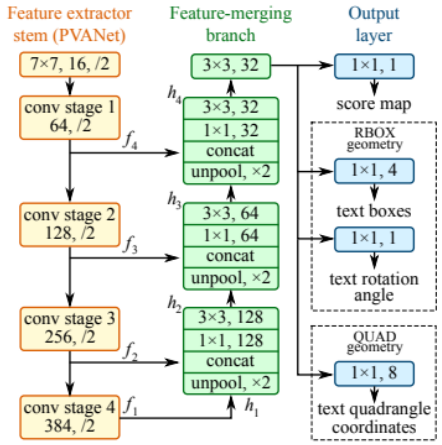

Рис. 4. Структура розпізнавання тексту FCN

Мережа дозволяє використовувати два типи виходів обмежувальних рамок з поворотом: стандартна обмежувальна рамка з кутом повороту (2×2 + 1 параметри), або «чотирикутник», який є обмежувальною рамкою з невеликим кутом повороту та координатами всіх вершин (рис. 5).

Рис. 5. Види обмежувальних рамок

Якщо реальні результати будуть такими, як у наведених вище рисунках, розпізнавання текстів не потребуватиме великих зусиль. Однак результати реального світу не є ідеальними.

CRNN. Конволюційно-рекурента нейронна мережа — описана у статті 2015 р., у якій пропонується гібридна неперервна архітектура, яка призначена для захоплення слів, підходом у три кроки.

Ідея полягає в такому: перший рівень — це стандартна повністю конволюційна мережа. Останній шар мережі визначається як особливий шар і поділяється на «стовпці». Кожен стовпець ознаки призначений для відображення певного розділу в тексті (рис. 6).

Після цього, стовпці ознак подаються у глибинний двонаправлений LSTM (рис. 7), який виводить послідовність, і призначений для пошуку зв'язків між символами. Третя частина є шаром транскрипції. Її мета — узяти безладну послідовність символів, у якій одні символи є зайвими, а інші — порожніми, і використовувати ймовірнісний метод для її уніфікації та осмислення.

Цей метод називається втратою CTC. Цей шар можна використовувати з, або без попередньо визначеного лексикону, що може полегшити передбачення слів.

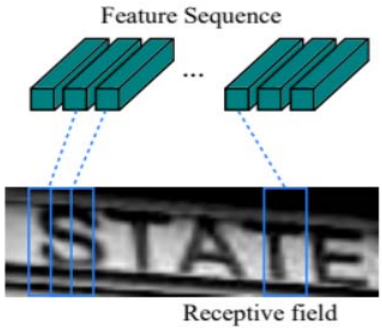

Рис. 6. Схема представлення тексту стовпцями ознак

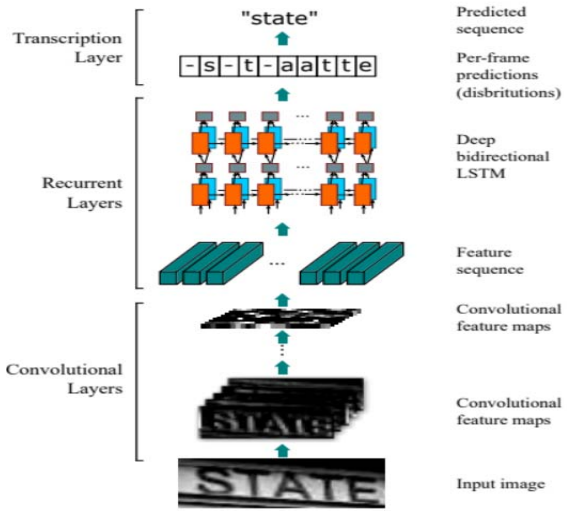

Рис.7 . Загальна схема мережі

Такий метод забезпечує високу точність (більше 95 %) з фіксованим текстовим лексиконом та різними показниками успішності без нього.

STN-net/SEE. SEE — Напівконтрольоване неперервне розпізнавання тексту розглянуто у дослідженнях Крістіана Барці [14]. Він та його колеги застосовують дійсно неперервну стратегію для виявлення та розпізнавання тексту. Вони використовують дуже слабкий нагляд (який вони називають напівнаглядом, в іншому значенні, ніж зазвичай). Оскільки вони навчають мережу лише з текстовими примітками (без обмежувальних рамок). Це дозволяє їм використовувати більше даних, але робить їх тренувальну процедуру досить складною, і вони обговорюють різні хитрощі, щоб змусити її працювати, наприклад, не тренувати мережу на зображеннях з більш ніж двома рядками тексту (принаймні на перших етапах навчання).

У статті є більш рання версія, яка називається STN OCR [15]. У підсумковому дослідженні автори вдосконалили свої методи та представлення, а також вони зробили більше акценту на загальності свого підходу зважаючи на високу якість результатів.

Назва STN-OCR натякає на стратегію використання просторового трансформатора (= STN, немає ніякого відношення до нещодавнього трансформатора Google).

Автори тренують дві з'єднані мережі, у яких перша мережа, трансформатор, вивчає перетворення на зображенні, щоб вивести простіший інтерпретаційний субобраз. Потім ще одна мережа з прямим зв'язком вперед з LSTM вгорі для розпізнавання тексту.

У цьому дослідженні наголошується на важливості використання ResNet (вони використовують її двічі), оскільки вона забезпечує «сильне» поширення на ранні шари, однак ця практика сьогодні досить прийнята.

Стандартний підхід глибинного навчання

Після виявлення «слів» можна застосувати стандартні підходи до виявлення глибокого навчання, такі як SSD [16], YOLO та Mask RCNN. Однак SSD та інші моделі виявлення мають певні труднощі коли ідеться про щільні, подібні класи.

Найбільш ефективні рішення, що використовують для розпізнавання зображень базуються на CNN — конволюційних (згорткових) нейромережах. Такого самого висновку дійшла велика кількість дослідників, зокрема у статтях [17] та [18]. Саме тому, система що розглядатиметься в цій роботі має нейромережеву архітектуру.

Модель YOLO вперше була описана у праці [19]. Підхід включає одну нейронну мережу, тренування якої відбувається неперервно і яка використовує фотографію як вхідні дані та прогнозує обмежувальні рамки та мітки класів для кожного обмежувального вікна безпосередньо. Ця методика пропонує меншу точність прогнозування (наприклад, більше помилок локалізації), хоча працює зі швидкістю 45 кадрів у секунду і до 155 кадрів у секунду для швидкісно-оптимізованої версії моделі (рис. 8).

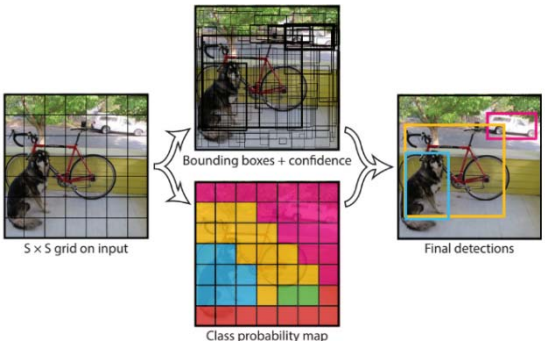

Рис. 8. Результат передбачень моделі YOLO

Реалізацію моделі [20] було натреновано на наборах даних Pascal VOC [21] від 2007 до 2012 р.

Модель було оновлено відповідні зміни розглядаються у дослідженні [22]. Подальші вдосконалення моделі були запропоновані у праці [23].

Головною перевагою YOLO безумовно є швидкодія.

Недоліки — досить високі вимоги до ресурсів апаратного забезпечення та нижчу точність розпізнавання порівняно з CNN.

Мета статті (постановка завдання)

Ураховуючи переваги та недоліки розглянутих систем доцільною є розробка системи розпізнавання зображень. ***Метою статті*** є дослідження ефективності запропонованої системи розпізнавання зображень з нейромережевою архітектурою на основі глибинного навчання.

Архітектура системи розпізнавання зображень

Глибинне навчання використовує послідовні шари обчислень, які розбивають простір ознак на ті, що відомі як представлення ознак вищого порядку. Використаємо *Rubix* ML — високорівневу бібліотеку машинного навчання та глибинного навчання для мови PHP [24].

Для тренування моделі глибинного навчання під назвою багатошаровий перцептрон (*Multilayer Perceptron*) для розпізнавання букв представлених рукописними літерами кирилиці.

Для розпізнавання букв, представлених рукописними літерами, класифікатор повинен мати можливість вивчити лінії, краї, кути та їх комбінації, щоб розрізнити літери на зображеннях. На рис. 9 представлено знімок ознак на одному шарі нейронної мережі, натренованої на наборі даних MNIST [25]. На рис. 9 показано, що на кожному шарі *learner* будує більш детальний опис навчальних даних доти, доки цифри не будуть легко розрізнювані вихідним шаром Softmax в кінці.

Запропонована архітектура системи розпізнавання зображень на основі глибинного навчання представлена на рис. 10.

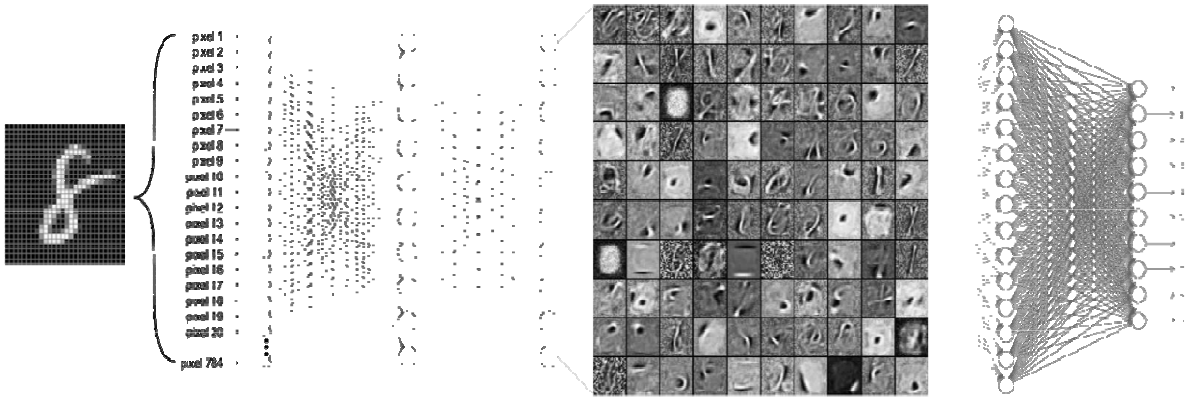

Рис. 9. Знімок ознак на одному шарі нейронної мережі, натренованої на наборі даних

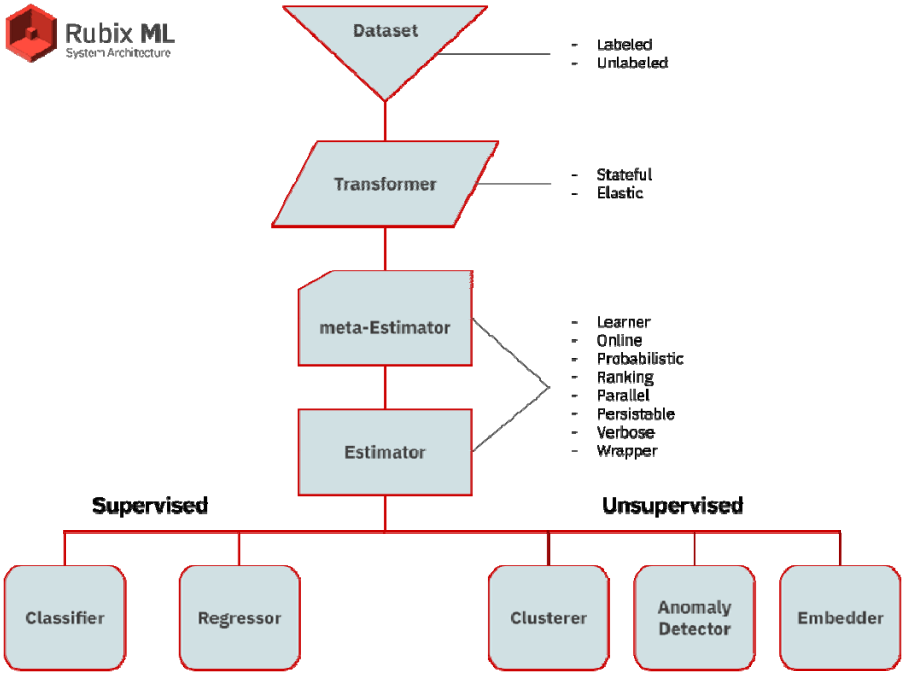

Рис. 10. Архітектура системи розпізнавання зображень

Архітектура системи. Rubix ML має універсальну модульну архітектуру, яка визначається кількома ключовими абстракціями та їх типами та інтерфейсами, які описано нижче.

Dataset Objects — являє собою спеціалізовані контейнери в пам'яті, куди передаються дані. Dataset objects (об'єкти набору даних) — це таблице-подібні структури, що використовують високорівневу систему типів, яка має операції для маніпулювання даними. Вони можуть містити гетерогенну суміш типів даних і спрощують транспортування даних канонічним способом. Наборам даних потрібна таблиця зразків, у якій кожен рядок становить собою зразок, і кожен стовпець представляє значення ознаки, представленої цим стовпцем. Вони мають додаткове обмеження, що кожен стовпець ознаки повинен бути однорідним, тобто вони повинні містити значення однакового типу даних високого рівня.

Elastic. Деякі transformers (перетворювачі) здатні адаптуватися до нових навчальних даних. Метод `update()` у *transformers*, які реалізують інтерфейс Elastic, може бути використаний для модифікації підгонки перетворювача з новими даними навіть після того, як була попередньо здійснена підгонка.

Meta-estimators (метаоцінювачі) покращують інші оцінювачі своїм власним доданим функціоналом. Вони включають ensembles, model selectors та інші покращувачі моделей, які «обгортають» сумісний базовий оцінювач.

Estimator. В Rubix ML усі learners реалізують інтерфейс Estimator. Він забезпечує основну функціональність інференції через метод `predict()`, який повертає набір прогнозів із набору даних. Крім того, він надає методи повернення декларацій сумісності типу оцінювача та типу даних.

Supervised Learning (контрольоване навчання) — це тип машинного навчання, яке містить в себе тренувальний сигнал у формі анотацій, наданих людиною, під назвою labels (мітки). Labels (Мітки) — це бажані вихідні дані learner (об'єкта навчання), враховуючи зразок, який ми йому показуємо. Саме тому, контрольоване навчання можна представити як навчання на прикладі. У Rubix ML є два типи контрольованого навчання. Графічне представлення розподілу даних для контрольованого навчання подано на рис. 11.

Classification. Для задач класифікації, learner тренується відрізняти зразки серед набору k можливих дискретних класів. Для такого роду проблем, тренувальні мітки — це класи, до яких належить кожен зразок. Приклади міток класів включають `cat`, `dog`, `human` тощо. Проблеми класифікації варіюються від простих до дуже складних і включають такі приклади, як: розпізнавання зображень з набору CIFAR-10 [26], аналіз емоційного тону тексту та класифікацію квітів — ірисів з відомого набору Іриси Фішера (Iris flower data set) [27].

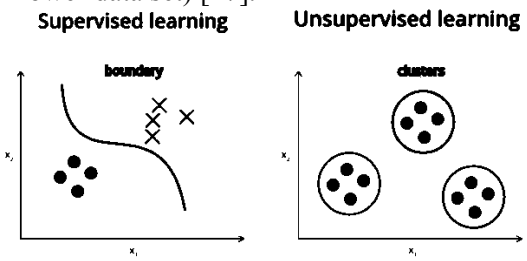

Рис. 11. Графічне представлення розподілу даних при Supervised та Unsupervised Learning

Regression (регресія) — це рід задач навчання, які мають на меті передбачити результат у безперервному режимі роботи. У цьому випадку, тренувальні мітки — це безперервні типи даних, такі як цілі числа та числа з плаваючою точкою. На відміну від класифікаторів, діапазон прогнозів, які може зробити регресор, нескінченний. Задачі регресії включають такі приклади: оцінку ціни продажу будинку, оцінку кредитоспроможності та визначення кута повороту безпілотного автомобіля.

Unsupervised Learning (навчання без нагляду або неконтрольоване навчання) — це форма навчання, яка не потребує тренувальних міток. Unsupervised learners надають першочергового значення освоєнню закономірностей у необроблених зразках. Існує три типи неконтрольованого навчання, які використовуються в Rubix ML. Графічне представлення розподілу даних для неконтрольованого навчання подано на рис. 4.

Clustering (кластеризація) приймає набір даних та присвоює кожному зразку дискретний номер кластера на основі його подібності з іншими зразками навчального набору. Це можна розглядати як слабкішу форму класифікації де імена класів невідомі. Кластеризація використовується для диференціації тканин із зображень PET (*positron emission tomography*) — сканування, сегментації ринку баз даних клієнтів, а також для виявлення спільнот у соціальних мережах. Графічне представлення кластеризації для неконтрольованого навчання подано на рис. 4.

Anomaly Detection (аномалії) визначаються як зразки, які були згенеровані процесом, відмінним від нормального, або такі, що не відповідають очікуваному розподілу тренувальних даних. Зразки можуть бути відмічені або упорядковані на основі оцінки їх аномальності. Виявлення аномалій використовується в інформаційній безпеці для виявлення вторгнень та відмов у доступі, та у фінансовій галузі для виявлення шахрайства.

Embedders (вбудовувачі) — це нелінійні редуктори розмірності, які створюють щільне представлення простору ознак вхідних даних, таким чином, щоб їх можна було візуалізувати або використати як вхідні дані нижчої розмірності до алгоритму навчання.

Попередня підготовка та екстракція даних

Набір даних CoMNIST (Cyrillic-oriented MNIST) [28] складається з більш ніж 28 000 png-зображень розміром 278×278 пікселів, що представляють 33 літери кирилиці та 26 літер латиниці. Літери на зображеннях були написані вручну на сенсорному екрані через краудсорсинг. Для нашого проекту знадобляться лише 33 літери кирилиці, тому скористаємось кириличною частиною символів що містить 15 233 зображень. Усі зображення розділено на два піднабори, отримавши таким чином 13 646 тренувальних та 1587 тестових зображень, організованих у підпапки, де назва папки — це анотована людиною мітка, надана зразку.

Наступним кроком є оптимізація набору даних, оскільки в нашому випадку обчислювальні потужності та ресурси є обмеженими. Використання зображень в оригінальному розмірі (278×278 пікселів) є практично неможливим, оскільки за результатами експериментів встановлено, що тренування на зразках лише двох літер використовує 15+ Гб оперативної пам'яті, що є максимально допустимим значенням для використовуваної апаратної конфігурації. Для порівняння — при використанні повного набору даних MNIST для тренування нейромережі було задіяно ~3 Гб оперативної пам'яті. Крім того, зображення з набору даних CoMNIST мають прозорий фон, що може спричинити неочікувану поведінку системи. Отже, необхідно попередньо обробити зображення з набору даних.

Для виконання цієї задачі було використано засоби автоматизації обробки партій зображень Adobe Photoshop CC. Розмір кожного зображення було пропорційно зменшено до 28×28 пікселів, після цього зображення було інвертоване та прозорий фон замінювався на чорний. Така послідовність операцій виявилась оптимальною з точки зору використання системних ресурсів. Для виявлення оптимальної послідовності було проведено декілька експериментальних обробок менших партій зображень. У результаті, отримані зразки являють собою білий контур літери на чорному фоні.

Зразки для тренування знаходяться у папці `training`. Для завантаження зображення у розроблений сценарій як ресурсу та присвоєння мітки з огляду на підпапку, використовуємо функцію `imagecreatefrompng()` з бібліотеки GD [29].

Потім створюється новий об'єкт міченого (labeled) набору даних із зразків та міток.

Підготовка набору даних під час виконання скрипта

Для формування набору даних у правильному форматі для нашого модуля learner використовуємо трансформуючий Pipeline. Відомо, що розмір кожного зразка зображення в нашому попередньо підготовленому наборі даних становить 28×28 пікселів. Для того, щоб переконатися, що майбутні зразки завжди мають правильний розмір, використовуємо Image Resizer. Потім, для конвертування зображення в необроблені дані пікселів, використовуємо Image Vectorizer, який екстрагує необроблені дані колірних каналів із зображення.

Оскільки зразки зображень є чорно-білими, нам потрібно використовувати лише 1 канал кольору на піксель.

У кінці pipeline будемо централізувати та масштабувати набір даних за допомогою Z Scale Standardizer, щоб сприяти прискоренню конвергенції нейронної мережі.

Створення одиниці learner

Наступним кроком необхідно інсталювати Multilayer Perceptron класифікатор.

Розглянемо архітектуру нейронної мережі, яка підходить для задачі MNIST, що складається з трьох груп Dense нейронних шарів, з подальшим шаром активації Leaky ReLU, а потім м'яким Dropout шаром, який буде діяти як регуляризатор.

Теоретично, кожен наступний шар мережі стає більш складним детектором ознак. Вихідний шар додає додатковий шар нейронів з Softmax активацією, що робить цю конкретну мережеву архітектуру 4-шаровою в глибину (рис. 12).

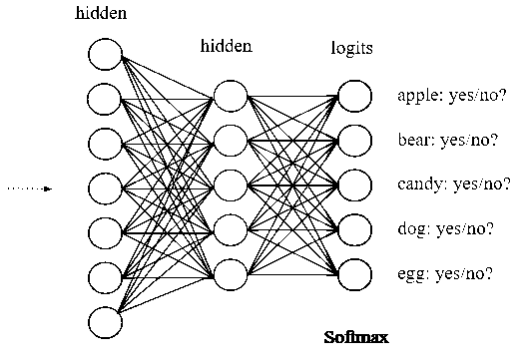

Рис. 12. Шар Softmax усередині нейронної мережі

Математично Leaky ReLU (Leaky (або Parameteric) Rectified Linear Unit) можна представити у вигляді формули

$$f(x)=\begin{cases} a\,x, \text{ для } x<0; \\ x, \text{ для } x\geq 0, \end{cases}$$

а Softmax

$$S(y_i)=\frac{e^{y_i}}{\Sigma_j e^{y_i}} i \text{ та } j=[0,...,K],$$

де K — розмірність вектору y.

Наступним кроком буде встановлення розміру партії еквівалентним 200. Тобто до 200 зразків з тренувального набору буде надсилатися за один раз через мережу. Adam Optimizer визначає крок оновлення алгоритму градієнтного спуску і використовує комбінацію Momentum та RMS Prop для його оновлення. Він використовує глобальну швидкість навчання для контролю розміру кроку, який встановимо рівним 0,001 для цього прикладу.

Алгоритм Momentum може бути представлений у математичному вигляді так.

Повторювати до моменту збіжності:

$$\left\{\nu_j=\eta\nu_{j-1}-\alpha dw\sum_m L_m(w)w_j=w_j+\nu_j\right\},$$

де j — поточна ітерація; ν_j — значення градієнту; η — коефіцієнт Momentum, що визначає відсоток градієнту, що повертається; w_j — направлення градієнтного спуску.

RMS Prop може бути представлений як:

$$E\left[g^2\right]_t=\beta E\left[g^2\right]_{t-1}+(1-\beta)\left(\frac{\delta C}{\delta w}\right)^2;$$

$$w_t=w_{t-1}-\frac{\eta}{\sqrt{E\left[g^2\right]_t}}\frac{\delta C}{\delta w},$$

де $E[g]$ — ковзне середнє квадратів градієнтів; $\delta C/\delta w$ — градієнт функції вартості відносно ваги w; η — швидкість навчання; β — параметр ковзного середнього (рекомендоване значення за замовчуванням — 0,9); wj — направлення градієнтного спуску.

Алгоритм Adam (Adaptive Moment Optimization) використовує комбінацію виразів вище та математично виглядає так:

$$\nu_t=\beta_1\nu_{t-1}-(1-\beta_1)\frac{\delta C}{\delta w};$$

$$s_t=\beta_2 s_{t-1}-(1-\beta_2)\left(\frac{\delta C}{\delta w}\right)^2;$$

$$w_{t+1}=w_t-\eta\frac{\nu_t}{\sqrt{s_t+\varepsilon}}\frac{\delta C}{\delta w},$$

де η — початкова швидкість навчання; $\delta C/\delta w$ — градієнт функції вартості відносно ваги w на момент часу t; st — ковзне середнє квадратів градієнтів вздовж wj; vt — ковзне середнє градієнтів уздовж wj; β_1 та β_2 — параметри ковзного середнього (рекомендоване значення за замовчуванням $\beta_1 = 0{,}9$; $\beta_1 = 0{,}99$); wj — направлення градієнтного спуску

Щоб мати можливість зберігати та завантажувати модель зі сховища, «обгорнемо» увесь pipeline у мета-оцінювач Persistent Model. Persistent Model надає додаткові методи `save()` та `load()` на додаток до методів базового оцінювача. Для цього потрібен об'єкт Persister, щоб повідомити, де модель має зберігатися. Для наших цілей будемо використовувати Filesystem persister, який приймає шлях до файлу моделі на диску. Встановлення режиму `history` в `true` означає, що persister буде відслідковувати кожне збереження.

Тренування системи розпізнавання зображень

Для того, щоб розпочати тренування, викличемо метод `train()` у екземпляра оцінювача з тренувальним набором у якості вхідних даних.

Результати та втрати валідації. Прогрес тренування можна візуалізувати на кожному етапі, вивантажуючи значення функції втрат та метрику валідації після тренування. Метод `steps()` виведе масив, що містить значення функції вартості Cross Entropy, а метод `scores()` поверне масив балів з метрики валідації F Beta.

Потім можна побудувати графіки значень (рис.13 та рис.14) за допомогою програмного забезпечення для побудови графіків, наприклад, Tableu або Excel.

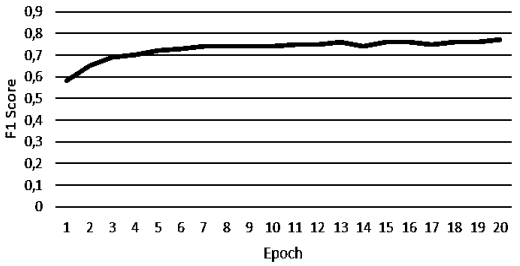

Рис. 13. Графік результатів валідації для Rubix ML Multi Layer Perceptron

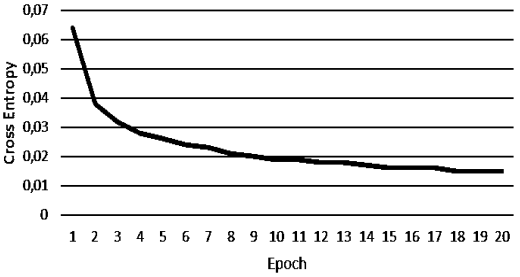

Рис. 14. Графік втрат валідації для Rubix ML Multi Layer Perceptron

Якщо система працює правильно, то величина втрат повинна знижуватися, в той час як значен-

ня показника валідації зростати. Враховуючи отримання миттєвих знімків, епоха, у якій значення показника валідації є найвищими, а втрати найнижчими — це точка, у якій беруться значення параметрів мережі.

За результатами аналізу графіків можна зробити висновок що система працює правильно. Величина втрат повинна знижуватися, в той час як значення показника валідації зростати. Враховуючи отримання миттєвих знімків, епоха, у якій значення показника валідації є найвищими, а втрати найнижчими — це точка, у якій беруться значення параметрів мережі.

Збереження. Натреновану мережу можна зберегти шляхом виклику методу `save()`, що надається обгорткою Persistent Model. Модель буде збережено у компактному серіалізованому форматі, такому як нативний формат серіалізації PHP.

Перехресна валідація — це методика оцінки того, наскільки добре learner може узагальнити свою підготовку до незалежного набору даних. Мета полягає в тому, щоб виявити помилку вибірки або надмірне навчання (перетренування), що призвело б до того, що модель неефективно працювала на небачених даних.

Для розроблюваної системи адаптований тестувальний піднабір, отриманий з набору даних CoMNIST, включає додаткові 1587 зображень з мітками, які можна використовувати для тестування моделі. Оскільки не було використано жоден із цих зразків для навчання мережі, можна ефективно використовувати їх для тестування продуктивності узагальнення моделі. Для початку екстрагуємо тестові зразки та мітки з папки `testing` в об'єкт міченого (labeled) набору даних.

Завантаження моделі зі сховища. У скрипті тренування необхідно переконатися, що модель збережено перед тим, як вийти. У скрипті валідації завантажимо натреновану модель зі сховища та використаємо її для прогнозування на тестовому наборі. Статичний метод `load()` на Persistent Model приймає об'єкт Persister, який вказує на модель в сховищі, в якості свого єдиного аргументу і повертає завантажений екземпляр оцінювача.

Прогнозування. Далі можна використовувати оцінювач для прогнозування на тестовому наборі. Метод `predict()` приймає набір даних на вхід і повертає масив прогнозів.

Формування звіту. Звіт про перехресну валідацію, який створюємо, насправді є комбінацією двох звітів — Multiclass Breakdown та Confusion Matrix. «Обгортаємо» кожен звіт у оболонку Aggregate Report, щоб генерувати обидва звіти одночасно. Multiclass Breakdown надасть детальну інформацію про ефективність роботи оцінювача на рівні класу. Confusion Matrix дасть уявлення про те, які мітки оцінювач плутає одну з одною [30].

Щоб згенерувати звіт, потрібно передати прогноз разом з мітками з тестового набору до методу `generate()`.

Приклади показано на рис. 15, 16). Як можна бачити, розроблена модель змогла досягти 98 % точності на тестовому наборі (рис. 17, 18).


```
[
  {
    "overall": {
      "accuracy": 0.9809454682660341,
      "precision": 0.7518986814403983,
      "recall": 0.7559231445374274,
      "specificity": 0.9900806066095389,
      "f1_score": 0.748640844981423,
      "cardinality": 1587
    },
    "classes": {
      "Е": {
        "accuracy": 0.963884430176565,
        "precision": 0.6823529411764706
      },
      "Ъ": {
        "accuracy": 0.9812091503267973,
        "precision": 0.8103448275862069
      },
...
      "Й": {
        "accuracy": 0.9804081632653061,
        "precision": 0.9565217391304348
      },
      "Р": {
        "accuracy": 0.988477366255144,
        "precision": 0.9142857142857143
      }
    }
```


Рис. 15. Результат Multiclass Breakdown в report.json для розпізнавання літер


```
    "Л": {
      "Л": 32
    },
    "Ц": {
      "Ц": 30,
      "Щ": 8,
      "И": 3,
      "Й": 3
    },
...
    "Н": {
      "М": 3,
      "И": 4,
      "Н": 40
    },
    "Й": {
      "Ц": 1,
      "И": 1,
      "Й": 44
    },
    "Р": {
      "Э": 1,
      "У": 2,
      "Р": 32
    }
  }
]
```


Рис. 16. Результат Confusion Matrix в report.json для розпізнавання літер

```php
<?php

include __DIR__ . '/vendor/autoload.php';

use ...

ini_set('memory_limit', '-1');

echo 'Loading data into memory ...' . PHP_EOL;

$samples = $labels = [];

chdir("./training");
$directories = glob("*", GLOB_ONLYDIR);
chdir("../");

foreach ($directories as $label) {
    foreach (glob("training/$label/*.png") as $file) {
        $samples[] = [imagecreatefrompng($file)];
        $labels[] = (string) $label;
    }
}

$dataset = new Labeled($samples, $labels);

$estimator = new PersistentModel(
    new Pipeline([
        new ImageResizer(28, 28),
        new ImageVectorizer(true),
        new ZScaleStandardizer(),
    ], new MultiLayerPerceptron([
        new Dense(100),
        new Activation(new LeakyReLU()),
        new Dropout(0.2),
        new Dense(100),
        new Activation(new LeakyReLU()),
        new Dropout(0.2),
        new Dense(100),
        new Activation(new LeakyReLU()),
        new Dropout(0.2),
    ], 200, new Adam(0.001))),
    new Filesystem('comnist.model', true)
);

$estimator->setLogger(new Screen('CoMNIST'));

echo 'Training ...' .  PHP_EOL;

$estimator->train($dataset);

$scores = $estimator->scores();
$losses = $estimator->steps();

$writer = Writer::createFromPath('progress.csv',
    'w+');
$writer->insertOne(['score', 'loss']);
$writer->insertAll(array_transpose(
    [$scores, $losses]));

echo 'Progress saved to progress.csv' . PHP_EOL;

if (strtolower(trim(readline(
    'Save this model? (y|[n]): '))) === 'y') {
    $estimator->save();
}
```

Рис. 17. Скрипт тренування train.php після оптимізації

```php
<?php

include __DIR__ . '/vendor/autoload.php';

use ...

ini_set('memory_limit', '-1');

echo 'Loading data into memory ...' . PHP_EOL;

$samples = $labels = [];

chdir("./training");
$directories = glob("*", GLOB_ONLYDIR);
chdir("../");

foreach ($directories as $label) {
    foreach (glob("testing/$label/*.png")
             as $file) {
        $samples[] =
            [imagecreatefrompng($file)];
        $labels[] = (string) $label;
    }
}

$dataset = new Labeled($samples, $labels);

$estimator = PersistentModel::Load(
    new Filesystem('comnist.model'));

echo 'Making predictions ...' . PHP_EOL;

$predictions = $estimator->predict($dataset);

$report = new AggregateReport([
    new MulticlassBreakdown(),
    new ConfusionMatrix(),
]);

$results = $report->generate($predictions,
    $dataset->labels());

file_put_contents('report.json', json_encode(
    $results, JSON_PRETTY_PRINT));

echo 'Report saved to report.json' . PHP_EOL;
```

Рис. 18. Скрипт валідації validate.php після оптимізації

Висновки

У статті запропоновано систему розпізнавання зображень з нейромережевою архітектурою на основі технології глибинного навчання. Такі системи є затребуваними в багатьох галузях та мають широкий спектр застосування. Впровадження та розгортання подібних систем значною мірою підвищує рівень автоматизації пов'язаних процесів та ефективності їх виконання.

Аналіз отриманих результатів

Отримана модель має високі показники точності розпізнавання, при цьому використання ресурсів максимально оптимізовано. Беручи до уваги той факт, що точність розпізнавання чисел становить 99 %, потрібно зазначити, що зображення чисел більш придатні для якісного розпізнавання патернів. Саме тому досягнення практично тієї самої точності розпізнавання (98 %) кириличного алфавіту є хорошим результатом, ураховуючи складність та різноманітну манеру написання кириличних літер різними людьми.

З огляду на більшу кількість категорій зразків порівняно з прикладом розпізнавання рукопис-

них цифр (33 літери проти 10 цифр) та більшу загальну складність розпізнавання, отримані результати наочно демонструють гнучкість, масштабованість та ефективність використовуваної архітектури та продуктивність отримуваної в результаті моделі.

Розроблені адаптовані версії скриптів тренування та валідації є більш гнучкими та універсальними щодо роботи зі зразками наборів даних різних типів Використаний у цій роботі набір даних CoMNIST (Cyrillic-oriented MNIST) є одним з найкращих та найбільших наборів рукописних зразків, що відкриті для вільного використання.

Усі зображення (зразки набору) розділені на два піднабори, отримавши таким чином окремо тренувальний та тестовий набори, зображення у яких не повторюються. На відміну від набору даних MNIST, зразки CoMNIST (Cyrillic-oriented MNIST) потребують певної попередньої обробки для використання в моделі.

Експериментальним шляхом було встановлено, що оптимальним є варіант приведення зразків у відповідність зі зразками набору MNIST (тобто використання таких же параметрів для зображень), оскільки такий підхід забезпечує найкраще відношення показників якість зразків / використання системних ресурсів.

У результаті численних експериментів та дослідних тренувань нейромережі було отримано важливі дані, що стали основою для стабільної та високопродуктивної конфігурації системи розпізнавання зображень з нейромережевою архітектурою на основі глибинного навчання.

ЛІТЕРАТУРА

1. **Bishop C. M.** Pattern Recognition and Machine Learning. Springer, 2006. ISBN 978-0-387-31073-2.
2. **Wikipedia**. Machine Learning. URL: https://en.wikipedia.org/wiki/Machine_learning (дата звернення: 24.01.2020).
3. **Schmidhuber J.** Deep Learning in Neural Networks: An Overview. *Neural Networks*. 2015. Vol. 61. Pp. 85–117.
4. **Bengio Yoshua**, LeCun Yann, Hinton Geoffrey Deep Learning. *Nature*. 2015. Vol. 521 (7553). Pp. 436–444.
5. **Image** Recognition. URL: https://sightcorp.com/knowledge-base/image-recognition/ (дата звернення 24.01.2020).
6. **Emerging** Tech Research: Artificial Intelligence & Machine Learning. URL: https://pitchbook.com/news/reports/3q-2019-emerging-tech-research-artificial-intelligence-machine-learning (дата звернення 24.01.2020).
7. **How** computer vision can improve buyer conversion and retention for E-commerce sites. URL: https://www.clarifai.com/blog/how-computer-vision-can-improve-buyer-conversion-and-retention (дата звернення 24.01.2020).
8. **Celebrating** one year of Pinterest Lens. URL: https://newsroom.pinterest.com/en/post/celebrating-one-year-of-pinterest-lens (дата звернення 27.01.2020).
9. **Artificial** intelligence in E-commerce: benefits, statistics, facts, use cases & case studies. URL: https://apiumhub.com/tech-blog-barcelona/artificial-intelligence-ecommerce/ (дата звернення 17.02.2020).
10. **Simple** Digit Recognition OCR in OpenCV-Python. URL: https://stackoverflow.com/a/9620295 (дата звернення 01.02.2020).
11. **A gentle** introduction to OCR. URL: https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa (дата звернення 01.02.2020).
12. **Scikit-image**. Image processing in Python. Label image regions. URL: https://scikit-image.org/docs/dev/auto_examples/segmentation/plot_label.html#sphx-glr-download-auto-examples-segmentation-plot-label-py (дата звернення 01.02.2020).
13. **EAST**: An Efficient and Accurate Scene Text Detector. URL: https://arxiv.org/pdf/1704.03155.pdf (дата звернення 01.02.2020).
14. **SEE**: Towards Semi-Supervised End-to-End Scene Text Recognition. URL: https://arxiv.org/pdf/1712.05404.pdf (дата звернення 01.02.2020).
15. **STN-OCR**: A single Neural Network for Text Detection and Text Recognition. URL: https://arxiv.org/abs/1707.08831 (дата звернення 01.02.2020).
16. **Review: SSD** — Single Shot Detector (Object Detection). URL: https://towardsdatascience.com/ review-ssd-single-shot-detector-object-detection-851a94607d11 (дата звернення 01.02.2020).
17. **Handwritten** Digit Recognition using Machine Learning. URL: https://medium.com/@himanshubeniwal/handwritten-digit-recognition-using-machine-learning-ad30562a9b64 (дата звернення 01.02.2020).
18. **Fast**, Simple and Accurate Handwritten Digit Classification by Training Shallow Neural Network Classifiers with the ‘Extreme Learning Machine’ Algorithm. URL: https://www.researchgate.net/figure/Comparison-of-our-results-on-the-MNIST-data-set-with-published-results-using-other_fig5_281815053 (дата звернення 01.02.2020).
19. **You Only Look Once**: Unified, Real-Time Object Detection. URL: https://arxiv.org/abs/1506.02640 (дата звернення 30.01.2020).
20. **YOLO:** Real Time Object Detection. URL: https://github.com/pjreddie/darknet/wiki/YOLO:-Real-Time-Object-Detection (дата звернення 30.01.2020).
21. **The PASCAL** Visual Object Classes Homepage. URL: http://host.robots.ox.ac.uk:8080/pascal /VOC/ (дата звернення 30.01.2020).

22. **YOLO9000:** Better, Faster, Stronger. URL: https://arxiv.org/abs/1612.08242 (дата звернення 30.01.2020).

23. **YOLOv3:** An Incremental Improvement. URL: https://arxiv.org/abs/1804.02767 (дата звернення 30.01.2020).

24. **Rubix ML.** A high-level machine learning and deep learning library for the PHP language. URL: https://rubixml.com/ (дата звернення 17.02.2020)

25. **MNIST** database. URL: https://en.wikipedia.org/ wiki/MNIST_database (дата звернення 03.02.2020).

26. **CIFAR-10.** URL: https://en.wikipedia.org/wiki/CIFAR-10 (дата звернення 03.02.2020).

27. **Iris flower** dataset. URL: https://en.wikipedia.org/ wiki/Iris_flower_data_set (дата звернення 03.02.2020).

28. **Cyrillic-oriented** MNIST. A dataset of Latin and Cyrillic letter images for text recognition. URL: https://github.com/GregVial/CoMNIST (дата звернення 03.02.2020).

29. **PHP**. Image Processing and GD. URL: https://www.php.net/manual/en/book.image.php (дата звернення 03.02.2020)

30. RubixML/MNIST. URL: https://github.com/RubixML/ MNIST (дата звернення 03.02.2020).

Холявкіна Т. В., Резаєв Я. О., Харченко О. О.

СИСТЕМА РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ З НЕЙРОМЕРЕЖЕВОЮ АРХІТЕКТУРОЮ НА ОСНОВІ ТЕХНОЛОГІЇ ГЛИБИННОГО НАВЧАННЯ

Машинне навчання дозволяє отримувати корисну інформацію з необроблених даних, щоб швидко та ефективно вирішувати складні, насичені даними задачі. Як підгалузь штучного інтелекту машинне навчання досліджує вивчення та побудову алгоритмів, які можуть формувати процес навчання та здійснювати прогнозування на основі даних, — такі алгоритми є значно ефективнішими методики використання строго статичних програмних інструкцій. Машинне навчання застосовують в ряді обчислювальних задач, у яких проектування та реалізація явних алгоритмів з належним рівнем продуктивності є складним або взагалі нездійсненним процесом.

Глибинне навчання є галуззю машинного навчання, що ґрунтується на наборі алгоритмів, які моделюють високорівневі абстракції в даних, застосовуючи глибинний граф із декількома обробними шарами, що побудовано з кількох лінійних або нелінійних перетворень. Дослідження в цій області намагаються зробити кращі представлення та створити моделі для навчання цих представлень з великомасштабних немічених даних. Різні архітектури глибинного навчання, такі як глибинні нейронні мережі, конволюційні глибинні нейронні мережі, глибинні мережі переконань та рекурентні нейронні мережі застосовуються в таких областях, як комп'ютерне бачення, автоматичне розпізнавання мовлення, обробка природної мови, розпізнавання звуків та біоінформатика, де вони представляють передові результати в різноманітних задачах.

Ця стаття охоплює поняття машинного навчання, глибинного навчання та розпізнавання зображень. Розглядається конкретний приклад (з поясненням кроків) використання глибинного навчання для побудови системи розпізнавання зображень з нейромережевою архітектурою. Отримана система надає широкі можливості для автоматизації технологічних процесів та підвищення їх ефективності. При цьому концепція системи може бути адаптована відповідно до типу нових задач.

Ключові слова: машинне навчання; глибинне навчання; дані; розпізнавання зображень.

Kholyavkina T., Rezaiev Y., Kharchenko O.

DEEP LEARNING BASED IMAGE RECOGNITION SYSTEM WITH NEURAL NETWORK ARCHITECTURE

Machine learning allows us to obtain useful information from raw data for quick and efficient solving of complex data-intensive tasks. As a sub-sector of artificial intelligence, machine learning explores study and construction of algorithms that make data-based predictions and are capable of shaping the learning process accordingly — such algorithms are far more effective than the technique of using strictly static program instructions. Machine learning algorithms are used in a wide variety of computational tasks, where it is difficult or infeasible to design and implement an explicit algorithm with decent performance.

Deep learning is a branch of machine learning, based on a set of algorithms that model high-level abstractions in data by applying a depth graph with multiple processing layers, built from several linear or non-linear transformations. Research in this area is aimed at getting better representations and creating models for training on these representations from large-scale unlabeled data. Deep learning architectures such as deep neural networks, convolutional deep neural networks, deep belief networks and recurrent neural networks have been applied to fields including computer vision, speech recognition, natural language processing, sound recognition, and bioinformatics where they have produced cutting-edge results in a variety of cases.

This article covers the concepts of machine learning, deep learning and image recognition. A specific example (with step-by-step explanation) of using deep learning for building an image recognition system with a neural network architecture is given. The resulting system provides ample opportunities to automate the technological processes and increase their efficiency. The concept of the system can be adapted to the tasks of a new type.

Keywords: machine learning; deep learning; data; image recognition; perceptron.

Холявкина Т. В., Резаев Я. О., Харченко А.А.

СИСТЕМА РАСПОЗНАВАНИЯ ИЗОБРАЖЕНИЙ С НЕЙРОСЕТЕВОЙ АРХІТЕКТУРОЙ НА ОСНОВЕ ТЕХНОЛОГИЙ ГЛУБИННОГО ОБУЧЕНИЯ

Машинное обучение позволяет получать полезную информацию из необработанных данных, чтобы быстро и эффективно решать сложные насыщенные данными задачи. Как подотрасль искусственного интеллекта машинное обучение исследует изучение и построение алгоритмов, которые могут формировать процесс обучения и осуществлять прогнозирование на основе данных, — такие алгоритмы значительно эффективнее методики использования строго статических программных инструкций. Машинное обучение применяют в ряде вычислительных задач, в которых проектирование и реализация явных алгоритмов с должным уровнем производительности затруднительно или полностью невыполнимый процесс.

Глубинное обучение является отраслью машинного обучения, основанной на наборе алгоритмов, которые моделируют высокоуровневые абстракции в данных, применяя глубинный граф с несколькими обрабатывающими слоями, которые построены из нескольких линейных или нелинейных преобразований. Исследования в этой области направлены на получение лучших представлений и создание моделей для обучения на этих представлениях с крупномасштабных немеченых данных. Различные архитектуры глубинного обучения, такие как глубинные нейронные сети, свёрточные глубинные нейронные сети, глубинные сети доверия и рекуррентные нейронные сети применяются в таких областях, как компьютерное зрение, автоматическое распознавание речи, обработка естественного языка, распознавание звуков и биоинформатика, где они представляют передовые результаты в различных задачах.

Эта статья охватывает понятие машинного обучения, глубинного обучения и распознавания изображений. Рассматривается конкретный пример (с объяснением шагов) использования глубинного обучения для построения системы распознавания изображений с нейросетевой архитектурой. Полученная система предоставляет широкие возможности для автоматизации технологических процессов и повышения их эффективности. При этом концепция системы может быть адаптирована в соответствии с типом новых задач.

Ключевые слова: машинное обучение; глубинное обучение; глубокое обучение; данные; распознавание изображений; перцептрон.

Стаття надійшла до редакції 24.02.2020 р.
Прийнято до друку 11.03.2020 р.