Хмельницький національний університет
Факультет програмування
та комп'ютерних і телекомунікаційних систем
Кафедра телекомунікацій, медійних та інтелектуальних технологій

ДИПЛОМНА РОБОТА МАГІСТРА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ КЛАСИФІКАЦІЇ МАРОК АВТОМОБІЛІВ З ВИКОРИСТАННЯМ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

Назва теми

Галузь знань 11 – Математика та статистика

Спеціальність 113 – Прикладна математика

Шифр ДРПМ.2019/115.01.26.00

Виконала:
студентка 2 курсу, група ПМм-19-1 _____ Підпис І.В. Руденко Ініціали, прізвище

Керівник:
канд.тех.наук, доцент _____ Підпис, дата Д.М. Медзатий Ініціали, прізвище

До захисту допускаю:

Зав. кафедри ТМІТ докт.тех.наук, доцент _____ Підпис, дата С.К. Підченко Ініціали, прізвище

16 12 2020 р.

Хмельницький, 2020

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ПРОГРАМУВАННЯ ТА КОМП'ЮТЕРНИХ І ТЕЛЕКОМУНІКАЦІЙНИХ СИСТЕМ

Кафедра ТЕЛЕКОМУНІКАЦІЙ, МЕДІЙНИХ ТА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ

Освітній рівень МАГІСТР

Галузь знань 11 МАТЕМАТИКА ТА СТАТИСТИКА

Спеціальність 113 ПРИКЛАДНА МАТЕМАТИКА

Освітня програма ОСВІТНЬО-ПРОФЕСІЙНА ПРОГРАМА ПІДГОТОВКИ МАГІСТРА

ЗАТВЕРДЖУЮ

Зав. кафедри д.т.н. доцент Підченко .

Сергій Костянтинович

" 03 " 09 2020 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Руденко Інна Вікторівна

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Інформаційна технологія для класифікації марок автомобілів звикористанням згорткової нейронної мережі

Керівник проекту (роботи) Медзатий Дмитро Миколайович

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

к.т.н., доцент, кандидат технічних наук

Затверджена наказом ректора університету від 01.09.2020 р. № 118

2. Строк подання студентом проекту (роботи) на кафедру 01.12.2020 р.

3. Вихідні дані до проекту (роботи) Проведення порівняльного аналізу підходів до вирішення задачі розпізнавання елементів на зображеннях на основі нейронних мереж. Дослідження існуючих алгоритмів для розпізнавання об'єктів на зображеннях. Створення оптимізованого алгоритму для розпізнавання об'єктів на зображеннях з дослідженням оптимальних гіперпараметрів згорткової нейронної мережі.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Створення математичної моделі для оптимізації алгоритму для розпізнавання марок автомобілів на зображеннях за допомогою підбору оптимальних гіперпараметрів. Проведення аналізу і сумування висновків на основі дослідження. Дослідження ефективності математичної моделі, реалізація моделі зі допомогою програмування.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Аналіз результатів незалежного тестового порівняння ефективності розробленого алгоритму розпізнавання марок автомобілів на зображеннях. Алгоритми та схеми розпізнавання елементів на зображеннях, їх порівняння. Архітектура згорткової нейронної мережі, аналіз продуктивності, висновки.

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Відповідальний за оформлення ДП			

7. Дата видачі завдання «_1_»__Лютого__2020р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
Розробка індивідуального графіку дослідження. Узгодження його з керівником магістерської дисертації	01.09 – 05.09	Виконано
Формулювання основних задач та перелік робіт, визначених на написання магістерської дисертації	05.09 – 10.09	Виконано
Ознайомлення з науково-інформаційними джерелами за спеціальністю, обрання технічної проблематики та формування бібліографії	10.09 – 15.09	Виконано
Збір, аналіз та обробка відповідними методами фактичного, фактологічного та статистичного матеріалу щодо згорткових нейронних мереж	15.09 – 30.09	Виконано
Детальне ознайомлення з нормативно документацією при розробці програмного забезпечення	30.09 – 15.10	Виконано
Написання алгоритму для навчання згорткової нейронної мережі	15.10 – 25.10	Виконано
Тестування алгоритму, порівняння результативних даних	25.10 – 15.11	Виконано
Оформлення магістерської дисертації	15.11 – 20.11	Виконано

Студент _____ Руденко І.В.
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Медзатий Д.М.
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема дипломної роботи: Інформаційна технологія для класифікації марок автомобілів з використанням згорткової нейронної мережі.

Автор роботи: Руденко Інна Вікторівна

Керівник роботи: Медзатий Дмитро Миколайович

Загальний обсяг роботи: 104 сторінки, 19 рисунків, 5 додатків, 40 посилань.

НЕЙРОННА МЕРЕЖА, КЛАСИФІКАЦІЙНА МОДЕЛЬ, ЗГОРТКОВІ РІВНІ, МАШИННЕ НАВЧАННЯ, ОПТИМІЗАЦІЯ ГІПЕРПАРАМЕТРІВ

Метою роботи є розробка програмного модуля, що базується на удосконаленні та оптимізації параметрів класифікаційної моделіз горткової нейронної мережі.

Дана дипломна робота присвячена розробці програмного продукту для розпізнавання марок автомобілів за допомогою навчання згорткової нейронної мережі і підбору оптимальних параметрів.

ANNOTATION

Thesis topic: Information technology for the classification of car brands using a convolutional neural network.

Author of the work: Rudenko Inna Viktorivna

Mentor: Dmitry Medzatiy

Total volume of work: 104 pages, 19 figures, 5 appendices, 40 references

NEURAL NETWORK, CLASSIFICATION MODEL, ROLLER LEVELS, MACHINE LEARNING, HYPERPARAMETER OPTIMIZATION

The aim of the work is to develop a software module based on the improvement and optimization the classification model parameters of the convolutional neural network.

A method for hyperparameters optimization based on the use of existing algorithms has been developed, with the help of which the machine learning model was able to optimally solve the learning problem.

8.12.2020р
Дата/ Date

Підпис студента/ Signature

ЗМІСТ

ВСТУП

Американський піонер, Артур Семюель, започаткував термін «машинне навчання» 1959 рокуу галузі комп'ютерних ігор та штучного інтелекту, працюючи в IBM. Машинне навчання з’явилосьіз штучного інтелекту, як наукове прагнення. Ще з ранніх днів деякі дослідники зацікавилися тим, щоб комп’ютери навчались з даних. Вони пробували наблизитися до розв'язання цієї задачі різними символьними способами, а також тим, чому згодом дали назву «нейронні мережі».

То були в більшості перцептрони та й інші моделі, що з часом виявилися повторними винаходами узагальнених лінійних моделей статистики. Також застосовувалося ймовірнісне міркування, зокрема в автоматизованій медичній діагностиці [1].

Проте зростання акценту на логічному, заснованому на знаннях підході викликало розрив між штучним інтелектом та машинним навчанням. Ймовірнісні системи страждали на практичні та теоретичні проблеми представлення даних та збирання.

Близько 1980 року були винайденні експертні системи, щоб домінувати над штучним інтелектом, а статистика була в недостатньо хороша. Робота на основі знань та символів над навчанням дійсно продовжувалася за допомогою штучного інтелекту, все ближче наближаючись до індуктивного логічного програмування, однак більш статистичні дослідження були тепер поза межами області справжнього штучного інтелекту, в інформаційному пошуку та розпізнаванні образів. Орієнтовно в цей же час інформатикою та штучним інтелектом було здійснено дослідження нейронних мереж. Таке дослідження також було продовжено за межами навчання штучного інтелекту та інформатики. Такі дослідниками з інших дисциплін, як Хопфілд, Румельхарт та Хінтон брали в тому участь. Після довгої роботи, вони діцйшли успіху у середині 1980-х років. Вони винайшли зворотне поширення, однак це було вдруге. Загалом, машинне навчання, відділене як окрема

область, почало сильно розвиватися в 1990-х роках. Дана область змінила свої цілі з досягнення штучногоінтелекту на розв'язання розв'язних задач практичного характеру. Вона змістила ракурс із багатьох символьних підходів, що були успадкованні цією областю від штучного інтелекту до моделей й методів, що є запозиченні із статистики та теорії ймовірності. Вона також виграла від можливості розповсюджування оцифрованої інформації через Інтернет та збільшуваної її доступності.

Добування даних та машинне навчання часто використовують однакові методи, тобто вони дійсно значно перекриваються між собою, але в той час як машинне навчання побудоване на передбаченні, опирається на відомі властивості, що є вивченні з даних для тренування, в свою чергу, добування даних побудоване на відкритті раніше невідомих властивостях даних. Це і було основним кроком аналізу для створення знань у базах даних. Наспраді, дуже багато методів машинного навчання використовуються ща допомогою добування даних, але з інакшими цілями. Хоч і машинне навчання також використовує методи добування даних, таких як: «Навчання без вчителя», воно також використовує данні методи для попередньої обробки й для покращення точності під час роботи механізму навчання. Загалом існує велика плутанина між цими двома спільнотами, яка базується на основних припущеннях, з якими ці спільноти працюють. У машинному навчанні прийнятто оцінювати продуктивність базуючись на відношенні до здатності передбачати відоме знання, в той час як у відкриванні знань та добуванні даних (knowledge discovery and data mining) основна задача полягає у відкриванні невідомого знання. Некерований метод буде значно гіршим ніж інші керовані методи під час оцінки по відношенню до відомих знань, в той час як у типовій задачі добування даних керовані методи не можуть бути застосованні через відсутність даних для тренування.

Машинне навчання також є тіснопов’язаним з оптимізацією: дуже багато задач навчання є зформульовані як мінімізація певної функції втрат на тренувальному наборі даних. Функції витрат показують розбіжність між дійсними зразками задачі й передбаченнями тренованої моделі. От для прикладу в

класифікації небхідно визначати мітки для зразків, і тоді моделі будуть тренуватись правильно передбачати попередньо визначені мітки. Взагалі основна різниця між данними двома областями з'являється з мети узагальнення, іншими словами, в той час як алгоритми оптимізації дійсно зможе мінімізувати витрати на наборі даних для тренування, то в інший час машинне навчання є зосереджене на небачених зразках й мінімізації витрат. Статистика та машинне навчання є дуже тісно пов'язаними сферами. Як казав Майкл І. Джорд, ідеї, враховуючи ідеї методологічних принципів та ідеї теоретичних інструментів, в свою чергу мали дійсно довгу історію в науці статистики [2][3]. Він також запровадив термін «наука про дані» для позначення загальної області вчення.

В свй час Лео Брейман зазначив власні парадигми статистичного моделювання. Він назвав їх як модель даних, що означає алгоритм машинного навчання, та алгоритмічна модель.

Інші фахівці статистики використовують існуючі методи машинного навчання, для існування спільної області, котру вони називали статистичним навчанням (statistical learning).

Згорткові нейронні мережі (convolutional neural network) в машинному навчанні — це набір глибинних штучних нейронних мереж прямого поширення, що успішно використовується у розпізнаванні і аналізу зображень. Конструкція згорткових нейронних мережстворена на основі зорових механізмів живих організмів. Протягом 1950-тих та 1960-тих років над цим терміном працювали Г'юбел та Візел. Врешті вони зрозуміли, що зорова кора приматів містить нейрони, що здатні реагувати на маленькі ділянки зорового поля. Якщо взяти до прикладу момент, коли очі не рухаються, то область зорового простору відома як його рецептивне поле, де область впливає на збудження одного нейрону. Клітини розташовані поруч мають подібні рецептивні поля, що перекривають одне одного. Розташування та розмір рецептивних полів змінюються по всій корі, і в свою чергу формують цілком повне відображення зорового простору. Кора кожної з півкуль являє собою перехресне зорове поле.

Згорткові нейронні мережі застосуються в розпізнаванні тексту, зображень та відео-файлів, а також у рекомендаціяї, рекламі та обробці мови.

Натхненням для мене стали машини компанії Тесла, які побудовані з наявністю у них автопілота, що може розпізнавати вулицю, знаки, дорожню розмітку. Автомобіль використовує камеру і «бачить» світ, реагує на нього.

Провідні розробники в сфері стартапів IT постійно наголошують: «Якщо хочете написати цікавий продукт і гарно його продати, потрібно знайти якусь проблему і вирішити її за допомогою вашого продукту.». Моєю проблемою став вибір автомобіля. Як і багато людей, котрі не розуміються на автомобілях, я отримала купу варіантів різних виробників на огляд, і часто, зустрічаючи автомобіль десь на пів дороги на роботу, я не могла визначити виробника, а часом й марку.

З часом я зауважила, що колеги на роботі можуть побачити машину з боку, тобто небачачи її логотипа, зрозуміти що то за марка, модель і приблизно її цінова категорія, з того часу я зрозуміла, що це реально, і якщо це може зробити людина, то чому цього не може зробити штучний інтелект ?

Механізм автопілота машини Тесла використовує не лише камеру, а й купу датчиків, що надають інформацію про навколишню обстановку. Хоч датчиків я не маю, проте я вирішила написати модуль, який буде використовуватись мобільним додатком і буде розпізнавати марки автомобілів за допомогою камери.

У моїй роботі ми розглянемо відомості про проект, також обговоримо наявні рішення щодо розпізнавання об’єктів на зображеннях, обговоримо що таке згортка і оптимізація гіперпараметрів, також буде описано дослідження і пошук оптимальних гіперпараметрів нашого «вчителя» (алгоритм), що буде навчати нашу згорткову нейронну мережу.

На основі аналітичного огляду інформаційних джерел й аналізу предметної області було уточнено завдання роботи: «Створити алгоритм для навчання згорткової нейронної мережі і підібрати оптимальні параметри для розпізнавання марок автомобілів», сформульовано мету дипломної магістерської роботи: «Удосконалення та оптимізація параметрів класифікаційної моделі».

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ І ТЕОРЕТИЧНІ ОСНОВИ

1.1 Машинне навчання

Машинне навчання є визначенне як підгалузь штучного інтелекту в навчанні інформатики, що досить часто використовує прийоми із статистики для навчання комп'ютерів навчатися. Іншими словами, машинне навчання поступово покращувати продуктивність у певній задачі вткористовуючи дані, які не програмуються програмістом явно.

Ще в далекому 1959 році Артурм Семюел виявив машинне навчання і дав йому таку назву. Його лінія досліджень розглядала вивчення та побудову алгоритмів, що зможуть навчатися й робити передбачення, він намагався поширити свої алгоритми до досліджень розпізнавання образів та теорії обчислювального навчання в галузі штучного інтелект. Такі алгоритми маю неабияку перевагу над статичним кодом і данними для роботи цього коду, бо алгоритм здійснює прогнози або ухвалює рішення шляхом побудови моделі з вибіркових вхідних даних.

Сферу машинного навчання досі застосовують для великої кількості обчислювальних задач, де безповередньо програмування алгоритмів та розробка з високою продуктивністю є достатньо складною або навіть нездійсненною. Наприклад це може бути фільтрування електронної пошти, або виявляння мережних вторгників, або, наприклад, зловмисних інсайдерів, що намагаються отримати доступ до даних, для розпізнавання символів, або для навчання ранжуванню і, безпосередньо й обов’язково, для комп'ютерного зору.

Машинне вчення є дійсно тісно пов'язаним й саме по собі часто перетинається з наукою обсилювальної статистики, що також створена для прогнозуванні за допомогою застосування програмного коду і комп'ютерів[4]. Машине вчення має справді тісні зв'язки з математичною оптимізацією, що надажє цій галузі теорією, методи та прикладні області.

Машинне вчення часто поєднують з знаходженням даних, де наступна підгалузь є сфокусованою здебільшого на розвідувальному аналізі даних, й також відома як навчання без вчителя. Машинне вчення може робити помилки і бути по-справжньому спонтанним, може застосовуватись для навчання та виявлення характерів поведінки різних прогнозів, а потім застосовуватись ще й для пошуку виразних аномалій. Якщо розглядати гплузь аналізу даних, то машинне вчення - це метод, що застосовується для знаходження алгоритмів та складних моделей, які в свою чергу є прогнозуванням. В комерційному досвіді його знають як аналітика для передбачення. Такі аналітичні моделі справді дозволяють науковцям, аналітикам й інженерам виробляти результати та повторювані рішення, й відкривати такі приховані розуміння за допомогою навчання з тенденцій в даних й з співвідношень історії.

Будь хто, хто працює з машинним навчанням знає всесвітньо відомий опис Тома Мітчелла щодо машинного навчання і як воно працює: «Кажуть, що комп'ютерна програма вчиться з досвіду E по відношенню до якогось класу задач T та міри продуктивності P, якщо її продуктивність у задачах з T, вимірювана за допомогою P, покращується з досвідом E». Так, це є найточніше визначення алгоритмів, яких називають алгоритмами машинного навчання. Таке визначення задач дає фундаментальне визначення, замість того, щоб визначати дану галузь у когнітивних термінах. А праці Алана Тюрінга згадано: «Обчислювальні машини та розум», де питання: «Чи можуть машини думати?» змінюється з питанням «Чи можуть машини робити те, що можемо робити ми?». Тюрінг виклав насправді різні характеристики, що саме зможе описати «машину, що думає», та й різні наслідки для побудови такої машини.

Як правило, задачі машинного навчання вирішено поділити на дві досить великі категорії, залежно від того, чи доступний системі навчальний «знак», або «зворотний зв'язок»:

1) навчання з учителем: комп'ютерові надають приклади вхідних даних та їхніх бажаних вихідних даних, що були вказані «вчителем», і метою таких дій є навчання загального правила, що відображає вхідні дані на вихідні. Вхідний

сигнал можна розглянути як доступний лише частково в інших випадках, або обмеженим зворотним зв'язком:

- напівавтоматичне навчання: машині дають лише один неповноцінний тренувальний сигнал, іншими словами набір тренувальних даних, де деякі цільові вихідні дані є відсутні;
- активне навчання: машина може отримувати мітки тренування тільки для певного обмеженого набору типів, а також має оптимізувати свою вибірку даних для отримання міток. За інтерактивного застосування, такі міки можуть надаватися для позначення користувачеві;
- навчання з підкріпленням: тренувальні дані, що представленні у вигляді покарань і винагород, надаються тільки як зворотний зв'язок на дії алгоритму в змінному середовищі, як, наприклад, під час керування автомобілем, або у грі з опонентом.

2) навчання без учителя, спонтанне навчання: комп’ютеру для навчання не дається жодних варіантів, даючи змогу йому самому знаходити правильні дані у своєму вході. Таке навчання дійсно може бути метою саме по собі, бо воно виявляє приховані закономірності у запропонованих даних, може групувати їх за певними само-знайденими критеріями або засобами навчання ознак.

Багато науковців зі статистики використовують методи машинного навчання, створюючи об'єднауї область, яку вони назвали статистичним навчанням (statistical learning).

1.2 Застосування на відношення машинного навчання до інших областей

Можна покласифікувати машине вчення по-іншому, при розгляді бажаних вхідних даних з системи з алгоритмом навчання:

1) при розгляді класифікації вхідні дані групують на декілька класів, і алгоритм-учень змушений створити модель, що віднесе вхідні дані до цих класів. Як правило, такі речі науковці намагаються розв'язувати керованим способом. Прикладом класифікації є фільтр спам листів, де вхідними даними є повідомлення на пошті, а класом виступає класифікація «спам» чи «не спам»;
2) при розгляді регресії (англ. regression), так само у керованій задачі, вихіді дані є не дискретними, а безперервними;
3) при розгляді кластерування, набір вхідних даних повинний бути поділеним на підгрупи. На відміну від класифікації, групи не відомі насамперед;
4) оцінка густоти дізнається про розподіл вхідних данних у якомусь просторі;
5) зниження розмірності спростовує вхідні дані за допомогою відображення їх на площину меншої розмірності. Подібної задачею є тематичне моделювання, де алгоритму дають колекцію документів словами людини, і дають задачу зрозуміти, які документи поєднують подібні теми.

Еволюційне навчання (developmental learning), створене для навчання роботів, також воно створює свої особисті послідовності навчальних процесів , так званих навчальним графіком, щоб отримувати набори нових навичок за допомогою соціальної взаємодії з вчителями, тобто з людьми, і само-дослідження, застосування основних механізмів, наприклад як дозрівання, активне навчання, імітація та рухова синергія [5].

Американський піонер, Артур Семюель, започаткував термін «машинне навчання» 1959 рокуу галузі комп'ютерних ігор та штучного інтелекту, працюючи в IBM. Машинне навчання з’явилось із штучного інтелекту, як наукове прагнення.

Ще з ранніх днів деякі дослідники зацікавилися тим, щоб комп’ютери навчались з даних. Вони пробували наблизитися до розв'язання цієї задачі різними символьними способами, а також тим, чому згодом дали назву «нейронні мережі».

То були в більшості перцептрони та й інші моделі, що з часом виявилися повторними винаходами узагальнених лінійних моделей статистики. Також застосовувалося ймовірнісне міркування, зокрема в автоматизованій медичній діагностиці [1].

Проте зростання акценту на логічному, заснованому на знаннях підході викликало розрив між штучним інтелектом та машинним навчанням. Ймовірнісні системи страждали на практичні та теоретичні проблеми представлення даних та збирання [5][6]. Існувало багато теоретичних і практичні проблем отримування та показування даних у ймовірнісних системах. Близько 1980 року з'явились експертні системи, для того щоб домінувати над звичайною штучною мережею, а статистика стала здебільшого непотрібною. Праця над навчанням на базі знань та символів по справжньому продовжувалася в алгоритмах штучного інтелекту, наближаючись до логічного програмування, проте більш статистична лінія досліджень була на той час далеко від області справжнього штучного інтелекту та й безпосередньо у розпізнаванні об'єктів на зображеннях і інформаційному пошуку. Орієнтовно в той час штучний інтелект та й інформатика в цілому були позбавленні дослідженнь нейронних мереж. Ці дослідження також були продовжені поза областею інформатики, науковцями з інших дисциплін, враховуючи Хінтона, Хопфілда та й Румельхарта. Головний успіх цих видатних людей прийшов близько 1980-х років, коли вони повторно винайшли зворотнє поширення.

Машинне вчення, було створене як окрема область і з того часу воно почало сильно рости в 1990-х роках. Ця область замість того, щоб досягати цілі штучного інтелекту, почала розв'язувати не розв'язані задачі практичного характеру. Вона змістила рух від успадкованих нею від штучного інтелекту символьних підходів, до моделей і методі, які є взятті зі теорії ймовірності й статистики. Вона також має бульшу перевагу, дивлячись на її розвиток від збільшування доступності інформації та можливості розповсюджування такої інформації за допомогою Інтернету.

Машинне вчення та винайдення даних за допомогою алгоритмів дуже часто використовують одні й ті ж способи, і по-спроавжньому перекривають одне одного, але тоді як машинне вчення є базованим на прогнозі на основі відомих переваг, вивчених з даних для тренування, то добування даних базоване на відкритті невідомих даних. Це і є справжньою сходинкою аналізу з дізнавання

знань у базах даних. Знаходження даних використовує багато способів машинного вчення, проте з інакшими цілями. З одного боку, машинне навчання безпосередньо також використовує способи знаходження даних як «навчання без учителя», або як сходинка обробки для покращення точності алгоритму навчання. Ці окремі спільноти часто мають окремі конференції та окремі журнали, і велика частина невідомого між ними з'являється з основних припущень, з якими вони працюють: в машинному вченні продуктивність часто вимірюється по відношенню до здатності прогнозувати наперед відоме знання, в той час як у знаходженні знань та видобутку даних, основною і ключовою задачею є знаходження невідомого раніше знання. При оцінюванні по відношенню до наперед відомих знань некерований неінформований спосіб буде вважатись безпосередньо набагато гіршим, ніж іншим керованим методам, в той час як у типовій даній задачі керовані методи застосовуватися не мають, через відсутність даних для тренування алгоритму нейронної мережі.

Машинне навчання можна також легко пов'язувати з оптимізацією: по-справжньому багато задач навчання базуються як мінімізація деякої функції втрат на наборі тренувальниз данних-прикладів. Функції витрат показують різновид даних між прогнозами моделі, що тренуться, та справжніми зразками задачі. Для прикладу, у класифікації потрібно призначати певні властивості вхідним даним, тобто дати відповідь так/ні, щоб мережа змогла погрупувати за допомогою людини. Ці моделі тренуються правильно прогнозувати попередньо призначені відповіді набору даних. З мети узагальнення виникає різність між цими областями: в той час як алгоритми оптимізації мінімізувують витрати на колекції тренувальних даних, в інший час машинне навчення дійсно зосереджене на мінімізації витрат на нічому, тобто на відсутності аби яких підказок.

Головна мета алгоритму, що навчається, — це робити узагальнення зі свого досвіду. Це є здатність комп'ютера і його алгоритму, що вчиться, працювати точно на нових, невідомих задачах та прикладах опісля отримання результату набору даних для навчання. Тренувальні алгоритми знайденні із загалом невідомого розподілу ймовірності, що представлений представницьким для простору випадків,

і система, що вчиться, повинна побудувати узагальнену модель данного простору, що дає їй виробляти досить точні прогнози в нових випадках.

В контексті узагальнення для найкращої продуктивності складність гіпотези повинна відповідати складності функції, що лежить в основі даних. Якщо гіпотеза є менш складною за ту функцію, то модель недо-пристосувалася до даних. Якщо у відповідь складність моделі підвищити, то похибка тренування сильно знижуватиметься [7]. Однак коли гіпотеза є дійсно складною, то модель піддається перенавчанню, а узагальнення в решті решт буде безповередньо гіршим.

Аналіз алгоритмів машинного вчення та продуктивності є класом теоретичної інформатики, сьогоденно відомої як теорія обчислювального навчання [6]. Оскільки набори даних для тренування є скінченними, а майбутнє є незрозумілим, варіант навчання, прийнято вважати, не дає сто відсоткових гарантій продуктивності алгоритмів. Натомість дійсно поширеними в науці є ймовірнісні обмеження продуктивності. Загалом одним зі шляхів кількісної оцінки похибки узагальнення є компроміс дисперсії та зсуву.

Науковці обчислювального навчання, додатково до обмежень продуктивності, досліджують складність по часу та реальність здійснення навчання. У теорії обчислювального навчання обчислення вважається реальним, коли його може бути виконано за конкретно вказаний мінімальний час. Існує два види результатів складності часу. Правильні або позитивні результати вказують, що функції одного класу можуть бути навчені за конкретний час. А неправильні або негативні результати вказують, що безпосередньо конкретних класів не може бути навчено за конкретний час.

1.3 Перелік алгоритмів машинного навчання

Існує досить багато варіантів реалізації алгоритмів машинного навчання. Розглянемо найголовніші і найвідоміші наразі з них:

1) навчання дерев рішень: дерева рішень у машинному навчанні використовуються як передбачувальні моделі, що відображають знання про об'єкт (представлені гілками) у множину рішень. Це один з підходів до передбачувального моделювання у статистиці, добуванні даних та машинному навчані;
2) навчання асоціативних правил: метод машинного навчання базованого на правилах для знаходження цікавих відношень між змінними у великих базах даних. Метою є ідентифікація сильних правил, які виявляються в базах даних з використанням деяких вимірів зацікавленості;
3) штучні нейронні мережі: алгоритм навчання штучної нейронної мережі є алгоритмом, що навчається. Вони є результатом натхнення справжніми людськими нейронами в мозку. Обраховування здійснюються в термінах взаємопов'язаних класів штучних нейронів, що обробляють інформацію із використанням конективістського підходу обчислення. Сьогоденні нейронні мережі виступають як нелінійні статистичні інструменти, що моделюють данні. Також їх зазвичай використовують для моделювання досить складних взаємозв'язків між вхідними даними та вихідними даними, для знаходження структури в невідомому розподілі прогнозування величин або для пошуку закономірностей в даних [7];
4) глибинне навчання: падіння цін на продукти бізнесу та невпинний розвиток графічних процесорів протягом останніх декількох років посприяли розвитку поняття глибинного навчання, що складається з кількох прихованих шарів штучної нейронної мережі. Такий підхід моделює спосіб, яким мозок людини оброблює інформацію, зокрема наприклад світло, звук чи слух. Основним з успішних використань глибинного навчання є комп'ютерне бачення об’єктів та розпізнавання мови [7][8];

5) індуктивне логічне програмування: індуктивне логічне програмування претендує себе як підхід до навчання певних правил за допомогою зворотного поширення, застосування логічного програмування як універсального виведення вхідних даних чи прикладів та гіпотез. Маючи набір прикладів як логічна база даних та кодування відомого основного знання, система індуктивного логічного програмування виводитиме гіпотетичний логічний алгоритм, що має у наслідкх жодні з негативних прикладів і усі позитивні. Індуктивне програмування – це пов'язана область, що для представлення гіпотез розглядає будь-які види мов програмування, такі як функційні алгоритми [4];
6) метод опорних векторів: такий метод виступає як набір пов'язаних методів навчання з вчителем, що застосовуються для регресії та класифікації. Маючи набір прикладів для тренування, де кожен з них відмічено як той, що належить до однієї з двох категорій, алгоритм тренування методу опорних векторів створює модель, що передбачає, до якої категорії потрапляє новий приклад: до однієї категорії, чи до іншої [4];
7) кластерування: кластерний аналіз являє собою поділ колекції спостережень на кластери (іншими словами - підмножини), таким чином, що спостереження в обсязі однієї й такої ж підмножини є досить схожими згідно з деякими спочатку встановленими критеріями, в той час як ті ж самі спостереження, що були отримані із різних підмножин, є різними. Насправді різні методики підмножування роблять різні припущення про структуру даних, а також часто визначені конкретними мірами схожості, і даванні оцінки. Для прикладу - внутрішньою компактністю, або іншими словами, - подібністю членів одного й того ж кластеру чи підмножини, також відокремленістю між різними підмножинами. Інші методи побудовані на зв'язності графа та оцінюваній густині. Загалом кластерування – це є метод навчання без учителя, і однозначно, можна сказати, що це є поширеною методикою статистичного аналізу даних [5];

8) баєсові мережі: мережа переконань, або баєсова мережа, або спрямована ациклічна графова модель можна назвати як - ймовірнісна графова модель, що безпосередньо представляє колекцію рандомних даних та величин й їхніх умовних незалежностей прямо через спрямований ациклічний граф. От розглянемо для прикладу, те, що мережа справді може представляти ймовірнісні взаємозв'язки між симптомами та хворобами. Якщо баєсова мережа має в себе симптоми, то таку мережу можна використовувати для вираховування і прогнозування наявності різних хвороб. Сьогодні існують дійсно ефективні алгоритми для виконання прогнозування та навчання [6];
9) навчання з підкріпленням: навчання з підкріпленням слідкує за тим, як агент повинен вчиняти дії в встановленому середовищі таким шлязом, аби максимізувати піібране деяке уявлення про довготермінову винагороду. Справді, алгоритми навчання з підкріпленням пробують віднайти політику, що показує стани світу на дії, які в свою чергу цей агент змушений вчиняти в таких станах. Таке навчання з підкріпленням різниться від навчання з учителем тим, що пари правильних вхвдних даних і вихідних даних йому ніколи не показуються, і не зовсім оптимальні дії ніколи явно не виправляються [6];
10) навчання представлень: деякі алгоритми навчання, або алгоритми навчання без учителя, мають за мету знайти кращі представленя вхідних даних, що були надані під час тренування. Такі класичні приклади включають в себе ніщо як кластерний аналіз та метод головних компонент. На сьогоднішній едень алгоритми навчання представлень дуже часто намагаються зберегти усю інформацію в своїх вхідних даних, але переструктуризувати її таким способом, щоб вона стала зручною, часто як схожинка попередньої обробки перед виконанням передбачень , прогнозів чи класифікаці, роблячи можливим відбудову вхідних даних, які йдуть з невідомого розподілу, що в свою чергу породжує дані, а також у той же час не являтись обов'язково точними для конфігурацій, що є

малоймовірними за данного розподілу. Такі алгоритми навчання багатьох видів намагаються робити це за певних обмежень, для того щоб навчені представлення мали низьку розмірність. В свою чергу алгоритми розрідженого кодування пробують робити це за обмеження, щоб навчені представлення були розрідженими, а тобто вони мають мати багато нулів). Якщо ж розглядати алгоритми навчання полілінійного підпростору, то вонибезпосередньо мають на меті ніщо як навчання представлень низької розмірності безпосередньо з тензорних представлень багатовимірних даних і це має бути обов'язково без конвертації їх на багатовимірні вектори. Алгоритми глибинного навчання показують ієрархію ознак або загалом кілька рівнів представлення, в якій абстрактніші, високорівневі ознаки визначаються в межах ознак нижчого рівня або навіть породжують їх. [9][10]. Якось було висловлено думку, що розумна машина — це такий комп'ютер, що може навчатись з породженням представлення, що розплутує чинники, котрі знаходяться в основі варіацій, що описують дослідуванні спостережувані дані [10];

11) навчання подібностей та мір: у такій задачі, комп'ютеру (алгоритму), що навчається, надають декілька пар прикладів, котрі розглядаються як схжі між собою, і декілька пар менш схожих об'єктів. Комп'ютеру необхідно навчитися функції міри відстані або, іншими словами, функції подібності, котра може прогнозувати, чи є нові об'єкти схожими. Це часто використовується в рекомендаційних системах або для підбору релевантного контенту чи реклами;
12) навчання розрідженних словників: в данному методі дані потрібно представити як лінійну комбінацію базисних функцій. Взагалі ппередбачається, що коефіцієнти є розрідженими. Якщо x є d-вимірними даними, а D є матрицею d на n, де кожен стовпчик представляє базисну функцію, r є коефіцієнтом для представлення x за допомогою D. З математичної точки зору, навчання розрідженого словника означає розв'язання x ~ Dr, де r є розрідженим. Чесно кажучи, насправді n є

більшим за d, щоб дати свободу для розрідженого представлення. Наразі навчання словника разом із розрідженим представленням є досить складним для наближеного розв'язання і досить NP-складним є також. є K-CPM – один з популярних евристичним методів навчання розріджених словників. Навчання розріджених словників використовувалось в декількох контекстах. Якщо розглядати лише класифікацію, то задачею є якраз визначення, до яких класів належать раніше невідомі дані. Нехай ми будемо уявляти, що для кожного словника класів наразі вже було побудовано визначення. Отож тоді нові дані порівнюються і сприймаються як дані з таким ж класом, у словникові котрого вони розріджено представлені якнайкраще. Навчання розріджених словників використовувалось також у позбавленні шуму на зображеннях. Ключова ідея полягає в тому, що чистий клаптик зображення насправді може бути розріджено представлено словником зображеннь, а шум, в свою чергу, - не може.

13) генетичні алгоритми: генетичний алгоритм — це евристичний алгоритм пошуку, що дублює поведінку процесу природного добору, і також використовує деякі методи, зокрема такі як мутація та схрещування щоб породити новоий генотип в повній надії знайти хороші розв'язки заданої задачі. Генетичні алгоритми знаходили, зокрема в машинному навчанні, деякі застосування в 1980-1990-х роках. І в іншому порядку, методики машинного навчання були використанні для покращення рівня продуктивності еволюційних та генетичних алгоритмів.
14) машинне навчання на основі правил: це загально прийнятий термін для будь-якого методу машинного навчання, що показує, навчається або виводить конкретні «правила» для маніпулювання, зберігання знань або їх застосування. Найбільш визначальною характеристикою системи машинного навчання на основі правил є ідентифікування та використання набору реляційних правил, що сукупно але точно представляють знання, є отримані безпосередньо системою. Це і являється протилежністю до

інших систем машинного навчання, котрі звикли ідентифікувати одиничну модель, котру для отримання передбачення реально універсально застосовувати до справді будь-якого випадку. До підходів машинного навчання на основі правил належать такі підходи як навчання асоціативних правил, системи навчання класифікації та штучні імунні системи [10].

15) системи навчання класифікації: такі системи являють собою сімейство алгоритмів машинного навчання на базі правил, котрі поєднують відкриваючу складову, зазвичай, генетичний алгорит з навчальною складовою, що виконує спонтанне навчання, кероване навчання, навчання з підкріпленням [10]. Вони прагнуть віднайти спосіб, щоб ідентифікувати набір контекстно-залежних правил, котрі сукупно застосовують та зберігають знання кусковим чином, аби зробити прогнозування й передбачення.

Хоч і машинне вчення виявилося дуже змінюючим в деяких з галузей, ефективне машинне вчення є досить складним, оскільки пошук закономірностей є достатньо важким, і наявних даних для тренування часто просто не достатньо, а в результаті, алгоритми машинного вчення дуже часто не вдається здати.

Якщо потрібно затверджувати класифікувальні моделі машинного навчання, то необхідно використати такі методики як оцінювання точності методу притримування, що розбиває дані на колекцію тренувальних даних та випробувальний набір та оцінює продуктивність моделі тренування на випробувальному наборі [11].

Загалом прийнято призначати 2 / 3 колекції тренувальних даних та 1 / 3 випробувального набору.

Приклад нейронної мережі у вигляді картинки зображено на рисунку 1.3.1.

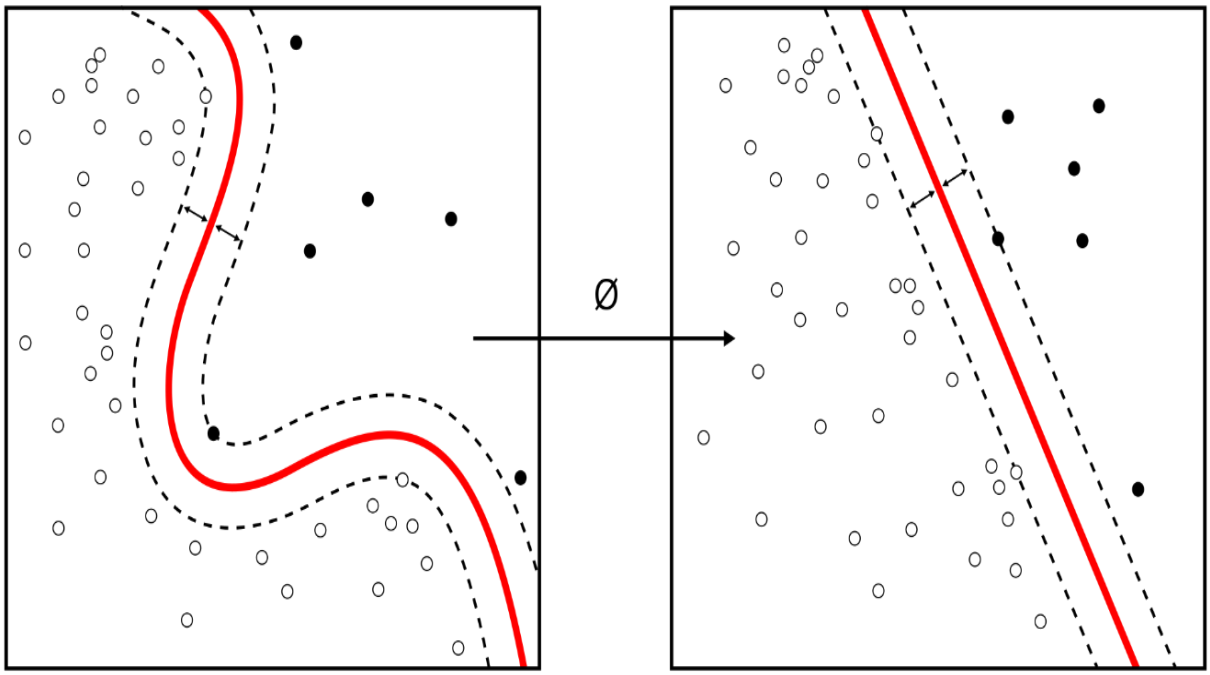


Рисунок 1.3.1 – Приклад нейронної мережі у вигляді картинки

Якщо ми хочемо порівняти, то метод k - кратного перехресного затверджування незрозумілим і випадковим способом розбиває дані на k підмножин, де k - 1 підмножин даних використовують для тренування алгоритму, в той час як k-ту використовують для спроби точності прогнозування і здатності до правильного тренування. Додатково до таких методів перехресної перевірки та притримування, для оцінювання точності моделі можна використовувати й натяжку, що сортує і обирає з набору даних n зразків із заміною.

Нейронні мережі з певною часовою затримкою дозволяють певним сигналам мовлення бути опрацьованими алгоритмами навчання інваріантно відносно часу, таким чином, аналогічно до інваріантності відносно паралельного перенесення, що запропоновують згорткові нейронні мережі. Справді, їх було представлено ще на початку 1980-х років, однак замінування вихідних даних нейронів може покривати часові періоди.

Тепер, додатково до загальної точності, науковці дуже часто наголошують про специфічність і чутливість, котрі в свою чергу означають справді позитивний та справді негативний рівні відповідно. Так само, науковці часом наголошують на хибно позитивному та хибно негативному рівнях [11][12].

Однак, такі рівні є відношеннями, котрі не мають наміру розкривти своїх знаменників і чисельників.

Робоча характеристика показує чисельники та знаменники рівнів, і надає більше інформації, ніж досить широко застосовувана робоча характеристика приймача, і безпосередньо також пов'язана з нею площа.

2 АНАЛІТИЧНИЙ ОГЛЯД ОБРОБКИ ДЖЕРЕЛ ІНФОРМАЦІЇ НЕЙРОННИХ МЕРЕЖ

Згорткові нейронні мережі (convolutional neural network) в машинному навчанні — це набір глибинних штучних нейронних мереж прямого поширення, що успішно використовується у розпізнаванні і аналізу зображень. Конструкція згорткових нейронних мережствореена на основі зорових механізмів живих організмів. Протягом 1950-тих та 1960-тих років над цим терміном працювали Г'юбел та Візел. Врешті вони зрозуміли, що зорова кора приматів містить нейрони, що здатні реагувати на маленькі ділянки зорового поля. Якщо взяти до прикладу момент, коли очі не рухаються, то область зорового простору відома як його рецептивне поле, де область впливає на збудження одного нейрону. Клітини розташовані поруч мають подібні рецептивні поля, що перекривають одне одного. Розташування та розмір рецептивних полів змінюються по всій корі, і в свою чергу формують цілком повне відображення зорового простору. Кора кожної з півкуль являє собою перехресне зорове поле.

Згорткові нейронні мережі застосуються в розпізнаванні тексту, зображень та відео-файлів, а такод у рекомендаціяї, рекламі та обробці мови [4][5].

Згорткові нейронні мережі використовують відносно мало попередньої обробки, якщо порівнбвати їх з іншими алгоритмами класифікації зображень і розпізнавання. Можна зробити висновок, що мережа навчається фільтрів, котрі в традиційних алгоритмах конструювалися руками. Така незалежність у конструюванні ознак від знань та людських зусиль є великою перевагою [13][15].

Такі нейронні мережі використовуються у розпізнаванні об’єктів на зображеннях та об’єктів на відео, підбору релевантного контенту та обробці мови спілкування людей.

Згорткова нейронна мережа складається з схожинок вхідних даних та ихідних даних, а також із декількох прихованих шарів [13][15]. Приховані шари в свою чергу у нейронній мережі часто складаються зі агрегувальних шарів,

згорткових шарів, чи наприклад повноз'єднаних шарів та шарів нормалізації. Такий процес складають в нейронних мережах як згортку за домовленням. Конкретно з математичної точки зору такий процес є шкидше взаємною кореляцією, аніж звичайною згорткою. Це має значення тільки для визначених індексів у матриці, а також необхідно звертати увагу, які ваги на якому індексі розташовуються [9][15]. Згорткові шари дуже часто застосовують до входу в операцію згортки, а також шари передають результат до наступного шару посліжовно. Згортка робить імітацію реакції окремого нейрону на зоровий стимул комп'ютера. Кожен згортковий нейрон працює з даними лише для свого рецептивного поля [13][18].

Насправді застосування цієї архітектури до зображень є непрактичним, хоч повноз'єднані нейронні мережі прямого поширення й можливо застосовувати к для навчання ознак, так і для класифікування даних [13]. Було би потрібним дуже велике число нейронів, навіть у поверхневій, іншими словами - протилежній до глибинної, архітектурі, по причині дуже великиз розмірів вхідних даних, пов'язані з зображеннями, де кожен піксель є відповідною змінною. Для прикладу, повноз'єднаний шар для певного маленького зображення розміром 100 × 100 має 10 000 ваг [13], що є досить великим навантаженням для опрацювання.

Насправді операція згортки дає змогу розв'язати таку проблему, оскільки вона зменшує кількість вільних параметрів, дозволяючи мережі бути глибшою і мати меншу кількість параметрів. Для прикладу, незважаючи на розмір зображення, області замощування розміру 5 × 5, кожна з одними й тими ж спільними вагами, потребують лише 25 вільних параметрів. Тобто можна зробити висновок, що таким чином, це розв'язує проблему вибуху градієнтів та зникання у тренуванні традиційних багатошарових згорткових нейронних мереж з багатьма шарами за допомогою зворотного поширення [5][7].

Якщо розглядати згорткові нейронні шари детальніше, то вони можуть включати сходинки локального або глобального агрегування, котрі поєднують виходи пдмножин нейронів одного шару до конкретного нейрону наступного шару. От для прикладу, максимізаційне агрегування в свою чергу використовує максимальне значення з кожного з підмножин нейронів попереднього шару [9].

Зовсім іншим прикладом є середнє агрегування, що використовує середнє значення з дійсно кожного з підмножин нейронів попереднього шару.

Справді, повноз'єднані шари поєднують кожен нейрон одного шару з кожним нейроном наступного шару. Це також є тим самим, що й традиційна нейронна мережа багатошарового перцептрону. Також, варто зазначити, що згорткові нейронні мережі використовують спільні ваги в згорткових шарах, що в свою чергу означає, що для кожного рецептивного поля шару використовується однаковий фільтр. То зменшує обсяг необхідної пам'яті для опрацювання та поліпшує продуктивність. Конструкція згорткових нейронних мереж наслідує зорові механізми в живих організмах.

2.1 Розрізнювання ознак

Тоді як існуючі моделі багатошарового перцептрону успішно використовувались для розпізнавання об’єктів на зображеннях, по причині повної з'єднаності між такими вузлами то такі моделі потерпають від проблеми розмірності, та таким чином, трохи погано масштабуються на каортинах з вищими роздільностями.

Для прикладу, в колекції CIFAR - 10 зображення мають розмір лише 32×32×3. В такій колекції ширина є 32, а висота є 32, де також є 3 канали кольору. Таким чином у першому прихованому шарі звичайної нейронної мережі один обраний повноз'єднаний нейрон матиме 32*32*3 = 3 072 ваг. Однак зображення 200 × 200 дасть такі нейроннів, що мають 200*200*3 = 120 000 ваг [14].

Варто зазначити, що такі мережеві архітектури не звертають увагу на просторову структуру даних, під час розгляду вхідних пікселів, що є далеко один від одного, таким же спочобом, як і пікселі, що є близько один від одного. Отож, можна зробити висновок, що повна з'єднаність нейронів для таких варіантів, як

розпізнавання об’єктів на зображеннях, де переважають локальні вхідні візерунки, є марнотратною [15].

Згорткові нейронні мережі є варіантами багатошарових перцептронів в природі тварин і людей. Такі мережі є розробленими для імітації поведінки зорової кори. Можна впевнено сказати, що нейронні мережі реалізовуються як спосіб бачити для комп’ютерів. Ці моделі пом'якшують виклики, поставлені архітектурою багатошарогово перцептону, використовуючи сильну просторово локальну кореляцію, присутню в природних зображеннях. На противагу до багатошаровим перцептонам, згорткові нейронні мережі мають наступні відмітні ознаки:

1) тривимірні ємності нейронів. Шари згорткових нейронних мереж вміщають в собі нейрони, котрі є впорядковані у трьох вимірах: глибина, ширина та висота (рис 2.1). Нейрони, що розташовані всередині шару, є поєднаними лише з маленькою областю попереднього шару. Його тьакож називають рецептивним полем. Щоб сформувати архітектуру згорткові нейронні мережі складають різні типи шарів, як локальні, так і повноз'єднані;

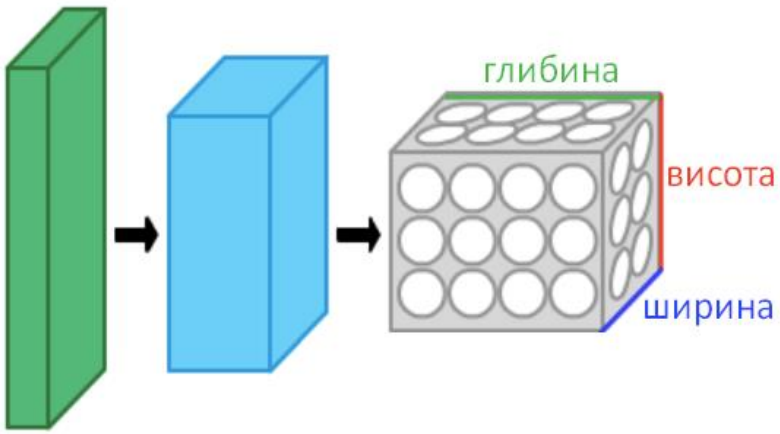


Рисунок 2.1 – Шари згорткової нейронної мережі,
розташовані в трьох вимірах

2) локальна з'єднаність: відповідно до концепції рецептивних полів, згорткові нейронні мережі користуються просторовою локальністю за

допомогою застосування способу локальної з'єднаності між нейронами сусідніх шарів. Така архітектура показує, що навчені «фільтри» дають найсильніший відгук до просторово локального вхідного образу. Безпосередньо, складання багатьох подібдних шарів веде до нелінійних фільтрів, котрі стають все більш глобальнішими, іншими словами - чутливими до більшої області піксельного простору, таким чином, що мережа з самого початку створює представлення дрібних деталей вхідних даних, а потім з них набирає представлення набагато більших областей;

3) спільні ваги: у згортковій нейронній мережі на всьому зоровому полі кожен фільтр повторюється. Такі повторні вузли використовують спільну параметризацію. Тобто вектор ваги та упередженості. Також вони формують карту ознаки. Можна сказати, що всі нейрони в даному згортковому шарі реагують на одну й ту ж саму ознаку в рамках свого рецептивного поля. Процес повторювання вузлів таким способом дозволяє ознакам бути виявленими і це відбувається незалежно від їхнього положення в зоровому полі, за допомогою забезпечення таким способом властивість інваріантності, однак відносно зсуву.

Такі, вище згаданні властивості, дозволяють згортковій нейронній мережі отримувати краще узагальнення на задачах зору. Спільне використання ваг достатньо зменшує кількість вільних параметрів, котрих навчається мережа, за допомогою знижування вимог до пам'яті для роботи алгоритму та роблячи можливим тренування більших, потужніших мереж.

2.2 Будівельні блоки

Згортковий шар (англ. convolutional layer) є основним будівельним блоком згорткової нейронної мережі (рис 2.2.1).

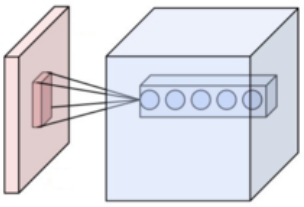


Рисунок 2.2.1 – Нейрони синього згорткового шару, котрі є з'єднані з їхнім червоним рецептивним полем

Параметри шару складаються з набору ядер або фільтрів для навчання, котрі мають маленьке рецептивне поле, однак вони простягаються на всю глибину вхідної ємності. Кожен фільтр протягом прямого проходу здійснює процес згортки за висотою та шириною вхідної ємності, також такий фільтр здійснює обчислювання скалярного добутоку даних фільтру та вхідних даних, і формуючи двовимірну карту реакції такого фільтру. Врешті решт, як результат, алгоритм навчається, котрі фільтри активуються, в той час як вона виявляє певну схожість (класифікацію чи групуваня) у певному просторовому положенні у вхідних даних [16]. Зазвичай прийнято застосовувати декілька різних типів шарів. Варто також звернути на них увагу.

Колекціонування карт реакції усіх фільтрів протягом виміру глибини формує повну ємність виходу згорткового шару. Тобто, як висновок, кожен запис в ємності вихідних даних може також трактуватися як вихідні дані нейрону, що звертає увагу на невеличку область у вхідних даних, та має спільні параметри з нейронами тієї ж карти збудження [16].

Якщо ми опрацьовуємо, наприклад, зображення, що є входом високої розмірності, то недоцільно поєднувати нейрони з всіма нейронами попередньої ємності, тому що така архітектура мережі не бере до уваги просторову структуру даних. Згорткові нейронні мережі використовують просторово локальну кореляцію за допомогою забезпечення схеми локальної з'єднаності між нейронами сусідніх шарів. Іншими словами, кожен нейрон з'єднано лише з невеликою областю вхідної ємності [17]. Простір такої з'єднаності називають гіперпараметром, що також можна назвати як рецептиве поле нейрону. Поєднання є локальними у просторі

вздовж ширини та висоти, однак завжди розширюються вздовж усієї глибини вхідної ємності. Тобто, така архітектура забезпечує, аби навчені фільтри виробляли найдоцільніший відгук до просторово локальних вхідних образів.

Графік загального використання параметрів використовується в згорткових шарах для того, щоб регулювати кількість вільних параметрів. Також схема опирається на одне досить розумне припущення, що якщо клаптикова ознака являється корисною для обчислення в певному просторовому положенні, то така ознака також змушена бути корисною і для обчислення й в інших положеннях. Можна сказати, що позначаючи двовимірний зріз за глибиною як зріз глибини, ми справді обмежуємо нейрони в кожному зрізі глибини використанням упередженості та одних і тих же ваг.

Якщо взяти до уваги, що усі нейрони, що розташовані в одному зрізі, розділяють спільну параметризацію, то прямий прохід у кожному зрізі глибини згорткового шару може бути обраховано як згортку ваг нейронів із вхідною ємністю. Так з’явилась і назва: згортковий шар. Отож, виглядає, що є досить буденним називати набори ваг фільтром або ядром, котрий згортається із входом. Карта збудження – те, що є результатом цієї згортки. Набори карт збудження для кожного з фільтрів збираються докупи вздовж виміру глибини для того, щоб отримати ємності виходу. Загальне використання параметрів призводить до інваріантності архітектури згорткової нейронної мережі відносно зсуву.

Часом спільне використання параметрів може бути безсенсовим. А здебільшого тоді, коди вхідні зображення до згорткової нейронної мережі мають конкретну особливу центровану структуру, у котрій ми очікуємо зовсім різних ознак для навчання в різних просторових положеннях. Першим із практичних прикладів є такий, коли вхід є обличчями людей, що було відцентровано в картинц. Тобто таким чином ми можемо очікувати, що будемо вчитись різних особливих ознак очей та волосся в різних частинах картинки. В такому разі замість того, щоб пом'якшувати схему спільного використання параметрів, можна просто називати шар локально з'єднаним.

Якщо ж розглядати інші важливі поняття згорткової нейронної мережі, то можна розглянути агрегування, що є різновидом нелінійного зниження дискретизації. Також існує декілька нелінійних функцій для реалізації агрегування, серед яких найпоширенішою є максимізаційне агрегування (рис. 2.2.2).

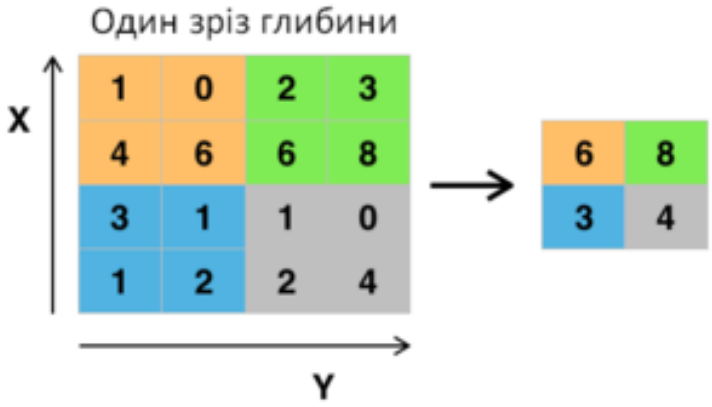


Рисунок 2.2.2 - Максимізаційне агрегування (англ. max pooling) із фільтром 2×2 та кроком = 2

Таке агрегування розділяє вхідну картинку на набір прямокутників без перекриття. Для кожної такого прямокутничка виводить її максимум. Основна ідея полягає в тому, що конкретне положення ознаки не виступає досить важливим, як наприклад її грубе положення відносно інших ознак. Такий агрегувальний шар слугує поступовому зменшенню просторового розміру представлення для скорочення кількості параметрів та об'єму обчислень у мережі. І можна сказати, що також для контролю перенавчання. У архітектурі згорткової нейронної мережі є нормою періодично вставляти агрегувальний шар між згортковими шарами, що розташовані послідовно. Процес агрегування забезпечує ще один різновид інваріантності відносно паралельного перенесення [20].

Агрегувальний шар працює незалежно на кожен зріз глибини входу, і також зменшує його просторовий розмір.

Найбільш найпоширенішим видом є агрегувальний шар з фільтрами, розмір коттрих досягає 2×2. Такі фільтри застосовуються з кроком 2, котрий зменшує дискретизацію кожного зрізу глибини входу в два рази як за висотою, так і за

шириною, відкидаючи 75 % збуджень [36]. У такому випадку кожна операція взяття максимуму діє з використанням 4 чисел. В свою чергу розмір за глибиною залишається без змін.

Агрегувальні вузли можуть використовувати й інші функції додатково до максимізаційного агрегування. Наприклад - усереднювальне агрегування та L2-нормове агрегування. Роками агрегування застосовувалася дуже часто, однак останніми роками зменшило свою популярність у порівнянні з дією максимізаційного агрегування, де робота на практиці виявилася кращою.

Тенденція йде до менших фільтрів через агресивне скорочення розміру представлення, або ж відмови від агрегувального шару взагалі [30].

ReLU – це є абревіатура від Rectified Linear Units. Такий шар застосовує ненасичувальну передавальну функцію :

$$f(x) = \max(0, x) \tag{2.2.1}$$

Агрегування областей інтересу — це такий різновид максимізаційного агрегування, де прямокутник входу є параметром, а розмір виходу є фіксовано. Насправді, агрегування є досить важливою складовою згорткових нейронних мереж для того, щоб виявляти об'єкти, котрі ґрунтуються на архітектурі швидких згорткових нейронних мереж на основі областей [18][19].

Дійсно, ReLU значно посилює нелінійні властивості мережі і функції ухвалення рішення, однак вона не зачіпає рецептивні поля свого згорткового шару. Щоб посилити нелінійність звикли застосовувати й інші функції, наприклад, насичувальні гіперболічний тангенс:

$$f(x) = tanh(x)\,, f(x) = |\ tanh(x)\,| \tag{2.2.2}$$

та сигмоїдна функція:

$$f(x) = (1 + e^{-x})^{-1} \tag{2.2.3}$$

Іншими словами, зрізаному лінійному вузлові досить часто надають перевагу у порівнянні з іншими функціями, тому що він тренує нейронну мережу набагато швидше без значної розплати точністю узагальнення [31].

В кінці кінців, максимізаційно агрегувальних шарів та після кількох згорткових шарів, високорівневі міркування в нейронній мережі здійснюються за допомогою повноз'єднаних шарів. У повноз'єднаному шарі нейрони з'єднуються з всіма збудженнями попереднього шару. Таке можна побачити у звичайних нейронних мережах. Такі їхні збудження може бути обраховано матричним множенням, за котрим слідує зсув упередженості.

Якщо ми хочемо визначити, як тренування штрафує відхилення між справжніми та передбаченими мітками, то це робить шар втрат. Для різних завдань в ньому застосовують різні функції втрат. Як правило, такий шар є останнім і завершальним шаром.

Ккспоненційні втрати використовуються для передбачення одного класу з K взаємно виключних класів. Сигмоїдні перехресно ентропійні втрати застосовуються для передбачення K незалежних значень ймовірності на проміжку від 0 до 1. А евклідові втрати використовуються для регресії до дійснозначних міток $(-\infty, \infty)$.

Згорткова нейронна мережа використовує більше гіперпараметрів, ніж стандартний багатошаровий перцептон. При оптимізації потрібно мати на увазі наступне, той час як звичайні правила для сталих регуляризацій і темпів навчання все ще використовуються:

1) кількість фільтрів: шари поруч із вхідним шаром мають менше фільтрів, оскільки карта ознак з глибиною зменшується. В той час як попередні шари можуть мати більше [37]. Прийнято підтримувати добуток ознак і піксельні позиції грубо сталим по всіх шарах, щоб вирівняти обчисленя на кожному шарі. Збереження великої кількості інформації про вхідні дані потребуватиме забезпечення, щоб кількість відображень ознак на кількість піксельних положень, а іншими словами, загальне число

збуджень, - не зменшувалося від одного шару до наступного. Кількість карт ознак напряму контролює ємність. Кільість карт ознак залежить від кількості доступних прикладів та складності завдання;

2) форма фільтрів: форми поля фільтрів сильно різняться, як відомо з літератури. В більшості вони обираються в залежності від колекції даних. Отож, таким чином, можна сказати, що безпосередньо складність полягає в знаходженні правильного рівня зернистості. Це надасть створювання абстракції в правильному масштабі для пеконкретноговного набору робочих даних;
3) форми максимізаційних агрегувань: типовими значеннями є 2×2. Однак, якщо ми працюємо з дуже великими вхідними об'єми, то вони можуть бути опрацьовані на нижчих шарах агрегування 4×4. Хоч і вибір більших форм може призводити до надмірної втрати інформації і різко знижуватиме розмірність сигналу. Та досить часто найкраще працюють вікна агрегування без перекривання.

2.3 Оптимізація гіперпараметрів

Оптимізація гіперпараметрів — задача машинного навчання для вибору колекції оптимальних гіперпараметрів для алгоритму машинного навчання [19]. Гіперпараметр – це є звичайний параметр, котрий використовується для маніпулювання процесом навчання. На відміну від значень вагових коефіцієнтів або параметрів, котрі потрібно просто вивчити. Однакові види моделей машинного навчання можуть мати різні ваги чи обмеження, або потребувати конкретної швидкості навчання для різних наборів даних. Такі параметри якраз і називаються гіперпараметрами, параметри слід підбирати так, щоб модель оптимально вирішувала завдання навчання. Для цього знаходиться кортеж гіперпараметрів, що

дає оптимальну модель, яка оптимізує задану функцію втрат на заданих незалежних даних. В свою чергу, цільова функція бере кортеж гіперпараметрів і повертає пов'язані з ними втрати.

2.3.1 Підхід: пошук по гратці

Найпоширенвішим методом оптимізації гіперпараметрів є пошук по ґратці, котру також називають ще як варіація параметрів. Такий метод просто робить повний перебір по заданій вручну підмножині простору гіперпараметрів навчального алгоритму [25].

Так як простір параметрів алгоритму машинного навчання для певних параметрів може зберігати простори з необмеженими значеннями або дійсними значеннями, тому, таким чином, реальна ситуація, коли необхідно задати дискретизацію і границю до застосування пошуку по ґратці. Такі два параметри є неперервними, тому для пошуку по ґратці обирають скінченну множину обґрунтованих значень, тобто:

$$C \in \{10, 100, 1000\},\ \gamma \in \{0.1, 0.2, 0.5, 1.0\} \qquad (2.3.1)$$

Зазвичай для порівняння використовують перехресне затверджування на наборі даних для тренуваня, або за допомогою оціни на фіксованому перевірочному наборі, тому що пошук по ґратці повинен супроводжуватися деякою мірою продуктивності. Для прикладу, звичайний класифікатор з ядровою радіально-базисною функцією та з незначним отворос на основі методу опорних векторів має як мінімум два гіперпараметри, котрі необхідно налаштувати для високої продуктивності на недоступних даних, тобто - гіперпараметр ядра γ і константа C регуляризації.

Пошук по ґратці потім перевіряє продуктивність на кожній парі вибраних параметрів на фіксованому перевірочному наборі, або за допомогою внутрішнього перехресного затверджування на наборі даних для тренування і в такому випадку декілька методів моторних векторів проганяють попарно, і також проганяє метод опорних векторів для кожної такої пари по декартовому добутку цих двох множин. Врешті решт, алгоритм пошуку по ґратці видає найкращий результат, котрий було досягнуто на процедурі перевірки [29].

Хоч і пошук по ґратці має недолік розмірності, однак він досить часто легко розпаралелюється, тому що зазвичай гіперпараметричні величини не залежать одна від одного, з якими алгоритм працює.

2.3.2 Підхід: випадковий пошук

Випадковий пошук — це така група методів числової оптимізації, котрі не потребують обчислення градієнту для розв'язування задач по оптимізації. Таким чином, випадковий пошук може бути застосований для функцій, які не є диференційованими або неперервними [38]. Схожі методи оптимізації безпосередньо називаються методами без використання похідної, методами прямого пошуку або методами «чорної скриньки». Прийнято вважати автором назви випадкового пошуку Растригіну. Він вперше почав роботу з такими методами базуючись на базовому математичному аналізі. Випадковий пошук працює за допомогою послідовного просування в пошуковому просторі між кращими позиціями. Кращі позиції в свою чергу обираються з гіперсфери, який оточує поточну позицію [20].

В літературі існує декілька варіантів випадкового пошуку:

1) випадковий пошук із фіксованим кроком — це такий базовий алгоритм, написаний Растригіном. Алгоритм обирає нові позиції з гіперсфери маючи заданий фіксований радіус;
2) випадковий пошук з оптимальним кроком — це є певна теоретична викладка з визначення оптимального радіусу гіперсфери. Така викладка створена для пришвидшеної конвергенції з оптимумом. Застосування данного методу вимагає апроксимації цього оптимального радіусу за допомогою багаторазової дискретизації. Можна сказати, що він є занадто вимогливим до ресурсів для застосування на практиці;
3) випадковий пошук з адаптивним кроком — складний алгоритм, який може евристично адаптувати радіус гіперсфери. Однак, даний алгоритм досить сильно ускладнений;
4) випадковий пошук із оптимізованим відносним кроком — такий пошук апроксимує оптимальний розмір кроку за допомогою простого експоненційного зменшення. Однак його формула для обчислення коефіцієнту згладжування дуже сильно ускладнена.

Можна сказати, що випадковий пошук замінює конкретний набір усіх комбінацій на їх випадковий вибір. Таке можна легко використати до дискретних випадків, про які ми говорили раніше. Хоча й метод можна узагальнити на змішані і неперервні простори. Такий пошук може бути оптимальнішим і кращим вибором, якщо обирати між ним і пошуком по ґратці, зокрема, коли лише маленька кількість гіперпараметрів впливає на оптимальність алгоритму машинного навчання.

2.3.3 Підхід: баєсова оптимізація

Байєсова оптимізація — це один з методів глобальної оптимізації для певної невідомої функції з шумом, яка виступає як чорний ящик. Якщо застосовувати

баєсівську оптимізацію до гіперпараметричної оптимізації, то баєсівстка будує стохастичну модель функції відображення використовуючи значення гіперпараметрів в цільову функцію. Така цільова функцію може бути застосована на затверджувальному наборі. За допомогою ітеративного застосування такої перспективної, заснованої на поточній моделі конфігурації гіперпараметрів, її оновлення, то можна сказати, що таку байєсівська оптимізація хоче зібрати якомога більше інформації про данну функці. Іншими словами вона прагне отримати місце оптимуму. Метод намагається збалансувати дослідження і використання, тобто гіперпараметри, для котріх результат найменш відомо і також гіперпараметри, котрі, в свою чергу, найбільше близькі до оптимуму [21]. У практиці й розробці байєсівська оптимізація показала найкращі результати, використовуючи менші обчислення, якщо порівнювати її з випадковим пошуком і пошуком по ґратці завдячуючи можливості судження про практичну точність експериментів ще до того, як вони будуть виконанні.

2.3.4 Підхід: оптимізація на основі градієнтів

Можна обрахувати і оптимізувати градієнт гіперпараметрів шляхом градієнтного спуску для конкретних алгоритмів навчання. Коли таке вирішили робити вперше, то ці техніки були задіяні з використанням нейронних мереж. Згодом ці методи були запозиченні іншим моделям, наприклад як логістична регресія чи метод опорних векторів.

Інакший спосіб використання градієнтів гіперпараметрів полягає в диференціюванні кроків алгоритму ітеративної оптимізації шляхом автоматичного диференціювання.

2.3.5 Підхід: еволюційна оптимізація

У розділі еволюційного моделювання існують еволюційні алгоритми. Такі алгоритми це є варіант реалізації алгоритмів шляхом застосування штучного інтелекту, що моделює і використовує біологічну еволюцію. Вирізняють багато різних алгоритмів: генетичне програмування, системи класифікаторів, генетичні алгоритми, еволюційні стратегії, еволюційне програмування тощо. Усі вони моделюють фундаментальні положення в теорії біологічної еволюції. Наприклад процеси відтворення, мутації і відбору. Поведінка агентів базується на довкіллі [21][23]. Колекцію агентів прийнято називати популяцією. Така популяція еволюціонує згідно з правил відбору базуючись на цільовій функції. Така функція задається довкіллям. Отже, таким шляхом, кожному індивідуумові або агентові популяції призначається значення його потрібності в навколишньому середовищі. Розмножуються лише найбільш придатні види. Мутація і рекомбінація дозволяють агентам пристосовуватися і змінюватись до середовища. Такі алгоритми належать до адаптивних пошукових механізмів [22].

Можна впевнено зауважити, що уволюційні алгоритми мають широке застосування на практиці. Однією з найпопулярніших сфер їх використання є комбінаторна оптимізація. Еволюційні алгоритми були використанні для розв'язання базових NP-повних проблем, наприклад: розбиття чисел, задача комівояжера, максимальна незалежна множина, задача пакування рюкзака та розфарбовування графів [26].

Планування, обчислення маршрутів, складання розкладів, задачі транспортування і розташування - не класичі задач, для розв'язання яких застосовують еволюційні алгоритми також. Якщо розглядати оптимізації структур та елетронних схем, наприклад і економіці чи медицині, то еволюційні алгоритми використовують для того також.

Для передбачення кристалічних структур останніми роками також часто роблять вибір у сторону використання еволюційних алгоритмів з допомогою програмного забезпечення USPEX.

У Австрії насамперед активно досліджується можливість використання еволюційних алгоритмів у сфері музики, зокрема при спробі відтворення і моделювання гри на музичних інструментах видатними музикантами різних епох і століть.

Як ми казали раніше, еволюційну оптимізацію можна назвати методологією для повної оптимізації невідомих функцій з шумом. Еволюційна оптимізація використовує еволюційні алгоритми під час оптимізації гіперпараметрів для пошуку гіперпараметрів для даного алгоритму. У данному випадку наголошують, що завдання оптимізації має низьку внутрішню розмірність. Випадковий пошук також досить легко паралелізується і, безпосередньо, є можливим використання попередніх даних за допомогою вибору розподілу для вибірки випадкових параметрів.

Еволюційна оптимізація гіперпараметрів дотримується процесу, навіяного біологічною концепцією еволюції:

1) генеруємо випадковий кортеж гіперпараметрів, тобто створюємо початкову популяцію випадкових рішень;
2) отримуємо функцію кортежів гіперпараметрів допасованості і оцінюємо їх. Для прикладу, за допомогою десятикратної точності перехресного затверджування програми машинного навчання з данними гіперпараметрами);
3) сортуємо і встановлюємо кортежі гіперпараметрів згідно з їх відносній придатності;
4) ставимо нові кортежі гіперпараметрів які були створенні шляхом схрещуванням і мутації замість кортежів гіперпараметрів з гіршою продуктивністю;

5) повторюємо вище згадану послідовність дій, поки не отримаємо ту продуктивність алгоритму, яка нам потрібна або поки продуктивність не перестане поліпшуватися до оптимального рівня.

Еволюційна оптимізація застосовується для оптимізації гіперпараметрів для статистичних алгоритмів у автоматизованому машинному навчанні, для пошуку архітектуриглибоких нейронних мереж, а також для формування ваг в глибоких нейронних мережах [24].

Процес навчання на базі заселення займається вивченням значенням гіперпараметрів і вагів мережі. Кілька процесів навчання працюють незалежно, за допомогою використання різних гіперпараметрів. Моделі, котрі досить погано працюють, ітеративно замінюються моделями, котрі застосовують модифіковані значення гіперпараметрів кращого виконавця [39]. Змінювання дає можливість еволюційно змінювати гіперпараметри і також безпосередньо виключає необхідність ручного налаштування гіперпараметрів. Данний процес не робить припущень щодо функції втрат, чи то архітектури моделі, або ж процедури навчання.

Відбувається також розвиток методів на базі радіально-базисної функції і спектрального методу.У порівнянні з галузями даних зображень, роботи із застосування згорткової нейронної мережі до класифікації відео є відносно досить мало. Для прикладу, зрозуміло, що відео є досить складнішим за зображення, тому що воно має ще й часовий вимір. Хоча, тим не менше, розширення згорткової нейронної мережі до області відео було наразі досліджено. Одним із підходів – це є можливість трактувати час і простір як рівноцінні виміри входу, а також безпосередньо виконувати згортку за часом і за простором. Злиття ознак двох згорткових нейронних мереж є іншим підходом. Однієї ознаки для просторового і також однієї ознаки для часового потоків. Також було представлено способи рандомного навчання для тренування методу незалежних підпросторів і просторово-часових ознак на базі згорткових вентильних обмежених машин Больцмана.

2.4 Застосування згорткових нейронних мереж

Згорткові нейронні мережі досить часто використовують у системах розпізнавання об’єктів на картинках. Проягом 2012 року було з’ясовано, на базі даних MNIST є рівень похибки в що 0.23-відсотковий рівень. Ще одна праця про застосування згорткової нейронної мережі для групування об’єктів на картинках заявила, що процес навчання був «досить швидким». В такій ж праці було досягнуто найкращих результатів на базах даних MNIST та NORB із опублікованих на 2011 рік результатів [25].

Згорткові нейронні мережі досягли великого зниження рівня похибки при застосуванні до розпізнавання облич. Досліження інших людей повідомила про 97.6-відсотковий рівень розпізнавання на «5 600 нерухомих зображеннях понад 10 суб'єктів». Безпосередньо, згорткову нейронну мережу застосовували для оцінки якості відео об'єктивним шляхом, зрозуміло, що опісля тренування вручну. Також, отримана в результаті система мала дуже низьку кореневу середньоквадратичну похибку [26].

Широкомасштабне випробування з візуального розпізнавання ImageNet є першим еталоном у класифікації та виявленні об'єктів на картинках, з сотнями класів об'єктів і мільйонами зображень. У 2014 році, великомасштабному змаганні з візуального розпізнавання, практично кожна команда використовувала згорткову нейронну мережу як свою основну схему, команда, котра досягла високого рівня розвитку [27].

Переможець, GoogLeNet знизив похибку класифікації до 0.06656 і збільшив очікувану середню точність виявлення об'єктів до 0.439329. То був найкращий результат на той момент. Його мережа використовувала більше ніж 30 шарів. Така продуктивність згорткових нейронних мереж у завданнях ImageNet була практично як у людей. Найкращі реалізації мережі продрвжують битися з об'єктами, які є дуже

маленькими або дуже тонкими, такими як маленька мурашка на листочку, або людина, що тримає зернятко у руці. Такі реалізації також мають проблеми із зображеннями, яке було спотворено з застосуванням фільтрів, що є досить часто використовуваним в наш час сучасних цифрових камер.

Такі типи зображень рідко викликали складність розуміння в людей. Хоч і люди схильні мати проблеми з іншими питаннями [26]. Для прикладу, люди не є дуже вправні у групуванні об'єктів на досить тонкі категорії, наприклад як окремі породи собак або види птахів, а згорткові нейронні мережі це роблять як легку задачу.

Протягом 2015 року багатошарова згорткова нейронна мережа показала здатність з досить швидко, точно і оптимально помічати обличчя з великого діапазону кутів, навіть із перевернутими обличчями і частково прикритимии шарфиком чи маскою. Така прекрасна мережа тренувалася на базі даних із використанням 200 000 картинок, де були зображенні обличчя у різних орієнтаціях і під різними кутами, і ще також з використанням двадцяти мільйонів картинок без облич. Данні алгоримтми використовували пакети із 128 зображенями у 50 000 ітерацій [28].

Сама собою нейромережа не варта абсолютно нічого, якби дивно це не звучало . Для того, щоб вона могла видавати якийсь результат – її потрібно навчити.

Взагалі, що таке навчання нейромережі? Це процес, в якому її параметри налаштовуються за певним алгоритмом (цей алгоритм називається алгоритмом навчання) таким чином, щоб вона на виході давала правильні результати.

Розрізняють алгоритми навчання з вчителем та без вчителя (рис 2.4.1, 2.4.2). Процес навчання з вчителем застосовується тоді, коли вхідні дані вже розділені на певні класи. Тобто нейромережа знає правильні відповіді й на їх основі підлаштовує свої параметри.

Отже, на вхід нейромережі поступає зображення. Далі, після «магії», що відбувається всередині, на виході ми отримуємо клас даного зображення [26][27]. Оскільки для кожного вхідного зразка нейромережа знає правильну відповідь, то

за допомогою функції втрат ми можемо порахувати помилку, тобто різницю між виходом нейромережі та правильним результатом, а вже після цього підкоригувати вагові коефіцієнти.

Рисунок 2.4.1 – Розподіл

Ці кроки повторюються, поки помилка не буде в допустимих межах (або не закінчиться кількість епох). Якщо вхідні дані розбиті на класи, то ми вирішуємо задачу класифікації. Класичними алгоритмами навчання є алгоритм зворотного поширення помилки та метод градієнтного спуску, які і було використано.

При навчанні без вчителя нейромережа не знає правильних відповідей. Тому, за певними закономірностями чи подібністю даних сама відносить їх до певних класів (кластерів). Ця задача називається кластеризацією. Кластеризація в декілька разів програє за точністю класифікації. Справді необхідно наголосити, що складна багатошарова мережа дійсно може навчитися видавати правильний результат на даному прикладі. Хоч з використанням такої логіки роботи їй доводиться обходити цю хибність класифікації через багато проміжних шарів, а це в свою чергу вимагає більшої кількості навчальних ваг і великої глибини мережі. Іншими словами така логіка пошуку об’єктів не є фундаментально вірною з біологічної точки зору [28].

Для зменшення розмірності зображення застосовують шар пулінгу (субдискретизації). Останнім шаром іде повнозв'язний шар, що формує вихід нейромережі.В найпростішому випадку архітектура згорткової нейронної мережі - набір шарів, які перетворюють образ зображення у якийсь можливий інший вихідний образ.

Рисунок 2.4.2 – Процес навчання з вчителем

Кожен шар відповідає за певний етап процесу обробки зображення. Кожен шар отримує на вході об'ємну 3D інформацію і перетворює її зі збереженням 3D-об'єму за допомогою певної функції [29].

Логічний сенс згортки такий – чим більше величина елемента згортки, тим більше ця частина матриці A була схожа на матрицю B (схожа всенсі скалярного добутку). Тому матрицю A називають зображенням, а матрицю B – фільтром.

Неодмінно, згорткові мережі можуть включати в себе шари глобального агрегування чи то локального агрегування, котрі поєднують виходи кластерів нейронів одного до наступного шару.

Зазвичай поширені в літературі форми поля фільтрів вибираються в залежності від набору даних і досить сильно різняться.

Тобто можна впевнено сказати, що складність полягає в знаходженні правильного рівня зернистості, для того щоб мати змогу створювати абстракції для певного набору даних в правильному масштабі.

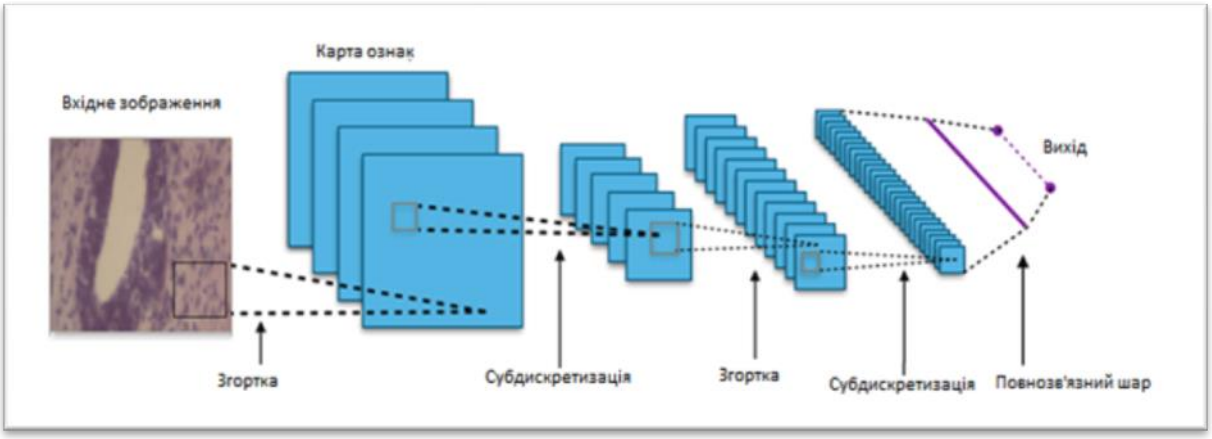


Рисунок 2.4.3 – Структура згорткової нейронної мережі

Наприклад, максимізаційне агрегування використовуємаксимальне значення з кожного з кластерів нейронів попереднього шару [40]. Зовсім інакшим прикладом є усереднене агрегування, що в свою чергу використовує усереднене значення з кожного кластеру нейронів у попередньому шарі.

2.5 Згортка (математичний аналіз)

На сьогоднішній день, процедура пов'язані операції із згорткою і безпосередньо самі згортки знаходять багато використання у інженерії, науці та математиці.

Згортка (англ. convolution) — математична операція деяких двох функцій f(t) та g(t), що дозволяє отримати третю функцію:

$$(f * g)(t) = \int_{-\infty}^{\infty} f(t-r)g(r)dr \qquad (2.5.1)$$

Головна і базова властивість згортки – це можливість робити фур'є-образ згортки пропорційним добутку фур'є-образів функцій.

Нехай G — група Лі, оснащена мірою Хаара m, і $f, g : G -> R$ — дві функції, визначенні на G.

Тоді їх згорткою називається функція:

$$f * g(x) = \int_G f(y)g(xy^{-1})\, m(dy), \quad \forall x \in G \qquad (2.5.2)$$

Пов’язані операції зі згорткою і безпосередньо самі згортки знаходять дуже багато використань в математиці, науці, інженерії тощо.

Наразі досить поширеною практикою стало застосування згортки у обробці зображень за допомогою багатьох фільтрів, наприклад для виявлення контурів чи розмиття.

Незфокусована фотографія є згорткою чіткого зображення з функцією лінзи. Прийнято називати фотографічний термін для цього поняття як Боке.

Також, неодмінно у статистиці, зважене рухоме середнє є нічим іншим як згорткою.

В свою чергу згорткові нейронні мережі застосовують багато каскадів ядер згорток щоб безпосередньо застосовуватись в областях штучного інтелекту і комп'ютерного зору [30].

3 РЕАЛІЗАЦІЯ АЛГОРИТМУ З ПІДБОРОМ ОПТИМАЛЬНИХ ГІПЕРПАРАМЕТРІВ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

3.1 Огляд існуючого рішення і його недоліки

Згорткові нейронні мережі часто застосовують у системах розпізнавання зображень (рис 3.1.1). Протягом 2012го року було повідомлено про 0.23-відсоткову похибку на базі даних MNIST. Ще одна праця про застосування згорткової нейронної мережі для групування зображень на класи повідомила, що процес навчання проходив достатньо швидко. Також, що було досягнуто найкращих із опублікованих на 2011 рік результатів на базах даних MNIST та NORB.

Рисунок 3.1.1 – Наочний приклад як працює розпізнавання

Згорткові нейронні мережі досягли великого зниження рівня похибки при застосуванні до розпізнавання облич. Інакша праця дослідників повідомила про 97.6-відсотковий рівень розпізнавання при використанні 5 600 нерухомих

картинок понад 10 суб'єктів. Згорткові нейронні мережі застосовуались після тренування вручну для оцінки рівня якості відео об'єктивним способом. В результаті отримана система мала досить низьку кореневу середньоквадратичну помилку.

Широкомасштабне випробування з візуального розпізнавання ImageNet є номер один у виявленні об'єктів і класифікації, з використанням сотнів класів об'єктів та мільйонів зображень. У 2014 році, в великому і машбатному змаганні з візуального розпізнавання об’єктів на картинках, практично кожна команда, котрі досягла високого рівня, використовувала згорткові нейронні мережі як свою фундаментальну систему. Переможець, GoogLeNet (основа DeepDream), отримав середню влучність близько 0.439329 для виявлення об'єктів і безпосередньо знизив похибку класифікації до 0.06656. Це був найкращий результат на той момент. Створена ним мережа використовувала близько 30 шарів [31].

Данна продуктивність згорткових нейронних мереж представленна у завданнях ImageNet була досить наближенною до людської. Наразі найдосконаліші алгоритми продовжують битись з об'єктами, які є досить тоненькими або маленькими, наприклад як маленький мураха на листочку, чи то людина, котрі тримає зернятко в руці. Але на превеликий жаль вони мають проблеми із зображеннями, на які було накладено фільтри, а це в свою чергу є дуже поширенним явищем з використанням сучасних цифрових камер. Але в свою чергу люди досить чітко і точно розпізнають данні зображення і рідко отримують утруднення. Однак на відмінну від переваги людей над мережею, вони схильні мати проблеми з іншими моментами розпізнавання. Наприклад, люди не досконалі у класифікуванні об'єктів на маленькі групки чи категорії, наприклад як окремі породи котиків або види пташок, але безпосередньо для згорткових нейронних мереж - це є досить легка задача.

З часом, 2015 року багатошарова згорткова нейронна мережа показала можливість розпізнвати досить швидко обличчя з великого діапазону кутів, наприклад перевернуті обличчя або частково прикритими.

Така мережа тренувалася на базі тестових даних, що вміщала в себе близько 200 000 картинок, де 20 мільйонів зображень без облич, а інші включали обличчя в різних орієнтаціях і під різними кутами. Такі мережі застосовували пакети зі 128 зображень у 50 000 ітерацій.

Мобільний додаток відсилає інформацію з камери, коли користувач буде наводити камеру на автомобіль, і передає цю інформацію на сервер, де безпосередньо є в дії наш алгоритм.

Задача алгоритму – швидко навчити комп'ютер розпізнавати марки автмобілів і повертати результат. Після отримання результату програма може перенаправити користувача мобільного додатку на сайт бренду автомобіля.

Щоб навчити нейронну мережу потрібно використати велику кількість картинок, які ми будемо шукати в інтернеті і безпосередньо віддавати алгоритму на навчання.

Створюючи модель глибокого навчання, нам доведеться зробити багато, здавалося б, довільних рішеннь: Скільки шарів слід скласти? Скільки одиниць або фільтрів має входитикожен шар? Чи слід використовувати relu як активацію або іншу функцію? Якщо ми використовуємо BatchNormalization після заданого шару? Скільки відсіву слід використовувати? Ці параметри архітектурного рівня називаються гіперпараметрами, що відрізняються від параметрів моделі, які тренуються шляхом зворотного розповсюдження [32].

На практиці досвідчені інженери машинного навчання та дослідники з часом формують інтуїцію щодо того, що працює, а що ні, коли йдеться про такий вибір - вони розвивають навички налаштування гіперпараметрів. Але офіційних правил немає.

Якщо ви хочете дійти до межі того, чого можна досягти за певним завданням, ви не можете бути задоволеними з довільним вибором, зробленим помилковою людиною. Ваші початкові рішення майже завжди неоптимальні, навіть якщо у вас хороша інтуїція. Ви можете уточнити свій вибір, налаштувавши їх вручну та повторно перекваліфікувавши модель - саме на це проводять більшу частину часу інженери та дослідники машинного навчання. Але це не повинно бути вашою роботою людини, щоб цілими днями возитися з гіперпараметрами - це краще довірити машині.

Таким чином, нам потрібно досліджувати простір можливих рішень автоматично, систематично, принципово. Нам потрібно шукати архітектурний простір і емпірично знаходити найкращі з них. Саме в цьому полягає сфера автоматичної оптимізації гіперпараметрів: це ціла область досліджень, і важлива.

Процес автоматизації гіперпараметрів виглядає приблизно так:

1) вибрати набір гіперпараметрів (автоматично);
2) побудувати відповідну модель;
3) пристосувати його до своїх навчальних даних і виміряти остаточну ефективність на валідації даних;
4) вибрати наступний набір гіперпараметрів, щоб спробувати (автоматично);
5) повторити;
6) зрештою, виміряти ефективність даних тесту.

Ключем цього процесу є алгоритм, який використовує цю історію виконання перевірки, враховуючи різні набори гіперпараметрів, для вибору наступного набору гіперпараметрів для оцінки. Можливо багато різних методів: байєсівська оптимізація, генетичні алгоритми, простий випадковий пошук тощо.

Навчання ваг моделі порівняно просто: обчислюємо функцію втрат на міні-партії даних, а потім використовуємо алгоритм зворотного розповсюдження для переміщення ваг у правильному напрямку. З іншого боку, оновлення гіперпараметрів надзвичайно складно [32][33]. Розглянемо наступне:

1) Обчислення сигналу зворотного зв'язку (чи цей набір гіперпараметрів призводить до високоефективної моделі для цього завдання?) Може бути надзвичайно дорогим: для створення набору даних потрібно навчити нову модель;
2) Простір гіперпараметрів, як правило, складається з дискретних рішень і, отже, не є безперервним або диференційованим. Отже, ми не можемо здійснити градієнтний спуск у просторі гіперпараметрів. Натомість мам слід покластись на безградієнтні методи оптимізації, які, природно, набагато менш ефективні, ніж градієнтний спуск.

Оскільки ці виклики важкі, а сфера все ще молода, ми наразі маємо доступ до дуже обмежених інструментів для оптимізації моделей. Часто виявляється, що

випадковий пошук (вибір гіперпараметрів для випадкової оцінки багаторазово) є найкращим рішенням, незважаючи на те, що він є найбільш наївним.

Але одним із інструментів, який я знайшлає надійнішим, ніж випадковий пошук, є Hyperopt (https://github.com/hyperopt/hyperopt), бібліотека Python для оптимізації гіперпараметрів, яка внутрішньо використовує дерева оцінювачів Parzen для прогнозування наборів гіперпараметрів, які, ймовірно, спрацюють Ну. Інша бібліотека під назвою Hyperas (https://github.com/maxpumperla/hyperas) інтегрує Hyperopt для використання з моделями Keras [34].

Тюнінг моделей машинного навчання є одним із завдань оптимізації. У нас є набір гіперпараметров, і ми прагнемо знайти правильну комбінацію їх значень, яка може допомогти нам знайти або мінімум (наприклад, втрати), або максимум (наприклад, точність) функції (рис. 3.1.2).

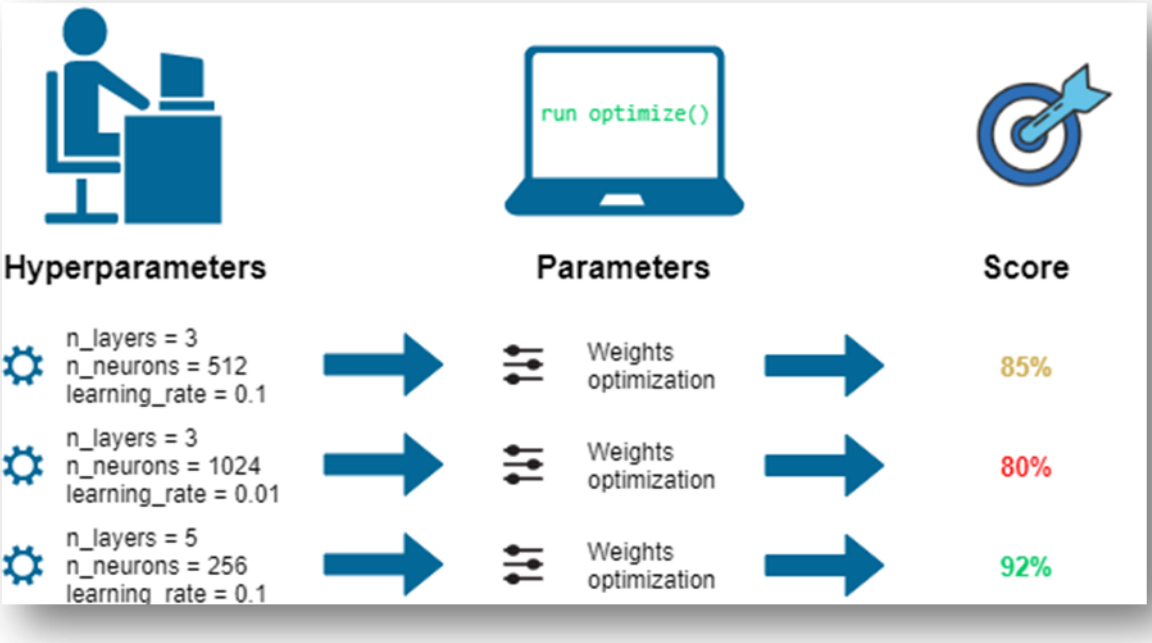


Рисунок 3.1.2 – Процес оптимізації

Це може бути особливо важливо при порівнянні того, як різні моделі машинного навчання працюють з набором даних.

Фактично, було б несправедливо, наприклад, порівнювати модель SVM з кращими параметрами (гіперпараметрами) з моделлю випадкового лісу, яка не була

оптимізована [35].

Тут ми пояснимо такі підходи до оптимізації гіперпараметрів:

1. ручний пошук;
2. Випадковий пошук;
3. Grid Search;
4. Автоматичненалаштування гіперпараметрів (Байєсова оптимізація, генетичні алгоритми);
5. Налаштування штучних нейронних мереж (ANNs).

Щоб продемонструвати, як виконати оптимізацію гіперпараметрів в Python, я вирішила виконати повний аналіз данних. Наша мета - правильно класифікувати, які автомобілі є конкретної моделі (наприклад Мерседес), які ні.

В цьому випадку я вирішила використовувати тільки підмножину набору даних, щоб прискорити час тренувань і забезпечити ідеальний баланс між двома різними класами. Крім того, тільки обмежена кількість функцій була використана, щоб зробити завдання оптимізації більш складними. Остаточний набір даних показаний на малюнку нижче (рис. 3.1.3).

	V17	V9	V6	V12	Class
0	0.207971	0.363787	0.462388	-0.617801	0
1	-0.114805	-0.255425	-0.082361	1.065235	0
2	1.109969	-1.514654	1.800499	0.066084	0
3	-0.684093	-1.387024	1.247203	0.178228	0
4	-0.237033	0.817739	0.095921	0.538196	0

Рисунок 3.1.3 – Остаточний набір даних

Отже, у нас стоїть завдання виділити на зображенні якийсь об'єкт - автомобіль. Людина легко розуміє що перед нею автомобіль і розпізнає його за тисячею дрібних

ознак. Але як навчити машину що "цей набір точок на зображенні – автомобіль"? Відповідь на це питання лежить не в понятті згорткової мережі - з цим завданням може впоратися і найстаріша нейронна мережа на персептроні.

Згорткова ж нейронна мережа за рахунок застосування спеціальної операції – власне згортки – дозволяє водночас зменшити кількість інформації, що зберігається в пам'яті інформації, за рахунок чого краще справляється з картинками більш високої роздільної здатності, і виділити опорні ознаки зображення, такі як ребра, контури або грані [20]. На наступному рівні обробки з цих ребер і граней можна розпізнати повторювані фрагменти текстур, які далі можуть скластися в фрагменти зображення.

Першим і, по суті, найбільш тривіальним завданням, яке навчилися вирішувати за допомогою нейронних мереж, стала класифікація зображень (рис. 3.1.4). Як ми вже писали вище, більшість прикладів вказано на зображеннях. Але, по суті, можна класифікувати будь-які сигнали.

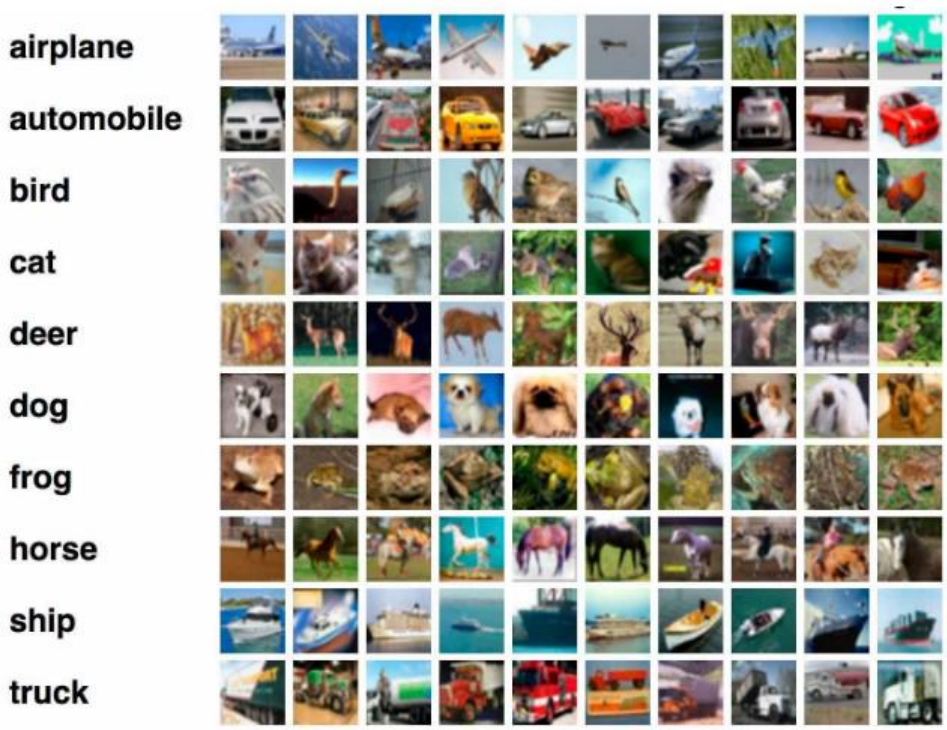


Рисунок 3.1.4 –Розпізнавання об’єктів

По суті кожен шар нейронної мережі використовує власне перетворення. Якщо на перших шарах мережа оперує такими поняттями як "ребра", "грані" і т.п, то далі

використовуються поняття "текстура", "частини об'єктів". В результаті такого опрацювання ми можемо правильно класифікувати картинку або виділити на кінцевому етапі потрібний об'єкт на зображенні.

Згорткові нейронні мережі застосовуються досить широко і в різних областях. Нижче ми розглянемо прості прикладні приклади того, як можна застосовувати GAN і CNN в бізнесі.

Класифікації за допомогою CNN активно застосовуються в медицині: можна навчити нейронну мережу класифікації хвороб або симптомів, наприклад, для МРТ-діагностики (рис.3.1.5).

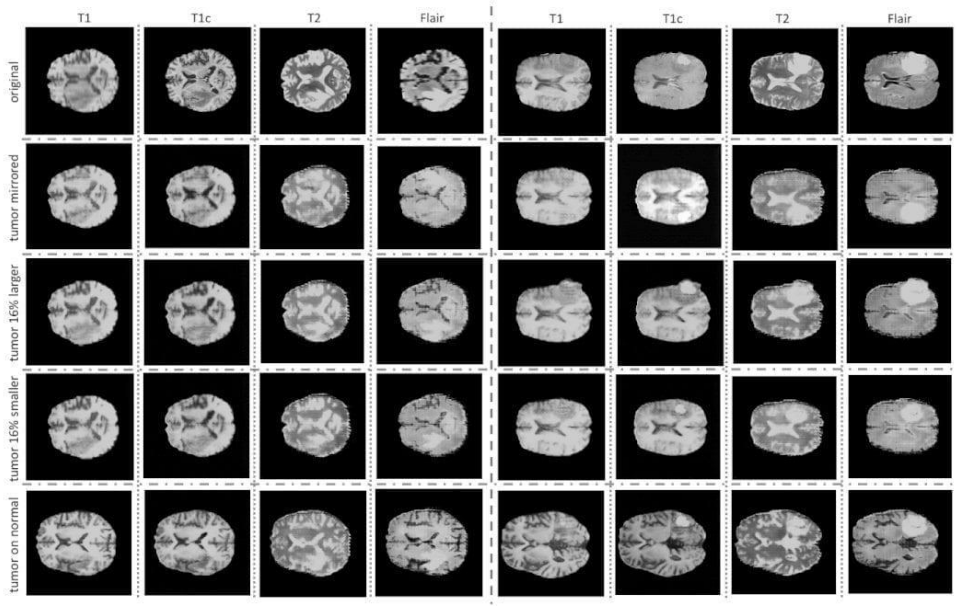


Рисунок 3.1.5 – Застосування в медицині

В агробізнесі розробляється і впроваджується методика аналізу та розпізнавання зображень, при якій дані отримують від відкритих супутників, таких як LSAT, і використовують для прогнозування майбутньої врожайності конкретних земель (рис. 3.1.6).

Розпізнавання об'єктів на фото і відео за допомогою нейронних мереж застосовується в безпілотному транспорті, відеоспостереженні, системах контролю доступу, системах "розумного будинку" і так далі.

Візуальний пошук – така написана технологія, котра дозволяє користувачеві фотографувати об’єкт або просто наводити на нього камеру, і в свою чергу програма

за декілька секунд розпізнає товар, на який було наведену камеру і шукає його в інтернет-магазині, повідомляє про його характеристики й ціну [4].

Такий спосіб застосування має очевидну складність людського фактору, чи то досвід прауюючого персоналу. Для прикладу, лише працюючий менеджер, маючи достатній досвід роботи, має змогу розпізнавати запчастини до авто, погризений собакою пульт із напівз'їденими кнопками. Однак лише працівник з досвідом може по картинці визначити комаху, яка вас укусила. А в свою чергу недавно найнятий на роботу працівник із таким завданням не зможе впоратись, бо його ще треба буде довгий час навчати.

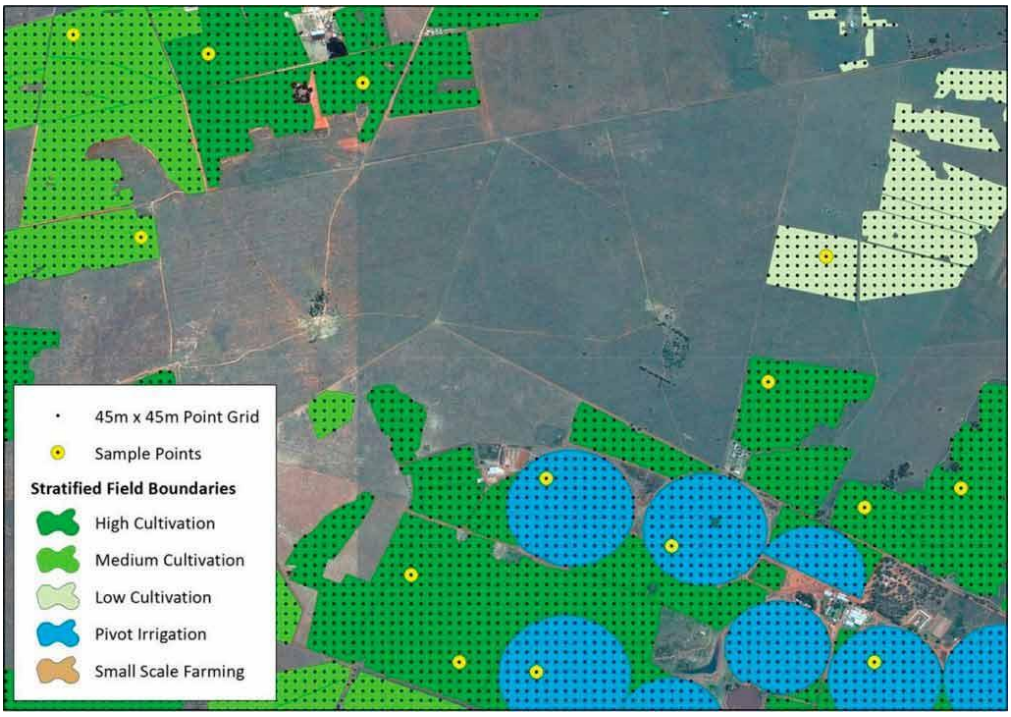


Рисунок 3.1.6 – Розпізнавання в агро-бізнесі

Нище ми можемо бачити технології, за допомогою яких будується програма для розпізнавання об'єктів на зображенях:

1. tensorFlow – це фреймворк для реалізації машинного навчання, на ньому можна створити нейронну мережу. Можна також застосувати Faster-RCNN-Inception-V2 – це така модель, яка дуже підходить для навчання згорткової нейронної мережа;
2. Google Cloud Vision, Soundex – технологія, що використовується для

розпізнавання об'єктів. Можна використати API Google Cloud Vision, а для пошуку найкращої відповідності серед отриманих результатів ми можемо застосувати Soundex або обчислення відстані Левінштейн. Soundex – це алгоритм порівняння двох рядків за їх звучанням.

3.2. Реалізація алгоритму

Створена мною система розпізнавання об'єктів на зображеннях загалом являє собою певний сервіс і може легко працювати з сайтом, чат-ботом чи то мобільним додатком. Це дозволить такій системі функціонувати як повністю автоматизований процес без участі людини.

Визначити конкретну модель і марку об'єкта серед кількох візуально дуже схожих? Можливо. Кейс втілювався за допомогою нейронних мереж із використанням фреймворку Tensorflow.

Давайте пройдемося по кожному шару в мережі. Нейронні мережі в цілому складаються з колекції нейронів, які організовані в шари, кожен зі своїми вагами та зміщеннями, які можна дізнатись. Давайте розберемо нейрону мережу на основні будівельні блоки:

1) тензор можна сприймати як n-вимірну матрицю. У нейронній мережі вище тензори будуть тривимірними, за винятком вихідного рівня;
2) Нейрон можна розглядати як функцію, яка приймає кілька входів і дає один вихід. Виходи нейронів представлені вище як червоні → сині карти активації;
3) Шар - це просто сукупність нейронів з однаковою операцією, включаючи однакові гіперпараметри;
4) Ваги та упередження ядра, хоча і є унікальними для кожного нейрона, налаштовуються на етапі навчання та дозволяють класифікатору адаптуватися до задачі та набору даних. Вони кодуються у візуалізації жовтим → зеленим кольором, що розбігається. Конкретні значення можна переглянути в Інтерактивному поданні формули, клацнувши нейрон або наводячи курсор на ядро / ухил у Звітному еластичному поясненні;

5) Мережа передає диференційовану функцію оцінки, яка представляється як оцінка класу у візуалізації на вихідному рівні.

Вхідний рівень (крайній лівий шар) представляє вхідне зображення в нейронній мережі. Оскільки ми використовуємо зображення RGB як вхідні дані, вхідний шар має три канали, що відповідають червоному, зеленому та синьому каналам відповідно, які показані в цьому шарі. Використовуйте кольорову шкалу, коли ви клацаєте на піктограмі деталей мережі вище, щоб відобразити детальну інформацію (про цей шар та інші).

Згорткові шари є основою згорткових нейронних мереж, оскільки вони містять вивчені ядра (ваги), які виділяють особливості, що відрізняють різні зображення одне від одного - це те, що ми хочемо для класифікації! Під час взаємодії із згортковим шаром ви помітите зв’язки між попередніми та згортковими шарами. Кожне посилання представляє унікальне ядро, яке використовується для операції згортки для створення поточного виведення або карти активації згорткового нейрона.

Згорнутий нейрон виконує безелементний точковий продукт з унікальним ядром і виходом відповідного нейрона попереднього шару. Це дасть стільки проміжних результатів, скільки унікальних ядер. Згорнутий нейрон є результатом усіх проміжних результатів, підсумованих разом із вивченим упередженням[34].

Наприклад, давайте розглянемо перший згортковий шар у архітектурі Tiny VGG вище. Зверніть увагу, що в цьому шарі 10 нейронів, але в попередньому шарі лише 3 нейрони. В архітектурі Tiny VGG згорткові шари повністю пов'язані, тобто кожен нейрон пов'язаний з кожним іншим нейроном попереднього шару. Зосереджуючись на виході верхнього згорткового нейрона з першого згорткового шару, ми бачимо, що є 3 унікальних ядра, коли ми наводимо курсор на карту активації.

Розмір цих ядер є гіперпараметром, заданим розробниками мережевої архітектури. Для того, щоб отримати вихід згорткового нейрона (карта активації), ми повинні виконати елементний крапковий продукт з виходом попереднього шару та унікальним ядром, засвоєним мережею. У TinyVGG операція точкового продукту використовує крок 1, що означає, що ядро зміщується на 1 піксель на крапковий продукт, але це гіперпараметр, який дизайнер мережевої архітектури може налаштувати, щоб краще відповідати їх набору даних. Ми повинні зробити це для всіх

3 ядер, що дасть 3 проміжні результати.

Потім виконується поетапна сума, що містить усі 3 проміжні результати разом із зміщенням, яке вивчила мережа. Після цього отриманий двовимірний тензор буде картою активації, яку можна побачити на інтерфейсі вище для самого верхнього нейрона першого згорткового шару. Цю саму операцію потрібно застосувати для створення карти активації кожного нейрона.

За деякою простою математикою ми можемо зробити висновок, що є 3 x 10 = 30 унікальних ядер, кожна розміром 3x3, нанесених у перший згортковий шар. Зв'язок між згортковим шаром та попереднім шаром є дизайнерським рішенням при побудові мережевої архітектури, що вплине на кількість ядер на згортковий шар.

Заповнення часто необхідне, коли ядро виходить за межі карти активації. Заповнення зберігає дані на кордонах карт активації, що призводить до кращої продуктивності, і це може допомогти зберегти просторовий розмір вводу, що дозволяє дизайнеру архітектури будувати деппер, високопродуктивні мережі. Існує безліч методів заповнення, але найчастіше застосовується підхід із заповненням нулем завдяки його продуктивності, простоті та обчислювальній ефективності. Техніка передбачає додавання нулів симетрично по краях введення. Цей підхід застосовується багатьма високопродуктивними нейронними мережами, такими як AlexNet.

Розмір ядра, який також називають розміром фільтра, відноситься до розмірів розсувного вікна над входом. Вибір цього гіперпараметра має значний вплив на завдання класифікації зображень. Наприклад, невеликі розміри ядра здатні витягувати набагато більший обсяг інформації, що містить дуже локальні особливості, із вхідних даних. Як ви можете бачити на візуалізації вище, менший розмір ядра також призводить до меншого зменшення розмірів шару, що дозволяє більш глибоку архітектуру.

І навпаки, великий розмір ядра витягує менше інформації, що призводить до швидшого зменшення розмірів шару, що часто призводить до гіршої продуктивності. Великі ядра краще підходять для вилучення функцій, які є більшими. Зрештою, вибір відповідного розміру ядра буде залежати від вашого завдання та набору даних, але загалом менші розміри ядра призводять до кращої продуктивності завдання класифікації зображень, оскільки дизайнер архітектури може складати все більше і

більше шарів разом, щоб дізнайся все більше і більше складних функцій!

Stride вказує, скільки пікселів ядро повинно бути переміщено одночасно. Наприклад, як описано у наведеному вище прикладі згорткового шару, Tiny VGG використовує крок 1 для своїх згорткових шарів, що означає, що крапковий виріб виконується у вікні 3x3 вхідного сигналу для отримання вихідного значення, а потім переміщується до право на один піксель для кожної наступної операції. Вплив удару на згорткову нейронну мережу подібний до розміру ядра.У міру зменшення кроку вивчається більше можливостей, оскільки витягується більше даних, що також призводить до збільшення вихідних шарів. Навпаки, із збільшенням кроку це призводить до більш обмеженого вилучення особливостей та менших розмірів вихідного шару. Одним із обов'язків дизайнера архітектури є забезпечення того, щоб ядро симетрично ковзало по вхідних даних при реалізації CNN. Використовуйте візуалізацію гіперпараметру вище, щоб змінити крок для різних розмірів вводу / ядра, щоб зрозуміти це обмеження!

Нагадаємо, що згорткові нейронні мережі — це така класифікація штучних нейронних мереж прямого поширення, котрий з успіхом використовувся для аналізу візуальних зображень. Вони застосовують колекцію різних багатошарових перцептронів, котрі розроблені таким чином, щоб вимагати використання мінімального обсягу попередньої обробки. Вони відомі також як просторово інваріантні штучні нейронні мережі або інваріантні відносно зсуву. Виходячи з характеристик інваріантності відносно паралельного перенесення і їхньої архітектури спільних ваг.

Згорткові мережі взяли за основу біологічніий процес, зокрема схему з'єднання нейронів зорової кори живих створінь. Рецептивні поля, що розташовані у різних нейронах частково перекриваються таким способом, що вони покривають усе зорове поле. А окремі нейрони кори реагують на стимули лише в певній області зорового поля, іншими словами у рецептивному полі.

Модель машинного навчання перетворює свої вхідні дані у значущі результати - процес, який «вивчається» завдяки впливу відомих прикладів входів та результатів. Тому центральною проблемою машинного навчання та глибокого навчання є змістовна трансформація даних: іншими словами, вивчення корисних уявлень про

наявні вхідні дані, які наближають нас до очікуваного результату. Що таке подання? По суті, це інший спосіб розглядати дані - представляти чи кодувати дані. Наприклад, кольорове зображення може бути закодовано у форматі RGB (червоно-зелено-синій) або у форматі HSV (значення насиченості відтінку): це два різні подання одних і тих самих даних. Деякі завдання, які можуть бути складними для одного представництва, можуть стати легкими для іншого. Наприклад, завдання «вибрати всі червоні пікселі на зображенні» простіше у форматі RG, тоді як «зробити зображення менш насиченим» простіше у форматі HSV. Моделі машинного навчання - це пошук відповідних подань для своїх вхідних даних - перетворення даних, які роблять їх більш придатними до вирішення завдання, наприклад, завдання класифікації.

Глибоке навчання є специфічним підполем машинного навчання: новий погляд на уявлення про навчання з даних, що робить акцент на вивченні послідовних шарів дедалі значущих уявлень. Поглиблене глибоке навчання - це не посилання на будь-яке глибше розуміння, досягнуте підходом; скоріше, це означає цю ідею послідовних шарів репрезентацій. Скільки шарів сприяє моделі даних, називається глибиною моделі. Іншими відповідними іменами для цього поля могли б бути багаторівневі навчання репрезентацій та навчання ієрархічних репрезентацій. Сучасне глибоке навчання часто включає десятки, а то й сотні послідовних рівнів уявлень - і всі вони вивчаються автоматично завдяки впливу навчальних даних. Тим часом інші підходи до машинного навчання, як правило, фокусуються на вивченні лише одного або двох шарів подання даних; отже, їх іноді називають неглибоким навчанням.

У процесі глибокого навчання ці шаруваті уявлення (майже завжди) вивчаються за допомогою моделей, званих нейронними мережами, структурованих буквальними шарами, складеними один на одного. Термін нейронна мережа є посиланням на нейробіологію, але хоча деякі центральні концепції глибокого навчання частково були розроблені, черпаючи натхнення з нашого розуміння мозку, моделі глибокого навчання не є моделями мозку. Немає доказів того, що мозок реалізує щось подібне до механізмів навчання, що використовуються в сучасних моделях глибокого навчання. Ви можете натрапити на науково-популярні статті, які проголошують, що глибоке навчання працює як мозок або було створено за зразком мозку, але це не так. Було б заплутаним та контрпродуктивним для новачків у цій галузі думати про

глибоке навчання як про те, що якимось чином пов'язане з нейробіологією; вам не потрібна та саванна містики та таємничості "так само, як наші уми", і ви можете забути все, що прочитали про гіпотетичні зв'язки між глибоким навчанням та біологією. Для наших цілей глибоке навчання - це математична основа для вивчення подань на основі даних.

На даний момент ви знаєте, що машинне навчання - це відображення вхідних даних (наприклад, зображень) на цілі (наприклад, мітка «кішка»), що робиться шляхом спостереження за багатьма прикладами введення та цілей. Ви також знаєте, що глибокі нейронні мережі роблять це відображення вхідних даних до цілі за допомогою глибокої послідовності простих перетворень даних (шарів), і що ці перетворення даних засвоюються шляхом впливу на приклади. А тепер давайте подивимось, як це навчання відбувається конкретно. Специфікація того, що шар робить із вхідними даними, зберігається у вагах шару, які, по суті, являють собою купу чисел. Технічно кажучи, ми б сказали, що перетворення, реалізоване шаром, параметризується його вагами. (Ваги також іноді називають параметрами рівня.) У цьому контексті навчання означає пошук набору значень для ваг усіх шарів у мережі, таких, що мережа правильно відображатиме приклади входів до пов'язаних з ними цілей. Але ось у чому річ: глибока нейронна мережа може містити десятки мільйонів параметрів. Пошук правильного значення для всіх з них може здатися непростим завданням, особливо враховуючи, що зміна значення одного параметра вплине на поведінку всіх інших!

Спочатку вагам мережі присвоюються випадкові значення, тому мережа просто реалізує серію випадкових перетворень. Природно, що його випуск далекий від того, яким він мав би бути в ідеалі, і відповідно оцінка збитків дуже висока. Але з кожним прикладом мережевих процесів ваги трохи коригуються у правильному напрямку, і оцінка втрат зменшується. Це навчальний цикл, який, повторюваний достатню кількість разів (як правило, десятки ітерацій протягом тисяч прикладів), дає значення ваги, які мінімізують функцію втрат. Мережа з мінімальними втратами - це мережа, вихід якої максимально наближений до цілей: навчена мережа. Знову ж таки, це простий механізм, який після масштабування закінчується виглядати як магія.

Незважаючи на те, що за останні роки глибоке навчання призвело до неабияких

досягнень, очікування щодо того, чого зможе досягти галузь у наступне десятиліття, як правило, значно вищі, ніж те, що, ймовірно, буде можливим. Хоча деякі програми, що змінюють світ, такі як автономні машини, вже доступні, набагато більше, ймовірно, залишаться незрозумілими протягом тривалого часу, такі як правдоподібні системи діалогу, машинний переклад на рівні людини на довільних мовах та розуміння природної мови на рівні людини. Зокрема, розмови про загальний інтелект на рівні людини не слід сприймати занадто серйозно. Ризик із високими очікуваннями на короткий термін полягає в тому, що, оскільки технологія не може забезпечити, інвестиції в дослідження будуть висихати, сповільнюючи прогрес на довгий час.

Це траплялося і раніше. У минулому двічі штучний інтелект пройшов цикл напруженого оптимізму, за яким слідували розчарування та скептицизм, в результаті чого бракувало фінансування. Це почалося із символічного штучний інтелект в 1960-х. У ті перші дні прогнози щодо штучного інтелекту летіли високо. Одним з найвідоміших піонерів та прихильників символічного підходу до штучного інтелекту був Марвін Мінський, який у 1967 р. Заявив: "Протягом одного покоління ... проблема створення" штучного інтелекту "буде істотно вирішена". Через три роки, в 1970 році, він зробив більш точний кількісний прогноз: "Через три-вісім років ми матимемо машину із загальним інтелектом середньої людини".

У 2016 році таке досягнення все ще здається далеким у майбутньому - настільки, що ми не маємо можливості передбачити, скільки часу це займе, - але в 1960-х і на початку 1970-х років кілька експертів вважали, що це вже не за горами (як це роблять сьогодні багато людей). Через кілька років, оскільки ці великі сподівання не змогли здійснитися, дослідники та державні кошти відвернулись від поля, ознаменувавши початок першої зими штучного інтелекту (згадка про ядерну зиму, оскільки це було незабаром після розпалу холодної війни).

Це було б не останнє. У 1980-х роках новий погляд на символічний ШІ, експертні системи, почав набирати популярність серед великих компаній. Кілька початкових історій успіху спричинили хвилю інвестицій: корпорації у всьому світі заснували власні внутрішні відділи ШІ для розробки експертних систем. Близько 1985 року компанії витрачали на технології понад 1 мільярд доларів щороку; але на початку 1990-х років ці системи виявилися дорогими в обслуговуванні, важкими в масштабі та

обмеженими в масштабах, і інтерес згас. Так розпочалася друга II зима.

Можливо, зараз ми спостерігаємо третій цикл ажіотажу на II та розчарування - і ми все ще перебуваємо у фазі напруженого оптимізму. Найкраще пом'якшити наші короткострокові очікування та переконатись, що люди, менш обізнані з технічною стороною галузі, чітко уявляють, що глибоке навчання може, а що не може дати.

Хоча ми можемо мати нереальні короткострокові очікування щодо штучного інтелекту, довгострокова картина виглядає яскравою. Ми лише починаємо застосовувати глибоке навчання до багатьох важливих проблем, для яких воно може стати трансформативним, від медичних діагнозів до цифрових асистентів. Протягом останніх п’яти років дослідження штучного інтелекту просуваються надзвичайно швидко, значною мірою завдяки рівню фінансування, якого ще ніколи не було за коротку історію штучного інтелекту, але до цього часу відносно невеликий із цього прогресу пробився до продуктів та процесів які формують наш світ. Більшість результатів досліджень глибокого навчання ще не застосовуються або, принаймні, не застосовуються до всього спектру проблем, які вони можуть вирішити в усіх галузях. Ваш лікар ще не використовує штучний інтелект, як і ваш бухгалтер.

Звичайно, ви можете задати своєму смартфону прості запитання та отримати розумні відповіді, ви можете отримати досить корисні рекомендації щодо продуктів на Amazon.com, а ви можете шукати “день народження” в Google Фото і миттєво знаходити ті фотографії з дня народження вашої дочки з минулого місяць. Це далеко не так, як колись стояли такі технології. Але такі інструменти все ще є лише аксесуарами для нашого повсякденного життя. Штучний інтелект ще не перейшов до того, щоб стати головним у способі роботи, мислення та життя.

Зараз може здатися важким повірити, що штучний інтелект може мати великий вплив на наш світ, оскільки він ще не широко розгорнутий - так само, як ще в 1995 році було б важко повірити у майбутній вплив Інтернету. Тоді більшість людей не розуміли, як Інтернет є для них актуальним і як це змінить їхнє життя. Те саме стосується глибокого навчання та штучний інтелект сьогодні. Але не помиляйся: штучний інтелект приходить. У недалекому майбутньому штучний інтелект стане вашим помічником, навіть вашим другом; він відповість на ваші запитання, допоможе навчати своїх дітей та стежить за вашим здоров’ям. Він доставить ваші продовольчі

товари до ваших дверей і підведе вас від пункту А до пункту Б. Це буде вашим інтерфейсом у все більш складний та інформаційно інтенсивний світ. І, що ще більш важливо, штучний інтелект допоможе людству в цілому рухатися вперед, допомагаючи вченим-людям у нових проривних відкриттях у всіх наукових галузях, від геномики до математики.

По дорозі, ми можемо зіткнутися з кількома невдачами і, можливо, новою зимою штучний інтелект - приблизно так само, як Інтернет-індустрія була перебільшена в 1998–1999 рр. І постраждала від аварії, яка висушила інвестиції на початку 2000-х. Але ми врешті дійдемо туди. В кінцевому підсумку штучний інтелект застосовуватиметься майже до кожного процесу, що складає наше суспільство та наше повсякденне життя, подібно до того, як сьогодні є Інтернет. Не вірте короткочасному ажіотажу, але вірте в довгострокове бачення. Може знадобитися деякий час, поки штучний інтелект буде розгорнуто до його справжнього потенціалу - потенціалу, про масштаби якого ще ніхто не наважився мріяти, - але штучний інтелект приходить, і він фантастично перетворить наш світ.

Поглиблене навчання досягло рівня уваги громадськості та галузевих інвестицій, яких раніше не було в історії штучного інтелекту, але це не перша успішна форма машинного навчання. Можна з упевненістю сказати, що більшість алгоритмів машинного навчання, що використовуються сьогодні в галузі, не є алгоритмами глибокого навчання. Поглиблене навчання не завжди є правильним інструментом для роботи - іноді недостатньо даних для глибокого навчання, а іноді проблему краще вирішити за допомогою іншого алгоритму. Якщо глибоке навчання - це ваш перший контакт з машинним навчанням, то ви можете опинитися в ситуації, коли все, що у вас є, - це молоток глибокого навчання, і кожна проблема машинного навчання починає виглядати як цвях. Єдиний спосіб не потрапити в цю пастку - це ознайомитись з іншими підходами та практикувати їх, коли це доречно. Детальне обговорення класичних підходів до машинного навчання виходить за рамки цієї книги, але ми коротко їх розглянемо та опишемо історичний контекст, у якому вони були розроблені. Це дозволить нам помістити глибоке навчання в ширший контекст машинного навчання та краще зрозуміти, звідки походить глибоке навчання та чому це важливо.

Імовірнісне моделювання - це застосування принципів статистики для аналізу даних. Це була одна з найперших форм машинного навчання, і вона все ще широко використовується донині. Одним з найвідоміших алгоритмів у цій категорії є алгоритм Наїв Байєса. Класифікація Байєса - це тип класифікатора машинного навчання, заснований на застосуванні теореми Байєса, при цьому припускаючи, що всі ознаки у вхідних даних є незалежними (сильне, або «наївне» припущення, звідки походить назва). Ця форма аналізу даних передує комп'ютерам і застосовувалася вручну за десятки років до її першої комп'ютерної реалізації (швидше за все, датується 1950-ми роками). Теорема Байєса та основи статистики датуються вісімнадцятим століттям, і це все, що вам потрібно, щоб почати використовувати наївні класифікатори Байєса. Тісно пов'язаною моделлю є логістична регресія, яку іноді вважають «привітним світом» сучасного машинного навчання. Нехай вас не вводить в оману його назва - logreg - це алгоритм класифікації, а не алгоритм регресії. Подібно до Naive Bayes, логрег давно випереджає обчислення, але все ще корисний донині завдяки своїй простій та універсальній природі. Це часто перше, що дослідник даних спробує використати набір даних, щоб отримати уявлення про завдання класифікації.

Приблизно в 2010 році, хоча наукові спільноти майже повністю уникали нейронних мереж, певна кількість людей, які все ще працюють над нейронними мережами, почали робити значні прориви: групи Джеффрі Хінтона з Університету Торонто, Йошуа Бенджо з Університету Монреаля, Янн ЛеКун з Нью-Йоркського університету та IDSIA у Швейцарії. У 2011 році Ден Сіресан з IDSIA почав перемагати в академічних змаганнях з класифікації зображень із глибокими нейронними мережами, що навчаються GPU - перший практичний успіх сучасного глибокого навчання. Але переломний момент настав у 2012 році, коли група Хінтона вступила у щорічну масштабну проблему класифікації зображень ImageNet. На той час виклик ImageNet був відомим складним завданням, який полягав у класифікації кольорових зображень з високою роздільною здатністю на 1000 різних категорій після навчання на 1,4 мільйона зображень. У 2011 році п’ятірка точності моделі-переможця, заснованої на класичних підходах до комп’ютерного зору, становила лише 74,3%. Потім, у 2012 році, команда під керівництвом Алекса Крижевського та за порадою Джеффрі Хінтона змогла досягти точності п’ятірки у 83,6% - значного прориву. З тих

пір у конкуренції домінують глибокі згорткові нейронні мережі. До 2015 року переможець досяг точності 96,4%, а завдання класифікації на ImageNet було визнано повністю вирішеною проблемою. Починаючи з 2012 року, глибокі згорткові нейронні мережі (convnets) стали алгоритмом переходу до всіх завдань комп'ютерного зору; загальніше, вони працюють над усіма перцептивними завданнями. На великих конференціях з комп'ютерного зору в 2015 та 2016 роках було майже неможливо знайти презентації, які б не залучали коннетів у тій чи іншій формі. У той же час глибоке навчання також знайшло застосування в багатьох інших типах проблем, таких як обробка природною мовою. Він повністю замінив SVM та дерева рішень у широкому діапазоні застосувань. Наприклад, протягом кількох років Європейська організація з ядерних досліджень (CERN) використовувала методи, що базуються на дереві рішень, для аналізу даних про частинки з детектора ATLAS на Великому адронному колайдері (LHC); але CERN врешті-решт перейшов на глибокі нейронні мережі на основі Keras через їхню більш високу продуктивність та простоту навчання на великих наборах даних.

Основною причиною глибокого навчання стало так швидко, що це забезпечило кращу ефективність у багатьох проблемах. Але це не єдина причина. Глибоке навчання також значно полегшує вирішення проблем, оскільки воно повністю автоматизує те, що раніше було найважливішим кроком у процесі машинного навчання: інженерія функцій. Попередні методи машинного навчання - неглибоке навчання - передбачали лише перетворення вхідних даних в один або два послідовних простори представлення, як правило, за допомогою простих перетворень, таких як високовимірні нелінійні проекції (SVM) або дерева рішень. Але вдосконалених уявлень, що вимагаються складними проблемами, як правило, неможливо досягти такими методами. Таким чином, людям довелося докласти максимум зусиль, щоб зробити вихідні вхідні дані більш підданими обробці цими методами: їм довелося вручну розробляти хороші рівні подань для своїх даних. Це називається інженерною функцією. Глибоке навчання, навпаки, повністю автоматизує цей крок: при глибокому навчанні ви вивчаєте всі функції за один прохід, а не потребуєте їх самостійного проектування. Це значно спростило робочі процеси машинного навчання, часто замінюючи складні багатоступеневі конвеєри єдиною простою наскрізною моделлю

глибокого навчання. Ви можете запитати, якщо суть проблеми полягає в тому, щоб мати кілька послідовних рівнів уявлень, чи можна неодноразово застосовувати неглибокі методи для наслідування ефектів глибокого навчання? На практиці скорочується повернення до послідовних застосувань методів неглибокого навчання, оскільки оптимальний перший рівень представлення в тришаровій моделі не є оптимальним першим шаром в одношаровій або двошаровій моделі. Що трансформує глибоке навчання, це те, що воно дозволяє моделі вивчати всі рівні представництва спільно, одночасно, а не послідовно (жадібно, як це називається). При спільному вивченні особливостей, коли б модель коригувала одну зі своїх внутрішніх особливостей, усі інші функції, які залежать від неї, автоматично пристосовуються до змін, не вимагаючи втручання людини. Все контролюється одним сигналом зворотного зв'язку: кожна зміна моделі служить кінцевій меті. Це набагато потужніше, ніж жадібне укладання неглибоких моделей, оскільки дозволяє вивчати складні абстрактні уявлення, розбиваючи їх на довгі ряди проміжних просторів (шарів); кожен простір є лише простим перетворенням від попереднього. Це дві основні характеристики того, як глибоке навчання вчиться на даних: поступовий, пошаровий спосіб, яким розвиваються дедалі складніші уявлення, і той факт, що ці проміжні додаткові уявлення вивчаються спільно, кожен рівень оновлюється, щоб відповідати обом представницькі потреби шару вгорі та потреби шару внизу. Разом ці дві властивості зробили глибоке навчання набагато успішнішим, ніж попередні підходи до машинного навчання.

Чудовим способом зрозуміти сучасний ландшафт алгоритмів та інструментів машинного навчання є перегляд змагань з машинного навчання на Kaggle. Завдяки високому конкурентному середовищу (деякі конкурси мають тисячі учасників та мільйондолярні призи) та широкому спектру проблем машинного навчання, Kaggle пропонує реалістичний спосіб оцінити, що працює, а що ні. Отже, який алгоритм надійно виграє змагання? Які інструменти використовують найкращі учасники? У 2016 та 2017 роках у Kaggle переважали два підходи: машини для підвищення градієнта та глибоке навчання. Зокрема, посилення градієнта використовується для проблем, де є структуровані дані, тоді як глибоке навчання використовується для перцептивних проблем, таких як класифікація зображень. Практики першого майже

завжди використовують чудову бібліотеку XGBoost, яка пропонує підтримку двох найпопулярніших мов науки про дані: Python та R. Тим часом, більшість учасників Kaggle, які використовують глибоке навчання, використовують бібліотеку Keras, завдяки простоті використання, гнучкість та підтримка Python. Ось дві техніки, з якими ви сьогодні повинні бути найбільш знайомими, щоб досягти успіху в прикладному машинному навчанні сьогодні: машини для підвищення градієнта, для проблем неглибокого навчання; та глибоке навчання для проблем сприйняття. У технічному плані це означає, що вам потрібно буде знати XGBoost та Keras - дві бібліотеки, які в даний час домінують у змаганнях Kaggle. З цією книгою в руках ви вже на один великий крок ближче.

У період з 1990 по 2010 рр. Готові процесори стали швидшими приблизно в 5000 разів. Як результат, сьогодні можна запускати невеликі моделі глибокого навчання на своєму ноутбуці, тоді як це було б важко вирішити 25 років тому. Але типові моделі глибокого навчання, що використовуються в комп'ютерному зорі або розпізнаванні мови, вимагають на порядок більшої обчислювальної потужності, ніж те, що може забезпечити ваш ноутбук. Протягом 2000-х років такі компанії, як NVIDIA та AMD, інвестували мільярди доларів у розробку швидких, масово паралельних мікросхем (графічних процесорів) для живлення графіки все більш фотореалістичних відеоігор - дешевих одноцільових суперкомп'ютерів, призначених для створення складних 3D-сцени на екрані в режимі реального часу. Ця інвестиція принесла користь науковому співтовариству, коли в 2007 році NVIDIA запустила CUDA (https://developer.nvidia.com/about-cuda), інтерфейс програмування для своєї лінійки графічних процесорів. Невелика кількість графічних процесорів почала замінювати масивні кластери центральних процесорів у різних сильно паралелізованих додатках, починаючи з фізичного моделювання. Глибокі нейронні мережі, що складаються здебільшого з безлічі малих матричних множень, також мають високий паралелізм; і приблизно в 2011 р. деякі дослідники почали писати реалізації нейронних мереж CUDA - Ден Сіресан та Алекс Крижевський були одними з перших.

3.3 Тренування, тестування і огляд результатів

Нейронна мережа моделює роботу нейронів людського мозку за допомогою штучних нейронів, що самонавчається, вирішує певне завдання з урахуванням попереднього досвіду. І з кожним разом робить усе менше помилок.

На сьогодні момент машинне навчання охоплює досить широкий спектр додатків від ресторанів, банків, заправок до роботів на виробництві. Нові завданн призводять до появи нових напрямків машинного навчання, що виникають практично щодня.

Перейдемо до розпізнавання марок автомобілів. Процес підготовки включає в себе тренування нейронної мережі.

Тренування нейронної мережі зокрема означає вибирання однієї моделі з множини дозволених моделей. В баєсовій системі, це є визначення розподілу над множиною дозволених моделей, що зводить витрати до мінімуму. Ми підготували багато різних зображень автомобілів (рис.3.3.1) для тренування згорткової нейронної мережі.

Рисунок 3.3.1 – Набір зображень автомобілів
для тренування нейронної мережі

На першому етапі попередньої обробки даних зображення стискаються у файли hdf5 (один для навчання та інший для тестування). Потім може прочитати працювати нейронна мережа.

У цьому сценарії ми хочемо переконатися, що можемо використовувати механізм виведення моделей, який підтримує обслуговування запитів від клієнта REST API. Для цього ми використовуємо власну модель, засновану на раніше

побудованій моделі Keras, щоб прийняти об'єкт JSONDataframe, який має кодоване зображення Base64 всередині. Лістинг коду зображено в додатку А.

Для будь-якої нестандартної операції, яка має треновані ваги, Keras дозволяє нам реалізувати власний шар. Маючи справу з величезною кількістю даних зображень, можна зіткнутися з проблемами пам'яті. Спочатку зображення RGB містять цілі дані (0-255). Під час запуску градієнтного спуску в рамках оптимізації під час зворотного розповсюдження виявиться, що цілочисельні градієнти не дають достатньої точності для правильного регулювання ваги мережі. Тому необхідно змінити на плаваючу точність.

Тут можуть виникати проблеми. Навіть коли зображення зменшено до 224*224*3, коли ми використовуємо десять тисяч навчальних зображень, ми розглядаємо понад 1 мільярд записів із плаваючою комою. На відміну від перетворення цілого набору даних на плаваючу точність, кращою практикою є використання «масштабного шару», який масштабує вхідні дані по одному зображенню за раз і лише тоді, коли це потрібно. Це слід застосовувати після нормалізації партії в моделі. Параметри цього масштабного шару - це також параметри, які можна дізнатись під час навчання.

Щоб використовувати цей спеціальний шар також під час підрахунку балів, ми повинні упакувати клас разом із нашою моделлю. За допомогою MLflow ми можемо досягти цього за допомогою словника Keras custom_objects, що відображає імена (рядки) у власні класи або функції, пов'язані з моделлю Keras. MLflow зберігає ці власні шари за допомогою CloudPickle і відновлює їх автоматично, коли модель завантажена в mlflow.keras.load_model() і mlflow.pyfunc.load_model().

Щоб детільніше розглянути суть оптимізації, можемо зобразити то на картинках. Щоб досягти кращого підбору гіперпараметрів необхідно порівняти два можливих пошуки (по гратці і випадковий пошук).

У методі випадкового пошуку ми створюємо сітку можливих значень для гіперпараметрів. Кожна ітерація пробує випадкову комбінацію гіперпараметрів із цієї сітки, реєструє продуктивність і, нарешті, повертає комбінацію гіперпараметрів, яка забезпечила найкращі показники.У методі пошуку сітки ми створюємо сітку можливих значень для гіперпараметрів. Кожна ітерація пробує комбінацію

гіперпараметрів у певному порядку. Він підходить для моделі на кожній можливій комбінації гіперпараметра та реєструє ефективність моделі. Нарешті, він повертає найкращу модель з найкращими гіперпараметрами. (рис. 3.3.2).

Налаштування та пошук правильних гіперпараметрів для вашої моделі - це проблема оптимізації. Ми хочемо мінімізувати функцію втрат нашої моделі, змінивши параметри моделі. Байєсова оптимізація допомагає нам знайти мінімальну точку за мінімальну кількість кроків. Байєсова оптимізація також використовує функцію збору даних, яка спрямовує вибірку на ділянки, де, ймовірно, покращення порівняно з найкращими спостереженнями на даний момент.

Ідея деревної оптимізації Парзена схожа на байєсівську. Замість того, щоб знаходити значення p (y | x), де y - це функція, яку слід мінімізувати (наприклад, втрата під час перевірки), а x - значення гіперпараметра, моделі TPE P (x | y) та P (y). Одним з найбільших недоліків деревоподібних оцінювачів Парзена є те, що вони не моделюють взаємодії між гіперпараметрами. Тим не менш, TPE працює надзвичайно добре на практиці та пройшов бойові випробування в більшості доменів.

Одними з найкращих бібліотек оптимізації гіперпараметрів є:

1) Scikit-learn (grid search, random search);
2) Hyperopt;
3) Scikit-Optimize;
4) Optuna;
5) Ray.tune.

Scikit-learn має реалізації для пошуку в сітці та випадкового пошуку, і це гарне місце для початку, якщо ви будуєте моделі за допомогою sklearn.

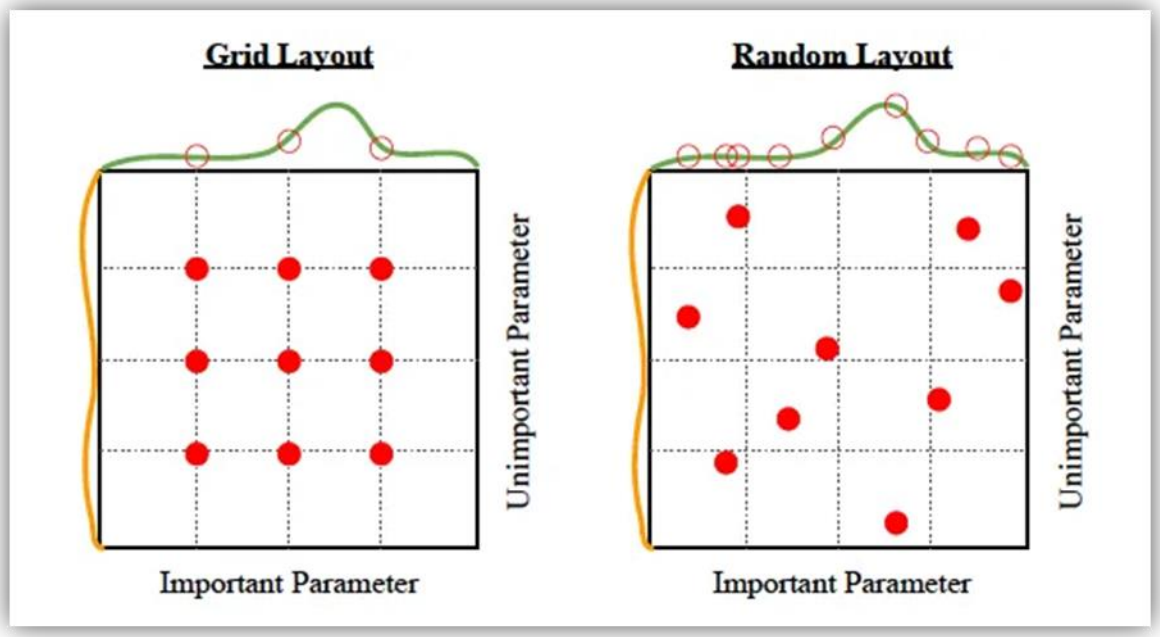


Рисунок 3.3.2 – Порівняння пошуку по гратці і випадкового пошуку

Для обох цих методів scikit-learn навчає та оцінює модель у k-кратному перехресному підтвердженні за різними варіантами параметрів та повертає найкращу модель. Зокрема:

1) випадковий пошук: за допомогою randomsearchcv виконує пошук за деякою кількістю випадкових комбінацій параметрів;
2) пошук погратці: gridsearchcv виконує пошук за всіма наборами параметрів у сітці.

Налаштування моделей за допомогою scikit-learn є хорошим початком, але є кращі варіанти, і вони часто мають стратегію випадкового пошуку.

Hyperopt - один із найпопулярніших доступних пакетів налаштування гіперпараметрів. Hyperopt дозволяє користувачеві описати пошуковий простір, в якому користувач очікує найкращих результатів, що дозволяє алгоритмам у hyperopt здійснювати більш ефективний пошук.

В даний час в алгоритмі hyperopt реалізовано три алгоритми:

1) Random Search;
2) Tree of Parzen Estimators (TPE);
3) Adaptive TPE.

Щоб використовувати гіперопт, спочатку слід описати:

4) цільову функцію щоб інмізувати;
5) простір, над яким можна шукати;
6) база даних, в якій зберігатимуться всі точкові оцінки пошуку;
7) алгоритм пошуку, який слід використовувати.

Scikit-optimize використовує алгоритм оптимізації на основі послідовної моделі для пошуку оптимальних рішень для проблем пошуку гіперпараметрів за менший час. Scikit-optimize забезпечує безліч функцій, окрім оптимізації гіперпараметрів, таких як:

1) зберігати та завантажувати результати оптимізації;
2) графіки конвергенції;
3) порівняння сурогатних моделей.

Optuna використовує історичний запис деталей маршрутів, щоб визначити перспективну область для пошуку оптимізації гіперпараметра, а отже, знаходить оптимальний гіперпараметр за мінімальний проміжок часу.

Він має функцію обрізки, яка автоматично зупиняє неперспективні стежки на ранніх етапах навчання. Деякі ключові функції, що надаються Optuna:

1) Lightweight, versatile, and platform-agnostic architecture;
2) Pythonic search spaces;
3) Efficient optimization algorithms;
4) Easy parallelization;
5) Quick visualization.

Tune - популярний вибір експериментів та налаштування гіперпараметрів у будь-якому масштабі. Ray використовує потужність розподілених обчислень для пришвидшення оптимізації гіперпараметрів і має реалізацію для декількох сучасних алгоритмів оптимізації в масштабі.

Деякі основні функції, які надає Ray Tune:

1) розподілена асинхронна оптимізація нестандартно, використовуючи Ray;
2) легко масштабується;
3) надані алгоритми SOTA, такі як ASHA, BOHB та навчання на основі

народонаселення;

4) підтримує Tensorboard і MLflow;
5) підтримує різноманітні фреймворки, такі як sklearn, xgboost, Tensorflow, pytorch тощо.

Keras Tuner - це бібліотека, яка допомагає вибрати оптимальний набір гіперпараметрів для вашої програми TensorFlow. Створюючи модель для налаштування гіперпараметрів, ви також визначаєте простір пошуку гіперпараметрів на додаток до архітектури моделі. Модель, яку ви налаштували для налаштування гіперпараметрів, називається гіпермоделью.

Можна визначити гіпермодель за допомогою двох підходів:

1) за допомогою функції конструктора моделей;
2) підкласуючи клас HyperModel API Keras Tuner.

Також можна використовувати два заздалегідь визначені класи HyperModel - HyperXception та HyperResNet для програм комп'ютерного зору.

Ми використовуємо набір даних з автомобілями, який містить 16 185 зображень 196 класів автомобілів. Дані розділені на 8144 навчальних зображень та 8041 тестових зображень, де кожен клас був розділений приблизно на 50-50 поділів (рис 3.3.3).

Отримали ми список зображень з простори Інтернету.Готувати вручну 8 тис. зображень – робота не з легких, тому ми вирішили знайти готові датасети (набір зображень) і використати їх.

Команди, за допомогою яких було отримано колекцію картинок з інтернету зображено в додатку Б. Колекція картинок вийшла досить великою, мені її вистачило для того, щоб добре натренувати нейронну мережу.

Рисунок 3.3.3 – Приклади зображень авто, які були підготовленні

Запускаємо наш алгоритм за допомогою команди:

1) python pre_process.py

Код даної команди можна переглянути в додатку А. Тепер час запускати тренування:

2) python train.py

Код алгоритму тренування знаходиться в додатку В. Якщо ми хочемо візуалізувати тренування, необхідно запустити у своєму терміналі:

3) tensorboard --logdir path_to_current_dir/logs

Щоб отримати інформацію проте, як відбувається навчання, було вирішено зробити прості діаграми і слідкувати за показниками тренування згорткової нейронної мережі.

В результаті бачимо візуалізацію виконання процесу тренування нашої згорткової нейронної мережі (рис.3.3.4).

Згорткові нейронні мережі на сьогодні – "робоча конячка" в області нейронних мереж. Згорткові нейронні мережі використовуються переважно для вирішення завдань комп'ютерного зору, хоча може застосовуватися також для роботи з аудіо і будь-якими даними, які можна представити у вигляді матриць. В нашому випадку ми використовуємо їх для розпізнавання елементів на зображенях.

Кількість та різноманітність зібраних даних має великий вплив на результати, яких можна досягти за допомогою моделей глибокого навчання.

Збільшення даних - це стратегія, яка може значно покращити результати навчання без необхідності фактичного збору нових даних.

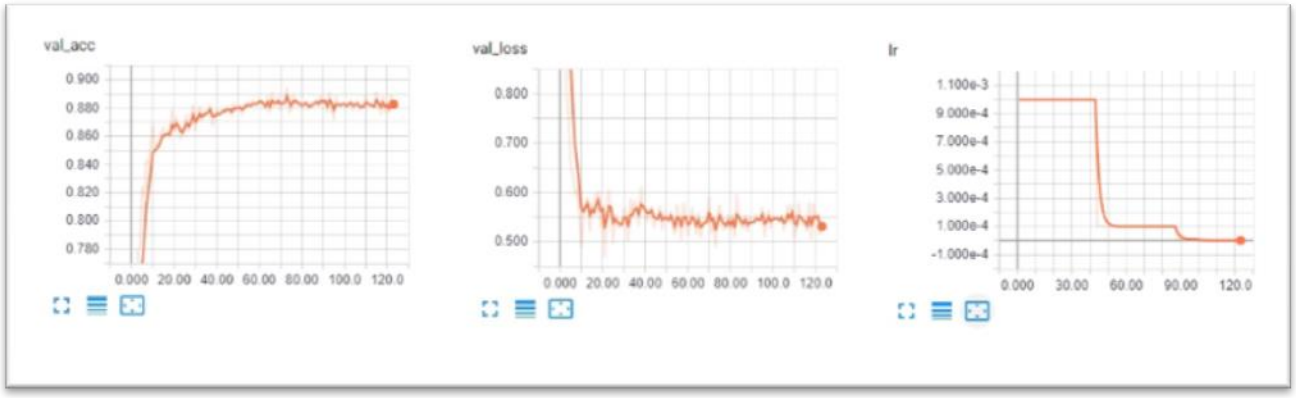


Рисунок 3.3.4 – Візуалізація тренування

Як ми говорили вище, існує дуже великий вибір різних фреймворків і бібліотек для розпізнавання і оптимізації, а писати з нуля такі бібліотеки варті іншої дипломної роботи, тому ми використали деякі бібліотеки, що допомогли реалізувати наш алгоритм:

1) NumPy;
2) Tensorflow;
3) Keras;
4) OpenCV.

Після тренуавння ми можемо побачити результат виконавши команду:

4) python demo.py --i [image_path]

Для наглядності передамо таке зображення автомобіля (рис.3.3.5).

Застосування доповнення до великого корпусу навчальних даних може бути дуже дорогим, особливо при порівнянні результатів різних підходів. За допомогою Koalas стає легко спробувати існуючі фреймворки для збільшення зображень у Python та масштабування процесу на кластері з кількома вузлами за допомогою знайомого Pandas API для науки про дані.

За допомогою різних методів, таких як обрізання, доповнення та горизонтальне перевертання, які зазвичай використовуються для тренування

великих нейронних мереж, набори даних можна штучно роздути, збільшивши кількість зображень для навчання та тестування.

Рисунок 3.3.5 – Зображення автомобіля

Ось наші запити і результат (код до demo.py знаходиться в додатку Г):

5) python demo.py

class_name: Lamborghini Reventon Coupe 2008

prob: 0.9999994

Так можна розпізнати будь який інший автомобіль (рис 3.3.6).

Рисунок 3.3.6 – Інші автомобілі розпізнанні нейронної мережею

Нагадаємо, що згортка — матриця, в більшості випадків малих розмірів, що застосовуєтсбся у обробці зображень як фільтр для підвищення різкості, розмиття чи виділення границь тощо. Обробка зображення базуєьбся на обчисленні нового значення обраного пікселя з урахуванням значення оточуючих його пікселів.

Згортка може викликати різні ефекти, залежності від елементів матриці:

- тотожне відображення (тотожна функція) — таке відображення, яке

переводить кожний елемент множини (області) визначення в себе;

- Sharpen;
- розмиття квадратом;
- розмивання Гауса — це метод фільтрації зображення за допомогою функції Гауса, котрий призводить до ефекту розмиття зображення. Такий ефект широко застосовується в графічних програмах, зокрема, для зниження деталізації чи то зменшення зашумленості зображенні. Візуальний ефект цієї фільтрації розмивання аналогічний погляду на зображення крізь напівпрозорий екран, суттєво відрізняючись від ефекту боке, котрий можна отримати шляхом несфокусованого об'єктива або тіні об'єкта при звичайному освітленні;
- 5×5 нерізке розмиття.

Згортка - це процес додавання кожного елемента зображення до його сусідів, зважених ядром. Хоча й виконувана матрична операція позначається зірочкою, варто зауважити, що це не звичайне множення.

Наприклад, якщо ми маємо дві 3x3 матриці, перша - ядро, друга - шматок зображення, згортка - це процес транспонування рядків і стовпчиків ядра з наступним множенням і додаванням. Елемент з координатами [2, 2] отриманого зображення буде зваженою комбінацією всіх елементів матриці зображення, з вагами взятими з ядра:

$$\left(\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} * \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}\right)[2,2] =$$

$$= (i * 1) + (h * 2) + (g * 3) + (f * 4) + (e * 5) + (d * 6) + (c * 7) + (b * 8) + (a * 9) \quad (3.3.1)$$

Так як ступінь перенавчання моделі визначається і її потужністю, і кількістю отримуваного нею тренування, забезпечення згорткової мережі більшою кількістю тренувальних даних може знижувати перенавчання. Таким чином, якщо ці мережі зазвичай тренують усіма існуючими даними, одним із підходів є, якщо це можливо, - це породжувати нові дані з нуля, або ж безпосередньо збурювати наявні

дані для створення нових. Для прикладу, вхідні зображення може бути асиметрично обрізувано на декілька відсотків для знаходження нових прикладів з таким же маркером, як і первинний [33].

Що сталося, так це те, що ігровий ринок субсидував суперкомп'ютери для наступного покоління програм штучного інтелекту. Іноді великі речі починаються як ігри. Сьогодні ігровий графічний процесор NVIDIA TITAN X, вартість якого на кінець 2015 року коштувала 1000 доларів, може забезпечити максимум 6,6 TFLOPS за одну точність: 6,6 трлн операцій float32 в секунду. Це приблизно в 350 разів більше, ніж можна отримати від сучасного ноутбука. На TITAN X потрібно лише кілька днів, щоб навчити модель ImageNet такого роду, яка виграла б конкурс ILSVRC кілька років тому. Тим часом великі компанії навчають моделі глибокого навчання на кластерах сотень графічних процесорів типу, розроблених спеціально для потреб глибокого навчання, таких як NVIDIA Tesla K80. Сама обчислювальна потужність таких кластерів - це те, що ніколи не було б можливим без сучасних графічних процесорів. Більше того, галузь глибокого навчання починає виходити за межі графічних процесорів і інвестує у все більш спеціалізовані та ефективні чіпи для глибокого навчання. У 2016 році на щорічній конвенції вводу-виводу Google розкрив проект свого блоку обробки тензорів (TPU): нова конструкція мікросхеми, розроблена з нуля для роботи глибоких нейронних мереж, яка, як повідомляється, в 10 разів швидша і набагато енергоефективніша, ніж верхня - лінійних графічних процесорів.

Оскільки глибоке навчання стало новим рівнем техніки для комп’ютерного зору в 2012–2013 роках, а зрештою і для всіх сприймаючих завдань, керівники галузі взяли це до відома. Далі послідувала поступова хвиля галузевих інвестицій, що значно перевищувала все, що раніше спостерігалося в історії ШІ. У 2011 році, безпосередньо перед тим, як поглиблене навчання потрапило в центр уваги, загальний обсяг інвестицій венчурного капіталу в ШІ становив близько 19 мільйонів доларів США, які майже повністю спрямовувались на практичне застосування неглибоких підходів до машинного навчання. До 2014 року вона зросла до приголомшливих 394 мільйонів доларів. За ці три роки десятки стартапів,

які намагалися скористатися ажіотажем із глибоким навчанням. Тим часом великі технологічні компанії, такі як Google, Facebook, Baidu та Microsoft, інвестували у відділи внутрішніх досліджень у розмірах, які, швидше за все, зменшать потік грошей венчурного капіталу. Виявилося лише декілька цифр: у 2013 році Google придбав глибокий стартап DeepMind за 500 мільйонів доларів, що було найбільшим придбанням компанії зі штучним інтелектом в історії. У 2014 році Байду відкрив у Силіконовій долині науково-дослідний центр, який інвестував у проект 300 мільйонів доларів. Поглиблений апаратний стартап Nervana Systems був придбаний компанією Intel у 2016 році за понад 400 мільйонів доларів. Машинне навчання - зокрема, глибоке навчання - стало центральним у продуктовій стратегії цих технічних гігантів. Наприкінці 2015 року генеральний директор Google Сундар Пічай заявив: «Машинне навчання - це основний, перетворюючий спосіб, за допомогою якого ми переосмислюємо, як ми все робимо. Ми вдумливо застосовуємо його у всіх наших продуктах, будь то пошук, реклама, YouTube або Play. І ми вже в перші дні, але ви побачите, як ми систематично застосовуємо машинне навчання у всіх цих сферах.

В роботі вирішено завдання створення математичного забезпечення для побудови моделей кількісних залежностей на основі згорткових нейронних мереж. Запропоновано архітектуру згорткової нейронної мережі, що може використовуватися для даних, в яких вхідні значення не пов’язані між собою. В запропонованій архітектурі в якості першого шару використовується повнозв’язний шар. Завдяки цьому в процесі навчання нейронної мережі між вихідними значеннями нейронів першого шару можуть з’явитись зв’язки, що необхідні для роботи наступних згорткових шарів. Як і в звичайних згорткових нейромережах, згорткові шари можуть чергуватися із шарами підвибірки, але при цьому використовується одновимірна згортка. Після згорткових шарів використовуються повнозв’язні. В якості функції активації останнього шару використовується функція softmax, що дозволяє визначати ймовірності належності розпізнаваного екземпляра до кожного з класів. Порівняно результати тестування всіх побудованих моделей. За результатами тестування визначено, що якість

запропонованої моделі вища, але на її навчання поребується більше часу.

Навчання - це подорож на все життя, особливо у галузі штучного інтелекту, де ми маємо набагато більше невідомих на наших руках, ніж відомих. Тож варто продовжувати вчитися, опитувати та досліджувати. Ніколи не зупинятися. Оскільки навіть з огляду на прогрес, досягнутий на сьогодні, більшість із принципових питання в штучному інтелекту залишаються без відповіді. Багато хто ще навіть не запитував належним чином.

4 ПРАКТИЧНІСТЬ ВИКОРИСТАННЯ ПРОГРАМИ І ЇЇ АКТУАЛЬНІСТБ

4.1 Чому розроблений алгоритм – це майбутнє

Чи є щось особливе в глибоких нейронних мережах, що робить їх «правильним» підходом для компаній, в які слід інвестувати, і для дослідників, яким слід з'їхатися? Або глибоке навчання - це просто примха, яка може не тривати? Чи будемо ми все ще використовувати глибокі нейронні мережі через 20 років? Глибоке навчання має кілька властивостей, що виправдовують його статус революції штучного інтелекту, і воно тут залишається. Ми можемо не використовувати нейронні мережі через два десятиліття, але все, що ми використовуємо, безпосередньо успадкує сучасне глибоке навчання та його основні концепції. Ці важливі властивості можна розділити на три категорії.

Простота - глибоке навчання позбавляє потреби в конструюванні особливостей, замінюючи складні, крихкі, важкі в інженерному процесі трубопроводи простими наскрізними навчальними моделями, які, як правило, будуються з використанням лише п'яти-шести різних тензорних операцій. Масштабованість - глибоке навчання дуже піддається паралелізації на графічних процесорах або TPU, тому воно може повною мірою скористатися законом Мура. Крім того, моделі глибокого навчання навчаються шляхом ітерації невеликих пакетів даних, що дозволяє їх навчати на наборах даних довільного розміру. (Єдиним вузьким місцем є кількість паралельної обчислювальної потужності, яка завдяки закону Мура є швидкозмінною перешкодою.) Ati Універсальність та багаторазове використання - На відміну від багатьох попередніх підходів до машинного навчання, моделі глибокого навчання можна навчити на додаткових даних без перезапуск з нуля, що робить їх життєздатними для постійного навчання в Інтернеті - важливою властивістю для дуже великих моделей виробництва. Крім того, навчені моделі глибокого навчання можуть бути багаторазово використані і,

таким чином, багаторазові: наприклад, можна взяти модель глибокого навчання, навчену для класифікації зображень, і залишити її в конвеєрі для відеообробки. Це дозволяє нам реінвестувати попередню роботу у все більш складні та потужні моделі. Це також робить глибоке навчання придатним для досить невеликих наборів даних.

Поглиблене навчання було в центрі уваги лише кілька років, і ми ще не встановили повний обсяг того, що він може зробити. З кожним місяцем ми дізнаємося про нові випадки використання та технічні вдосконалення, що скасовують попередні обмеження. Після наукової революції прогрес, як правило, йде за сигмовидною кривою: він починається з періоду швидкого прогресу, який поступово стабілізується, коли дослідники досягають жорстких обмежень, а потім подальші вдосконалення стають поступовими. Поглиблене навчання в 2017 році, здається, припадає на першу половину цієї сигмовидної кишки, а набагато більше прогресу настане в найближчі кілька років.

Поле сучасного глибокого навчання, яке ми знаємо сьогодні, має лише кілька років, незважаючи на довгу повільну передісторію, що тягнеться десятиліттями. З експоненціальним збільшенням фінансових ресурсів та кількості наукових досліджень з 2013 року, область у цілому зараз рухається шаленими темпами. Те, що ви дізналися в цій книзі, не залишатиметься актуальним назавжди, і це не все, що вам знадобиться до кінця кар'єри. На щастя, існує безліч безкоштовних Інтернет-ресурсів, якими можна користуватися, щоб бути в курсі подій та розширювати свій кругозір.

Одним із ефективних способів набути реального досвіду є випробувати свої сили на змаганнях з машинного навчання на Kaggle (https://kaggle.com). Єдиний реальний спосіб навчання - це практика та власне кодування - ось філософія цієї книги, і змагання Kaggle є природним продовженням цього. На Kaggle ви знайдете безліч постійно оновлюваних змагань з науки про дані, багато з яких передбачають глибоке навчання, підготовлене компаніями, зацікавленими в отриманні нових рішень деяких із найскладніших проблем машинного навчання. Досить великі грошові призи пропонуються найкращим учасникам. Більшість змагань

виграються за допомогою бібліотеки XGBoost (для неглибокого машинного навчання) або Keras (для глибокого навчання). Тож ви точно впишетесь! Беручи участь у кількох змаганнях, можливо, в складі команди, ви ознайомитесь з практичною стороною деяких передових передових практик, описаних у цій книзі, особливо з налаштуванням гіперпараметрів, уникаючи переобладнання наборів перевірки та складання моделей.

Розроблена мною програма справді має застосування. Хоч і це не було щось неймовірне, однак ми використали вже існуючі технології для того, щоб створити шось нове, що можна продати, побудувати бізнес і навіть розглянути математичну модель.

Навчання - це подорож на все життя, особливо у галузі штучного інтелекту, де ми маємо набагато більше невідомих на наших руках, ніж відомих. Тож варто продовжувати вчитися, опитувати та досліджувати. Ніколи не зупинятися. Оскільки навіть з огляду на прогрес, досягнутий на сьогодні, більшість із принципових питання в штучному інтелекту залишаються без відповіді. Багато хто ще навіть не запитував належним чином.

ВИСНОВКИ

За результатами виконаних теоретичних та практичних досліджень у дипломній роботі розроблено програмний модуль для розпізнавання марок автомобілів по зображенню.

Також протягом досліджень було здійснено поглиблення навичок самостійної наукової роботи, розширення наукового світогляду, дослідження проблем розпізнавання об’єктів на зображеннях та вміння пов’язувати їх з обраним теоретичним напрямком дослідження. Також було поглиблено теоретичні знання в сфері штучного навчання і алгоритмах нейронної мережі, підібрано фактичний матеріал для дослідження, а саме: можливі способи оптимізації і удосконалення параметрів класифікаційної моделі штучної нейронної мережі. Було оформлено звіт, опрацьовано матеріал, виписано усі використанні джерела.

Машинне навчання — це така підгалузь штучного інтелекту в сфері інформатики, котрі часто використовує статистичні прийоми для надання комп'ютерам здатності навчатися. Тобто іншими словами вона надає можливість поступово покращувати продуктивність у певній задачі. Навчання відбувається з даних і після того комп’ютер буде розпізнавати, а не бути програмованим явно. Коли машинне навчання стало відділене до окремії області навчання, воно почало бурхливо розвиватися в 1990-х роках. Данна область змінила свої цілі з досягання штучного інтелекту на розв'язання розв'язних задач практичного характеру. Також вона безпомередньо змістила фокус із символьних підходів, котрі були успадковані нею від штучного інтелекту, в бік методів та моделей, позичених зі статистики та теорії ймовірності. Ця область також виграла від збільшуваної доступності оцифрованої інформації та можливості розповсюджування її через Інтернет, як ми говорили у роботі.

Добування даних і машинне навчання часто використовують одні й ті ж методи, і достатньо перекриваються, але в той ж час як машинне навчання фокусується на передбаченні на основі відомих властивостей, вивчених з даних для

тренування, добування даних фокусується на відкритті раніше невідомих властивостей даних. Тобто це є кроком аналізу з відкривання знань у базах даних. Добування даних застосовує дуде багато методів машинного навчання, але з іншими цілями. З однієї сторони, машинне навчання використовує методи добування даних як крок попередньої обробки для покращення точності механізму навчання або як «навчання без учителя». В свою чергу у машинному навчанні продуктивність зазвичай оцінюється по відношенню до здатності відтворювати відоме знання, тоді як у відкриванні знань та добуванні даних ключовою задачею є відкривання не відомого раніше знання.

У роботі проведено аналіз базових принципів та архітектурних складових згорткових мереж, досліджено виявлені проблеми у сучасних згорткових нейронних мережах при розпізнаванні зображень. Також було обґрунтовано необхідність створення нового типу передачі даних між шарами мережі замість існуючого методу агрегування.

І хоча згорткові нейронні мережі успішно працюють та виконують свої задачі, однак логіка інваріантності здається хибною при розпізнаванні зображень. Правильно буде очікувати, що результат зміститься так само, відображаючи зміну положення об’єкта на зображенні. А неправильно очікувати, що при зміні положення чого-небудь на зображенні, результат буде один і той самий.

Можна сказати, що основні ідеї згортки нейронів працюють логічно і вірно, однак саме чергування згорткових шарів з агрегувальними шарами є ключовим неправильним принципом котрий забирає з процесу навчання важливу інформацію про знайдені ознаки. В наш час згорткові нейронні мережі, які показують найкращі результати у розпізнаванні зображень, грунтуються на наступних принципах:

1) використовується багато шарів згортки навчених детекторів ознак;
2) детектори ознак є локальними і кожен тип детектору поширений на всю мережу;
3) набори детекторів ознак стають все більшими у верхніх шарах;
4) шари виділення ознак чергуються з шарами підвибірки, які агрегують виходи сусідніх детекторів ознак одного типу.

За результатами виконаних практичних і теоретичних досліджень було оптимізовано класифікаційну модель згорткової нейронної мережі за допомогою підбору оптимальних гіперпараметрів для алгоритму навчання.

На основі досліджень та тестування визначено існуючі проблеми при розпізнаванні об'єктів на картинках за допомогою згорткових нейронних мережам. До них належать: необхідність попереднього штучного розширення колекції даних для розпізнавання невеликих трансформаційних змін об'єкта, можливість обману мережі зміною одного пікселя у вхідному зображенні, а також можливість обману мережі додаванням обрахованого градієнту до вхідного зображення.

Згорткові нейронні мережі є інваріантними до зсуву ознак вхідного зображення, однак насправді правильною логікою можна вважати зміну характеристик результату при зміні початкових положень різних об'єктів картинки.

Тому є актуальна потреба у створенні нового типу роутингу нейронів нижчого рівня до високорівневих шарів, котрий би враховував просторові властивості кожної знайденої ознаки, і безпосередньо також їх взаємопов'язаність при формуванні всього знайденого об'єкта.

В роботі вирішено завдання створення математичного забезпечення для побудови розпізнаванні об'єктів на основі згорткових нейронних мереж. Запропоновано архітектуру згорткової нейронної мережі, що може використовуватися для даних, в яких вхідні значення не пов'язані між собою. Порівняно результати тестування всіх побудованих моделей. За результатами тестування визначено, що якість запропонованої моделі вища, але на її навчання поребується більше часу.

Навчання - це подорож на все життя, особливо у галузі штучного інтелекту, де ми маємо набагато більше невідомих на наших руках, ніж відомих. Тож варто продовжувати вчитися, опитувати та досліджувати. Ніколи не зупинятися. Оскільки навіть з огляду на прогрес, досягнутий на сьогодні, більшість із принципових питання в штучному інтелекту залишаються без відповіді. Багато хто ще навіть не запитував належним чином.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. James A. Scott. Getting Started with Apache Spark / Scott. A. James. – USA: MapR technologies Inc, 2015. – С. 88.

2. Karau H. Learning Spark: Lightning-Fast Big Data Analysis / H. Karau, A. Konwinski, P. Wendell, M. Zaharia, Karau H. Learning Spark. – USA: O'Reilly Media Inc, 2015. – С. 257.

3. Jacek Laskowski. Mastering Apache Spark / Laskowski Jacek. – USA: Gitbooks Inc, 2018. – С. 1352.

4. Plattner H. In Memory Data Management: Technology and Applications / H. Plattner. A. Zeier. – Springer Science & Business Media, 2012 – С. 266.

5. Nils J. Nilsson - Introduction to Machine Learning 2015. – С. 35.

6. Trevor Hastie, Robert Tibshirani and Jerome H. Friedman - The Elements of Statistical Learning 2001. – С. 408.

7. Pedro Domingos, - The Master Algorithm, Basic Books, 2015. – С. 48.

8. Ian H. Witten and Eibe Frank - Data Mining: Practical machine learning tools and techniques Morgan Kaufmann, 2011. – С. 40-60.

9. Ethem Alpaydin - Introduction to Machine Learning, 2004. – С. 9-10.

10. David J. C. MacKay. - Information Theory, Inference, and Learning Algorithms Cambridge. 2015. – С. 90-100.

11. Richard O. Duda, Peter E. Hart, David G. Stork - Pattern classification (2nd edition, 2001. – С. 43-54.

12. Christopher Bishop - Neural Networks for Pattern Recognition, 1995. – С. 89.

13. Stuart Russell & Peter Norvig, - Artificial Intelligence — A Modern Approach. Prentice Hall, 2002. – С. 121-199.

14. Ray Solomonoff, - An Inductive Inference Machine, IRE Convention Record, Section on Information Theory, 1957. – С. 56-62.

15. Finnsson, Hilmar, and Yngvi Björnsson. - Simulation-Based Approach to General Game Playing, 2008. – С. 45.

16. Офіційна документація Keras Documentation. [Електронний ресурс] : [Веб-сайт]. – Електронні дані. — США: Keras Documentation – Режим доступу: https://keras.io (дата звернення: 24.09.2018) – Назва із екрана.

17. Vonod Khosla, - Do We Need Doctors or Algorithms?, 2012 – С.12-44.

18. Why Machine Learning Models Often Fail to Learn: QuickTake Q&A. Bloomberg.com. 2016-11-10. Процитовано 2017-04-10. (англ.)

19. Simonite, Tom. Microsoft says its racist chatbot illustrates how AI isn't adaptable enough to help most businesses. MIT Technology Review (en). Процитовано 2017-04-10. (англ.)

20. Kohavi, Ron, - A Study of Cross-Validation and Bootstrap for Accuracy Estimation and Model Selection. International Joint Conference on Artificial Intelligence. 1995 – С. 90-98.

21. Pontius, Robert Gilmore; Si, Kangping, - The total operating characteristic to measure diagnostic ability for multiple thresholds. International Journal of Geographical Information Science 28, 2014. – С. 15-16.

22. Bostrom, Nick, - The Ethics of Artificial Intelligence, 2011. Процитовано 11 серпня 2016 р. (англ.).

23. Edionwe, Tolulope. The fight against racist algorithms. The Outline. Процитовано 17 November 2017. (англ.)

24. Jeffries, Adrianne. Machine learning is racist because the internet is racist. The Outline. Процитовано 17 November 2017. (англ.)

25. Hao, Karen; Stray, Jonathanm - Can you make AI fairer than a judge? Play our courtroom algorithm game [Чи можете ви зробити ШІ справедливішим за суддю? Зіграйте в нашу гру з алгоритмом для суду]. MIT Technology Review, 2019 – С. 302-303.

26. Ширяев А. Н. Вероятность, — М.: Наука. 1989 – С. 40-56.

27. Колмогоров А. Н., Фомин С. В. Элементы теории функций и функционального анализа. — 4-е изд. — Москва : Наука, 1976. – С. 56-70.

28. Fogel, David Blondie24: Playing at the Edge of AI. San Francisco, CA: Morgan Kaufmann, 2001 – С. 44-56.

29. Clark, Christopher; Storkey, Amos. «Teaching Deep Convolutional Neural Networks to Play Go», 2014 – С. 103-107.

30. Maddison, Chris J.; Huang, Aja; Sutskever, Ilya; Silver, David. «Move Evaluation in Go Using Deep Convolutional Neural Networks». 2014 – С. 67-79.

31. AlphaGo – Google DeepMind. Архів оригіналу за 30 січня 2016. Процитовано 30 January 2016. (англ.)

32. Durjoy Sen Maitra; Ujjwal Bhattacharya; S.K. Parui, "CNN based common approach to handwritten character recognition of multiple scripts," in Document Analysis and Recognition (ICDAR), 2015 13th International Conference on, vol., no., pp.1021–1025, 23–26 Aug. 2015 (англ.) – С. 60-100.

33. Mnih, Volodymyr (2015). Human-level control through deep reinforcement learning. Nature 518 (7540): 529–533. Bibcode:2015Natur.518..529M. PMID 25719670. doi:10.1038/nature14236. (англ.) – С. 40-50.

34. Sun, R.; Sessions, C. (June 2000). Self-segmentation of sequences: automatic formation of hierarchies of sequential behaviors. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics) 30 (3): 403–418. ISSN 1083-4419. doi:10.1109/3477.846230. (англ.) – С. 67-89.

35. Lee, Honglak; Grosse, Roger; Ranganath, Rajesh; Ng, Andrew Y. (1 January 2009). Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. Proceedings of the 26th Annual International Conference on Machine Learning – ICML '09 (ACM): 609–616. ISBN 9781605585161. doi:10.1145/1553374.1553453 — через ACM Digital Library. (англ.) – С. 43-56

36. Convolutional Deep Belief Networks on CIFAR-10. (англ.)

37. Офіційна документація Apache Hadoop. [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Сан Франциско : Apache Software Foundation – Режим доступу: http://hadoop.apache.org/docs/current/ (дата звернення: 25.09.2018) – Назва з екрана.

38. Офіційна документація Databricks [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Сан Франциско : Databricks Inc. – Режим доступу: https://docs.databricks.com/ (дата звернення: 01.10.2018) – Назва з екрана.

39. Литвинов О. А. Розподілена обробка інформації / О. А. Литвинов, В. С. Хандецький – Д.: ТОВ «Баланс-Клуб», 2013. – 314 с.

40. Пашков О. М. Автоматизовані системи зберігання і видачі документів: Виникнення та тенденції розвитку / О. М. Пашков. – Бібл. планета. – Київ, 2003. – №2. – С. 33-34.

ДОДАТОК А

Лістинг коду для стискання зображення

```
import os
import random
import shutil
import tarfile

import cv2 as cv
import numpy as np
import scipy.io
from tqdm import tqdm

def ensure_folder(folder):
    if not os.path.exists(folder):
        os.makedirs(folder)

def save_train_data(fnames, labels, bboxes):
    src_folder = 'cars_train'
    num_samples = len(fnames)

    train_split = 0.8
    num_train = int(round(num_samples * train_split))
    train_indexes = random.sample(range(num_samples), num_train)

    for i in tqdm(range(num_samples)):
        fname = fnames[i]
        label = labels[i]
        (x1, y1, x2, y2) = bboxes[i]

        src_path = os.path.join(src_folder, fname)
        src_image = cv.imread(src_path)
        height, width = src_image.shape[:2]
```

```
        # margins of 16 pixels
        margin = 16
        x1 = max(0, x1 - margin)
        y1 = max(0, y1 - margin)
        x2 = min(x2 + margin, width)
        y2 = min(y2 + margin, height)
        # print("{} -> {}".format(fname, label))

        if i in train_indexes:
            dst_folder = 'data/train'
        else:
            dst_folder = 'data/valid'

        dst_path = os.path.join(dst_folder, label)
        if not os.path.exists(dst_path):
            os.makedirs(dst_path)
        dst_path = os.path.join(dst_path, fname)

        crop_image = src_image[y1:y2, x1:x2]
        dst_img = cv.resize(src=crop_image, dsize=(img_height, img_width))
        cv.imwrite(dst_path, dst_img)

def save_test_data(fnames, bboxes):
    src_folder = 'cars_test'
    dst_folder = 'data/test'
    num_samples = len(fnames)

    for i in tqdm(range(num_samples)):
        fname = fnames[i]
        (x1, y1, x2, y2) = bboxes[i]
        src_path = os.path.join(src_folder, fname)
        src_image = cv.imread(src_path)
        height, width = src_image.shape[:2]
        # margins of 16 pixels
        margin = 16
```

```
        x1 = max(0, x1 - margin)
        y1 = max(0, y1 - margin)
        x2 = min(x2 + margin, width)
        y2 = min(y2 + margin, height)
        # print(fname)

        dst_path = os.path.join(dst_folder, fname)
        crop_image = src_image[y1:y2, x1:x2]
        dst_img = cv.resize(src=crop_image, dsize=(img_height, img_width))
        cv.imwrite(dst_path, dst_img)

def process_train_data():
    print("Processing train data...")
    cars_annos = scipy.io.loadmat('devkit/cars_train_annos')
    annotations = cars_annos['annotations']
    annotations = np.transpose(annotations)

    fnames = []
    class_ids = []
    bboxes = []
    labels = []

    for annotation in annotations:
        bbox_x1 = annotation[0][0][0][0]
        bbox_y1 = annotation[0][1][0][0]
        bbox_x2 = annotation[0][2][0][0]
        bbox_y2 = annotation[0][3][0][0]
        class_id = annotation[0][4][0][0]
        labels.append('%04d' % (class_id,))
        fname = annotation[0][5][0]
        bboxes.append((bbox_x1, bbox_y1, bbox_x2, bbox_y2))
        class_ids.append(class_id)
        fnames.append(fname)

    labels_count = np.unique(class_ids).shape[0]
```

```
    print(np.unique(class_ids))
    print('The number of different cars is %d' % labels_count)

    save_train_data(fnames, labels, bboxes)


def process_test_data():
    print("Processing test data...")
    cars_annos = scipy.io.loadmat('devkit/cars_test_annos')
    annotations = cars_annos['annotations']
    annotations = np.transpose(annotations)

    fnames = []
    bboxes = []

    for annotation in annotations:
        bbox_x1 = annotation[0][0][0][0]
        bbox_y1 = annotation[0][1][0][0]
        bbox_x2 = annotation[0][2][0][0]
        bbox_y2 = annotation[0][3][0][0]
        fname = annotation[0][4][0]
        bboxes.append((bbox_x1, bbox_y1, bbox_x2, bbox_y2))
        fnames.append(fname)

    save_test_data(fnames, bboxes)


if __name__ == '__main__':
    # parameters
    img_width, img_height = 224, 224

    print('Extracting cars_train.tgz...')
    if not os.path.exists('cars_train'):
        with tarfile.open('cars_train.tgz', "r:gz") as tar:
            tar.extractall()
```

```
print('Extracting cars_test.tgz...')
if not os.path.exists('cars_test'):
    with tarfile.open('cars_test.tgz', "r:gz") as tar:
        tar.extractall()
print('Extracting car_devkit.tgz...')
if not os.path.exists('devkit'):
    with tarfile.open('car_devkit.tgz', "r:gz") as tar:
        tar.extractall()

cars_meta = scipy.io.loadmat('devkit/cars_meta')
class_names = cars_meta['class_names']  # shape=(1, 196)
class_names = np.transpose(class_names)
print('class_names.shape: ' + str(class_names.shape))
print('Sample class_name: [{}]'.format(class_names[8][0][0]))

ensure_folder('data/train')
ensure_folder('data/valid')
ensure_folder('data/test')

process_train_data()
process_test_data()

# clean up
shutil.rmtree('cars_train')
shutil.rmtree('cars_test')
# shutil.rmtree('devkit')
```

ДОДАТОК Б
ЛІСТИНГ КОДУ ДЛЯ ОТРИМАННЯ КОЛЕКЦІЇ ЗОБРАЖЕНЬ

```
$ cd Car-Recognition
$ wget http://imagenet.stanford.edu/internal/car196/cars_train.tgz
$ wget http://imagenet.stanford.edu/internal/car196/cars_test.tgz
$ wget --no-check-certificate  https://ai.stanford.edu/~jkrause/cars/car_devkit.tgz
```

ДОДАТОК В
ЛІСТИНГ КОДУ АЛГОРИТМУ ТРЕНУВАННЯ

```
import keras
from resnet_152 import resnet152_model
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import CSVLogger, ModelCheckpoint, EarlyStopping
from keras.callbacks import ReduceLROnPlateau

img_width, img_height = 224, 224
num_channels = 3
train_data = 'data/train'
valid_data = 'data/valid'
num_classes = 196
num_train_samples = 6549
num_valid_samples = 1595
verbose = 1
batch_size = 16
num_epochs = 100000
patience = 50

if __name__ == '__main__':
    # build a classifier model
    model = resnet152_model(img_height, img_width, num_channels, num_classes)

    # prepare data augmentation configuration
    train_data_gen = ImageDataGenerator(rotation_range=20.,
                                        width_shift_range=0.1,
                                        height_shift_range=0.1,
                                        zoom_range=0.2,
                                        horizontal_flip=True)
    valid_data_gen = ImageDataGenerator()
    # callbacks
```

```
        tensor_board = keras.callbacks.TensorBoard(log_dir='./logs', histogram_freq=0,
write_graph=True, write_images=True)
        log_file_path = 'logs/training.log'
        csv_logger = CSVLogger(log_file_path, append=False)
        early_stop = EarlyStopping('val_acc', patience=patience)
        reduce_lr = ReduceLROnPlateau('val_acc', factor=0.1, patience=int(patience / 4), verbose=1)
        trained_models_path = 'models/model'
        model_names = trained_models_path + '.{epoch:02d}-{val_acc:.2f}.hdf5'
        model_checkpoint = ModelCheckpoint(model_names, monitor='val_acc', verbose=1,
save_best_only=True)
        callbacks = [tensor_board, model_checkpoint, csv_logger, early_stop, reduce_lr]

        # generators
        train_generator = train_data_gen.flow_from_directory(train_data, (img_width, img_height),
batch_size=batch_size,
                                                             class_mode='categorical')
        valid_generator = valid_data_gen.flow_from_directory(valid_data, (img_width, img_height),
batch_size=batch_size,
                                                             class_mode='categorical')

        # fine tune the model
        model.fit_generator(
            train_generator,
            steps_per_epoch=num_train_samples / batch_size,
            validation_data=valid_generator,
            validation_steps=num_valid_samples / batch_size,
            epochs=num_epochs,
            callbacks=callbacks,
            verbose=verbose)
```

ДОДАТОК Г
ЛІСТИНГ КОДУ ДЛЯ ТЕСТУВАННЯ

```
import json
import os
import random

import cv2 as cv
import keras.backend as K
import numpy as np
import scipy.io

from utils import load_model

if __name__ == '__main__':
    img_width, img_height = 224, 224
    model = load_model()
    model.load_weights('models/model.96-0.89.hdf5')

    cars_meta = scipy.io.loadmat('devkit/cars_meta')
    class_names = cars_meta['class_names']  # shape=(1, 196)
    class_names = np.transpose(class_names)

    test_path = 'data/test/'
    test_images = [f for f in os.listdir(test_path) if
                   os.path.isfile(os.path.join(test_path, f)) and f.endswith('.jpg')]

    num_samples = 20
    samples = random.sample(test_images, num_samples)
    results = []
    for i, image_name in enumerate(samples):
        filename = os.path.join(test_path, image_name)
        print('Start processing image: {}'.format(filename))
```

```
    bgr_img = cv.imread(filename)
    bgr_img = cv.resize(bgr_img, (img_width, img_height), cv.INTER_CUBIC)
    rgb_img = cv.cvtColor(bgr_img, cv.COLOR_BGR2RGB)
    rgb_img = np.expand_dims(rgb_img, 0)
    preds = model.predict(rgb_img)
    prob = np.max(preds)
    class_id = np.argmax(preds)
    text = ('Predict: {}, prob: {}'.format(class_names[class_id][0][0], prob))
    results.append({'label': class_names[class_id][0][0], 'prob': '{:.4}'.format(prob)})
    cv.imwrite('images/{}_out.png'.format(i), bgr_img)

print(results)
with open('results.json', 'w') as file:
    json.dump(results, file, indent=4)

K.clear_session()
```

ДОДАТОК Д
ТЕЗИ НА ВСЕУКРАЇНСЬКУ КОНФЕРЕНЦІЮ

Руденко І.В.

Хмельницький національний університет

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ ДЛЯ КЛАСИФІКАЦІЇ МАРОК АВТОМОБІЛІВ З ВИКОРИСТАННЯМ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

Розглянуто прикладні аспекти розробки інформаційної системи для класифікації та розпізнаванню марок автомобілів на фотографіях та візуалізації отриманих результатів для подальшого аналізу. Запропонована інформаційна система забезпечує точну і швидку класифікацію розпізнавання об'єктів на фото відповідно до заданих категорій.

Applied aspects of information system development for classification and car brands in the photo retrieval and visualization of the obtained results for further analysis are considered. The offered information system provides accurate and fast classification of the objects in photo according to the given categories.

З розвитком нових технологій та постійним підвищенням рівня інформатизації суспільства проблема машинної класифікації та розпізнавання об'єктів на фото набуває особливого значення.

Оптимізація гіперпараметрів — задача машинного навчання по вибору множини оптимальних гіперпараметрів для алгоритму машинного навчання. Гіперпараметр є параметром, значення якого використовується для керування процесом навчання. На відміну від значень інших параметрів (наприклад, вагових коефіцієнтів), які потрібно вивчити.

Одні й ті ж види моделей машинного навчання можуть мати різні обмеження, ваги або потребувати певної швидкості навчання для різних видів даних. Ці параметри називаються гіперпараметрами і їх слід підбирати так, щоб модель могла оптимально вирішити завдання навчання. Для цього знаходиться кортеж гіперпараметрів, який дає оптимальну модель, що оптимізує задану функцію втрат на заданих незалежних даних. Цільова функція бере кортеж гіперпараметрів і повертає пов'язані з ними втрати. Часто використовується перехресне затверджування для оцінки цієї узагальнюючої здатності.

Метою роботи є розробка інформаційної системи для класифікації марок автомобілів з використанням згорткової нейронної мережі та візуалізації отриманих результатів для подальшого аналізу.

З існуючих підходів розглянемо найвідоміші дієві варіанти реалізації:

- пошук по ґратці;
- випадковий пошук;
- байєсова оптимізація;
- оптимізація на основі градієнтів;
- уволюційна оптимізація;
- на основі заселення;
- інше (на основі радіально-базисної функції (РБФ) і спектрального методу).

Налаштування гіперпараметрів згорткової нейронної мережі трохи складніше, ніж налаштування щільних мереж. Це пов'язано з тим, що алгоритм використовує випадковий пошук найкращої з можливих моделей, що, в свою чергу, може призвести до невідповідності кількох умов, щоб цього не сталося, нам потрібно розробити архітектуру згорткової нейронної мережі, а потім відрегулювати гіперпараметри в алгоритмі, щоб отримати нашу найкращу модель.

Згорткові нейронні мережі (рис. 1) мають кілька різних фільтрів / ядер, що складаються з параметрів, що піддаються навчанню, які можуть обертатися на даному зображенні просторово для виявлення таких функцій, як краї та фігури.

Рисунок 1 – Алгоритм згортки

Ця велика кількість фільтрів по суті вчиться фіксувати просторові особливості із зображення на основі вивчених ваг шляхом зворотного розповсюдження, а складені шари фільтрів можуть бути використані для виявлення складних просторових форм з просторових об'єктів на кожному наступному рівні. Отже, вони можуть успішно звести дане зображення у дуже абстраговане зображення, яке легко передбачити.

У щільних мережах ми намагаємось знайти закономірності у значеннях пікселів, що вводяться як напр. якщо пікселі 25 і 26 перевищують певне значення, це може належати до певного класу та кількох складних варіацій того самого. Воно може легко вийти з ладу, якщо ми можемо мати об'єкти де завгодно на зображенні і не обов'язково в центрі.

Функція активації ReLU (рис.2) - це індивідуальна математична операція.

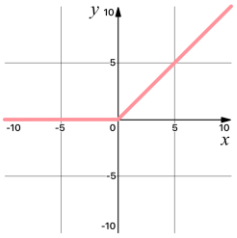


Рисунок 2 - Графічна функція активації ReLU, яка ігнорує всі негативні дані.

Ця функція активації застосовується поетапно до кожного значення вхідного тензора. Наприклад, якщо застосувати ReLU до значення 2.24, результат буде 2.24, оскільки 2.24 більше 0. Ви можете спостерігати, як застосовується ця функція активації, клацнувши нейрон ReLU у мережі вище. Функція випрямленої лінійної активації (ReLU) виконується після кожного згорткового рівня в мережевій архітектурі, описаній вище. Зверніть увагу на вплив цього шару на карту активації різних нейронів у всій мережі!

$$\text{Softmax}(x_i)=\frac{\exp(x_i)}{\sum_j \exp(x_j)} \quad (1)$$

Операція softmax має ключову мету: переконатися, що виходи мережі складають 1. Через це операції softmax корисні для масштабування результатів моделей на ймовірності. Для візуальної індикації впливу кожного логіту (масштабоване скалярне значення) вони кодуються за допомогою світло-оранжевої → темно-оранжевої кольорової шкали. Після проходження функції softmax кожен клас тепер відповідає відповідній ймовірності.
Формула зображена вижче (1).

Нейронні мережі надзвичайно поширені в сучасних технологіях - адже вони настільки точні! Сьогодні найефективніші мережі складаються з великої кількості шарів, які здатні вивчати все більше і більше можливостей. Частина причин, завдяки яким ці новаторські мережі можуть досягти таких надзвичайних точностей, полягає в їх нелінійності. ReLU застосовує в моделі вкрай необхідну нелінійність. Нелінійність необхідна для отримання нелінійних меж рішення, так що вихідні дані не можуть бути записані як лінійна комбінація входів. Якби нелінійна функція активації була відсутня, глибокі архітектури мережі перетворилися б на єдиний, еквівалентний згортковий рівень, який не працював би майже так само добре. Функція активації ReLU спеціально використовується як нелінійна функція активації, на відміну від інших нелінійних функцій, таких як Sigmoid, оскільки емпірично зауважено, що мережа, що використовують ReLU, швидше навчається, ніж її аналоги.

Отже, розроблений алгоритм, який використовує згорткову нейронну мережу, може бути застосований для розв'язання завдань онлайн-розпізнавання об'єктів на фото. Точність розпізнавання достатньо висока. Ефективність реалізованої системи залежить від розміру навчальної та тестової вибірок, кількості шарів та нейронів у кожному шарі. Точність вирішення поставленого завдання з використанням моделі згорткової нейронної може бути поліпшена практично до 99%.

Перелік посилань.

1. Nils J. Nilsson, Introduction to Machine Learning. (англ.)
2. Trevor Hastie, Robert Tibshirani and Jerome H. Friedman (2001). The Elements of Statistical Learning, Springer. ISBN 0-387-95284-5. (англ.)
3. Pedro Domingos (September 2015), The Master Algorithm, Basic Books, ISBN 978-0-465-06570-7 (англ.)

Дані про авторів:

ПІБ автора	Телефон	Email
Руденко Інна Вікторівна	+380969635486	nieraud20@gmail.com

Тема:
Інформаційна технологія для класифікації марок автомобілів з використанням згорткової нейронної мережі

Автор: Руденко І.В.
Керівник: Медзатий Д.М., к.т.н., доцент

Об’єкт дослідження:
Проблема керування процесом навчання нейронної мережі

Предмет дослідження:
Оптимізація гіперпараметрів для оптимального навчання згорткової нейронної мережі

Мета: Удосконалення та оптимізація параметрів класифікаційної моделі

Завдання:
Створити алгоритм для навчання згорткової нейронної мережі і підібрати оптимальні параметри для розпізнавання марок автомобілів

Аналіз предметної області і теоретичні основи

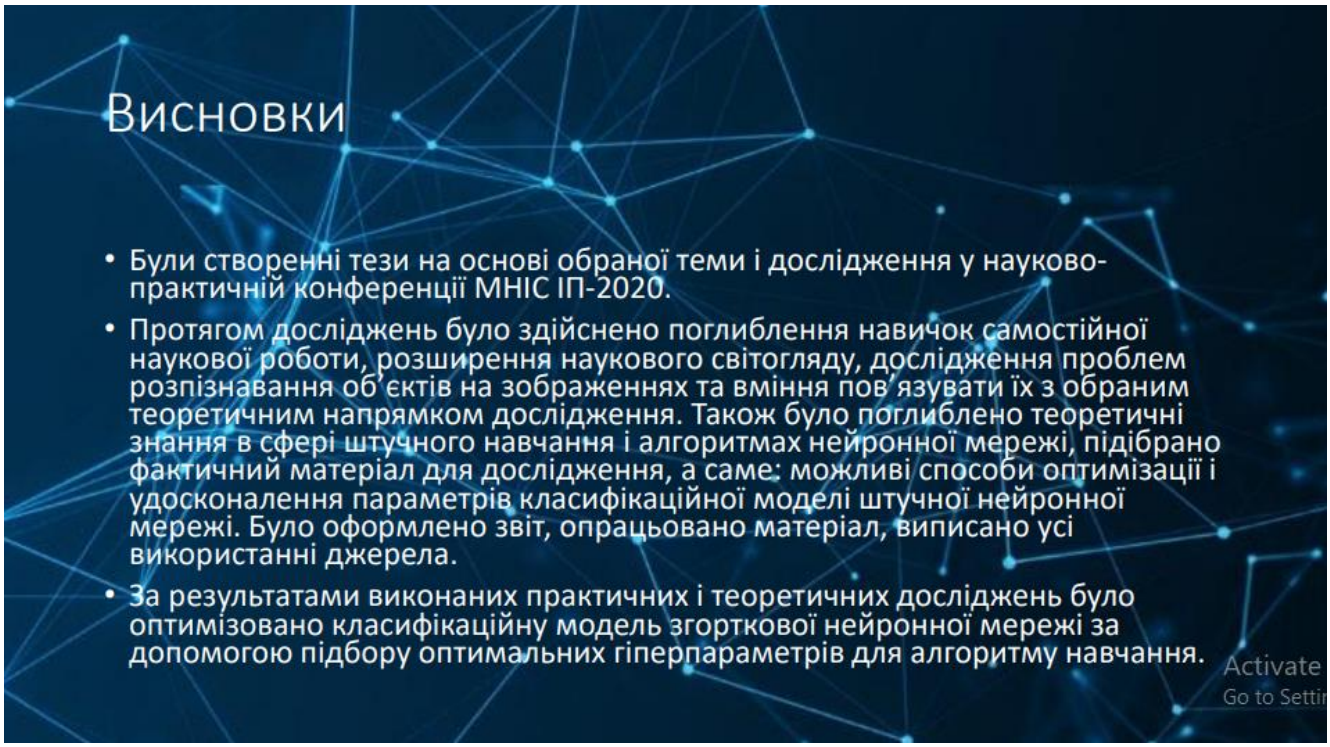
- У розділі ми ознайомлюємося з термінологією, історією і поняттями, що таке машинне навчання в цілому, де використовується. Також наводиться перелік існуючих математичних моделей і для чого вони були створенні. Також говоримо про їх переваги і недоліки.

Аналітичний огляд обробки джерел інформації нейронних мереж

- У данному розділі ми вже починаємо детально розглядати згорткову нейронну мережу, як вона працює і для чого була створена. Також розглядаємо роботу математичної моделі згорткової нейронної мережі і її модель навчання, застосування в реальному світі. Проаналізуємо існуючі підходи до реалізації і обговоримо їхні мінуси та плюси.
  Також почнемо роботу з підбором гіперпараметрів, проаналізуємо які вже алгоритми існують і запропонуємо свій варіант згідно дослідження.
  Наведемо плюси і мінуси нашої математичної моделі підбору гіперпараметрів.

Реалізація алгоритму з підбором оптимальних гіперпараметрів згорткової нейронної мережі

- В данному розділі ми почнемо детальніше пояснення нашого дослідження, розкажемо, який саме метод підбору гіперпараметрів ми використали і як удосконалили для оптимальної роботи згорткової нейронної мережі.
  Отримаємо результати і запишемо в таблицю, порівняємо з результатами існуючих підходів підбору оптимальних гіперпараметрів.
  Напишемо алгоритм, який буде розпізнавати марки автомобілів на фото.


Висновки
• Були створенні тези на основі обраної теми і дослідження у науково-практичній конференції МНІС ІП-2020.
• Протягом досліджень було здійснено поглиблення навичок самостійної наукової роботи, розширення наукового світогляду, дослідження проблем розпізнавання об'єктів на зображеннях та вміння пов'язувати їх з обраним теоретичним напрямком дослідження. Також було поглиблено теоретичні знання в сфері штучного навчання і алгоритмах нейронної мережі, підібрано фактичний матеріал для дослідження, а саме: можливі способи оптимізації і удосконалення параметрів класифікаційної моделі штучної нейронної мережі. Було оформлено звіт, опрацьовано матеріал, виписано усі використанні джерела.
• За результатами виконаних практичних і теоретичних досліджень було оптимізовано класифікаційну модель згорткової нейронної мережі за допомогою підбору оптимальних гіперпараметрів для алгоритму навчання.
Activate
Go to Settin

UNICHECK

Имя пользователя:
Kafedra TMIT KhNU

ID проверки:
1005430202

Дата проверки:
11.12.2020 10:59:27 EET

Тип проверки:
Doc vs Internet

Дата отчета:
11.12.2020 11:07:34 EET

ID пользователя:
100005657

Название файла: Руденко_ПМм-19-1 (повторно)

Количество страниц: 103 **Количество слов:** 22583 **Количество символов:** 171145 **Размер файла:** 2.10 MB **ID файла:** 1005721648

3072 слова помечены как "исключенные" и не учитываются в подсчете слов

7.94% Совпадения

Наибольшее совпадение: 1.75% с Интернет-источником (https://evergreens.com.ua/ua/articles/cnn.html)

7.94% Источники из Интернета	131	..... Страница 105

Поиск совпадений с Библиотекой не производился

0.56% Цитат

Цитаты	12	..... Страница 106

Не найдено ни одной ссылки

0.01% Исключений

Некоторые источники исключены автоматически (фильтры исключения: количество найденных слов меньш...

0.01% Исключений из Интернета	1	..... Страница 107

Нет исключенных библиотечных источников

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы	13

Anti-Plagiarism v-15.257

Максимальное совпадение с одним документом 4.0%

Словари проверки: en_US, ru_RU, ua_UA. **Ошибок в документах: 12%**

	Документ		Суммарное совпадение по Базе Данных	
ID: 81377 Название: Інформаційна технологія для класифікації марок автомобілів з використанням згорткової нейронної мережі Добавлено в БД: 2020-11-26 Авторы: Руденко Інна Вікторівна Руководители: Медзатий Дмитро Миколайович Консультанты: Опоненты:	Символы	Лексемы	Символы	Лексемы
	144651	1068	17252 (12%)	139 (13%)

Источник плагиата

ID	Описание	Наличие плагиата в документе	
		Символы	Лексемы

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

освітнього ступеня «магістр»

Магістр Руденко Інна Вікторівна

Тема Інформаційна технологія для класифікації марок автомобілів з використанням згорткової нейронної мережі

Спеціальність 113 «Прикладна математика»

спеціалізація «Математика та статистика»

Обсяг дипломної роботи освітнього ступеня «магістр»:

кількість листів креслень 0; кількість сторінок записки 104

1. Короткий зміст ДР та прийнятих рішень В рамках магістерської роботи проведено аналіз моделей, методів і алгоритмів підвищення ефективності алгоритму для розпізнавання марок автомобілів на основі нейронних мереж. Розроблено модель методу навчання нейронної мережі. На її основі запропоновано оптимізований метод для кращого і швидшого розпізнавання

2. Висновок про відповідність ДР дипломному завданню Дипломна робота освітнього ступеня «магістр» у повній мірі відповідає поставленому завданню як в теоретичній, так і в практичній частині дипломної роботи.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У вступі обгрунтовується актуальність теми роботи, дається аналіз досліджуваної проблеми і обгрунтовується застосовуваний підхід до її вирішення, формулюються цілі і завдання дослідження, описується наукова новизна і практична значимість отриманих результатів. У першому розділі якісно та в повній мірі проаналізовано сучасний стан алгоритмів для розпізнавання об'єктів на зображення, сфера машинного навчання. Наступні розділи присвячені розробці моделі огляду згорткових нейронних мереж, їх застосування в світі і науці, запропоновані існуючі алгоритми для рішення проблеми, також оптимізація існуючих рішень за допомогою оптимальних гіперпараметрів у використанні для алгоритму навчання. Розглянуто питання застосування розробленого методу.

4. Позитивні сторони проекту Дипломна робота містить ряд інноваційних рішень, зокрема, розроблено метод оптимізації існуючого рішення розпізнавання маленьких об'єктів на зображеннях за допомогою правильного підбору гіперпараметрів.

5. Негативні сторони проекту Розробка представленого методу дозволяє досягти мети, тобто підвищити ефективність розпізнавання марок автомобілів, лише після попереднього навчання нейроної мережі, що суттєво вплине на швидкість проведення дослідження виявлення заражених файлів.

6. Оцінка графічного оформлення та пояснювальної записки роботи Графічне оформлення виконане відповідно до теми дипломної роботи з дотриманням стандартів. В загальному графічне оформлення виконане на достатньому рівні. Пояснювальна записка відповідає нормам для її оформлення.

7. Відгук про роботу в цілому В загальному дипломна робота заслуговує позитивної оцінки. Весь матеріал дипломної роботи структурований, чіткий та послідовний. Усі розділи роботи послідовні та логічні, що дозволяє чітко розуміти викладений матеріал в рамках тематики дипломної роботи. Графічний матеріал дозволяє наочно побачити доцільність та ефективність рішень, які були прийняті за основу для досягнення поставленої задачі.

8. Інші зауваження.

9. Оцінка дипломної роботи Розглянувши позитивні та негативні сторони представленої дипломної роботи, можна зробити висновок, що вона заслуговує оцінку «добре».

РЕЦЕНЗЕНТ (прізвище, ім'я, по батькові, посада, місце роботи) Гурман Іван Васильович доцент кафедри Інженерії програмного забезпечення

« 11 » грудня 2020 . Гурман І.В. (підпис)

ХНУ. Зам. 54, тир. 6000, 2007.

Завідувачу кафедри ТМЗТ
доц. Підченку С.К.

здобувача вищої освіти (студента
ПІБ, факультет, «курс», «група»)
Руденко Інни Вікторівни
2 курс, гр. ПМт-19-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

24.11.2020р.
дата

підпис

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ ТЕЛЕКОМУНІКАЦІЙ, МЕДІЙНИХ ТА ІНТЕЛЕКТУАЛЬНИХ ТЕХНОЛОГІЙ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Інформаційна технологія для розпізнавання марок автомобілів за допомогою згорткової нейронної мережі

Автор: Руденко Інна Вікторівна

Спеціальність: 113 – прикладна математика

Освітня програма: освітньо-професійна

Науковий керівник: Медзатий Дмитро Миколайович, к.т.н доцент

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних). Робота приймається до захисту.	+
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Співпадіння складають 7.94%

Запозичення виявлені в роботі є законними і не є плігіатом, оскільки:

1) найбільші співпадіння з джерелами:
   1. https://www.wikiwand.com/uk3горткова_нейронна_мережа 1,57%
   2. https://evergreens.com.ua/ua/articles/cnn.html 1,75%
   3. https://uk.wikipedia.org/wiki/Машинне_навчання 1,28%

Розміщені в розділах, які не описують безпосередньо авторське дослідження, і не стосуються результатів роботи

2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

3) усі інші джерела запозичення мають відсоток співпадінь менший 1,5%.

16.12.20р. Підченко С.К.

Дата Підпис

Медзатий Д.М.

Підпис