Київський національний торговельно-економічний університет
Кафедра інженерії програмного забезпечення та кібербезпеки

ВИПУСКНИЙ КВАЛІФІКАЦІЙНИЙ ПРОЄКТ

на тему:

«Розробка моделі та архітектури аналітичної системи розпізнавання рукописних цифр»

Студента 2м курсу, 2 групи,
спеціальності 121 «Інженерія
програмного забезпечення»
спеціалізації «Інженерія
програмного забезпечення» _____ Авдєєнко Олексія
підпис студента Юрійовича

Науковий керівник
Кандидат педагогічних наук,
доцент кафедри інженерії
програмного забезпечення та
кібербезпеки _____ Котенко Наталія
підпис керівника Олексіївна

Гарант освітньої програми
доктор економічних наук,
професор кафедри інженерії
програмного забезпечення та
кібербезпеки _____ Токар Володимир
підпис керівника Володимирович

КИЇВ – 2021

Київський національний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення та кібербезпеки
Освітній ступінь магістр
Спеціальність 121 «Інженерія програмного забезпечення»
Спеціалізація «Інженерія програмного забезпечення»

Затверджую
Зав. кафедри інженерії програмного забезпечення та кібербезпеки
Криворучко О. В.
"10" листопада 2020 р.

Завдання
на випускний кваліфікаційний проєкт студента

Авдєєнка Олексія Юрійовича
(прізвище, ім'я, по батькові)

Тема випускного кваліфікаційного проєкту «Розробка моделі та архітектури аналітичної системи розпізнавання рукописних цифр»

Затверджена наказом ректора від "30" листопада 2020 р. № 3923

2. Строк здачі студентом закінченого проєкту 25 листопада 2021

3. Цільова установка та вихідні дані до проєкту

Мета проєкту розробка моделі та архітектури аналітичної системи розпізнавання рукописних цифр.

Об'єктом дослідження виступають алгоритми для вирішення поставленої задачі.

Предметом дослідження є модель нейронної мережі.

4. Консультанти проєкту із зазначенням розділів, які консультують:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв

5. Зміст випускного кваліфікаційного проєкту (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1. ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Machine and Deep Learning

1.2. Система розпізнавання рукописних цифр

1.3. Проблеми з рукописними цифрами

1.4. Набір даних MNIST

1.5 Доступні файли в наборі даних для дипломної роботи

1.6 Висновки до розділу1

РОЗДІЛ 2. ОГЛЯД АЛГОРИТМІВ ВИРІШЕННЯ ПРОБЛЕМИ

2.1. One-Shot Learning

2.2. k-Nearest Neighbors

2.3. Support Vector Machine

2.4. Convolutional Neural Network

2.5. Висновки та вибір алгоритму до розділу 2

РОЗДІЛ 3. ОПИС ПРАКТИЧНОЇ ЧАСТИНИ

3.1. Підготовка даних

3.2. Моделювання та оцінювання CNN

3.3. Оцінка моделі

3.4. Висновки до розділу 3

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ДОДАТКИ

6. Календарний план виконання проєкту:

№ пор.	Назва етапів випускного кваліфікаційного проєкту	Строк виконання етапів проєкту	
		за планом	фактично
1	2	3	4
1.	*Вибір теми випускного кваліфікаційного проєкту*	*21.09.2020*	*21.09.2020*
2.	*Розробка та затвердження завдання на проєкт магістра (стац/заоч)*	*06.11.2020*	*22.12.2020*
3.	*Вступ та перелік літературних джерел*	*27.02.2021*	*27.02.2021*
4.	*Розробка технічного завдання*	*20.03.2021*	*20.03.2021*
5.	*Розділ 1. Опис предметної області*	*16.04.2021*	*16.04.2021*
6.	*Розділ 2. Огляд алгоритмів вирішення проблеми*	*24.05.2021*	*24.05.2021*
7.	*Розділ 3. Практична частина*	*21.06.2021*	*21.06.2021*
8.	*Розробка програми та методики тестування*	*18.10.2021*	*18.10.2021*
9.	*Написання наукової статті*	*22.05.2021*	*22.05.2021*
10.	*Керівництво користувача*	*21.10.2021*	*21.10.2021*
11.	*Висновки та пропозиції*	*01.11.2021*	*01.11.2021*
12.	*Здача випускного кваліфікаційного проєкту на кафедру (перша перевірка)*	*03.11.2021*	*03.11.2021*
13.	*Підготовка автореферату та презентації доповіді*	*03.11.2021*	*03.11.2021*
14.	*Попередній захист випускного кваліфікаційного проєкту*	*22.11.2021 – 25.11.2021*	*22.11.2021*
15.	*Здача зброшуровано ївипускного кваліфікаційного проєкту*	*25.11.2021*	*25.11.2021*
16.	*Зовнішнє рецензування випускного кваліфікаційного проєкту*	*26.11.2021*	*26.11.2021*
17.	*Підготовка до публічного захисту випускного кваліфікаційного проєкту*		

7. Дата видачі завдання « 10 » листопада 2021 р.

8. Науковий керівник випускного кваліфікаційного проєкту
Котенко Н.О.
(прізвище, ініціали, підпис)

9. Гарант освітньої програми Токар В.В.
(прізвище, ініціали, підпис)

10. Завдання прийняв до виконання студент Авдєєнко О.Ю.
(прізвище, ініціали, підпис)

11. Відгук керівника випускного кваліфікаційного проєкту

_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Науковий керівник випускного кваліфікаційного проєкту

_____
(підпис, дата)

Відмітка про попередній захист ___Котенко Н.О._____
(ПІБ, підпис, дата)

12. Висновок про випускний кваліфікаційний проєкт

Випускний кваліфікаційний проєкт студента ____Авдєєнко О.Ю._____
(прізвище, ініціали)

може бути допущена до захисту екзаменаційній комісії.

Гарант освітньої програми _____Токар В. В._____
(прізвище, ініціали, підпис)

Завідувач кафедри _____Криворучко О. В._____
(підпис, прізвище, ініціали)

«_____»_____20 ___ р.

АНОТАЦІЯ

Метою роботи є розробка моделі та архітектури аналітичної системи розпізнавання рукописних цифр

У процесі роботи досліджувалися алгоритми для вирішення поставленої задачі.

Результатом роботи є модель нейронної мережі.

Обсяг роботи: 33 сторінки, 18 зображень, 7 використаних джерел.

Ключові слова: нейронні мережі, machine learning

ABSTRACT

The purpose of the work is to develop a model and architecture of the analytical system for handwritten number recognition

In the course of work algorithms for the decision of the set task were investigated.

The result is a neural network model.

Volume of work: 33 pages, 18 pictures, 7 used sources.

Keywords: neural networks, machine learning

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

ООП — Об'єктно-орієнтоване програмування.

UI — User Interface.

API — Application Programming Interface.

БД — База Даних.

ЗМІСТ

ВСТУП

Машинне навчання та глибоке навчання відіграють важливу роль у комп'ютерних технологіях та штучному інтелекті. Розпізнавання рукописних цифр набуло популярності, починаючи від початківця машинного навчання і глибокого навчання, до фахівця, який практикує протягом багатьох років та шукає кращі алгоритми для вирішення поставленої задачі. З появою потужних обчислювальних машин, за останні два десятиліття впроваджувалося все більше і більше методів, а також вивчався рівень їх помилок, продуктивність і час виконання. За допомогою глибокого навчання та машинного навчання людські зусилля можуть бути зменшені в розпізнаванні, навчанні, прогнозуванні та багатьох інших легких або складних обчисленнях.

РОЗДІЛ 1.
ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Машинне та глибоке навчання.

У цій роботі представлено розпізнавання рукописних цифр (від 0 до 9) із відомого набору даних MNIST, порівняння таких класифікаторів, як One-Shot learning, k-Nearest Neighbors, Support Vector Machine, Neural Network та Convolutional Neural Network, на основі продуктивності, точності, часу, чутливості, позитивної продуктивності та специфіки з використанням різних параметрів з класифікаторами.

Розробка такої системи включає програму для розуміння і класифікації зображень рукописних 10 цифр (0–9). Дану роботу на наборі даних MNIST можна реалізувати за допомогою кількох методів. Для верифікації та тестування мого алгоритму був запропонований та використаний модифікований набір даних MNIST. База даних модифікованого інституту стандартів і технологій нації (MNIST) складається з 60 000 зображень для навчання і 10 000 зображень для тестування. Для перевірки точності алгоритмів навчання з одноразовим вивченням та скороченням наборів даних, були створені додаткові набори даних. Це дозволило ввести однакові набори даних для кожного алгоритму навчання. Для тестування кожного з алгоритмів знань на класифікацію, були використані повні 10 000 зображень.

1.2 Система розпізнавання цифр

Система розпізнавання цифр - це робота машини для самопідготовки або розпізнавання цифр з різних джерел, таких як електронні листи, банківські чеки, папери, зображення тощо, і в різних реальних сценаріях для розпізнавання рукописного вводу в Інтернеті на комп'ютерних планшетах або в системі, розпізнавання номера номерні знаки транспортних засобів, обробка банківських чекових сум, числові записи у формах, заповнених вручну (скажімо - податкові форми) тощо.

1.3 Проблеми з рукописними цифрами

Рукописні цифри не завжди мають однаковий розмір, ширину, орієнтацію, оскільки вони відрізняються в написанні, тому загальна проблема полягає в класифікації цифр через схожість між цифрами, такими як 1 і 7, 5 і 6, 3 і 8, 2 і 5, 2 і 7 тощо. Багато людей пишуть одну цифру різними рукописними написами. Нарешті, унікальність та різноманітність почерку різних людей також впливають на формування та зовнішній вигляд цифр.

1.4 Набір даних MNIST

Зразки, надані з набору даних MNIST (Модифікований національний інститут стандартів і технологій), включають рукописні цифри на загальну кількість 70 000 зображень, що складаються з 60 000 прикладів у навчальному наборі та 10 000 прикладів у наборах для тестування, обидва з позначеними зображеннями з 10 цифр (від 0 до 9). Це невеликий сегмент із широкого набору, де розмір нормалізували відповідно до розміру 20 * 20 пікселів та не змінювали співвідношення сторін.

Рукописні цифри - це зображення у вигляді 28 * 28 інтенсивності шкали сірого зображення, що представляють зображення разом із першим стовпцем, який буде міткою (від 0 до 9) для кожного зображення. Те саме вибрано для випадку тестового набору, як 10 000 зображень з міткою від 0 до 9.

Янн Лекун, Корінна Кортес та Крістофер Берджес розробили цей набір даних MNIST для оцінки та вдосконалення моделей машинного навчання щодо рукописної проблеми класифікації цифр. Набір даних MNIST був розроблений на основі спеціального набору даних NIST із спеціальною базою даних 3 (співробітники Бюро перепису населення США) та спеціальною базою даних 1 (учні старших класів), яка складається з двійкових зображень від руки написаних цифр. Раніше SD-3 (спеціальна база даних -3) розглядалася як навчання, а SD-1 (спеціальна база даних -1) як набір для тестування з легшим розпізнаванням рівня SD-3. Тому, щоб залишатись складним, роз'єднаним та справедливим серед різних класифікаторів навчання, набір даних NIST був змішаний. Поділ MNIST відбувся за 30000 зразків з SD-3 та 30000 зразків з SD-1 з 250 авторами прибл. та 5000 зразків з SD-3 та інші 5000 зразків з SD-1, щоб сформувати інший набір тестів.

Зображення цифр були зроблені з різних відсканованих цифр, нормалізовані за розміром і виправдані як відцентровані. Це робить його чудовим набором даних для оцінки моделей і дозволяє претендентам на машинне навчання зосередитись на глибокому навчанні та машинному навчанні з дуже невеликою кількістю очищення даних.

Говорячи про нову або більш модифіковану версію, яка схожа на стандартну MNIST, у 2017 році з'явився EMNIST або Розширений MNIST, де в навчальному наборі були встановлені зразки 2 40000 зображень разом із збільшенням до 40000 зображень у наборі для тестування що складається з рукописних цифр.

1.5 Доступні файли в наборі даних

Отже, перед тим, як починати глибше в цій темі, кращим моментом є ознайомлення з наданим набором даних. Наступні пункти однакові в навчальному та тестовому наборі, а також у наборі файлів зображень та лейблів.

Пікселі розташовані по рядках, від 0 до 255, відповідно до кольорового коду RGB. Фон як білий (значення 0 від RGB), а передній план як чорний (значення 255 від RGB).

Лейбли цифр, класифіковані від 0 до 9.

Існує чотири файли навчання та тестування:

1. Файли зображень навчального набору (train-images-idx3-ubyte)
2. Файл лейблів навчальних наборів (train-labels-idx1-ubyte)
3. Тестовий набір файлів зображень (t10k-images-idx3-ubyte)
4. Тестовий набір файлів лейблів (t10k-labels-idx1-ubyte)

1.6 Висновки до розділу 1

Машинне навчання та глибоке навчання відіграють важливу роль у комп'ютерних технологіях та штучному інтелекті. Розпізнавання рукописних цифр набуло популярності, починаючи від початківця машинного навчання і глибокого навчання, до фахівця, який практикує протягом багатьох років та шукає кращі алгоритми для вирішення поставленої задачі. З появою потужних обчислювальних машин, за останні два десятиліття впроваджувалося все більше і більше методів, а також вивчався рівень їх помилок, продуктивність і час виконання. За допомогою глибокого навчання та машинного навчання людські зусилля можуть бути зменшені в розпізнаванні, навчанні, прогнозуванні та багатьох інших легких або складних обчислюваннях.

РОЗДІЛ 2.
АЛГОРИТМИ ВИРІШЕННЯ ПОСТАВЛЕНОЇ ЗАДАЧІ

Зараз я представлю поняття та алгоритми глибокого навчання та машинного навчання. В цьому розділі, я хочу показати всі ці методи та алгоритми, для вирішення моєї задачі, порівняти їх та пояснити чому я обрав один алгоритм, а не інший. Ці алгоритми включають Perceptron, k-NN, 3 версії SVM і CNN.

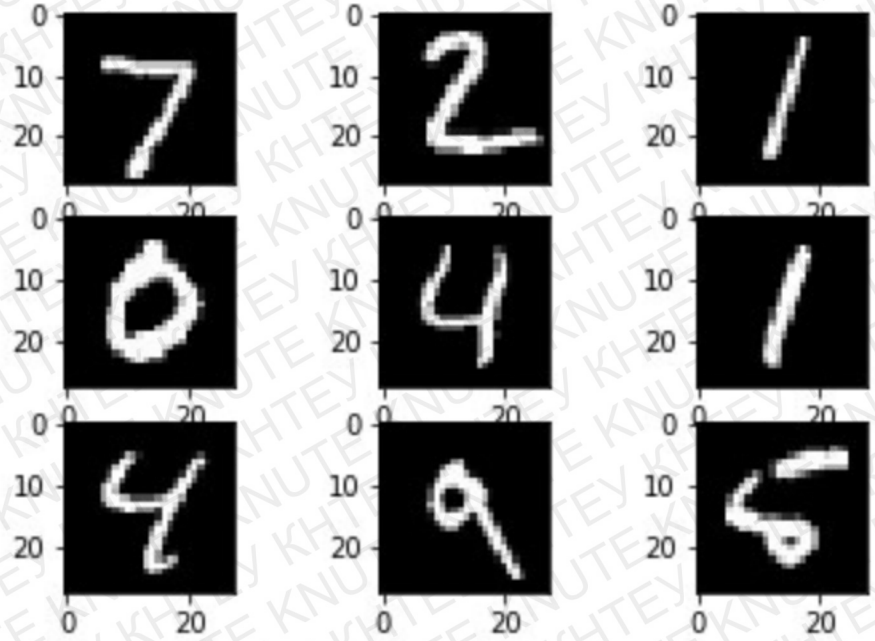


Рис. 2.1. Приклади рукописних цифр

Однак одним із обмежень сучасних методів машинного навчання є те, що вони вимагають багато прикладів.

Наприклад, якщо ви хочете навчити комп’ютер розпізнавати щось, то спочатку потрібно надати комп’ютеру багато фотографій чогось, щоб він міг зрозумів, як це виглядає. Цих прикладів може існувати не дуже багато або вони можуть бути дуже “дорогими” (з точки зору складності алгоритму, часу виконання і тд.) для отримання і розпізнавання. А що, якби ми могли навчити комп’ютер вивчати нове поняття, наприклад, «кішка», лише з одного-двох прикладів? А це якраз саме те, що зробили троє дослідників, Лейк, Салаххутдінов та Тенебаум з MIT.

2.1 One-Shot Learning

Першим алгоритмом для вирішення поставленної задачі є алгоритм - One-Shot Learning. Давайте розглянемо цей алгоритм, що він собою являє та які переваги та недоліки даного алгоритму.

Дослідники цього алгоритму особливо зосередились на розпізнаванні персонажів. Вони запитали: чи можемо ми навчити комп’ютер розпізнавати нових символів, просто побачивши один приклад? Кінцевим результатом був алгоритм, який був настільки ж гарний, як і люди, дізнавшись, як виглядають нові персонажі. Зокрема, алгоритм виконувався настільки ж добре, як і людина.

2.1.1 Розпізнавання символів

Щоб оцінити ефективність алгоритму в цьому тесті, дослідники порівняли ефективність свого алгоритму з ефективністю людей. Спочатку вони зібрали групу рукописних символів з різних алфавітів. Потім дослідники дали кожному учаснику приклад персонажа, якого вони ніколи раніше не бачили, і попросили учасника знайти персонажа в наборі з 20 нових символів того ж алфавіту. Вони попросили їх алгоритм зробити те саме. Дивно, але

алгоритм (рівень помилок 3,3%) працював так само добре, як і люди (середній коефіцієнт помилок 4,5%)!

Рис. 2.2. Символи для тестування алгоритму One-Shot Learning

2.1.2 Генерація символів

Використовуючи ту саму групу рукописних символів, вони давали кожному учаснику приклад персонажа, якого вони ніколи не бачили, а потім просили учасника створити новий приклад цього персонажа. Вони попросили їх алгоритм зробити те саме. Щоб перевірити, наскільки добре працював алгоритм, вони показали судді групу персонажів, створених комп’ютером, та групу символів, написаних людиною, щоб перевірити, чи може суддя розрізнити їх. Судді могли ідентифікувати персонажів, створених комп'ютером, як 52% випадків, а це є не набагато краще, ніж випадкові події (50%).

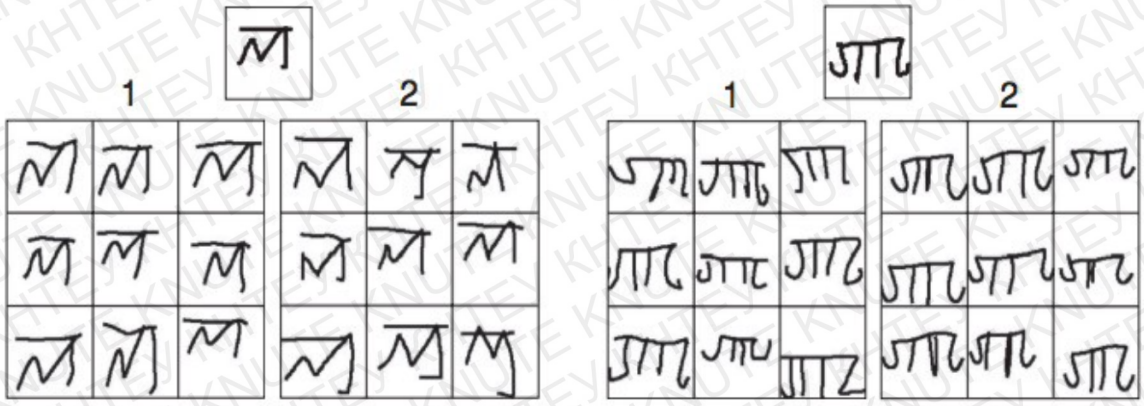


Рис. 2.3. Символи намальовані людиною та комп’ютером.

Важко знайти, які приклади написані машиною. У цьому прикладі сітка 1 ліворуч та сітка 2 праворуч були сформовані машинами.

То як вони це зробили? З огляду на те, наскільки добре працює алгоритм лише на одному прикладі, виникає природне запитання: як вони це зробили?

Основною інтуїцією алгоритму є усвідомлення того, що персонажа можна розглядати як серію обведень. Дослідники навчали алгоритм, як розкласти зображення персонажа на послідовність штрихів, які могли бути використані для написання персонажа. Потім алгоритм може використовувати це подання на основі обведень як основу, на якій можна генерувати нові приклади (наприклад, беручи до уваги інші способи написання обведення) або бачити, які символи можуть бути зіставлені з тим самим шаблоном обведення.

Щоб навчити комп’ютер перетворювати символи на штрихи, дослідники застосували метод, що називається баєсівською програмою навчання. Вони розбили завдання переходу від символу до штриху на частини і змоделювали кожну частину як розподіл ймовірностей (наскільки ймовірно, що є три штрихи, враховуючи, що персонаж виглядає так ... І т. Д.). Перш ніж запускати алгоритм, вони дали комп'ютеру символи з 30 алфавітів, щоб навчити комп'ютер, як повинні виглядати розподіли ймовірностей. Хоча для вивчення початкових ймовірностей йому все ще потрібні були деякі дані, тепер, замість

того, щоб знадобитися тисяча прикладів нового символу, зараз йому потрібен лише один!

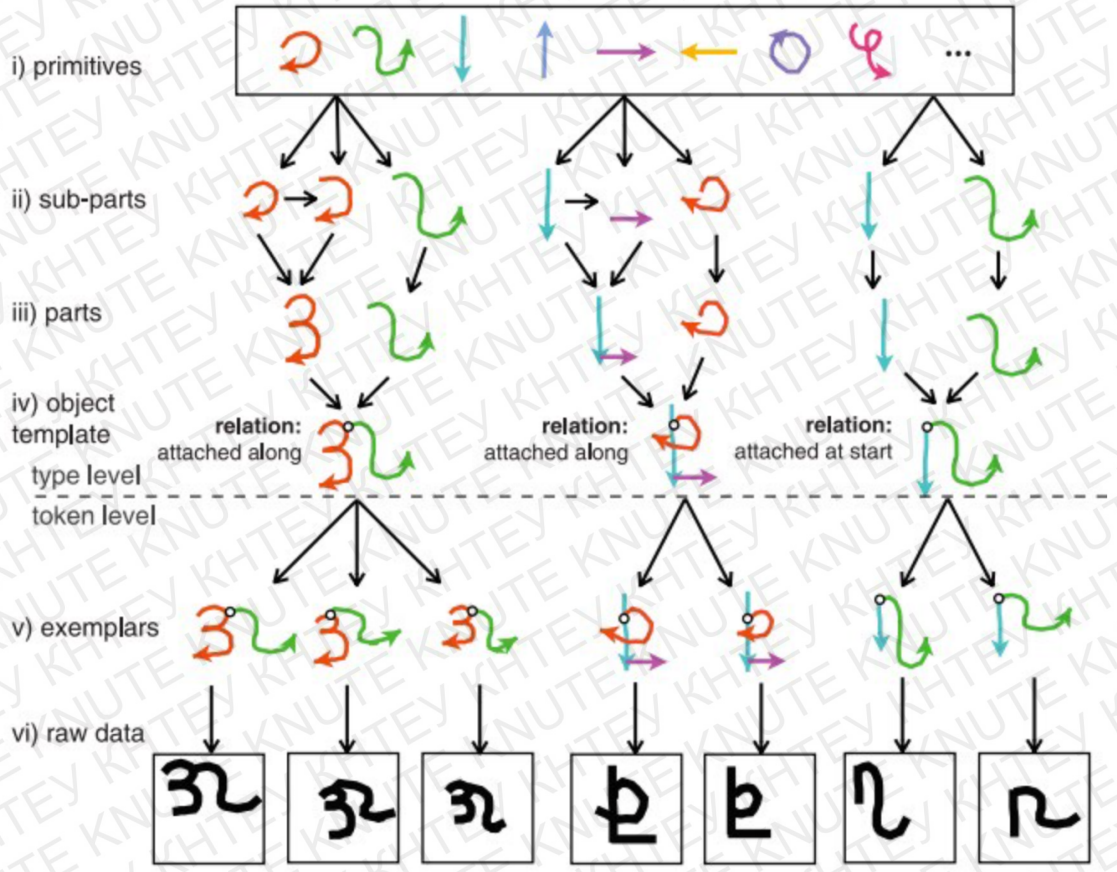


Рис. 2.4. Алгоритм One-Shot Learning

2.1.3 Наступні кроки

Незважаючи на вражаючі успіхи, ще багато роботи ще потрібно зробити. Люди бачать не лише штрихи, коли дивляться на персонажа; вони також можуть помітити такі риси, як паралельні лінії або симетрія. Крім того, додаткові функції можуть викликати багато труднощів. Розглянемо символ “7”. Алгоритм може моделювати його як одночерепний символ, коли вперше бачить його. Однак, побачивши «7» із рискою, він може вважати, що це інший символ, оскільки для цього потрібні два обведення, і ніколи не бачить «7» із

тире. Людина, однак, може зробити висновок, що "7" з рискою - це те саме, що "7" без тире, через контекст чи інші фактори.

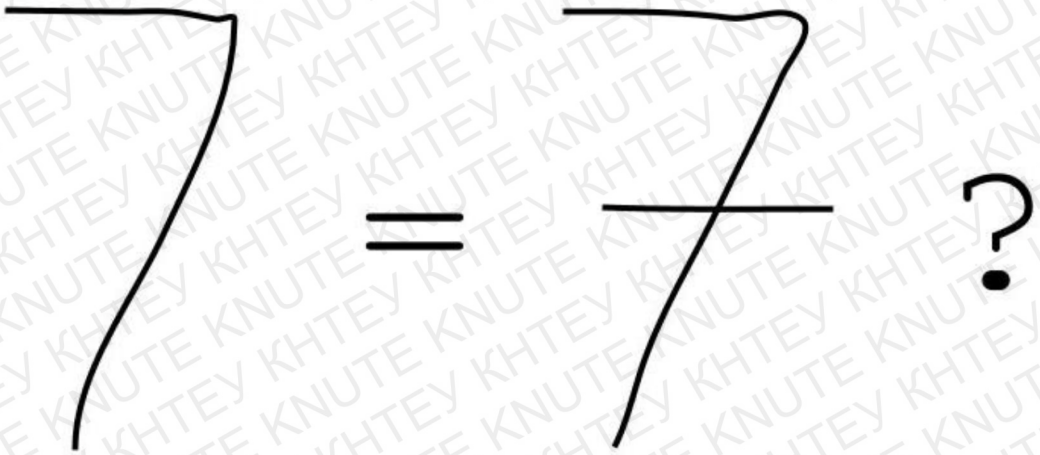
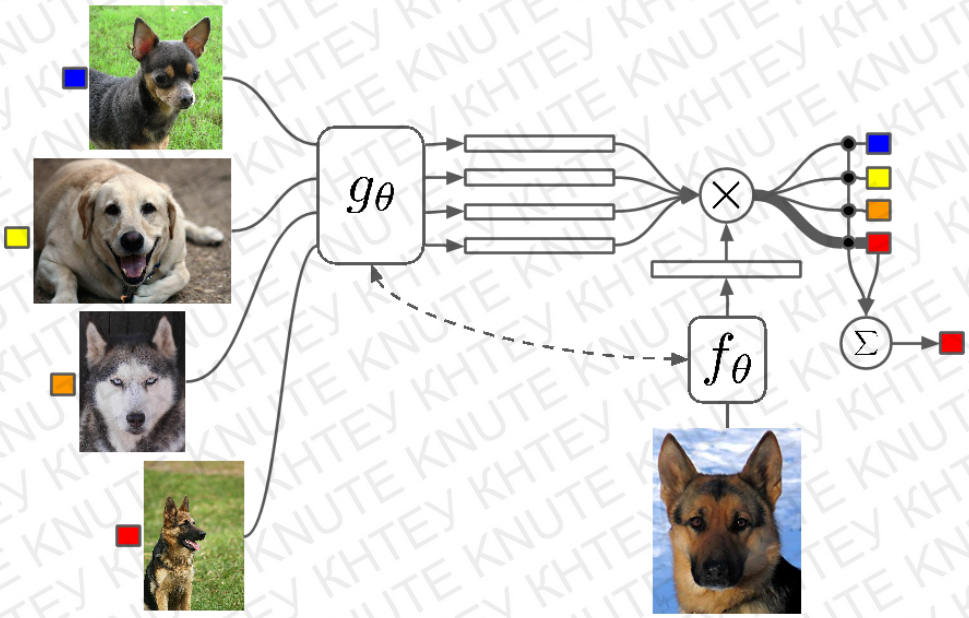


Рис. 2.5. Однакові цифри написані по-різному



2.6. Приклад роботи алгоритму One-Shot Learning

Більшість методів класифікації зображень вимагають багато вхідних даних для отримання значущих результатів. One-Shot Learning намагається імітувати навчання людини з точки зору класифікації. Люди можуть класифікувати дуже мало прикладів зображень і здатні добре виконувати лише

один приклад. Метод One-Shot Learning бере 1 або декілька зображень кожного класу і намагається класифікувати тестовий набір на основі невеликої інформації, яку він отримав. Щоб спробувати протестувати наші алгоритми навчання на One-Shot Learning, були створені набори даних зменшеного розміру.

2.2 K- Nearest Neighbors (k-NN)

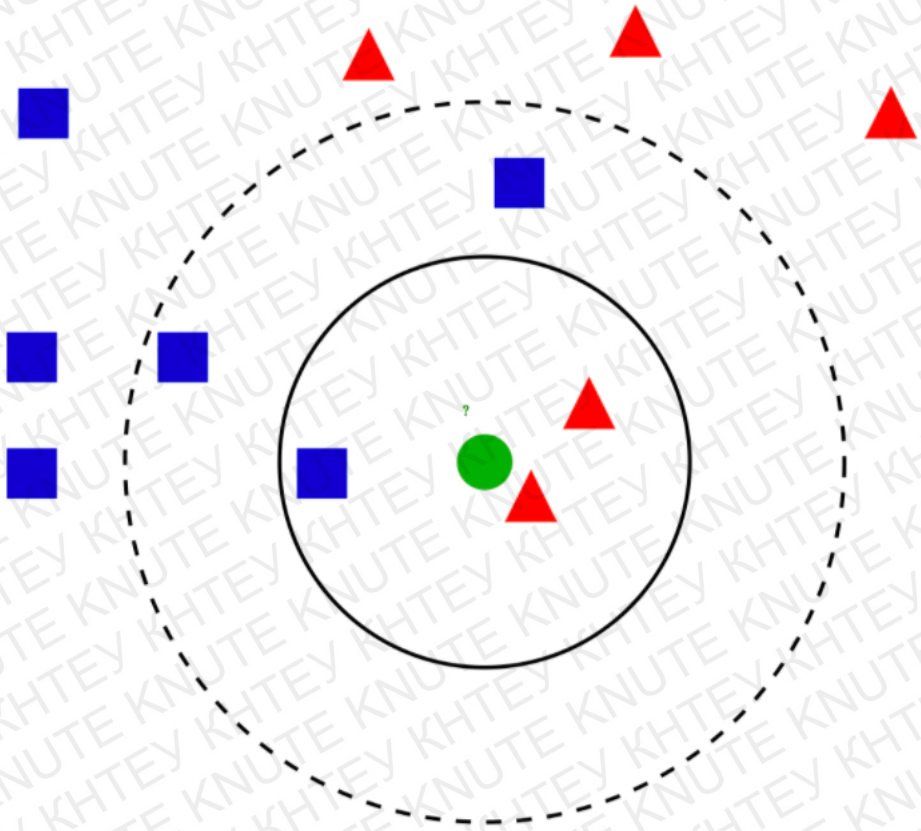


Рис. 2.7. K- Nearest Neighbors (k-NN)

Алгоритм класифікації k-NN (k-Найближчі сусіди) - це, мабуть, найпростіший алгоритм ML, але ефективний, щоб розпочати вашу подорож до ML. Я збираюся розробити алгоритм k-NN для ідентифікації цифр рукописного вводу у текстовому форматі. Приклад цифр рукописного вводу, відформатованих як текст.

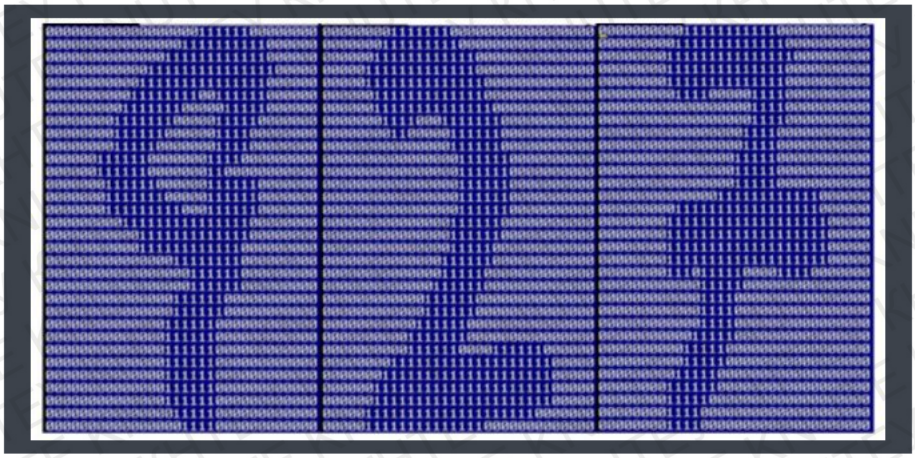


Рис. 2.8. Приклад цифр рукописного вводу, відформатованих як текст

```
For every vector/sample in our training dataset:
calculate the distance between inX (new sample vec) and the current vector/sampl
sort the distances in increasing order
take k items with lowest distances to inX
find the majority class among these items
return the majority class as our prediction for the class of inX
```

Рис. 2.8. Псевдокод алгоритму.

KNN - це непараметричний метод або класифікатор, що використовується для класифікації, а також для задач регресії. Це алгоритм класифікації лінивого або пізнього навчання, де всі обчислення виведені до останнього етапу класифікації, а також це алгоритми навчання на основі екземплярів, де наближення відбувається локально. Будучи найпростішим та найпростішим у реалізації, раніше не існує чіткого етапу навчання, а алгоритм не виконує жодного узагальнення даних навчання.

2.2.1 Коли використовувати?

Прямим рішенням було б, коли це нелінійні межі прийняття рішень між класами або коли обсяг даних досить великий. Вхідні ознаки можуть мати як якісний, так і кількісний характер. Тоді як вихідні ознаки можуть бути категоріальними значеннями, які є типовими класами, що спостерігаються в даних.

KNN пояснює категоричне значення, використовуючи більшість голосів найближчих сусідів, де значення K може відрізнятися, тому при зміні значення K значення голосів також може змінюватися.

2.2.2 Виберіть параметр K на основі даних.

Потрібна метрика відстані для визначення близькості між будь-якими двома точками даних. Цю відстань можна обчислити з евклідової відстані, відстані Махаланобіса, відстані Хеммінга тощо.

2.2.3 Алгоритм

1. Обчисліть метрику відстані між точкою тестових даних та усіма позначеними точками даних.

2. Впорядкуйте позначені точки даних у порядку збільшення метрики відстані.

3. Виберіть найвищі точки, позначені K, і подивіться на мітки класів.

4. Шукайте мітки класів, які мають більшість цих точок даних із мітками K, і призначте їх для тестування точок даних.

2.2.4 Що слід врахувати?

1. Вибір параметра – найкращий вибір K залежить від даних. Більше значення K зменшує вплив шуму на класифікацію, але чітко визначає межі рішення між безкласовими. На меншу величину K, як правило, впливає шум з чітким поділом між класами.

2. Наявність шуму

3. Вибір функцій та масштабування – важливо зменшити непотрібні функції. Коли кількість функцій занадто велика і існує підозра, що вона надмірно зайва, потрібно буде вилучити функції. Якщо ретельно підібрати ознаки, тоді очікується, що класифікація буде кращою.

4. Проблеми розмірності

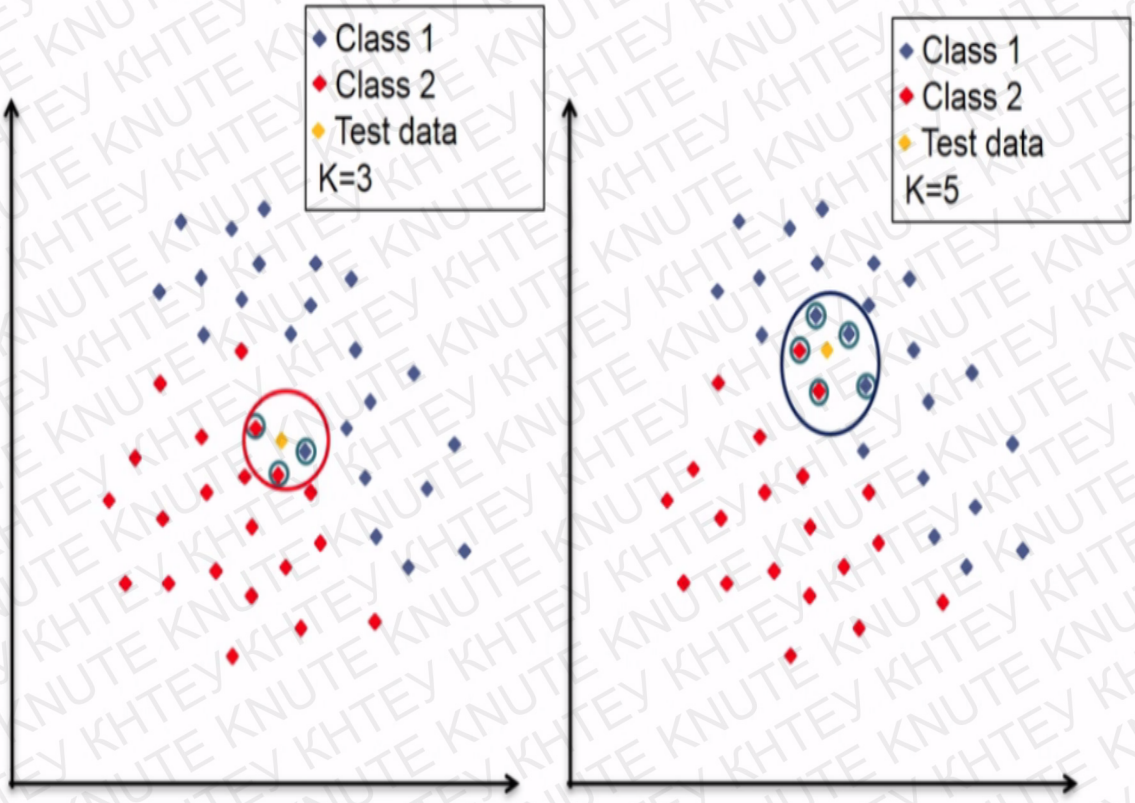


Рис. 2.9 Проблеми розмірності для алгоритму k-NN

Щоб краще зрозуміти, давайте подивимось на різні значення для K. Скажімо, у випадку 1, значення для K дорівнює 3. Тоді клас для точки тестових даних мав би червоний колір серед класів для червоного та синього. Для K = 5 у випадку 2, тоді передбачуваний клас буде мати синій колір з алгоритму KNN. Отже, для зміни значення K, вихід для точки тестових даних також може змінюватися. Отже, необхідно правильно підібрати значення K. Велике значення K може зменшити загальний шум, але немає жодних гарантій.

2.2.5 Різні функції відстані, що використовуються в KNN

1. Евклідова функція
2. Манхеттенська функція
3. Мінковський
4. Відстань Хеммінга
5. Відстань Махаланобіса

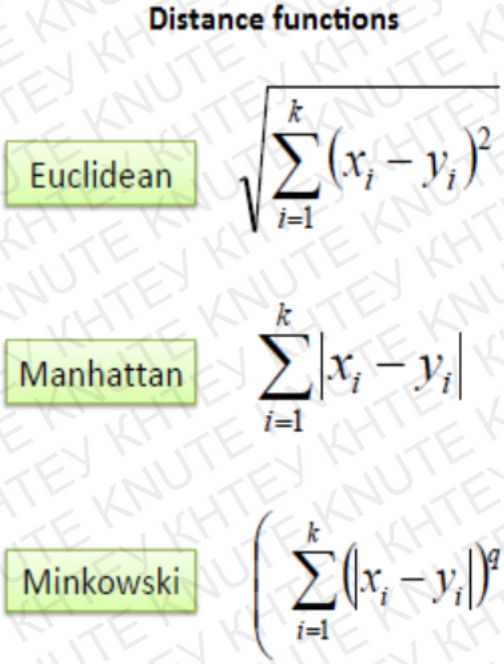


Рис. 2.10. Функції відстані, що використовуються в KNN

2.2.6 Команда для виконання KNN

Ми можемо змінювати параметри класифікатора та спостерігати за зміною вилучення класифікатора та проводити порівняння того, наскільки якісно та ефективно працюють з різними параметрами та гіперпараметрами.

клас sklearn.neighbors.KNeighborsClassifier (n_neighbors = 5, ваги = 'рівномірний', алгоритм = 'авто', leaf_size = 30, p = 2, metric = 'minkowski', metric_params = None, n_jobs = 1, ** kwargs)

2.2.7 Висновок

Алгоритм k-Найближчі сусіди - це простий та ефективний спосіб класифікації даних. Але алгоритм повинен містити повний набір даних; для великих наборів даних це передбачає великий обсяг пам'яті. Крім того, вам потрібно розрахувати вимірювання відстані для кожного фрагмента даних у базі даних, і це може бути громіздким.

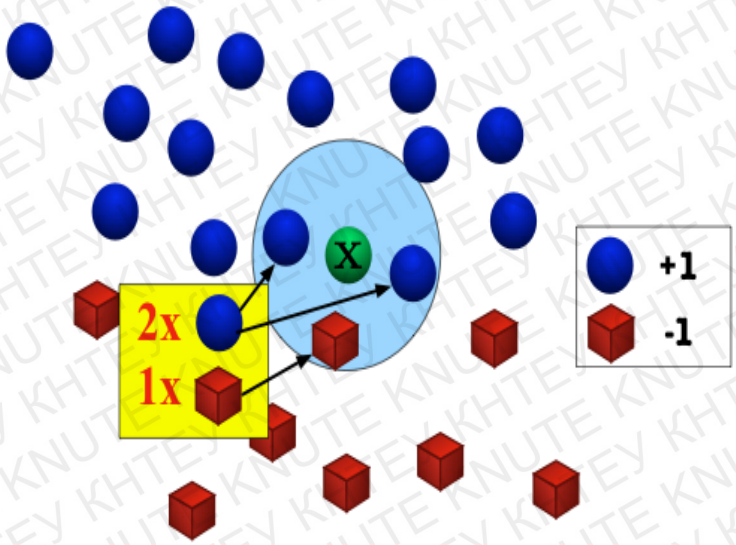


Рис. 2.11. Графічне представлення алгоритму k-NN

Алгоритм навчання k-NN базується на класифікації евклідової відстані між тестовими зображеннями і тренувальними. Кожне тестування зображень порівнюється з кожним тренувальним зображенням і беруться labels k найбільш близьких зображень, причому найчастіше використовується як label для навчання. Значення k вказує, скільки з найближчих зображень розглядаються для створення результату. Коли k дорівнює 1, для тестування зображення використовується label найближчого тренувального зображення. k, як правило, вибирається як непарне число, щоб уникнути зв'язку. Метод k-NN навчання не вимагає часу навчання, однак, кожне зображення тестування має порівнюватися з кожним навчальним зображенням. Це призводить до великої кількості часу для ідентифікації зображень, коли набір даних великий. Тому для реалізації нашої задачі цей алгоритм не дуже відповідає нашим вимогам, це може спричинити як погану продуктивність, так і погані результати самої моделі.

2.3 Support Vector Machine (SVM)

Support Vector Machine (SVM) - це алгоритм автоматизованого навчання, який можна використовувати як для класифікації, так і для регресії. Проте він в основному використовується в проблемах класифікації. У цьому алгоритмі ми будуємо кожен елемент даних як точку в n-мірному просторі (де n - кількість ознак), причому значення кожної ознаки є значенням певної координати. Потім ми виконуємо класифікацію, знаходячи гіперплощину, яка дуже добре розрізняє два класи.

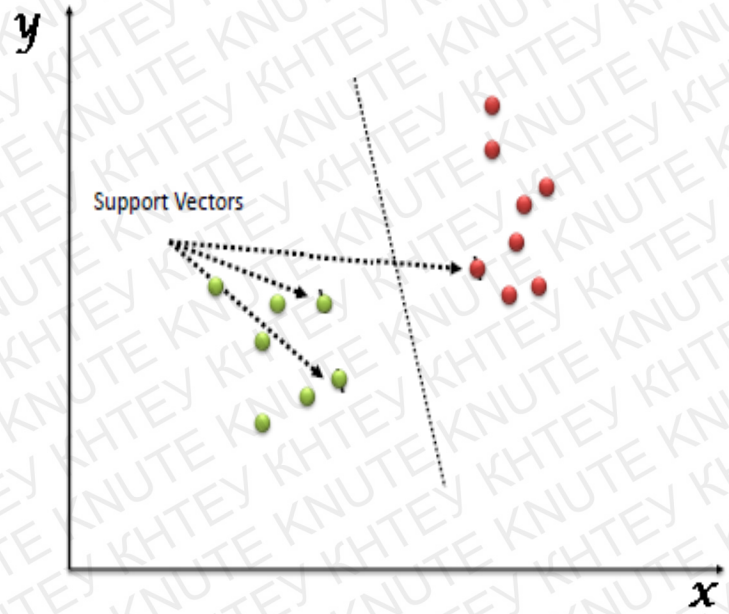


Рис. 2.12. Графічне представлення алгоритму SVM

Support Vectors - це просто координати індивідуального спостереження. Support Vector Machine - це межа, яка найкраще розділяє два класи (гіпер-площину / лінію). У SVM дуже легко мати лінійну гіперплощину між цими двома класами. Але, ще одне гостре питання, яке виникає, чи потрібно нам додавати цю функцію вручну, щоб мати гіпер-площину. Ні, SVM має техніку, яка називається трюк ядра. Це функції, які приймають низький розмірний вхідний простір і перетворюють його в більш високомірний простір, тобто перетворює нероздільну задачу в роздільну задачу, ці функції називаються ядрами. В основному це корисно в нелінійній задачі розділення. Простіше кажучи, це робить деякі надзвичайно складні перетворення даних, а потім з'ясовує процес розділення даних на основі міток або виходів, які ви визначили. Тому в нашому випадку, краще не обирати даний алгоритм, для того, щоб уникнути надмірної установки і скоротити час навчання.

2.4 Convolution Neural networks (CNN)

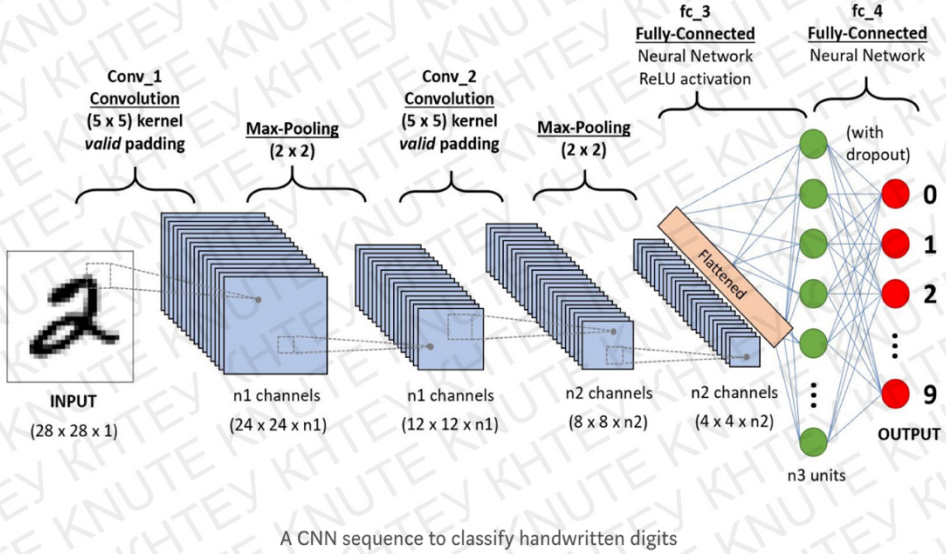


Рис. 2.13. Графічне представлення алгоритму k-NN

Згорткова нейронна мережа (CNN) - це тип штучної нейронної мережі, що використовується для розпізнавання та обробки зображень, спеціально розроблених для обробки піксельних даних. CNN використовує глибоке навчання для виконання як генеративних, так і описових завдань, часто використовуючи машинне бачення, яке включає в себе розпізнавання зображень і відео. Нейронна мережа - це система апаратного забезпечення та / або програмного забезпечення, сформованого після функціонування нейронів людського мозку. Традиційні нейронні мережі не ідеально підходять для обробки зображень. CNN мають власні «нейрони», подібні до таких, як фронтальна частка, що відповідає за обробку візуальних стимулів у людей та інших тварин. Шари нейронів розташовані таким чином, щоб охопити все поле зору, уникаючи проблему обробки фрагментів зображення традиційними нейронними мережами. CNN використовує систему, схожу на багатошаровий персептрон, розроблений для зниження вимог до обробки.

Шари CNN складаються з вхідного шару, вихідного шару і прихованого шару, який включає в себе безліч згорткових шарів, шарів об'єднання, повністю з'єднаних шарів і нормалізаційних шарів. Усунення обмежень і підвищення ефективності обробки зображень призводить до набагато ефективнішої, простішої для тренування обмеженої обробки зображень.

2.5 Висновки та вибір алгоритму до розділу 2

У цій роботі досліджено кілька методів, включаючи Perceptron, k-NN, SVM і CNN. K-NN дає хороші результати для малих наборів даних, але, оскільки набір даних стає більшим і більшим, потрібно більше часу, k-NN дає набагато кращі результати в порівнянні з одношаровим Perceptron. SVM надав багатообіцяючі результати за короткий проміжок часу, використовуючи всі три методи, однак лінійний SVM запропоновано використовувати для набору даних MNIST замість нелінійних SVM, оскільки набір MNIST не був знайдений як нелінійний або нероздільний. CNN має найвищий час виконання, він перевершує всі інші методи функціональністю, швидкістю, та цей алгоритм дав один з найкращих результатів, тому я і обрав саме цей алгоритм.

РОЗДІЛ 3
ОПИС ПРАКТИЧНОЇ ЧАСТИНИ

Моя робота - це 5-ти шарова послідовна згорткова нейронна мережа (Convolutional Neural Network) для розпізнавання цифр, що навчається на наборі даних MNIST. Для побудови я обрав Keras API (Tensorflow. По-перше, я підготував дані (рукописні цифри зображень), для того, щоб з ними було зручно працювати, проводити деякі операції для спостережень і статистики, а потім вже зосереджувався на моделювання та оцінці обраного алгоритму CNN. Я досяг 98,528% точності, використовуючи алгоритм CNN. В моїй роботі можна виділити три основні частини:

1. Підготовка даних
2. Моделювання та оцінювання CNN
3. Результати прогнозування і подання самих результатів

3.1. Підготовка даних

3.1.1. Завантаження даних

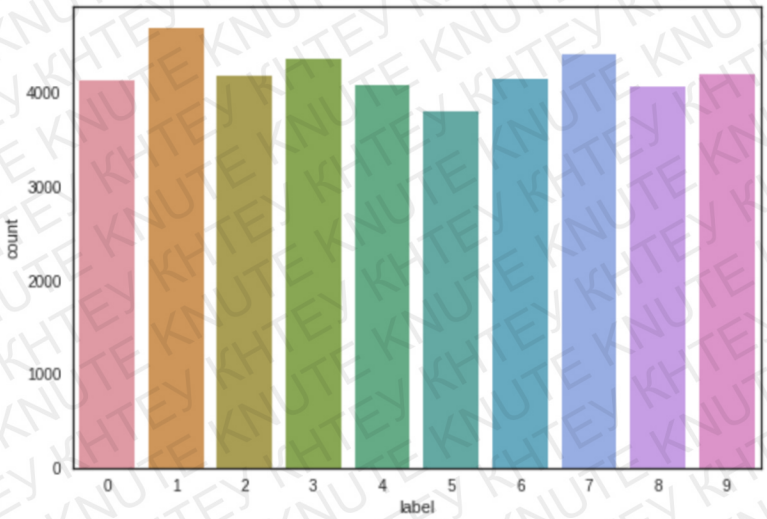


Рис. 3.1. Завантаження даних та візуальне відображення

За нашими підрахунками, ми маємо такі відповідні значення для наших 10 цифр.

3.1.2. Перевірка нульових та відсутніх значень

Я перевірив наявність пошкоджених зображень (відсутні значення всередині). Немає відсутніх значень у наборі натренованих даних і тестів. Тому я продовжив і перейшов до нормалізації.

3.1.3. Нормалізація

Для зниження ефекту відмінностей освітлення я виконав нормалізацію градацій сірого. Крім того, CNN швидше перетворюється на [0..1] дані, ніж на [0..255].

3.1.4. Зміна початкової форми

Натреновані і тестові зображення (28px x 28px) закріплені в Pandas. Dataframe представлені як 1D вектори 784 значень. Змінюємо всі дані на матриці розміром 28x28x1. Keras вимагає додаткового виміру в кінці, який відповідає каналам. Зображення MNIST мають сірий масштаб, тому він використовує тільки один канал. Для RGB-зображень є 3 канали, тому ми переробили 784px вектори на 3D матриці 28x28x3.

3.1.5. Кодування міток (Labels).

Мітки складаються з 10 цифр від 0 до 9. Необхідно кодувати мітки до одного гарячого вектора (наприклад: 2 -> [0,0,1,0,0,0,0,0,0,0]).

3.1.6. Розділення натренованого набору та набору перевірки

Я вирішив розділити натренований набір на дві частини: невелика частка (10%) стала валідаційною системою, яка оцінює саму модель, а решта (90%) використовується для навчання моделі. Оскільки у нас є 42 000 навчальних зображень збалансованих лейблів (див. 1.1 Завантаження даних), випадкове розбиття натренованого набору не призводить до надмірного представлення позначок у наборі перевірки. Треба бути обережним з деяким незбалансованим набором даних, простий випадковий поділ може спричинити неточну оцінку під час перевірки. Щоб уникнути цього, також можна скористатися опцією stratify = True у функції train_test_split (тільки для > = 0.17 sklearn versions). Ми можемо отримати кращий сенс для одного з цих прикладів, візуалізуючи зображення і подивившись на лейбл.

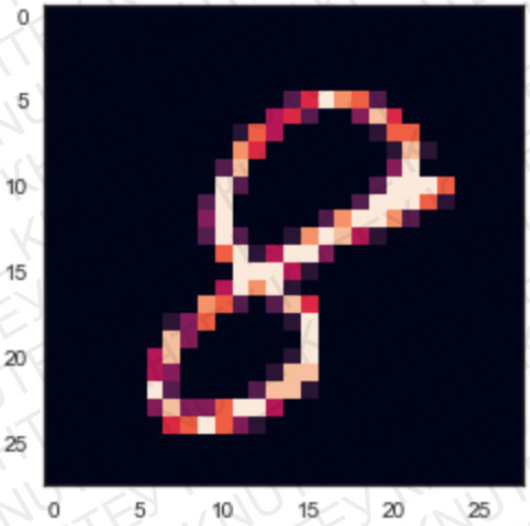


Рис. 3.2. Початкова форма натренованих зображень

3.2. Моделювання та оцінювання CNN

3.2.1. Визначення моделі

Я використовував Keras Sequential API, де я додавав шари один за одним, починаючи з вводу даних.

Першим є згортковий (Conv2D) шар. Це схоже на набір фільтрів, які можна вивчати. Я обрав для встановлення 32 фільтри для двох шарів Conv2D і 64 фільтри для двох останніх. Кожен фільтр перетворює частину зображення (визначається розміром ядра) за допомогою фільтра ядра. Матриця фільтра ядра застосовується до всього зображення. Фільтри можна розглядати як перетворення зображення. CNN може ізолювати особливості, які є корисними всюди від цих трансформованих зображень (feature maps).

Другим важливим шаром в CNN є шар об'єднання (MaxPool2D). Цей шар просто виступає як фільтр зниження. Він дивиться на 2 сусідніх пікселя і обирає максимальне значення. Вони використовуються для зниження обчислювальних витрат і до певної міри також зменшують перенасичення. Ми повинні обирати розмір об'єднання (тобто розмір площі, що об'єднується

кожного разу), більший розмір об'єднання є високим, більш важливим є зменшення обсягу.

Об'єднуючи згорткові та об'єднувальні шари, CNN здатні поєднувати локальні функції та вивчати більш глобальні особливості зображення.

Dropout є методом регуляризації, де частка вузлів у шарі випадково ігнорується (встановлюючи їхні рівні до нуля) для кожної навчальної вибірки. Це випадковим чином ломає мережу і змушує мережу вивчати функції розподіленим способом. Ця методика також покращує узагальнення і зменшує перенавчання. 'relu' є випрямлячем (функція активації max (0, x). Функція активації випрямляча використовується для додавання нелінійності в мережу.

Пласт Flatten використовується для перетворення остаточної карти властивостей в єдиний одиничний 1D вектор. Цей етап вирівнювання необхідний для того, щоб можна було використовувати повністю пов'язані шари після деяких згорткових / максимальних шарів. Він поєднує в собі всі знайдені локальні особливості попередніх згорткових шарів. Зрештою, я використав особливості в двох повністю пов'язаних (щільних) шарах, що є простими штучними класифікаторами нейронних мереж (ANN). В останньому шарі (Dense (10, activation = "softmax")) чисті виходи розподілу ймовірності кожного класу.

3.2.2. Встановлення оптимізатора

Після того, як наші шари додані до моделі, нам потрібно налаштувати функцію оцінки, функцію втрат і алгоритм оптимізації.

Ми визначаємо функцію втрат, щоб виміряти, наскільки погано наша модель працює на зображеннях з відомими мітками. Це рівень помилок між закладеними мітками і прогнозованими. Ми використовуємо специфічну форму для категоріальних класифікацій (> 2 класу), так званих "categorical_crossentropy".

Найважливішою функцією є оптимізатор. Ця функція ітеративно поліпшить параметри (фільтрує значення ядра, ваги і зміщення нейронів ...), щоб мінімізувати втрати.

Я обрав RMSprop (із значеннями за замовчуванням), це дуже ефективний оптимізатор. Оновлення RMSProp дуже простим способом коригує метод Адаграда, намагаючись зменшити його агресивну, монотонно знижуючу швидкість навчання. Ми також могли б використати оптимізатор Stochastic Gradient Descent ('sgd'), але він повільніше, ніж RMSprop.

Метрична функція "accuracy" використовується для оцінки продуктивності нашої моделі. Ця метрична функція аналогічна функції втрат, за винятком того, що результати оцінки метрики не використовуються при навчанні моделі (тільки для оцінки).

Для того, щоб оптимізатор сходився швидше і ближче до глобального мінімуму функції втрат, я використовував метод відпалу швидкості навчання (LR). LR - це крок, за допомогою якого оптимізатор проходить через “loss landscape”. Чим вище LR, тим більше кроків і тим швидше відбувається зближення. Однак вибірка дуже погана з високим LR і оптимізатор, ймовірно, може потрапити в локальні мінімуми. Краще мати зниження швидкості навчання під час навчання для ефективного досягнення глобального мінімуму функції втрати. Щоб зберегти перевагу швидкого часу обчислення з високим LR, я зменшив LR динамічно з кожним кроком X (epochs) залежно від необхідності (коли точність не покращується).

З функцією ReduceLROnPlateau з Keras.callbacks, я вирішив зменшити LR наполовину, якщо точність не покращилася після 3 epochs.

3.2.3. Збільшення даних

Щоб уникнути проблеми з перенасиченням, ми повинні штучно розширити наш набір рукописних цифр. Ми можемо зробити наш існуючий

набір даних ще більшим. Ідея полягає в тому, щоб змінити навчальні дані з невеликими перетвореннями, щоб відтворити зміни, що відбуваються, коли хтось пише цифру.

Наприклад, число не центрується, масштаб не є однаковим (деякі, хто пише з великими / малими числами) Зображення повертається ...

Підходи, які змінюють навчальні дані способами, які змінюють подання масиву, зберігаючи те ж саме, відомі як методи збільшення даних. Деякі популярні доповнення людей використовують сірі тони, горизонтальні сальто, вертикальні сальто, випадкові культури, спалахи кольору, переклади, обертання та багато іншого.

Застосовуючи лише кілька цих перетворень до наших навчальних даних, ми можемо легко подвоїти або потроїти кількість навчальних прикладів і створити дуже надійну модель. Покращення важливо:

Без збільшення даних я отримав точність 97,114%

При збільшенні даних і досягнуто 98,528% точності

Для збільшення даних я обрав:

Довільно повертати деякі навчальні зображення на 10 градусів

Довільно збільшити на 10% деякі навчальні зображення

Випадково переміщувати зображення по горизонталі на 10% від ширини

Випадково переміщувати зображення вертикально на 10% від висоти

Я не застосовував вертикальний або горизонтальний flip, оскільки він міг призвести до неправильної класифікації симетричних чисел, таких як 6 і 9.

Після того, як наша модель буде готова, ми вміщуємо навчальний набір даних.

3.3. Оцінка моделі

3.3.1. Криві навчання та перевірки

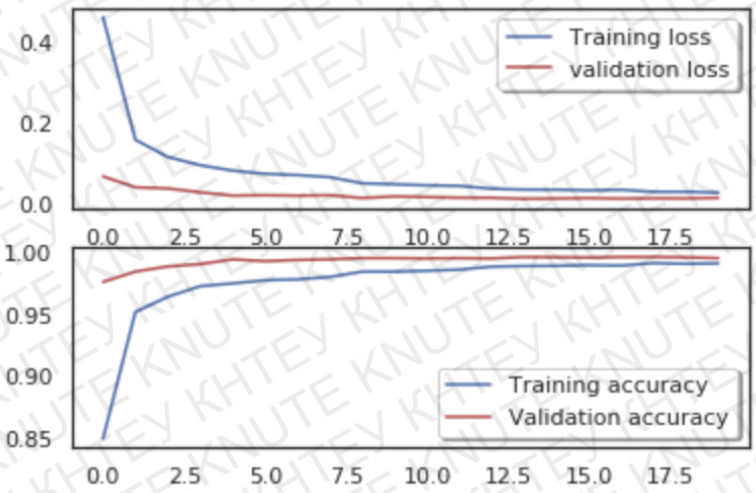


Рис. 3.3. Криві навчання та перевірки

Модель досягає майже 99% (98,528 %) точності на наборі валідації після 2 epochs. Точність перевірки більша, ніж точність підготовки, майже на час навчання. Це означає, що наша модель не переповнює тренувальний набір. Тому можна зробити лише один висновок, що наша модель дуже добре підготовлена.

3.3.2. Матриця типових помилок

Матриця типових помилок може бути дуже корисною, щоб побачити мої недоліки в моделі. Я побудував матрицю плутанини результатів перевірки.

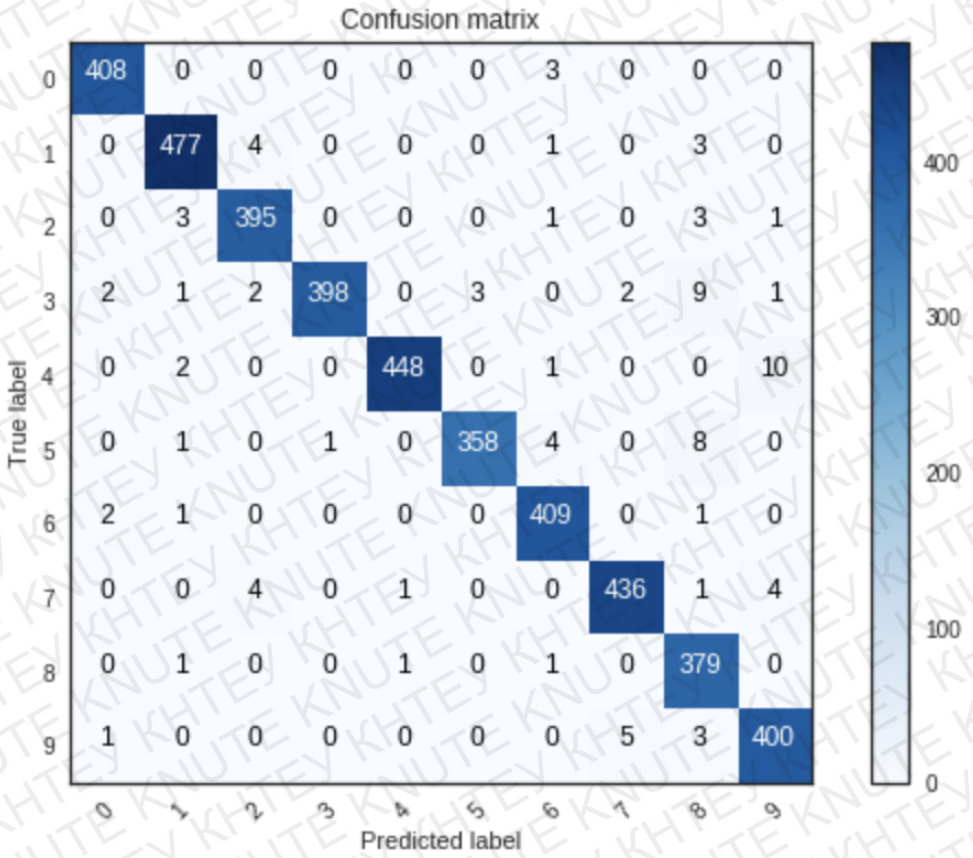


Рис. 3.4. Матриця типових помилок

Тут ми бачимо, що CNN добре працює на всіх цифрах з невеликою кількістю помилок з урахуванням розміру набору (4 200 зображень). Тим не менш, здається, що мій CNN має деякі невеликі неприємності з цифрою 4, вони неправильно класифіковані як 9. Це можна пояснити тим, що дуже важко впізнати різницю між 4 і 9, коли криві гладкі. Тепер я хочу дослідити помилки, які дає моя модель. Я хочу побачити найважливіші помилки. Для цього потрібно отримати різницю між ймовірностями реального value і прогнозованого в результатах.

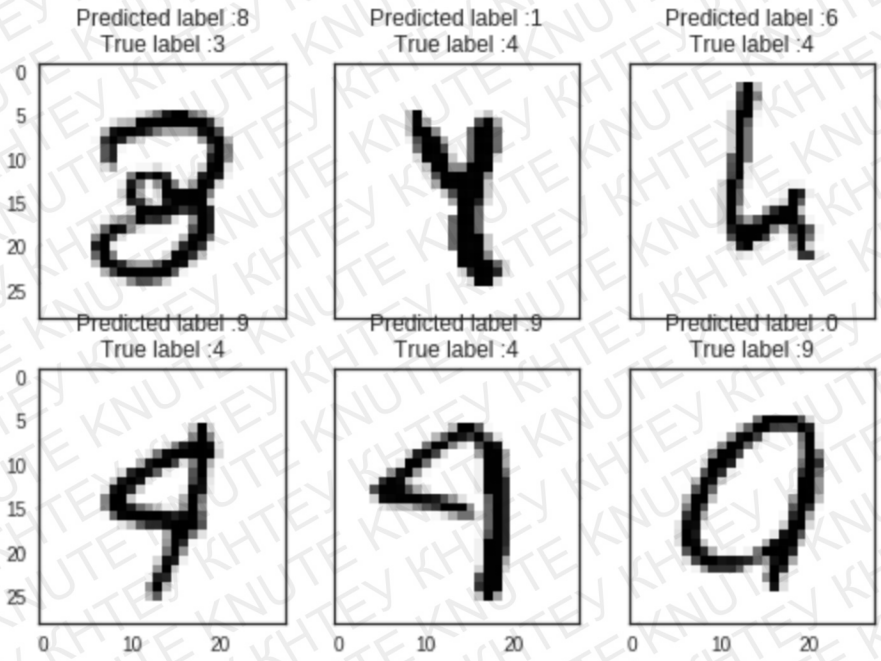


Рис. 3.5. Типові помилки

Серед найбільш важливих помилок є також інтриги. Деякі з цих помилок також можуть бути зроблені людьми, особливо з 9, яка дуже близька до 4.

3.4 Висновки до розділу 3

Для задачі розпізнання рукописних цифр використовуються алгоритми машинного навчання, такі як KNN, SVM, нейронні мережі разом з різними параметрами та векторами масштабування функцій. Ця робота допомогла мені побачити різницю між багатьма класифікаторами з точки зору найважливішої особливості, такої як точність та синхронізація. Точність може змінюватися, оскільки вона залежить від розбиття даних, підготовки та тестування, і це може бути покращено, якщо надається додаткова кількість даних. Завжди є шанс підвищити точність, якщо розмір даних збільшиться. Кожен класифікатор має власну точність і час споживання. Ми можемо також додати той факт, що якщо потужність процесора змінюється на GPU, класифікатор може виконувати з більшою точністю і з меншим часом і можна спостерігати кращі результати. Продуктивність класифікатора може бути виміряна з точки

зору: здатності правильно ідентифікувати стан (чутливість), частки справжніх результатів (точність) і кількості позитивних результатів від процедури класифікації (позитивні прогнози). У цьому, я побачив коротке порівняння з класифікаторами машинного навчання та глибокого навчання. Дотепер алгоритми глибокого навчання краще виконувалися при застосуванні рукописного розпізнавання цифр. Майбутні дослідження можуть розглянути можливість використання архітектури згорткової мережі, яка дала найкращий результат на базі даних MNIST. Такі додаткові системи можуть бути розроблені для розпізнавання рукописних символів, розпізнавання об'єктів, сегментації зображень, розпізнавання рукописного тексту, розпізнавання мови тексту, а також майбутні дослідження щодо апаратної реалізації в системі розпізнавання розрядів, в режимі онлайн, з більшою продуктивністю та ефективністю, з результатами тестування в реальному часі сценарії.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

Для вирішення поставленої в роботі мети було проведено аналіз мов програмування для створення моєї моделі. В процесі виконання випускної кваліфікаційної роботи було зроблено наступне:

1. Провівши аналіз технологій було визначено технології, за допомогою яких найзручніше реалізувати поставлену задачу.
2. Проведений аналіз алгоритмів для вирішення поставленої задачі.

В другому розділі був проведений аналіз алгоритмів для вирішення поставленої задачі та аналізувалися технології за допомогою чого можна вирішити задачу.

В третьому розділі здійснювалась розробка алгоритму. Здійснювалась розробка алгоритмів роботи програми, що дозволило оптимізувати процес відповідно до вимог випускної кваліфікаційної роботи. На останньому етапі здійснювалась оцінка моделі.

За результатами розробки отримано модель, яку можна використовувати для розпізнавання рукописних цифр.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Introduction to Machine Learning with Python: A Guide for Data Scientists, by Sarah Guido
2. Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition, by Aurélien Géron
3. Applied Deep Learning, A Case-Based Approach to Understanding Deep Neural Networks, by Umberto Michelucci
4. Data Science from Scratch, by Joel Grus
5. Fluent Python, by Luciano Ramalho
6. Інструкція для початківців Tensorflow. Режим доступу: https://www.tensorflow.org/tutorials/
7. Машинне навчання. Режим доступу: https://www.coursera.org/learn/machine-learning

ДОДАТКИ

ДОДАТОК А

Алгоритм:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import seaborn as sns
%matplotlib inline
np.random.seed(2)

from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import itertools

from keras.utils.np_utils import to_categorical # convert to one-hot-encoding

from keras.models import Sequential
from keras.layers import Dense, Dropout, Flatten, Conv2D, MaxPool2D
from keras.optimizers import RMSprop
from keras.preprocessing.image import ImageDataGenerator
from keras.callbacks import ReduceLROnPlateau


sns.set(style='white', context='notebook', palette='deep')

# Load the data
trainSet = pd.read_csv("train.csv")
testSet = pd.read_csv("test.csv")
```

Продовження дод.А

```
Y_trainSet = trainSet["label"]

# Drop 'label' column
X_trainSet = trainSet.drop(labels = ["label"],axis = 1)

# free some space
del trainSet

valueCounts = sns.countplot(Y_train)

Y_trainSet.value_counts()
# print(Y_train)

# Check the data
X_trainSet.isnull().any().describe()

testSet.isnull().any().describe()

# Normalize the data
X_trainSet = X_trainSet / 255.0
testSet = testSet / 255.0

# Reshape image in 3 dimensions (height = 28px, width = 28px , canal = 1)
X_trainSet = X_trainSet.values.reshape(-1,28,28,1)
testSet = testSet.values.reshape(-1,28,28,1)

# Encode labels to one hot vectors (ex : 2 -> [0,0,1,0,0,0,0,0,0,0])
Y_trainSet = to_categorical(Y_trainSet, num_classes = 10)

# Set the random seed
random_seed = 2

# Split the train and the validation set for the fitting
X_trainSet, X_val, Y_trainSet, Y_val = train_test_split(X_trainSet, Y_trainSet, test_size =
0.1, random_state=random_seed)
```

Продовження дод.А

```
valueCounts = plt.imshow(X_trainSet[0][:,:,0])

# Set the CNN model
# my CNN architechture is In -> [[Conv2D->relu]*2 -> MaxPool2D -> Dropout]*2 -> Flatten -> Dense -> Dropout -> Out

myModel = Sequential()

myModel.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',
                   activation ='relu', input_shape = (28,28,1)))
myModel.add(Conv2D(filters = 32, kernel_size = (5,5),padding = 'Same',
                   activation ='relu'))
myModel.add(MaxPool2D(pool_size=(2,2)))
myModel.add(Dropout(0.25))


myModel.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',
                   activation ='relu'))
myModel.add(Conv2D(filters = 64, kernel_size = (3,3),padding = 'Same',
                   activation ='relu'))
myModel.add(MaxPool2D(pool_size=(2,2), strides=(2,2)))
myModel.add(Dropout(0.25))


myModel.add(Flatten())
myModel.add(Dense(256, activation = "relu"))
myModel.add(Dropout(0.5))
myModel.add(Dense(10, activation = "softmax"))

# Define the optimizer
myOptimizer = RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0)

# Compile the model
myModel.compile(optimizer = myOptimizer , loss = "categorical_crossentropy", metrics=["accuracy"])
```

```
# Set a learning rate annealer
learningRateReduction = ReduceLROnPlateau(monitor='val_acc',
                                          patience=3,
                                          verbose=1,
                                          factor=0.5,
                                          min_lr=0.00001)
epochs = 1 # Turn epochs to 30 to get 0.9967 accuracy
batch_size = 86

# Without data augmentation i obtained an accuracy of 0.98114
history = myModel.fit(X_trainSet, Y_trainSet, batch_size = batch_size, epochs = epochs,
      validation_data = (X_val, Y_val), verbose = 2)

# With data augmentation to prevent overfitting (accuracy 0.99286)

dataGeneration = ImageDataGenerator(
      featurewise_center=False,  # set input mean to 0 over the dataset
      samplewise_center=False,  # set each sample mean to 0
      featurewise_std_normalization=False,  # divide inputs by std of the dataset
      samplewise_std_normalization=False,  # divide each input by its std
      zca_whitening=False,  # apply ZCA whitening
      rotation_range=10,  # randomly rotate images in the range (degrees, 0 to 180)
      zoom_range = 0.1, # Randomly zoom image
      width_shift_range=0.1,  # randomly shift images horizontally (fraction of total width)
      height_shift_range=0.1,  # randomly shift images vertically (fraction of total height)
      horizontal_flip=False,  # randomly flip images
      vertical_flip=False)  # randomly flip images


dataGeneration.fit(X_trainSet)

# Fit the model
history = myModel.fit_generator(dataGeneration.flow(X_trainSet,Y_trainSet,
batch_size=batch_size),
```

```
                    epochs = epochs, validation_data = (X_val,Y_val),
                    verbose = 2, steps_per_epoch=X_trainSet.shape[0] // batch_size
                    , callbacks=[learningRateReduction])

# Plot the loss and accuracy curves for training and validation
fig, ax = plt.subplots(2,1)
ax[0].plot(history.history['loss'], color='b', label="Training loss")
ax[0].plot(history.history['val_loss'], color='r', label="validation loss",axes =ax[0])
legend = ax[0].legend(loc='best', shadow=True)

ax[1].plot(history.history['acc'], color='b', label="Training accuracy")
ax[1].plot(history.history['val_acc'], color='r',label="Validation accuracy")
legend = ax[1].legend(loc='best', shadow=True)

# Look at confusion matrix

def plot_confusion_matrix(cm, classes,
                    normalize=False,
                    title='Confusion matrix',
                    cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """
    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]

    thresh = cm.max() / 2.
```

Продовження дод.А

```
        for i, j in itertools.product(range(cm.shape[0]), range(cm.shape[1])):
            plt.text(j, i, cm[i, j],
                     horizontalalignment="center",
                     color="white" if cm[i, j] > thresh else "black")

        plt.tight_layout()
        plt.ylabel('True label')
        plt.xlabel('Predicted label')

    # Predict the values from the validation dataset
    Y_pred = myModel.predict(X_val)
    # Convert predictions classes to one hot vectors
    Y_pred_classes = np.argmax(Y_pred,axis = 1)
    # Convert validation observations to one hot vectors
    Y_true = np.argmax(Y_val,axis = 1)
    # compute the confusion matrix
    confusion_mtx = confusion_matrix(Y_true, Y_pred_classes)
    # plot the confusion matrix
    plot_confusion_matrix(confusion_mtx, classes = range(10))

    # Display some error results

    # Errors are difference between predicted labels and true labels
    errors = (Y_pred_classes - Y_true != 0)

    Y_pred_classes_errors = Y_pred_classes[errors]
    Y_pred_errors = Y_pred[errors]
    Y_true_errors = Y_true[errors]
    X_val_errors = X_val[errors]

    def display_errors(errors_index,img_errors,pred_errors, obs_errors):
        """ This function shows 6 images with their predicted and real labels"""
        n = 0
        nrows = 2
        ncols = 3
```

Продовження дод.А

```
        fig, ax = plt.subplots(nrows,ncols,sharex=True,sharey=True)
        for row in range(nrows):
            for col in range(ncols):
                error = errors_index[n]
                ax[row,col].imshow((img_errors[error]).reshape((28,28)))
                ax[row,col].set_title("Predicted label :{}\nTrue label :{}".format(pred_errors[error],obs_errors[error]))
                n += 1

    # Probabilities of the wrong predicted numbers
    Y_pred_errors_prob = np.max(Y_pred_errors,axis = 1)

    # Predicted probabilities of the true values in the error set
    true_prob_errors = np.diagonal(np.take(Y_pred_errors, Y_true_errors, axis=1))

    # Difference between the probability of the predicted label and the true label
    delta_pred_true_errors = Y_pred_errors_prob - true_prob_errors

    # Sorted list of the delta prob errors
    sorted_dela_errors = np.argsort(delta_pred_true_errors)

    # Top 6 errors
    most_important_errors = sorted_dela_errors[-6:]

    # Show the top 6 errors
    display_errors(most_important_errors, X_val_errors, Y_pred_classes_errors, Y_true_errors)

    # predict results
    results = myModel.predict(testSet)

    # select the indix with the maximum probability
    results = np.argmax(results,axis = 1)

    results = pd.Series(results,name="Label")
```

Продовження дод.А

```
submission = pd.concat([pd.Series(range(1,28001),name = "ImageId"),results],axis = 1)

submission.to_csv("cnn_mnist_datagen.csv",index=False)
```