Міністерство освіти і науки України
Прикарпатський національний університет імені Василя Стефаника
Кафедра комп'ютерної інженерії та електроніки

Славський Михайло Васильович
Slavskyi Mykhailo

УДК 004:681.5

Спеціальність 123 «Комп'ютерна інженерія»
(шифр та назва спеціальності)

Кваліфікаційна робота
на здобуття освітньо-кваліфікаційного рівня бакалавр
(бакалавр, спеціаліст, магістр)

Розпізнання рукописного тексту нейромережею
Recognition of handwritten text by a neural network

Науковий керівник:
к.т.н, доцент
Голота В.І.

Рецензент:
к.ф.-м.н., проф.каф.
фізики і хімії твердого тіла
Никируй Л. І.

Івано-Франківськ
2023

Форм.	Зона	Поз.	Позначення	Найменування	К-ть	Прим.
			6.050102. УДК 004:681.5	Пояснювальна записка	1	
			6.050102. УДК 004:681.5	Блок-реалізації	1	

					6.050102. УДК 004:681.5
Змн.	Арк.	№ докум.	Підп.	Дата	

Разробив	Славський М.В.			Специфікація	Літ.	Арк.	Аркушів
Перевірив	Голота В.І					2	
Н. Конт.							
Затвердив							

АНОТАЦІЯ

Дана робота містить 57 ст., 30 рис., 1 дод., 15 джерел.

Для розв`язання даної задачі використовувалися та роглядалися існуючі системи аналізу, сучасні інформаційні технології, опис всіх деталей та частин, а також розробка програмного коду для розпізнання рукописного тексту нейромережею.

Результати: науковий - запропонований даний комплекс рішення для перевірки рукописних текстів, швидкий та точний у виконанні поданими та запрограмованими командами.

Область застосування: в автоматичному заповненню полів з пошуком в базі даних. b

Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розробив		Славський М.В.			Анотація	*Літ.*	*Арк.*	*Аркушів*
Перевірив		Голота В.І.					3	1
Н. Контр.		.						
Затвердив								

ABSTRACT

This work contains 57 articles, 30 figures, 1 appendix, and 15 sources.

To solve this problem, existing analysis systems, modern information technologies, a description of all details and parts, as well as the development of a software code for recognizing handwritten text by a neural network were used and considered.

Results: scientific - the proposed set of solutions for checking handwritten texts is fast and accurate in execution by the given and programmed commands.

Field of application: in the automatic filling of fields with a search in the database.

Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		Славський М.В.			Блок-реалізації	*Літ.*	*Арк.* 5 / *Аркушів* 1
Перевірив		Голота В.І.					
Н. Контр.		.					
Затвердив							

Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Фізико-технічний факультет
Кафедра комп’ютерної інженерії та електроніки

Пояснювальна записка
до кваліфікаційної роботи на тему
Розпізнання рукописного тексту нейромережею

					6.050102. УДК 004:681.5		
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розробив		Славський М.В.			. Пояснювальна записка	*Літ.*	*Арк.* 6 / *Аркушів* 65
Перевірив		Голота В.І.					
Н. Контр.		.					
Затвердив							

ПЕРЕЛІК СКОРОЧЕНЬ

ШНМ – Штучна нейронна мережа

SNN – імітована нейронна мережа

CSV - Comma-Separated Values

БШП – багатошаровий перцептрон

ЗНМ - Згорткова нейронна мережа

РФБ - радіальних базових функцій

LSTM – довготривала короткочасна пам’ять

WSGI - Web Server Gateway Interface

ЗМІСТ1

ВСТУП

Інтелект людини відрізняє її від комп'ютера. Людина може виконувати різноманітні завдання, які машини ще не можуть виконати самостійно. Одним із таких завдань є розпізнавання рукописного тексту. Незважаючи на те, що протягом останніх кількох десятиліть розпізнавання тексту в рукописних документах вивчалося як одна з важливих областей досліджень різними дослідниками, і тому в минулому різними дослідниками було розроблено багато автоматичних рукописних систем. Однак питання алгоритму розпізнавання та його ефективності залишається відкритим. Через значну неузгодженість стилів рукописного тексту найсучасніші системи розпізнавання рукописного тексту часто не можуть забезпечити задовільну продуктивність на різних типах зразків рукописного тексту.

Доступні підходи до розпізнавання рукописного тексту зазвичай складаються з різних етапів, які в основному включають: попередню обробку, вилучення ознак, класифікацію, постобробку. Однак виділення ознак і дизайн класифікатора є двома основними кроками будь-якої системи розпізнавання. Багато дослідників створили різні типи систем розпізнавання рукописного тексту для різних мов, таких як англійська, китайська, арабська, японська бангла, тощо. проблеми розпізнавання цих сценаріїв не можна вважати повністю вирішеними.

Штучна нейронна мережа (ШНМ) розробляє ефективні та точні системи розпізнавання рукописного тексту. Одним із основних засобів, за допомогою яких комп'ютери володіють людськими здібностями, є використання ШНМ у проектуванні. Нейронні мережі працюють над проектуванням людського мозку, і вони особливо корисні для вирішення таких проблем, які неможливо описати як серію простих кроків, таких як розпізнавання шаблонів, класифікація об'єктів у різні класи, інтелектуальний аналіз даних і прогнозування серій.

Найпоширенішим використанням нейронних мереж є, мабуть, розпізнавання образів. Нейронна мережа представлена з іншим класом цільових векторів, а також з відповідними вхідними векторами (вектор, який містить інформацію про шаблон). Вхідні дані можуть варіюватися від простих

одновимірних (1-D) даних до багатовимірних даних. Після того, як ШНМ буде навчено за допомогою даних, його можна використовувати для визначення нових вхідних даних.

Основною метою цього дослідження є розробка ефективної системи розпізнавання рукописних символів і цифр.

Завдання:

- Аналіз існуючих системи розпізнання рукописного тексту нейромережею
- Вибір основних компонентів та редакторів для створення власної нейронної мережі
- Дослідити методи для обробки вхідних даних
- Розробка основного продукту
- Тестування даної програми

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ СИСТЕМ ДЛЯ РОЗПІЗНАННЯ РУКОПИСНОГО ТЕКСТУ НЕЙРОМЕРЕЖЕЮ

1.1. Огляд існуючих аналогів

Робота над штучними нейронними мережами, які зазвичай називають «нейронними мережами», з самого початку була мотивована визнанням того, що людський мозок обчислює зовсім інакше, ніж звичайний цифровий комп'ютер. Мозок — це дуже складний, нелінійний і паралельний комп'ютер (система обробки інформації).

Він має здатність організовувати свої структурні складові, відомі як нейрони, щоб виконувати певні обчислення (наприклад, розпізнавання образів, сприйняття та керування моторикою) у багато разів швидше, ніж найшвидший цифровий комп'ютер, який існує сьогодні. Розглянемо, наприклад, людський зір, який є завданням обробки інформації. Функція зорової системи полягає в тому, щоб забезпечити уявлення про навколишнє середовище і, що важливіше, надати інформацію, необхідну для взаємодії з навколишнім середовищем.

Точніше кажучи, мозок зазвичай виконує завдання перцептивного розпізнавання (наприклад, розпізнавання знайомого обличчя в незнайомій сцені) приблизно за 100–200 мс, тоді як завдання набагато меншої складності на потужному комп'ютері займають набагато більше часу.

Розвиток нейронних мереж сягає початку 1940-х років. Він пережив сплеск популярності наприкінці 1980-х років. Це стало результатом відкриття нових технологій і розробок, а також загального прогресу в технології комп'ютерного обладнання(рис.1.1.)[1].

Деякі НМ є моделями біологічних нейронних мереж, а деякі – ні, але історично склалося так, що значна частина натхнення для галузі НМ прийшла з бажання створювати штучні системи, здатні виконувати складні, можливо, «інтелектуальні» обчислення, подібні до тих, які виконує людина. мозок регулярно виконує, і таким чином, можливо, покращити наше розуміння людського мозку.

В принципі, НМ можуть обчислювати будь-яку обчислювану функцію, тобто вони можуть робити все, що може робити звичайний цифровий комп'ютер.

Майже будь-яке відображення між векторними просторами можна апроксимувати з довільною точністю за допомогою прямих НМ.

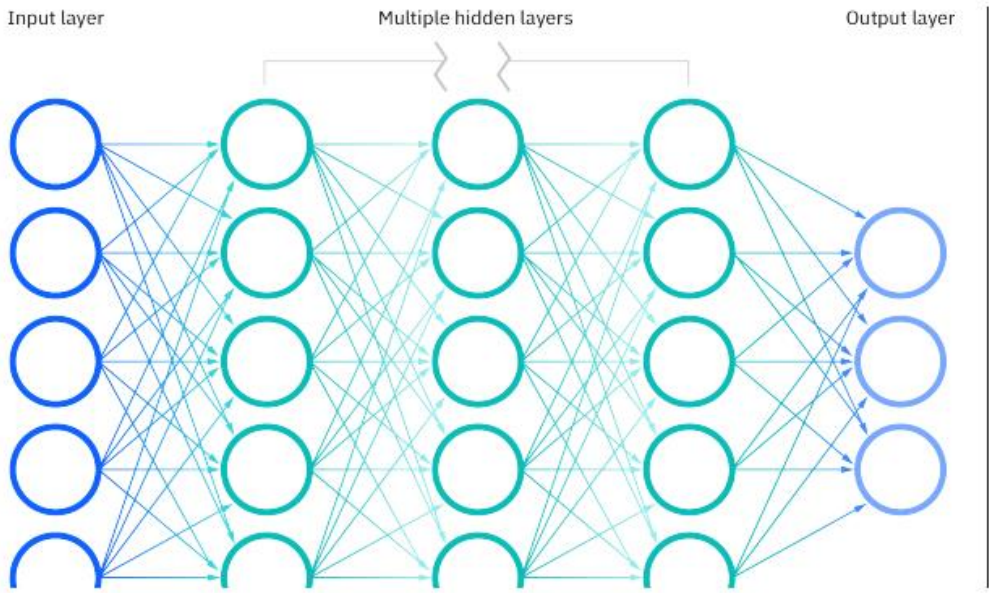


Рисунок 1.1. Нейронна мережа

Розглянемо деякі нейронні мережі для розпізнавання рукописного тексту:

1. Neuroph

Neuroph — це легкий Java Neural Network Framework для розробки поширених архітектур нейронних мереж. Графічний редактор інтерфейсу користувача полегшує вивчення та використання. Neuroph був повністю розроблений і закодований на Java. Це проект із відкритим початковим кодом, розміщений на SourceForge, а остання версія 2.9 була випущена під ліцензією Apache 2.0. Попередні версії були ліцензовані згідно з LGPL.

Neuroph написаний мовою Java і складається з двох блоків, як показано на схемі нижче(рис.1.2):

- Бібліотека Neuroph
- Neuroph Studio

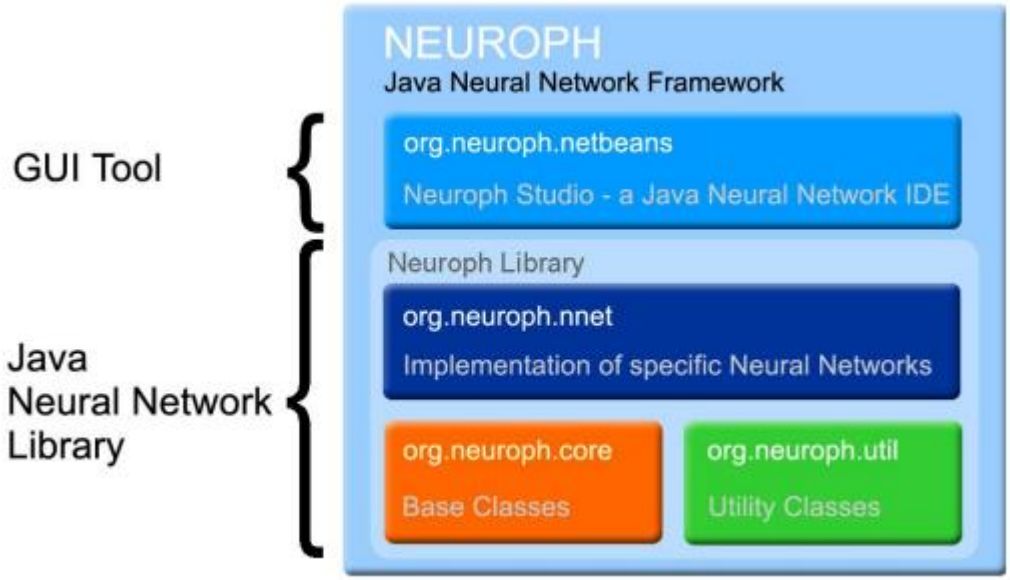


Рисунок 1.2. Структура Neuroph

Бібліотека Neuroph — це бібліотека нейронної мережі Java, яка складається з кількох пакетів Java, що дає розробникам як готові частини функціональності, так і створювати спеціальні додаткові компоненти, які не пов'язані з архітектурою нейронних мереж або алгоритмами навчання за допомогою плагінів.

Neuroph studio — це інструмент графічного інтерфейсу користувача, створений на основі платформи Netbeans і бібліотеки Neuroph, який надає прості у використанні майстри й інструменти нейронних мереж, щоб розробники могли створювати, тестувати та розгортати різноманітні компоненти Java на основі нейронних мереж у даному середовищі, чого раніше не було, оскільки воно мало окрему програму для розробки Java та іншу програму для побудови та створення нейронних мереж.

Програма Neuroph демонструє, як можна застосувати нейронні мережі для розпізнавання рукописного тексту. Додаток jHRT використовує штучну нейронну мережу для розпізнавання рукописних літер і перетворення їх у редагований документ (як MS Word .doc або файл Notepad і Wordpad .txt). Він заснований на нейронній мережі, яка може навчитися розпізнавати більше символів(рис.1.3).

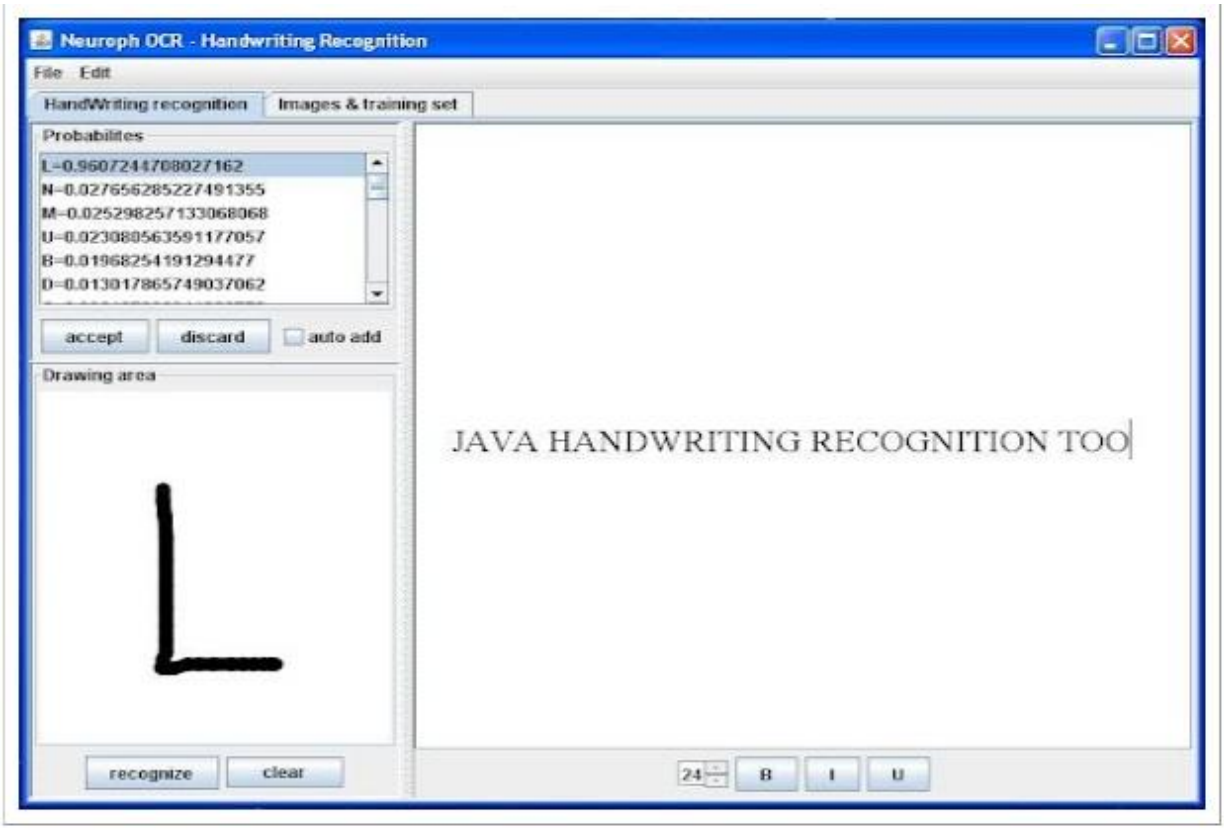


Рисунок 1.3. Програма Neuroph

2. TensorFlow.js

Розпізнавання рукописного тексту дозволяє перетворювати рукописні документи в цифрову форму[2]. Ця технологія зараз використовується в багатьох напрямках: зчитування поштових адрес, сум банківських чеків, оцифрування історичної літератури.

Завдяки tensorflow.js ця потужна технологія реалізована у браузері. У цій статті створимо веб-програму, яка зможе передбачити цифру, яка малюється на полотні.

Нижче наведено кінцевий результат веб-додатку, як видно, отримано досить хороший результат, він передбачає, шо намальовано цифру 5 із ймовірністю 99%(рис.1.4)

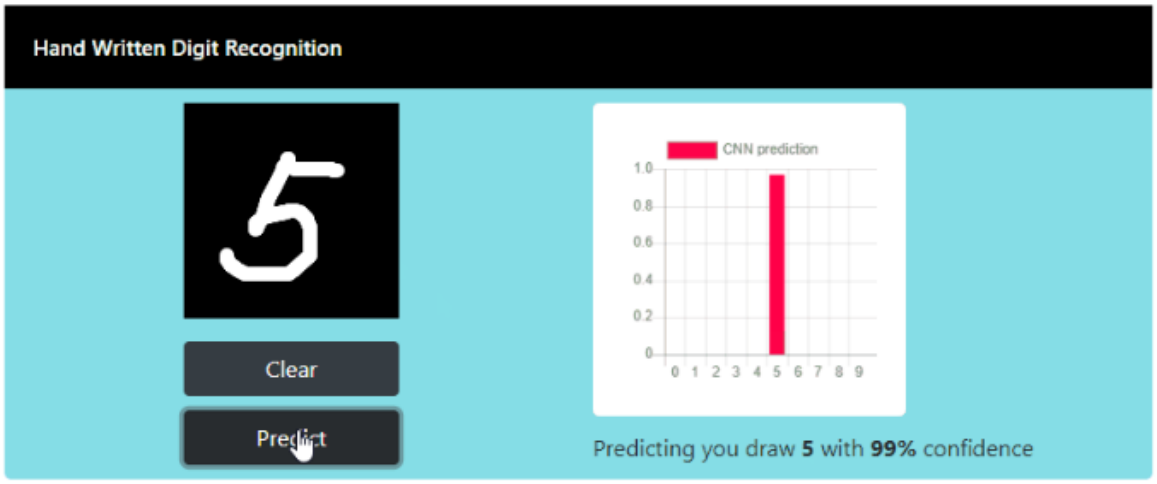


Рисунок 1.4. Демонстрація розпізнавання рукописної цифри

Ця програма спочатку використовує сценарій Python для навчання та збереження моделі, потім використовує бібліотеку JavaScript tensorflow.js, щоб завантажити модель у браузер і передбачити, яке число є цифрою, намальованою рукою.

1.2. Аналіз архітектури нейронних мереж

Архітектура нейронної мережі складається з окремих одиниць, званих нейронами, які імітують біологічну поведінку мозку.

Основні компоненти нейрона показані на рис.1.5.

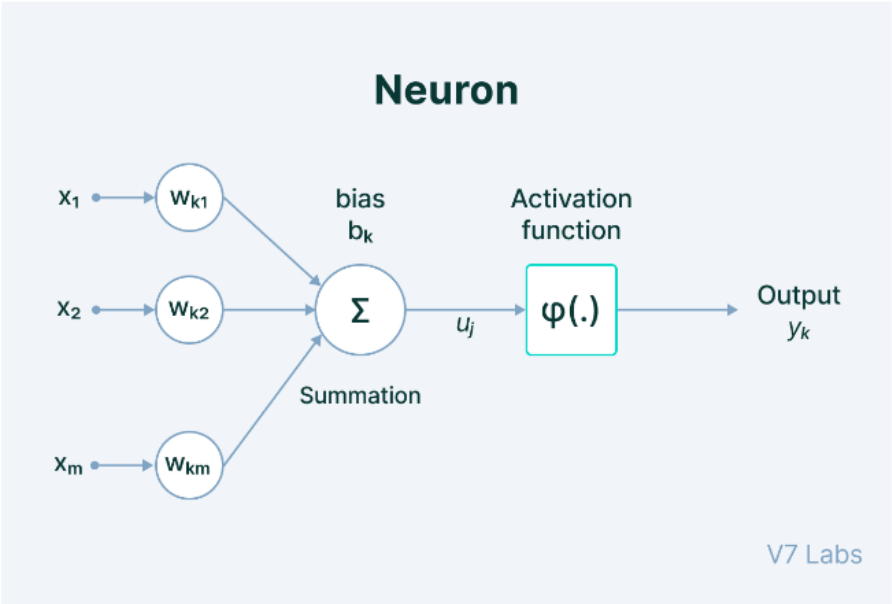


Рисунок 1.5. Нейрон у штучній нейронній мережі

Input – це набір функцій, які вводяться в модель для процесу навчання. Наприклад, вхідними даними для виявлення об'єктів може бути масив значень пікселів, що відносяться до зображення.

Weight – його головна функція полягає в тому, щоб надати значення тим характеристикам, які більше сприяють навчанню.

Transfer function. Завдання функції передачі полягає в об'єднанні кількох входів в одне вихідне значення, щоб можна було застосувати функцію активації. Це робиться шляхом простого підсумовування всіх вхідних даних для функції передачі.

Activation Function – вводить нелінійність у роботу персептронів, щоб враховувати різну лінійність із входами. Без цього вихід був би просто лінійною комбінацією вхідних значень і не міг би ввести нелінійність у мережу.

Bias. Роль зміщення полягає у зсуві значення, створеного функцією активації. Його роль подібна до ролі константи в лінійній функції.

Коли кілька нейронів розташовані разом в ряд, вони утворюють шар, і кілька шарів, розташованих поруч один з одним, називаються багатошаровою нейронною мережею.

Нижче описано основні компоненти цього типу структури (рис.1.6).

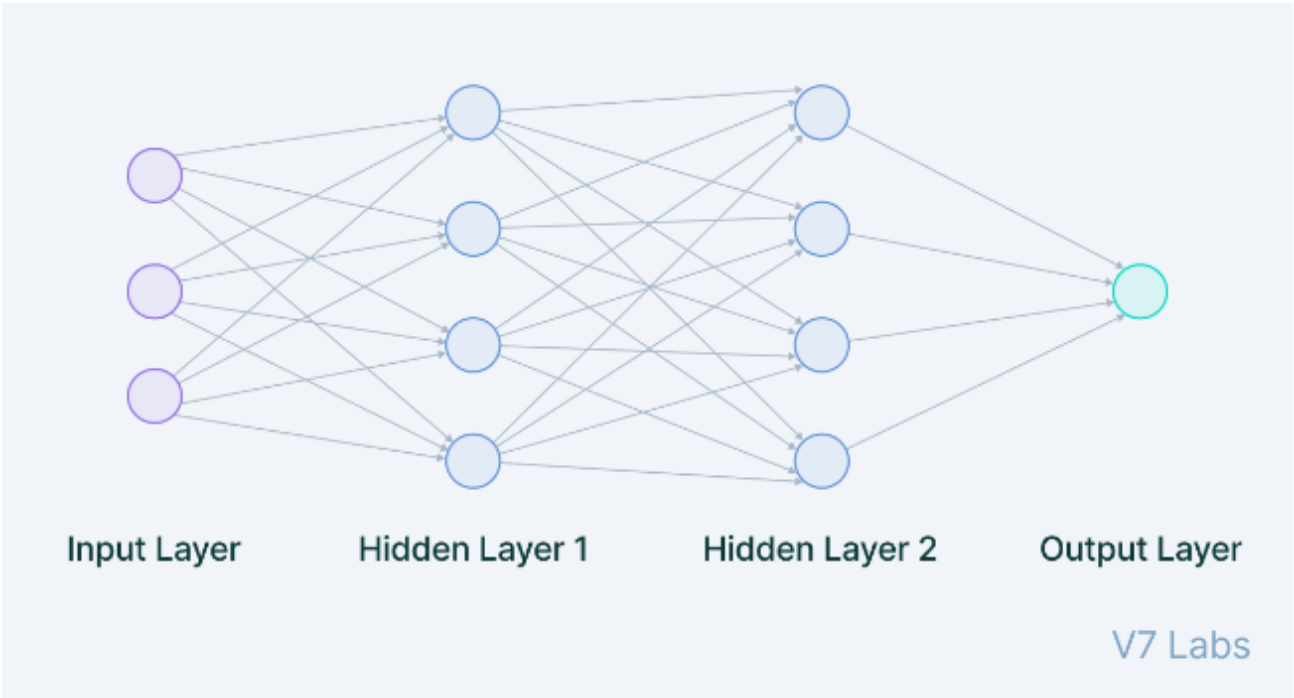


Рисунок 1.6. Багатошарова нейронна мережа

- Input Layer

Дані, які подаємо в модель, завантажуються на вхідний рівень із зовнішніх джерел, таких як файл CSV або веб-служба. Це єдиний видимий рівень у повній архітектурі нейронної мережі, який передає повну інформацію із зовнішнього світу без будь-яких обчислень.

- Hidden Layers

Приховані шари – це те, що робить глибоке навчання таким, яким воно є сьогодні. Вони є проміжними рівнями, які виконують усі обчислення та витягують характеристики з даних.

Може бути кілька взаємопов'язаних прихованих шарів, які відповідають за пошук різних прихованих функцій у даних. Наприклад, під час обробки зображень перші приховані шари відповідають за функції вищого рівня, такі як краї, форми чи межі. З іншого боку, пізніші приховані шари виконують складніші завдання, такі як ідентифікація повних об'єктів (автомобіль, будівля, людина).

- Output Layer

Вихідний рівень отримує вхідні дані з попередніх прихованих шарів і робить остаточний прогноз на основі знань моделі. Це найважливіший шар, де отримуємо кінцевий результат.

У випадку моделей класифікації та регресії вихідний рівень зазвичай має один вузол. Однак це повністю залежить від конкретної проблеми та від способу створення моделі.

Висновок до 1 розділу

При виконанні даної роботи був проведений аналіз декількох існуючих систем розпізнавання рукописного тексту, а саме:

1. Neuroph — це легкий Java Neural Network Framework для розробки поширених архітектур нейронних мереж. Графічний редактор інтерфейсу користувача полегшує вивчення та використання.
2. TensorFlow.js - Розпізнавання рукописного тексту дозволяє перетворювати рукописні документи в цифрову форму, яка використовується в багатьох

напрямках: зчитування поштових адрес, сум банківських чеків, оцифрування історичної літератури.

Також, було розглянуто архітектуру нейронної мережі, яка складається з окремих одиниць, званих нейронами, які імітують біологічну поведінку мозку.

РОЗДІЛ 2. АНАЛІЗ ФУНКЦІОНАЛЬНИХ ХАРАКТЕРИСТИК ТА ДОСЛІДЖЕННЯ НЕЙРОННИХ МЕРЕЖ

2.1. Вибір мови програмування

Для створення нейронної мережі розпізнавання рукописного тексту було обрано мову програмування Python.

Python — це мова програмування, яка була розроблена в кінці 1980-х і на початку 1990-х років Гвідо ван Руссомом у Нідерландах. Python походить від інших мов, таких як Java, C, C++ і Smalltalk, які на нього сильно вплинули.

Python є динамічно типізованою мовою програмування, що означає, що змінні можуть змінюватися під час виконання, і мова не змушує програміста визначати, який це тип даних, мова зробить це за програміста. Це одна з причин, чому цю мову часто сприймають як мову, зручну для початківців, оскільки вона не вимагає великих знань з інформатики, щоб почати роботу.

Python є інтерпретованою мовою, а не компільованою, і вона має можливості для об’єктно-орієнтованого програмування з використанням класів. З роками Python виріс і створив велику спільноту, яка використовує її для різноманітних програм і зараз є однією з найпопулярніших мов програмування. Ця мова програмування має широку підтримку більшості функцій, які необхідні для мови загального призначення. Наприклад, він підтримує використання веб-фреймворків, фреймворків настільних програм і має широкий спектр бібліотек [3].

Порівняльна характеристика Python з іншими мовами програмування:

Java, Javascript і Python значною мірою залежать від використання змінних. Змінні використовуються для збереження різних типів даних, які можна використовувати в різних місцях програми.

Спосіб роботи змінних у Python простий, немає необхідності писати тип даних перед іменем змінної та призначенням даних . Оскільки Python є динамічно типізованою мовою, кожна змінна може змінювати тип даних динамічно під час виконання. Змінна, яка містить текст, може раптово представляти замість нього число.

Змінні Javascript подібні до Python тим, що їм не потрібно оголошувати певний тип даних. Можна змінити дані, які може містити змінна, призначивши її в іншій точці програми.

В Java потрібно вказати, який тип даних повинна мати змінна, щоб використовувати її. Можна розібрати значення змінної в інший тип даних в іншій змінній за допомогою таких методів, як «valueOf». Однак це не так просто, як у javascript або python, де немає потреби використовувати функції, щоб змінити дані, які містить змінна.

Python використовує # для коментування одного рядка і не підтримує функцію формального блокового коментаря. Проте автори виявили, що можна імітувати блоковий коментар, перетворивши потрібний код у рядок. Це не елегантне рішення, і його краще не використовувати.

Java і Javascript використовують // для коментування одного рядка та /* */ для коментування блоків[4].

Python не має символу для оголошення розриву рядка, а натомість використовує hitespace, щоб побачити, чи відбувся розрив, тобто чи було згенеровано маркер нового рядка. Python має можливість об’єднувати рядки коду. Ситуації, в яких можна використовувати розриви рядків, це оператори if, які мають перевірити кілька факторів, щоб зробити їх більш читабельними.

Java використовує крапку з комою, щоб відокремити кожен оператор один від одного. Кожен оператор має закінчуватися крапкою з комою, інакше видаватиметься повідомлення про помилку.

Javascript використовує крапку з комою та розрив рядка як роздільники операторів. Це означає, що немає необхідності відокремлювати кожен оператор крапкою з комою, достатньо використовувати лише розрив рядка. Однак якщо більше одного оператора в тому самому рядку, тоді потрібно буде розділити її крапкою з комою.

2.1.1. Вибір середовища розробки

Visual Studio Code, вперше випущений у 2015 році, є редактором форматованого тексту з відкритим кодом, який працює в Windows, macOS і Linux. На відміну від Visual Studio, VSCode — це набагато легша інсталяція, яка потребує використання плагінів для кращого досвіду кодування. Ви можете використовувати VSCode як є для розробки, але ви можете виявити, що деякі функції, які очікуєте від IDE, відсутні, наприклад автозавершення, підсвічування синтаксису, лінтери коду та розумні компілятори. VSCode найкраще підходить для розробників, яким зручно налаштовувати своє середовище відповідно до своїх потреб, тому що їм це, ймовірно, доведеться (рис.2.1).

Рисунок 2.1. Платформи Visual Studio Code

Є кілька причин використовувати VSCode для розробки PX4:

1. Налаштування займає лише кілька хвилин.
2. Багата екосистема розширень, яка надає широкий спектр інструментів, необхідних для розробки PX4: C/C++ (з надійною інтеграцією cmake), Python , Jinja2 , повідомлення ROS і навіть UAVCAN dsdl.
3. Відмінна інтеграція Github

2.2. Вибір бібліотеки з відкритим кодом

Глибинні нейронні мережі, як випливає з терміну — це мережі нейронів, де кожен нейрон навчається виконувати власну операцію як частину більшої

картини. Такі дані, як зображення, надходять у цю мережу як вхідні дані та проходять через мережу, коли вони адаптуються під час навчання або прогнозують результати в розгорнутій системі.

TensorFlow — це стандартний спосіб представлення даних у навчанні. Простіше кажучи, TensorFlow — це просто багатовимірні масиви, розширення двовимірних таблиць (матриць) до даних з більшою вимірністю. Подібно до того, як чорно-білі (відтінки сірого) зображення представлені у вигляді «таблиць» значень пікселів, зображення RGB представлені у вигляді тривимірних масивів, де кожен піксель має три значення, що відповідають червоному, зеленому та синьому компонентам.

TensorFlow було розроблено з урахуванням портативності, що дає змогу виконувати обчислювальні графіки в різноманітних середовищах і апаратних платформах. Завдяки практично ідентичному коду ту саму нейронну мережу TensorFlow можна, наприклад, навчити в хмарі, розподіленій у кластері на багатьох машинах або на одному ноутбуці. Він може бути розгорнутий для надання прогнозів на виділеному сервері або на платформах мобільних пристроїв, таких як Android або iOS, або одноплатних комп'ютерах Raspberry Pi. TensorFlow також сумісний з операційними системами Linux, macOS і Windows.

Ядро TensorFlow на C++, і воно має дві основні мови інтерфейсу високого рівня та інтерфейси для вираження та виконання обчислювальних графіків. Найбільш розвинений зовнішній інтерфейс на Python, яким користується більшість дослідників і фахівців з обробки даних. Інтерфейс C++ надає досить низькорівневий API, корисний для ефективного виконання у вбудованих системах та інших сценаріях

Keras — це API, призначений для людей, а не для машин. Keras дотримується найкращих практик для зменшення когнітивного навантаження: він пропонує узгоджені та прості API, мінімізує кількість дій користувача, необхідних для типових випадків використання, і надає чіткі та дієві повідомлення про помилки. Він також містить розширену документацію та посібники для розробників.

Keras є центральною частиною тісно пов’язаної екосистеми TensorFlow 2, яка охоплює кожен етап робочого процесу машинного навчання, від керування даними до навчання гіперпараметрів і рішень для розгортання.

Keras використовується CERN, NASA, NIH та багатьма іншими науковими організаціями по всьому світу. Keras має гнучкість для реалізації довільних дослідницьких ідей, пропонуючи додаткові функції високого рівня зручності для прискорення циклів експериментів.

Завдяки своїй простоті у використанні та зосередженості на взаємодії з користувачем Keras є найкращим рішенням для глибинного навчання для багатьох університетських курсів. Його широко рекомендують як один із найкращих способів навчитися глибинному навчанню.

2.3. Види нейронних мереж

Складні штучні нейронні мережі розроблено для того, щоб моделі могли відображати нелінійний процес прийняття рішень людським мозком. Це означає, що моделі можна навчити приймати складні рішення або розуміти абстрактні поняття та об’єкти. Модель будуватиме від функцій низького рівня до складних функцій, розуміючи складні концепції. Кожен вузол у мережі зважується залежно від його впливу на інші вузли штучної нейронної мережі.

Як і інші моделі машинного навчання, оптимізація штучних нейронних мереж базується на функції втрат[5]. Це різниця між прогнозованим і фактичним результатом. Зважування кожного вузла та шару регулюється моделлю для досягнення мінімальних втрат. Моделі штучних нейронних мереж можуть розуміти кілька рівнів функцій даних і будь-який ієрархічний зв’язок між функціями. Таким чином, коли модель штучної нейронної мережі використовується для проблеми класифікації, вона може зрозуміти складні концепції, обробляючи кілька рівнів ознак.

Існує багато типів нейронних мереж, які можуть бути на стадії розробки. Їх можна класифікувати залежно від: структури, потоку даних, використовуваних нейронів та їхньої щільності, шарів та їх глибинних фільтрів активації тощо.

Відомі дев'ять типів нейронних мереж:

- перцептрон;
- нейронна мережа прямого поширення;
- багатошаровий перцептрон;
- згорткова нейронна мережа;
- радіальна базова функціональна нейронна мережа;
- рекурентна нейронна мережа;
- LSTM – довготривала короткочасна пам’ять;
- моделі «послідовність до послідовності»;
- модульна нейронна мережа.

2.3.1. Перцептрон

Персептрон є найважливішою моделлю нейронної мережі, яка активно вивчалася протягом останніх шести десятиліть. Його історія починається з початку 1940-х років, коли МакКаллох і Піттс виявили, що нейрон можна моделювати як пороговий пристрій, який може виконувати логічні функції. Пізніше в 1950-х роках Розенблат запропонував модель персептрона та її алгоритм навчання.

Для моделей перцептронів використовується контрольоване навчання. Контрольоване навчання порівнює вихід мережі та бажаний результат. Це замкнута система з помилкою як сигналом зворотного зв’язку. У контрольованому навчанні спочатку визначається цільова функція. Щоб об’єктивна вартість наблизилася до нуля, зазвичай застосовують процедуру градієнтного спуску, таку як алгоритми LMS і BP (рис. 2.2.).

Переваги перцептрона:

- Перцептрони можуть реалізувати такі логічні елементи, як AND, OR або NAND.

Недоліки перцептрона:

- Перцептрони можуть вивчати лише задачі з лінійним розділенням, такі як логічна проблема I. Для нелінійних задач, таких як логічна проблема XOR, це не працює.

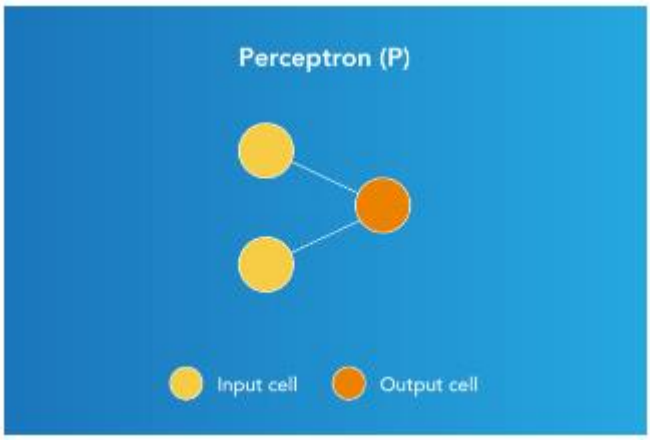


Рисунок 2.2. Перцептрон

2.3.2. Нейронна мережа прямого поширення

Як випливає з назви, вхідні дані подаються в прямому напрямку через мережу. Кожен прихований рівень приймає вхідні дані, обробляє їх відповідно до функції активації та передає на наступний рівень.

Вхідні дані слід подавати лише в прямому напрямку. Дані не повинні текти у зворотному напрямку під час генерації вихідних даних, інакше вони утворять цикл і вихідні дані ніколи не можуть бути згенеровані. Такі конфігурації мережі відомі як мережа прямого зв'язку . Мережа прямого зв'язку допомагає в прямому поширенні(рис.2.3).

Переваги прямої нейронної мережі

1. Менш складна, проста у проектуванні та обслуговуванні.
2. Швидка.
3. Висока чутливість до зашумлених даних

Недоліки прямої нейронної мережі :

1. Не можна використовувати для глибинногоо навчання [6]через відсутність щільних шарів і зворотного поширення]

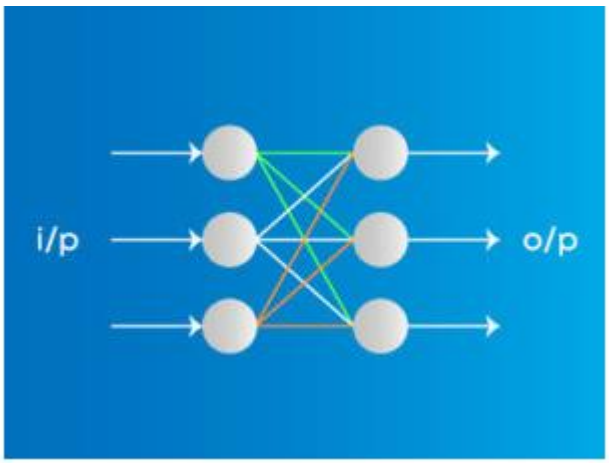


Рисунок 2.3. Нейронна мережа прямого поширення

2.3.3. Багатошаровий перцептрон

Багатошаровий перцептрон (БШП, англ. multilayer perceptron, MLP) (рис. 2.4) використовується для:

- Розпізнавання мови.
- Машинного перекладу.
- Комплексної класифікації.

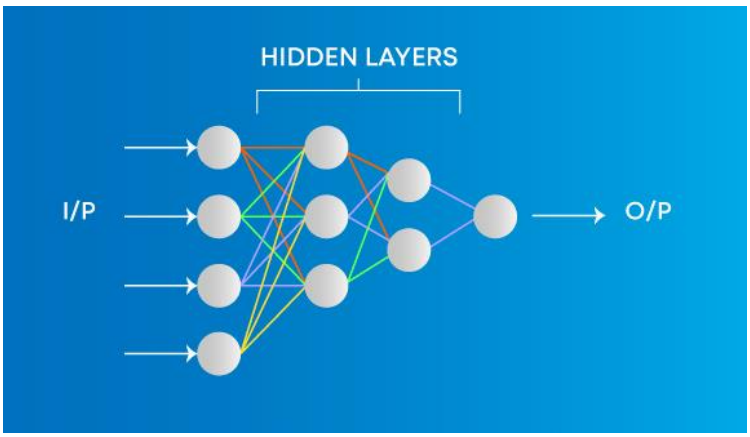


Рисунок 2.4. Багатошаровий перцептрон

Багатошаровий перцептрон є найвідомішим і найчастіше використовуваним типом нейронної мережі. У більшості випадків сигнали передаються в мережі в одному напрямку: від входу до виходу. Немає циклу, вихід кожного нейрона не впливає на сам нейрон. Ця архітектура називається прямою передачею.

Переваги багатошарового перцептрона

1. Використовується для глибокого навчання (через наявність щільних повністю з’єднаних шарів і зворотного поширення).

Недоліки багатошарового перцептрона:

1. Порівняно складний у проектуванні та обслуговуванні.

2.3.4. Згорткова нейронна мережа

Згорточна нейронна мережа, також відома як CNN або ConvNet, є класом нейронні мережі який спеціалізується на обробці даних, що мають сіткову топологію, наприклад зображення. Цифрове зображення — це двійкове представлення візуальних даних. Він містить серію пікселів, упорядкованих у вигляді сітки, яка містить значення пікселів, щоб позначити, наскільки яскравим і якого кольору повинен бути кожен піксель.

Переваги згорткової нейронної мережі:

1. Використовується для глибинного навчання з невеликою кількістю параметрів.

2. Менше параметрів для вивчення порівняно з повністю підключеним рівнем.

Недоліки згорткової нейронної мережі:

- Порівняно складна у проектуванні та обслуговуванні.
- Порівняно повільна (залежить від кількості прихованих шарів).

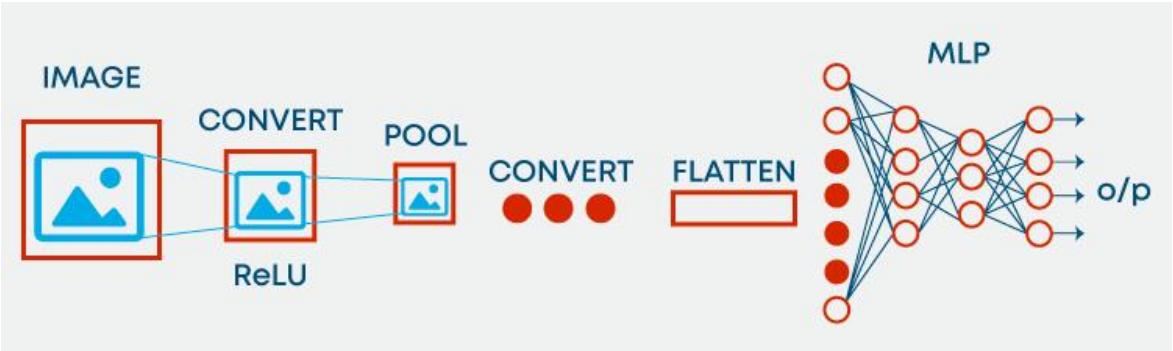


Рисунок 2.5. Згорткова нейронна мережа

2.3.5. Радіальна базова функціональна нейронна мережа

Мережа радіальних базових функцій (РБФ, англ. radial basis functions, RBF) складається з вхідного вектора, за яким слідує шар нейронів РБФ і вихідного рівня з одним вузлом на категорію, рис. 2.6.

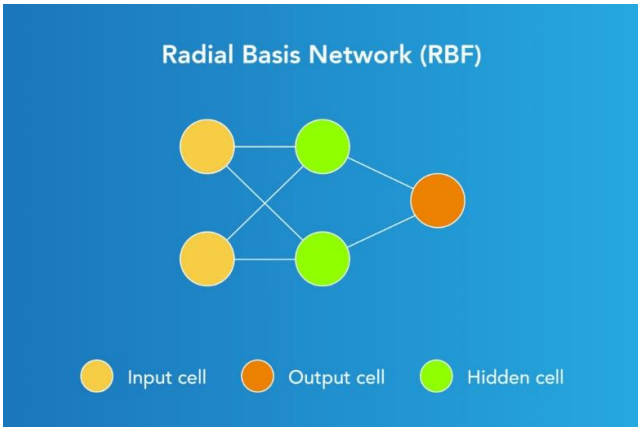


Рисунок 2.6. Радіальна базова функціональна нейронна мережа

Мережі радіальної базисної функції (RBF) є поширеним типом використання в штучних нейронних мережах для задач апроксимації функцій. Радіальні функціональні мережі відрізняються від інших нейронних мереж завдяки їх глобальній апроксимації та швидкій швидкості навчання. і нейронні мережі RBF також є типом прямої мережі, навченої за допомогою контрольованого алгоритму навчання. Основна перевага мережі RBF полягає в тому, що вона має лише один прихований шар і використовує функцію радіального базису як функцію активації. Ці функції дуже потужні в наближенні.

Робота радіальної базисної функції:

- RBFN виконують класифікацію, вимірюючи схожість вхідних даних із прикладами з навчального набору.
- RBFN мають вхідний вектор, який подається на вхідний рівень. Вони мають шар нейронів RBF.
- Функція знаходить зважену суму вхідних даних, а вихідний рівень має один вузол на категорію або клас даних.
- Нейрони в прихованому шарі містять функції передачі Гауса, які мають виходи, обернено пропорційні відстані від центру нейрона.
- Вихід мережі є лінійною комбінацією радіально-базисних функцій входу та параметрів нейрона.

2.3.6. Рекурентна нейронна мережа

Повторювані нейронні мережі, також відомі як RNN, є класом нейронних мереж, які дозволяють використовувати попередні виходи як вхідні, маючи приховані стани (рис.2.7).

Моделі RNN здебільшого використовуються в області обробки природної мови та розпізнавання мовлення.

Переваги рекурентних нейронних мереж

1. Можливість обробки вхідних даних будь-якої довжини.
2. Розмір моделі не збільшується разом із розміром вхідних даних.
3. Обчислення враховує історичну інформацію
4. Вагові коефіцієнти розподіляються в часі

Недоліки рекурентних нейронних мереж

1. Обчислення повільні.
2. Складнощі з доступом до інформації, отриманої давно.
3. Неможливо розглянути будь-який майбутній вхід для поточного стану.

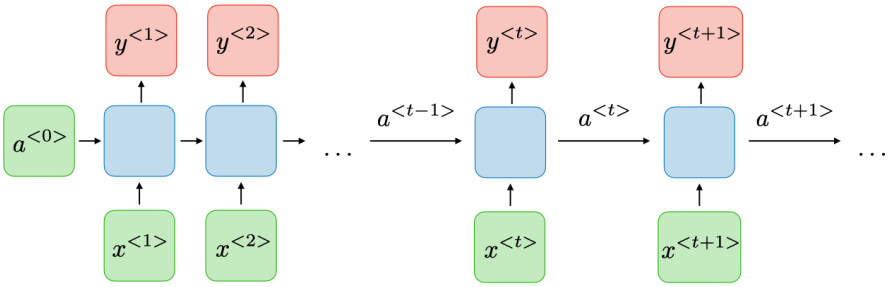


Рисунок 2.7 – Рекурентна нейронна мережа

2.3.7. LSTM – довготривала короткочасна пам’ять

Модель довготривалої короткочасної пам’яті (скорочено: LSTM) є підтипом рекурентних нейронних мереж (RNN). Він використовується для розпізнавання шаблонів у послідовностях даних, наприклад тих, що з’являються в даних датчиків, курсах акцій або природній мові. RNN можуть зробити це, оскільки, окрім фактичного значення, вони також включають його положення в послідовності у прогнозі.

Проблема повторюваних нейронних мереж полягає в тому, що вони просто зберігають попередні дані у своїй «короткочасній пам’яті». Коли пам’ять у ньому

вичерпується, він просто видаляє найдовше збережену інформацію та замінює її новими даними. Модель LSTM намагається уникнути цієї проблеми, зберігаючи вибрану інформацію в довготривалій пам’яті. Ця довготривала пам'ять зберігається в так званому стані клітини. Крім того, існує також прихований стан, який ми вже знаємо зі звичайних нейронних мереж і в якому зберігається короткочасна інформація з попередніх кроків обчислення. Прихований стан - це короткочасна пам'ять моделі. Це також пояснює назву довгострокових мереж. (рис.2.8).

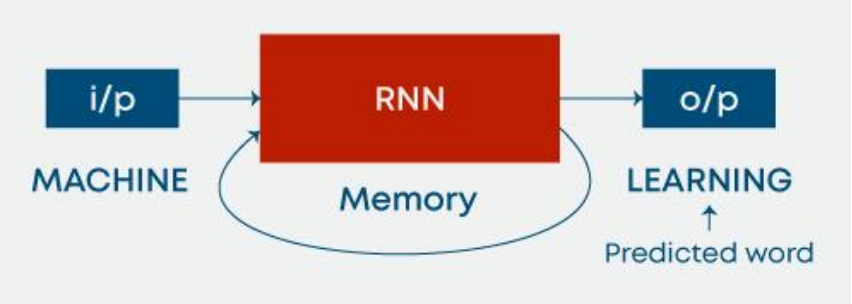


Рисунок 2.8. ДКЧП

Це одна з реалізацій осередків LSTM, однак існує багато інших архітектур (рис.2.9).

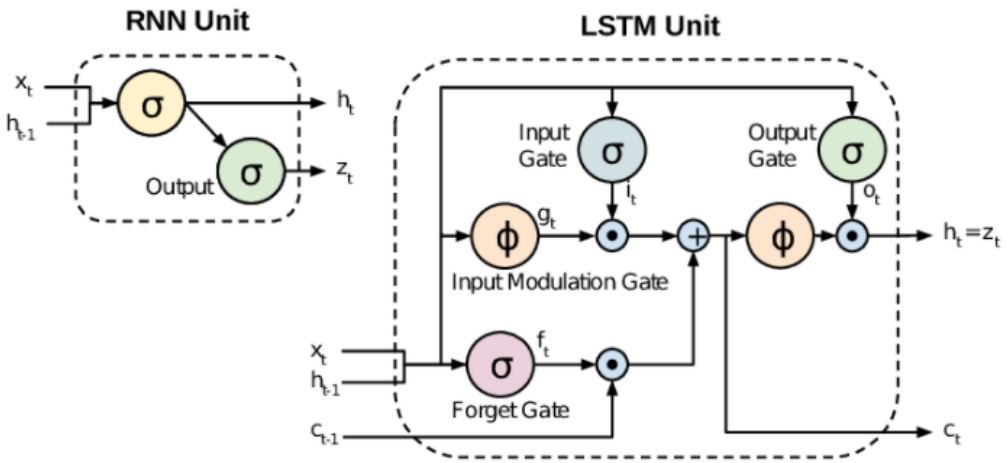


Рисунок 2.9. Архітектура LSTM

2.3.8. Моделі «послідовність-до-послідовності»

Модель “послідовність-до-послідовності” це потужна техніка машинного навчання, яка революціонізувала спосіб обробки природної мови (NLP). Це

дозволяє обробляти вхідні послідовності різної довжини та створювати вихідні послідовності різної довжини, що робить його особливо корисним для таких завдань, як переклад мови, розпізнавання мовлення та розробка чат-ботів. Послідовне моделювання також забезпечує чудову основу для створення текстових підсумків, систем відповідей на запитання, систем аналізу настроїв тощо. Завдяки широкому спектру застосувань вивчення концепцій послідовного моделювання є важливим для кожного, хто хоче працювати в галузі обробки природної мови. (рис.2.10).

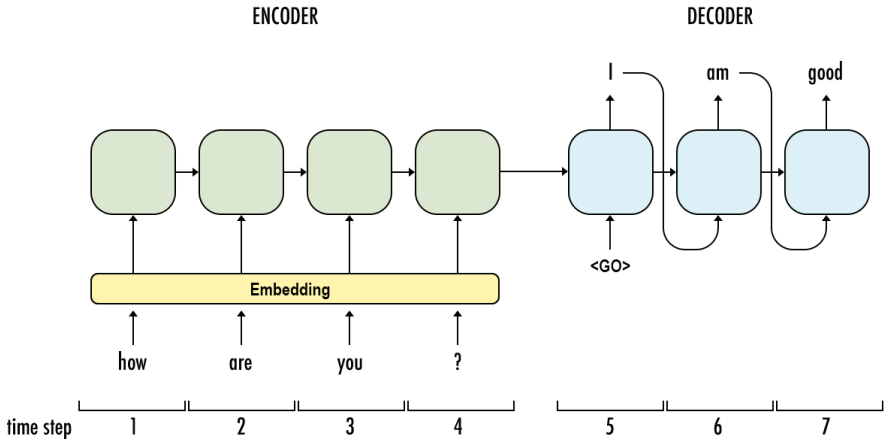


Рисунок 2.10. Модель послідовність-до-послідовгості

2.3.9. Модульна нейронна мережа

Застосування модульної нейронної мережі:

1. Системи прогнозування фондового ринку
2. Адаптивний MNN для розпізнавання символів

Модульні нейронні мережі – це використання ряду нейронних мереж для вирішення проблем. Тут різні нейронні мережі поводяться як модулі для вирішення частини проблеми. Усе завдання поділу проблеми на різні модулі, а також інтеграція відповідей модулів для генерування кінцевого результату системи виконується інтегратором. (рис.2.11).

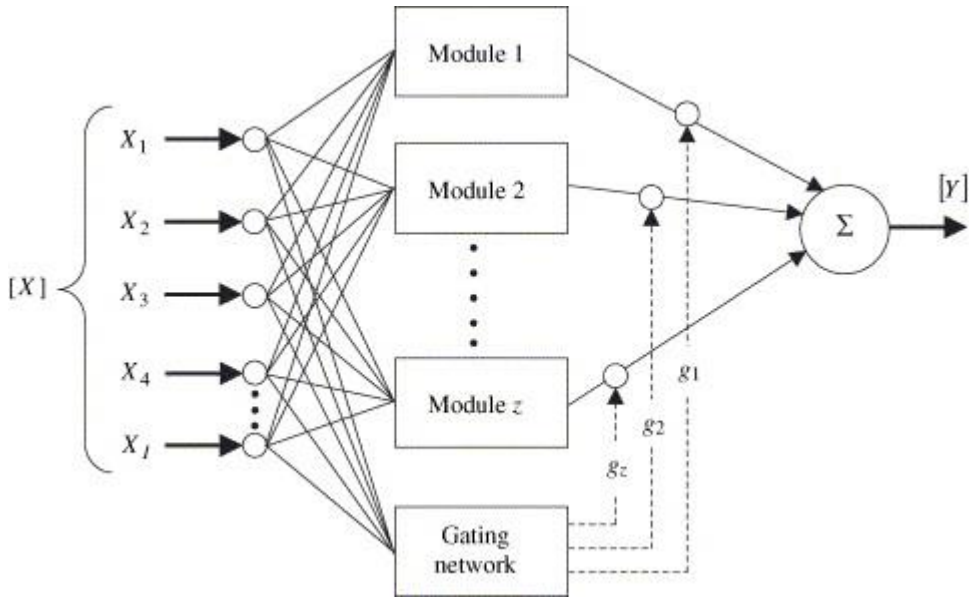


Рисунок 2.11. Модульна нейронна мережа

Переваги модульної нейронної мережі:

1. Ефективність.
2. Самостійне навчання.
3. Стійкість.

Недоліки модульної нейронної мережі

1. Проблеми з рухомою ціллю.

Висновки до 2 розділу

Було розглянуто та охарактеризовано вибір мови програмування, наведено його властивості та порівняння з іншими мовами програмування, такими як: Java, C++, JavaScript. Вибрано середовище розробки Visual Studio Code, наведено його переваги та недоілки, властивості та описано його характеристику. Також розглянуто бібліотеку з відкритим кодом TensorFlow.

Описано і розглянуто дев’ять видів нейронних мереж, таких як:

1. Перцептрон;
2. Нейронна мережа прямої подачі;
3. Багатошаровий персептрон;
4. Згорточна нейронна мережа;
5. Радіальна базова функціональна нейронна мережа;
6. Рекурентна нейронна мережа;

7. LSTM – довготривала короткочасна пам’ять;
8. Моделі «послідовність до послідовності»;
9. Модульна нейронна мережа.

РОЗДІЛ 3. МЕТОДИ ОБРОБКИ РУКОПИСНОГО ТЕКСТУ

3.1. Розпізнавання символів

Коли документ піддається візуальному розпізнаванню, очікується, що він складатиметься з друкованих (або рукописних) символів, що відносяться до одного або кількох написів або шрифтів. Однак цей документ може містити інформацію, окрім лише оптичних символів. Наприклад, може містять малюнки та кольори, які не надають жодної корисної інформації для миттєвого розпізнавання символів. Крім того, символи, які потрібно проаналізувати окремо, можуть існувати як кластери слів або можуть бути розташовані в різних місцях документа. Зазвичай таке зображення обробляється для зменшення шумів і виділення окремих символів з документа. Зручно для розуміння припустити, що представлене зображення звільнене від шумів і що окремі символи вже розташовані (за допомогою, наприклад, відповідного алгоритму кластеризації). Ця ситуація є синонімом ситуації, коли системі для розпізнавання надіслано єдиний безшумний символ[7].

Процес оцифрування важливий для нейронної мережі, яка використовується в системі. У цьому процесі вхідне зображення дискретизується у двійковому вікні, яке формує вхідні дані для системи розпізнавання. На рис. 3.1 букву А оцифровано на 6×8=48 цифрових клітинок, кожна з яких має один колір, чорний або білий. Тепер важливо закодувати цю інформацію у формі, зрозумілій для комп’ютера. Для цього присвоюється значення +1 кожному чорному пікселю та 0 кожному білому пікселю та створюється двійкова матриця зображення I, як показано на рис. 3.1c. Такої кількості перетворень достатньо для нейронної мережі. Оцифрування зображення в бінарну матрицю заданих розмірів робить вхідне зображення інваріантним щодо його фактичних розмірів. Таким чином, зображення будь-якого розміру перетворюється на двійкову матрицю фіксованих заздалегідь визначених розмірів. Це встановлює однаковість у розмірах вхідних і збережених шаблонів, як вони переміщатися по системі розпізнавання.

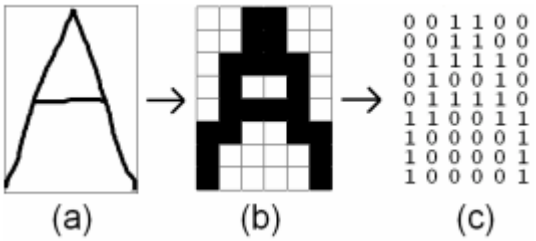


Рисунок 3.1. Символ для розпізнання

У застосовуваній системі використовується сильно спрощена архітектура штучних нейронних мереж. У використаному методі різні символи навчаються у мережі з вчителем. Екземпляр подається системі та отримує певну позначку. Декілька варіантів візерунків одного екземпляру навчаються у мережі під тією самою позначкою. Таким чином, мережа вивчає різні можливі варіації одного шаблону і стає адаптивною за своєю природою. Під час навчання входом до нейронної мережі є вхідна матриця M, яка визначається наступним чином:

$$If\ I(i,j)=1\ Then\ \ M(i,j)=1$$
$$Else:$$
$$If\ I(i,j)=0\ Then\ \ M(i,j)=-1$$

Вхідна матриця M тепер подається як вхід до нейронної мережі. Для будь-якої нейронної мережі характерно навчання контрольованим чи неконтрольованим способом шляхом коригування своїх ваг. У поточному методі навчання кожен символ-кандидат, який визначається в мережі, має відповідну вагову матрицю. Для k-го символу, який потрібно навчити до мережі вагова матриця позначається W_k. При зміні екземплярів символів з однаковою позначкою вагова матриця оновлюється. Спочатку (кероване навчання), матриця ініціалізується нулем. Кожного разу, коли екземпляр символу навчає мережу, надсилається шаблон введення, що задає цей символ у мережі. Потім мережа отримує вказівку ідентифікувати цей екземпляр як, скажімо, k-й символ у базі знань символів. Це означає, що екземпляру присвоєно позначку k. У відповідності з цим вагова матриця W_k оновлюється таким чином:

```
for all i=1 to x
{
  for all j=1 to y
  {
    W_k(i, j) = W_k(i, j) + M(i, j)
  }
}
```

Тут x і y — розміри матриці W_k (та M).

На рис. 3.2 показано оцифровані шаблони трьох різних написань символу S, які надаються системі для навчання.

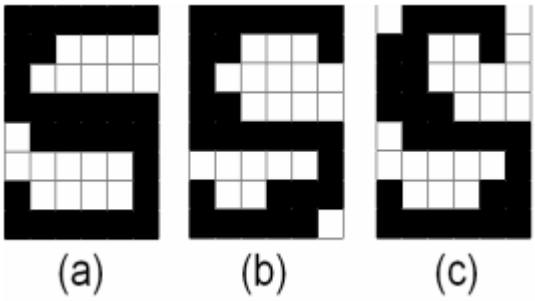


Рисунок 3.2. Шаблони символу S

Зауважте, що шаблони дещо відрізняються один від одного, так само як почерк відрізняється між людьми і як друковані символи відрізняються від машинного набору.

На рис. 3.3 подано вагову матрицю, скажімо, WS, що відповідає символу S. Матриця була оновлена тричі, щоб вивчити символ S. Слід зазначити, що ця матриця є специфічною лише для символу S. Інші символи повинні мати свою відповідну вагову матрицю.

$$W_S = \begin{pmatrix} 1 & 3 & 3 & 3 & 3 & 1 \\ 3 & 3 & -3 & -3 & -1 & -1 \\ 3 & -1 & -3 & -3 & -3 & -3 \\ 3 & 3 & 1 & -1 & -1 & -1 \\ -1 & 3 & 3 & 3 & 3 & 3 \\ -3 & -3 & -3 & -3 & -3 & 3 \\ 3 & -3 & -3 & -1 & 1 & 3 \\ 3 & 3 & 3 & 3 & 3 & 1 \end{pmatrix}$$

Рисунок 3.3. Вагова матриця символу S

Уважне спостереження за матрицею дозволяє зауважити наступне:

1. Елементи матриці з вищими (позитивними) значеннями є тими, які позначають пікселі зображення, які найчастіше зустрічаються.

2. Елементи з меншими або від'ємними значеннями означають пікселі, які рідше з'являються на зображеннях.

Нейронні мережі навчаються через таке оновлення своїх ваг. Щоразу вагові коефіцієнти регулюються таким чином, щоб результат був ближчим до бажаного, ніж раніше. Ваги можуть задавати важливість або пріоритет параметра, що в даному випадку є поява певного пікселя в шаблоні символів. Можна побачити, що ваги найбільш частих пікселів є вищими і зазвичай позитивними, а незвичайних пікселів нижчими і часто негативний. Таким чином, матриця призначає пікселям важливість на основі їх частоти появи в шаблоні.

Іншими словами, високо ймовірним пікселям призначається вищий пріоритет, тоді як менш часті пікселі штрафуються. Проте всі позначені шаблони розглядаються без упередженості, щоб включати неупередженість адаптації в системі.

3.2. Впровадження мережі для класифікації букв

Коефіцієнт розпізнавання (Q): ця статистика показує, наскільки добре система розпізнавання визначає вхідний шаблон як відповідний кандидат для одного з багатьох вивчених шаблонів[8]:

$$Q(k) = \frac{\psi(k)}{\mu(k)}$$

Чим більше значення Q, тим більше впевненості система надає шаблону введення як схожому на вже відомий шаблон. Класифікація вхідних шаблонів тепер дотримується такої тривіальної процедури:

1. Для вхідного шаблону-кандидата I обчислити коефіцієнт розпізнавання (Q(k)) для кожного вивченого шаблону k.

2. Визначити значення k, для якого Q(k) має максимальне значення.

3. Занадто низьке максимальне значення Q(k) (скажімо, менше 0,5) вказує на погане розпізнавання. У такому випадку:

- Потрібно зробити висновок, що шаблон кандидата не існує в базі знань АБО
- Навчити шаблон-кандидат мережі, доки не буде отримано задовільне значення Q(k).

1. Умовно визначити шаблон-кандидат введення як подібний до k-го вивченого шаблону АБО продовжувати навчання для кращої продуктивності.

На рис. 3.4 селектор дає результат k, роблячи найкращий вибір. Адаптивну ефективність мережі можна легко перевірити на прикладі: надсилаються два намальовані від руки шаблони, що задають символи S і P відповідно, системі, яка вже вивчила лише символ S. Коефіцієнт розпізнавання, отриманий навченою системою, вказується поряд(рис.3.5)

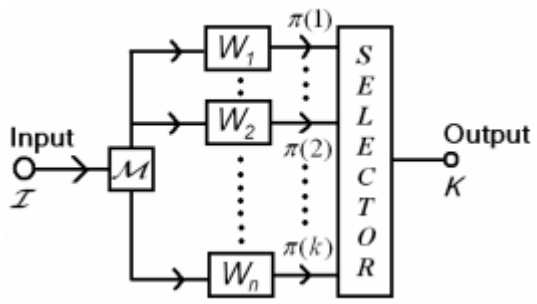


Рисунок 3.4. Селектор вибору результатів

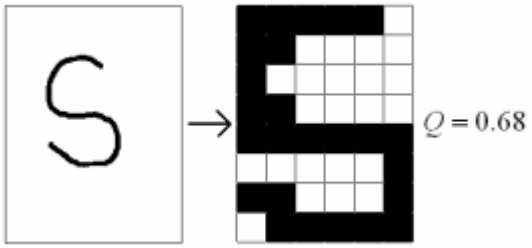


Рисунок 3.5. Результат селектора вибору для символу S

Зауважимо, що шаблон на рис. 3.4 не зовсім подібний на три шаблони на рис. 3.2, які були навчені системі. Однак, будучи адаптивною, система тим не менш дає хороший коефіцієнт Q=0,68 для шаблону, що вказує на відповідність. Щоб покращити розпізнавання цього конкретного шаблону, той самий шаблон можна багаторазово вводити в систему та навчати її, як і раніше, під тією самою позначкою. У результаті значення Q наближається до одиниці після кожного навчання шаблону, що і ілюструє навчання[9].

За допомогою регулярного навчання можна спостерігати, що система розвивається на основі своєї здатності ідентифікувати відповідний шаблон і відхиляти невідповідні шаблони. Таким чином, регулярне навчання з вчителем відзначає покращену продуктивність системи.

Висновки до 3 розділу

У даному розділі наведено методи обробки рукописного тексту, приклади оцифровування шаблонів, на прикладі букви S та подано її вагову матрицю, яка є специфічною лише для букви S. Впроваджено мережу для класифікації букв коефіцієнтом Q , яка показує наскільки добре система розпізнавання визначає вхідний шаблон як відповідний кандидат для одного з багатьох вивчених шаблонів.

РОЗДІЛ 4. РОЗРОБКА НЕЙРОМЕРЕЖЕВОЇ СИСТЕМИ ДЛЯ РОЗПІЗНАННЯ РУКОПИСНОГО ТЕКСТУ

4.1. Огляд засобів програмного забезпечення

Flask — це веб програмна платформа (англ. Framework), яка не містить ORM (Object Relational Manager) або подібних функцій.

Flask має багато цікавих функцій, таких як маршрутизація URL-адреси, система шаблонів. Це структура веб-додатків WSGI.

Інтерфейс шлюзу веб-сервера (Web Server Gateway Interface, WSGI) використовувався як стандарт для розробки веб-додатків на Python[10-12].

Werkzeug — це набір інструментів WSGI, який реалізує запити, об'єкти відповіді та службові функції. Це дозволяє створити на ньому веб-фрейм. Фреймворк Flask використовує Werkzeg як одну зі своїх основ.

jinja2 — популярний механізм шаблонів для Python. Система веб-шаблонів поєднує шаблон із певним джерелом даних для відтворення динамічної веб-сторінки.

На відміну від програмної платформи Django, Flask дуже Pythonic. Розпочати роботу з Flask легко, тому що для нього не потрібно довго вчитися.

Переваги Flask:

Маштабованість. Розмір – це все, і статус Flask, як мікро програмної платформи. означає, що можна використовувати його для неймовірно швидкого розвитку технологічного проекту, наприклад веб-додатка. Якщо потрібно створити програму, яка починається з малого, але має потенціал для швидкого розвитку та в напрямках, які ще не повністю опрацювані, тоді це ідеальний вибір. Його простота у використанні та невелика кількість залежностей дозволяють йому працювати безперебійно, навіть якщо він масштабується все більше і більше.

Гнучкість. Це основна функція Flask і одна з його найбільших переваг. Перефразовуючи один із принципів дзен Python , простота краща за складність, тому що її можна легко переставляти та переміщувати.

Це не тільки корисно з точки зору того, що дозволяє вашому проекту легко рухатися в іншому напрямку, це також гарантує, що конструкція не зруйнується, коли частина буде змінена. Мінімальний характер Flask і його здатність до розробки невеликих веб-програм означає, що він навіть більш гнучкий, ніж сам Django.

Легкість. Коли використовуємо цей термін по відношенню до інструменту чи фреймворку, говоримо про його дизайн — є кілька складових частин, які потрібно зібрати та повторно зібрати, і він не покладається на велику кількість розширень, щоб функціонувати. Цей дизайн дає веб-розробникам певний рівень контролю.

Flask також підтримує модульне програмування, де його функціональні можливості можна розділити на кілька взаємозамінних модулів. Кожен модуль діє як незалежний будівельний блок, який може виконувати одну частину функціональності. Разом це означає, що всі складові частини конструкції є гнучкими, рухливими та перевіряються самостійно.

Документованість. Дотримуючись власної теорії творця про те, що «гарний дизайн документації змушує користувача насправді писати документацію», користувачі Flask знайдуть велику кількість прикладів і порад, упорядкованих у структурований спосіб. Це спонукає розробників використовувати фреймворк, оскільки вони можуть легко познайомитися з різними аспектами та можливостями інструменту. Можна знайти документацію Flask на їхньому офіційному веб-сайті .

Недоліки Flask: не так багато інструментів. Є деякі недоліки легкої природи цього мікро програмної платформи. Головним з них є те, що на відміну від Django, у Flask відсутній великий інструментарій. Це означає, що розробникам доведеться вручну додавати розширення, наприклад бібліотеки. І, якщо додати величезну кількість розширень, це може почати гальмувати саму програму через велику кількість запитів.

Важко ознайомитися з великою програмою Flask. Через те, що розробка веб-програми за допомогою Flask може потребувати різноманітних розгалужень, веб-розробник, який приходить до проекту на півдорозі, може важко зрозуміти, як він

був розроблений. Модульний характер мікро програмної платформи, про яку згадувалося раніше, може знову зацікавити програмістів, яким доведеться ознайомитися з кожною складовою частиною.

Витрати на технічне обслуговування. Оскільки він настільки універсальний у плані того, з якими технологіями він може взаємодіяти, компанія, яка використовує Flask, часто несе додаткові витрати на підтримку цих технологій. Наприклад, якщо технологія, яка взаємодіє з вашим додатком Flask, застаріє або припиниться, то компанії доведеться з усіх сил шукати нову сумісну. Чим складнішою стає програма, тим вищі потенційні витрати на обслуговування та впровадження[13-14].

4.2. Розробка програмної мережі

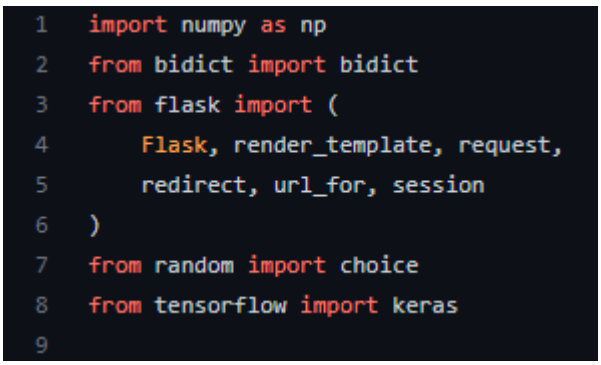
Для розробки нейронної мережі було створено декілька файлів, які базувались на використанні програмної платформи Flask, а також було встановлені всі вище згадані бібліотеки.

Структура файлів наведена нижче на рисунку 4.1.

Рисунок 4.1. Структура файлів наведена з Github

Яку вже було сказано, для написання нейронної мережі використовувався Flask. Отже, почнемо описувати розробку програмної мережі.

Для початку було створено файл apру.ру, в якому були імпортовані наступні бібліотеки, які наведені на рисунку 4.2.



```
import numpy as np
from bidict import bidict
from flask import (
    Flask, render_template, request,
    redirect, url_for, session
)
from random import choice
from tensorflow import keras

```

Рисунок 4.2. Імпортовані бібліотеки

Дана програма розбивалась на дві частини:

1. Перша частина програми давала можливість користувачу додавати букви до нейронної мережі, запам’ятовувати її для подальшого використання.

2. Друге – це те, що дана програма включає в себе можливість вікторини, яка дає зіграти в гру. Тобто програма показує яку букву потрібно вписати, і після введення букви, перевіряється чи введена буква відповідає потрібному значенню.

Пропишемо маршрут шляху для головної сторінки в переглядачі:

```
@app.route('/')
def index():
    session.clear()
    return render_template("index.html")


@app.route('/add-data', methods=['GET'])
def add_data_get():
        message = session.get('message', '')
```

Потім створюється каталог з назвою templates, де були створені html-файли (рис.4.3.) :

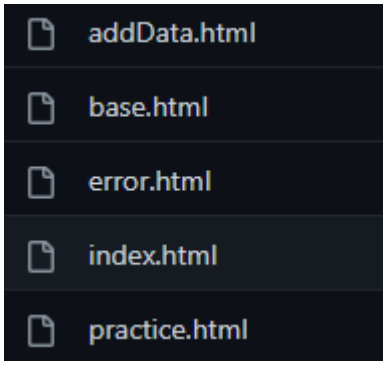


Рисунок 4.3. Html-файли

Далі створюємо тип словника, за допомогою бібліотеки from bidict import bidict, для того щоб створити повний словник, у якому кожна літера буде відповідати свої цифрі (рис.4.4.).

```
ENCODER = bidict({
    'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6,
    'G': 7, 'H': 8, 'I': 9, 'J': 10, 'K': 11, 'L': 12,
    'M': 13, 'N': 14, 'O': 15, 'P': 16, 'Q': 17, 'R': 18,
    'S': 19, 'T': 20, 'U': 21, 'V': 22, 'W': 23, 'X': 24,
    'Y': 25, 'Z': 26
})
```

Рисунок 4.4. Словник символів

Завдяки бібліотеці random, зроблено випадкову вибірку букв, для вікторини. Особливе зауваження, літери, які будуть вписуватись від руки на вікторині повинні бути у верхньому регістрі. Нижче наведено функцію, яка відповідає за випадкову вибірку літер у вікторині (рис.4.5.).

```
3  @app.route('/practice', methods=['GET'])
4  def practice_get():
5      letter = choice(list(ENCODER.keys()))
6      return render_template("practice.html", letter=letter, correct='')
```

Рисунок 4.5. Функція випадкової вибірки букв

Функція, зображена на рисунку 4.6, визначає пікселі у рукописному тексті, тобто, при введенні літери на сторінці програми, вона записується в масив, у якому буде показано її параметри, тобто клітинки на яких вона була замальована, а яка ні.

```python
@app.route('/add-data', methods=['POST'])
def add_data_post():

    label = request.form['letter']
    labels = np.load('data/labels.npy')
    labels = np.append(labels, label)
    np.save('data/labels.npy', labels)

    pixels = request.form['pixels']
    pixels = pixels.split(',')
    img = np.array(pixels).astype(float).reshape(1, 50, 50)
    imgs = np.load('data/images.npy')
    imgs = np.vstack([imgs, img])
    np.save('data/images.npy', imgs)

    session['message'] = f'"{label}" added to the training dataset'

    return redirect(url_for('add_data_get'))
```

Рисунок 4.6. Визначення пікселів рукописного тексту

Отже, дана програма не є досить важкою в розумінні, вона легка у використанні та зрозуміла для користувачів. В даному розділі були наведені основні фрагменти коду, а решта коду відображена в додатку А.

4.3. Тестування розробленої системи

Тестування даної програми проводилось на ПК з операційно системою Windows. Для запуску програми потрібно завантажити репозиторій, і встановити з requirements.txt всі потрібні залежності, які були встановлені для роботи даної нейронної мережі.

Після підключення до локального сервера, користувачеві надаються два вікна, одне з них для вікторини, інше для додавання букви у нейронну мережу(рис.4.7).

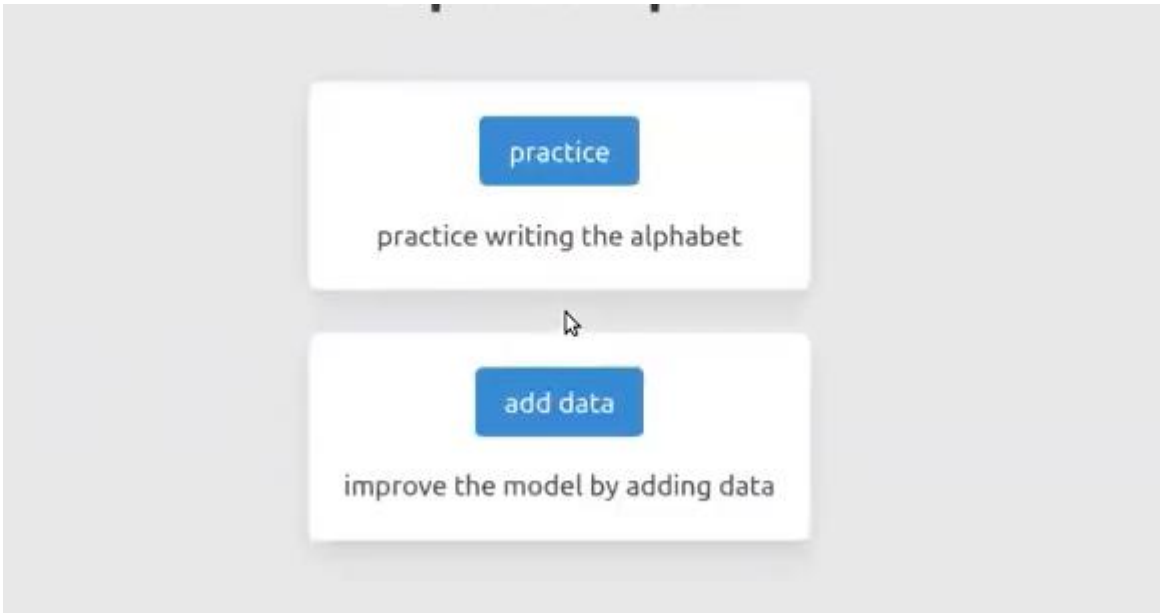


Рисунок 4.7. Графічні вікна для введення символів

Коли користувач вибирає вікторину, то йому потрібно ввести букву, яка зображена в надписі нижче, наприклад(рис.4.8.)

Рисунок 4.8. Приклад введення букви

Після чого користувач вводить цю букву у відповідне поле (рис.4.9.)

Рисунок 4.9. Введення букви

При веденні правильної букви буде виведено повідомлення True, якщо все правильно, і False – в іншому випадку.

Висновки до 4 розділу

У даному розділі описаний огляд засобів програмного забезпечення, вибрано фреймоврк. Наведено його переваги та недоліки у порівнянні з Django. Розроблено програмний код для розпізнавання рукописного тексту у вигляді гри, в якому користувач повинен ввести букву показану на сторінці. Також даний програмний код було протестовано і наведено результати відповідної роботи .

РОЗДІЛ 5. ЕКОНОМІЧНА ЧАСТИНА

5.1. Розрахунок економічної ефективності розробленої програми

У даному розділі наведені поточні ціни розробки даного програмного забезпечення[15].

Оплата програміста для створення програмного коду «Розпізнавання рукописного тексту нейромережею», можна розділити на дві можливості:

1. Розрахунок програміста відразу після створення даної програми буде оцінена в 1500$, і не підтримуватиметься більше даним програмістом;
2. Розрахунок програміста по-місячно, в результаті чого виходить 200$, тобто якщо врахувати цілий рік, то 2400$. Але при цьому програміст повинен при необхідності вдосконалювати певні блоки коду або бути на сапорті при певній поломці даної програми.

Якщо програміст не має певного обладнання для написання коду, то можна надати йому новий пристрій, але з урахуванням на те що ціна за виконаний продукт буде нища вказаного на початку, тобто розрахунок програміст буде оцінено в 800$, якщо не підтримуватиме його більше. І 50$ щомісячно при сапорті даного продукту, тобто 600$ за рік.

Вартість комп’ютерного обладнання оцінено в 500$, якщо програміст не має власного пристрою.

Час виконаний на розробку даного проекту оцінено в місяць. Тобто при наданні необхідних компонентів розробнику буде надано половину суми відразу, решта половини буде розбито на рік, якщо продукт потребує підтримки, або відразу після здачі і тестуванні програмного коду.

Висновок до 5 розділу

В даному розділі описано економічну частину розробки програмного продукту, наведено оцінку самого продукту, та розглянуто розрахунок при необхідності надання певних обладнань програмісту.

ВИСНОВКИ

Було виявлено, що символи з двома прихованими шарами, кожен із яких містить 100 нейронів, забезпечують найвищу точність розпізнавання 90,19%. Система розпізнавання рукописного тексту, описана в цьому документі, знайде потенційне застосування для розпізнавання рукописного імені, читання документів, перетворення будь-якого рукописного документа у структурну текстову форму та розпізнавання поштової адреси.

По-перше, в даній роботі була розроблена нейронна мережа, за допомогою встановлених бібліотек:

- Neuroph — це легкий Java Neural Network Framework для розробки поширених нейронних мереж. Він включає бібліотеку Java з відкритим початковим кодом із невеликою кількістю базових класів, які відповідають основним концепціям NN, а графічний редактор інтерфейсу користувача полегшує вивчення та використання.
- TensorFlow.js - Розпізнавання рукописного тексту дозволяє перетворювати рукописні документи в цифрову форму, яка використовується в багатьох напрямках: зчитування поштових адрес, сум банківських чеків, оцифрування історичної літератури.

Також, було розглянуто архітектуру нейронної мережі, яка складається з окремих одиниць, званих нейронами, які імітують біологічну поведінку мозку.

По-друге, було розглянуто та охарактеризовано вибір мови програмування, наведено його властивості та порівняння з іншими мовами програмування, такими як: Java, C++, JavaScript. Вибрано середовище розробки Visual Studio Code, наведено його переваги та недоілки, властивості та описано його характеристику. Також розглянуто бібліотеку з відкритим кодом TensorFlow.

По-третє, наведено методи обробки рукописного тексту, приклади оцифровування шаблонів, на прикладі букви S та подано її вагову матрицю, яка є специфічною лише для букви S. Впроваджено мережу для класифікації букв коефіцієнтом Q, яка показує наскільки добре система розпізнавання визначає

вхідний шаблон як відповідний кандидат для одного з багатьох вивчених шаблонів.

По-четверте, описаний огляд засобів програмного забезпечення, вибрано фреймоврк Flask. Наведено його переваги та недоліки у порівнянні з Django. Розроблено програмний код для розпізнавання рукописного тексту у вигляді гри, в якому користувач повинен ввести букву показану на сторінці. Також даний програмний код було протестовано і наведено результати відповідної роботи.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Vats I. Offline Handwritten English Numerals Recognition using Correlation Method / I. Vats, S. Singh. // International Journal of Engineering Research and Technology. – 2014. – №6. – С. 1626–1629.
2. Singh G. Recognition of Handwritten Hindi Characters using Back propagation Neural Network / G. Singh, S. Lehri. // International Journal of Computer Science and Information Technologies. – 2012. – №4. – С. 4892–4895.
3. Devnagiri Character Recognition Using Neural Networks / [S. S. Sayyad, A. Jadhav, M. Jadhav та ін.]. // International Journal of Engineering and Innovative Technology. – 2013. – №1. – С. 476–480.
4. Patel K. Offline Handwritten Character Recognition: A Review / K. Patel, M. Gandhi. // International Journal of Scientific & Engineering Research. – 2016. – №5. – С. 193–196.
5. Fuzzy Based Handwritten Character Recognition System / S.Borde, E. Shah, P. Rawat, V. Patil. // International Journal of Engineering Research and Applications. – 2012. – С. 151–154.
6. Dongare S. A. Handwritten Devanagari Character Recognition using Neural Network / S. A. Dongare, D. B. Kshirsagar, S. V. Waghchaure. // IOSR Journal of Computer Engineering. – 2014. – №14. – С. 74–79
7. Patil M. B. Recognitio0n of Handwritten Devnagari Characters through Segmentation and Artificial neural networks / M. B. Patil, V. Narawade. // International Journal of Engineering Research & Technology. – 2012. – №1. – С. 1–15.
8. Kaur M. A recognition system for handwritten gurmukhi characters / M. Kaur, S. Kumar. // International Journal of Engineering Research & Technology. – 2012. – №1. – С. 1–5.
9. Nohaj M. Image preprocessing for optical character recognition using neural networks / M. Nohaj, R. Jaka. // Journal of Patter Recognition Research. – 2011. – С. 1–11.
10. Sharma N. Recognition for Handwritten English Letters: A Review / N. Sharma,

T. Patnaik, K. Bhupendra. // International Journal of Engineering and Innovative Technology. – 2013. – №2. – С. 318–320.

11. Ghouzam Y. Digit Recognizer [Електронний ресурс] / Yassine Ghouzam. – 2017. – Режим доступу до ресурсу: https://www.kaggle.com/yassineghouzam/introduction-to-cnn-keras-0-997-top-6.
12. EMNIST: an extension of MNIST to handwritten letters. Gregory Cohen, Saeed Afshar, Jonathan Tapson, and Andr´e van Schaik The MARCS Institute for Brain, Behaviour and Development Western Sydney University (https://arxiv.org/abs/1702.05373v1).
13. The EMNIST Dataset (https://www.nist.gov/itl/iad/image-group/emnistdataset)
14. https://en.wikipedia.org/wiki/Visual_Studio_Code
15. Foltyniewicz, R. (1995, September). Efficient high order neural network for rotation, translation and distance invariant recognition of gray scale images. In International Conference on Computer Analysis of Images and Patterns (pp. 424-431). Springer, Berlin, Heidelberg.

ДОДАТКИ

Додаток А . Код реалізованої програми на мові Python

```
import numpy as np
from bidict import bidict
from flask import (
    Flask, render_template, request,
    redirect, url_for, session
)
from random import choice
from tensorflow import keras



ENCODER = bidict({
    'A': 1, 'B': 2, 'C': 3, 'D': 4, 'E': 5, 'F': 6,
    'G': 7, 'H': 8, 'I': 9, 'J': 10, 'K': 11, 'L': 12,
    'M': 13, 'N': 14, 'O': 15, 'P': 16, 'Q': 17, 'R': 18,
    'S': 19, 'T': 20, 'U': 21, 'V': 22, 'W': 23, 'X': 24,
    'Y': 25, 'Z': 26
})



app = Flask(__name__)
app.secret_key = 'alphabet_quiz'



@app.route('/')
def index():
    session.clear()
```

```
    return render_template("index.html")

@app.route('/add-data', methods=['GET'])
def add_data_get():
    message = session.get('message', '')

    # labels = np.load('data/labels.npy')
    # count = {k: 0 for k in ENCODER.keys()}
    # for label in labels:
    #     count[label] += 1
    # count = sorted(count.items(), key=lambda x: x[1])
    # letter = count[0][0]

    letter = choice(list(ENCODER.keys()))

    return render_template("addData.html", letter=letter, message=message)

@app.route('/add-data', methods=['POST'])
def add_data_post():

    label = request.form['letter']
    labels = np.load('data/labels.npy')
    labels = np.append(labels, label)
    np.save('data/labels.npy', labels)

    pixels = request.form['pixels']
    pixels = pixels.split(',')
    img = np.array(pixels).astype(float).reshape(1, 50, 50)
    imgs = np.load('data/images.npy')
    imgs = np.vstack([imgs, img])
```

```
    np.save('data/images.npy', imgs)

    session['message'] = f'"{label}" added to the training dataset'

    return redirect(url_for('add_data_get'))

@app.route('/practice', methods=['GET'])
def practice_get():
    letter = choice(list(ENCODER.keys()))
    return render_template("practice.html", letter=letter, correct='')

@app.route('/practice', methods=['POST'])
def practice_post():
    try:
        letter = request.form['letter']

        pixels = request.form['pixels']
        pixels = pixels.split(',')
        img = np.array(pixels).astype(float).reshape(1, 50, 50, 1)

        model = keras.models.load_model('letter.model')

        pred_letter = np.argmax(model.predict(img), axis=-1)
        pred_letter = ENCODER.inverse[pred_letter[0]]

        correct = 'yes' if pred_letter == letter else 'no'
        letter = choice(list(ENCODER.keys()))

        return render_template("practice.html", letter=letter, correct=correct)
```

```
    except Exception as e:
        print(e)
        return render_template('error.html')


if __name__ == '__main__':
    app.run(debug=True)
```