МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ДМИТРУК МАКСИМ АНДРІЙОВИЧ (_____)

Допускається до захисту:
в.о. завідувача кафедри прикладної
математики та кібербезпеки,
д-р. філ. з матем., старший викладач
_____ Алла ЛУЦЕНКО
«___» _____2024 р.

ЗАСТОСУВАННЯ МАТЕМАТИКИ ДЛЯ ДОСЯГНЕННЯ НАЙБІЛЬШОЇ ЕФЕКТИВНОСТІ НЕЙРОННИХ МЕРЕЖ ТА МІНІМІЗАЦІЇ ПОТРІБНИХ РЕСУРСІВ КОМП'ЮТЕРА

Спеціальність 113 Прикладна математика

Кваліфікаційна (бакалаврська) робота

Керівник:
Оксана ДАНИЛЬЧУК, доцент кафедри
прикладної математики та кібербезпеки,
к. пед. наук, доцент _____
(підпис)

Оцінка _____/_____/_____
(бали за шкалою ЄКТС/ національною шкалою)
Голова ЕК:_____
(підпис)

Вінниця – 2024

АНОТАЦІЯ

Дмитрук М.А перед Спеціальність 113 «Прикладна математика», освітня програма «Прикладна математика». Донецький національний університет імені Василя Стуса, Вінниця, 2024.

У кваліфікаційній (бакалаврській) роботі представлені описи архітектур та моделей нейронних мереж та був проведений аналіз різних архітектур з ціллю досягнення найбільшої ефективності ШІ. Також в представлена математична складова яку використовують при застосуванні ШІ.

Ключові слова: математика, штучний інтелект, нейронні мережі, аналіз.

93 с., 5 табл., 24 рис., 8 дод., 36 джерел.

Рис. 24. Таблиць 5. Бібліограф.: 36 найм.

ABSTRACT

Dmytruk M.A. Application of mathematics to achieve the greatest efficiency of neural networks and minimize the required computer resources. Specialty 113 «Applied Mathematics», educational program "Applied Mathematics". Vasyl’ Stus Donetsk National University, Vinnytsia, 2024.

In the qualifying (bachelor's) thesis, descriptions of architectures and models of neural networks are presented, and an analysis of various architectures was carried out with the aim of achieving the highest efficiency of artificial intelligence. The mathematical component used in the application of artificial intelligence is also presented.

Keywords: mathematics, artificial intelligence, neural networks, analysis.

93 pp., 5 table., 24 fig., 8 applications, 36 items

Fig. 24. Table 5. Bibliography: 36 items

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі сфера ШІ (ШІ) пропонує нам перед собою нескінченні можливості, перетворюючи наші уявлення про те, що може здійснити людський розум. Штучний інтелект, здатний до навчання, самонавчання та розв’язання завдань, раніше заданих виключно людськими, тепер переплітається з усіма сферами нашого життя – від медицини та фінансів до транспорту та мистецтва.

Сучасні організації мають доступ до величезних масивів даних, що потребує ефективного аналізу та обробки. Вибір оптимальної моделі та параметрів машинного навчання може значно підвищити ефективність роботи з цими даними.

Ця кваліфікаційна (бакалаврська) робота присвячена вивченню математичних основ ШІ та їх застосуванню у ключових областях, таких як машинне навчання, глибинне навчання, обробка природної мови та комп'ютерний зір. Вона спрямована на аналіз важливості математики як основи розвитку та застосування інтелектуальних систем, а також на розгляд конкретних математичних методів, що лежать в основі сучасних алгоритмів ШІ.

Новизна. Прогрес у глибокому навчанні, зокрема у нейронних мережах, вимагає нових підходів до їх оцінки та оптимізації, оскільки ці моделі мають велику кількість параметрів та складну структуру.

Мета кваліфікаційної (бакалаврської) роботи полягає в оцінці ефективності різних моделей машинного навчання при варіації їх параметрів для визначення оптимальних конфігурацій, що забезпечують найкращу продуктивність.

Для досягнення даної цілі нам необхідно розглянути та розв’язати наступні ***завдання дослідження***:

- зробити підбір даних, які будуть використані для навчання моделей;
- зробити вибір найбільш популярних архітектур нейронних мереж;
- експериментально зробити аналіз різних архітектур та параметрів у моделях нейронних мереж;

- оцінити результати та визначення залежності між продуктивністю та архітектурою мереж.

Об'єктом дослідження є аналіз моделей нейронних мереж.

Предметом дослідження є аналіз найкращих параметрів та моделей для класифікації об'єктів за зображенням.

У вступі до кваліфікаційної (бакалаврської) роботи розглянемо огляд сфери ШІ, його історію, розвиток та сучасний стан, зосереджуючись на значенні математики як важливого інструменту для розуміння та розвитку ШІ ШІ. Далі розглянемо основні математичні концепції, такі як лінійна алгебра, математичний аналіз, теорія ймовірностей, статистика, теорія інформації, та їх застосування в штучному інтелекті. Після цього звернемо увагу на конкретні області застосування математики в ШІ, такі як машинне навчання, глибинне навчання, обробка природної мови та комп'ютерний зір, розглядаючи математичні принципи та методи, що лежать в їх основі.

Ця кваліфікаційна (бакалаврська) робота має на меті сприяти глибокому розумінню математичних аспектів ШІ та їх впливу на розвиток інтелектуальних систем, а також висвітленню перспектив подальших досліджень у цій захоплюючій галузі.

Структура роботи: дана робота складається з вступу, трьох розділів, висновків, списку використаних джерел та додатків.

РОЗДІЛ 1. ЗНАЧЕННЯ МАТЕМАТИКИ ДЛЯ РОЗВИТКУ, СУЧАСНИЙ СТАН ТА ЗАСТОСУВАННЯ ШІ

1.1. Лінійна алгебра. Вектори, матриці, операції над ними. Застосування у нейронних мережах та алгоритмах машинного навчання

Лінійна алгебра є однією з ключових математичних галузей, що знаходить широке застосування у сфері ШІ. У цьому пункті розглянемо основні концепції лінійної алгебри, такі як вектори, матриці та операції над ними, і їх важливість для розуміння та розвитку інтелектуальних систем.

Вектори та матриці є основними поняттями лінійної алгебри, які використовуються для представлення даних та параметрів у багатьох алгоритмах ШІ. Операції над векторами та матрицями, такі як додавання, множення на число та знаходження оберненої матриці, є фундаментальними для багатьох операцій у нейронних мережах та алгоритмах машинного навчання.

Застосування лінійної алгебри у нейронних мережах полягає в представленні ваг та вхідних даних у векторній формі, а також у виконанні операцій з цими векторами та матрицями для навчання та передбачення результатів. Наприклад, у процесі навчання нейронних мереж ваги мережі оновлюються за допомогою алгоритмів градієнтного спуску, які базуються на математичних принципах лінійної алгебри. Отже, розуміння основ лінійної алгебри є критичним для розвитку та застосування ШІ у нейронних мережах та алгоритмах машинного навчання.

У 1949 році професор Гарвардського університету Василій Леонтьєв використовував комп’ютер Mark II для аналізу економіки США. Він розділив економіку на 500 секторів та записав лінійні рівняння для кожного сектора. Хоча Mark II не міг вирішити систему 500 рівнянь з 500 невідомими, Леонтьєв змінив проблему до системи 42 рівнянь з 42 невідомими. Програмування комп’ютера Mark II для цих рівнянь зайняло кілька місяців, але пристрій здатний був розв’язати їх за 56 годин. Леонтьєв, який отримав Нобелівську премію з економічних наук у 1973 році, відкрив нову еру математичного моделювання в економіці. З тих пір комп’ютери використовуються для аналізу математичних

моделей у багатьох галузях. Лінійна алгебра стала дуже важливою з появою потужних обчислювальних пристроїв, і комп'ютерні науки тісно пов'язані з лінійною алгеброю через зростання потужності обчислювальних систем. Вчені та інженери працюють над проблемами, які були немислимі кілька десятиліть тому, і лінійна алгебра має великий потенціал для здобувачів у багатьох галузях науки та бізнесу.

Лінійне рівняння зі змінними $x_1, x_2, \ldots, x_n$ – це рівняння, яке можна записати у вигляді:

$$a_1x_1 + a_2x_2 + \cdots + a_nx_n = b$$

де b та коефіцієнти a_1, $a_2, \ldots, a_n$ є дійсними або комплексними числами, зазвичай відомими заздалегідь. Підпис *n* може бути будь-яким позитивним цілим числом. У вправах підручника *n* зазвичай знаходиться в діапазоні від 2 до 5. У реальних проблемах n може бути 50 або 5000, або навіть більше.

Лінійна система рівнянь представляє собою набір однорідних лінійних рівнянь, що містять ті ж змінні. Розглянемо наступний приклад:

$$2x_1 - 5x_2 - 4{,}5x_3 = 12$$
$$x_2 + 3x_3 = 9$$
$$x_1 + 2x_2 = 6$$

Розв'язок лінійної системи представляє собою набір числових значень $s_1, s_2, \ldots, s_n$, які задовольняють усім рівнянням системи при підстановці цих значень замість відповідних змінних $x_1, x_2, \ldots, x_n$. Наприклад, набір {9.6, -1.8, 3.6} є розв'язком системи вище, оскільки при підстановці цих значень у рівняння перетворюються на тотожності

$$19{,}2 + 9 - 16{,}2 = 12$$
$$-1{,}8 + 10{,}8 = 9$$
$$9{,}6 - 3{,}6 = 6$$

Множина усіх можливих розв'язків лінійної системи називається множина розв'язків системи. Система лінійних рівнянь може мати:

- жодного розв'язку;
- тільки один розв'язок;

- нескінченну кількість розв'язків.

Систему лінійних рівнянь вважають сумісною, якщо вона має або один розв'язок, або нескінченну кількість розв'язків, система вважається несумісною, якщо вона не має розв'язків [1].

Основна інформація про лінійну систему може бути компактно представлена у вигляді прямокутної матриці. Розглянемо систему:

$$3x_1 - x_2 + 4x_3 = 7$$
$$2x_2 - 4x_3 = 0$$
$$3x_1 - 3x_3 = 10$$

Матриця коефіцієнтів змінних виглядає так:

$$\begin{bmatrix} 3 & -1 & 4 \\ 0 & 2 & -4 \\ 3 & 0 & -3 \end{bmatrix}$$

називається коефіцієнтною матрицею (або матрицею коефіцієнтів), а матриця

$$\begin{bmatrix} 3 & -1 & 4 & 7 \\ 0 & 2 & -4 & 0 \\ 3 & 0 & -3 & 10 \end{bmatrix}$$

називається розширеною матрицею системи. Розширена матриця системи складається з коефіцієнтної матриці з доданим стовпцем, що містить константи з правих частин рівнянь [1].

Сумою матриць A та *B* розміром *m* на *n* є матриця *A+B* розміром *m* на *n*, де кожен елемент представляє собою суму відповідних елементів матриць *A* та *B*, де *m* це кількість стовбців, а *n* це кількість рядків, тобто: $A + B = (a_{ij} + b_{ij})_{m \times n}$.

Добутком матриці A розміром *m* на *n* на дійсне число *b* є матриця *A*, такого ж розміру *m* на *n*, де кожен елемент представляє собою добуток відповідного елемента матриці *A* на число *b*, тобто: $bA = (b * a_{ij})_{m \times n}$.

Якщо кількість стовпців матриці *A* відповідає кількості рядків у матриці *B*, то матриці вважаються узгодженими.

Добуток матриць запроваджують лише для узгоджених матриць. Добутком рядка $\vec{x} = (x_j)_n$ завдовжки n на стовпець $\vec{y} = (y_i)_n$ заввишки n

називають число, яке дорівнює сумі добутків елементів рядка на відповідні елементи стовпця, тобто

$$\overleftarrow{x} * \vec{y} = (x_1\ x_2\ x_3 \dots\ x_4) * \begin{matrix} y_1 \\ y_1 \\ \cdot \\ \cdot \\ y_n \end{matrix} = x_1y_1 + x_2y_2 \dots x_ny_n$$

Добутком матриці $A_{m\times \mathrm{l}}$ на матрицю $B_{l\times \mathrm{n}}$ називають матрицю $C = AB$ розміром $m\times n$, кожний елемент c_{ij} якої дорівнює добуткові i -го рядка матриці A на j -й стовпець матриці B, тобто

$$C_{m\times n} = A_{m\times l} * B_{l\times n} = (c_{ij})_{m\times n} = (\overleftarrow{a}_i * \vec{b}_j)_{m\times n}$$

Заміну рядків матриці на її стовпці, а стовпців – на рядки, називають *транспонуванням матриці* і позначають її як $\overleftarrow{x} \overset{T}{\to} \vec{x}$ або $\vec{x} \overset{T}{\to} \overleftarrow{x}$

Матрицю розміром $n \times m$, яку одержують з матриці A розміром $m \times n$ транспонуванням стовпців (рядків), називають *транспонованою матрицею* до А і позначають A^T.

Обернення матриця, ділення для матриць не запроваджують, але для квадратних матриць можна побудувати аналог ділення – множення на обернену матрицю. Оберненою матрицею до квадратної матриці A порядку n називають матрицю A^{-1} таку, що

$$A^{-1}A = AA^{-1} = E_n$$

Матрицю A, для якої існує обернена матриця, називають *оберненою*. З означення випливає, що матриці A та A^{-1} взаємо обернені й переставні [2].

Для того, щоб перейти до застосування лінійної алгебри у алгоритмах машинного навчання потрібно розглянути найбільш популярні базові визначення у цій сфері.

Нейронні мережі:

- нейронні мережі – це клас моделей машинного навчання, натхненних структурою та функціонуванням біологічних нейронних мереж;
- Основними компонентами нейронних мереж є:

- нейрони – обчислювальні одиниці, які отримують вхідні сигнали, виконують над ними певні операції та генерують вихідний сигнал;
- ваги – числові параметри, що визначають силу зв'язків між нейронами. Ваги налаштовуються під час навчання нейронної мережі;
- активаційні функції – функції, що застосовуються до суми зважених вхідних сигналів нейрона для визначення його вихідного сигналу.

Навчання моделей:

- Навчання моделей – процес налаштування параметрів моделі (наприклад, ваг нейронної мережі) на основі навчальних даних.
- Алгоритми навчання, такі як градієнтний спуск, використовують математичні принципи, зокрема, обчислення градієнтів, для оптимізації параметрів моделі.
- Регуляризація – техніки, що застосовуються для запобігання перенавчання моделі та покращення її узагальнюючої здатності.

Задачі машинного навчання:

- Класифікація – задача, в якій модель має визначити, до якого класу належить даний об'єкт.
- Регресія – задача, в якій модель має передбачити числове значення цільової змінної.
- Кластеризація – задача, в якій модель має згрупувати об'єкти в кластери на основі їх схожості.

У нейрона з одним входом вхідними параметрами є скалярні значення входу p, які множаться на коефіцієнт ваги w, та число 1, яке множиться на значення зсуву b. Ці значення формують вхідні сигнали для суматора. Вихід суматора, що є входом мережі, передається функції активації, яка формує вихід нейрона у вигляді скалярного значення y. Ця функція активації може бути як лінійною, так і нелінійною (рис. 1.1).

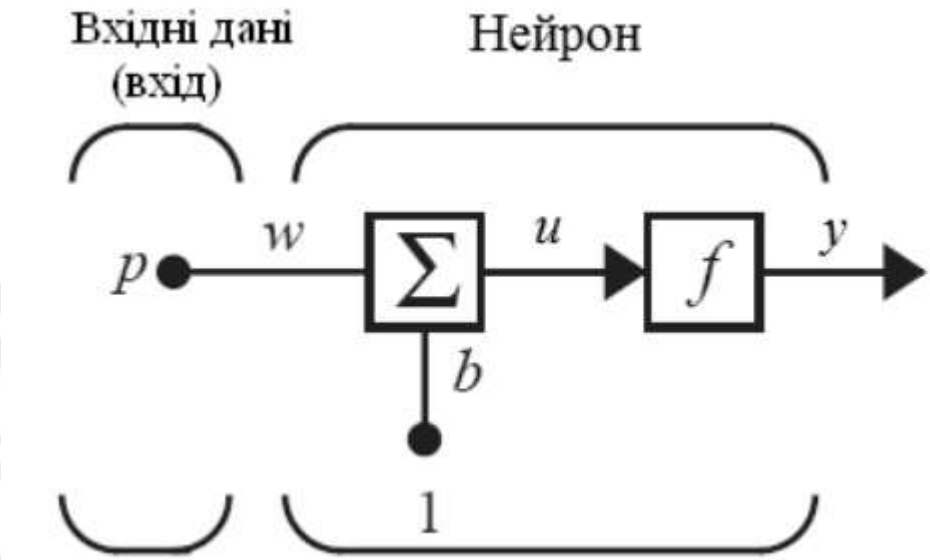


Рис. 1.1 – Модель нейрона з одним входом: $y = f(wp + b)$

Вихід нейрона обчислюють за формулою $y = f(wp + b)$. Зазвичай скалярні параметри *w* і *b* налаштовуються на вхідні дані задачі згідно з певним навчальним алгоритмом, щоб нейрон, який відображає залежність між вхідними та вихідними даними, досягнув певної поставленої мети.

Нейрон із декількома входами. У більшості випадків нейрон має більше одного входу. Модель нейрона із *R* входами зображено на рис. 1.2

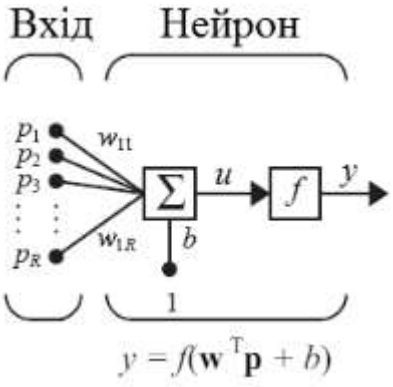


Рис. 1.2 – Модель нейрона із *R* входами

Користувач мережі задає вектор входу $p = [p_1 \dots p_R]^T$, якому відповідає вектор вагових коефіцієнтів $=w = [w_{11} \dots w_{1R}]^T$. Зсув нейрона *b* підсумовується зі зваженими вхідними елементами та формує вхід мережі *u*:

$$u = w_{11}p_1 + \cdots + w_{1R}p_R + b$$

Цей вираз можна записати у матричній формі: $u = w^T p + b$, а вихід нейрона [3]:

$$y = f(w^T p + b).$$

Лінійна алгебра в машинному навчанні використовується для моделювання взаємозв'язків між ознаками та прогнозування вихідних змінних. Вона забезпечує основні методи для оптимізації параметрів моделей, обчислення відстаней між об'єктами та кластеризації, а також розкладання матриць для зменшення розмірності даних. У більш загальному сенсі, лінійна алгебра становить основу для багатьох алгоритмів та методів у машинному навчанні, допомагаючи аналізувати, обробляти та використовувати дані для рішення різноманітних завдань.

1.2 Математичний аналіз. Функції, похідні, інтеграли. Значення для оптимізації в штучному інтелекті

Часто увага акцентується на важливості лінійної алгебри, яка безперечно відіграє вирішальну роль у багатьох аспектах цієї галузі. Однак важливо визнати, що лінійна алгебра не єдиний розділ математики, який має велике значення для машинного навчання. Математичний аналіз, зокрема, відіграє ключову роль у розв'язанні багатьох завдань, пов'язаних з оптимізацією, обчисленнями градієнтів, аналізом функцій втрат та статистичним моделюванням. Цей розділ математики надає інструменти для аналізу, розуміння та оптимізації складних моделей, що розвиваються в машинному навчанні. У цьому пункті ми розглянемо важливість математичного аналізу для машинного навчання, проаналізуємо його основні концепції та покажемо, як він впливає на розвиток цієї сфери.

Розглянемо базові визначення, Функція – це правило, за яким кожному значенню x з певної множини відповідає єдине значення змінної y. Це можна записати у вигляді $y = f(x)$. Змінну x називають незалежною змінною або аргументом, а змінну y – залежною змінною або значенням функції. Множину всіх можливих значень незалежної змінної x називають областю визначення функції. Множину всіх можливих значень залежної змінної y називають областю значень функції або просто множиною значень функції [4].

Похідна є основним поняттям диференціального числення, що визначає швидкість зміни функції. Вона визначається як границя відношення приросту

значення функції до приросту її аргументу, коли приріст аргументу збільшується до нуля (якщо така границя існує). Функція, яка має скінченну похідну, називається диференційованою. Процес знаходження похідної функції називається диференціюванням. Зворотним процесом є інтегрування – процес знаходження первісної функції.

Означення похідної полягає в тому, що, якщо взяти довільне число x в деякому околі точки x_0, то приріст аргументу (позначений як Δx) визначається як $x - x_0$, а приріст функції (Δy) як $f(x) - f(x_0)$**.** Тоді, якщо існує $\lim\limits_{\Delta x \to 0} \frac{\Delta y}{\Delta x}$, то вона називається похідною функції f в точці x_0 [5].

Для функції $f: R^n \to R, x \to f(x), x \in R^n$, часткові похідні для n змінних $x_1, \dots, x_n$ визначаються як:

$$\frac{\partial f}{\partial x_1} = \lim_{h \to 0} \frac{f(x_1+h, x_2, \dots, x_n) - f(x)}{h} \qquad \frac{\partial f}{\partial x_n} = \lim_{h \to 0} \frac{f(x_1+h, \dots, x_{n-1}, x_n) - f(x)}{h}$$

У машинному навчанні похідні використовуються для розв'язання різноманітних задач. Ось декілька основних використань похідних у машинному навчанні:

1. *Оптимізація моделей.* Похідні використовуються для оптимізації параметрів моделей шляхом знаходження мінімуму (або максимуму) функції втрат. Наприклад, метод градієнтного спуску використовує градієнт функції втрат для оновлення параметрів моделі в напрямку мінімізації функції втрат.

2. *Адаптивні методи навчання*. Деякі алгоритми машинного навчання, такі як методи оптимізації з адаптивним кроком (наприклад, алгоритм Adam), використовують інформацію про градієнт для динамічного налаштування кроку навчання.

3. *Регуляризація моделей.* Похідні використовуються для розрахунку градієнту регуляризаційних штрафів, які допомагають уникнути перенавчання та підвищити загальну роботу моделі.

Градієнтний спуск – це ітераційний метод оптимізації першого порядку, в якому для знаходження локального мінімуму функції кроки робляться пропорційно протилежному значенню градієнту (або наближеного градієнту)

функції у поточній точці. Якщо ж кроки здійснюються пропорційно самому значенню градієнту, це призводить до наближення до локального максимуму цієї функції, і така процедура відома як градієнтний підйом. Використовуючи часткові похідні які були описані вище ми можемо представити градієнт як вектор:

$$\nabla_x f = gradf = \frac{df}{dx} = \left[\frac{\partial f(x)}{\partial x_1}, \frac{\partial f(x)}{\partial x_2} \dots \frac{\partial f(x)}{\partial x_n}\right] \in R^{1\times n}$$

де n це кількість змінних , а 1 – розмір області значень / діапазону / область визначення f [6].

Як градієнтний спуск використовується у алгоритмі Adam та відрізняється від класичного стохастичного градієнтного спуску (SGD). Стандартний стохастичний градієнтний спуск зберігає один коефіцієнт навчання (названий alpha) для всіх змін у значеннях ваг та коефіцієнт навчання не змінюється протягом навчання. У методі Adam для кожної ваги мережі (параметра) зберігається окремий коефіцієнт навчання, який адаптується окремо по мірі розвитку моделі (її навчання). Метод обчислює індивідуальні адаптивні коефіцієнти для різних параметрів на основі оцінок перших та других моментів градієнтів.

Автори цього алгоритму описують Adam як поєднання переваг двох інших розширень стохастичного градієнтного спуску. А саме: адаптивний градієнтний алгоритм (AdaGrad), який підтримує коефіцієнт навчання для кожного параметра, що покращує результати на проблемах з розрідженими градієнтами (наприклад, проблеми обробки природної мови та комп'ютерного зору).

Метод розповсюдження кореня середнього квадрату градієнта (RMSProp), який також підтримує коефіцієнти навчання для кожного параметра, які адаптуються на основі середнього значення останніх величин градієнтів ваг (наприклад, як швидко вони змінюються).

Adam реалізує переваги як AdaGrad, так і RMSProp. Замість адаптації коефіцієнтів навчання параметрів на основі середнього (mean) першого моменту як у RMSProp, Adam також використовує середнє значення другого моменту

градієнтів (ненормовану дисперсію). Конкретно, алгоритм обчислює експоненціальне згладження середнього значення градієнту та квадрату градієнту, і параметри beta1 та beta2 контролюють швидкість згасання цих згладжених середніх. Початкове значення згладжених середніх та значення beta1 та beta2 близькі до 1.0 (вони ж і є рекомендованими) призводять до зміщення оцінок моменту до нуля. Це викривлення подолано шляхом спочатку обчислення зміщених оцінок, а потім обчислюючи оцінки з поправкою на зміщення [7].

У математиці, статистиці та задачах машинного навчання, регуляризація використовується для введення додаткової інформації з метою знаходження рішення некоректно поставленої задачі або для уникнення перенавчання. У задачах класифікації застосовують регуляризацію. Під час емпіричного навчання класифікаторів на обмеженому наборі даних завжди виникає недостатньо визначена задача, оскільки загалом ми намагаємося вивести функцію для довільного x за декількома заданими прикладами $x_1, x_2, \ldots, x_n$. Загалом, регуляризуючий вираз $R(f)$ додається до функції втрат:

$$\min_f \sum_{i=1}^{n} V(f(\hat{x}_i), \hat{y}_i) + \lambda R(f)$$

де V – функція, що визначає похибку передбачення $f(x)$ для значень y (наприклад, квадрати похибок), а параметр λ визначає важливість доданка для регуляризації. Зазвичай $R(f)$ визначається як штраф за складність функції f. Зокрема, поняття складності включає обмеження на гладкість та на норму векторного простору. Регуляризація може використовуватись як спосіб покращення узагальнення для моделі у машинному навчанні. Основна задача машинного навчання полягає в тому, щоб знайти функцію, сумарна похибка передбачень якої для всіх можливих значень була б мінімальною. Очікувана похибка f_n виражається як:

$$I[f_n] = \int_{X \times Y}^{\infty} V(f_n(x), y) p(x, y) dx dy$$

Без обмежень складності кінцевої функції моделі, модель може навчитися так, що відповідна їй функція проходить через кожну точку наявних даних x_i навіть якщо у даних є помилки. Це може привести до занадто великих помилок

при використанні моделі на нових даних (які не були використані при навчанні). Регуляризація вводить штраф за включення зайвих областей функціонального простору, що використовується для побудови моделі, що в свою чергу може поліпшити загальну точність моделі [8]. Це все лише мала частина машинного навчання в якому математичний аналіз відіграє важливу роль.

1.3 Теорія імовірностей і статистика. Розподіл імовірностей, статистичне методи. Застосування в Байєсових мережах та алгоритмах прогнозування

Теорія імовірностей та статистика відіграють ключову роль у фундаменті машинного навчання. Вони надають математичний апарат для розуміння і моделювання невизначеності, властивої даним, і допомагають виробляти рішення на основі інформації, зібраної з обмежених або неповних даних. Від коректного застосування цих теоретичних підходів залежить якість та ефективність моделей машинного навчання. У цьому розділі ми детально розглянемо основні поняття теорії імовірностей та статистики, які лежать в основі алгоритмів машинного навчання. Ми розглянемо базові статистичні методи для аналізу даних, важливість імовірнісних розподілів у моделюванні, а також вплив невизначеності на процеси прийняття рішень в машинному навчанні. Ось деякі ключові області, де ці концепції є невід'ємною частиною:

1. *Імовірнісні розподіли:* Розподіли імовірностей, такі як нормальний, біноміальний, Пуассона та інші, часто використовуються для моделювання випадкових величин у машинному навчанні. Наприклад, багато алгоритмів передбачення використовують нормальний розподіл для оцінки ймовірностей вихідних результатів.

2. *Статистичні методи:* Статистичні методи, такі як довірчі інтервали, *p*-значення, аналіз дисперсії (ANOVA) та інші, допомагають в оцінці надійності результатів, отриманих з моделей машинного навчання.

3. *Регуляризація:* Це техніка, яка використовується для управління перенавчанням моделей. Вона базується на статистичних методах, таких як L1 та L2 регуляризація, які допомагають знизити складність моделі, збільшуючи її здатність до генералізації на нових даних.

4. *Валідація моделі:* Перехресне затверджування (cross-validation), яка є статистичним методом оцінки ефективності моделі на основі доступних даних, використовується для оцінки, наскільки добре модель може узагальнити на невідомих даних.

5. *Байєсівські методи:* Баєйсівське висновування є основою для багатьох алгоритмів машинного навчання, таких як наївний байєсівський класифікатор та байєсівські мережі. Ці методи дозволяють оновлювати наші вірування на основі нових даних.

6. *Оцінка ризиків:* Статистичні методи можуть бути використані для оцінки ризиків в машинному навчанні, наприклад, для оцінки ризику перенавчання або недооцінки.

Розподіл ймовірностей – це математична функція, яка визначає ймовірності появи різних можливих результатів експерименту. Це математичний опис випадкового явища в термінах його вибіркового простору та ймовірностей подій (підмножин вибіркового простору). Наприклад, якщо *X* використовується для позначення результату підкидання монети («експеримент»), тоді розподіл ймовірностей *X* матиме значення 0,5 (1 із 2 або 1/2) для *X* = голови та 0,5 для *X* = решка (за умови, що монета чесна). Частіше розподіли ймовірностей використовуються для порівняння відносної появи багатьох різних випадкових значень. Розподіли ймовірностей можна визначити різними способами для дискретних або неперервних змінних. Дистрибутивам зі спеціальними властивостями або для особливо важливих додатків даються спеціальні назви [9].

Біноміальний розподіл є фундаментальним дискретним розподілом випадкових величин, який безпосередньо виникає зі схеми повторних незалежних експериментів(випробувань), відомої як схема Бернуллі. Цей розподіл описує кількість випадків, коли певна подія відбувається у серії з n повторних незалежних випробувань, де ймовірність появи цієї події (успіху) в кожному випробуванні стала і дорівнює p. Відповідно, ймовірність непояви події (невдачі) в кожному випробуванні дорівнює $q = 1 - p$. Біноміальний розподіл для дискретної випадкової величини *X*, що приймає цілі значення $m = 0,1,\dots n$,

визначається формулою $P = (X = m) = C_n^m p^m q^{n-m}$, де $n \geq 0$ (кількість випробувань), $1 \leq p$ – ймовірність успіху в одному випробуванні, $q = 1 - p$ – ймовірність невдачі, а C_n^m – біноміальний коефіцієнт.

Перехресне затверджування (cross-validation), також відоме як ротаційне оцінювання (rotation estimation) або позавибіркове тестування (out-of-sample testing), є методикою перевірки моделі для оцінки її здатності узагальнюватися на нові, незалежні дані. Цей метод часто використовується в контексті передбачення, де потрібно визначити, наскільки ефективно модель працює на практиці. У процесі перехресного затверджування модель тренується на відомому наборі даних (тренувальний набір), а потім перевіряється на незалежному наборі даних (випробувальний набір), який не використовувався при її навчанні. Мета цього підходу – виявити можливі проблеми, такі як перенавчання або вибіркове упередження, і зрозуміти, як модель буде працювати на реальних, невідомих даних. Перехресне затверджування включає розбиття набору даних на комбіновані піднабори, проведення аналізу на одному з цих піднаборів (тренувальний набір), а затвердження результатів на іншому (затверджувальний або випробувальний набір). Щоб отримати стабільні результати, зазвичай проводять декілька ітерацій перехресного затверджування з різними комбінаціями розбиття, а результати цих ітерацій комбінують, наприклад, усереднюють. Отже, перехресне затверджування усереднює показники точності прогнозування для отримання більш надійної оцінки продуктивності моделі [10].

Теорема Байєса, також відома як закон Байєса або правило Байєса, є фундаментальним принципом у теорії ймовірностей та статистиці, який дозволяє оновлювати ймовірності подій на основі нової інформації. Наприклад, якщо є інформація про вік особи і ймовірність захворювання на рак, то за допомогою теореми Байєса можна оцінити ймовірність того, що ця особа дійсно має рак. В теоремі ймовірності можуть інтерпретуватися різними способами. У байєсовому підході до статистичних висновків теорема Байєса допомагає оновити суб’єктивні впевненості з урахуванням нових даних. Це є основою байєсової

статистики. Однак, теорема також широко використовується в інших контекстах, де потрібно обчислювати ймовірності з використанням різних джерел інформації [11].

Класифікатор наївного Байєса – це алгоритм навчання з учителем, який часто використовується для класифікаційних завдань, наприклад, для класифікації тексту. Він використовує ймовірнісний підхід для вирішення таких завдань. Наївний Байєс належить до генеративних моделей навчання, що означає, що він моделює розподіл вхідних даних для конкретного класу або категорії. У відмінність від дискримінативних класифікаторів, таких як логістична регресія, наївний Байєс не визначає, які особливості є ключовими для розрізнення між класами. Він працює на основі двох основних припущень: умовна незалежність прогнозуючих змінних та рівний вплив кожної ознаки на результат. Хоча ці припущення не завжди відповідають реальності (наприклад, послідовне слово в тексті може залежати від попереднього), вони роблять задачу класифікації менш витратною з обчислювальної точки зору. Це дозволяє спростити обчислення моделі, але незважаючи на це спрощення, алгоритм наївного Байєса добре справляється з класифікацією, особливо з обмеженими обсягами даних [12].

Байєсова мережа, також відома як мережа переконань, мережа Байєса, байєсова модель або ймовірнісна орієнтована ациклічна графова модель, є графовою моделлю, яка використовує орієнтований ациклічний граф (DAG) для представлення залежностей між випадковими змінними. У таких мережах вершини відповідають випадковим змінним, які можуть бути спостережуваними, латентними або параметрами. Ребра між вершинами представляють умовні залежності між цими змінними. Якщо між двома вершинами немає ребра, це означає, що ці змінні є умовно незалежними одна від одної в контексті інших змінних в мережі. Кожна вершина асоціюється з функцією ймовірності, яка відображає ймовірність цієї змінної при заданих значеннях її «батьківських» змінних. Ця функція може бути представлена у вигляді таблиці, де кожен рядок представляє комбінацію значень «батьківських» змінних, а значення в кожному рядку показує ймовірність або розподіл ймовірності для даної змінної. Хоча

основна структура байєсової мережі є ациклічною і орієнтованою, ідеї, які лежать в її основі, можуть бути застосовані і в інших типах графів, наприклад, в марковських мережах [13].

У цьому пункті ми поверхнево проаналізували теоретичні основи теорії імовірностей та статистики, які є ключовими для розуміння принципів машинного навчання. Ці дисципліни надають математичний апарат для вивчення та моделювання невизначеності, яка є невід'ємною частиною реальних даних. Ми докладно розглянули імовірнісні розподіли і їх роль у моделюванні даних. Це дозволило нам зрозуміти, як різні імовірнісні моделі можуть бути використані для опису різноманітних явищ у наборах даних. Особливий акцент був зроблений на байєсівських методах, таких як теорема Байєса, класифікатор наївного Байєса та байєсові мережі. Хоч у сьогоденні перевага надається нейронним мережам ці методи безумовно мають великий вклад у розвиток машинного навчання та математики у цілому.

1.4 Теорія інформації. Ентропія, застосування в алгоритмах компресії та кодування інформації

Теорія інформації є ключовим компонентом для розуміння, як інформація кодується, передається та обробляється в різних системах та алгоритмах. Вона відіграє важливу роль у різних галузях, включаючи телекомунікації, обчислювальну техніку, статистику та машинне навчання. У цьому розділі ми детально розглянемо ключові поняття теорії інформації, такі як ентропія та взаємна інформація. Ми також розглянемо застосування цих концепцій у алгоритмах компресії та кодування інформації. Кодування та компресія інформації є важливими завданнями у сучасних телекомунікаційних системах та зберіганні даних. Розуміння теорії інформації допомагає розробляти ефективні методи кодування, які дозволяють зберігати та передавати інформацію з мінімальними втратами та використовувати обмежені ресурси більш ефективно.

Клауд Шеннон(Claude Shannon) запозичив визначення ентропії зі статистичної фізики, де ентропія представляє випадковість або невпорядкованість системи. Зокрема, передбачається, що система має набір

можливих стани, в яких він може перебувати, і в даний момент часу існує розподіл ймовірностей за цими станами. Ентропія тоді визначається як:

$$H(S) = \sum_{s \epsilon S} p(s) \log_2 \frac{1}{p(s)}$$

де S – множина можливих станів, а $p(s)$ – ймовірність стану $s \in S$. Це визначення вказує на те, що чим більш рівномірні ймовірності, тим вища ентропія (безлад) і тим більше зміщення ймовірності, чим нижча ентропія, наприклад, якщо ми точно знаємо, в якому стані перебуває система $H(S) = 0$. Можна згадати, що другий закон термодинаміки в основному говорить, що ентропія закритої системи може тільки зростати. У контексті теорії інформації Шеннон просто замінив «стан» на «повідомлення», тому S є набір можливих повідомлень, а $p(s)$ – це ймовірність повідомлення $s \in S$. Шеннон також визначив поняття власної інформації повідомлення як [14]: $i(s) = \log_2 \frac{1}{p(s)}$.

Відносна ентропія є мірою відстані між двома розподілами. У статистиці вона описана як очікуваний логарифм до відношення правдоподібності. Відносна ентропія $D(p||q)$ є мірою неефективність припущення, що розподіл дорівнює Q, коли справжній розподіл дорівнює p. Наприклад, якби ми знали справжній розподіл випадкової змінної, то ми могли б побудувати код із середньою довжиною опису $H(p)$. Якщо натомість ми використали код для розповсюдження q, нам знадобилося б $H(p) + D(p||q)$ бітів у середньому для опису випадкової змінної. Відносна ентропія або відстань Кульбака-Лейблера між двома ймовірнісними масовими функціями $p(x)$ та $q(x)$ визначається як:

$$D(p||q) = \sum_{x \in X} p(x) \log \frac{p(x)}{q(x)} = E_p \log \frac{p(X)}{q(X)}$$

У наведеному вище визначенні ми використовуємо домовленість (на основі неперервності аргументів), що $0 \log \frac{0}{q} = 0$ та $p \log \frac{p}{0} = \infty$. відносна ентропія завжди невід'ємна і дорівнює нуль тоді і тільки тоді, коли $p = q$. Однак це не є істинною відстанню між розподілами, оскільки вона не є симетричною і не задовольняє трикутник нерівності. Тим не менш, часто корисно думати про

відносну ентропію як про «відстань» між розподілами. Взаємна інформація, яка є мірою кількості інформації, яку одна випадкова величина містить про іншу випадкову величину про іншу випадкову величину. Це зменшення невизначеності однієї випадкової змінної завдяки знанню іншої.

Розглянемо дві випадкові величини X та Y зі спільною масовою функцією розподілу ймовірностей $p(x, y)$ та граничними масовими функціями розподілу ймовірностей $p(x)$ і $p(y)$. Взаємна інформація $I(X;Y)$ – це відносна ентропія між спільним розподілом і розподілом добутку $p(x)p(y)$ [15].

$$I(X;Y) = \sum_{x\in X}\sum_{y\in Y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} = D(p(x,y)||p(x)p(y)) =$$

$$= E_{p(x,y)}\log\frac{p(X,Y)}{p(X)p(Y)}$$

Ми можемо переписати визначення взаємної інформації $I(X, Y)$ як

$$I(X;Y) = \sum_{x,y} p(x,y)\log\frac{p(x,y)}{p(x)p(y)} = \sum_{x,y} p(x,y)\log\frac{p(x|y)}{p(x)} =$$

$$= -\sum_{x,y} p(x,y)\log p(x) + \sum_{x,y} p(x,y)\log p(x|y) =$$

$$= -\sum_{x,y} p(x)\log p(x) - (-\sum_{x,y} p(x,y)\log p(x|y)) = H(X) - H(X|Y)$$

Взаємна інформація $I(X;\ Y)$ – це зменшення невизначеності X за рахунок знання Y. За симетрією також випливає, що

$$I(X;Y) = H(Y) - H(Y|X)$$

Саме тому, X знає про Y стільки ж, скільки Y знає про X. Оскільки $H(X,Y) = H(X) + H(Y|X)$, як показано вище, маємо

$$I(X;Y) = H(X) + H(Y) - H(X,Y)$$

$$I(X;X) = H(X) - H(X|X) = H(X)$$

Взаємна інформація випадкової величини з самою собою це ентропія випадкової величини. Це є причиною того, що ентропію іноді називають самоінформацією [16. с.19-20].

Отже, що таке стиснення даних і навіщо воно нам потрібно? Більшість чули про JPEG та MPEG, які є стандартами для представлення зображень, відео та аудіо. Стиснення даних у цих стандартах використовуються для зменшення кількості бітів, необхідних для представлення зображення, відеоряду або музики. Коротко кажучи, стиснення даних – це мистецтво або наука представлення інформації у компактній формі. Ми створюємо ці компактні репрезентації визначаючи та використовуючи структури, які існують у даних. Дані можуть бути символами в текстовому файлі, числами, які є зразками мовних чи графічних сигналів, або послідовностями чисел які згенеровані іншими процесами. Причина, по якій нам потрібне стиснення даних, полягає в тому, що все більше і більше інформації, яку ми генеруємо і використовуємо, знаходиться у цифровому вигляді – у вигляді чисел, представлених байтами даних. А кількість байт, необхідних для представлення мультимедійних даних може бути величезною. Наприклад, для того, щоб представити в цифровому вигляді 1 секунду відео без стиснення (використовуючи формат CCIR 601), нам потрібно більше 20 мегабайт, або 160 мегабіт. Враховуючи кількість секунд у фільмах, якщо ми розглянемо кількість секунд у фільмі, ми легко зрозуміємо, чому нам потрібно стиснення даних або інакше кажучи компресія інформації.

У сучасних системах обробки даних існують два основних методи стиснення: *безвтратне* та *втратне*. *Безвтратне стиснення* використовується для збереження даних без втрати інформації, перетворюючи їх у формат, який займає менше місця, але може бути повністю відновлений. Цей метод застосовується для текстових даних, програмного коду та документів. З іншого боку, *втратне стиснення* використовується для медіа-файлів, таких як зображення, аудіо та відео. Під час цього процесу, частина інформації може бути втрачена для досягнення більшого ступеня стиснення, що може призвести до зниження деталізації або якості, але загальна сприйнятливість залишається на прийнятному рівні.

LZ77 та LZ78 – це два безвтратних алгоритми стиснення даних, опубліковані в роботах Абрахама Лемпеля та Якова Зіва у 1977 та 1978 роках

відповідно. Вони також відомі як LZ1 та LZ2 відповідно. Ці два алгоритми утворюють основу для багатьох варіацій, включаючи LZW, LZSS, LZMA та інші. Окрім їхнього академічного впливу, ці алгоритми лягли в основу кількох всепроникних схем стиснення, включаючи GIF та алгоритм DEFLATE, що використовується в PNG та ZIP. Теоретично обидва вони є словниковими кодерами. LZ77 під час стиснення підтримує ковзне вікно. Пізніше було показано, що це еквівалентно явному словнику, побудованому LZ78, проте вони еквівалентні лише тоді, коли вся дана має бути розкодована. Оскільки LZ77 кодує та декодує з ковзного вікна через раніше побачені символи, розкодування завжди повинно починатися з початку вводу. Концептуально, розкодування LZ78 може дозволяти випадковий доступ до вводу, якщо весь словник відомий наперед. Проте на практиці словник створюється під час кодування та декодування за допомогою створення нової фрази, кожного разу, коли виводиться токен. Алгоритми LZ77 досягають стиснення шляхом заміни повторюваних даних посиланнями на одну копію цих даних, яка вже існує раніше в нестисненому вигляді. Збіг кодується парою чисел, які називаються парою довжина-відстань (Відстань іноді називають зміщенням замість цього). Для виявлення збігів енкодер повинен відстежувати певну кількість останніх даних, таких як останні 2 КБ, 4 КБ або 32 КБ. Структура, в якій зберігаються ці дані, називається ковзним вікном, тому іноді LZ77 називається стисненням з ковзним вікном. Енкодер повинен зберігати ці дані, щоб шукати збіги, а декодер повинен зберігати ці дані, щоб інтерпретувати збіги, на які посилається енкодер. Чим більше ковзне вікно, тим далі назад може шукати енкодер для створення посилань [17].

MPEG-1 – це стандарт для втратного стиснення відео та аудіо. Він призначений для стиснення цифрового відео якості VHS та аудіо CD до приблизно 1,5 Мбіт/с (відповідно стиснення в 26:1 та 6:1), з мінімальною втратою якості, що робить практичними відео CD, цифрове кабельне/супутникове телебачення та цифрове аудіо-радіомовлення (DAB). В наші дні MPEG-1 став найбільш широко сумісним втратним аудіо / відео форматом у світі, і використовується в великій кількості продуктів і технологій. Можливо,

найвідоміша частина стандарту MPEG-1 – це перша версія аудіо-формату MP3, яку він представив. Відео MPEG-1 використовує методи перцептивного стиснення для значного зменшення потрібної швидкості передачі даних у відеопотоці. Він зменшує або повністю видаляє інформацію в певних частотах та областях зображення, які людське око має обмежену здатність повністю сприймати. Він також використовує тимчасові (на проміжку часу) та просторові (по зображенню) подібності, які є загальними для відео, щоб досягти кращого стиснення даних, ніж це було б можливо в іншому випадку. Перед кодуванням відео у формат MPEG-1 простір кольорів перетворюється на Y′CbCr (Y′= Люма, Cb = Хрома синього, Cr = Хрома червоного). Люма (яскравість, роздільна здатність) зберігається окремо від хроми (колір, відтінок, фаза) і навіть додатково розділяється на складові червоного і синього. Хрома також піддискретизується до 4:2:0, що означає, що вона зменшується до половинної роздільної здатності по вертикалі і горизонталі, тобто до однієї чверті кількості зразків, використаних для компоненту люми відео. Це використання вищої роздільної здатності для деяких колірних компонент схоже за концепцією на фільтр зі способом Байєра, який зазвичай використовується для сенсорів у цифрових кольорових камерах. Оскільки людське око набагато чутливе до невеликих змін у яскравості (компонент Y) ніж у кольорі (компоненти Cr і Cb), піддискретизація хроми є дуже ефективним способом зменшення кількості відеоданих, які потрібно стиснути [18].

Поняття ентропії, запропоноване Клодом Шенноном, є важливим інструментом для розуміння і вимірювання кількості інформації в повідомленні. Ентропія може бути використана для оцінки ефективності алгоритмів компресії даних, а також для розробки оптимальних методів кодування, що дозволяють зменшити розмір передаваних даних без втрати інформації. Застосування теорії інформації у сфері компресії даних є важливим, оскільки вона допомагає розробляти ефективні методи стиснення, які забезпечують оптимальне використання доступного простору для зберігання або передачі даних. Багато алгоритмів стиснення, таких як безвтратні алгоритми Хаффмана та LZ77,

ґрунтуються на концепціях теорії інформації для досягнення ефективної компресії. Крім того, теорія інформації має широкі застосування в кодуванні інформації, такому як передача даних через канали зв’язку. Знання про ентропію дозволяє розробляти кодові схеми, які мінімізують ймовірність помилкового розпізнавання сигналів у шумному середовищі та забезпечують надійну передачу даних.

РОЗДІЛ 2. КЛЮЧОВІ ОБЛАСТІ ШТУЧНОГО ІНТЕЛЕКТУ

2.1. Машинне навчання. Навчання з учителем, без учителя, з частковим учителем

Машинне навчання є галуззю, що базується на математичних принципах та алгоритмах, і використовується для створення систем, які можуть автоматично навчатися та покращувати свою ефективність зі зростанням даних та досвіду. Воно стає дедалі більш важливим у сучасному світі і ні для кого не секрет що зараз відбувається справжня гонка у створенні найкращої моделі для різноманітних сфер життя. З кожним днем усе більше створюється нових моделей які здатні прогнозувати погоду, керувати машиною, генерувати фото та відео, класифікувати хвороби та інше.

Кероване навчання, також відоме як навчання з учителем або контрольоване навчання, – це метод у машинному навчанні, де модель тренується на парах даних, що складаються з вхідних об’єктів (наприклад, векторів ознак) та відповідних вихідних значень (також відомих як мітки або керівні сигнали), наданих експертом. Під час тренування модель створює функцію, яка відображає нові вхідні дані на очікувані вихідні значення. Метою є навчити алгоритм узагальнюватися з тренувальних даних на нові, раніше не бачені ситуації, тобто правильно прогнозувати вихідні значення для невідомих вхідних даних. Якість алгоритму вимірюється за допомогою похибки узагальнення, яка вказує, наскільки точно модель прогнозує на невідомих даних.

Щоб розв’язати задачу керованого навчання, потрібно пройти через кілька кроків:

1. *Визначення типу тренувальних прикладів*: Користувач повинен визначити, який тип даних він буде використовувати як тренувальний набір. Наприклад, це може бути рукописний текст, фотографії, аудіофайли тощо.

2. *Збір тренувального набору:* Потрібно зібрати репрезентативний тренувальний набір, що відображає реальне використання моделі. Це включає збір об’єктів входу та відповідних даних виходу.

3. *Визначення ознак входу:* Важливо визначити, які ознаки будуть використовуватися для опису об’єктів входу. Це може бути набір числових характеристик, що описують об’єкт.

4. *Вибір структури моделі та алгоритму навчання:* Після визначення ознак потрібно вибрати структуру моделі та відповідний алгоритм навчання, який буде використовувати ці ознаки для побудови моделі.

5. *Розробка моделі:* Виконання алгоритму навчання на тренувальному наборі. Під час цього кроку може знадобитися налаштування керівних параметрів моделі.

6. *Оцінка точності моделі:* Після навчання моделі її точність оцінюється на випробувальному наборі. Це допомагає визначити, наскільки добре модель узагальнюється на нові дані.

Мультишаровий персептрон (MLP) – це алгоритм навчання з учителем, який навчається функції: $f(\cdot): R^m \rightarrow R^o$ на наборі даних, де m – це кількість вимірів для введення, а o – це кількість вимірів для виведення. Заданий набір ознак $X = x_1, x_2, x_3, \dots, x_m$ та ціль y, він може навчити нелінійний функціональний апроксиматор для класифікації або регресії. Він відрізняється від логістичної регресії тим, що між входом та виходом може бути один або кілька нелінійних шарів, які називаються прихованими шарами. На рисунку 2.1 (додаток А) представлена модель мультишарового пермертрону.

Крайній шар з ліва, відомий як вхідний шар, складається з набору нейронів $\{x_i | x_1, x_2, x_3, \dots, x_m\}$, що представляють вхідні ознаки. Кожен нейрон у прихованому шарі перетворює значення з попереднього шару за допомогою зваженої лінійної суми $w_1x_1 + w_2x_2 + w_3x_3 + \cdots + w_mx_m$ за якою слідує нелінійна функція активації $g(\cdot): R \rightarrow R$ наприклад, гіперболічний тангенс. Вихідний шар отримує значення з останнього прихованого шару та перетворює їх у вихідні значення [19].

Як працює кероване навчання? Дані, які використовуються в керованому навчанні, марковані, тобто вони містять приклади як вхідних даних (називаються ознаками), так і правильних вихідних даних (мітки). Алгоритми аналізують

великий набір цих навчальних пар, щоб зробити висновок про бажане значення вихідних даних, коли їх просять зробити прогноз на нових даних.

Наприклад, уявімо, що ми хочемо навчити модель розпізнавати зображення дерев. Ми надаємо маркований набір даних, що містить багато різних прикладів видів дерев і назви кожного виду. Дозволяємо алгоритму визначити, який набір характеристик належить кожному дереву на основі маркованих вихідних даних. Потім ми можемо протестувати модель, показавши їй зображення дерева і запитавши, який це вид. Якщо модель надає неправильну відповідь, ми можете продовжувати її навчання і налаштовувати параметри з більшою кількістю прикладів, щоб покращити її точність і зменшити помилки. Після того, як модель була навчена і протестована, можна використовувати її для прогнозування невідомих даних на основі попередніх знань, які вона отримала.

Поняття та структура керованогоі некерованого навчання представлена у додатку Б.

*Кероване навчання проти некерованого навчання***.** Основна відмінність між керованим навчанням і некерованим навчанням полягає у типі вхідних даних, які ви використовуєте. На відміну від некерованих алгоритмів машинного навчання, кероване навчання покладається на марковані тренувальні дані, щоб визначити, чи є розпізнавання шаблонів у наборі даних точним. Цілі моделей керованого навчання також заздалегідь визначені, що означає, що тип вихідних даних моделі вже відомий до застосування алгоритмів. Іншими словами, вхідні дані відображаються на вихідні дані на основі тренувальних даних [20].

2.2 Глибинне навчання. Архітектури нейронних мереж, зокрема, конволюційні та рекуренційні нейронні мережі.

Архітектура нейронних мереж відноситься до структури нейронної мережі або кількості та типів шарів. У цьому розділі ми розглянемо чотири типи нейронних мереж та їх архітектури: нейромережа прямо поширення (Feedforward neural network), рекурентні нейронні мережі(Recurrent neural networks), згорткові нейронні мережі(Convolutional neural networks) та генеративні змагальні

мережі(Generative adversarial networks). Теоретична складова представлено у додатку Г.

Конволюційна нейронна мережа (CNN або ConvNet) - це клас нейронних мереж, що спеціалізується на обробці даних, які мають топологію у вигляді сітки, наприклад, зображення. Цифрове зображення – це двійкове представлення візуальних даних. Воно містить серію пікселів, розташованих у вигляді сітки, які містять значення пікселів, що позначають яскравість і колір кожного пікселя. Людський мозок обробляє величезну кількість інформації в ту ж секунду, коли ми бачимо зображення. Кожен нейрон працює у своєму власному рецептивному полі і з'єднаний з іншими нейронами так, що вони покривають все поле зору. Так само як кожен нейрон реагує на стимули тільки в обмеженій області візуального поля, званої рецептивним полем у біологічній системі зору, кожен нейрон в CNN також обробляє дані тільки в своєму рецептивному полі. Шари розташовані таким чином, що спочатку виявляють простіші шаблони (лінії, криві тощо), а далі більш складні шаблони (обличчя, об'єкти тощо). Використовуючи CNN, можна надати комп'ютерам можливість бачити.

Архітектура конволюційної нейронної мережі. Зазвичай CNN має три шари: конволюційний шар, пулінговий шар і повністю зв'язаний шар рис.2.3.

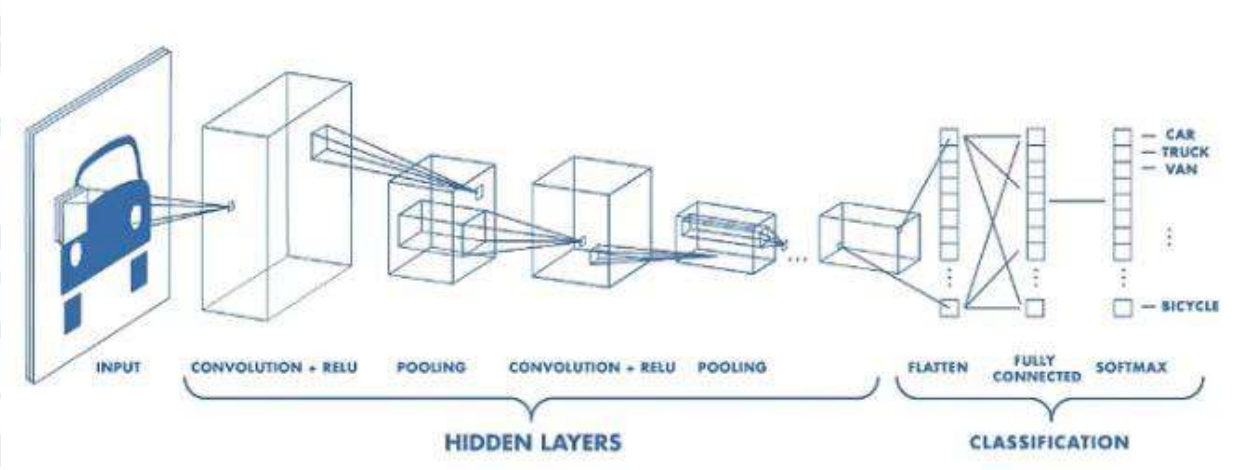


Рис.2.3 - Архітектура конволюційної нейронної мережі [21]

Конволюційний шар є основним будівельним блоком CNN. Він несе основне навантаження обчислень мережі.

Цей шар виконує скалярний добуток між двома матрицями, де одна матриця є набором навчальних параметрів, відомих як ядро, а інша матриця – обмежена частина рецептивного поля. Ядро є просторово меншим за зображення, але має більшу глибину. Це означає, що якщо зображення складається з трьох каналів (RGB), висота і ширина ядра будуть просторово невеликими, але глибина охоплюватиме всі три канали. Під час прямого проходу ядро ковзає по висоті і ширині зображення, створюючи представлення зображення цієї рецептивної області. Це створює двовимірне представлення зображення, відоме як карта активації, яка дає відповідь ядра на кожну просторову позицію зображення. Розмір кроку ядра називається кроком (stride). Якщо у нас є вхід розміром *W x W x D* і Dout кількість ядер із просторовим розміром *F* з кроком *S* і кількістю заповнення *P*, тоді розмір вихідного об'єму можна визначити за такою формулою:

$$W_{out} = \frac{W - F + 2P}{S} + 1$$

Це призведе до вихідного об'єму розміром Wout x Wout x Dout.

Мотивація за конволюційними мережами. Конволюція використовує три важливі ідеї, які мотивували дослідників комп'ютерного зору: розріджену взаємодію(sparse interaction), спільне використання параметрів та еквіваріантне представлення(equivariant representation). Давайте детально розглянемо кожну з них.

Розріджена Взаємодія. Тривіальні шари нейронної мережі використовують множення матриці на матрицю параметрів, що описують взаємодію між вхідними та вихідними одиницями. Це означає, що кожна вихідна одиниця взаємодіє з кожною вхідною одиницею. Проте, у конволюційних нейронних мережах присутня розріджена взаємодія. Це досягається за рахунок того, що ядро є меншим за вхідні дані, наприклад, зображення може містити мільйони або тисячі пікселів, але при обробці його за допомогою ядра ми можемо виявляти значущу інформацію, що складається з десятків або сотень пікселів. Це означає,

що нам потрібно зберігати менше параметрів, що не лише зменшує вимоги до пам’яті моделі, але й підвищує статистичну ефективність моделі.

Спільне Використання Параметрів. Якщо обчислення однієї ознаки в просторовій точці (x_1, y_1) є корисним, то це також має бути корисним в іншій просторовій точці, скажімо (x_2, y_2). Це означає, що для одного двовимірного шару, тобто для створення однієї карти активації, нейрони обмежені використанням одного і того ж набору ваг. У традиційній нейронній мережі кожен елемент матриці ваг використовується один раз і потім більше не використовується, тоді як у конволюційній мережі параметри є спільними, тобто для отримання вихідного сигналу, ваги, застосовані до одного вхідного сигналу, ті ж самі застосовуються й в іншому місці.

Завдяки спільному використанню параметрів, шари конволюційної нейронної мережі матимуть властивість еквіваріантності до трансляції. Це означає, що якщо ми змінимо вхідні дані певним чином, то вихід також зміниться тим же чином. Ці три ключові ідеї допомагають забезпечити ефективність та продуктивність конволюційних нейронних мереж у завданнях обробки зображень та інших даних з топологією сітки.

Пулінговий шар замінює вихідні дані мережі в певних місцях, отримуючи підсумкову статистику з найближчих виходів. Це допомагає зменшити просторовий розмір представлення, що знижує необхідну кількість обчислень та ваг. Операція пулінгу виконується на кожному шарі представлення окремо.

Існує кілька функцій пулінгу, таких як середнє прямокутного околу, L2-норма прямокутного околу та зважене середнє на основі відстані від центрального пікселя. Однак найпопулярнішим процесом є макс-пулінг, який визначає максимальний вихід із околу представлено на рис. 2.4 (додаток Д).

Якщо ми маємо карту активації розміром *W x W x D*, пулінгове ядро просторового розміру *F* та з кроком *S*, тоді розмір вихідного об’єму можна визначити за допомогою наступної формули:

$$W_{out} = \frac{W - F}{S} + 1$$

Це призведе до вихідного об'єму розміром $Wout \times Wout \times D$

У всіх випадках пулінг забезпечує певну інваріантність до трансляцій, що означає, що об’єкт буде розпізнаваним незалежно від того, де він з’являється на кадрі.

Повністю Зв'язаний Шар. Нейрони в цьому шарі мають повну зв'язність з усіма нейронами у попередньому та наступному шарах, як це спостерігається у звичайній повністю зв'язаній нейронній мережі (FCNN). Тому його можна обчислити як зазвичай за допомогою множення матриць, за яким слідує додавання зміщення.

Повністю зв'язаний шар допомагає зіставити представлення між вхідними та вихідними даними.

Нелінійні Шари. Оскільки конволюція є лінійною операцією, а зображення далекі від лінійних, нелінійні шари часто розміщуються безпосередньо після конволюційного шару, щоб ввести нелінійність у карту активації.

Існує кілька типів нелінійних операцій, найпопулярніші з яких:

Сигмоїда. Нелінійність сигмоїди має математичну форму $\sigma(к) = 1/(1+e^{-к})$. Вона бере дійсне число і "стискає" його в діапазон між 0 і 1.

Однак, дуже небажаною властивістю сигмоїди є те, що коли активація знаходиться на будь-якому з кінців, градієнт стає майже нульовим. Якщо локальний градієнт стає дуже малим, то під час зворотного поширення він фактично "вбиває" градієнт. Крім того, якщо дані, що надходять до нейрона, завжди позитивні, то вихід сигмоїди буде або всі позитивні, або всі негативні, що призведе до зигзагоподібної динаміки оновлень ваг.

Tanh. Функція Tanh стискає дійсне число в діапазон [-1, 1]. Як і сигмоїда, активація насичується(saturates), але — на відміну від нейронів сигмоїди — її вихід є центрованим відносно нуля.

ReLU. Останніми роками великої популярності набула виправлена лінійна одиниця ($ReLU$). Вона обчислює функцію $f(к) = max(0, к)$. Іншими словами, активація просто обмежується нулем.

У порівнянні з сигмоїдою та *Tanh*, *ReLU* є більш надійною та прискорює збіжність у шість разів.

На жаль, недоліком *ReLU* є те, що вона може бути вразливою під час навчання. Великий градієнт, що проходить через неї, може оновити її таким чином, що нейрон більше не буде оновлюватись. Однак, з цим можна працювати, встановивши правильну швидкість навчання [24].

Генеративні змагальні мережі. Генеративна змагальна мережа відрізняється від наведених вище моделей тим, що вона насправді є двома окремими мережами. Працюючи як команда, ці два алгоритми генерують новий контент на основі навчальних даних. Одна з цих нейронних мереж, генератор, створює нове зображення або текст на основі навчальних даних. Друга нейронна мережа, дискримінатор, оцінює роботу генератора, щоб визначити, чи виглядає вона реальною або підробленою. Ці дві моделі працюють разом, поки дискримінатор не зможе відрізнити реальні навчальні дані від фальшивої роботи генератора. Генеративні змагальні мережі можуть створювати 3D моделі з 2D зображень, генерувати зображення або створювати навчальні набори даних для інших нейронних мереж, які подібні, але відрізняються від існуючих наборів даних.

Як працює архітектура генеративної змагальної мережі? Основна архітектура генеративної змагальної мережі – це дві окремі нейронні мережі, які працюють у тандемі для створення виходу з входу. В межах цієї категорії нейронних мереж є підтипи, які мають унікальні архітектури, такі як:

Vanilla GAN: Це базова версія генеративної змагальної нейронної мережі, яку потрібно адаптувати для багатьох конкретних реальних застосувань.

CycleGAN: Генеративна змагальна нейронна мережа зі збереженням циклічної послідовності, або CycleGAN, корисна для перекладу зображень, переміщення зображення з однієї області до іншої.

DCGAN: Глибока згорткова генеративна змагальна нейронна мережа, або DCGAN, використовує згорткові нейронні мережі для більш потужної генерації зображень.

Text-2-image: Генеративна змагальна нейронна мережа text-2-image може створювати нові зображення з текстових описів, таких як додавання певного кольору очей до згенерованого обличчя [25].

2.3. Обробка природної мови (NLP). Моделі для аналізу та генерації тексту

Обробка природної мови (ОПМ) є однією з найгарячіших сфер ШІ (ШІ) завдяки застосуванням, таким як генератори тексту, що складають зв'язні есе, чат-боти, які можуть обдурити людей, змусивши їх думати, що вони свідомі, і програми перетворення тексту в зображення, які створюють фото реалістичні зображення будь-чого, що можна описати. Останні роки принесли революцію у здатності комп'ютерів розуміти людські мови, мови програмування і навіть біологічні та хімічні послідовності, такі як ДНК і білкові структури, що нагадують мову. Останні моделі ШІ відкривають ці галузі для аналізу значення вхідного тексту та генерування змістовного, виразного вихідного тексту.

Теоретична складова, чому важлива ОПМ, для чого використовується, як працює представлено у додатку Е.

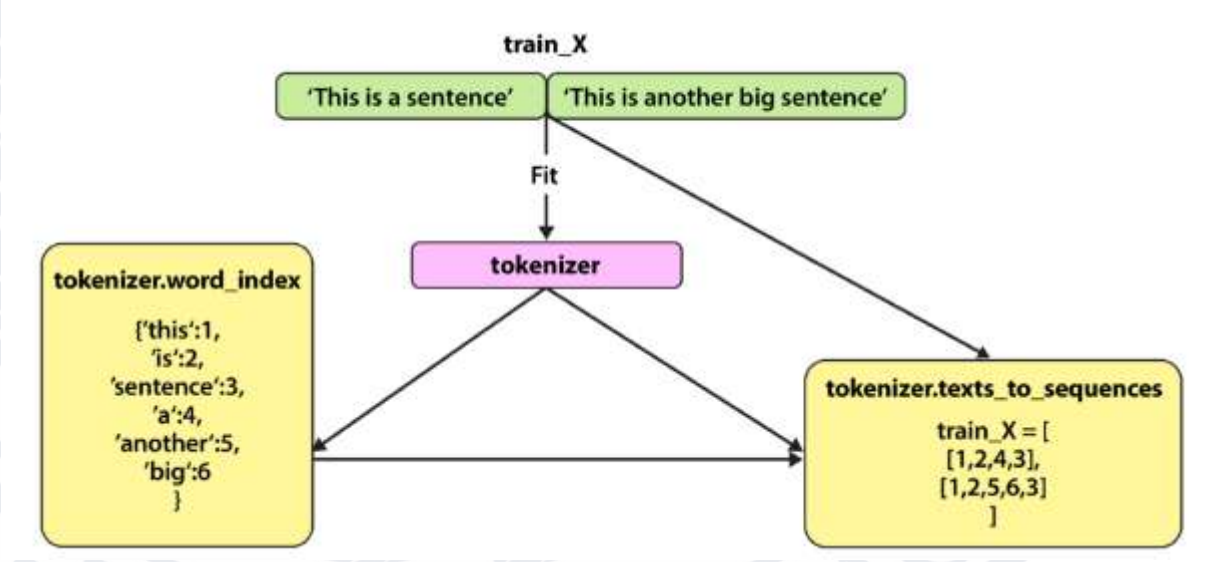


Рис. 2.5 – Приклад токенізації

Виділення ознак: Більшість традиційних технік машинного навчання працюють на основі ознак – зазвичай чисел, які описують документ щодо корпусу, що його містить, – створених за допомогою методу «мішка слів» (Bag-of-Words), TF-IDF(term frequency-inverse document frequency) або загального

виділення ознак, таких як довжина документа, полярність слів і метадані (наприклад, якщо текст має асоційовані теги або оцінки). Більш сучасні техніки включають Word2Vec, GLoVE та навчання ознак під час тренування нейронної мережі. Мішок слів (Bag-of-Words): Мішок слів рахує кількість разів, коли кожне слово або *n*-грам (комбінація з *n* слів) з'являється в документі. Наприклад, нижче модель мішка слів створює числове представлення набору даних на основі того, скільки разів кожне слово з word_index зустрічається в документі.

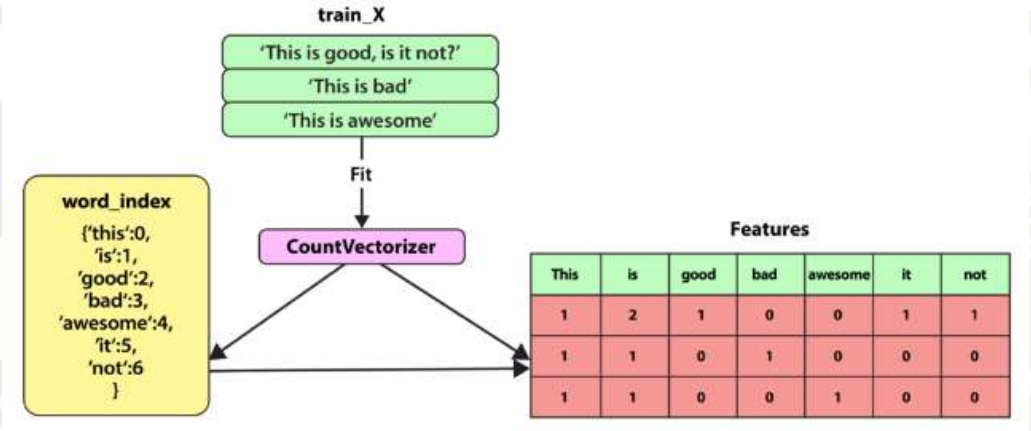


Рис. 2.6 -Приклад, як працює мішок слів

У методі мішка слів ми рахуємо кількість появ кожного слова або n-грам у документі. На противагу цьому, у TF-IDF ми зважуємо кожне слово за його важливістю. Щоб оцінити значущість слова, ми розглядаємо дві речі: Частота терміну(TF): Наскільки важливе слово в документі? TF(слово в документі) = Кількість появ цього слова в документі / Кількість слів у документі Зворотна частота документів(IDF): Наскільки важливий термін у всьому корпусі? IDF(слово в корпусі) = log(кількість документів у корпусі / кількість документів, що містять це слово). Слово є важливим, якщо воно зустрічається багато разів у документі. Але це створює проблему. Такі слова, як «a» і «the», зустрічаються часто. Відповідно, їхній TF бал завжди буде високим. Ми вирішуємо цю проблему, використовуючи зворотну частоту документів (IDF), яка є високою, якщо слово рідкісне, і низькою, якщо слово поширене по всьому корпусу. TF-IDF бал терміну є добутком TF і IDF.

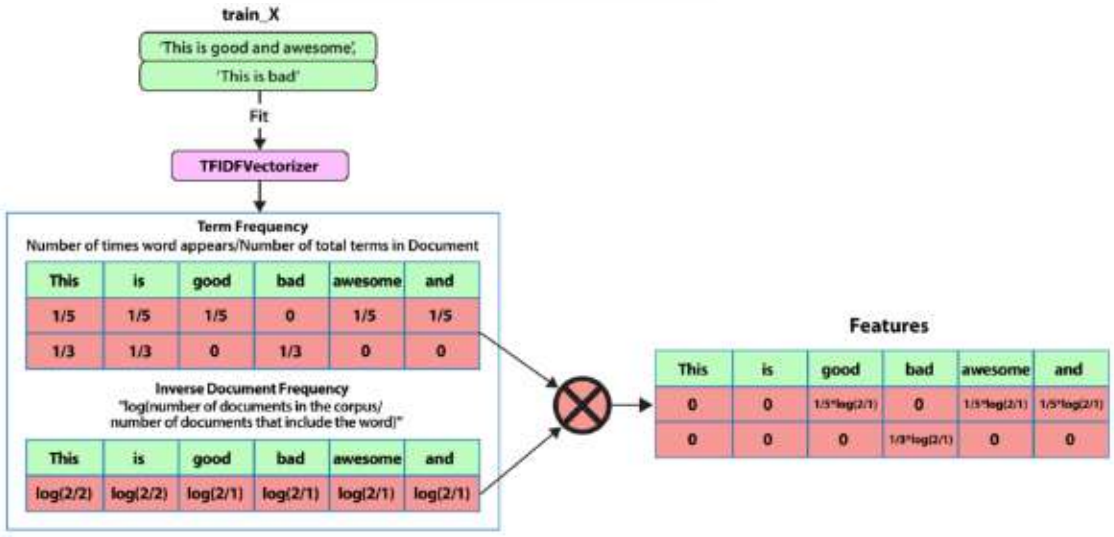


Рис. 2.7 – Приклад обрахунків TF-IDF

Word2Vec, запропонований у 2013 році, використовує звичайну нейронну мережу для навчання багатовимірних слів вбудовуваних із сирого тексту. Він має дві варіації: Skip-Gram, у якій ми намагаємося передбачити оточуючі слова за заданим цільовим словом, і Continuous Bag-of-Words (CBOW), який намагається передбачити цільове слово за оточуючими словами. Після видалення останнього шару після тренування, ці моделі приймають слово на вході і видають вбудовування слова, яке можна використовувати як вхід до багатьох завдань ОПМ. Вбудовування з Word2Vec захоплюють контекст. Якщо певні слова з’являються в схожих контекстах, їх вбудовування буде схожим.

GLoVE подібний до Word2Vec, оскільки також навчає вбудовування слів, але робить це за допомогою технік факторизації матриці, а не нейронного навчання. Модель GLoVE будує матрицю на основі глобальних рахунків співвідношень слів до слів.

Моделювання: Після того, як дані попередньо оброблені, їх передають в архітектуру ОПМ, яка моделює дані для виконання різних завдань.

Числові ознаки, вилучені за допомогою описаних вище технік, можна передавати в різні моделі залежно від завдання. Наприклад, для класифікації, вихід від векторизатора TF-IDF може бути переданий логістичній регресії, наївному Баєсу, деревам рішень або градієнтним підсиленим деревам. Або, для

розпізнавання іменованих сутностей, ми можемо використовувати приховані моделі Маркова разом із *n*-грамами.

Глибокі нейронні мережі зазвичай працюють без використання вилучених ознак, хоча ми все ще можемо використовувати ознаки TF-IDF або Bag-of-Words як вхід.

Мовні моделі(Language Model): У дуже базових термінах мета мовної моделі – передбачити наступне слово, якщо дано потік вхідних слів. Ймовірнісні моделі, які використовують припущення Маркова, – як приклад: $P(Wn) = P(Wn|Wn-1)$, де Wn представляє собою слово у тексті, а $Wn{-}1$ – попереднє слово перед ним. Права частина рівняння, $P(Wn|Wn-1)$, означає умовну ймовірність, що слово Wn з'являється після слова $Wn{-}1$ у тексті Ліва частина рівняння, $P(Wn)$, представляє собою загальну ймовірність появи слова Wn у тексті без будь-яких обмежень чи умов. Отже, рівняння $P(Wn) = P(Wn|Wn-1)$ стверджує, що загальна ймовірність появи слова Wn у тексті дорівнює умовній ймовірності появи слова Wn, якщо перед ним з'явилось слово $Wn{-}1$. Для створення таких мовних моделей також використовують глибоке навчання. Моделі глибокого навчання приймають на вході вбудовування слова і на кожному кроці часу повертають розподіл ймовірностей наступного слова як ймовірність для кожного слова у словнику. Завчасно треновані мовні моделі вивчають структуру першого конкретного мовлення за допомогою обробки великого корпусу, такого як Вікіпедія. Потім їх можна налаштувати для конкретного завдання. Наприклад, BERT було налаштовано для різних завдань, від перевірки фактів до написання заголовків.

Методи обробки природних мов (NLP) можна розділити на дві категорії: традиційні методи машинного навчання та методи глибокого навчання.

Традиційні методи машинного навчання NLP:

Логістична регресія – це алгоритм навчання з учителем, який спрямований на прогнозування ймовірності виникнення події на основі вхідних даних. У NLP моделі логістичної регресії можуть бути застосовані для вирішення завдань,

таких як аналіз настроїв, виявлення спаму та класифікація токсичних повідомлень.

Наївний Байєс – це алгоритм класифікації з учителем, який знаходить умовний розподіл ймовірностей P(label | text) за допомогою формули Байеса та прогнозує на основі того, який спільний розподіл має найвищу ймовірність. Наївне припущення в моделі наївного Баєса полягає в тому, що окремі слова є незалежними. Таким чином:

P(text|label) = P(word_1|label)*P(word_2|label)*…P(word_n|label).

У NLP такі статистичні методи можуть бути застосовані для вирішення завдань, таких як виявлення спаму або пошук помилок у програмному коді.

Дерева рішень – це клас моделей класифікації з учителем, які розбивають набір даних на основі різних ознак для максимізації приросту інформації в цих розбиттях.

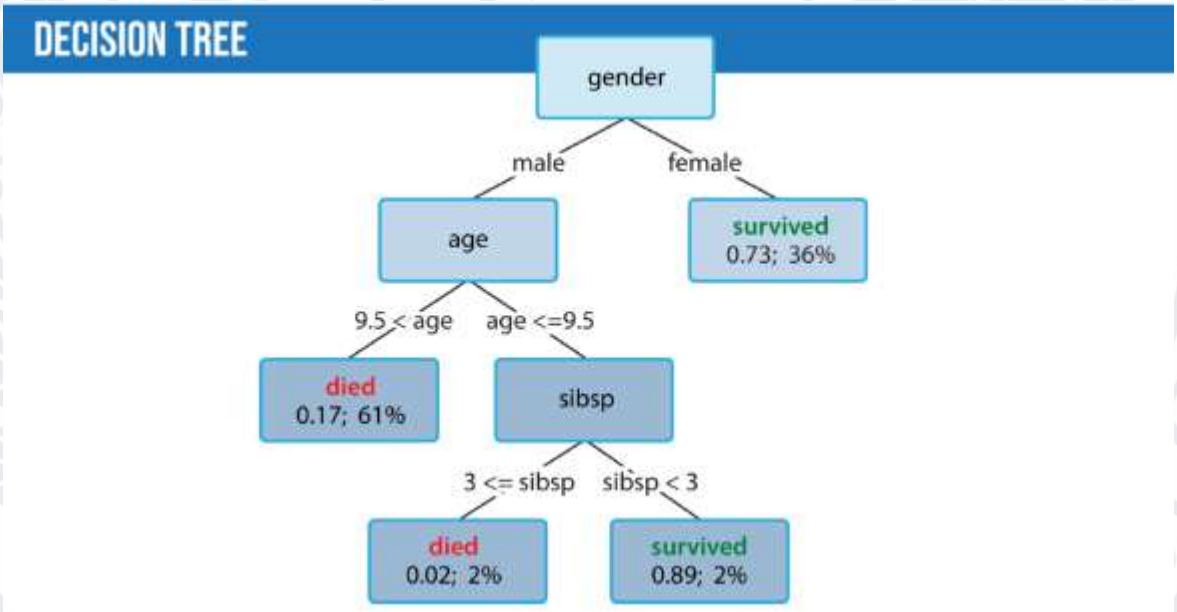


Рис. 2.8 – Приклад дерева рішень

Латентна алокація діріхле (Latent Dirichlet Allocation, LDA) використовується для моделювання тем. LDA намагається розглядати документ як колекцію тем, а тему як колекцію слів. LDA – це статистичний підхід. Інтуїція за ним полягає в тому, що ми можемо описати будь-яку тему за допомогою лише невеликого набору слів із корпусу.

Приховані моделі Маркова: Моделі Маркова – це ймовірнісні моделі, які вирішують наступний стан системи на основі поточного стану. Наприклад, у NLP ми можемо запропонувати наступне слово на основі попереднього слова. Ми можемо моделювати це як модель Маркова, де ми можемо знайти ймовірності переходу від слова1 до слова2, тобто $P(word1|word2)$. Потім ми можемо використовувати добуток цих ймовірностей переходу, щоб знайти ймовірність речення. Прихована модель Маркова (ПММ) – це ймовірнісний метод моделювання, який вводить прихований стан до моделі Маркова. Прихований стан – це властивість даних, яка не спостерігається безпосередньо. ПММ використовуються для маркування частин мови (POS), де слова речення – це спостережувані стани, а POS-теги – це приховані стани. ПММ додає концепцію ймовірності емісії; ймовірність спостереження за умови прихованого стану. У попередньому прикладі це ймовірність слова за умови його POS-тегу. ПММ передбачає, що цю ймовірність можна обернути: знаючи речення, ми можемо розрахувати тег частини мови для кожного слова на основі як ймовірно слово матиме певний тег частини мови, так і ймовірності того, що певний тег частини мови слідує за тегом частини мови, призначеним попередньому слову. На практиці це вирішується за допомогою алгоритму Вітербі.

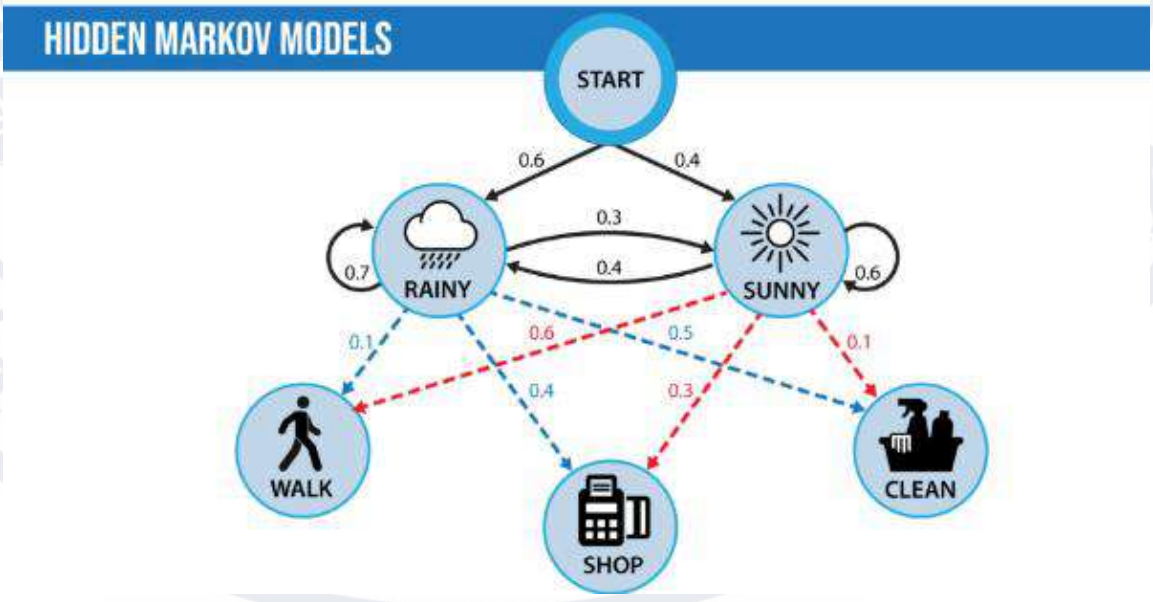


Рис. 2.9 – Приклад моделі Маркова

Техніки глибокого навчання в NLP. Згорткові нейронні мережі (CNN): Ідея використання CNN для класифікації тексту вперше була представлена в роботі «Згорткові нейронні мережі для класифікації речень» Юна Кіма. Центральна інтуїція полягає в тому, що документ сприймається як зображення. Проте, замість пікселів, вхідними є речення або документи, представлені у вигляді матриці слів.

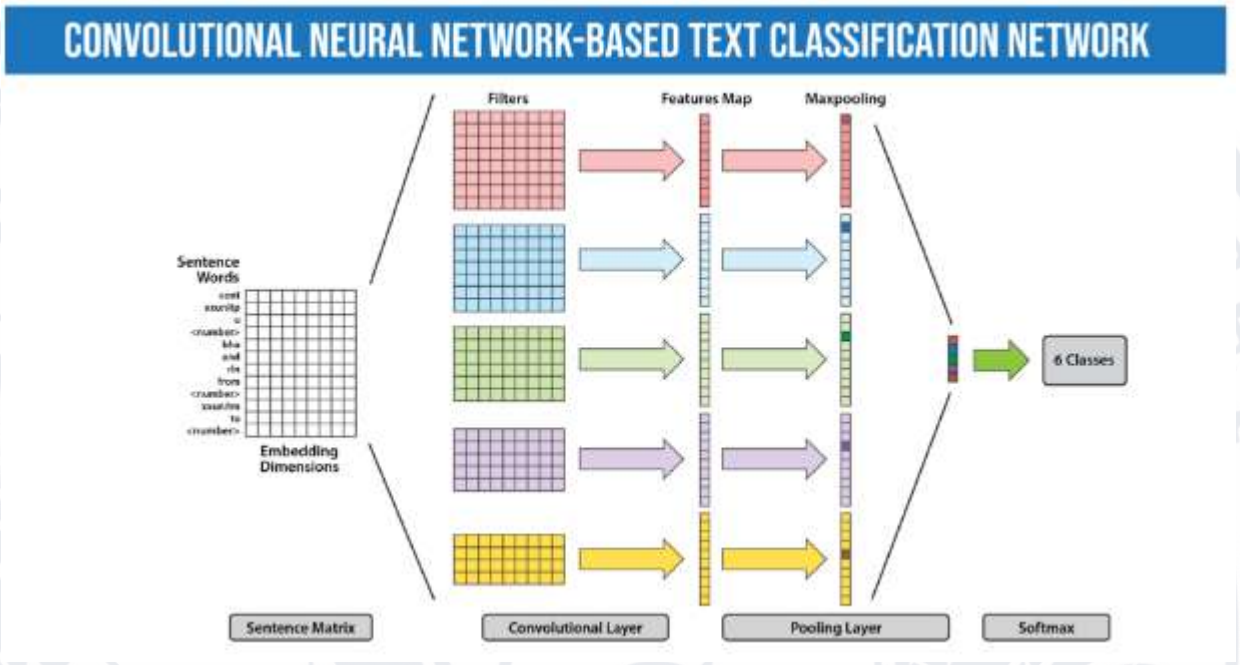


Рис. 2.10 – Конволюційна нейронна мережа для класифікації тексту

Рекурентні нейронні мережі (RNN): Багато технік класифікації текстів, які використовують глибоке навчання, обробляють слова, розташовані поруч, використовуючи *n*-грами або вікно (CNN). Вони можуть бачити «New York» як один екземпляр. Проте вони не можуть захопити контекст, наданий конкретною послідовністю тексту. Вони не вивчають послідовну структуру даних, де кожне слово залежить від попереднього слова або слова у попередньому реченні. RNN запам’ятовують попередню інформацію за допомогою прихованих станів і зв’язують її з поточним завданням. Архітектури, відомі як блоки затворів (GRU) і короткострокова пам’ять (LSTM), є типами RNN, призначеними для запам’ятовування інформації на тривалий період. Крім того, двонаправлені LSTM/GRU зберігають контекстуальну інформацію в обох напрямках, що корисно для класифікації тексту. RNN також використовувалися для генерації математичних доказів та перекладу людських думок у слова.

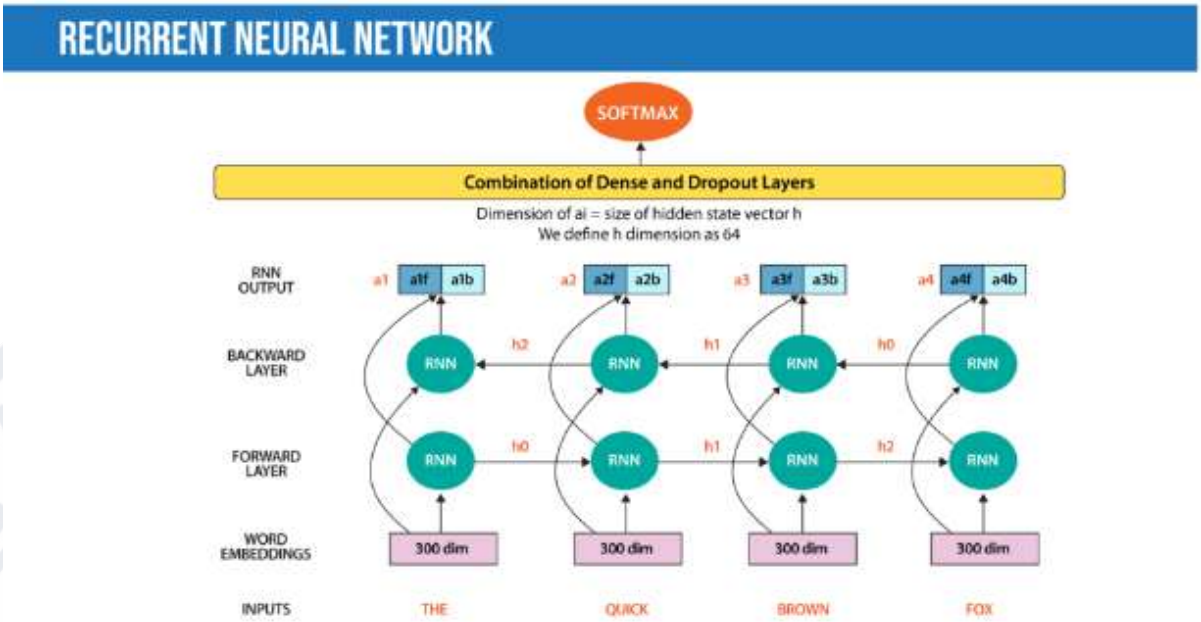


Рис. 2.11 – Рекуренційна нейронна мережа для класифікації тексту

Автокодувальники – це глибокі нейронні мережі кодувальник-декодувальник, які наближають відображення від X до X, тобто вихід=input. Спочатку вони стискають вхідні ознаки у більш низькорозмірне представлення (іноді його називають латентним кодом, латентним вектором або латентним представленням) і вивчають відновлення вхідного сигналу. Вектор представлення може бути використаний як вхід до окремої моделі, тому цю техніку можна використовувати для зменшення розмірності. Серед фахівців у багатьох інших галузях генетики застосовують автокодувальники для виявлення мутацій, пов’язаних з хворобами у послідовностях амінокислот.

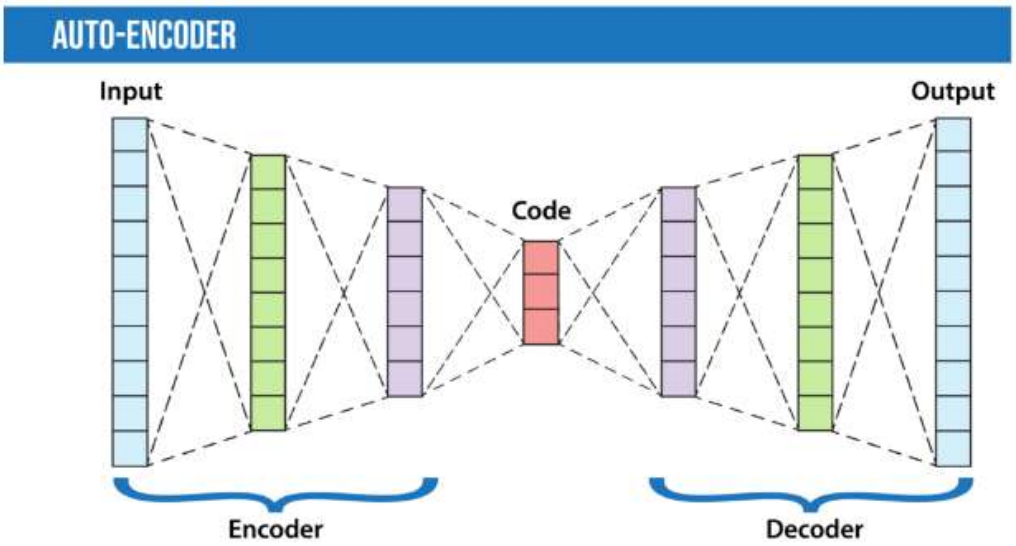


Рис. 2.12 – Приклад архітектури автокодувальника

NLP є однією з найшвидше зростаючих доменів досліджень в галузі ШІ, з застосуваннями, що включають завдання, такі як переклад, узагальнення, генерація тексту та аналіз настроїв. Бізнеси використовують NLP для реалізації зростаючої кількості застосунків, як внутрішні – такі як виявлення шахрайства зі страховками, визначення настрою клієнтів та оптимізація технічного обслуговування повітряних суден – так і з спрямуванням на клієнта, наприклад, Google Translate [26].

РОЗДІЛ 3. ОЦІНКА РІЗНИХ МОДЕЛЕЙ МАШИННОГО НАВЧАННЯ ПРИ ВАРІАЦІЇ ПАРАМЕТРІВ РЕСУРСУ КОМП'ЮТЕРІВ

Зображення є важливим джерелом інформації в сучасному світі, і їх класифікація за допомогою методів машинного навчання має великий потенціал у різних областях, таких як медицина, автоматизоване виробництво, безпека, інтернет-магазини та багато інших. Однак вибір правильної моделі машинного навчання та належне налаштування її параметрів може виявитися складним завданням, особливо у випадку класифікації зображень.

Цей розділ присвячений порівнянню різних моделей машинного навчання, що використовуються для класифікації зображень, з різними параметрами. Мета цього дослідження – визначити, яка модель машинного навчання та які параметри найефективніші для класифікації зображень у нашій конкретній задачі.

Розвиток технологій комп'ютерного зору та зростання обсягів зображень, що генеруються, створює потребу у швидкому та точному класифікуванні зображень. Для досягнення цієї мети розробники використовують різні моделі машинного навчання. Однак вибір найбільш підходящої моделі та оптимальних параметрів для класифікації може бути складною задачею, оскільки він залежить від багатьох факторів, таких як обсяг даних, складність задачі та ресурси, доступні для навчання моделі.

Основною метою цього дослідження є порівняння різних моделей машинного навчання з різними параметрами для класифікації зображень. Крім того, ми прагнемо визначити, які моделі та параметри найефективніше працюють у нашій конкретній задачі класифікації зображень.

Цей розділ організований наступним чином: спочатку розглянемо основні моделі машинного навчання, що використовуються для класифікації зображень, і їхні принципи роботи. Потім перейдемо до експериментального порівняння цих моделей з різними параметрами за допомогою нашого набору даних. На завершення проаналізуємо результати експерименту та зробимо висновки щодо

найефективніших моделей та параметрів для класифікації зображень у нашому контексті.

Опис набору даних CIFAR-10.

CIFAR-10 (Canadian Institute For Advanced Research) – це широко використовуваний набір даних для задач класифікації зображень, розроблений в рамках проекту Tiny Images, що був створений Alex Krizhevsky, Vinod Nair та Geoffrey Hinton. Він містить низькорозмірні зображення, які є корисними для навчання та тестування алгоритмів машинного навчання, особливо нейронних мереж.

Набір даних CIFAR-10 складається з 60,000 кольорових зображень розміром 32x32 пікселів, розподілених по 10 класах. Кожен клас містить по 6,000 зображень. Набір даних розділений на дві частини: тренувальний набір з 50,000 зображень і тестовий набір з 10,000 зображень.

Класи набору даних включають:

1. Літак (airplane)
2. Автомобіль (automobile)
3. Птах (bird)
4. Кіт (cat)
5. Олень (deer)
6. Собака (dog)
7. Жаба (frog)
8. Кінь (horse)
9. Корабель (ship)
10. Вантажівка (truck)

Особливості набору даних:

- *Розмір зображень*: 32x32 пікселів, що робить його зручним для швидкого навчання моделей, навіть на обмежених обчислювальних ресурсах.
- *Кольорові зображення:* Кожне зображення має три канали (RGB), що дозволяє моделювати більш реалістичні сценарії обробки зображень.

- *Рівноважний розподіл класів:* Кожен клас представлений рівною кількістю зображень, що сприяє рівномірному навчанню моделей без перекосів у бік одного з класів.
- *Розбиття на тренувальний та тестовий набори*: Наявність окремого тестового набору дозволяє об’єктивно оцінювати продуктивність моделей після навчання.

Використання в машинному навчанні CIFAR-10 часто використовується для:

- *Навчання і тестування моделей нейронних мереж:* Набір даних ідеально підходить для експериментів з простими та складними моделями, такими як MLP, CNN, та інші.
- *Дослідження нових архітектур:* Вчені та інженери використовують CIFAR-10 для тестування нових архітектур нейронних мереж та порівняння їх ефективності.
- *Навчання на малих даних*: CIFAR-10 є корисним для досліджень, де потрібно швидко тренувати моделі на невеликих обсягах даних.

Аналоги набору даних CIFAR-10

1. **MNIST**

- *Опис*: MNIST (Modified National Institute of Standards and Technology) - це набір даних з 70,000 чорно-білих зображень розміром 28x28 пікселів, що містить рукописні цифри від 0 до 9.
- *Кількість класів:* 10 класів.
- *Призначення:* Використовується для задач розпізнавання рукописних цифр.
- *Особливості*: Простий для візуалізації та швидкого навчання моделей.

2. **Fashion-MNIST**

- *Опис*: Fashion-MNIST є альтернативою MNIST і містить 70,000 зображень розміром 28x28 пікселів, які представляють предмети одягу, такі як футболки, пальто, сумки та взуття.
- *Кількість класів:* 10 класів.

- ***Призначення**:* Використовується для задач класифікації зображень предметів одягу.
- *Особливості:* Більш складний, ніж MNIST, через різноманітність зображень.

3. **CIFAR-100**

- *Опис:* CIFAR-100 є розширеною версією CIFAR-10 і містить 60,000 кольорових зображень розміром 32x32 пікселів, розподілених по 100 класах.
- *Кількість класів:* 100 класів.
- *Призначення:* Використовується для більш детальних класифікаційних задач.
- *Особливості*: Кожен клас містить лише 600 зображень, що робить його більш складним для класифікації.

4. **SVHN (Street View House Numbers)**

- *Опис:* SVHN - це набір даних, що містить понад 600,000 кольорових зображень розміром 32x32 пікселів, які представляють цифри на зображеннях будинкових номерів.
- *Кількість класів:* 10 класів.
- *Призначення:* Використовується для задач розпізнавання цифр у реальному світі.
- *Особливості:* Має варіативність у вигляді освітлення, перспективи і фону, що робить його більш реалістичним для розпізнавання.

Набір даних CIFAR-10 є важливим інструментом для досліджень у сфері машинного навчання. Його збалансованість, компактний розмір та різноманітність класів роблять його ідеальним для експериментів з різними моделями нейронних мереж. У порівнянні з аналогами, CIFAR-10 забезпечує добрий баланс між простотою використання та складністю задачі, що дозволяє його використовувати як початківцям, так і досвідченим дослідникам, саме тому я вирішив використовувати цей набір даних.

Опис наступних моделей нейронних мереж представлено у додатку Ж.

Обираючи моделі для порівняння, я обрав ці п’ять типів нейронних мереж з таких причин:

1. **Простий персептрон**: Ця модель є найбільш базовою та простою для розуміння. Вона дозволить порівняти ефективність більш складних моделей з базовим підходом.
2. **Багатошаровий персептрон (MLP)**: MLP є стандартним вибором для багатьох задач класифікації. Вона може добре пристосовуватися до складних шаблонів у даних.
3. **Згорткова нейронна мережа (CNN)**: CNN є потужним інструментом для обробки зображень. Вона здатна виявляти просторові шаблони та ієрархічні функції у зображеннях.
4. **Рекурентна нейронна мережа (RNN)**: RNN корисна для аналізу послідовних даних, таких як часові ряди або текстові дані. Вона здатна враховувати контекст та залежності між даними.
5. **Довга короткострокова пам'ять (LSTM)**: LSTM є розвиненою версією RNN з здатністю враховувати довгострокові залежності. Вона корисна для задач, де важливі довгострокові залежності між даними.

Обравши ці п’ять різних типів нейронних мереж, ми зможемо провести комплексний аналіз їх ефективності на задачі класифікації зображень та зрозуміти, яка модель працює найкраще для даного набору даних.

Було вирішено провести тестове навчання на 10 епохах, так як вважаю, що це достатня кількість щоб провести дослідження та визначити найкращих кандидатів. Навчання проводилося використовуючи CPU(центральний процесор), щоб уникнути несправедливого порівняння та помилок пов’язаних з обчисленнями і бібліотекою CUDA.

Ці графіки (рис.3.1) були створені за допомогою tensorboard. Відразу можна побачити та підтвердити що CNN виконує те для чого була створена найкраще, найбільше мене здивувала продуктивність lstm хоч він і тривав набагато довше але досягнув кращих результатів ніж багатошаровий персептрон.

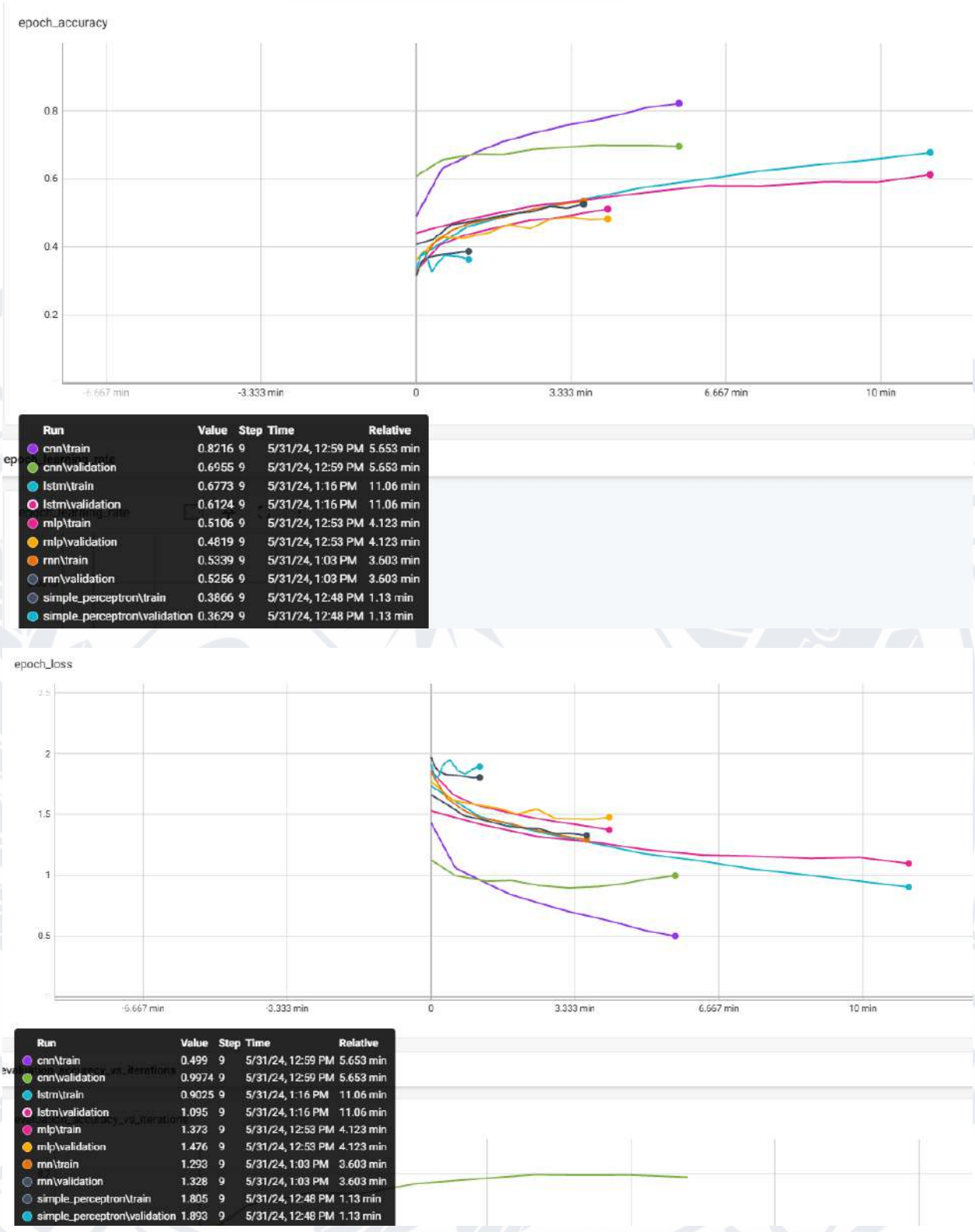


Рис. 3.1 – Результати тестування

Наступним кроком було зібрати дані за допомогою табличного процесора Excel для побудови графіків (таблиця 3.1).

Таблиця 3.1 – Дані по архітектурних моделях

Архітектура моделі	Кількість нейронів до тренування	Кількість пам’яті (KB)	Точність першої епохи	Точність десятої епохи	Час тренування (у секундах)
SLP	30,730	120,04	0,284	0,3854	44
MLP1	1,707,274	6510	0,2803	0,5159	220
CNN1	167,562	654,54	0,3854	0,7959	350
RNN	29,910	116,84	0,2702	0,5365	250
LSTM	160,210	625,82	0,3102	0,6805	760

Рис. 3.2 – Графік кількості нейронів

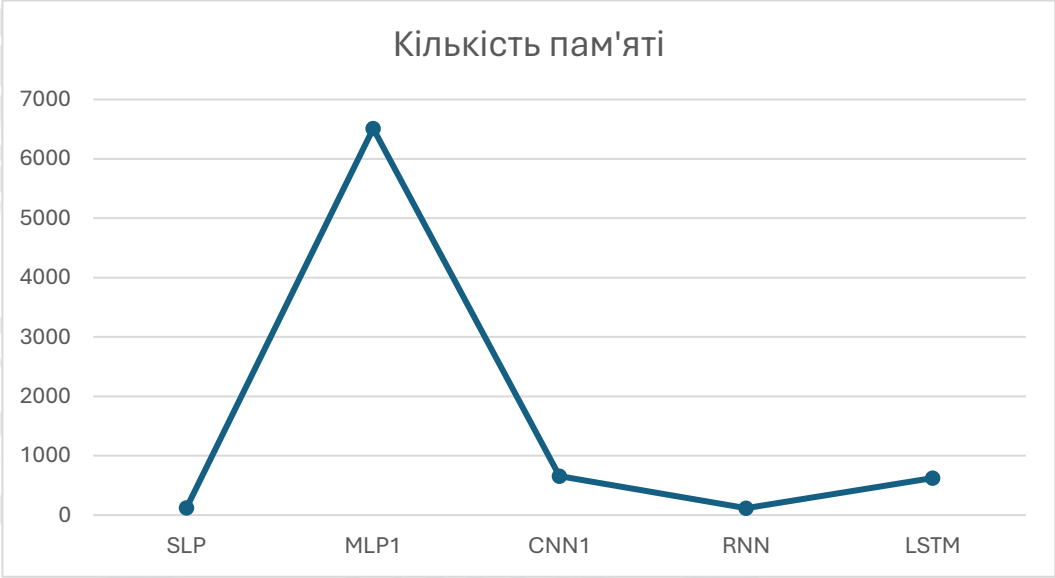


Рис. 3.3 – Графік кількості пам’яті

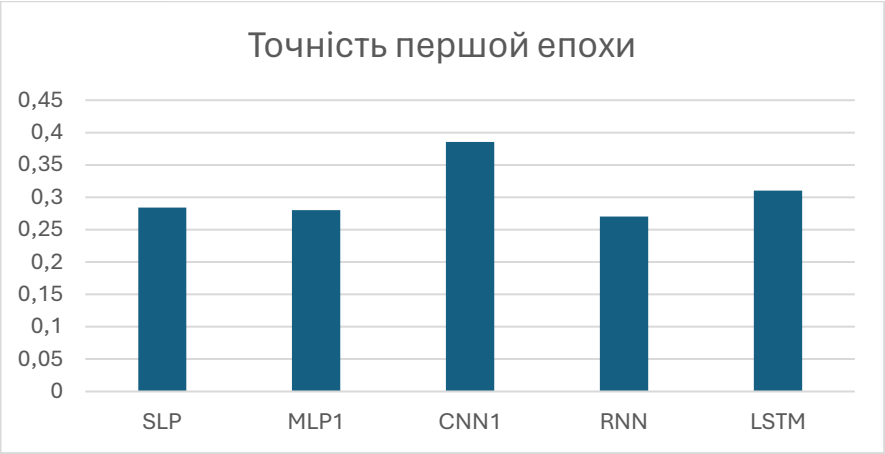


Рис. 3.4 – Графік точності першої епохи

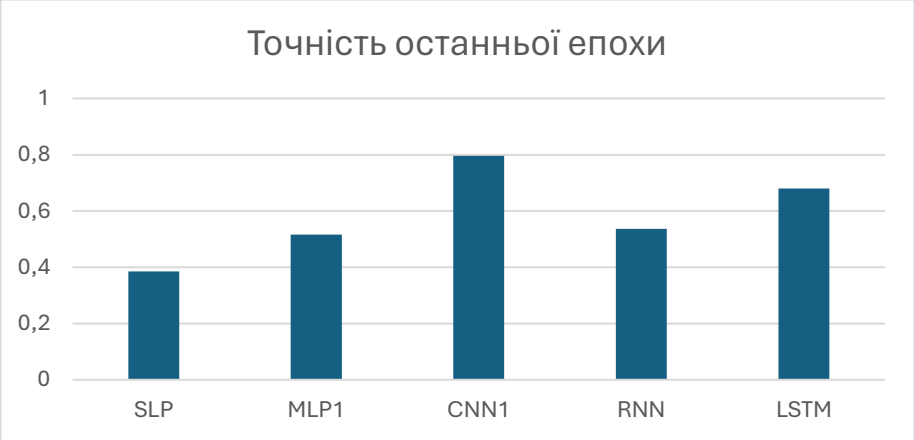


Рис. 3.5 – Графік точності останньої епохи

Рис. 3.6 – Графік часу тренування

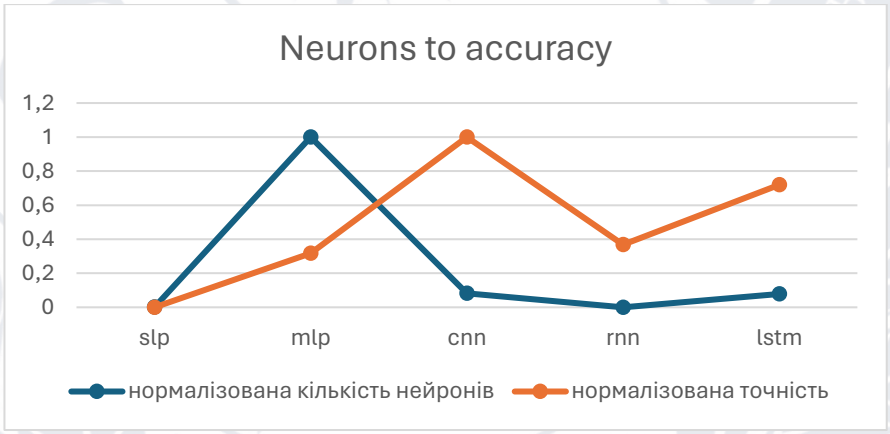
Далі знормалізовуємо дані, щоб спробувати знайти кореляцію (таблиця 3.2).

Таблиця 3.2 – Результати даних для знаходження кореляції

мінімальна кількість нейронів	29910	мінімальна точність	0,3854	мінімальний час	44
максимальна кількість нейронів	1707274	максимальна точність	0,7959	максимальний час	760

Таблиця 3.3 – Результати по моделям

назва моделі	нормалізована кількість нейронів	нормалізована точність	нормалізований час
slp	0,000488862	0	0
mlp	1	0,317904994	0,245810056
cnn	0,082064477	1	0,427374302
rnn	0	0,368087698	0,287709497
lstm	0,07768141	0,718879415	1



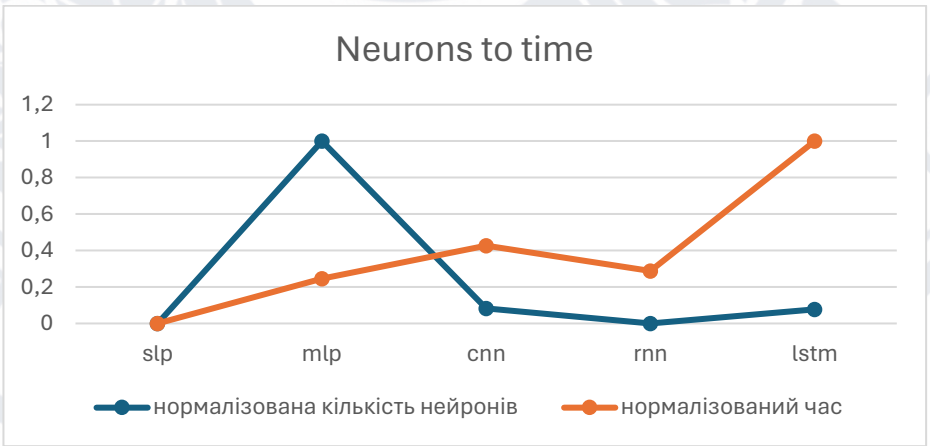


Рис. 3.7 - Графіки співвідношень

Висновки щодо попередніх графіків:

- Кількість нейронів сама по собі немає значущого впливу на точність.
- Кількість пам’яті яку займає модель прямо пропорційна кількості нейронів.
- Кількість нейронів як таких немає значущого впливу на час тренування.
- Кількість часу яку витрачено на тренування має вплив на точність.
- Архітектура, а саме шари мають значущий вплив на ефективність моделі.

Хоч MLP і не досягнув найкращих результатів я хотів би зробити порівняння між ним та конволюційною мережею використовуючи різні параметри та можливо знайти якісь залежності представлено у додатку И.

Результати вносимо у таблиці, які представлені у додатку К та Л.

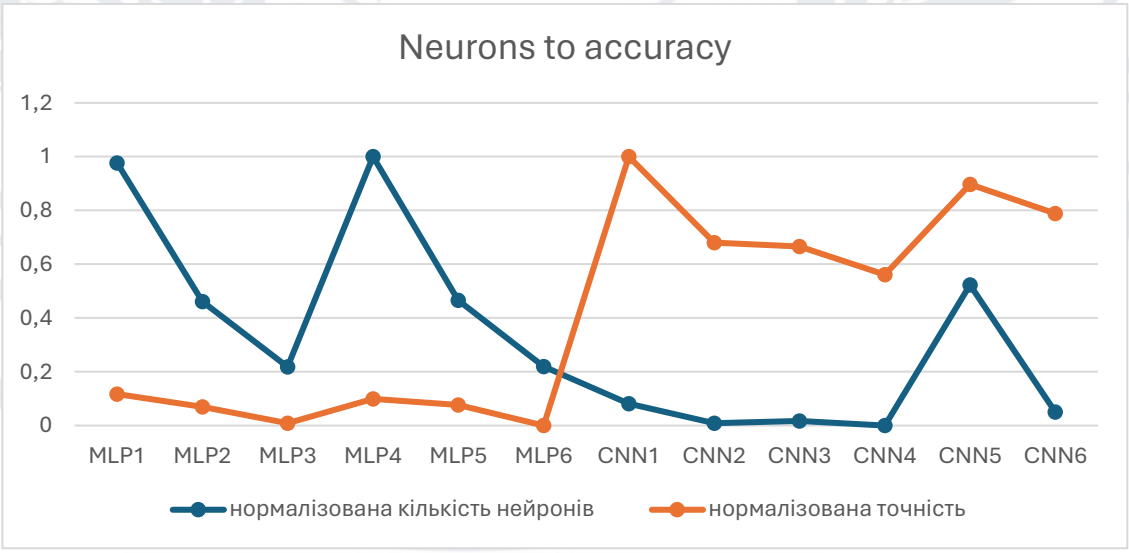


Рис. 3.8 – Графіки нормалізації

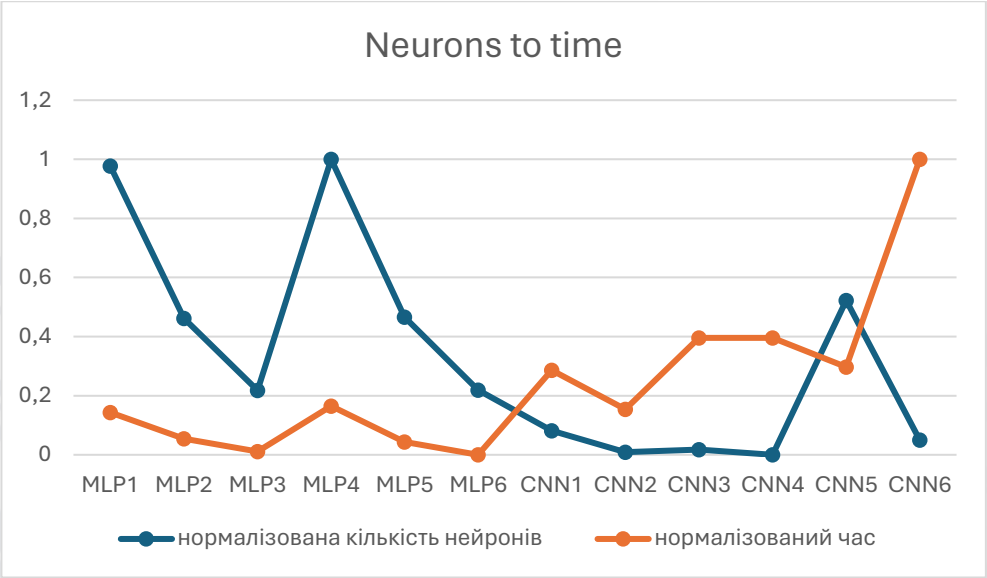


Рис. 3.9 – Графіки нормалізації

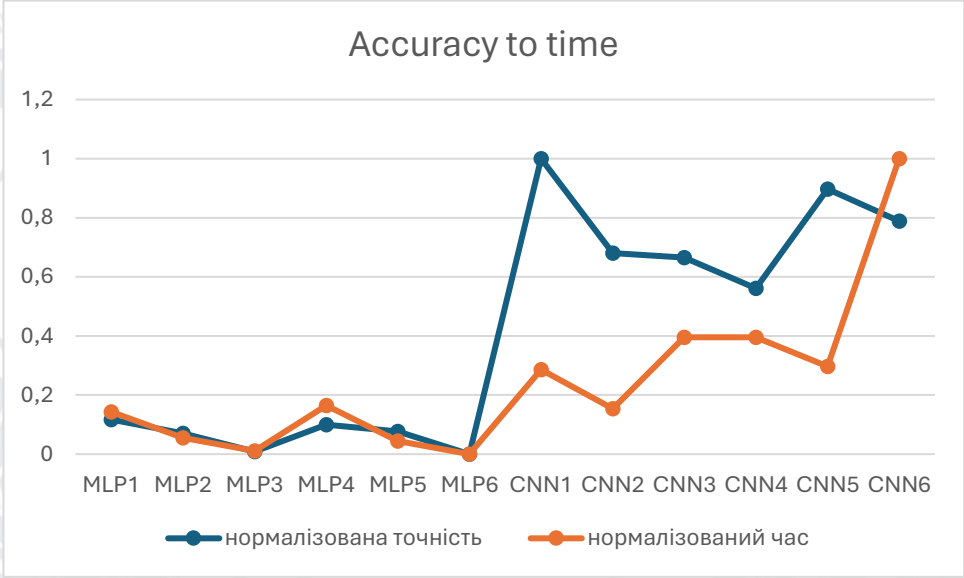


Рис. 3.10 – Графіки нормалізації

Зважаючи на попередні висновки підтвердилися й у цьому експерименті.

Для подальшого аналізу ефективності та впливу параметрів у різних архітектурах нейронних мереж, можна використати та техніки та напрямки:

1. Збільшення різноманітності та кількості архітектур та моделей нейронних мереж:

- Розробка та тестування гібридних моделей, що комбінують елементи різних архітектур.
- Включення нових перспективних архітектур, таких як графові нейронні мережі (GNN) та капсульні мережі.

2. Використання різних зборів даних для різного типу завдань:
- Збір та аналіз даних з різних джерел та доменів, щоб забезпечити широкий спектр задач для тестування моделей.
- Вивчення впливу якості та обсягу даних на ефективність різних моделей.
- Використання технік розширення даних (data augmentation) для поліпшення якості тренування моделей.
3. Використання не тільки центрального процесору для навчання, а й відповідних відеокарт:
- Оптимізація використання GPU та TPU для прискорення процесу навчання моделей.
- Дослідження можливостей розподіленого навчання для обробки великих обсягів даних.
- Порівняння ефективності та продуктивності різних апаратних платформ у навчанні нейронних мереж.
4. Врахування більшої кількості параметрів для аналізу:
- Додавання параметрів, таких як кількість шарів конкретного типу, розмір шарів, функції активації, методи регуляризації, та інші архітектурні параметри.
- Аналіз впливу різних гіперпараметрів, таких як швидкість навчання, розмір пакету, та кількість епох на результати моделей.
- Використання технік пошуку гіперпараметрів, таких як сітковий пошук (grid search) та байєсовий оптимізаційний пошук.
5. Вивчення методів попереднього навчання та трансферного навчання:
- Дослідження впливу попереднього навчання моделей на великих наборах даних на ефективність в задачах з малими наборами даних.
- Використання технік трансферного навчання для адаптації моделей до нових задач та доменів.
6. Оцінка стійкості та узагальнюваності моделей:
- Аналіз стійкості моделей до змін у вхідних даних, таких як шум або перекоси.

- Тестування моделей на різних наборах даних, щоб перевірити їх здатність узагальнювати на нові дані.
- Використання методів валідації, таких як крос-валідація та оцінка на тестових наборах даних.

7. Використання різних метрик для оцінки ефективності моделей:

- Застосування широкого спектру метрик, таких як точність, F1-міра, середньоквадратична похибка (MSE), та інших, для комплексної оцінки моделей.
- Аналіз компромісів між різними метриками, такими як точність та швидкість обробки.

8. Дослідження інтерпретованості та пояснюваності моделей:

- Розробка методів для пояснення прийнятих моделей та їх рішень.
- Використання технік, таких як LIME (Local Interpretable Model-agnostic Explanations) та SHAP (SHapley Additive exPlanations), для підвищення прозорості моделей.

9. Автоматизація процесу налаштування та вибору моделей:

- Використання методів автоматизованого машинного навчання (AutoML) для автоматизації процесу налаштування гіперпараметрів та вибору архітектури.
- Інтеграція методів оптимізації, таких як генетичні алгоритми та алгоритми рою частинок, для покращення результатів.

ВИСНОВКИ

В результаті проведеного дослідження було виявлено, що кількість нейронів у моделі машинного навчання не має значущого впливу на її точність. Проте, кількість нейронів прямо пропорційно впливає на об’єм пам’яті, який займає модель. Крім того, кількість нейронів не має суттєвого впливу на час тренування. Натомість, час, витрачений на тренування, безпосередньо впливає на точність моделі.

Важливим чинником ефективності моделі є її архітектура, зокрема кількість і структура шарів. Архітектура моделі виявилася значущим фактором, що впливає на її ефективність при вирішенні задач класифікації зображень.

Основною метою дослідження було порівняння різних моделей машинного навчання з різними параметрами для класифікації зображень. Дослідження дозволило визначити, які моделі та параметри є найбільш ефективними для конкретної задачі класифікації зображень.

В процесі роботи були підібрані ряд моделей для аналізу та обробки даний нейронних мереж, проведена характеристика різних архітектур нейронних мереж, де були вказані як і недоліки так і пропозиції удосконалення даних моделей. Проведено експериментально аналіз архітектур різних нейронних мереж та оцінено результати залежності між продуктивністю та архітектурою мереж.

Проведене дослідження внесло важливий внесок у розуміння того, як різні параметри впливають на продуктивність моделей машинного навчання. Це дозволяє краще налаштовувати моделі для досягнення оптимальних результатів у класифікації зображень. Зокрема, результати вказують на те, що правильний вибір архітектури моделі та оптимальний час тренування є ключовими факторами, що впливають на точність і ефективність моделі. Ці висновки можуть бути корисними для подальших досліджень та практичного застосування у сфері машинного навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. David C. Lay, Steven R. Lay, Judi J. McDonald, Linear Algebra and Its Applications Fifth Edition, Pearson 5th edition 2014, с. 1-4.
2. В. В. Булдигін, І. В. Алєксєєва, В. О. Гайдей, О. О. Диховичний, Н. Р. Коновалова, Л. Б. Федорова, ЛІНІЙНА АЛГЕБРА ТА АНАЛІТИЧНА ГЕОМЕТРІЯ, Київ 2011, с. 11-17.
3. Л. М. Добровська І. А. Добровська, Теорія та практика нейронних мереж Навчальний посібник, Київ НТУУ «КПІ» 2015, с. 15-17
4. А.Я. Дороговцев, Математичний аналіз частина перша, Київ ‘Либідь’ 1993, с. 12
5. https://uk.wikipedia.org/wiki/%D0%9F%D0%BE%D1%85%D1%96%D0%B4%D0%BD%D0%B0 (дата доступу 4/20/2024)
6. Marc Peter Deisenroth A. Aldo Faisal Cheng Soon Ong, MATHEMATICS FOR MACHINE LEARNING, Cambridge University Press 2020, с. 146
7. Diederik P. Kingma Jimmy Lei Ba, ADAM: A METHOD FOR STOCHASTIC OPTIMIZATION, публіковано на конференції ICLR 2015.
8. Вікіпедія поняття регуляризація в математиці та статистиці URL: https://uk.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B3%D1%83%D0%BB%D1%8F%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D1%96%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) (дата доступу 4/04/2024)
9. Поняття теорії ймовірностей URL: https://en.wikipedia.org/wiki/Probability_distribution (дата доступу 21.04.2024)
10. Н. М. Куссуль А.Ю. Шелестов С. А. Тарасенко Г.О. Яйлимова, Аналіз даних Лабораторний практикум, КПІ ім. Ігоря Сікорського 2022, розділ 4.1
11. Теорема Байєса. Вікіпедія URL: https://uk.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B5%D0%BC%D0%B0_%D0%91%D0%B0%D1%94%D1%81%D0%B0 (дата доступу 21.04.2024)

12. Матеріал про наївні байєсівські класифікатори URL: https://www.ibm.com/topics/naive-bayes (дата доступу 21.04.2024)
13. Сторінка вікіпедія. Байєсова мережа URL: https://uk.wikipedia.org/wiki/%D0%91%D0%B0%D1%94%D1%81%D0%BE%D0%B2%D0%B0_%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D0%B0 (дата доступу 21.04.2024)
14. Матеріали іноземних сайтів про поняття ентропія URL: https://en.wikipedia.org/wiki/Entropy_(information_theory) (дата доступу 03.05.2024)
15. Thomas M. Cover Joy A. Thomas, Elements of Information Theory, 1991, с. 18
16. Thomas M. Cover Joy A. Thomas, Elements of Information Theory, 1991, с. 19-20
17. Матеріали іноземних сайтів про алгоритми LZ77 і LZ78 стиснення даних URL: https://en.wikipedia.org/wiki/LZ77_and_LZ78 (дата доступу 11.05.2024)
18. Матеріали іноземних сайтів про MPEG-1 на цифрове стиснення даних URL: https://en.wikipedia.org/wiki/MPEG-1 (дата доступу 11.05.2024)
19. https://scikit-learn.org/stable/modules/neural_networks_supervised.html (дата доступу 11.05.2024)
20. Матеріали іноземних сайтів про моделі нейронних мереж URL: https://cloud.google.com/discover/what-is-unsupervised-learning (дата доступу 30.05.2024)
21. Матеріали іноземних сайтів про повторні нейронні мережі (RNN) URL: https://aws.amazon.com/what-is/recurrent-neural-network/ (дата доступу 28.05.2024)
22. Відео-матеріали іноземних сайтів про глибинне навчання: згорткові нейронні мережі URL: https://www.mathworks.com/videos/introduction-to-deep-learning-what-are-convolutional-neural-networks--1489512765771.html (дата доступу 28.05.2024)

23. Матеріали іноземних сайтів про поняття глибокого навчання URL: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939 (дата доступу 28.05.2024)
24. Матеріали іноземних сайтів про згорткові нейронні мережі URL: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939 (28/05/2024)
25. Матеріали іноземних сайтів про нейронні мережі та глибоке навчання URL: https://www.coursera.org/articles/neural-network-architecture (дата доступу 27.05.2024)
26. Матеріали іноземних сайтів про обробку природної мови (NLP) з глибоким навчанням URL: https://wikidocs.net/202231 (дата доступу 01.06.2024)
27. Камінський А. Б. Нейромережеві технології в управлінні портфелем простроченої заборгованості / А. Б. Камінський, В. О. Сікач // Моделювання та інформаційні системи в економіці: зб. наук. праць. — К. : КНЕУ, 2011. — Вип. 84. — С. 5—19.
28. Савіна С. С. Об’єднання моделей logit-регресій як комітету експертів для оцінки кредитоспроможності позичальника / С. С. Савіна, Вибір архітектури нейромережі для розв’язання задачі. С. С. Савіна, В. П. Бень 151 В. П. Бень // Нейро-нечіткі технології моделювання в економіці. — 2015. — № 4. — С. 154—188.
29. Генаш, М. Г. Застосування нейронних мереж архітектури UNet, DeepLabV3, PSPNet для семантичної сегментації обличчя на фотографії / Генаш Максим Геннадійович, Олійник Володимир Валентинович. – 2018. – Вып. 10(42), ч. 2. – С. 69–74.
30. Nguyen A. Рекурентна нейронна мережа для обробки великих текстових даних / A. Nguyen, Y. Sidorov // Системи управління, навігації та зв’язку. Збірник наукових праць. – Полтава: ПНТУ, 2018. – Т. 4 (50). – С. 135-138.
31. Khanam, Z., Alwasel, B. N., Sirafi, H., Rashid, M. (2021). Fake News Detection Using Machine Learning Approaches. IOP Conference Series: Materials Science

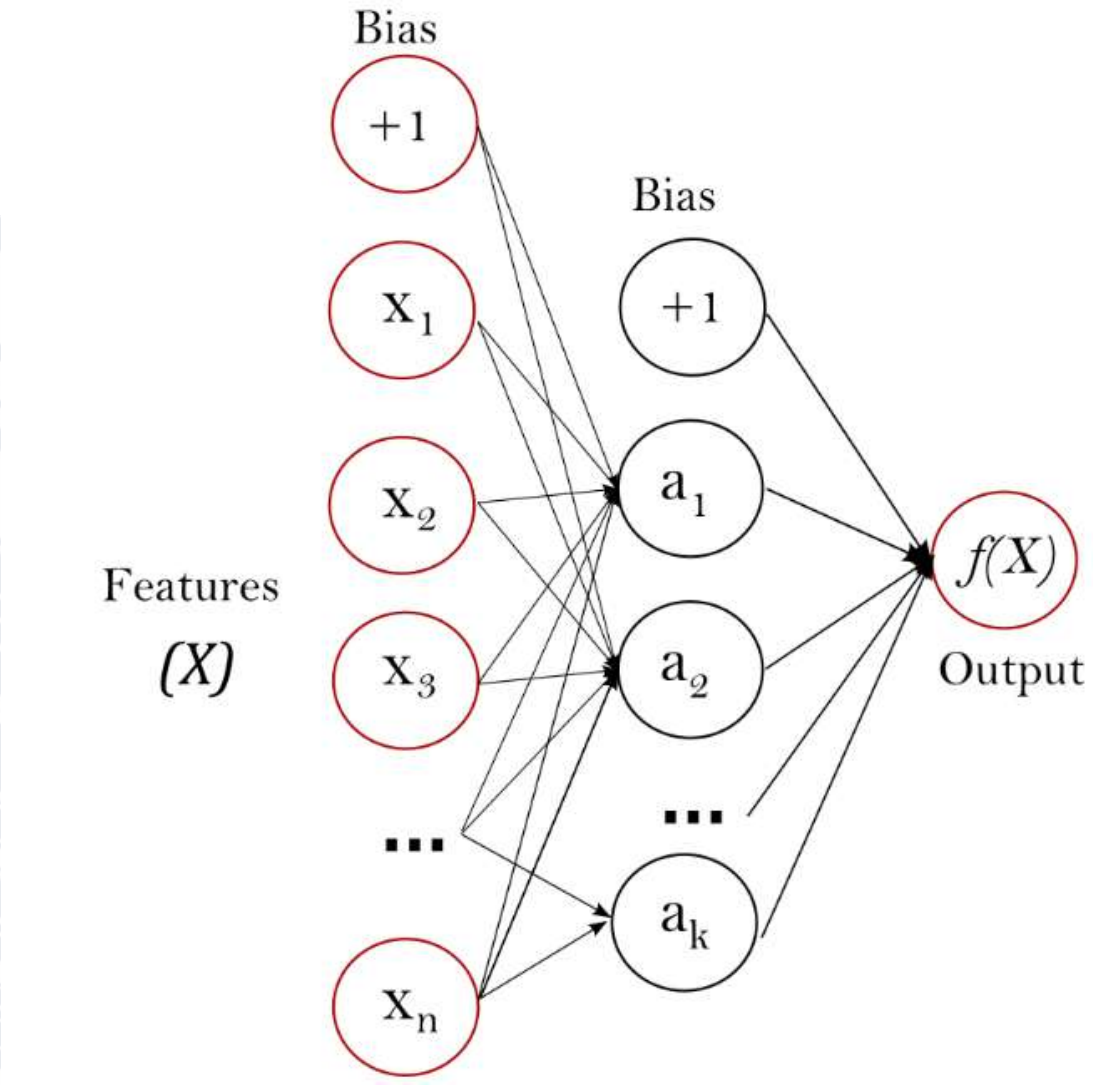
and Engineering, 1099(1), 012040. https://doi.org/10.1088/1757-899x/1099/1/012040

32. Transformer neural networks are shaking up AI | TechTarget. Enterprise AI. https://www.techtarget.com/searchenterpriseai/feature/Transformer-neural-networks-are-shaking-up-AI
33. Chen, W., Zhang, Y., Yeo, C. K., Lau, C. T., Lee, B. S. (2018). Unsupervised rumor detection based on users' behaviors using neural networks. Pattern Recognition Letters, 105, 226–233. https://doi.org/10.1016/j.patrec.2017.10.014
34. Inc., V. Fakebox · Docs · Machine Box · Machine learning in a box. Fakebox · Docs · Machine Box · Machine Learning in a Box. https://machinebox.io/
35. Patwari, K., Hafiz, S. M., Wang, H., Homayoun, H., Shafiq, Z., Chuah, C. N. (2022). DNN Model Architecture Fingerprinting Attack on CPU-GPU Edge Devices. 2022 IEEE 7th European Symposium on Security and Privacy (EuroS&P). https://doi.org/10.1109/eurosp53844.2022.00029
36. Rocca, J. (2021). Understanding Generative Adversarial Networks (GANs). Medium. https://towardsdatascience.com/understanding-generative-adversarial-networks-gans-cd6e4651a29

ДОДАТКИ

ДОДАТОК А

Рис. 2.1 – Модель мультишарового перспетрону



ДОДАТОК Б

Типи керованого і некерованого навчання

*Типи керованого навчання***.** Кероване навчання в машинному навчанні зазвичай ділиться на дві категорії: класифікація та регресія.

*Класифікація***.** Алгоритми класифікації використовуються для групування даних шляхом прогнозування категоріальної мітки або вихідної змінної на основі вхідних даних. Класифікація використовується, коли вихідні змінні є категоріальними, тобто існує два або більше класів. Одним із найпоширеніших прикладів використання алгоритмів класифікації є спам-фільтр у нашій електронній пошті. Тут модель керованого навчання навчається прогнозувати, чи є електронний лист спамом чи ні, використовуючи набір даних, який містить марковані приклади як спамових, так і легітимних листів. Алгоритм витягує інформацію про кожен лист, включаючи відправника, тему, текст листа тощо. Потім він використовує ці ознаки та відповідні мітки вихідних даних, щоб навчитися виявляти шаблони та призначати оцінку, яка вказує на те, чи є лист справжнім або спамом.

Регресія. Алгоритми регресії використовуються для прогнозування реального або неперервного значення, де алгоритм виявляє взаємозв’язок між двома або більше змінними. Загальним прикладом завдання регресії може бути прогнозування зарплати на основі досвіду роботи. Наприклад, алгоритм керованого навчання отримує вхідні дані, пов'язані з досвідом роботи (наприклад, тривалість часу, галузь або сфера, місце розташування тощо) та відповідну призначену суму зарплати. Після навчання моделі її можна використовувати для прогнозування середньої зарплати на основі досвіду роботи.

Некероване навчання в штучному інтелекті – це тип машинного навчання, який вчиться на даних без людського нагляду. На відміну від навчання з наглядом, моделі некерованого машинного навчання отримують немарковані дані і можуть

самостійно виявляти шаблони та розуміти їх без будь-яких явних вказівок або інструкцій.

Як працює некероване навчання? Як випливає з назви, некероване навчання використовує алгоритми самонавчання – вони вчаться без жодних міток або попереднього навчання. Замість цього модель отримує сирі, немарковані дані і повинна сама вивести свої правила та структурувати інформацію на основі схожостей, відмінностей і шаблонів без явних інструкцій щодо того, як працювати з кожним елементом даних.

Алгоритми некерованого навчання краще підходять для більш складних завдань обробки, таких як організація великих наборів даних у кластери. Вони корисні для виявлення раніше невиявлених шаблонів у даних і можуть допомогти визначити ознаки, корисні для категоризації даних. Наприклад, у нас є великий набір даних про погоду. Алгоритм некерованого навчання пройде через ці дані і виявить шаблони в точках даних. Він може згрупувати дані за температурою або схожими погодними умовами. Хоча сам алгоритм не розуміє цих шаблонів на основі будь-якої попередньої інформації, яку йому надали, можна пройти через ці групи даних і спробувати класифікувати їх на основі нашого розуміння набору даних. Наприклад, ми можемо визнати, що різні температурні групи представляють усі чотири сезони або що погодні умови розділені на різні типи погоди, такі як дощ, мокрий сніг або сніг представлено на рис. 2.2. (додаток В).

Методи некерованого машинного навчання Загалом, існує три типи завдань некерованого навчання: кластеризація, асоціативні правила та зниження вимірності. Нижче ми розглянемо кожен тип техніки некерованого навчання більш детально.

Кластеризація – це техніка для дослідження сирих, немаркованих даних та розбиття їх на групи (або кластери) на основі схожостей або відмінностей. Вона використовується в різних застосуваннях, включаючи сегментацію клієнтів, виявлення шахрайства та аналіз зображень. Алгоритми кластеризації ділять дані на природні групи, знаходячи схожі структури або шаблони в некатегоризованих даних. Кластеризація є одним із найпопулярніших підходів некерованого

машинного навчання. Існує кілька типів алгоритмів некерованого навчання, які використовуються для кластеризації, включаючи ексклюзивну, перехресну, ієрархічну та ймовірнісну кластеризацію.

Ексклюзивна кластеризація: Дані групуються таким чином, що одна точка даних може належати лише одному кластеру. Це також називається «жорсткою» кластеризацією. Загальним прикладом ексклюзивної кластеризації є алгоритм *K*-means, який розподіляє точки даних на *K* кластерів, де *K* визначається користувачем.

Перехресна кластеризація: Дані групуються таким чином, що одна точка даних може належати двом або більше кластерам з різними ступенями членства. Це також називається «м'якою» кластеризацією.

Ієрархічна кластеризація: Дані розділяються на окремі кластери на основі схожостей, які потім повторно об'єднуються і організовуються на основі їхніх ієрархічних відносин. Існує два основних типи ієрархічної кластеризації: агломеративна та дивізивна кластеризація. Цей метод також називається ієрархічним кластерним аналізом (ІКА).

Ймовірнісна кластеризація: Дані групуються в кластери на основі ймовірності кожної точки даних належати до кожного кластеру. Цей підхід відрізняється від інших методів, які групують точки даних на основі їх схожості з іншими в кластері.

Асоціація. Видобуток асоціативних правил – це підхід, заснований на правилах, для виявлення цікавих взаємозв'язків між точками даних у великих наборах даних. Алгоритми некерованого навчання шукають часті асоціації типу «якщо-то», які також називаються правилами, щоб виявити кореляції та співпадіння в даних і різні зв'язки між об'єктами даних. Найчастіше це використовується для аналізу роздрібних кошиків або наборів даних, щоб показати, як часто певні предмети купуються разом. Ці алгоритми виявляють шаблони покупок клієнтів і раніше приховані взаємозв'язки між продуктами, що допомагає створювати рекомендаційні системи або інші можливості для перехресних продажів. Найвідоміші приклади застосування це «Часто купують

разом» і «Люди, які купили цей товар, також купили» на вашому улюбленому інтернет-магазині. Асоціативні правила також часто використовуються для організації медичних наборів даних для клінічних діагнозів. Використання некерованого машинного навчання та асоціативних правил може допомогти лікарям визначити ймовірність конкретного діагнозу, порівнюючи взаємозв'язки між симптомами з минулих випадків пацієнтів. Зазвичай алгоритми Apriori є найбільш широко використовуваними для навчання асоціативних правил для виявлення пов'язаних колекцій елементів або наборів елементів. Однак також використовуються інші типи, такі як алгоритми Eclat і FP-growth.

Зниження вимірності. Зниження вимірності – це техніка некерованого навчання, яка зменшує кількість ознак або вимірів у наборі даних. Більше даних зазвичай краще для машинного навчання, але це також може ускладнити візуалізацію даних. Зниження вимірності витягує важливі ознаки з набору даних, зменшуючи кількість неактуальних або випадкових ознак. Цей метод використовує алгоритми аналізу головних компонент (PCA) і сингулярного розкладу (SVD) для зменшення кількості вхідних даних без компрометації цілісності властивостей у вихідних даних.

Приклади некерованого навчання в реальному світі. Тепер, коли ми розуміємо основи того, як працює некероване навчання, розглянемо найбільш поширені випадки використання, які допомагають бізнесам швидко досліджувати великі обсяги даних. Ось кілька реальних прикладів некерованого навчання:

- *Виявлення аномалій:* Некерована кластеризація може обробляти великі набори даних і виявляти точки даних, які є нетиповими для набору даних.
- *Рекомендаційні системи*: Використовуючи асоціативні правила, некероване машинне навчання може допомогти досліджувати транзакційні дані, щоб виявити шаблони або тенденції, які можуть бути використані для створення персоналізованих рекомендацій для онлайн-ритейлерів.
- *Сегментація клієнтів*: Некероване навчання часто використовується для створення профілів покупців шляхом кластеризації загальних ознак або

поведінки покупців. Ці профілі можуть бути використані для направлення маркетингових і інших бізнес-стратегій.

- *Виявлення шахрайства:* Некероване навчання корисне для виявлення аномалій, виявляючи незвичайні точки даних у наборах даних. Ці осяяння можуть допомогти виявити події або поведінку, що відхиляються від нормальних шаблонів у даних, виявляючи шахрайські транзакції або незвичайну поведінку, наприклад, активність ботів.
- *Обробка природної мови (NLP):* Некероване навчання часто використовується для різних застосувань NLP, таких як категоризація статей у новинних розділах, переклад і класифікація тексту або розпізнавання мови в розмовних інтерфейсах.
- *Генетичні дослідження:* Генетична кластеризація є ще одним поширеним прикладом некерованого навчання. Ієрархічні алгоритми кластеризації часто використовуються для аналізу ДНК-патернів і виявлення еволюційних зв’язків.

Некероване навчання добре підходить для завдань, що вимагають дослідження великої кількості немаркованих даних. Такий підхід полегшує бізнесу отримання інсайтів із даних, коли відсутні мітки, допомагаючи зрозуміти основну структуру набору даних і виявити шаблони та взаємозв’язки між наборами даних без потреби вчити їх людиною.

ДОДАТОК В

Рис. 2.2 – Приклад кластеризації

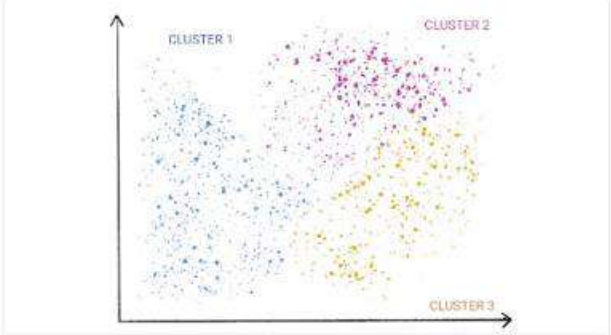


Figure 1. An ML model clustering similar data points.

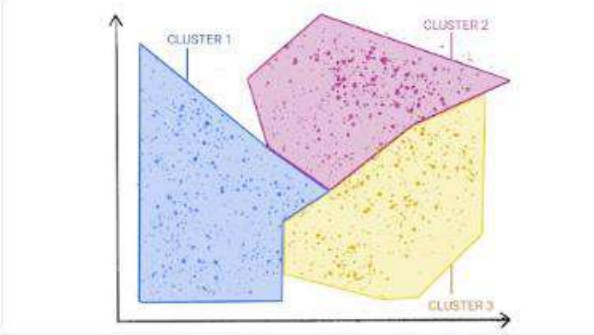


Figure 2. Groups of clusters with natural demarcations.

ДОДАТОК Г

Теоретична складова архітектури нейронних мереж

Нейромережа прямого поширення. Нейромережа прямо поширення є однією з найпростіших форм нейронних мереж, і її архітектуру часто використовують для створення більш спеціалізованих мереж. Як підказує назва, ці нейронні мережі передають дані від входу до виходу без петель або кругових зв'язків. Хоча це одна з найпростіших структур нейронних мереж, приховані шари між входом і виходом все ще можуть бути складними. Цей тип нейронної мережі можна використовувати для різних завдань, таких як розпізнавання зразків і зображень, регресійний аналіз та класифікація.

Як працює архітектура цієї нейронної мережі? Нейромережа прямого поширення має вхідний шар, за яким слідує серія прихованих шарів, і завершується вихідним шаром. Дані надходять в алгоритм через вхід і проходять через вузли в першому шарі. Перший шар вузлів обчислює дані на основі ваг вузлів і передає обчислення наступному шару вузлів. Кожен вузол в кожному шарі з'єднується з кожним вузлом наступного шару, але дані можуть передаватися лише в напрямку виходу.

Рекурентна нейронна мережа (RNN). Рекурентна нейронна мережа (RNN) – це модель глибокого навчання, яка навчається обробляти та перетворювати послідовні дані на конкретний послідовний вихід. Послідовні дані – це дані, такі як слова, речення або часові ряди, де компоненти послідовності взаємопов'язані на основі складних семантичних і синтаксичних правил. RNN – це програмна система, що складається з багатьох взаємопов'язаних компонентів, що імітують те, як люди виконують перетворення послідовних даних, наприклад, перекладають текст з однієї мови на іншу.

RNN значною мірою замінюються трансформерними моделями ШІ (AI) та великими мовними моделями (LLM), які набагато ефективніше обробляють послідовні дані.

Як працює рекурентна рейронна Мережа (RNN)? Рекурентні нейронні мережі (RNN) складаються з нейронів: вузлів обробки даних, які працюють разом для виконання складних завдань. Нейрони організовані в шари: вхідний, вихідний і прихований. Вхідний шар отримує інформацію для обробки, вихідний шар надає результат, а обробка даних, аналіз і прогнозування відбуваються у прихованому шарі.

Прихований Шар. RNN працюють шляхом послідовної передачі даних, які вони отримують, до прихованих шарів, по одному кроку за раз. Однак, вони також мають самоповторюваний або рекурентний робочий процес: прихований шар може запам'ятовувати та використовувати попередні вхідні дані для майбутніх прогнозів у компоненті короткострокової пам'яті. Він використовує поточний вхід і збережену пам'ять для прогнозування наступної послідовності.

Наприклад, розглянемо послідовність: «Яблуко червоне». Ви хочете, щоб RNN передбачила «червоне», коли вона отримує вхідну послідовність «Яблуко є». Коли прихований шар обробляє слово «Яблуко», він зберігає його копію в пам'яті. Далі, коли він бачить слово «є», він згадує «Яблуко» з пам'яті та розуміє повну послідовність: «Яблуко є» для контексту. Потім він може передбачити «червоне» з підвищеною точністю. Це робить RNN корисними для розпізнавання мови, машинного перекладу та інших завдань моделювання мови.

RNN використовуються у завданнях, де послідовні дані мають велике значення, оскільки вони можуть враховувати контекст попередніх елементів послідовності для більш точних прогнозів.

Типи Рекурентних Нейронних Мереж. RNN часто характеризуються архітектурою «один до одного»: одна вхідна послідовність асоціюється з одним виходом. Однак їх можна гнучко налаштовувати для різних цілей. Нижче наведено кілька поширених типів RNN.

*Один до багатьох***.** Цей тип RNN спрямовує один вхід до кількох виходів. Це дозволяє застосовувати такі лінгвістичні додатки, як створення підписів до зображень, генеруючи речення з одного ключового слова.

Багато до багатьох. Модель використовує кілька входів для прогнозування кількох виходів. Наприклад, ви можете створити перекладач мов за допомогою RNN, який аналізує речення та правильно структурує слова іншою мовою.

Багато до одного. Кілька входів відображаються на один вихід. Це корисно в таких додатках, як аналіз настроїв, де модель прогнозує настрої клієнтів, такі як позитивні, негативні та нейтральні, на основі вхідних відгуків.

Як рекурентні нейронні мережі порівнюються з іншими мережами глибокого навчання? Рекурентна нейронна мережа проти прямої нейронної мережі (прямого поширення). Як і RNN, прямі нейронні мережі є штучними нейронними мережами, які передають інформацію від одного кінця до іншого. Пряма нейронна мережа може виконувати прості завдання класифікації, регресії або розпізнавання, але вона не може запам’ятовувати попередній вхід, який вона обробляла. Наприклад, вона забуває «Яблуко» до того моменту, як її нейрон обробляє слово «є». RNN долає це обмеження пам’яті, включаючи прихований стан пам’яті в нейрон.

Рекурентна нейронна мережа проти згорткової нейронної мережі (конволюційна). Згорткові нейронні мережі (CNN) є штучними нейронними мережами, які призначені для обробки просторових даних. Ви можете використовувати згорткові нейронні мережі для отримання просторової інформації з відео та зображень, пропускаючи їх через серію згорткових і пулінгових шарів у нейронній мережі. RNN призначені для захоплення довгострокових залежностей у послідовних даних.

Архітектура RNN поклала основу для моделей машинного навчання з можливістю обробки мови. З’явилося кілька варіантів, які спільно зберігають її принцип пам’яті та покращують її початкові функції.

Двостороння рекурентна нейронна мережа (BRNN) обробляє послідовності даних з передніми та задніми шарами прихованих вузлів. Передній шар працює подібно до RNN, яка зберігає попередній вхід у прихованому стані і використовує його для передбачення наступного виходу. Тим часом, задній шар

працює в протилежному напрямку, беручи як поточний вхід, так і майбутній прихований стан, щоб оновити поточний прихований стан. Комбінація обох шарів дозволяє BRNN покращити точність передбачення, враховуючи минулий та майбутній контексти.

Довга короткотермінова пам’ять (LSTM) - це варіант RNN, який дозволяє моделі розширити свою пам’яті, щоб вміщувати більший часовий проміжок. RNN може запам’ятати лише найближчий минулий вхід. Він не може використовувати вхідні дані з кількох попередніх послідовностей, щоб покращити прогноз. LSTM мережі додають спеціальний блок пам’яті, що називається клітинами, в прихований шар. Кожна клітина керується входовою, вихідною та забутковою воротами, які дозволяють шару запам’ятовувати корисну інформацію.

Блок забороненої пам’яті (GRU) – це RNN, яка дозволяє вибіркове збереження пам’яті. Модель додає ворота оновлення та забуткові ворота до свого прихованого шару, які можуть зберігати або видаляти інформацію з пам’яті.

Обмеження рекурентних нейронних мереж. З моменту введення RNN в машинне навчання, інженери зробили значний прогрес у застосуванні їх та їх варіантів у обробці природної мови (NLP). Проте, сім’я моделей RNN має кілька обмежень.

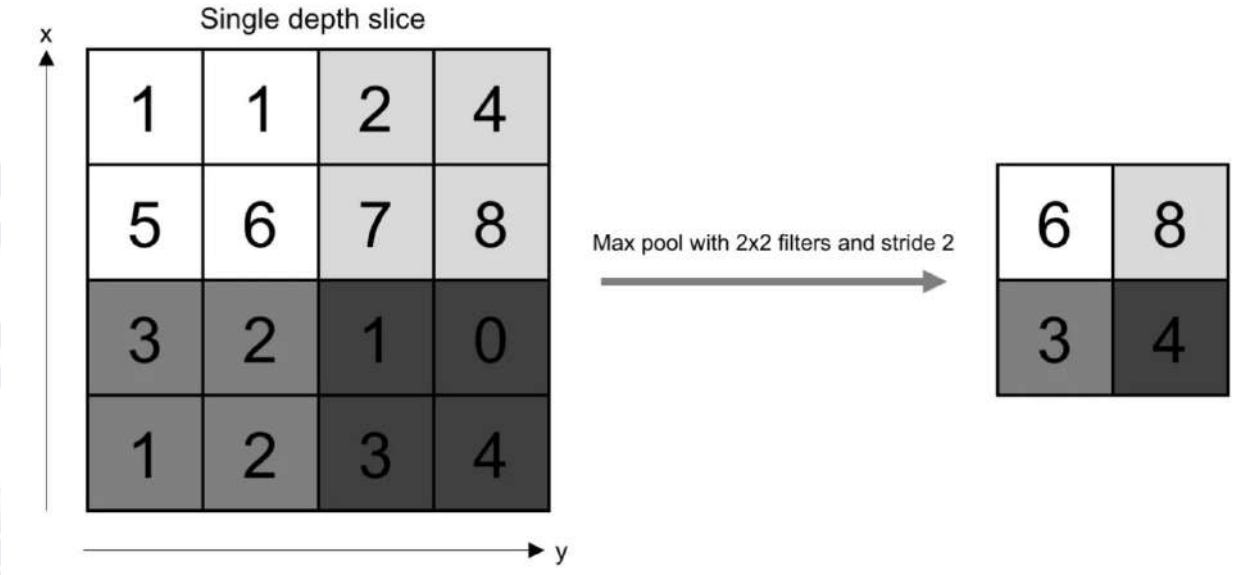
Вибух градієнту. RNN може неправильно передбачити вихідні дані на початковому етапі навчання. Потрібно декілька ітерацій, щоб налаштувати параметри моделі та зменшити рівень помилок. Градієнт описує чутливість рівня помилок відносно параметрів моделі. Про градієнт можна думати як про нахил, який ви берете, щоб спуститися з гори. Більш крутий градієнт дозволяє моделі навчатися швидше, а менш крутий градієнт зменшує швидкість навчання. Вибух градієнту виникає, коли градієнт зростає експоненційно до того моменту, поки RNN не стає стійким. Коли градієнти стають нескінченно великими, RNN веде себе непередбачувано, що призводить до проблем з продуктивністю, такими як перенавчання. Перенавчання – це явище, коли модель може точно передбачати з тренувальними даними, але не може цього зробити з реальними даними.

Проблема зникнення градієнту – це стан, коли градієнт моделі наближається до нуля під час навчання. Коли градієнт зникає, RNN не може ефективно вчитися з тренувальних даних, що призводить до недонавчання. Модель з недостатнім навчанням не може ефективно працювати в реальних застосуваннях, оскільки її ваги не були налаштовані належним чином. RNN піддаються ризику проблем зникнення та вибуху градієнту, коли вони обробляють довгі послідовності даних.

Повільний час навчання. RNN обробляє дані послідовно, що обмежує його здатність ефективно обробляти велику кількість текстів. Наприклад, модель RNN може проаналізувати настрій покупця за кілька речень. Проте для того, щоб підсумувати сторінку есе, потрібні великі обчислювальні потужності, обсяг пам’яті та час [21].

ДОДАТОК Д

Рис. 2.4 – Схема представлення прикладу макс-пулінгу [23]



ДОДАТОК Е

Теоретична складова, чому важлива ОПМ, для чого використовується, як працює

Обробка природної мови (ОПМ) – це дисципліна, що займається створенням машин, здатних маніпулювати людською мовою – або даними, які нагадують людську мову – таким чином, як вона написана, вимовлена та організована. Вона еволюціонувала з обчислювальної лінгвістики, яка використовує комп'ютерні науки для розуміння принципів мови, але, на відміну від розробки теоретичних рамок, ОПМ є інженерною дисципліною, що прагне створювати технології для виконання корисних завдань. ОПМ можна поділити на два взаємопов'язані частини: розуміння природної мови (РПМ), яке зосереджено на семантичному аналізі або визначенні наміченого значення тексту, і генерація природної мови (ГПМ), що зосереджена на створенні тексту машиною. ОПМ відрізняється від – але часто використовується разом із – розпізнаванням мови, яке прагне аналізувати усну мову в слова, перетворюючи звук у текст і навпаки.

Чому обробка природної мови (ОПМ) важлива? ОПМ є невід'ємною частиною повсякденного життя і стає все більш значущою, оскільки технологія обробки мови застосовується в різних галузях, таких як роздрібна торгівля (наприклад, у чат-ботах для обслуговування клієнтів) і медицина (інтерпретація або підсумовування електронних медичних записів). Розмовні агенти, такі як Amazon Alexa і Apple Siri, використовують ОПМ для прослуховування запитів користувачів і пошуку відповідей. Найбільш складні такі агенти, як GPT-3, який нещодавно став доступним для комерційного використання, можуть генерувати складні тексти на різноманітні теми, а також забезпечувати роботу чат-ботів, здатних вести зв'язні розмови. Google використовує ОПМ для покращення результатів своєї пошукової системи, а соціальні мережі, такі як Facebook, використовують його для виявлення та фільтрації мови ворожнечі. ОПМ стає дедалі складнішою, але ще багато роботи належить виконати. Поточні системи схильні до упередженості та нелогічності, а іноді поводяться непередбачувано. Незважаючи на виклики, інженери з машинного навчання мають багато

можливостей для застосування ОПМ у способах, які стають все більш важливими для функціонування суспільства.

Для чого використовується обробка природної мови (ОПМ)? ОПМ використовується для широкого спектра завдань, пов'язаних з мовою, включаючи відповідь на запитання, класифікацію тексту різними способами та спілкування з користувачами.

Ось кілька завдань, які можна вирішити за допомогою ОПМ:

- *Аналіз емоції* у тексті – це процес класифікації емоційного наміру тексту. Зазвичай, вхідними даними для моделі класифікації емоції є фрагмент тексту, а вихідними даними є ймовірність того, що виражена емоція є позитивною, негативною або нейтральною. Зазвичай ця ймовірність базується на або ручних ознаках, словосполученнях *n*-грамів, ознаках TF-IDF, або на використанні моделей глибокого навчання для захоплення послідовних довго- та короткотермінових залежностей. Аналіз емоцій використовується для класифікації відгуків клієнтів на різних онлайн-платформах, а також для спеціальних застосувань, таких як виявлення ознак психічних захворювань в онлайн-коментарях.
- *Машинний переклад* автоматизує переклад між різними мовами. Вхідними даними для такої моделі є текст на вказаній вихідній мові, а вихідними – текст на вказаній цільовій мові. Google Translate, мабуть, є найвідомішим загальновідомим застосуванням. Такі моделі використовуються для покращення спілкування між людьми на соціальних медіаплатформах, таких як Facebook або TikTok. Ефективні підходи до машинного перекладу можуть розрізняти слова зі схожими значеннями. Деякі системи також виконують ідентифікацію мови; тобто класифікують текст як такий, що написаний тією чи іншою мовою.
- *Виявлення спаму* є поширеною проблемою бінарної класифікації в ОПМ, де метою є класифікація електронних листів як спам або ні. Детектори спаму приймають на вхід текст електронного листа разом з іншими підтекстами, такими як заголовок і ім'я відправника. Вони прагнуть

визначити ймовірність того, що лист є спамом. Провайдери електронної пошти, такі як Gmail, використовують такі моделі для покращення користувацького досвіду, виявляючи небажані та небажані електронні листи і переміщуючи їх до спеціальної папки для спаму.

- *Моделі виправлення граматичних помилок* кодують граматичні правила для виправлення граматики в тексті. Це головним чином розглядається як завдання послідовність-послідовність, де модель тренується на неграматичному реченні як вхідних даних і правильному реченні як вихідних даних. Онлайн-перевірки граматики, такі як Grammarly, і текстові процесори, такі як Microsoft Word, використовують такі системи для покращення досвіду письма своїх користувачів. Школи також використовують їх для оцінювання учнівських творів.
- *Генерація тексту*, більш формально відома як генерація природної мови (ГПМ), створює текст, подібний до написаного людиною. Такі моделі можуть бути налаштовані для створення тексту в різних жанрах і форматах – включаючи твіти, блоги і навіть комп'ютерний код. Генерація тексту здійснювалася з використанням марковських процесів, LSTM, BERT, GPT-2, LaMDA та інших підходів. Це особливо корисно для автозаповнення та чат-ботів. Автозаповнення передбачає, яке слово буде наступним, і такі системи різної складності використовуються в чат-додатках, таких як WhatsApp. Google використовує його для передбачення пошукових запитів. Однією з найвідоміших моделей для автозаповнення є GPT-2, яка використовувалася для написання статей, текстів пісень і багато іншого. Чат-боти автоматизують одну сторону розмови, тоді як людський співрозмовник зазвичай забезпечує іншу сторону. Їх можна розділити на наступні дві категорії: Запит до бази даних: Маємо базу даних із запитаннями та відповідями, і ми хочемо, щоб користувач запитував її за допомогою природної мови. Генерація розмов: Ці чат-боти можуть імітувати діалог із людським партнером. Деякі здатні брати участь у широкомасштабних розмовах. Відомим прикладом є LaMDA від Google,

яка надавала настільки людяні відповіді на запитання, що один із її розробників був переконаний, що вона має почуття.

Як працює обробка природної мови (ОПМ)? Моделі ОПМ працюють, знаходячи взаємозв'язки між складовими частинами мови – наприклад, літерами, словами і реченнями, що зустрічаються в текстових наборах даних. Архітектури ОПМ використовують різні методи для попередньої обробки даних, вилучення ознак та моделювання. Деякі з цих процесів це:

Попередня обробка даних: Перш ніж модель обробляє текст для конкретного завдання, текст часто потрібно попередньо обробити, щоб покращити продуктивність моделі або перетворити слова та символи у формат, який модель може зрозуміти. ШІ орієнтований на даних – це зростаючий рух, який надає пріоритет попередній обробці даних. Різні методи можуть використовуватися при попередній обробці даних: Стемінг та лематизація: Стемінг – це неформальний процес перетворення слів у їх базові форми за допомогою евристичних правил. Наприклад, «university», «universities» і «university's» можуть бути зведені до базової форми univers. (Одним з обмежень цього підходу є те, що «universe» також може бути зведене до univers, хоча universe і university не мають близького семантичного зв'язку.) Лематизація – це більш формальний спосіб знаходження коренів шляхом аналізу морфології слова з використанням словника. Стемінг і лематизація надаються такими бібліотеками, як spaCy і NLTK. Сегментація речень розбиває великий фрагмент тексту на лінгвістично значущі реченнєві одиниці. Це очевидно в таких мовах, як англійська, де кінець речення позначається крапкою, але все ж це не є тривіальним завданням. Крапка може використовуватися для позначення абревіатури, а також для закінчення речення, і в цьому випадку крапка повинна бути частиною токену самої абревіатури. Процес стає ще складнішим у мовах, таких як стародавня китайська, де немає розділового знаку, що позначає кінець речення. Видалення стоп-слів має на меті видалити найбільш поширені слова, які не додають багато інформації до тексту. Наприклад, «the», «a», «an» тощо. Токенізація розбиває текст на окремі слова та фрагменти слів. Результат зазвичай

складається з індексу слів та токенізованого тексту, в якому слова можуть бути представлені як числові токени для використання в різних методах глибокого навчання. Метод, який інструктує мовні моделі ігнорувати неважливі токени, може покращити ефективність.

ДОДАТОК Ж

Опис моделей нейронних мереж

Python – це високорівнева, інтерпретована мова програмування, що використовується для різних застосувань, включаючи веб-розробку, наукові дослідження, обробку даних, автоматизацію задач і, звісно, машинне навчання. Python відомий своєю простотою і читабельністю, що робить його ідеальним для швидкого прототипування та розробки.

Основні особливості Python:

1) Простота та зрозумілість: Код на Python легко читати та писати завдяки його зрозумілій синтаксичній структурі. Це дозволяє зосередитися на вирішенні задачі, а не на синтаксисі мови.
2) Багатофункціональність: Python підтримує кілька програмних парадигм, включаючи об’єктно-орієнтоване, функціональне та імперативне програмування.
3) Величезна кількість бібліотек та фреймворків: Python має безліч бібліотек для різних сфер застосування, таких як NumPy, Pandas, Matplotlib, SciPy для наукових обчислень та аналізу даних, а також TensorFlow, Keras, PyTorch для машинного навчання.
4) Кросплатформеність: Python підтримується на різних операційних системах, включаючи Windows, macOS, та Linux, що дозволяє розробляти програми для різних платформ без зміни коду.
5) Велика спільнота: Python має активну спільноту розробників, що сприяє постійному розвитку мови, наявності численних навчальних матеріалів і прикладів коду, а також швидкому вирішенню проблем.

TensorFlow – це відкрита платформа для машинного навчання, розроблена Google Brain Team. Вона надає широкий спектр інструментів та бібліотек для створення, тренування та розгортання моделей машинного навчання. TensorFlow підтримує як дослідження в галузі машинного навчання, так і виробниче використання завдяки своїй гнучкості та масштабованості.

Основні компоненти та особливості TensorFlow:

1) Tensor: Основна структура даних в TensorFlow. Тензори – це багатовимірні масиви, аналогічні масивам NumPy, але з можливістю виконання на GPU.
2) Графи обчислень: TensorFlow використовує графи обчислень для визначення потоку даних та операцій, які виконуються над цими даними. Граф обчислень складається з вузлів (операцій) та ребер (тензорів).
3) Сесії: В TensorFlow 1.x для виконання графів обчислень використовуються сесії. У TensorFlow 2.x, сесії більше не потрібні завдяки режиму «eager execution», який дозволяє виконувати операції одразу після їх виклику.
4) Автоматичне диференціювання: TensorFlow забезпечує автоматичне обчислення градієнтів, що є критичним для оптимізації моделей.
5) Розподілене навчання: TensorFlow підтримує розподілене навчання, дозволяючи використовувати декілька GPU або навіть декілька машин для паралельного тренування моделей.
6) TensorFlow Serving: Інструмент для розгортання моделей машинного навчання в продукційне середовище, що дозволяє легко інтегрувати моделі у веб-додатки або інші системи.

Keras – це високорівневий API для нейронних мереж, який спочатку був розроблений як незалежний проєкт, але тепер інтегрований в TensorFlow як його основний API для роботи з нейронними мережами. Keras забезпечує простий та інтуїтивний інтерфейс для побудови та тренування моделей.

Основні компоненти та особливості Keras:

1) Модульність та простота: Keras пропонує простий спосіб створення моделей нейронних мереж за допомогою високорівневих абстракцій, таких як Sequential та функціональна API.
2) Sequential API: Цей інтерфейс дозволяє створювати моделі шляхом додавання шарів один за одним, що є зручним для побудови простих моделей.
3) Функціональна API: Цей інтерфейс дозволяє створювати складніші моделі, включаючи багатовхідні, багатовихідні та моделі з розгалуженнями.

4) Широкий вибір шарів: Keras надає різноманітні шари для створення нейронних мереж, включаючи щільні (Dense), згорткові (Convolutional), рекурентні (Recurrent) та багато інших.
5) Підтримка зворотного зв'язку: Keras підтримує різні алгоритми оптимізації, функції втрат, та метрики для оцінки продуктивності моделей.
6) Сумісність з TensorFlow: Keras інтегрований в TensorFlow 2.x, що дозволяє користувачам використовувати всі можливості TensorFlow, такі як розподілене навчання, автоматичне диференціювання та TensorFlow Serving.

Переваги використання TensorFlow та Keras:

- Гнучкість: TensorFlow дозволяє створювати моделі будь-якої складності, від простих регресійних моделей до складних нейронних мереж.
- Масштабованість: TensorFlow підтримує масштабовані обчислення на багатьох GPU та кластерних середовищах.
- Простота: Keras забезпечує простий та інтуїтивний спосіб створення моделей, що дозволяє швидко прототипувати ітерації моделей.
- Розширена підтримка: Велика спільнота користувачів та розробників, численні навчальні матеріали та документація.

Kaggle — це онлайн платформа для змагань з аналізу даних і машинного навчання. Вона надає користувачам доступ до великих наборів даних, середовища для виконання кодів та інструментів для співпраці з іншими дослідниками. Kaggle є чудовим ресурсом як для початківців, так і для досвідчених фахівців з аналізу даних.

Опис моделей.

- ***Простий персептрон (Single-Layer Perceptron):***
- *Опис*: Ця модель має один повнозв’язаний шар без прихованих шарів. Після вхідного шару, який плоско розгортає зображення (Flatten), йде ще один повнозв’язаний шар з 10 виходами, які активуються функцією softmax.
- *Параметри:*

- Вхідний шар: Плоский шар (Flatten) з вхідною формою (32, 32, 3).
- Повнозв’язаний шар: 10 нейронів з функцією активації softmax.

Layer (type)	Output Shape	Param #
flatten_1 (Flatten)	(None, 3072)	0
dense_1 (Dense)	(None, 10)	30,730

Total params: 30,730 (120.04 KB)
Trainable params: 30,730 (120.04 KB)

- ***Багатошаровий персептрон (MLP)***:
- *Опис*: Ця модель має два приховані повнозв'язані шари перед вихідним шаром. Після вхідного шару, який плоско розгортає зображення, йдуть два повнозв’язані шари з функцією активації ReLU, після чого йде вихідний шар з 10 виходами, які активуються функцією softmax.
- *Параметри:*
  - Вхідний шар: Плоский шар (Flatten) з вхідною формою (32, 32, 3).
  - Повнозв’язаний шар 1: 512 нейронів з функцією активації ReLU.
  - Повнозв’язаний шар 2: 256 нейронів з функцією активації ReLU.
  - Вихідний шар: 10 нейронів з функцією активації softmax.

Layer (type)	Output Shape	Param #
flatten_2 (Flatten)	(None, 3072)	0
dense_2 (Dense)	(None, 512)	1,573,376
dense_3 (Dense)	(None, 256)	131,328
dense_4 (Dense)	(None, 10)	2,570

Total params: 1,707,274 (6.51 MB)
Trainable params: 1,707,274 (6.51 MB)

- **Згорткова нейронна мережа (CNN)**:
- *Опис*: Ця модель має два згорткових шари з пулінгом, після яких йде два повнозв’язані шари. Згорткові шари виявляють просторові шаблони в зображеннях.

- *Параметри:*
  - Згортковий шар 1: 32 фільтри розміром (3, 3) з функцією активації ReLU, після якого йде пулінг (MaxPooling2D) з розміром (2, 2).
  - Згортковий шар 2: 64 фільтри розміром (3, 3) з функцією активації ReLU, після якого йде пулінг (MaxPooling2D) з розміром (2, 2).
  - Повнозв'язаний шар 1: 64 нейрони з функцією активації ReLU.
  - Вихідний шар: 10 нейронів з функцією активації softmax.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 30, 30, 32)	896
max_pooling2d (MaxPooling2D)	(None, 15, 15, 32)	0
conv2d_1 (Conv2D)	(None, 13, 13, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten_3 (Flatten)	(None, 2304)	0
dense_5 (Dense)	(None, 64)	147,520
dense_6 (Dense)	(None, 10)	650

Total params: 167,562 (654.54 KB)
Trainable params: 167,562 (654.54 KB)

- **Рекурентна нейронна мережа (RNN)**:
- *Опис:* Ця модель має один шар LSTM для обробки послідовних даних. LSTM дозволяє врахувати довгострокові залежності в послідовних даних.
- *Параметри:*
  - LSTM шар: 50 нейронів з вхідною формою (32, 96) (32 часових кроки, кожен з 96 ознак), після якого йде вихідний шар з 10 нейронами з функцією активації softmax.

Layer (type)	Output Shape	Param #
lstm (LSTM)	(None, 50)	29,400
dense_7 (Dense)	(None, 10)	510

Total params: 29,910 (116.84 KB)
Trainable params: 29,910 (116.84 KB)

- **Довга короткострокова пам'ять (LSTM)**:
- *Опис*: Ця модель має два LSTM шари для обробки послідовних даних з додатковими довгостроковими залежностями.
- *Параметри:*
  - LSTM шар 1: 100 нейронів з вхідною формою (32, 96) (32 часових кроки, кожен з 96 ознак) з поверненням послідовності, що виводиться, після якого йде другий LSTM шар.
  - LSTM шар 2: 100 нейронів без повернення послідовності.
  - Вихідний шар: 10 нейронів з функцією активації softmax.

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 32, 100)	78,800
lstm_2 (LSTM)	(None, 100)	80,400
dense_8 (Dense)	(None, 10)	1,010

Total params: 160,210 (625.82 KB)

Trainable params: 160,210 (625.82 KB)

ДОДАТОК И

Код програми MLP2

Layer (type)	Output Shape	Param #
flatten_3 (Flatten)	(None, 3072)	0
dense_8 (Dense)	(None, 256)	786,688
dense_9 (Dense)	(None, 128)	32,896
dense_10 (Dense)	(None, 10)	1,290

MLP3

Layer (type)	Output Shape	Param #
flatten_4 (Flatten)	(None, 3072)	0
dense_11 (Dense)	(None, 128)	393,344
dense_12 (Dense)	(None, 64)	8,256
dense_13 (Dense)	(None, 10)	650

MLP4

Layer (type)	Output Shape	Param #
flatten_5 (Flatten)	(None, 3072)	0
dense_14 (Dense)	(None, 512)	1,573,376
dense_15 (Dense)	(None, 256)	131,328
dense_16 (Dense)	(None, 128)	32,896
dense_17 (Dense)	(None, 64)	8,256

dense_18 (Dense)	(None, 10)	650

MLP5

Layer (type)	Output Shape	Param #
flatten_6 (Flatten)	(None, 3072)	0
dense_19 (Dense)	(None, 256)	786,688
dense_20 (Dense)	(None, 128)	32,896
dense_21 (Dense)	(None, 64)	8,256
dense_22 (Dense)	(None, 10)	650

MLP6

Layer (type)	Output Shape	Param #
flatten_7 (Flatten)	(None, 3072)	0
dense_23 (Dense)	(None, 128)	393,344
dense_24 (Dense)	(None, 64)	8,256
dense_25 (Dense)	(None, 32)	2,080
dense_26 (Dense)	(None, 10)	330

CNN2

Layer (type)	Output Shape	Param #
conv2d_2 (Conv2D)	(None, 30, 30, 16)	448
max_pooling2d_2 (MaxPooling2D)	(None, 15, 15, 16)	0

conv2d_3 (Conv2D)	(None, 13, 13, 32)	4,640
max_pooling2d_3 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_8 (Flatten)	(None, 1152)	0
dense_27 (Dense)	(None, 32)	36,896
dense_28 (Dense)	(None, 10)	330

CNN3

Layer (type)	Output Shape	Param #
conv2d_4 (Conv2D)	(None, 30, 30, 64)	1,792
max_pooling2d_4 (MaxPooling2D)	(None, 15, 15, 64)	0
conv2d_5 (Conv2D)	(None, 13, 13, 32)	18,464
max_pooling2d_5 (MaxPooling2D)	(None, 6, 6, 32)	0
flatten_9 (Flatten)	(None, 1152)	0
dense_29 (Dense)	(None, 32)	36,896
dense_30 (Dense)	(None, 10)	330

CNN4

Layer (type)	Output Shape	Param #
conv2d_6 (Conv2D)	(None, 30, 30, 64)	1,792
max_pooling2d_6 (MaxPooling2D)	(None, 15, 15, 64)	0

conv2d_7 (Conv2D)	(None, 13, 13, 32)	18,464
max_pooling2d_7 (MaxPooling2D)	(None, 6, 6, 32)	0
conv2d_8 (Conv2D)	(None, 4, 4, 16)	4,624
max_pooling2d_8 (MaxPooling2D)	(None, 2, 2, 16)	0
flatten_10 (Flatten)	(None, 64)	0
dense_31 (Dense)	(None, 32)	2,080
dense_32 (Dense)	(None, 10)	330

CNN5

Layer (type)	Output Shape	Param #
conv2d_9 (Conv2D)	(None, 30, 30, 64)	1,792
max_pooling2d_9 (MaxPooling2D)	(None, 15, 15, 64)	0
flatten_11 (Flatten)	(None, 14400)	0
dense_33 (Dense)	(None, 64)	921,664
dense_34 (Dense)	(None, 32)	2,080
dense_35 (Dense)	(None, 10)	330

CNN6

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 30, 30, 128)	3,584
max_pooling2d_10 (MaxPooling2D)	(None, 15, 15, 128)	0

conv2d_11 (Conv2D)	(None, 13, 13, 64)	73,792
max_pooling2d_11 (MaxPooling2D)	(None, 6, 6, 64)	0
conv2d_12 (Conv2D)	(None, 4, 4, 32)	18,464
max_pooling2d_12 (MaxPooling2D)	(None, 2, 2, 32)	0
flatten_12 (Flatten)	(None, 128)	0
dense_36 (Dense)	(None, 128)	16,512
dense_37 (Dense)	(None, 10)	1,290

ДОДАТОК К

Таблиця 3.4 – Дані по архітектурам моделей

Архітектура моделі	Кількість нейронів до тренування	Кількість пам’яті (KB)	Точність першої епохи	Точність десятої епохи	Час тренування (у секундах)
MLP1	1,707,274	6510	0.2803	0.5159	220
MLP2	820,874	3130	0.2766	0.501	140
MLP3	402,250	1530	0.284	0.4817	100
MLP4	1,746,506	6660	0.2595	0.5104	240
MLP5	828,490	3160	0.2701	0.5031	130
MLP6	404,010	1540	0.2549	0.4788	90
CNN1	167,562	654.54	0.3854	0.7959	350
CNN2	42,314	165.29	0.3177	0.6945	230
CNN3	57,482	224.54	0.3219	0.6898	450
CNN4	27,290	106.6	0.2858	0.6567	450
CNN5	925,866	3530	0.3518	0.7632	360
CNN6	113,642	443.91	0.3193	0.729	1000

ДОДАТОК Л

Таблиця 3.5 – Нормалізація даних по архітектурам моделей

Назва	нормалізована кількість нейронів	нормалізована вага	нормалізована точність	нормалізований час
MLP1	0.977180296	0.977111118	0.116997792	0.142857143
MLP2	0.461596449	0.461348308	0.070009461	0.054945055
MLP3	0.218099413	0.217200232	0.00914538	0.010989011
MLP4	1	1	0.099653106	0.164835165
MLP5	0.466026375	0.465926084	0.076631977	0.043956044
MLP6	0.219123135	0.218726157	0	0
CNN1	0.081590679	0.08361156	1	0.285714286
CNN2	0.008738867	0.008955657	0.680227058	0.153846154
CNN3	0.017561493	0.017996765	0.665405235	0.395604396
CNN4	0	0	0.56102176	0.395604396
CNN5	0.522666145	0.522385327	0.896877956	0.296703297
CNN6	0.050227546	0.051470992	0.789025544	1