Plag

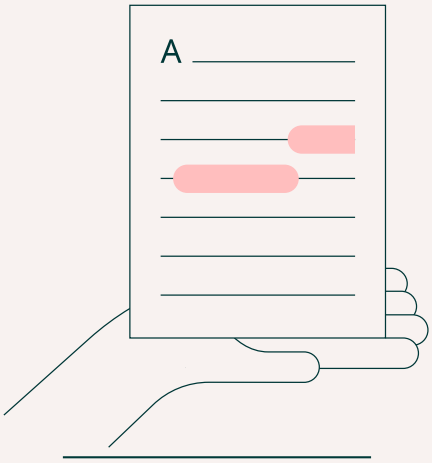
Звіт про оригінальність

● Оцінка схожості 21%

● Ризик плагіату НАЙВИЩИЙ

Ігор Кагало 2025-06-14 10:14

Посилання на звіт: 10bhR / Посилання користувача: qAHy



Ось вона –
Ваша звіт про оригінальність!

Ми раді повідомити, що перевірка вашого документа завершена, і результати вже готові! Наші алгоритми старанно працювали, щоб знайти збіги в наших базах даних.

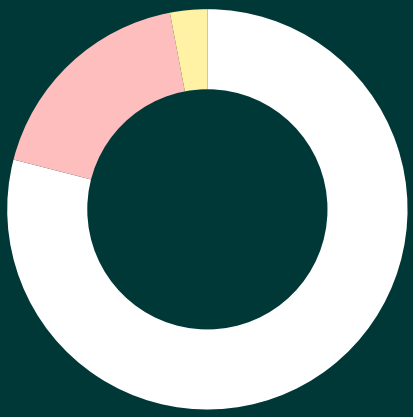
На наступних сторінках ви знайдете результати перевірки:

Бали
Збіги
Посилання

Ваш документ було перевірено за такими джерелами:

- ✓ База даних інтернет-джерел
- ✓ База даних наукових статей
- ○ Глибока перевірка (наш вдосконалений алгоритм)

Бали



Збіги тексту	18%
Перефразування	3%
Цитований текст	0%
Неправильне цитування	0%
Збігів не знайдено	79%

Ризик плагіату НАЙВИЩИЙ

Ризик плагіату вказує, як збіги тексту розподілені по документу. Вищий ризик виникає, коли збіги з'являються близько один до одного, наприклад, у тому самому абзаці або розділі.

Оцінка схожості % 21

Оцінка схожості показує, скільки слів або символів у вашому документі збігаються з текстами інших документів, включаючи перефразовані тексти або неправильні цитати.

Збіги

1 ЗАГАЛЬНА ІНФОРМАЦІЯ ТА АНАЛІЗ ПІДХОДІВ ДО ПРОЕКТУВАННЯ І 15 НАВЧАННЯ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ

15 Штучні нейронні мережі (ШНМ) – це алгоритми, котрі надають комп'ютерам змогу "вчитися", розпізнаючи визначені структури всередині даних. Вони сконструйовані на основі роботи людського мозку: кожна ланка мережі – це щось на кшталт "нейрона", що приймає дані, обробляє їх та пересилає далі.

Щоб нейронна мережа працювала, вхідні дані – текст, зображення чи звук – мусять бути перетворені на числову форму. У такому вигляді інформація прямує через три ключові шари: вхідний шар (де дані потрапляють до системи), прихований шар або декілька таких шарів (де здійснюється основна обробка), та вихідний шар (де отримуємо кінцевий результат).

Найбільш цікавим є те, що така мережа не має заздалегідь встановлених інструкцій – вона набуває "досвіду" у процесі навчання, тобто змінює свої внутрішні зв'язки (ваги між нейронами) згідно з успішністю вирішення задачі. З часом вона починає дедалі точніше передбачати або класифікувати дані.

Кожне з'єднання між нейронами має свою "вагу" – чисельне значення, котре вказує на важливість певного сигналу. Коли загальна сума сигналів, що приходять до нейрона, перевищує певне значення, нейрон активується й надсилає сигнал далі. Усе це відбувається у кілька етапів, аж поки дані не дійдуть до вихідного шару.

Отже, нейронні мережі не тільки обробляють інформацію – вони вчаться та покращуються, стаючи потужним інструментом для вирішення важких задач, зокрема – для аналізу тексту природною мовою, що й є темою цієї роботи.

1.1Виникнення штучних нейронних мереж

Ідея про те, як працюють нейрони, вперше з’явилася в 1943 році завдяки нейрофізіологу Уоррену Мак-Каллоху та математику Уолтеру Пітсу. У їхній спільній праці було запропоновано просту модель штучної нейронної мережі (ШНМ), яку було втілено у вигляді електричних кіл. Використовуючи поєднання математичних методів

та логічних схем, що їх вони охрестили як "порогову логіку", ці дослідники спробували відтворити елементарний процес мислення [1].

Ця праця послужила основою для подальших досліджень та розколола розробку нейронних мереж на два головні напрями: вивчення біологічних принципів функціонування мозку та створення моделей для обчислювальних систем. Саме ця концепція започаткувала сучасні технології штучного інтелекту, що нині широко використовуються у галузі аналізу даних, розпізнавання мовлення, зображень і тексту. На рисунку 1.1 наведено стуктуру нейрона.

Ядро

Ядро

Мієлінова оболонка

Мієлінова оболонка

Шваннівська клітина

Шваннівська клітина

Перехоплення Ранв’є

Перехоплення Ранв’є

Тіло нейрона

Тіло нейрона

Дендрит

Дендрит

Аксон

Аксон

Термінали Аксону

Термінали Аксону

Рисунок 1.1 - Структура нейрона

Наступним важливим етапом у становленні нейронних мереж стали дослідження Дональда Гебба, представлені в його праці "Організація поведінки" у 1949 році. Він

показав, що нервові шляхи посилюються з кожним використанням. Це стало наріжним каменем для розуміння процесів навчання, як у живих організмах, так і в штучних нейронних мережах. Відкриття зробило вагомий внесок у теорію нейропластичності, демонструючи зміни нейронних зв'язків під впливом досвіду. Саме Гебб запропонував один з перших ефективних алгоритмів навчання нейронних мереж, заклавши основу для подальшого розвитку штучного інтелекту [2].

У 1957 році Джон фон Нейман запропонував імітувати функції нейронів за допомогою телеграфних реле або вакуумних трубок, що стало значною віхою в розвитку концепції штучних нейронних мереж. З розвитком комп'ютерних технологій, у 1959 році вчені зі Стенфордського університету розробили дві ключові моделі — "ADALINE" та "MADALINE". "ADALINE" була зосереджена на обробці та розпізнаванні бінарних шаблонів, зокрема для прогнозування наступного біта інформації в телефонних лініях, тоді як "MADALINE" використовувалася для усунення ефекту відлуння, що стало першим практичним застосуванням штучної нейронної мережі для розв'язання реальних проблем.

Того ж року Френк Розенблатт, спираючись на перцептрон створив один з перших нейрокомп'ютерів "Марк-1", здатний розпізнавати кілька літер англійського алфавіту. Математична модель перцептрона, запропонована Розенблатом у 1957 році, справедливо вважається однією з перших штучних нейронних мереж. На рисунку 1.2 наведено структуру перцептрона.

Рисунок 1.2. Структура перцептрона

Перші успішні експерименти з перцептронами викликали хвилю наукових пошуків та сповнені оптимізмом передбачення щодо створення інтелектуальних систем, які мали б значно перевершувати здібності людського мислення. Але у 1969 році світ побачила книга "Перцептрон" Марвіна Мінського та Сеймура Пейперта, де було розглянуто обмеження та вади цієї технології. Як результат, автори дійшли висновку про низьку перспективність подальших розробок у сфері перцептронів. Труднощі проектування та використання, поряд з низькою продуктивністю та критикою значної частини таких систем, призвели до відчутного скорочення фінансування у цій галузі. Це, у свою чергу, зумовило "заморожування" розвитку штучних нейронних мереж на більше десятиліття [3].

У 1972 році Теуво Кохонен представив новий різновид нейронних мереж, які здатні функціонувати як пам'ять, відомі як самоорганізаційні карти Кохонена (рисунок 1.3.). Кохонен працював над теорією асоціативної пам'яті та запропонував 2 спосіб навчання ШНМ з учителем для мереж векторного квантування [4].

Рисунок 1.3 - Самоорганізаційна Карта Кохонена

Відродження зацікавлення до нейронних мереж стало реальністю у 1982 році завдяки Джону Гопфілду, який представив свою розробку, відому як Мережа Гопфілда. Йому вдалося вирішити проблеми попередніх поколінь ШНМ та оптимізувати їхню роботу, що стало поштовхом до відновлення досліджень і розробок нейрокомп'ютерів.

Алгоритм зворотного поширення, хоч і був запропонований раніше, у 1970-х роках, набув широкого визнання лише у 1986 році завдяки Девіду Румельгарту, Джеффрі Гінтону та Рональду Вільямсу. Дослідникам вдалося продемонструвати, що зворотне поширення працює значно швидше, ніж попередні методи навчання, що дозволило застосовувати нейронні мережі для розв'язання задач, які раніше здавалися неможливими без такого підходу. Сьогодні алгоритм зворотного поширення залишається одним з найпопулярніших методів навчання нейронних мереж.

У 1985 році Американський інститут фізики організував щорічну зустріч під назвою "Нейронні Мережі в Обчисленнях", після якої відбулася перша у світі міжнародна конференція з нейронних мереж, проведена Інститутом інженерів з електротехніки та електроніки (IEEE) у 1987 році. Конференцію відвідало понад 1800 фахівців.

У 1987 році США, Японія та Європа розпочали масштабне фінансування досліджень у сфері ШНМ, що суттєво сприяло розвитку технології. З 1989 по 1990 роки дослідження штучних нейронних мереж набули великої популярності, і ці дослідження активно вели найбільші електротехнічні компанії.

Протягом 1992-1998 років дослідження ШНМ продовжували стрімко розвиватися, відбувалися міжнародні форуми та конференції, зростала кількість спеціалізованих наукових публікацій, а також кількість фахівців у цій галузі.

1999 рік став важливим етапом завдяки появі графічних прискорювачів (GPU) та прогресу в комп'ютерних технологіях, що дозволило суттєво прискорити обробку даних. Це значно підвищило ефективність ШНМ, і вони почали конкурувати з опорними векторними машинами (SVM). Незважаючи на те, що нейронні мережі можуть бути повільнішими за векторні машини, їхньою перевагою є здатність покращуватися зі збільшенням обсягу даних, що дозволяє досягати значно кращих результатів з тими ж наборами даних.

У 2001 році у дослідницькому звіті компанії Gather були описані потенційні можливості та проблеми, пов’язані зі збільшенням тривимірних даних через збільшення кількості джерел та типів даних. Цей звіт став першим закликом до

підготовки до ери Big Data (Великі Дані).

У 2007 році Джеффрі Гінтон в університеті Торонто розробив алгоритми для глибинного навчання багатошарових нейронних мереж, використовуючи обмежену машину Больцмана (RBM) для тренування нижніх шарів мережі. Це було важливим досягненням, яке зробило глибинне навчання більш ефективним і поширеним на практиці [5].

Значне покращення продуктивності графічних процесорів протягом наступних десяти років дозволило з 2011 року тренувати згорткові нейронні мережі (CNN) без попереднього навчання шар за шаром. З ростом швидкості обробки даних стало очевидно, що глибинне навчання значно перевершує традиційні методи. Одним з найвідоміших прикладів успіху є мережа AlexNet, згорткова нейронна мережа, яка виграла декілька міжнародних змагань у 2011–2012 роках.

У 2016 році команда розробників з компанії DeepMind, що належить Google, за допомогою своєї нейронної мережі AlphaGo змогла перемогти чемпіона світу з Ґо — китайської настільної гри, яка вважається значно складнішою за шахи.

На сьогодні штучні нейронні мережі продовжують активно розвиватися. Вчені з усього світу працюють над новими алгоритмами навчання та інноваційними архітектурами. Вже існують сотні комерційних та некомерційних проектів, які довели свою ефективність у різних сферах.

1.2 Функції активації нейронів

Найважливішою складовою штучної нейронної мережі є функція активації. Вона диктує вихідні дані (результат) моделі, її точність та продуктивність навчання. Функція активації перетворює підсумований вхідний сигнал на конкретний вихідний, що передається нейроном наступному шару або рівню.

Ключовим параметром функції активації є її дієвість, враховуючи її застосування до сотень, тисяч чи навіть мільйонів нейронів у масштабних мережах. Алгоритм навчання зі зворотним розповсюдженням помилки (backpropagation) ще більше збільшує навантаження на функцію активації, оскільки саме вона вирішує, наскільки швидко мережа скоригує свої ваги під час навчання.

Однією з найвідоміших функцій активації є сигмоїдна функція. Її діапазон значень переважно від -1 до 1, хоча зустрічаються варіації з діапазоном від 0 до 1. Ця функція активно використовується в багатьох задачах, зокрема в моделях, де результат можна представити як ймовірність, або в бінарних класифікаціях. Однак сигмоїдна функція має обмеження, наприклад, проблему "зникаючого градієнта", яка може виникати при

навчанні глибоких мереж, особливо з великими вхідними значеннями або надто глибокою структурою мережі. На рисунку 1.4 можна побачити 2 графік сигмоїдних функцій активації.

2 Рисунок 2 1.4 - 2 Графік сигмоїдних функцій активації

2 Ще 2 однією 2 популярною функцією активації є ReLU 11 (Rectified Linear Unit, Випрямлений Лінійний Блок). Це одна з найпоширеніших функцій активації, яка використовується в багатьох сучасних моделях нейронних мереж завдяки своїй простоті та ефективності. Вона описується наступною формулою:

де ???? — це вхідний сигнал нейрона.

ReLU має кілька варіантів і підвидів, серед яких найпоширенішими є:

ReLU Softplus: цей варіант є плавною версією стандартної ReLU функції, яка використовує формулу:

Це сприяє уникненню проблеми "нейронів-мертвяків", коли вихідне значення фіксується на нулі 11 для всіх від'ємних вхідних даних.

ReLU Rectifier (або Parametric ReLU): у цьому випадку негативні значення опрацьовуються з додатковим параметром, що дозволяє гнучкіше регулювати функцію активації.

Функції ReLU та її різновиди можна візуалізувати на рисунку 1.5, де показано їх графічне зображення.

Рисунок 1.5 - 2 Графік функцій активації ReLU: Softplus та Rectifiler

2 Не дивлячись на існування численних альтернативних рішень для ReLU, ця проста й ефективна функція не зазнала повної заміни. Її основна перевага полягає в здатності значно скоротити час навчання та зберігати високу продуктивність у задачах глибинного навчання, не стикаючись із серйозними проблемами, як-от зникнення градієнта, характерного для сигмоїдальних функцій.

2 У 2017 році команда Google Brain презентувала нову активаційну функцію, відому як Swish (схематичний вигляд наведено на рисунку 1.6). Вона дещо нагадує ReLU, але у певних ситуаціях демонструє значно кращі показники. В експериментах із 40-шаровою штучною нейронною мережею, де застосовувалися 2 ReLU та Swish, результати обох функцій були приблизно однаковими. Проте зі збільшенням кількості шарів і обсягу навчальних 2 даних, Swish значно перевершила ReLU. Дослідникам вдалося покращити точність аж на 0.9% у деяких задачах.

Математичний опис цієї функції має такий вигляд:

f(x)=x□sigmoid(x) (1.2)

де sigmoid(x)=1/(1+exp(-x))

У 2 2018 році дослідник Ерік Алкайде презентував модифікований варіант функції Swish, названий ним E-swish. У математичному виразі ця функція визначається за допомогою формули:

f(x)=βx□sigmoid(x), (1.3)

де sigmoid(x)=1/(1+exp(-x))

Характерні риси E-swish суголосні з Swish, і за умови β=1 вона абсолютно тотожна Swish. Однак, підсумки дослідів вказали, що E-swish демонструє покращену точність нейромережі на 0.35% у порівнянні з ReLU та на 0.6% проти Swish. На рисунку 1.6. продемонстровано графіки Swish, E-Swish, та інших популярних функцій активації.

Рисунок 1.6 - Графіки Swish, E-Swish, та інших популярних функцій активації

Раніше досить популярна, але зараз застосовується рідше, функція Binary step може приймати лише два значення: 0 та 1. Вона повертає значення 1 (істина), коли вхід перевищує заданий поріг, і 0 (хиба), якщо вхід не досягає цього порогу. Функцію Binary step часто використовують у задачах бінарної класифікації. Формула, що описує цю ступінчасту функцію:

Майже всі класичні логічні функції піддаються реалізації через штучні нейронні мережі. Відтак, порогові функції переважно застосовуються у простих нейромережах без прихованих шарів або ж у одношарових мережах. Цей тип мережі здатен розв'язувати задачі з лінійно роздільними даними, зокрема логічні операції AND та OR. Інакше кажучи, всі можливі класи (0 та 1) можуть бути розділені за допомогою єдиної прямої лінії. На рисунку 1.7 наведено графік функції Binary step.

Рисунок 1.7 - Графік функції Binary step

Існує величезна кількість функцій активації для штучних нейронів, і їхній набір безперервно розширюється. Вибір конкретної функції залежить від багатьох аспектів, включаючи специфіку поставленої задачі, що розв'язується нейронною мережею, та особливостей процесу її тренування. Сьогодні серед найбільш популярних та ефективних функцій варто виділити ReLU та Swish.

Сигмоїдні функції та їхні комбінації часто демонструють добрі результати при вирішенні задач класифікації. Однак в деяких ситуаціях їх краще уникати через проблему

зникнення градієнту.

ReLU — одна з найпоширеніших функцій активації у різних штучних нейронних мережах. Вона є стандартним вибором для багатьох моделей, нерідко забезпечуючи найкращу продуктивність. Проте, її рекомендується використовувати лише в прихованих шарах мережі.

У деяких випадках ефективною альтернативою ReLU є Swish та E-swish, які можуть показувати значно кращі результати за певних обставин. Графіки цих функцій характеризуються більш плавним вихідним ландшафтом (рисунок 1.8), що суттєво полегшує оптимізацію та обчислення помилок. Плавність ландшафту помилок безпосередньо впливає на ефективність процесу навчання, тому чим плавніший графік, тим простіше віднайти та обчислити оптимальну помилку.

Рисунок 1.8 - 1 Візуалізація вихідного ландшафту Swish та ReLU

1 1.3. Архітектура нейронних мереж

1 Перцептрон 1 є першим поколінням 1 штучних нейронних мереж. Це проста обчислювальна модель одного нейрона, математична модель якої була запропонована Ф. Розенблатом на початку 1960-х 1 років. 1 Стандартна архітектура перцептрона – це мережа прямого поширення (feedforward neural network), де вхідні дані надходять до 1 нейрона, 1 обробляються, і результат передається на вихід. На рисунку 1.9 можна побачити архітектуру багатошарового перцептрона (MLP).

Рисунок 1.9 - 1 Архітектура багатошарового перцептрона (MLP)

1 Цей архітектурний підхід передбачає обробку даних у мережі з лівого краю до правого. Суть в тому, що числова інформація проходить через систему, де над нею виконуються різні дії. Щоб ці дії були коректними та кожен вхід давав потрібний результат, необхідне 1 навчання перцептрона. Навчання полягає у визначенні оптимальних параметрів, котрі забезпечать найкращі показники, та їх впровадженні в мережу.

Перцептрон знаходить застосування в різних сферах, наприклад, у комп'ютерному зорі (системи розпізнавання об'єктів на цифрових 1 зображеннях та відео), в обробці природної мови, а також як основа для побудови складніших нейронних мереж.

Проте, перцептрон має значні обмеження. 1 Він не може розпізнавати 1 патерни після певних перетворень. 1 Марвін Мінський та Сеймур Пейперт, критикуючи перцептрон, відзначають, що його навчена частина не може адаптуватися, 1 якщо перетворення утворюють групи патернів. Для вирішення цієї проблеми перцептрону

необхідно використовувати кілька функціональних блоків для розпізнавання змінних інформативних підпатернів. Отже, для вирішення складних завдань розпізнавання патернів варто використовувати спеціальні, ручно запрограмовані детектори, замість того, щоб покладатися виключно на процес навчання.

На початку нульових років команда науковців під керівництвом Я. ЛеКуна створила нейромережу, що мала на меті безпомилково розпізнавати рукописні цифри. Назву їй дали LeNet. Для досягнення цієї мети дослідники використали алгоритм зворотного поширення похибки у зворотно-згортковій мережі, яка містила багато 1 прихованих шарів та велику кількість карт повторюваних елементів. Вони інтегрували вихідні дані з сусідніх копій у розгалужену мережу, що дозволяло одночасно обробляти декілька символів, навіть за умови їх часткового накладання [6]. Згодом ця архітектура отримала офіційну назву - 1 згорткова нейронна мережа (CNN). На 1 рисунку 1.10 наведено архітектуру 1 CNN для розпізнавання рукописних цифр.

1 Рисунок 1 1.10 - Архітектура CNN

1 Згорткові нейронні мережі (CNN) 1 продовжують активно використовуватися в сучасності 1 для вирішення завдань розпізнавання та класифікації. Вони застосовуються, зокрема, для розпізнавання чисел, ідентифікації об'єктів та обробки 1 природної мови. CNN оперують трансформованою версією базової функції, інтегруючи її для створення функцій, які нечутливі до зсувів. Використання однієї базової функції в різних позиціях сприяє зменшенню кількості навчальних параметрів, що дає змогу ефективніше навчати мережу 12 з меншою кількістю тренувальних прикладів, порівняно з ситуацією, де кожна позиція вивчалася б окремо за допомогою різних базових функцій 1 [7].

1 Згорткові мережі мають значні відмінності від інших видів мереж. Їхнє основне призначення — це класифікація зображень, але їх також успішно використовують 1 для обробки звукової інформації. Найбільш поширеним випадком використання CNN є аналіз зображень, які надходять, наприклад, з інтернету, з метою їх подальшої класифікації. Здебільшого CNN отримують дані через вхідний сканер (рисунок 1.11), 1 який не призначений для одночасного аналізу всіх даних.

Рисунок 1.11 - 1 Приклад роботи вхідного сканера CNN

1 Наприклад, 1 якщо маємо картинку розміром 1 50 на 50 пікселів, щоб уникнути створення вхідного шару з 2500 вузлами (сумарна кількість пікселів зображення), можна організувати вхідний шар-сканер, припустимо, розміром 5 на 5. Спочатку на цей шар надсилаються 1 перші 25 пікселів зображення. Далі 1 сканер зміщується 1 на один піксель праворуч, і 1 на вхід подаються 1 наступні 25 пікселів. Такі вхідні дані сканування передаються через згорткові шари замість звичайних шарів, де з'єднання

між вузлами повні. У згорткових шарах кожен вузол працює лише з найближчими сусідніми клітинками. 1 Ці згорткові шари, зазвичай, зменшуються в розмірі, оскільки з глибиною мережі їх структура спрощується, зокрема, за рахунок зменшення вхідних даних.

Також CNN мають властивість трансляційної інваріантності, що є невід'ємним аспектом їхньої архітектури. Але недолік такого підходу 1 полягає в тому, що архітектура може обробляти лише трансляційну інваріантність [8]. Скажімо, мережа не зможе об'єднати частини зображення, що представляють собою ті самі об'єкти, але повернуті, або розпізнати 1 складніші варіації, такі як повороти поза площиною.

1 Рекурентні нейронні мережі (RNN) відрізняються від мереж прямого поширення тим, 1 що є різновидом 1 рекурсивної мережі, де з'єднання 1 між вузлами створюють циклічну структуру. 1 Це означає, що результат роботи мережі залежить не тільки від поточних 1 даних, але й 1 від стану нейронів з попереднього циклу. Така «пам'ять» дозволяє використовувати їх для вирішення завдань обробки природної мови (NLP), включаючи розпізнавання рукописного тексту або обробку мовних сигналів.

RNN зберігають інформацію про минуле, і її рішення залежать від 1 попередніх результатів. Інші типи мереж також «запам'ятовують» дані, але лише те, що було засвоєно 1 під час навчання. Наприклад, мережа для розпізнавання 1 цифр вчиться розрізняти цифру «5» 1 під час навчання, а потім використовує ці знання для класифікації під час роботи. Натомість, RNN, крім навчання під час тренування, зберігають інформацію, отриману від попередніх вхідних даних, що дозволяє генерувати 1 вихідні результати.

1 RNN можуть приймати один або декілька вхідних векторів і генерувати один або декілька вихідних векторів. На 1 результат впливають не лише ваги, які застосовуються до входів, як у 1 звичайних нейронних мережах, але й прихований вектор стану, що 1 формує 1 контекст на основі попередніх 1 даних [9]. Через це, 1 один і той самий вхід може давати різні 1 результати 1 залежно від попередніх входів мережі.

1 На рисунку 1.12. наведено рекурентну нейронну мережу 1 із прихованим шаром, де: – вхідні вузли, – приховані вузли, yn – вихідні вузли, – ваги.

1 Рисунок 1 1.12 - 1 Рекурентна нейронна мережа із прихованим шаром

Команда китайських науковців: К. Лю, С. 1 Лай та Д. Джао у своїй 1 праці "Генерування природної мови, перефразування 1 та узагальнення відгуків користувачів з використанням 1 рекурентних нейронних мереж" продемонструвала RNN-модель 1 для класифікації тексту, уникнувши необхідності задавати

класифікаційні ознаки вручну. Вони здійснили порівняльний аналіз своєї мережі з іншими 1 методами класифікації тексту, зокрема LR, SVM, LDA, RNN та CNN. Отримані результати засвідчили, що їхня модель демонструє значно кращу ефективність у вирішенні поставлених завдань, випереджаючи традиційні методи на всіх використаних наборах даних.

Узагальнюючи, стандартні нейронні мережі трансформують 1 вхідний вектор фіксованого розміру у вихідний вектор того ж фіксованого розміру. Проте, якщо мережа багаторазово 1 застосовує перетворення до серії вхідних 1 даних і формує 1 серію вихідних векторів, таку мережу правомірно називати рекурентною (рисунок 1.13). Варто наголосити на відсутності заздалегідь визначених обмежень щодо розміру вектора. Додатково, під час генерації результату, залежного від 1 вхідного та прихованого стану, відбувається оновлення прихованого стану на основі отриманої вхідної інформації, що потім застосовується при обробці наступних вхідних даних.

Мережа Елмана

Мережа Джордана

Вхідний шар

Прихований шар

Вихідний шар

Вхідний шар

Прихований шар

Вихідний шар

Шар стану

Шар стану

Рисунок 1.13 - 1 Приклад рекурентних мереж Елмана та Джордана

1 1.4. Навчання штучних нейронних мереж

1 Першим кроком навчання є пряме поширення, яке здійснюється завдяки активаційним функціям. В рамках цього процесу, мережа навчається, використовуючи тренувальні дані, 1 що проходять через всю нейронну мережу з метою прогнозування результатів. Інакше кажучи, вхідні дані транслюються 1 через мережу таким чином, що кожен 1 нейрон обробляє інформацію, отриману 1 від нейронів попереднього

шару, шляхом 1 певного перетворення, після чого передає 1 її нейронам наступного шару. Коли дані проходять всі шари, а всі нейрони виконують свої обчислення, формується фінальний результат, який представляє собою прогноз міток для вхідних зразків [10].

Наступним етапом є 1 використання функції втрат для оцінки помилки, яка показує, наскільки точним чи неправильним 1 був прогноз мережі відносно коректного результату. В ідеальному випадку, втрата має дорівнювати нулю, тобто 1 між отриманим та очікуваним результатом не має бути жодних розбіжностей. Тому, в процесі навчання моделі, ваги зв'язків між нейронами поступово коригуються до моменту отримання точних прогнозів.

1 Після обчислення втрати, ця інформація поширюється у зворотному напрямку через мережу. Звідси походить назва процесу: 10 зворотне поширення помилки. Розпочинаючи 1 з вихідного шару, дані про втрати передаються до нейронів прихованих шарів, які безпосередньо впливають на вихід. Однак, нейрони прихованих шарів отримують лише певну 1 частину загального сигналу втрати, що залежить від внеску кожного нейрона у результат. 1 Цей процес повторюється пошарово, поки всі нейрони не отримають сигнал втрати, який демонструє 1 їх відносний вплив на 1 загальну помилку. Зворотне поширення 10 помилки можна розуміти як механізм корекції параметрів (ваг та 10 зміщень) нейронної мережі у потрібному напрямку. Процес починається 1 з обчислення втрати, після 1 чого параметри 1 мережі коригуються у зворотному порядку за допомогою алгоритму оптимізації з врахуванням цієї втрати.

Після зворотного поширення інформації необхідне налаштування ваг 1 зв'язків між нейронами. Для цього використовується метод, відомий як градієнтний спуск. Цей метод здійснює корекцію ваг невеликими кроками, обчислюючи 1 похідну (градієнт) функції втрат, що допомагає визначити напрямок, у якому слід "спускатись", щоб досягти глобального мінімуму. Градієнтний спуск є одним 10 з найбільш поширених 1 алгоритмів оптимізації в машинному та глибокому навчанні. Процес 1 передбачає прив'язку 1 похідних втрат кожного прихованого шару до похідних втрат його вищого 1 шару та включення 1 його функції активації у 1 обчислення (відтак 1 функції активації повинні мати 1 похідні).

1 Рисунок 1.14 - 1 Візуалізація градієнтного спуску

1 Під 1 час кожного повторення, 1 як тільки всі нейрони отримують величину градієнта функції втрат, що відповідає їхньому внеску, відбувається корекція значень параметрів у напрямку, протилежному напрямку градієнта. Це обумовлено тим, що градієнт завжди показує напрямок зростання 1 функції втрат [10]. Відтак, застосування

від'ємного градієнта дає можливість виявити напрям, де 1 значення функції втрат знижується, що й забезпечує ефективне навчання моделі.

Таким чином, кроки, які слід виконати для тренування 1 штучної нейронної мережі, такі:

• Ініціалізувати параметри мережі (ваги та зсуви) випадковими величинами.

• Вибрати навчальний набір даних та пропустити його через мережу для отримання передбачень.

• Зіставити одержані прогнози 1 з очікуваними значеннями та обчислити втрати (похибку).

• Здійснити зворотне поширення помилки, щоб визначити вплив кожного з параметрів на сукупні втрати.

• Відкоригувати параметри мережі з використанням методу градієнтного спуску на підставі отриманих градієнтів, щоб зменшити втрати та поліпшити якість моделі.

• Повторювати ітерації до досягнення бажаного рівня точності мережі.

1 2. ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

1 2.1. Бібліотеки обробки природної мови

1 Обробка природної мови (Natural Language Processing, NLP) – це автоматизований процес 1 опрацювання природної людської мови, включаючи 1 текст та мовлення. 1 Цінність цієї технології полягає у широкому полі застосування, яке постійно розширюється. Нижче наведено приклади практичного використання NLP у різних сферах:

• Охорона здоров’я. 1 NLP дозволяє розпізнавати та прогнозувати захворювання, використовуючи 1 електронні медичні записи. Наприклад, сервіс 1 Amazon Comprehend Medical використовує NLP для виявлення згадок про хвороби, медикаменти та результати лікування у записах пацієнтів, звітах клінічних досліджень та інших даних. Ці інструменти допомагають у діагностиці серцево-судинних захворювань, депресії та інших розладів.

• Аналіз настроїв у соціальних мережах. NLP дає змогу організаціям дізнаватися про те, що клієнти думають про їхні послуги чи продукти, аналізуючи коментарі у соціальних мережах. Такий аналіз дозволяє зрозуміти вподобання споживачів, ринкові тенденції та реакцію на новинки.

• Персоналізовані пошукові асистенти. Один з винахідників IBM створив віртуального помічника, що функціонує 1 як персоналізована пошукова система. Цей асистент вивчає звички користувача та може нагадати ім'я, пісню або іншу важливу інформацію у потрібний момент.

• Фільтрація електронної пошти. 1 Такі компанії, як Yahoo та Google, застосовують NLP для класифікації електронної пошти та блокування спаму. Алгоритми аналізують текст повідомлень, щоб заздалегідь виявити небажані листи та не допустити їх потрапляння до вхідної скриньки користувача.

• Інтелектуальні голосові інтерфейси. Пристрої, як-от Amazon Alexa та Apple Siri, 1 використовують NLP для обробки голосових команд. Вони здатні знаходити магазини, надавати інформацію про погоду, пропонувати маршрути та виконувати інші завдання, реагуючи на природну людську мову.

• Фінансова аналітика. У сфері торгівлі 1 NLP використовується для відстеження новин, фінансових 1 звітів та публікацій 1 про можливі злиття компаній. Такі дані інтегруються в торгові алгоритми, покращуючи прийняття рішень трейдерами.

1 За останні роки NLP зазнав значного розвитку (рисунку 2.1), особливо у сфері 1 охорони здоров'я. Ця технологія не 1 лише підвищує ефективність 1 надання медичної допомоги та точність діагнозів, але й сприяє зменшенню витрат. Широке використання 1 електронних медичних записів у медичних установах робить NLP незамінним інструментом для покращення клінічної документації, що, в свою чергу, дає змогу краще розуміти пацієнтів та забезпечувати якісніше лікування.

Рисунок 2.1 - 1 Кількість публікацій в сфері NLP у період 1978 – 2018 років

1 SpaCy 1 — це сучасна 1 бібліотека обробки природної мови (PNL), 1 що 1 створена для роботи з Python. Вона надає дієвий та адаптивний набір інструментів для детального дослідження текстових даних, що донедавна не був доступним у більшості звичайних PNL-рішень [11]. На рисунку 2.2 можна побачити архітектуру бібліотеки SpaCy.

Рисунок 2.2 - 12 Архітектура бібліотеки SpaCy

12 Дана бібліотека вміє розпізнавати складні граматичні категорії та вміло знаходити текстові зразки, уникаючи надмірних повторів, застосовуючи попередньо навчені нейронні мережі (рисунок 2.3).

Рисунок 2.3 - Алгоритм навчання нейронної мережі 1 SpaCy

1 SpaCy - це потужний інструмент для виконання складних операцій з обробки

природної мови. Вона вміє виконувати токенізацію (розбиття тексту на окремі токени) та забезпечує поглиблений аналіз текстової інформації. Значною перевагою є підтримка лематизації — надання слову його базової (словникової) форми. SpaCy вирізняється з-поміж інших бібліотек високого рівня тим, що реалізує цю функцію з високою точністю.

Аналіз частин мови (POS-tagging), який застосовує SpaCy, характеризується високою швидкістю та точністю, роблячи цю бібліотеку однією з найшвидших у світі у галузі синтаксичного аналізу. SpaCy поширюється 4 з відкритим вихідним кодом під ліцензією MIT, що дає змогу легко інтегрувати бібліотеку, адаптувати її під індивідуальні потреби користувачів та широко застосовувати у задачах глибокого навчання та обробки природної мови.

Основні функції SpaCy:

• Токенізація;

• POS-tagging (розпізнавання частин мови);

• Класифікація тексту;

• Лематизація;

• Виявлення граматичних зв’язків;

• Векторне представлення слів (Word Embeddings).

Користувачі бібліотеки SpaCy. здебільшого дослідники даних, котрі мають справу з великими текстовими масивами, виявляють загальні тенденції та створюють високопродуктивні додатки з покращеними передбачувальними властивостями. Орієнтована на практичне застосування, бібліотека активно використовується не лише в наукових розробках, а й при розробці масштабних корпоративних систем, що опрацьовують значні обсяги текстових даних з метою виявлення прихованих семантичних структур.

SpaCy розроблено з акцентом на обробку 6 великих обсягів даних, де важлива швидкість обробки разом із точністю. Утім, для задач, які не передбачають виробничого використання або складної аналітики, ця бібліотека може виявитись надлишковою. Водночас, її чудово структурована та зрозуміла документація значно полегшує інтеграцію складних алгоритмів обробки тексту. Навіть без звернення до документації, завдяки добре продуманій архітектурі, бібліотека інтуїтивно зрозуміла для фахівців, які не мають поглибленої лінгвістичної підготовки.

SpaCy створює потужну екосистему інструментів для обробки природної мови,

спрямовану на практичне застосування. Бібліотека дає можливість розробляти як корпоративні рішення, так і кастомізовані NLP-системи для кількох мов. Вона сумісна з сучасними методами глибокого навчання, залишаючись при цьому у форматі відкритого коду та доступною для всіх.

Варто відзначити:

• Сучасний механізм розпізнавання іменованих сутностей (NER),

• Високу точність морфологічного аналізу,

• Надзвичайно швидкий синтаксичний аналіз.

Сильні сторони бібліотеки SpaCy:

• Орієнтація на використання у виробничих умовах.

• Простота використання та швидка інтеграція.

• Один з найшвидших у світі синтаксичних аналізаторів.

Слабкі сторони:

• Обмежена кількість мов, що підтримуються.

• Відносна негнучкість порівняно з іншими бібліотеками (наприклад, NLTK).

• Менший набір інструментів для обробки, якщо порівнювати з деякими іншими NLP-рішеннями.

TextBlob — це Python-бібліотека й API для обробки природної мови (NLP), котра вдало поєднує в собі простоту та потужність. Працює як з Python 2, так і з Python 3. Завдяки багатому функціоналу, бібліотека здатна задовольнити більшість потреб з видобутку інформації з тексту — від розпізнавання іменників до аналізу емоційної складової. TextBlob поширюється безкоштовно з ліцензією GPL. Хоча офіційної підтримки за гроші немає, навколо неї існує активна спільнота розробників, котрі регулярно оновлюють та виправляють помилки.

Бібліотека розроблена спеціально для Python, тому інтегрується з загальними підходами до роботи з текстом у цьому середовищі. TextBlob забезпечує швидке прототипування та може легко розгортатися за допомогою стандартного інтерпретатора Python. 6 Це дає змогу швидко отримати доступ до ключових інструментів NLP без складних конфігурацій.

TextBlob не є виключно інструментом для початківців, вона містить розширені функції,

серед яких визначення мови та інтеграція з Google Translate, що дає змогу майже миттєво перекладати текст. Також бібліотека підтримує базову синтаксичну обробку, як-от побудова дерева залежностей, а рівень абстракції дозволяє реалізувати навіть складні операції всього за кілька рядків коду. Наприклад, аналіз настроїв реалізується так само просто, як і базова токенізація.

TextBlob також має функцію виправлення орфографії, автоматично пропонуючи найбільш імовірні варіанти виправлень. Хоча основна функціональність дуже проста, за потреби користувач може отримати більш деталізовані дані. Це надає гнучкість бібліотеці для різних рівнів користувачів.

Основні можливості TextBlob:

• Токенізація.

• POS-тегування 1 (визначення частин мови).

1 • Класифікація тексту.

• Аналіз настроїв.

• Пошук граматичних залежностей.

• Перевірка орфографії.

• Інтеграція з Google Translate.

• Інтеграція з WordNet.

1 Переваги бібліотеки TextBlob:

1 • Простота у використанні.

• Підтримка перекладу та визначення мови.

• Вбудована перевірка орфографії.

• Гнучкість та зручність для швидкого прототипування.

Недоліки:

• Повільніша швидкість обробки, порівняно з іншими бібліотеками.

• Обмежений функціонал — для складних задач потрібне поєднання з іншими NLP-бібліотеками.

• Відсутність векторного представлення слів.

TextBlob — відмінний інструмент для швидкого створення NLP-прототипів та вирішення типових завдань 1 з обробки тексту. Бібліотека орієнтована на легкість у використанні, пропонуючи зрозумілий API та скорочені цикли розробки, що робить її популярною серед дослідників, аналітиків та розробників.

1 2.2. Бібліотеки проектування та навчання нейронних мереж

1 Машинне та глибоке навчання (ML/DL) переживають вибуховий розвиток в останні роки, що радикально змінило потенціал 6 штучного інтелекту в багатьох сферах. Цьому сприяє злагоджена праця спільнот програмістів, котрі безперестанно вдосконалюють інструменти для розробки інтелектуальних систем. На сьогодні налічуються десятки бібліотек, спеціалізованих на машинному та глибокому навчанні, і над багатьма з них трудяться сотні активних розробників по всіх куточках світу (рисунок 2.4).

Рисунок 2.4 - 1 Графік бібліотек ML/DL за кількістю контрибюторів та правок

1 Близько 50 бібліотек досягли рівня зрілості, що дозволяє їх успішно застосовувати в реальних проєктах – як у наукових дослідженнях, так і у виробничих рішеннях. Вони надають широкий набір функціональних можливостей, включно з розробкою, навчанням, тестуванням та оптимізацією моделей штучного інтелекту. Найбільш вживані з них – TensorFlow, Keras, PyTorch, Theano, MXNet та інші, які активно задіюються для побудови нейронних мереж, зокрема в завданнях 16 обробки природної мови.

16 Машинне навчання (ML) застосовує універсальні математичні моделі для отримання відповідей на поставлені питання, використовуючи доступні дані. Ця технологія знайшла застосування у багатьох областях, включно з виявленням спаму в електронній пошті, розробкою інтелектуальних роботів, розпізнаванням об'єктів за допомогою комп'ютерного зору (CV), створенням систем "розумного будинку", розпізнаванням мовлення, генерацією текстів (зокрема, романів чи віршів), розробкою рекомендаційних систем для онлайн-магазинів, а також прогнозуванням цін на товари та послуги.

Раніше подібні завдання вирішувалися вручну – розробники самостійно писали математичні алгоритми та статистичні формули, що вимагало значних витрат часу, зусиль та спеціалізованих знань. Сьогодні ж машинне навчання стало значно доступнішим та ефективнішим завдяки появі великої кількості потужних бібліотек, зокрема для мови програмування Python.

Python є однією з найпопулярніших мов для реалізації рішень у сфері машинного

навчання, завдяки зрозумілому синтаксису, великій спільноті користувачів 4 та широкому спектру спеціалізованих бібліотек. Ці бібліотеки полегшують створення, навчання та використання моделей машинного та глибокого навчання.

Найбільш поширені бібліотеки Python для задач машинного навчання:

• TensorFlow – одна з найбільш потужних бібліотек для глибокого навчання, розроблена компанією Google.

• Keras – високорівнева надбудова над TensorFlow, зручна для швидкого прототипування моделей.

• PyTorch – популярна бібліотека з гнучкими динамічними обчисленнями графів, особливо популярна серед дослідників.

• Scikit-learn – універсальна бібліотека для традиційного машинного навчання (класифікація, регресія, кластеризація).

• NumPy – бібліотека для роботи з багатовимірними масивами та матрицями.

• Pandas – зручний інструмент для аналізу, очищення та обробки табличних даних.

• Theano – бібліотека, що дозволяє ефективно виконувати математичні операції над багатовимірними масивами.

• Caffe – бібліотека, орієнтована на застосування в комп'ютерному зорі.

• Seaborn – бібліотека для візуалізації статистичних даних.

• Matplotlib – одна з найстаріших бібліотек Python для створення графіків та діаграм.

TensorFlow – це бібліотека 4 з відкритим кодом, що використовується для числових розрахунків, організованих у вигляді графів потоку даних. Розвинута командою Google Brain, яка входить до дослідницької структури Google Machine Intelligence, вона головно спрямована на рішення завдань машинного навчання та конструювання нейронних мереж. Попри те, TensorFlow є універсальним інструментом, здатним знайти застосування в широкому спектрі сфер, де важлива висока обчислювальна потужність.

Стабільна версія 1.0 бібліотеки TensorFlow вперше побачила світ у лютому 2017 року. Відтоді проєкт динамічно розвивається, про що свідчить наявність більше ніж 21 000 контрибуторів на платформі GitHub [13], що є показником визнання серед дослідників і розробників.

TensorFlow надає необхідне підґрунтя для конструювання моделей як у традиційному

машинному навчанні, так і в глибинному навчанні. У випадку створення комплексних моделей другого типу, зокрема при великих обсягах вихідних даних, виникає необхідність у розподіленому навчанні. Архітектура TensorFlow, що базується на графах, забезпечує ефективний розподіл обчислень між різними обчислювальними елементами: центральними процесорами (CPU), графічними процесорами (GPU), а також у хмарних платформах.

На рисунку 2.5 наведено 1 процес навчання ШНМ в TensorFlow.

1 Рисунок 1 2.5 - 1 Процес навчання ШНМ в TensorFlow

1 З поточною версією TensorFlow для початку роботи потрібно створити обчислювальний граф, а вже потім його запустити. 1 Граф в TensorFlow представляє собою структуру даних, що повністю визначає обчислення, які необхідно 1 виконати [13]. Така структура надає ряд важливих переваг:

• Портативність: граф може бути негайно запущений або збережений для подальшого використання. Він здатний працювати на різних платформах, включаючи центральні процесори (CPU), графічні процесори (GPU), процесори для тензорів (TPU), а також мобільні платформи. Більше того, граф можна розгорнути 1 у виробництво без необхідності залежати від коду, який його побудував.

• Трансформація та оптимізація: граф можливо перетворювати, щоб отримати більш оптимальні версії для певної платформи. Це також дає змогу оптимізувати використання 1 пам'яті або обчислення, знаходячи 1 компроміс між ними.

1 • 1 Підтримка розподіленого виконання: API високого рівня TensorFlow разом 1 з обчислювальними графами забезпечують потужне 1 та гнучке середовище для розробки та можуть гарантувати масштабованість у виробничих умовах.

• Швидке виконання: ще одна суттєва перевага TensorFlow – підтримка імперативного стилю кодування. Активувавши функцію швидкого 1 виконання, ядра TensorFlow будуть виконуватися негайно, що значно полегшує налагодження коду, оскільки відсутня потреба будувати графи для пізнішого виконання.

Ці можливості роблять можливим легку перевірку та налагодження проміжних значень в графах та використання потоків керування 1 Python в API TensorFlow, включаючи цикли, умови, функції тощо. Підтримка швидкого виконання також значно спрощує процес налагодження. Після того, як код TensorFlow функціонує, його можна автоматично перетворити в граф, що полегшує збереження, розгортання та розповсюдження цих 1 графів.

1 TensorFlow та спільнота 4 з відкритим вихідним кодом: бібліотека 1 була відкрита

для загального використання, що дозволяє спільноті активно брати участь у покращенні. Команда TensorFlow налаштувала процеси керування запитами, переглядом та поданням проблем, а також надає відповіді на питання розробників стосовно цієї технології. TensorFlow має понад 76 000 зірок (оцінки користувачів) на GitHub, і кількість репозиторіїв, які використовують TensorFlow, збільшується щомісяця, досягнувши понад 20 000. Значна їх частина – це навчальні посібники, моделі, переклади та проєкти, створені спільнотою. Вони слугують прекрасним джерелом прикладів для новачків у сфері машинного навчання [13]. Команда TensorFlow активно відповідає на запитання користувачів на платформі Stack Overflow (система питань-відповідей для програмістів), де на сьогоднішній день налічується понад 8 000 відповідей. На рисунку 2.6 можна побачити графік популярності бібліотек машинного навчання.

Рисунок 2.6 - Графік популярності бібліотек машинного навчання

Отже, TensorFlow визначає суворі критерії для оцінювання продуктивності та зрозумілості програмного коду. Група розробників створила докладний комплекс рекомендацій, які ретельно інтегровані у документацію. Вони постійно взаємодіють з користувачами, беручи до уваги їхні коментарі та поради. TensorFlow пропонує відмінну підтримку різноманітних архітектур, забезпечуючи безперебійний запуск обчислень на багатьох платформах, від звичайних комп'ютерів до серверів та мобільних пристроїв. Ще одна суттєва перевага TensorFlow полягає у його абстракції, яка дозволяє програмістам сконцентруватися на суті задачі, замість детального вивчення реалізації алгоритмів. Це значно прискорює процес створення моделей машинного навчання.

Scikit-learn – це надзвичайно потужна, універсальна бібліотека для машинного навчання, розроблена на основі мови програмування Python. Вона включає в себе великий набір алгоритмів кластеризації, регресії та класифікації, надаючи ефективну реалізацію сучасних технік, що широко використовуються в наукових розробках та різних сферах практичного застосування [14]. Python, завдяки своїй інтерактивності та модульності, дозволяє Scikit-learn сприяти швидкому створенню прототипів моделей машинного навчання.

Усі об'єкти та алгоритми Scikit-learn функціонують на основі двовимірних масивів для вхідних даних. Для об'єктів, призначених для навчання, визначено єдиний набір методів, який залежить від функцій: оцінювачі використовують дані для побудови моделей, прогнозувальники застосовуються для передбачення нових значень, а трансформери призначені для перетворення даних між різними форматами [14].

Scikit-learn ідеально поєднується з іншими науково-орієнтованими бібліотеками Python, зокрема, з NumPy та SciPy, що дозволяє створювати продуктивні та масштабовані

моделі. Бібліотека підтримує як контрольоване, так і неконтрольоване навчання, 5 що робить її універсальним інструментом для розв'язання різних задач машинного навчання.

Основні переваги Scikit-learn:

• Алгоритми для навчання з учителем та без вчителя.

• Регресійні алгоритми, включаючи лінійну та логістичну регресію.

• Класифікаційні алгоритми, включаючи алгоритм K-найближчих сусідів.

• Кластеризація, у тому числі алгоритм K-середніх.

• Інструменти вибору моделі.

• Методи попередньої обробки даних, включаючи нормалізацію Min-Max.

Ці можливості роблять Scikit-learn 5 потужним інструментом для аналізу даних та розробки моделей машинного навчання.

Keras — провідна відкрита бібліотека Python, призначена для розробки нейромереж 5 та машинного навчання. Вона функціонує як високорівневий API, який взаємодіє з різними низькорівневими платформами, зокрема Deeplearning4j, MXNet, Microsoft Cognitive Toolkit (CNTK), Theano чи TensorFlow [15]. Це надає Keras значної гнучкості при створенні та навчанні моделей глибокого навчання.

Основною перевагою Keras є легкість освоєння. Бібліотека надає великий вибір незалежних компонентів: оптимізаторів, шарів нейронів, активаційних функцій, схем ініціалізації, функцій втрат та регуляризації [15]. Така структура дозволяє користувачам просто будувати та змінювати моделі, додаючи нові функції та класи, враховуючи потреби.

Ще однією важливою рисою Keras є відсутність потреби в окремих файлах конфігурації моделі. Модель визначається безпосередньо в коді, що значно спрощує процес розробки та інтеграції з іншим програмним забезпеченням. Всі ці якості роблять Keras зручним інструментом для дослідження та створення прототипів нейромереж, особливо для новачків у сфері глибокого навчання. На рисунку 2.7 наведено застосування бібліотеки Keras.

Рисунок 2.7 - Застосування бібліотеки Keras

Keras значно полегшує 3 створення та навчання штучних нейронних мереж, пропонуючи зрозумілий інтерфейс для конструювання комплексних моделей. Ця

бібліотека відмінно працює зі згортковими нейронними мережами та включає великий вибір алгоритмів для нормалізації, оптимізації та активації шарів. Важливо відзначити, що Keras не є окремою бібліотекою машинного навчання, а являє собою розширюваний інтерфейс, що збільшує модульність та гнучкість, що дозволяє легко адаптувати її під індивідуальні вимоги.

3. РОЗРОБКА ШТУЧНОЇ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ АНАЛІЗУ ТЕКСТОВИХ ДАНИХ

Для створення програми тональної класифікації текстових даних було використано мову програмування Python та операційну систему Manjaro Linux 19.03. Було залучено такі бібліотеки:

• TensorFlow для створення та навчання нейронних мереж.

• Keras, як зручний API для TensorFlow.

• Scikit-learn для класифікації та кластеризації.

• Бібліотеки SpaCy та TextBlob для обробки текстових даних.

• Додаткові бібліотеки Pandas та Numpy.

Алгоритм розробки штучної нейронної мережі (ШНМ) включає три ключові етапи:

Підготовка даних для навчання та тестування: Необхідно ретельно відібрати навчальні дані та підготувати їх у відповідному форматі. З використанням регулярних виразів та бібліотек обробки природної мови (NLP), дані очищаються від зайвих елементів та перетворюються у векторний формат.

Розробка нейронної мережі: В процесі розробки було розглянуто три архітектурні рішення: мережа прямого поширення, згорткова нейронна мережа та рекурентна мережа. Проектування кожної мережі здійснювалось з використанням TensorFlow та Keras, що забезпечує доступ до всіх необхідних функцій для оптимізації, розрахунку втрат та активації.

Навчання та тестування нейронної мережі: Створена мережа проходить навчання протягом шести епох. Після завершення навчання проводиться тестування на підготовлених даних для оцінки точності. Результати навчання представлено у вигляді графіків, що дозволяє порівняти ефективність різних типів ШНМ.

3.1. Вибір та підготовка набору даних

Наріжним каменем машинного навчання є практика на найрізноманітніших задачах, що коливаються від обробки зображень до розпізнавання та опрацювання природної

мови. Кожна з них має свої унікальні підходи та тонкощі, вирішення яких потребує високоякісних даних для навчання ШІ-моделей. Значна частина наукових досліджень використовує власні набори даних, які зазвичай не публікуються у вільному доступі для широкого загалу.

Набори даних класифікуються на три основні групи: обробка зображень, опрацювання природних мов і аналіз аудіо. Існує безліч способів застосування цих наборів даних. Вони можуть бути використані для експериментування з різними методами машинного навчання, вдосконалення власних навичок, а також для розуміння, як правильно формулювати та структурувати кожну окрему проблему.

Для тренування штучних нейронних мереж у цьому дослідженні буде застосовано набір даних "IMDB Dataset" з платформи Kaggle (спільнота дослідників та спеціалістів з машинного навчання). Представлення цього набору даних наведено на рисунку 3.1.:

Рисунок 3.1 - Набір даних IMDB Dataset

Набір даних представлено у форматі CSV, який містить 50 000 рядків. Кожен рядок складається з двох стовпців: "Огляд фільму" та "Тональність". У колонці "Огляд" розташовано текст рецензії, написаної користувачем, а в колонці "Тональність" зазначено настрій рецензії. Можливі значення тональності обмежуються двома варіантами: "позитивний" або "негативний". Це дозволяє розглядати задачу як бінарну класифікацію. Для прикладу, на рисунку 3.2. наведено вигляд перших 30 записів:

Рисунок 3.2 - Вигляд 30 перших рядків набору даних

Аби підготувати вхідні дані для роботи нейронної мережі, потрібен певний попередній етап обробки тексту. Ключові кроки включають наступне:

• Видалення HTML-тегів: Оскільки HTML-теги не несуть в собі корисної інформації для аналізу тональності тексту, їх потрібно прибрати. Для цього можна використати регулярні вирази або бібліотеки, такі як BeautifulSoup чи re.

• Видалення знаків пунктуації та чисел: Пунктуація та цифри не мають значущості при визначенні тональності, тому їх теж необхідно видалити. Це можна зробити за допомогою регулярних виразів, що виявлять всі пунктуаційні знаки та цифри, а потім вилучать їх з тексту.

• Токенізація: Розбиття тексту на окремі слова, або токени. Це дозволяє перетворити текст у структуру, придатну для подальшої обробки нейронною мережею.

• Лемматизація або стемінг: Приведення слів до їхньої базової форми, щоб зменшити варіативність (наприклад, "біжить" перетвориться на "бігти"). Лемматизація часто дає

кращі результати в задачах 3 обробки природної мови.

3 • Перетворення тексту у векторний вигляд: Для тренування нейронних мереж текст необхідно перевести у векторну форму. Для цього можна використовувати такі методи, як Bag-of-Words, TF-IDF або 3 векторні представлення слів (наприклад, Word2Vec, GloVe чи FastText).

• Нормалізація тексту: Переведення всіх слів до нижнього регістру, щоб уникнути проблем із різними варіантами написання (наприклад, "Фільм" і "фільм" мають сприйматися як одне й те саме).

Залишаючи лише корисну інформацію та мінімізуючи шум у даних, ви покращуєте якість навчання моделі та збільшуєте її точність на тестових даних (рис 3.3).

Рисунок 3.3 - Приклад негативної рецензії з індексом №2

Один з найдієвіших методів очищення тексту від непотрібних знаків — застосування регулярних виразів (regex). Регулярні вирази - це зразки, котрі дають змогу відшукувати та замінювати конкретні комбінації символів. Їх підтримує безліч мов програмування, зокрема Python. Алгоритм для очищення відгуків за допомогою регулярних виразів 3 виглядає наступним чином:

3 • Вилучення всіх символів, окрім літер англійського алфавіту (як великих, так і малих). Для цього застосовується вираз: “ [^a-zA-Z] ”.

• Видалення поодиноких літер, оскільки вони не містять корисних відомостей. Для цього використовується вираз: “ \s+[a-zA-Z]\s+ ”.

• Усунення зайвих пробілів і відступів, котрі йдуть підряд, за допомогою виразу: “ \s+ ”.

• Очищення тексту від HTML-тегів, для чого використовується вираз: “ <[^>]+> ”.

Після очищення даних, рецензія з індексом №2 буде мати вигляд (рис 3.4.):

Рисунок 3.4 - Приклад очищеної рецензії №2

Отже, після обробки вхідних даних, непотрібні символи було вилучено з усіх рецензій. Далі весь текст за допомогою бібліотек NLP буде перетворено у векторний простір. На завершення, стовпець "Тональність" з текстових значень перетвориться на числові: позитивні рецензії отримають значення 1, а негативні – 0. Графік тональності відгуків про фільми матиме такий вигляд (рис 3.5):

Рисунок 3.5 - Графік тональності відгуків набору даних IMDB Dataset

3.2. Побудова програми на основі MLP

У цій конкретній реалізації програми застосовується класична архітектура нейронної мережі – багатошаровий перцептрон (його схема наведена на рисунку 3.6). Процес оптимізації навчання здійснюється 3 за допомогою алгоритму Adam, а як міра помилки (функція втрат) використовується binary_crossentropy. В якості функції активації на вихідному шарі застосовується сигмоїдна функція.

Рисунок 3.6 - Архітектура MLP

Варто підкреслити, що процес тренування моделі відбувається винятково на тренувальному наборі даних. Він охоплює 80% від сукупного обсягу наявних даних. Перевірка здатності мережі до узагальнення (або точність) реалізується на інших 20% даних, що не залучалися до процесу тренування. Усі графічні представлення результатів базуються на функціоналі бібліотеки TensorFlow. Після завершення фази навчання та перевірки точності, навчена нейронна мережа продемонструвала такі показники (рисунку 3.7 та рисунку 3.8):

Рисунок 3.7 - Точність MLP

Рисунок 3.8 - Похибка MLP

Отже, у процесі тренування на навчальному наборі нейронна мережа досягла рівня точності 82%. В той же час, оцінюючи її роботу на даних, які раніше не бачила (тестування), модель показала результат 74% правильно розпізнаних відповідей.

3.3. Побудова програми на основі CNN

Згорткові нейронні мережі (CNN) - це особливий вид нейронних мереж, який найчастіше використовується для класифікації двовимірних даних, зокрема зображень. Їхній принцип дії базується на виявленні ключових характеристик безпосередньо на вхідному зображенні вже на першому шарі. Потім, на наступних шарах, ці первинно виявлені ознаки об'єднуються, формуючи складніші групи характеристик, що дозволяє мережі поступово обробляти всю інформацію, яка представлена на зображенні. Варто зазначити, що дослідження продемонстрували 8 ефективність використання згорткових нейронних мереж також і для аналізу текстових даних.

У цій конкретній реалізації 8 згорткової 17 нейронної мережі функція активації ReLU використовується для першого шару, а сигмоїдна функція - для другого. Навчання оптимізується 3 за допомогою алгоритму Adam, а функція втрат - binary_crossentropy. Структурна схема використаної 8 згорткової нейронної мережі представлена нижче (рисунок 3.9):

Рисунок 3.9 - Архітектура згорткової мережі.

Проаналізувавши здобуті дані, маємо підстави стверджувати, що 8 згорткова нейронна мережа (CNN) виявляє вищу точність на обох етапах – як під час навчання, так і при випробуванні на незнайомих даних, якщо порівнювати з звичайною багатошаровою нейронною мережею. Зокрема, точність навчання для CNN наближається до 92%, що є суттєво кращим результатом, аніж у простої нейронної мережі. Схожу картину спостерігаємо і при тестуванні: CNN демонструє точність біля 84%, що теж помітно переважає 74%, отримані для простої нейронної мережі.

Водночас, незважаючи на кращі абсолютні результати, потрібно враховувати, що модель CNN все ще має певний рівень похибок, про що свідчить помітна різниця між точністю, продемонстрованою на тренувальних даних, та точністю на тестовому наборі.

Графічне представлення динаміки точності навчання та тестування міститься на рисунках 3.10 та 3.11.

Рисунок 3.10 - Точність CNN.

Рисунок 3.11 - Похибка CNN.

3.4. Побудова програми на основі RNN

Рекурентні нейронні мережі (RNN) — це тип нейромереж, який показує високу ефективність при 7 роботі з послідовними даними. Оскільки текст, по суті, є впорядкованою низкою слів, використання рекурентних нейромереж є логічним методом для вирішення широкого спектру завдань, що стосуються аналізу тексту.

У наведеній рекурентній нейронній мережі передбачено 128 вхідних нейронів та 128 нейронів у прихованому шарі. Для покращення процесу навчання застосовується алгоритм Adam, функцією втрат обрано binary_crossentropy, а функція активації представлена сигмоїдною функцією. Схематичне зображення архітектури цієї 7 рекурентної нейронної мережі подано на рисунку 3.12 (де: - вхідні вузли, A - приховані вузли, – вихідні вузли, t =128.

:

Рисунок 3.12 - Архітектура 7 рекурентної нейронної мережі

7 Згідно з підсумками тестового етапу (), точність на тестовому масиві сягає приблизно 85%. Це практично відповідає показникам, отриманим під час фази тренування. Важливо зауважити, що цей показник тестової точності є кращим, ніж у згорткових 3 нейронних мереж, та істотно перевершує результати звичайних багатошарових

нейронних мереж. До того ж, незначна різниця між точністю на тренувальній та тестовій вибірках 3 вказує на відсутність перенавчання моделі, що свідчить про правильний підбір вагових коефіцієнтів. Візуалізацію точності та похибки RNN наведено на рисунках 3.13 і 3.14.

Рисунок 3.13. Точність RNN

Рисунок 3.14 - Похибка RNN

Посилання

Це джерела виділених збігів у вашому документі. Кожен збіг позначено темно-зеленим числом, яке відповідає вказаному тут джерелу. Джерела впорядковані за схожістю — чим вищий бал, тим сильніше збіг.

#	Джерело	%
1	194.44.152.155	10.7%
2	lib.pnu.edu.ua	3.9%
3	dspace.ksaeu.kherson.ua	0.9%
4	library.econom.zp.ua	0.3%
5	lemon.school	0.2%
6	moodle.znu.edu.ua	0.2%
7	ela.kpi.ua	0.2%
8	researchgate.net	0.2%
9	ela.kpi.ua	0.2%
10	duikt.edu.ua	0.2%
11	api.man.gov.ua	0.2%
12	lib.pu.if.ua	0.1%
13	researchgate.net	0.1%
14	researchgate.net	0.1%
15	dut.edu.ua	0.1%
16	eoss-conf.com	0.1%
17	openarchive.nure.ua	0.1%

Plag

Дякуємо, що перевірили
свій документ за допомогою
Plag!