Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розробка і дослідження застосування нейронних мереж при обробці і аналізі медичних даних
(тема)

Виконав:
студент 2 курсу, групи СШІм-19-2
Скорик І. С.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту
(повна назва спеціалізації)

Керівник проф. Філатов В.О
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ В.О. Філатов
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)

Кафедра Штучного інтелекту
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системи штучного інтелекту (СШІ)
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Скорику Івану Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка і дослідження застосування нейронних мереж при обробці і аналізі медичних даних

затверджена наказом університету від 29 березня 20 21 р. № 390

2. Термін подання студентом роботи до екзаменаційної комісії _____ _____ 20 ___ р.

3. Вихідні дані до роботи Функція: Огляд сучасного стану використання нейронних мереж. Розробка прототипу нейронної мережі для обробки медичних даних. Перелік використовуваних програмних засобів: ОС Windows 10, середовище розробки Jupyter Notebook, середовище розробки Google Colab. Технічне забезпечення: ноутбук зі встановленим програмним середовищем Jupyter Notebook

4. Перелік питань, що потрібно опрацювати в роботі Аналіз предметної області; Постановка задач досліджень; Характеристика предметної області; Проблематика предметної області; Аналіз методів побудови рекомендацій, заснованих на знаннях; Розробка прототипу нейронної мережі; Огляд використаного середовища та набору даних; Передобробка даних; Реалізація методів класифікації; Демонстрація роботи системи

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Рисунок 1.1 – Нейрон мозку людини; Рисунок 1.2 – Штучна НМ; Рисунок 1.3 – Проста згорткова НМ; Рисунок 1.4 – Система «Цельс»; Рисунок 1.5 – Блок-схема NLP; Рисунок 2.1 – Модель нейрона; Рисунок 2.2 – Функція активації Хевісайда; Рисунок 2.3 – Лінійна функція активації; Рисунок 2.4 – Сигмоїдальна функція; Рисунок 2.5 – Функція гіперболічного тангенсу; Рисунок 2.6 – Функція ReLU; Рисунок 2.7 – Функція LReLU; Рисунок 2.8 – Проста. штучна НМ; Рисунок 2.9 – НМ із прямим поширенням сигналу; Рисунок 2.10 – Рекурентна НМ; Рисунок 2.11 – Багатошарова НМ; Рисунок 3.1 – Демонстрація швидкості обробки запитів при підрахуванні записів; Рисунок 3.2 – Демонстрація швидкості обробки запитів при перегляді усіх записів; Рисунок 3.3 – Демонстрація швидкості обробки запитів при вставці; Рисунок 3.4 – Логічна БД; Рисунок 3.5 – Нормалізація для числових ознак; Рисунок 3.6 – Нормалізація для строк; Рисунок 3.7 – Нормалізація для числа; Рисунок 3.8 – Головне вікно; Рисунок 3.9 – Ввід даних; Рисунок 3.10 – Вивід результату; Рисунок 3.11 – Попередні результати; Рисунок 3.12 – Вивід бібліотеки Keras; Рисунок 3.13 – Графік зміни помилки; Рисунок 3.14 – Графік зміни точності;

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на атестаційне проектування	29.03.2021	виконано
2	Аналіз завдання, пошук літератури та аналогів з теми	30.03.2021	виконано
3	Опрацювання літератури та аналіз об'єкту	31.03.2021	виконано
4	Вибір оптимального підходу до розробки	01.04.2021	виконано
5	Проектування програмного засобу	01.04.2021 –15.04.2021	виконано
6	Вибір технічних засобів для розробки ПЗ	16.04.2021	виконано
7	Розробка програмного засобу	16.04.2021 – 20.04.2021	виконано
8	Аналіз результатів отриманих за допомогою ПЗ	21.04.2021	виконано
9	Оформлення пояснювальної записки та документації	21.04.2021 –22.04.2021	виконано
10	Оформлення графічної та презентаційної частини	22.04.2021	виконано
11	Попередній захист	14.05.2021	виконано
12	Захист перед ЕК		

Дата видачі завдання 29 березня 20 21 р.

Студент _____
(підпис)

Керівник роботи _____ _____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка: 93 с., 4 табл., 30 рис. 2 дод., 37 джерел.

АНАЛІЗ ДАНИХ, МЕДИЧНІ ДАНІ, МОБІЛЬНИЙ ДОДАТОК, НЕЙРОННІ МЕРЕЖІ, ОБРОБКА ДАНИХ, IOS

Об’єкт дослідження – нейронна мережа для обробки та аналізу медичних даних.

Предмет дослідження – процес обробки та аналізу медичних даних за допомогою нейронних мереж.

Мета роботи – обробка та аналіз медичних даних за допомогою нейронної мережі, шляхом розробки мобільного додатку.

Методи розробки – методи створення мобільних додатків під платформу iOS, використовуючи середовище розробки XCode, мову програмування Swift та технології Realm, Storyboard і Alamofire, методи розробки та тренування нейронних мереж за допомогою мови програмування Python, середовища розробки PyCharm.

Результати кваліфікаційної роботи – мобільний додаток, розрахований під платформу iOS, обробки та аналізу медичних даних.

Область застосування – програмний продукт використовується для обробки медичних даних та подальшого їх аналізу для надання попереднього діагнозу, щодо можливих захворювань.

РЕФЕРАТ

Пояснительная записка: 93 с., 4 табл., 30 рис. 2 прил., 37 источников.

АНАЛИЗ ДАННЫХ, МЕДИЦИНСКИЕ ДАННЫЕ, МОБИЛЬНОЕ ПРИЛОЖЕНИЕ, НЕЙРОННЫЕ СЕТИ, ОБРАБОТКА ДАННЫХ, IOS

Объект исследования – нейронная сеть для обработки и анализа медицинских данных.

Предмет исследования – процесс обработки и анализа медицинских данных с помощью нейронных сетей.

Цель работы – обработка и анализ медицинских данных с помощью нейронной сети, путем разработки мобильного приложения.

Методы разработки – методы создания мобильных приложений под платформу iOS, используя среду разработки XCode, язык программирования Swift и технологии Realm, Storyboard и Alamofire, методы разработки и тренировки нейронных сетей с помощью языка программирования Python, среды разработки PyCharm.

Результаты работы – мобильное приложение, рассчитанный под платформу iOS, обработки и анализа медицинских данных.

Область применения – программный продукт используется для обработки медицинских данных и дальнейшего их анализа для предоставления предварительного диагноза, о возможных заболеваниях.

ABSTRACT

Explanatory note: 93 p, 4 tables, 30 fig., 2 ann., 37 sources.

DATA ANALYSIS, DATA PROCESSING, IOS, MEDICAL DATA, MOBILE APPLICATION, NEURAL NETWORKS

The object of the research – a neural network for processing and analyzing medical data.

The subject of the research – the process of processing and analyzing medical data using neural networks.

The purpose of the work – to process and analyze medical data using a neural network by developing a mobile application.

Development methods – methods for creating mobile applications for the iOS platform using the XCode development environment, the Swift programming language and Realm technologies, Storyboard and Alamofire, methods for developing and training neural networks using the Python programming language, the PyCharm development environment.

The results of the work – a mobile application designed for the iOS platform, processing and analysis of medical data.

Scope – the software product is used for processing medical data and their further analysis to provide a preliminary diagnosis of possible diseases.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Гб – гігабайт;

НМ – нейронна мережа;

СУБД – система управління баз данних;

ШІ – штучний інтелект;

ШНМ – штучна нейронна мережа;

API – application programming interface – програмний інтерфейс додатка;

GPU – Graphics Processing Unit – графічний процесор;

MSE – Mean Squared Error – середньоквадратична помилка;

SQL – Structured Query Language – мова структурованих запитів;

UI – user interaface – інтерфейс користувача;

URL – uniform resource locator – універсальний локатор ресурса;

ВСТУП

З кожним роком розвиток інформаційних технологій в усіх галузях діяльності людства викликає необхідність пошуку та розробки нових способів використання Інтернету та його можливостей. Цей стійкий та послідовний рух до побудови інформаційного суспільства і є наслідком інтенсивного розвитку комп'ютерних і телекомунікаційних технологій.

Технології машинного навчання все активніше проникають в повсякденне життя, і ми навіть не замислюємося про те, що нашу стрічку в Instagram та інших соціальних мережах сформував саме штучний інтелект. Звичайно, у нього є і серйозніші завдання – наприклад, прогноз попиту на товари, розпізнавання осіб, відбитків або голосу. Однак важко уявити більш важливу і перспективну сферу застосування ШІ ніж медицина. Від нейромереж чекають серйозних досягнень в цій області – це і діагностика різних захворювань, і розробка нових ліків, і нові методи лікування.

Нейронні мережі являють собою нелінійні системи, що дозволяють набагато краще класифікувати дані, ніж зазвичай використовуються лінійні методи. У додатку до медичній діагностиці вони дають можливість значно підвищити специфічність методу, не знижуючи його чутливості.

Технології машинного навчання можуть застосовуватися при роботі з різними видами інформації. Найбільш широке поширення нейромережі в медицині отримали саме в області роботи з зображеннями. Робочі процеси медичних установ нерозривно пов'язані зі збором, обробкою і аналізом різних медичних зображень: рентген, КТ, цифрові гістологічні дослідження і так далі.

Зважаючи на вищесказане, темою даної науково-дослідної практики обрано дослідження використання нейронних мереж у сфері обробки та аналізу медичних даних.

Для досягнення поставленої мети необхідно розв’язати наступні задачі:

– провести аналіз предметної області;

– зробити опис предметної області;

– провести огляд і аналіз існуючих аналогів, що реалізують функції предметної області;

– розробити специфікацію вимог до предметної області;

– реалізувати проектні та технічні рішення;

– розробити логічну постановку задачі;

– розробити програмний засіб;

– провести опис необхідних інструментів для розгортання програмного продукту.

Платформою для розробки мобільного додатку було обрано операційну систему iOS, оскільки програмний продукт оперує конфіденційною інформацією і повинен буду захищеним від стороннього втручання. Нейронна мережа буде розроблена за допомогою мови програмування Python та бібліотек Keras, Pandas, TensorFlow, NumPy.

В результаті кваліфікаційної роботи буде розроблено компоненти мобільного додатку, який дозволить асистувати лікареві прогнозувати появлення захворювання.

1 АНАЛІЗ СУЧАСНОГО СТАНУ ВИКОРИСТАННЯ НЕЙРОННИХ МЕРЕЖ

1.1 Загальна характеристика нейронних мереж та основних задач, які вирішуються за їх допомогою.

Галузь машинного навчання, нейронні мережі (НМ), також відомі як штучні нейронні мережі (ШНМ), є обчислювальними моделями – по суті алгоритмами. Нейронні мережі мають унікальну здатність витягувати сенс із неточних або складних даних, щоб знаходити закономірності та виявляти тенденції, занадто заплутані для людського мозку або для інших комп'ютерних технік..

Нейронні мережі з'явилися на початку 1940-х років, коли математики Уоррен Маккаллох і Уолтер Піттс створили просту систему, засновану на алгоритмах, призначену для імітації функції людського мозку. Робота в цій області прискорилася в 1957 році, коли Френк Розенблат з Корнельського університету задумав перцептрон, революційний алгоритм, розроблений для виконання складних завдань розпізнавання [1]. У наступні чотири десятиліття нестача обчислювальної потужності, необхідної для обробки великих обсягів даних, гальмувала прогрес. У 2000-х роках, завдяки появі більшої обчислювальної потужності і більш складного обладнання, а також існування величезних наборів даних, з яких можна було витягувати дані, комп'ютерні вчені, нарешті, отримали те, що їм було потрібно. Щоб зрозуміти, наскільки ця область розширилася в новому тисячолітті, візьміть до уваги, що дев'яносто відсотків інтернет-даних було створено з 2016 року. Ці темпи будуть продовжувати рости завдяки розвитку Інтернету речей (IoT).

Подібні людині атрибути нейронних мереж та здатність виконувати завдання в нескінченних перестановках та комбінаціях роблять їх унікальними для сучасних великих даних. Оскільки нейронні мережі також мають унікальну здатність (відому як нечітка логіка) розуміти неоднозначні,

суперечливі або неповні дані, вони можуть використовувати контрольовані процеси, коли відсутні точні моделі.

Згідно з доповіддю, опублікованою Statista [7], у 2017 році загальний обсяг даних сягав близько 100 000 петабайт (тобто, один мільйон гігабайт) на місяць; прогнозується, що вони досягнуть 232 655 петабайт вже у 2021 році. Оскільки підприємства, приватні особи та пристрої генерують величезний обсяг інформації, усі ці великі дані є цінними, і нейронні мережі можуть це зрозуміти.

Володіючи людської здатністю вирішувати проблеми і застосовувати цю навичку до величезних наборів даних, нейронні мережі володіють такими потужними атрибутами:

− адаптивне навчання: як і люди, нейронні мережі моделюють нелінійні і складні відносини і спираються на попередні знання. Наприклад, програмне забезпечення використовує адаптивне навчання для навчання математики та мовним мистецтвам;

− самоорганізація: здатність кластеризувати і класифікувати величезні обсяги даних робить нейронні мережі унікальними для організації складних візуальних проблем, що виникають при аналізі медичних зображень;

− робота в реальному часі: нейронні мережі можуть (іноді) надавати відповіді в реальному часі, як у випадку з безпілотними автомобілями та навігацією за допомогою дронів;

− прогноз: здатність НМ прогнозувати на основі моделей має широкий спектр застосувань, в тому числі для погоди і дорожнього руху;

− відмовостійкість: коли значна частина мережі втрачена або відсутня, нейронні мережі можуть заповнити прогалини. Ця здатність особливо корисна при освоєнні космосу, де завжди можливий вихід з ладу електронних пристроїв.

Нейронні мережі дуже цінні, тому що вони можуть виконувати завдання з аналізу даних, зберігаючи при цьому всі свої інші атрибути. Ось основні завдання, які виконують нейронні мережі:

− класифікація: нейронні мережі організовують шаблони або набори даних в заздалегідь визначені класи;

− прогноз: вони виробляють очікуваний результат на основі заданих вхідних даних;

− кластеризація: вони ідентифікують унікальну особливість даних і класифікують її, не знаючи попередніх даних;

− зв'язування: ви можете навчити нейронні мережі «запам'ятовувати» шаблони. Коли ви показуєте незнайому версію патерну, мережа пов'язує її з найбільш порівнянної версією в своїй пам'яті і повертається до останньої.

Нейронні мережі мають фундаментальне значення для глибинного навчання, надійного набору методів нейронних систем, що дозволяє вирішити абстрактні проблеми, такі як біоінформатика, розробка лікарських засобів, фільтрація соціальних сетей та переклад на естетичний мову. Глибинне навчання – це те місце, де вирішуються найскладніші завдання в науці та техніці, включаючи передову робототехніку. За допомогою того, як нейронні мережі стають розумнішими та швидшими, суспільство щодня вдосконалюється.

Найбільш проривним аспектом нейронних мереж є те, що після тренування вони продовжують навчатися самостійно в процессі їх використання. Таким чином, вони імітують людський мозок, який складається з нейронів, фундаментального будівельного матеріалу для передачі інформації як для людини, так і для нейронної мережі.

Весь мозок ссавців складається з взаємопов’язаних нейронів, які передають електрохімічні сигнали. Нейрони мають кілька компонентів: тіло, що включає ядро і дендрити; аксони, які з’єднуються з іншими клітинами; і аксонні термінали або синапси, які передають інформацію або подразники від одного нейрона до іншого. У поєднанні цей блок виконує функції зв’язку та інтеграції в нервовій системі. Мозок людини має величезну кількість процесорних одиниць (86 мільярдів нейронів), які дозволяють виконувати дуже складні функції.

На рисунку 1.1 продемонстрована структура нейрону мозку людини.

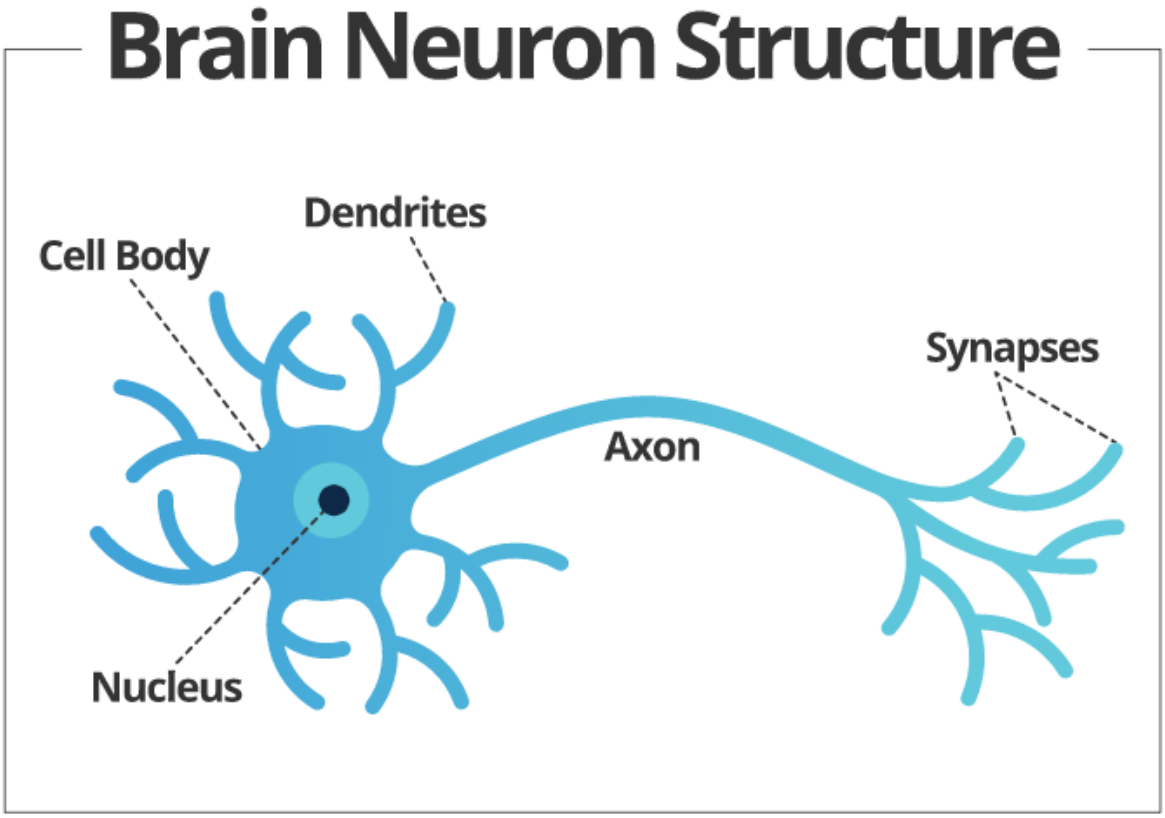

Рисунок 1.1 – Структура нейрону мозку людини

Штучні нейронні мережі (ШНМ) – це статистичні моделі, призначені для адаптації та самопрограмування за допомогою алгоритмів навчання, щоб зрозуміти та розібрати поняття, зображення та фотографії. Щоб процесори виконували свою роботу, розробники розташовують їх по шарах, які працюють паралельно. Вхідний шар є аналогом дендритів у нейронній мережі людського мозку. Прихований шар порівнянний з тілом клітини і знаходиться між вхідним шаром та вихідним шаром (що є подібним до синаптичних виходів у мозку). Прихований шар – це місце, де штучні нейрони приймають набір входів на основі синаптичної ваги, яка є амплітудою або силою зв’язку між вузлами. Ці зважені входи генерують вихід через функцію передачі на вихідний рівень.

На рисунку 1.2 продемонстрована архітектура нейронної мережі.

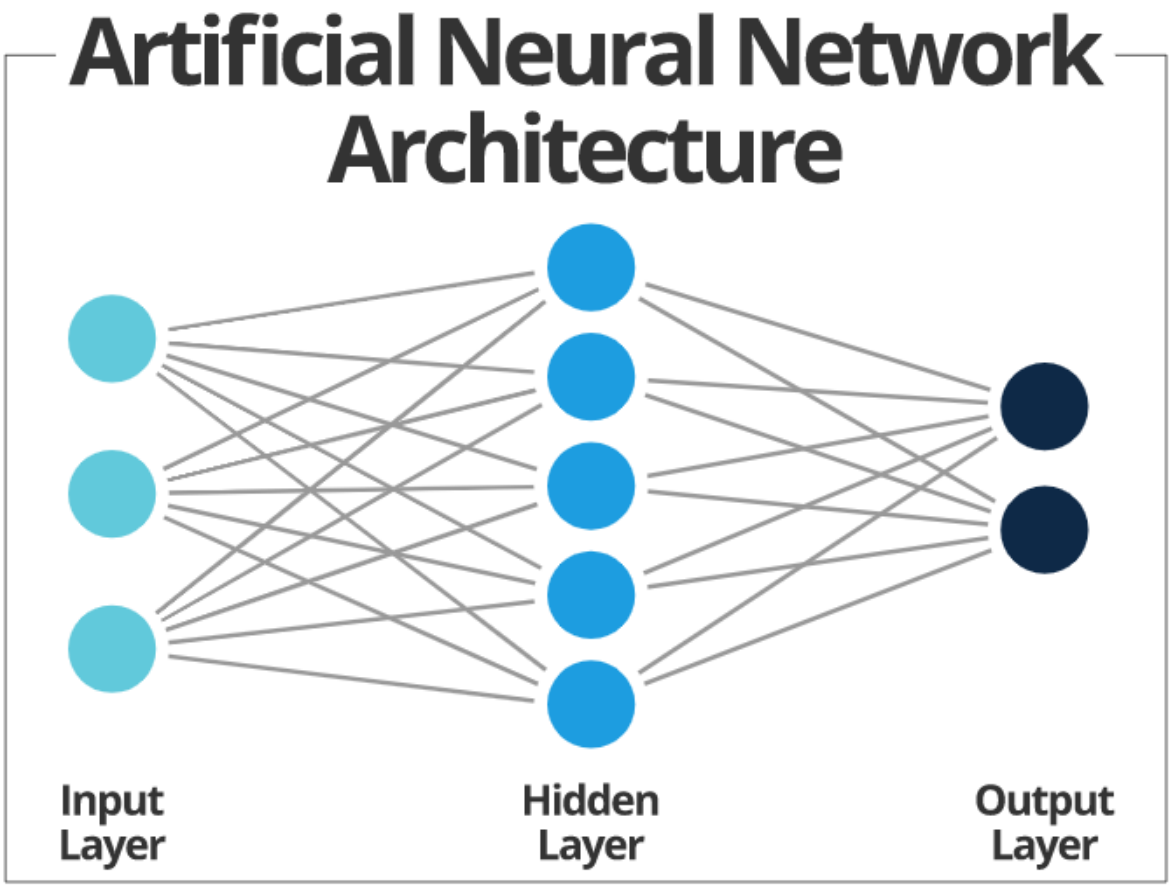

Рисунок 1.2 – Архітектура штучної нейронної мережі.

Після того, як ви структуруєте мережу для конкретного додатка, починається тренування (тобто навчання). Існує два підходи до навчання. Контрольоване навчання або навчання з учителем забезпечує мережу бажаними результатами шляхом ручного оцінювання продуктивності мережі або шляхом надання бажаних результатів та входів. Навчання без нагляду відбувається, коли мережа осмислює вхідні дані без сторонньої допомоги чи інструкцій.

Щодо не контрольованого навчання або навчання без учителя, то Йошуа Бенджо з Університету Монреаля у статті «Підйом нейронних мереж та глибоке навчання у нашому повсякденному житті» зауважує, що отримання інформації з немічених даних зараз є дуже гарячою темою у світі машинного навчання, тому у цій галузі ведуться досить велика кількість досліджень.

Бенджо посилається на той факт, що кількість нейронних мереж не може збігатися з кількістю зв’язків у мозку людини, але здатність нейронних мереж наздоганяти може бути просто надзвичайною. Закон Мура, який стверджує, що загальна обчислювальна потужність комп’ютерів подвоюватиметься кожні два роки, дає нам підказку щодо напрямку, в якому рухаються нейронні мережі та ШІ. Генеральний директор Intel Брайан Крзаніч на виставці комп’ютерної електроніки 2017 року підтвердив, що «Закон Мура живий, здоровий і процвітає». З моменту свого створення в середині 20 століття здатність нейронних мереж «думати» змінювала наш світ неймовірними темпами.

1.2 Характеристика використання нейронних мереж в медицині

Використання нейронних мереж в медицині зазвичай пов'язане з системами діагностики захворювань. Однак нейронні мережі здатні не тільки розпізнавати приклади, але і зберігати дуже важливу інформацію. З цієї причини однією з основних областей застосування нейронних мереж є інтерпретація медичних даних.

Нейронні мережі в медицині можуть використовуватися як лікарями, так і пацієнтами лікарень для аналізу стана здоров’я.

Найбільш широке поширення нейромережі в медицині отримали саме в області роботи з зображеннями. Робочі процеси медичних установ нерозривно пов'язані зі збором, обробкою і аналізом різних медичних зображень: рентген, КТ, цифрові гістологічні дослідження і так далі.

1.2.1 Використання нейронних мереж для обробки та аналізу медичних зображень

Піднапрямок штучного інтелекту, який займається роботою з зображеннями і відеопотоком, отримав назву Computer Vision або

комп'ютерний зір. Цей напрямок є найбільш перспективним в медичній діагностиці та скринінгу патології.

Сервіси з застосуванням технології комп'ютерного зору розробляються по всьому світу і допомагають лікарям виявляти ознаки різних захворювань, в тому числі онкології. Один з таких проектів – «Цельс» [8], розробка російської компанії «Медичні скринінг системи». Даний сервіс використовує технології згорткових нейронних мереж, для роботи із зображеннями.

Згорткові нейронні мережі, також звані ConvNets, були вперше представлені в 1980-х роках Яном ЛеКуном, докторантом-дослідником в області комп'ютерних наук. ЛеКун спирався на роботу, виконану Куніхіко Фукусіма, японським ученим, який кількома роками раніше винайшов неокогнітрон, дуже просту нейронну мережу розпізнавання зображень.

На рисунку 1.3 продемонстрована топологія простої згорткової неронної мережі.

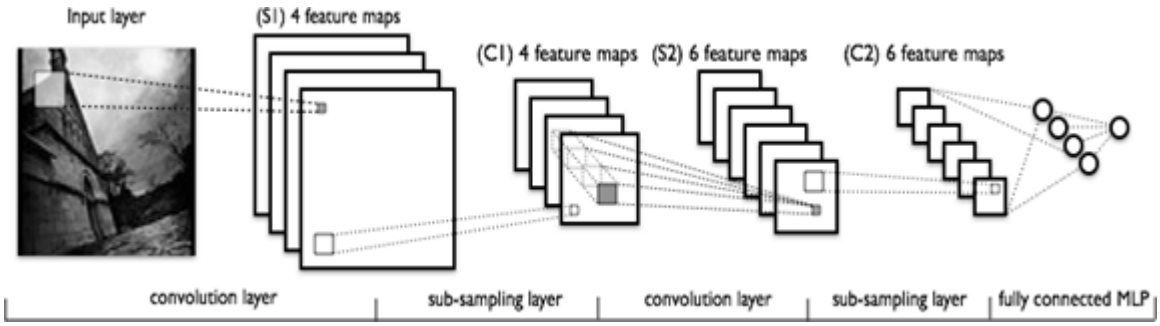

Рисунок 1.3 – Топологія простої згорткової нейронної мережі

Рання версія CNN, названа LeNet (в честь LeCun), могла розпізнавати рукописні цифри. CNN знайшли нішу на ринку банківських, поштових і банківських послуг, де вони зчитують поштові індекси на конвертах і цифри на чеках.

Але, не дивлячись на свою винахідливість, ConvNets залишалися осторонь від комп'ютерного зору і штучного інтелекту, тому що зіткнулися з серйозною проблемою: вони не могли масштабуватися. CNN потрібно

багато даних і обчислювальних ресурсів для ефективної роботи з великими зображеннями. У той час цей метод був застосований лише до зображень з низьким дозволом.

У 2012 році AlexNet показав, що, можливо, прийшов час повернутися до глибокого навчання – галузі штучного інтелекту, що використовує багаторівневі нейронні мережі. Доступність великих наборів даних, а саме набору даних ImageNet з мільйонами помічених зображень, і величезних обчислювальних ресурсів дозволили дослідникам створювати складні CNN, які могли виконувати завдання комп'ютерного зору, які раніше були неможливі.

На даний момент сервіс «Цельс» працює за чотирма напрямами діагностики – мамографія, флюорографія, комп'ютерної томографія легенів і гістологія. Робота лікаря з системою відбувається наступним чином:

– лікар завантажує в систему зображення (по одному або цілим пакетом). Далі система ранжує список досліджень по пріоритетності – від найбільшої ймовірності наявності патології до найменшої. Таким чином лікар в першу чергу перегляне знімки тих пацієнтів, у яких система запідозрила наявність новоутворення. Це дозволить оперативно провести дообстеження, поставити діагноз і почати лікування;

– лікар відкриває конкретне дослідження зі списку і бачить зображення, на якому система маркером виділила саме ті області, на яких імовірно візуалізуються ознаки патології;

– потім лікар переглядає опис знімка, автоматично сформованої системою, і при необхідності вносить до нього свої зауваження.

Таким чином, основні завдання сервісів на основі технологій комп'ютерного зору – полегшення рутинної роботи лікаря, скорочення часу на дослідження і як наслідок більш оперативна допомога пацієнту.

На рисунку 1.4 продемонстрований приклад роботи системи «Цельс».

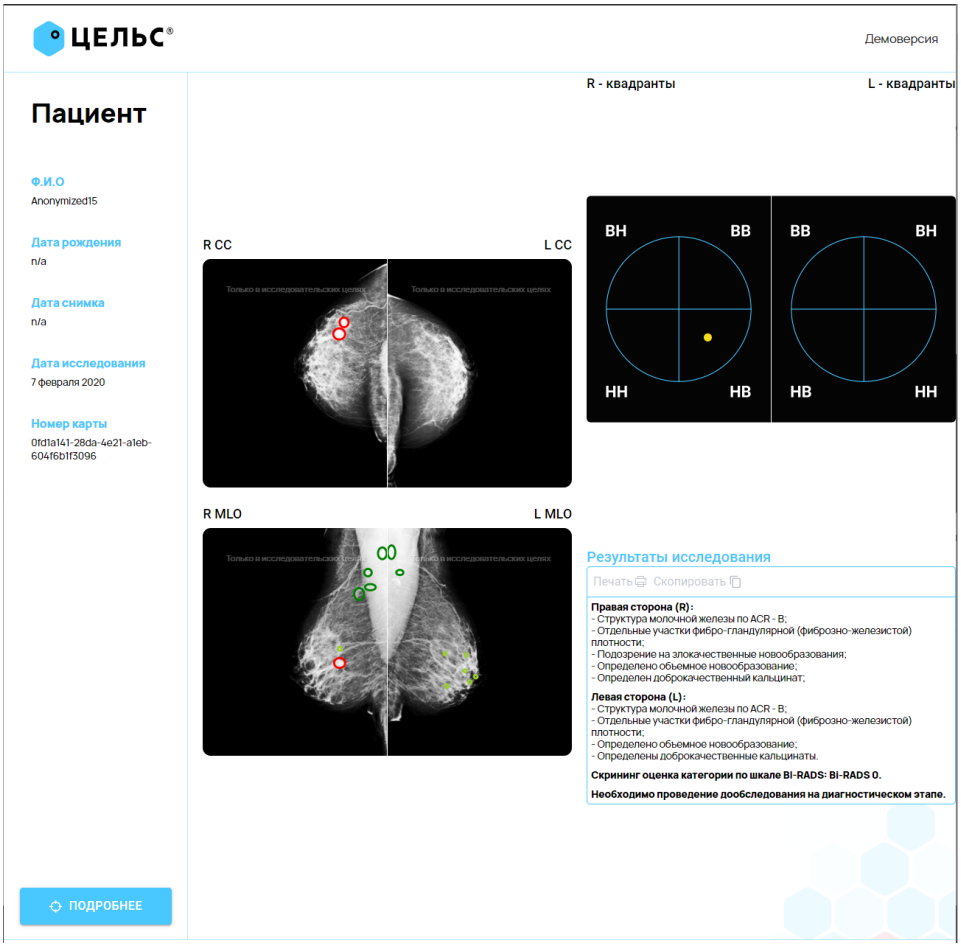

Рисунок 1.4 – Приклад роботи системи «Цельс»

Наведемо ще кілька прикладів того, в яких сферах медицини може застосовуватися машинне навчання, а також розглянемо основні обмеження і складності застосування нейромережевих технологій в реальній клінічній практиці, які заважають розпочати їх масове використання тут і зараз.

1.2.2 Використання нейронних мереж для аналізу ДНК

Аналіз ДНК – ще один перспективний і активно розвивається напрямок застосування нейромереж. Наприклад, інструмент, розроблений Університетом штату Мічиган [9], здійснює генетичні дослідження і дозволяє по геному людини встановити його зростання з точністю до трьох сантиметрів, спрогнозувати розвиток у нього таких серйозних захворювань як рак, інсульт та інфаркт, виявити мутації, що впливають на щільність

кісткової тканини, і навіть передбачити рівень освіти, якого може досягти людина.

1.2.3 Використання нейронних мереж для розробки ліків

Першим ліками, створеним за допомогою штучного інтелекту і вийшли на етап клінічних випробувань, став препарат DSP-1 181. Він розроблений компанією Exscientia [10] спільно з японською фармацевтичною компанією.

DSP-1 181 є агоністом 5-HT1A рецептора серотоніну і призначений для лікування пацієнтів з обсесивно-компульсивним розладом (ОКР). Зазвичай на розробку таких ліків (етап Drug discovery) у дослідників йде близько п'яти років. Штучний інтелект впорався з цим завданням всього за рік.

Перший етап клінічних випробувань препарату був запланований на березень 2020 року. Поки немає інформації про те, чи вплинула на ці плани пандемія COVID-19.

1.2.4 Використання нейронних мереж для обробки голосових команд лікаря

Розпізнавання голосу. Робочі процеси лікаря включають в себе не тільки консультації пацієнтів або проведення досліджень. Значна частина часу йде на заповнення різного роду документації. З цією рутинною роботою лікаря також можуть допомогти нейромережеві технології.

Програма Voice2Med [11] економить час лікаря завдяки голосовому заповнення медичної документації. В сервіс включені спеціалізовані словники, що дозволяє вірно розпізнавати медичні терміни. Даний сервіс для своєї роботи використовує технології обробки природної мови (NLP).

Обробка природної мови (NLP) – це перетин комп'ютерних наук, лінгвістики і машинного навчання. Ця область фокусується на спілкуванні

між комп'ютерами і людьми на природній мові, а NLP – на тому, щоб змусити комп'ютери розуміти і генерувати людську мову. Застосування методів NLP включає голосових помічників, таких як Amazon Alexa і Apple Siri, а також такі речі, як машинний переклад і фільтрація тексту.

На рисунку 1.5 наведена блок-схема процесу обробки речення за допомогою методів NLP.

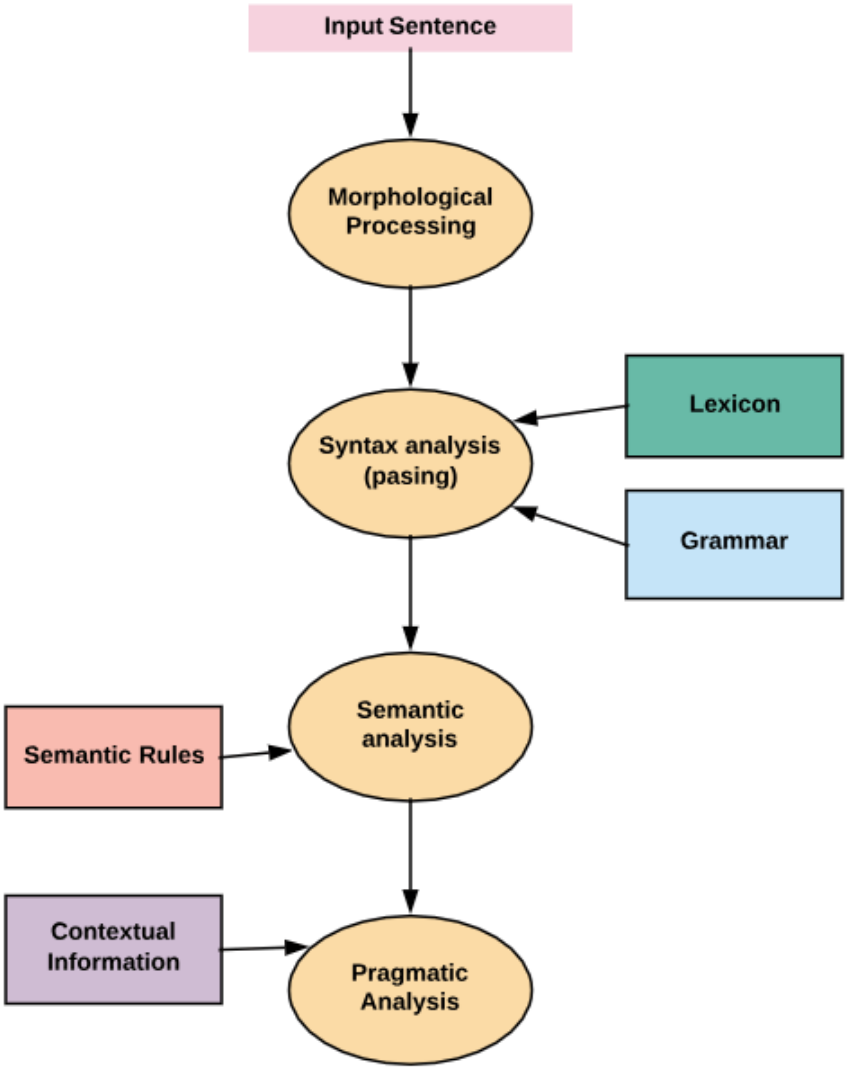

Рисунок 1.5 – Блок-схема процесу обробки речення за допомогою методів NLP

Методи обробки природної мови сильно виграли від останніх досягнень в області машинного навчання, особливо від методів глибокого навчання. Галузі використання поділені на три частини:

– розпізнавання мови – переклад усного мовлення в текст;

– розуміння природної мови – здатність комп'ютера розуміти те, що ми говоримо;

– генерація природної мови – генерація природної мови комп'ютером.

Обробка природної мови в медицині дає можливість комп'ютерам робити те, що їм необхідно. Для проведення аналітики, кодування коригування ризиків виникнення захворювань, функцій бек-офісу і аналізу набору пацієнтів, не заважаючи спілкуванню з лікарем.

NLP в медицині відкриває нові захоплюючі можливості для надання медичної допомоги та поліпшення якості обслуговування пацієнтів. Незабаром спеціалізоване розпізнавання кодування NLP дозволить лікарям проводити більше часу з пацієнтами, допомагаючи робити проникливі висновки на основі точних даних. У найближчі роки ми будемо чути новини і бачити можливості цієї технології, оскільки вона дозволяє постачальникам послуг позитивно впливати на результати щодо здоров'я.

1.2.5 Використання нейронних мереж для діагностики та прогнозування захворювань

У пошуках різних застосувань у різних областях наук, для області медичної діагностики також знайдено застосування штучної нейронної мережі за допомогою біостатистики в клінічних службах. Вона використовується для діагностики раку, склерозу, діабету, сердцевих захворювань та т.д. Розробка та застосування адаптивного алгоритму забезпечують максимальну точність результатів статистики в клінічних випробуваннях.

Серцево-судинні захворювання – це сукупність захворювань, що вражають серце, серцеві м’язи, судини, вени. Національний центр

статистики Америки повідомляє про те, що основна причина смерті в США є сердцево-судинні захворювання. У минулому дані, зібрані з пацієнтами, використовувались для розробки моделей штучної нейронної мережі з алгоритмом зворотнього поширення помилок. Ця модель змогла досягти 91,2% точної діагностики цих захворювань на основі отриманих даних. Також були і інші моделі з точністю менше 90%, що також використовувались для діагностики конкретних типових серцевих захворювань.

Ракові пухлини. У 2012 році в звітах Американського онкологічного товариства говорилося, що було виявлено понад 1,6 мільйона вперше діагностованих випадків. Отже, виникла необхідність в розробці швидкого і відповідного діагнозу для клінічного ведення. Відповідна інформація для діагностики була зібрана за допомогою передових аналітичних методів, таких як мас-спектрометрії, і застосована в клінічній діагностиці раку грудей і яєчників. Штучна нейронна мережа також використовується для діагностики різних типів пухлин головного мозку, раку легенів. В кінцевому рахунку, штучна нейронна мережа була помічена з використанням базових даних, які варіюються від клінічних даних до результатів біохімічних аналізів і забезпечують максимальну діагностичну точність для різних типів раку.

Діагнотика діабету. Діабет став серйозною проблемою для здоров'я як у розвинутих, так і в країнах, що розвиваются. Число випадків діабету у світі оцінюється в 366 мільйонів. Діабет II типу є стандартним типом цього захворювання, яке виникає із-за неправильної клітинної реакції на інсулін, що приводить до гіперглікемії. Інформація про такі параметри, як вік, стать, вага і рівень глюкози, була зібрана та використана у якості вхідних даних для створення штучної нейронної мережі, яка може дати результати з точністю 90%. Штучні нейронні мережі використовуються для оцінювання рівня глюкози, а також для діагностики діабету відповідно до біостатистичних досліджень клінічних випробувань.

1.3 Обмеження і проблеми у використанні нейронних мереж в медицині

Перша перешкода пов'язана не стільки із застосуванням медичної нейромережі, скільки з її розробкою. Для навчання штучного інтелекту необхідна велика кількість даних. У випадку з аналізом медичних зображень потрібні знімки з виконаною на них розміткою на об'єкти.

Існують публічно доступні датасети (набори даних), але використання більшості з них допускається тільки в некомерційних цілях. До того ж, розмітка на них може бути різною – і не завжди підходить під конкретну задачу.

Саме тому розробникам не обійтися без збору власних датасетів для навчання своєї моделі. А це, в свою чергу, вимагає безпосередньої участі лікарів. Втім, їх участь потрібно не тільки в зборі і розмітці даних, але і на інших етапах розробки. Без зворотного зв'язку продукт буде «відірваний» від реальної клінічної практики і не зможе в достатній мірі враховувати специфіку роботи лікарів.

Ще одна перешкода стосується процесу впровадження вже готового продукту в робочі процеси медичних установ: це відсутність в законодавстві конкретних стандартів, що регламентують застосування таких технологій в медицині.

Також досить важлива проблема майбутнього штучного інтелекту в медицині полягає в тому, наскільки добре можуть бути забезпечені конфіденційність і безпеку даних. Існує ризик виявлення конфіденційних даних пацієнта з історії хвороби. Більш того, є ризик навмисного злому алгоритму для нанесення шкоди людям у великих масштабах, наприклад передозування інсуліну у діабетиків.

Інша проблема – неточна робота алгоритмів. Недавній приклад – алгоритм IBM Watson Health (відомий як Watson for Oncology). Використовуваний сотнями лікарень по всьому світу для рекомендацій по лікуванню хворих на рак, алгоритм був заснований на невеликій кількості

синтетичних випадків і дуже обмеженій кількості реальних даних. Багато з його рекомендацій по лікуванню були помилковими, наприклад, пропонували використовувати несумісне ліки для пацієнта з сильною кровотечею, що представляє явне протипоказання.

Ще одна проблема – упередженість. Низький соціально-економічний статус – основний фактор ризику передчасної смертності. Непропорційне використання штучного інтелекту у «заможних», на відміну від «незаможних», може збільшити існуючий розрив у стані здоров'я. З цією проблемою тісно пов'язано зміщення результатів через відсутність включення меншин в набори даних. Приклад – алгоритми в дерматології, які діагностують меланому, але не враховують колір шкіри. Потрібно викорінити забобони і прагнути до медичних досліджень, які забезпечують дійсно репрезентативне уявлення населення.

1.4 Постановка задачі дослідження

В ході роботи планується розробити та натренувати нейронну мережу, яка зможе робити прогноз щодо можливої появи серцевих нападів або діабету у користувача. Також планується помістити розроблену нейронну мережу в оболонку мобільного додатку під платформу iOS.

Для розробки нейронної мережі необхідно буде вирішити такі питання:

– який тип завдань буде вирішуватися нейронною мережею;

– який тип навчання нейронної мережі потрібно застосувати;

– нормалізувати набір даних, що буде використовуватися для навчання та тестування нейронної мережі.

Під час аналізу предметної області було визначено, що мобільний додаток має реалізовувати наступний функціонал:

– зручний спосіб вводу даних про користувача;

– зручний і зрозумілий перегляд результату прогнозу;

– перегляд попередніх результатів роботи системи;

– додаток повинен добре взаємодіяти із розробленою нейронною мережею.

Нейронна мереже буде розроблена за допомогою мови програмування Python та бібліотек Keras, TensorFlow, NumPy та Pandas. Потім нейронну мережу потрібно буде конвертувати у зрозумілий для середи розробки Xcode та бібліотеки CoreML формат, щоб потім її можна було використовувати при розробці мобільного додатку. Цільовою аудиторією можуть бути як лікарі так і звичайні користувачі, які хочуть отримати попередній діагноз щодо можливих захворювань.

2 ТЕОРЕТИЧНІ ДОСЛІДЖЕННЯ

2.1 Штучні нейронні мережі

Штучна нейрона мережа (ШНМ) – це математична модель, що представляє собою систему з'єднаних і взаємодіючих між собою простих процесорів (штучних нейронів). Штучні нейрони використовують біологічну аналогію з нейронами головного мозку людини. Біологічні нейрони влаштовані дуже складно, але спрощено їх основні компоненти можна описати наступним чином. В центрі знаходиться ядро, в якому накопляється електричний заряд. Він поступає від інших нейронів через відростки, які називаються дендритами. Після того, як всередині ядра накопичився певний обсяг заряду, нейрон спрацьовує і видає електричний сигнал на вихідний відросток, який називається аксоном. Аксон в свою чергу прикріпляється до дендритів інших нейронів через так звані синапси, які можуть змінювати сигнал, що ними передається. Якщо сигнал в синапсі збільшується, то такий синапс називається збуджуючим, а якщо зменшується – гальмуючим.

Нейрон – це обчислювальна одиниця, яка отримує інформацію, здійснює над нею прості обчислення і передає далі. За моделлю Мак-Каллока і Піттса у нейрона є кілька входів (аналог біологічних дендритів), на які подаються вхідні сигнали $x_1, x_2 \dots x_n$ і один вихід a, через який видається вихідний сигнал, по аналогії з аксоном у біологічному нейроні. Кожному входу назначається деяка вага $w_1, w_2 \dots w_n$ по аналогії з роботою синапсів головного мозку [13]. Велика додатня вага посилює сигнал, як збуджуючий синапс, а від'ємна послаблює вхідний сигнал, як гальмуючий синапс. Вхідні сигнали нейрона, що подаються на кожний вхід, помножуються на відповідну вагу, потім складаються і подаються на вхід деякій нелінійній функції, що називається функцією активації. Функція активації визначає спрацьовує нейрон чи ні. Загальну схему нейрона зображено на рисунку 2.1.

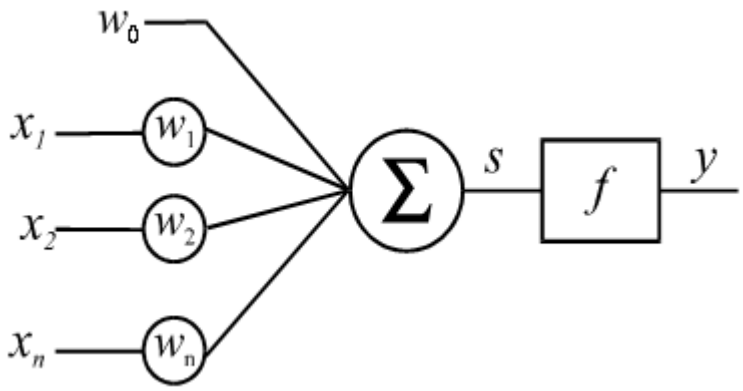

Рисунок 2.1 – Модель нейрона

Вихідний сигнал нейрона обчислюється за формулою 2.1.

$$y = f\left(\sum_{i=1}^{N} w_i x_i\right), \tag{2.1}$$

де N – кількість входів нейрона;

x_i – вхідне значення;

w_i – ваговий коефіцієнт на i-му вході;

f – деяка функція активації.

2.2 Функції активації

Одним з найважливіших аспектів глибинної нейронної мережі є функції активації. Функція активації – f (x) визначає вихідне значення нейрона в залежності від результату зваженої суми входів і порогового значення [13].

Розглянемо нейрон, у якого зважена сума входів:

$$z = \sum_i w_i x_i + b, \tag{2.2}$$

де w_i – вага i-ого входу;

x_i – вхідний сигнал і-ого входу;

b – зсув.

Отриманий результат передається в функцію активації, яка вирішує розглядати цей нейрон як активоване, або його можна ігнорувати.

Одна із найпростіших і найпоширеніших функцій активації являється функція Хевісайда, також відома як ступінчата або порогова функція (рисунок 2.2). Ця функція активації перевіряє вихідний сигнал і якщо він менше заданої величини, то на виході нейрону буде нуль, в іншому випадку – 1 (формула 2.3).

$$\theta(x) = \begin{cases} 0, x < 0 \\ 1, x > 0 \end{cases}, \tag{2.3}$$

де $\theta(x)$ – вихідний сигнал;

x – вхідний сигнал.

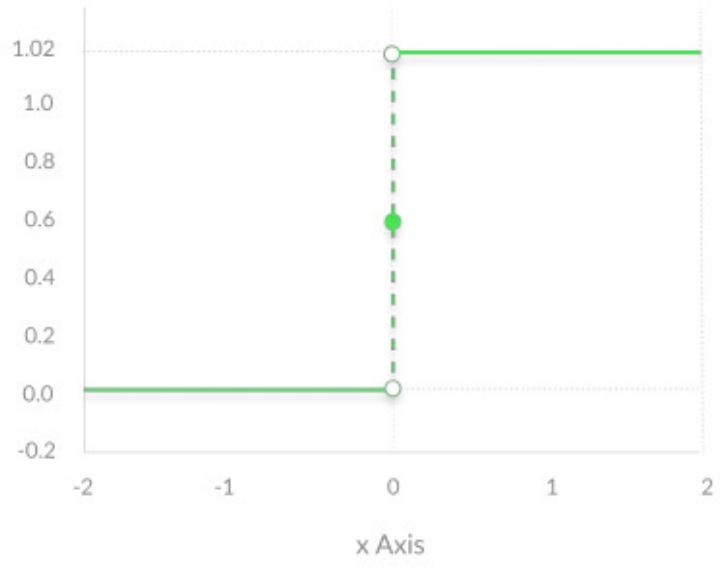

Рисунок 2.2 – Графік функції активації Хевісайда

Така функція відмінно працює для бінарної класифікації. Але вона не працює, коли для класифікації потрібна більша кількість нейронів і кількість можливих класів більше двох.

Лінійна функція являє собою пряму лінію (формула 2.4), а це значить, що результат цієї функції активації пропорційний переданому аргументу. На відміну від попередньої функції, вона дозволяє отримати діапазон значень на виході, а не тільки бінарні 0 і 1, що вирішує проблему класифікації з великою кількістю класів. Графік лінійної функції зображено на рисунку 2.3.

$$a(x) = \sum_i c_i x_i \,, \tag{2.4}$$

де $a(x)$ – вихідний сигнал;

x – вхідний сигнал;

c – деяка константа.

Рисунок 2.3 – Графік лінійної функції активації

Але у лінійної функції є дві основні проблеми:

– неможливість використання методу зворотного поширення помилки. Так як в основі цього методу навчання лежить градієнтний спуск, а для того щоб його знайти, потрібно взяти похідну, яка для даної функції активації – константа і не залежить від вхідних значень. Тобто при оновленні ваг не можна сказати чи покращується емпіричний ризик на поточному кроці чи ні;

– розглянемо нейронну мережу з декількома шарами з цією функцією активації. Так як для кожного шару вихідне значення лінійно, то вони утворюють лінійну комбінацію, результатом якої є лінійна функція. Тобто фінальна функція активації на останньому шарі залежить тільки від вхідних значень на першому шарі. А це означає, що будь-яку кількість шарів може бути замінено всього одним шаром, і, отже, немає сенсу створювати багатошарову мережу.

Головна відмінність лінійної функції від інших в тому, що її область визначення не обмежена: $(-\infty; +\infty)$. Отже, її потрібно використовувати, коли вихідне значення нейрона повинно відповідати $\in R$, а не обмеженому інтервалу.

Сігмоїдальна функція, яку також називають логістичною, є гладкою монотонно зростаючою нелінійної функцією:

$$\sigma(x) = \frac{1}{1 + e^{-x}}, \tag{2.5}$$

де, $\sigma(x)$ – вихідний сигнал;

x – вхідний сигнал.

І так як ця функція нелінійна, то її можна використовувати в нейронних мережах з великою кількістю шарів, а також навчати ці мережі методом зворотного поширення помилки. Сигмоїда обмежена двома горизонтальними асимптотами $y = 1$, $y = 0$, що дає нормалізацію вихідного значення кожного нейрона. Крім того, для сігмоідной функції характерний

гладкий градієнт, який запобігає «стрибкам» при підрахунку вихідного значення. Крім того, у цієї функції є ще одна перевага, для значень $x > 2$ і $x < -2$, y «притискається» до однієї з асимптот, що дозволяє робити чіткі прогнози класів. Графік сигмоїдальної функції зображено на рисунку 2.4.

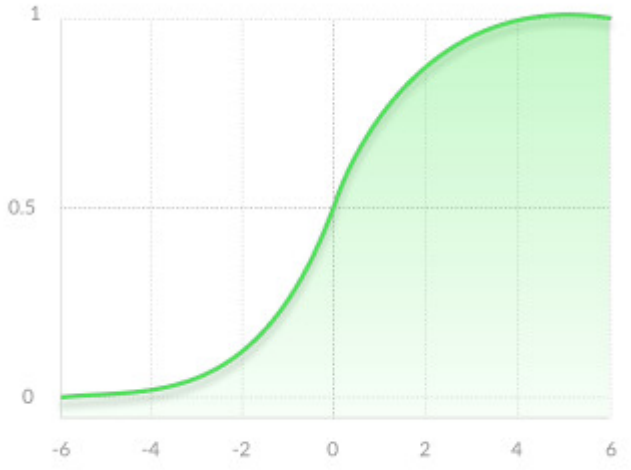

Рисунок 2.4 – Графік сигмоїдальної функції

Незважаючи на безліч сильних сторін сігмоідальної функції, у неї є значний недолік. Похідна такої функції вкрай мала у всіх точках, крім порівняно невеликого проміжку. Це сильно ускладнює процес поліпшення ваг за допомогою градієнтного спуску. Більш того, ця проблема посилюється в разі, якщо модель містить багато шарів. Дана проблема називається проблемою зникаючого градієнта.

Що стосується використання сігмоїдальної функції, то її перевага над іншими – в нормалізації вихідного значення. Іноді, це буває вкрай необхідно. Наприклад, коли підсумкове значення шару повинно представляти ймовірність випадкової величини. Крім того, цю функцію зручно застосовувати при вирішенні задачі класифікації, завдяки властивості «притискання» до асимптот.

Функція гіперболічного тангенса має вигляд:

$$th\,(x) = \frac{2}{1 + e^{-2x}} - 1, \tag{2.6}$$

де $th(x)$ – вихідний сигнал;

x – вхідний сигнал.

Графік функції гіперболічного тангенсу продемонстрований на рисунку 2.5

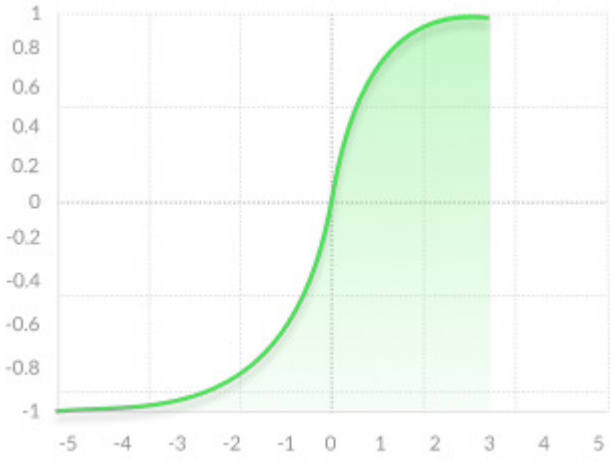

Рисунок 2.5 – Графік функції гіперболічного тангенсу

Ця функція є скоригованою сігмоїдальною функцією

$$th(x) = 2\,\sigma(2x) - 1, \tag{2.7}$$

тобто вона зберігає ті ж переваги і недоліки, але вже для діапазону значень $(-1;\ 1)$.

Зазвичай, функція гіперболічного тангенсу використовується частіше сигмоїдальної у випадках, коли немає необхідності в нормалізації. Це відбувається через те, що область визначення даної функції активації центрована щодо нуля, що знімає обмеження при підрахунку градієнта для переміщення в певному напрямку. Крім того, похідна гіперболічного тангенса значно вище поблизу нуля, даючи велику амплітуду градієнтному спуску, а отже і більш швидку збіжність.

Наразі, функція активації, якою користуються частіше за всі інші для глибинного навчання, є функція ReLU (Rectified Linear Unit). Ця функція повертає 0, якщо приймає негативний аргумент, в іншому випадку функція повертає саме число. Тобто вона може бути записана як

$$f(\mathrm{x}) = max\,(0, \mathrm{x}), \tag{2.8}$$

де $f(\mathrm{x})$ – вихідний сигнал;

x – вхідний сигнал.

Графік функції ReLU відображений на рисунку 2.6.

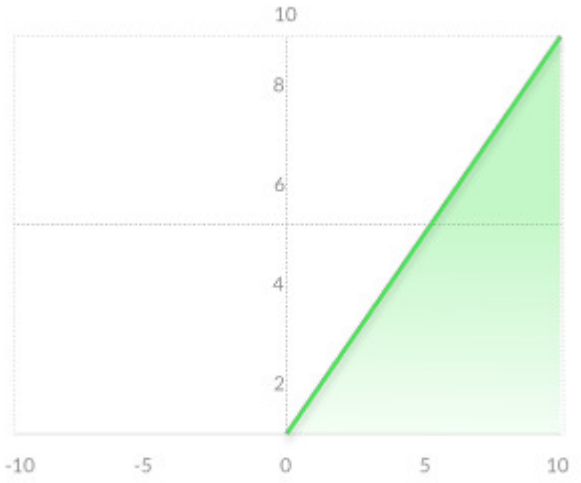

Рисунок 2.6 – Графік функції ReLU

На перший погляд може здатися, що вона лінійна і має ті ж проблеми що і лінійна функція, але це не так і її можна використовувати в нейронних мережах з великою кількістю шарів. Функція ReLU володіє декількома перевагами перед сигмоїдальною функцією і функцією гіперболічного тангенса:

– дуже швидко і просто рахується похідна. Для від'ємних значень – 0, для позитивних – 1;

– розрідженість активації. У мережах з дуже великою кількістю нейронів використання сігмоїдальної функції або гіперболічного тангенса

як активаційний функції тягне активацію багатьох нейронів, що може позначитися на продуктивності навчання моделі. Якщо ж використовувати ReLU, то кількість вмикаємих нейронів стане менше, в силу характеристик функції, і сама мережа стане легше.

У цієї функції є один недолік, який має назву проблемою вмираючого ReLU. Так як частина похідної функції дорівнює нулю, то і градієнт для неї буде нульовим, а то це означає, що ваги не будуть змінюватися під час спуску і нейронна мережа перестане вчитися.

Функцію активації ReLU слід використовувати, якщо немає особливих вимог для вихідного значення нейрона, на кшталт необмеженої області визначення. Але якщо після навчання моделі результати вийшли не оптимальні, то варто перейти до інших функцій, які можуть дати кращий результат.

Однією з проблем стандартного ReLU є затухаючий, а саме нульовий, градієнт при негативних значеннях. При використанні звичайного ReLU деякі нейрони вмирають, а відстежити вмирання нейронів не просто. Щоб вирішити цю проблему іноді використовується підхід ReLU з «витоком» (leak) – графік функції активації на негативних значеннях виглядає як не горизонтальна пряма, а похила, з маленьким кутовим коефіцієнтом (близько 0,01) (рисунок 2.7). Тобто функція може бути записана так:

$$f(x) = \begin{cases} 0.01x, x < 0 \\ x, x \geq 0 \end{cases}, \tag{2.9}$$

де $f(x)$ – вихідний сигнал;

x – вхідний сигнал.

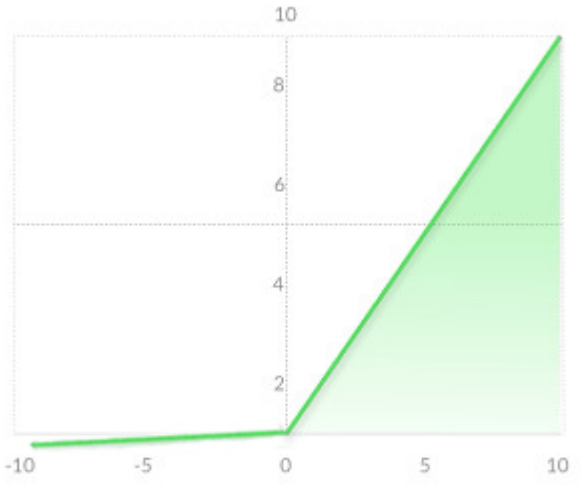

Рисунок 2.7 – Графік функції LReLU

Таке невелике від'ємне значення допомагає домогтися ненульового градієнта при негативних значеннях. Однак, функція Leaky ReLU має деякі недоліки:

– складніше розраховувати похідну, в порівнянні зі стандартним підходом (так як значення вже не дорівнюють нулю), що уповільнює роботу кожної епохи;

– кутовий коефіцієнт прямої також є гіперпараметром, який треба налаштовувати;

– на практиці, результат не завжди сильно поліпшується щодо ReLU.

Варто відзначити, що крім проблеми вмираючих нейронів, у ReLU є і інша – проблема затухаючого градієнта. При занадто великій кількості шарів градієнт буде приймати дуже маленьке значення, поступово зменшуючись до нуля. Через це нейронна мережа працює нестабільно і неправильно. Leaky ReLU (LReLU) вирішує першу проблему, але в по-справжньому глибоких мережах проблема загасання градієнта все ще зустрічається і при використанні цього підходу.

На практиці LReLU використовується не так часто. Практичний результат використання LReLU замість ReLU відрізняється не занадто сильно. Однак в разі використання LReLU потрібно додатково

налаштовувати гіперпараметр (рівень нахилу при негативних значеннях), що вимагає певних зусиль. Ще однією проблемою є те, що результат LReLU не завжди краще ніж звичайний ReLU, тому найчастіше такий підхід використовують як альтернативу. Досить часто на практиці використовується PReLU (Parametric ReLU), який дозволяє домогтися більш значних поліпшень в порівнянні з ReLU і LReLU. Також, в разі параметричної модифікації ReLU, кут нахилу не є гіперпараметром і налаштовується нейромережею.

Функція м’якого максимуму (SoftMax), часто використовується в нейронних мережах в якості функції активації при вирішенні задачі класифікації. Функція SoftMax задається формулою 2.10:

$$\sigma(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{N} e^{x_j}}, \tag{2.10}$$

де $\sigma(x)_i$ – вихідний сигнал *i*-го нейрона;

x_i – значення на виході з *i*-го нейрона до активації;

N – загальна кількість нейронів в шарі.

Функцію активації м'якого максимуму зручно застосовувати для задач класифікації, тому що вона дозволяє трактувати вихідні значення нейронів як ймовірність приналежності до даного класу. Це означає, що кожне значення має бути в діапазоні від 0 до 1, і сума всіх значень повинна дорівнювати одиниці. Також функція SoftMax забезпечує те, щоб тільки одне вихідне значення було близьке до одиниці за рахунок застосування експоненти.

Слід зазначити, що немає теоретичної необхідності використовувати саме SoftMax для задач класифікації. При наявності достатнього обсягу даних нейронна мережа може навчитися з іншими функціями активації на останньому шарі. Але з функцією м'якого максимуму навчання відбудеться швидше, тому що дана функція добре підходить саме для класифікації з непересічними класами.

Вибір активаційної функції визначається специфікою поставленого завдання або обмеженнями, що накладаються деякими алгоритмами навчання.

Для поєднання штучних нейронів в нейронні мережі вихід одного нейрона подається на вхід іншому нейрону. Нейронна мережа складається з деякої кількості нейронів, об’єднаних в шари (рисунок 2.8). Вхідний шар нейронів отримує сигнали від зовнішнього світу, вихідний – видає сигнали у зовнішній світ, а прихований шар нейронів (таких шарів може бути багато) отримує сигнали від нейронів вхідного шару і видає сигнал на вихідний шар. Прихований шар має таку назву тому що те, що в ньому відбувається, не видно із зовнішнього середовища.

Зміщення – це ваги, додані до прихованих шарів. Вони теж ініціалізуються випадковим чином і оновлюються так само, як і звичайні ваги прихованих шарів. Роль ваги прихованого шару полягає в тому, щоб визначити форму базової функції в даних, в той час як роль зміщенння – змістити знайдену функцію в сторону так, щоб вона частково збіглася з вихідною функцією.

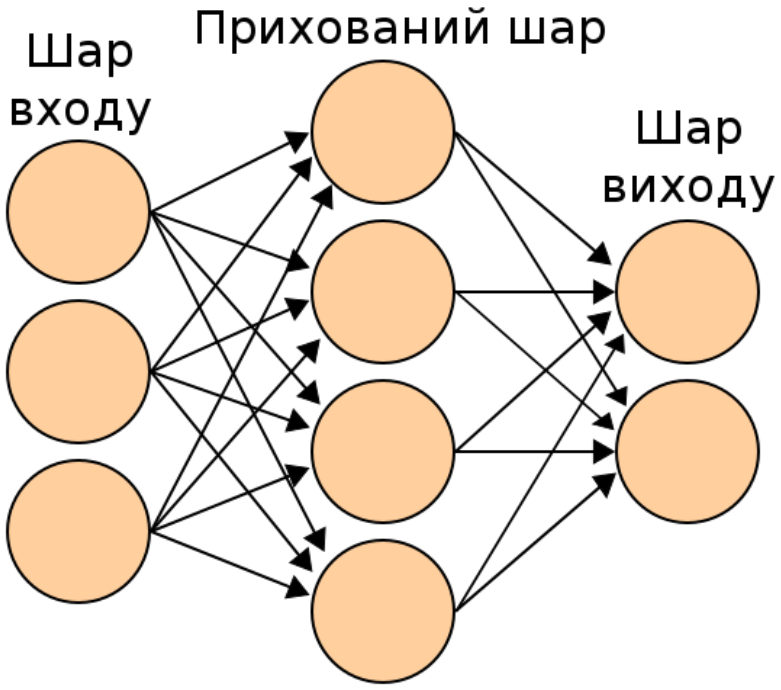

Рисунок 2.8 – Архітектура простої штучної нейронної мережі

Є два основні типи нейронних мереж: мережі з прямим поширенням сигналу (рисунок 2.9) і рекурентні мережі (рисунок 2.10).

В мережі з прямим поширенням сигналу немає циклів: сигнал, який поступає на вхідний шар передається на прихований шар (їх може бути кілька), далі на вихідний шар і у зовнішнє середовище.

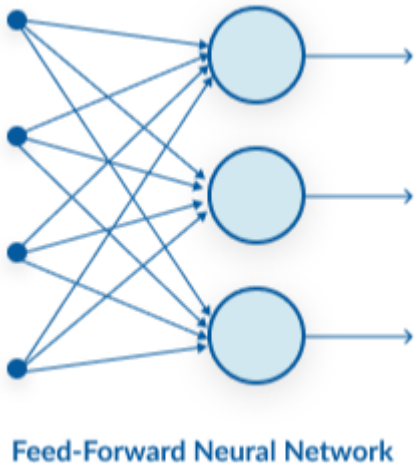

Рисунок 2.9 – Нейронна мережа із прямим поширенням сигналу

В рекурентних мережах можливі цикли. Це означає, що вихідний сигнал від нейрону може поступати на вхід до цього нейрона, інших нейронів цього ж шару або навіть до нейронів попереднього шару.

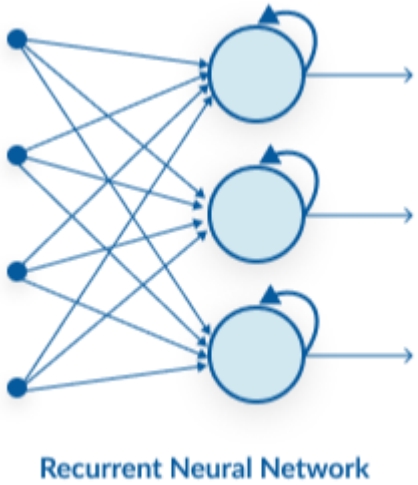

Рисунок 2.10 – Рекурентна нейронна мережа

Звичайні мережі з прямим поширенням сигналу, такі як CNN, беруть до уваги лише поточний вхідний сигнал. Тому вони не пам'ятають, що було в минулому. Таким чином, їм важко передбачати, що буде відбуватися в подальшому.

RNN відрізняються тим, що вони зберігають інформацію про раніше отримані вхідні дані. Це мережі з петлями зворотного зв’язку, які дозволяють отримувати інформацію, що зберігається – риса, аналогічна короткостроковій пам’яті Ця послідовна пам'ять зберігається у векторі прихованого стану рекурентної мережі та представляє контекст, що базується на попередніх входах та виходах. На відміну від мережі прямого поширення зв'язку, один і той же вхід може давати різні вихідні дані в залежності від попередніх входів.

Нейронні мережі можуть включати в себе кілька прихованих шарів. Виявилося, що такі багатошарові мережі мають більші можливості, ніж одношарові, і в останні роки були розроблені алгоритми для їх навчання. В такому випадку вони називаються глибокими нейронними мережами (рисунок 2.11).

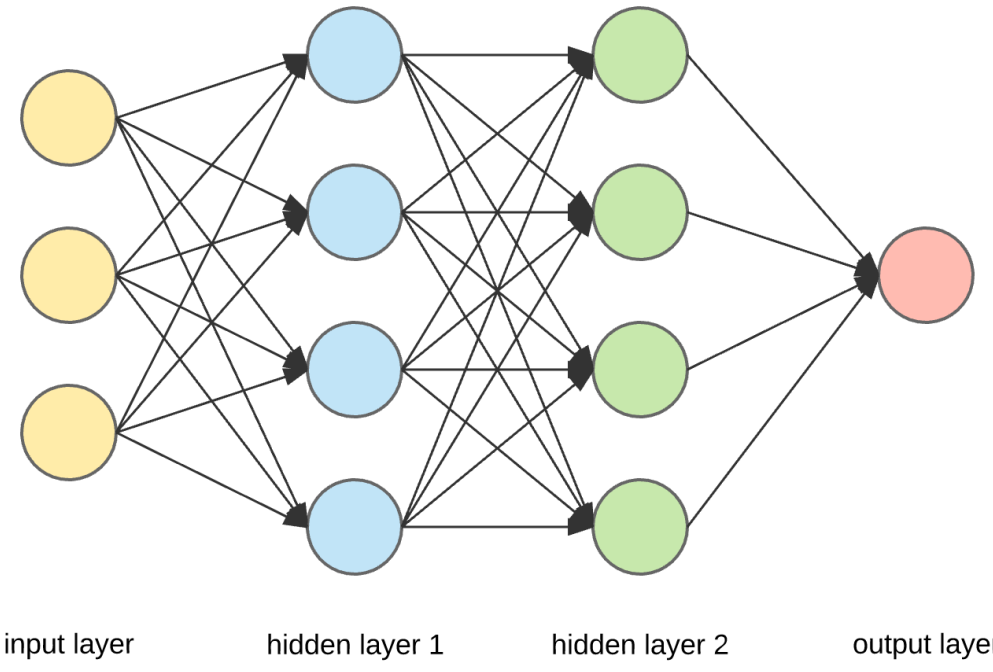

Рисунок 2.11 – Багатошарова нейронна мережа або глибока мережа

Отже, чим більше число прихованих шарів, тим більше можливості навчання мережі. Однак збільшення числа нейронів в одному шарі робить мережу широкою, а не глибокою, і не призводить до глибокого розуміння даних, а призводить до вивчення більшого числа ознак.

Дуже заманливо використовувати глибокі і широкі нейронні мережі для кожного завдання. Але це може бути поганою ідеєю, тому що:

– обидві вимагають значно більшої кількості даних для навчання, щоб досягти мінімальної бажаної точності;

– обидві мають експонентну складність;

– занадто глибока нейронна мережа спробує зламати фундаментальні уявлення, але при цьому вона буде робити хибні припущення і намагатися знайти псевдо-залежності, яких не існує;

– занадто широка нейронна мережа буде намагатися знайти більше ознак, ніж є. Таким чином, подібно до попередньої, вона почне робити неправильні припущення про дані.

Також виділяють повнозв'язні нейронні мережі (коли всі вузли одного шару з'єднані з усіма вузлами суміжних шарів) і неповнозв'язні (коли деякі синаптичні зв'язки відсутні).

2.3 Навчання нейронних мереж

Навчання нейронної мережі – це підбір вагових коефіцієнтів таким чином, щоб мережа вирішувала поставлену задачу [14]. Можливість навчання – одна з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. В процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними даними і вихідними, а також виконувати узагальнення. Це означає, що в разі успішного навчання мережа зможе повернути вірний результат на основі даних, які були відсутні в навчальній вибірці, а також неповних і/або «зашумлених», частково спотворених даних.

Існують наступні типи навчання нейронних мереж:

– навчання з учителем. Має бути набір даних (сигналів, що подаються на вхід нейронної мережі), для яких заздалегідь відомі правильні відповіді;

– навчання без учителя. Наявні дані, правильні відповіді для яких заздалегідь невідомі. Навчання без учителя ґрунтується на виявленні структурних відмінностей в даних;

– навчання з підкріпленням. Немає заздалегідь підготованих наборів даних, мережа діє як агент у зовнішньому середовищі і отримує сигнали від зовнішнього середовища про те, правильні дії виконуються чи ні.

Першим підхід до навчання нейронних мереж сформулював Хебб у 1949 році. Він використовував біологічні передумови: якщо нейрони головного мозку спрацьовують разом, то їх зв’язки зміцнюються, а якщо вони спрацьовують окремо, то зв’язки між ними послаблюються. Хебб розглядав нейрон, що видає сигнали 0 або 1, і задачу навчання з учителем, де необхідний набір вхідних сигналів, для яких заздалегідь відомі правильні відповіді. Початкові значення вагових коефіцієнтів вхідних сигналів задаються випадковим чином. Вхідні сигнали по черзі подаються на нейрон, перевіряється вихід нейрона і за необхідності змінюються значення вагових коефіцієнтів. Вагові коефіцієнти змінюються лише якщо нейрон видає неправильний сигнал. Якщо сигнал нейрона неправильний і дорівнює 0, то необхідно збільшити вагу тих входів, на які було подано 1. І навпаки, якщо сигнал нейрона неправильний і дорівнює 1, то вагу тих входів, на які було подано 1, необхідно зменшити.

Френк Розенблатт розробив інший підхід до навчання нейронної мережі. Навчання його персептрона відбувається наступним чином: якщо вихідний сигнал неправильний і дорівнює 0, то значення, які подаються на вхід, додаються до ваги, а якщо персептрон видає невірний сигнал рівний 1, то значення, які подаються на вхід, віднімаються до ваги. [1].

Зараз для навчання нейронних мереж найчастіше використовується метод зворотного поширення помилки. Вихідний сигнал, на відміну від перспетрона Розенблатта та правил Хебба, не бінарне, а дійсне число.

Необхідно задати певну міру помилки – на скільки дійсне число, яке видає мережа, відрізняється від правильної відповіді. Можливі різні варіанти міри помилки, найчастіше використовується середньоквадратична

Середньоквадратична міра помилки (MSE) – це різниця між значенням, виданим мережею, і правильною відповіддю, піднесена до квадрату. Вона повинна бути порахована для всіх об’єктів, що будуть використані для навчання. Середньоквадратична міра помилки розраховується за формулою 2.6.

$$\varepsilon = \frac{1}{2}\sum_{i=1}^{M}(a_i - y_i)^2\,, \tag{2.11}$$

де a_i – значення, видане мережею;

y_i – правильна відповідь;

M – кількість об'єктів, використаних для навчання.

Але MSE – це не єдиний варіант функції помилки, для мереж з вихідним шаром Softmax звичайно використовують середню крос-ентропію по всім об’єктам навчальної вибірки (формула 2.12):

$$E = \sum_{i=1}^{k}(-\log(p_i))\,, \tag{2.12}$$

де k – кількість об’єктів навчальної вибірки;

p_i – видана класифікатором ймовірність приналежності об’єкта до свого класу.

Середня крос-ентропія дозволяє отримати сумарне значення всіх нейронів на виході з шару рівне 1. Це дозволяє трактувати вихід із шару, який використовує таку функцію, як ймовірність.

Навчання полягає у зміні вагових коефіцієнтів таким чином, щоб міра помилки зменшувалася. Для цього використовується метод мінімізації градієнтного спуску. Градієнтний спуск – це спосіб знаходження

локального мінімуму або максимуму функції за допомогою руху уздовж градієнта. Необхідно знайти напрямок найшвидшого змінення функції помилки, який називається градієнтом, і рухатись у цьому напрямку. Щоб знайти напрямок градієнта треба взяти частинну похідну по ваговим коефіцієнтам. При зміні вагових коефіцієнтів для нейрона використовується так зване дельта-правило, що для простого лінійного нейрона описується формулою 2.13.

$$w_i = w_i - \eta \sum_{j=1}^{M} x_j^i \, (a_j - y_j), \tag{2.13}$$

де w_i – значення вагового коефіцієнта на попередньому кроці;

$\sum_{j=1}^{M} x_j^i \, (a_j - y_j)$ – частинна похідна помилки відповіді;

η – параметр швидкості навчання.

Якщо використовується нейрон з нелінійною функцією активації, то в дельта-правило додається похідна цієї функції активації.

Параметр швидкості навчання говорить про те, наскільки сильно змінюються вагові коефіцієнти. Цей параметр має бути достатньо маленьким, щоб не пропустити мінімальне значення функції помилки, але з іншої сторони достатньо великим, щоб навчання проводилось за допустимий час.

Існують три варіанти реалізації навчання:

– повне навчання. Вагові коефіцієнти змінюються після обробки всіх елементів навчальної вибірки. Однак коли даних багато такий підхід є неефективним;

– онлайн навчання (його ще інколи називають стохастичним). Вагові коефіцієнти змінюються після обробки кожного об’єкта;

– міні-вибірки. Вагові коефіцієнти змінюються після обробки 10-100 об’єктів.

Метод зворотного поширення помилки полягає в тому, що по значенню помилки на вихідному шарі розраховується значення помилки на

виході прихованого шару, потім на вході наступного прихованого шару і так далі до вхідного шару. Далі використовується дельта-правило для визначення того як потрібно змінити вагові коефіцієнти прихованого шару. Таким чином розраховується помилка від вихідного шару через приховані шари до вхідного шару, і саме тому метод навчання називається зворотне поширення помилки.

Щоб отримати помилку на вході в нейрон, якщо відома помилка на виході з нього, для лінійного нейрона використовується формула 2.14.

$$\varepsilon_i^h = \sum_{m=1}^{M} \varepsilon_i^m w_{hm}, \tag{2.14}$$

де ε_i^m– значення помилки на виході з нейрона;

w_{hm} – вага цього виходу.

Вибір даних для навчання мережі та їх обробка є найскладнішим етапом вирішення задачі. Набір даних для навчання повинен задовольняти декільком критеріям:

– репрезентативність – дані повинні ілюструвати справжній стан речей в предметній області;

– несуперечливість – суперечливі дані в навчальній вибірці призведуть до поганої якості навчання мережі.

Вихідні дані перетворюються до вигляду, в якому їх можна подати на входи мережі. Кожен запис у файлі даних називається навчальною парою або навчальним вектором. Навчальний вектор містить по одному значенню на кожен вхід мережі і, в залежності від типу навчання (з учителем або без), по одному значенню для кожного виходу мережі. Навчання мережі на «сирому» наборі, як правило, не дає якісних результатів. Існує ряд способів поліпшити «сприйняття» мережі. До них відноситься нормування, яке виконується, коли на різні входи подаються дані різної розмірності. Наприклад, на перший вхід мережі подаються величини зі значеннями від нуля до одиниці, а на другий – від ста до тисячі. При відсутності нормування

значення на другому вході будуть завжди надавати істотно більший вплив на вихід мережі, ніж значення на першому вході. При нормуванні розмірності всіх вхідних і вихідних даних зводяться воєдино.

В процесі навчання мережа в певному порядку переглядає навчальну вибірку. Мережі, які навчаються з учителем, переглядають вибірку багато разів, при цьому коли нейронна мережа пройшла один навчальний сет, вважається що вона пройшла одну ітерацію. Ітерація – це своєрідний лічильник, який збільшується кожен раз, коли нейронна мережа проходить один тренувальний сет. Іншими словами це загальна кількість тренувальних сетів, пройдених нейронною мережею. Один повний прохід по вибірці називається епохою навчання. При ініціалізації нейронної мережі ця величина встановлюється в 0, а її максимальне значення задається вручну. Чим більше епоха, тим краще натренована мережа, і відповідно краще її результат. Епоха збільшується кожного разу, коли мережа проходить весь набір тренувальних сетів.

При навчанні з учителем набір вихідних даних ділять на дві частини – навчальну вибірку і тестову вибірку; принцип поділу може бути довільним. Навчальні дані подаються мережі для навчання, а тестові використовуються для оцінки якості роботи мережі після завершення навчання (тестові дані ніколи для навчання мережі не застосовуються). Таким чином, якщо на тестових даних помилка зменшується, то мережа дійсно виконує узагальнення. Якщо помилка на навчальних даних продовжує зменшуватися, а помилка на тестових даних збільшується, значить, мережа перестала виконувати узагальнення і просто «запам'ятовує» навчальні дані. Це явище називається перенавчанням мережі або оверфіттінгом. У таких випадках навчання зазвичай припиняють. Перенавчання виникає, коли нейронна мережа адаптується до конкретних особливостей навчальної вибірки, а не до загальних закономірностей в даних. Щоб уникнути перенавчання, не варто довго тренувати ШНМ на одних і тих же або дуже схожих даних. Також, перенавчання може бути викликано неправильним підбором гіперпараметрів або неправильною архітектурою.

Гіперпараметри – це значення, які потрібно підбирати вручну. Серед таких значень можна виділити момент і швидкість навчання, кількість прихованих шарів, кількість нейронів у кожному шарі, наявність або відсутність нейронів зміщення, кількість епох навчання.

У процесі навчання можуть проявитися інші проблеми, такі як параліч або потрапляння мережі в локальний мінімум поверхні помилок. Неможливо заздалегідь передбачити прояв тієї чи іншої проблеми, так само як і дати однозначні рекомендації щодо їх вирішення.

2.4 Передобробка даних для тренування нейронної мережі

Нормалізація даних – це одна з операцій перетворення ознак (Feature Transformation), яка виконується при їх генерації (Feature Engineering) на етапі підготовки даних (Data Preparation).

Нормалізація – це процедура предобработки вхідної інформації (навчальних, тестових і валідаційних вибірок, а також реальних даних), при якій значення ознак у вхідному векторі приводяться до деякого заданому діапазону, наприклад, $[0, 1]$ або $[-1, 1]$.

Необхідність нормалізації вибірок даних зумовлена природою використовуваних алгоритмів і моделей машинного навчання. Вихідні значення ознак можуть змінюватися в дуже великому діапазоні і відрізнятися один від одного на кілька порядків. Припустимо, датасета містить відомості про концентрацію діючої речовини, яка вимірюється в десятих або сотих частках відсотків, і показники тиску в сотнях тисяч атмосфер. Або, наприклад, в одному вхідному векторі присутня інформація про вік і доходи клієнта.

Будучи різними за фізичним змістом, дані сильно розрізняються між собою за абсолютними величинами. Робота аналітичних моделей машинного навчання (нейронних мереж, карт Кохонена і т.д.) з такими показниками виявиться некоректною: дисбаланс між значеннями ознак може викликати нестійкість роботи моделі, погіршити результати навчання

і уповільнити процес моделювання. Зокрема, параметричні методи машинного навчання (нейронні мережі, що ростуть дерева) зазвичай вимагають симетричного і унімодального розподілу даних. Популярний метод найближчих сусідів, часто використовуваний в задачах класифікації та іноді в регресійному аналізі, також чутливий до діапазону змін вхідних змінних.

Після нормалізації всі числові значення вхідних ознак будуть приведені до однакової області їх зміни – деякого вузького діапазону. Це дозволить звести їх разом в одній моделі машинного навчання і забезпечить коректну роботу обчислювальних алгоритмів.

Data Preparation (підготовка даних) – досить трудомісткий ітеративний процес, який займає до 80% всіх витрат ресурсів і часу в життєвому циклі Data Mining і включає такі задачі опрацювання вхідних («сирих») даних:

– вибірка даних – відбір ознак (features або предикторов) і об'єктів з урахуванням їх релевантності для цілей Data Mining, якості і технічних обмежень (обсягу і типу);

– очищення даних – видалення помилок, неправильних значень (наприклад, число в строковому параметрі і ін.), відсутніх значень (Missing values або NA), виключення дублів і різних описів одного і того ж об'єкта, відновлення унікальності, цілісності і логічних зв'язків;

– генерація ознак – створення похідних ознак і їх перетворення в вектори для моделі Machine Learning, а також трансформація для підвищення точності алгоритмів машинного навчання;

– інтеграція – злиття даних з різних джерел (інформаційних систем, таблиць, протоколів тощо.), включаючи їх агрегацію, коли нові значення обчислюються шляхом підсумовування інформації з безлічі існуючих записів;

– форматування – синтаксичні зміни, які не змінюють значення даних, але потрібні для інструментів моделювання, наприклад, сортування

в певному порядку або видалення непотрібних знаків пунктуації в текстових полях, обрізка «довгих» слів, округлення дійсних чисел до цілого і т.д.

Далеко не завжди вихідні дані отримані з корпоративного сховища або вітрини даних і мають чітку структуру. А, всупереч громадській думці, машинне навчання не працює автономно і самостійно. Для адекватного функціонування цього інструменту, як і будь-якого IT-засоби, необхідні чітко визначені вихідні дані і інструкції. Неможливо завантажити в алгоритм Machine Learning всі накопичені великі дані різних форматів і отримати на виході коректні результати. Крім того, вихідні дані часто спотворені і ненадійні: в них можуть бути присутніми значення, що виходять за межі допустимих діапазонів (шуми), аномальні значення (викиди), а також пропуски (відсутність значень).

До того ж, часто виникає завдання попередньої підготовки вихідних даних. Наприклад, якщо стоїть завдання визначення тональності клієнтських відгуків, необхідно спочатку розбити текст на смислові вирази (токени), «оцифрувати» слова і перетворити їх в числові вектора. У географічних даних можуть зустрічатися помилки в адресах і помилки визначення координат через особливості місцевості, зокрема, в підвальних приміщеннях, серед пагорбів і т.д. У числових рядах можуть зустрічатися значення, що виходять за межі можливого діапазону, наприклад, цифра 7 в п'ятибальною шкалою оцінок. Також числові значення вихідних даних можуть сильно варіюватися за абсолютними величинами: від декількох сотих відсотків до десятків тисяч одиниць. Подібні похибки спотворять результати моделювання і не дозволять отримати модель машинного навчання з прийнятною якістю.

Вибірка даних для машинного навчання – це оброблена і структурована інформація в табличному вигляді. Рядки такої таблиці називаються об'єктами, а стовпці – ознаками. Розрізняють 2 види ознак:

– незалежні змінні – предиктори;

– залежні змінні – цільові ознаки, які обчислюються на основі одного або декількох предикторов.

Ознаковий опис характерний для задач класифікації, коли є вибірка – кінцева безліч об'єктів, для яких відома, до яких класів вони належать. Класова приналежність інших об'єктів невідома. В процесі машинного навчання будується модель, здатна класифікувати довільний об'єкт з початкової множини. Практичний сенс завдань класифікації полягає в передбаченні можливих результатів на основі сукупності вхідних змінних, наприклад, діагностика захворювань, попередня оцінка ефективності родовищ корисних копалин, кредитний скоринг, розпізнавання мови, прогнозування відтоку клієнтів і т.д.

В залежності від варіанту завдання класифікації, цільова ознака може виглядати по-різному:

– один стовпець з двійковими значеннями (1/0, TRUE/FALSE і ін.): двухкласова класифікація (binary classification), коли кожен об'єкт належить тільки одному класу;

– декілька стовпців з двійковими значеннями: завдання класифікації з пересічними класами (multi-label classification), коли один об'єкт може належати кільком класів;

– один стовпець з дійсними значеннями: регресійний аналіз, коли прогнозується одна величина;

– декілька стовпців з дійсними значеннями: завдання множинної регресії, коли прогнозується кілька величин.

Первинний набір вихідних даних прийнято називати генеральною сукупністю. Процес формування вибірок з генеральної сукупності називається породження даних. Вибірка – це кінцеве підмножина елементів генеральної сукупності, вивчивши яке можна зрозуміти поведінку вихідного безлічі. Наприклад, генеральна сукупність складається з 150 тисяч відвідувачів сайту, а в вибірку потрапили 250 з них.

Імовірнісна модель породження даних передбачає, що вибірка з генеральної сукупності формується випадковим чином. Якщо все її елементи однаково випадково і незалежно один від одного розподілені по вихідного безлічі (генеральної сукупності), вибірка називається простий.

Проста вибірка є математичною моделлю серії незалежних дослідів і, як правило, використовується для машинного навчання. При цьому для кожного етапу Machine Learning необхідний свій набір даних:

– для безпосереднього навчання моделі потрібна навчальна вибірка (training sample), по якій проводиться налаштування (оптимізація параметрів) алгоритму;

– для оцінки якості моделі використовується тестова (контрольна) вибірка (test sample), яка, в ідеальному випадку, не повинна залежати від навчальної;

– для вибору найкращої моделі машинного навчання знадобиться перевірочна (валідаційну) вибірка (validation sample), яка також не повинна перетинатися з навчальною.

Вибірка, отримана в результаті першого етапу підготовки даних (Data Preparation), ще поки не придатна для обробки алгоритмами машинного навчання, оскільки інформацію необхідно очистити.

Очищення даних – процес обробки вибірки для інтелектуального аналізу інформації (Data Mining) за допомогою алгоритмів машинного навчання. Цей етап, на якому виконується виявлення і видалення помилок і невідповідностей в даних з метою поліпшення якості датасета, також називається data cleaning, data cleansing або scrubbing. Некоректна інформація, інформація, що дублюється або втрачена інформація може стати причиною неадекватної статистики і невірних висновків в контексті бізнесу. Тому очищення даних є обов'язковою процедурою Data Preparation.

Існують наступні 2 види проблем з даними, від яких позбавляє процедура їх очищення :

– проблеми з ознаками – значеннями змінних, стовпцями в табличному поданні датасета;

– проблеми із записами – об'єктами, які є рядками датасета і описуються значеннями ознак.

На рівні ознак виділяють 6 основні проблеми:

– неприпустимі значення, які лежать поза потрібного діапазону, наприклад, цифра 7 в поле для шкільних оцінок за п'ятибальною шкалою;

– відсутні значення, що не введені, безглузді або не визначені, наприклад, число 000-0000-0000 як телефонного номера;

– орфографічні помилки – неправильне написання слів: «водітл» замість «водій» або «Омськ» замість «Томськ», що спотворює первинний сенс змінної, підставляючи замість одного міста інший;

– багатозначність: використання різних слів для опису одного і того ж за змістом значення, наприклад, «водій» і «шофер» або застосування однієї абревіатури для різних за змістом значень, наприклад, «БД» може бути скороченням для словосполучення «великі дані» або «база даних»;

– перестановка слів, зазвичай зустрічається в текстових полях вільного формату;

– вкладені значення – кілька значень в одному ознаці, наприклад, в поле

– вільний формат.

На рівні записів виділяють 4 основні проблеми:

– порушення унікальності, наприклад, паспортного номера або іншого ідентифікатора;

– дублювання записів, коли один і той же об'єкт описаний двічі;

– суперечливість записів, коли один і той же об'єкт описаний різними значеннями ознак;

– невірні посилання – порушення логічних зв'язків між ознаками.

Метод очищення даних повинен задовольняти ряду критеріїв:

– бути здатним виявляти і видаляти всі основні помилки і невідповідності, як в окремих джерелах даних, так і при інтеграції декількох джерел;

– підтримуватися певними інструментами, щоб скоротити обсяги ручної перевірки та програмування;

– бути гнучким в плані роботи з додатковими джерелами.

Генерація ознак – мабуть, самий творчий етап підготовки даних (Data Preparation) для машинного навчання (Machine Learning). Цей етап ще називають Feature Engineering. Він настає після того, як вибірка сформована і очищення даних завершена.

Ознака (фіча, feature) – це змінна, яка описує окрему характеристику об'єкта. У табличному вигляді вибірки ознаки – це стовпці таблиці, а об'єкти – рядки. Вхідні, незалежні, змінні для моделі машинного навчання називаються предикторами, а вихідні, залежні, – цільовими ознаками. Всі ознаки можуть бути наступних видів:

– бінарні, які приймають два значення, наприклад, {true, false}, {0,1}, {1,1}, { «так», «ні»} і т.д.;

– номінальні (фактори), які мають кінцеве кількість рівнів, наприклад, фактор «день тижня» має іменованих 7 рівнів: понеділок, вівторок і т. д. Фактори можуть бути впорядкованими та неупорядкованими. Наприклад, фактор «час доби» має 24 рівня і він впорядкований. Фактор «район міста» з 32 рівнями не впорядкований, оскільки всі рівні мають рівну значущість. Якщо фактор впорядкований, це варто явно вказати при його оголошенні;

– кількісні (числові) значення в діапазоні від мінус нескінченності до плюс нескінченності.

Ознаки можуть вилучатись з даних будь-якого типу, в т.ч. з тексту, зображень та геоданих. При обробці текстової інформації спочатку виконується її токенізація, а потім лематизація і цифровізація – переклад слів в числові вектора. У разі зображень часто аналізується не тільки зміст картинки як набору пікселів різного кольору, але і метадані графічного файлу: дата зйомки, розширення, модель камери і т.д. Географічні дані найчастіше представлені у вигляді адрес (текст) або пар «широта + довгота» (числових наборів – точок).

Генерація ознак включає в себе 3 взаємопов'язані завдання, кожній з яких ми присвятили окрему статтю:

– витяг ознак (feature extraction) – перетворення даних, специфічних для предметної області, в зрозумілі для моделі числові вектори. Зокрема, саме тут виконується токенізація і лематизація текстів, обробка зображень і геоданих;

– перетворення ознак (feature transformation) – зміна даних для підвищення точності алгоритму, наприклад, нормалізація або зміна імовірнісного розподілу;

– відбір ознак (feature selection) – відсікання непотрібних ознак за допомогою алгоритмів машинного навчання, які дозволяють оцінити важливість предиктора, наприклад, жадібний алгоритм, логістична регресія, випадковий ліс, градієнтний бустінг.

Після того, як генерація ознак завершена, наступають етапи інтеграції та форматування датасета, щоб, нарешті, приступити до моделювання, тобто машинного навчання.

3 ПРОЕКТНІ РІШЕННЯ

3.1 Вибір програмних засобів для розробки нейронної мережі

Для вирішення поставленої задачі обрано мову програмування Python. Вона широко використовується для машинного навчання, так як писати на цій мові доволі просто і швидко – для реалізації тої ж самої функціональності на C++ або Java знадобилось би набагато більше строк коду. Також Python має простий синтаксис і динамічну типізацію. При використанні нейронних мереж необхідно проводити велику кількість експериментів: визначити архітектуру нейронної мережі, що буде підходити для поставленої задачі, оцінити гіперпараметри навчання і таке інше. Для цього добре підходить мова Python, тому що дозволяє легко модифікувати програму для проведення різноманітних експериментів, і в той же час програма при таких модифікаціях залишається такою, що легко читається і добре підтримується.

Та все ж одна з найбільших переваг Python і головна причина чому він використовується для машинного навчання полягає в тому, що у нього є безліч фреймворків і бібліотек, які спрощують процес написання коду і скорочують час на розробку. Основними фреймворками і бібліотеками, які використовуються у машинному навчанні, а також будуть використані у цій роботі є: NumPy, SciPy, Scikit-learn, Matplotlib, Pandas, Tensor Flow та Keras.

NumPy – основна бібліотека Python, яка спрощує роботу з векторами і матрицями. Містить готові методи для різних операцій: від створення, зміни форми, множення і розрахунку детермінанта матриць до вирішення лінійних рівнянь і сингулярного розкладання [16].

SciPy – бібліотека для наукових розрахунків, що ґрунтується на NumPy і розширює її можливості. Включає методи лінійної алгебри і методи для роботи з ймовірнісними розподілами, інтегральним обчисленням і перетвореннями Фур'є [17].

Scikit-learn – бібліотека, що ґрунтується на NumPy і SciPy. Надає алгоритми для машинного навчання та інтелектуального аналізу даних: кластеризації, регресії і класифікації [18].

Matplotlib – низькорівнева бібліотека для створення двовимірних діаграм і графіків. Важливою задачею машинного навчання є візуалізація даних. За допомогою бібліотеки Matplotlib можна побудувати будь-який графік [19].

Pandas – бібліотека, що надає структури даних і інструменти для аналізу. Підходить для обробки неповних, невпорядкованих і немаркованих даних (як раз такі найчастіше і зустрічаються в житті). Pandas дозволяє замінити досить складні операції з даними на одну-дві команди. Містить багато готових методів групування, фільтрації, об'єднання даних, а також можливість розпізнавання різних типів джерел. У бібліотеці можна об'єднувати таблиці по аналогії з SQL JOIN. При цьому дані беруться прямо з файлів, завдяки чому відпадає необхідність в організації баз даних. Ще одна перевага Pandas – швидкість роботи [20].

TensorFlow – бібліотека, що використовується для налаштування, тренування і застосування штучних нейронних мереж з численними наборами даних. TensorFlow – бібліотека машинного навчання від компанії Google, яка зараз є одною з провідних компаній, що використовують на практиці машинне навчання. TensorFlow – це бібліотека з відкритими вихідними кодами. Два основних компоненти, які відображені у назві – це робота з багатовимірними матрицями (тензорами), які широко використовуються при навчанні нейронних мереж і розрахунки на графах потоків даних (flow) між якими передаються тензори. Недолік TensorFlow полягає у тому, що замість нейронної мережі необхідно описувати граф потоків даних, який відповідає потрібній нейронній мережі, тобто поверх ефективних обчислень з тензорами самостійно реалізувати нейронну мережу [21].

Тому для розробки програми буде використана ще одна допоміжна бібліотека глибокого навчання – Keras, яка використовує TensorFlow для

виконання ефективних обчислень. Особливість бібліотеки Keras полягає в тому, що вона дозволяє на Python описувати нейронну мережу. За допомогою цієї бібліотеки можна вказати з яких шарів буде складатися нейронна мережа, які будуть використовуватись функції активації, метод оптимізації для зменшення помилки та інші параметри важливі для навчання нейронної мережі. Далі бібліотека Keras сама побудує необхідну нейронну мережу і для проведення обчислень буде викликати високоефективні методи з TensorFlow [22].

Традиційні програми на Python працюють повільно, але наявні бібліотеки глибокого навчання використовують різноманітні методи для пришвидшення програм – використовуються оптимізовані математичні бібліотеки, написані на C, деякі бібліотеки на основі коду на Python автоматично генерують код на C і виконується вже цей код, що працює значно швидше. Для ще більшого підвищення продуктивності є можливість використовувати графічні прискорювачі (GPU) за допомогою NVIDIA cuDNN.

3.2 Вимоги та вибір програмних засобів для розробки мобільного додатку

На основі функціональних вимог до інформаційної системи, можна зробити такі висновки, що база даних повинна мати такі властивості:

– можливість легко розширювати функціонал;
– можливість працювати з мовою програмування Swift;
– простота у використанні;
– швидкість роботи запитів.

Розглянемо декілька варіантів баз даних, що так чи інакше задовольняють цим умовам, а саме: SQLite та Realm. Розглядати MySQL не має сенсу бо не можливо на пряму використовувати цю базу даних з мовою програмування Swift.

SQLite – це вбудована в бібліотеку мови програмування С база даних, розроблена Річардом Хіппом у 2000 році. Це безкоштовне і відкрите джерело. Основна ідея SQLite – позбутися архітектури сервер-клієнт і зберегти всю інформацію про додаток безпосередньо на мобільному пристрої. БД написана на С і може підтримувати більше 30 мов програмування. Існують тисячі мобільних додатків, які використовують SQLite у своїй архітектурі. Проте, незважаючи на всі великі можливості SQLite DB, він також має цілий ряд недоліків, наприклад:

– безпека. Перш за все, потрібно звернути увагу на те, що SQLite DB зберігається в одному файлі. Цей файл можна помістити, де б ви не хотіли, в ієрархію каталогів. На перший погляд це може здатися дуже зручним, але існує ризик того, що базу даних можна відкрити і переписати будь-яким нечесним процесом. Ось чому безпеку слід надавати на рівні файлу;

– ще одна проблема SQLite щодо безпеки виникає, коли мова йде про журналювання. Щоб мати можливість відкату, база даних створює тимчасовий файл журналу. Як правило, вони видаляються після операції. Однак у деяких випадках ці файли не видаляються. У результаті дані в такій базі даних можуть бути пошкоджені через цю особливість бази даних SQLite. Якщо відключити журналювання, база даних буде пошкоджена, якщо програма аварійно завершиться. Як опція, ви можете зашифрувати ваші дані перед тим, як помістити їх у БД. Крім того, SQLite як спеціальне розширення для цих цілей;

– масштабованість. Логічно, що база даних додатка буде рости з часом. Разом з тим, зростатиме потреба в написанні більш складних запитів. Більшість молодих програмістів стикаються з багатьма проблемами під час цієї фази, оскільки вони мають невеликий досвід роботи з структурованою мовою запитів. Така ж проблема виникає під час міграції даних.

Враховуючи усі мінуси, можна зробити висновок, що SQLite є не найкращім рішенням для бази даних.

Іншим варіантом є база даних Realm. Realm – крос-платформна мобільна база даних для iOS (доступна в Swift & Objective-C) і Android.

Realm була створена, щоб стати краще і швидше, ніж SQLite і Core Data. Вона не тільки краще і швидше, але і проста у використанні, так ви можете зробити багато з допомогою всього декількох рядків коду. Realm є абсолютно безкоштовною, і ви можете використовувати її без будь-яких обмежень. Realm розроблена, щоб бути простою у використанні, так як вона не ORM, і вона використовує свій власний механізм персистентності для більшої продуктивності і швидкості виконання [24]. Realm неймовірно швидка і проста у використанні, Вам буде потрібно всього пара рядків коду для виконання будь-якого завдання незалежно від того чи читає вона або записує в базу даних. Можна виділити декілька дуже важливих переваг Realm:

– простота. База даних Realm надзвичайно проста в порівнянні з SQLite. Realm набагато лаконічніший і вимагає написання мінімального коду;

– швидкість. Декілька прикладів порівняння Realm, SQLite та бібліотек на основі SQLite наведені у рисунках 3.1 – 3.3. Порівняння проводилося за кількома параметрами, а саме: підрахунок записів, які відповідають умовам запиту в базі на 200 тис. записів, перегляд всіх записів, що відповідають запиту та вставка 200 тис. записів в рамках одної транзакції. Таким чином, можна зробити висновок, що Realm дійсно досить швидкий та у більшості випадків швидше навіть за чистий SQLite;

– документація. Це завжди великий плюс для будь-якого інструмента, який використовується. Realm добре задокументований, і якщо є які-небудь питання, ви завжди можете відвідати його офіційний сайт для більш докладної інформації;

– додаткові функції. Оскільки Realm є значно молодшим інструментом у порівнянні з SQLite, він може похвалитися багатьма новими функціями, які є великим бонусом для всіх розробників. Наприклад, можна отримати підтримку JSON, підтримку шифрування та вільний API, використовуючи Realm у проектах.

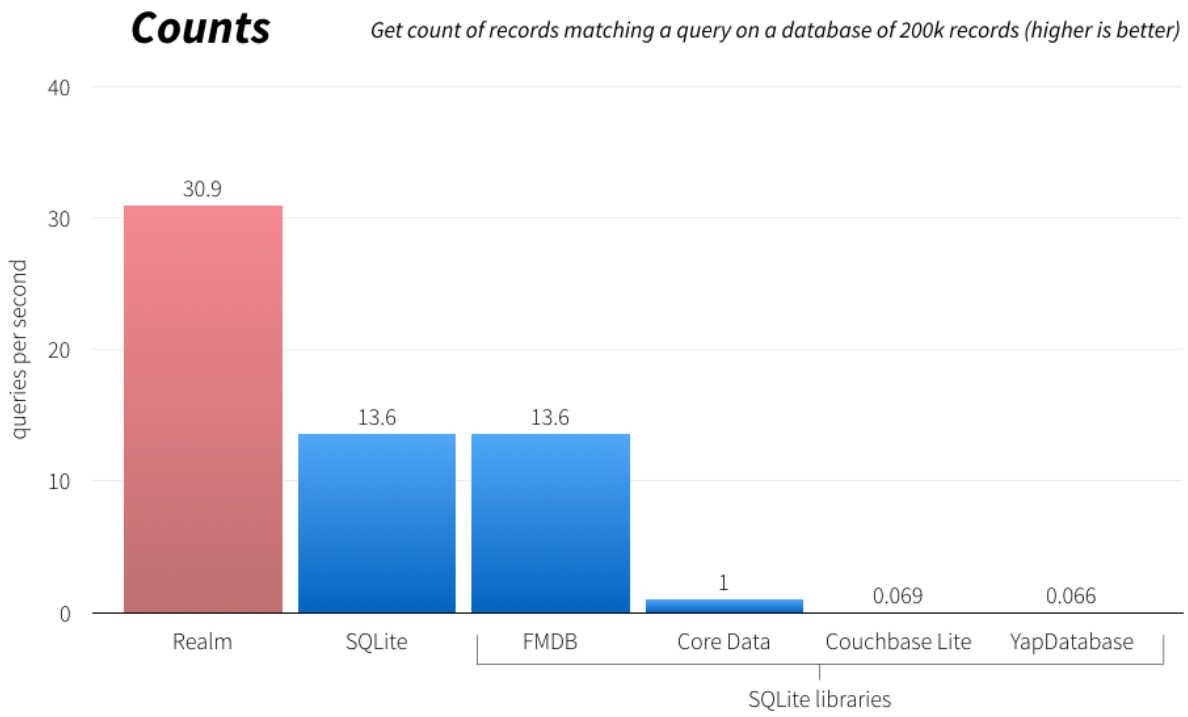

Рисунок 3.1 – Демонстрація швидкості обробки запитів при підрахуванні записів, що відповідають запитові

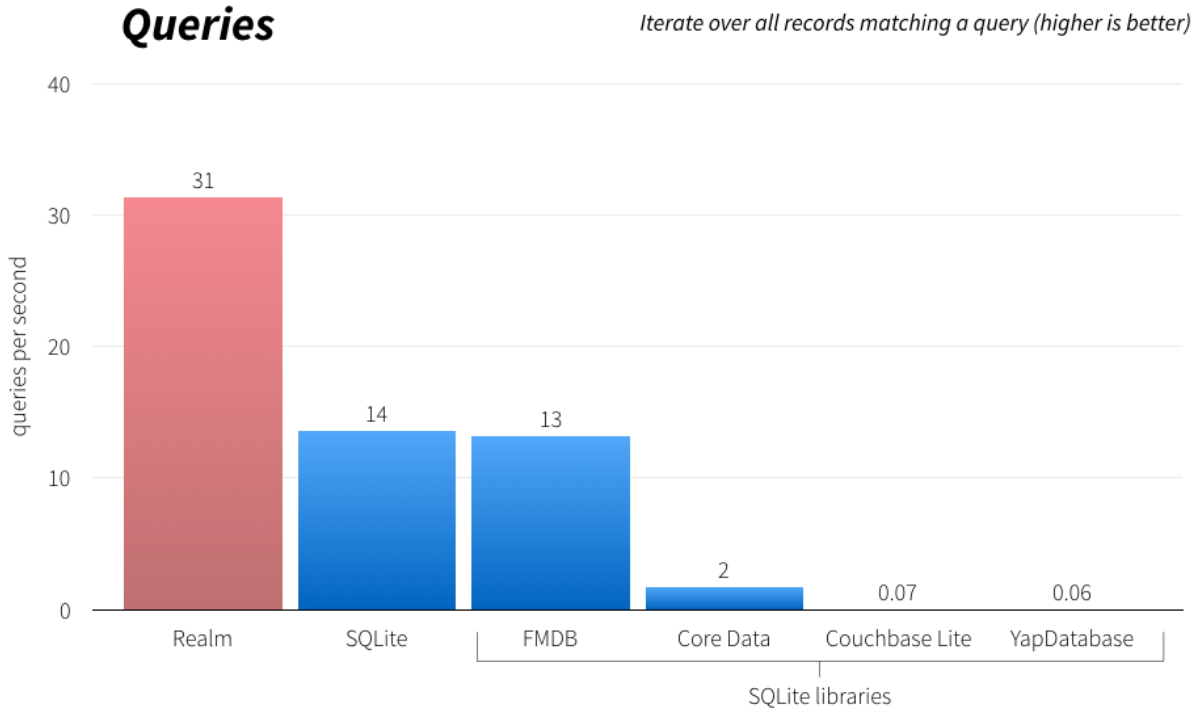

Рисунок 3.2 – Демонстрація швидкості обробки запитів при перегляді усіх записів, що відповідають запитові

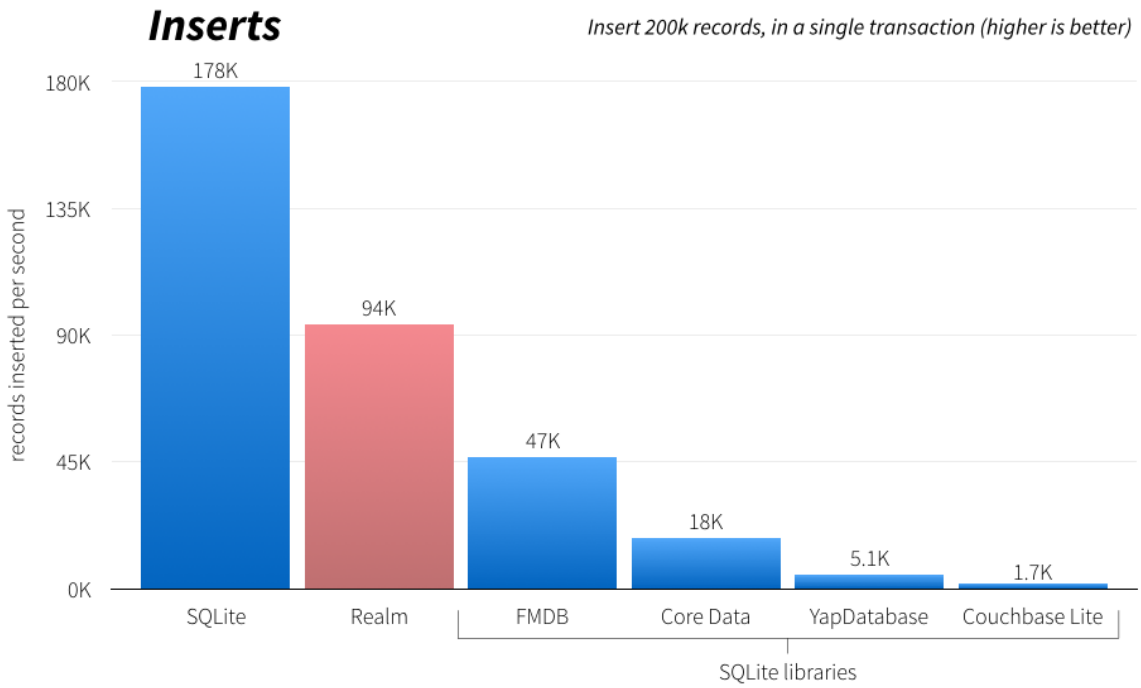

Рисунок 3.3 – Демонстрація швидкості обробки запитів при вставці 200 тис. записів в рамках однієї транзакції

Також можна додати до списку порівнянь вбудовану в операційну систему iOS базу даних – Core Data. По суті ця база даних являється обгорткою над чистою SQLite, та працює навіть гірше за неї. До того ж робота з даною базою хоча й нативна, але досить складна, що не робить її більш привабливою для вибору у якості основної бази даних для розробки мобільного додатку.

Таким чином, порівнявши три бази данних (CoreData, SQLite, Realm), очевидний вибір пав на Realm. Отож основною базою даних у інформаційній системі стане саме база даних Realm.

3.3 Логічне та фізичне моделювання бази даних.

Логічне проектування бази даних – це процес створення моделі використовуваної на підприємстві інформації на основі обраної моделі

організації даних, але без урахування типу цільової СУБД і інших фізичних аспектів реалізації.

Другий етап проектування бази даних називається логічним проектуванням бази даних. Його мета полягає в створенні логічної моделі даних для досліджуваної частини підприємства. Концептуальна модель даних, створена на попередньому етапі, уточнюється і перетворюється в логічну модель даних. Логічна модель даних враховує особливості обраної моделі організації даних в цільової СУБД (наприклад, реляційна модель).

Якщо концептуальна модель даних не залежить від будь-яких фізичних аспектів реалізації, то логічна модель даних створюється на основі обраної моделі організації даних цільової СУБД. Інакше кажучи, на цьому етапі вже має бути відомо, яка СУБД буде використовуватися в якості цільової – реляційна, мережева, ієрархічна або об'єктно-орієнтована. Однак на цьому етапі ігноруються всі інші характеристики обраної СУБД, наприклад, будь-які особливості фізичної організації її структур зберігання даних і побудови індексів.

В процесі розробки логічна модель даних постійно тестується і перевіряється на відповідність вимогам користувачів.

Створена логічна модель даних є джерелом інформації для етапу фізичного проектування і забезпечує розробника фізичної бази даних засобами пошуку компромісів, необхідних для досягнення поставлених цілей, що дуже важливо для ефективного проектування. Логічна модель даних відіграє також важливу роль на етапі експлуатації і супроводу вже готової системи. При правильно організованому супроводі підтримувана в актуальному стані модель даних дозволяє точно і наочно уявити будь-які вносяться в базу даних зміни, а також оцінити їх вплив на прикладні програми та використання даних, вже наявних в базі [25].

Фізичне проектування бази даних – процес підготовки опису реалізації бази даних на вторинних запам'ятовуючих пристроях; на цьому етапі розглядаються основні відносини, організація файлів і індексів,

призначених для забезпечення ефективного доступу до даних, а також всі пов'язані з цим обмеження цілісності і засоби захисту.

Фізичне проектування є третім і останнім етапом створення проекту бази даних, при виконанні якого проектувальник приймає рішення про способи реалізації розроблюваної бази даних. Хоча ця структура не залежить від конкретної цільової СУБД, вона створюється з урахуванням обраної моделі зберігання даних, наприклад реляційної, мережевої або ієрархічної. Однак, приступаючи до фізичного проектування бази даних, перш за все необхідно вибрати конкретну цільову СУБД. Тому фізичне проектування нерозривно пов'язане з конкретною СУБД. Між логічним і фізичним проектуванням існує постійний зворотний зв'язок, так як рішення, що приймаються на етапі фізичного проектування з метою підвищення продуктивності системи, здатні вплинути на структуру логічної моделі даних.

Як правило, основною метою фізичного проектування бази даних є опис способу фізичної реалізації логічного проекту бази даних [26].

Тому для того, щоб безпомилково розробити модель бази даних, було проведено логічне моделювання бази даних. Було вирішено, що для обраної предметної області достатньо всього однієї таблиці. Результат логічного моделювання зображений на рисунку 3.4.

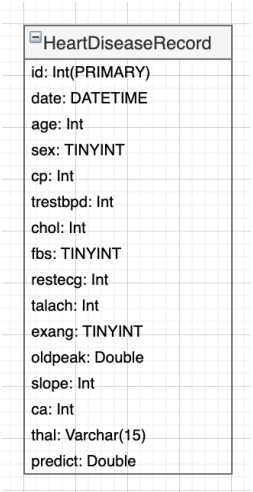

Рисунок 3.4 – Логічне моделювання бази даних

Далі наводиться фізичний опис усіх об’єктів бази даних, які приймають участь у системі моніторингу параметрів самопочуття користувача.

База даних складається з однієї таблиці – HeartDiseaseRecord. Таблиця містить у собі параметри, які використовуються при роботі з нейронною мережею, а також унікальний ідентифікатор запису, дату створення запису, а також результат прогнозу зроблений нейронною мережею для цього набору даних.

Нижче приведено опис усіх полів таблиць:

– id – унікальний ідентифікатор запису;

– date – дата створення запису;

– age – вік користувача у повних роках;

– sex – стать користувача;

– cp (chest pain type) – тип болю в грудях;

– trestbpd – артеріальний тиск у спокої (у мм рт. ст.);

– chol – холестерин в сироватці крові в мг/дл;

– fbs – рівень цукру в крові вище 120 мг / дл (1 = істинно; 0 = хибно);

– restecg – результати електрокардіограми в спокої;

– thalach – максимальна частота серцевих скорочень;

– exang – ангіна, спричинена фізичними вправами (1 = так; 0 = ні);

– oldpeak – депресія ST, спричинена фізичними вправами щодо відпочинку;

– slope – нахил пікового сегмента вправи ST;

– ca – ількість основних судин (0-3), пофарбованих флюороскопією;

– thal – 3 = нормальний; 6 = виправлений дефект; 7 = оборотний дефект;

– predict – спрогнозована ймовірність виникнення захворювання.

У результаті було отримано чітку картину щодо того, як повинна виглядати база даних, поля має містити у собі таблиця. Слід додати, що стан бази даних може відрізнятися від фінальної версії.

3.4 Обгрунтування вибору мови програмування

Виходячи с того, що метою роботи є розробка мобільного додатку під платформу iOS, то нижче наведені інструменти розробки, які задовольняють цій вимозі:

– React Native;
– Xamarin;
– Flutter;
– Objective-C;
– Swift.

Усі мови підходять для того, щоб розробити мобільний додаток під платформу iOS, але три з них використовуються для розробки кросплатформених додатків і лише дві використовуються у якості нативних інструментів розробки. Отже, розглянемо усі вище названі мови, їх переваги та недоліки.

Почнемо з Flutter. Flutter, новачок серед вище названих інструментів розробки, забезпечує легку і безперервну крос-платформну розробку мобільних додатків. Ви не повинні окремо створювати додатки для iOS та Android. Все, що вам потрібно, це єдина кодова база для обох платформ.

Що таке «Flutter»? Це безкоштовна платформа з відкритим вихідним кодом. Він заснований на Dart – швидкій, об'єктно-орієнтованій мові програмування, яку легко вивчити. Вона забезпечує свої власні гаджети, намальовані власним чудовим двигуном рендеринга. Вони швидкі, красиві і регульовані. Архітектура Flutter заснована на реактивному програмуванні, яке сьогодні популярне.

Переваги Flutter:

– більш швидке написання коду, що зазвичай займає кілька секунд, допомагає командам додавати функції, виправляти помилки та швидше експериментувати;

– розробникам потрібно написати лише одну кодову базу, яка використовується для додатків, що охоплюють як платформи Android, так і iOS;

– наявність подібного додатка для двох платформ означає, що для тестування потрібно менше часу, а процес забезпечення якості може бути швидшим;

– просто зробити свої власні віджети або налаштувати існуючі віджети, такі як віджети для дизайну матеріалів або віджети Cupertino;

– ідеально підходить для MVP, тому що вам потрібно побудувати додаток за менший час.

Недоліки Flutter:

– Flutter все ще перебуває у бета-версії та ще не випустила стабільну версію. Він також не підтримується платформами безперервної інтеграції (CI), такими як Travis або Jenkins. Таким чином, для того, щоб досягти автоматичного створення, тестування додатків і розгортання, ваша команда розробників повинна використовувати і зберігати спеціальний контент;

– Flutter повністю підтримується Google і існує кілька корисних бібліотек з функціональними можливостями, які готові до реалізації. Проте, Flutter до цих пір зовсім не так багатий на рідний розвиток.

Далі розглянемо Xamarin. Xamarin – це платформа, розроблена компанією Xamarin, яка була придбана компанією Microsoft в 2016 році, призначена для створення додатків для Android, iOS, macOS, Tizen, GTK# і Windows з однією спільною .NET базою коду. Будівельні програми з Xamarin дозволяють розробникам створювати програми для всіх основних платформ одночасно, які не відрізняються від рідних додатків. Фактично, програми Xamarin є рідними, використовуючи всі переваги власних елементів керування інтерфейсом користувача, специфічні для платформи API та специфічне для платформи апаратне прискорення.

Переваги Xamarin:

– структура використовує мову C #, яка працює бездоганно через безліч платформ (Android, iOS і Windows);

– більше 1,4 мільйона розробників використовували Xamarin в першому кварталі 2017 року, завдяки його здатності писати рідну коду UI для розробки мобільних додатків;

– якщо ви створюєте додаток для декількох платформ (наприклад, Android, iOS тощо), ви можете спільно використовувати або повторно використовувати більшу частину свого коду у вашому додатку для інших платформ;

– доступ до рідного API: у вашому додатку можна отримати функції, специфічні для певної платформи;

– декілька елементів керування та макетів, що надаються в Xamarin, допомагають при швидшому процесі розробки;

– швидше розробка додатків за допомогою плагінів NuGet і Xamarin;

– використовується багатьма відомими брендами, ця межплатформенная платформа розробки додатків пропонує конкурентоспроможність у розробці видатних інтерфейсів нативних програм, які успішно перевершують обмеження гібридних додатків.

Недоліки Xamarine:

– Xamarin не є безкоштовним і вам потрібно не тільки придбати рамки, але й ціну. Отже, це зовсім не для підприємств стартапів або бюджетних обмежень;

– обмежений доступ до різних життєво необхідних бібліотек для створення програми, що використовує цей фреймворк;

– розробка інтерфейсу користувача (UI) вимагає багато часу, оскільки створення інтерфейсу користувача не є мобільним;

– Xamarin страждає від низької активності громади, що не є корисним для розробки рамки.

Далі розглянемо React Native. React Native – це фреймворк з відкритим вихідним кодом, створений Facebook, Instagram та великою спільнотою розробників. Він використовує JavaScript і JSX (JavaScript-XML) для

створення власних програм для iOS і Android. За допомогою React Native ви можете створити повне мобільне додаток так само, як це було б у Java, Objective-C або Swift.

Використовуючи власні компоненти компонування, React Native дозволяє створювати переконливі інтерфейси користувача, які не відрізняються від рідної програми (Android або iOS). Код розділяється між цими двома різними кодами, але для компонування і-е елементи дизайну, такі як види, стилі тощо, використовують рідні компоненти кожної платформи. Саме там розміщується розмітка JSX. JSX використовується для створення макета для iOS або Android, використовуючи відповідні рідні компоненти, таким чином, ім'я React NATIVE.

Переваги React Native:

– React Native пропонує різні готові до застосування елементи, які можуть прискорити час розробки;

– Hot Reload: Ви можете перезавантажити програму швидко, без перекомпіляції;

– пряме використання власного коду для оптимізації вашого додатку до великого рівня;

– спільне використання коду на обох платформах – iOS і Android;

– швидше розробка додатків з вбудованими елементами;

– доступ до функціональних можливостей, таких як камера, акселерометр тощо;

– якісний мобільний інтерфейс користувача.

Недоліки React Native:

– навігація у вашому мобільному додатку не буде рівнозначно нативній навігації. Навігація, розроблена в React Native, не така гладка;

– як Xamarin, React Native навіть дає вам можливість розробити чудові якісні програми. Але програми, побудовані за допомогою React Native, повільніші, ніж нативні програми для Android, створені за допомогою Java і рідних програм iOS, побудованих за допомогою Objective-C і Swift.

Отже, розглянувши 3 кроссплатформені інструменти розробки, можна зробити висновок, що вони усі являються чудовими і якісними інструментами розробки, але додатки розроблені за допомогою них суттєво гірші за додатки написані нативними інструментами розробки, такими як Objective-C та Swift.

Так як метою даної роботи є розробка мобільного додатку під платформу iOS, було проаналізовано і вивчено переваги та недоліки кросплатформених інструментів розробки та було вирішено не використовувати їх у якості основних інструментів розробки, а натомість використовувати нативні мови програмування такі як Objective-C та Swift.

Розглянемо мову програмування Objective-C, як інструмент розробки мобільного додатку під платформу iOS.

Objective-C – це давня мова програмування, створена компанією Stepstone на початку 1980-х років. Вона була випущена для громадськості в 1988 році, коли Бред Кокс і Том Лав опублікували книгу «Об'єктно-орієнтоване програмування: еволюційний підхід». Наприкінці 1980-х років компанія Objective-C була ліцензована компанією NeXT Computer, Inc. для розробки рамок NeXTStep і в кінцевому підсумку була придбана Apple. Саме так він став стандартом протягом багатьох років у сфері розробки додатків iOS.

Objective-C був створений під впливом двох інших мов програмування: C і Smalltalk. Ось чому він має такий складний синтаксис. Він виводить свій синтаксис об'єкта з Smalltalk, тоді як синтаксис для не-об'єктно-орієнтованих операцій такий же, як у C. Objective-C використовує динамічне введення і передачу повідомлень. Вона також вимагає розділення класів на два блокових коду: інтерфейс і реалізацію.

Переваги Objective-C:

– вона існує багато років і добре випробувана. Є мільйони рядків коду, написаних на Objective-C. Є багато добре задокументованих, сторонніх бібліотек, і є відповідь на майже кожне питання;

– хороша сумісність з C і C++. Так кк Objective-C є надмножеством C, то вона працює з C або C++ кодом працює відносно гладко;

– це стабільно. Якщо ви розробляєте програму в Objective-C, вам, мабуть, не доведеться витрачати гроші на перенесення програми на нову мовну версію через кілька місяців.

Недоліки Objective-C:

– важко вчитися. Objective-C істотно відрізняється від багатьох інших популярних мов програмування, і управління пам'яттю досить складне. Саме тому легше навчитися Swift розробниками, знайомими з Objective-C, ніж навпаки;

– зменшення кількості розробників. Оскільки Objective-C важче вивчити, існує більше нових розробників, які вивчають Swift, ніж навчання Objective-C. З іншого боку, досвідчені розробники, знайомі з Objective-C, зазвичай знайомі з Swift або, принаймні, готові до його вивчення;

– додаток, розроблений в Objective-C, може бути потенційно легше зламати, ніж його Swift альтернатива. Оскільки Objective-C добре відома і існує багато років, це також означає, що інструменти зворотного проектування також добре розвинені.

І, нарешті, розглянемо останню в списку мову програмування, що дозволяє створювати додатки під платформу iOS – Swift. Swift – це сучасна мова програмування загального призначення та багатопартійної парадигми, розроблена компанією Apple для побудови своїх пристроїв, що працюють на iOS, і всієї наступної екосистеми. Програми можуть бути розроблені для роботи також на MacOS (для комп'ютерів Apple), watchOS (AppleWatch), tvOS (цифровий мультимедійний програвач Apple TV) і, що може бути трохи дивно, що z / OS, що живить комп'ютери IBM Mainframe. Наразі мова поширюється на ліцензію Apache, що робить її доступною для спільноти. Це відносно новий проект, запущений у червні 2014 року, через сім років після запуску iPhone. На відміну від Objective-C, в якому для кожного класу необхідно створювати файли *.h і *.m з інтерфейсом і реалізацією відповідно, в Swift потрібно створити лише один файл *.swift, в якому

містяться і інтерфейс, і реалізація. Це означає, що вихідних файлів в проекті буде в 2 рази менше, що є плюсом [27]. Та звісно у Swift є свої дуже вагомі переваги над конкурентами:

– читаність. Перевагою номер один у виборі Swift, ймовірно, є його чистий синтаксис, що полегшує читання та запис. Кількість рядків коду, необхідних для реалізації опції Swift, набагато менше, ніж для Objective-C. Причина цього полягає в тому, що Swift видаляє багато успадкованих умовностей, таких як крапки з комою до кінцевих рядків або дужок, які оточують умовні вирази всередині операторів if/else. Іншою важливою зміною є те, що виклики методів не сидять усередині один одного, в результаті чого виникає безлад в дужках. Замість цього, виклики методів і функцій у Swift використовують розділені комою список параметрів у дужках. В результаті, код є чистішим з спрощеним синтаксисом. Код Swift більш нагадує звичайну англійську мову, що робить написання коду більш природним, дозволяючи розробникам витрачати набагато менше часу на пошук проблемного коду. Ця читаність також полегшує існуючим програмістам з JavaScript, Java, Python, C# і C++ прийняття Swift в їхню інструментальну ланцюжок;

– технічне обслуговування. Неможливо, щоб Objective-C розвивався без того, щоб C розвивався першим. Натомість Swift не має цих залежностей, що значно полегшує їх обслуговування. C вимагає від програмістів підтримувати два файли коду, щоб поліпшити час і ефективність створення коду, який також переноситься на Objective-C. Однак Swift відкидає цю вимогу для двох файлів, поєднуючи заголовок Objective-C (.h) і файли реалізації (.m), в єдиний файл коду (.swift). У Objective-C, вам доведеться вручну синхронізувати імена методів і коментарі між файлами. У Swift програмісти можуть витрачати більше часу на створення логіки та покращення якості коду, коментарів та функцій, які підтримуються;

– безпечніша платформа. На конкурентному ринку мобільних додатків розробка безпечної програми має бути пріоритетом. Синтаксис і

мовні конструкції Свіфта виключають декілька можливих типів помилок в Objective-C. Ця стабільність означає, що буде менше аварій і випадків проблемної поведінки. Це не заважає програмістам писати поганий код, а навпаки, зменшує ймовірність помилок. Це додає додаткового рівня контролю якості під час розробки. Swift приймає нульовий код і генерує помилку компілятора, коли програмісти пишуть поганий код. За допомогою Swift ви можете компілювати та виправляти помилки під час написання коду, що неможливо з Objective-C. Як результат, Swift працює краще і швидше в порівнянні з Objective-C, коли мова йде про тестування помилок. Все це дає підстави вважати Swift безпечною і безпечною мовою програмування;

– менше коду та менше спадщини. За допомогою Objective-C існує багато проблем, які викликають збій програм. Swift надає код, який є менш схильним до помилок через його вбудовану підтримку для маніпулювання текстовими рядками та даними. Крім того, класи не розділені на дві частини; інтерфейс і реалізація. Це скорочує кількість файлів у проекті в два рази, що робить його набагато простішим у використанні. Swift в кінцевому рахунку вимагає менших зусиль кодування при написанні повторюваних операцій або викликаючи маніпуляцію рядками. Під час роботи з Objective-C вам потрібно буде об'єднати два рядки, які роблять його довгим. За допомогою Swift потрібно лише додати знак «+», щоб об'єднати два рядки;

– швидкість. Swift також забезпечує різні швидкісні переваги при розробці, в свою чергу, економить на витратах. Наприклад, складний тип об'єкта буде працювати на 3,9 разів швидше, ніж реалізація того ж самого алгоритму в Python. Це також краще, ніж Objective-C, що на 2,8 рази швидше, ніж версія Python. Його продуктивність наближається до C++, що вважається найшвидшою арифметикою алгоритму розрахунку. У грудні 2014 року лабораторія приматів опублікувала звіт про продуктивність Swift та C++. Apple зрозуміла, що вони присвячені підвищенню швидкості, з якою Swift може запускати логіку додатків;

– Swift підтримує динамічні бібліотеки. Динамічні бібліотеки – це виконувані фрагменти коду, які можна зв'язати з додатком. Ця функція дозволяє поточним додаткам Swift зв'язуватися з новими версіями мови Swift, оскільки вона розвивається з плином часу. Динамічні бібліотеки в Swift безпосередньо завантажуються в пам'ять, зменшуючи початковий розмір програми і в кінцевому рахунку збільшуючи продуктивність програми;

– Playgrounds заохочує інтерактивне кодування. Playgrounds – це функція, яка дозволяє програмістам перевіряти новий алгоритм без необхідності створювати цілі програми. Компанія Apple додала виконання Inline коду до Playgrounds, щоб допомогти програмістам створити фрагмент коду або написати алгоритм, отримуючи зворотний зв'язок. Цей контур зворотного зв'язку може поліпшити швидкість, з якою код може бути написаний за допомогою візуалізації даних. Ігрові майданчики та Swift разом вказують на спроби Apple зробити розробку програм простішим та доступнішим;

– відкрите джерело. У 2015 році Swift було оголошено відкритим вихідним кодом, що відкриває мову до потенціалу, який буде використовуватися в різних платформах і для інфраструктури бекенда. Open-Sourcing Swift означає, що Apple зможе отримати зворотний зв'язок від спільноти, щоб зробити послідовне вдосконалення, оскільки незалежні розробники сприяють успіху мови. Swift не тільки успішно злетів, тому що він добре структурований і розроблений, а також тому, що його підтримали багато розробників [28].

Та що стосується недоліків, то можна сказити, що їх майже не існує. Можна виділити такі невагомі недоліки:

– мова ще досить молода. Swift може бути найшвидшою і найпотужнішою мовою в світі, але ще занадто молодою. Вона має багато питань, які необхідно вирішити, і «відчувати біль». Зрештою, три роки – це занадто мало часу для того, щоб будь-яка мова дозріла, навіть якщо вона є Swift. Більше того, Swift все ще має дуже обмежену кількість «рідних»

бібліотек і інструментів: багато доступних ресурсів і інструментів, присвячених попереднім версіям Swift, марні з новими релізами;

– відсутність підтримки попередніх версій iOS. Можна використовувати лише Swift у програмах, націлених на iOS7 та пізніші. При цьому Swift не може використовуватися для застарілих проектів, що працюють на старих версіях операційної системи. Проте, за недавніми дослідженнями, менше 5% пристроїв Apple в даний час працюють на iOS6 або раніше. [29].

Таким чином, у більшості випадків розробка програми мовою Swift є кращим вибором. Проте є лише два випадки, коли ви повинні вибрати Objective-C:

– додаток вже написано в Objective-C, і вам потрібно лише додати нові функції. Swift є сумісним з Objective-C, що означає, що ви можете використовувати ці дві мови в одному проекті успішно. Однак це може бути не найкращим варіантом. Підтримка програми, розробленої двома мовами, може викликати проблеми. Ви повинні знати, що можуть виникнути деякі крайні випадки. Вам потрібен розробник, який володіє обома мовами. Крім того, перемикання контекстів робить роботу розробника повільнішою, оскільки їм необхідно щодня перемикатися між мовами програмування.

– необхідно широко використовувати рамки сторонніх розробників C або C++. Така ситуація зустрічається порівняно рідко. У такому випадку, оскільки Objective-C є надмножкою C, варто подумати про використання цієї мови.

Отже, коли йдеться про запуск нового iOS-проекту, рішення про те, яка мова програмування використовувати, має вирішальне значення. Це вплине на інші варіанти, такі як архітектура або використані фреймворки. Розвиток у Swift є більш швидким, безпечним і більш приємним з точки зору розробника. Саме тому в якості основного інструменту розробки була вибрана мова програмування Swift.

3.5 Демонстрація роботи системи

Для навчання нейронної мережі було використано набір даних надано Клівлендською клінічною фундацією для серцевих захворювань (Cleveland Clinic Foundation for Heart Disease) [30]. Це файл CSV із 303 рядками. Кожен рядок містить інформацію про пацієнта (зразок), а кожен стовпець описує атрибут пацієнта (ознаку). Використаємо ознаки для прогнозування наявності у пацієнта серцевих захворювань (бінарна класифікація).

Опис кожної з ознак представлений в таблиці 3.1.

Таблиця 3.1 – Опис ознак для обраного датасету.

Ознака	Пояснення	Тип ознаки
Age	Вік у роках	Числовий
Sex	Стать (1 – ч, 0 – ж)	Категорія
CP	Тип болю в грудях (0, 1, 2, 3, 4)	Категорія
Trestbpd	Артеріальний тиск у спокої (у мм рт. ст.)	Числовий
Chol	Холестерин в сироватці крові в мг / дл	Числовий
FBS	рівень цукру в крові вище 120 мг / дл (1 = істинно; 0 = хибно)	Категорія
RestECG	Результати електрокардіограми в спокої (0, 1, 2)	Категорія
Thalach	Максимальна частота серцевих скорочень	Числовий
Exang	Ангіна, спричинена фізичними вправами (1 = так; 0 = ні)	Категорія
Oldpeak	Депресія ST, спричинена фізичними вправами щодо відпочинку	Числовий
Slope	Нахил пікового сегмента вправи ST	Числовий
CA	Кількість основних судин (0-3), пофарбованих флюороскопією	Числовий, категорія
Thal	3 = нормальний; 6 = виправлений дефект; 7 = оборотний дефект	Категорія
Target	Діагностовано серцеві захворювання (1 = правда; 0 = хибно)	Target

В таблиці 3.2 можно переглянути як представлені дані в CSV файлі.

Таблиця 3.2 – Представлення даних в CSV файлі.

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	1	145	233	1	2	150	0	2.3	3	0	fixed	0
1	67	1	4	160	286	0	2	108	1	1.5	2	3	normal	1
2	67	1	4	120	229	0	2	129	1	2.6	2	2	reversible	0
3	37	1	3	130	250	0	0	187	0	3.5	3	0	normal	0
4	41	0	2	130	204	0	2	172	0	1.4	1	0	normal	0

Останній стовпець «target» вказує, чи є у пацієнта серцеве захворювання (1) чи ні (0).

Для навчання нейронної мережі набір даних було розділено на навчальну (training set) та перевірочну (validation set) вибірки у класичному співвідношенні 80:20. Таким чином до навчальної вибірки потрапило 242 зразки, а до перевірочної – 61 зразок.

Далі були розроблені функції, що нормалізують дані з нашого набору даних (рисуноки 3.5 – 3.7). Виходячи з таблиці 3.1, видно, що таких функцій знадобилось 3, відповідно до кількості типів ознак, що присутні у датасеті.

```
def encode_numerical(feature, name, dataset):
    norm_layer = Normalization()

    ds = dataset.map(lambda x, y: x[name])
    ds = ds.map(lambda x: tf.expand_dims(x, -1))

    norm_layer.adapt(ds)

    new_feature = norm_layer(feature)
    return new_feature
```

Рисунок 3.5 – Функція нормалізації для числових ознак

```
def encode_string_categorical(feature, name, dataset):
    index = StringLookup()

    ds = dataset.map(lambda x, y: x[name])
    ds = ds.map(lambda x: tf.expand_dims(x, -1)

    index.adapt(ds)

    new_feature = index(feature)

    encoder = CategoryEncoding(output_mode="binary")

    ds = ds.map(index)

    encoder.adapt(ds)

    new_feature = encoder(new_feature)
    return new_feature
```

Рисунок 3.6 – Функція нормалізації для ознак типу категорія, які записані у вигляді рядку

```
def encode_integer_categorical(feature, name, dataset):
    encoder = CategoryEncoding(output_mode="binary")

    ds = dataset.map(lambda x, y: x[name])
    ds = ds.map(lambda x: tf.expand_dims(x, -1))

    encoder.adapt(ds)

    new_feature = encoder(feature)
    return new_feature
```

Рисунок 3.7 – Функція нормалізації для ознак типу категорія, які записані у вигляді числа

Для експерименту топологія нейронної мережі змінювалася, шляхом зміни кількості прихованих шарів (1, 3, 5 шарів).

В якості функції помилки використана бінарна крос-ентропія, яка гарно підходить для задачі бінарної класифікації.

В якості метрики при навчанні мережі використовується точність (accuracy) – доля правильних відповідей мережі.

Для вибору оптимального оптимізатора було проведено порівняння точності роботи мережі на тестових даних при використанні трьох найпопулярніших оптимізаторів – SGD, Rmsprop та Adam. Результати експерименту відображено у таблиці 3.3. Видно, що найвищу точність мережа показала при використанні оптимізатора Rmsprop.

Таблиця 3.3 – Точність і помилка мережі на тестових даних в залежності від використаного оптимізатора

Оптимізатор	Точність			Помилка		
	1 шар	3 шари	5 шарів	1 шар	3 шари	5 шарів
SGD	80.06%	68.90%	69.86%	48.62%	55.74%	60.44%
Rmsprop	85.43%	89.71%	92.99%	30.81%	23.63%	19.45%
Adam	82.40%	90.56%	90.65%	35.50%	24.04%	20.64%

Розмір міні-вибірок обрано рівним 32, тобто мережа змінює вагові коефіцієнти після обробки кожних 32 об’єктів.

Дослідним шляхом було визначено оптимальну кількість епох навчання. Було проведено навчання мережі при різній кількості епох – 15, 20, 25, 30, 35. Результати експерименту відображено у таблиці 3.4. Таким чином оптимальна кількість епох навчання склала 30, так як при меншій або більшій кількості епох навчання мережа показувала нижчу точність на тестовій вибірці. При малій кількості епох це зумовлено тим, що мережа не встигла виділити достатню кількість ознак на навчальних даних, а при більшій кількості епох – через початкові ознаки перенавчання.

Таблиця 3.4 – Точність мережі на тестових даних в залежності від кількості епох навчання

Кількість епох навчання	15	20	25	30	35
Точність на тестовій вибірці	92.99%	90.93%	92.96%	97.89%	94.50%

Далі навчену нейронну мережу було збережено у форматі «*.mlmodel», щоб її можливо було використовувати при розробці мобільного додатку.

На рис 3.8 – 3.11 продемонстровані робочі екрани розробленого мобільного додатку.

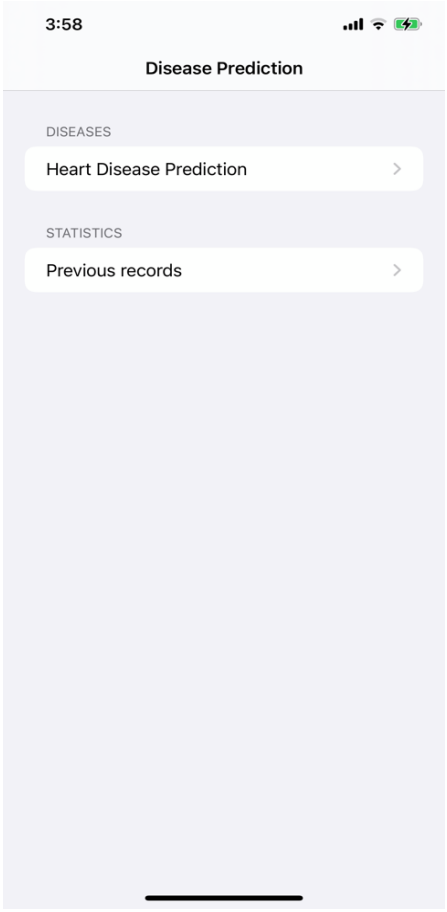

Рисунок 3.8 – Головне вікно мобільного додатку

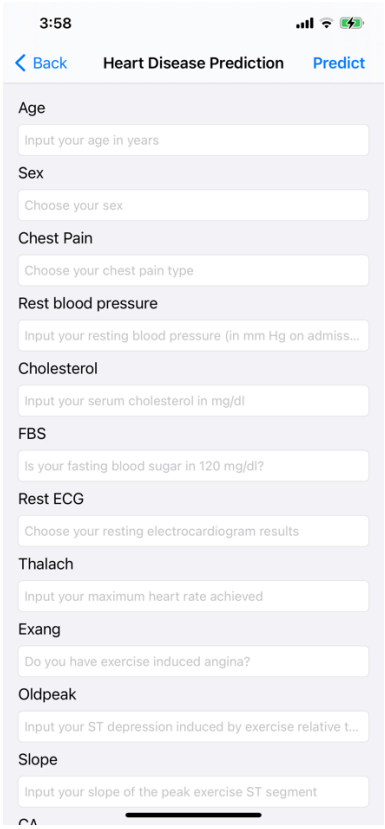

Рисунок 3.9 – Екран вводу даних для подальшої роботи системи

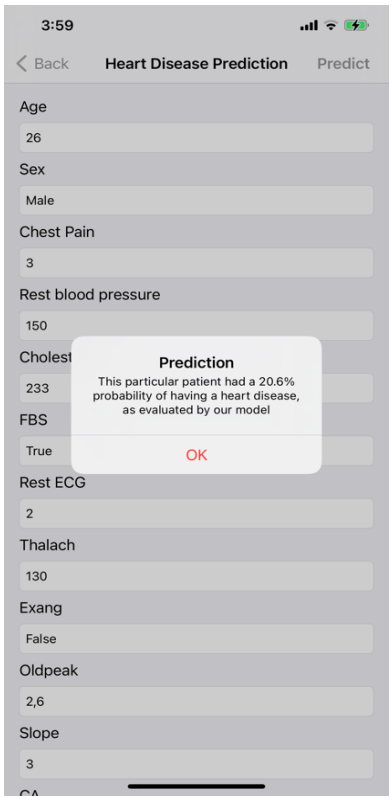

Рисунок 3.10 – Екран виводу результату

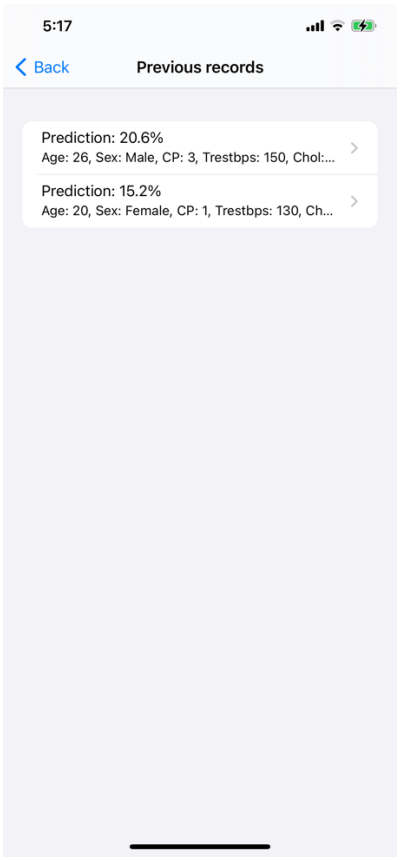

Рисунок 3.11 – Екран перегляду попередніх результатів

На даному екрані користувач може переглянути у вигляді звіту раніше зроблені прогнози щодо наявності захворювання.

3.6 Аналіз результатів

Готова підсистема була реалізована за допомогою мови програмування Python і біблотек машинного навчання TensorFlow та Keras. Вона дозволяє ввести дані про користувача, що будуть класифіковане нейронною мережею і з певною ймовірністю зробить висновок чи можливе захворювання серця у користувача чи ні.

В кінці навчання мережі точність на даних, на яких проводилося навчання, склала 97.89%, а на перевірочній вибірці (validation data) – 81.97%. Значення функції втрат (loss) відповідно склали 9.81% на

навчальній вибірці 55.24% на перевірочній вибірці (validation loss) (рисунок 3.12)

```
7/7 [==============================] - 0s 9ms/step - loss: 0.5023 - accuracy: 0.6945 - val_loss: 0.4969 - val_accuracy: 0.7802
Epoch 8/30
7/7 [==============================] - 0s 9ms/step - loss: 0.4889 - accuracy: 0.7360 - val_loss: 0.4748 - val_accuracy: 0.7802
Epoch 9/30
7/7 [==============================] - 0s 10ms/step - loss: 0.5173 - accuracy: 0.7129 - val_loss: 0.4615 - val_accuracy: 0.7802
Epoch 10/30
7/7 [==============================] - 0s 9ms/step - loss: 0.4758 - accuracy: 0.7436 - val_loss: 0.4493 - val_accuracy: 0.7802
Epoch 11/30
7/7 [==============================] - 0s 8ms/step - loss: 0.4656 - accuracy: 0.7740 - val_loss: 0.4390 - val_accuracy: 0.7912
Epoch 12/30
7/7 [==============================] - 0s 10ms/step - loss: 0.4398 - accuracy: 0.8096 - val_loss: 0.4290 - val_accuracy: 0.7912
Epoch 13/30
7/7 [==============================] - 0s 9ms/step - loss: 0.4231 - accuracy: 0.7728 - val_loss: 0.4196 - val_accuracy: 0.7802
Epoch 14/30
7/7 [==============================] - 0s 10ms/step - loss: 0.3922 - accuracy: 0.8248 - val_loss: 0.4109 - val_accuracy: 0.7912
Epoch 15/30
7/7 [==============================] - 0s 10ms/step - loss: 0.4348 - accuracy: 0.7879 - val_loss: 0.4039 - val_accuracy: 0.7802
Epoch 16/30
7/7 [==============================] - 0s 9ms/step - loss: 0.4405 - accuracy: 0.7950 - val_loss: 0.3962 - val_accuracy: 0.8022
Epoch 17/30
7/7 [==============================] - 0s 9ms/step - loss: 0.3801 - accuracy: 0.8293 - val_loss: 0.3893 - val_accuracy: 0.8022
Epoch 18/30
7/7 [==============================] - 0s 9ms/step - loss: 0.3899 - accuracy: 0.8171 - val_loss: 0.3834 - val_accuracy: 0.8132
Epoch 19/30
7/7 [==============================] - 0s 9ms/step - loss: 0.4232 - accuracy: 0.8036 - val_loss: 0.3782 - val_accuracy: 0.8242
Epoch 20/30
7/7 [==============================] - 0s 9ms/step - loss: 0.3697 - accuracy: 0.8293 - val_loss: 0.3737 - val_accuracy: 0.8352
Epoch 21/30
7/7 [==============================] - 0s 10ms/step - loss: 0.3770 - accuracy: 0.8278 - val_loss: 0.3698 - val_accuracy: 0.8242
Epoch 22/30
7/7 [==============================] - 0s 10ms/step - loss: 0.3551 - accuracy: 0.8304 - val_loss: 0.3665 - val_accuracy: 0.8352
Epoch 23/30
7/7 [==============================] - 0s 10ms/step - loss: 0.3252 - accuracy: 0.8506 - val_loss: 0.3638 - val_accuracy: 0.8352
Epoch 24/30
7/7 [==============================] - 0s 9ms/step - loss: 0.2856 - accuracy: 0.9135 - val_loss: 0.3614 - val_accuracy: 0.8352
Epoch 25/30
7/7 [==============================] - 0s 10ms/step - loss: 0.2846 - accuracy: 0.8994 - val_loss: 0.3586 - val_accuracy: 0.8242
Epoch 26/30
7/7 [==============================] - 0s 9ms/step - loss: 0.2804 - accuracy: 0.9044 - val_loss: 0.3570 - val_accuracy: 0.8352
Epoch 27/30
7/7 [==============================] - 0s 6ms/step - loss: 0.2958 - accuracy: 0.8572 - val_loss: 0.3560 - val_accuracy: 0.8352
Epoch 28/30
7/7 [==============================] - 0s 6ms/step - loss: 0.2953 - accuracy: 0.8732 - val_loss: 0.3548 - val_accuracy: 0.8352
Epoch 29/30
7/7 [==============================] - 0s 6ms/step - loss: 0.2633 - accuracy: 0.8964 - val_loss: 0.3544 - val_accuracy: 0.8352
Epoch 30/30
7/7 [==============================] - 0s 6ms/step - loss: 0.3023 - accuracy: 0.8677 - val_loss: 0.3534 - val_accuracy: 0.8352
```

Рисунок 3.12 – Вивід бібліотеки Keras

Було побудовано графіки зміни помилки (рисунок 3.13) і точності мережі в процесі навчання (рисунок 3.14). На даних графіках видно, що під час навчання мережі на кожній епосі помилка на наборі даних для навчання і наборі даних для перевірки зменшується, а якість навчання збільшується, тобто перенавчання не виникло. Якщо спочатку, на перших епохах навчання помилка знижується як на даних для навчання, так і на перевірочних даних, то ближче до кінця навчання значення помилки зменшується лише на даних для навчання, а на наборі даних для перевірки значення помилки майже не змінюється. Також на наборі даних для перевірки майже не збільшується якість навчання. Це свідчить про те, що якщо продовжити навчати мережу, її узагальнююча здатність знизиться – вона буде гірше розпізнавати нові зображення, які не бачила в процесі навчання. Тому навчання мережі було припинене, щоб уникнути перенавчання.

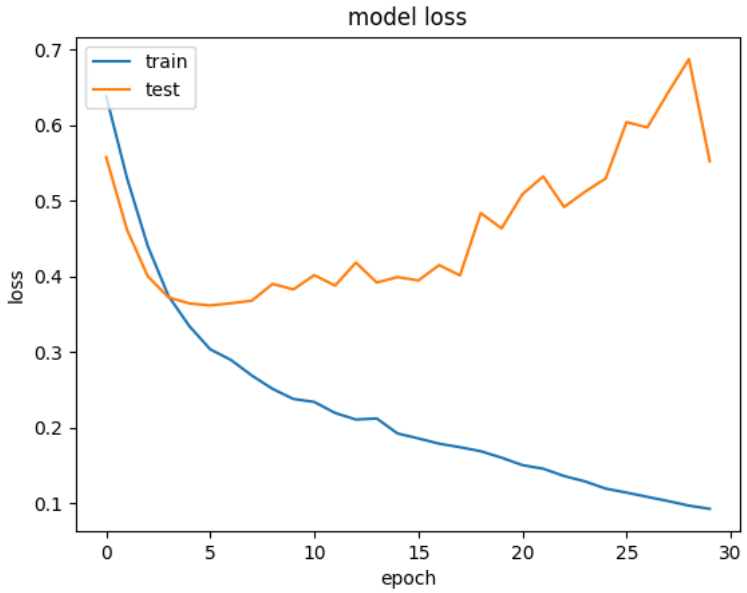

Рисунок 3.13 – Графік зміни помилки при навчанні нейронної мережі

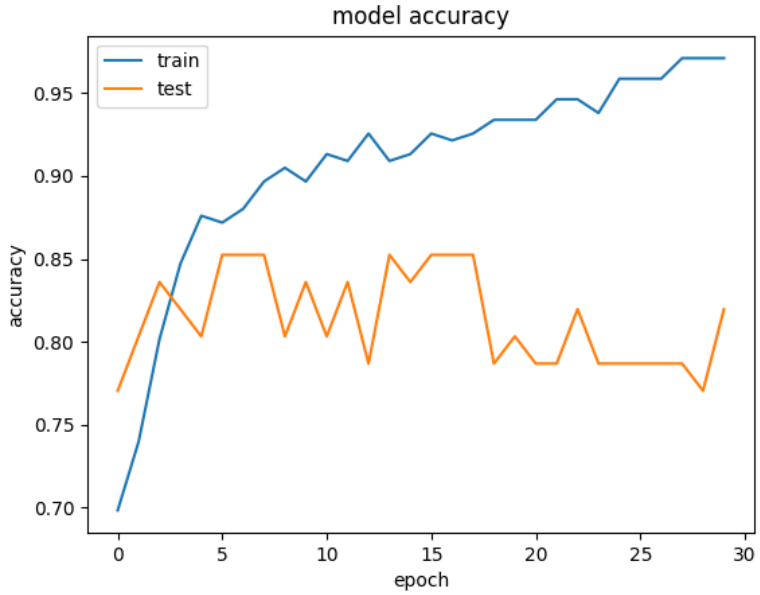

Рисунок 3.14 – Графік зміни точності при навчанні нейронної мережі

Видно, що графіки мають вигляд ламаних прямих. Це зумовлено тим, що кількість даних у оброблюємій вибірці обмежений. Для підвищення точності системи можна збільшити навчальну вибірку.

ВИСНОВКИ

В даній кваліфікаційній роботі було показано, як використовуються нейронні мережі для обробки та аналізу медичних даних, а саме для прогнозу виникнення захворювань серцево-судинної системи у людини.

Було проведено детальний аналіз предметної області та огляд існуючих аналогів використання нейронних мереж в медицині.

виявлені переваги та недоліки кожного з них. Після чого було встановлено, що потрібно реалізувати в системі моніторингу параметрів самопочуття користувача, усунувши ці недоліки, та зберігши переваги кожного із розглянутих додатків.

Перед розробкою системи було визначено функціональні вимоги до неї та обрано технології, за допомогою яких вона була реалізована.

Були розроблені нейронна мережа для прогнозу виявлення серцево-судинних захворювань та мобільний додаток в якості графічного інтерфейсу та надання розширених можливостей використання нейронної мережі. За допомогою даної системи, користувачі можуть зручно взаємодіяти з нейронною мережею, переглядати попередні зроблені записи та інше.

Розроблений програмний засіб буде мати цінність та буде використовуватися серед працівників медичних структур, бо дозволить їм скоротити час за який вони зможуть встановити та виявити ті чи інші захворювання.

База даних додатку була реалізована за допомогою СУБД Realm для мобільних пристроїв. Нейронна мереже розроблена на мові програмування Python за допомогою бібліотек Pandas, TensorFlow, Keras та Numpy. Мобільний додаток написано на мові програмування Swift у середовищі розробки XCode, для пристроїв з операційною системою iOS.

Розроблений додаток відповідає сформульованим вимогам завдання, таким чином, завдання на кваліфікаційну роботу виконано в повному обсязі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Ф. Розенблатт. Принципи нейродинаміки. Перцептрони і теорія механізмів мозку. Москва: Мир, 1965. 478 с.

2. Minsky M., Papert A. Perceptrons – Expanded Edition. Cambridge, MA: The MIT Press, 1988. 308 p.

3. Aggarwal C. Neural Networks and Deep Learning. London: Springer International Publishing, 2018. 405 p.

4. Agostinelli F., Hoffman F., Sadowski P. Learning Activation Functions to Improve Deep Neural Networks. Workshop paper contribution at the International Conference on Learning Representations (ICLR). 2015. P. 555-680.

5. Хайкін С. Нейронні мережі: повний курс. Київ: Діалектика Вільямс, 2018. 2-е вид. 1104 с.

6. Volume of data/information created, captured, copied, and consumed worldwide from 2010 to 2024. URL: https://www.statista.com/statistics/871513/worldwide-data-created (дата звернення: 17.04.2021)

7. Сайт програмного засобу «Цельс». URL: https://celsus.ai/ (дата звернення: 17.04.2021).

8. New DNA tool predicts height, shows promise for serious illness assessment URL: https://msutoday.msu.edu/news/2018/new-dna-tool-predicts-height-shows-promise-for-serious-illness-assessment (дата звернення: 17.04.2021)

9. Сайт програмного засобу «Exscientia» URL: https://www.exscientia.ai/ (дата звернення: 17.04.2021)

10. Сайт програмного засобу «Voice2Med» URL: https://www.speechpro.ru/product/programmy-dlya-raspoznavaniya-rechi-v-tekst/voice2med (дата звернення: 17.04.2021)

11. Ніколенко С. І., Кадурін А. А., Архангельска Е. О. Глибоке навчання. Занурення у світ нейронних мереж. Пітер, 2016. 480 с.

12. S. Kajan. GUI for classification using multilayer perceptron network, Technical Computing Prague, 2009

13. Agostinelli F., Hoffman F., Sadowski P. Learning Activation Functions to Improve Deep Neural Networks. Workshop paper contribution at the International Conference on Learning Representations (ICLR). 2015. P. 555-680.

14. Бодянский Е. В. Искусственные нейронные сети: архитектуры, обучение, применения. Харьков ТЕЛЕТЕХ, 2004. 372 с.

15. M. Negnevitsky, Artificial Intelligence. Pearson Education Limited, 2005

16. Визначення NumPy. URL: https://en.wikipedia.org/wiki/NumPy (дата звернення: 17.04.2021).

17. Визначення SciPy. URL: https://en.wikipedia.org/wiki/SciPy (дата звернення: 17.04.2021).

18. Визначення scikit-learn. URL: https://en.wikipedia.org/wiki/scikit-learn (дата звернення: 17.04.2021).

19. Визначення Matplotlib. URL: https://en.wikipedia.org/wiki/Matplotlib (дата звернення: 17.04.2021).

20. Визначення Pandas. URL: https://en.wikipedia.org/wiki/Pandas_%28software%29 (дата звернення: 17.04.2021).

21. Визначення TensorFlow. URL: https://en.wikipedia.org/wiki/TensorFlow (дата звернення: 17.04.2021).

22. Визначення Keras. URL: https://en.wikipedia.org/wiki/Keras (дата звернення: 17.04.2021).

23. Apple. Human Interface Guidelines. URL: https://developer.apple.com/design/human-interface-guidelines/ios/overview/themes (дата звернення: 17.04.2021)

24. Yarmolchuk, Habr. Создание приложения ToDo с помощью Realm и Swift. URL: https://habr.com/ru/post/272393 (дата звернення: 17.04.2021)

25. Karpov K. Логическое проектирование базы данных. URL: http://karpov-k.me/computernaya-nauka/bd/353-logicheskoe-proektirovanie-bd (дата звернення: 17.04.2021)

26. Borabai. Физическое проектирование базы данных. URL: https://bourabai.ru/dbt/dbms/03.htm (дата звернення: 17.04.2021)

27. Dev_ninja, Habr. Swift: проблемы и перспективы. URL: https://habr.com/ru/post/227243 (дата звернення: 17.04.2021)

28. Kinjal Dua, ClearBridge. 8 Advantages of Using Swift for iOS Development. URL: https://clearbridgemobile.com/8-advantages-choosing-swift-objective-c-ios (дата звернення: 17.04.2021)

29. Altexsoft. The Good and the Bad of Swift Programming Language. URL: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-swift-programming-language (дата звернення: 17.04.2021)

30. Kaggel Heart Disease Dataset. URL: https://www.kaggle.com/ronitf/heart-disease-uci (дата звернення: 17.04.2021)

31. ДСТУ 3008:2015. Документація. Звіти у сфері науки і техніки. Структура та правила оформлюванню. – К.: ДП «УкрНДНЦ», 2016. – 31 с.

32. Загальні методичні вказівки з дипломного проектування в університеті / Упоряд.: Ковтун П.С., Дудар З.В., Журавльов В.Я., Шкіль О.С. – Харків: ХНУРЕ, 2003. – 40 с.

33. Методичні вказівки щодо структури, змісту та оформлення навчально–методичної літератури для авторів (упорядників) оригіналів / Упорядн.: П.С. Ковтун, І.О. Мілютченко, Б.П. Косіковська. – Харків: ХНУРЕ, 2002. – 60 с.

34. Паттер проектування MVC. URL: https://tproger.ru/articles/mvc/.

35. Віктор Гольцман. MySQL 5.0. Библиотека программиста. – 1-е вид. – П.: Питер, 2010. – 174 с.

36. Realm Database. URL: https://realm.io/products/realm-database (дата звернення: 17.04.2021)

37. Xcode // Developer Apple. URL: https://developer.apple.com/xcode (дата звернення: 17.04.2021)