Вінницький національний технічний університет
Факультет інтелектуальних інформаційних систем та автоматизації
Кафедра комп'ютерних систем управління

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Нейромережевий ансамбль для розпізнавання зображень як елемент людино-машинної взаємодії»

Виконав: студент 2 курсу, групи 2АКІТ-21м
спеціальності 151 – Автоматизація та комп'ютерно-інтегровані технології

___Антон ШАМАТІЄНКО___
Ім'я ПРІЗВИЩЕ

Керівник:
__к.т.н., доцент кафедри КСУ_____
степінь, звання, посада

__Олена НИКИТЕНКО__
Ім'я ПРІЗВИЩЕ

«__12__»_____грудня_____2022 р.

Опонент:
_____доцент кафедри АІІТ_____
степінь, звання, посада

__Юрій ІВАНОВ___
Ім'я ПРІЗВИЩЕ

«__17___»_____грудня____2022 р.

Допущено до захисту
Зав. кафедри КСУ
В'ячеслав КОВТУН
«14 »_грудня_2022

Вінниця ВНТУ – 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 15 – Автоматизація та приладобудування
Спеціальність – 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо - професійна програма – Інтелектуальні комп'ютерні системи

ЗАТВЕРДЖУЮ
Завідувач кафедри КСУ
В'ячеслав КОВТУН
"03" жовтня 2022 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Шаматієнко Антону Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Нейромережевий ансамбль для розпізнавання зображень як елемент людино-машинної взаємодії
керівник роботи к.т.н., доцент кафедри КСУ Никитенко Олена Дмитрівна
затверджені наказом ВНТУ від "14" вересня 2022 року №203
2. Термін подання студентом роботи "12" грудня 2022 року
3. Вихідні дані до роботи:
науково-технічні публікації, інтернет-ресурси по темі, наукові конференції, публікації в наукових журналах, збірники конференцій, документація мови програмування Python
4. Зміст текстової частини: провести огляд комбінованих нейронних мереж як засобу підвищення ефективності розпізнавання об'єктів; провести огляд і аналіз структурного проектування нейромережевого ансамблю; провести огляд і аналіз алгоритмів навчання нейромереж; розробити і протестувати програмне забезпечення
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
рисунків – 33шт., таблиць – 6шт., презентація 16 слайдів

1. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв
1-3	Ковтун В. В., д.т.н., доцент, зав. кафедри КСУ		
4	Небава М.І., пофесор кафедри ЕПВМ		

2. Дата видачі завдання "03" жовтня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Затвердження теми	14.09.22	14.09.22	
2	Аналіз предметної області	15.09.22	25.09.22	
3	Постановка задачі	20.09.22	25.09.22	
4	Проектування системи	25.09.22	09.10.22	
5	Розробка програмного забезпечення	10.10.22	15.11.22	
6	Підготовка економічної частини	15.11.22	20.11.22	
7	Підготовка пояснювальної записки	20.11.22	30.11.22	
8	Отримання відгуку опонента	12.12.22	18.12.22	
9	Отримання відгуку кервіника	12.12.22	18.12.22	
10	Попередній захист магістерської кваліфікаційної роботи	14.12.22	14.12.22	
11	Захист магістерської кваліфікаційної роботи	20.12.22	20.12.22	

Студент (підпис) Антон ШАМАТІЄНКО (*Ім'я ПРІЗВИЩЕ*)

Керівник роботи (підпис) Олена НИКИТЕНКО (*Ім'я ПРІЗВИЩЕ*)

АНОТАЦІЯ

УДК. 004.514

Шаматієнко А.В. Нейромережевий ансамбль для розпізнавання зображень як елемент людино-машинної взаємодії. Магістерська кваліфікаційна робота зі спеціальності 151 – Автоматизація та комп’ютерно-інтегровані технології, освітня програма – Інтелектуальні комп’ютерні системи. Вінниця: ВНТУ, 2022. 114ст.

На укр. мові. Бібліогр.: 23 назв; рис.: 33; табл. 6.

У магістерській кваліфікаційній роботі розроблено програмне забезпечення, яке використовує створений асамбль нейронних мереж-класифікаторів для ідентифікації пневмонії на рентгенограмах.

В першому розділі було детально досліджено існуючу науково-технічну літературу по темі нейронні мережі та ансамблі нейронних мереж. В другому розділі за комплексним критерієм визначено найкращі топології мереж, які входять в ансамбль для вирішення задачі розпізнавання, в третьому розділі описано створення ансамблю нейронних мереж із застосуванням різноманітних методів навчання ансамблів нейромереж. У економічній частині проаналізований технічний рівень і розрахована собівартість реалізації розробки.

Ілюстративна частина складається з 16 зображень із результатами роботи.

Ключові слова: нейронні мережі, ансамбль, навчання нейронних мереж, класифікація, розпізнавання.

ABSTRACT

Shamatienko A.V. Neural network ensemble for image recognition as an element of human-machine interaction. Master's thesis on specialty 151 - Automation and computer-integrated technologies, educational program - Intelligent computer systems. Vinnytsia: VNTU, 2022. 114p.

In English speech Bibliography: 23; pic.: 33; table 6.

In the master's thesis was developed software that uses the created assembly of neural networks-classifiers for the identification of pneumonia on radiographs.

In the first chapter, the existing scientific and technical literature on the topic of neural networks and ensembles of neural networks was studied in detail. In the second section, the best topologies of networks included in the ensemble for solving the recognition problem are determined based on a complex criterion, in the third section, the creation of an ensemble of neural networks using various methods of learning ensembles of neural networks is described. In the economic part, the technical level is analyzed and the cost of development implementation is calculated.

The illustrative part consists of 16 images with work results.

Keywords: neural networks, ensemble, neural network training, classification, recognition.

ВІДГУК
керівника магістерської кваліфікаційної роботи

студента _____Шаматієнко Антон Віталійович_____
(прізвище, ім'я, по батькові)
на тему: Нейромережевий ансамбль для розпізнавання зображень як елемент людино-машинної взаємодії_____

Актуальність роботи в контексті спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» доведена результатами інформаційного пошуку та аналізу літературних джерел. Додатковим підтвердженням актуальності роботи слугують опубліковані тези на Науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації.

Представлені в магістерській кваліфікаційній роботі рішення обґрунтовані послідовним ланцюгом дій, які включають пошук інформації про проблему, її узагальнення, формулювання прикладних задач для її вирішення, проектування відповідного програмного засобу для вирішення поставлених задач, його тестування і формулювання висновків. Раціональність та ефективність прийнятих рішень доведені результатами тестування.

Дипломник показав хороший рівень спеціальних знань і «м'яких» навичок. Дипломник продемонстрував вміння: - вирішувати поставлені керівником завдання самостійно, згідно власноруч розробленої схеми заходів; - здійснювати пошук і узагальнення інформації; - комунікативні навички. Доведенням ерудиції та креативності дипломника є вчасно представлена магістерська кваліфікаційна робота.

Основні результати, представлені в роботі отримані дипломником самостійно. Матеріалу роботи властивий високий ступінь оригінальності, що доведено результатами перевірки на наявність запозичень.

Дипломник працював ритмічно, без суттєвих відхилень від затвердженого графіку. Втрати зв'язку з керівником не було.

Недоліки: Автор не акцентує увагу на тім, які саме з отриманих результатів опубліковані у вигляді тез. Бажано було аналізувати свіжіші літературні джерела. Не всі рисунки достатньо якісні. Автор навів лише необхідні і достатні діаграми, що описують процес проектування створеної системи. Результати багатоваріантного аналізу бажано було звести в таблицю.

Загалом магістерська кваліфікаційна робота ***відповідає*** спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», заслуговує на оцінку _____A_____, а її автор ***заслуговує*** присудження кваліфікації: ступінь вищої освіти магістр, спеціальність «Автоматизація та комп'ютерно-інтегровані технології», освітня програма «Інтелектуальні комп'ютерні системи».

Керівник магістерської кваліфікаційної роботи

к.т.н., доцент, доцент кафедри КСУ _____ Олена НИКИТЕНКО
(посада, науковий ступінь, вчене звання) (підпис) (*Ім'я ПРІЗВИЩЕ*)

ВІДГУК
опонента на магістерську кваліфікаційну роботу

студента Шаматієнко Антон Віталійович
(прізвище, ім'я, по батькові)
на тему: Нейромережевий ансамбль для розпізнавання зображень як елемент людино-машинної взаємодії

Актуальність роботи в контексті спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології» доведена результатами інформаційного пошуку та аналізу літературних джерел. Додатковим підтвердженням актуальності роботи слугують опубліковані тези на Науково-технічній конференції факультету інтелектуальних інформаційних технологій та автоматизації.

Перший розділ магістерської кваліфікаційної роботи повністю присвячений огляду літературних та інформаційних джерел за обраною темою. Проаналізовано не менш ніж три аналоги створюваної системи. Функціональність та форм-фактор створеної системи цілком визначені на основі результатів критичного огляду літератури.

Прийняті рішення обґрунтовані результатами огляду літератури, результатами проектування та втілені в функціонуючу програмну систему. Результати її тестування доводять правильність прийнятих рішень.

Експериментальні дослідження продумані і повні. Тести охоплюють як функції інтерфейсу створеної системи, так і доводять якість та повноту виконання нею функціонального призначення, обґрунтованого на етапі проектування.

Вміст графічної частина повною мірою репрезентує всі отримані в магістерській кваліфікаційній роботі результати. Якість рисунків в графічній частині прийнятна.

Недоліки: В роботі недостатньо обґрунтовано, чому для розпізнавання зображень були обрані саме нейромережі. Інтерфейс створеної системи нестандартний. Не зрозуміло, як впливають параметри зображень на загальну схему організації системи ансамблю.

Загалом магістерська кваліфікаційна робота ***відповідає*** спеціальності 151 «Автоматизація та комп'ютерно-інтегровані технології», заслуговує на оцінку A, а її автор ***заслуговує*** присудження кваліфікації: ступінь вищої освіти магістр, спеціальність «Автоматизація та комп'ютерно-інтегровані технології», освітня програма «Інтелектуальні комп'ютерні системи»

Опонент

доцент кафедри АІІТ _____ Юрій ІВАНОВ
(посада, науковий ступінь, вчене звання) (підпис) (*Ім'я ПРІЗВИЩЕ*)

Печатка установи,
організації опонента

Зміст

ВСТУП

Актуальність магістерської роботи залежить від того, що вирішення проблеми розпізнавання образів є важливою частиною процесу прийняття рішень. Вирішення цієї проблеми дозволить штучним системам більш точно і швидко аналізувати навколишній простір, розділяючи його на окремі логічні частини і миттєво реагуючи на будь-які зміни.

Сьогодні для вирішення цієї проблеми використовується велика кількість різноманітних методів, але традиційні аналітичні методи не дають очікуваних результатів для вирішення сучасних завдань розпізнавання, а методи з використанням штучного інтелекту не повністю придатні для вирішення конкретної задачі, оскільки вони не адаптовані. використовувати велику кількість параметрів для ідентифікації, а відсоток успіху досить високий. Менший відсоток результатів розпізнавання знижує ефективність використання цих ідентифікаторів.

Серед існуючих методів розпізнавання найбільш перспективним є метод нейронної мережі. Існує нагальна потреба у вдосконаленні моделей і алгоритмів розпізнавання, включаючи використання ансамблів нейронних мереж. Два параметри системи, що вирішує задачу розпізнавання, особливо важливі: швидкість і точність розпізнавання. Тому існує два напрямки досліджень: удосконалення, оптимізація та створення нових моделей, а також удосконалення існуючих алгоритмів для підвищення їх швидкості роботи шляхом застосування різних методик і архітектур процесорів.

Серед відомих проблем можна виділити задачі розпізнавання об'єктів на зображеннях, вирішення яких є важливою частиною систем штучного зору. Рішення починається зі створення візуальних навчальних зразків для штучних нейронних мереж. На сьогоднішній день не існує єдиного покрокового підходу до створення ансамблю, так званого навчального набору даних. Крім того, рідко досліджується вплив різних параметрів зразків зображень і попередньої обробки візуальних даних на якість навчання нейронних мереж.

Метою магістерської роботи є вдосконалення існуючих методів розпізнавання шляхом застосування концепції ансамблевих нейронних мереж.

Завдання магістерської роботи вирішується з використанням методів математичного моделювання штучної нейронної мережі та обробки зображень. Досягнення цілей магістерської роботи можна розділити на такі окремі завдання:

– вибрати топологію нейронної мережі для включення в набір нейронних мереж;

– вибір методів побудови та формування ансамблів нейронних мереж;

– створити колекцію з обраним способом побудови;

– розробити програмне забезпечення для класифікації зображень на основі створених наборів мереж.

Об’єктом дослідження є автоматичний метод розпізнавання зображень на основі нейронної мережі.

Предмет дослідження базується на теоретичному ознайомленні з існуючими програмними аналогами, застосуванні можливостей бібліотеки OpenCV та мови програмування Python для створення програмних модулів розпізнавання зображень на основі нейронних мереж.

Наукова новизна ефективної ідентифікації об’єктів базується на використанні ансамблів нейронних мереж, які, в свою чергу, покращують узагальнювальну здатність нейронних мереж, не сповільнюючи їх роботу.

Апробація і публікація результатів роботи. Результати роботи були представлені на всеукраїнській науково-практичній інтернет-конференції студентів аспірантів та молодих науковців «Молодь в науці: дослідження, проблеми, перспективи»

Шаматієнко А.В. Розробка інформаційної системи фітнес-центру [Електронний ресурс] / А.В. Шаматієнко // ВНТУ. – 2022. – Режим доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2021/paper/view/12584

1. КОМБІНОВАНІ НЕЙРОМЕРЕЖІ ЯК ЗАСІБ ПІДВИЩЕННЯ ЕФЕКТИВНОСТІ РОЗПІЗНАВАННЯ ОБ’ЄКТІВ

1.1 Передумови використання нейромереж

Сьогодні зростає інтерес до штучного інтелекту, включаючи нейронні мережі. Зазвичай використовується, коли потрібно вирішити такі завдання, як прогнозування, ідентифікація та класифікація, управління та прийняття рішень.

Вони отримують такий розповсюдження завдяки схожості з нашим мозком, великій кількості можливостей і перспективам ефективного застосування, оскільки, за словами вчених, наш мозок використовує лише 10% свого власного потенціалу. Нейронні мережі є дуже потужним методом моделювання, який дозволяє нам відтворювати складні залежності. Нейронні мережі є нелінійними і тому можуть мати справу з розмірністю, яка не дозволяє моделювати лінійні залежності з багатьма змінними. Нейронні мережі включають досвід, дуже схожий на наш мозок.[1]

Нейронні мережі з'явилися в результаті спроб відтворити здатність мозку до навчання і виправлення помилок. Для створення штучного інтелекту необхідно створити систему зі схожою архітектурою та принципами роботи.

Але на відміну від нашого мозку, штучні нейронні мережі не мають властивостей втоми, що збільшує час для отримання результату. Тому ми можемо виконувати дуже складні завдання, подібні до завдань нашого мозку. Наприклад, автономне водіння зараз широко використовується на магістралях або паркінгах, що є звичайним явищем. За допомогою комп’ютерного зору ці системи можуть автоматично миттєво виявляти перешкоди на дорозі, дотримуючись дистанції та правил дорожнього руху на певній ділянці дороги.[2]

Такі мережі також використовуються в таких областях:

- Економіка: прогнозуйте курси валют, ціни на товари на основі різних умов або факторів як параметрів.

– Медицина: використовується для діагностики та виявлення різних захворювань, включаючи аналіз медичних зображень тощо.

– Авіоніка: для безпілотних літальних апаратів їх зв’язок може бути перервано, а дрон захоплений.

– Роботи: швидко та точно ідентифікують об’єкти та перешкоди, визначають траєкторії руху та зберігають рівновагу перед початком роботи.

Сьогодні ми бачимо велику кількість прикладів застосування нейронних мереж у різних сферах, взявши за приклад FaceID, який використовують користувачі мобільних телефонів Apple. Все це можливо завдяки тому, що нейронні мережі забезпечують надзвичайно потужний і гнучкий набір інструментів для вирішення різноманітних завдань обробки та аналізу даних, що надаються як вхідні дані.[3]

1.2 Математичний опис штучного нейрона

З біологічної точки зору основними елементами нервової системи є нервові клітини, або нейрони. Нейрон має тіло клітини — сому, з якої виходять два відростки: він приймає і передає інформацію відповідно за допомогою відростка — дендрита й аксона. Кожен нейрон має вихідний процес, який передає імпульси до кількох інших нейронів.

Нейрон запускається великою кількістю нейронів. Вважається, що мозок людини складається приблизно з 10 мільярдів нейронів. Кожен нейрон передає збудження іншим нейронам через нейронні зв’язки, які називаються синапсами, і процес передачі сигналів має складні електрохімічні властивості. Синапси виконують роль підсилювачів, тому нейрони сприймають сигнали, один з яких підсумовує збудження, інший - гальмівні імпульси. Нейрони передають збудливі та гальмівні імпульси. Якщо їх алгебраїчна сума трансформується до певного порогу, сигнал із виходу нейрона надсилається до інших нейронів через аксон.[4]

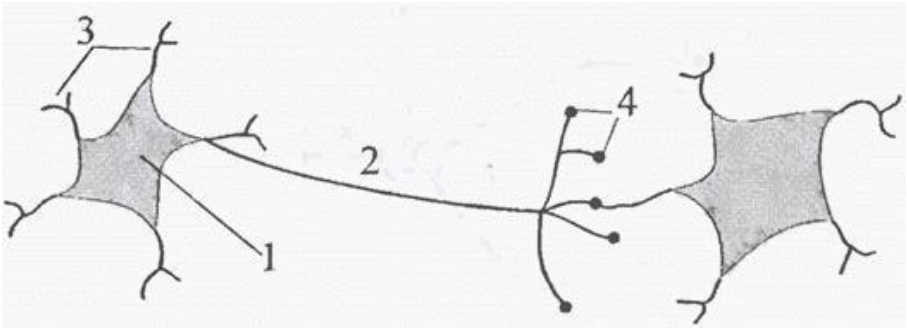


Рисунок 1.1 – З’єднані нейрони: 1 – клітина, 2 – аксон, 3 – дендрити. 4 – синапси

Аналогію можна провести зі штучним нейроном, який отримує вхідний сигнал за допомогою певного зв’язку, зваженого за силою. Вона дорівнює синаптичній активності біологічних нейронів. Кожен нейрон має свій поріг. Щоб обчислити активацію нейрона, потрібно відняти поріг із суми всіх ваг. Це значення також відоме як постсинаптичний потенціал нейрона. Перетворення сигналу активації на вихідний сигнал за допомогою функції передачі.

Нейрон - це інформаційна одиниця нейронної мережі, яку можна представити наступним рисунком. 1.2.:

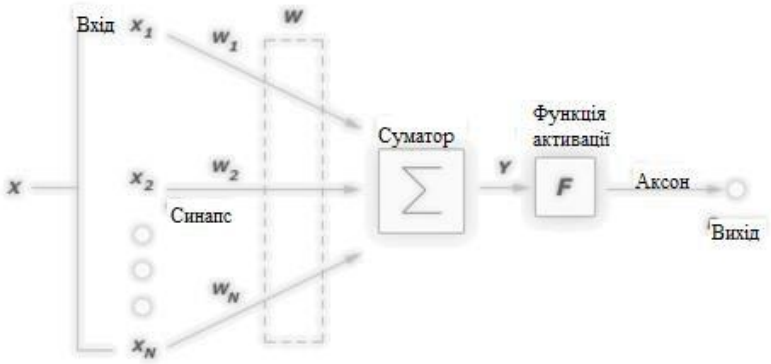


Рисунок 1.2 – Модель штучного нейрону

Ця модель була запропонована Маккалохом і Піттсом у 1943 році [16].

Враховуючи вищесказане, ви можете побачити три основні елементи штучного нейрона:

1. Синапси - характеризуються вагою. Він з’єднує нейрони. Введіть добуток хі та синаптичної ваги wi, щоб охарактеризувати силу синаптичного зв’язку;

2. Суматор - тіло нейрона, яке підсумовує вхідні сигнали інших нейронів.

3. Функція активації – обчислює вихідний рівень одного нейрона, від якого інші нейрони отримують вхідні дані.[8]

Формула, яка описує, як працює нейрон, іншими словами, функція активації, яка обчислює вихідний сигнал. Як параметр на виході вхідного суматора надходить сигнал. У загальному вигляді функція активації може бути виражена як:

$$0ut = F(S - \Theta)$$

де

F(x) – функція активації;

S – середньозважене значення суми, отримане під час першого етапу обчислення вихідного значення нейрона;

Θ – поріг спрацьовування функції активації.

Практично найчастіше використовують такі функції активації як:

1. Одиничний стрибок (порогова функція):

$$0ut = \begin{cases} 0, & S < \Theta \\ 1, & S \geq \Theta \end{cases}$$

Якщо значення менше за порогове значення, значення функції активації є мінімально допустимим значенням, інакше це максимально допустиме значення.

2. Сигмоїдна функція (сигмоїд):

$$0ut = \frac{1}{1 + e^{-\alpha S\prime}}$$

де α – параметр нахилу сигмоїдальної функції активації. Змінивши цей параметр, ви можете побудувати функції з різною крутизною.

Монотонне зростання диференціала всюди описується як відносно нелінійна функція з насиченням. Сигмовид дозволяє посилити слабкі сигнали. Гросберг (1973) виявив, що така нелінійна функція активації вирішує дилему насичення шумом.[14]

3. Гіперболічний тангенс:

$$0ut = th(S), \text{або } 0ut = \frac{e^{-s} - e^{-s}}{e^{-s} - e^{-s}}$$

Така функція часто використовується в мережах з безперервними сигналами, оскільки вона може повертати від'ємне значення результату.[10]

1.3 Класифікація нейромереж

Нейронні мережі мають різні архітектури, кожна з яких призначена для виконання завдання, яке найкраще підходить для цієї архітектури. Нейронна мережа також може бути представлена у вигляді графа зі зваженими зв'язками, де нейрони є вузлами. Залежно від архітектури нейронних зв’язків їх можна об’єднати в дві категорії (рис. 1.3): мережі з прямим поширенням, де граф не має циклів, і рекурентні або інверсні мережі.[6]

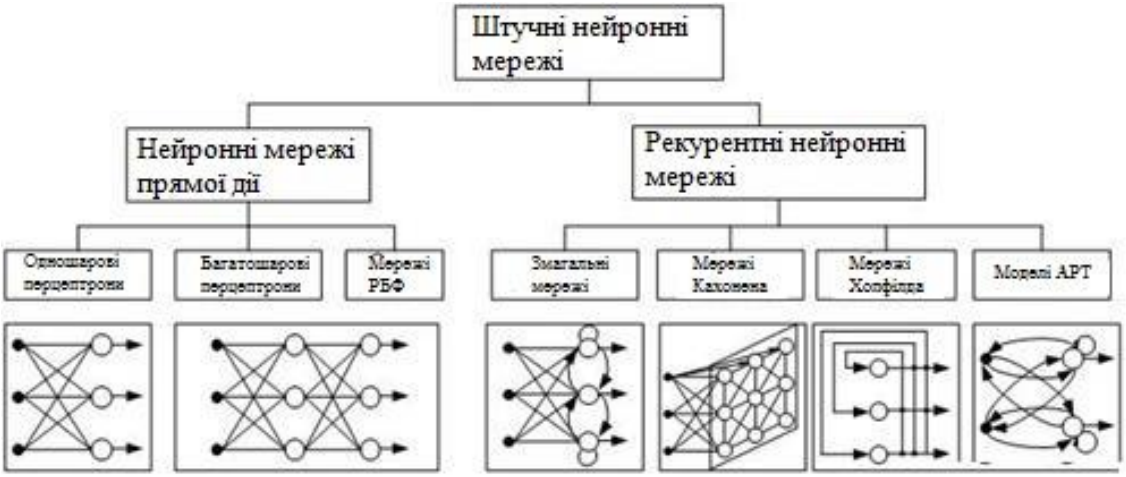


Рисунок 1.3 – Класифікація нейромереж за архітектурою

Наприклад, мережі RBF — це мережі, які використовують радіальні базисні функції як функції активації.[18]

ART (Adaptive Resonance Theory) — це штучна нейронна мережа, заснована на теорії адаптивного резонансу Стівена Гроссберга та Гейла Карпентера.

Нейронні мережі також відрізняються за такими ознаками:

– структура мережі;

– характеристики моделей нейронів;

- Функція електронного навчання.

За структурою нейронна мережа поділяється на рис.1. 1.4. На:

- Частково і повністю пов'язані;

– мають випадкові та періодичні зв’язки;

– Має симетричні та асиметричні зв’язки.[12]

Рисунок 1.4 – Класифікація нейронних мереж за структурою

Частково зв'язані нейронні мережі поділяються на одношарові та багатошарові, з прямим, перехресним і зворотним зв'язком. У мережі з прямими з'єднаннями нейрон j-го рівня на вході може бути підключений тільки до нейрона і-го типу, де j > і, тобто до нейронів нижнього рівня. У мережах із перехресними з’єднаннями допускаються з’єднання всередині рівня, тобто замініть наведену вище нерівність на j >= і. У нейронній мережі зі зворотним зв'язком, коли j < і, і-те з'єднання j-го рівня використовується з і-м входом.

За топологією можна виділити три основних типи нейронних мереж:

– повністю з’єднані (рис. 1.5.а);

– багатошарові (рис. 1.5.б);

– з слабкими зв’язками (рис. 1.5.с).

У повністю підключеній нейронній мережі будь-який нейрон передає свої дані іншим, включаючи себе. Вхідний сигнал передається всім нейронам. Вихід мережі може бути повністю або частково вихідним сигналом нейронів після того, як мережа проходить кілька циклів.[19]

У мережі з великою кількістю шарів нейрони групуються в шари. Кожен нейрон, у свою чергу, містить набір нейронів з унікальним вхідним сигналом і може мати будь-яку кількість нейронів і не залежить від кількості нейронів в інших шарах.

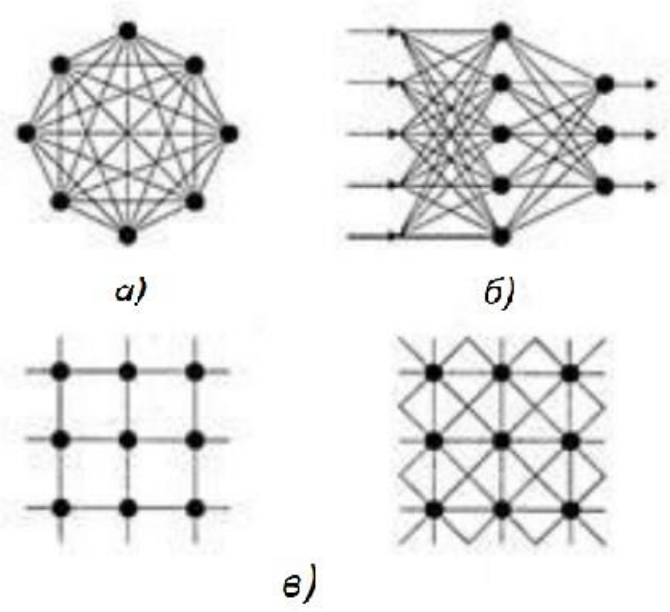


Рисунок 1.5 – Архітектури нейромереж: а) – повнозв’язна мережа, б) – багатошарова мережа з послідовним зє'нанням, в) – слабозв'язна мережа

Взагалі кажучи, мережа складається з Q рівнів, пронумерованих зліва направо. Зовнішній вхідний сигнал подається на вхідні нейрони вхідного шару, а на виході мережі є вихідний сигнал останнього шару. Крім вхідного та вихідного рівнів, багатошарова нейронна мережа має один або кілька прихованих шарів. З'єднання виходу нейрона в шарі q з входом нейрона в наступному рівні (q+1) називається послідовним.

Щодо багатошарові нейронної мережі, можна виділити вид:

1) Монотонний. Це окремий випадок багатошарових мереж із підключенням і додатковими нейронами. За винятком останнього шару, кожен шар поділяється на дві частини: збудливу і гальмівну. Зв'язки між блоками також поділяються на гальмівні і збудливі.

Якщо існує лише пов’язане збудження від нейрона в блоці А до нейрона в блоці В, це означає, що будь-який вихідний сигнал блоку є монотонно неспадною функцією будь-якого вихідного сигналу блоку А.

Якщо ці з’єднання є просто придушенням, тоді будь-який вихідний сигнал блоку В є незростаючою функцією будь-якого вихідного сигналу блоку А. Для

нейронів монотонної мережі необхідна монотонна залежність вихідного сигналу нейрона від параметрів вхідного сигналу.

2) Мережа без зворотного зв'язку. У цих мережах нейрони вхідного рівня отримують вхідні сигнали, перетворюють і передають їх нейронам першого прихованого шару і так далі, а потім передають їх на вихід, який надає сигнал інтерпретатору та користувачеві.

Якщо не вказано інше, кожен вихідний сигнал шару q надсилається на вхід усіх нейронів у шарі (q + 1); однак рівень q можна комбінувати з будь-яким (q

+ p) го шару. Багаторівневі мережі без зворотного зв’язку розрізняють як повнозв’язані мережі (вихід кожного нейрона на рівні q з’єднаний із входом кожного нейрона (q+1)) і частково повнозв’язані.[15]

Класичним варіантом мережі є повноз’єднана мережа прямого розподілу (рис. 1.6).

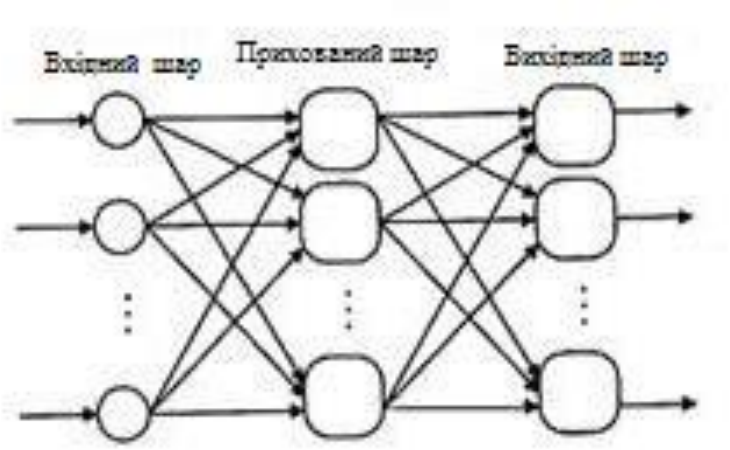


Рисунок 1.6 – Багатошарова (двошарова) нейромережа прямого поширення

3) Мережа зі зворотним зв'язком. У такій мережі інформація з наступних рівнів передається на попередні рівні.

Як приклад мережі зі зворотним зв'язком на рисунку 1. 1.7 Представлені частково рекурентні мережі Елмана та Джордана.

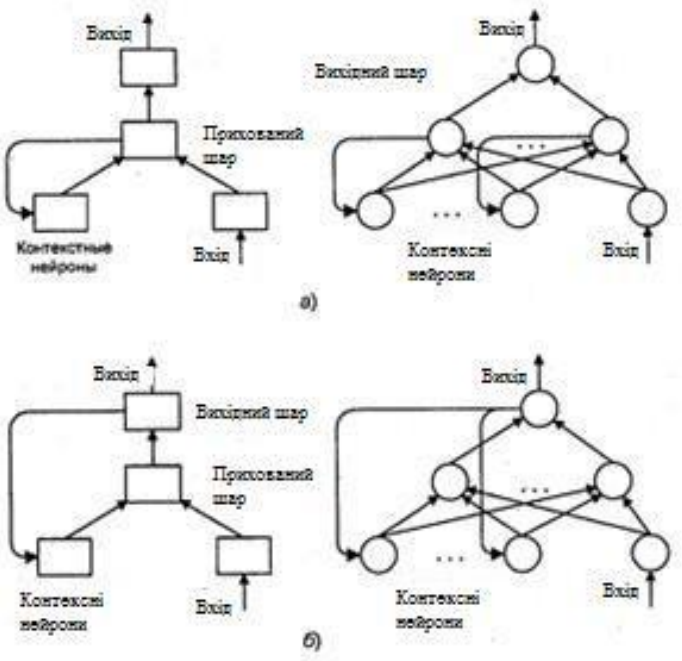


Рисунок 1.7 – Частково-рекурентні мережі: а) – Елмана, б) – Жордана

У слабозв'язаної нейронної мережі нейрони розташовані у вузлах прямокутної або гексагональної решітки. Кожен нейрон пов'язаний з чотирма, шістьма або вісьмома своїми сусідами.

За вхідним і вихідним сигналом ШНМ його можна розділити на:

- моделювання,

- Двійковий.

Двійковий працює лише з двійковими сигналами, і вихід кожного нейрона може приймати значення логічного нуля або логічної одиниці, щоб бути заблокованим або активованим відповідно.

До найпоширеніших моделей відносяться:

– модель Хопфілда;

– машини Больцмана;

– Мережа Кохонена;

– рекурентні нейронні мережі;

– Згорточні нейронні мережі;

– Багатошарові та одношарові персептрони.

Прикладом класичної архітектури мережі прямого поширення є повнозв’язана нейронна мережа прямого поширення або FNN.

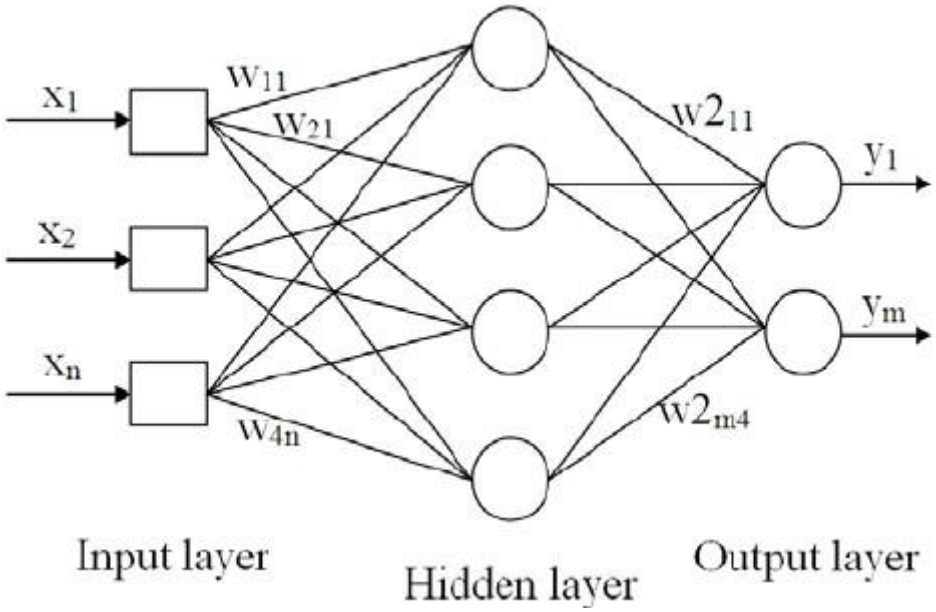


Рисунок 1.8 – Тришаровий персептрон

Перцептрон, який ми бачимо на рисунку. 1.8 Також відома як класична нейронна мережа, це повноцінна багатошарова мережа. Повністю з’єднаний, оскільки кожен нейрон з’єднаний з усіма нейронами попереднього шару. Мережа може дуже успішно виконувати завдання класифікації. Однак у нього є дві проблеми:

1) Багато параметрів

Наприклад, якщо ви використовуєте 3-шарову приховану нейронну мережу для обробки 100*100 пк зображень, це означає, що вхідні дані матимуть 10 000 пк і їх буде зменшено до 3 шарів. Якщо враховувати всі параметри, то в такій мережі буде близько мільйона. Існує багато посібників зі створення нейронних мереж із мільйонами параметрів.[5]

Крім того, мережі з багатьма параметрами мають додаткову тенденцію до повторного використання. Він може згадувати щось, чого насправді не існує: шум у наборі даних.

2) Градієнт розпаду

Коли нейронна мережа має багато шарів, у кінці може залишитися лише невелика частка.

1.4 Принципи навчання нейромереж

Навчання ШНМ – це процес налаштування параметрів нейронної мережі шляхом імітації середовища, в яке вбудована модель. Тип навчання представлений параметризованим методом. Існує різниця між алгоритмами навчання мереж з викладачами та без них.

Перш ніж нейронна мережа отримає завдання, її необхідно навчити, як показано. (Див. Рисунок 1.9).[7]

Навчальний процес викладача складається з подання мережі навчальних прикладів. Зразки надсилаються на вхід мережі, який потім обробляється в структурі нейронної мережі, а вихід мережі обчислюється та порівнюється з відповідним значенням цільового вектора, що представляє бажаний вихід мережі.

Потім розраховується похибка за певними правилами, а ваговий коефіцієнт проміжного з’єднання мережі змінюється відповідно до обраного алгоритму. Послідовне введення векторів для навчальної множини, обчислення помилок і обчислення ваг

– Коригуйте кожен вектор, поки помилка в навчальній матриці не досягне прийнятного рівня.

Під час неконтрольованого навчання навчальний набір містить лише вхідні вектори. Алгоритм навчання регулює ваги мережі для отримання узгоджених вихідних векторів, тобто вхідні вектори, які представлені досить близько, дадуть той самий результат.

Таким чином, сам процес навчання розрізняє статистичні властивості навчального набору та групує подібні вектори в класи. Заданий вхідний вектор даного класу дасть конкретний вихідний вектор, але неможливо передбачити, який вихід дасть цей клас вхідних векторів, не знаючи.

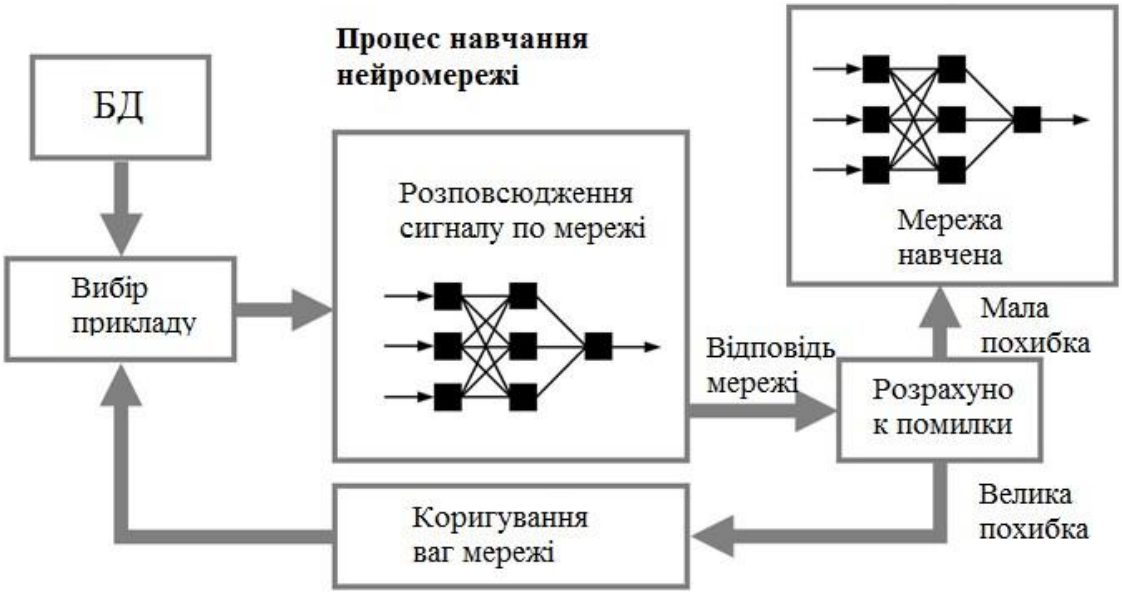


Рисунок 1.9 – Графічна інтерпретація процеса навчання нейромережі

Тому результати такої мережі повинні бути перетворені в зрозумілу форму, зумовлену процесом навчання. Це не головна проблема. Як правило, неважко визначити зв'язок між входом і виходом, встановленим мережею.

Сигнальний метод Гебба і Оя використовується для навчання нейронних мереж без вчителя.

Математично процес навчання можна описати так. Під час роботи нейронна мережа формує вихідний сигнал Y шляхом реалізації функції Y = G(X). Якщо вказана архітектура мережі, тип функції G визначається значенням синаптичного ваги та зміщенням мережі.[17]

Нехай розв’язком задачі є функція Y = F(X), задана параметрами вхідних і вихідних даних (X 1, Y 1), (X 2, Y 2), ..., (X N, Y N), де Y k = F (X k) (k = 1, 2, ..., N).

Навчання складається з пошуку (синтезу) функції G, яка близька до F у сенсі відповідності функції помилки E, див. рис. 1.9.

Якщо вибрати багато навчальних прикладів - пар (X N, Y N) (де k = 1, 2, ..., N) і вибрати метод обчислення функції помилок E, тоді навчання нейронної мережі стає багатовимірним. Оптимізація задачі, має дуже великі розміри, оскільки функція E

взагалі може бути багатоопуклою задачею з довільною формою навчання та налаштуванням часу.

Для вирішення цієї проблеми можна використати наступний алгоритм:

1) Алгоритм локальної оптимізації для обчислення приватних похідних першого порядку:

– градієнтний алгоритм (метод найшвидшого спуску),

– 1D та 2D методи оптимізації для цільових функцій у зворотному градієнтному напрямку,

– Метод спряженого градієнта,

- Спосіб розгляду зворотних напрямків градієнта за кілька кроків алгоритму.

2) Алгоритм локальної оптимізації для обчислення приватних похідних першого та другого порядку:

- Метод Ньютона,

– Методи оптимізації для розріджених матриць Гессе,

– метод квазіньютона,

– метод Гаусса-Ньютона,

– метод Левенберга-Маркара та ін.

3) Алгоритм випадкової оптимізації:

- випадковий пошук,

– Імітація відпалу,

– методи Монте-Карло (числові методи статистичного тестування);

4) Алгоритм глобальної оптимізації (задача глобальної оптимізації вирішується сортуванням значень змінних, від яких залежить цільова функція).

Існує чотири основні правила навчання, пов’язані з відповідними мережевими архітектурами: виправлення помилок, правило Больцмана, правило Геба та закон про конкуренцію.[9]

1) Виправлення помилок. Для кожного вхідного прикладу вказується бажаний вихід (ціль), який може не відповідати дійсному значенню (очікуваному). Правила коригувального навчання неправильно використовують різницю між цільовою змінною та змінною-прогностикою, щоб змінити вагові коефіцієнти, щоб зменшити

помилку невідповідності. Тренуйтеся тільки в разі помилкових результатів. Існує багато модифікацій цього правила навчання.

2) Закон Больцмана. Правило Больцмана — це стохастичне правило навчання, засноване на аналогічних термодинамічних принципах. В результаті його реалізації вагові коефіцієнти нейронів підлаштовуються відповідно до бажаного розподілу ймовірностей. Дослідження правила Больцмана можна розглядати як окремий випадок виправлення помилок, де помилка розуміється як різниця в кореляції стану в двох модах.

3) Закон Хебба. Першим підходом, використаним для вивчення ШНМ без вчителя, є правило Д. Хебба, яке в нейрофізіології формулюється так:

Якщо аксон клітини А знаходиться досить близько до клітини В і бере участь у її руйнуванні безперервно і періодично, процес метаболічних змін відбудеться в одному або обох нейронах. Це призводить до підвищення ефективності нейрона А як стимулятора нейрона В.

4) Закон про конкуренцію. На відміну від правила Хебба, де багато вихідних нейронів можуть запускатися одночасно, тут вихідні нейрони конкурують. А вихідний нейрон із найбільшою зваженою сумою є «переможцем». Виходи інших вихідних нейронів налаштовані на неактивні. Під час навчання змінюється лише вага нейрона-переможця зі збільшенням наближення до цього вхідного прикладу.[20]

Існує багато алгоритмів навчання, призначених для вирішення різноманітних завдань. Серед них алгоритм зворотного поширення, який є одним із найпотужніших. Його основна ідея полягає в тому, що зміни синаптичних ваг враховують локальний градієнт функції помилки.

Різниця між фактичною відповіддю мережі та правильною відповіддю мережі, визначеною на вихідному рівні, поширюється в протилежних напрямках (рис. 1.10.) – у напрямку потоку сигналу.

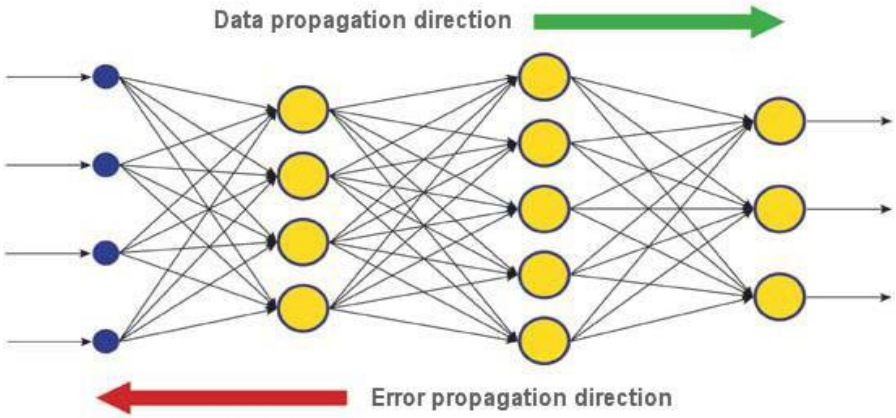


Рисунок 1.10 – Схема поширення інформації та помилки в нейронній мережі під час навчання за методом зворотного поширення помилки

У результаті нейрон здатний визначити внесок кожного зі своїх вагових коефіцієнтів у загальну помилку мережі. Найпростішим практичним правилом є найшвидший метод спуску.

Звичайно, навчаючи нейронну мережу таким чином, немає впевненості, що вона навчається якнайкраще, тому що завжди є шанс, що алгоритм досягне локального мінімуму. Для цього використовуються спеціальні методики, що дозволяють «відсікати» знайдені рішення від локальних екстремумів.

Якщо після кількох таких дій нейронна мережа сходиться до одного і того ж рішення, можна зробити висновок, що знайдене рішення, швидше за все, буде оптимальним.

1.5 Ансамбль нейромереж та передумови його створення

Кожен, хто майже відразу стикався з інтелектуальним аналізом даних, знає, що жоден із цих методів не є ідеальним, тому їх так багато. Більше року дослідники та ентузіасти в цій галузі думали над тим, як знайти компроміс між точністю, простотою та можливістю інтерпретації кожної моделі. Однак слід зазначити, що більшість експертів віддають перевагу точності. Якщо подумати, то саме ця якість робить модель корисною.

Одним із способів підвищення точності моделі є створення ансамблів моделей, які являють собою набори моделей, які використовуються для вирішення типових проблем.

Ансамблі нейронних мереж — це набори моделей нейронних мереж, які приймають рішення шляхом усереднення продуктивності окремих моделей. Залежно від того, як побудований ансамбль, його використання може вирішити одну з двох проблем: тенденцію до поганої роботи базової архітектури нейронної мережі.

Елемент ансамблю — це параметрична модель, яка використовується при побудові колекції.[13]

Формування ансамблю стосується формування остаточного набору базових класифікаторів і подальшого об’єднання їхніх передбачень у єдине передбачення складеного класифікатора. Зрозуміло, що комплексний класифікатор дасть точніші результати, особливо в таких випадках:

– кожен класифікатор має хорошу точність,

- вони призводять до різних результатів (їх плутають з різними наборами).

Щоб моделі (класифікатори) можна було об’єднати в ансамбль, ми демонструємо наступні переваги:

– Зведіть до мінімуму ефект випадковості.

Кумулятивний класифікатор є середнім значенням помилок кожного основного класифікатора, тому вплив випадковості на середнє припущення значно зменшується.

– Зменшити дисперсію.

Загальна думка про велику кількість моделей краще, ніж про одну модель. В економіці це називається диверсифікацією – розширення асортименту виробленої продукції дозволяє підвищити ефективність виробництва та уникнути банкрутства. Набір моделей з більшою ймовірністю знайде загальний оптимум, оскільки пошук бере початок з іншого набору початкових припущень.[21]

3. Уникати перевищення встановлених лімітів. Глобальні припущення знаходяться поза основними припущеннями. Коли ви будуєте комбіновані гіпотези будь-яким способом (логістична регресія, усереднення, голосування), усі гіпотези розвиваються, тому результати не перевищують їх.

Крім того, дослідження показали, що колекції ідентифікаторів часто точніші, ніж один класифікатор. Одна з таких колекцій показана на рисунку 1.11.a.

Він використовує кілька класифікаторів, кожен з яких приймає рішення щодо об'єкта, представленого у вхідних даних. Потім ці окремі рішення об’єднуються в об’єднувач. На стороні виводу інтегрована мітка класу випущеного об’єкта. Інтуїтивно зрозуміло, що дати чітке визначення набору класифікаторів неможливо

Очевидно, що неможливо дати чітке визначення набору класифікаторів. Ця загальна невизначеність показана на рисунку 1.11.bd. Насправді будь-яка колекція сама є класифікатором (рис. 1.11.б.). Компоненти його базового класифікатора витягуватимуть складні (часто неявні) закономірності з потоку даних, а уніфікатор стане простим класифікатором, який поєднує ці функції.

З іншого боку, ніщо не заважає нам викликати цей ансамбль за допомогою стандартного класифікатора нейронної мережі (рис. 1.11.c.). Нейрони в його передостанньому шарі можна розглядати як окремі класифікатори. Їхні рішення має «розшифрувати» об'єднувач, роль якого бере на себе верхівка.

Тепер ми можемо розглядати функції як примітивні класифікатори, а класифікатори — як їхні комплексні об’єднувачі (рис. 1.11.d.).

Ми поєднуємо прості класифікатори, щоб отримати точні класифікаційні рішення. У своїй статті 2002 року «Комбінування класифікаторів: уроки та наступні кроки» Тін Хо писав:

"Зараз ми не шукаємо найкращий набір функцій і класифікаторів, ми шукаємо найкращий набір класифікаторів, а потім найкращий спосіб їх поєднання. Як ви можете собі уявити, незабаром ми шукатимемо найкращі класифікатори. Якщо ми не розглянемо фундаментальні питання, пов’язані з цим викликом, ми неминуче зіткнемося з цим нескінченним повторенням, витягуючи все більш складні комбінації та теорії, поступово забуваючи початкову проблему».[11]

Для прикладу розглянемо доцільність використання ансамблю згорткових нейронних мереж для задач розпізнавання образів.

Враховуючи статтю та відповідну презентацію, використання ансамблю згорткових нейронних мереж може значно зменшити помилки. Це показано на прикладі розпізнавання образів цифр у базі даних MNIST.[22]

Ми бачимо, що зі збільшенням кількості моделей в ансамблі зростає і точність розпізнавання:

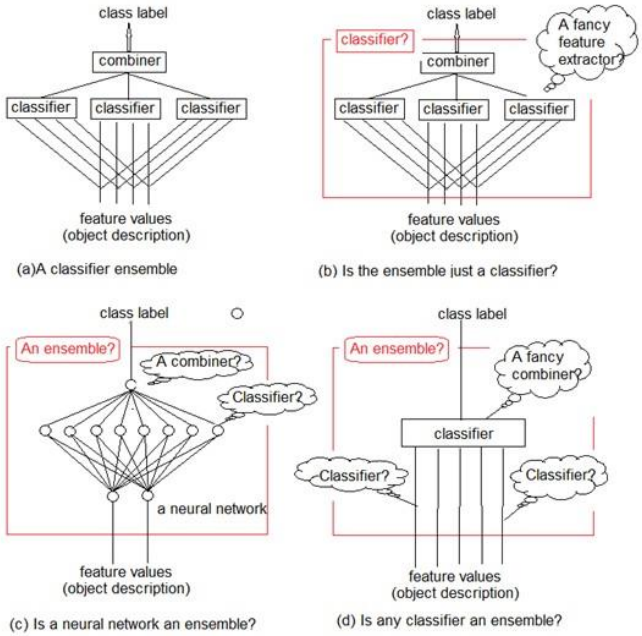


Рисунок 1.11 – Ансамбль класифікаторів

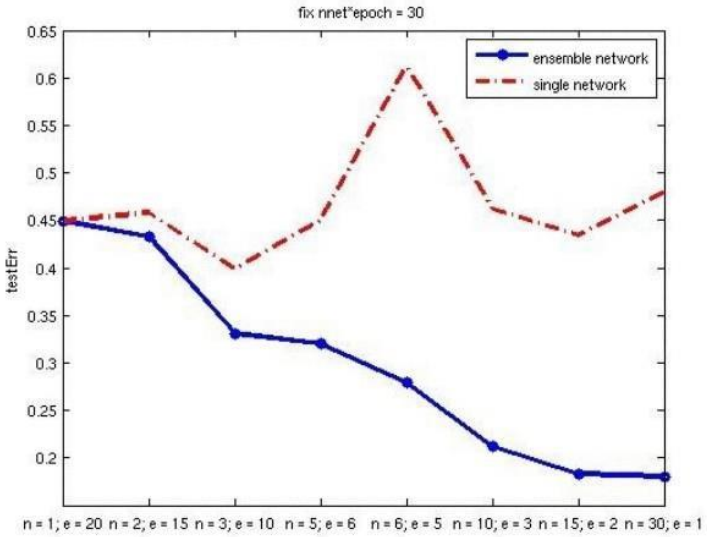


Рисунок 1.12 – Порівняння точності розпізнавання при застосуванні однієї нейромережі та ансамбля

Алгоритм ансамблю може не тільки підвищити точність розпізнавання, але й скоротити час навчання.

Ви також можете стежити за цими статтями 2012 року:

«Практичні проблеми використання згорткових нейронних мереж та їх комітетів у розпізнаванні цифрових образів» (Н.Н. Кузьмицький, 2012, pdf)

Ці статті та роботи наочно демонструють ефективність використання комітетів CNN, сформованих на основі різних стилів написання.[23]

Це одна з таблиць порівняння точності розпізнавання Neural Network Board і KADMOS:

Комітет CNN	MNIST _test	FONT _test	KNI _test	Середнє для 3-х	NIST_HSF 4	OPTDIGI TS	USPS	Середнє для 3-х
MAX_COM	98.23	98.62	98.56	98.47	97.19	94.51	97.80	96.50
AVER_COM	97.64	99.11	98.10	98.28	96.37	93.73	96.52	95.54
MAJOR_COM	93.82	98.50	98.86	97.06	92.01	87.68	92.95	90.88
KADMOS	95.84	98.68	84.95	93.15	94.45	94.66	97.09	95.4

Рисунок 1.13 – Точність розпізнавання (в%) паттернів контрольних та тестових вибірок комітетами нейромереж

Як підсумок:

1. Оглянуто аспекти застосування нейронних мереж у різних сферах діяльності. Наведено приклад використання нейронної мережі.

2. Оглянуто порівняння біологічних нейронних мереж і штучних нейронних мереж. Оглянуто математичні моделі штучних нейронів. Наведено приклад функції активації нейрона.

3. Наведено детальну класифікацію нейронних мереж. Оглянуто найбільш поширені нейронні мережі, які використовуються для класифікаційних завдань: згорточні та рекурентні нейронні мережі та мережі Хопфілда.

4. Оглянуто принципи формування нейронної мережі.

5. Дано визначення поняття нейромережевого ансамблю та визначено можливість використання нейромережевого ансамблю. Завдяки підвищеній точності та швидкості класифікації необхідно створити набір нейронних мереж.

2 СТРУКТУРНЕ ПРОЕКТУВАННЯ НЕЙРОМЕРЕЖЕВОГО АНСАМБЛЮ

2.1 Постановка задачі

У розділі 1.3 представлена класифікація нейронних мереж з ієрархічною структурою: класи, підкласи та їх елементи являють собою єдину нейронну мережу зі своїми описаними властивостями та категоріями вхідних даних. Розділ 1.5 представляє критерії ефективності для штучних нейронних мереж і підкреслює необхідність створення набору нейронних мереж для підвищення точності. Відповідно до наведеної вище синтетичної формули інтегрованої авторської розробки ми визначаємо наступне.

У цій роботі запропоновано модульний принцип організації гібридних нейронних мереж, згідно з яким елементи створеної топології є модулями, що містяться в нейронних мережах різних топологій, і модулі певним чином об’єднані, утворюючи набір. Модуль може складатися з однієї або кількох мереж, які виконують певні частини роботи. Збірник складається з модулів, які використовуються для вирішення задачі класифікації в цілому.

2.2 Загальні принципи створення ансамбля

Як ми вже бачили в розділі 1.5, набір нейронних мереж називається набором топологій, об’єднаних у структуру, яка може відрізнятися за архітектурою, алгоритмами навчання, критеріями навчання та типами нейронів. генератор нейронів. Вхідні дані можна розділити на певні групи обробки різних модулів або надіслати всім модулям одночасно.

В даний час існує декілька типів структур, які використовуються для побудови наборів нейронних мереж: послідовні та паралельні. Послідовні типи побудови зондів нейронної мережі включають послідовні типи помилок і підсилення та їх різноманітні варіації. В даний час немає досліджень послідовних структур, що з'єднують модулі в ансамблі. Приклади побудови мережевих і послідовних модулів представлені в [13, 17]. Приклад паралельного з'єднання модулів наведено в [6].

Основна складність для набору мереж полягає в тому, щоб навчитися вирішувати всі компоненти задачі. Для підвищення ефективності навчання нейронні мережі формуються окремо (по можливості), а потім об’єднуються в єдину структуру. Однак, якщо алгоритм, який налаштовує обрану топологію, належить до різних навчальних класів, всі модулі, що належать до набору, повинні формуватися синхронно, тому необхідно розробити унікальний алгоритм для налаштування всіх модулів навчального модуля. в цілому

У даній магістерській роботі ми використовуємо метод навчання гібридних нейронних мереж з модульною структурою. Тому необхідно враховувати різні топологічні характеристики мереж, що містяться в одному наборі.

2.3 Типізація структур ансамблів гібридних нейромереж

Давайте розглянемо принципи побудови мережевого набору з послідовними та паралельними структурами.

2.3.1 Послідовне з’єднання нейромереж

Організація послідовної множини множин, яка повинна забезпечити вихід одного модуля на вхід іншого модуля, називається послідовною. Ця структура використовується для відновлення вхідних даних або покращення їх невідповідності для виконання основного завдання (прогнозування, апроксимація, класифікація тощо). Перший модуль такого ансамблю заснований на використанні згорткових мереж.

Коли одна мережа розширює можливості іншої, іншим варіантом є використання послідовних модульних з’єднань. Прикладом такої взаємодії є інтеграція багатошарових персептронів у мережу ART. Мережа ART виконує завдання одновимірної кластеризації самостійно, але попередня обробка даних за допомогою кількох рівнів зептрона дозволяє їй виконувати завдання багатовимірної кластеризації.

Загальна схема послідовного з’єднання модуля наведена на рисунку 2.1.

Рисунок 2.1 – Послідовне з’єднання

Підвищення стосується конкатенації модулів, але перш ніж говорити про прискорення, давайте згадаємо два терміни інтелектуального аналізу даних — сильна модель і слабка модель. Надійна модель — це та, яка допускає найменшу кількість помилкових класифікацій. З іншого боку, слабкі моделі допускають помилки без цифр

– Тобто є неточним (або втрачає достовірність).

Таким чином, посилення — це метод, метою якого є перетворення слабкої моделі на сильну шляхом створення набору класифікаторів. У процесі підкріплення класифікатори навчаються послідовно. Таким чином, навчальний набір даних для кожного наступного етапу залежить від точності передбачення попереднього базового класифікатора.

Наприклад, перший алгоритм Boost 1 використовував три основні класифікатори. У цьому випадку перший класифікатор навчається відповідати «все так», другий класифікатор навчається на вибірках, а третій класифікатор навчається на наборі даних, де прогнози перших двох класифікаторів відрізняються.

Сучасна модифікація першого алгоритму передбачає використання нескінченної кількості класифікаторів, кожен з яких навчається на наборі прикладів, а потім по черзі застосовує їх до іншого прикладу.

2.3.2. Паралельне з’єднання нейромереж

Набір мереж, який одночасно передає вхідні дані всім модулям, що утворюють гібридну нейронну мережу, називається паралельною мережею. Основним елементом

створення цієї асоціації є «рівень об’єднання», який відповідає за агрегування продуктивності різних компонентів набору мереж.

Загальна структура ансамблю паралельних нейронних мереж показана на рисунку 1. 2.2.

Модуль може являти собою одну мережу (RBF, ANFIS, TSK, багатошаровий перцептрон тощо) або набір послідовних мереж.

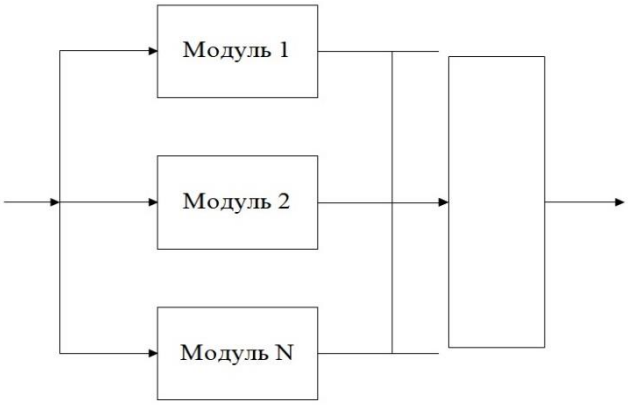


Рисунок 2.2 – Паралельне з’єднання

На відміну від попередніх методів, початкові агрегати формуються з використанням одночасного формування базових класифікаторів (з точки зору математичної логіки, пакетування покращує асоціації, а прискорення покращує пе-сітки).

У процесі розпакування відбуваються такі події:

– Випадково виберіть кілька підмножин з вихідного набору даних, що містить кілька прикладів, що відповідає кількості прикладів у вихідному наборі.

- Оскільки вибір є випадковим, набір прикладів завжди буде різним: деякі приклади будуть згруповані в кілька підмножин, а деякі не підійдуть ні в одну.

– На кожному зразку будується класифікатор.

Узагальніть висновки класифікатора (голосуванням або по рядках).

– Як і у випадку з прискоренням, прогнози класифікатора агрегатора мають бути набагато точнішими, ніж прогнози моделі на наборі даних.

Основні проблеми паралельної побудови кількох груп мереж полягають у наступному:

Для кожного модуля в комплекті найкраще підходять розділи входу. Кожен модуль повинен вирішувати ту частину завдання, в яку він «вписується».

- Основним недоліком використання паралельних наборів є надто складний алгоритм навчання з імовірнісною збіжністю.

2.4 Критерії залучення нейромережі у набір

Крім процедури навчання необхідно сформувати оптимальну структуру мережі: визначити кількість нейронів і топологію зв'язків між ними, вибрати функцію активації і визначити, чи потрібен зворотний зв'язок.

Якщо експерти нейронної мережі налаштовані неправильно (наприклад, наявність надмірних ваг спричиняє погіршення якості загальних ваг, оскільки прогрес нейронної мережі відіграє важливу роль, вони можуть приймати будь-які значення, що також призведе до ваг впустити). потребує більше ітерацій алгоритму навчання), буде отримано рішення. , заснований на висновках некомпетентних експертів.

Для нашого завдання класифікації об’єктів пропонується критерій фільтрації попередніх гіпотез нейронної мережі, який враховує якість розпізнавання штучної нейронної мережі >80%, а також кількість помилок нейронної мережі.

Якщо ці критерії виконуються, нейронна мережа буде частиною ансамблю.

Щоб оцінити кожну модель і якість ансамблю, ми розрахували середню квадратичну помилку (MSE) у тестовій вибірці.

На рис. 2.3 Бачимо, що мінімум похибки в множині (на графіку – вихід).

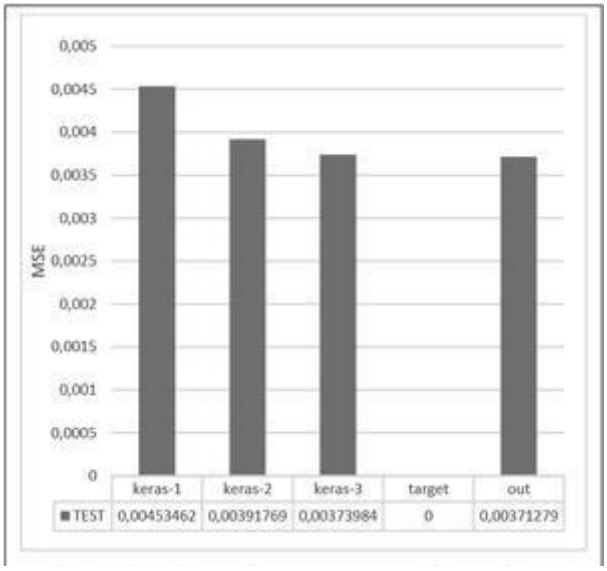


Рисунок 2.3 – Середньоквадратична помилка

Оскільки лінійна регресія використовується як метамодель, лінії регресії можна оцінити в кожній моделі похідного значення. Ці значення записані в EnsembleItem.properties: 1.06192747 keras-1, -0.13015426 keras-2, 0.17757505 keras-3.

Значення ваги можна інтерпретувати так: вага з найбільшою вагою відповідає першій моделі (keras-1), третя модель (keras3) робить невелике коригування, а друга модель (keras-2) забезпечує компенсацію) .

На рис. 2.4 Представлення прогнозів, отриманих за допомогою ансамблю:

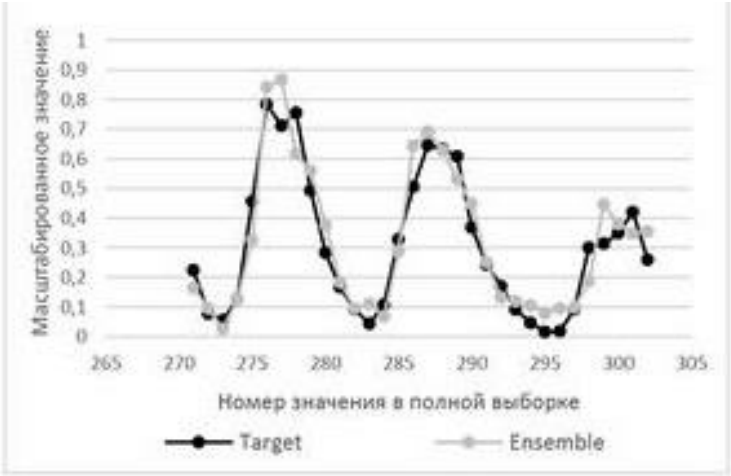


Рис. 2.4 – Приклад прогнозу, отриманого за допомогою ансамблю

2.5 Алгоритм відбору нейронних мереж

Початковою умовою для вибору базової структури є те, що структура повинна виконувати завдання класифікації.

Виходячи з цього твердження, не можна сказати, що нова мережа повинна базуватися на тій самій мережі.

Оскільки набір мереж є функціонально єдиною мережею, можна використовувати одну мережу та послідовно-паралельний набір як основу.

Тому, незалежно від його структури, виникає проблема створення загального алгоритму навчання для обраної бази.

Рішенням цієї проблеми є генетичний алгоритм навчання, який майже не чутливий до структури мережі.

Однак обчислювальна потужність, необхідна для генетичних алгоритмів, є більшою, ніж та, що потрібна для градієнтних методів, тому здатність вивчати генетичні алгоритми суворо обмежена специфікою даного завдання.

Типовий формат ансамблевих навчальних зразків нічим не відрізняється від ініціалізації однієї мережі.

У таблиці 2.1 наведено експериментальні результати різних варіантів формування модулів у гібридній нейронній мережі.:

Таблиця 2.1 – Розрахунок точності різних з’єднань

Тип мономережі	Тип з’єднання	Час роботи	Точність
ANFIS	Мономережа	10	82,6%
	Рекурентна мережа+ Мономережа	21	88,2%
	Мономережа + Мережа Хопфілда	22	87,7%
	Рекурентна мережа + Мономережа +Мережа Хопфілда	25	97,8%
NEFCLASS	Мономережа	8	81,4%
	Рекурентна мережа + Мономережа	15	84%
	Мономережа + Мережа Хопфілда	17	87%
	Рекурентна мережа + Мономережа + Мережа Хопфілда	20,6	96,4%

Рекурентна мережа + мономережа + мережа Хопфілда 20,6 96,4%.

Як показано в таблиці. 2.1 Топологічні варіанти модулів нейронної мережі: оптимальна мережа прямого поширення, базова нейронна мережа (ANFIS або NEFCLASS), асоціативна пам'ять.

Перехідні та періодичні мережі можуть використовуватися як мережі прямого призначення, мережі Хопфілда можуть використовуватися як асоціативна пам'ять, а мережі ANFIS і NEF-CLASS можуть використовуватися як первинні мережі.

Функція зупинки — це метод регулярізації, який дозволяє уникнути переобладнання великих нейронних мереж. Це трохи схоже на пробник, але насправді це модель.

Для формування набору створених мереж я рекомендую побудувати набір моделей з різними структурами або параметрами. Ви хочете, щоб моделі мали дещо різні упередження, робили різні помилки та виправляли одна одну. Наприклад, набір, що містить багато різних моделей, отримав нагороду Netflix.

Навчання розраховане на слабких учнів, а нейронні мережі (особливо глибинні) дуже потужні. Якщо одна нейронна мережа може досягти точності навчальних даних, то покращувати нічого.

Але чому б не спробувати деякі з цих методів і не порівняти їх із отриманими даними перевірки? Правильна відповідь може залежати від набору або навіть комбінації кількох підходів. Приклад генетичного алгоритму можна побачити на рисунку 1. 2.5.

Генетичний алгоритм складається з наступних етапів:

1. Розрахувати функцію пристосованості:

$$E_j({}_p w_j(0)) = 1 - \frac{y}{y_0}$$

де у – значення сигналу, отримане в результаті роботи мережі з вагами, які відповідають j-ої популяції, y_0 – еталонне значення сигналу в навчаючій вибірці.

2. Хромосоми розташовані в порядку зменшення значення функції пристосованості. Менш придатні видаляються з натовпу.

3. Якщо найкраща хромосома має значення, яке найкраще відповідає початковим критеріям, алгоритм припиняє роботу. В іншому випадку перейдіть до кроку 2.

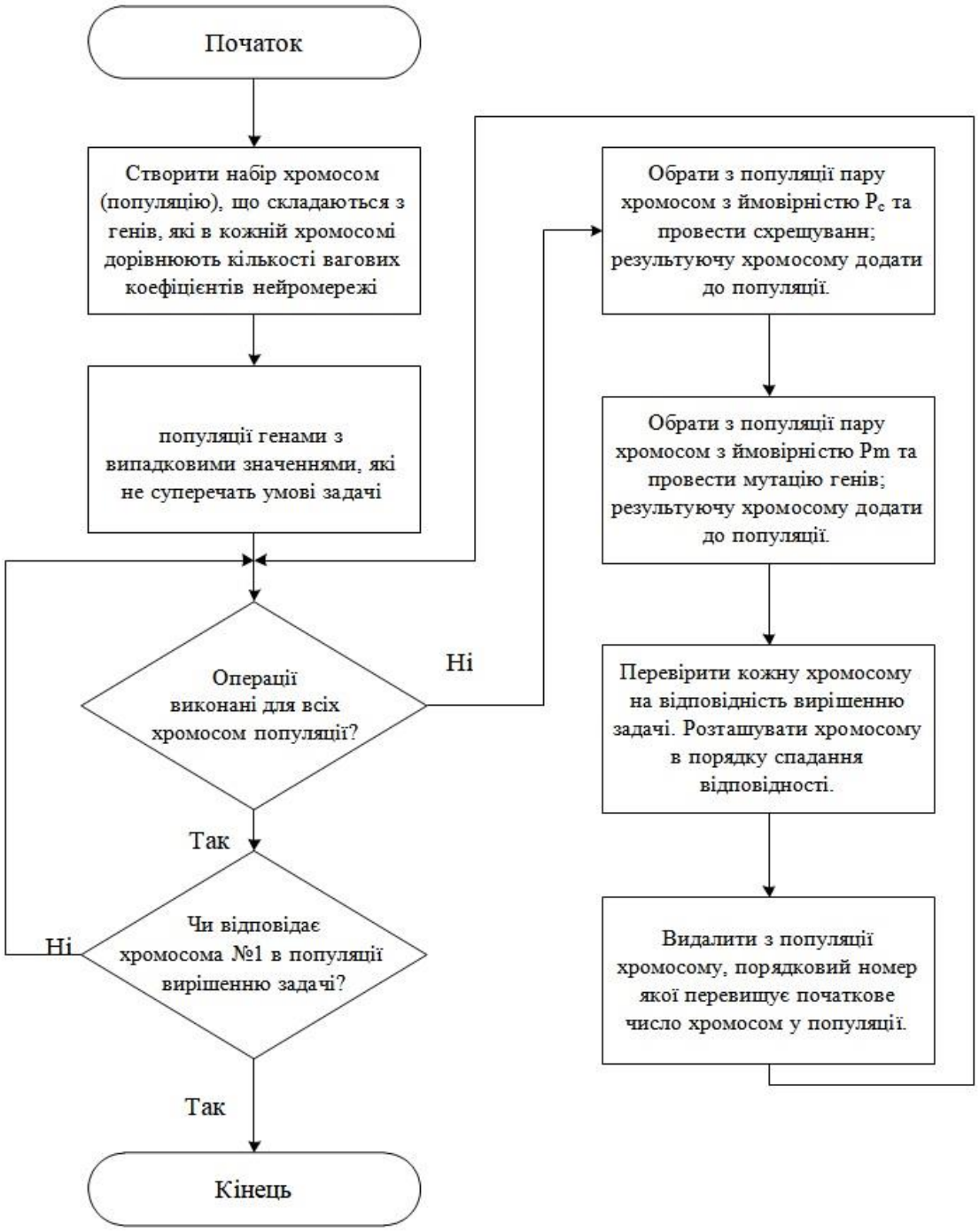


Рисунок 2.5 – Генетичний алгоритма навчання ансаблю

У разі навчання мережевих ансамблів утворення нових популяцій відбувається за рахунок хромосом дітей мережі, батьківські хромосоми постійні. Але оскільки генетичні алгоритми мають різне походження та мають конкретну мету, у нашому дослідженні ми обрали паралельний підхід до побудови та навчання ансамблю

нейронних мереж – використання методів скорочення для модифікації прослуховування.

Тому ми досягли наступних результатів:

1. Розглянемо основні методи створення набору нейронних мереж. Описано переваги та недоліки двох методів створення інтеграцій.

2. Визначає критерії включення нейронних мереж як цілого. Ці критерії виявляють якість оцінок і кількість помилок.

3. Розглянемо алгоритм вибору мережі, яка буде частиною набору. Наведено приклад використання методу паралельної побудови для класифікації нейронної мережі, яка є частиною ансамблю, і визначено, що найкращим результатом є ансамбль, створений трьома мережами з різною архітектурою.

3 НАВЧАННЯ НЕЙРОМЕРЕЖЕВОГО АНСАМБЛЮ

3.1 Постановка задачі нвачання

Добре відомо, що ансамблеві методи складаються з кількох незалежно сформованих моделей і поєднують їхні прогнози, щоб зробити загальний прогноз.

Кожна мережа, що входить в ансамбль, формується за окремим алгоритмом. Багато уваги приділено тому, які типи слабких учнів підбираються та як вони підбираються. Це дуже сильний технічний курс, тому він дуже популярний. Нагадуємо про ансамблеві методи навчання:

- сприяти;

– Завантажувальна агрегація (Bagging);

– AdaBoost;

– стекове узагальнення (змішування);

– Gradient Boosting Machine (GBM);

– Gradient Boosted Regression Tree (GBRT);

- Випадковий ліс.

Необхідно визначити, які проблеми можуть виникнути під час навчання множини, і визначити найкращий спосіб їх вирішення, особливо для задач класифікації.

Прочитавши список прикладів проблем машинного навчання, ви побачите схожість.

Це цінна навичка, оскільки заглиблення в суть проблеми дозволить вам ретельно продумати потрібні вам дані та тип алгоритму, який слід застосувати. Існують загальні категорії проблем машинного навчання. Наведений нижче клас проблем є прототипом для більшості проблем, які ми обговорюємо в машинному навчанні.

1) Класифікація: дані вказують на те, що вони віднесені до таких категорій, як спам/не спам або шахрайство/не шахрайство.

Моделювання рішень передбачає призначення міток новим даним без міток. Це можна розглядати як проблему дискримінації, моделювання відмінностей або подібності між різними групами.

2) Регресія: дані містять справжні значення, а не мітки.

Легким для розуміння прикладом є дані часових рядів за певний проміжок часу, наприклад, курси акцій.

Рішення моделювання полягає в тому, щоб передбачити важливість нових випадків.

3) Кластеризація: дані не маркуються, але можуть бути згруповані на основі схожості в природній структурі даних та інших показників.

Наведений вище список — це розташування зображень облич без назви, коли користувач призначає назву групі, наприклад iPhoto на Mac.

4) Видалення правила: дані служать основою для видалення пропозиційних правил (попереднього/наступного).

Такі правила можуть бути спрямованими, але зазвичай такими не є, тобто ці методи знаходять статистично підтверджуючі зв’язки між атрибутами даних, які не обов’язково корелюються, як очікувалося.

3.2 Алгоритми тренування нейромереж

У попередніх розділах були розглянуті методи побудови ансамблю гібридних нейронних мереж, оскільки нейронні мережі формуються за власними алгоритмами, ці методи побудови можна назвати методами навчання та побудови. Наприклад, при послідовному побудові ансамблів з використанням методів підвищення, нейронні мережі навчаються послідовно. У магістерській роботі ми будемо використовувати алгоритм навчання прослушки та вдосконалювати його.

Модифікований алгоритм навчання включає наступне:

1. Формат навчальних зразків і тестових зразків.

По-перше, дані поділяються на навчальні та тестові. У нашому прикладі навчальні дані складають 80%, а тестові – 20%.

Потім кожен вхідний пакет S повинен бути розділений на X вибірок вхідних значень і Y вибірок вихідних значень.

2. Навчальні модулі. Модуль може включати більше двох нейронних мереж, один або кілька типів архітектур.

Модуль вивчається в три етапи:

Фаза 1. Навчання згорткових нейронних мереж.

Щоб почати будувати нашу мережу, вам потрібно вирішити, як вимірювати якість розпізнавання. У нашому прикладі ми будемо використовувати найпоширенішу функцію в теорії нейронних мереж, середню квадратичну помилку (MSE, MSE) [3]:

$$E^p = \frac{1}{2}(D^p - O(I^p, W))^2$$

У цій формулі Ep — помилка розпізнавання p-ої навчальної пари, Dp — бажаний вихід мережі, а O(Ip, W) — вихід мережі, який залежить від p-го входу та вагові коефіцієнти W, включаючи ядро згортки, зміщення, вагові коефіцієнти для шарів Si та F. Навчальна задача полягає в тому, щоб налаштувати ваги W так, щоб для будь-якої пари приводів (Ip, Dp) вони давали мінімальну похибку Ep. Щоб обчислити похибку для всієї навчальної вибірки, ми просто беремо середнє арифметичне помилок для всіх навчальних пар. Такі помилки ми позначаємо як єп.

Для мінімізації функції помилки Ep найбільш ефективним є градієнтний метод. Давайте розглянемо природу градієнтних методів на найпростішому одновимірному прикладі (тобто коли у нас є лише одна вага). Якщо ми розкладемо функцію помилок Ep в ряд Тейлора, то отримаємо наступний вираз:

$$E(W) = E(Wc) + (W - Wc)\frac{dE(Wc)}{dW} + \frac{1}{2}(W - Wc)^2\frac{d^2E(Wc)}{dW^2} + \dots$$

Тут E — та сама функція похибок, а Wc — початкове значення вагових коефіцієнтів. У шкільній математиці ми пам’ятаємо, що для того, щоб знайти екстремум функції, її потрібно продиференціювати та порівняти з нулем. Тож давайте візьмемо похідну функції вагової помилки та вилучимо доданки вище другого порядку:

$$\frac{dE(W)}{dW} = \frac{dE(Wc)}{dW} + (W - Wc)\frac{d^2E(Wc)}{dW^2}$$

з цього виразу випливає, що вага, при якому значення функції помилки буде мінімальним, можна обчислити з наступного виразу:

$$Wmin = Wc - (\frac{d^2E(Wc)}{dW^2})^{-1}\frac{dE(Wc)}{dW}$$

Тобто оптимальна вага обчислюється як поточна вага мінус похідна функції помилки ваги, поділена на другу похідну функції помилки.

Однак для багатовимірного випадку (тобто для вагової матриці) лише перша похідна стає градієнтом (вектором приватних похідних), а друга похідна стає Гессеном (матрицею приватних похідних). Звідси є два варіанти.

Якщо опустити другу похідну, ми отримаємо найшвидший алгоритм градієнтного спуску. Гессе зазвичай замінюють на щось простіше. Наприклад, одним із найвідоміших і найефективніших методів є метод Левенберга-Маркара (LM), який замінює метод Гессе, але два методи, які нам потрібно зрозуміти, полягають у тому, що алгоритм LM має обробляти всю навчальну вибірку, тоді як градієнтний спуск Алгоритм можна використовувати при кожній окремій роботі з навчальними зразками.

В останньому випадку алгоритм називається стохастичним градієнтом. Оскільки наша база даних містить 60 000 навчальних зразків, ми краще підходимо до стохастичних градієнтів.

Ще однією перевагою стохастичних градієнтів порівняно з LM є їх менша тенденція досягати локальних мінімумів.

Існує також випадкова модифікація алгоритму LM, яка буде обговорюватися пізніше.

Запропонована формула допомагає розрахувати похибку, отриману за ваговими коефіцієнтами у вихідному шарі. Обчислення помилки в прихованому шарі дозволяє використовувати добре відомий метод зворотного поширення помилок у розділі 2.

Фаза 2. Навчання рекурентної нейронної мережі.

На цьому етапі реалізується класифікатор для об’єктів, заданих послідовністю векторів, за допомогою методу формування рекурентної мережі Елмана за схемою «багато до одного» (рис. 3.1).

Правильно використовувати той самий градієнтний метод [3], що й традиційна мережа прямого розповсюдження, але з деякими модифікаціями.

Він обчислюється за допомогою модифікованого методу зворотного поширення [3], що називається зворотним поширенням у часі (метод BPTT) [6].

Ідея методу полягає в тому, щоб розширити послідовність шляхом перетворення рекурентної мережі в «нормальну» мережу. Як і у випадку з методом зворотного поширення, для мережі прямого поширення [3]:

1. Прямий перехід - стан нашого обчислювального рівня,

2. Зворотне поширення - ми обчислюємо похибку шару,

3. Розрахувати зміну ваги за даними, отриманими на першому та другому етапах.

Розглянемо ці фази докладніше:

1. Прямий прохід: для кожного вектора послідовності {x(1),...x(n)}{x(1),...x(n)}: Обчислюємо стану прихованого шару {s(1),...s(n)}{s(1),...s(n)}і виходи прихованого шару {h(1),...h(n)}{h(1),...h(n)}

$$s(t) = V \cdot x(t) + U \cdot h(t-1) + a \quad s(t) = V\,x(t) + U\,h(t-1) + a$$

$$h(t) = f(S(t)) \quad h(t) = f(s(t))$$

обчислюємо вихід мережі y

$$y(N) = g(W \cdot h(n) + b) \quad y(n) = g(W \cdot h(n) + b).$$

2. Зворотний прохід: обчислюємо помилку вихідного шару δo

$$\delta_O = y - d \quad \delta_O = y - d,$$

– Розраховуємо похибку δh(N)δh(n) прихованого шару в кінцевому стані: δh(N) = WT · δo⊙ f'(S (n))δh(n)=WT · δo⊙f' (s (n)), обчислюємо похибку прихованого шару в проміжному стані

$$\delta h(T)\ \delta h(t) \quad (T = 1, \ldots nt = 1, \ldots n)$$

$$\delta_h(T) = UT \cdot \delta_h(T+1) \odot f(S(n)) \quad \delta_h(t) = UT \cdot \delta_h(t+1) \odot f(s(n))$$

4. Обчислюємо зміну ваг:

$$\Delta W = \delta_0 \cdot (h(n))T\Delta W = \delta_0 \cdot (h(n))T$$

$$\Delta\delta_y = \Sigma\,\delta_0\,\Delta\delta_y = \Sigma\,\delta_0$$

$$\Delta V = \Sigma t\delta_h(T) \cdot (x (t))\, T\,\Delta V = \Sigma t\delta_h(t) \cdot (x (t))\, T$$

$$\Delta U = = \Sigma t\delta_h(T) \cdot (h (t - 1))T\,\Delta U = \Sigma t\delta_h(t) \cdot (h (t\text{-}1))\, T$$

$$\Delta b_h = \Sigma\,\Sigma t\delta_h(T)\,\Delta b_h = \Sigma\,\Sigma t\delta_h(t)$$

Знайшовши спосіб обчислення градієнта функції похибок, ми можемо застосувати модифікацію методу градієнтного спуску, детально описану в [3].

Фаза 3. Навчання нейронної мережі Хопфілда.

Навчання мережі Хопфілда початкових образів ξ_μ^{in} полягає у обчисленні значень J_{ij} елементів матриці. Процес навчання можна формально описати так: нехай нейронна мережа буде навчена розпізнавати позначені моменти. Вхідней $\{\xi_\mu^{in}, \mu = 1, \dots, M\}$ образ $\bar{\xi}_\mu^{in}$ такий: $\bar{\xi}_\mu^{in} = \xi_\mu^{in} + {}'\xi$ де ${}'\xi$ шум накладається на вихідний образ $\left\|\xi_\mu^{in} - {}'\xi_\mu^{in}\right\|$. Насправді, вивчення нейронної мережі – це визначення норми в просторі зображень. Потім, видалення шумового вхідного зображення можна описати як мінімізацію виразу $\left\|\xi_\mu^{in} - {}'\xi_\mu^{in}\right\|$

Важливою особливістю нейронної мережі є відношення кількості ключових образів M, які можуть бути збережені, до кількості нейронів у мережі N: $\alpha = \frac{M}{N}$. Для Хопфілда значення $\alpha = \frac{M}{N}$ не перевищує 0,14.

Обчислення квадратної матриці J_{ij} розміру $N \times N$ для M ключових кадрів проводиться за правилом Хебба:

$J_{ij} = \frac{1}{N} \cdot \sum_{\mu=1}^{M}\left[\xi_{i\mu}^{in} \cdot \xi_{j\mu}^{in}\right]; i \neq j, J_{ii} = 0$ де $\xi_{i\mu}^{in}$ означає i-ий елемент ξ_μ^{in} способу.

Слід зазначити, що через комутативність операції множення дотримується рівність $J_{ij} = J_{ji}$

Вхідне зображення, яке потрібно ідентифікувати, відповідає виходу системи як початкового стану динамічної системи. Це рівняння достатньо для визначення штучної нейронної мережі Хопфілда та може бути реалізоване.

Тоді вам доведеться навчити основам. Загальним методом навчання, який не залежить від базової структури, є генетичний алгоритм. У цьому випадку хромосома - це набір генів, кожен з яких є мережевою вагою, представленою у вигляді двійкового коду.

Хромосоми відповідають алгоритмам навчання індивідуальної топології. Якщо мережева колекція використовується як модуль, слід ввести батьківську хромосому C0, до якої приєднані хромосоми Ci всіх мереж колекції. Під час першого етапу навчання ваги мережі визначаються випадковим чином у діапазоні можливих значень.

3. Застосування обрізки. Відсікання використовується для підвищення точності класифікації - класифікація тестових даних визначає точність класифікації мережі, якщо отримана точність не відповідає очікуваному значенню (наприклад, точність класифікації менше ніж 70% і повинна перевищувати 80%), Нейронні мережі до складу ансамблю не входять.

4. Класифікація тестових зразків за нейронною мережею.

5. Застосування обрізки. Знову застосовуючи метод обрізання, ми отримуємо остаточний набір нейронних мереж, які будуть частиною ансамблю.

Нейронні мережі можна формувати за допомогою різних алгоритмів: вони засновані на еволюційних принципах - імунні та генетичні алгоритми, алгоритми зворотного зв'язку з помилками, алгоритми стохастичної оптимізації, тому ви повинні вибрати алгоритм, виходячи із завдання.

У цьому дослідженні сучасний паралельний алгоритм навчання та формування помилок було застосовано до томографічних і радіологічних зображень, отриманих із ресурсів платформи kaggle, наданих Національними інститутами охорони здоров’я.

Kaggle — це платформа для збору та обробки даних. У своїй роботі пошук використовує принципи краудсорсингу.

Базу даних було створено після обробки оригінального набору чорно-білих зразків розміром 400 х 400 пікселів. База даних містить 4000 навчальних і 1000 тестових зображень. Половина зразків, які використовуються для навчання та тестування, належать до навчальної множини, а друга половина – до тестової множини.

База даних була створена після обробки необробленого чорно-білого набору зразків розміром 400x400 пікселів. База даних містить 4000 зображень для навчання та 1000 зображень для тестування.

Половину навчальної та тестової вибірок відбирають із навчальної множини, іншу половину — з тестової.

Приклад зображення в цій базі даних показано на рисунку 3.1.

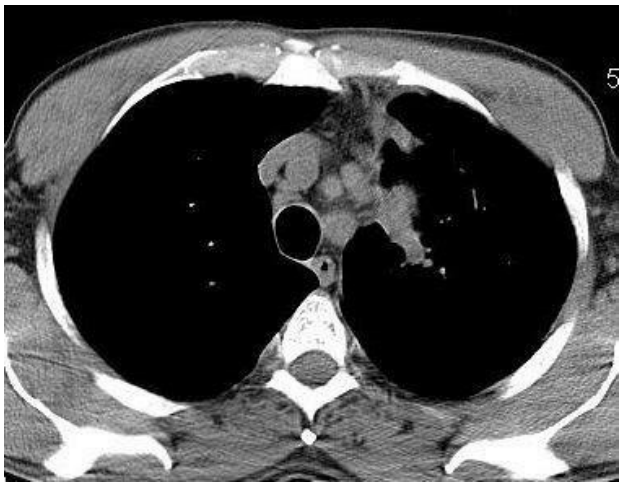


Рисунок 3.1 – Приклад даних з латасету

Результати порівняння розв’язання класичної задачі класифікації за допомогою стандартного алгоритму прослуховування на рисунку 2. 3.2.

Рисунок 3.2 – Результати розпізнавання зображень

Порівняльні результати розв’язання задачі ідентифікації пухлини за допомогою сучасного алгоритму підслуховування наведені на рисунку 1. 3.3.

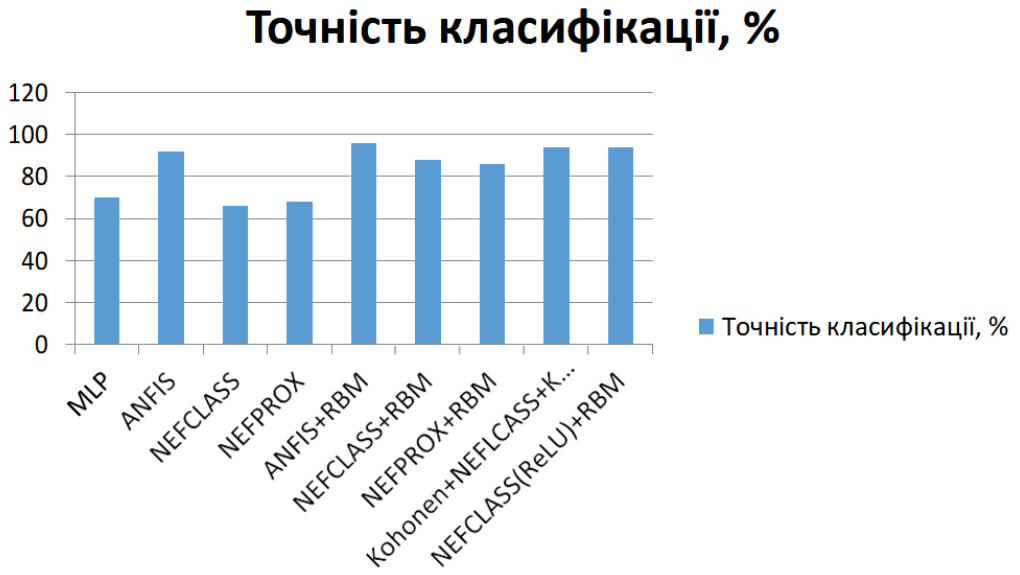


Рисунок 3.3 – Результати реалізації класичної класифікації

Давайте підсумуємо те, що описано в цьому розділі:

1. Оглянуто задачу навчання загальної структури. Проблеми включають класифікацію, регресію та кластеризацію.

2. Запропоновано алгоритм навчання ансамблевої структури. Одним із таких алгоритмів є генетичний алгоритм, який підходить для початку структури ансамблю. Змінено алгоритм прослуховування за допомогою скорочення.

3. Проведено огляд алгоритму навчання нейронної мережі, архітектура якої буде частиною ансамблю. Серед розглянутих алгоритмів можна виділити досить поширений алгоритм зворотного поширення помилок.

4 РЕАЛІЗАЦІЯ І ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Структура програмного забезпечення

Для застосування розробленого набору нейронних мереж розроблено веб-сервіс для медичного закладу, для якого обрано трирівневу архітектуру додатку, що включає такі частини:

– сервер бази даних;

- сервер додатків;

- Клієнтський розділ.

Крім того, для реалізації розробленої архітектури були обрані такі технології:

– Мова програмування Python як основний інструмент розробки;

– мова JavaScript для роботи з клієнтською частиною;

- Sqlite3, як інструмент управління базами даних;

– Angular – для створення веб-додатків;

– Tensorflow – бібліотека машинного навчання від Google;

– HTML – веб-сторінки, які використовуються для створення програм;

– Каскадні таблиці стилів CSS;

– Платформа Bootstrap для дизайну сторінки, включаючи елементи керування користувача.

Оскільки система розроблена як багатокористувацька та масштабована, тобто розширювана без зайвих зусиль, структура системи базуватиметься на трьох основних частинах:

1) Клієнт — це видима частина, з якою користувач взаємодіє під час використання програми. В даному випадку це веб-додаток, тобто клієнтом системи є веб-браузер.

2) Вся необхідна бізнес-логіка реалізована на стороні сервера. Сервер додатків є посередником між користувачем і сховищем даних. Сервер приймає на клієнті запит, надісланий користувачем, при необхідності його обробляє - звертається до сервера бази даних і повертає відповідь на запит.

3) У цьому випадку браузер користувача взаємодіє з фреймворком Angular. Деякі веб-сторінки отримують інформацію від сервера під час обробки запитів JQuery та AJAX. Технологія міграції Django на стороні сервера дозволяє створювати класи моделі, подібні до тих, що створюються в базі даних. Однак ця технологія дозволяє працювати з сутностями, робити вибірку, створювати та редагувати дані. Це полегшує розробникам роботу з даними, оскільки вони використовують моделі сутностей.

Схема компонентів показана на рисунку. 4.1.

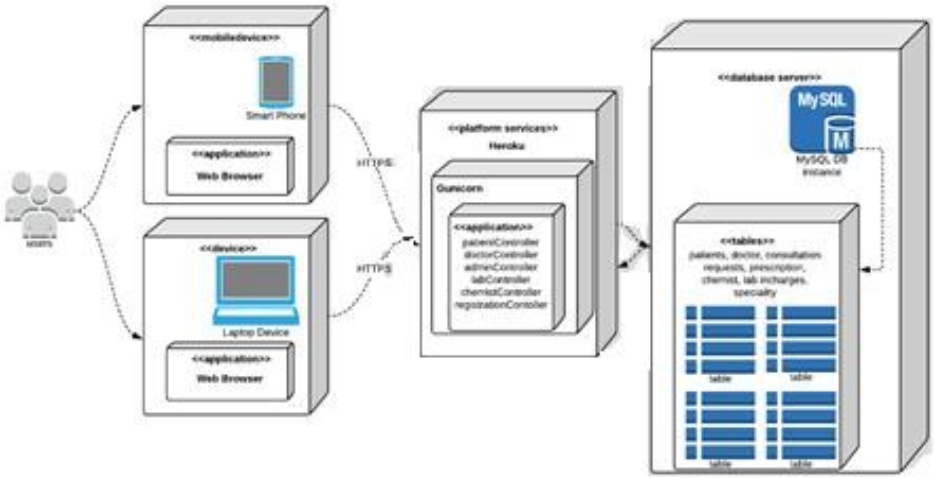


Рисунок 4.1 – Діаграма компонентів програмного забезпечення

Розроблена система повинна бути надійною, відмовостійкою і виконувати всі покладені на неї завдання. У разі помилки необхідно надати спеціальну інформацію про цього хакера. Система відповідає за правильну, швидку та ефективну обробку вхідних даних і виконання всіх функцій, які містяться в них.

Система, що розробляється, буде веб-додатком, доступним з будь-якого веб-браузера. Різні групи користувачів мають різні функції. Тільки зареєстровані співкористувачі можуть отримати доступ до системи.

При використанні програми користувач буде працювати із записами та даними в базі даних. Також варто враховувати, що можуть виникнути непередбачені обставини.

Вихідні дані:

Інформація про зареєстрованих пацієнтів (ім’я, прізвище, номер телефону, адреса електронної пошти);

Інформація про лікаря (прізвище, ім'я, по батькові, телефон, адреса електронної пошти, паспортні дані);

Інформація про час заїзду та виїзду, тип та номер кімнати;

Додаткові послуги після реєстрації користувача (назва послуги, вартість)

Вихідні дані:

Інформація про бронювання користувача (дата заїзду, дата виїзду, статус бронювання, тип номера, ім'я користувача).

Незаплановані ситуації:

а) помилки введення даних;

б) надмірний час очікування відповіді чи запиту; в) втрата зв’язку.

Розробка моделей баз даних

Для зручного та ефективного керування даними спочатку необхідно визначити об’єктну модель системи та бази даних. Ця модель використовується для опису даних, які використовуватиме система.

У системі, що розробляється, важливо наступне:

- спеціалізовані.

- лікар.

- терплячий.

- Прийом пацієнтів.

- проконсультуватися.

- рецепти.

- АПТЕКА.

Перераховані вище моменти взаємопов'язані.

Сутність «спеціальності» пов’язана з сутністю «лікаря», оскільки лікарі мають спеціальність.

Суть «консультації» стосується трьох сутностей «пацієнт», «рецепт» і «рецепт», оскільки пацієнт звертається до лікаря, а лікар призначає та називає пацієнта та лікаря для консультації.

Схема бази даних наведена в додатку Г.

4.2 Інтерфейс користувача

Відкриваючи веб-додаток, усі користувачі бачать сторінку входу.

Наступні сторінки доступні для неавторизованих користувачів:

"Вхід";

«Зареєструвати нового користувача».

Для авторизованих користувачів із роллю пацієнта доступні наступні сторінки:

1) Додавайте, редагуйте та детально переглядайте власну інформацію.

2) Детальний перегляд інформації про ліки в системі.

3) Повідомте лікаря.

4) Перегляньте написані рецепти.

5) Перегляньте запис до лікаря.

Наступні сторінки доступні для авторизованих користувачів як лікарів:

1) Доповнення, зміна та детальне відображення особистої інформації.

2) Перегляньте місця, які варто відвідати.

3) Надіслати повідомлення.

4) Доповнення, редагування та уточнення висновків.

Щоб зрозуміти роботу програми, діаграма стану діаграми. 4.2.

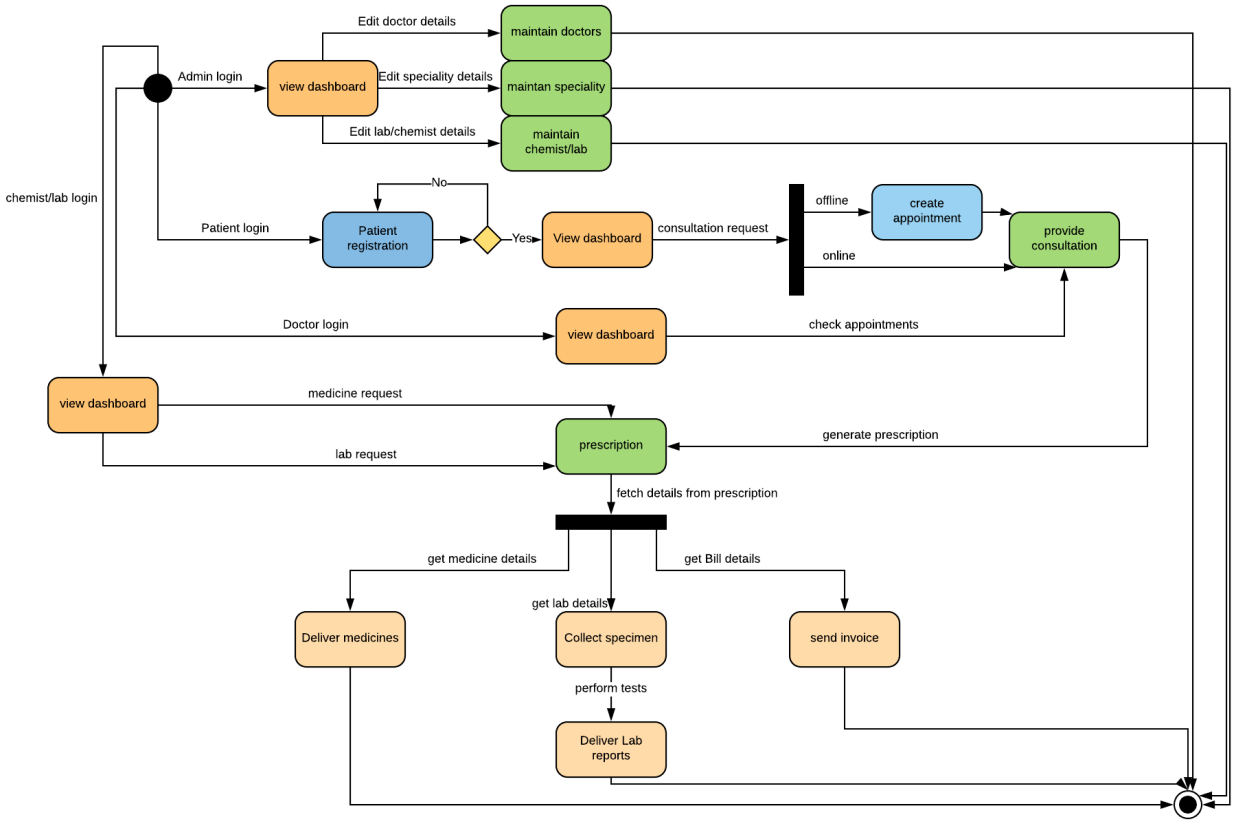


Рисунок 4.2 – Діаграма станів програмного забезпечення

Розглянемо його докладніше:

1. Сторінка «Реєстрація».

Сторінка реєстрації (рис. 4.3) вимагає створення нового облікового запису користувача, а потім авторизації, використовуючи раніше введені дані. Після заповнення необхідних полів запит виконується і викликається дія Зберегти. Також, якщо все в порядку, ці дані перевіряються на відповідність правилам перевірки - дані користувача зберігаються.

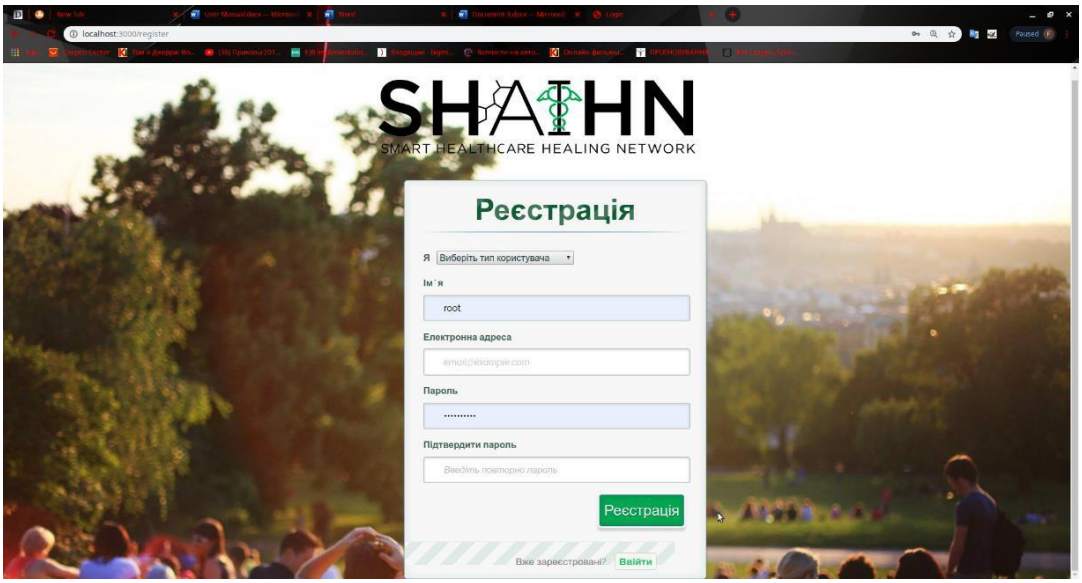


Рисунок 4.3 – Реєстрація нового користувача

2. Сторінка «Вхід».

Сторінки доступні для зареєстрованих і неавторизованих користувачів «Вхід» (рис. 4.4). Існують сторінки, доступні для різних типів користувачів відповідно до дозволів користувача.

Рисунок 4.4 – Авторизація користувачів

3. Сторінка «Домашня сторінка лікаря»

Після реєстрації в лікарні домашня сторінка доступна, як показано. 4.5.

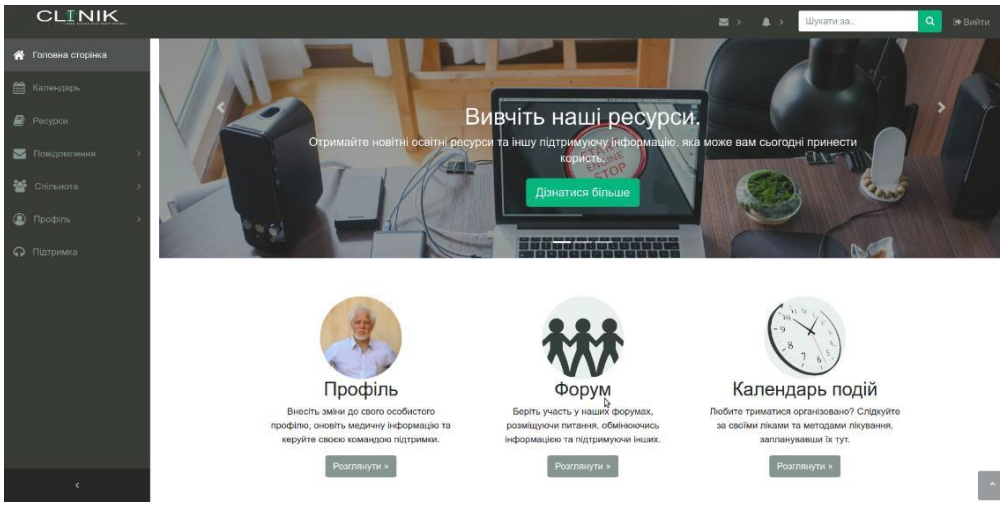


Рисунок 4.5 – Головна сторінка

4. Сторінка «Повідомлення»

При переході на вкладку «Повідомлення» стає доступною сторінка «Вхідні повідомлення», як показано. 4.6.

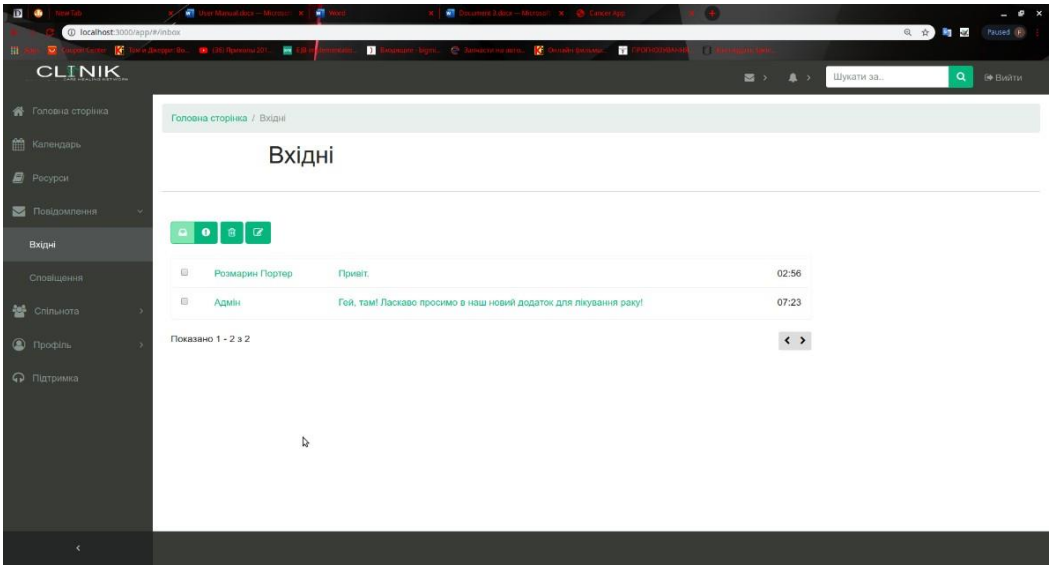


Рисунок 4.6 – Вхідні повідомлення

5. Сторінка « Календар прийому»

Коли ви переходите на вкладку «Календар» з облікового запису лікаря, стає доступною сторінка «Календар записів», де лікар може побачити запис на прийом до пацієнта, рис. 4.7.

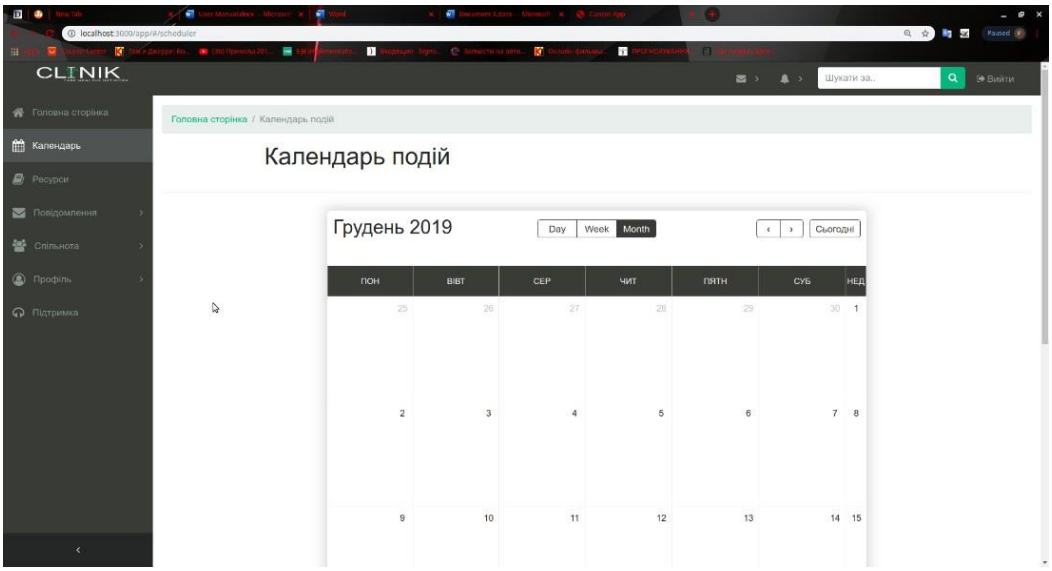


Рисунок 4.7 – Сторінка «Календар прийому»

6. Сторінка «Профіль пацієнта»

Після реєстрації пацієнт отримує доступ до сторінки «Профіль пацієнта», де може змінити свої дані. 4.8.

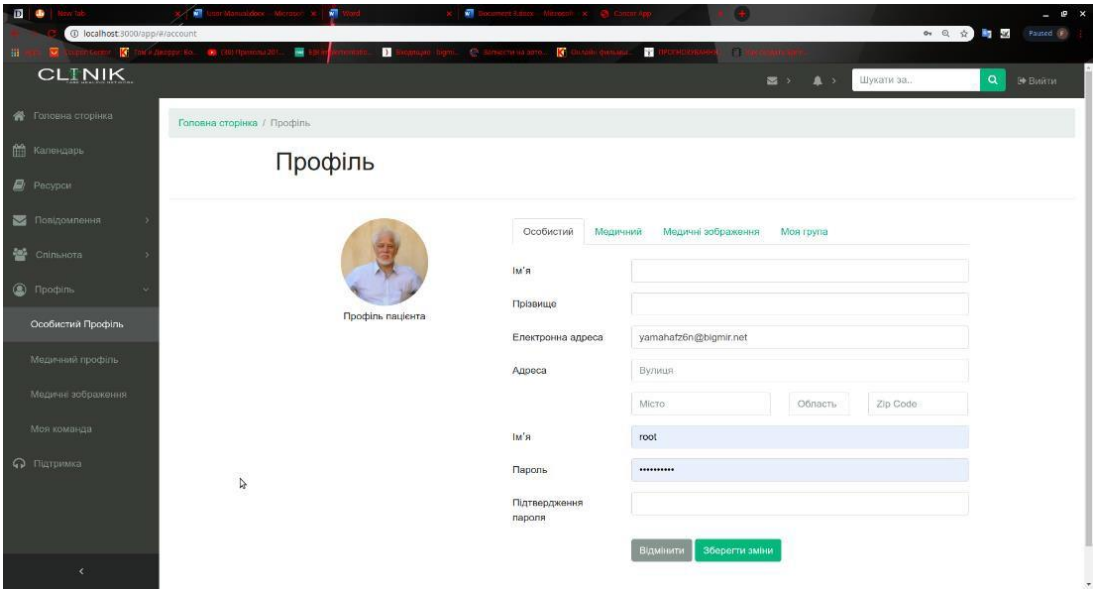


Рисунок 4.8 – Персональна сторінка пацієнта

7. Сторінка «Ресурси»

Після реєстрації пацієнт може скористатися сторінкою «Ресурси», де він може переглядати новини, пов’язані з онкологічними дослідженнями та різними подіями. 4.9.

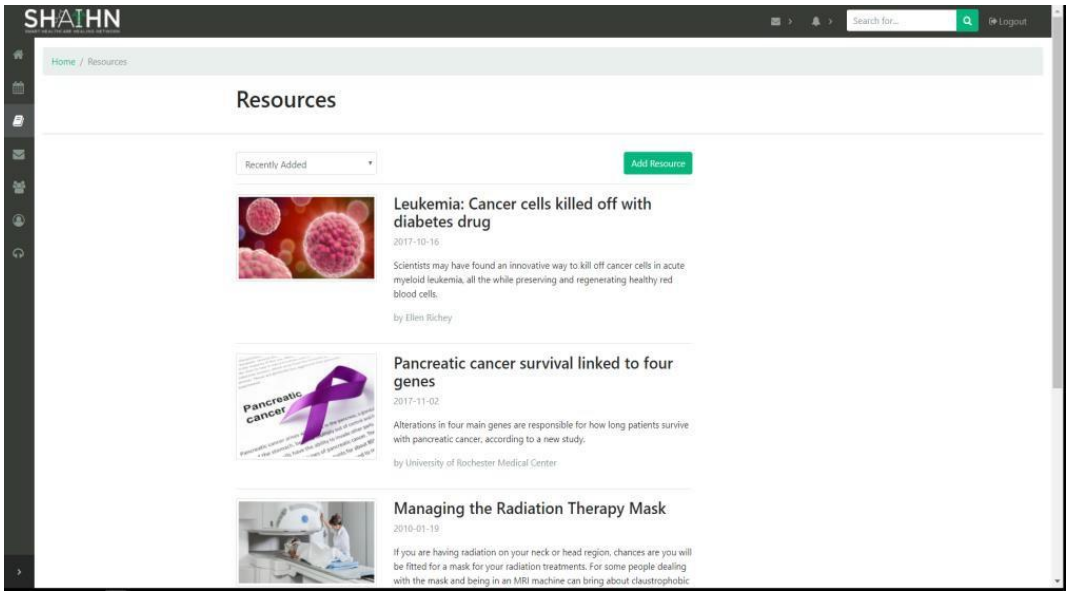


Рисунок 4.9 – Сторінка «Ресурси»

8. Сторінка «Медичний профіль пацієнта»

Після реєстрації пацієнт може скористатися сторінкою «Медичний профіль пацієнта», де він може побачити свій діагноз та характеристики для запису на прийом. 4.10.

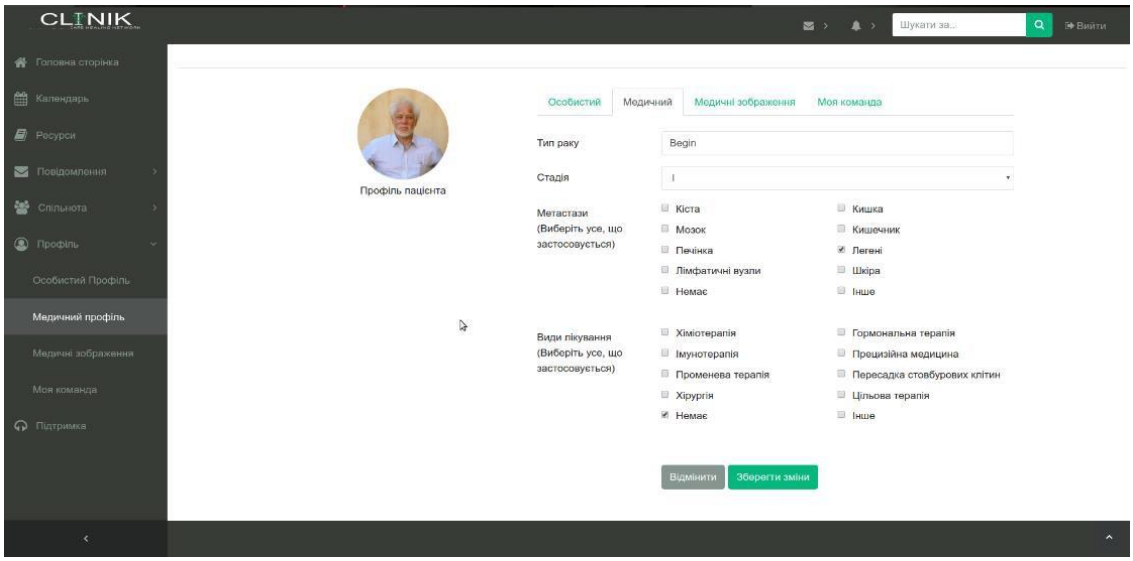


Рисунок 4.10 – Медичний профіль пацієнта

9. Сторінка « Медичні зображення пацієнта »

Якщо натиснути вкладку «Призначення пацієнта», стане доступною сторінка «Медичні зображення пацієнта», Рис. 4.11.

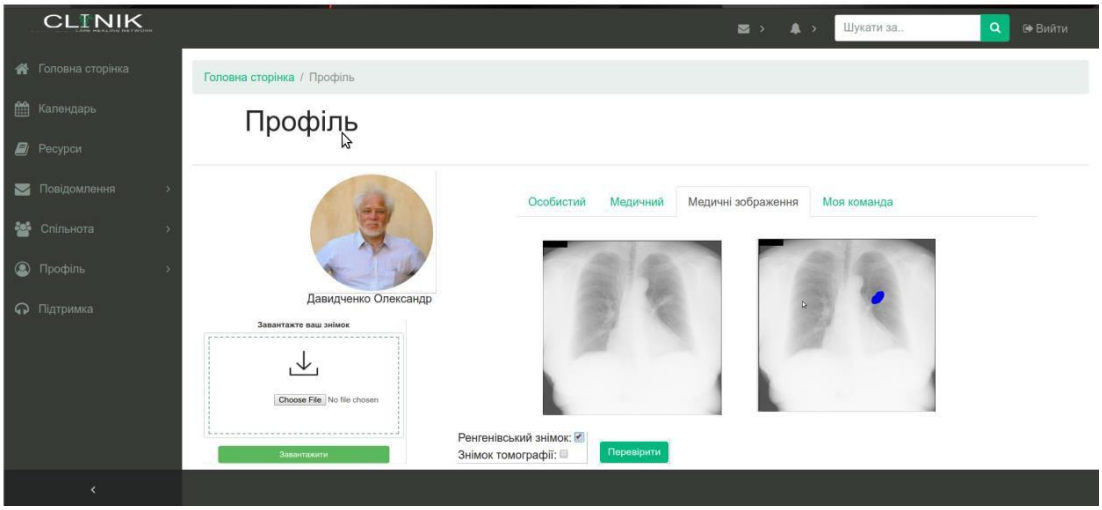


Рисунок 4.11 – Медичні зображення пацієнта

10. Сторінка « Календар прийому ліків та подій »

Перейшовши на вкладку «Календар», ви можете скористатися сторінкою «Календар ліків» і «Календар подій», де лікар може побачити пацієнта, як показано. 4.12.

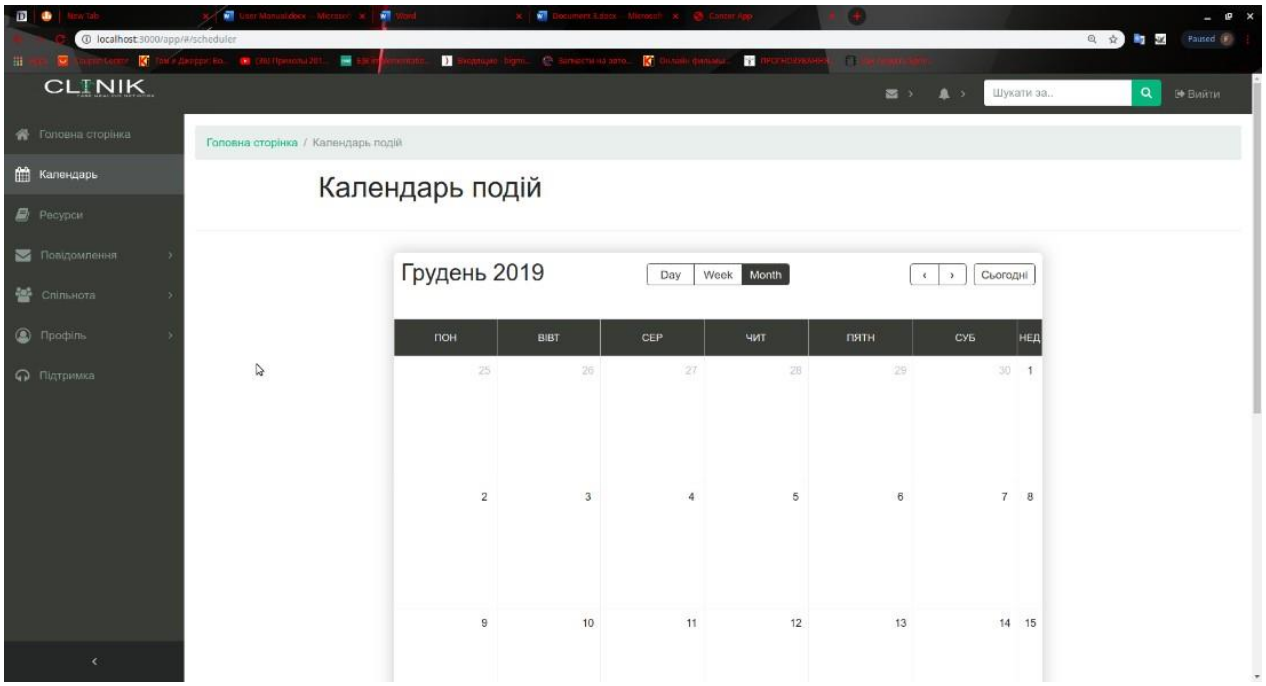


Рисунок 4.12 – Календар прийому ліків та подій.

Підсумок:

1. Проведено огляд структури розробки програмного забезпечення. Архітектура програми визначена і стає трирівневою. Надано схеми компонентів.

2. Розроблено модель бази даних.

3. Створено саму базу даних.

4. Розроблено серверну та клієнтську компоненти системи.

5. Створено і деталізовано сторінки, які можуть використовувати різні групи користувачів.

5. ЕКОНОМІЧНА ЧАСТИНА

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту, в даному розробки набору нейронних мереж для розпізнавання пневмонії на рентгенівських знімках, що застосовується в веб-сервісі медичного закладу.

Особливістю програми є те, що дана технологія застосовує концепції ансамблевих нейронних мереж для розпізнавання пневмонії на рентгенівських знімках. Звичайні аналітичні методи не дають бажаного результату для вирішення сучасних завдань розпізнавання, а методи, які використовують штучний інтелект чи не в повній мірі підходять для вирішення конкретної задачі, оскільки вони не пристосовані до розпізнавання з використанням великої кількості параметрів, чи мають недостатньо високий відсоток успішних результатів. Аналогом може бути Matlab – 37600 грн., Mathcad –28000 грн.

Для проведення комерційного та технологічного аудиту залучають не менше 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 5.1.

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Бали (за 5-ти бальною шкалою)					
Крите-рій	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах

Продовження табл. 5.1

Ринкові переваги					
2	Багато аналогів на малому ринку	Ринкові п Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практик на здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так із комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження табл. 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 – Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	3	3	4
Наявність аналогів на ринку	3	3	4
Цінова політика	4	4	4
Технічні та споживчі властивості виробу	4	3	4
Експлуатаційні витрати	3	4	3
Ринок збуту	4	3	4
Конкурентоспроможність	3	4	3
Фахівці з технічної і комерційної реалізації	4	3	4
Фінансування	4	4	3
Матеріально-технічна база	3	3	3
Термін реалізації ідеї	4	4	4
Супровідна документація	4	4	4
Сума	43	42	44
Середньоарифметична сума балів	(43+42+44) / 3 = 43		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Таблиця 5.3 - Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ , розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, так як рішення задачі розпізнавання образів є важливою складовою прийняття рішень і вирішення цієї задачі дасть змогу штучним системам набагато точніше і швидше аналізувати навколишній простір. Звичайні аналітичні методи не дають бажаного результату для вирішення сучасних завдань розпізнавання, а методи, які використовують штучний інтелект чи не в повній мірі підходять для вирішення конкретної задачі, оскільки вони не пристосовані до розпізнавання з використанням великої кількості параметрів, чи мають недостатньо високий відсоток успішних результатів.

5.2 Прогнозування витрат на виконання науково-дослідної (дослідно-конструкторської) роботи

5.2.1 Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \,, \tag{5.1}$$

де М – місячний посадовий оклад конкретного розробника (дослідника), грн.;

$Т_р$ – число робочих днів в місяці, 23 днів;

t – число днів роботи розробника (дослідника).

Результати розрахунків зведемо до таблиці 5.1.

Таблиця 5.2.1 – Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	42000	1826,09	38	69391,304
Програміст	39000	1695,65	38	64434,783
Тестувальник	31000	1347,83	38	51217,391
Всього				185043,48

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

5.2.2 Додаткова заробітна плата розробників, які приймали участь в розробці обладнання.

Додаткова заробітна плата прийнято розраховувати як 12 % від основної заробітної плати розробників та робітників:

$$З_д = З_о \cdot 12\ \% / 100\ \% \qquad (5.2)$$

$З_д = (185043{,}48 \cdot 12\ \% / 100\ \%) = 22205{,}22$ (грн.)

5.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Н_з = (З_о + З_д) \cdot 22\ \% / 100\% \qquad (5.3)$$

$Н_з = (185043{,}48 + 22205{,}22) \cdot 22\ \% / 100\ \% = 45594{,}71$ (грн.)

5.2.4. Оскільки для розроблювального пристрою не потрібно витрачати матеріали та комплектуючи, то витрати на матеріали і комплектуючі дорівнюють нулю.

5.2.5 Амортизація обладнання, яке використовувалось для проведення розробки.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді амортизація обладнання, що використовувалась для розробки розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12} \text{ [грн.].} \qquad (5.4)$$

де Ц – балансова вартість обладнання, грн.;

Т – термін корисного використання обладнання згідно податкового законодавства, років

$t_{вик}$ – термін використання під час розробки, місяців

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 20000 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,65 міс.

$$A_{обл} = \frac{20000}{2} \times \frac{1{,}65}{12} = 1376{,}81\ грн.$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 5.2. Для розрахунку амортизації нематеріальних ресурсів використовується формула:

$$A_{н.р.} = Ц_{н.р.} * Н_а * \frac{t_{вик.}}{12} \quad (5.5)$$

Але, так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $B_{нем.ак.}$ = 8470 грн.

Таблиця 5.2 – Амортизаційні відрахування матеріальних і нематеріальних ресурсів для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна переферія	20000	2	1,65	1376,812
Офісне обладнення	21500	4	1,65	740,036
Приміщення	980000	20	1,65	6746,377
Всього				8863,22

5.2.6 Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.6)$$

де В – вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 6,2 грн./кВт;

П – встановлена потужність обладнання, кВт. П = 0,6 кВт;

Ф – фактична кількість годин роботи обладнання, годин.

$К_п$ – коефіцієнт використання потужності, $К_п$ = 0,9.

$В_е = 0,9 \cdot 0,6 \cdot 8 \cdot 38 \cdot 6,2 = 1017,792$ (грн.)

5.2.7 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.7)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$I_в = 185043,48 * 92\% / 100\% = 170240$ (грн.)

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$H_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.8)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$H_{нзв}$ = 185043,48 * 111 % / 100 % = 205398 (грн.)

5.2.9 Витрати на проведення науково-дослідної роботи.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$B_{заг}$ = 185043,48+22205,22+45594,71+8863,22+8470+1017,79+170240+

+205398 = 646832,69 грн.

5.2.11 Розрахунок загальних витрат на науково-дослідну (науково-технічну) роботу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються *ЗВ*, визначається за формулою:

$$ЗВ = \frac{B_{заг}}{\eta} \text{ (грн)}, \quad (5.9)$$

де η – коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то

η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$ЗВ$ = 646832,69 / 0,5 = 1293665 грн.

5.3 Розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

а) вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

б) зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

в) кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

г) визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання

прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

- абсолютного економічного ефекту (чистого дисконтованого доходу);
- внутрішньої економічної дохідності (внутрішньої норми дохідності);
- терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

5.3.1 Розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.10)$$

де $\pm\Delta Ц_o$ – зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Цб$ – вартість програмного продукту у році до впровадження результатів розробки;

ΔN – збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт $\lambda = 0{,}8333$.

p – коефіцієнт, який враховує рентабельність продукту;

ϑ – ставка податку на прибуток, у 2022 році ϑ =18%.

Припустимо, що при прогнозованій ціні 12000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Після завершення розробки і її вдосконалення, можна буде підняти її ціну на *1000* грн. Кількість одиниць реалізованої продукції також збільшиться (кількість державник лікарень в Україні у 2022 році 1186 закладів, кількість приватник клінік – близько 35000 закладів): протягом першого року – на 1500 шт., протягом другого року – на 1100 шт., протягом третього року на 900 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

*$\Delta П_1$ = (0*1000 + (*12000 + *1000)*1500)* 0,8333* 0,25) * (1 - 0,18) = 3074999,877 грн.*

*$\Delta П_2$ = (0*1000 + (*12000 + *1000)*(1500+1100)* 0,8333* 0,25) * (1 - 0,18) = 5774166,436 грн.*

*$\Delta П_3$ = (0*1000 + (*12000 + *1000)*(1500+1100+900)* 0,8333* 0,25) * (1 - 0,18) = 7772916,356 грн.*

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 16622082,67 грн.

5.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності.

Розраховуємо приведену вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т – період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0,05...0,15$;

t – період часу (в роках).

Збільшення прибутку ми отримаємо починаючи з першого року:

ПП= $(3074999,877/(1+0,1)^1)+(5774166,436/(1+0,1)^2)+(7772916,356/(1+0,1)^3)$ = 2795454,43 + 4772038,377 + 5839907,104 = 13407399,91 грн.

Далі розраховують величину початкових інвестицій *PV*, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.12)$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}=2...5$, але може бути і більшим;

ЗВ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$PV = 2 * 1293665 = 2587330,74$ *грн.*

Тоді абсолютний економічний ефект $E_{абс}$ або чистий приведений дохід *(NPV, Net Present Value)* для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV, \quad (5.13)$$

$E_{абс}$ = 13407399,91 – 2587330,74 = 10820069,17 грн.

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розра-хувати внутрішню економічну дохідність або показник внутрішньої но-рми дохідності *(IRR, Internal Rate of Return)* вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.14}$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$E_в = \sqrt[3]{(1 + 10820069,17/2587330,74} - 1 = 0,730$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.15}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d = (0,09…0,14);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{\min} = 0,14 + 0,05 = 0,19.$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \qquad (5.16)$$

$T_{ок}$ = 1 / 0,730 = 1,37 р.

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 1,37 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку нового програмного продукту, сума яких складає 1293665 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. В результаті аналізу розрахунків можна зробити висновок, що розроблений програмний продукт за ціною дешевший за аналог і є висококонкурентоспроможним. Період окупності складе близько 1,37 роки.

ВИСНОВКИ

1. Аналіз існуючих методів вирішення задач розпізнавання зображень визначає вибір штучних нейронних мереж для вирішення конкретної задачі.

2. Розглянути основні поняття, класифікації штучних нейронних мереж, класифікацію основних методів навчання.

3. Обговорюється базова концепція ансамблю нейронної мережі, надається метод навчання та базова структура.

4. Дано огляд існуючих методів побудови нейромережевих ансамблів. Особливу увагу приділено використанню алгоритмів помилок для формування множин.

5. Для підвищення точності роботи запропоновано метод модифікації існуючих алгоритмів для формування ансамблю нейронних мереж.

6. ЕКСПЕРИМЕНТИ Експерименти проводились з використанням розробленого коду, що реалізує запропоновану модель системи. Експериментальні результати показують, що нова структура має вищу точність у розв’язуванні задач і не залежить від навчальних зразків.

7. Розроблено програмне забезпечення, яке використовувало створений комплекс для ідентифікації пневмонії на рентгенограмах.

8. В рамках розробки стартап проекту було проведено опитування щодо основних аспектів виходу на ринок додатків для визначення діагностики пацієнтів. Продукт підходить для користувачів, які хочуть поставити остаточний діагноз за короткий проміжок часу через його здатність виявляти пухлини. Визначте перелік слабких, сильних і нейтральних характеристик і атрибутів потенційного продукту для формування його конкурентоспроможності.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Агєєв А. Д., Балухто А. Н., Бичков А. В., Верещагін С. А. та ін нейроматема- тика. Книга 6: навчальний посібник для вузів. - М .: ІПРЖР, 2002. - 448 с.

2. Барський, А. Б. Нейронні мережі: розпізнавання, управління, прийняття рішень. - М .: Фінанси і статистика, 2004. - 176 с.

3. Хайкін С. Нейронні мережі. Повний курс. Друге видання. - М .: Вільямс, 2006. - 1104 с.

4. Саттон Р.С. Навчання з підкріпленням. - М .: БИНОМ. Лабораторія знання, 2012. - 399 с

5. Harris Drucker, Yann Le Cun. Improving Generalization Performance Using Back- propagation / Harris Drucker, Yann Le Cun // IEEE Transactions on Neural Net- works. - 1992. - Vol.3, №5 - p.991-997.

6. Боровиков, В. П. Мистецтво аналізу даних, 2-е видання - СПБ .: Пітер, 2005. - 412 с.

7. Васенко, Д. В. Методи навчання штучних нейронних мереж. Інформатизація освіти, 2007. - С. 20-29.

8. Васильєв, В. І. розпізнати системи. Довідник. 2-е видання - Київ: Наукова думка, 1983. - 424 с.

9. Воронцов, К. В. Комбінаторні оцінки якості навчання по прецедентах // До- повіді РАН. - 2004. - Т. 394. - №2. - С. 175-178.

10. Галушкин, А. І. Нейрокомп'ютери і їх застосування: навчальний посібник для вузів. Книга 1 - Теорія нейронних мереж / А. І. Галушкин. - М .: ІПРЖР, 2000. - 416 с.

11. Гонсалес, Р. Цифрова обробка зображень / Р. Гонсалес, Р. Вудс. - М .: Техно- сфера, 2005. - 1007 с.

12. Горбань, А. Н. Нейроінформатика / А. Н. Горбань, В. Л. ДунінБарковскій, А. Н. Кірдіна. - Новосибірськ: Наука. Сибірське підприємство РАН, 1998. - 296 с.

13. Желтов, С. Ю. Обробка та аналіз зображень в задачах машинного зору / С. Ю. Желтов. - М .: Фізматкніга, 2010. - 672 с.

14. Заєнцев, І. В. Нейронні мережі. Основні моделі / І. В. Заєнцев. - Воронеж: ВДУ, 1999. - 76 с.

15. Комарцова, Л. Г. Нейрокомп'ютери: навчальний посібник / Л. Г. Комарцова, А. В. Максимов. - М .: МГТУ, 2002. - 318 с.

16. Корн, Г. Довідник з математики для науковців та інженерів / Г. Корн, Т. Корн. - М .: Наука, 1970. - 246 с.

17. Крісілов, В. А. Методи прискорення навчання нейронних мереж / В. А. Крісілов, Д. Н. Олешко. - М .: Гардарика, 2005. - 1042 с.

18. Круглов, В. В. Штучні нейронні мережі. Теорія та практика. - 2-е видання / В. В. Круглов, В. В. Борисов. - М .: Гаряча лінія - Телеком, 2002. - 382 с.

19. Рутковська, Д. Нейронні мережі, генетичні алгоритми та нечіткі системи / Д. Рутковська, М. Піліньскій, Л. Рутковський. - М .: Гаряча лінія. - Телеком, 2006. - 147 с.

20. Сергієнко, А. Б. Цифрова обробка сигналів: підручник для вузів / А. Б. Серги- енко. - СПб .: Пітер, 2002. - 608 с.

21. Сойфер, В. А. Методи комп'ютерної обробки збережений / В. А. Сойфер. - М.: Фізматліт, 2003. - 459 с.

22. Ту, Дж. Принципи розпізнавання образів / Дж. Ту, Р. Гонсалес. - М .: Світ, 1978. - 412 с.

23. Форсайт, Д. А. Комп'ютерне зір. Сучасний похід / Д. А. Форсайт, П. Джин. - М .: Вільямс, 2004. - 928 с.

Додаток А
(обов'язковий)

ВНТУ

ЗАТВЕРДЖЕНО
Зав. кафедри КСУ ВНТУ,
д.т.н., доцент
_____ _____В'ячеслав КОВТУН
" 30 " _жовтня_ 2022 р.

ТЕХНІЧНЕ ЗАВДАННЯ

На виконання магістерської кваліфікаційної роботи

Нейромережевий ансамбль для розпізнавання зображень як елемент людино-машинної взаємодії

(тема)

08-33.МКР.010.00.000 ТЗ
номер

Студент групи __2АКІТ-21м_

_Антон ШАМАТІЄНКО__
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник _к.т.н., доцент, доцент кафедри КСУ_

__Олена НИКИТЕНКО_
Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2022

1. Назва та галузь застосування

1.1. Назва – Нейромережевий ансамбль для розпізнавання зображень як елемент людино-машинної взаємодії.

1.2. Галузь застосування – розпізнавання образів.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від “14” вересня 2022 року №203

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є вдосконалення існуючих методів розпізнавання шляхом застосування концепції ансамблевих нейронних мереж.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1 Агєєв А. Д., Балухто А. Н., Бичков А. В., Верещагін С. А. та ін нейроматема- тика. Книга 6: навчальний посібник для вузів. - М .: ІПРЖР, 2002. - 448 с.

2 Галушкин, А. І. Нейрокомп'ютери і їх застосування: навчальний посібник для вузів. Книга 1 - Теорія нейронних мереж / А. І. Галушкин. - М .: ІПРЖР, 2000. - 416 с.

3 Круглов, В. В. Штучні нейронні мережі. Теорія та практика. - 2-е видання / В. В. Круглов, В. В. Борисов. - М .: Гаряча лінія - Телеком, 2002. - 382 с.

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- розпізнавання зображень;
- відображення особистої інформації;
- формування результату

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 10;
- Visual Studio 11.

5.2.2. Умови експлуатації системи:

- робота на веб-додатках;
- редагування і перегляд даних;
- декілька користувачів системи.

6. Стадії та етапи розробки.

6.1 Пояснювальна записка:

1. Детальне дослідження існуючої науково-технічної літератури по темі нейронних мереж та ансамблів нейронних мереж. Постановка задач дослідження «03»_09__ 2022 р.
2. За комплексним критерієм визначення найкращих топологій мереж, які входять в ансамбль для вирішення задачі розпізнавання «10»__11_ 2022 р.
3. Проектування ансамблю нейронних мереж «20»_11__ 2022 р.
4. Розробка програмного забезпечення системи «27»_11__ 2022 р.

6.2 Графічні матеріали:

1. Розробка UML-діаграм системи «12»_10__ 2022 р.
2. Розробка моделі функціонування системи «25»_10__ 2022 р.
3. Тестування програмного забезпечення «15»_11__ 2022 р.

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «28»__11_ 2022 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «16»__12_ 2022 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «23»__12_ 2022 р.

Додаток Б

ПРОТОКОЛ
ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ
НА НАЯВНІСТЬ ТЕКСТОВИХ ЗАПОЗИЧЕНЬ

Назва роботи: _«Нейромережевий ансамбль для розпізнавання зображень як елемент людино-машинної взаємодії»_____

Тип роботи: _____Магістерська кваліфікаційна робота_____
(БДР, МКР)

Підрозділ _____КСУ, ФІТА_____
(кафедра, факультет)

Показники звіту подібності Unicheck

Оригінальність ___88,4%_____ Схожість___11,6%_____

Аналіз звіту подібності (відмітити потрібне)

☐ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.
☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на розгляд експертної комісії кафедри.
☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірк_____ ____Галущак А.В._____
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unichek щодо роботи.

Автор роботи ____ _____ __Шаматієнко А.В. _____
(підпис) (прізвище, ініціали)

Керівник роботи _____ _____Никитенко О.Д._____
(підпис) (прізвище, ініціали)

Додаток В
Лістинг програми

Файл NeuralNetwork

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.IO;


namespace iTellUtils
{
public class Vertex
{
public List<Vertex> next; public List<double> wnext; public double delta; public double value; public double threshold;


public Vertex()
{
next = new List<Vertex>(); wnext = new List<double>();
Random rand = new Random((int)DateTime.Now.Ticks); threshold = rand.NextDouble() * 0.4;
delta = 0.0;
value = 0.0;
}
}

public class Layer
{
public List<Vertex> vertex; public Layer(int size)
{
vertex = new List<Vertex>(); for (int i = 0; i < size; ++i)
vertex.Add(new Vertex());
}
}


public class NeuralNetworkException: Exception
{
public NeuralNetworkException(string message)
: base(message) { }
}
```

```
public class NeuralNetwork
{
private const double speed = 0.05; private List<Layer> layers;
static public NeuralNetwork Current = null; public int getDesision() {
double best = -1e100;
int bestv = -1; double[] x = Output();

for(int i=0; i<x.Length; ++i)
{
if (x[i] > best)
{
best = x[i]; bestv = i;
}
}
return bestv;
}
public double[] Output()
{
double[] result = new double[layers[layers.Count - 1].vertex.Count];



for(int i=0; i<result.Length; ++i)
{
result[i] = layers[layers.Count - 1].vertex[i].value;
}
return result;
}

public void Save(string filename)
{
StreamWriter sw = null; try
{
sw = new StreamWriter(filename);


sw.WriteLine(layers.Count);
for (int i = 0; i < layers.Count; ++i) sw.Write(layers[i].vertex.Count.ToString() + " ");
sw.WriteLine();


for (int i = 0; i < layers.Count; ++i)
```

```
for (int j = 0; j < layers[i].vertex.Count; ++j)
{
Vertex v = layers[i].vertex[j];
sw.Write(v.threshold.ToString() + " " + v.next.Count.ToString() + " "); for (int k = 0; k < v.next.Count; ++k)
{
sw.Write(v.wnext[k]);
sw.Write(" ");
}
sw.WriteLine();
}
}
catch (Exception e)
{
throw new NeuralNetworkException(e.Message);
}
finally
{

if (sw != null)
{
sw.Flush();
sw.Close();
}
}
}



public void Load(string filename)
{
StreamReader sw = null; try
{
sw = new StreamReader(filename); string s = sw.ReadToEnd();
string[] _ss = s.Split(' ', '\n', '\r'); List<string> ss = new List<string>(); for (int i = 0; i < _ss.Length; ++i)
if (_ss[i].Length > 0) ss.Add(_ss[i]);

int p = 0;
int n = int.Parse(ss[p++]); int[] size = new int[n];
for (int i = 0; i < n; ++i) size[i] = int.Parse(ss[p++]);



layers = new List<Layer>(); for (int i = 0; i < n; ++i)
layers.Add(new Layer(size[i]));
```

```
for (int i = 0; i < n - 1; ++i)
for (int j = 0; j < layers[i].vertex.Count; ++j)
for (int k = 0; k < layers[i + 1].vertex.Count; ++k) layers[i].vertex[j].next.Add(layers[i + 1].vertex[k]);


for (int i = 0; i < n; ++i)


{
for (int j = 0; j < size[i]; ++j)
{
Vertex v = layers[i].vertex[j]; v.threshold = double.Parse(ss[p++]); int m = int.Parse(ss[p++]);
for (int k = 0; k < m; ++k) v.wnext.Add(double.Parse(ss[p++]));
}
}
}
catch (Exception e)
{
throw new NeuralNetworkException(e.Message);
}
finally
{

if (sw != null) sw.Close();
}
}

public NeuralNetwork(string filename)
{
Load(filename);
}

public NeuralNetwork(int[] sizes)
{
Random rand = new Random((int)DateTime.Now.Ticks); layers = new List<Layer>();
for (int i = 0; i < sizes.Length; ++i) layers.Add(new Layer(sizes[i]));


for (int i = 0; i < sizes.Length - 1; ++i)
{
for (int j = 0; j < layers[i].vertex.Count; ++j)
```

```
for (int k = 0; k < layers[i + 1].vertex.Count; ++k)
{
layers[i].vertex[j].next.Add(layers[i + 1].vertex[k]); layers[i].vertex[j].wnext.Add(rand.NextDouble() * 0.4 - 0.2);
}
}
}

public void clear()
{
for (int i = 0; i < layers.Count; ++i)
for (int j = 0; j < layers[i].vertex.Count; ++j) layers[i].vertex[j].value = 0.0;

}

private double sigmoida(double x)
{
return 1.0 / (1.0 + Math.Exp(-x));
}

public int InputLayerSize { get { return layers[0].vertex.Count; } } public void Run(double[] input)
{
clear();
if (input.Length != layers[0].vertex.Count) throw new Exception("Input layer and input has different number of items");


// init first layer
for (int i = 0; i < input.Length; ++i)
{
layers[0].vertex[i].value = input[i];
for (int j = 0; j < layers[0].vertex[i].next.Count; ++j) layers[0].vertex[i].next[j].value += input[i] *
layers[0].vertex[i].wnext[j];
}

// next layers
for (int t = 1; t < layers.Count; ++t)
for (int i = 0; i < layers[t].vertex.Count; ++i)
{


Vertex v = layers[t].vertex[i];
v.value = sigmoida(v.value - v.threshold); for (int j = 0; j < v.next.Count; ++j)
v.next[j].value += v.value * v.wnext[j];
```

```
}
}

public void Learn(double[] input, double[] need)
{
if (need.Length != layers[layers.Count - 1].vertex.Count)
throw new Exception("Test is something different from expected"); Run(input);
// last layer:
for (int i = 0; i < layers[layers.Count - 1].vertex.Count; ++i)
{
Vertex v = layers[layers.Count - 1].vertex[i];
v.delta = v.value * (1.0 - v.value) * (need[i] - v.value);
}

// other layers:
for (int t = layers.Count - 2; t >= 0; --t) foreach (Vertex v in layers[t].vertex)
{
v.delta = 0.0;
for (int i = 0; i < v.next.Count; ++i) v.delta += v.next[i].delta * v.wnext[i];
v.delta *= v.value * (1.0 - v.value);
}


// learning:

for (int t = layers.Count - 1; t >= 0; --t)
for (int i = 0; i < layers[t].vertex.Count; ++i)
{
Vertex v = layers[t].vertex[i];
for (int j = 0; j < v.next.Count; ++j)
{
Vertex y = v.next[j];
v.wnext[j] += speed * y.delta * v.value;
}
}
}
}
}
```

Файл WordDB

```
using System;
```

```
using System.Collections.Generic; using System.Linq;
using System.Text; using System.IO;
using System.Runtime.Serialization.Formatters.Binary; using System.Runtime.Serialization;

namespace iTellUtils
{

public class WordDBCantSaveException: Exception
{
public WordDBCantSaveException(string s):base(s) { }
}
public class WordDBCantLoadException : Exception
{
public WordDBCantLoadException(string s) : base(s) { }
}

[Serializable] public class WordDB
{
static private WordDB instance = null;
static public WordDB Instance { get { if (instance == null) instance = new WordDB(); return instance; } }

private SortedSet<string> words = new SortedSet<string>();

private double getProb(string s, FuzzyVector v) { if (s.Length != v.chars.Count) return 0.0; double prob = 1.0;
for(int i=0; i<s.Length; ++i)
prob *= v.chars[i][(int)(s[i]-'a')]; return prob;
}


public int WordsCount { get { return words.Count; } } private bool checkPref(string s1, string s2)
{
if (s1.Length<s2.Length) return false;
int i = 0; while (i < s2.Length) if (s1[i] != s2[i]) return false; else ++i; return true;
}
public string[] getWords(string pref)
{
string[] w = words.ToArray(); List<string> list = new List<string>(); for (int i = 0; i < w.Length; ++i)
{
if (checkPref(w[i], pref)) list.Add(w[i]);
}
return list.ToArray();
}
```

```
public void LoadFromTextFile(string filename)
{

try
{

words.Clear();
string[] w = File.ReadAllLines(filename);
for (int i = 0; i < w.Length; ++i) if (w[i].Length > 0)
{


AddWord(w[i]);
}
}
catch (Exception e)
{
throw new WordDBCantLoadException(e.Message);
}
}
public void SaveToTextFile(string filename)
{


try
{


}

string[] s = words.ToArray(); File.WriteAllLines(filename, s);

catch (Exception e)
{
throw new WordDBCantSaveException(e.Message);
}
}

public WordDB() { }
public WordDB(string[] words)
{
```

```
for (int i = 0; i < words.Length; ++i) this.words.Add(words[i]);
}
public void AddWord(string s)
{
string ss="";
for (int i = 0; i < s.Length; ++i) ss += Char.ToLower(s[i]); if (words.Contains(ss)) return;
words.Add(ss);
}
public void RemoveWord(string s)
{
if (!words.Contains(s)) return; words.Remove(s);
}


public WordDBSearchResult findBestWords(FuzzyVector word)
{


WordDBSearchResult res = new WordDBSearchResult(10); string[] W = getWords("");
for (int i = 0; i < W.Length; ++i)
{
res.AddWord(W[i], getProb(W[i], word));
}
return res;
}
}

public class WordDBSearchResult
{
private List<double> prob; private List<string> word;
public WordDBSearchResult(int limit) { prob = new List<double>();
word = new List<string>();
}

public void AddWord(string word, double prob)
{
this.prob.Add(prob); this.word.Add(word);
}
public int Count()
{return 0;}

public string GetFirstWord()
```

```
{
double best = 0.0; string res = "";
for (int i=0; i < prob.Count; ++i)
{
if (prob[i] > best)
{
best = prob[i]; res = word[i];
}
}
return res;
}
public string GetWordAt(int index)
{ return null; }
public double GetProbAt(int index)
{ return 0.0; }
}

public class FuzzyVector
{
public List<double[]> chars = new List<double[]>();
}
}
```

Файл BlockRecognizer

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.Drawing;


namespace iTellUtils
{
public class BlockRecognizer
{
static private int Width; static private int Height; static int[,] bits = null;

static public bool Loaded() { return (bits != null); } static public void StartImage(int [,] img, int W, int H)
{
Width = W; Height = H; bits = img; line = 0;
column = 0;

}
```

```
static public void Reset()
{
line = column = 0;
}

public static int dx0 = 2; public static int dy0 = 0; public static int dx = 7; public static int dy = 16;

static public int line; static private int column;

static public Bitmap NextBitmapChar()
{
if ((line+1)*dy + dy0 > Height) return null; int y = line * dy + dy0;
int x = column * dx + dx0; if (x + dx > Width) {
++line; column = 0;
return NextBitmapChar();
}
Bitmap res = new Bitmap(dx, dy); int sum = 0;
for (int i = 0; i < dx; ++i) for (int j = 0; j < dy; ++j) {
//res.SetPixel(i, j, Color.Red); sum += bits[x + i, y + j];
res.SetPixel(i, j, Color.FromArgb(255 - bits[x + i, y + j], 255 - bits[x + i, y + j], 255
- bits[x + i, y + j]));
}
++column; return res;
}


static public double[] NextDoubleChar()
{
if ((line + 1) * dy + dy0 > Height) return null; int y = line * dy + dy0;
int x = column * dx + dx0; if (x + dx > Width)
{
++line; column = 0;
return NextDoubleChar();
}
double[] res = new double[dx*dy]; int sum = 0;
for (int j = 0; j < dy; ++j) for (int i = 0; i < dx; ++i)
{
res[sum++]= (double)bits[x + i, y + j]/255.0;
}
++column; return res;
}
```

```
}
}
```

Файл ImageFilter

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using System.Drawing;


namespace iTellUtils
{
public class ImageFilter
{
static int ColorToInt(Color v)
{
return Math.Max(v.B,Math.Max(v.G,v.R));
}
static Color IntToColor(int x) {


return Color.FromArgb(x, x, x);
}

public static Bitmap DoFilteringSimple(Bitmap source) { Bitmap result = (Bitmap)source.Clone();
for (int x = 0; x < result.Width; ++x)
for (int y = 0; y < result.Height; ++y)
{
result.SetPixel(x,y,IntToColor(ColorToInt(source.GetPixel(x, y))));
}
return result;
}

public static Bitmap DoFiltering(Bitmap source)
{
Bitmap result = new Bitmap(source.Width, source.Height); int[] pixels = new int[8];
int[] dx = {-1,-1,-1,0,1,1, 1, 0};
int[] dy = {-1, 0, 1,1,1,0,-1,-1};
for (int x = 0; x < result.Width; ++x)
{
for (int y = 0; y < result.Height; ++y)
{
```

```
if (x == 317 && y == 106) { Console.WriteLine();
}
int n = 0;
for(int i=0; i<8; ++i)
{
int xx=x+dx[i]; int yy=y+dy[i];
if (xx<0 || yy<0 || xx>=source.Width || yy>=source.Height) continue; pixels[n++] = ColorToInt(source.GetPixel(xx,yy));
}
for(int i=0; i<n-1; ++i) for(int j=0; j<n-1; ++j)
if (pixels[j]>pixels[j+1])
{

}

if (n<8) {

pixels[j]^=pixels[j+1]; pixels[j+1]^=pixels[j]; pixels[j]^=pixels[j+1];

result.SetPixel(x,y,IntToColor(ColorToInt(source.GetPixel(x,y))));
}
else
{
int X = ColorToInt(source.GetPixel(x,y));
double med = (double)(pixels[3]+pixels[4])/2.0d; double r1 = ((double)X<=med)?pixels[0]-X:X-pixels[7]; double r2 = ((double)X<=med)?pixels[1]-X:X-pixels[6]; double r3 = ((double)X<=med)?pixels[2]-X:X-pixels[5]; double r4 = ((double)X<=med)?pixels[3]-X:X-pixels[4];

if (r1>8.0 || r2>20.0 || r3>40.0 || r4>80.0)
{

}
else
{

}
}
}
}

result.SetPixel(x,y,IntToColor( (int)(1.0*med + 0.0*X + 0.5) ));
```

```
result.SetPixel(x,y,IntToColor( (int)(0.0*med + 1.0*X + 0.5) ));

return result;
}
}
}
```

Файл Recognizer

```
using System;
using System.Collections.Generic; using System.Linq;
using System.Text; using iTellUtils;


using System.Drawing;
using System.Drawing.Imaging;

namespace iTell
{
public class Recognizer
{
static public List<double[]> getData(int[][] images)
{
List<double[]> res = new List<double[]>(); for (int i = 0; i < images.Length; ++i)
{
double[] a = new double[NeuralNetwork.Current.InputLayerSize]; for (int r = 0; r < images[i].Length; ++r)
a[r] = (double)images[i][r] / 256.0; res.Add(a);
}
return res;
}


static public FuzzyVector GetVector(List<double[]> data)
{
FuzzyVector vect = new FuzzyVector(); for (int i = 0; i < data.Count; ++i) {
NeuralNetwork.Current.Run(data[i]);  double[] v = NeuralNetwork.Current.Output(); double sum = 0.0;
for (int j = 0; j < 26; ++j) sum += v[j]; for (int j = 0; j < 26; ++j) v[j] /= sum; vect.chars.Add(v);
}
return vect;
}

static public FuzzyVector GetVector(string word)
```

```
{
FuzzyVector vect = new FuzzyVector(); for (int i = 0; i < word.Length; ++i)
{
double[] y = new double[26]; for (int j = 0; j < 26; ++j)
y[j]=((int)(word[i]-'a')==j)?1.0:0.0;

vect.chars.Add(y);
}
return vect;
}
}
}
```

Файл MainForm

```
using System;
using System.Collections.Generic; using System.ComponentModel; using System.Data;
using System.Drawing; using System.Linq; using System.Text;
using System.Windows.Forms; using iTellUtils;
using System.IO;


namespace iTell
{
public partial class MainForm : Form
{
private ConsoleForm consoleForm = new ConsoleForm(); public void showConsole(bool yes)
{
UserSettings.Instance.ShowConsole = yes; вивестиКонсольToolStripMenuItem.Checked = yes; if (yes)
consoleForm.Show(); else
consoleForm.Hide();
}

public MainForm()
{
InitializeComponent();
}

private void LoadWordDB(string filename)
{
Console.Message("Завантаження бази слів з файлу " + filename); bool ok = true;
try { WordDB.Instance.LoadFromTextFile(filename); }
```

```
catch (WordDBCantLoadException e) { ok = false; Console.Error("LoadWordDB", "Помилка завантаження слів з файлу: " + e.Message); }
if (ok) Console.Message("База слів завантажена");
}

private void SaveWordDB(string filename)
{
Console.Message("Збереження бази слів до файлу " + filename); bool ok = true;
try { WordDB.Instance.SaveToTextFile(filename); }
catch (WordDBCantSaveException e) { ok = false; Console.Error("SaveWordDB", "Помилка збереження слів до файлу: " + e.Message); }
if (ok) Console.Message("База слів збережена");
}

private void Form1_Load(object sender, EventArgs e)
{
consoleForm.Width = 600;
consoleForm.Height = 200; Console.setForm(consoleForm);
if (UserSettings.Instance.ShowConsole)
{
consoleForm.Show(); вивестиКонсольToolStripMenuItem.Checked = true;
}
else
{
consoleForm.Hide(); вивестиКонсольToolStripMenuItem.Checked = false;
}


// todo loading words
if (UserSettings.Instance.CurrentDB != "") LoadWordDB(UserSettings.Instance.CurrentDB);
// todo loading net
if (UserSettings.Instance.CurrentNetwork != "")
NeuralNetwork.Current = new NeuralNetwork(UserSettings.Instance.CurrentNetwork); else
{
int[] sizes = new int[] { 112,100,26 }; NeuralNetwork.Current = new NeuralNetwork(sizes);
}
}

private void button1_Click(object sender, EventArgs e)
{
}
```

```
private void button2_Click(object sender, EventArgs e)
{

}

private void button1_Click_1(object sender, EventArgs e)
{
if (openFileDialog1.ShowDialog() == DialogResult.OK)
{
pictureBox1.Load(openFileDialog1.FileName);
Bitmap bits = ImageFilter.DoFilteringSimple(new Bitmap(pictureBox1.Image)); pictureBox1.Image = bits;
int [,] img = new int[bits.Width,bits.Height]; for (int i = 0; i < bits.Width; ++i)
for (int j = 0; j < bits.Height; ++j)
{
img[i, j] = 255-(int)bits.GetPixel(i, j).B;




}
BlockRecognizer.StartImage(img, bits.Width, bits.Height);
}
}

string goWork(FuzzyVector v, List<double[]> input)
{
string s = WordDB.Instance.findBestWords(v).GetFirstWord(); for (int i = 0; i < s.Length; ++i) {
int a = (int)s[i] - 'a'; double[] need = new double[26];
for (int j = 0; j < s.Length; ++j)
{
need[j] = (j == a) ? 1.0 : 0.0;
}
NeuralNetwork.Current.Learn(input[i], need);
}
return s;
}

private void button2_Click_1(object sender, EventArgs e)
{
DateTime beginTime = DateTime.Now; int total=0; BlockRecognizer.Reset();
FuzzyVector word = new FuzzyVector(); List<double[]> imgs = new List<double[]>(); StringBuilder sb = new StringBuilder();  int lastline = BlockRecognizer.line;
int last = 0; while(true) {
```

```
last = BlockRecognizer.line;
double[] x = BlockRecognizer.NextDoubleChar();
++total;
if (x == null) break;
int spaces = 0; for (int i = 0; i < x.Length; ++i) if (x[i] < 0.1) spaces++; if (spaces > x.Length * 90 / 100)
{
if (word.chars.Count > 0)
{
if (last != lastline)
{
sb.Append("\n");
}

}
}
else
{

lastline=last; sb.Append(goWork(word, imgs)+" "); word.chars.Clear();
imgs.Clear();

imgs.Add(x); NeuralNetwork.Current.Run(x);
word.chars.Add(NeuralNetwork.Current.Output());
}
}
if (word.chars.Count > 0)
{
sb.Append(goWork(word, imgs)); word.chars.Clear(); imgs.Clear();
}

richTextBox1.Text = sb.ToString();
double v=(DateTime.Now - beginTime).TotalSeconds; MessageBox.Show("Затрачений час: " + v.ToString() + "
сек.\n" +
"Всього символів: " + total.ToString() + "\n" +
"Час обробки одного символу: " + (v / (double)total).ToString() + "
сек/символ");
}

private void вивестиКонсольToolStripMenuItem_Click(object sender, EventArgs e)
{
showConsole(!UserSettings.Instance.ShowConsole);
}
```

```
private void вихідToolStripMenuItem_Click(object sender, EventArgs e)
{
Close();
}

private void налаштуванняToolStripMenuItem_Click(object sender, EventArgs e)
{
SettingsForm settingsForm=new SettingsForm(this); settingsForm.ShowDialog();
}

private void редагуванняБазиToolStripMenuItem_Click(object sender, EventArgs e)
{
DBForm form = new DBForm(); form.ShowDialog();
}

private void імпортуватиЗФайлуToolStripMenuItem_Click(object sender, EventArgs e)
{
if (openFileDialog1.ShowDialog() == DialogResult.OK) LoadWordDB(openFileDialog1.FileName);
}

private void експортуватиВФайлToolStripMenuItem_Click(object sender, EventArgs e)
{
if (saveFileDialog1.ShowDialog()==DialogResult.OK) SaveWordDB(saveFileDialog1.FileName);
}

private void button4_Click(object sender, EventArgs e)
{
if (BlockRecognizer.Loaded()) { TestForm form = new TestForm(); form.ShowDialog();
}
}


private void завантажитиНейроннуМережуToolStripMenuItem_Click(object sender, EventArgs e)
{
if (openFileDialog1.ShowDialog() == DialogResult.OK) NeuralNetwork.Current.Load(openFileDialog1.FileName);
}

private void зберегтиНейроннуМережуToolStripMenuItem_Click(object sender, EventArgs e)
{
if (saveFileDialog1.ShowDialog() == DialogResult.OK) NeuralNetwork.Current.Save(saveFileDialog1.FileName);
}
```

```
private void навчанняНаОсновіШаблонних СимволівToolStripMenuItem_Click(object sender, EventArgs e)
{
LearningForm form = new LearningForm(); form.ShowDialog();
}

private void button3_Click(object sender, EventArgs e)
{
if (saveFileDialog1.ShowDialog() == DialogResult.OK) {
File.WriteAllText(saveFileDialog1.FileName,richTextBox1.Text);
}
}

private void завантажитиЗображенняToolStripMenuItem_Click(object sender, EventArgs e)
{
button1_Click_1(sender, e);
}

private void зберегтиРезультатиРозпізнаванняToolStripMenuItem_Click(object sender, EventArgs e)
{
button3_Click(sender, e);
}
}
}
```

Файл LearningForm

```
using System;
using System.Collections.Generic; using System.ComponentModel; using System.Data;
using System.Drawing; using System.Linq; using System.Text;
using System.Windows.Forms; using iTellUtils;

namespace iTell
{
public partial class LearningForm : Form
{
public LearningForm()
{
InitializeComponent();
}

private void button1_Click(object sender, EventArgs e)
{
```

```
if (folderBrowserDialog1.ShowDialog() == DialogResult.OK)
{
textBox1.Text = folderBrowserDialog1.SelectedPath;
}
}

Random rand = new Random((int)DateTime.Now.Ticks); private void button2_Click(object sender, EventArgs e)
{
List<double[]> bits = new List<double[]>(); List<double[]> need = new List<double[]>(); for(int i=0; i<26; ++i)
{
double[] res = new double[7*16];
Bitmap p = new Bitmap(textBox1.Text+"\\"+(char)((int)'A'+i)+".bmp"); int pos=0;
for(int y=0; y<16; ++y) for(int x=0; x<7; ++x) res[pos++] = (double)(255- p.GetPixel(x,y).B)/255.0;
bits.Add(res);
}

for(int i=0; i<26; ++i)
{
double[] a = new double[26];
for(int j=0; j<26; ++j) a[j] = (i==j)?1.0:0.0; need.Add(a);
}


StringBuilder sb = new StringBuilder(); int iter = (int)numericUpDown1.Value; int total=0;
int wins=0; progressBar1.Minimum=0; progressBar1.Maximum=iter;
for (int i = 0; i < iter; ++i)
{
progressBar1.Value = i; Application.DoEvents();
for (int a = 0; a < 100; ++a)
{
int x = rand.Next() % 26;

NeuralNetwork.Current.Run(bits[x]);
if (NeuralNetwork.Current.getDesision()==x) {sb.Append("+");++wins;} else sb.Append("o");
++total;
NeuralNetwork.Current.Learn(bits[x], need[x]);
}
progressBar1.Value = i + 1;
}
richTextBox1.Text = sb.ToString() + "\n"+"["+wins.ToString()+" of "+total.ToString()+"]";
}
```

```
private void button4_Click(object sender, EventArgs e)


{
if (openFileDialog1.ShowDialog() == DialogResult.OK) NeuralNetwork.Current.Load(openFileDialog1.FileName);
}

private void button3_Click(object sender, EventArgs e)
{
if (saveFileDialog1.ShowDialog() == DialogResult.OK) NeuralNetwork.Current.Save(saveFileDialog1.FileName);
}
}
```

Додаток Г
(обов'язковий)

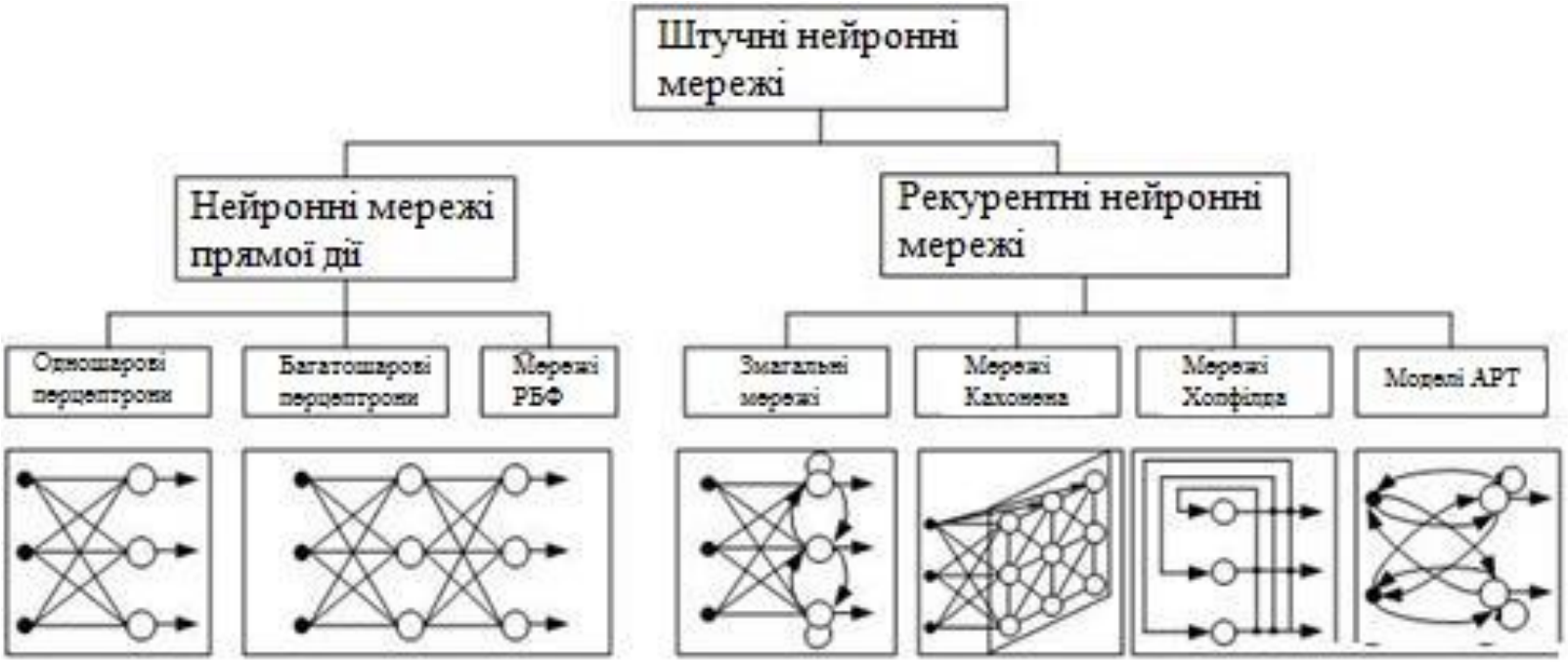
ІЛЮСТРАТИВНА ЧАСТИНА

Студента групи 2АКІТ-21м

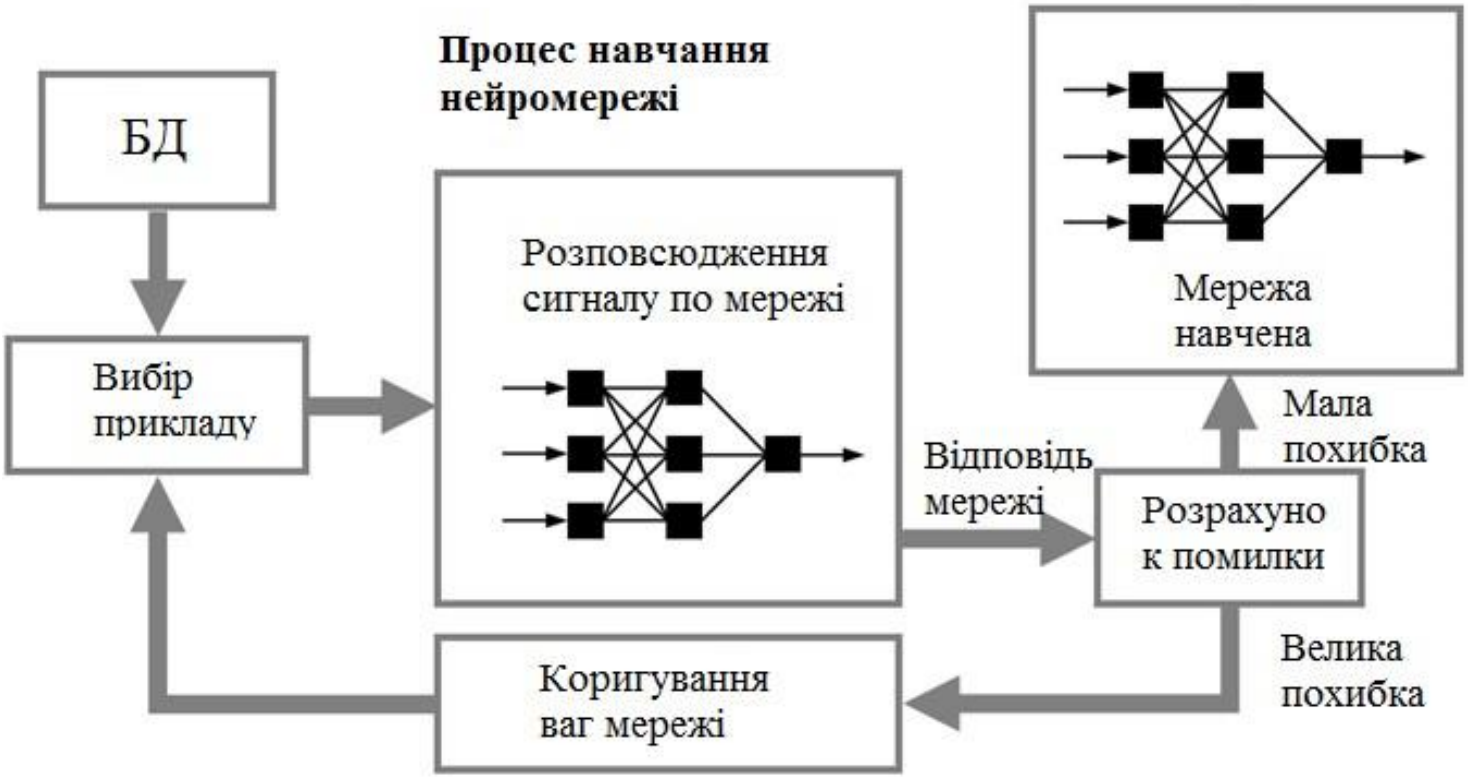
Підпис Антон ШАМАТІЄНКО
Ім'я ПРІЗВИЩЕ

Керівник к. т.н., доцент, доцент кафедри КСУ

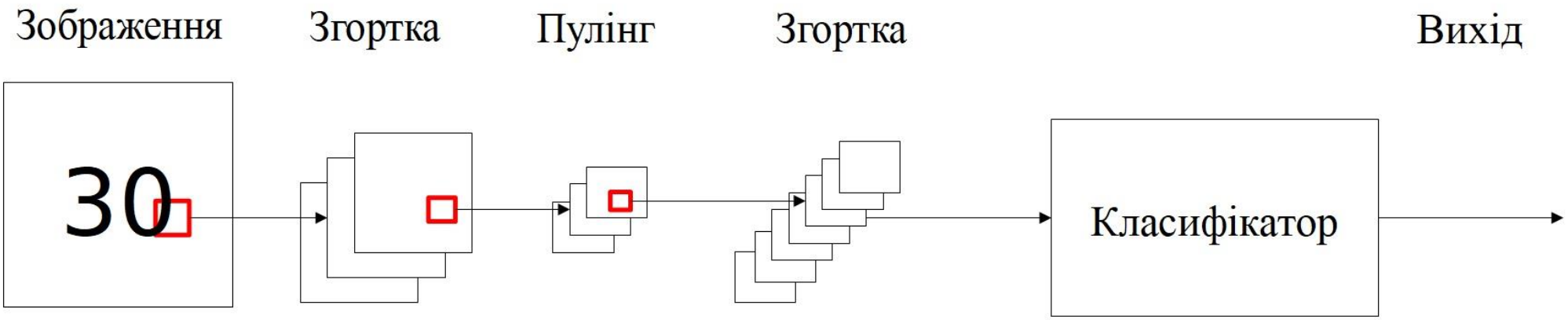
Підпис Олена НИКИТЕНКО
Ім'я ПРІЗВИЩЕ



Класифікація нейромереж за архітектурою



Графічна інтерпретація процеса навчання нейромережі



Згорткова нейронна мережа

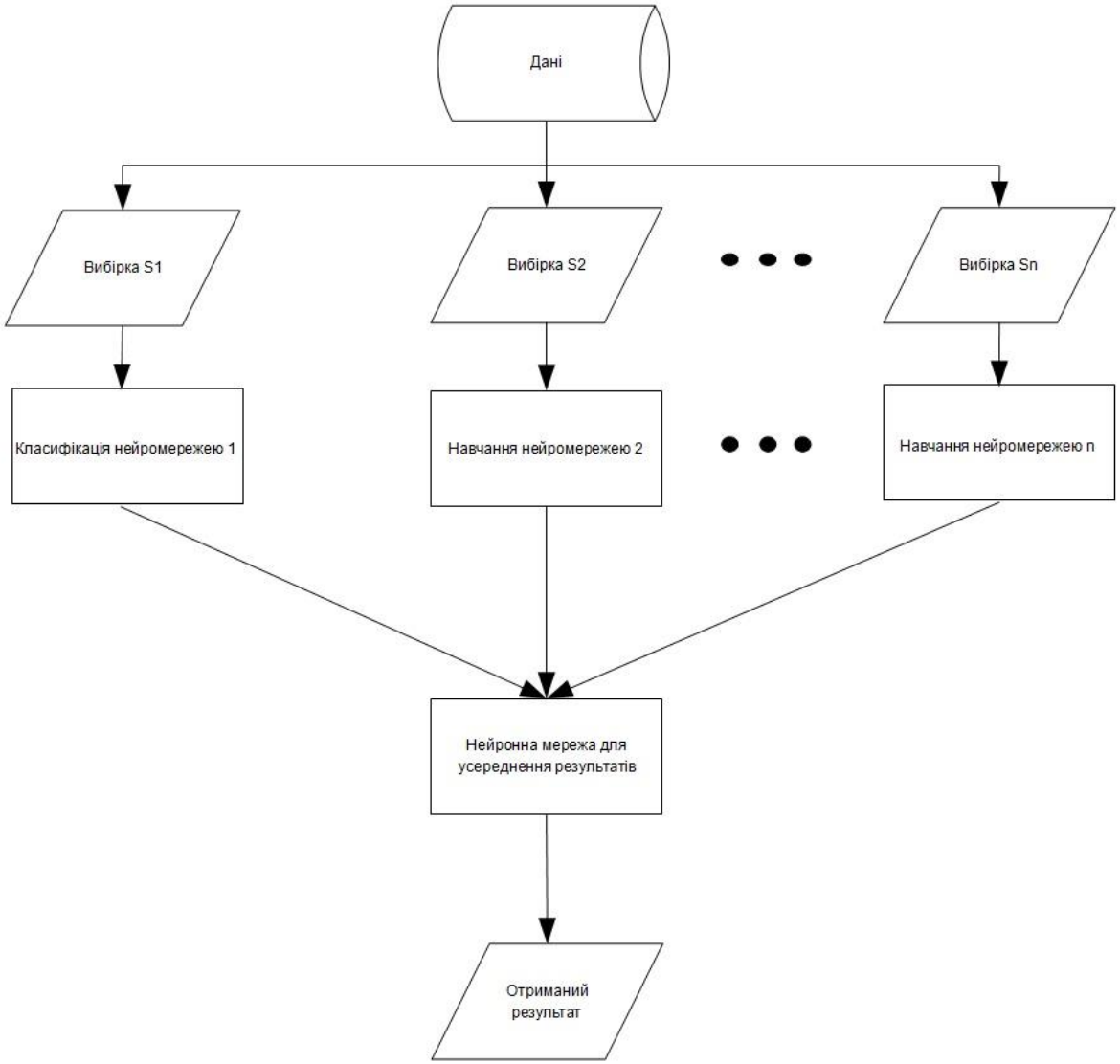
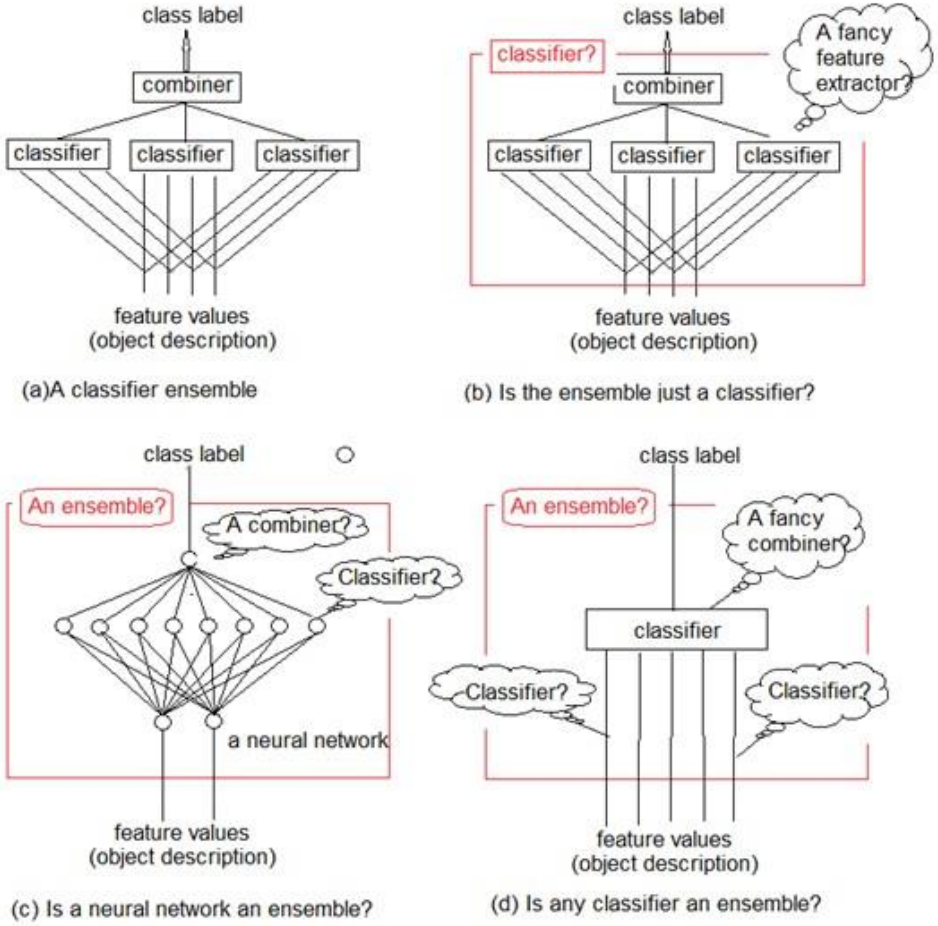
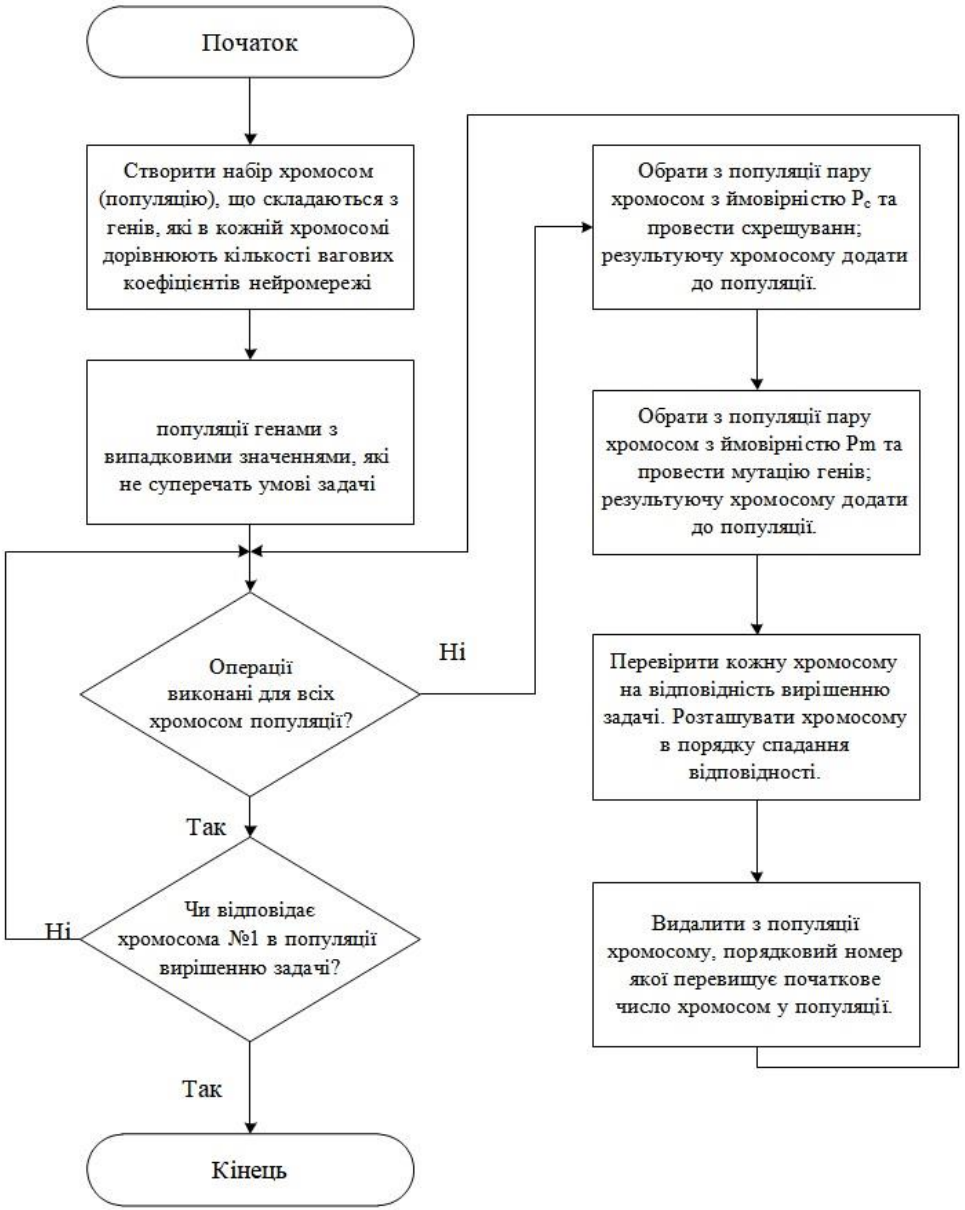


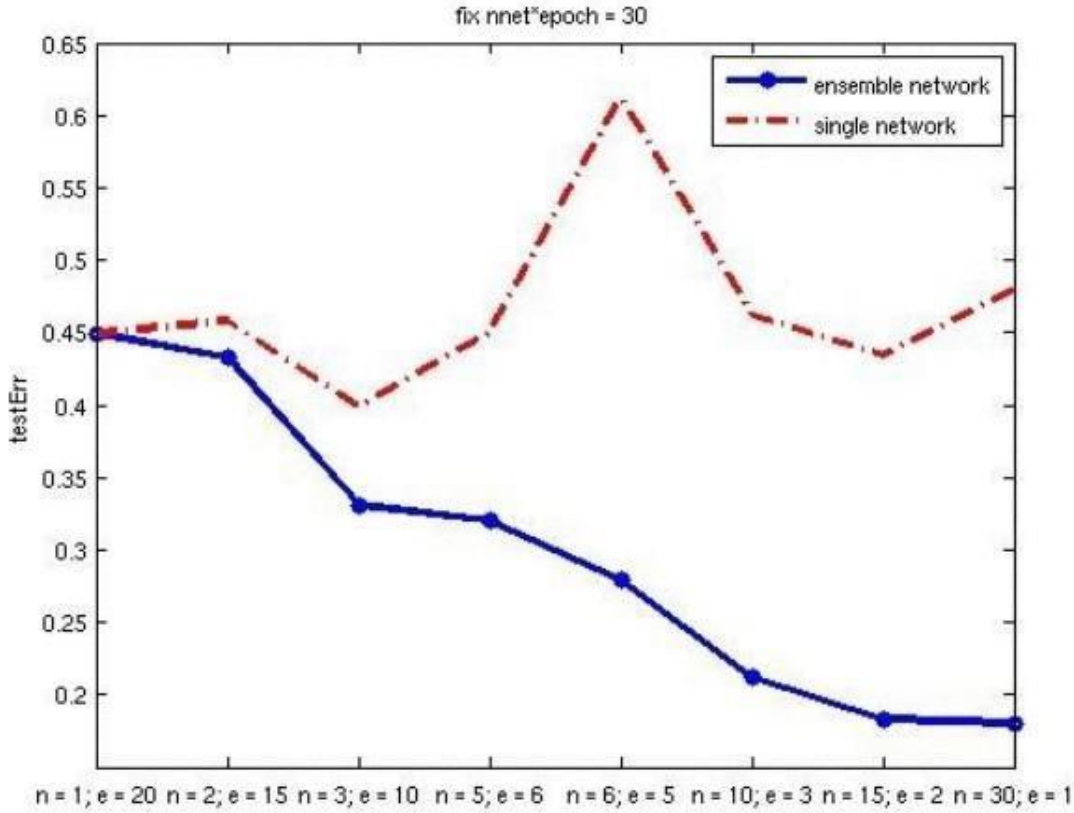
Схема методу паралельного навчання нейромереж



Ансамбль класифікаторів



Алгоритм навчання ансаблю



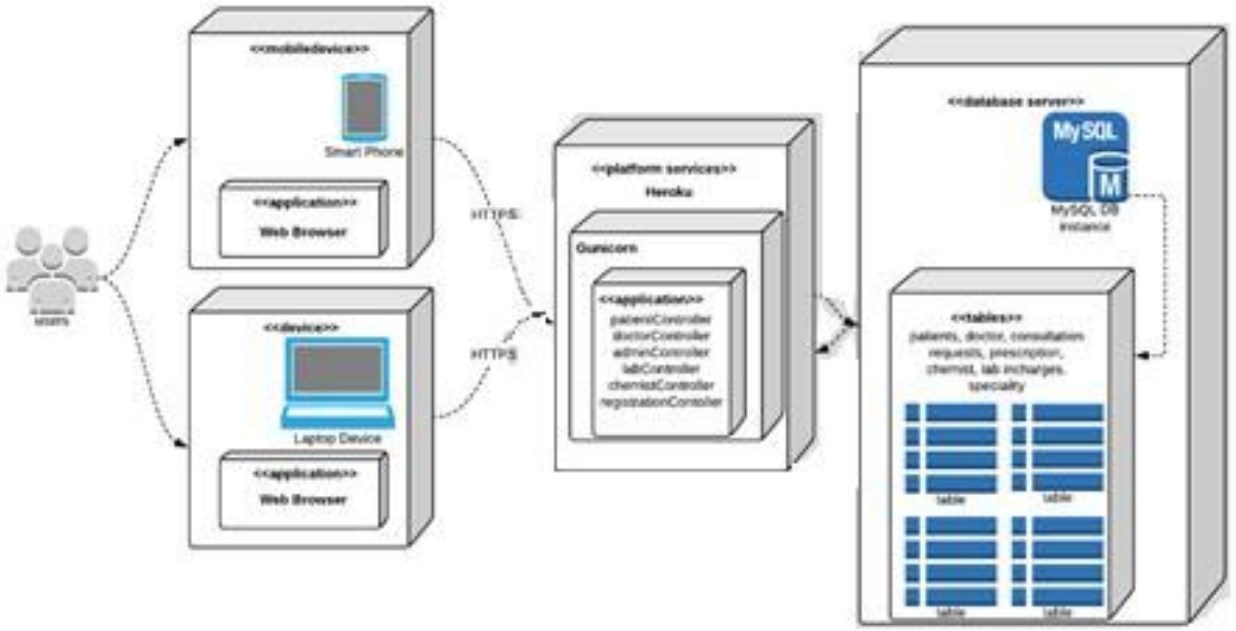
Порівняння точності розпізнавання при застосуванні однієї нейромережі та ансамбля

Комітет CNN	MNIST _test	FONT _test	KNI _test	Середнє для 3-х	NIST_HSF 4	OPTDIGI TS	USPS	Середнє для 3-х
MAX_COM	98.23	98.62	98.56	98.47	97.19	94.51	97.80	96.50
AVER_COM	97.64	99.11	98.10	98.28	96.37	93.73	96.52	95.54
MAJOR_COM	93.82	98.50	98.86	97.06	92.01	87.68	92.95	90.88
KADMOS	95.84	98.68	84.95	93.15	94.45	94.66	97.09	95.4

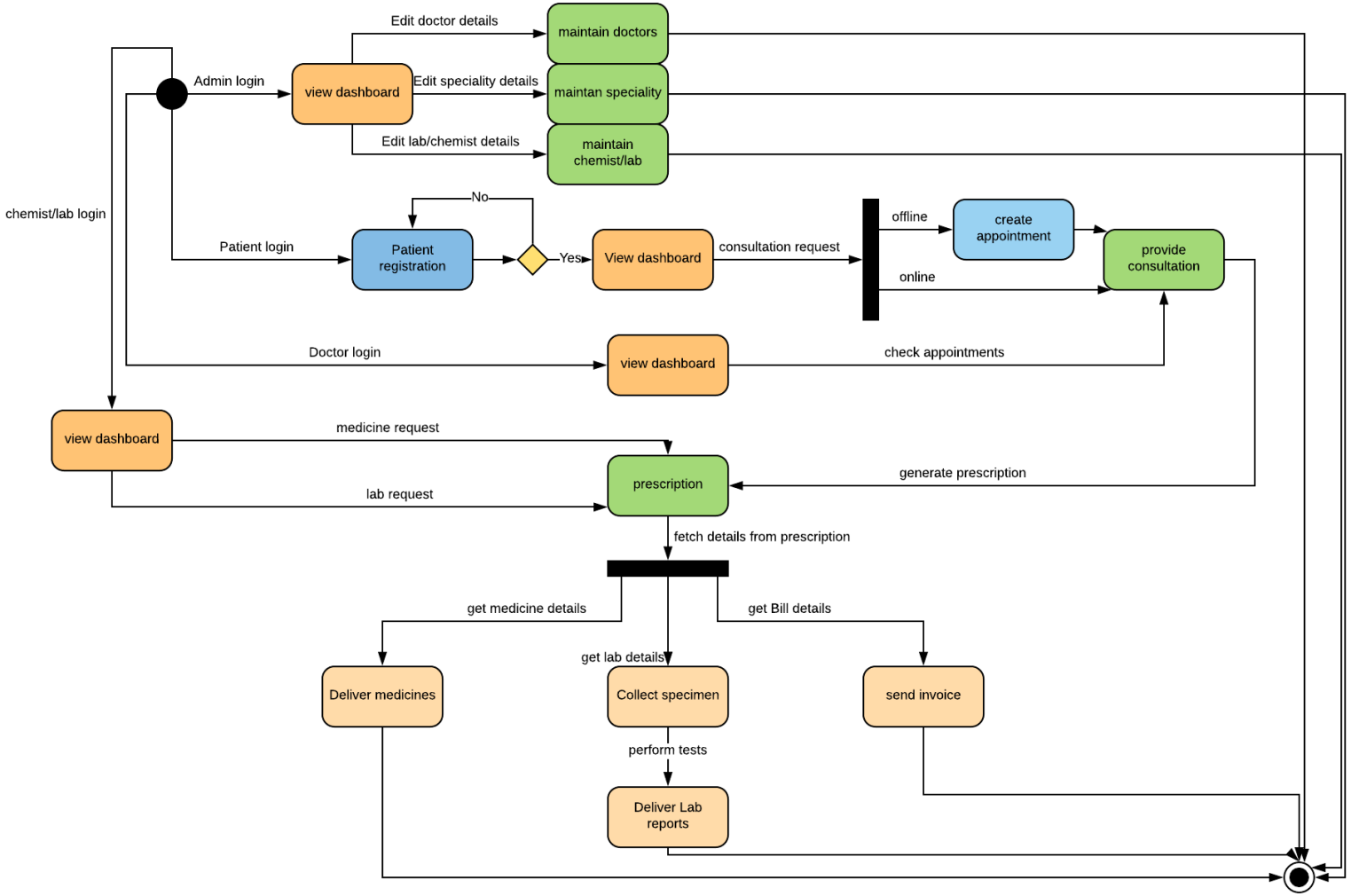
Точність розпізнавання (в%) паттернів контрольних та тестових вибірок комітетами нейромереж

Тип мономережі	Тип з’єднання	Час роботи	Точність
ANFIS	Мономережа	10	82,6%
	Рекурентна мережа+ Мономережа	21	88,2%
	Мономережа + Мережа Хопфілда	22	87,7%
	Рекурентна мережа + Мономережа +Мережа Хопфілда	25	97,8%
NEFCLASS	Мономережа	8	81,4%
	Рекурентна мережа + Мономережа	15	84%
	Мономережа + Мережа Хопфілда	17	87%
	Рекурентна мережа + Мономережа + Мережа Хопфілда	20,6	96,4%

Розрахунок точності різних з’єднань



Діаграма компонентів програмного забезпечення



Діаграма станів програмного забезпечення

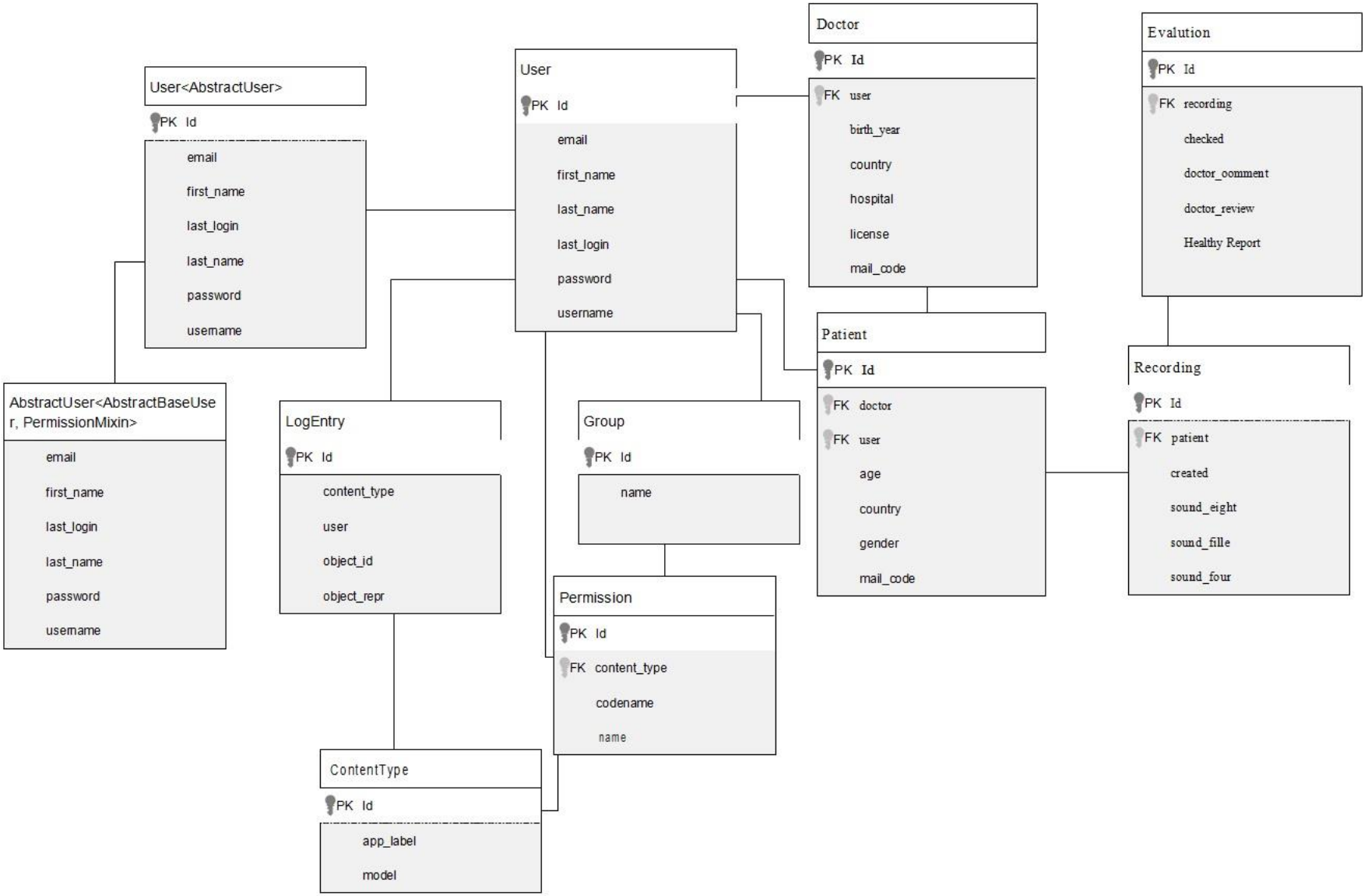
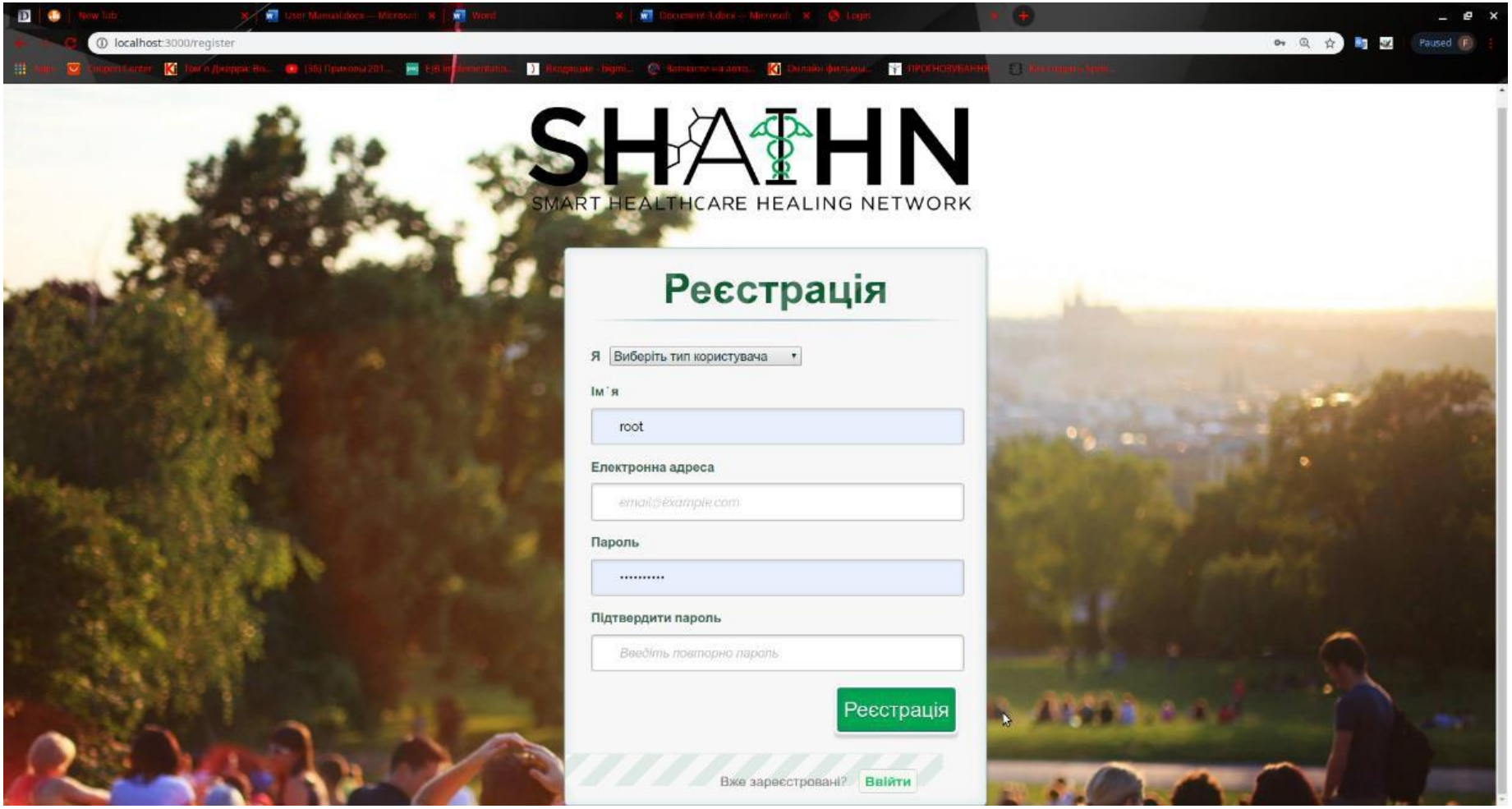
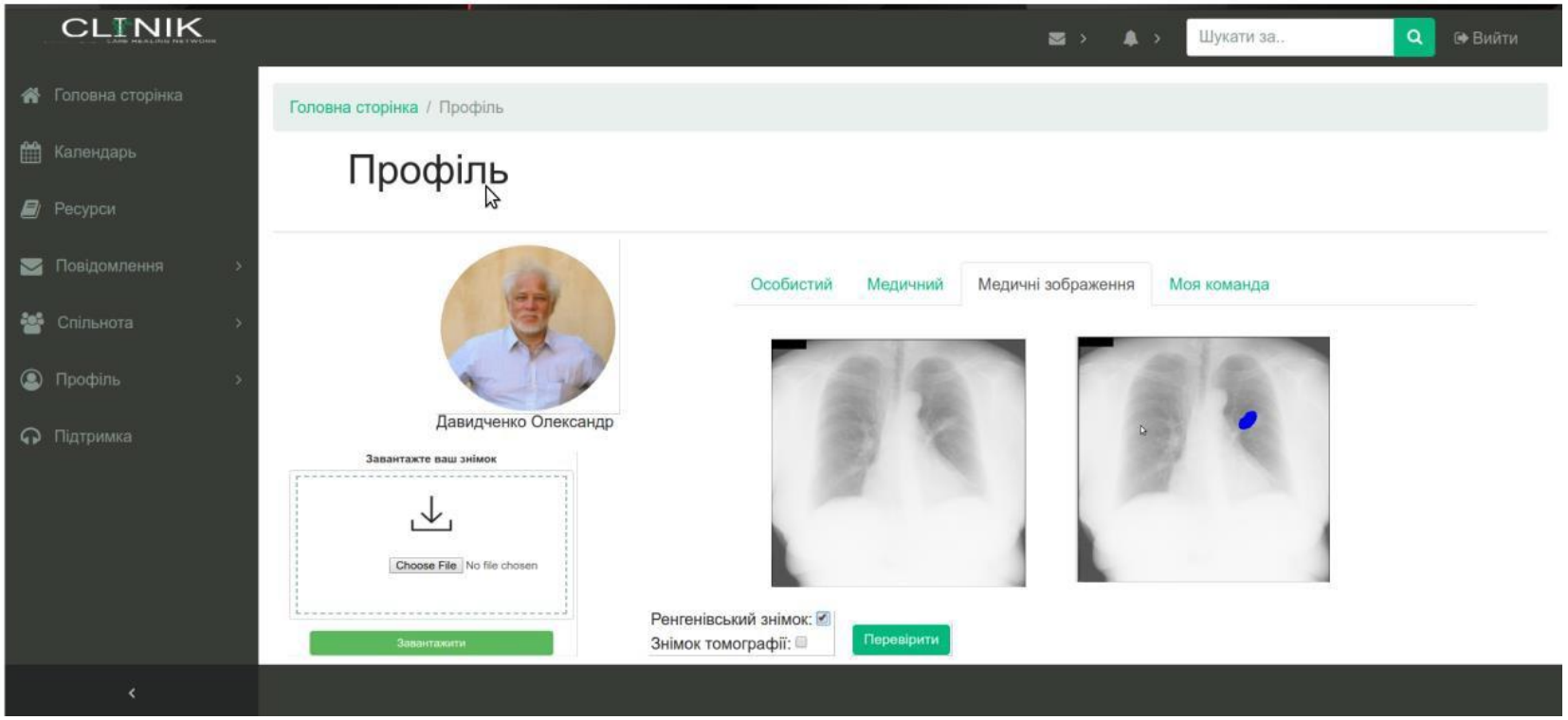


Схема бази даних системи

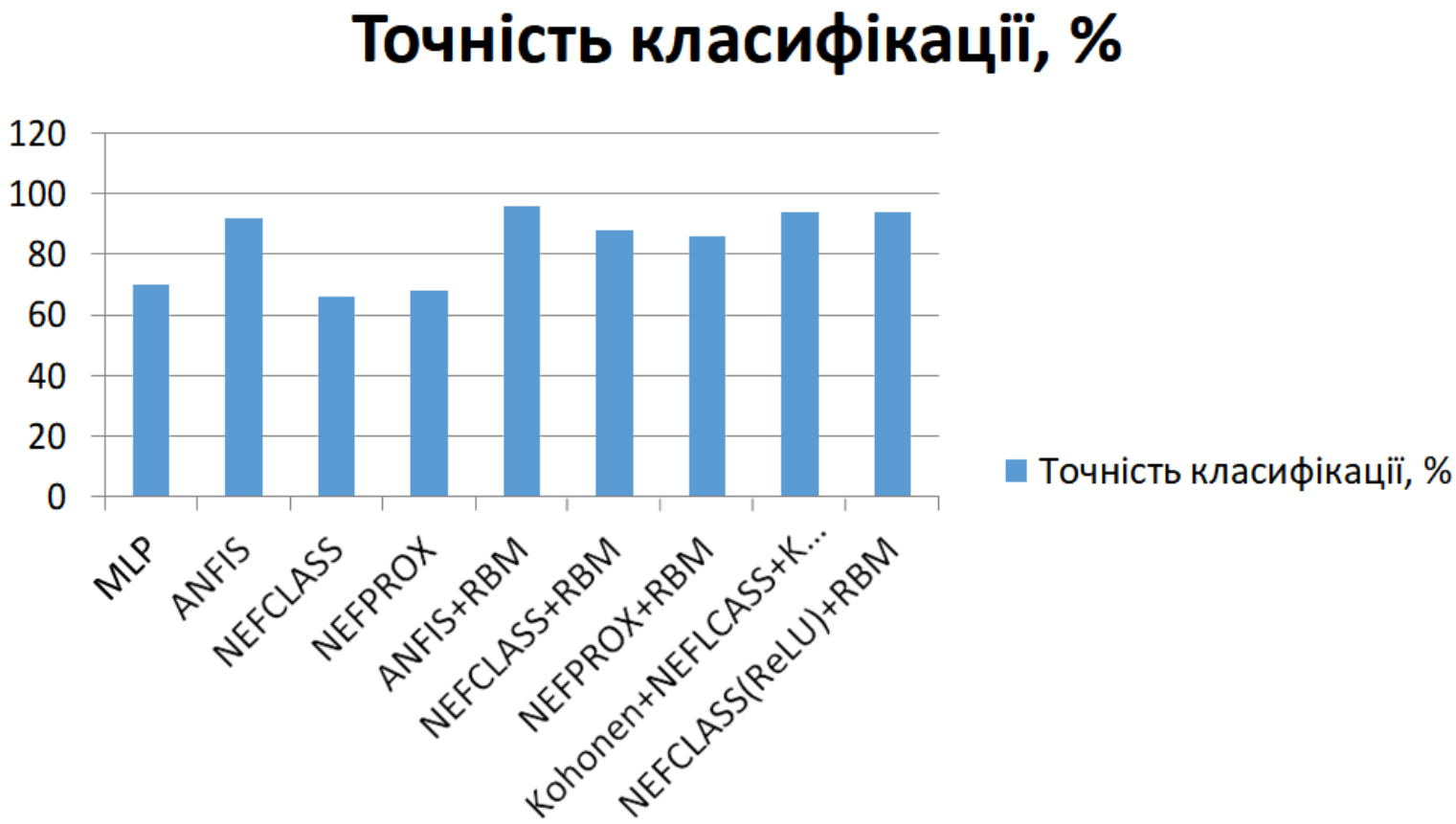


Реєстрація нового користувача



Медичні зображення пацієнта

Результати розпізнавання зображень



Результати реалізації класичної класифікації