Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

«На правах рукопису»
УДК 004.8

«До захисту допущено»
Завідувач кафедри
Ролік О.І.
(підпис) (ініціали, прізвище)
“_____” _____ 20***20*** р.

Магістерська дисертація

зі спеціальності: *126. Інформаційні системи та технології*
(код та назва напряму підготовки або спеціальності)

Спеціалізація: *126. Інтегровані інформаційні системи*

на тему: Інтелектуальна система діагностування легеневих захворювань людини за рентгенівськими знімками

Виконав: студент II курсу, групи ІА-92мп
(шифр групи)

Тарановський Микола Володимирович
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник к.т.н., доцент Писаренко А.В.
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант доц. каф. ТК, к.т.н., доцент Ткач М.М.
(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент

(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент
(підпис)

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет (інститут) *Інформатики та обчислювальної техніки*
(повна назва)

Кафедра *Автоматики та управління в технічних системах*
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною програмою
Спеціальність *126. Інформаційні системи та технології*
(код і назва)

Спеціалізація *126. Інтегровані інформаційні системи*
(код і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____.
(підпис) (ініціали, прізвище)
« » _____ 20***20*** р.

ЗАВДАННЯ
на магістерську дисертацію студенту
Тарановському Миколі Володимировичу
(прізвище, ім'я, по батькові)

1. Тема дисертації «Інтелектуальна система діагностування легеневих захворювань людини за рентгенівськими знімками»
Науковий керівник дисертації к.т.н., доцент Писаренко А.В.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «_» 2020 р. №

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження легеневі захворювання людини

4. Предмет дослідження діагностування легеневих захворювань засобами штучного інтелекту.

5. Перелік завдань, які потрібно розробити: огляд існуючих систем, огляд методів розпізнавання, дослідження нейронних мереж, проектування та програмна реалізація системи

6. Дата видачі завдання

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Строк виконання етапів дисертації	Примітка
1	Дослідження літератури та документації	15.09.2020	Викон.
2	Огляд існуючих рішень	20.09.2020	Викон.
3	Аналіз теоретичних методів	01.10.2020	Викон.
4	Проектування системи	10.10.2020	Викон.
5	Програмна реалізація системи	25.10.2020	Викон.
6	Тестування та виправлення помилок	30.10.2020	Викон.
7	Оформлення документації	15.11.2020	Викон.

Студент _____ _Тарановський М.В.___
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ __Писаренко А.В.
(підпис) (ініціали, прізвище)

Реферат

Магістерська дисертація: 106 с., 62 рис., 24 табл., 9 додатків, 29 джерел.

Актуальність проблеми. Сьогодні весь світ зіткнувся із проблемою Covid-19, дана проблема є дуже актуальною. Через велику кількість заражених і переповненість лікарень огляди і аналізи хворих забирають багато часу. Одним із основних рішень лікарів при огляді пацієнта із симптомами Covid-19 є проходження хворим рентгенівського знімку легень, що дозволяє отримати дані чи має пацієнт пневмонію. Для того, щоб прискорити процес і отримати більше інформації розроблена система, що дозволяє за рентгенівськими знімками визначати чи людина має ураження легень, і якщо так, то вказує чи спричинені вони Covid-19.

Мета і задачі дослідження. Метою роботи є підвищення швидкості встановлення діагнозу Covid-19 за рентгенівським знімками за допомогою штучного інтелекту.

Об’єкт дослідження. Легеневі захворювання людини.

Предмет дослідження. Діагностування легеневих захворювань засобами штучного інтелекту.

Новизна. Запропоновано новий засіб діагностування Covid-19, який за рахунок використання нейронних мереж, дозволяє ефективніше і точніше розшифровувати рентген знімки та КТ.

Ключові слова. *Нейронні мережі, машинне навчання, CNN, навчання нейронних мереж, розпізнавання, класифікація, рентген, Covid-19*

Abstract

Master's Thesis: 106 pp., 62 figs., 24 tables, 9 appendix, 29 sources.

The urgency of the problem. Today, the whole world is faced with the problem of Covid-19, this problem is very relevant. Due to the large number of infected and overcrowded hospitals, all examinations and tests of patients take a long time. One of the main decisions doctors make when examining a patient with Covid-19 symptoms is to have an X-ray of the patient's lungs to see if the patient has pneumonia. In order to speed up the process and get more information, it was decided to make a system that allows you to analyze the image to deduce the result of whether a person has lung damage, and if so, indicates whether they are caused by Covid-19.

The purpose and objectives of the study. The task of this work is to study the possibility of recognizing lung diseases from X-rays. The goal is to develop a neural network-based system capable of recognizing Covid-19 lesions and providing data to the user using a cross-platform application.

Object of study. The process of recognizing human activity using elements of the neural network.

Subject of study. Methods of analysis and processing of image classification.

Novelty. A new method of diagnosing Covid-19 has been proposed, which, through the use of neural networks, allows more efficient and accurate decoding of X-rays and CT scans..

Keywords. *Neural Networks, Machine Learning, CNN, Neural Network Learning, Recognition, Classification, X-Ray, Covid-19*

ЗМІСТ

ПЕРЕЛІК ТЕРМІНІВ ТА СКОРОЧЕНЬ

AI	(англ. Artificial intelligence) — скорочення англійською мовою визначення «Штучний інтелект».
CV	(англ. Computer vision) підгалузь машинного навчання. Наука про створення комп’ютерів, що дозволяють визначати, відслідковувати та класифікувати об'єкти.
KNN	(англ. k-nearest neighbors) алгоритм в якому для класифікації необхідні відстані до об’єктів.
ML	(англ. Machine learning) підгалузь AI.
OS	(англ. operating system) Операційна система.
SVM	(англ. Support Vector Machine) алоритм аналізу даних, що використовує моделі керованого навчання.
Валідація	процедура, що дозволяє перевірити правильність введених даних.
Хостинг	виділення пам’яті для розміщення даних на віддаленому пристрої, сервері.

ВСТУП

В сучасному світі люди зробили великий крок вперед – з'являється багато сфер, де замість людини працює штучний інтелект. На це є багато причин: виконання монотонної роботи, помилки, що виникають внаслідок людського фактору, реакція технічних приладів набагато краща, непристосованість людського тіла для роботи в критичних умовах. Керування авто при поганих погодніх умовах може спричинити ДТП, автомобіль, що оснащений сучасними технологіями може використовувати спеціальні прилади, щоб уникнути поганих наслідків.

На сьогоднішній день дуже швидко розвиваються технології комп'ютерних наук, IoT і машинного навчання. Дані технології займають лідируюче місце в сьогоднішньому науковому світі. Часто у завданнях програмування необхідно використовувати алгоритми машинного навчання, класифікації та кластеризації. У звичайному житті людей і у промисловій індустрії все більше впроваджуються нові технології, що знищують грань між віртуальним і реальним світом. Задачі класифікації та кластеризації, що раніше вважалися складними, швидко і без проблем вирішуються технічним пристроями звичайних людей.

Пришвидшення роботи технічних пристроїв у задачах розпізнавання об'єктів у роботизації виробництва, транспорту, екології, заходів для порятунку людей, військових, що здійснюють роботу в інтересах держави, дозволяє зробити людське життя краще та загалом зберегти його.

Протягом тривалого періоду предметом дослідження інженерів є моделювання навколишнього середовища за допомогою комп'ютерів. Для цього необхідно володіти великим обсягом даних і обробити їх для отримання результатів моделювання з чим добре справляють алгоритми машинного навчання.

Основним завданням дисертації є класифікація об'єктів на зображенні. Оскільки зараз світ зіткнувся із проблемою Covid-19, вирішено присвятити увагу

цій проблемі. Через велику кількість заражених і переповненість лікарень всі огляди і аналізи хворих забирають багато часу. Одним із основних рішень лікарів при огляді пацієнта із симптомами Covid-19 є проходження хворим рентгенівського знімку легень, що дозволяє отримати дані чи має пацієнт пневмонію. Для того, щоб прискорити процес і отримати більше інформації було прийняте рішення зробити систему, що дозволяє проаналізувавши знімок виводити результат чи людина має ураження легень, і якщо так, то вказує чи спричинені вони Covid-19.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Машинне навчання

Машинне навчання (Machine learning) – це підгалузь розділу інформатики штучний інтелект (Artificial intelligence), що використовуючи спеціальні алгоритми надає комп’ютеру вміння навчатись. Дана технологія надає можливість програмам ставати точнішими у прогнозуванні даніих, не будучи при цьому явно запрограмованими. Головною передумовою роботи машинного навчання є розроблення таких алгоритмів, що можуть отримувати початкові дані, що надаються системі і використовувати статистичний аналіз таким чином, що матиме змогу прогнозувати результат, коли нові вхідні дані будуть доступними [7].

Процеси, що пов'язані із машинним навчанням, схожі до процесів, що виконуються для аналізу та моделювання даних. Для роботи обох необхідно виконати пошук даних, щоб обрати необхідні шаблони і згідно з ними керувати діями програми. Більшість людей стикалася із машинним навчанням здійснюють покупки в інтернет-магазинах. Користувачев ресурсу отримує таргетовані оголошення, що зацікавлять його. Це стає можливим через те, що інтернет-магазини користуються машинним навчанням, щоб персоналізувати відображення оголошень в реальному часі. Крім відображення персональної реклами, інші поширені можливості використання алгоритмів машинного навчання включають виявлення шахрайства, фільтрацію листів електронної пошти, виявлення небезпечних дій в мережі, прогнозування метеоролічгих даних.

Алгоритми машинного навчання розділяються на контрольовані та неконтрольовані. Контрольовані алгоритми вимагають, щоб спеціаліст, що програмуватиме мимтему володів навичками машинного навчання. Він визначає, які признаки модель повинна аналізувати і використовувати, щоб зробити прогноз. Після завершення навчання алгоритм використає ті дані, що було вивчено до нових. Неконтрольовані алгоритми нема необхідності

тренувати з вихідними даними. Замість цього способу дані алгоритми використовують ітераційний підхід, що має назву глибинне навчання [4,6]. Неконтрольовані алгоритми [5] – їх ще називають нейронними мережами – використовуються для складніших завдань обробки даних, ніж контрольовані, в тому числі для класифікації та розпізнавання об'єктів на зображенні, перетворення аудіофайлу із мовленням до текстового виду та іншого. Дані нейронні мережі працюють, аналізуючи багато прикладів тренувальних даних і автоматично визначають кореляції між багатьма змінними. Після того, як алгоритм пройшов процес навчання, він виуористовує власний банк асоціацій для аналізу нових даних. Дані алгоритми потребують великих об'ємів навчальних даних. На рисунку 1.1 зображено алгоритм роботи Machine learning.

Рисунок 1.1 – Алгоритм роботи Machine learning

1.2 Комп’ютерне бачення

Комп'ютерне бачення – це підгалузь Machine learning, основним завданням якої є надання комп'ютерам можливості «бачити».

На абстрактному рівні робота комп'ютерного зору полягає в тому, що використовуючи зображення об’єктів, що спостерігаються, необхідно зробити певний висновок.

Комп’ютерний зір можна назвати частиною машинного навчання та штучного інтелекту, що також містить використання спеціалізованих методів та алгоритмів навчання. Зв'язок AI, ML та CV зображено на рисунку 1.2

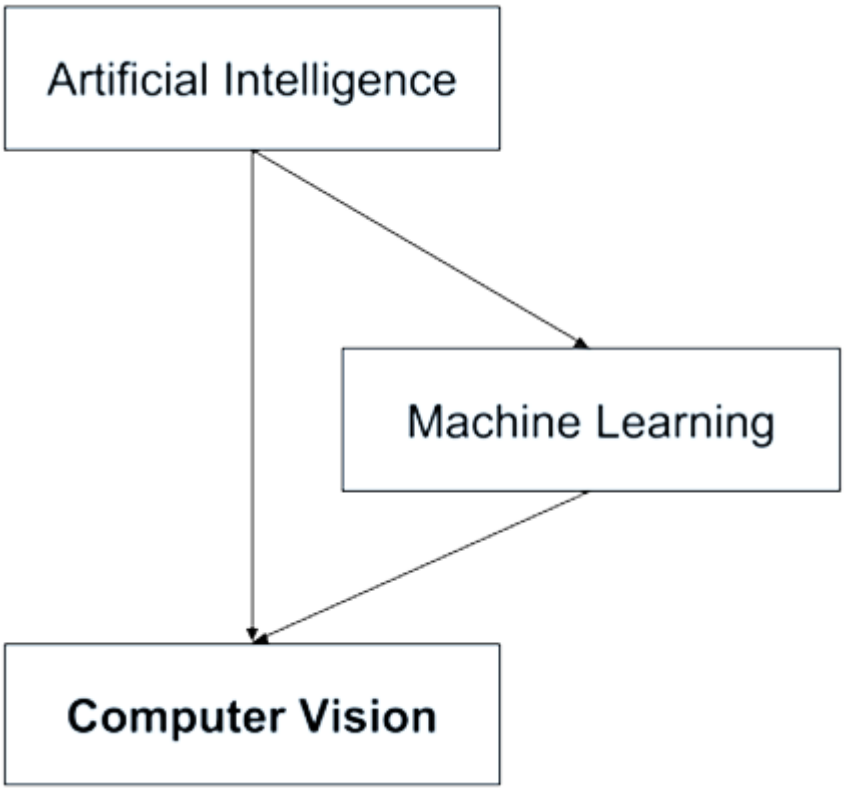


Рисунок 1.2 – Зв'язок комп'ютерного бачення із AI та ML [17]

Дана область дослідження є мультидисциплінарною. В ній використовуються методи, що запозичені і повторно використовуються з різних сфер технічних наук.

Одне спецефічне завдання для комп’ютерного бачення можна з невеликими зусиллями вирішити за допомогою ручного статистичного методу, проте

інший можливо вимагати складного поєднання роботи різних алгоритмів машинного навчання.

Метою комп'ютерного бачення отримання даних із цифрових зображень. Для цього необхідно здійснити розробку методів, які зможуть відтворити людськмй зір.

Результат аналізу цифрового зображення нейронною мережею має містити опис того, що на ньому міститься: певний об’єкт, текстовий надпис, багатовимірна модель і т.д.

1.3 Теорія розпізнавання образів

Теорія розпізнавання образів це підрозділ таких наукових дисципліп як математична статистика [3], комп’ютерні науки та інших розділів. Її основне завдання – дослідження класифікації та виявлення різних об'єктів: текстових знаків, предметів, ситуацій, що мають множину певних ознак. Для розпізнавання зображень використовується строга математична мова, ґрунтуючись на логічному мисленні і доведених математичних законах. Проте з іншої сторони, є способи розпізнавання об’єктів за допомогою машинного навчання та штучних нейронних мереж, що формуються не дуже строго формалізованими принципами для розпізнавання, проте часто вони показують не гірший, а в деяких випадках набагато якісніший результат розпізнавання, кращий за класичні методи. Проблеми пов’язані із розпізнаванням об’єктів також обмежують сферу застосування різних класичних методів тільки спеціалізованими напрямками, в кожному з яких найефективнішими являються одні методи проте зовсім неефективними інші. Це і є причиною складності знаходження способу для спільного вирішення проблеми.

1.4 Основні методи класифікації зображень

1.4.1 Наївний байєсівський класифікатор

Наївний байєсівський класифікатор – це представник сімейства простих ймовірнісних класифікаторів, що засновані на використанні теорема Байєса із наївними припущеннями про незалежність змінних. Нижче приведена формула теореми Байєса:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \qquad (1)$$

де P(A|B) – імовірність припущення A при настанні події B (апостеріорна ймовірність), P(B|A) – імовірність того, що подія B відбудеться при істинності припущення A, P (A) – апріорна ймовірність припущення A і P(B) – повна імовірність того, що подія B відбудеться.

Наївні байєсівські класифікатори володіють однією дуже вагомою якістю: дані класифікатори не приводять до надмірного зростання точності. Для цього наївні байєсівські класифікатори спочтаку роблять припущення про імовірнісне розподілення відповіді. Також недоліком цього підходу є те, що у даного алгоритму мало параметрів.

Якщо перейти від теорії до практики, можна виділити наступні сфери застосування наївного байєсівського класифікатораю:

- фільтрація електронної пошти;
- сегментація публікацій по спільній темі;
- визначення чи певний блок тексту емоційно забарвлений;
- програмні продукти, що розпізнають людей.

1.4.2 K-найближчих сусідів

K-найближчих сусідів (k-nearest neighbors, KNN) – це непараметричний, простий метод класифікації. Його алгоритмом роботи є використання бази даних, де маркери вхідних даних розділені на кілька класів для передбачення класифікації вихідної прогнозованої точки.

Визначення того, що алгоритм непараметричний означає, що він не робить ніяких припущень відносно початково розподілу даних. Іншими словами, структура моделі для класифікації визначається використовуючи дані. В зовнішньому середовищі більшість даних не підкоряються класичному теоретичному припущенню (як у згаданих моделях лінійної регресії). Тому KNN – це перше, що спадає на думку при виборі алгоритму для класифікації об'єктів, коли початкових знань про розподіл даних зовсім мало або вони взагалі відсутні.

KNN – також являється лінивим алгоритмом. Дана характеристика ощначає, що він не використовує дані навчальних точок, щоб зробити будь-яке узагальнення. Інакше кажучи, фаза тренування відсутня або вона є дуже малою. Також це означає, що фаза тренування є досить швидкою. Відсутність узагальнення означає, що метод K-найближчих сусідів зберігає майже усі тренувальні дані оскільки вони необхідні під час етапу тестування.

Результатом роботи метод K-найближчих сусідів є класифікація нових об'єктів. Для цього вибирається значення ознаки декількох найближчих сусідів, і перевіряється до якого саме класу вони відносяться, передбачуваний об'єкт належатиме до того класу який переважає серед сусідів. Приклад роботи методу K-найближчих сусідів зображений на рисунку 1.3.

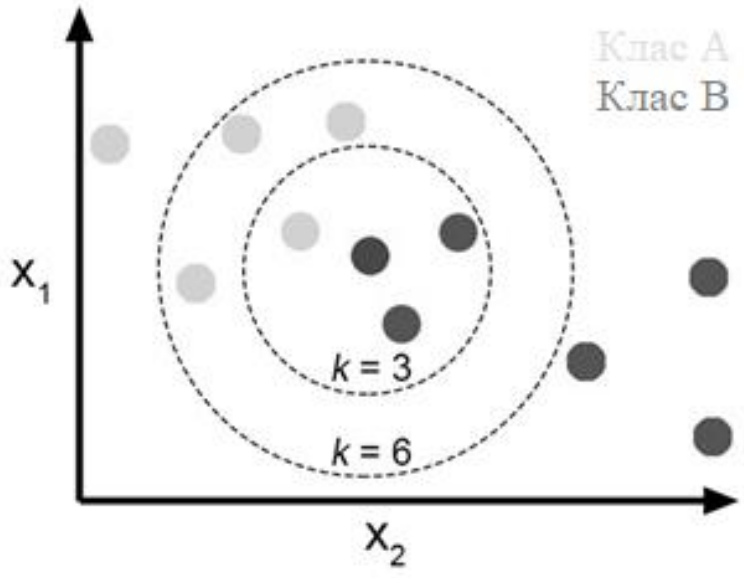


Рисунок. 1.3 – Приклад KNN класифікації [26]

1.4.3 Метод опорних векторів

Методи опорних векторів (support vector machines, SVM) знаходять межу, що розділяє класи з найбільшою шириною. Якщо значення двох класів можна чітко розділити, алгоритми методу опорних векторів знайдуть найкращу границю, яку зможуть. Методи опорних векторів, що містять тільки два класи роблять даний процес для прямої лінії (користуючись термінами даного методу, вони користуються лінійним ядром). Методи опорних векторів використовують лінійну апроксимацію, то час виконання роботи досить малий. Дані методи добре проявляють себе у роботі із даними, що містять велику кількістю функцій, такими як аудіо чи текстові файли. Під час роботи із цими даними методи опорних векторів можуть розділити класи з більшою швидкістю і з меншою надлишкової точністю, ніж інші алгоритми; вагомою перевагою також є те, що вони використовують малий об’єм пам'яті.

Інший метод, що часто використовується – це двокласовий локально глибокий SVM. Даний метод позиціонується як нелінійний варіант методу опорних векторів. Він зберігає велику швидкість роботи і використання малого об’єму пам'яті лінійного варіанту методу. Даний спосіб підходить для випадків, з якими робота лінійного методу не дає результатів необхідної точності.

Перевагою роботи є те що спеціалістам вдалось зберегти швидкість роботи методу розділивши вхідне завдання на декілька менших завдач методу опорних векторів.

Використовучи ефективне розширення нелінійних методів метод опорних векторів із одним класом проводить межу, шо відокремлює весь початковий набір даних. Даний механізм зручно використовувати для виявлення аномалій. Всі нові точки, що знаходяться далеко за цією межею не грають важливої ролі. Приклад SVN класифікації зображений на рисунку 1.4.

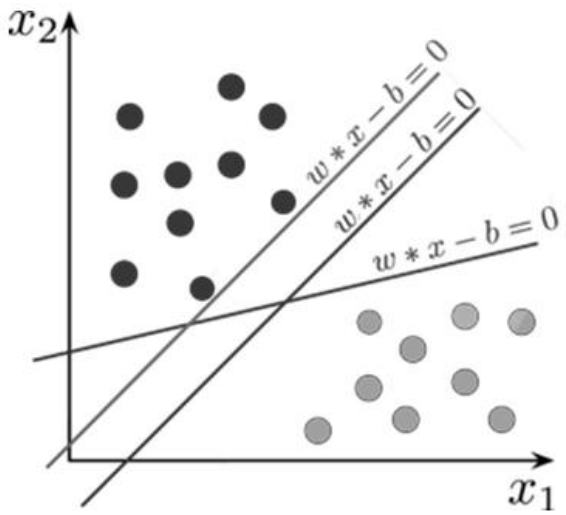


Рисунок. 1.4 – Приклад SVN класифікації [27]

1.5 Нейронна мережа

Застосування нейронних мереж вирішує великий спектр проблем, що стосуються таких задач, як визначення об’єктів, апроксимація функцій, прогнозування, покращення роботи, керування і створення пам'яті, що переадресовується за змістом. У задачі розпізнавання об’єктів вхідний об’єкт відносять до найбільш визначеного класу. Популярним прикладом використання нейронних є розпізнавання об’єктів на зображенні перетворення рукописного тексту у друкований, класифікація популяції тварин, аналіз даних у медичній сфері.

Для вирішення наукових і технічних задач моделювання, нейронні мережі можуть допомогти проводити апроксимацію функцій для отримання оцінки невідомої функції, провести максимізацію або мінімізацію цільової функції і оптимізувати рішення, що задовільняє дану систему обмежень функції. У різних сферах сучасного світу штучні нейронні мережі відіграють надважливу роль. Для прикладу, нейронні мережі можна застосовувати в таких сферах як автомобільна промисловість (аналіз природніх умов, класифікація зображень для використання автопілоту). Також можливий варіант використання нейронних мереж є пам'ять, яка адресується за змістом даних. Виклик елементів цієї пам'яті виконується за вказівкою заданого змісту. Дана

процедура можлива навіть при неповному або непослідовному змісті. Пам’ять такого типу можна використовувати у сфері де є мультимедійні інформаційні бази даних. Завданням систем управління є прогнозування таких вхідних даних, при яких комп’ютерна система веде себе так як нам необхідно, відповідно до заданої еталонної моделі.

1.5.1 Штучний нейрон

Одиничним елементом біологічної та штучної нейронної мережі є нейрон.

Людський мозок складається із 10000000000 нейронів . На один нейрон може припадати декілька тисяч зв'язків з іншими клітинами нейронної мережі.

На рисунку 1.5 зображений біологічний нейрон, що являється клітиною живого організму. Її функціонал полягає в розприділянні хімічних та електричних сигналів.

У біологічному нейроні інформація приходить до нього через дендрити, а від нього через аксони. Аксони створюють зв'язок з дендритами інших клітин використовуючи хімічний процес – синапс. Нейрон стає активним, коли сумарний рівень сигналів, що приходять до дендритів, стає більшим за поріг активації. Під час цього процесу нейрон надсилає хімічні та електричні сигнали.

Вчений Дональд Хебом, що являється спеціалістом в області нейрофізіології, вважає що навчання нейронних мереж відбувається при зміні сили синаптичних зв’язків.

1.5.2 Персептрон

У 50- 60 роках XX ст. американський вчений Френк Розенблат запропонував модель , елемента нейронної мережі, персептрона, що є, по суті, спрощеною математичною моделлю нейрона. З того часу модель в її

почптковому вигляді застаріла, і сьогодні використовують покращенні моделі штучних нейронів, таких як сигмоїд.

Отже, класична математична модель персептрона, що зображена на рисунку 1.6 має декілька бінарних входів (1..X) на початку і один бінарний вихід.

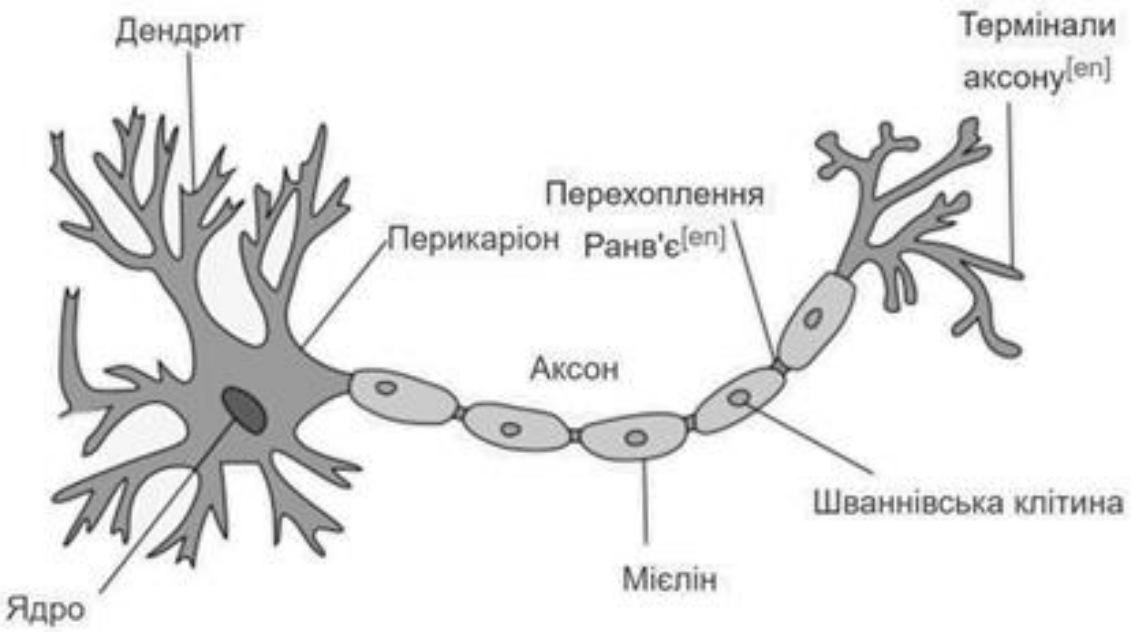


Рисунок. 1.5 – Біологічний нейрон [15]

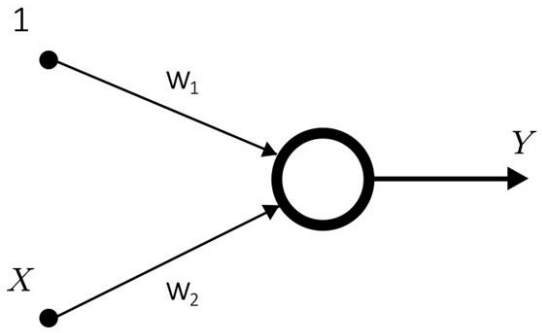


Рисунок 1.6 – Модель персептрона [24]

Френк Розенблатт запропонував просте правило для обрахувпння кінцевого значення: він вирішиа ввемти ваги w1, w2 ..., дійсні числа виражають істотність відповідного вхідного значення для фінального рішення. Кінцеве значення нейрона 0 або 1 визначається тим, виявилася чи зважена сума менше або більше певного кількістного порога, threshold.

$$\begin{cases} 0 \text{ if } \sum_j w_j x_j \leq \text{ threshold}, \\ 1 \text{ if } \sum_j w_j x_j > \text{ threshold}. \end{cases} \quad (2)$$

Навчальні алгоритми в автоматичному режимі налаштовують ваги і відхилення мережі нейронів у відповідь на зовнішній подразник, без безпосередньої участі програміста.

Процес навчання передбачає малу зміну ваг і відхилень, щоб в підсумку отримати невелику зміну в кінцевому результаті, наближаючись до найкращого результату класифікації. Проте, мережа персептронів працює не зовсім так, мале відхилення може викликати серйозні зміни, приводячи до наслідків, що неможливо передбачити. Дана проблема вирішується використанням сигмоїда – це вид штучного нейрона.

1.5.3 Сигмоїд

Сигмоїд – це штучний нейрон, що зображений на рисунку 1.7. Він є гладкою функцією. Дана властивість нейрона відіграє важливу роль.

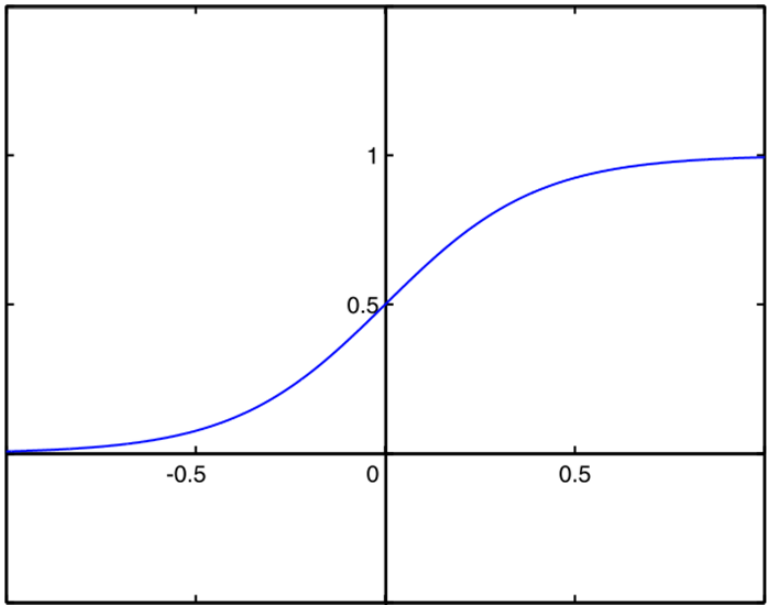


Рисунок. 1.7 – Сигмоїд

Дана функція зветься функцією активації. Існує декілька видів таких функцій. Їхня форма впливає на швидкість навчання. Потрібно зазначити, що якщо б на її місці була одинична функція, даний штучний нейрон перетворився би в персептрон. Гладкість цієї функції дозволяє отримати невеликі зміни виходу при невеликих змінах ваг і відхилень.

Стає зрозуміло, що вихід даного нейрона вже є не бінарним і він може отримувати дійсні значення від 0 до 1.

$$\Delta_{\text{вихід}} \approx \sum_j \frac{\partial_{\text{вихід}}}{\partial w_j} \Delta w_j + \frac{\partial_{\text{вихід}}}{\partial b} \Delta b \qquad (3)$$

1.5.4 Архітектура нейронних мереж

Нейронні мережі складаються з вхідного (початково) шару, вихідного (кінцевого) і одного або декількох внутрішніх (прихованих) шарів, що знаходяться між ними, що показано на рисунку 1.8. Нейрони, що містяться в одному шарі не пов'язані між собою.

Допустимо, що нейронна мережа повинна прорахувати чи знаходиться на фото розміром 32x32 пікселі об'єкт певного класу. Даним способом можна обчислити кількість нейронів у вхідному шарі по кількості пікселів на фото 32 x 32 = 1024, а у вихідному – один нейрон, що розраховує відповідь 0..0,5 (так) чи 0,5..1 (ні). З іншої сторони дизайн прихованих шарів не є настільки прямолінійним і дослідники в сфері нейронних мереж розробили безліч евристичних рішень цієї проблеми.

1.6 Класифікація зображень

Класифікація зображень – це процес під час якого відбувається отримання класів інформації з багатоканального вхідного зображення. Згорткова нейронна мережа добре справляється із завданням катигоризації та знаходження об’єктів [9].

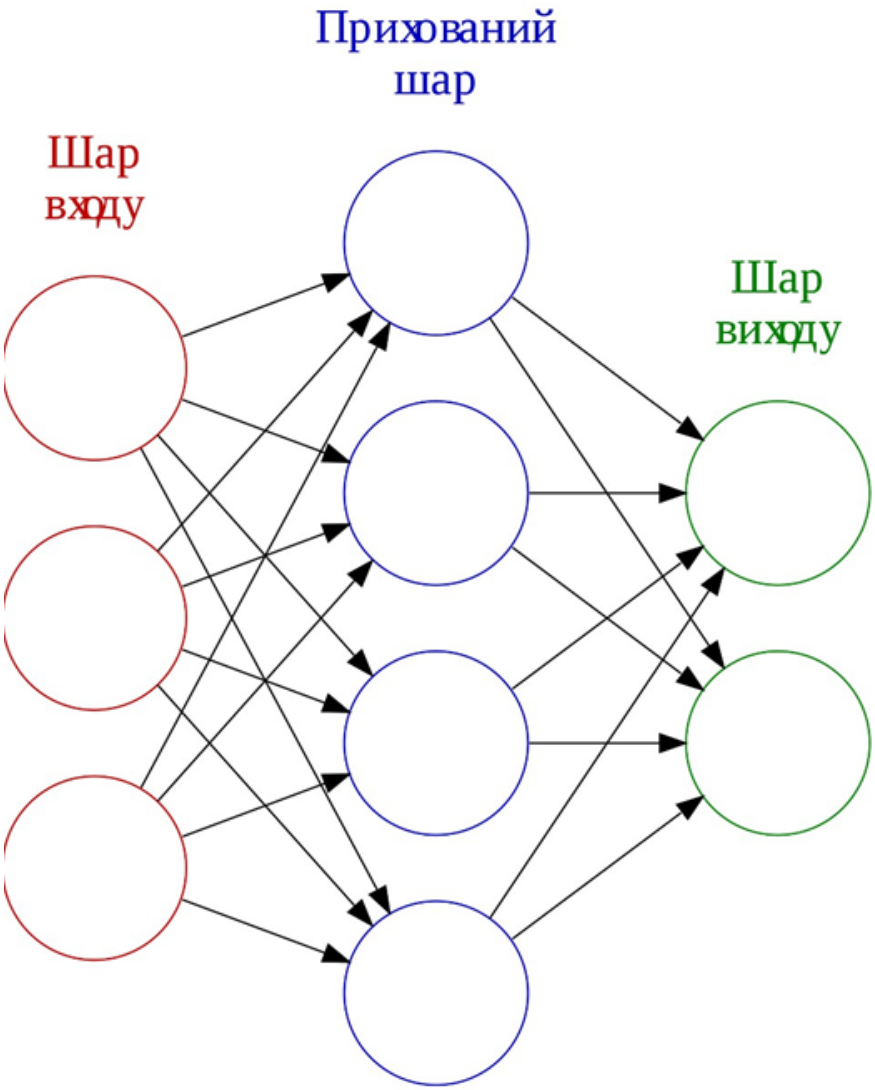


Рисунок 1.8 – Модель нейронної мережі [23]

1.6.1 Двовимірна згорткова мережа

У даній мережі є ядро, що являється матрицею ваг (weight matrix) "проходить" над двовимірним зображенням, в кожному елементі виконуючи операцію множення з тими початковими даними, над якими воно зараз знаходиться. В кінці ядро підсумовує всі обраховані значення в один кінцевий піксель [8].

Ядро повторює дану процедуру із кожним елементом, над яким воно "проходить". Далі воно перетворює двовимірну матрицю в іншу також двовимірну матрицю ознак, що зображено на рисунку 1.9.

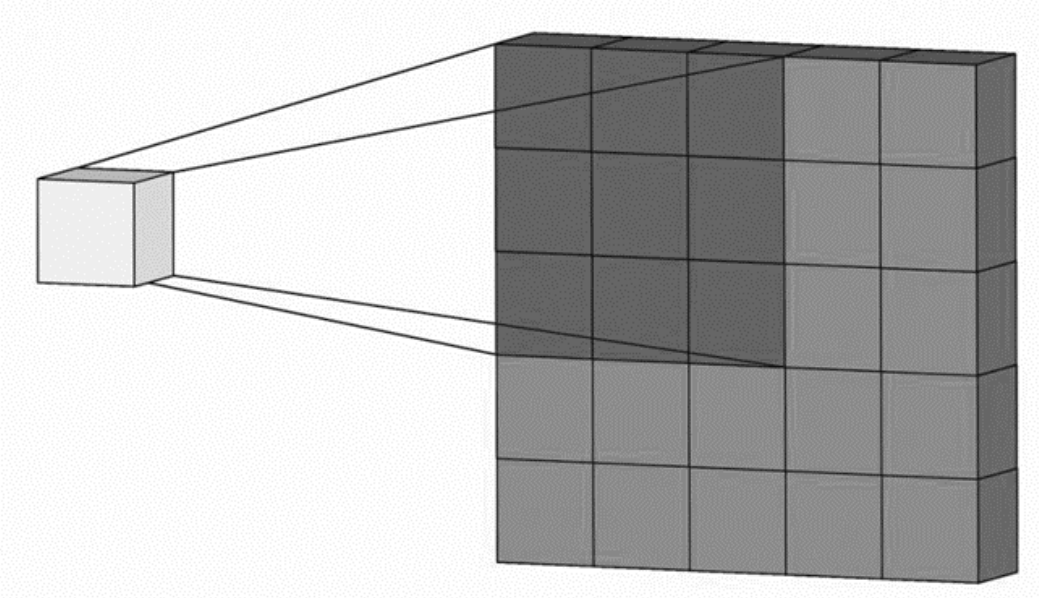


Рисунок 1.9 – Двовимірна згортка [16]

Ознаки наприкінці є зваженими сумами (де ваги є значеннями самого ядра) ознак на вході, що розташовані майже в тому ж місці, що і результуючий піксель на вхідному (початковому) шарі.

Не важливо, чи потрапляє вхідна ознака в приблизно те ж місце. Дана ознака визначається в залежності від того, чи знаходиться вона в зоні ядра, що генерує вихідні дані, чи ні. Зважаючи на це, розмір ядра згорткової нейронної мережі обраховує кількість ознак, що в кінцевому результаті будуть з'єднані для отримання нової ознаки на виході [13].

На рисунку 1.10 міститься зображення 5 x 5 = 25 ознак на початку і 3 x 3 = 9 ознак на виході. Для класичного шару ("standard connected layer") було б отримано матрицю ваг із 25 x 9 = 225 параметрів, а кожна обчислена (кінцева) ознака була би розрахованою сумою всіх вхідних ознак. Двовимірна згорткова мережа дозволяє зробити дану операцію всього із дев'ятьма параметрами, бо кожна ознака на виході обраховується аналізом не кожного признака на вході, а тільки одного вхідного, що знаходиться в "майже такому ж місці"[18].

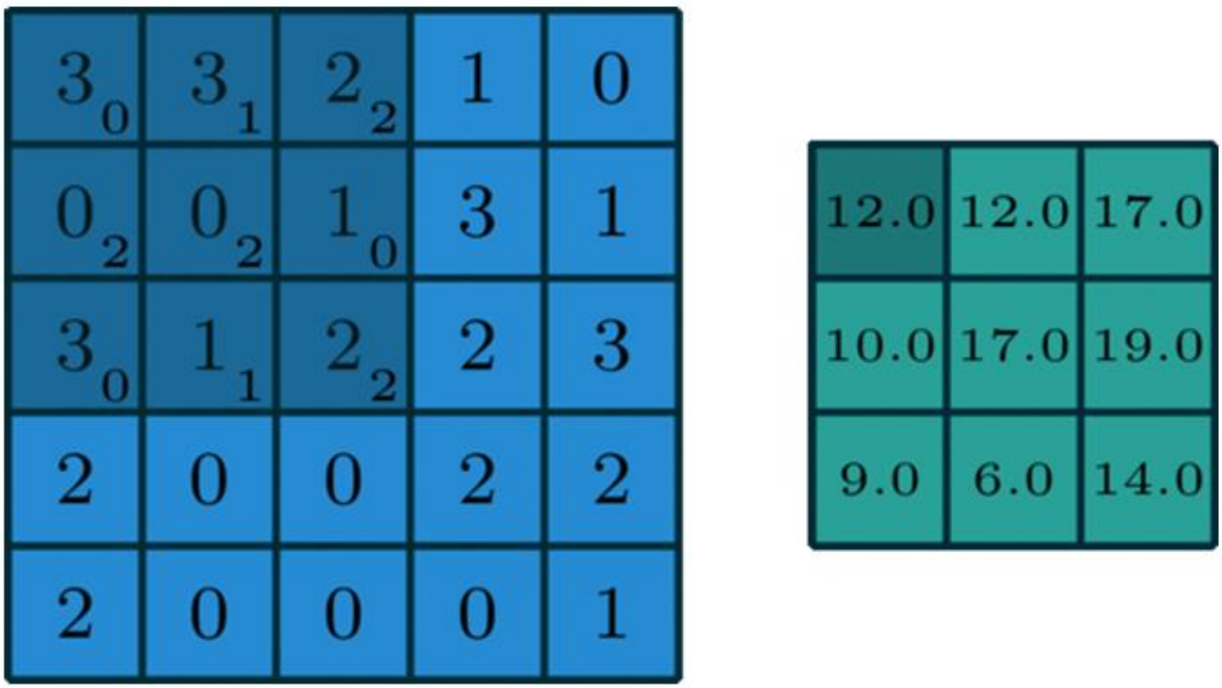


Рисунок 1.10 – Операція згортання [25]

1.6.2 Багатоканальна згорткова нейронна мережа

Рисунки, що знаходяться вище стосуються тільки того випадку, коли у зображення є тільки один вхідний канал. Зазвичай, у реальній роботі, зображення мають 3 вхідних канали.

Часто ми стикаємось з зображеннями із кольоровою моделлю RGB з трьома каналами, що зображено на рисунку 1.11.

Необхідно розуміти ключові відмінності у термінах : колм у випадку з 1 вхідним каналом терміни «фільтр» і «ядро» замінюють один одного, то в загальному випадку вони різні.

Кожен фільтр в дійсності являє собою колекцію ядер, причому для кожного окремого вхідного каналу даного шару є окреме одне ядро, і кожне ядро є унікальним [10,11].

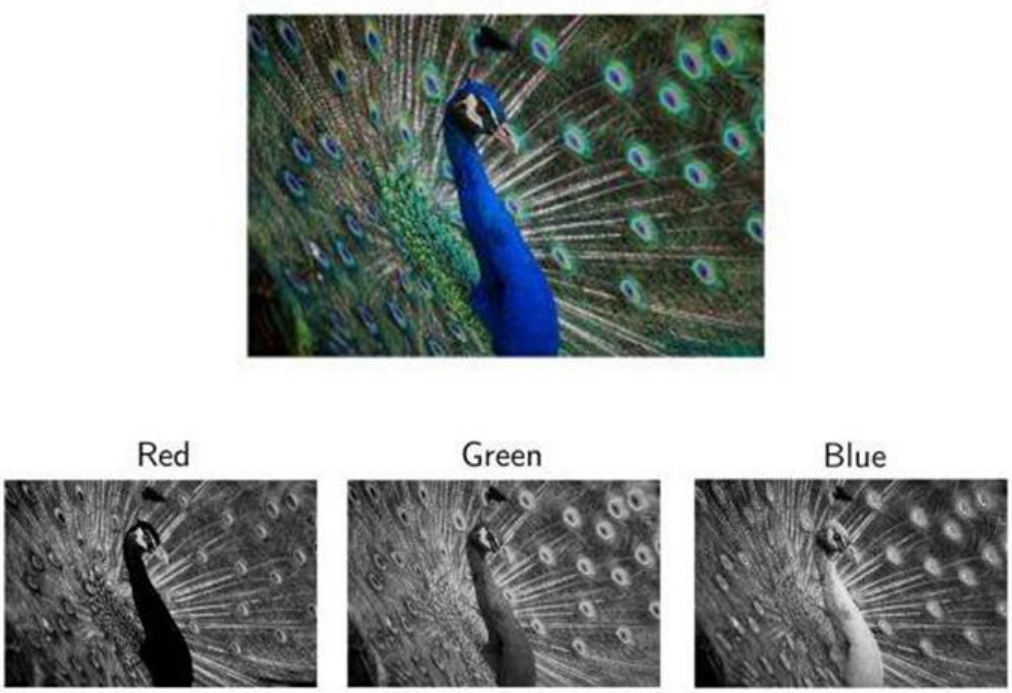


Рисунок 1.11 – зображення RGB з трьома каналами [19]

Для кожного фільтру в згортковому шарі створюється тільки один вихідний канал і роблять вони це так: кожне з ядер даного фільтра «проходить» по їх відповідним вхідним каналам і при цьому створює оброблену версію кожного з них, що зображено на рисунку 1.12.

Деякі ядра можуть мати більшу вагу, ніж інші. Це зроблено для того, щоб приділяти більше уваги необхідним вхідним каналам (наприклад, фільтр має можливість задати зеленому каналу ядра більшу вагу, ніж червоному та синьому, і тому сильніше реагувати на відмінності в образах з зеленого каналу).

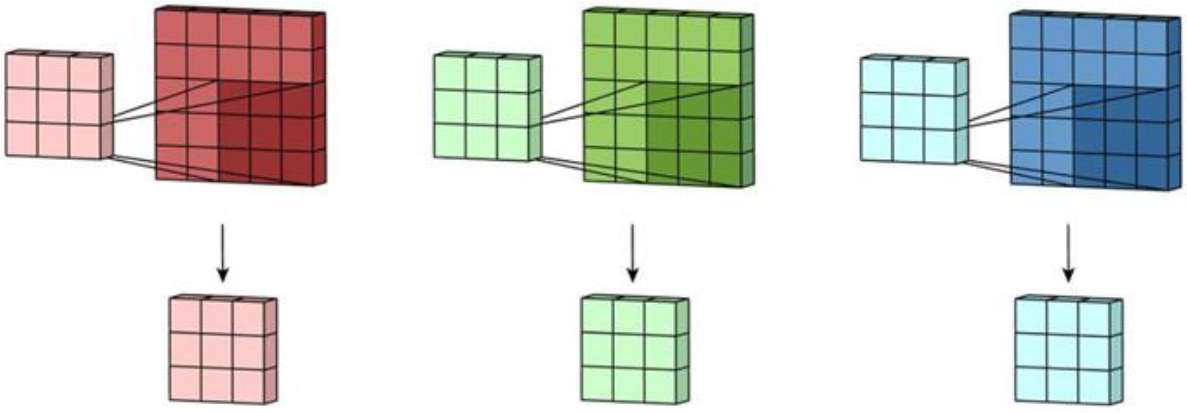


Рисунок 1.12 – Фільтр створює тільки один канал[16]

Далі кожна з оброблених в каналі варіантів сумуються разом для формування спільного каналу. Ядра кожного фільтра утворюють один варіант кожного каналу, а фільтр створює єдиний спільний вихідний канал, як це зображено на рисунку 1.13.

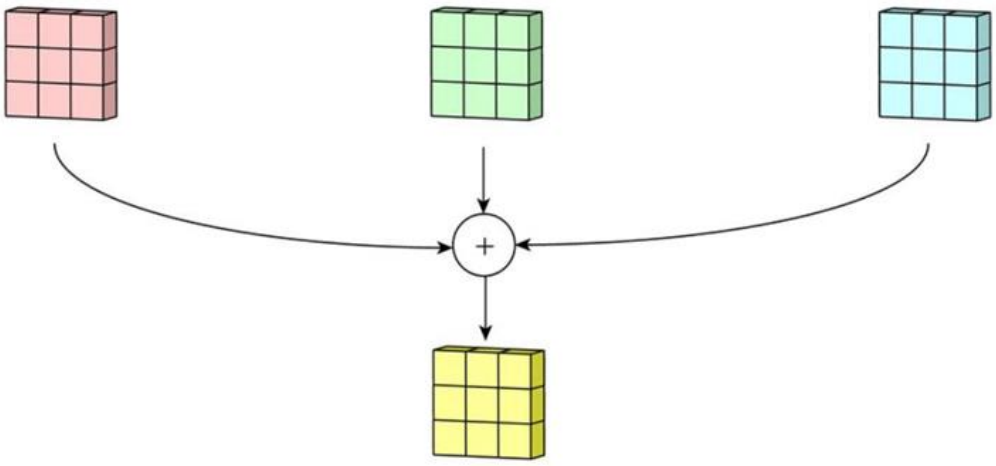


Рисунок 1.13 – Загальний вихідний канал [25]

На наступному кроці роботи нейронної мережі зсув додається до вихідного каналу для генерування остаточного вихідного каналу, як це зображено на рисунку 1.14.

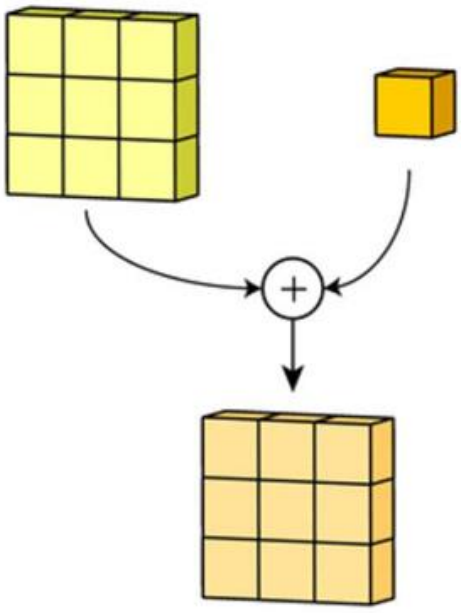


Рисунок 1.14 – Зсув додається до вихідного каналу[16]

Далі вони об'єднуються разом для генерування загального виходу, з умовою, щл кількість вихідних каналів така сама як і число фільтрів. При цьому часто використовується нелінійність перед початковим входом іншого шару згортки, що потім повторює даний процес.

Згорткові нейронні мережі надають можливість комп'ютерному зору виконувати не тільки прості завдання, але й складні продукти та послуги, починаючи від керування транспортним засобом і закінчуючи покращенням діагностування медичних захворювань. Дані нейронні мережі відіграють важливу роль в комп'ютерному зорі, як хороший засіб для виконання розпізнавання та класифікації об’єктів, що є ключовим завданням програмного продукту.

1.7 Рентгенографія

Важкий епідеміологічний стан в світі і швидке розповсюдження коронавіруса Covid-19 визначає високу значимість апаратної діагностики, а саме комп'ютерної томографії, як способу, що може ідентифікувати головні прояви Covid-19 (зони в легенях по типу «матового скла») навіть на початкових стадіях їх появи, що позволяє вчасно почати контролювати перебіг хвороби.

КТ (комп'ютерна томографія) – це метод ренгенівського сканування. Підозрюючи ураження коронавірусом Covid-19 сімейний лікар у поліклініці (приватній чи державній) при необхідності направляє на КТ легенів.

Комп'ютерна томографія легенів – це дуже точний метод діагностики дихальних органів, що складають основу системи дихання людини.

Комп'ютерний томограф складається з таких елементів як рентгенівська трубка і детектори.

На першому кроці рентгенівська трубка випромінює рентгенівський промінь, що проходить крізь пацієнта. Даний промінь вловлють детектори і реконструюєть для одержання зображення в 2D і 3D.

Рисунок 1.15 – Процес комп’ютерної томографії [22]

Інакше кажучи, томограф обертається навколо людини, в процесі цього робить знімки, далі вони опрацьовуються комп'ютером. Отримані результати аналізує, розшифровує та досліджує лікар.

Комп’ютерну томографію часто проводять із використанням контрастного матеріалу, шо допомагає оприділити в людськму організмі структури однакової щільності. Комп’ютерна томографія пропонує кращу диференціацію між різними щільностями людських органів, ніж рентгенограма.

Як і у випадку звичної рентгенографії, в КТ для одержання зображень використовується рентгенівське випромінювання, проте дози випромінювання під час проведення комп’ютерної томографії вищі через неодноразову експозицію.

До слова, високе променеве навантаження на пацієнта (хоч його і вдалося значно знизити за час існування цього методу діагностики) є основним недоліком комп'ютерної томографії. Крім цього, проведення комп'ютерної томографії збільшує кількість випадків виникнення пошкоджень у ДНК організму.

Протипоказання для КТ

- вагітність у жінок чи період лактації. У Міжнародній організації охорони здоров'я вважають, при можливості, відкласти комп'ютерну томографію на термін після періоду вагітності чи використовувати такі методи візуалізації, що не засновані на шкідливому випромінюванні. До таких відносяться ультразвукове дослідження та магнітно-резонансна томографія;
- проведення комп'ютерної томографії з використанням контрастного матеріалу протипоказано для хворих із нирковою недостатністю, багаторазовою мієломою (злоякісна пухлина із плазматичних клітин) та серйозними захворюваннями серцевосудинної системи (постійною серцевою недостатністю чи аневризмі гирла аорти);
- при сильному діабеті, зневодненні, ураженні лімфатичної системи, захворюваннях щитовидної залози;
- вага тіла, що є більгою максимальну для приладу комп'ютерної томографії (томографи розраховані на невелику масу, стандартне обмеження до 150 кг);
- дуже часте проходження процедур із рентгенівським опроміненням.

Відносно обмежень за віком, то їх як таких немає. проте враховуючи дозу рентгенівського випромінювання комп'ютерного томографа, багато лікарів не рекомендують процедуру для осіб до 3 років. Дітям комп'ютерну томографію необхідно призначати тільки при наявності серйозних захворювань.

Зазвичай після проведення комп'ютерної томографії не спостерігається ускладнень. Вони можуть виникнкти внаслідок застосування рентген-контрастних матеріалів. Наприклад, при використанні йодовмісних контрастних матеріалів слід заздалегідь знати про наявність непереносимості чи алергії.

Переваги КТ:

- дана процедура є безболісною. Комп'ютерна томографія триває протягом декількох хвилин. За цей короткий період часу хворий отримує тривимірний чи двовимірний знімок ділянки дослідження;

- найбільш точний діагностичний метод захворювань різних органів: головного та спинного мозку, скелету людини, органів середостіння, печінки, серця, підшлункової залози, щитовидних залоз і легенів.

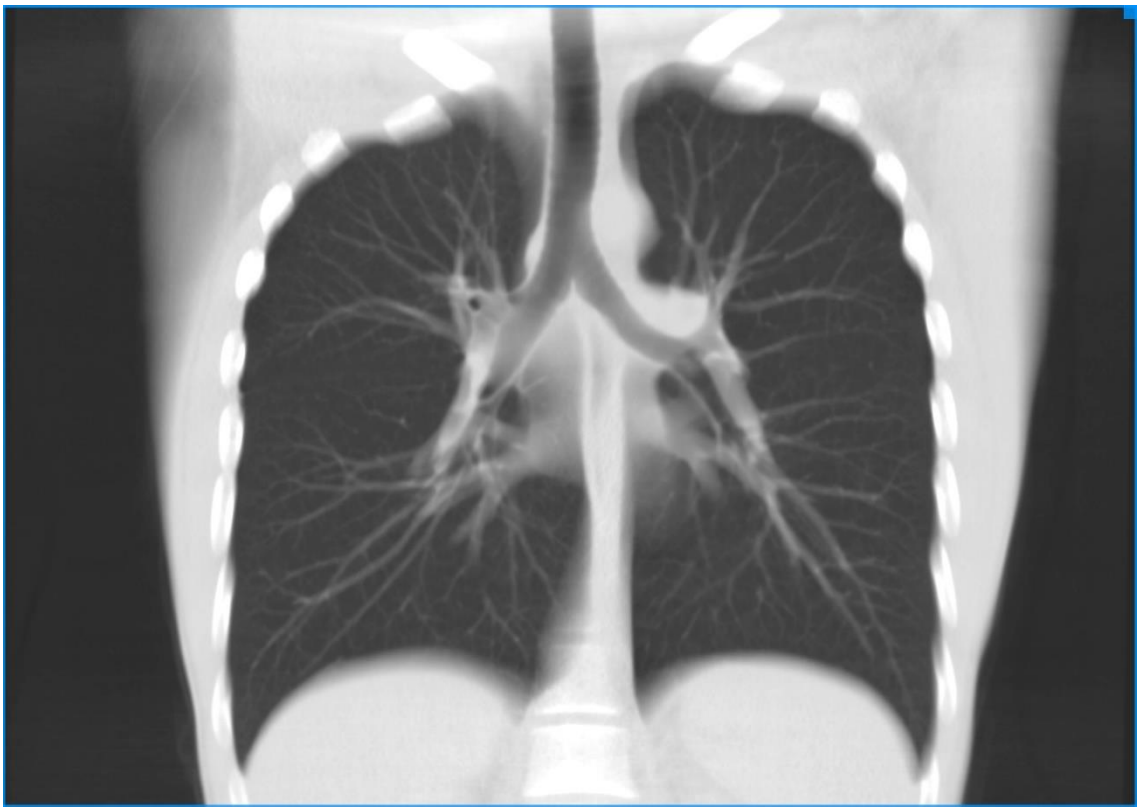


Рисунок 1.16 – Приклад рентгенографії легень [20]

При дослідженні легень, даний метод діагностики дає більше інформації про їх стан, ніж рентген або флюорографія. Комп'ютерна томографія покаже більшість видів запальних захворювань дихальних шляхів.

На комп'ютерній томографії крім легенів також добре видно бронхи, альвеоли, трахею, аорту, серцеві судини, порожнисті вени.

Застосування багатоспірального сканера комп'ютерній томографії для діагностики легень дозволяє не тільки використовуючи рентгенівськ променів зробити знімки органів, яких не буде видно при магнітно-резонансній томографії, але й отримати тривимірну модель органу, що на відміну від рентгену легень тільки в одній проекції. Використовуючи тривимірні зображення та моделі, комп'ютерну томографію легенів, лікар-рентгенолог має можливість:

- перевірити зони ущільнення легеневих тканин;

- побачити на комп’ютерній томографії ознаки нездорових змін, що є індикатором наявності пневмонії, вірусних захворювань чи коронавірусів;

- перевірити стан бронх, наявність їх ушкоджень;

- поміряти розміри легеневих артерій, серцевої аорти і альвеол;

- перевірити наявність засвітів і їх розміри при дослідженні комп’ютерною томографію тканин грудної клітини.

При виявлення ознак вірусних захворювань чи коронавірусів після проведеної комп’ютерної томографії органів грудної клітки лікар-рентгенолог надає рекомендації щодо подальших дій.

Комп'ютерна томографія при виявленні симптомів лікарем є основним і першочерговим методом діагностики при підозрі на коронавірус або вірусну пневмонію. На комп'ютерній томографії легенів зображено зміни в тканинах грудної клітини, їх об’єм, розміри і масштаб поширеності. Аналізуючи дані комп'ютерної томографії легенів, наприклад, про наявність односторонньої моносегментарної пневмонії при наявності лабораторного обстеження виключається або підтверджується інфекціям Covid-19.

Враховуючи те, що коронавірусна інфекція часто проходить без симптомів захворювання, щоб виключети ознак вірусної пневмонії на початковому етапі як показника Covid-19, проходження комп'ютерної томографії легень має рекомендаційний характер. Діагностування Covid-19 на початковій стадії його розвитку дає більший спектр для контролю перебігу поширення вірусу і запобігання серйозних ускладнень.

Процедура комп'ютерної томографії триває протягом 5-6 хвилин, під час яких більшу частину часу спеціаліст вкладає пацієнта у вірне положення, щоб зробити точну комп'ютерну томографію органів грудної клітини. Під час комп’ютерної томографії потрібно затримати дихання 4 рази на 5 секунд. Під час наступного кроку система виконує реконструкцію зображення і хворий покидає кабінет комп’ютерної томографії.

Перший етап діагностики на ураження коронавірусом Covid-19, включає проведення комп'ютерної томографії на зміни такого типу як "матове скло".

Комп'ютерна томографія на Covid-19 показує 4 рівні важкості запалення в легенях:

- I рівень важкості – легка. Вражається до 25% легеневої паренхіми;
- II рівень важкості – середня. Вражається до 50% легеневої паренхіми;
- III рівень важкості – важка. Вражається до 75% легеневої паренхіми;
- IV рівень важкості – вкрай важка. Вражається більше 75% легеневої паренхіми.

Перший і другий рівні важкості означають подальше лікування вдома в умовах самоізоляції. При третьому та четвертому рівні важкості необхідно лікуватись в умовах стаціонару, необхідна госпіталізація.

Комп'ютерна томографія легень є пергочерновою та дуже важливою складовою комплексного підходу для діагностики коронавірусної інфекції, Covid-19. Так, діагностика вірусу Covid-19 базується не тільки на комп'ютерній діагностиці легень, але і потребі зробити тест полімеразно ланцюгової реакції (аналіз РНК) чи тест на антитіла IgM та IgG, тоді коли підтверджується ураження коронавірусом Covid-19.

1.7.1 Сучасні томографи

Далі розглянуто три найпопулярніші апарати КТ, що використовуються у медзакладах.

1.7.1.1 Апарат КТ Emotion 6

Emotion 6 надає користувачеві унікальні за зручністю і продуктивністю клінічні інструменти інтерактивної візуалізації анатомічного сканування. Дозволяє проводити скринінгові обстеження легенів і товстого кишечника, віртуальну ендоскопію, перфузію, інтервенції під контролем КТ, кількісну оцінку васкулярних порушень, і багато іншого.

Апарат забезпечує високу якість зображень, мінімальне променеве навантаження, а також оснащений ефективною повітряною системою охолодження. Апарат зображений на рисунку 1.17.

Рисунок 1.17 – Апарат КТ Emotion 6 [20]

Базова комплектація включає потужні засоби тривимірної постобробки зображень, ангіографії та мультипланарної реконструкції.

Emotion 6 має розширений набір додаткових програмних і програмно-апаратних модулів, які вирішують повний спектр діагностичних завдань в хірургії, онкології, неврології і нейрохірургії, в відділеннях швидкої допомоги, а також в педіатрії та інших клінічних областях.

Emotion дозволяє проводити рутинні обстеження головного мозку, КТ-ангіографію з контрастним посиленням, повний комплекс обстежень, пов'язаних з діагностикою та кількісною оцінкою васкулярних порушень, включаючи діагностику вроджених судинних мальформацій, як нормальних вен (венозна ангіома) так і артерій безпосередньо переходять у вени (артеріо -венозна мальформація або АВМ), динамічні функціональні дослідження порушень перфузії головного мозку при оцінці інсультів і пухлин на ранній стадії.

Крім того, система забезпечує велику швидкість субміліметрового анатомічного покриття, що визначає відмінну візуалізацію при проведенні досліджень судин з контрастним посиленням, динамічних функціональних досліджень в режимі сканування без руху столу пацієнта. Висока деталізація анатомічних структур дозволяє застосовувати апарати класу Emotion при дослідженні кісткової тканини в травматології та ортопедії.

Сканери класу Emotion дозволяють проводити обстеження всього тіла пацієнта, включаючи рутинні радіологічні обстеження, скринінгові онкологічні обстеження і дослідження перфузії внутрішніх органів, і навіть базові обстеження в області кардіології, такі, як оцінка ступеня кальцифікації коронарних судин.

1.7.1.2 Апарат КТ Ingenuity Elite

У Ingenuity Elite 128, Philips використовує ряд інтегрованих технологій, що дозволяють знизити променеве навантаження і обсяг введеного контрастної речовини без будь-яких компромісних рішень і зі збереженням діагностичної якості зображень. Апарат зображений на рисунку 1.18.

Рисунок 1.18 – Апарат КТ Ingenuity Elite [21]

Четверте покоління технології зниження дози iDose4 дозволяє істотно знизити рівень шуму на зображеннях і помітно поліпшити візуалізацію м'яких тканин і судин при зниженні дози на величину до 80%.

Функція SyncRight підвищує узгодженість введення контрастної речовини і дозволяє знизити його дозу на величину до 15%.

Комп'ютерний томограф Ingenuity дозволяє швидше отримувати зображення для аналізу.

При використанні алгоритму реконструкції, що входить до складу технології iDose4, зображення реконструюються за 60 секунд і більш короткий час для 67% протоколів дослідження.

Клінічні можливості апарату КТ Ingenuity Elite:

– збільшення просторової роздільної здатності при тому ж рівні дози до 68%;

– зниження променевого навантаження до 50% при збільшенні просторової роздільної здатності 35%;

– зниження променевого навантаження до 80% при тій самій діагностичній якості зображень;

– персоналізовані ін'єкції контрастної речовини для підтримки високої якості зображень.

Для охолодження трубки MRC Ice використовується система рідинного охолодження, що дозволяє проводити термінові дослідження пацієнтів без перерви.

MRC Ice є однією з найнадійніших в галузі. Вона розрахована на великий обсяг досліджень і цілодобову роботу. Трубці не потрібно часу на нагрів перед проведенням дослідження і на охолодження після завершення сканування

Система реконструкції зображень RapidView IR працює на 137% швидше аналогічної системи першого покоління, встановленої на томографах Brilliance.

У стандартному режимі реконструкції вона займає менше 60 секунд для 86% протоколів досліджень.

При використанні технології iDose4 реконструкція займає менше 60 секунд для 67% протоколів досліджень.

Система Ingenuity розрахована на широку інтеграцію з інформаційними системами HIS / RIS, ін'єктором контрастних речовин, системою PACS, робочою станцією Brilliance Workspace, рішеннями Intellispace Portal і іншими пристроями. Ці можливості дозволяють оптимізувати робочий процес і підвищити якість медичної допомоги пацієнтам.

1.7.1.3 Апарат КТ BrightSpeed 16.

Нові апаратні і програмні рішення дозволили створити унікальний з точки зору зручності і ергономіки простір. Можливість оптимізації робочого місця для дослідження пацієнтів в положеннях сидячи і лежачи сприяє зменшенню навантаження і зняття втоми. Крім того, консоллю можна керувати віддалено. Апарат зображений на рисунку 1.19.

Рисунок 1.19 – BrightSpeed 16 [22]

Технологія Xtream FX дозволяє обробляти до 16 зображень в секунду з повною реконструкцією всієї структури, через яку проходить рентгенівський

пучок конічної форми. Алгоритм мультипланарної реконструкції дозволяє проспективно реконструювати аксіальні, корональні і косі проекції.

Томограф серії BrightSpeed дозволяє отримувати високоякісні зображення всіх областей, від голови до пальців ніг в широкому діапазоні додатків, необхідних для лікарів.

Сканер BrightSpeed забезпечує оптимальну швидкість, потужність необхідні для неінвазивної діагностики раку прямої кишки і захворювань серцево-судинної системи, а також багатьох інших досліджень. Крім того, за рахунок зменшення розмірів багатьох компонентів він поміщається на такій же площі, як і «одношарові» сканери.

Покращені умови роботи для оператора дозволяють забезпечити догляд за пацієнтами.

Апарат дозволяє проводити дослідження в положеннях лежачи і сидячи, а також у вертикальному і горизонтальному положеннях, дає можливість збільшити кількість досліджень і допомагає розмістити пацієнта в зоні прямої видимості.

Томограф BrightSpeed забезпечує високу роздільну здатність зображень – до 0,35 мм у всіх напрямках при оптимальній дозі. Цей рівень якості досягається завдяки матричному детектору і новій цифровій системі отримання даних GE Volara, яка сприяє збільшенню швидкості їх обробки і знижує електронні шуми на 30-40 відсотків.

Томограф BrightSpeed всього за кілька хвилин отримує всю необхідну інформацію. Завдяки високій швидкості обертання (один оборот за півсекунди) і регульованої швидкості сканування істотно скорочується тривалість затримки дихання пацієнтів.

1.8 Висновки до розділу 1

В основі штучного інтелекту та комп’ютерного зору, закладена можливість проектування та розроблення комп’ютерних систем, що можуть аналізувати

фото та відео документи. Дана сфера науки вивчає такі програмні продукти, що можуть класифікувати та розпізнавати об’єкти.

У даному розділі вивчена теорія виявлення об’єктів, які містяться на зображенні, що дозволяє їх класифікувати. Аналіз методів виконання роботи комп’ютерного зору показав, що використання нейронних мереж має найбільше переваг. Способи роботи їхніх нейронів показують перевагу над лінійними алгоритмами. Це визначається такими параметрами як функція активації і вага.

Проаналізовані принципи роботи і навчання нейронних мереж дозволяють зробити висновок, що найефективнішої роботи розпізнавання та класифікації об’єктів можна досягти згортковими нейронними мережами.

2 ПРОЕКТУВАННЯ СИСТЕМИ

2.1 Опис предметної області

Теорія розпізнавання об’єктів – це підрозділ комп’ютерних наук та суміжних дисциплін, що досліджує методи класифікації та розпізнавання об'єктів різного типу: сигналів, предметів, ситуацій що характеризуються достатньою кількістю деяких ознак.

Наука про розпізнавання об’єктів також віднесена в розділ міждисциплінарних досліджень – в тому числі містить роботу зі створення нейронних мереж, і доволі часто застовується для вирішення прикладних завдань у галузі комп'ютерного зору. При постановці типової задачі розпізнавання об’єктів необхідно використовувати математичну мову, спираючись на логічні міркування і математичні принципи. Протилежними до даного підходу, використовують методи детекції об’єктів із використанням Machine Learning і штучних нейронних мереж, що створені на не таких формалізованих підходах до завдань розпізнавання, і зазвичай демонструють не гірший, а часто і набагато точніший результат.

Клас – це масив об'єктів, що мають однпкові властивості. Для об'єктів, що наледжать до одного класу необхідна наявність «схожості». Для завдання розпізнавання об’єків можло визначити будь-яку кількість класів, один чи більше. Кожен клас ідентифікується своєю міткою класу.

Класифікація – процес присвоєння міток класу відповідним об'єктам, згідно із таблицею властивостей даних об'єктів. Класифікатор – це інструмент, що присвоє мітки класам, який в якості початкових даних отримує список властивостей об'єкта. До одного з найпопулярніших методів класифікації можна віднести такий, що спеціалізується на класифікації об'єктів з використанням ознак, під час роботи якого кожен об'єкт має власний набір числових або текстових ознак. Проте існують такі дані для яких числові чи текстові ознаки не дають достатньої точності класифікації, наприклад, кольои пікселів зображень

чи оцифрований аудіо сигнал. Загальна класифікація фотографій дерев і будівель є простою задачею для людини і в той же час складною для обчислювальних приладів. Причиною цього є уміння людини розуміти «приховані ознаки», що недоступні для машини, такі як крона дерева чи дах будівлі.

Прикладами задач класифікації об’єктів є розпізнавання слів, діагностування медичиних захворювань, виявлення проблем із здоров’ям у популяцій тварин, розпізнавання обличь, класифікація документації, тощо.

Найчастіше початковим даними для класифікації об’єктів на зображені є зроблене із камери фото. Завданням класифікації об’єктів є одержання таких векторів, що складаються із ознак для всіх класів на фотографії. Процес кодування полягає в тому, що необхідно присвоїти значення кожної ознаки із їхнього масиву для кожного класу.

2.2 Визначення вимог і завдань

Основним завданням програмного продукту є класифікація об’єктів на фотографії. Для демонстрації роботи вирішено розробити систему, що дозволяє розпізнавати легеневі захворювання. Дана система обробляє рентген знімки за допомогою нейронних мереж і видає результат хвора людина чи ні. Інтерфейс організований у вигляді мобільного додатку для пацієнта та комп’ютерного додатку для лікаря.

Отже, основними функціями програмного продукту є:

- розпізнавання хвороби;
- обліковий запис для пацієнта;
- обліковий запис для лікаря;
- зберігання даних;
- відображення історії розпізнавання;
- відправка даних на розпізнавання та отримання результату в режимі реального часу.

Основні вимоги до додатку:

- точність;
- зрозумілий дизайн;
- універсальність;
- легка інтеграція;
- система повинна показувати повідомлення про помилку користувачеві, якщо виникають проблеми з підключенням до сервера.

2.3 Опис функціоналу програмного продукту

Зі сторони пацієнта система повинна надавати можливість виконувати наступні дії і схематично зображена на рисунку 2.1:

1) реєстрація профілю у системі. Для вдалої операції необхідно ввести унікальну електронну адресу, що буде виступати логіном у системі, своє ім’я, та прізвище, а також пароль;
2) авторизація до системи. Для цього потрібно ввести логін і пароль, які були вказані при реєстрації профілю;
3) відновлення паролю;
4) переглядати інформацію про медичні заклади, де проводились обстеження;
5) переглядати інформацію про свої обстеження;
6) самому надсилати фото для аналізу;
7) змінювати дані профілю.

Зі сторони лікаря система повинна надавати можливість виконувати наступні дії і схематично зображена на рисунку 2.2:

1) реєстрація профілю у системі. Для вдалої операції необхідно ввести унікальну електронну адресу, що буде виступати логіном у системі, своє ім’я, та прізвище, а також пароль;
2) авторизація до системи. Для цього потрібно ввести логін і пароль, які були вказані при реєстрації профілю;

3) відновлення паролю;
4) переглядати інформацію про своїх пацієнтів;
5) додавати своїх пацієнтів;
6) зробити рентген знімок і система автоматично проаналізує його;
7) додати знімок для аналізу;
8) змінювати дані профілю.

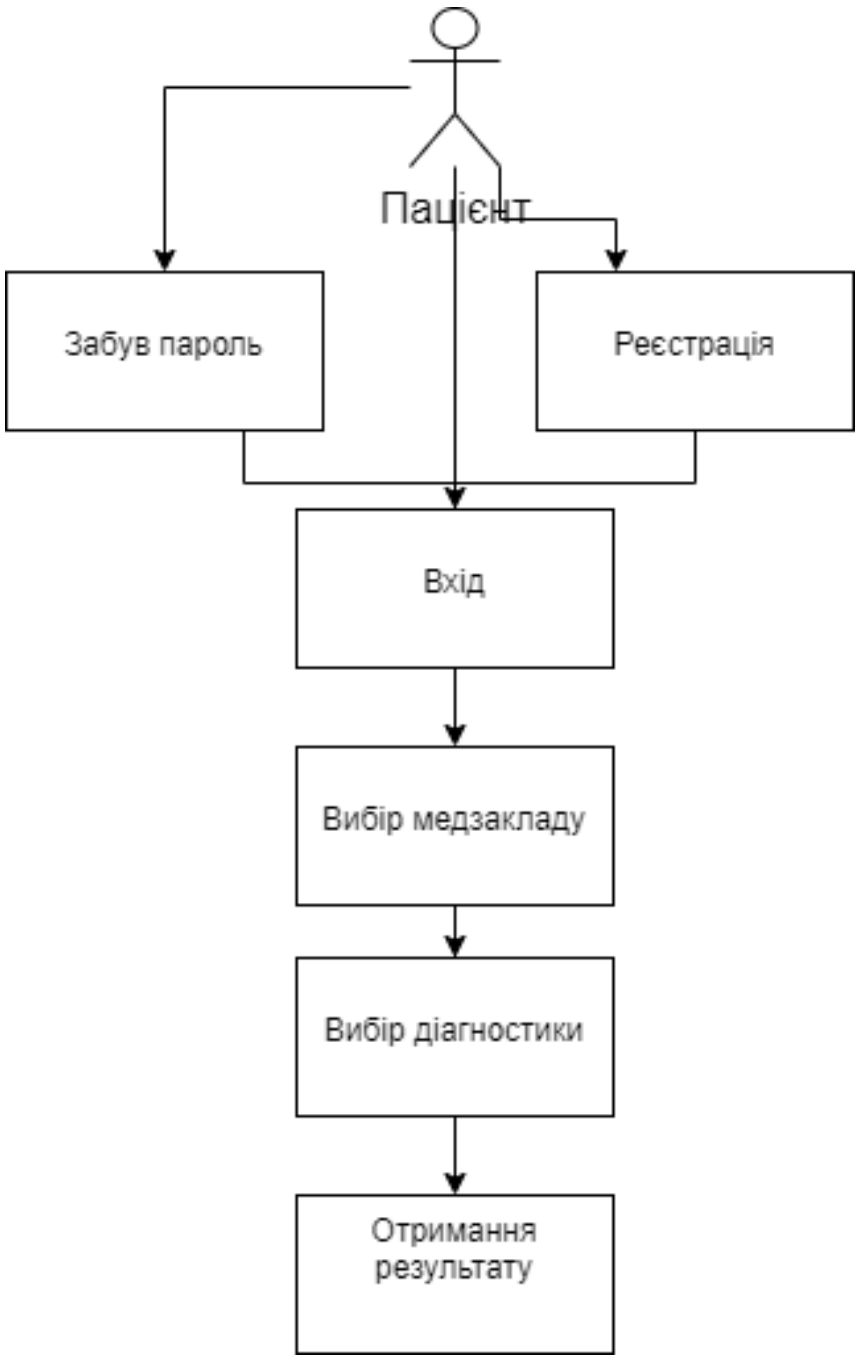


Рисунок 2.1 – Сценарій роботи пацієнта

Рисунок 2.2 – Сценарій роботи лікаря

2.4 Структура системи

Роботу комп’ютерної системи можна розділити на дві частини і схематично зображена на рисунку 2.3:

- front-end у вигляді мобільного та десктопного застосунків;
- back-end у вигляді веб застосунку, що разом із базою даних та нейронною мережею знаходяться на сервері.

Рисунок 2.3 – Зв’язок між вузлами комп’ютерної системи

2.5 Організація Back-end частини

Back-end частина починає свою роботу після натиску на кнопку “Надіслати фотографію”. Після цього програмний продукт надсилає фотографію на серевер, де проходитиме аналіз. Далі відбуватиметься попередня обробка фотографії. На наступному кроці програмний продукт сегментує зображення. Дана процедура дозволяє отримати ознаки. Використовуючи даний результат можна проводити класифікацію зображення. Програма записує ці дані. Далі оновлюються дані на пристроях лікаря та пацієнта. Робота схематично зображена на рисунку 2.4.

Рисунок 2.4 – Зв’язок між вузлами комп’ютерної системи

2.6 Розроблення графічного інтерфейсу

Перш за все необхідно розробити дизайн мобільного застосунку. Перша сторінка міститиме такі поля як ім’я, прізвище, e-mail, номер телефону.

Сторінка реєстрації зображена на рисунку 2.5.

Рисунок 2.5 – Реєстрація пацієнта

При правильному введені даних користувач отримає повідомлення, що зображене на рисунку 2.6.

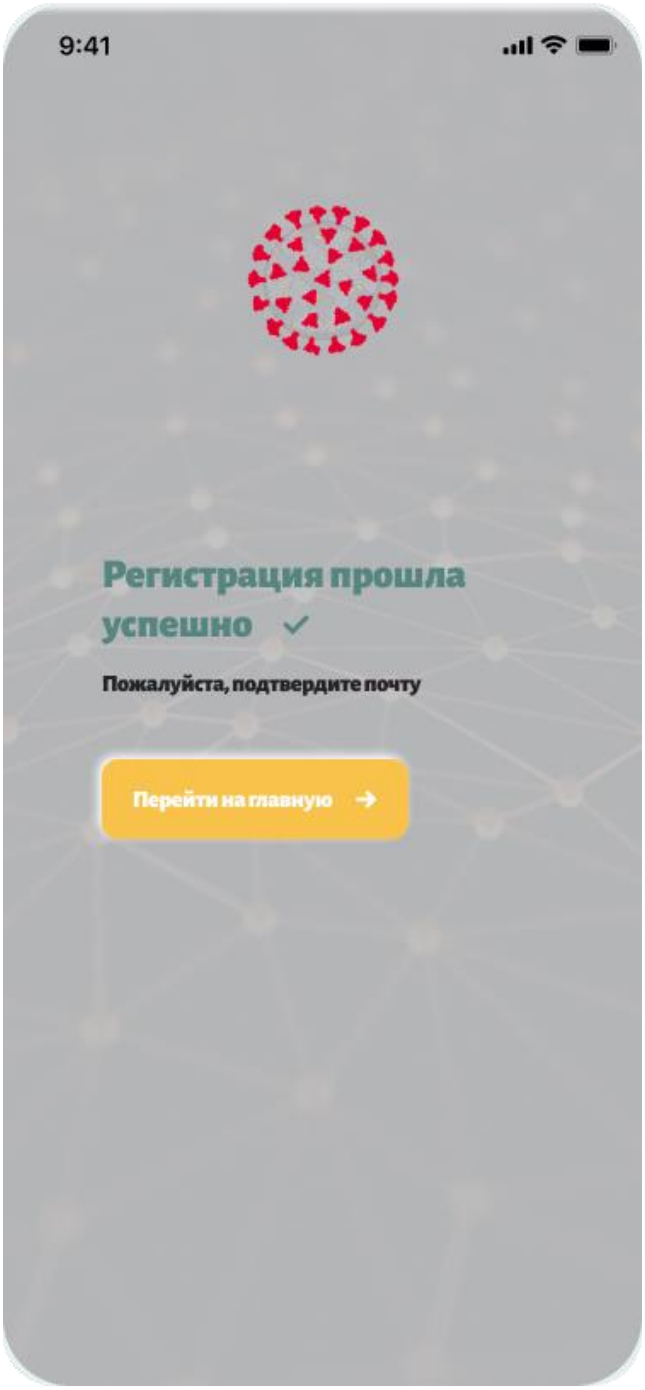


Рисунок 2.6 – Успішна реєстрація

Користувачу доступна сторінка авторизації, де необхідно внести номер телефону і пароль для входу. Сторінка входу зображена на рисунку 2.7.

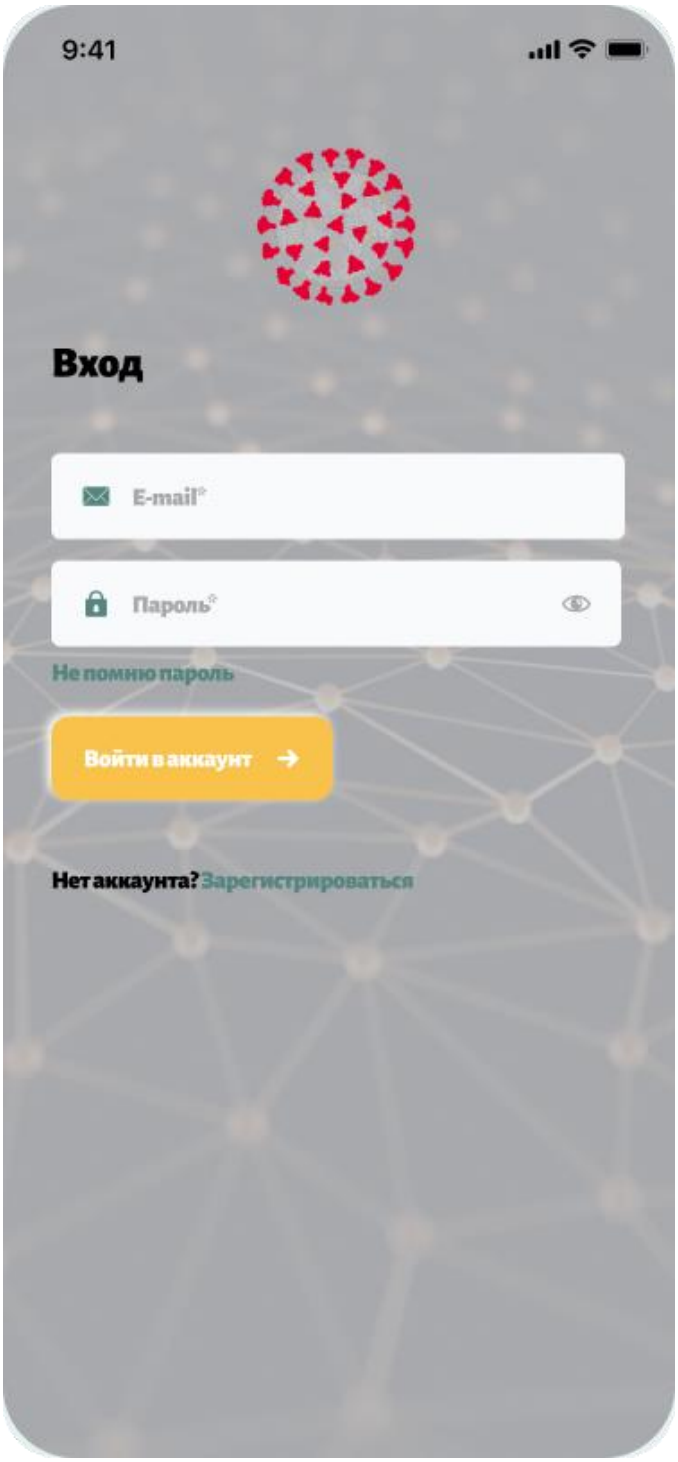


Рисунок 2.7 – Сторінка входу

Якщо користувач забув пароль, він може скористатись сторінкою відновлення пароля, де можна змінити пароль.

Сторінки відновлення пароля на рисунках 2.8 -2.10

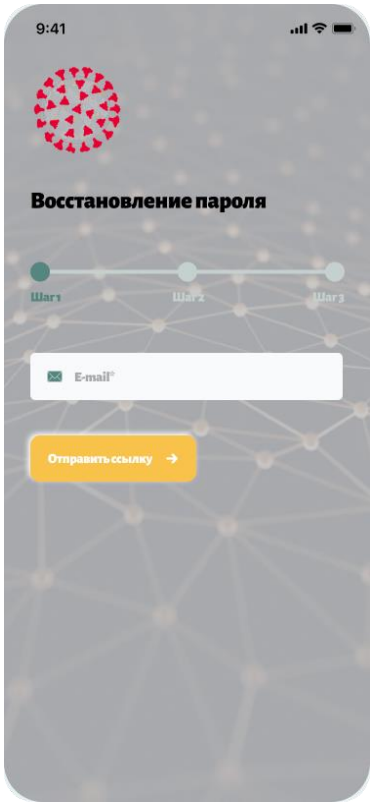


Рисунок 2.8 – Відновлення паролю крок 1

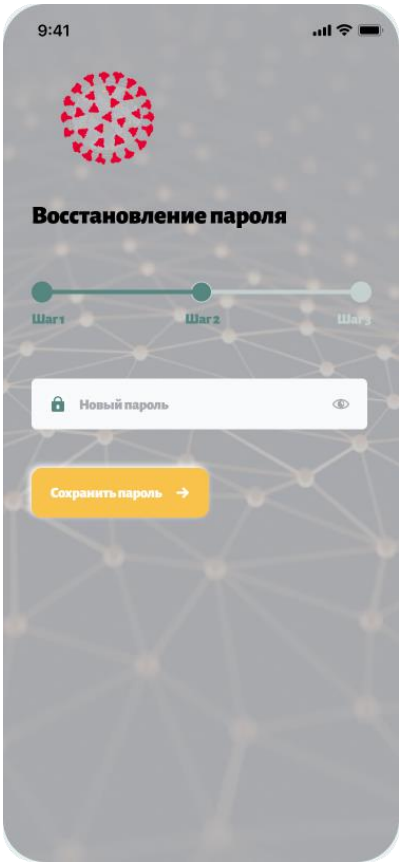


Рисунок 2.9 – Відновлення паролю крок 2

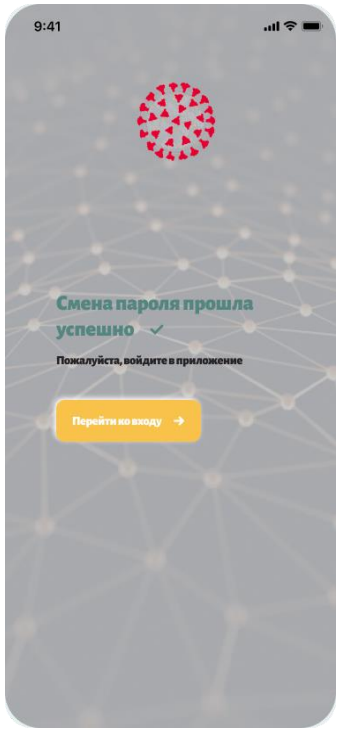


Рисунок 2.10 – Відновлення паролю крок 3

Наступна сторінка, що доступна користувачеві це сторінка зі списком медзакладів, де проходив обстеження пацієнт, що видно з рисунку 2.11.

Рисунок 2.11 – Сторінка медзакладів

Наступна сторінка, що доступна користувачеві це сторінка зі списком його рентген знімків. Ця сторінка зображена на рисунку 2.12.

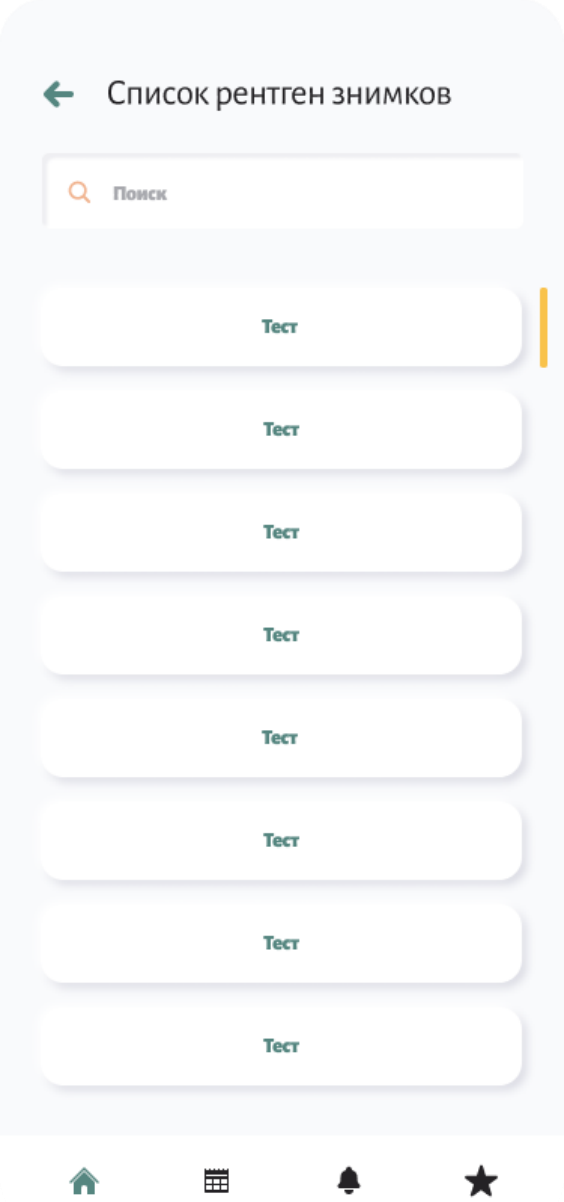


Рисунок 2.12 – Сторінка зі списком знімків

Наступна сторінка, що доступна користувачеві це сторінка із результатами аналізу, вона зображена на рисунку 2.13

Розробка дизайну комп’ютерного додатку для лікаря.

Користувачу доступна сторінка авторизації, де необхідно внести номер телефону і пароль для входу.

Сторінка авторизації зображена на рисунку 2.14.

Рисунок 2.13 – Сторінка із результатом

Наступна сторінка, що доступна користувачеві це сторінка зі списком його пацієнтів.

Сторінка із пацієнтами лікаря зображена на рисунку 2.15.

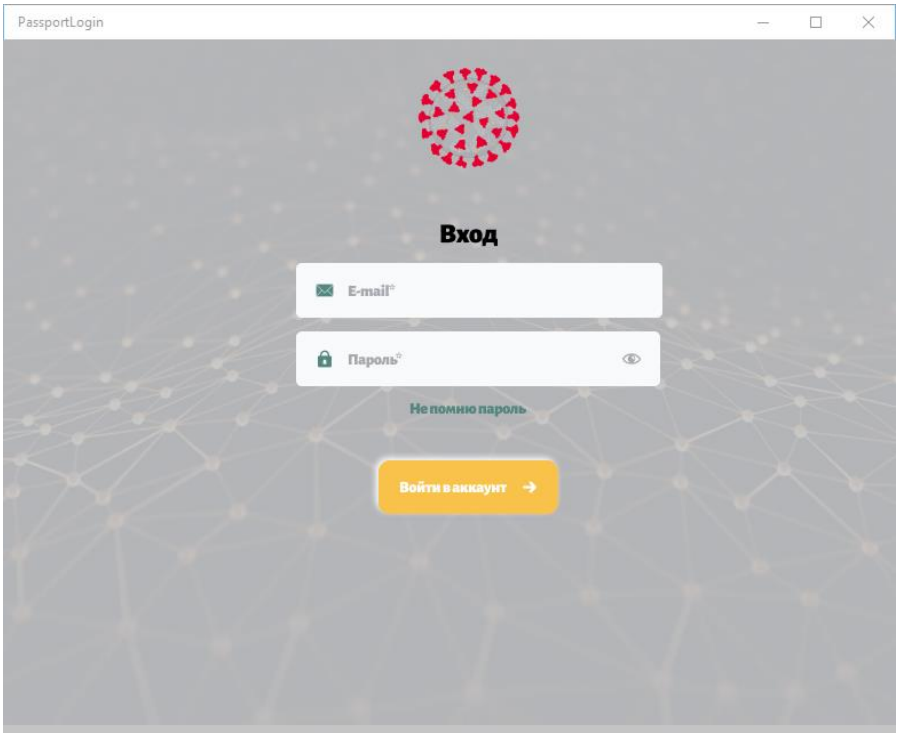


Рисунок 2.14 – Сторінка входу для лікаря

Наступна сторінка для реєстрації нового пацієнта міститиме такі поля як ім’я, прізвище, e-mail , номер телефону. Сторінка додавання пацієнта зображена на рисунку 2.16.

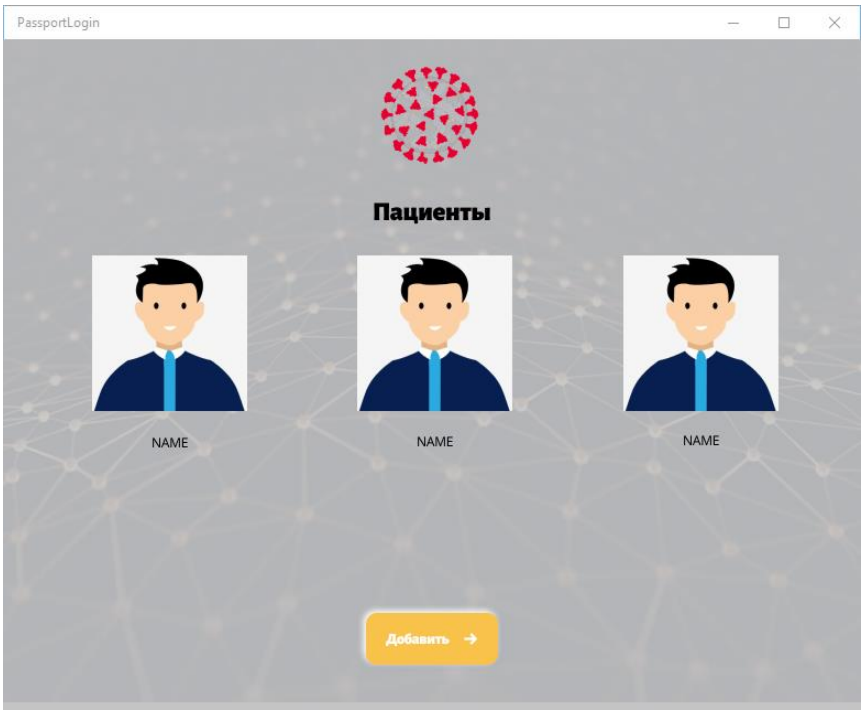


Рисунок 2.15 – Пацієнти лікаря

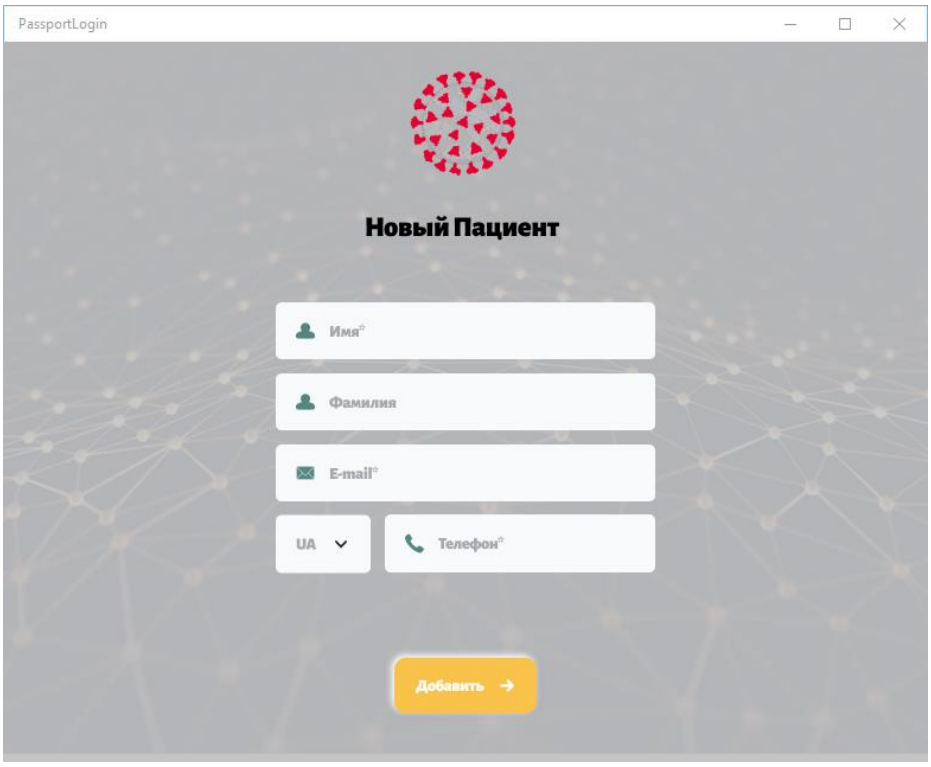


Рисунок 2.16 – Додавання пацієнта

На наступній сторінці лікар може додати фото для аналізу чи запустити автоматичний процес. Ця сторінка зображена на рисунку 2.17.

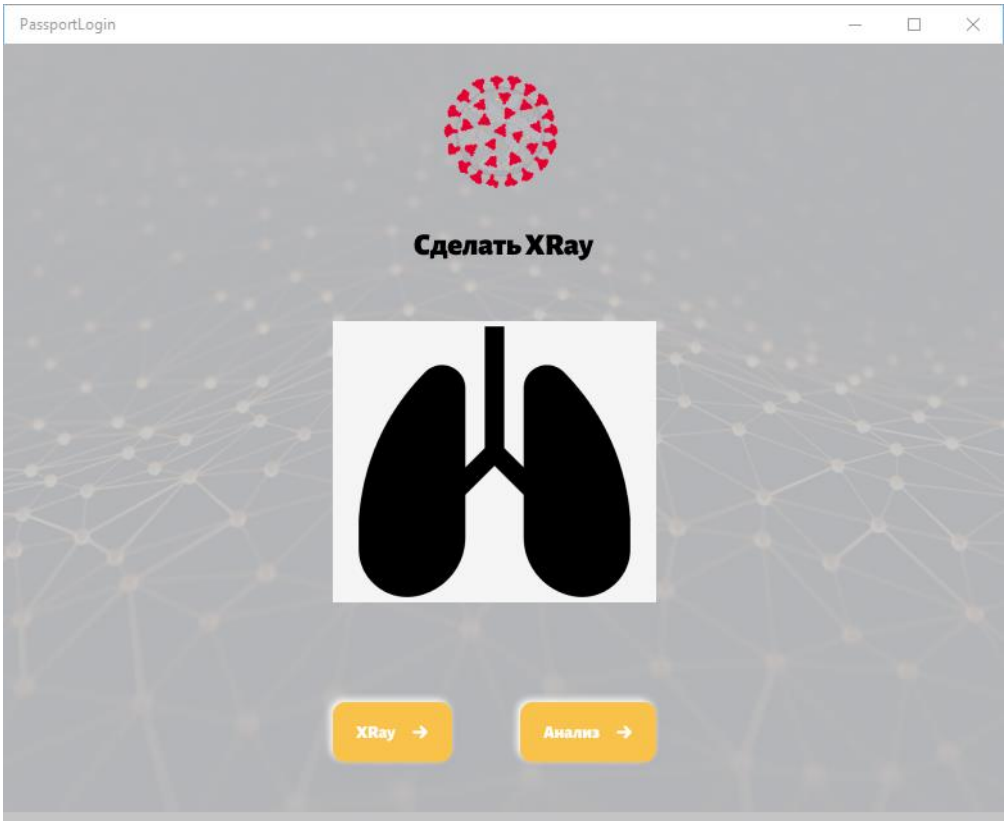


Рисунок 2.17 – Сторінка додатку для лікаря

На наступній сторінці лікарю доступна інформація про аналіз легеневого захворювання. Дана сторінка зображена на рисунку 2.18.

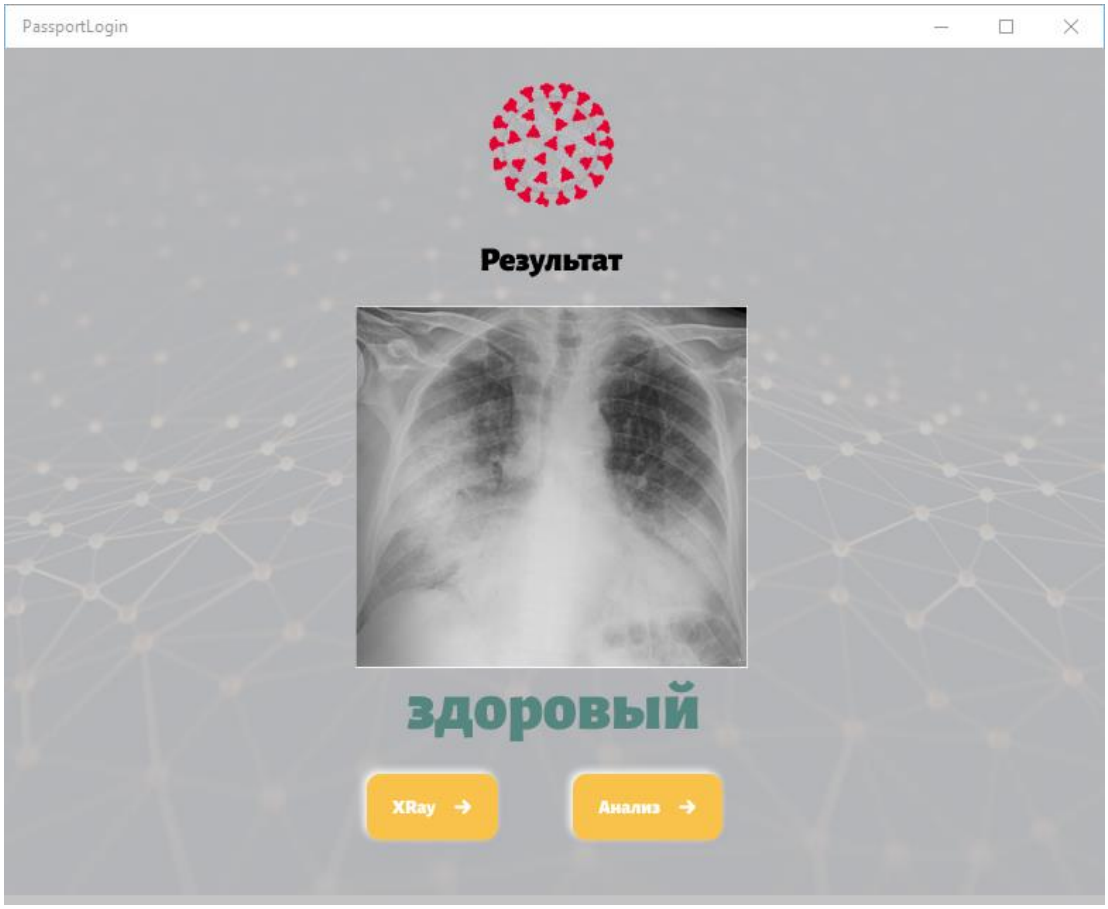


Рисунок 2.18 – Сторінка із діагнозом пацієнта

2.7 Висновки до розділу 2

В даному розділі було зроблено загальний опис предметної області задачі, виконано формування основних вимог та функціоналу для системи розпізнавання.

Вирішено, що система складатиметься із front-end (містить різні додатки для лікаря та пацієнта) та back-end частини, що складається із web-додатку, бази даних та нейронної мережі.

Розроблений дизайн інтерфейсу для мобільного та комп’ютерного додатків.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

3.1. Вибір технологій та їх обґрунтування

3.1.1. Вибір платформи для системи

Відповідно до вимог, що були поставлені до системи, потрібно досягти найбільш можливої доступності для широкого спектру користувачів та зрозумілості у використанні системи. Останнім часом спостерігається така тенденція, що кількість користувачів мобільних пристроїв зростає з кожним роком. Проте частка людей, що використовують персональні комп’ютери, також знаходиться на доволі великому рівні. Для розробки продукту необхідно обрати спосіб реалізації комп’ютерної системи, що буде максимально задовільняти потреби користувачів.

Так як простому користувачеві продукту не потрібно мати доступ до функціоналу, що дає можливість перенавчати нейронну мережу, а тільки до функціоналу, що дозволяє розпізнавати легеневі захворювання, тому саму систему для прискорення роботи краще розмістити на сервері, оскільки всі обчислення необхідні для роботи нейронної мережі відбуватимуться на серверних відеокартах.

Інтерфейс для програмного продукту необхідно реалізувати з досягненням максимальної швидкодії у їхній роботі. Тому необхідно реалізовувати його використовуючи засоби, що дають можливість користуватись системою розпізнавання необхідних для користувача мобільних операційних систем, серед них: Android, iOS, для зручності лікарів – Windows, MacOs, а також систему з можливістю надання доступу до API.

Програмний продукт дає можливість розпізнавати легеневі захворювання на основі знімків зроблених рентген апаратом та відображує результати аналізу лікарю та пацієнту. Користувачі взаємодіють з системою розпізнавання легеневих захворювань за допомогою графічного інтерфейсу, що буде розроблений за допомогою фреймворків, що дозволяють проводити крос-

платформенну розробку для операційних систем, які використовує більшість користувачів даної системи.

Проаналізуємо, які операційні системи мобільних приладів є найпопулярнішими. У 2020 році, iOS та Android є найпопулярнішими операційними системами смвртфонів за кількістю користувачів.

Android – це мобільна платформа та операційна система (ОС) для смартфонів, планшетів, телевізорів, побутової техніки та ін. Вона створена комп'ютерним гігантом Google використовуючи за основу Linux. Дана ОС підтримується кооперацією Open Handset Alliance (OHA). Використовується для мобільних та портативних пристроїв із сенсорним дисплеєм. Крім смартфонів та планшетів, Google покращила Android TV для сучасних телевізорів, AndroidAuto для автомобілів та AndroiWear для смарт-годинників, кожен із цим приладів має спеціальний інтерфейс. Програмні продукти під ОС Android реалізовані в нестандартному байт-коді для створеної на регістрах віртуальної машини Dalvik, для них був створений спеціальний формат для встановлювання пакетів із розширенням .APK. У 87% смартфонів, що були продані у третьому кварталі 2016 року, встановлено операційну систему Android. На конференції для розробників у квітні 2019 року Google надала інформацію, що за весь час існування Android було придбано більше 3 млрд Android-пристроїв.

iOS – це мобільна платформа та операційна система (ОС) для смартфонів, планшетів, портативних медіаплеєрів та деяких приладів іншого типу, що конструюються і виробляються відомою компанією Apple [8]. Дана ОС розроблена в 2007 році. Смартфони та портативні медіаплеєри спочатку стали першими продуктами на IOS, наступними стали електронні планшети та приставки до телевізора. У 2014 році розробив автомобільні мультимедійні системи CarPlay. Пристрої на iOS мають перевагу над Android, оскільки дана ОС підтримується тільки на пристроях вироблених компанією Apple. Через це вони більш оптимізовані і показують більшу швидкодію. Для розробки додатків необхідне спеціальне середовище Xcode, що підтримується тільки на MacBook.

В цьому середовищі можна розробити додатки для більшості приладів Apple, після цього їх необхідно опублікувати в мар і опубліковані в App Store – платформі для їхнього поширення та продажу.

3.1.2. Вибір мінімальної версії мобільного додатку

Для операційної системи Android як мінімальну найоптимальніше використати версію 7.0, адже таким чином можна охопити більшу частину ринку Android пристроїв. Ще однією причиною чому доцільно обрати версію 7.0 є те, що більшість технологій, що є популярними на даний момент прцюють на даній версії операційної системи. Якщо ж вибрати нижчу версію ОС, то це може призвести до вагомого збільшення вартості розробки, оскільки старші версії дуже відрізняються від актуальних на даний момент. Підтримка Android 7.0 і новіше дозволить використовувати більшу частину графіки у векторному форматі, а бібліотеки, що відповідають за сумісність програми, написаної для старших приладів, з ненайновішими версіями системи, включати в проект не потрібно. Даний процес скоротить розмір програми на 30-55%. Отже, доцільно використовувати ОС Android – версію 7.0, як мінімальну версію.

Схожа ситуація відбувається з пристроями, що використовують операційну систему iOS. На даний момент остання версія операційної системи яку компанія Apple підтримує – 9, але розробка під дану версію збільшить фінансові затрати та збільшить витрати часу на розробку проекту. Тому було прийнято рішення, що доцільно буде використовувати як мінімальну версію iOS 11. Кількість користувачів iOS 9 менше 1%.

Відмінності між iOS 9 і iOS 11 величезні, і для підтримки iOS 9 трудовитрати збільшилися б на 50% заради менш ніж 1% користувачів. Більш того, потрібно витратити набагато більше часу на виправлення проблемних моментів, яких немає у більш нових версіях.

3.2 Вибір мови програмування

3.2.1 Xamarin

При розгляді питання, як створити iOS або Android програми, багато людей вважають, що рідні мови, Objective-C, Swift, і Java, є єдиним вірним рішенням. Проте, за кілька останніх років, виникла нова ціла екосистема платформ для створення мобільних додатків.

Xamarin унікальний в цьому просторі, пропонуючи одну мову – C #, бібліотеку класів і середовища виконання, яка працює у всіх з трьох з мобільних платформ – iOS, Android і UWP. Приклад роботи Xamarin застосунку на двох ОС зображено на рисунку 3.1.

Кожна з існуючих платформ розробки мобільних додатків має безліч специфічних особливостей і кожна різниться своїми можливостями створювати "нативні" додатки – тобто, з додатками, які компілюються в рідній для кінцевого пристрою машинний код і які плавно взаємодіють з базовою підсистемою Java. Наприклад деякі платформи дозволяють створювати додатки тільки в HTML і JavaScript, а деякі з них дуже низького рівня і дозволяють використовувати тільки C / C ++. Деякі платформи навіть не задіють "нативний" інструментарій управління.

Xamarin є унікальним в тому, що він поєднує в собі всю силу рідних платформ і додає цілий ряд потужних функцій, включаючи:

– цільові моделі до базових SDK – Xamarin містить цільові моделі майже до всієї базової платформі SDK в iOS і Android. Крім того ці моделі, є строго типізованими (strongly-typed), що робить їхнє використання зручним і забезпечують надійну модель компіляції при налагодженні і під час розробки. Це призводить до мінімізації помилок часу виконання і високої якості додатків;

– objective-C, Java, C, і C ++ взаємодія – Xamarin забезпечує можливості для безпосереднього виклику Objective-C, Java, C і C ++ бібліотек, надаючи ефективні засоби для використання широкого спектру коду вже створених сторонніх бібліотек. Це дозволяє скористатися вже існуючими iOS і Android

бібліотеками, написаними на Objective-C, Java або C ++. Крім того, Xamarin пропонує цільові проекти, які дозволяють легко зв'язати нативні Objective-C і Java бібліотеки за допомогою декларативного синтаксису;

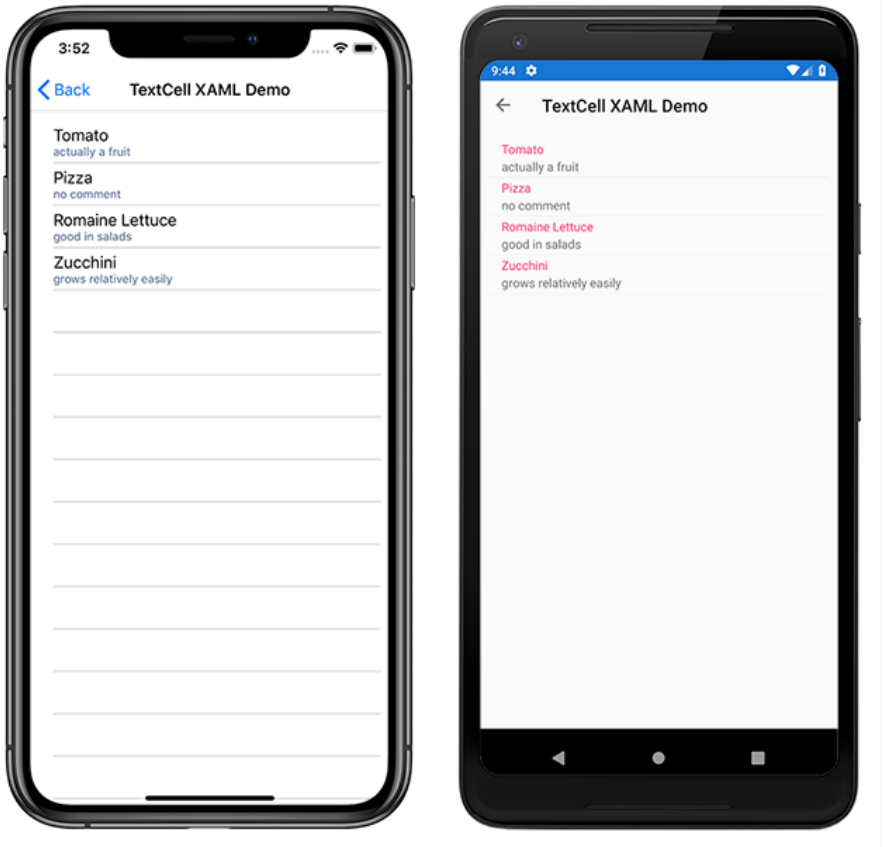


Рисунок 3.1 – Приклад Xamarin додатку[27]

– сучасна конструкція мови – додатки Xamarin написані на C #. Це сучасна мова, яка включає в себе значні поліпшення в порівнянні з Objective-C і Java, такі як динамічні мовні особливості, функціональні конструкти, такі як Лямбда-вирази, LINQ, функції паралельного програмування, універсальні шаблони, і багато іншого;

– базова бібліотека класів (BCL) – додатки Xamarin використовують .NET BCL, велику колекцію класів, які містят оптимізовані функції, такі як XML, робота з базами даних, серіализація, функції введення-виведення, роботи з рядками, підтримка мережі, та інші. Крім того, існуючий код може бути скомпільований для використання в додатках, що забезпечує доступ до тисяч бібліотек, які дозволять робити речі, які ще не охоплені в BCL;

– сучасне інтегроване середовище розробки (IDE) – Xamarin пропонує використовувати Xamarin Studio на Mac OS X, а також Xamarin Studio або Visual Studio на Windows. Це обидва сучасних IDE які включають в себе такі функції, як автоматичне завершення коду, вдосконалена системи управління проектом і рішенням, бібліотека шаблонів комплексного проекту, вбудовану систему керування версіями і багато інших;

– підтримка мобільної крос-платформної розробки – Xamarin пропонує вдосконалену підтримку крос-платформної розробки для трьох основних мобільних платформ iOS, Android і Windows Phone;

– створювані додатки можуть використовувати до 90% загального коду, а бібліотека Xamarin.Mobile пропонує єдиний API для доступу до загальних ресурсів на всіх трьох платформах. Це може значно знизити витрати на розробку і час виходу на ринок для розробників мобільних додатків, націлених на три найбільш популярних мобільних платформи.

Через потужний і комплексний набір функцій Xamarin чудово підійде для розробників додатків, які хочуть використовувати сучасну мову і сучасну платформу для розробки крос-платформних мобільних додатків.

Роботу Xamarin графічно представлена на рисунку 3.2.

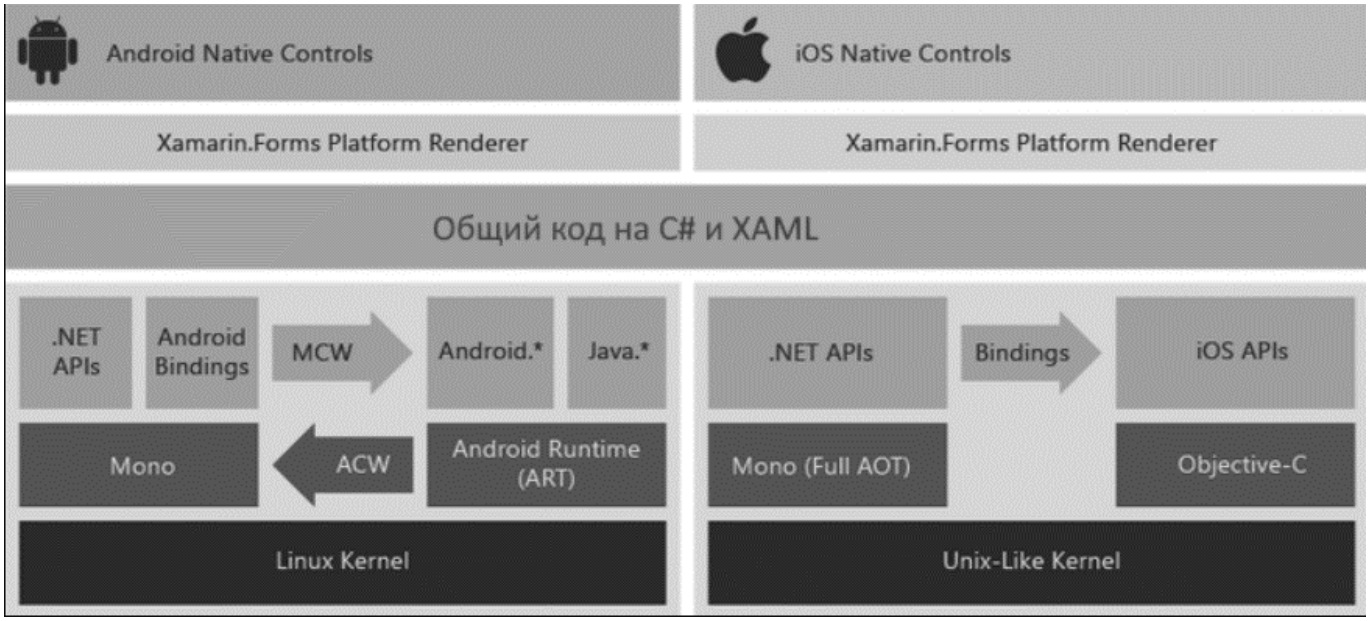


Рисунок 3.2 – Робота Xamarin [27]

Xamarin пропонує два комерційних продукти: Xamarin.iOS і Xamarin.Android. Вони обидва побудовані поверх платформи Mono, версії .NET Framework з відкритим вихідним кодом на основі опублікованих стандартів .NET ECMA. Моно повсюдно відома майже так само довго, як .NET Framework і працює на практично всіх платформах, включаючи Linux, Unix, FreeBSD і Mac OS X.

Для iOS Xamarin – середовище компілює додаток Xamarin.iOS в Ahead-of-Time (AOT), безпосередньо в машинний ARM асемблер-код. Для Android Xamarin – середовище компілює додаток аж до Intermediate Language - IL, який потім компілюється в власну Just-in-Time (JIT) збірку під час запуску програми.

В обох випадках Xamarin додатки використовують середовище виконання, яке автоматично регулює такі речі, як виділення пам'яті, прибирання сміття, взаємодія з базової платформи і т.д

Додатки Xamarin побудовані на підмножині .NET BCL відомій як Xamarin Mobile Profile. Цей профіль був створений спеціально для мобільних додатків, і упакований в MonoTouch.dll і Mono.Android.dll (для iOS і Android відповідно).

Профіль додатків включає методи, дуже схожих на методи Silverlight (і Moonlight), побудованих паралельно Silverlight / Moonlight .NET Profile. Насправді, Xamarin Mobile Profile еквівалентний Silverlight Profile 4.0, із включеними BCL класами.

На додаток до BCL, ці .dll включають бібліотеки оболонок для майже всього iOS SDK і Android SDK, які дозволяють посилатися на базові інтерфейси API SDK безпосередньо з C #.

При остаточному складанні програми, результатом є пакет програми, або файл .app в iOS, або файл .apk в Android. Ці файли нічим не відрізняються від пакетів прикладних програм, побудованих в IDE на платформі нативних бібліотек платформ та здатних до розгортання аналогічним чином.

Створення кроссплатформенних додатків Xamarin.Forms в VS призначений шаблон , який називається Мобільний додаток (XF). Для швидкого пошуку потрібного можна відфільтрувати шаблони за словом Xamarin.Forms

За замовчуванням в створеному програмному продукті вже є згенерований код, що необхідний для роботи. Саме виконання програми починається з файлів App, що зображено на рисунку 3.2.

```
namespace HelloApp
{
    public partial class App : Application
    {
        public App()
        {
            InitializeComponent();

            MainPage = new MainPage();
        }

        protected override void OnStart()
        {

        }

        protected override void OnSleep()
        {

        }
```

Рисунок 3.2 – Клас App

Фактично єдине, що він робить, це встановлює головну сторінку додатка через властивість MainPage в конструкторі.

В якості головної сторінки встановлюється об'єкт класу MainPage, тобто єдина в проекті сторінка.

Перейдемо до визначення цієї сторінки. MainPag епредставляє візуальний інтерфейс сторінки у вигляді коду XAML (як і в Windows Presentation Foundation), який аналогічний Hypertext Markup Language, що зображено на рисунку 3.3.

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="HelloApp.MainPage">
    <StackLayout>
        <Label Text="Welcome to Xamarin.Forms!"
           HorizontalOptions="Center"
           VerticalOptions="CenterAndExpand" />
    </StackLayout>
</ContentPage>
```

Рисунок 3.3 – Xaml код

При створенні сторінок використовується клас ContentPage, в якості кореневого елемента в цьому файлі визначено саме елемент ContentPage.

Весь його вміст складається з одного елемента Label. Клас Label служить для виведення простого тексту на екран. Текст задається за допомогою атрибута Text. Попутно у мітки встановлюються деякі властивості позиціонування через атрибути VerticalOptions і HorizontalOptions.

Таким чином, інтерфейс сторінки досить простий – мітка з текстом. Але також і файл з кодом логіки сторінки – файл MainPage.xaml.cs, що зображено на рисунку 3.4.

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Xamarin.Forms;

namespace HelloApp
{
    [DesignTimeVisible(false)]
    public partial class MainPage : ContentPage
    {
        public MainPage()
        {
            InitializeComponent();
        }
    }
}
```

Рисунок 3.4 – MainPage.xaml.cs

Через виклик InitializeComponent () на сторінці формується інтерфейс, який визначений в файлі MainPage.xaml.

Варто відзначити, що щоб зв'язати обидва визначення сторінки, клас MainPage визначається як частковий (partial), а в файлі MainPage.xaml у визначенні сторінки ContentPage вказаний клас, назва якого збігається з класом MainPage.xaml.cs.

```
x:Class="HelloApp.MainPage"
```

До класу застосовується атрибут DesignTimeVisible, який дозволяє в процесі розробки в Visual Studio в режимі превью побачити створений інтерфейс. Головний проект з усіма розглянутими вище файлами компілюється в бібліотеку dll, а решта три проекти містять посилання на нього. Так, якщо відкрити вузол References у кожного проекту, то побачимо там посилання на бібліотеку HelloApp - тобто посилання на головний проект.

Крім цієї бібліотеки проекти містять ще ряд важних посилань, зокрема кожен проект містить посилання на свою специфічну бібліотеку:

– XFP.Android
– XFP.iOS
– UWP

Дані бібліотеки визначають для своєї мобільної платформи статичний метод XF.Init (), який виконує ініціалізацію системи XF.

Виклик додатку на Android

Наприклад, візьмемо проект для Android. У ньому за умовчанням є файл MainActivity.cs, з якого власне починається виконання проекту на Android:

```
public class MainActivity : global::Xamarin.Forms.Platform.Android.FormsAppCompatActivity
{
    protected override void OnCreate(Bundle savedInstanceState)
    {
        TabLayoutResource = Resource.Layout.Tabbar;
        ToolbarResource = Resource.Layout.Toolbar;

        base.OnCreate(savedInstanceState);

        Xamarin.Essentials.Platform.Init(this, savedInstanceState);
        global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
        LoadApplication(new App());
    }
```

Рисунок 3.5 – MainActivity.cs

Перш за все звернемо увагу на дві обов'язкові рядки:

```
Xamarin.Essentials.Platform.Init(this, savedInstanceState);
global::Xamarin.Forms.Forms.Init(this, savedInstanceState);
```

Тут відбуваєтьсяться ініціалізація XE і XF.

Потім екземпляр класу App передається в метод LoadApplication (), який визначений у класу XFPA.FosAppCompatAcivity, і власне відбувається запуск програми Xamarin Forms.

```
1  LoadApplication(new App());
```

Виклик програми на iOS.

У проекті для iOS початок роботи застосунку XF відбуватиметься в класі AppDelegate в його методі FinishedLaunching.

```
1  public partial class AppDelegate : global::Xamarin.Forms.Platform.iOS.FormsApplicationDelegate
2  {
3      public override bool FinishedLaunching(UIApplication app, NSDictionary options)
4      {
5          global::Xamarin.Forms.Forms.Init();
6          LoadApplication(new App());
7
8          return base.FinishedLaunching(app, options);
9      }
10 }
```

Рисунок 3.6 – FinishedLaunching

Виклик global :: XF.Init () ініціалізує XF, а виклик LoadApplication (new App ()) запускає додаток XF на виконання.

UWP (Universal Windows Platform)

В проекті для UWP всі подібні дії розбиті на два етапи. Перш за все, в файлі App.xaml.cs в методі OnLaunched() визначена ініціалізація XF. А в файлі MainPage.xaml.cs запуск програми.

```
1  protected override void OnLaunched(LaunchActivatedEventArgs e)
2  {
3      Frame rootFrame = Window.Current.Content as Frame;
4
5      if (rootFrame == null)
6      {
7          rootFrame = new Frame();
8
9          rootFrame.NavigationFailed += OnNavigationFailed;
0
1          Xamarin.Forms.Forms.Init(e);
```

3.2.2 Побудова модуля розпізнавання

У відкритих джерелах знайдено датасет “COVID-19 Radiography Database”, що містить рентгенівські знімки грудної клітки. Зображення розділялись на 3 класи: “Здоровий”, “Вірусна пневмонія”, “Covid-19”. На основі цих даних буде реалізований програмний продукт.

Першим кроком в розробці програмного продукту є імпортування необхідних бібліотек, що зображено на рисунку 3.7.

```
#import neccessery libs
%matplotlib inline
import random
import torch
import torchvision
import os
from matplotlib import pyplot as plt

import shutil
import pandas as pd

import numpy as np

from PIL import Image

torch.manual_seed(0)
from torch.autograd import Variable
```

Рисунок 3.7 – Імпорт бібліотек

На наступному кроці необхідно розділити дані на навчальні та тестові, що зображено на рисунку 3.8.

Далі після підготовки папки із зображеннями є створення власного класу набору даних,. Цей клас успадковує абстрактний клас Dataset і містить два важливі методи, як показано на рисунку 3.9:

- len - повертає розмір набору даних;
- getitem може бути використаний для отримання і-го зразка.

```
#split data to training and test
 classification_types = ['health', 'virus_pneu', 'covid']
 root_dir = 'Database'
 source_dirs = ['Health', 'Virus_pneu', 'COVID-19']

 if os.path.isdir(os.path.join(root_dir, source_dirs[1])):
     os.mkdir(os.path.join(root_dir, 'test'))

     for i, d in enumerate(source_dirs):
         os.rename(os.path.join(root_dir, d), os.path.join(root_dir, classification_types[i]))

     for c in classification_types:
         os.mkdir(os.path.join(root_dir, 'test', c))

     for c in classification_types:
         images = [x for x in os.listdir(os.path.join(root_dir, c)) if x.lower().endswith('png')]
         selected_images = random.sample(images, 30)
         for image in selected_images:
             source_path = os.path.join(root_dir, c, image)
             target_path = os.path.join(root_dir, 'test', c, image)
             shutil.move(source_path, target_path)
```

Рисунок 3.8 – Розділення даних

```
# custom dataset

class XRayData(torch.utils.data.Dataset):
    def __init__(self, data_location, remake):
        def img(classification_type):
            imgs = [x for x in os.listdir(data_location[classification_type]) if x[-3:].lower().endswith('png')]
            return imgs

        self.images = {}
        self.classification_types = ['health', 'virus_pneu', 'covid']

        for classification_type in self.classification_types:
            self.imgs[classification_type] = img(classification_type)

        self.data_location = data_location
        self.remake = remake


    def __len__(self):
        return sum([len(self.images[classification_type]) for classification_type in self.classification_types])


    def __getitem__(self, index):
        classification_type = random.choice(self.classification_types)
        index = index % len(self.images[classification_type])
        image_name = self.images[classification_type][index]
        image_path = os.path.join(self.data_location[classification_type], image_name)
        image = Image.open(image_path).convert('RGB')
        return self.remake(image), self.classification_types.index(classification_type)
```

Рисунок 3.9 – class XRayData

Ще однією проблемою було те, що зразки були не одного розміру, а нейронна мережа очікує зображення фіксованого розміру. Тому ми виконає ми

попередню обробку навчального та тестового наборів даних, як показано на рисунку 3.10.

```
# transform image (preprocessing)
train_transform = torchvision.transforms.Compose([
    torchvision.transforms.Resize(size = (224,224)),
    torchvision.transforms.RandomHorizontalFlip(),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize(mean =[0.485,0.456,0.406],std=[0.229,0.224,0.225]),
])
test_transform = torchvision.transforms.Compose([
    torchvision.transforms.Resize(size = (224,224)),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize(mean =[0.485,0.456,0.406],std=[0.229,0.224,0.225]),
])
```

Рисунок 3.10 – Обробка наборів даних

Далі необхідно створити словники, де ключем буде назва класу зображення, а значенням – відповідна папка, де зберігаються зображення класу. Після цього створюється 2 об’єкти класу, передаючи тестові і навчальні словники з відповідними перетвореннями, як параметри класу, як показано на рисунку 3.11.

```
#get DATA
train_dirs = {
    'health': 'Database/health',
    'virus_pneu': 'Database/virus_pneu',
    'covid': 'Database/covid'
}

train_dataset = XRayData(train_dirs,train_transform)

test_dirs = {
    'health': 'Database/test/health',
    'virus_pneu': 'Database/test/virus_pneu',
    'covid': 'Database/test/covid'
}

test_dataset = XRayData(test_dirs,test_transform)

batch_size = 6

dl_train = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
dl_test = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size, shuffle=True)

print('Number of training batches', len(dl_train))
print('Number of test batches', len(dl_test))
```

Рисунок 3.11 – Створення словників

Далі виконується візуалізація. Створюємо функцію, яка братиме правильні надписи, передбачені надписи та набір зображень. Якщо передбачення правильне, то надпис буде зелений, якщо ні, то червоний. Оскільки модель ще не навчена, то поки будемо передавати правильні надписи, що зображено на рисунку 3.12.

```
# show date using matplotlib
classification_types = train_dataset.classification_types


def show_images(images, labels, preds):
    plt.figure(figsize=(8, 4))
    for i, image in enumerate(images):
        plt.subplot(1, 6, i + 1, xticks=[], yticks=[])
        image = image.numpy().transpose((1, 2, 0))
        mean = np.array([0.485, 0.456, 0.406])
        std = np.array([0.229, 0.224, 0.225])
        image = image * std + mean
        image = np.clip(image, 0., 1.)
        plt.imshow(image)
        col = 'green'
        if preds[i] != labels[i]:
#             print("label",labels[i])
            col = 'red'

        plt.xlabel(f'{classification_types[int(labels[i].numpy())]}')
        plt.ylabel(f'{classification_types[int(preds[i].numpy())]}', color=col)
    plt.tight_layout()
    plt.show()
images,labels = next(iter(dl_train))
show_images(images,labels,labels)
```

Рисунок 3.12 – Візуалізація

Результат показаний на рисунку 3.13

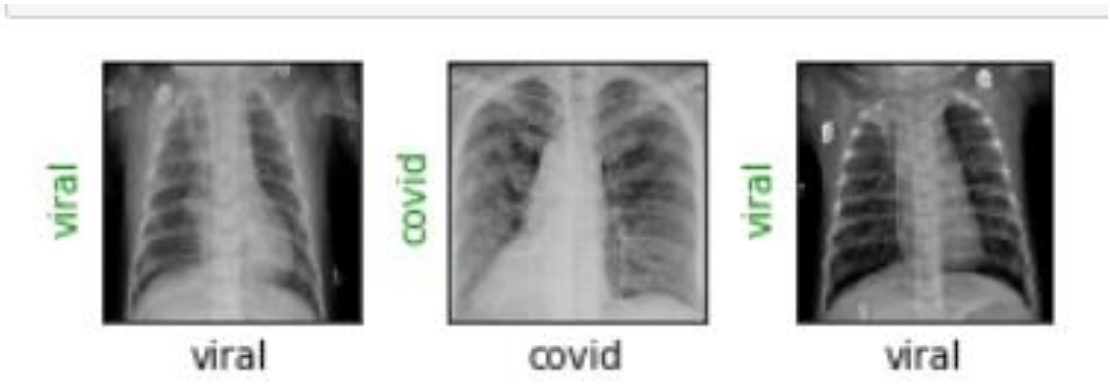


Рисунок 3.13 –Приклад відображення

Для нашого завдання було вирішено використати переднавчену модель resnet18 і потім використовуючи трансферне навчання вирішувати проблеми класифікації, як показано на рисунку 3.14.

```
#creating model
resnet18 = torchvision.models.resnet18(pretrained=True)
print(resnet18)
resnet18.fc = torch.nn.Linear(in_features=512, out_features=3)
loss_fn = torch.nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(resnet18.parameters(), lr=4e-5)
def show_preds():
    resnet18.eval()
    images, labels = next(iter(dl_test))
    outputs = resnet18(images)
    _, preds = torch.max(outputs, 1)
    show_images(images, labels, preds)


loader = torchvision.transforms.Compose([
    torchvision.transforms.Resize(size = (224,224)),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize(mean =[0.485,0.456,0.406],std=[0.229,0.224,0.225]),
])
def image_loader(image_name):
    image = Image.open(image_name).convert('RGB')
    image = loader(image).float()
    image = Variable(image, requires_grad=True)
    image = image.unsqueeze(0)
    return image
```

Рисунок 3.14 – Створення моделі

Потім ми створюємо ще одну функцію для заключної частини навчання моделі. Ми перебираємо дані, що передаємо в мережу і оптимізуємо до необхідної точності, що зображено на рисунку 3.15.

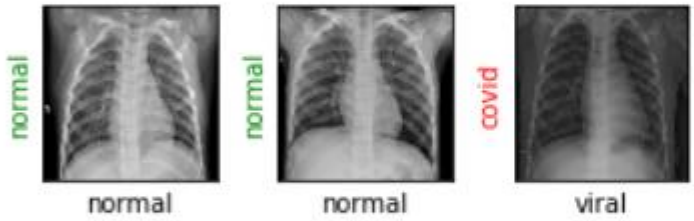
```
#train our model
def train(epochs):
    print('Starting training..')
    for e in range(0, epochs):
        print('='*20)
        print(f'Starting epoch {e + 1}/{epochs}')
        print('='*20)

        train_loss = 0.
        val_loss = 0.

        resnet18.train()

        for train_step, (images, labels) in enumerate(dl_train):
            optimizer.zero_grad()
            outputs = resnet18(images)
            loss = loss_fn(outputs, labels)
            loss.backward()
            optimizer.step()
            train_loss += loss.item()
            if train_step % 20 == 0:
                accuracy = 0
                resnet18.eval()

                for val_step, (images, labels) in enumerate(dl_test):
#                     print(type(images),images)
                    outputs = resnet18(images)
                    loss = loss_fn(outputs, labels)
                    val_loss += loss.item()

                    _, preds = torch.max(outputs, 1)
                    accuracy += sum((preds == labels).numpy())

                val_loss /= (val_step + 1)
                accuracy = accuracy/len(test_dataset)
                show_preds()

                resnet18.train()

                if accuracy >= 0.95:
                    break

        train_loss /= (train_step + 1)


with experiment.train():
    train(epochs=1)
```

Рисунок 3.14 – Навчання моделі

На рисунку 3.15 зображено процес навчання нейронної мережі.

```
Starting training..
====================
Starting epoch 1/1
====================
Evaluating at step 0
Validation Loss: 0.9623, Accuracy: 0.6333
```


```
Evaluating at step 20
Validation Loss: 0.7243, Accuracy: 0.7444
```




```
Evaluating at step 40
Validation Loss: 0.5893, Accuracy: 0.8333
```


```
Evaluating at step 60
Validation Loss: 0.2687, Accuracy: 0.9556
```

Рисунок 3.15 – Відображення навчання моделі

Зберігаємо модель, як показано на рисунку 3.16.

```
# save model
torch.save(resnet18,'./model.pth')
```

Рисунок 3.16 – Зберігання моделі

Далі проводиться тестування моделі при завантаженні одного зображення. Для цього потрібно спочатку перетворити вхідні дані так само, як перетворювались тестові зображення, що показано на рисунку 3.17.

```
#test our model
loader = torchvision.transforms.Compose([
    torchvision.transforms.Resize(size = (224,224)),
    torchvision.transforms.ToTensor(),
    torchvision.transforms.Normalize(mean =[0.485,0.456,0.406],std=[0.229,0.224,0.225]),
])
def image_loader(image_name):
    image = Image.open(image_name).convert('RGB')
    image = loader(image).float()
    image = Variable(image, requires_grad=True)
    image = image.unsqueeze(0)
    return image

image = image_loader('health.png')

res_dict = {0:"Health",1:"Virus_pneu",2:"Covid"}
output = model(image)
_, preds = torch.max(output, 1)
print(res_dict[preds.tolist()[0]])
```

Рисунок 3.17 – Тестування моделі

Результат роботи модуля зображено на рисунку 3.18.

```
# print(output)
_, preds = torch.max(output, 1)
# print(preds)
print(res_dict[preds.tolist()[0]])

Covid
```

Рисунок 3.18 – Результат роботи нейронної мережі

3.2.3 Web Застосунок та база даних

Налаштування MongoDB.

Якшо використовується Window, *MongoDB* за замовчуванням встановлюється в *C: \ Program Files \ MongoDB* . Необхідно додати *C: \ Program Files \ MongoDB*

\ *Server* \ <*номер_версії*> \ *b* до системних змінних. Ця зміна забезпечує доступ до MongoDB з будь-якого місця.

Використовується оболонка mongo у наступних кроках для створення бази даних, створення колекцій та зберігання документів:

1) необхідно обрати каталог для зберігання даних. Наприклад, C: \ BooksData *у* Windows. Потрібно створити каталог, якщо він не існує. Монго Shell не створює нових каталогів;
2) відкрити командний рядок. Далі необхідно підключитися до MongoDB, що за замовчуванням створюється на порті 27017. Потрібно замінити <data_directory_path> на каталог, який обраний на попередньому кроці.

Створення проекту веб-API ASP.NET Core:

1) у середовищі перейти до Файл > Створити > Проект;
2) вибрати тип проекту веб-програми ASP.NET Core;
3) обрати назву проекту;
4) вибрати цільову структуру .NET Core та ASP.NET Core3.0;
5) перейти до галереї NuGet: MongoDB.Driver, щоб визначити останню стабільну версію драйвера .NET для MongoDB. У вікні консолі диспетчера пакетів перейти до кореня проекту. Виконати команду, що зображена на рисунку 3.19, щоб встановити драйвер .NET для MongoDB.

```
Install-Package MongoDB.Driver -Version {VERSION}
```

Рисунок 3.19 – Встановлення MongoDB.Driver

3.2.4 Розгортання проекту на Amazon Server

.NET проект розгортається на AWS Elastic Beanstalk за допомогою набору інструментів AWS для Visual Studio.

Необхідно зробити такі кроки, щоб створити середовище:

1) спочатку скористатись майстром створення нового додатка на консолі Elastic Beanstalk для створення середовища програми. Для платформи вибрати .NET;
2) відкрити консоль Elastic Beanstalk за цим попередньо налаштованим посиланням;
   console.aws.amazon.com/elasticbeanstalk/home#/newApplication?applicationName=tutorials&environmentType=LoadBalanced
3) вибрати платформу та гілку платформи, які відповідають мові, що використовується додатком;
4) підтвердити створення.

Налаштування середовища VS для публікації в AWS

Спочатку необхідно відкрити контекстне меню, потім вибрати опублікувати в AWS, як показано на рисунку 3.20.

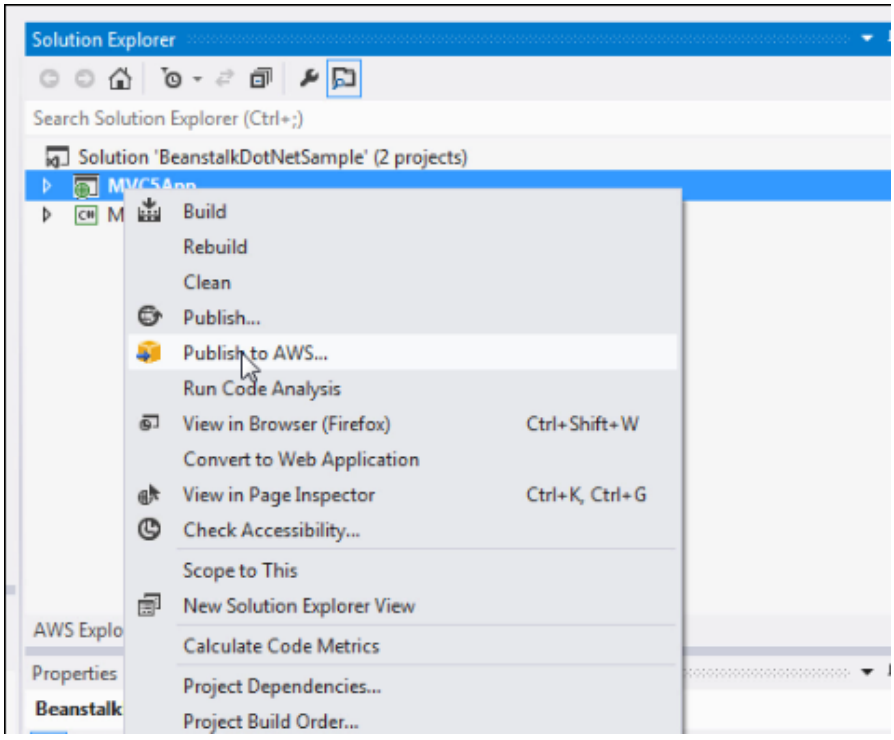


Рисунок 3.20 – Публікація в AWS

На сторінці Опублікувати в AWS Elastic Beanstalk для цілі розгортання необхідно вибрати щойностворене середовище, а потім натиснути Далі, що зображено на рисунку 3.21.

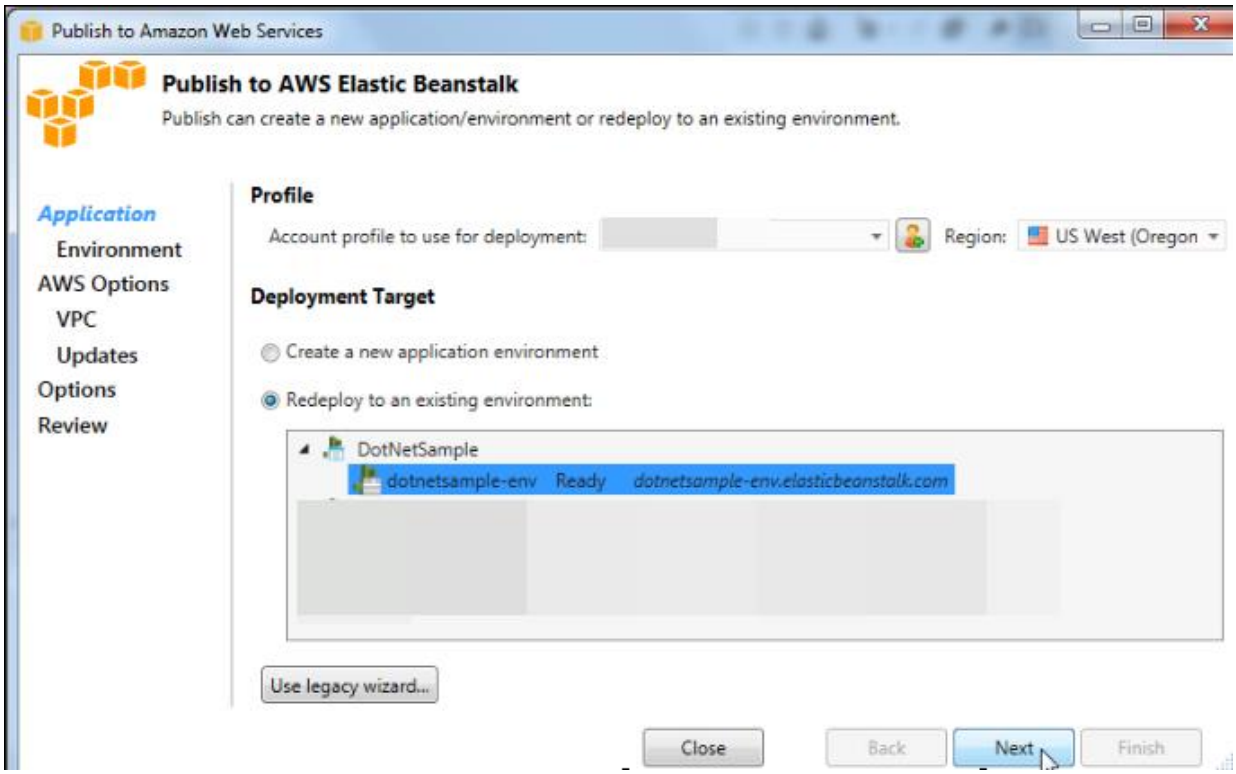


Рисунок 3.21 – Вибір середовища

На сторінці Параметри програми необхідно вибрати усі значення за замовчуванням, як показано на рисунку 3.22.

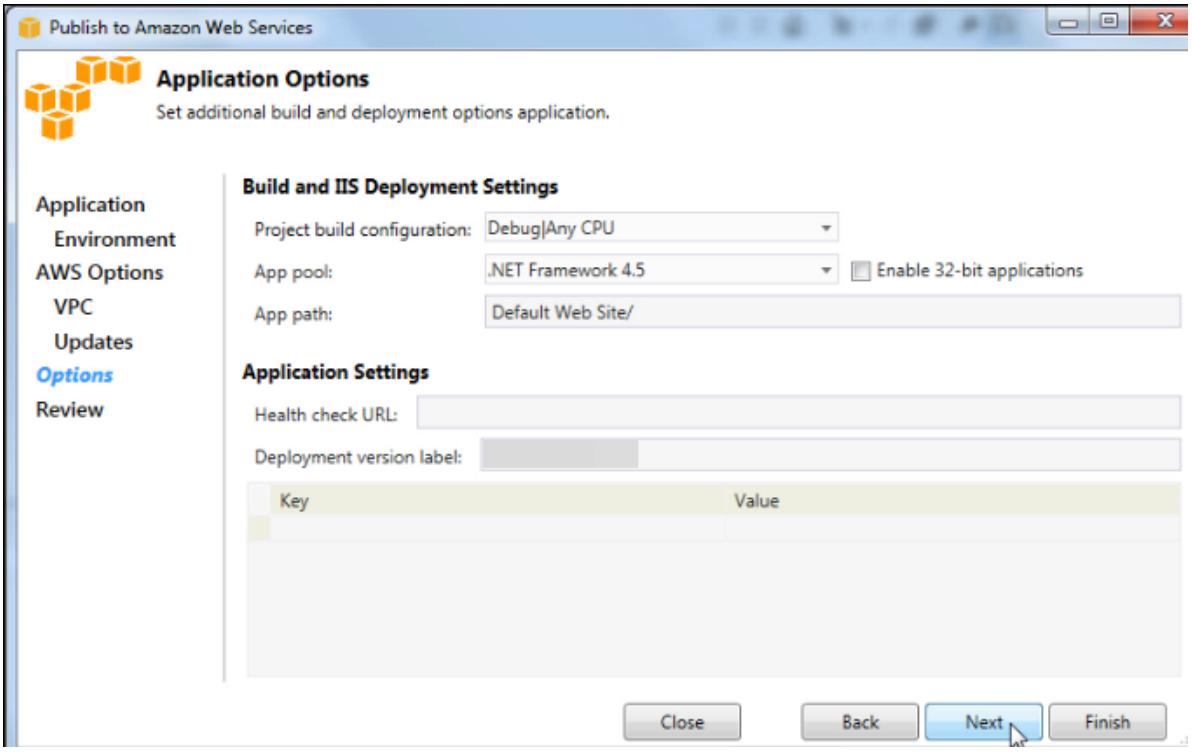


Рисунок 3.22 – Параметри програми

На сторінці огляду потрібно обрати Deploy, що зображено на рисунку 3.23.

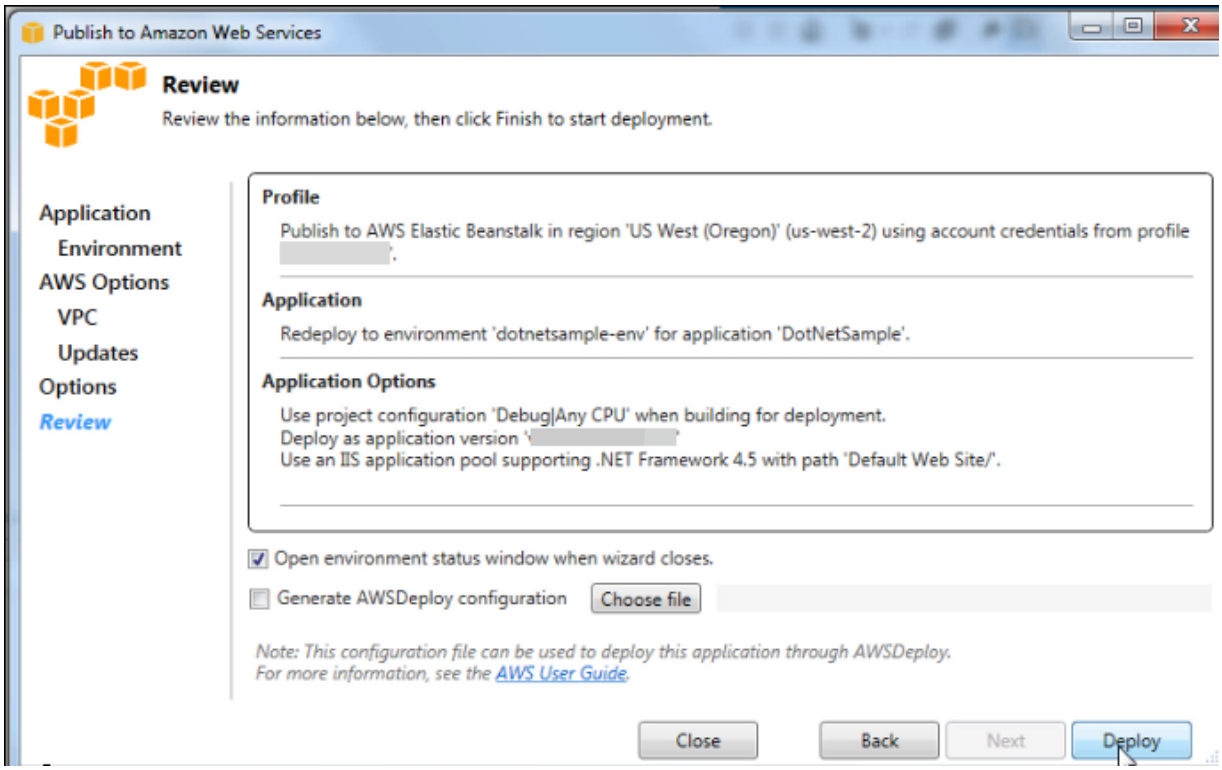


Рисунок 3.23 – Deploy

Якщо є необхідність відстежувати стан розгортання, то можна використовувати NuGet Package Manager у Microsoft Visual Studio, як показано на рисунку 3.24.

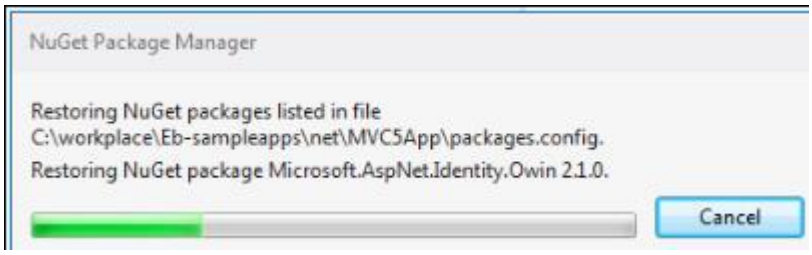


Рисунок 3.24 – Стан розгортання

Після успішного розгортання програми вікно Вивід відображається успішно завершеним. Далі необхідно повернутись до консолі Elastic Beanstalk. На панелі навігації обрати Перейти до середовища. В результаті програма

ASP.NET відкриється в новій вкладці. Дані процеси зображені на рисунках 3.25 -3.26.

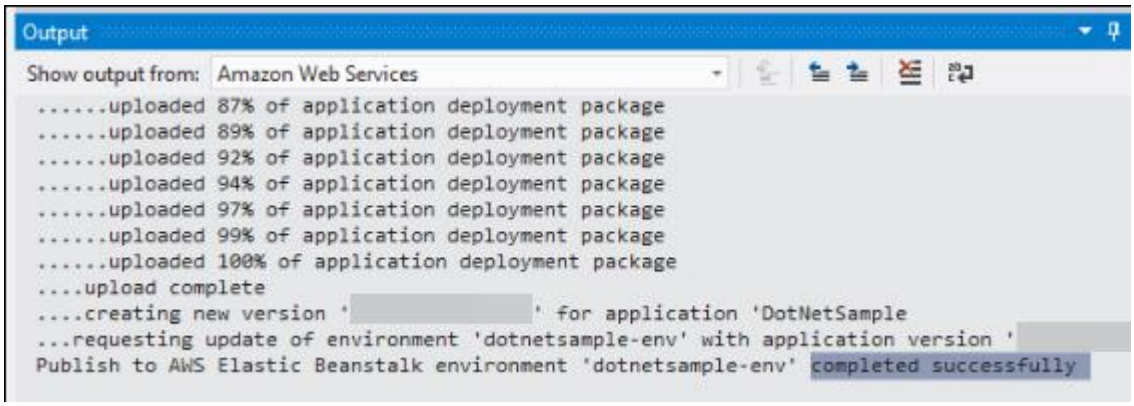


Рисунок 3.25 – Завершення встановлення

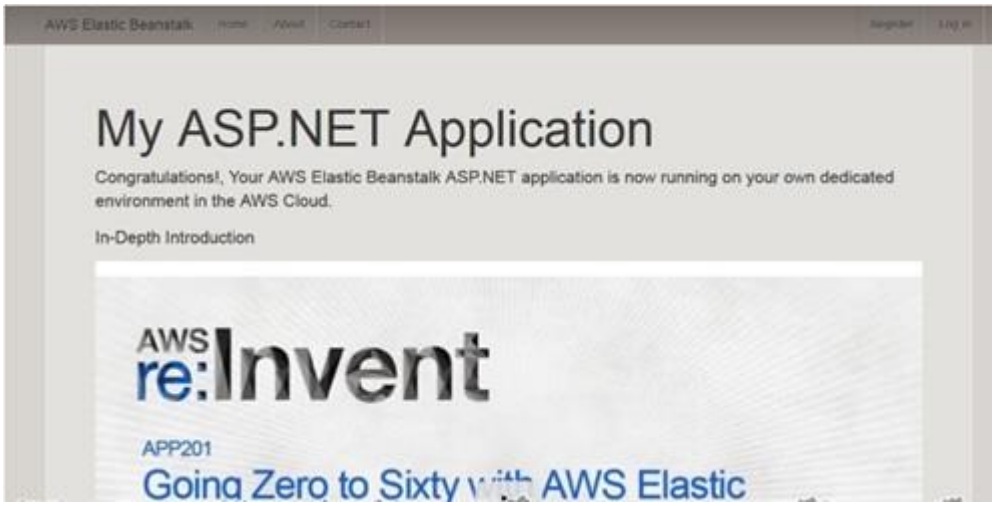


Рисунок 3.26 – Результат виконання публікації додатку на AWS

3.3 Висновки до розділу 3

Розглянуто аспекти вибору стеку технологій та бібліотек для реалізації додатку, наведено обґрунтування вибору платформи для розробки програмного продукту, мови написання додатку та множини пристроїв, які підтримуватимуться. Враховуючи вимоги до програмного продукту вибрано мову програмування та платформи, обрано допоміжні бібліотеки, які спростять процес розробки та дозволять побудувати гнучку архітектуру. Також розглянуті

можливі варіанти реалізації нейронної мережі для класифікації об’єктів на зображенні.

Проведений опис основних рішень та підходів щодо реалізації проекту. Розроблено гнучку структуру управління станами програмного продукту, яка проектувалась із розрахунком на можливе розширення, підхід для позиціювання ефектів на зображенні та графічний інтерфейс користувача.

4 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

У даному розділі буде проведено аналіз та розробку стартап-проекту інтелектуальна система діагностування легеневих захворювань людини за рентгенівськими знімками.

4.1. Опис ідеї проекту

Почнемо з опису можливих варіацій застосування продукту стартапу. Деталі наведено в таблиці 4.1.

Таблиця 4.1 – Опис ідеї стартап проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Створення системи діагностування легеневих захворювань людини за рентгенівськими знімками	Системи, що використовуються у медичних закладах	Ефективніше використання показників пристроїв. Зменшення людської помилки внаслідок використання нейронних мереж.
	Системи, що використовуються у мобільних лабораторіях	Не змінюючи розмірів мобільної системи, збільшується її ефективність. Можливість керувати системою оператору, адже попередня постановка діагнозу не вимагає обробки результатів спеціалістом

Таблиця 4.1 – Продовження

	Система, що дозволяє людям перевіряти власні рентген знімки	Людина має можливість перевірити власні знімки, якщо має сумніви отриманих результатів від медпрацівників

Далі слід зробити такі пункти:

– аналіз техніко-економічних можливостей;

– аналіз характеристик ідеї та порівняння з попереднім колом конкурентів;

До основних конкурентів можна віднести:

1) Системи, що проводять діагностування за рентгенівськими знімками, що вимагають втручання спеціаліста для діагностування.

Таблиця 4.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

№	Техніко-економічні характеристики ідеї	Мій проект	Конкуренти	W	N	S
			1			
1	Висока швидкість розпізнавання	+	-			+
2	Можливість інтеграції системи як елемент іншої системи	+	+		+	

Таблиця 4.2 – Продовження

3	Необхідність додаткового устаткування у медзакладах	+/-	+	+		
4	Покращена якість діагностування хвороби	+	-			+
5	Відсутність людської похибки	+	-			+

4.2. Технологічний аудит ідеї проекту

Для того, щоб ефективно реалізувати ідею системи діагностування легеневих захворювань людини за рентгенівськими знімками, необхідно провести технологічний аудит. Результати аудиту наведені в таблиці 4.3.

Таблиця 4.3 – Технологічна здійсненність проекту

№	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Система діагностування легеневих захворювань людини за рентгенівськими знімками	Python 3.8.0+	Наявна	У відкритому доступі
2		PostgreSQL database	Наявна	У відкритому доступі
3		JWT	Наявна	У відкритому доступі

Таблиця 4.3 – Продовження

4		matplotlib.pyplot	Наявна	У відкритому доступі
5		pandas	Наявна	У відкритому доступі
6		numpy	Наявна	У відкритому доступі
7		keras	Наявна	У відкритому доступі
8		sklearn.preprocessing	Наявна	У відкритому доступі
9	Мобільний застосунок	Xamarin.Forms	Наявна	У відкритому доступі

Відповідно до результатів технологічного аудиту, всі необхідні технології знаходяться у вільному доступі.

4.3. Аналіз ринкових можливостей запуску стартап-проекту

Наступним кроком стане аналіз ринку. Результат наведено в таблиці 4.4.

Таблиця 4.4 – Попередня характеристика потенційного ринку стартап-проекту

№	Показники стану ринку	Характеристика
1	Кількість головних гравців, од	5-8
2	Загальний обсяг продаж, грн/ум.од	Середня вартість томографа – 150000$. Зарплата спеціаліста, що розшифровує отримані дані 500 – 1000$
3	Динаміка ринку	Швидко зростає через пандемію
4	Наявність обмежень для входу	Якісний продукт. Отримання медичних сертифікатів.
5	Специфічні вимоги до стандартизації та сертифікації	Необхідні спеціальні медичні сертифікати.

Приведемо ще одну таблицю. в ній наведемо потенційні групи клієнтів, їхні потреби та вимоги до продукту.

Таблиця 4.5 – Характеристика потенційних клієнтів стартап-проекту

№	Потреба, що формує ринок	Цільова аудиторія	Відмінності у поведінці різних потенційних цільових груп	Вимоги споживачів до товару

Таблиця 4.5 – Продовження

1	Потреба у ефективному API для розпізнавання	Великий бізнес, використання API в різноманітних проектах	високі вимоги до точності розпізнавання, різний бюджет, який компанія готова витрачати на отримання доступу до API	Гарантія надійності системи, вичерпна документація для використання API
2	Потреба у ефективному кросс-платформенному мобільному додатку для розпізнавання	Пересічні клієнти, які хочуть самі перевірити свої знімки	високі вимоги до точності розпізнавання	Гарантія надійності системи, простота у використанні, нативний візуальний інтерфейс
3	Потреба у простоті інтеграції	Медичні заклади	високі вимоги до точності розпізнавання, робота із різними приладами для рентгенології	Гарантія надійності системи, простота у використанні, нативний візуальний інтерфейс

Після визначення потенційних груп клієнтів проведено аналіз ринкового середовища на фактори загроз (табл. 4.6) та можливостей (табл. 4.7).

Таблиця 4.6 – Фактори загроз

№	Фактор	Зміст загрози	Можлива реакція компанії
1	Людський фактор	Неможливість повністю врахувати людський фактор у процесі отримання даних	Вдосконалення моделі нейронної мережі
2	Консерватизм серед пересічних клієнтів	Деякий відсоток клієнтів може відмовитись переходити на використання нової системи	Реклама та просування продукту в інформаційному полі
3	Висока частота помилок системи	Неправильні налаштування моделі в ході навчання/донавчання	Часті оновлення, рефакторинг методології навчання нейронної мережі

Таблиця 4.7 – Фактори можливостей

№	Фактор	Зміст можливості	Можлива реакція компанії
1	Залучення інвестицій	Можливість залучення інвестицій від інвестиційних фондів та інших компаній	Можливість розширення штату, та найму найкращих спеціалістів з області безпеки
2	Розширення клієнтської бази	З допомогою продуманої PR-кампанії, можна переманити клієнтів	Спланувати рекламну кампанію, проводити презентації, в яких для

		конкурентів запропонувавши більш вигідні умови	демонстрації буде використано розроблюваний продукт
3	Відсутність конкуренції	Відсутність високої конкуренції на ринку розпізнавання людської активності	Збільшення ціни на послуги компанії

Риси конкуренції, притаманні розроблювальній системі, наведені в таблиці 4.8.

Як можна побачити в даній таблиці, конкуренція на міжнародному ринку є не надто високою, однак той факт, що великі гравці рідко надають перевагу старим перевіреним системам більше, ніж новим, ускладнює конкуренцію на цьому ринку.

Таблиця 4.8 – Ступеневий аналіз ринку конкуренції

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Вказати тип конкуренції: чиста	Досить велика кількість незалежних гравців на ринку	Фокусування уваги на точність розпізнавння
2. За рівнем конкурентної боротьби: міжнаціональна	Компанії працюють на міжнародному ринку	Необхідність працювати на світовому ринку

Таблиця 4.8 – Продовження

3. За галузевою ознакою: міжгалузева	Постачальники займаються не лише виробництвом вузько-направленого ПО	Сконцентруватися на суті завдання розпізнавння та його ефективнотості та якості
4. Конкуренція за видами товарів: товарно-видова	Наявність на ринку кількох авторитетних постачальників	Здобувати авторитет влаштовуючи спеціалістів з розробки систем штучного інтелекту
5. За характером конкурентних переваг: цінова	Великі відмінності в ціні від різних постачальників	Забезпечити конкурентну ціну на продукт
6. За інтенсивністю: марочна	Постачальники виступають під певним брендом	Створення впізнаваного бренду

Виокремимо прямих конкурентів, що є на даний момент на ринку розпізнавння людської активності, займають найбільші частки ринку та можуть становити загрозу розвитку стартапу. Серед таких: Google Activity Recognition, Samsung Health, Mi Fit, Huawei Health. Згідно з цим, виконаємо аналіз конкуренції в галузі за М. Портером.

Робота на ринку можлива, однак необхідно надати конкурентоздатну ціну та набір характеристик продукту. Також, система має працювати справно, оскільки замовники напряму ризикують власними коштами.

Таблиця 4.9 – Аналіз конкуренції в галузі за М. Портером

Складові аналізу	Прямі конкуренти в галузі	Потенційні конкуренти в галузі	Постача-льники	Клієнти	Товари-замінники
	Toshiba, Philips, Siemens	-	Siemens	Медичні заклади	Достатня кількість
Висновки	Інтенсивна конкурента боротьба	Вихід на ринок можливий за умови надання конкуренто здатних умов	Постачальники встановлюють високу вартість на ліцензії	Клієнти висувають вимоги до виробників оскільки конкуренція висока	Відсутність локальних товарів-замінників

Обґрунтування факторів конкурентоспроможності наведено в табл. 4.10.

Таблиця 4.10 – Обгрунтування факторів конкурентоспроможності

№	Фактор конкурентоспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Універсальність системи	Можливість використання без суттєвих змін, та використання кросплатформеного додатку

2	Швидкість роботи	За рахунок навченної моделі нейронної мережі, розпізнавння даних відбувається значно швидше, що дозволяє ефективніше реагувати на зміни.
3	Економія на зарплатах співробітників	Можливість керувати системою оператору, адже попередня постановка діагнозу не вимагає обробки результатів спеціалістом
4	Удосконалення	Система завдяки штучному інтелекту буде перенавчатись і ставати кращою

Перейдемо до порівняльного аналізу сильних та слабких сторін у порівнянні з конкуруючим продуктом. Результат порівняння зведений в таблицю 4.11.

Таблиця 4.11 – Порівняльний аналіз сильних та слабких сторін

№	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів						
			-3	-2	-1	0	1	2	3
1	Універсальність системи	16				x			

Таблиця 4.11 – Продовження

2	Швидкість роботи	14					x		
3	Економія на зарплатах співробітників	17				x			

Фінальним кроком є SWOT-аналіз відображений у таблиці 4.12.

Таблиця. 4.12 – SWOT-аналіз стартап-проекту

Сильні сторони: універсальність, швидкість обробки даних, економія, можливість довчатись	Слабкі сторони: можлива похибка, якщо дані будуть нестандартного типу
Можливості: багато інвестицій через пандемію, невелика конкуренція	Загрози: несприйняття людьми даної системи

На основі аналізу розроблено альтернативи ринкової поведінки та перелік заходів з їх впровадження відображені у таблиці 4.13.

Таблиця 4.13 – Альтернативи ринкового впровадження стартап-проекту

№	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Зацікавлення інших гравців	Продаж проекту чи його поглинання іншою компанією	Декілька років

Таблиця 4.13 – Продовження

2	Переведення продукту на модель open source	Можливість безкоштовного залучення програмістів та медичних спеціалістів	До року
3	Продаж рішень	Продаж системи лідерам ринку рентгенології	Декілька місяців

4.4. Розроблення ринкової стратегії проекту

Наведемо опис цільових потенційних споживачів продукту, що продукується стартап-проектом. Такими групами є пересічні клієнти та розробники програмного забезпечення. Дані зведено до таблиці 4.14

Таблиця. 4.14 – Вибір цільових груп потенційних споживачів

№	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент

Таблиця. 4.14 - Продовження

1	Пересічні клієнти	Ці клієнти мають потребу у використанні подібних систем	Попит високий через високу точність системи та зрозумілість інтерфейсу	На даний момент відсутня	Вхід у сегмент простий, адже в даний момент немає загрози інтенсивної конкуренції та інших перепон
2	Розробники програмного забезпечення	Розробники потребують доступну та високоточну систему розпізнавання	Попит високий через високу точність системи та хорошу документацію до використання API	На ринку майже немає продуктів які надають доступ до якісного API систему розпізнавання	Вхід у сегмент простий, адже в даний момент немає загрози інтенсивної конкуренції та інших перепон
3	Медичні заклади	Медзаклади потребують даний продукт для економії та конкуренції	Попит високий через високу точність системи та швидку інтеграцію	Інтенсивність низька	Вхід у сегмент простий, проте необхідні медичні сертифікати
Які цільові групи обрано: пересічні клієнти, розробники програмного забезпечення, медзаклади					

Таблиця 4.15 – Визначення базової стратегії ринку

№	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Залучення іноземних інвестицій	Робить ставку на універсальності продукту	Вдосконалення продукту, його потенційна конкретизація у певній сфері споживання	Стратегія диференціації

У таблиці 4.16. наведемо базову стратегію конкурентної поведінки майбутнього продукту

Таблиця 4.16 – Визначення базової стратегії конкурентної поведінки

№	Чи є проект «пер-шопрохідцем» на ринку?	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Проект є першопрохідцем на ринку	Буде шукати нових споживачів, та забирати у існуючих конкурентів	Компанія не буде копіювати основні характеристики товарів конкурентів	Наступальна

На основі вимог групи споживачів до продукту та постачальника, обраної стратегії конкурентної поведінки та базової стратегії розвитку продукту визначимо стратегію позиціонування, яку відображено у таблиці 4.17.

Таблиця 4.17 – Визначення стратегії позиціонування

№	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Простота системи, точність розпізнавання, висока продуктивність	Стратегія диференціації	Вдосконалення продукту, його потенційна конкретизація у певній сфері споживання	Простота, швидкість, точність.

4.5. Розроблення маркетингової програми стартап-проекту

Сформуємо маркетингову концепцію продукту. Результат конкурентоспроможності наведено в таблиці 4.18.

З огляду на результат, на ринку відсутній продукт, спрямований на універсальність розпізнавання з впровадженням нейронних мереж.

Таблиця 4.18 – Визначення ключових переваг концепції потенційного товару

№	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити)
1	Точність	Забезпечує більше точність розпізнавання	Краще розпізнавання, що сприяє його подальшій швидкій обробці
2	Швидкість	Забезпечує високу швидкість розпізнавання	Великі обсяги даних для розпізнавання обробляються набагато швидше, що економить час користувачів
3	Простота	Простий та зрозумілий дизайн, легкість у використанні	Легкість використання із зручним клієнтським інтерфейсом, як веб-сервіс, так і мобільний додаток. Повна та зрозуміла документація для користувачів API

Розробка трирівневої маркетингової моделі товару наведена у таблиці 4.19.

Таблиця. 4.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові
1. Товар за задумом	Швидке розпізнавання захворювання людини на основі рентген знімку із забезпеченням високої точності результату. Зручний та інтуїтивно зрозумілий клієнтський інтерфейс

Таблиця. 4.19 – Продовження

2. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Точність 2. Швидкодія 3. Швидкість клієнт-серверної взаємодії 4. Попередня обробка зображень 5. Зовнішній вигляд 6. Вартість	Нм Нм Нм Нм М М	Тл Тх Тх Тл Ор Вр
	Якість: тестування за допомогою перевірки працездатності моделі на тренувальному датасеті		
	Продаж: на звичайних маркетплейсах		
	Марка: назва організації-розробника + назва товару		
3. Товар із підкріпленням	До продажу: консультація з приводу застосування		
	Після продажу: консультація та допомога, вирішення проблем та помилок		
За рахунок чого потенційний товар буде захищено від копіювання: патенти на використання товару відповідно до державного та правового законодавства.			

В таблиці 4.20 наведено цінові межі, якими необхідно керуватися при утворенні ціни.

Розглянуто можливий збут та розробимо систему збуту. Відповідні дані наведені у таблиці 4.21.

Таблиця 4.20 – Визначення меж встановлення цін

№	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	$3000	500 – 1000$ міс	~$50000/міс	$2000-10000
2	$1000/рік	-	~$50000/міс	$800-$6000/рік

Таблиця 4.21 – Формування системи збуту

№	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Разова покупка; Річна підписка на API для розробників;	Зворотній зв’язок та технічна підтримка. Аналіз використання сервісу та швидке реагування на випадкові неочікувані ситуації.	Однорівневий	Збут продукту на маркетплейсах

Останнім кроком стане визначення концепції маркетингових комунікацій, наведений у 4.22.

Таблиця 4.22 – Концепція маркетингових комунікацій

№	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користую-ться цільові клієнти	Ключові позиції, обрані для позиціонування	Завдання рекламного повідомлення	Концепція рекламного звернення
1	Пошук рішення проблеми розпізнавання у пошукових системах	Формальні/ неформальні канали комунікацій	Простота, швидкість, точність.	Справити враження та зацікавити клієнта	Демонстрація результатів роботи

4.6 Висновки до розділу 4

У даному розділі було розроблено документацію та аналіз стартап-проекту, визначено слабкі та сильні сторони стартапу, визначено технічну можливість реалізації продукту.

Досліджено, що на ринку нема сильної конкуренції програмного забезпечення для діагностування легеневих захворювань.

Потенційна стратегія ринку має орієнтацію на медзаклади та розробників програмного забезпечення.

Згідно з визначеною стратегією позиціонування було визначено, що потрібно зосередитися на простоті, високій ефективності, точності рішень та постійному оновлені продукту. Під час дослідження вирішено розповсюджувати послуги на використання програмного продукту за моделлю річної підписки, та одноразовим продажем.

ВИСНОВКИ

Під час виконання магістерської роботи розроблено інтелектуальну систему для діагностування легеневих захворювань людини за рентгенівськими знімками використовуючи нейронні мережі.

Досліджено актуальність концепції даного програмного продукту. Визначено акцент на реалізацію даного функціоналу в рамках Android, iOS, UWP додатків. Досліджено методи машинного навчання та комп'ютерного зору, визначено їх головні переваги та недоліки і на основі цих даних складений список основних вимог до додатку.

Проведено дослідження предметної області, визначено вимоги і завдання, які повинен вирішувати додаток, наведено перелік прецедентів та варіантів використання програмного продукту. Основні прецеденти розглянуто детально із, можливими, альтернативними сценаріями. Дані про це структуровано та зображено у вигляді схем.

Виходячи із заданих вимог проведено аналіз можливості використання наявних технологій та платформ для реалізації застосунків. Обрано Xamarin, як платформу для реалізації та C# мову написання застосунку. Також проведено аналіз допоміжних бібліотек для написання застосунку, визначено та обґрунтовано доцільність їх використання.

Розроблено модуль розпізнавання легеневих захворювань на мові Python, що аналізуючи рентгенівський знімок показує результат: здоровий, легені уражені вірусним захворюванням чи легені уражені Covid-19.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Android [Електронний ресурс] – Режим доступу до ресурсу: https://developer.android.com/docs.

2. Android vs iOS [Електронний ресурс] – Режим доступу до ресурсу: https://www.ixbt.com/news/2018/02/24/ios-android-99-9.html.

3. Bellman, R. E. Perturbation techniques in mathematics, engineering and physics / R. E. Bellman. — Courier Corporation, 2003. — 214 pp.

4. Bengio, Y. Learning deep architectures for AI / Y. Bengio // Foundations and trends in Machine Learning. — 2009. — no. 1. — Pp. 1–127.

5. Bengio, Y. Representation learning: A review and new perspectives / Y. Bengio, A. Courville, P. Vincent // Pattern Analysis and Machine Intelligence. — 2013. — no. 35(8). — Pp. 1798–1828.

6. Bengio, Y. Why does unsupervised pre-training help deep learning? / Y. Bengio, D. Erhan // The Journal of Machine Learning Research. — 2010. — no. 11. — Pp. 625–660.

7. Bishop, C. M. Pattern recognition and machine learning / C. M. Bishop. — New York: Springer, 2006. — 12 pp.

8. Bradski, G. The OpenCV library / G. Bradski // Doctor Dobbs Journal. — 2000. — no. 25.11. — Pp. 120–126.

9. CNNs trained models [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@RaghavPrabhu/a-simple-tutorial-to-classify-images-using-tensorflow-step-by-step-guide-7e0fad26c22.

10. Fei-Fei, L. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories / L. Fei-Fei, R. Fergus, P. Perona // Computer Vision and Image Understanding. — 2007. — no. 106(1). — Pp. 59– 70.

11. Fei-Fei, L. Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories / L. Fei-Fei, R.

Fergus, P. 117 Perona // Computer Vision and Image Understanding. — 2007. — no. 106(1). — Pp. 59– 70.

12. Heroku [Електронний ресурс] – Режим доступу до ресурсу: https://devcenter.heroku.com/.

13. Hinton, G.E. Learning multiple layers of features from tiny images / A. Krizhevsky, G. Hinton // Computer Science Department, University of Toronto, Tech. Rep. — 2009. — no. 1.4. — Pp. 7–10.

14. iOS [Електронний ресурс] – Режим доступу до ресурсу: https://developer.apple.com/documentation/.

15. Python [Електронний ресурс] – Режим доступу до ресурсу: https://docs.python.org/3/.

15. Біологічний нейрон [Електронний ресурс] – Режим доступу до ресурсу: https://teletype.in/files/13/130fcb50-188d-49db-9d78-18949cdac03d.png.

16. Двовимірна згортка [Електронний ресурс] – Режим доступу до ресурсу: https://teletype.in/files/13/130fcb50-188d-49db-9d78-18949cdac03d.png.

17. Зв'язок комп'ютерного бачення із AI та ML [Електронний ресурс] – Режим доступу до ресурсу: https://sonix.ai/packs/media/images/corp/articles/classification-of-ai-ml-dl-44838b636b886c49760da4ab7f6b6fa4.jpg

18. Згорткова нейронна мережа [Електронний ресурс] – Режим доступу до ресурсу:https://uk.wikipedia.org/wiki/%D0%A4%D0%B0%D0%B9%D0%BB:Max_pooling_uk.png.

19. Зображення RGB з трьома каналами [Електронний ресурс] – Режим доступу до ресурсу: https://commons.wikimedia.org/wiki/File:RGB_illumination.jpg?uselang=ru

20. КТ BrightSpeed 16 [Електронний ресурс] – Режим доступу до ресурсу: http://euromedcompany.ru/upload/iblock/59f/6-16-32-srezovie140-ge-brightspeed-16-srezovoy.jpg

21. КТ Emotion 6 [Електронний ресурс] – Режим доступу до ресурсу: https://radio-med.ru/assets/cache_image/assets/resourceimages/111/Somatom_Emotion_6_450x365_a2a.png

22. КТ Ingenuity Elite [Електронний ресурс] – Режим доступу до ресурсу: https://radio-med.ru/assets/cache_image/assets/resourceimages/126/Philips_Ingenuity_128_1_450x365_a2a.png

23. Модель нейронної мережі [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wiki/BB:Colored_neural_network_uk.svg

24. Модель персептрона [Електронний ресурс] – Режим доступу до ресурсу: https://intuit.ru/EDI/21_01_20_1/1579558792-21349/tutorial/25/objects/11/files/11_4.png

25. Операція згортання [Електронний ресурс] – Режим доступу до ресурсу: https://uk.wikipedia.org/wikiBB:Conv_layers_uk.png

26. Приклад KNN класифікації [Електронний ресурс] – Режим доступу до ресурсу: wikipedia.org/wiki/Метод_k-ближайших_соседей#/media/Файл:KnnClassification.svg

27. Робота Xamarin [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/xamarin/ios/internals/architecture-images/ios-arch.png#lightbox

28. Хуршудов, А.А. Визуальный трекинг объектов для обучения локальным признакам / А.А. Хуршудов, В.Н. Марков // Новейшие исследования в современной науке: опыт, традиции, инновации: Сборник научных статей III Международной научно-практической конференции. — 2015. — С. 67–71.

29. Ярбус, А.Л. К вопросу о роли движений глаз в процессе зрения / А.Л. Ярбус // Биофизика. — 1959. — № 6. — С. 41–51.