МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СУМСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ ЕЛЕКТРОНІКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК
СЕКЦІЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ПРОЕКТУВАННЯ

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: «Програмний додаток розпізнавання хвороб дерев та рослин по зображенню листя»

за спеціальністю 122 «Комп'ютерні науки та інформаційні технології»,
освітньо-професійна програма «Інформаційні технології проектування»

Виконавець роботи: студентка групи ІТдн-61лц Михальчук Марія Олегівна

Кваліфікаційна робота бакалавра захищена на засіданні ЕК з оцінкою _____ «___»_____2020 р.

Науковий керівник _____ к.т.н., доц., Парфененко Ю.В.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Голова комісії _____ Шифрін Д. М.
(підпис) (науковий ступінь, вчене звання, прізвище та ініціали)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Суми-2020

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра комп'ютерних наук
Секція інформаційних технологій проектування
Спеціальність 122 «Комп'ютерні науки та інформаційні технології»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
Зав. секцією ІТП

_____В. В. Шендрик
«____»_____2020 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТЦІ

Михальчук Марії Олегівни

1 Тема роботи *Програмний додаток розпізнавання хвороб дерев та рослин по зображенню листя*

керівник роботи Парфененко Юлія Вікторівна, *к.т.н., доцент* ,

затверджені наказом по університету від « 21 » травня 2020 р. № 0608-III

2 Строк подання студентом роботи «12» червня 2020 р.

3 Вхідні дані до роботи Навчальна вибірка з результатами виявлення хвороб дерев та рослин по зображенню листя

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз предметної області
2) Постановка задачі
3) Проектуванні інформаційної системи розпізнавання хвороб дерев та рослин по зображенню листя
4) Розробка інформаційної системи розпізнавання хвороб дерев та рослин по зображенню листя

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Мета, задачі, актуальність, проаналізовані системи, порівняльна характеристика систем та методів прогнозування , архітектура мережі, засоби реалізації ,діаграми варіантів використання, діаграма процесів з декомпозиціями, навчальна вибірка,висновки

6. Консультанти бакалаврської роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

Дата видачі завдання _____.

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної проекту	Термін виконання етапів проекту	Примітка
1	Аналіз предметної області	02.04.20-11.04.20	
2	Постановка задачі та мети дослідження	11.04.20-20.04.20	
3	Планування проекту	20.04.20-11.05.20	
4	Проектування додатку	11.05.20-23.05.20	
5	Проектування інтерфейсу	23.05.20-04.05.20	
6	Розробка проекту	04.05.20-03.06.20	
7	Тестування проекту	03.06.20-05.06.20	
8	Оформлення документації по проекту	05.06.20-10.06.20	

Студент _____ Михальчук М.О.

Керівник роботи _____ к.т.н., доц. Парфененко Ю.В.

РЕФЕРАТ

Темою кваліфікаційної роботи бакалавра є «Програмний додаток розпізнавання хвороб дерев та рослин по зображенню листя».

Пояснювальна записка включає в себе вступ, 4 розділи, висновки, список використаних джерел із 20 найменувань та 2 додатків. Загальний обсяг роботи − 60 сторінок, у тому числі 42 сторінки основного тексту, 2 сторінки списку використаних джерел, 15 сторінок додатків.

Перший розділ кваліфікаційної роботи розкриває сутність та актуальність нашого напрямку дослідження, а також результати аналізу існуючих програмних продуктів. Ми провели аналіз ефективності використання інформаційних систем для вирішення задач розпізнавання хвороб дерев та рослин по зображенню листя. Наведено всі ключові переваги та недоліки на прикладі існуючих аналогів, а також обґрунтовано актуальність роботи та потреба в створенні власної інформаційної системи на базі загорткової нейронної мережі.

Другий розділ розкриває сутність визначеної мети та поставлених завдань для досягнення кінцевої цілі, визначення основних методів дослідження та ефективних засобів удосконалення вихідних результатів.

В третьому розділі даної роботи розкрита суть та особливості проектування роботи і розробки архітектури програмного додатку на базі нейронної мережі. Також подано інформацію про структурно-функціональне моделювання та моделювання варіантів використання інформаційної системи. Визначено базові етапи проектування роботи.

Четвертий розділ розкриває суть процесу розробки інформаційної системи на базі нейронної мережі зі згорткою через архітектуру програмного додатку. На підтвердження додаються знімки екрану, що демонструють розробку інформаційної системи та використання продукту.

Результатом виконання роботи виступає розробка програмного додатку розпізнавання хвороб дерев та рослин по зображенню листя.

Ключові слова: інформаційна система, машинне навчання, архітектура, нейронна мережа зі згорткою, розпізнавання хвороб.

Зміст

ВСТУП

Віруси викликають глибокі незворотні зміни в рослинах, призводять до порушення вуглецевого та азотного обміну, пригнічують їх ріст та розвиток, що значно знижує урожайність (сільськогосподарських культур), а у багатьох випадках стають причиною загибелі рослин та дерев. Незалежно від підходу та методів лікування рослин, правильне визначення захворювання при його появі є важливим кроком для ефективного управління ним. Останнім часом такі зусилля додатково підтримуються обміном інформації для діагностики захворювань через Інтернет, що сприяє зростанню обізнаності та комплексному підходу до боротьби з захворюваннями рослин у всьому світі. Поєднання Інтернету, зростаючого глобального проникнення смартфонів та недавніх досягнень штучного інтелекту, дали поштовх для розвитку нейронних мереж, які прокладають шлях для діагностики захворювань, сприйнятих та опрацьованих смартфонами та іншими видами сучасної техніки.

Об’єктом дослідження дипломного проектування є: процес виявлення та розпізнавання захворювань дерев та рослин за ступенем ураження їх листя через програмний додаток на базі архітектури згорткової нейронної мережі.

Предметом дослідження є метод, який презентує особливості розпізнавання та класифікації захворювань листя рослин за допомогою згорткової нейронної мережі.

Метою кваліфікаційної роботи бакалавра є розроблення програмного додатку, що буде використовуватися для демонстрації та підвищення точності результатів нейронної мережі зі згорткою для виявлення захворювань рослин за ступенем ураження їх листя.

Для реалізації мети бакалаврської роботи необхідно вирішити задачі:

- провести огляд предметної області розробки програмних додатків з функцією класифікації зображень;
- виконати планування робіт та обрати засоби реалізації програмного додатку;
- виконати побудову та навчання нейронної мережі;

- розробити програмний додаток розпізнавання захворювань рослин за ступенем ураження їх листя по фото, одержаного з мобільного пристрою.

Для досягнення виконання побудови та навчання нейронної мережі потрібно вирішити такі задачі:

1. Знайти навчальний набір даних для подальшого його масштабування, створення тестів на його основі.

2. Налаштувати вихідний шар вихідного об’єму даних через карти ознак (формувати окрему двомірну карту активації), які складаються вздовж «глибини» вхідного об’єму.

3. Створити архітектуру нейронної мережі зі згорткою з врахуванням параметрів шару згортки тобто набору фільтрів для навчання (також можна зустріти назву – ядра Кернела).

4. Здійснити навчання нейронної мережі зі згорткою через фільтри, що активуються при виявленні певної візуальної особливості.

5. Провести оцінку точності моделі.

Причиною ініціалізації проекту є потреба створення інформаційної платформи для розпізнавання та класифікації захворювань листя рослин. Розроблена нейронна мережа може бути також використана для подальшого користування.

Результатом проекту буде програмний додаток, у складі якого буде модель, реалізована на базі нейронної мережі зі згорткою, що надаватиме змогу потенційним користувачам мати максимально повне уявлення про виявлення захворювань рослин за ступенем ураження їх листя.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Актуальність проблеми

Розробка програмного додатку на базі нейронної мережі зі згорткою є одним з напрямків у сфері розпізнавання та класифікації захворювань листя рослин, з метою підвищення точності та швидкості результатів діагностики, через впровадження автоматизованих моделей для виявлення хвороб рослин на основі традиційних алгоритмів машинного навчання (random forest та support vector machine (SVM)). Актуальність проблеми полягає в доведенні ефективності встановлення багатошарової моделі нейронної мережі для реалізації ідентифікації хвороб листя шляхом обчислення значень хроматичності, зміни забарвлення та форми листя аналізуючи використання методів розпізнавання підтримуючої векторної машини (SVM), дискримінантного методу аналізу для виявлення уражень з врахуванням особливостей структури листя, а також аналіз екологічних факторів впливу.

Щоб вирішити таку проблему, модель нейронної мережі зі згорткою (CNN) пропонує багато можливостей, включаючи: здатність до узагальнення різних мережевих структур CNN для різних типів хвороб листя; розробку різноманітних підходів для запобігання втрати врожаю, лісових насаджень через хвороби рослин; підвищення точності визначення захворювання для ефективного управління заходами щодо його ліквідації; удосконалення механізмів автономної селективності для обприскування пестицидами ураженого листя що сприятиме зниженню виробничих витрат у сільському та лісовому господарстві, використовуючи автоматичних роботів-дронів для виявлення рослин на посівах та проведення селективної ін'єкції; покращення спектроскопії та візуалізації виявлених захворювань рослин [1].

Таким чином, на базі архітектури нейронної мережі зі згорткою, можна створити досить точну модель для аналізу особливостей ураженості листя рослин з метою класифікації видів хвороб на зображеннях, яких модель раніше не бачила.

1.2. Аналіз архітектури нейронних мереж зі згорткою

Процес розробки архітектури, навіть, найелементарнішої нейронної мережі немислимий без попередньої постановки завдань і цілей а також ретельного аналізу навчального набору даних для подальшого його масштабування та створення тестів на його основі. Аналіз архітектури нейронних мереж, аналіз цільових споживачів (користувачів) та підвищення точності майбутньої моделі, виявлення «сильних» і «слабких» сторін проекту – розробка архітектури нейронних мереж завжди починається з виконання подібних завдань.

Для досягнення мети було проведення дослідження архітектур нейронних мереж, які брали участь у ILSVRC з 2012 по 2017 рік та досягли хороших результатів класифікації:

– AlexNet – нейронна мережа зі згорткою, яка містить вісім шарів з ваговими коефіцієнтами. Лише перших п’ять шарів є згортковими, а решта три - звязними. Вихідні дані аналізуються з використанням функції втрат softmax, яка формує розподіл, в середньому до 1000 міток класів. Нейронна мережа оптимізує багатолінійну логістичну регресію, що еквівалентно оптимізації середнього по всіх навчальних спробах тренування з використанням логарифма ймовірності правильного відображення з розподілу очікування. Особливість побудови ядра другого, четвертого і п'ятого згорткових шарів пов'язані тільки з тими картами ядра в попередньому шарі, які знаходяться на одному і тому ж графічному процесорі. Ядра третього згорткового шару пов'язані з усіма картами ядер другого шару. Нейронні звязки в повнозв'язних шарах активуються через активацію нейронів попередніх шарів. Можна дійти до висновку про те, що AlexNet містить 5 шарів зі згорткою і 3 повнозв'язних шари, а ReLu (лінійна функція активації) застосовується після кожного згорткового і повнозв'язаного шару. DropOut використовується перед першим і другим повнозв'язаними шарами. Мережа містить 62,3 мільйона параметрів і витрачає 1,1 мільярда обчислень при прямому проході (рис. 1.1) [2].

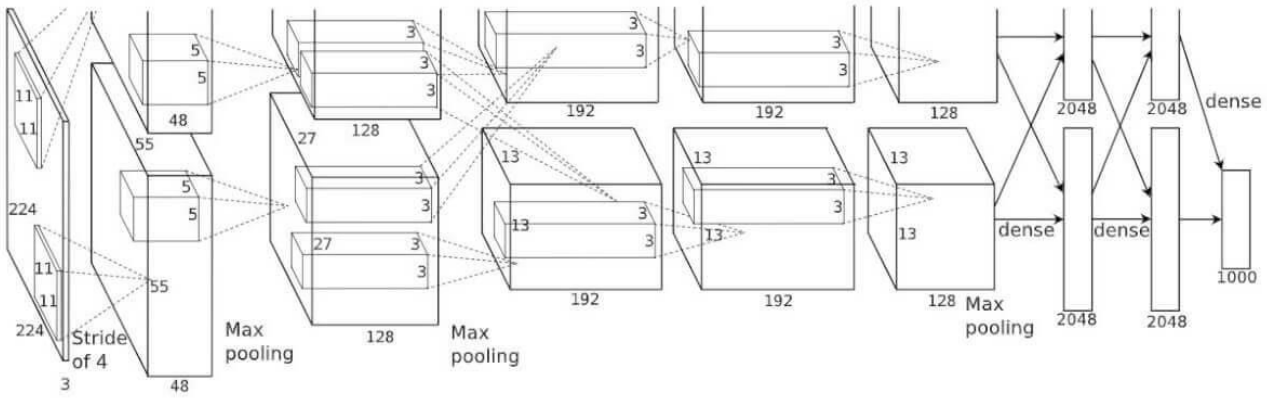


Рисунок 1.1 - Архітектура мережі AlexNet

– VGGNet (мережа Visual Geometry Group) – складається з 16 згорткових шарів і рівномірною архітектурою, що легко реалізувати на простих обчислювальних пристроях. В одній з конфігурацій використовується фільтр згортки 1x1, який може бути представлений як лінійна трансформація вхідних каналів (з подальшою нелінійністю). Крок згортки фіксується на значенні 1 піксель. Просторовий пулінг здійснюється за допомогою п'яти max-pooling шарів, які слідують за одним з згортальних шарів (не всі згорткові шари мають наступні max-pooling). Після стека згортальних шарів (який має різну глибину в різних архітектурах) йдуть три повнозв'язних шари: перші два мають по 4096 каналів, третій - 1000 каналів (так як в змаганні ILSVRC потрібно класифікувати об'єкти по 1000 категоріям; отже, класу відповідає один канал). Всі приховані шари забезпечені ReLU (рис.1.2) [3].

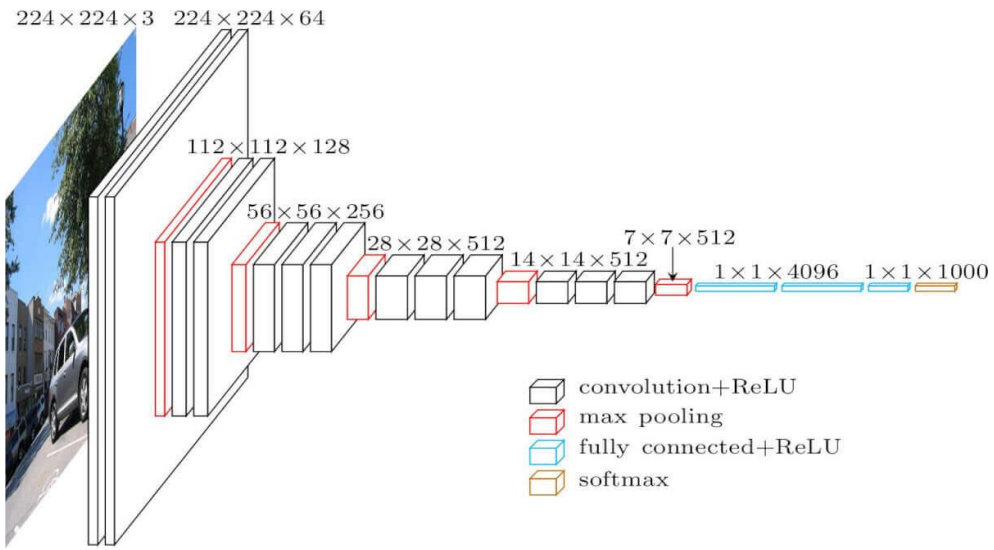
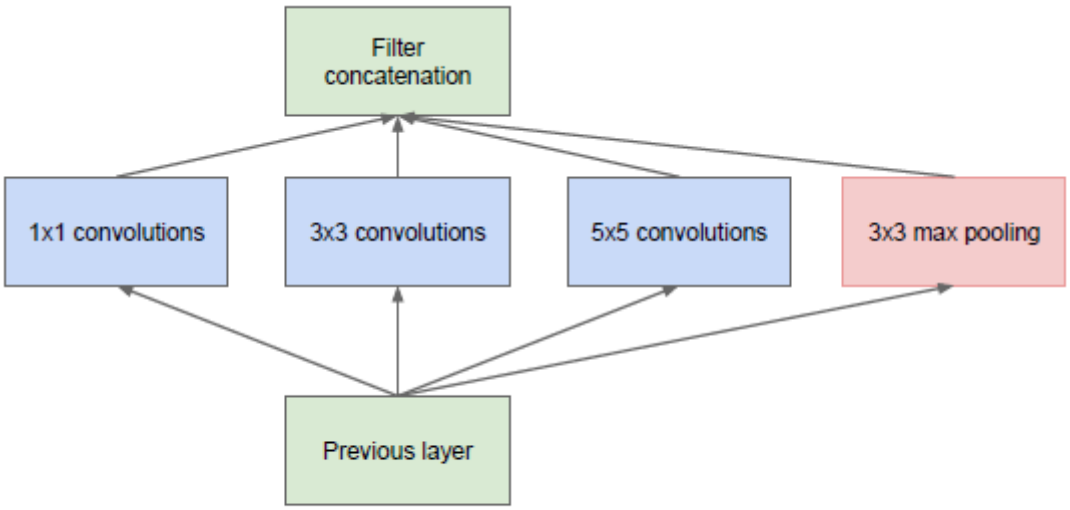


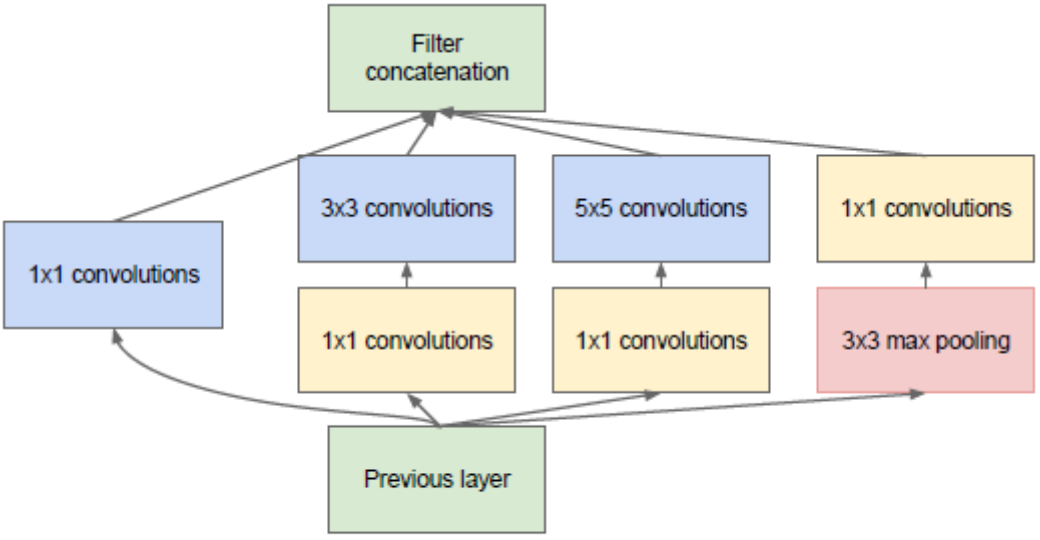
Рисунок 1.2 - Архітектура мережі VGGNet

– GoogLeNet (мережа Google) – відображає те, що більшість активацій у глибокій мережі є або непотрібними (нульовими значеннями), або надмірними (рис.1.3). Тому найефективніша архітектура глибокої мережі матиме рідкісний зв'язок між активаціями, що означає, що всі 512 вихідних каналів не будуть мати з'єднання з усіма 512 вхідними каналами. Існують способи знищити такі з'єднання, що призведе до незначного полегшення при обробці. В цьому випадку є три training head (прямокутники, відмічені жовтим на рис.1.4) для простішого тренування глибокої мережі. У кожному додатковому training head є шари FC layers, які прогнозують той же клас на основі низьких рівнів, щоб до нижніх рівнів сигнал доходив швидше.



(a) Inception module, naïve version

Рисунок 1.3 - Типовий модуль архітектури GoogLeNet.



(b) Inception module with dimensionality reduction

Рисунок 1.4 - Типовий модуль архітектури GoogLeNet зі згорткою.

– ResNet (залишкова мережа), в якій розмір зображення змінюється за допомогою випадкової вибірки його короткої сторони для збільшення масштабу. Кадрування 224 × 224 вибирається випадковим чином з зображення або його горизонтального зміщення з вирахуванням середнього значення для кожного пікселя. Швидкість навчання стартує з 0,1 і ділиться на 10, коли зміна помилок виходить на плато, моделі навчаються аж до 60 × 10000 ітерацій (рис.1.5) [3].

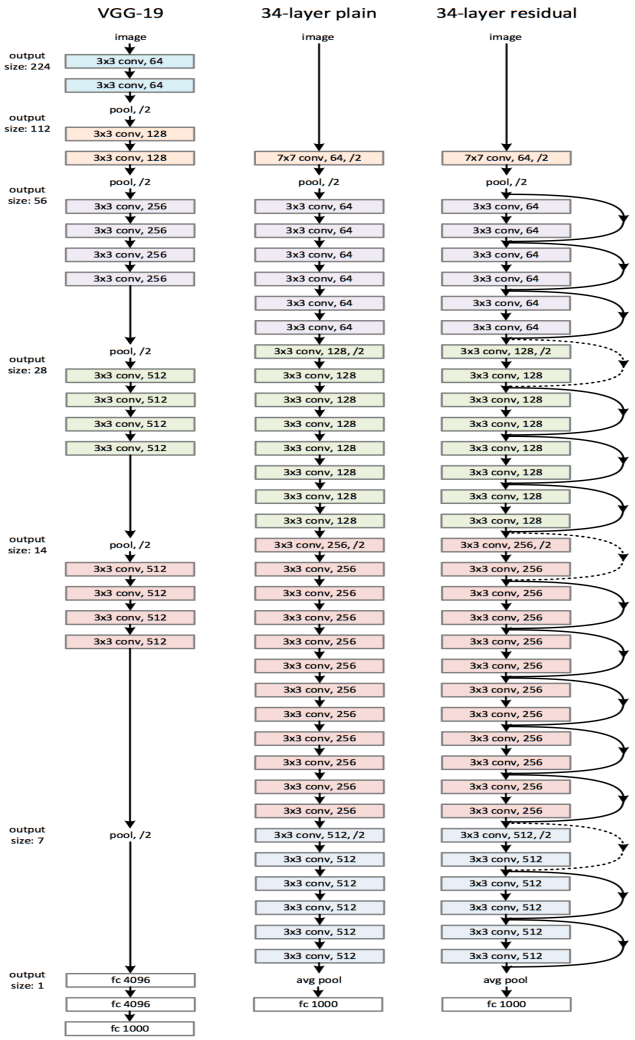


Рисунок 1.5 - Архітектура **ResNet**

– SENet (Squeeze-and-Excitation Networks, мережа стискання та збудження) має так званий будівельний блок для CNN, що покращує взаємозалежність каналу майже без обчислювальних витрат. Через блок "Стиснення та збудження" (SE), який адаптивно рекалібруються канальні відповіді функції, чітко моделюючи взаємозалежності між каналами. Основна ідея полягає в додаванні параметрів до кожного каналу згорткового блоку, щоб мережа могла адаптивно налаштувати зважування кожної картки функцій. Принцип роботи нейромережі

полягає в створенні функції для отримання вхідним згортковим блоком і поточною кількістю каналів, які вона має внаслідок стиснення каналу на одне числове значення, використовуючи середнє об'єднання. Повністю з'єднаний шар з подальшим функцією ReLU додає необхідної нелінійності. Другий повністю з'єднаний шар з подальшим активацією Sigmoid надає кожному каналу функцію гладкого з'єднання (smooth gating function) (рис.1.6).

–

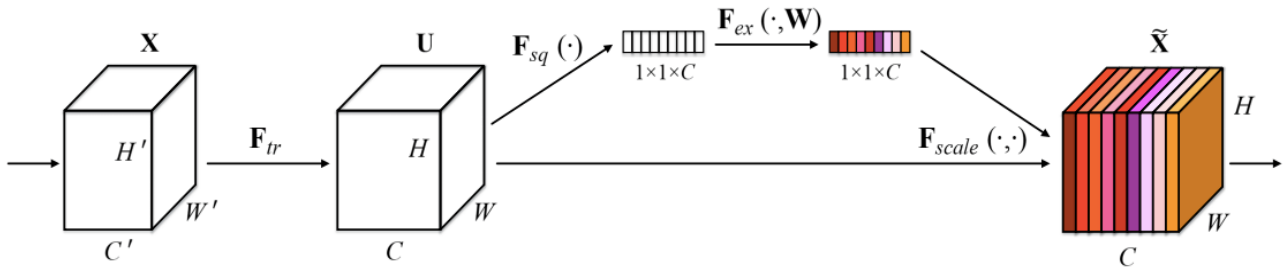


Рисунок 1.6 - Архітектура SENet

Проаналізувавши нейронні мережі зі згорткою, які представляються користувачеві було проведено порівняльний аналіз (табл.1.1 та рис. 1.7), де було виявлено позитивні та негативні сторони, які при створені дипломного проекту треба уникати.

Недоліки:

- наявність великої кількості параметрів зумовлює складність обробки зображень;
- час навчання нейромереж подвоюється з показником DropOut 0,5;
- згорткові шари, на які припадає 6% всіх параметрів, здійснюють 95% обчислень;
- Local Response Normalisation (нормалізація) веде до збільшення споживання пам'яті та часу виконання коду;

До переваг аналогів архітектур нейромереж зі згорткою можна віднести:

- функцію активації Relu замість арктангенса для додавання в модель не лінійності (за рахунок цього при однаковій точності методу швидкість зростає);
- використання дропаутів замість регуляризації (вирішує проблему перенавчання);
- перекриття об'єднань для зменшення розміру мережі (за рахунок цього рівень помилок першого і п'ятого рівнів знижуються до менш ніж 1%).

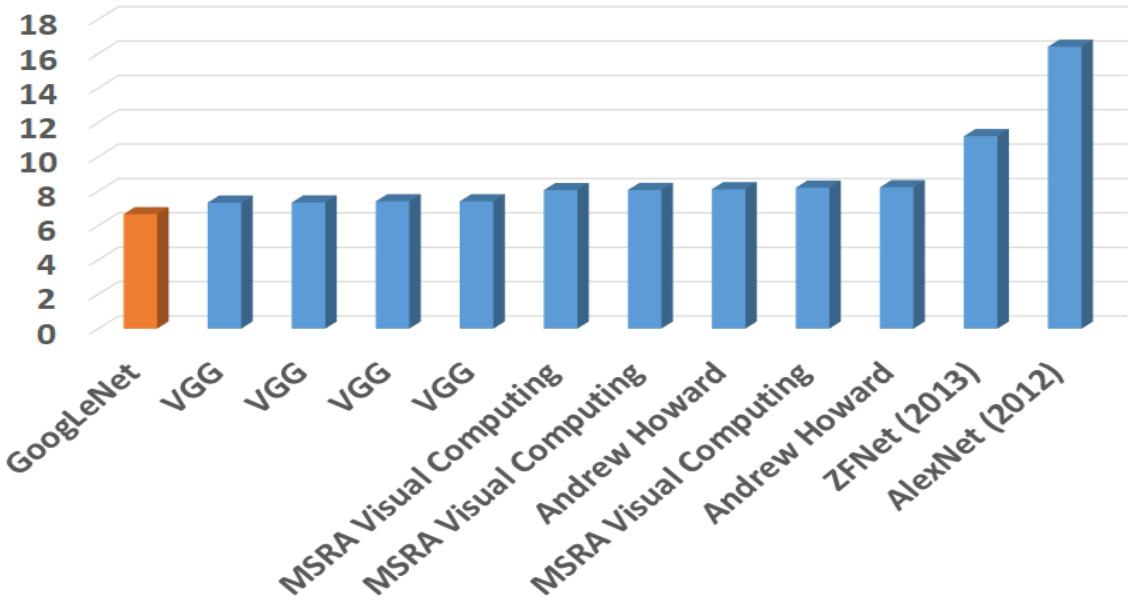


Рисунок 1.7 - Порівняння нейронних мереж зі згорткою за рівнем отриманих помилок при навчанні

Таблиця 1.1 – Аналіз нейронних мереж зі згорткою

	Team	Year	Place	Error (top-5)	Uses external data
AlexNet	SuperVision	2012	1st	16.4%	no
	SuperVision	2012	1st	15.3%	Imagenet 22k
ZFNet	Clarifai	2013	1st	11.7%	no
SPPNet	Clarifai	2013	1st	11.2%	Imagenet 22k
	MSRA	2014	3rd	7.35%	no
VGGNet	VGG	2014	2nd	7.32%	no
GoogLeNet	GoogLeNet	2014	1st	6.67%	no

У роботі для описаної вище проблеми класифікації буде використано нейронну мережу зі згорткою, а саме архітектури AlexNet та GoogLeNet.

Цей вибір пояснюється тим, що GoogleNet – це набагато глибша та ширша архітектура ніж розглянуті аналогічні з 22 шарами, при цьому вона має значно меншу кількість параметрів (5 мільйонів параметрів) у мережі, ніж AlexNet (60 мільйонів параметрів). Застосування архітектури “мережа в мережі” у вигляді вхідних модулів є ключовою особливістю архітектури GoogLeNet.

Модуль створення використовує шари згортки 1 × 1, 3 × 3 та 5 × 5 разом із шаром максимального об'єднання, що дозволяє йому паралельно фіксувати

різноманітні ознаки. З точки зору практичності реалізації, нам доведеться постійно контролювати кількість пов'язаних обчислень тому, що шар згортки 1 × 1 з вищезгаданими 3 × 3, 5 × 5 шарами згортки (а також після шару максимального об'єднання) додається для зменшення розмірності.

Нарешті, шар конкатенації фільтрів просто об'єднує виходи всіх цих паралельних шарів. Хоча це утворює єдиний модуль створення та обєднання, у виді архітектури GoogLeNet, яку ми використовуємо в наших експериментах, будемо використувлвати 9 модулів створення (inception modules).

Ефективність обох цих архітектур буде проаналізовано на наборі даних PlantVillage, навчаючи модель з нуля в одному випадку, а потім шляхом адаптації вже підготовлених моделей (навчених на наборі даних ImageNet), використовуючи трансферне навчання (transfer learning). У випадку навчання з передачі ми повторно ініціалізуємо ваги шару fc8 у випадку AlexNet та втрати {1,2,3} / шари класифікатора у випадку GoogLeNet. Тоді, навчаючи модель, ми не обмежуємо дослідження будь-якого з шарів, як це іноді робиться для передачі навчання.

Іншими словами, ключова відмінність між цими двома підходами до навчання (передача проти тренування з нуля) полягає в початковому стані ваг у кількох шарах).

1.3. Аналіз програмних додатків з можливістю розпізнавання хвороб по зображенню

Оскільки нейронна мережа зі згорткою буде поданою у вигляді мобільного додатку на GooglePlay, то обов’язковим етапом бакалаврської роботи є огляд інформаційних систем для інтелектуальної обробки зображень, а саме Leafsnap, Intello Track та Face2Gene.

Leafsnap – це перший в світі мобільний додаток для ідентифікації рослин за допомогою візуального пошуку, поданий на рис. 1.8., який дозволяє користувачам ідентифікувати види дерев, сфотографувавши листя дерева. Окрім назви, виду, Leafsnap надає фотографії з високою роздільною здатністю. Мобільний додаток дає

доступ до інформації про квіти, плоди, насіння та кору дерева – розкриваючи користувачам всеохоплюючу інформацію про вид, родину тощо.

Рисунок 1.8 - Мобільний додаток Leafsnap на GooglePlay

Даний алгоритм, створений за допомогою методів глибокого навчання, був навчений на 400 тисячах зображень рослин, що мешкають на території Коста-Ріки і Франції, може правильно ідентифікувати рослини в 90 відсотках випадків. Алгоритми для допомоги дослідникам в визначенні невідомих видів рослин вже давно розробляються: наприклад, програма LeafSnap може визначати рослини з території Північної Америки за знімком листа. Однак, всі відомі на сьогоднішній день програми з автоматичного визначення рослин були розроблені з використанням невеликих баз даних, через що точність їх роботи сильно обмежена.

У своїй новій роботі дослідники представили алгоритм автоматичного розпізнавання рослин за допомогою комп'ютерного зору. Їх технологія створена на основі нейронних мереж зі згорткою CNN, яка відповідає за автоматичну обробку зображень з метою виокремити певні характеризують його патерни (наприклад, колір або форму зображеного об'єкта) [5].

Для тренування і тестування нейромережі були використані п'ять баз даних: дві бази даних містили відсканованої зображення рослин з гербаріїв, дві - зображення живих рослин в природі та відсканованого листя і ще одна база даних (один мільйон зображень, взятих з ImageNet) була використана для попереднього тренування і не пов'язана з рослинами.

Intello Track – це мобільний додаток на базі штучного інтелекту який дозволяє оцінювати якість харчових продуктів (вирощеного урожаю). Користувач завантажує фото свого продукту, натиснувши на обране зображення та отримує повну інформацію про нього (рис. 1.9) [6].

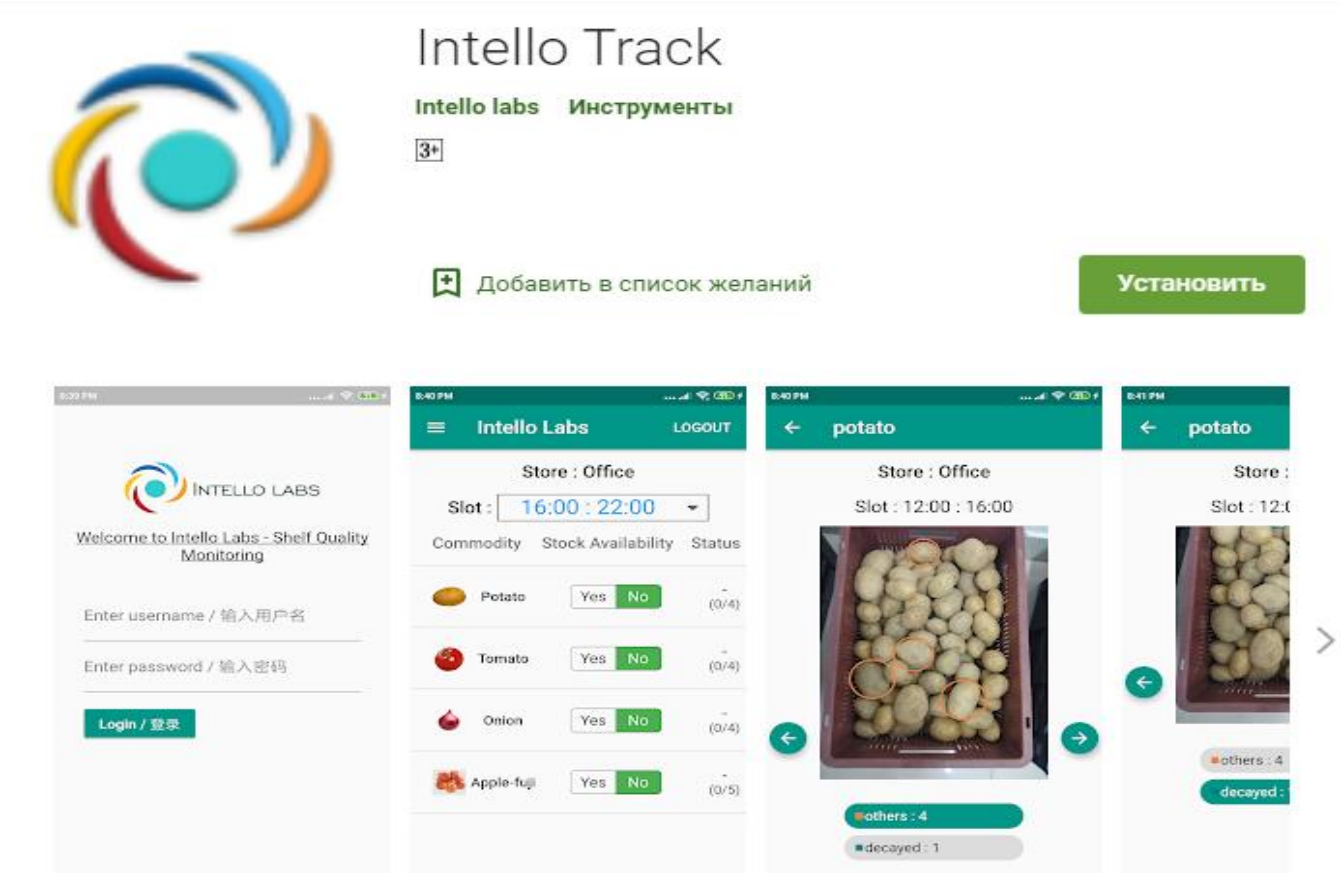


Рисунок 1.9 - Мобільний додаток Intello Labs на GooglePlay.

Потім зроблена фотографія переходить до хмарних сховищ, де над нею працює алгоритм Intello Labs. Після цього результати надсилаються на мобільний пристрій, де висвітлюється рейтинг продукту [6].

Face2Gene – це мобільний додаток розроблений компанією FDNA, що розробляє проекти в сфері цифрової охорони здоров'я в Бостоні. Додаток використовує алгоритм глибокого навчання Face2Gene для виявлення рідкісних генетичних порушень, спочатку надавши йому понад 17 000 зображень людей з

діагнозом одного із 216 генетичних синдромів та відхилень. Виходячи з цих даних, він навчився шукати відмінні риси обличчя, пов'язані з конкретними розладами [8].

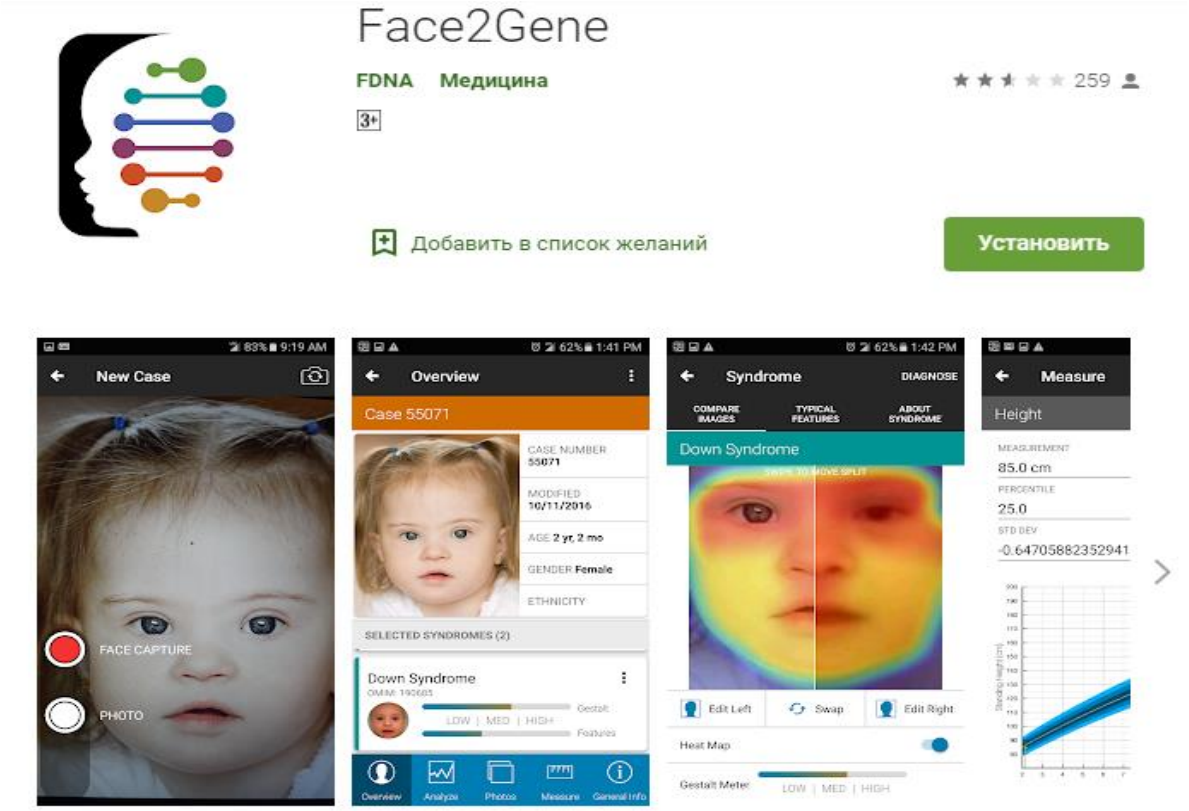


Рисунок 1.10. Мобільний додаток Face2Gene на GooglePlay.

Коли дослідники протестували додаток на 502 зображеннях, яких раніше не бачили, Face2Gene поставив правильний діагноз приблизно в 65% випадків. Коли була надана можливість вибору 10 можливих діагнозів, правильна з них становила список Face2Gene в 91% випадків [8].

Команда також перевірила здатність нейронної мережі зі згорткою Face2Gene по виявленню розладів по вроджених дефектах.

Порівнявши інтерфейс та можливості вищевказаних мобільних додатків було зроблено висновок, що найбільш підходящим є поєднання елементів двох інтерфейсів таких додатків як Leafsnap та Face2Gene. Leafsnap має великий набір готових даних для аналізу різного виду рослин, плодів та квітів і хороший алгоритм роботи нейронної мережі зі згорткою, але для нашого додатку, також основою якого буде нейронна мережа зі згорткою, в подальшому важливо не тільки діагностувати та запобігати поширенню загальних захворювань рослин, але й специфічних (які рідко, але всеж трапляються) і саме для цього ми будемо

використовувати деякі елементи та ідеї з Face2Gene, що дасть нам суттєву перевагу над іншими мобільними додатками [8].

Порівняння основних характеристик програмних додатків для розпізнавання захворювань таблиці 1.2.

Таблиця 1.2 − порівняльна характеристика програмних додатків

	Leafsnap	Intello Track	Face2Gene
Точність розпізнавання	90%	90%	91%
Мобільний додаток	+	+	+/-
Зручність	+	+/-	-
Наявність великої бази данних для аналізу	+	+	+
Висока вартість	-	+	+/-
Швидкість обробки зображень	-	+/-	-

Отже для конкурентоспроможності потрібно створити інформаційну систему для розпізнавання хвороб дерев та рослин по зображенню листя з точністю більше 90% та великою базою даних для більш кращого прогнозування. Також потрібно приділити достатню увагу зручності інтерфейсу та швидкості обробки зображень. Ця система буде корисна аграріям, працівникам лісового господарства для розпізнавання хвороб дерев, рослин по зображенню листя з метою вчасного лікування, зменшенню використання пестицидів, ефективному використанні добрив та запобігання розповсюдження хвороб і вірусів.

2. ПОСТАНОВКА ЗАДАЧІ ТА МЕТОДИ ДОСЛІДЖЕННЯ

2.1. Мета та задачі дослідження

Дипломний проект призначений для того, щоб користувачі програмного додатка на базі нейронної мережі могли використовувати багатошарову модель нейронної мережі для реалізації ідентифікації хвороб листя, а саме для виявлення уражень з врахуванням особливостей структури листя.

Інформаційна система повинна бути реалізована у вигляді програмного продукту на базі нейронної мережі зі згорткою, що забезпечуватиме відображення між входом (наприклад, зображенням хворої рослини) на вихід – наприклад, парою хвороби врожаю.

Написати, якою саме нейронною мережею, якого типу.

В нашій нейронній мережі зі згорткою будемо використовувати ReLU (Rectified Linear Unit).

Отже, використання інформаційної системи спрямоване на ідентифікацю хвороб дерев та рослин, шляхом обчислення значень хроматичності, зміни забарвлення та форми листя.

Впровадження інформаційної системи передбачає такі результати:

- введення і зберігання інформації по видам рослин, дерев (зображення листя);
- редагування інформації про вид захворювання та ступінь ураження рослини;
- пошук інформації по певним видам рослин, критеріям та захворюванням;
- отримання точного результату аналізу даних по фотографії для конкретного користувача програмного додатка.

Також було сформоване та затверджене технічне завдання, яке було розміщене у додатку А.

2.2. **Вибір методів та інструментів розробки програмного додатку розпізнавання хвороб дерев та рослин по зображенню листя на основі штучного інтелекту**

Високорівневі вбудовані структури даних (набори даних) поєднуються з динамічним набором тексту і через алгоритм функціонального зв'язування роблять її дуже привабливою для розробки додатків, а також, в якості мови сценаріїв. Інтерпретатор Python безкоштовний на всіх основних платформах, а також має велику стандартну бібліотеку.

Для розробки програмного додатку було обрано версію Python 3.8.3. Python – високорівнева, інтерпретована, об'єктно-орієнтована мова програмування з динамічною семантикою. Переваги цієї версії над іншими полягають в тому, що вона підтримує протокол PEP 590 (Vectorcall) який підтримує новий API для оптимізації викликів об'єктів та протокол PEP 574 (Pickle protocol 5 with out-of-band data), який охоплює додаткові метадані, необхідні для буферів даних (новий тип PickleBuffer для реалізацій __reduce_ex__ для повернення даних поза діапазоном буферів даних) саме через новий параметр buffer_callback при підборі, для обробки позадіапазонних буферів даних. Python підтримує різноманітні модулі та пакети, що заохочують модульність програми і повторне використання коду [9].

Також для розробки програмного додатку потрібно підключити бібліотеки для машинного навчання.

TensorFlow – це відкрита програмна бібліотека для машинного навчання, розроблена Google для вирішення завдань, побудованих та на навчанні нейронних мереж з автоматичним набором даних та класифікацією форм. Використовується як для дослідження, так і для розробки власних продуктів Google. Основний API для роботи з бібліотекою, що реалізується для Python, також сумісний з такими мовами програмування як R, C Sharp, C ++, Haskell, Java, Go і Swift. Принцип роботи даної бібліотеки дозволяє розробникам створювати графіки потоку даних – структури, що описують, як саме дані переміщуються через граф або набір вузлів обробки. Кожен вузол у графі

представляє математичну операцію, і кожне з'єднання або край між вузлами є багатовимірним масивом даних або тензором [10].

Keras – це бібліотека для тестування та навчання нейронних мереж з відкритим кодом, написана на Python. Дана бібліотека сумісна з TensorFlow, Microsoft Cognitive Toolkit, PlaidML, R або Theano. Вона призначена для експериментів з глибокими нейронними мережами і фокусується на зручності користування, модульності та розмірності. Keras містить численні реалізації часто використовуваних нейронних мережних будівельних блоків, таких як шари, цілі, функції активації, оптимізатори, різноманітні інструменти для полегшення роботи з зображеннями, текстовими даними, з метою спрощеного написання глибокого нейронного мережевого коду. Окрім стандартних нейронних мереж, Keras має підтримку згорткових і рециндентних нейронних мереж [11].

NumPy – це бібліотека для мови програмування Python, яка підтримує функціонування великих, багатовимірних масивів і матриць та математичних функцій високого рівня для роботи над цими масивами. NumPy націлений на реалізацію еталонної програми Python, яка є оптимізатором інтерфейсу байт-коду. NumPy дозволяє виконати основні операції над n-масивами та матрицями: додавання, віднімання, ділення, множення, транспортування, вивільнення визначення тощо.

Pandas – це бібліотека на мові Python для обробки та аналізу даних, їх структуризації у вигляді числових таблиць і тимчасових рядів. За допомогою даної бібліотеки можна індексувати масиви двовимірних даних, переформатовувати, додавати, видаляти дані, обробляти інформацію тимчасових рядів, формувати тимчасові періоди та змінні інтервали [11].

Для підключення нейронної мережі зі згорткою та коректної роботи програмного додатка обрано інструмент ML Kit, який дозволяє легко застосовувати методи ML при розробці програмних додатків, поєднуючи технології ML та Google, такі як API Cloud Vision Google, TensorFlow Lite та API нейронних мереж на Android разом, в один SDK.

ML Kit – це мобільний SDK, який дає можливіть машинного навчання Google для додатків Android та iOS в потужному але простому у використанні

пакеті. Незалежно від можливості реалізованих у реальному часі моделей на мобільних пристроях чи ПК, оптимізованих для даних пристроїв, або гнучкості користувацьких моделей TensorFlow Lite, ML Kit дозволяє лише за допомогою декількох рядків коду підключити нейронну мережу будь-якого рівня складності. Завдяки API можливе виявлення та відстеження об’єктів з підключенням ML Kit на пристрої, які локалізується та відстежуються в режимі реального часу. Виявивши та відфільтрувавши об’єкти, ми можемо перемістити їх до хмарного сховища, що в свою чергу допоможе зменшити навантаження на ПК чи смартфон та дозволить пришвидшити обробку даних [13].

Для розробки архітектури та дизайну програмного додатку використано Android Studio. Це інтегроване середовище розробки (IDE) для роботи з платформою Android. Android Studio забезпечує гнучку систему побудови на основі Gradle, швидку та багатофункціональний емуляцію створених об’єктів, застосування зміни коду і зміни ресурсів у своєму запущеному додатку без перезавантаження програми, використання шаблонів коду та інтеграції з GitHub, широкі інструменти та рамки тестування, а також інструментів оптимізації, зручності використання, сумісності версій. Вбудована підтримка Google Cloud Platform спрощує інтеграцію Google Cloud Messaging та App Engine.

2.3 Набір даних для навчання нейронної мережі

Найважливішим етапом перед тренуванням нейронної мережі зі згорткою є створення набору даних для навчання, оскільки це напряму вплине на точність та коректність ідентифікації захворювання рослин по зображенню листя.

Ми використовуємо готовий набір даних PlantVillage, який розміщений на платформі GitHub. Наш набір даних включатиме в себе такі види наборів, як навчальний (training dataset), підкріплення (validation dataset), тестовий (test dataset) та допоміжний (holdout set).

Навчальний набір буде використовуватися в алгоритмі машинного навчання для активації зв’язків між функціями та цільовою змінною. У контрольованому навчальному наборі позначаються з врахуванням відомих параметрів виходу.

Набір підкріплення також є підмножиною основного набору даних, до яких ми застосовуємо алгоритм машинного навчання, з метою підвищення точності прогнозу та додаткової активації визначених зв’язків після вихідних результатів навчального набору для цільової змінної й іншими функціями набору даних.

Тестовий набір, є бере участь у кінцевому етапі тренування нейронної мережі з метою забезпечення ефективної оцінки кінцевої моделі. Важливо перевірити роботу алгоритму, та виявити помилки і правильно налаштувати вагові коефіцієнти, щоб покращити результати виходу нейронної мережі.

Допоміжний набір, є підмножиною вхідних даних, який дає остаточну оцінку ефективності моделі навчання нейромережі після того, як вона пройшла навчання та затвердження. Він буде використовуватися лише для прийняття складних рішень про те, які алгоритми застосовувати для вдосконалення нейронної мережі зі згорткою, особливо у випадках мутації вірусів та виведенню нових стійкіших видів рослин.

Приклад зображень, які містяться в наборі даних подані на рисунку рисунку 2.1.

Рисунок 2.1 Зображення, які містяться в наборі даних PlantVillage

Набір даних буде зберігатися та оновлюватися через базу даних. Спочатку буде зазначатися вид рослини, тип вірусу, ID фотографії, вид набору та розширення файлу, наприклад:

- raw/color/Tomato___Tomato_Yellow_Leaf_Curl_Virus/5fee1f4a-3bcb-4d04-910b-1749ee96bfd1___YLCV_GCREC 2724.JPG SVM/test/35/30df07de-932a-4ca2-965d-afcc11250962.JPG
- raw/color/Cherry_(including_sour)___healthy/d488e8a6-948b-4805-9852-1a1825890e04___JR_HL 9554.JPG SVM/test/6/83cc56f9-beb8-4f92-9a49-d4e2822158cf.JPG

Набір даних PlantVillage буде використовувати картинки RGB, однакового розміру 256 х 256 пікселів та розширення JPEG. В першу чергу в наборі будуть фотографії для ідентифікації найбільш поширеніших хвороб, вірусів рослин та дерев, які можна виявити по зображенню листя, таких як білий наліт, вірус мозаїка, поморшклість, бактеріальні ураження, антракноз, ризоманія, фітоплазмоз, борошниста роса, біла гниль, леоціоміцети, ріжки пурпурові, сажка тощо.

3 МОДЕЛЮВАННЯ РОБОТИ ПРОГРАМНОГО ДОДАТКУ РОЗПІЗНАВАННЯ ХВОРОБ ДЕРЕВ ТА РОСЛИН ПО ЗОБРАЖЕННЮ ЛИСТЯ

3.1 Структурно-функціональне моделювання процесу розпізнавання хвороб дерев та рослин по зображенню листя за допомогою програмного додатку

З метою кращого розуміння цілей та функцій інформаційного процесу у рамках розробки програмного додатку на базі нейронної мережі зі згорткою використаємо структурно-функціональне моделювання SADT (IDEF0). В моделі IDEF0 найбільше акцентується увага на ієрархічності об'єктів.

Ми зосередимо свою увагу на SADT моделі через особливості функціонування системи та на її основних об'єктах. Декомпозиція моделювання процесу розпізнавання IDEF0 подана на рис. 3.1.

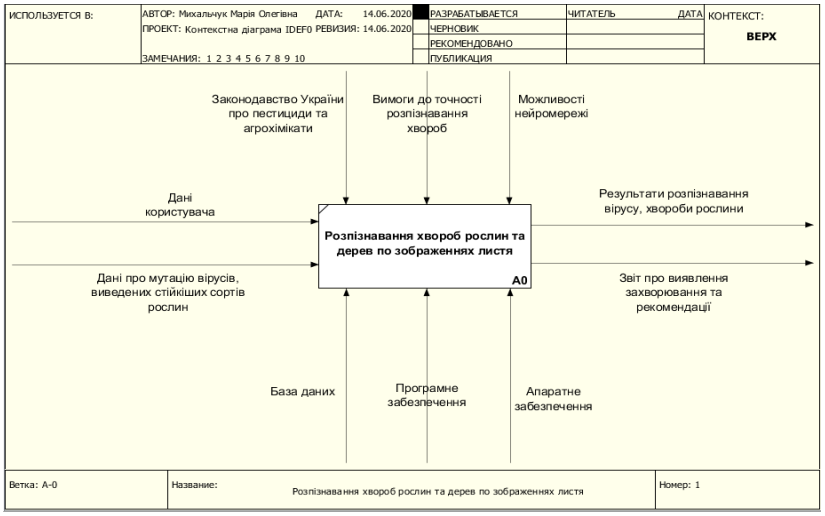


Рисунок 3.1 – Контекстна діаграма IDEF0

Структурована методика аналізу та проектування (SADT) – це методологія розробки систем та інженерії програмного забезпечення для опису об’єктів через функціональну ієрархію. SADT-модель є моделлю структурованого аналізу, яка використовує два типи діаграм: моделі діяльності та моделі даних.

IDEF0 виступає методологією моделювання, сутність якої полягає в описі функцій проектування інформаційних систем, переваг різноманітного програмного забезпечення. Галузь інженерії програмного забезпечення включає IDEF0 до мов моделювання IDEF за ознакою технічної структури та побудови методу структурованого аналізу.

Модель була створена за допомогою програми Ramus Educational 1.1, яка дозволяє створювати різні типи діаграм у форматах IDEF0 та DFD за допомогою утиліти для їх перегляду та редагування. Безкоштовна версія підтримує основні функції, такі як імпорт / експорт у формат IDL, сумісний із стороннім програмним забезпеченням, створення потоків за допомогою стрілок напряму, додавання або видалення елементів, заміна та переміщення їх тощо.

Основний блок діяльності поданий в центральній частині діаграми. Задача контекстної діаграми IDEF0 це відображення процесу розпізнавання хвороб рослин та дерев по зображенню листя. Також показано елементи які впливають на реалізацію даної задачі, а саме забезпечення певні даними, вимогами до точності кінцевого результату, можливостей нейронної мережі та програмно-апаратного забезпечення.

В ролі вхідних даних виступають:

- дані користувача;
- дані про мутацію вірусів, виведення стійкіших сортів рослин;
- набір даних.

До вихідних даних відноситься результати розпізнавання вірусу, хвороби рослини та формування звіту і рекомендацій.

Розпізнавання хвороб дерев та рослин по зображенню листя буде керуватися Законом України про пестициди та агрохімікати і можливостями нейронної мережі зі згорткою для даної системи.

З метою кращого розуміння внутрішніх процесів при розробці програмного додатка необхідно їх деталізувати за допомогою декомпозиції IDEF0 на діаграми

нижчого рівня.

Під час декомпозиції ми виділили декілька основних блоків, які подані на рис. 3.2:

– завантаження мобільного додатку;
– отримання зображення з камери (галереї фотографій);
– розпізнавання захворювання на отриманому зображенні (якщо листя дійсно уражене);
– виведення результату розпізнавання на інтерфейс програмного додатку (звіт).

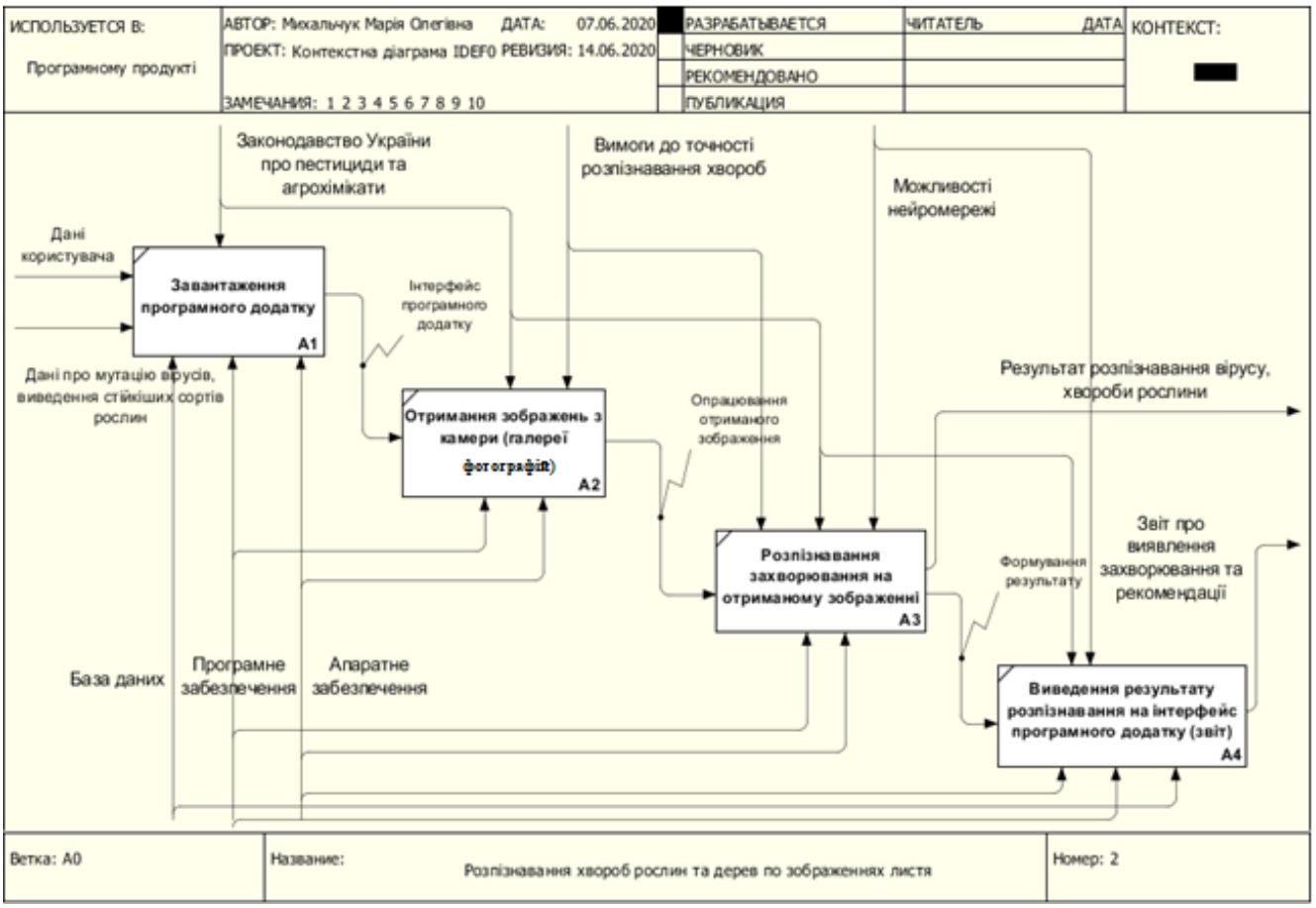


Рисунок 3.2 – Діаграма декомпозиції 1-го рівня у нотації IDEF0

Вимоги з приводу точності розпізнавання хвороб рослин в першу чергу будуть задаватися до зображень які отримані з камери мобільного пристрою користувача або зроблені раніше користувачем та збережені до галереї (потрібно зробити чітке фото листя), а в другу чергу до нейронної мережі зі згорткою.

Для кращого розуміння принципу роботи додатку наведемо UML діаграму послідовності виконання дій програмного продукту та взаємодію користувача, що подана на рис. 3.3.

Спершу потенційному користувачу потрібно встановити програмний додаток з Play Market, авторизуватися та зробити фото ураженого листя рослини, за допомогою камери смартфона або вибрати фото які були створені раніше (завантажити в програмний додаток з галереї зображень). Далі в програмному додатку відбувається обробка вхідного зображення через підготовку (встановлення однакового розміру та контрастності), розпізнавання зображення за допомогою нейронної мережі зі згорткою та порівняння результатів з набором даних, що підвищує точність ідентифікації захворювання рослини. Після опрацювання та обробки зображення в програмному додатку формується звіт про захворювання рослини та рекомендації з приводу лікування.

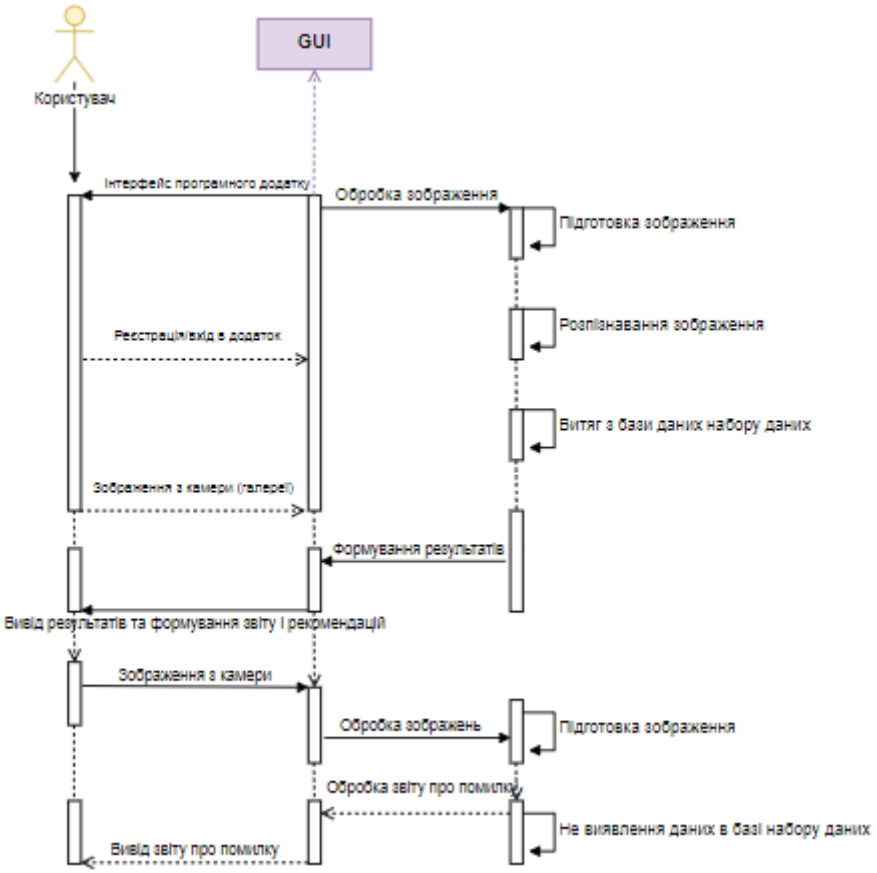


Рисунок 3.3 – Діаграма послідовності виконання дій програмного продукту та взаємодії користувача

Отже, було проведено в достатньому обсязі аналіз SADT та побудовано IDEF0, UML діаграми роботи програмного додатку.

3.2 Моделювання варіантів використання програмного додатку розпізнавання хвороб дерев та рослин по зображенню листя

Щоб краще зрозуміти роботу програмного додатку, передбачити ризики та взаємодію між об’єктами інформаційної системи, необхідно створити діаграму варіантів використання, яка подана на рис. 3.4. Даний вид діаграм відноситься до діаграм прецендентів, які задаються графами з множиною векторів, варіантів використання з обмеженням системи за допомогою границі у вигляді прямокутника. В діаграмі відображається асоціативний зв’язок між прецедентами та акторами.

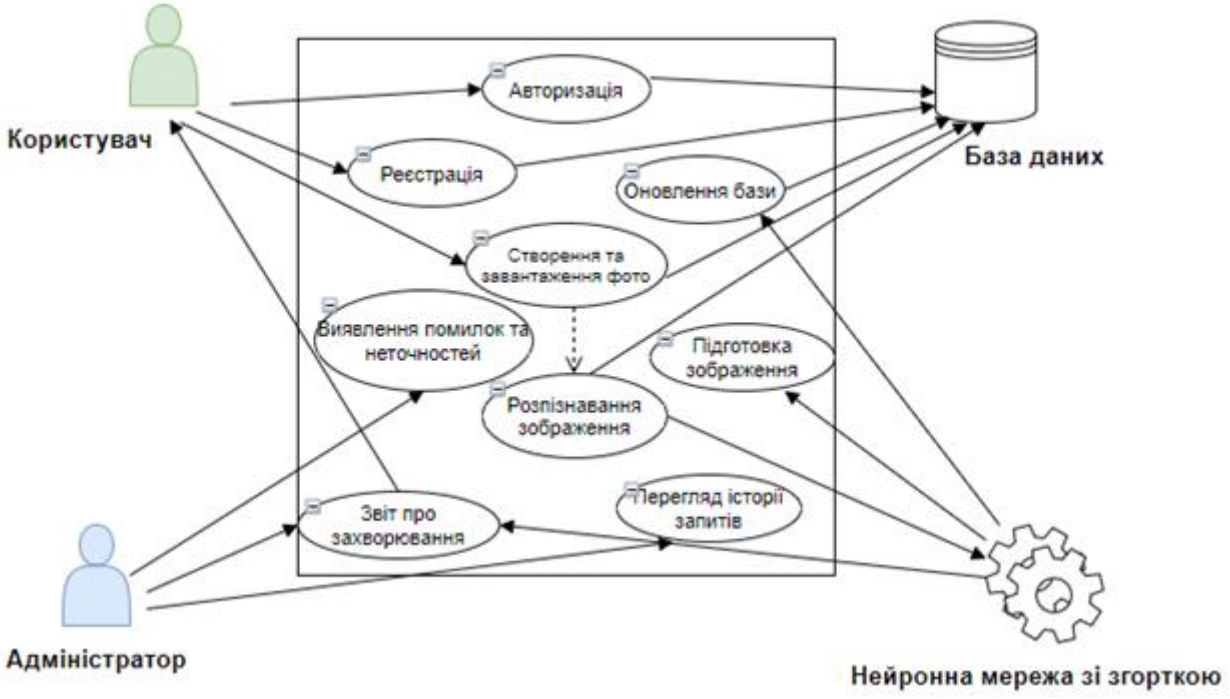


Рисунок 3.4 – Діаграма варіантів використання

Варіанти використання (англ. use case) відображаються через систему проектування. Їх використовують з метою детального опису функцій, які передбачені системою для актора (набір дій при взаємодії з актором).

В нашому випадку ключовими елементами системи виступають “База даних Data Set” та “Нейронна мережа зі згорткою”, які взаємодіють через програмний додаток. При запиті користувача виконується звернення інформаційної системи до цих елементів та очікування відповіді або результатів обробки вхідної інформації і перевірці її допустимості.

База даних зберігає таку інформацію:

- інформація про авторизацію користувачів (логін, пароль);
- інформація звіту, результатів розпізнавання захворювань;
- набір даних по кожному захворюванню рослин та ступеню ураження;
- дані по ефективності використання пестицидів та інших засобів лікування;
- дані про мутації, інформація по виявленню нових захворювань.

Нейронна мережа зі згорткою приймає дані від користувача та бази даних підготовлюючи та оброблюючи інформацію з кінцевою видачею користувачеві результату у вигляді звіту та рекомендацій по лікуванню захворювання.

Користувач та адміністратор виступають акторами даної діаграми. Для роботи з програмним додатком користувачу необхідно спершу встановити програму на мобільний пристрій, а далі авторизуватись. Авторизація буде підтримуватись також через Gmail, GitHub та Instagram. Після авторизації користувач може зробити фото листя рослини із захворюванням або завантажити раніше створені фото з галереї для подальшої обробки зображення за допомогою нейронної мережі зі згорткою. Робота нейронної мережі складається з трьох етапів: підготовка, обробка та видача інформації (в нашому випадку це вхідна інформація від користувача, а також робота з базою даних та оновлення бази даних). Вихідна інформація подається у формі звіту про захворювання в якому зазначається тип захворювання, ступінь ураження та симптоми проходження. Також користувач матиме доступ до рекомендацій щодо лікування, які будуть поділятися на три блоки: біологічні засоби, хімічні засоби, підтримка користувачів-аграріїв.

Адміністратор за бажанням може переглядати історію запитів та результати звітів, а також виправити помилки та неточності (передбачається налаштування точності по нових виведених сортах рослин та ефективності засобів боротьби з захворюванням).

4. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ДОДАТКУ РОЗПІЗНАВАННЯ ХВОРОБ ДЕРЕВ ТА РОСЛИН ПО ЗОБРАЖЕННЮ ЛИСТЯ

4.1 Розробка архітектури програмного додатку

Для створення програмного додатку ми обрали багатошарову архітектуру (Layered (n-tier) architecture) [16]. Дана архітектура є найбільш популярнішою та базово складається з трьох шарів: представлення, логіки та даних. Це означає, що ці різні шари розміщуються на декількох віртуальних машинах або кластерах, забезпечуючи надання відповідей на запити користувача без спільного використання ресурсів. Програмним додатком, розробленими на даній архітектурі, легше управляти та оновлювати. Це тому, що ми працюємо над одним розділом (шаром чи модулем), внесені вами зміни не вплинуть на інші функції. І якщо є проблема, ви можете легко визначити, де вона виникає та швидко виправити помилку.

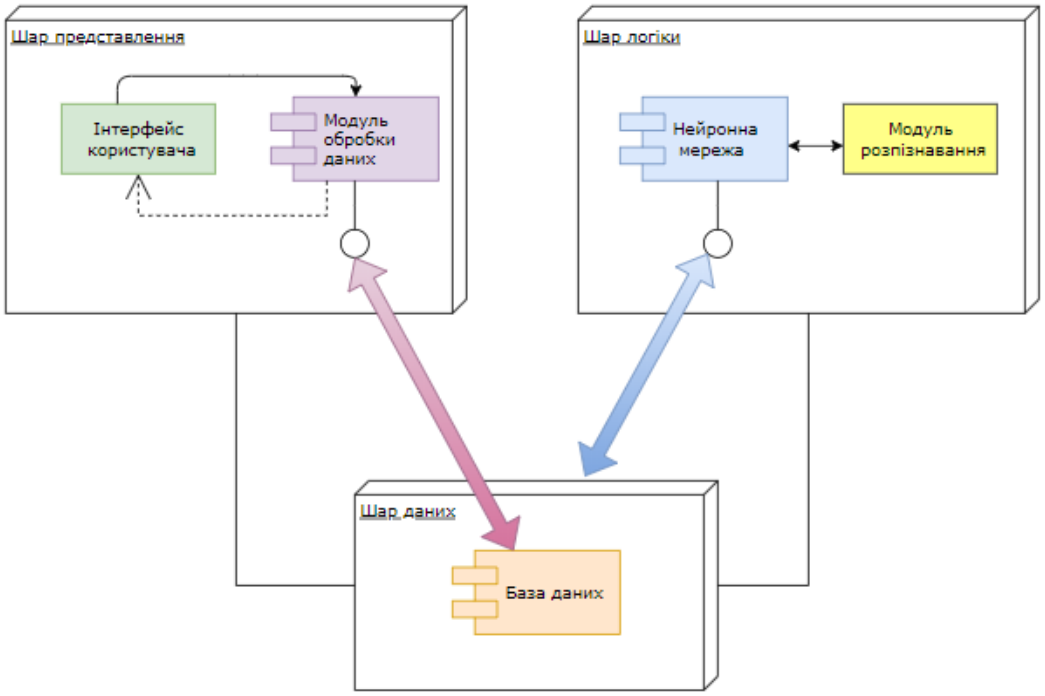


Рисунок 4.1 – Архітектура мобільного додатку

Окреме фізичне розташування цих шарів – це те, що відрізняє n-ярусну архітектуру від структури моделі перегляду контролера, яка лише розділяє рівні презентації, логіки та даних. N-ярусна архітектура також відрізняється

від структури MVC тим, що перша має середній рівень або логічний рівень, що полегшує всі комунікації між різними шарами. Коли ми використовуємо рамку MVC, взаємодія, яка відбувається, є трикутною; замість того, щоб пройти логічний рівень, це керуючий шар, який здійснює доступ до моделі та шари перегляду, тоді як шар моделі отримує доступ до шару представлення. Крім того, керуючий шар створює модель, використовуючи об'єктивні вимоги задані системою, а потім пересилає вихідну інформацію з цієї моделі у шар представлення.

Також багатошарова архітектура забезпечує масштабованість та гнучкість. Якщо нам потрібно додати більше ресурсів (оновити набір даних), ми можемо робити це в конкретному шарі, не впливаючи на інші рівні. Гнучкість проявляється через розширення кожного рівня будь-яким способом, який передбачений для покращення інформаційної системи та в кінцевому результаті покращить точність ідентифікації захворювань рослин нейронною мережею.

4.2 Розробка макету мобільного додатку

Розробка макету програмного додатку здійснюється за допомогою Android Studio 4.0 для Windows 64-bit з використанням мови Java. Для зручності використання додатку нами була розроблена детальна структура, яка складається з таких елементів (рис. 4.2):

- Реєстрація(вхід в програмний додаток);
- Меню;
- Про додаток;
- Дані для аналізу;
- Ідентифікація захворювання;
- Звіт та рекомендації;
- Поширити через соціальні мережі;
- Історія;
- Новини;

- Відгуки;
- Політика використання персональних даних.

Ми придумали назву нашому програмному додатку – Planticus.

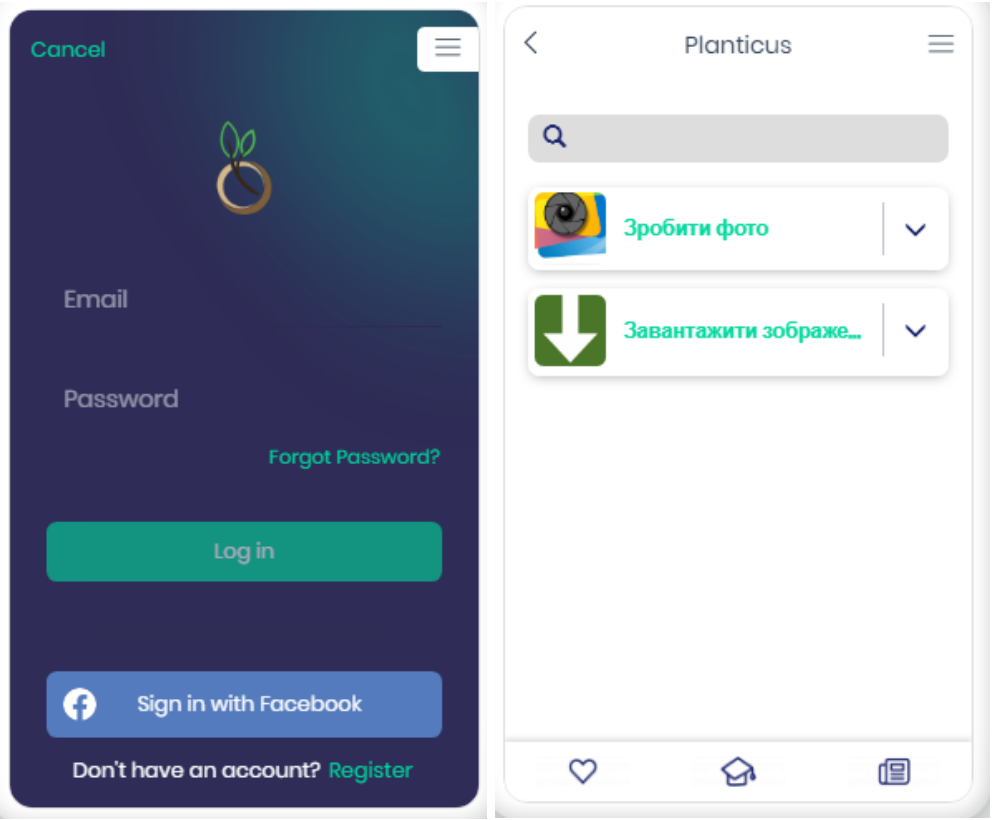


Рисунок 4.2 – Макет інтерфейсу мобільного додатку

4.3 Вибір налаштування, розроблення та навчання нейронної мережі на ресурсі Google Colab, Firebase та ML Kit

За допомогою Google Colab ми змогли імпортувати набір даних, тренувати класифікатор зображень та оцінити вихідну модель. Google Colab виконує код на хмарних серверах Google, тобто ми можемо використовувати потужність обладнання Google, включаючи GPU та TPU, незалежно від потужності нашого компютера чи смартфона.

Firebase слугує платформою для розробки мобільних і веб-додатків. Станом на березень 2020 року платформа Firebase налічує 19 продуктів, які використовують понад 1,5 мільйона додатків. Також на цій платформі є інструменти ML Kit, які виконують функцію підключення нейронної мережі до програмного додатку.

ML Kit є базовим інструментом для удосконалення навчання нейронної мережі зі згорткою. API ML Kit мають різноманітні функції, включаючи оптичне розпізнавання символів, виявлення облич, сканування штрих-кодів, маркування зображень та розпізнавання орієнтирів. Зараз він доступний для розробників iOS або Android. Ми також можемо імпортувати власні моделі TensorFlow Lite, якщо даних API нам буде не достатньо. Також зручність даного API полягає в тому, що його можна використовувати на пристрої (смартфоні) або в хмарних сховищах Google.

Проект доступний для перегляду на ресурсі Google Colab (рис. 4.3)

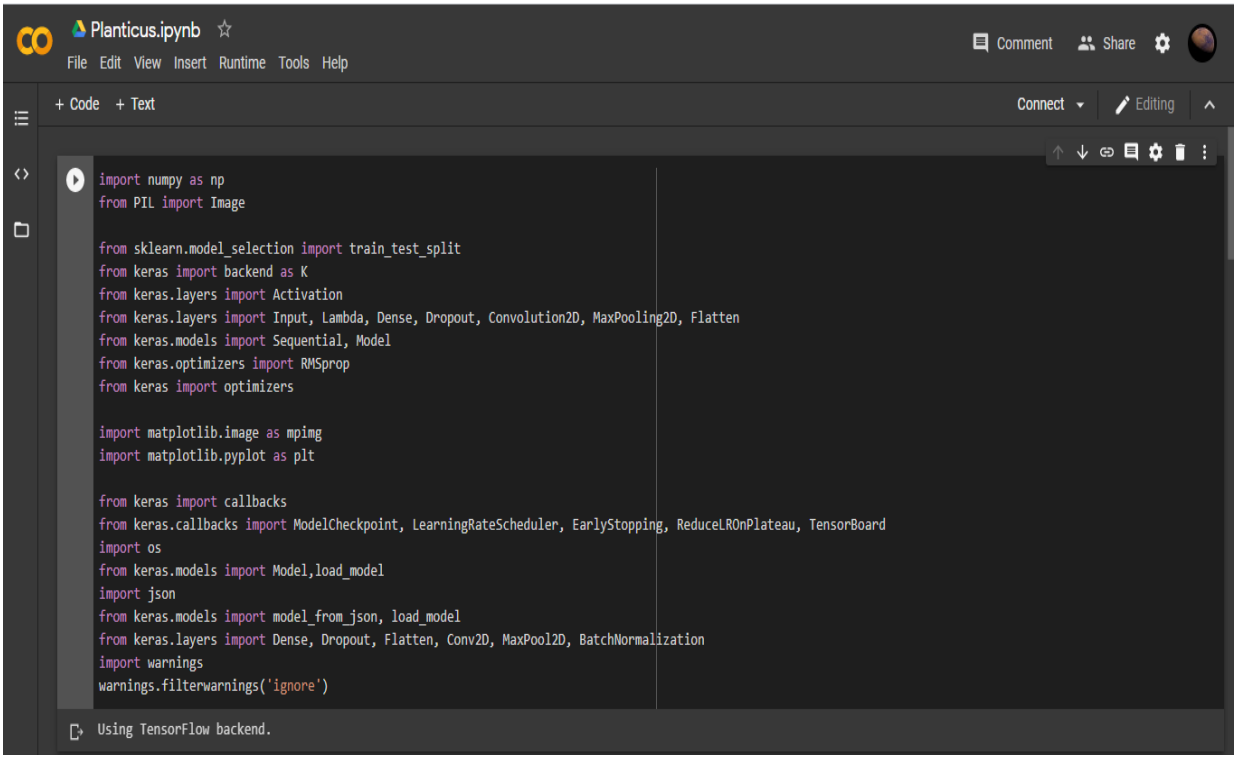


Рисунок 4.3 – Ресурс Google Colab

Створюємо базову модель нейронної мережі на якій будемо проводити експерименти з налаштуваннями:

```
def build_base_network(input_shape):

    seq = Sequential()

    nb_filter = [16, 32, 16]
    kernel_size = 3


    #convolutional layer 1
    seq.add(Convolution2D(nb_filter[0], kernel_size, kernel_size,
```

```
input_shape=input_shape,border_mode='valid', dim_ordering='th'))
    seq.add(Activation('relu'))
    seq.add(MaxPooling2D(pool_size=(2, 2)))
    seq.add(Dropout(.25))

    #convolutional layer 2
    seq.add(Convolution2D(nb_filter[1], kernel_size, kernel_size,
border_mode='valid', dim_ordering='th'))
    seq.add(Activation('relu'))
    seq.add(MaxPooling2D(pool_size=(2, 2), dim_ordering='th'))
    seq.add(Dropout(.25))

    #convolutional layer 2
    seq.add(Convolution2D(nb_filter[2], kernel_size, kernel_size,
border_mode='valid', dim_ordering='th'))
    seq.add(Activation('relu'))
    seq.add(MaxPooling2D(pool_size=(2, 2), dim_ordering='th'))
    seq.add(Dropout(.25))

    #flatten
    seq.add(Flatten())
    seq.add(Dense(128, activation='relu'))
    seq.add(Dropout(0.1))
    seq.add(Dense(50, activation='relu'))
    return seq
```

де, Sequential() – ініціалізація моделі моделі: seq.add – додаємо шари до моделі, MaxPooling2D – шар підвибірки, Convolution2D – згортковий шар,
Dropout – повно зв’язний шар.

Проведемо навчання вибраного налаштування нейронної мережі в 20 епох з набором даних на 13 600 зображень та тестового набору 3 400 зображень:

```
Epoch 1/20
13600/13600 [==============================] - 10s 764us/step - loss: 0.2560 - accuracy: 0.5660 - val_loss: 0.2959 - val_accuracy: 0.5329
Epoch 2/20
13600/13600 [==============================] - 6s 459us/step - loss: 0.2105 - accuracy: 0.6762 - val_loss: 0.2961 - val_accuracy: 0.5235
Epoch 3/20
13600/13600 [==============================] - 6s 453us/step - loss: 0.1790 - accuracy: 0.7382 - val_loss: 0.1709 - val_accuracy: 0.7441
```

```
Epoch 4/20
13600/13600 [==============================] - 6s 467us/step - loss: 0.1570 - accuracy:
0.7854 - val_loss: 0.1511 - val_accuracy: 0.7894
Epoch 5/20
13600/13600 [==============================] - 6s 462us/step - loss: 0.1319 - accuracy:
0.8375 - val_loss: 0.1197 - val_accuracy: 0.8479
Epoch 6/20
13600/13600 [==============================] - 6s 465us/step - loss: 0.1139 - accuracy:
0.8715 - val_loss: 0.1212 - val_accuracy: 0.8403
Epoch 7/20
13600/13600 [==============================] - 6s 468us/step - loss: 0.0971 - accuracy:
0.9030 - val_loss: 0.0970 - val_accuracy: 0.8888
Epoch 8/20
13600/13600 [==============================] - 6s 474us/step - loss: 0.0862 - accuracy:
0.9234 - val_loss: 0.0725 - val_accuracy: 0.9209
Epoch 9/20
13600/13600 [==============================] - 6s 469us/step - loss: 0.0761 - accuracy:
0.9414 - val_loss: 0.0663 - val_accuracy: 0.9282
Epoch 10/20
13600/13600 [==============================] - 6s 466us/step - loss: 0.0680 - accuracy:
0.9510 - val_loss: 0.0603 - val_accuracy: 0.9412
Epoch 11/20
13600/13600 [==============================] - 6s 463us/step - loss: 0.0605 - accuracy:
0.9597 - val_loss: 0.0438 - val_accuracy: 0.9609
Epoch 12/20
13600/13600 [==============================] - 6s 458us/step - loss: 0.0562 - accuracy:
0.9626 - val_loss: 0.0428 - val_accuracy: 0.9650
Epoch 13/20
13600/13600 [==============================] - 6s 465us/step - loss: 0.0517 - accuracy:
0.9699 - val_loss: 0.0388 - val_accuracy: 0.9638
Epoch 14/20
13600/13600 [==============================] - 6s 459us/step - loss: 0.0469 - accuracy:
0.9757 - val_loss: 0.0506 - val_accuracy: 0.9559
Epoch 15/20
13600/13600 [==============================] - 6s 468us/step - loss: 0.0444 - accuracy:
0.9764 - val_loss: 0.0286 - val_accuracy: 0.9768
Epoch 16/20
13600/13600 [==============================] - 6s 463us/step - loss: 0.0413 - accuracy:
0.9800 - val_loss: 0.0336 - val_accuracy: 0.9724
Epoch 17/20
13600/13600 [==============================] - 7s 481us/step - loss: 0.0381 - accuracy:
0.9832 - val_loss: 0.0312 - val_accuracy: 0.9750
Epoch 18/20
13600/13600 [==============================] - 6s 467us/step - loss: 0.0365 - accuracy:
0.9847 - val_loss: 0.0260 - val_accuracy: 0.9815
Epoch 19/20
13600/13600 [==============================] - 6s 466us/step - loss: 0.0349 - accuracy:
0.9827 - val_loss: 0.0333 - val_accuracy: 0.9768
Epoch 20/20
13600/13600 [==============================] - 6s 473us/step - loss: 0.0336 - accuracy:
0.9843 - val_loss: 0.0218 - val_accuracy: 0.9829
saved
```

За цими даними, можна дійти до висновку про те, що найкращі показники на тестовому наборі були отримані за номером конфігурації № 1, 16 – 20. Ми відслідкувати те, що якщо модель вивела високу точність на тренувальному наборі, вона буде найкращою моделлю для підключення в

програмний додаток.. Однак можна побачити декілька непоганих конфігурацій: конфігурації №.14 та 15 отримали 97,57% і 97,64% точності відповідно. Середня точність становить 98,29%.

Це є хороший результат того, що модель наближається до навчальних даних і належним чином ідентифікує зображення з хворобами рослин. Чим більше епох проходить процес тренування нейронної мережі зі згорткою, тим точніші результати і тим краща модель.

Після підключення натренованої нейронної мережі зі згорткою, програмний додаток буде швидко видавати відповіді на запити користувачів з метою виявлення захворювання рослини по її листю з високою точністю (рис. 4.4)

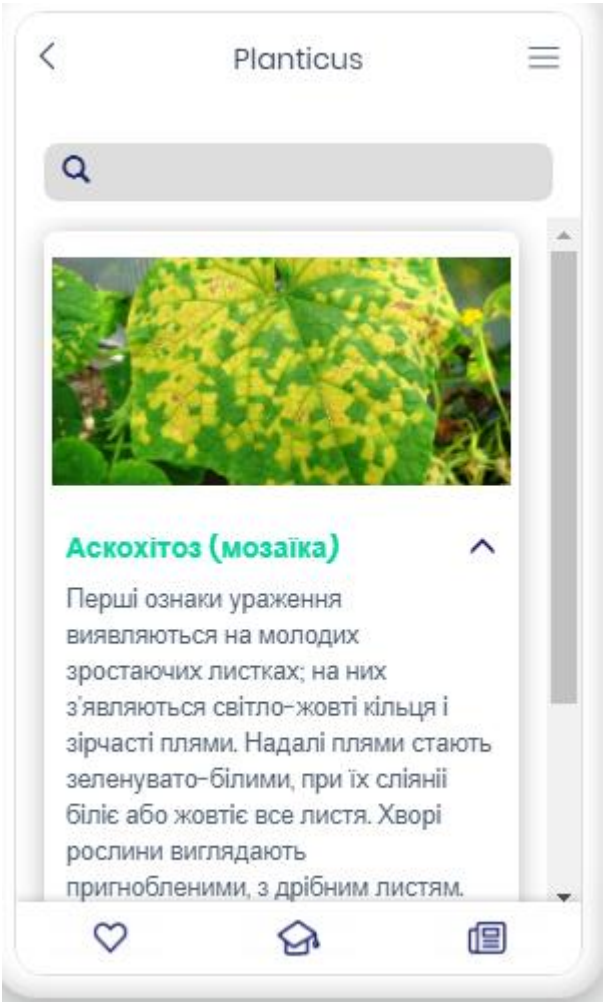


Рисунок 4.4 – Результат виявлення хвороби рослини по ураженому листю

ВИСНОВКИ

Під час виконання кваліфікаційної роботи проведено аналіз проектної області програмного додатку з розпізнавання хвороб рослин та дерев по зображенню листя на базі нейронної мережі зі згорткою. Результатом виконання є реалізація задачі щодо створення власної інформаційної системи ідентифікації захворювань рослин, що забезпечуватиме відображення між входом (наприклад, зображенням листя хворої рослини) на вихід – звіт та рекомендації по засобах та способах боротьби із захворюванням.

Аналітична частина кваліфікаційної роботи розкриває сутність вимог до інформаційної системи та обґрунтування вибору програмного забезпечення для виконання поставлених завдань, методів і алгоритмів на базі яких була розроблена інформаційна система.

Нами було виявлено переваги та недоліки архітектур нейронних мереж зі згорткою, які допоможуть в проектуванні програмного додатка для виявлення захворювань рослин та дерев по зображенню листя.

Вирішено основні задачі:

– пошук та обробка актуальної інформації;

– аналіз схожих програмних додатків з метою виявлення ключових переваг та недоліків;

– вибір, тестування та реалізація програмного додатку для реалізації поставленої мети;

– розробка архітектури програмного додатку, технічного завдання, матриці відповідальності, терміни виконання проекту відображені в діаграмі Ганта, також проаналізовано ризики по виконанню даного проекту та впроваджено заходи їх мінімізації.

У проектній частині роботи для реалізації програмного додатку на базі нейронної мережі зі згорткою були розроблені: модель процесів, діаграма варіантів використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1 Jie Hang, Dexiang Zhang, Peng Chen, Jun Zhang and Bing Wang, [Електронний ресурс]. – Режим доступу: https://www.ncbi.nlm.nih.gov/ pmc/articles/PMC6806268/– Classification of Plant Leaf Diseases Based on Improved Convolutional Neural Network.

2 AlexNet – згорткова нейронна мережа для класифікації зображень [Електронний ресурс]. – Режим доступу: https://neurohive.io/ru/vidy-nejrosetej/alexnet-svjortochnaja-nejronnaja-set-dlja-raspoznavanija-izobrazhenij/

3 Вибір архітектури штучної нейронної мережі на основі порівняння ефективності методів розпізнання зображень [Електронний ресурс]. – Режим доступу: https://cyberleninka.ru/article/n/vybor-arhitektury-iskustvennoy-neyronnoy-seti-na-osnove-sravneniya-effektivnosti-metodov-raspoznavaniya-izobrazheniy

11263-015-0816-y – ImageNet Large Scale Visual Recognition Challenge.

4 Russakovsky O., Deng J., Su H., Krause J., Satheesh S., Ma S., et al. Int. J. Comput. Vis. 115, 211–252. 10.1007/s11263-015-0816-y [Електронний ресурс]. – Режим доступу:https://link.springer.com/article/10.1007%2Fs

11263-015-0816-y – ImageNet Large Scale Visual Recognition Challenge.

5 LeafSnap - Plant Identification [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=plant.

identification.snap&hl=ru

6 Мобільний додаток Intello Labs використовує штучний інтелект для визначення якості харчових продуктів [Електронний ресурс]. – Режим доступу:https://aggeek.net/ru-blog/mobilnij-dodatok-intello-labsvikoristovue

-shtuchnij-intelekt-dlya-viznachennya-yakosti-harchovih-produktiv

7 LeafSnap - Plant Identification [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=plant.identification.snap&hl=ru

8 Face2Gene [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=com.fdna.face2gene&hl=ru

9 Python 3.8.3 [Електронний ресурс]. – Режим доступу: https://programy.com.ua/ua/python/?

10 TensorFlow [Електронний ресурс]. – Режим доступу:

https://www.tensorflow.org/about

11 Бібліотеки для глибокого навчання: Keras [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/ods/blog/325432/

12 Sharada P. Mohanty, David P. Hughes, and Marcel Salathé, [Електронний ресурс]. – Режим доступу: https://www.ncbi.nlm.nih.gov/pmc/articles /PMC5032846/ – 22.09.2016р. – Using Deep Learning for Image-Based Plant Disease Detection.

13 Detect and Track Objects with ML Kit on Android [Електронний ресурс]. – Режим доступу: https://firebase.google.com/docs/ml-kit/android/detect-objects

14 Pang S, Du A, Orgun MA, Yu Z., [Електронний ресурс]. – Режим доступу: https://www.ncbi.nlm.nih.gov/pubmed/30003400 – 12.07.2018р. – A novel fused convolutional neural network for biomedical image classification.

15 Syafiqah Ishaka, Mohd Hafiz Fazalul Rahimana, Siti Nurul, Hashim Saad [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/publication/286763616_Leaf_disease_classification_using_artificial_neural_network – 15.10.2015р. – Leaf disease classification using artificial neural network.

16 N Tier(Multi-Tier), 3-Tier, 2-Tier Architecture with EXAMPLE [Електронний ресурс]. – Режим доступу: https://www.guru99.com/n-tier-architecture-system-concepts-tips.html

17 Молчание вентиляторов. Google Colab, Javascript и TensorflowJS [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/avito/blog/488936/

18 The top 5 software architecture patterns: How to make the right choice [Електронний ресурс]. – Режим доступу: https://techbeacon.com/app-dev-testing/top-5-software-architecture-patterns-how-make-right-choice

19 How Do Convolutional Layers Work in Deep Learning Neural Networks? [Електронний ресурс]. – Режим доступу: https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/

20 PlantVillage-Dataset [Електронний ресурс]. – Режим доступу: https://github.com/spMohanty/PlantVillageDataset/blob/master/README.md

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку програмного додатку «Розпізнавання хвороб рослин та дерев по зображенню листя»

ДОДАТОК А

1 Призначення й мета створення інформаційної системи

1.1 Призначення інформаційної системи

Інформаційна система призначена для того, щоб користувачі нейронної мережі (за допомогою мобільного додатку на базі якого буде створена нейронна мережа зі згорткою) могли використовувати багатошарову модель нейронної мережі для реалізації ідентифікації хвороб листя шляхом обчислення значень хроматичності, зміни забарвлення та форми листя аналізуючи використання методів розпізнавання підтримуючої векторної машини (SVM), дискримінантного методу аналізу для виявлення уражень з врахуванням особливостей структури листя, а також аналіз екологічних факторів впливу.міг піднести себе різній аудиторії людей, сформувати свій імідж, представити його у вигідному ракурсі перед потенційними клієнтами..

1.2 Мета створення інформаційної системи

Метою інформаційної системи дипломного проекту є розроблення інформаційної системи, що буде використовуватися для демонстрації та підвищення точності результатів нейронної мережі зі згорткою для виявлення захворювань рослин за ступенем ураження їх листя.

2 Вимоги до інформаційної системи в цілому

2.1 Вимоги до структури й функціонування інформаційної системи.

Інформаційна система повинна бути реалізована у вигляді мобільного додатка, доступного в мережі Інтернет через GooglePlay. Мобільний додаток повинен складатися із взаємозалежних розділів із чітко розділеними функціями.

2.2 Вимоги до персоналу

Для підтримки мобільного додатка (desktop application) й експлуатації алгоритму нейронної мережі зі згорткою від персоналу не повинно вимагатися

спеціальних технічних навичок, знання технологій або програмних продуктів, за винятком загальних навичок роботи з смартфоном та комп'ютером, стандартними операційними системами такими, як Android, iOS, Windows, а також Google Fuchsia та веб-браузером (наприклад, MS Internet Explorer 8.0 або вище).

2.3 Вимоги до стилістичного оформлення мобільного додатку

Додаток повинен бути розроблений з використанням мови Python і C++, а також містити елементи SQL, обов'язково має мати добре розроблений інтерфейс та алгоритм навчання нейронної мережі зі згорткою за допомогою Tensorflow. Дизайн повинен бути витриманий в діловому форматі і в строгих тонах.

3 Основні вимоги

3.1 Структура інформаційної системи

Інформаційна система повинна складатися з наступних розділів:

– сторінка **Identify plants diseaces** – містить кнопки для завантаження фото рослин з галереї (на телефоні), папки, та доступу до камери (для миттєвої ідентифікації) тощо;

– сторінка **Instant identify Plants** – містить функцію доступу до вибору відповідної категорії, тобто Leaf, Flower, Bark, Fruit тощо.

– сторінка **Get results** – містить подібні фотографії (з ураженням та без ураження) рослини, загальний короткий опис за ознаками Genus, Family та двома описами Description (перший опис буде надавати коротку інформацію про особливості та різновиди даного виду, а в другому – опис стадії та особливостей захворювання рослини;

– сторінка **Build you Plant diseace Collection** – містить збережену та оброблену інформацію , яку можна з часом моніторити для покращення боротьби з захворюваннями рослин.

Кожна директорія конфігурації нейронної мережі зі згорткою може містити

наступні файли та папки:

- caffe.log : в навчальному етапі експерименту
- deploy.prototxt : Caffe implemen.prototxt в конкретному експерименті, який можна використовувати для прогнозів
- hdf5_dumps: тримає скидання hdf5 на всі ітерації на етапі тестування. Через розмір окремих файлів ці папки не будуть додаватися до цієї роботи, але можуть бути надані за запитом.
- labels.txt: зберігає відображення міток класу для всіх класів. Номер рядка (починаючи з 0) відноситься до внутрішнього значення мітки класу caffe.
- slurm-*.out : ииходи, визначені для кластеру, можуть ігноруватися, і в майбутньому вони будуть видалені з бази.
- solver.prototxt : Конфігурація рішення для кафе для конкретного експерименту
- test_logs : вміщує окремі журнали кави на етапі тестування для всіх знімків, створених під час навчальної фази
- test_prototxts : Зберігає конфігурації прототексту для фази тестування, що відповідає всім знімкам, згенерованим під час фази тренувань
- test.sh : Сценарій, щоб ініціювати тестування моделей на всіх знімках
- train.sh : Сценарій для початку навчання моделей за допомогою конкретної конфігурації експерименту
- train_val.prototxt : конфігурації кавових поїздів
- results : Папка, що містить результати та аналіз фази тестування. Структура папки з результатами така:
    - classification_reports: Утримує звіти про класифікацію для всіх знімків, створених під час навчального етапу
    - confusion_matrices: Тримає матриці плутанини для всіх знімків, створених під час тренувального етапу
    - evaluation_graphs: Зберігає звіт про класифікацію склеарну для всіх

знімків, створених на етапі тренувань

- generate_graphs.py: Сценарій, який використовується для генерування всіх графіків та звітів для зазначеного досвіду
- parse_log.py: Аналізує журнали Caffe, щоб отримати та зберігати як CSV значення ітерації, точності та втрат на етапі тренувань.
- generate_results.sh: Простий скрипт для обгортки над parsed_log.pyта generate_graphs.py.
- parsed_caffe_output: Папка, що містить проаналізовані журнали кави, і створений журнал оцінки.

3.2 Навігація

Користувацький інтерфейс сайту повинен забезпечувати наочне, інтуїтивно зрозуміле представлення структури розміщеної на ньому інформації, швидкий і логічний перехід до розділів і сторінок. Наявність навігації на всіх сторінках сайту. Правильна структура інформації дозволяє користувачам без побоювання продовжувати дослідження сайту, залишаючись впевненими в тому, що вони завжди зможуть без зусиль повернутися до раніше переглянутих сторінок.

3.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

– Веб-браузер: Internet Explorer 7.0 і вище, або Firefox 7.5 і вище, або Opera 9.5 і вище, або Safari 6.1 і вище, або Chrome 7 і вище;

– Включена підтримка javascript і Flash.

3.4 Функціональні вимоги

– перегляд портфоліо;

– перегляд відео контенту;

– перегляд соціальних мереж фотографу;

– можливість розповсюджувати (ділитися) фото в соціальні мережі;

– зворотній зв'язок;

– замовлення фотосесії.

3.5 Наповнення програмного додатку (контент)

Первинна розробка та верстка контенту (інформаційного вмісту) програмного додатку повинна проводитися силами Виконавця за безпосередньої участі Замовника. Замовник надає всі необхідні Виконавцю текстові та графічні матеріали, а також коментарі щодо їх змісту, обсягу, оформлення і розміщення. Додаток буде підтримувати дві мови: англійську та українську. Повинен мати зручну навігацію.

ДОДАТОК Б

Планування робіт

1 Ідентифікація ідеї проекту

Метою інформаційної системи дипломного проекту є допомога у сприянні продемонструвати творчі здібності (фото та відео роботи) фотографа.

Дипломний проект призначений для того, щоб фотограф міг піднести себе різній аудиторії людей, сформувати свій імідж, представити його у вигідному ракурсі перед потенційними клієнтами.

Інформаційна система повинна бути реалізована у вигляді сайту, доступного в мережі Інтернет. Сайт повинен складатися із взаємозалежних розділів із чітко розділеними функціями.

2 Деталізація мети методом SMART

Конкретна (Specific). Створити програмний продукт для захвату більшої аудиторії кліентів.

Вимірювана (Measurable). Використовуючи мінімум ресурсів розробити якісний програмний продукт.

Досяжна (Achievable). Поставлена мета випливає у результаті актуальних проблем, які наявні у сфері фотомистецтва.

Реалістична (Relevant). У наявності є всі необхідні технічні та програмні засоби. Розробники достатньо кваліфіковані для виконання поставлених задач.

Обмежена у часі (Time—framed). Ціль має часове обмеження. Терміни досягнення мети проекту визначаються за домовленість замовником та виконавцем.

3 Описання фази розробки IT—проекту

3.1 Планування змісту структури робіт IT—проекту (WBS)

Ієрархічна структура робіт (Work Breakdown Structure) є інструментом, який дозволяє розділити проект на частини. Вона встановлює ієрархічно структурований поділ праці з реалізації проекту для всіх залучених до нього працівників.

Під час побудови WBS відбувається послідовне розбиття проекту на підпроекти, роботи різних рівнів, детальні робочі пакети. Розподіл - це розподіл результатів проекту на менші, простіші для керування компоненти пакетів. Пакети робіт зазвичай відповідають найнижчому рівню деталізації та складаються з окремих робіт. Декомпозиція повинна бути коректною, тобто елементи будь-якого рівня WBS повинні бути необхідними та достатніми для створення відповідного елемента верхнього рівня.

Ієрархічна структура робіт в основному являє собою перелік завдань проекту. Вона може бути представлена графічно або у формі опису, який показує включення робіт. Ієрархічна структура робіт організовує і визначає весь зміст проекту. Роботи, не включені в WBS, не є роботами проекту.

Виконаємо побудову WBS структури, у якій зазначимо всі виконувані роботи в залежності від головних етапів:

1. Формування технічного завдання — розробка технічного завдання, що встановлює основне призначення, показники якості, техніко-економічні та спеціальні вимоги до розроблюваного інструментального засобу. Формування технічного завдання включає в себе підпункти (визначення предметної області, призначення програмного продукту, визначення мови написання. визначення цільової аудиторії, визначення вимог до програмного продукту).

2. Розробка програмного продукту - написання відповідних модулів, що забезпечують функціонування програмного продукту.2.1 Математична (формальна) постановка проблеми;

2.1. Розробка інтерфейсу програмного продукту.

2.2. Розробка стилю засобами CSS.

2.3. Наповнення контентом.

2.4. Програмування програмного продукту мовами HTML та PHP.

2.5. Розробка бази даних програмного продукту.

3. Тестування – перевірка роботи програмного продукту, виявлення помилок.

3.1. Тестування взаємодії.

3.2. Тестування навантаження.

4. Здача програмного продукту.

4.1. Документація.

4.2. Програмний продукт.

WBS-структура для даного проекту представлена на рисунку Б.1

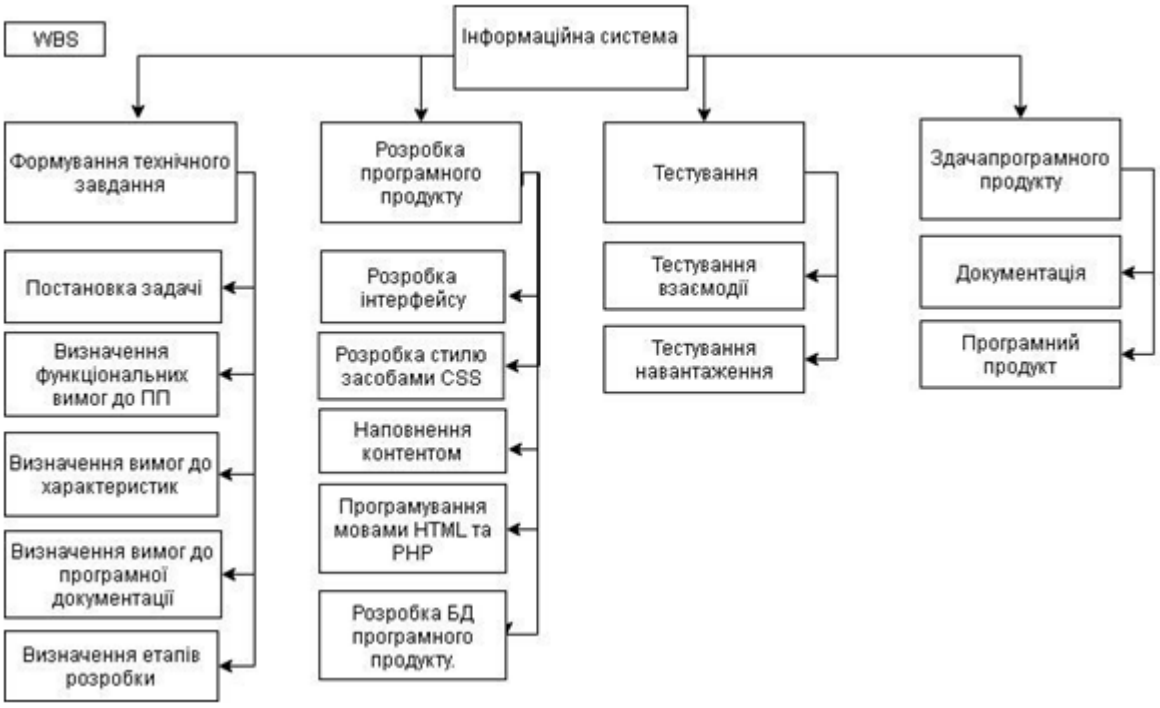


Рисунок Б.1 - WBS-структура інформаційної системи програмного додатку Plunticus

3.2 Планування структури організації, для впровадження готового проекту (OBS)

OBS-структура проекту – організаційна структура виконавців (організацій) проекту. Визначається за переліком пакетів робіт нижнього рівня кожної гілки WBS—структури. Представляється відповідальними (відповідальні – це не обов'язково керівники організацій (відділів), а ті люди які безпосередньо організують виконання робіт) за виконання пакетів робіт.

Організаційна структура представляє собою графічне відображення учасників проекту та їх відповідальних осіб, які задіяні в реалізації проекту. На верхньому рівні OBS розташована команда проекту.

На наступному рівні фіксуються виконавці: організації, відділи тощо. Потім, рівнем нижче, для кожного виконавця вказують прізвища конкретних осіб, які будуть відповідати за виконання елементарних робіт WBS. Потрібно пам’ятати, що відповідальні – це не обов’язково керівники, а ті співробітники, які безпосередньо організовують і відповідають у виконавця за виконання елементарної роботи, зазначеної у WBS. Для них ця елементарна робота також є проектом (у порівнянні з загальним проектом). Для себе вони також можуть побудувати WBS— структуру й застосовувати інші інструменти планування.

3.3 Побудова матриці відповідальності (виконавців пакетів робіт)

На підставі OBS та WBS структур було побудовано матрицю відповідальності. Для кожного з виконавців була визначена його роль:

– відповідальний (В) – повністю відповідає за виконання задачі та має право приймати рішення щодо способу її реалізації;

– консультант (К) – наглядає за ходом виконання завдання і висловлює свої міркування стосовно способу та якості реалізації. Несе відповідальність, якщо не помітить явного недоліку.

– спостерігач (С) – те ж саме що і консультант, але відповідальності не несе.

Матриця відповідальності представлена в табл. Б.1

Таблиця Б.1 – Матриця відповідальності

WBS\OBS		Парфененко	Рецензент
1Формування технічного завдання	В	К	
1.1 Постановка задачі	В	К	
1.2 Визначення функціональних вимог до ПП	В	К	
1.3 Визначення вимог до характеристик	В	К	
1.4 Визначення вимог до програмної документації	В	К	
1.5 Визначення етапів розробки	В	К	
2 Розробка ПП	В	С	
2.1 Розробка інтерфейсу ПП	В	С	
2.2. Розробка стилю засобами CSS	В	С	
2.3. Наповнення контентом.	В	С	
2.4. Програмування програмного продукту мовами Python та Java	В	С	
2.5. Розробка бази даних програмного продукту.	В	С	
3 Тестування	В	К	С
3.1 Тестування взаємодії.	В	К	С
3.2 Тестування навантаження.	В	К	С

Продовження таблиці Б.1– Матриця відповідальності

WBS\OBS		Парфененко	Рецензент
4 Здача програмного продукту	В	К	
4.1 Документація	В	К	
4.2 Програмний продукт	В	К	

4 Побудова календарного графіку виконання ІТ—проекту (включаючи побудову часткових мережевих моделей у вигляді діаграм Ганта)

Діаграма Ганта – горизонтальна лінійна діаграма, на якій задачі проекту представляються протяжними в часі відрізками, що характеризуються датами початку та закінчення, затримками і, можливо, іншими тимчасовими параметрами. Для отримання реального уявлення про тривалість виконання робіт з урахуванням обмеженості у використанні ресурсів, на підставі часткової мережевої моделі, а також, проекту в цілому з урахуванням вихідних та святкових днів, було побудовано календарний графік робіт. Цей графік представлено за допомогою програмного засобу MS Project.

Графік виконання дипломного проекту представлено у вигляді Діаграми Ганта на рисунку Б.3.

				Квітень							Травень							Червень						
Назва задачі	Тривалість	Початок	Кінець	Пн	Вт	Ср	Чт	Пт	Сб	Нд	Пн	Вт	Ср	Чт	Пт	Сб	Нд	Пн	Вт	Ср	Чт	Пт	Сб	Нд
1. Формування технічного завдання	35	02.04.2020	11.05.2020																					
1.1 Постановка задачі	8	02.04.2020	10.04.2020																					
1.2 Визначення функціональних вимог до ПП	8	11.04.2020	19.04.2020																					
1.3 Визначення вимог до характеристик	8	20.04.2020	28.04.2020																					
1.4 Визначення вимог до програмної документації	2	29.04.2020	30.04.2020																					
1.5 Визначення етапів розробки	9	01.05.2020	10.05.2020																					
2. Розробка ПП	22	11.05.2020	03.06.2020																					
2.1 Розробка інтерфейсу ПП	2	11.05.2020	12.05.2020																					
2.2. Розробка стилю засобами CSS	2	13.05.2020	15.05.2020																					
2.3. Наповнення контентом.	3	16.05.2020	18.05.2020																					
2.4. Програмування програмного продукту мовами Python та Java	10	19.05.2020	28.05.2020																					
2.5. Розробка бази даних програмного продукту.	5	29.05.2020	03.06.2020																					
3. Тестування	4	03.06.2020	06.06.2020																					
3.1 Тестування взаємодії.	2	03.06.2020	04.06.2020																					
3.2 Тестування навантаження.	2	05.06.2020	06.06.2020																					
4. Здача програмного продукту	5	07.06.2020	11.06.2020																					
4.1 Документація	2	07.06.2020	08.06.2020																					
4.2 Програмний продукт	3	09.06.2020	11.06.2020																					

Рисунок Б.3 - Діаграма Ганта

5 Ідентифікація ризиків

Ідентифікація ризиків – це виявлення ризиків, здатних вплинути на проект, і документальне оформлення їх характеристик. Це ітеративний процес, який періодично повторюється на всьому протязі проекту, оскільки в рамках його життєвого циклу можуть виявлятися нові ризики.

Якісний аналіз передує кількісному. Він передбачає визначення факторів ризику, ідентифікацію потенційних областей виникнення ризику, виявлення напрямків діяльності та етапів, на яких може реалізуватися ризик. Протягом якісного аналізу також встановлюється можливість кількісної оцінки ризиків, реалізація яких може вплинути на діяльність підприємства.

Кількісна оцінка ризиків часто супроводжує якісну оцінку і також вимагає процес ідентифікації ризиків. Кількісна і якісна оцінка ризиків можуть використовуватися окремо або разом, залежно від наявного часу і бюджету, необхідності в кількісній або якісній оцінці ризиків.

Планування реагування на ризики – це процес розробки шляхів і визначення дій із збільшення можливостей і зниження погроз для цілей проекту. Даний процес зачинається після проведення якісного і кількісного аналізу ризиків. На рисунку Б.5 розташовується класифікація ризиків.

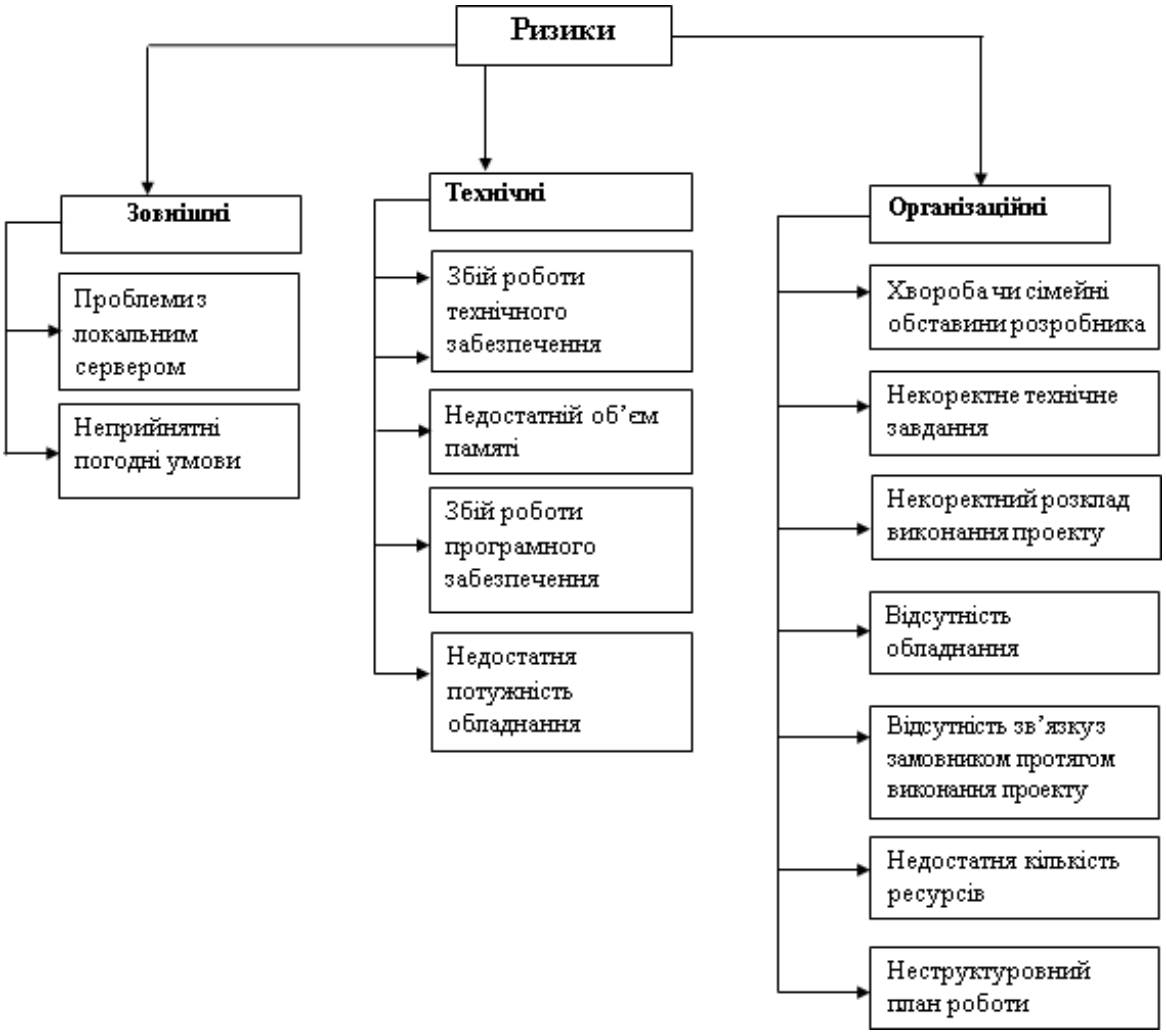


Рисунок Б.5 – Ризики

6.1 Матриця ризиків

Ризики представити за допомогою RBM матриці (Risk Breakdown Matrix) на рис. Б.6.

Класифікація ризиків за імовірністю виникнення:

– слабоімовірнісні;

– малоімовірнісні;

– імовірні;

– досить імовірні;

– майже імовірні.

Класифікація ризиків за імовірністю виникнення за величиною втрат:

– мінімальна;

– низька

– середня

– висока

– максимальна

Виконаємо класифікацію ризиків даного проекту. Для цього складемо табл. Б.2.

Таблиця Б.2 – Класифікація ризиків дипломного проекту

Ризик		ймовірність виникнення	величина втрат
Проблеми з локальним сервером	R1	3	5
Неприйнятні погодні умови	R2	4	2
Збій роботи технічного забезпечення	R3	3	2
Збій роботи програмного забезпечення	R4	2	5
Недостатня потужність обладнання	R5	2	2
Недостатній об’єм памяті	R6	2	2
Хвороба чи сімейні обставини розробника	R7	1	3
Некоректне технічне завдання	R8	3	3
Некоректний розклад виконання проекту	R9	3	3
Недостатня кількість ресурсів	R10	2	4
Неструктуровний план роботи	R11	1	3
Відсутність зв’язку з замовником	R12	3	4
Відсутність обладнання	R13	1	5

Рисунок Б.6 – Матриця імовірності втрат

6.2 Класифікація за ступенем впливу та за рівнем ризику (табл. Б.3)

Класифікація за ступенем впливу:

– ігноровані (1≤R≤4);

– незначні (5≤R≤8);

– помірні (9≤R≤11);

– вагомі (12≤R≤19);

– критичні (20≤R≤25).

Класифікація за рівнем ризику:

– прийнятні ризики;

– виправданні ризики;

– недопустимі ризики;

Таблиця Б.3 – Класифікація за ступенем впливу та за рівнем ризику

Ризик		Ступінь впливу	Рівень ризику
Проблеми з локальним сервером	R1	15	недопустимі ризики
Неприйнятні погодні умови	R2	8	виправданні ризики
Збій роботи технічного забезпечення	R3	6	виправданні ризики

Продовження таблиці Б.3 - Класифікація за ступенем впливу та за рівнем ризику

Збій роботи програмного забезпечення	R4	10	недопустимі ризики
Недостатня потужність обладнання	R5	4	прийнятні ризики
Недостатній об’єм памяті	R6	4	прийнятні ризики
Хвороба чи сімейні обставини розробника	R7	3	прийнятні ризики
Некоректне технічне завдання	R8	9	виправданні ризики
Некоректний розклад виконання проекту	R9	9	виправданні ризики
Недостатня кількість ресурсів	R10	8	виправданні ризики
Неструктуровний план роботи	R11	3	прийнятні ризики
Відсутність зв’язку з замовником	R12	12	виправданні ризики
Відсутність обладнання	R13	10	недопустимі ризики

План по усуненню ризиків:

– вибір потужного обладнання для виконання проекту;

– зіставлення структурованого плану роботи;

– періодичні поставки тестових версій сайту замовнику;

– безперервна взаємодія з замовником;

– враховувати досвід проектів-аналогів;

– резервувати час на випадок помилок планування та виникнення непередбачених обставин;

– організувати зустрічі та переговори для вирішення проблем, що виникають;

– використовувати програми страхування технічних ризиків;