

Одеський національний політехнічний університет  
Інститут комп'ютерних систем  
Кафедра інформаційних систем

**ПОЯСНЮВАЛЬНА ЗАПИСКА**  
до кваліфікаційної роботи  
магістра

на тему «Розробка та дослідження методики нейромережової класифікації листів рослин для використання в мобільних пристроях»

---

---

Виконав: студент У курсу, групи AI –131  
спеціальності 122 – «Комп’ютерні науки»  
спеціалізації – «Інформаційні управлюючі  
системи та технології»

Іщенко А.Д.

(прізвище та ініціали)

Керівник доц. Годовиченко М.А.

(прізвище та ініціали)

Одеса – 2018 року

## ЗМІСТ

Зміст .....	4
Перелік умовних позначень, символів, одиниць, скорочень і термінів.....	8
Вступ.....	9
1 Аналіз стану прикладної області дослідження .....	11
1.1 Аналіз предметного середовища .....	11
1.2 Актуальність дослідження.....	13
1.3 Аналіз існуючих методів та проблем.....	15
1.3.1 Класифікація по найближчому середнього значення.....	16
1.3.2 Класифікація по відстані до найближчого сусіда.....	18
1.3.3 Структурний метод класифікації .....	19
1.3.4 Байєсівський підхід до прийняття рішення.....	19
1.3.5 Нейронні мережі.....	20
1.4 Виділення значущих ознак на зображенні.....	21
1.4.1 Оператор кенні .....	22
1.4.2 Оператор прюіт .....	23
1.4.3 Оператор собеля .....	25
1.5 Метод laplacian of gaussian (log) .....	26
1.6 Висновки до первого розділу .....	26
2 Розробка методики класифікації листів рослин на зображенні.....	28
2.1 Виділення значущих ознак на зображенні .....	28
2.1.1 Використання оператору прюіт для виділення контурів .....	29
2.1.2 Аналіз виділених контурів та формування значущих ознак.....	30
2.2 Опис структури нейромережі в мобільному застосуванні.....	33
2.2.1 Опис реалізації алгоритму зворотного поширення помилки .....	39
2.3 Висновки до другого розділу .....	43

Змін.	Арк.	№ докум.	Підпис	Дата	ІС КРМ 122 036 ПЗ		
Розробник		Іщенко А.Д.			Розробка та дослідження	Літ.	Лист
Перев.		Годовиченко М.			методики нейромережової		Листів
Реценз.					класифікації листів рослин для	5	120
Н. Контр.		Ядрова М.В.			використання в мобільних	ОНПУ, ІКС, каф. ІС.	
Затвердж.		Арсірій О.О.			пристроях		

3 Розробка мобільного застосування для класифікації листів рослин на зображенні .....	45
3.1 Користувацький інтерфейс мобільного застосування.....	45
3.2 Загальна архітектура застосування.....	49
3.3 Представлення шарів.....	55
3.4 Інтеграційний процес .....	56
3.5 Управління програмним кодом IC.....	58
3.6 Розрахунок метрик програмного коду IC.....	59
3.7 Контрольний список по якості реалізації IC.....	59
3.8 Документація IC.....	60
3.9 Забезпечення якості IC .....	60
3.10 Функціональне тестування.....	61
3.11 Модульне тестування .....	63
3.12 Тестування навантаження .....	63
3.13 Висновки до третього розділу .....	64
4 Дослідження методики нейромережової класифікації листів рослин.....	66
4.1 Вибір критеріїв оцінки .....	66
4.2 Опис набору даних.....	67
4.3 Тестування розробленого програмного застосування на реальній тестовій виборці .....	74
4.4 Висновки до четвертого розділу.....	76
5 Охорона праці та безпека у надзвичайних ситуаціях .....	77
5.1 Організація та управління охороною праці у фірми розробника програмного забезпечення у мобільних пристроях .....	77
5.2 Обґрунтування заходів з покращення умов праці.....	78
5.3 Індивідуальне завдання. Розрахунок захисного заземлення .....	81
5.4 Надзвичайні ситуації та шляхи їх запобігання .....	84

Змін	Лист	№ докум.	Підпис	Дата

5.5 Індивідуальне завдання. Безпека в надзвичайної ситуації радиоактивного забруднення.....	85
5.6 Заходи з охорони праці та безпеки у надзвичайних ситуаціях.....	86
5.7 Висновки до п'ятого розділу .....	87
Висновки .....	89
Перелік посилань.....	91
Додаток А.....	93
Додаток Б.....	112
Додаток В .....	119
Додаток Г.....	120

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

# ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ІС – інформаційна система

ДСТУ – державні стандарти України

ДНАОП – державні нормативні акти про охорону праці

ДСН – державні санітарні норми

ССБТ – системи стандартів безпеки праці

ІДЕ – інтегроване середовище розробки

GUI (graphical user interface) – графічний користувальницький інтерфейс

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

ІС КРМ 122 036 ПЗ

Лист

8

## ВСТУП

Проблема швидкої класифікації рослин відома здавна та повністю не вирішена до сьогоднішнього часу внаслідок великої кількості різноманітних рослин, складності та схожості листків та індивідуальними особливостями виду.

Задача розробка системи автоматичної класифікації рослин має великий потенціал внаслідок своєї практичної актуальності та можливості застосування її на багатьох сільськогосподарських та фармацевтичних підприємствах. Не менш важливим є спостереження за популяціями рідких видів рослин у ботанічних садах тощо.

Поточні реалізації даної технології дозволяють вирішувати завдання виявлення і класифікації листів дерев, витративши певний час, але подібні технології використовують або ресурсномісткі алгоритми, які вимагають дорогої обладнання, яке не може використовуватися в польових умовах, або взагалі не застосовують алгоритми розпізнавання, вимагаючи від користувача ввести опис рослини.

У даній роботі пропонується методика класифікації рослин, в основі який лежить виділення форми листа та її нейромережева класифікація. Ця методика позитивно зарекомендувала себе з точки зору швидкості роботи та оптимальності використання обчислювальних ресурсів, тому була реалізована у вигляді мобільного застосування для приладів під управлінням системи Android.

Метою та завданням роботи є розробка методики нейромережової класифікації листів на фотографії із застосуванням сучасних методів обробки і аналізу зображень для прискорення і здешевлення роботи технологічного процесу.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

- проаналізувати проблеми при розпізнаванні листів на зображеннях;
- провести аналітичний огляд методів і моделей класифікації листів;
- провести аналітичний огляд методів для аналізу зображення і нейромережової класифікації;

Змін	Лист	№ докум.	Підпис	Дата

- розробити архітектуру системи з використанням методів аналізу зображення і нейромережової класифікації;
- реалізувати систему класифікації за розробленою архітектурі;
- отримати і проаналізувати швидкість роботи і вартість розробленої системи.

Об'єкт дослідження – процес класифікації об'єктів на зображені за допомогою нейронних мереж.

Предмет дослідження – методика використання сучасних методів обробки зображень для поліпшення точності роботи нейронної мережі розпізнавання зображень.

Науково-практична значимість роботи – розроблена методика та Android застосування, що дозволяє виконувати класифікацію листів рослин за невеликий проміжок часу та отримувати придатний результат.

Робота складається з чотирьох розділів. В першому розділі проводиться аналіз предметної області та існуючих систем автоматизованої класифікації листів. В другому розділі розробляється методика класифікації листів. В третьому розділі розробляється Android застосування для класифікації листів. В четвертому розділі робиться тестування проекту.

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

# 1 АНАЛІЗ СТАНУ ПРИКЛАДНОЇ ОБЛАСТІ ДОСЛІДЖЕННЯ

## 1.1 Аналіз предметного середовища

За оцінками, в світі налічується майже півмільйона видів рослин. Класифікація видів історично проблематична і часто призводить до дублювання інформації.

Типовий листок складається з листкової пластинки (лат. *lamina*), черешка (лат. *petiolus*) та прилистків (лат. *stipulae*). Проте наявність двох останніх частин зовсім необов'язкова. Листки без черешка називаються сидячими. Отже, листкова пластинка — основна частина листка. (рис 1.1) [14]

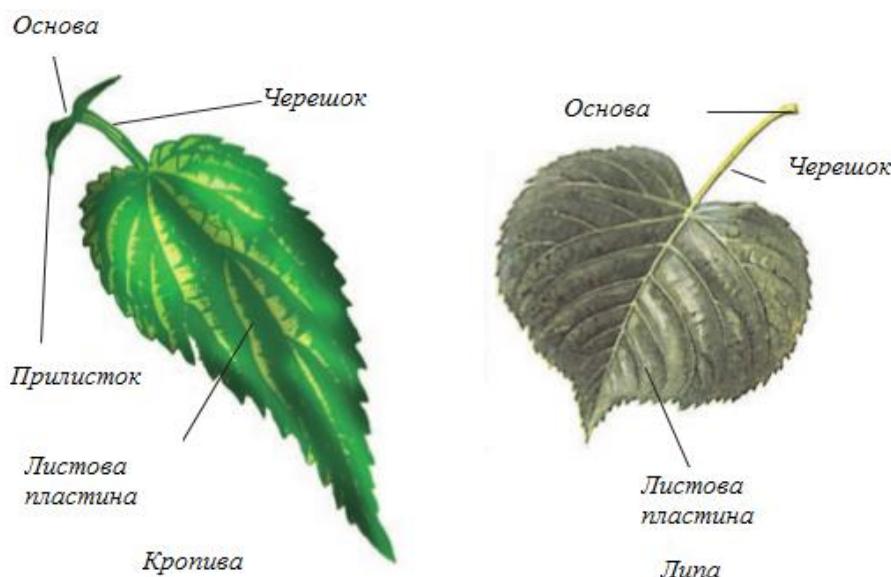


Рисунок 1.1 – Зовнішня будова листка

Розрізняють прості та складні листки. Простий листок складається з однієї листкової пластинки й одного черешка та опадає цілком (дуб, береза, клен) (рис. 1.2) [15]. Складний листок складається з декількох простих, розміщених на спільному (іноді гілчастому) черешку, листкових пластинок (каштан, акація) (рис. 1.2). Під час листопаду опадає окремими частинами.

Змін	Лист	№ докум.	Підпис	Дата

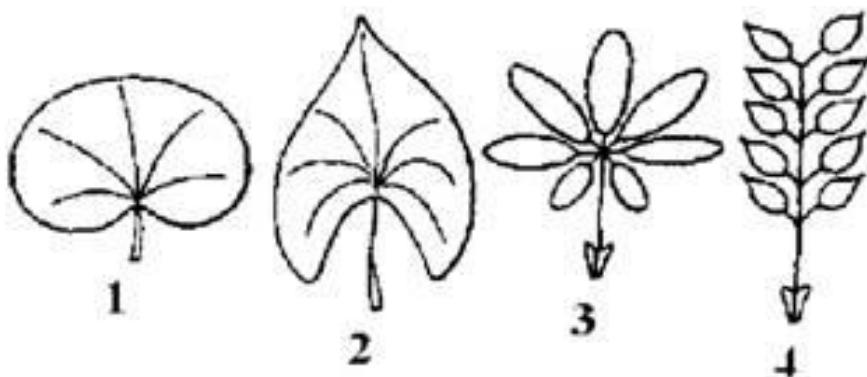


Рисунок 1.2 – Прості та складні листки (за Яковлевим, Челомбітко):

1, 2 – простий листок; 3, 4 – складний листок.

Складні листки класифікують за:

- будовою листкових пластинок;
- кількістю листкових пластинок та взаємним їх розташуванням – трійчастоскладні (суниця, конюшина, малина), пальчастоскладні (каштан, люпин), парноперистоскладні – з парної кількості листкових пластинок (акація жовта, горох), непарноперистоскладні – з непарної кількості листкових пластинок (шипшина, троянда, горобина).

Листкові пластинки у різних напрямках пронизані жилками – системою судинно-волокнистих пучків, що надають листковій міцності та поєднують у єдине ціле мезофіл листка. Порядок розташування жилок у листковій пластинці називається жилкуванням листка.

Розрізняють такі типи жилкування (рис. 1.3):

- просте – листкову пластинку пронизує лише один провідний пучок;
- центральна жилка (мохоподібні, плауноподібні, деякі хвойні);
- дихотомічне – кожна з жилок галузиться на дві бічні рівноцінні (гінго дволопатевий);
- сітчасте – від однієї або кількох великих жилок відгалужуються бічні, більш тонкі, які при подальшому галуженні утворюють густу сітку дрібних жилок (переважно у дводольних);
- дугове – в листок входить одна жилка, бічні жилки відходять від головної

Змін	Лист	№ докум.	Підпис	Дата

і продовжуються дугоподібно, не розгалужуючись (конвалія);

– паралельне – листкову пластинку від основи до верхівки пронизує декілька однакових паралельних нерозгалужених жилок (злакові, осокові).

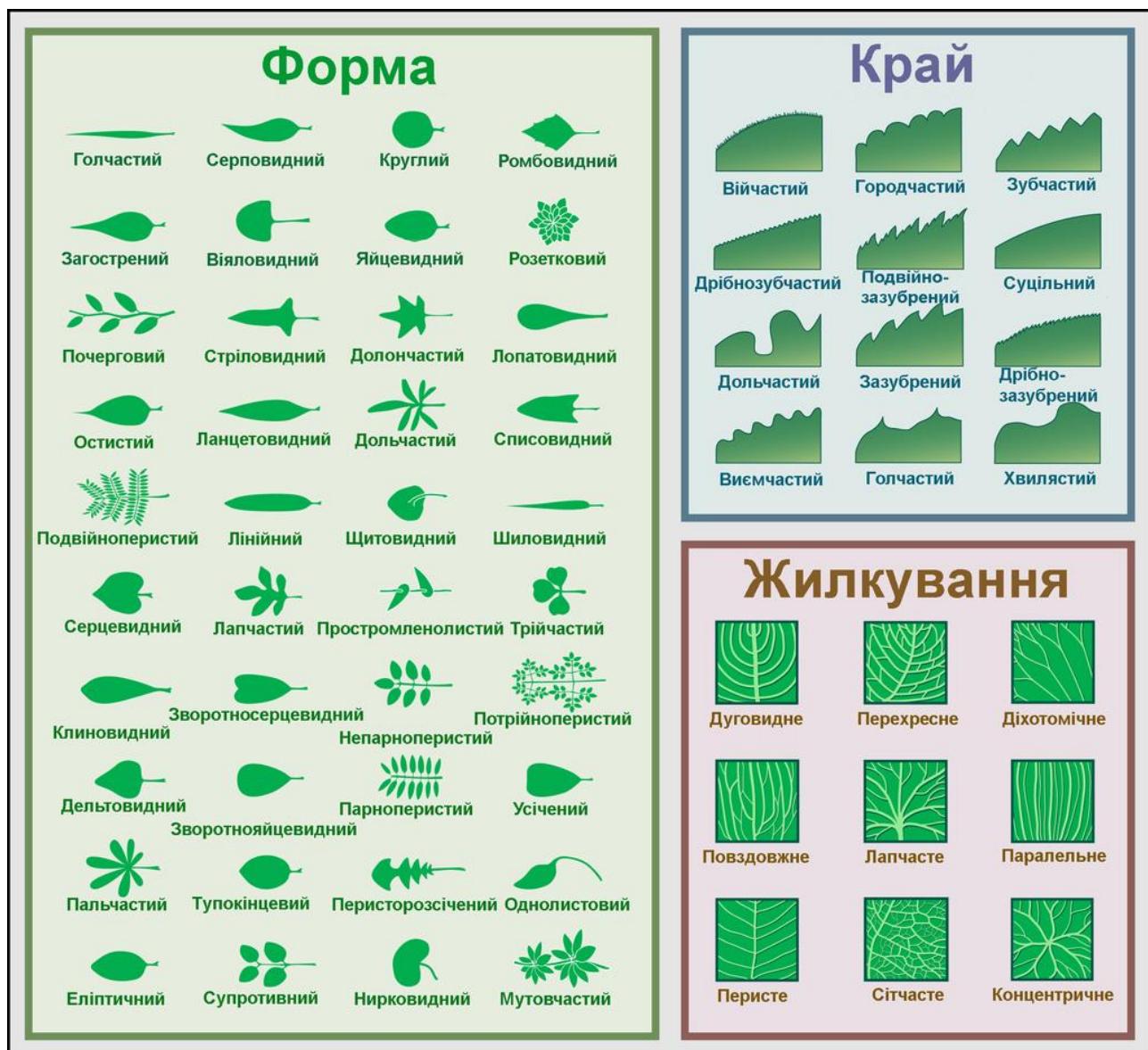


Рисунок 1.3 – Форми листкової пластинки, край та жилкування листків

## 1.2 Актуальність дослідження

Автоматизація розпізнавання рослин може мати безліч практичний застосувань. Деякі з них наведені нижче:

Спостереження і збереження популяцій видів.

Змін	Лист	№ докум.	Підпис	Дата

Зміна навколошнього середовища під впливом людини, трансформація місцеперебувань, руйнування квазінатурального рослинного покриву призводять до роздроблення і зменшення чисельності популяцій рослин, вимирання окремих видів, загальному збіденню флори, непоправної втрати генетичних ресурсів рослинного світу.

Найбільш уразливими елементами регіональних флор зазвичай виявляються ендемічні, реліктові, а також деякі корисні рослини (декоративні, лікарські, харчові).

Для того щоб своєчасно вжити заходів з порятунку рідкісних і зникаючих рослин, необхідно знати стан їх популяцій. У ряді контрольних пунктів повинні бути організовані спостереження за популяціями найцікавіших і важливих в науковому і практичному відношенні видів рослин. При цьому необхідно враховувати щільність, чисельність популяцій, просторову і вікову структуру, їх динаміку, реакцію на антропогенний вплив. Особливо цінним показником служить вікова структура популяцій: якщо вона набуває регресивний характер, це вже серйозний сигнал тривоги.

Фармацевтичні дослідження на основі рослин.

Перспективність досліджень лікарських рослин для застосування в сучасній фармації безсумнівна. При введенні таких рослин в фармацевтичну і медичну практику в першу чергу слід проводити цілий комплекс досліджень, що встановлюють їх видову приналежність.

Застосування в сфері продовольчої і сільськогосподарської промисловості.

Сьогодні аграрії все частіше звертаються до сучасних технологій, щоб збільшити виробництво в умовах обмежених ресурсів і підвищити ефективність землеробства. Так, наприклад, автоматизація розпізнавання дозволить прискорити відбір корисних сировинних рослин і як наслідок поліпшити якість виробленого продукту.

У всіх сферах використовується виділення об'єктів (листів рослин) на зображені і віднесення цих об'єктів до певного класу. Поточні реалізації даної технології дозволяють вирішувати завдання виявлення і класифікації листів дерев,

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

витративши певний час, але подібні технології використовують або ресурсномісткі алгоритми, які вимагають дорогого обладнання, яке не може використовуватися в польових умовах, або взагалі не застосовують алгоритми розпізнавання, вимагаючи від користувача ввести опис рослини. У зв'язку з цим, актуальним завданням буде використання менш витратних алгоритмів, що видають прийнятний результат.

### 1.3 Аналіз існуючих методів та проблем

При класифікації листів використовуються різні складні методи, при використанні яких сильно збільшується час класифікації. У той час як багато досліджень в цій області показують те, що цей показник можливо поліпшити. Тому виберемо в якості системного показника його.

Деякі з аналогів, такі, як прибор «Листомір» використовують згорткову нейронну мережу для класифікації зображень листів рослин, але таке рішення вимогливо до ресурсів и потребує значних затрат на навчання та тестування мережі.

В умовах обмежених ресурсів найбільш ефективно зарекомендували себе методи, основані на виділенні значимих ознак та здійснення на їх основі класифікації. У поточних реалізаціях таких методів найчастіше значущі ознаки вводяться вручну спеціалістом, але це займає багато часу та велика ймовірність помилки. Після виділення значущих ознак, вони передаються на вхід класифікатору, який відносить лист рослини до одного з класів.

Розділяють 3 групи методів розпізнавання листів:

- Порівняння з зразком. До цієї групи належать структурні методи і методи, які використовують наближення і відстань (класифікації по найближчому середньому і по відстані до найближчого сусіда.)
- Статистичні методи. Прикладом цієї групи служить байесовський метод прийняття рішення. Статистичні методи засновані на обчисленні ймовірності.
- Нейронні мережі. Окремий клас методів розпізнавання. На відміну від

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

інших методів, нейронні мережі здатні навчатися вже в процесі розпізнавання і володіють хорошим потенціалом розвитку. Далі розглянемо різні методи, що відносяться до різних груп.

У разі перших двох груп методів ознак виділяються вручну, що займає велику кількість часу. Після виділення ознак лист може бути класифікований одним з методів перших двох груп. Перевагою ручного виділення ознак є можливість введення додаткових ознак, таких як розмір самої рослини, а не тільки листа. Група методів з нейронними мережами дозволяє не виділяти значимі ознаки на зображені в деяких випадках, наприклад, при використанні складних надточних нейронних мереж, однак, без тривалого навчання на правильний навчальний вибірці такі мережі дають велику помилку та потребують багато ресурсів.

### 1.3.1 Класифікація по найближчому середнього значення

У класичному підході до систем розпізнавання вектор ознак, що характеризує кожен клас, виходить в наслідок навчання системи і відомий заздалегідь або на основі будь-яких моделей передбачається в режимі реального часу. Один з найпростіших алгоритмів класифікації використовує вектор математичного очікування класу (середнє значення). –  $j$ -й еталонний ознака класу  $i$ , – кількість еталонних векторів класу  $i$ . Отже, невідомий об'єкт буде ставитися до класу  $i$ , якщо він істотно близче до вектору математичного очікування класу  $i$ , ніж до векторів математичних очікувань інших класів (рис 1.4). Такий метод можна застосовувати, коли точки ознак розташовані дуже купчасто і далеко від точок інших класів.

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------



Рисунок 1.4 – Класифікація по найближчому середнього значення

Нижче наведено приклад ситуації, коли даний метод не буде працювати. Як видно, клас 2 розділений на два непересічних безлічі ознак, а клас 3 занадто витягнутий, що призводить до ситуації, коли його віддалені точки ближче до середнього значення іншого класу, ніж до його власного середнього значення.

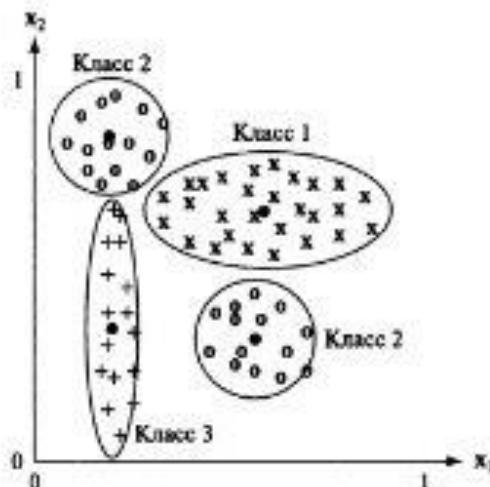


Рисунок 1.5 – Недоліки методу класифікації по найближчому середнього значення

Описана проблема в деяких випадках може бути вирішена зміною розрахунку відстані. Будемо враховувати характеристику «розкиду» значень класу, уздовж кожного координатного напрямку  $i$ . Середньоквадратичне відхилення дорівнює квадратному кореню з дисперсії. Евклідова відстань між вектором  $x$  і

Змін	Лист	№ докум.	Підпис	Дата

вектором математичного очікування визначається за формулою 1.1.

$$\|x - x_c\| = \sqrt{\sum_{t=1,d} \left( \frac{x[i] - x_c[i]}{\sigma_i} \right)^2} \quad (1.1)$$

Ця формула відстані дозволяє зменшити кількість помилок класифікації, але на ділі більшість завдань не вдається уявити таким простим класом.

### 1.3.2 Класифікація по відстані до найближчого сусіда

Цей метод відносить невідомий вектор ознак до класу, окрім зразки якого знаходяться ближче всіх. Такі зразки називаються найближчими сусідами. При класифікації за найближчому сусіду не потрібно знати моделей розподілу класів в просторі, необхідна тільки інформація про еталонних зразках. Принцип роботи алгоритму побудований на визначені мінімальної відстані до зразка ознаки з бази даних. Так само рішення можна поліпшити, якщо шукати серед сусідів. Для класифікації кожного з об'єктів тестової вибірки необхідно послідовно виконати наступні операції:

- обчислити відстань до кожного з об'єктів навчальної вибірки;
- відібрати k об'єктів навчальної вибірки, відстань до яких мінімально;
- клас об'єкта – це клас, який найчастіше трапляється серед k найближчих сусідів.

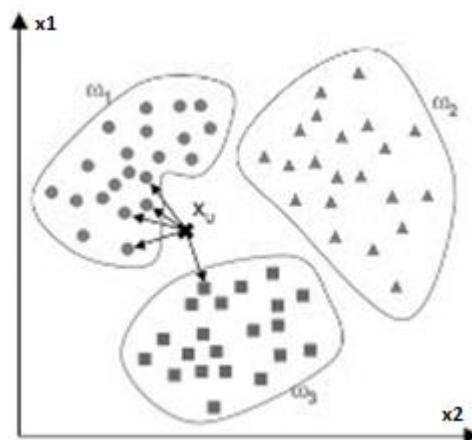


Рисунок 1.6 – Класифікація по відстані до найближчого сусіда

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

Переваги такого підходу очевидні:

- в будь-який момент можна додати нові зразки в базу даних;
- деревовидні і сіткові структури даних дозволяють скоротити кількість обчислюваних відстаней.

Даний алгоритм – один з найпростіших алгоритмів класифікації, тому на реальних завданнях він часто виявляється неефективним. Крім точності класифікації, проблемою цього класифікатора є швидкість класифікації: якщо в навчальній вибірці  $N$  об'єктів, в тестовому виборі  $M$  об'єктів, а розмірність простору –  $K$ , то кількість операцій для класифікації тестової вибірки може бути оцінений як  $O(K \times M \times N)$ .

### 1.3.3 Структурний метод класифікації

Такий метод розпізнавання застосовується для об'єктів, які можна структурно розділити на складові. Так само важливо, щоб при розпізнаванні система знайшла такі ознаки, які точно дозволяють сказати, що об'єкт належить до цього класу і до ніякому іншому.

Як приклад використання, можна привести задачу розпізнавання нарисної символів або фігур.

Друга назва цього методу – синтаксичний, так як він має на увазі використання мови опису образів, який структурно описує кожен елемент і підементи, структурно розділяючи образ на підобрази. Метод буде корисний для розпізнавання складних образів, що складаються з багатьох образів нижчого, простого рівня.

### 1.3.4 Байесівський підхід до прийняття рішення

Даний метод заснований на теоремі Байеса і визначені апріорних ймовірностей, тобто ймовірність результатів або належність об'єкта певного класу змінюється після отримання нових експертних оцінок (підтвердження наявності

Змін	Лист	№ докум.	Підпис	Дата

новых ознак).

Поява того чи іншого способу є випадковою подією і ймовірність цієї події можна описати за допомогою закону розподілу ймовірностей багатовимірної випадкової величини  $\xi$  в тій чи іншій формі. Знаючи елементи навчальної вибірки можна відновити імовірнісні характеристики цього середовища.

Байєсівський класифікатор на основі спостережуваних ознак відносить об'єкт до класу, до якого цей об'єкт належить з найбільшою ймовірністю. Як уже зазначалося, в основі методу лежить теорема Байеса.

Однак, наше рішення залежить від тих параметрів, які ми знаємо. Наприклад, ми можемо знати тільки значення априорної ймовірності, а інші значення оцінити неможливо. Недолік даного методу полягає в тому, що в окремому випадку ми представляємо об'єкт парою чисел – колір і розмір аркуша, але в більш складних завданнях розмірність ознак може бути в рази вище і для оцінки ймовірності багатовимірної випадкової величини може не вистачити числа спостережень зі списку з історичними даними.

### 1.3.5 Нейронні мережі

Нейронні мережі дозволяють вирішувати широке коло завдань і вдають із себе структуру з декількох шарів – штучних нейронів (обчислювальних елементів) і зв'язків між ними. Структура імітує структуру і властивості організації нервової системи живих організмів. Нейромережа отримує на вході набір сигналів і на виході видає відповідний відповідь (вихідні сигнали), які описують рішення деякої задачі. (рис 1.7)

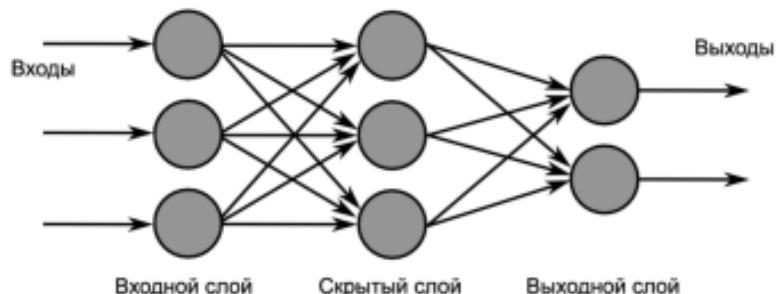


Рисунок 1.7 – Схема нейромережі

Змін	Лист	№ докум.	Підпис

Щоб описати принцип роботи мережі, уявімо штучний нейрон (рис 1.8)

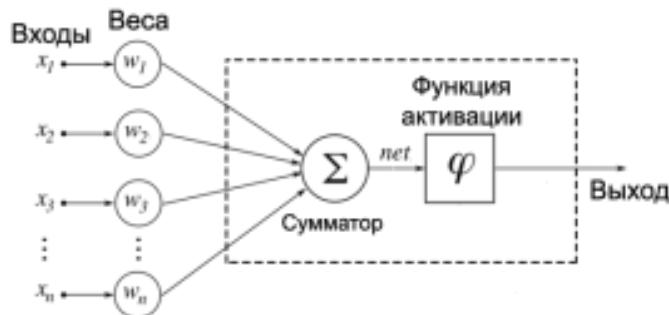


Рисунок 1.8 – Штучний нейрон

Відповідно на кожен нейрон входного шару надходить сигнал, який множиться на відповідний йому вагу. Функція активації є або порогове значення, передає на вихід одиничний сигнал, або сигмоїдальною функцією, яка перетворює значення суми всіх, хто прийде сигналів, в число, що знаходиться в діапазоні від 0 до 1. Таким чином на виході мережі виходить розподіл усіх значення, схоже на результат методу Байеса. У порівнянні з лінійними методами статистики, нейромережі дозволяють ефективно будувати нелінійні залежності, точніше описують набори даних. Що ж стосується байєсівського класифікатора, що буде квадратично розділяє поверхню, нейронна мережа може побудувати поверхню більш високого порядку. Висока нелінійність розділяє поверхні найвного байєсівського класифікатора (він не використовує коваріаційні матриці класів, як класичний Байес, а аналізує локальні щільності ймовірності) вимагає значного сумарного числа прикладів для можливості оцінювання ймовірностей при кожному поєднанні інтервалів значень змінних – тоді як нейросеть навчається на всій вибірці даних, не фрагментуючи її, що підвищує адекватність налаштування мережі.

#### 1.4 Виділення значущих ознак на зображені

Від виділення значущих ознак залежить якість роботи класифікатора. Значимі ознаки можуть бути різні, такі ознаки як колір, форма, кількість і форма прожилок і т.д. Край листа, так само як і його загальна форма, підказує ботаніків

Змін	Лист	№ докум.	Підпис	Дата

приналежність рослини до того, чи іншого виду. Для аналізу форми листа на зображення в першу чергу потрібно виділити його контури. Для цих цілей використовуються методи виділення контурів на зображення.

#### 1.4.1 Оператор Кенні

Оператор Кенні – оператор виявлення кордонів зображення. Був розроблений в 1986 році Джоном Кенні (англ. John F. Canny) і використовує багатоступінчастий алгоритм для виявлення широкого спектра кордонів в зображеннях.



Рисунок 1.9 – Виділення контурів листа за допомогою оператору Кенні

Кенні вивчив математичну проблему отримання фільтра, оптимального за критеріями виділення, локалізації та мінімізації декількох відгуків одного краю. Він показав, що шуканий фільтр є сумою чотирьох експонент. Він також показав, що цей фільтр може бути добре наблизений першої похідної Гауссіана. Кенні ввів поняття придушення немаксімумов (англ. Non-Maximum Suppression), яке означає, що пікселями кордонів оголошуються пікселі, в яких досягається локальний максимум градієнта в напрямку вектора градієнта.

Перед початком роботи з алгоритмом Кенні необхідно перетворити зображення в градації (відтінки) сірого, для зменшення обчислювальних витрат.

Першим кроком алгоритму є згладжування зображення для видалення шуму. Зазвичай для цієї мети застосовується фільтр Гаусса, який використовує функцію Гаусса (1.2) в якості імпульсної перехідної функції.

Змін	Лист	№ докум.	Підпис	Дата

$$G_\sigma = \frac{1}{\sqrt{(2\pi\sigma^2)}} \exp\left(-\left(\frac{m^2 + n^2}{2\sigma^2}\right)\right) \quad (1.2)$$

Таким чином, якщо вихідне зображення задано яскравістю  $f(m, n)$ , то розмиття по Гауса буде вважатися за формулою 1.3

$$g(m, n) = G_\sigma(m * n) * f(m, n) \quad (1.3)$$

Потім обчислюється градієнт функції  $g(m, n)$ , максимальне значення якого визначає положення кордонів.

$$M(m, n) = \sqrt{(g_m^2(m, n) + g_n^2(m, n))} \quad (1.4)$$

$$\theta(m, n) = \tan^{-1} \left[ \frac{g_n(m, n)}{g_m(m, n)} \right] \quad (1.5)$$

Третім кроком алгоритму є придушення немаксимумов, що дозволяє зробити кордону тонше і точніше. проводиться порівняння значення градієнта кожного пікселя з двома сусідніми в напрямку вектора градієнта. Якщо таке значення виявляється більше сусідніх, то піксель належить кордоні, інакше його значення градієнта встановлюється рівним нулю.

Подвійна порогова фільтрація дозволяє визначити знаходження кордону в конкретній точці зображення. Оператор Кенні використовує два пороги фільтрації. Піксель зі значенням градієнта більше верхньої кордони приймає максимальне значення (межа вважається достовірною), зі значенням нижче нижньої межі – пригнічується. Точки зі значеннями, потрапляють в діапазон між порогами, приймають фіксоване середнє значення.

Останнім етапом є трасування області неоднозначності. Пікселі, які отримали на попередньому етапі середнє значення, придушуються або відносяться до групи граничних. Критерієм такого поділу служить наявність зіткнення пікселя з певною групою по одному з восьми напрямків.

#### 1.4.2 Оператор Прюітт

Оператор Прюітт (англ. Prewitt operator), в дисципліні комп'ютерного зору – метод виділення кордонів в обробці зображень, який обчислює максимальний

Змін	Лист	№ докум.	Підпис	Дата

відгук на безлічі ядер згортки для знаходження локальної орієнтації кордону в кожному пікселі. Він був створений доктором Джудіт Прюйтт (Judith Prewitt) для виявлення меж медичних зображень.



Рисунок 1.10 – Виділення контурів листа за допомогою оператору Прюйтта

Суть цього методу полягає в обчисленні максимального відгуку на безлічі ядер згортки для знаходження локальної орієнтації кордону в кожному пікселі. Оператор використовує два ядра  $3 \times 3$ , згортуючи вихідне зображення для обчислення наближених значень похідних: одне по горизонталі і одне по вертикалі. Покладемо  $A$  вихідним зображенням, і  $G_x, G_y$  – двома зображеннями, в яких кожна точка містить горизонтальне і вертикальне наближення похідної, яка розраховується за формулами (1.6) і (1.7).

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * A \quad (1.6)$$

$$G_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} * A \quad (1.7)$$

$G_x$  і  $G_y$  – дві матриці, де кожна точка містить наближені похідні по  $x$  і по  $y$ . Вони обчислюються наступним чином шляхом множення матриці  $G_x$  і  $G_y$  і підсумовуванням обох матриць, в результаті отриманий результат записується в поточні координати  $x$  і  $y$  в нове зображення (1.8):

$$G = \sqrt{G_x^2 + G_y^2} \quad (1.8)$$

Змін	Лист	№ докум.	Підпис	Дата

### 1.4.3 Оператор Собеля

Оператор Собеля – дискретний диференціальний оператор, який обчислює наближене значення градієнта яскравості зображення. Результатом застосування оператора Собеля в кожній точці зображення є або вектор градієнта яскравості в цій точці, або його норма. Використовується в області обробки зображень, зокрема, часто застосовується в алгоритмах виділення кордонів.



Рисунок–1.11 Виділення контурів листа за допомогою оператору Собеля

Даний оператор заснований на згортку зображення з цілочисельними фільтрами. У найпростішому випадку оператор побудований на обчисленні згорток вихідного зображення з ядрами  $G_x$  і  $G_y$ , що забезпечують обчислення перших похідних за двома напрямками.

$$G_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} * A \quad (1.9)$$

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} * A \quad (1.10)$$

Оператор Собеля обчислює градієнт яскравості зображення в кожній точці. Таким чином знаходитьсья напрямок найбільшого збільшення яскравості і величина її зміни в цьому напрямку. Результат показує, наскільки «різко» або «плавно» змінюється яскравість зображення в кожній точці, а значить, ймовірність знаходження точки на межі, а також орієнтацію межі. На практиці обчислення величини зміни яскравості (ймовірності приналежності до межі) надійніше і

Змін	Лист	№ докум.	Підпис	Дата

простіше в інтерпретації, ніж розрахунок напрямку.

### 1.5 Метод Laplacian of Gaussian (LoG)

Даний метод орієнтований на підвищення різкості зображень. Фільтр Лапласа є похідним фільтром, використовуваним для виявлення перепадів яскравості в зображені, які, як правило, є контурами зображення. Метод LoG заснований на застосуванні першої або другої похідних.



Рисунок – 1.11 Виділення контурів листа за допомогою оператору LoG

Оскільки похідні фільтри дуже чутливі до шуму, то перед застосуванням Лапласіан застосовують згладжує фільтр Гаусса. Цей двоступінчастий процес називають оператором Лапласіан–Гауссіана (LoG). У загальному вигляді об'єднання Лапласіан і гаусом функції в одному вираженні записується формулою 1.11.

$$\log(x,y) = \frac{-1}{\pi \sigma^4} \left(1 - \frac{x^2 + y^2}{2\sigma^2}\right) e^{\left(\frac{-x^2 + y^2}{2\sigma^2}\right)} \quad (1.11)$$

### 1.6 Висновки до первого розділу

При проведенні аналізу предметного середовища були визначені методи класифікації, що застосовуються для розпізнавання листів. В результаті було зроблено висновки, що нейронні мережі є найбільш перспективним напрямком в даний момент.

Також було з'ясовано, що найбільш повну картину о класі можливо

Змін	Лист	№ докум.	Підпис	Дата

отримати, аналізуючи форму листів рослин. Для виділення форми використовуються методи виділення контурів на зображеннях. Для задачі виділення контуру листів по якості виділених контурів найбільш ефективно показав себе оператор Кенні, але по швидкості обробки зображення найкраще показав себе метод Прюітт і було вирішено використовувати його.

Таблиця 1.1 – Порівняння обробки зображення різними операторами

Оператор	Час (с)	Переваги	Недоліки
Прюітт	0,13	Простота реалізації, мала ресурсомісткість, детектування країв і їх орієнтації	Неточність виділення кордонів об'єктів, чутливий до шуму, можливість появи розривів в контурі, вплив шуму кутових елементів трохи вище, ніж у оператора Собеля
Laplacian of Gaussian	0,16	Досить високе швидкодія і відносно невелика складність обчислень, нечувствітельність до орієнтації кордонів областей; збільшення чіткості країв і дрібних деталей на зображення	Сильна чутливість до шумів навіть після застосування фільтра Гаусса, можливо появі розривів в контурі, а також їх подвоєння
Кенні	0,30	Ефект згладжування для видалення шуму, правильне визначення положення кордону, підвищення відносини сигнал / шум, єдиний відгук на одну кордон, виняток помилкового виявлення контуру там, де об'єктів немає	Межі мають деяку кінцеву товщину. Тому необхідно виконати потоншення ліній придущенням немаксимальних точок в перпендикулярному до кордону напрямку, тобто в напрямку градієнта
Собель	0,16	Простота реалізації, мала ресурсомісткість, детектування країв і їх орієнтації, вплив шуму кутових елементів кілька менше, ніж у методу Прюітт	Неточність виділення кордонів об'єктів, неможливість виробляти точне виявлення країв з тонкими і гладкими краями і можливість появи розривів в контурі

## 2 РОЗРОБКА МЕТОДИКИ КЛАСИФІКАЦІЇ ЛИСТІВ РОСЛИН НА ЗОБРАЖЕННІ

### 2.1 Виділення значущих ознак на зображенні

Не існує універсального чи точного визначення, що собою являє ознака, і точне визначення часто залежить від задачі або типу застосування. Враховуючи це, ознака визначається як «цікава» частина зображення, і ознаки використовуються як відправні точки для багатьох алгоритмів комп'ютерного зору. Оскільки ознаки використовуються як відправні точки та основні примітиви для наступних алгоритмів, загальний алгоритм часто буде лише настільки добрим, наскільки добрий є його детектор ознак. Отже, бажаною властивістю детектора ознак є повторюваність: чи буде одну й ту ж ознаку виявлено на двох або більше різних зображеннях однієї тієї ж сцени, чи ні.

Виявлення ознак є низькорівневою операцією обробки зображень. А саме, вона зазвичай виконується як перша операція на зображенні, і перевіряє кожен піксель, щоби побачити, чи присутня ознака в цьому пікселі. Якщо це є частиною більшого алгоритму, то цей алгоритм зазвичай перевірятиме зображення лише в областях ознак. Як будована передумова для виявлення ознак, вхідне зображення зазвичай згладжується гаусовим ядром у масштабно–просторовому представленні, що обчислюється одне або декілька зображень ознак, часто виражених в термінах операцій локальних похідних зображень.

Іноді, коли виявлення ознак є обчислювально витратним, і присутні часові обмеження, може застосовуватися алгоритм вищого рівня для скерування етапу виявлення ознак, так що пошук ознак здійснюватиметься лише деякими частинами зображення.

Багато алгоритмів комп'ютерного зору використовують виявлення ознак як перший крок, так що в результаті було розроблено дуже велику кількість детекторів ознак. Вони сильно різняться за типами ознак, що виявляють, за обчислювальною складністю та повторюваністю.

Змін	Лист	№ докум.	Підпис	Дата

### 2.1.1 Використання оператору Прюітт для виділення контурів

В ідеальному випадку, результатом застосування до зображення детектора контурів може бути набір з'єднаних кривих, що позначають межі об'єктів, межі забарвлення поверхонь, а також усі криві, що відповідають розривам в орієнтації поверхонь. Таким чином, застосування алгоритму виявлення контурів до зображення може значно зменшувати кількість даних, що підлягають обробці, відфільтровуючи інформацію, яка може розглядатися як менш значуща, але зберігаючи важливі структурні властивості зображення. Якщо крок виявлення контурів є успішним, то подальшу задачу інтерпретування інформаційного вмісту первісного зображення може бути істотно спрощено. Проте не завжди можливо отримувати такі ідеальні контури в картинах реального світу середньої складності.



Рисунок – 2.1 Виділення контурів листа за допомогою оператору Прюітта

Даний метод обробки зображень був створений доктором Джудіт Прюітт (Judith Prewitt) для виявлення меж медичних зображень. Суть цього методу полягає в обчисленні максимального відгуку на безлічі ядер згортки для знаходження локальної орієнтації кордону в кожному пікселі. Оператор використовує два ядра  $3 \times 3$ , згортуючи вихідне зображення для обчислення наблизених значень похідних: одне по горизонталі і одне по вертикалі. Покладемо  $A$  вихідним зображенням, і  $G_x, G_y$  – двома зображеннями, в яких кожна точка містить горизонтальне і вертикальне наближення похідної, яка розраховується за

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

формулами (2.1) і (2.2).

$$Gx = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} * A \quad (2.1)$$

$$Gy = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} * A \quad (2.2)$$

$Gx$  і  $Gy$  – дві матриці, де кожна точка містить наближені похідні по  $x$  і по  $y$ . Вони обчислюються наступним чином шляхом множення матриці  $Gx$  і  $Gy$  і підсумовуванням обох матриць, в результаті отриманий результат записується в поточні координати  $x$  і  $y$  в нове зображення (2.3):

$$G = \sqrt{G_x^2 + G_y^2} \quad (2.3)$$

### 2.1.2 Аналіз виділених контурів та формування значущих ознак

Багато алгоритмів комп'ютерного зору використовують виявлення ознак як перший крок, так що в результаті було розроблено дуже велику кількість детекторів ознак. Вони сильно різняться за типами ознак, що виявляють, за обчислювальною складністю та повторюваністю.

Після виділення кордонів на зображені листа, потрібно провести дискретизацію отриманих ліній. Після дискретизації проводимо спрошення краю листа, з'єднуючи точки дискретизації. (рис. 2.2)

Частота дискретизації задається користувачем та залежить від плавності країв листа. На рисунку 2.2 також показано зеленою лінією форму листового зображення після успішного виявлення межі. Червоним квадратом показана точка на формі листового зображення, з якої проводиться лінія до наступного квадрату.

Синя лінія з'єднання центри двох квадратів, з якого ми збираємося розрахувати косинус і синус кута, які виступають у вигляді номінальних признаків. Така синя лінія являє собою символічне зображення листа.

Змін	Лист	№ докум.	Підпис	Дата

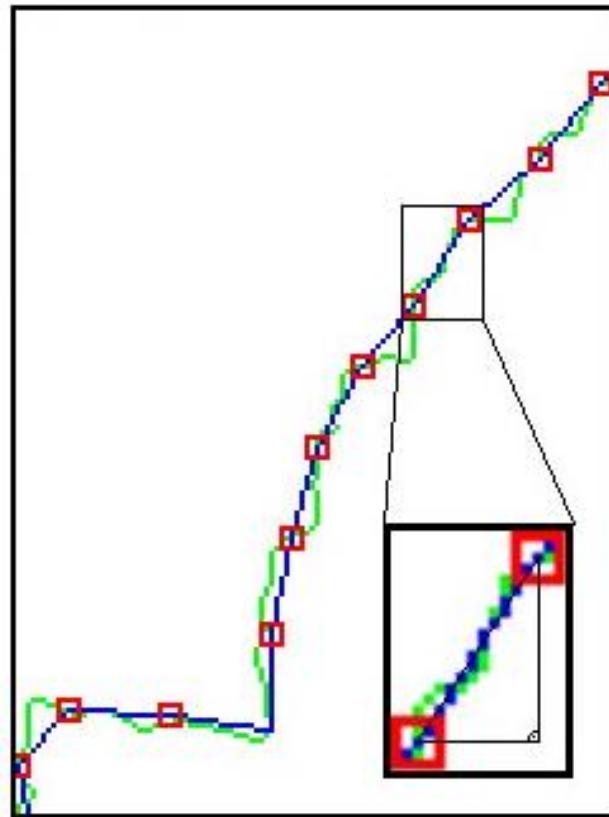


Рисунок 2.2 – Аналіз виділених контурів та спрощення форми листа

Отже, аналізуючи форму листа можна сформувати признований простір з таких номінальних признаків, як  $\sin$  і  $\cos$  першого гострого кута прямокутного трикутника, у якого лінія спрощеної геометрії листа виступає гіпотенузою, а катетами є дві лінії, проведенні паралельно краям зображення до їх перетину.

Після цього стане можливим знаходження кута між лініями форми листа (рис. 2.3). Для отримання повної картини форми листа знайдемо  $\sin A$  и  $\cos A$ , це дозволить побудувати ознаку, яка характеризує довжину лінії, напрям та кут її нахилу.

Після видлення ознак стає можливим подати ці ознаки на вход нейронної мережі для навчання і подальшого тестування на тестовій вибірці.

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

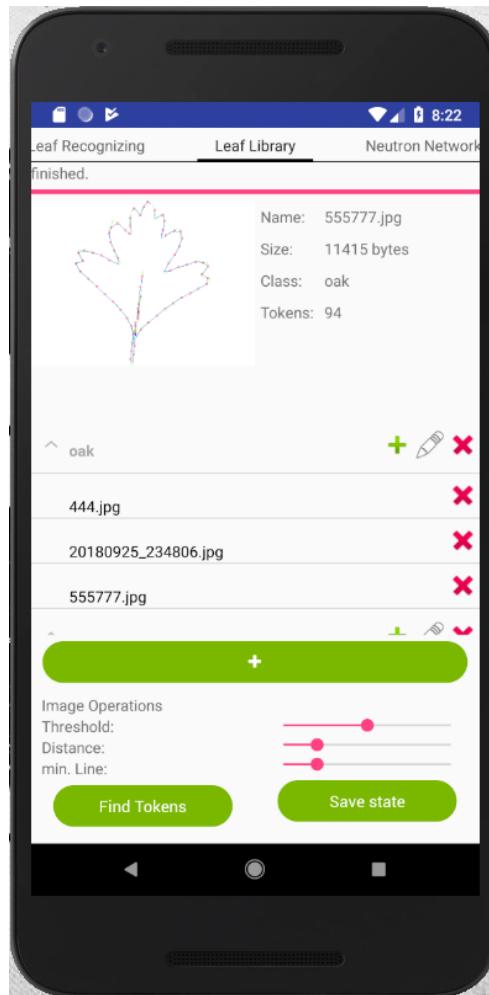


Рисунок 2.3 – Аналіз листа дуба у розробленому мобільному додатку

Приведемо приклад збереження точки у конфігураційному файлі формату xml:

```

<leafSpecies name="oak" IDlen="2">
    <ID value="1.0"></ID><ID value="0.0"></ID>
    <leafImage file="/storage/emulated/0/Download/species1_02-min.jpg">
        <leafToken x1="106" y1="3" x2="119" y2="20">
        </leafToken>
        <leafToken x1="106" y1="3" x2="102" y2="8">
        </leafToken>
        <leafToken x1="119" y1="20" x2="129" y2="30">
        </leafToken>
        ...
    </leafImage>
</leafSpecies>

```

Змін	Лист	№ докум.	Підпис	Дата

У конфігураційному файлі записується інформація о класі листа (блок leafSpecies), та самі листів (блок leafImage). У блоці leafImage описується файл зображенням листа (leafImage) та токени листа (leafToken). Для мобільності застосування при не знайдені файлу зображення листа, буде використовуватися його модель на основі збережених токенів. В описанні токену зберігаються 2 точки X та Y положення на координатний площині. З даних токену вираховуються параметри sinA та cosA для передачі на вхід нейромережі.

## 2.2 Опис структури нейромережі в мобільному застосуванні

Багатошарова нейронна мережа може моделювати функцію практично будь-якого ступеня складності, причому число шарів і число елементів в кожному шарі визначають складність функції. Визначення числа проміжних шарів і числа елементів в них є важливим питанням при конструкції. Саме тому було вирішено перенести ці параметри з внутрішньої бізнес логіки мобільного застосування як елементи користувальського інтерфейсу (рис. 2.4).

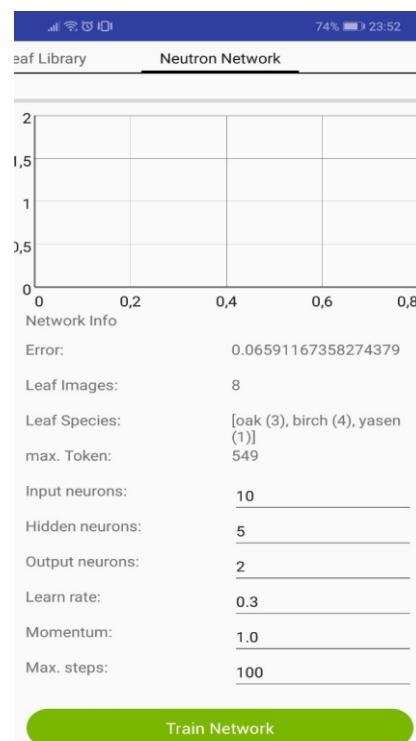


Рисунок 2.4 – Задання параметрів у користувальському інтерфейсі

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

Нейронна мережа також зберігається у конфігураційному файлі типу xml. У конфігураційний файл зберігаються параметри, за яких навчалася мережа, такі як кількість вхідних нейронів, кількість нейронів у скритому слої та кількість вихідних нейронів, коефіцієнт швидкості навчання, інерційний момент, ваги та зміщення нейронів:

```
<backProp input="10" hidden="5" output="2" alpha="0.3" momentum="1.0">
<hidden>
<hiddenW H="0.5012115767551585"></hiddenW>
...
<hiddenW H="-0.7870215678164689"></hiddenW>
<biasH H="-0.17145088949747075"></biasH>
</hidden>
...
<hidden>
<hiddenW H="-0.9401779477188791"></hiddenW>
...
<hiddenW H="1.0690450226196784"></hiddenW>
<biasH H="0.3855046117801849"></biasH>
</hidden>
<output>
<outputW O="0.6413747264713401"></outputW>
...
<biasO O="0.21608408852030903"></biasO>
</output>
<Error Error="0.014684572612121796"></Error>
</backProp>
```

Нейронна мережа складається з декількох шарів нейронів. Формальна модель нейрона представлена на рисунку 2.5. Нейрон має  $n$  входів  $x_1, x_2, \dots, x_n$ . Кожен вход являє собою числове значення з деякого діапазону. Для кожного входу діапазон може відрізнятися. У нейроні значення входу множиться на коефіцієнт – вага входу  $w$ .  $x_1, x_2, \dots, x_n, w_1, w_2, \dots, w_n$ . Результати множення складаються:  $A =$

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

$\Sigma_i(x_i w_i)$ . Від результату обчислюється деяка функція:  $y = f(A)$ . Існує кілька різновидів функцій: лінійна, нелінійна, стрибкоподібне. Шар нейронів складається з декількох однотипних нейронів. Однотипні нейрони обчислюють однакові функції.

Нейронна мережа складається з одного або кількох шарів. Шар приймає вихід попереднього шару в якості входу. Будемо називати вагові матриці, з'єднані з входами, вагами входу шару, а вагові матриці для сигналів, що йдуть від шару, назовемо вагами виходу шару. Далі, будемо використовувати верхні індекси, щоб вказати джерело і адресат для різних ваг і інших елементів нейронної мережі. Щоб пояснити це, розглянемо спочатку тільки один, перший шар багатошарової мережі (рис. 2.5)

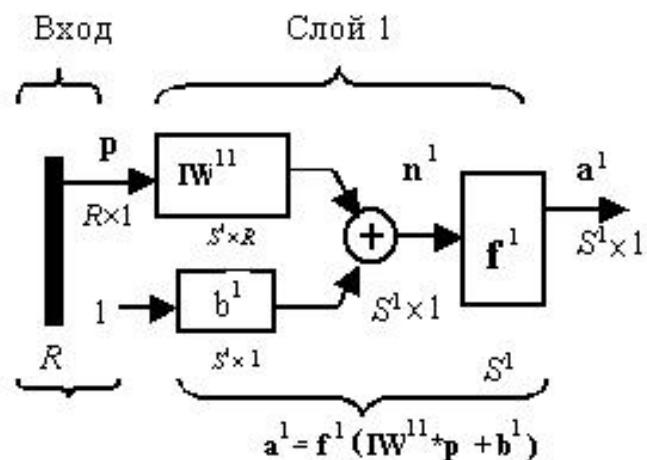


Рисунок – 2.5 Перший шар багатошарової мережі

Позначимо вагову матрицю, пов'язану з входами, через  $IW^{11}$ , верхні індекси якої вказують, що джерелом входів є перший шар (другий індекс) і адресатом є також перший шар (перший індекс). Елементи цього шару, такі, як зміщення  $b^1$ , вхід функції активації  $n^1$  і вихід шару  $a^1$ , мають верхній індекс 1, щоб позначити, що вони пов'язані з першим шаром. Надалі для матриць ваг входу і виходу шару будуть використані позначення  $IW$  (Input Weight) і  $LW$  (Layer Weight) відповідно.

Коли мережа має кілька шарів (багатошарові нейронні мережі), то кожен шар має свою матрицю ваг  $W$ , вектор зміщення  $b$  і вектор виходу  $a$ . Щоб розрізняти

Змін	Лист	№ докум.	Підпис	Дата

вагові матриці, вектори виходу і т. д. для кожного з цих шарів, введемо номер шару як верхній індекс для представляє інтерес змінної. Використання цієї системи позначень для мережі з трьох шарів можна бачити на показаної нижче структурній схемі із рівнянь, наведених в нижній частині рис. 2.6.

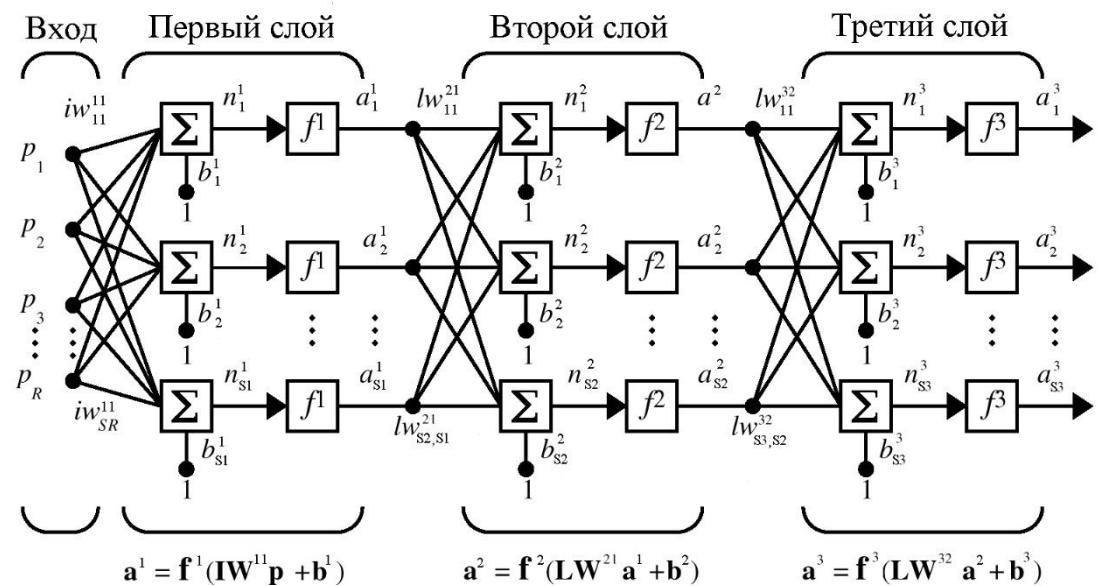


Рисунок 2.6 – Структурна схема трьохшарової мережі

Мережа, показана вище, має R входів, S1 нейронів в першому шарі, S2 нейронів у другому шарі. Для спільноті будемо вважати, що різні шари мають різне число нейронів. На зміщення для кожного нейрона поданий постійний вхідний сигнал 1. Зауважимо, що виходи кожного проміжного шару служать входами для наступного шару. Таким чином, шар 2 може бути розглянуто як один шар мережі з S1 входами, S2 нейронами і S1 'S2 матрицею ваг W2. Вхід до шару 2 є 1, а вихід – 2. Тепер, коли позначені всі вектори і матриці шару 2, можна трактувати його як самостійну одношарову мережу. Такий підхід може бути використаний до будь-якого шару мережі.

Шари багатошарової мережі мають різні призначення. Шар, який утворює вихід мережі, називається шаром виходу. Всі інші верстви називаються прихованими шарами. Тришарова мережа, показана вище, має вихідний шар (шар

3) і 2 прихованих шару (шар 1 і шар 2). Ця ж тришарова мережа може бути представлена у вигляді укрупненої структурної схеми (рис. 2.7).

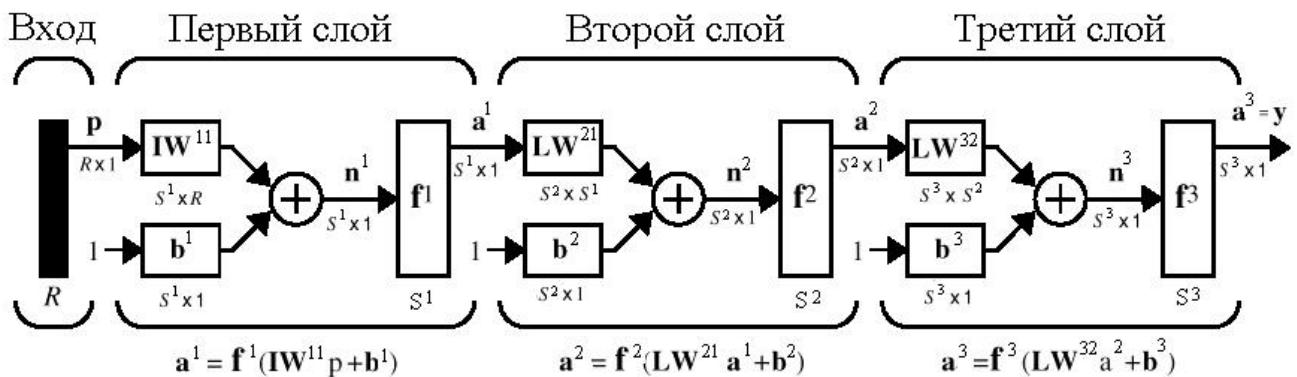


Рисунок 2.7 – Повна структурна схема трьохшарової мережі

Зауважимо, що вихід третього шару  $a^3$  позначений через  $y$ . Це зроблено для того, щоб підкреслити, що вихід останнього шару є виходом мережі.

Багатошарові мережі мають досить потужними можливостями. Наприклад, двошарова мережа, в якій перший шар містить сигмоїдальну, а другий шар – лінійну функцію активації, може бути навчена апроксимувати із довільною точністю будь-яку функцію з кінцевим числом точок розриву.

У даній роботі функцією активації нейронів мережі є сигмоїдальна функція активації  $OUT = \frac{1}{1+\exp(-\alpha Y)}$  (рис. 2.8), де  $\alpha$  – параметр нахилу сигмоїдальної функції. Змінюючи цей параметр, можна побудувати функції з різною крутизною. Слід зазначити, що сігмоїдной функція диференційована на всій осі абсцис, що широко використовується в багатьох алгоритмах навчання. Крім того, вона має властивість підсилювати слабкі сигнали краще, ніж сильні, і запобігає насичення від сильних сигналів, так як вони відповідають областям аргументів, де сигмоїд має пологий нахил. Ці особливості важливі для задачі розпізнавання листів, так як форма листа рослини характеризується однорідністю, яка полягає в тому, що листів після обробки мають однакову кількість токенів, а інші ознаки, такі як кількість прожилок та зубців листа не враховуються. Єдина відмінність однотипних листів різних класів – це їх кути. Тому різниця між однотипними листівми, що подаються

Змін	Лист	№ докум.	Підпис	Дата

на вхід нейронної мережі, буде невелика, що пред'являє до чутливості функції додаткові вимоги. Нейрони повинні «розділувати» слабо розрізняються вхідні сигнали. Це робить сигмоїдальну функцію найбільш прийнятною.

Сигмоид звужує діапазон зміни так, що значення OUT лежить між нулем і одиницею. Багатошарові нейронні мережі мають більшу представляє потужністю, ніж одношарові, тільки в разі присутності нелінійності. Стискаюча функція забезпечує необхідну нелінійність. Насправді є безліч функцій, які могли б бути використані. Для алгоритму зворотного поширення помилки потрібно лише, щоб функція була всюди диференцируема. Сигмоид задовольняє цій вимозі. Його додаткова перевага полягає в автоматичному контролі посилення. Для слабких сигналів (тобто коли OUT близько до нуля) крива вхід–виход має сильний нахил, що дає велике посилення. Коли величина сигналу стає більше, посилення падає. Таким чином, великі сигнали сприймаються мережею без насичення, а слабкі сигнали проходять по мережі без надмірного ослаблення.

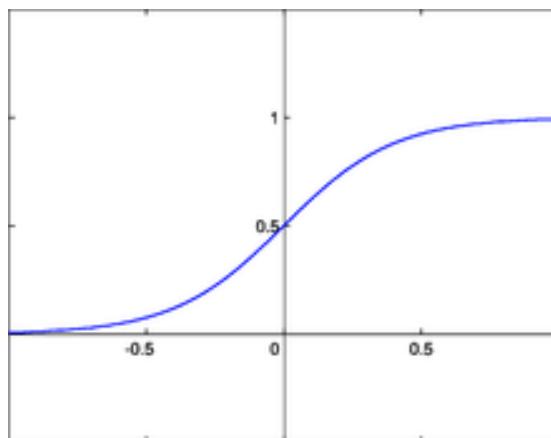


Рисунок 2.8 – Сигмоїдальна функція активування

Якщо останній шар багатошарової мережі використовує такі функції активування, то виходи мережі будуть обмежені. Коли в вихідному шарі використовуються лінійні нейрони, то виходи мережі можуть приймати довільні значення.

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

## 2.2.1 Опис реалізації алгоритму зворотного поширення помилки

Розглянемо процес навчання нейронної мережі, що передбачає наявність «вчителя», тобто процес навчання, при якому навчання відбувається шляхом надання мережі послідовності навчальних прикладів з правильними відповідями.

Метою навчання мережі алгоритмом зворотного поширення помилки є така підстроювання її ваг, щоб додаток деякого безлічі входів призводило до необхідного безлічі виходів. Для стисlostі ці безлічі входів і виходів будуть називатися векторами. При навчанні передбачається, що для кожного вхідного вектора існує парний йому цільової вектор, що задає необхідний вихід. Разом вони називаються навчальною парою. Мережа навчається на багатьох парах.

Алгоритм зворотного поширення помилки є одним з методів навчання багатошарових нейронних мереж прямого поширення, званих також багатошаровими персепtronами. Багатошарові персепtronи успішно застосовуються для вирішення багатьох складних завдань.

Навчання алгоритмом зворотного поширення помилки передбачає два проходи по всім верствам мережі: прямого і зворотного. При прямому проході вхідний вектор подається на вхідний прошарок нейронної мережі, після чого поширюється по мережі від шару до шару. В результаті генерується набір вихідних сигналів, який і є фактичною реакцією мережі на даний вхідний образ. Під час прямого проходу все синаптичні ваги мережі фіксовані. Під час зворотного проходу все синаптичні ваги налаштовуються відповідно до правила корекції помилок, а саме: фактичний вихід мережі віднімається з бажаного, в результаті чого формується сигнал помилки. Цей сигнал згодом поширюється по мережі в напрямку, протилежному напрямку синаптичних зв'язків. Звідси і назва – алгоритм зворотного поширення помилки. Синаптичні ваги налаштовуються з метою максимального наближення вихідного сигналу мережі до бажаного.

Алгоритм зворотного поширення помилки наступний:

1. ініціалізувати синаптичні ваги маленькими випадковими значеннями;

Змін	Лист	№ докум.	Підпис	Дата

2. вибрати чергову навчальну пару з навчальної множини, подати вхідний вектор на вхід мережі;
3. обчислити вихід мережі;
4. обчислити різницю між виходом мережі і необхідним виходом;
5. підкоригувати ваги мережі для мінімізації помилки;
6. повторювати кроки з 2 по 5 для кожного вектора навчальної множини до тих пір, поки помилка на всій множині не досягне прийнятного рівня.

Операції, що виконуються кроками 2 і 3, подібні до тих, які виконуються при функціонуванні вже навченої мережі, тобто подається вхідний вектор і обчислюється виходить вихід. Обчислення виконуються пошарово. На рис. 1 спочатку обчислюються виходи нейронів шару В (шар А вхідний, а значить ніяких обчислень в ньому не відбувається), потім вони використовуються в якості входів шару С, обчислюються виходи OUT\_{CN} нейронів шару С, які і утворюють вихідний вектор мережі OUT. Кроки 2 і 3 утворюють так званий «прохід вперед», так як сигнал поширюється по мережі від входу до виходу.

Кроки 4 і 5 складають «зворотний прохід», тут вираховується сигнал помилки поширюється назад по мережі і використовується для підстроювання ваг.

Розглянемо детальніше 5 крок – коригування ваг мережі. Тут слід виділити два нижчеописаних випадку.

#### Випадок 1. Коригування синаптичних ваг вихідного шару

Наприклад, для моделі нейронної мережі на рис. 2.9, це будуть ваги мають такі позначення:  $W_{B1-C1}$  і  $W_{B2-C1}$ . Визначимося, що індексом р будемо позначати нейрон, з якого виходить синаптична вага, а q – нейрон в який входить:

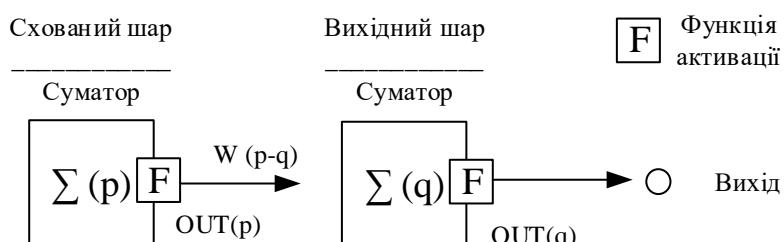


Рисунок 2.9 – Модель нейронної мережі

Змін	Лист	№ докум.	Підпис	Дата

Введемо величину  $\delta$ , яка дорівнює різниці між необхідним  $T_q$  і реальним  $OUT_q$  виходами, помноженої на похідну логістичної функції активації:

$$\delta_q = OUT_q (1 - OUT_q)(T_q - OUT_q) \quad (2.4)$$

Тоді, ваги вихідного шару після корекції будуть рівними:

$$w_{p-q}(i+1) = w_{p-q}(i) + \eta \delta_q OUT_p \quad (2.5)$$

де:

$i$  – номер поточної ітерації навчання;

$W_{p-q}$  – величина синаптического ваги, що з'єднує нейрон  $p$  з нейроном  $q$ ;

$\eta$  – коефіцієнт «швидкості навчання», дозволяє управляти середньою величиною зміни ваг;

$OUT_p$  – вихід нейрона  $p$ .

Наведемо приклад обчислень для синаптического ваги  $W_{B1-C1}$ :

$$\delta_{C1} = OUT_{C1} (1 - OUT_{C1})(T_1 - OUT_{C1})$$

$$w_{B1-C1}(i+1) = w_{B1-C1}(i) + \eta \delta_{C1} OUT_{B1}$$

Випадок 2. Коригування синаптических ваг прихованого шару

Для моделі нейронної мережі на рис. 2.10, це будуть ваги відповідні верствам А і В. Визначимося, що індексом  $p$  будемо позначати нейрон з якого виходить синаптична вага, а  $q$  – нейрон в який входить.

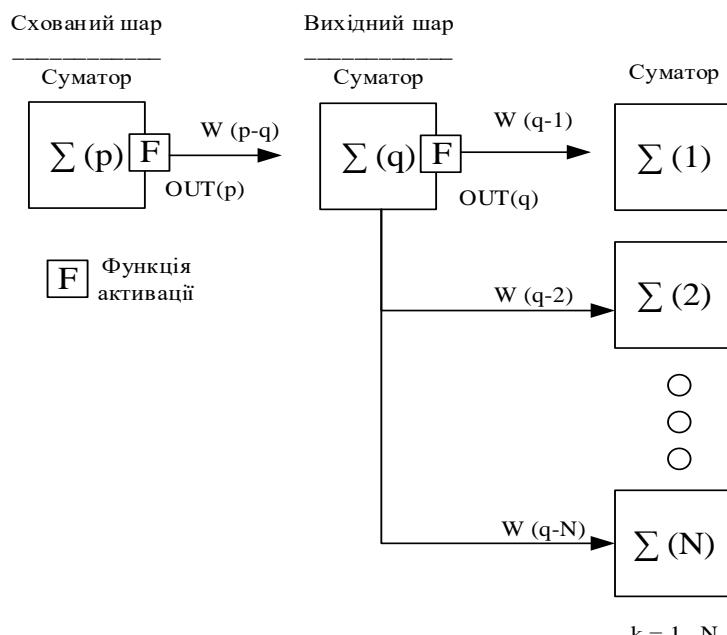


Рисунок 2.10 – Модель нейронної мережі

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

Введемо величину  $\delta$ , яка дорівнює:

$$\delta_q = OUT_q(1 - OUT_q) \sum_{k=1}^M \delta_k w_{q-k} \quad (2.6)$$

, де  $\sum_{k=1}^N \delta_k w_{q-k}$  – сума від 1 до N

Тоді, ваги прихованих шарів після корекції будуть рівними:

$$w_{p-q}(i+1) = w_{p-q}(i) + \eta \delta_q OUT_p \quad (2.7)$$

Наведемо приклад обчислень для синаптического ваги  $W_{A1-B1}$ :

$$\delta_{B1} = OUT_{B1} (1 - OUT_{B1})$$

$$w_{A1-B1}(i+1) = w_{A1-B1}(i) + \eta \delta_{B1} OUT_{A1}$$

Для кожного нейрона в прихованому шарі має бути обчислено  $\delta$  і налаштовані все ваги, асоційовані з цим шаром. Цей процес повторюється шар за шаром у напрямку до входу, поки все ваги не будуть скоректовані.

Вибрані гіперпараметри можуть значно вплинути на фінальний результат роботи алгоритму. В загальному випадку для підбору гіперпараметрів використовуються наступні підходи:

1. Перебiranня усіх можливих варіантів з деякого простору гіперпараметрів. Це один з найбільш витратних підходів, який, проте, дає достатньо надійні результати. Для деяких алгоритмів він неможливий, або майже неможливий.

2. Перебiranня випадкових варіантів з деякого простору гіперпараметрів. Перевагою над першим підходом є можливість задати певну кількість спроб вибору гіперпараметрів, що дозволяє приблизно задати час, необхідний для перебiranня. Недоліком — майже гарантована неоптимальність фінальної комбінації гіперпараметрів.

3. Використання алгоритмів оптимізації для підбору гіперпараметрів. Вибрана метрика класифікації (як якість роботи алгоритму) виступає як значення цільової функції, яку необхідно оптимізувати.

4. Емпіричний метод. Дослідник має спиратися на своє розуміння даних, алгоритму та досвід для підбору найбільш ефективних гіперпараметрів. Незважаючи на значне збільшення обчислювальних потужностей, цей метод залишається одним із основних методів підбору гіперпараметрів.

Змін	Лист	№ докум.	Підпис	Дата

У данній роботі використовувалась наступні гіперпараметри: швидкість навчання рівна 0.03, кількість епох рівна 1000.

### 2.3 Висновки до другого розділу

На закінчення можна сформулювати наступні висновки. Після виділення кордонів на зображені листа, потрібно провести дискретизацію отриманих ліній. Після дискретизації проводимо спрощення краю листа, з'єднуючи точки дискретизації.

Аналізуючи форму листа можно сформувати признаковий простір з таких номінальних признаків, як  $\sin i$  і  $\cos$  першого гострого кута прямокутного трикутника, у якого лінія спрощеної геометрії листа виступає гіпотенузою, а катетами є дві лінії, проведені паралельно краям зображення до їх перетину.

Після цього стане можливим знаходження кута між лініями форми листа. Для отримання повної картини форми листа знайдемо  $\sin A$  и  $\cos A$ , це дозволить побудувати ознаку, яка характеризує довжину лінії, напрям та кут її нахилу.

Після виділення ознак стає можливим подати ці ознаки на вход нейронної мережі для навчання і подальшого тестування на тестовій вибірці. Вхід функції активації нейрона визначається зміщенням і сумою зважених входів. Вихід нейрона залежить як від входів нейрона, так і від виду функції активації. Один нейрон не може вирішувати складні завдання, проте кілька нейронів, об'єднаних в один або кілька шарів, мають більші можливості.

Архітектура мережі складається з опису того, скільки шарів має мережу, кількості нейронів в кожному шарі, виду функції активації кожного шару та інформації про з'єднання шарів. Архітектура мережі залежить від тієї конкретної задачі, яку повинна вирішувати мережу.

Робота мережі полягає в обчисленні виходів мережі на основі відомих входів з метою формування бажаного відображення вхід / вихід. Конкретне завдання визначає число входів і число виходів мережі. Крім числа нейронів у вихідному шарі мережі, для проектувальника важливо число нейронів в кожному шарі. Більша

Змін	Лист	№ докум.	Підпис	Дата

кількість нейронів в прихованих шарах забезпечує більш потужну мережу. Якщо має бути реалізовано лінійне відображення, то слід використовувати нейрони з лінійними функціями активації.

При цьому треба пам'ятати, що лінійні нейронні мережі не можуть формувати нелінійні відображення. Використання нелінійних функцій активації дозволяє налаштовувати нейронну мережу на реалізацію нелінійних зв'язків між входом і виходом.

Мережі зі зміщенням дозволяють формувати більш складні зв'язки між входами і виходами, ніж мережі без зміщення. Наприклад, нейрон без зміщення, коли всі входи нульові, буде завжди ставити вхід функції активації рівним нулю, проте нейрон зі зміщенням може бути навчений так, щоб при тих же умовах задати вхід функції активації довільної форми.

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

### 3 РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУВАННЯ ДЛЯ КЛАСИФІКАЦІЇ ЛИСТІВ РОСЛИН НА ЗОБРАЖЕННІ

#### 3.1 Користувацький інтерфейс мобільного застосування

Для реалізації застосування були розроблені мокапи інтерфейсу користувача. Застосування матиме три вікна, реалізовані за допомогою технології Fragment, що дозволить плавний перехід з одного вікна до іншого (рис. 3.1)

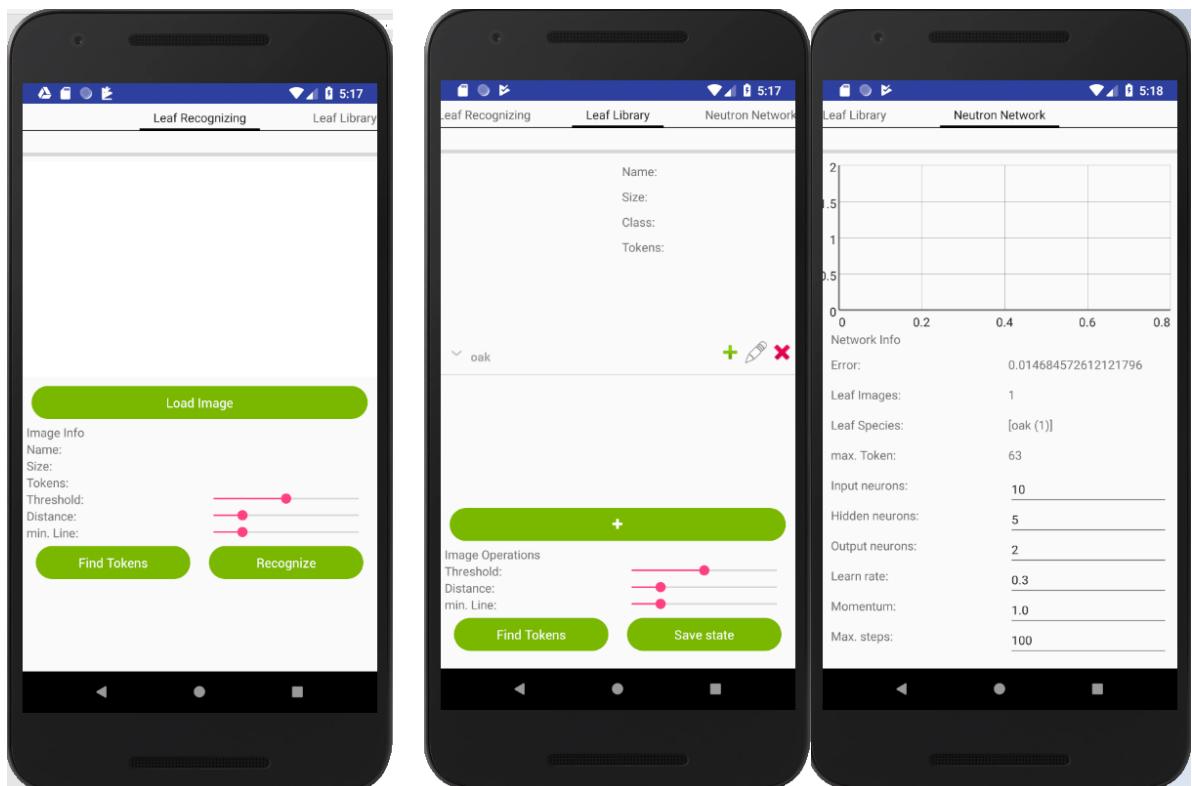


Рисунок 3.1 – Три головних вікна застосування

Детально розглянемо кожен з них:

Перше вікно дає можливість користувачу завантажити фотографію або зображення з пам'яті пристрою за допомогою кнопки «Load Image». (рис 3.2) За допомогою та повзунків користувач здатен обрати параметри для аналізу зображення та побудування спрощеної форми листа.

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

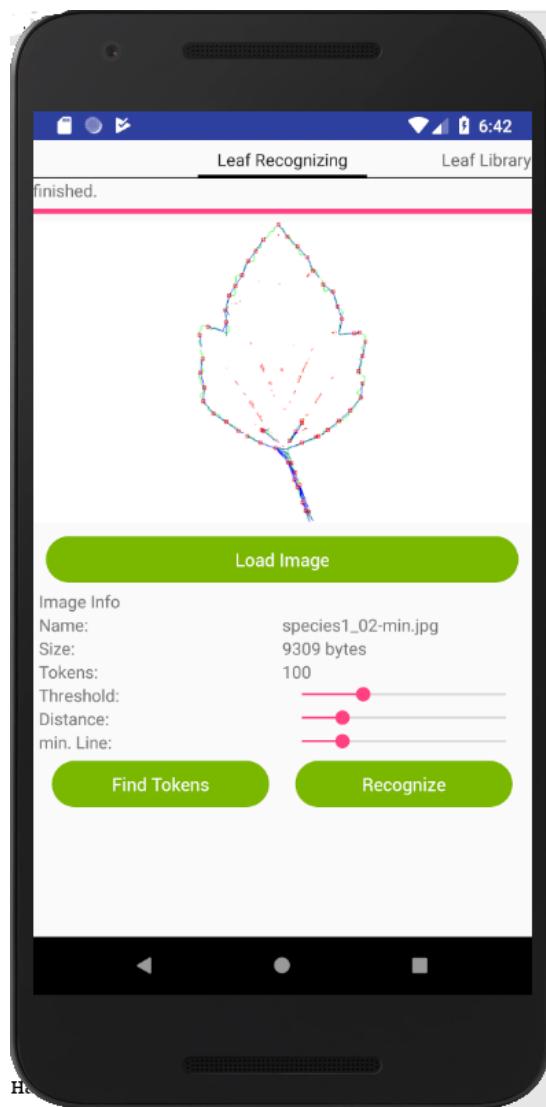


Рисунок 3.2 – Вікно розпізнавання листа

Користувач здатен задати три параметра:

- Threshold – поріг, який застосовуються для того щоб вирішити, знаходиться чи ні межа в даній точці зображення. Чим менший поріг, тим більше границь буде знаходитися, але тим більше сприймаючу до шуму стане результатом, виділяючи інші дані зображення. Навпаки, високий поріг може пропустити слабкі краю або отримати фрагменти фрагмента.
- Distance – відстань між точками дискретизації геометрії форми листа.
- Min. line – мінімальна кількість пікселів лінії, яка буде визнана як частина форми листа.

Останні кнопки «Find Tokens» та «Recognize» запускають процедури пошуку

Змін	Лист	№ докум.	Підпис	Дата

токенів на зображенні, використовуючи задані параметри та розпізнавання класу листа за допомогою навченої нейронної мережі та токенів листа, отриманих на попередньому етапі.

Друге вікно призначення для зберігання, огляду, налаштування колекції листків, які використовуються для навчання нейронної мережі (рис 3.3).

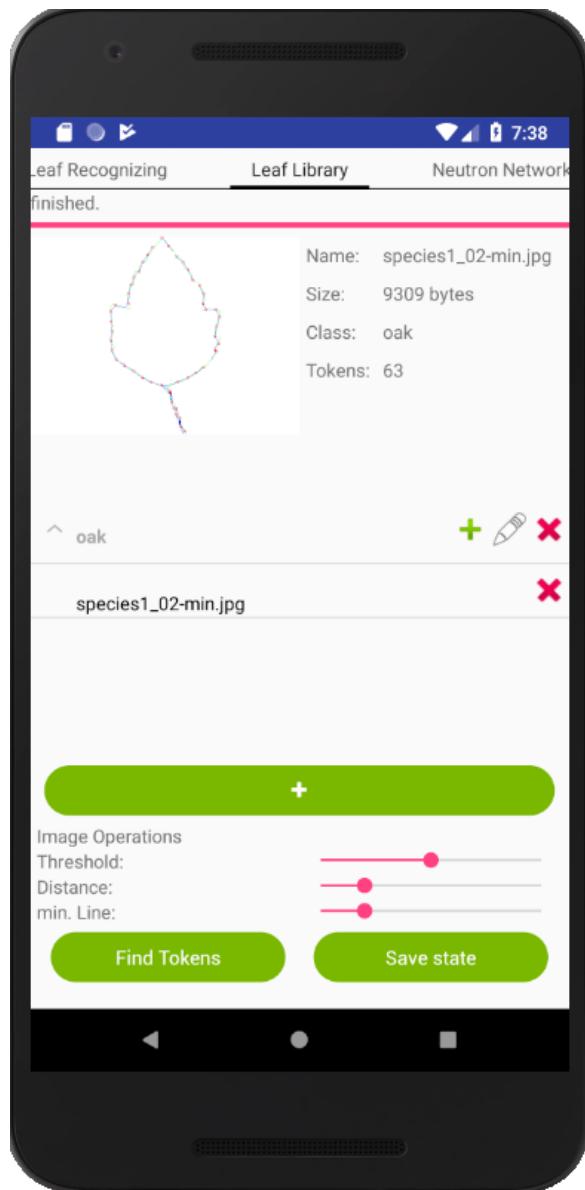


Рисунок 3.3 – Вікно колекції листів

Користувачу доступна кнопка «Додати клас», яка добавляє в систему клас с заданим ім'ям, та після цього стає можливим завантаження у нього зображень листів цього класу. Кожне зображення підлягає обробці та пошуку на ньому токенів

Змін	Лист	№ докум.	Підпис	Дата

листа. Пошук відбувається за тим же сценарієм, що і на першому вікні. Кожен клас можна редагувати або видалити за допомогою кнопок управління. Також користувачу доступна кнопка збереження колекції листів та навченої нейронної мережі у конфігураційний файл (рис. 2.5 та рис. 2.7).

Останнє вікно це вікно налаштування нейронної мережі. Користувач має можливість задавати параметри навчання, кількість кроків для навчання та стежити за зміною значення помилки на графіку (рис. 3.4). Кнопка «Train network» дозволяє почати процес навчання нейронної мережі.

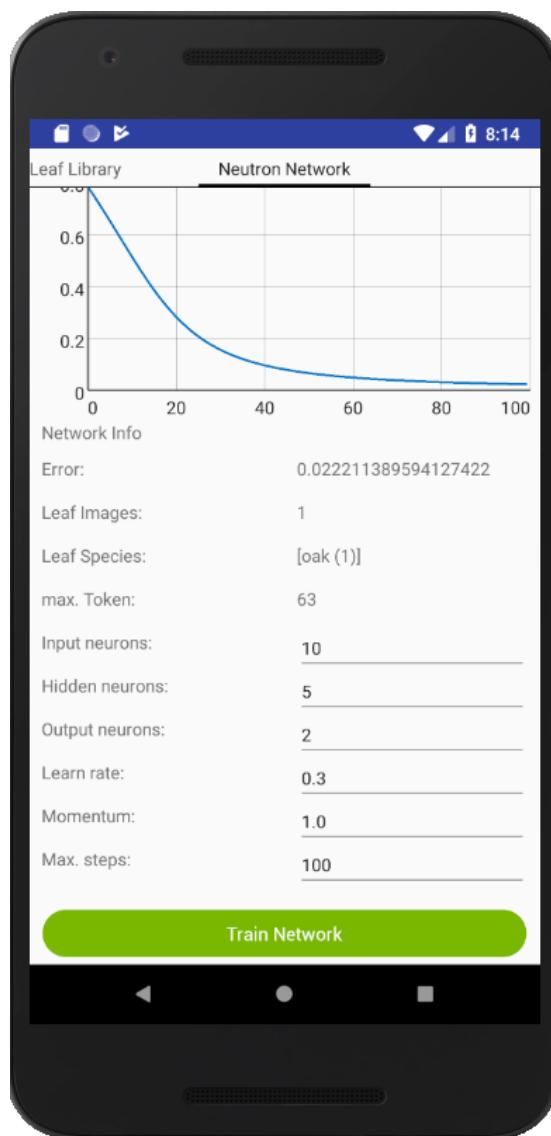


Рисунок 3.4 – Вікно нейронної мережі

Отже приведемо use-case діаграму застосування (рис.3.5):

Змін	Лист	№ докум.	Підпис	Дата

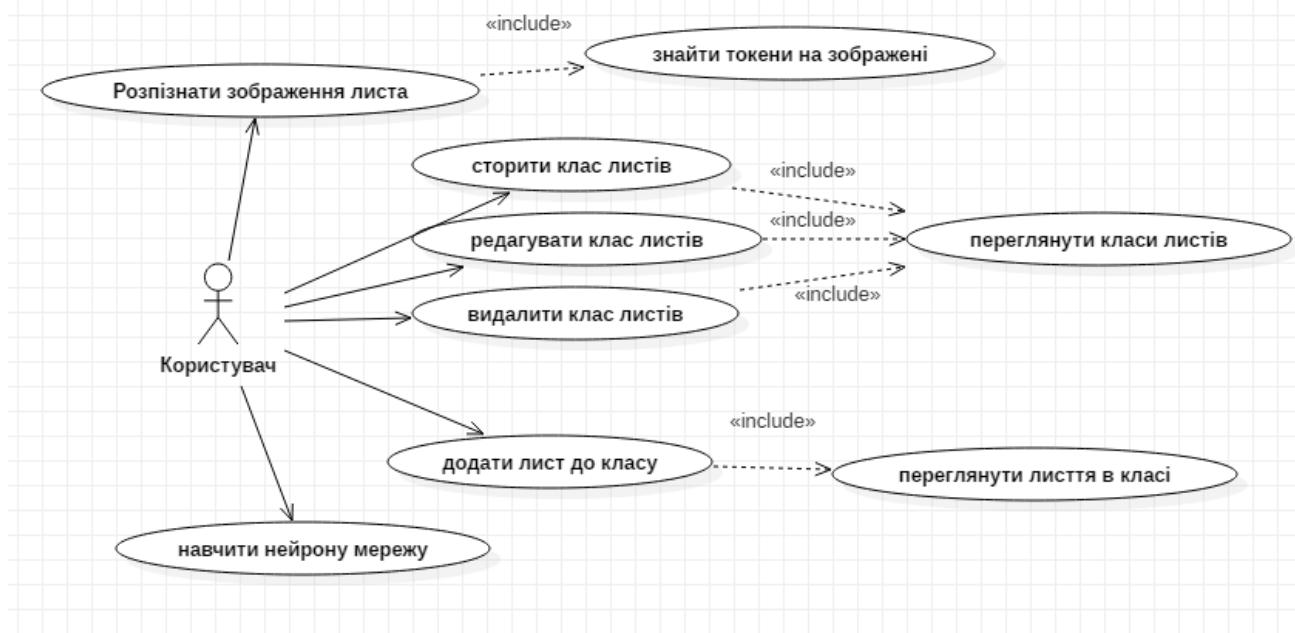


Рисунок 3.5 – діаграма застосування (Use case)

### 3.2 Загальна архітектура застосування

Для реалізації застосування була розроблена діаграма потоків даних, яка зображена на рисунку 3.6:

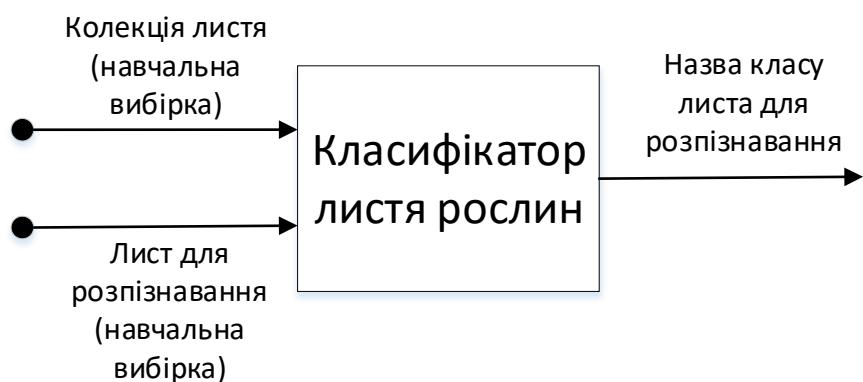


Рисунок 3.6 – Діаграма потоків даних

Розглянемо детально класифікатор, який структурно складається з трох модулей, зображених на рисунку 3.7:

Змін	Лист	№ докум.	Підпис	Дата	Лист
					ІС КРМ 122 036 ПЗ

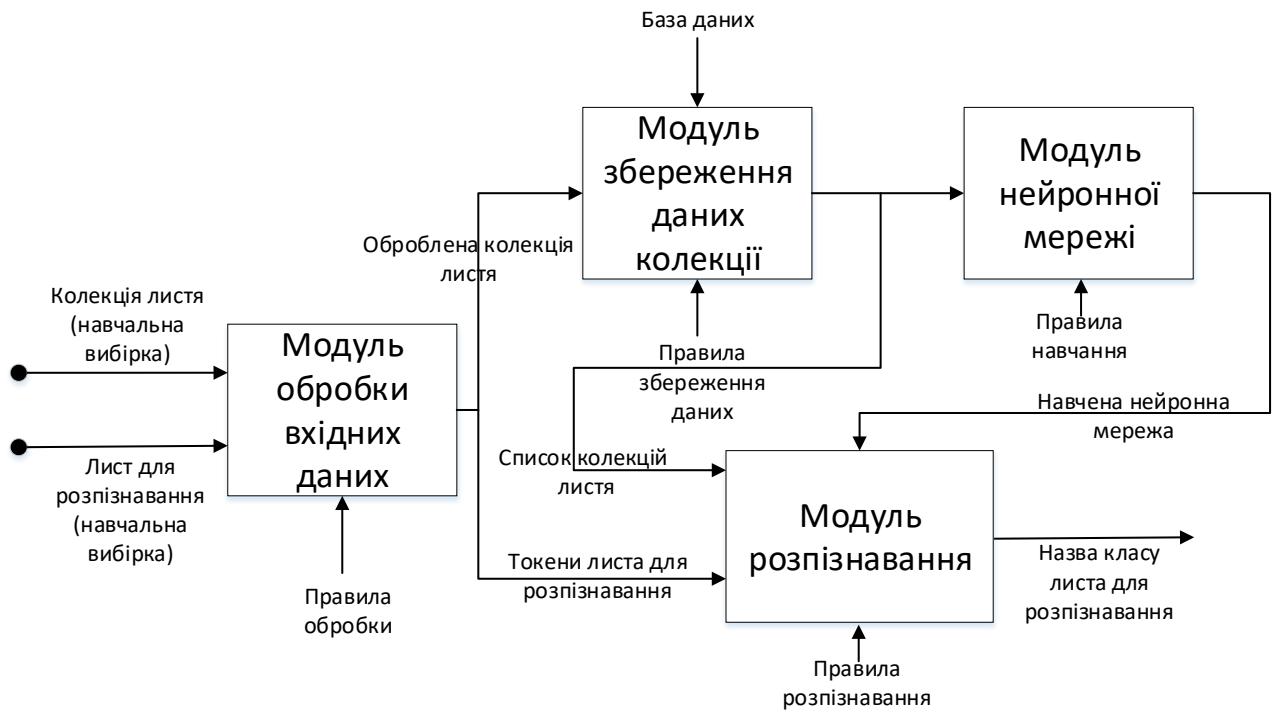


Рисунок 3.7 – Структура класифікатора

Як видно з схеми необроблені фотографії колекції листів та листів для розпізнавання попадають до модуля обробки вхідних даних (в програмі представлених класом ImageProcessor), в якому зображення оброблюється за допомогою оператора Прюйт (рис. 3.8) та виділяються токени, які передаються до модуля збереження даних, якщо зображення навчальне, або у модуль розпізнавання, якщо зображення тестове. Навчання нейронної мережі на навчальній виборці проходить у модулі нейронної мережі, на виході якого навчена нейрона мережа.

```

for (x = width - 2; x > 0; x--) {
    for (y = height - 2; y > 0; y--) {
        int dx = (getGrayPixel(x - 1, y + 1) + getGrayPixel(x, y + 1) + getGrayPixel(x + 1, y + 1)) -
            (getGrayPixel(x - 1, y - 1) + getGrayPixel(x, y - 1) + getGrayPixel(x + 1, y - 1));
        int dy = (getGrayPixel(x + 1, y - 1) + getGrayPixel(x + 1, y) + getGrayPixel(x + 1, y + 1)) -
            (getGrayPixel(x - 1, y - 1) + getGrayPixel(x - 1, y) + getGrayPixel(x - 1, y + 1));
        int z = (int)(Math.sqrt(dx * dx + dy * dy) / 3);
    }
}

```

Рисунок 3.8 – Обробка оператором Прюйт, фрагмент лістингу класу ImageProcessor

Змін.	Лист	№ докум.	Підпис	Дата

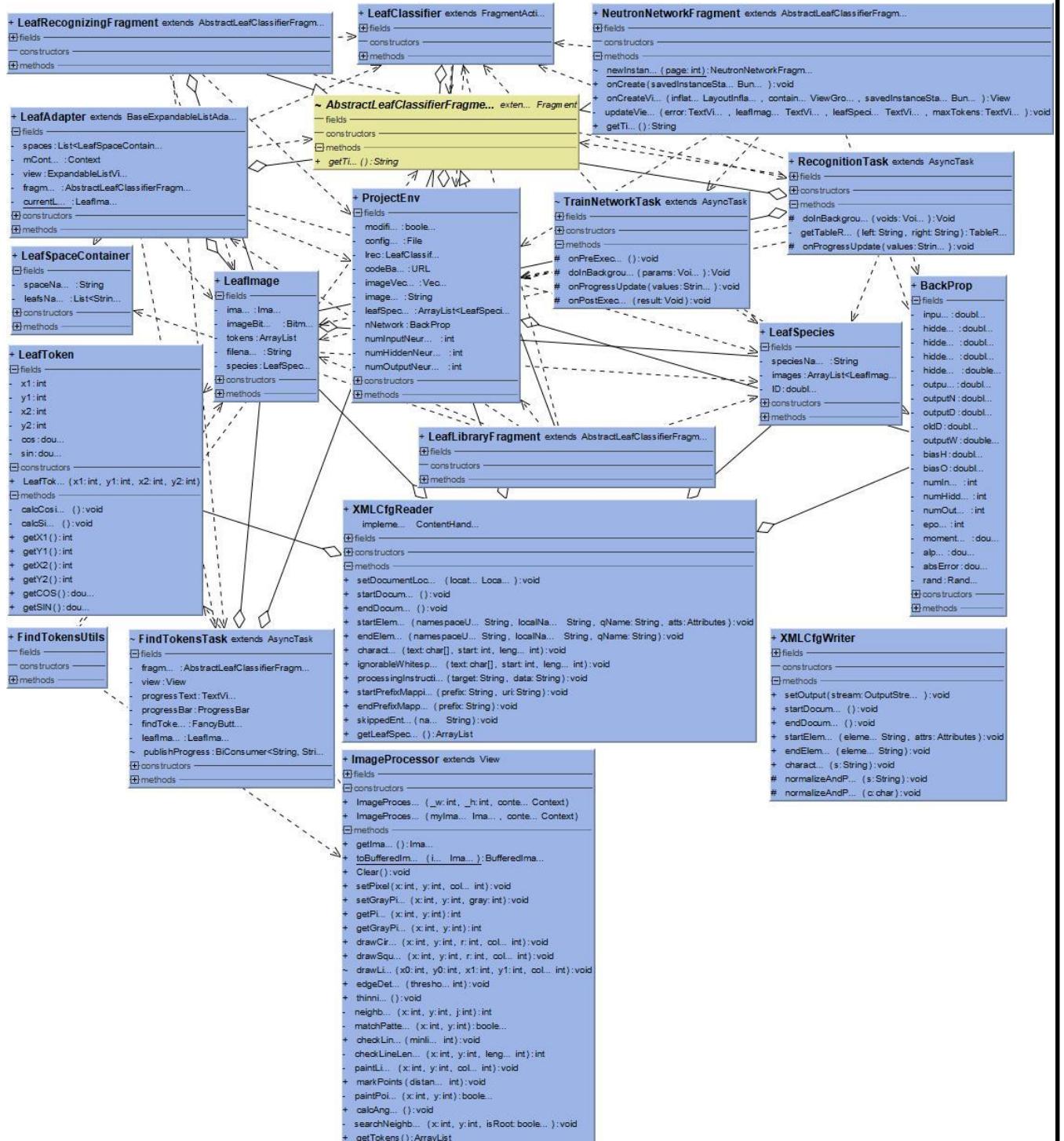


Рисунок 3.9 – Діаграма класів мобільного застосування

Також для реалізації застосування була розроблена діаграма класів (рис 3.9).

Розглянемо три фрагменти застосування, описані вище унаслідуються від абстрактного класу AbstractFragment. Внаслідок цієї реалізації стає можливим повторне використання загального коду у таких фрагментах, як розпізнавання листів та колекція листів. Для цих цілей був створений клас FindTokensTask, але

Змін	Лист	№ докум.	Підпис	Дата

загальна реалізація була зроблена у утильному класі `FindTokensUtils` (рис 3.10). У фрагменті з розпізнаванням листів також викликається схожий клас `RecognizingTask` (рис 3.11). Ці класи унаслідуються від класу `AsyncTask`, його мета – це виконання важких завдань і передача в UI–потік результатів роботи.

```

ImageProcessor imgProc = new ImageProcessor(leafImage.getImage(), view.getContext());
publishProgress.accept("Edge detection...", "5");
imgProc.edgeDetect(threshold.getProgress() * 10);
publishProgress.accept("Thinning...", "20");
imgProc.thinning();
publishProgress.accept("Line checking...", "40");
imgProc.checkLines(minLine.getProgress() * 10);
publishProgress.accept("Distance points...", "60");
imgProc.markPoints(distance.getProgress() * 10);
publishProgress.accept("Searching tokens...", "70");
// now we calculate the tokens of the image by calculating
// the angles
imgProc.calcAngels();
// set the TextField for the amount of tokens
List leafTokens = imgProc.getTokens();
leafImage.setTokens(leafTokens);
BufferedImage bufferedImage = ImageProcessor.toBufferedImage(imgProc.getImage());
ByteArrayOutputStream baos = new ByteArrayOutputStream();
try {
    ImageIO.write(bufferedImage, "png", baos);
    baos.flush();
    byte[] imageInByte = baos.toByteArray();
    Bitmap bmp = BitmapFactory.decodeByteArray(imageInByte, 0, imageInByte.length);
    leafImage.setImageBitmap(bmp);
    fragment.getActivity().runOnUiThread(() -> imageView.setImageBitmap(bmp));
} catch (IOException e) {
    e.printStackTrace();
}
publishProgress.accept("finished.", "100");

```

Рисунок 3.10 – Фрагмент лістингу класу `FindTokensUtils`

```

BackProp nNetwork = LeafClassifier.getProjectEnv().getNetwork();
double[] resultID = nNetwork.propagate(leafImageForRecognizing.getTokens(nNetwork.numInput()));
publishProgress("classifying.", "85");

```

Змін	Лист	№ докум.	Підпис	Дата

```

List leafSpecies =LeafClassifier.getProjectEnv().getLeafSpecies();
double error=0.0;
int size = leafSpecies.size();
TreeMap<String, Double> results = new TreeMap<>();
for (int i = 0; i < size; i++, error = 0.0) {
    LeafSpecies lSpecies =(LeafSpecies) leafSpecies.get(i);
    double[] ID=lSpecies.getID();
    for(int j = 0; j < nNetwork.numOutput();j++) {
        if (ID[j] >= resultID[j]) {
            error += ID[j] - resultID[j];
        } else error += resultID[j] - ID[j];
    }
    double probabilityValue =(100.0 -(100 / size) * error);
    results.put(lSpecies.getName(),probabilityValue);
}
results.entrySet().stream().
sorted(reverseOrder(Map.Entry.comparingByValue())).forEach((entry) -> {
    TableRow row =getTableRow(entry.getKey(),String.format("%.2f", entry.getValue()));
    fragment.getActivity().runOnUiThread() -> {
        tableLayout.addView(row,new
TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT,
TableLayout.LayoutParams.WRAP_CONTENT));
    });
});
publishProgress("finished.", "100");

```

Рисунок 3.11 – Фрагмент лістингу класу RecognitionTask

З діаграми класів можна бачити, що застосування працює, використовуючи декілька класів контейнерів, такі як:

- LeafSpaceContainer – клас–контейнер, для збереження інформації о класах листів.
- LeafImage – клас, який описує лист. Містить у собі такі дані, як зображення у форматі jpg, якщо воно існує та зображення, відновлене, використовуючи токени листа.
- LeafToken – клас, який описує токен. Містить в собі координаті двох точок, використовуючи які можливо відновити геометрію листа та розрахувати параметри для подачі на вхід нейронної мережі.

Змін	Лист	№ докум.	Підпис	Дата

Також у застосуванні є два класи XMLCfgReader та XMLCfgWriter, які відповідають за збереження та відновлення даних з xml. Вони надають однопрохідний доступ лише для читання XML–даних без кешування. Розглянемо читання xml файлу (рис 3.12).

```

if(qName.compareToIgnoreCase("leafSpecies")==0)
{
    actSpecies = new LeafSpecies(atts.getValue("name"));
    if(atts.getValue("IDlen")!=null)
    {
        actSpecies.ID = new double[Integer.parseInt(atts.getValue("IDlen"))];
    }
    numID = -1;
}
else if(qName.compareToIgnoreCase("ID") == 0)
{
    numID++;
    actID = Double.parseDouble(atts.getValue("value"));
}
else if(qName.compareToIgnoreCase("leafImage") == 0)
{
    String filePath = atts.getValue("file");
    File image = new File(filePath);
    BitmapFactory.Options bmOptions = new BitmapFactory.Options();
    Bitmap bitmap = BitmapFactory.decodeFile(image.getAbsolutePath(),bmOptions);
    actImage = new LeafImage(bitmap , filePath);
    actImage.setSpecies(actSpecies);
}
else if(qName.compareToIgnoreCase("leafSize") == 0)
{
    height = Integer.parseInt(atts.getValue("height"));
    width = Integer.parseInt(atts.getValue("width"));
}
else if(qName.compareToIgnoreCase("leafToken") == 0)
{
    int x1 = Integer.parseInt(atts.getValue("x1"));
    int y1 = Integer.parseInt(atts.getValue("y1"));
    int x2 = Integer.parseInt(atts.getValue("x2"));
    int y2 = Integer.parseInt(atts.getValue("y2"));

    actToken = new LeafToken(x1,y1,x2, y2);
}
else if(qName.compareToIgnoreCase("backProp") == 0)
{
    int input = Integer.parseInt(atts.getValue("input"));
    int hidden = Integer.parseInt(atts.getValue("hidden"));
    int output = Integer.parseInt(atts.getValue("output"));
    double alpha = Double.parseDouble(atts.getValue("alpha"));
    double momentum= Double.parseDouble(atts.getValue("momentum"));

    actNetwork = new BackProp(input,hidden,output,alpha,momentum);
    numHidden = numHiddenW = -1;
    numOutput = numOutputW = -1;
}
else if(qName.compareToIgnoreCase("hidden") == 0)
{
    numHidden++;
}

```

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

```

numHiddenW = -1;
}
else if(qName.compareToIgnoreCase("hiddenW") == 0)
{
    numHiddenW++;
    actHiddenW = Double.parseDouble(atts.getValue("H"));
}
else if(qName.compareToIgnoreCase("biasH") == 0)
{
    actBiasH = Double.parseDouble(atts.getValue("H"));
}
else if(qName.compareToIgnoreCase("output") == 0)
{
    numOutput++;
    numOutputW = -1;
}
else if(qName.compareToIgnoreCase("outputW") == 0)
{
    numOutputW++;
    actOutputW = Double.parseDouble(atts.getValue("O"));
}
else if(qName.compareToIgnoreCase("biasO") == 0)
{
    actBiasO = Double.parseDouble(atts.getValue("O"));
}
else if(qName.compareToIgnoreCase("Error") == 0) {
    error = Double.parseDouble(atts.getValue("Error"));
}

```

Рисунок 3.12 – Фрагмент лістингу класу XMLCfgReader

Нейрона мережа у застосувані описана класом BackProp. Процес навчання описаний класом TrainNetworkTask.

### 3.3 Представлення шарів

Мобільне застосування можна розділити на шари (рис.3.13). Це дозволить зробити процес розробки частин програми більш логічним та упорядкованим. Також розділення на шари дозволить в майбутньому змінити частини програми за бажанням і при цьому не переробляти інші готові частини.

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

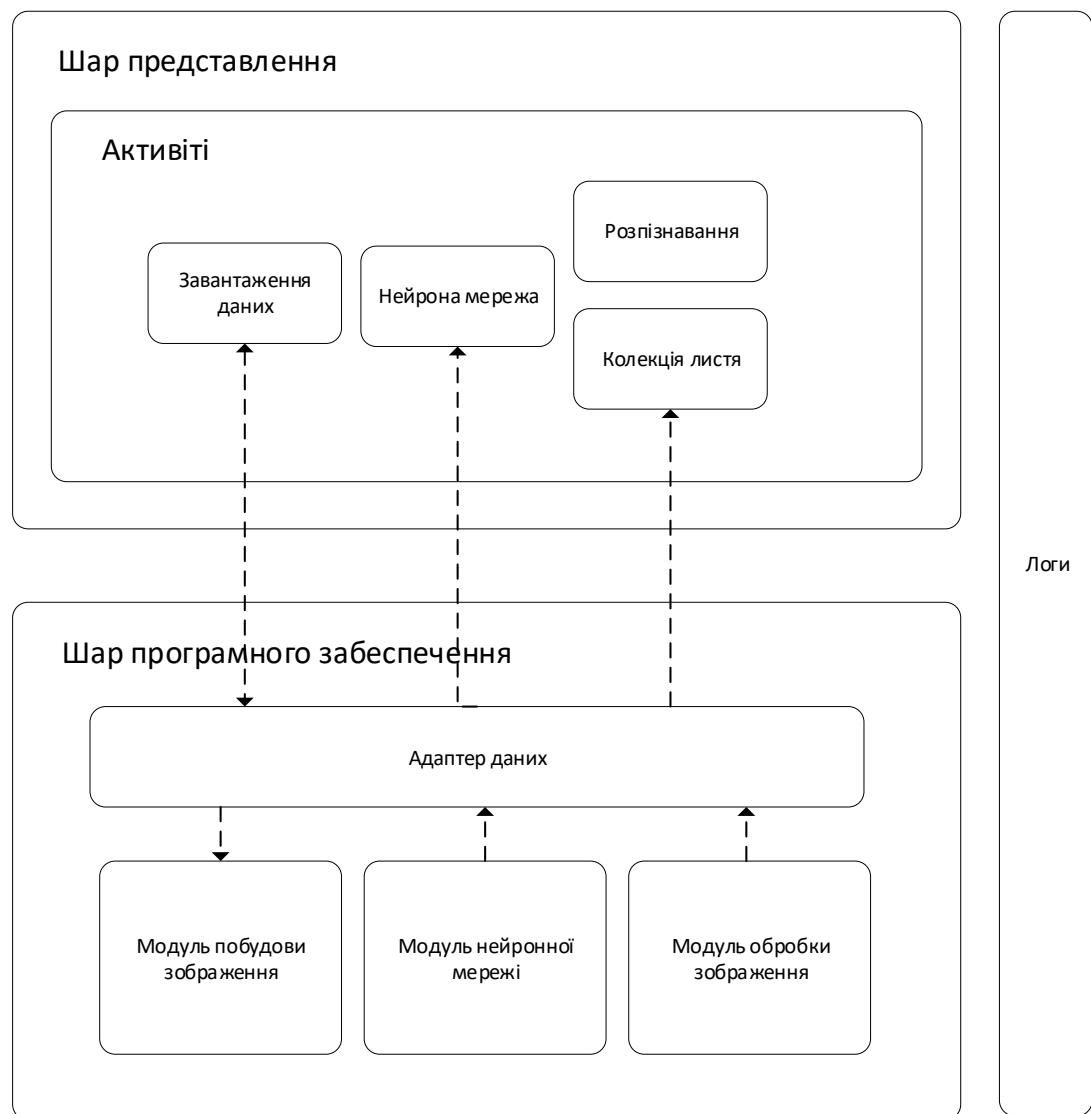


Рисунок 3.13 – Шари ПЗ мобільного застосування

### 3.4 Інтеграційний процес

Для управління кодом проекту та збірки apk–файлу була застосована система Gradle. Це система автоматичного збирання, яка далі розвиває принципи, закладені в Apache Ant та Apache Maven і використовує предметно–орієнтовану мову (DSL) на основі мови Groovy замість традиційної XML–подібної форми представлення конфігурації проекту. Для визначення порядку виконання завдань Gradle використовує орієнтований ацикличний граф ("DAG"). Gradle було розроблено для побудови мультипроектів, які можуть розростатися, і підтримує інкрементальне збирання. Вона визначає, які частини було змінено, і виконує тільки ті задачі, які

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

залежать від цих частин. Нижче приведений файл build.gradle (рис. 3.14), який дозволяє завантажувати вказані залежності з центрального репозіторія.

```
apply plugin: 'com.android.application'

android {
    compileSdkVersion 28
    buildToolsVersion '28.0.3'
    defaultConfig {
        applicationId "com.ishchenko.artem.leafclasifierandroid"
        minSdkVersion 14
        targetSdkVersion 28
        versionCode 4
        versionName "1.0"
        testInstrumentationRunner "android.support.test.runner.AndroidJUnitRunner"
        multiDexEnabled true
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
    compileOptions {
        targetCompatibility 1.8
        sourceCompatibility 1.8
    }
    productFlavors {
    }

    signingConfigs {
        release {
        }

        buildTypes {
            release {
                minifyEnabled true
                signingConfig signingConfigs.release
                proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
            }
        }
    }
}
```

Змін	Лист	№ докум.	Підпис	Дата

```

    }

    compileOptions {
        targetCompatibility 1.8
        sourceCompatibility 1.8
    }
}

dependencies {
    compile files('libs/android-awt-1.0.0.jar')
    implementation 'com.github.jrvansuita:PickImage:2.2.4'
    compile 'xerces:xercesImpl:2.8.0'
    compileOnly 'org.projectlombok:lombok:1.16.20'
    compile group:'commons-io',name:'commons-io',version:'2.6'
    annotationProcessor 'org.projectlombok:lombok:1.16.20'
    implementation fileTree(include: ['*.jar'], dir: 'libs')
    implementation 'com.android.support.constraint:constraint-layout:1.1.3'
    implementation 'com.jjoe64:graphview:4.2.2'
    implementation 'com.github.medyo:fancybuttons:1.9.0'
    implementation 'com.obsez.android.lib.filechooser:filechooser:1.1.11'
    //noinspection GradleCompatible
    implementation "com.android.support:appcompat-v7:28.0.0-alpha3"
    testImplementation 'junit:junit:4.12'
    androidTestImplementation 'com.android.support.test:runner:1.0.2'
    androidTestImplementation 'com.android.support.test.espresso:espresso-core:3.0.2'
}

```

Рисунок 3.14 – Файл build.gradle

### 3.5 Управління програмним кодом ІС

Проект мобільного застосування знаходиться на репозиторії (<https://github.com/proteus1121/LeafClasifierAndroid>) та розповсюджується під ліцензією Apache 2.0.

Під час розробки було дотримано правил щодо не прямих комітів у master. Дляожної функції було створено окрему гілку. Усі pull request's було закрито у гілку develop, після створення базової версії проекту було проведено злиття з гілки develop у master.

Змін	Лист	№ докум.	Підпис	Дата

Таблиця 3.1 – Управління кодом

Метрика	Значення
Кількість комітів в master бранче	22
Загальна кількість бранчей	6
Кількість закритих PR	9

### 3.6 Розрахунок метрик програмного коду ІС

Для перевірки якості написаного коду було проведено заміри метрик проекту (табл. 3.2).

Таблиця 3.2 – Метрики коду проекту

Метрика	Значення
Загальна кількість строк коду в проекті ІС	3415
Середня кількість строк коду в одному класі	86
Максимальна кількість строк коду в одному класі	176
Середня кількість строк коду в одному методі	36
Максимальна кількість строк коду в одному методі	67
Максимальна глибина дерева наслідування	4
Середня цикломатична складність метода	1,67
Максимальна цикломатична складність метода	4
Коментування коду, %	36
Покриття коду модульними тестами, %	64

Для написання тестів було використано бібліотеку Junit 4.

### 3.7 Контрольний список по якості реалізації ІС

Щоб оцінити якість створеної інформаційної системи було складено список по можливим критеріями.

Список по якості ІС приведено у таблицях 3.4 та 3.5.

Змін	Лист	№ докум.	Підпис	Дата	Лист	ІС КРМ 122 036 ПЗ	59

Таблиця 3.4 – Список по якості мобільного застосування

Твердження	Відповідь	Коментарі
Використання Dependency Injection	Так	
Використання логування (logging)	Так	
Використання модульного тестування	Так	
Захист від SQL ін'єкцій	Так	
Захист від Javascript/XSS ін'єкцій	Так	
Використання валідації даних у всіх полях вводу для користувачьких інтерфейсів	Так	
Використання інсталяторів/інтернет магазинів	Так	Play Market
Використання засобів синхронізації даних у випадку багатопоточного застосування	Так	
Підтримка глобалізації ІС (інтернаціоналізація, локалізація)	Ні	
Відсутність захищих в програмний код конфігураційних параметрів застосування	Ні	
Використання UI патернів при розробці UI	Так	
Врахування coding style guide lines для обраної мови програмування	Так	
Врахування рекомендованих guide lines при розробці користувачького інтерфейсу для тієї чи іншої ОС	Так	
Підтримка локалізації та глобалізації ІС	Так	

### 3.8 Документація ІС

Інструкція по зборці проекту мобільного застосування:

- Завантажте код з репозіторія

<https://github.com/proteus1121/LeafClasificatorAndroid>

- Відкрийте проект у середі програмування, що підтримує Java.
- Натиснути Build.

### 3.9 Забезпечення якості ІС

Для забезпечення якості ІС були проведені наступні тести:

- функціональне тестування;
- модульні тести за допомогою бібліотеки JUnit 4;

Змін	Лист	№ докум.	Підпис	Дата	ІС КРМ 122 036 ПЗ	Лист
						60

- тести навантаження.

### 3.10 Функціональне тестування

Мета функціонального тестування – виявлення невідповідностей між реальною поведінкою реалізованих функцій і очікуваною поведінкою відповідно до специфікації і вимог [6].

Функціональні тести повинні охоплювати всі реалізовані функції. Тестові сценарії, що поєднують окремі тести, орієнтовані на перевірку якості розв'язку функціональних задач.

Для перевірки основних функцій програми, були проведені наступні функціональні тести.

Тест кейс №1. Створення колекції листів:

1. Запустити програму.
2. Натиснути кнопку «Додати колекцію».
3. Ввести необхідні данні.
4. Натиснути кнопку «Створити».

Тест кейс №2. Збереження колекції листів:

1. Запустити програму.
2. Створити колекцію листів.
3. Натиснути кнопку «Поділитися».
4. Обрати за допомогою якого застосування буде відправлений файл.

Тест кейс №3. Збереження заповненої колекції:

1. Запустити програму.
2. Створити колекцію листів.
3. Завантажити зображення листів
4. Натиснути кнопку «Знайти токени».
5. Натиснути кнопку «Зберегти».

Тест кейс №4. Завантаження заповненої колекції:

Змін	Лист	№ докум.	Підпис	Дата

1. Запустити програму.
2. Натиснути кнопку «Завантажити».
3. Обрати шлях до файлу.

Передумова: у обраному шляху міститься файл колекції листів.

Тест кейс №5. Навчання нейронної мережі:

1. Запустити програму.
2. Завантажити колекцію листів.
3. Вибрати параметри навчання
4. Натиснути на кнопку «Навчити нейрону мережу».

Тест кейс №6. Розпізнавання листа:

1. Запустити програму.
2. Завантажити колекцію листів.
3. Натиснути на кнопку «Запустити».
4. Натиснути на кнопку «Навчити нейрону мережу».
5. Завантажити фото листа з тестової виборки

Результати функціонального тестування приведені в таблиці 3.5.

Таблиця 3.5 – Результати функціонального тестування

№	Очікуваний результат	Отриманий результат	Коментарі
1	Створена колекція листів	Створена колекція листів з'явилася на екрані	
2	З'явився файл з назвою «dataset.xml»	З'явився файл з назвою «dataset.xml»	
3	У файлі «dataset.xml» присутні розпізнані токени	У файлі «dataset.xml» присутні розпізнані токени	
4	Колекція листів з'явилася на екрані	Завантажена колекція листів з'явилася на екрані	
5	Графік поширення помилки з'явиться на екрані	Графік поширення помилки з'явився на екрані	
6	Лист розпізнаний правильно	Імовірність принадлежності листа до правильного класу найвища	
	Кількість успішних тест–кейсів	6	
	Кількість провалених тест–кейсів	0	

Змін	Лист	№ докум.	Підпис	Дата

За допомогою функціонально тестування були перевірені основні функції бізнес–логіки ІС, за результатами тестів з'ясовано, що функції працюють, як очікувано.

### 3.11 Модульне тестування

Модульне тестування, або юніт–тестування – це метод тестування програмного забезпечення, який полягає в окремому тестуванні кожного модуля коду програми. Модулем називають найменшу частину програми, яка може бути протестована. У процедурному програмуванні модулем вважають окрему функцію або процедуру. В об'єктно–орієнтованому програмуванні — інтерфейс, клас. Модульні тести, або unit–тести, розробляються в процесі розробки програмістами та, іноді, тестувальниками білої скриньки (white–box testers).

Зазвичай unit–тести застосовують для того, щоб упевнитися, що код відповідає вимогам архітектури та має очікувану поведінку

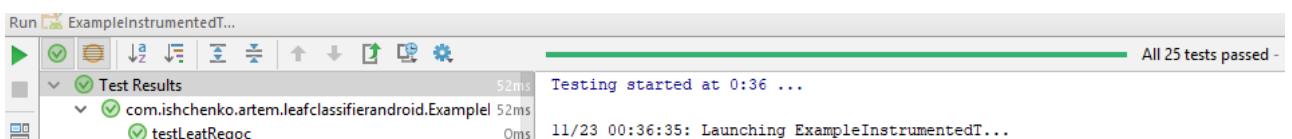


Рисунок 3.15 – Запуск unit–тестів

Покриття кодом було отримано завдяки вбудованому у середу програмування інструменту.

### 3.12 Тестування навантаження

Для перевірки роботи програми було проведено тестування продуктивності (рис 3.16).

Змін	Лист	№ докум.	Підпис	Дата

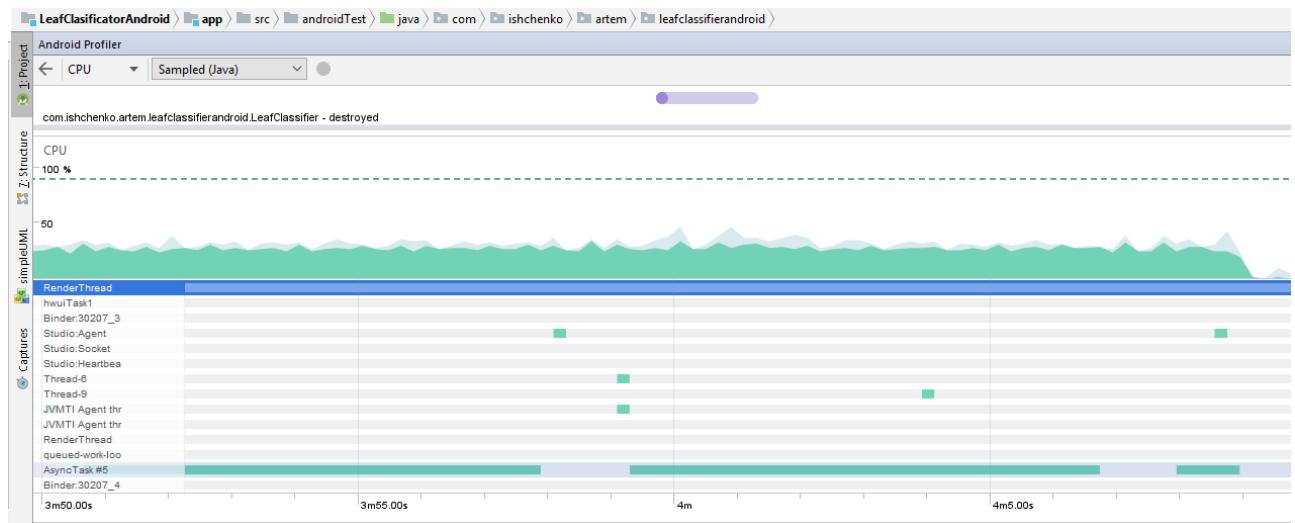


Рисунок 3.16 – Навантаження процесора під час тестування

Тест проводився на смартфоні, що має наступну конфігурацію:

- екран (5.2 ", IPS, 1920x1080);
- процесор: Qualcomm Snapdragon 808 (1.8 ГГц);
- основна камера: 12.3 Мп;
- фронтальна камера: 5 Мп;
- RAM 2 ГБ;
- 32 ГБ вбудованої пам'яті;
- система – Android 6.0 Marshmallow;
- ємність батареї – 2700 мА × ч.

### 3.13 Висновки до третього розділу

На етапі реалізації було створено мобільне застосування для класифікації листів рослин. Кожна нова версія коду була створена за допомогою системи контролю версій. Таким чином, можна побачити прогрес реалізації функцій, легше знаходити та виправляти помилки в коді. За допомогою використання сучасних технологій і шаблонів було створено більш якісних код, ніж без них. Окрім того, отримані метрики коду також показують якість створеного коду. Загалом було описано біля 23 класів на мові Java. Під час розробки було використано шаблон

Змін	Лист	№ докум.	Підпис	Дата	Лист
					ІС КРМ 122 036 ПЗ

Dependency Injection, додано логування, перевірка даних, що вводяться.

Для загальної перевірки системи були проведені наступні тести:

- функціональні тести – для перевірки функціональних вимог;
- модульне тестування – щоб перевірити окремі ізольовані компоненти;
- тестування навантаженням, що показує, як програма завантажує систему під час важких задач.

Для користувачів було створено інструкції, в яких описується використання основних функцій системи. Також було розроблено інструкція по збірці проектів для розробників.

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

## 4 ДОСЛІДЖЕННЯ МЕТОДИКИ НЕЙРОМЕРЕЖЕВОЇ КЛАСИФІКАЦІЇ ЛИСТІВ РОСЛИН

### 4.1 Вибір критеріїв оцінки

Для оцінки роботи застосування проведемо тестування застосування за наступними критеріями:

- час розпізнавання;
- час обробки зображення;
- час навчання нейронної мережі;
- точність розпізнавання.

Для цих цілей був використані 2 колекції листів [8] з сайту UCI Machine Learning Repository – найбільший репозиторій реальних і модельних задач машинного навчання. Містить реальні дані по прикладних задачах в області біології, медицини, фізики, техніки, соціології, і ін. набори даних (data set) саме цього сховища найчастіше використовуються науковим співтовариством для емпіричного аналізу алгоритмів машинного навчання. Репозиторій UCI створений в університеті г.Ірвін (Каліфорнія, США)



Рисунок 4.1 – Приклад різних форм листів першого датасету (10 класів)

У работі викорисаний набір даних, який був створений Педро Ф. Б. Сільвою

Змін	Лист	№ докум.	Підпис	Дата

та Андре Р. С. Марсалем за допомогою зразків листів, зібраних Рубімом Альмеїдою да Сілва на факультеті науки Університету Порту, Португалія. (рис. 4.1)

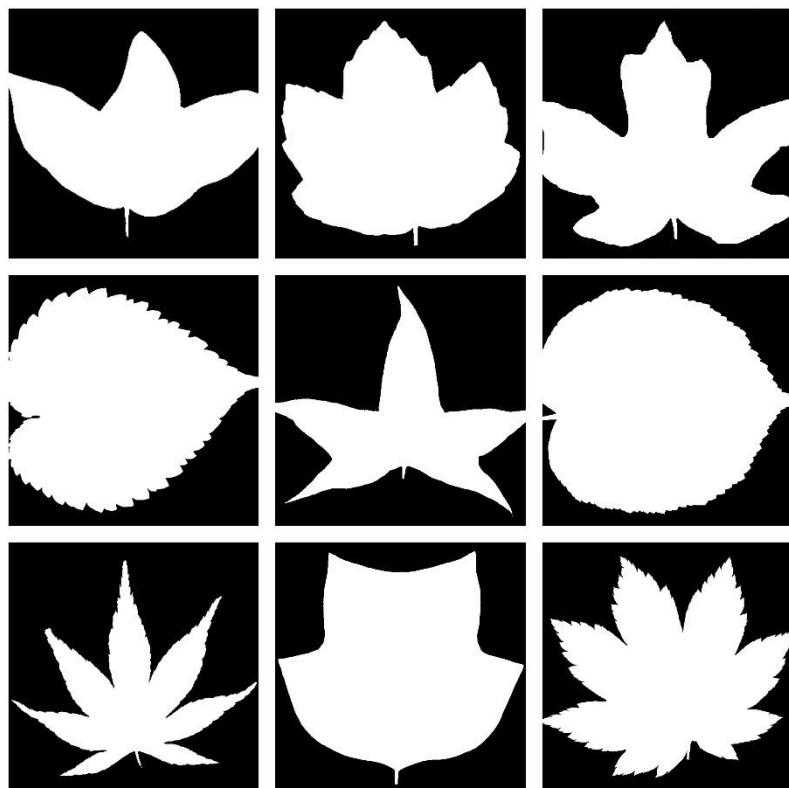


Рисунок 4.2 – Приклад різних форм листів другого датасету (9 класів)

Та набір даних, що складається з робіт, виконаних Джеймсом Коупом, Чарльзом Маллахом та Джеймсом Орвеллом, Кінгстонський університет Лондон (рис. 4.2).

#### 4.2 Опис набору даних

У першій базі даних зображену розглянуто 40 різних видів рослин, середня кількість 10 листкових зразків з кожної рослини. Всього було зібрано 443 зображення.

У другій базі розглянуто 100 видів, по шістнадцять зразків листів го з виду. Всього було зібрано 1600 зображень.

Кожен зразок листів був сфотографований за допомогою двох різних

Змін	Лист	№ докум.	Підпис	Дата	Лист
					ІС КРМ 122 036 ПЗ 67

пристроїв, камери Canon EOS 40D і планшету Apple iPAD2. Для розробки запропонованої в цій теорії системи було зосереджено на  $720 \times 920$  пікселях, 24–бітових зображеннях RGB, отриманих за допомогою пристрою Apple iPAD2.

Зібрани листи сфотографувались на контрастному полотні. Для зелених листів використовувався червоний колір. Для особливого випадку листів Acer Palmatum був використаний сірий фон. Вибір кольорів був довільний, поважаючи виключно попередньо зазначений стан придбання зображення листа на контрастному полотні. У другому датасеті наведена тільки форма листів.

В таблиці 4.2 зібрана інформація про види рослин та кількість зразків, а на рисунку 4.3 наведено огляд аспекту розглянутих різних листів.

Таблиця 4.2 – Інформація про види рослин та кількість зразків

Клас	Назва	Кількість зразків
1	<i>Quercus suber</i>	12
2	<i>Salix atrocinera</i>	10
3	<i>Populus nigra</i>	10
4	<i>Alnus sp.</i>	8
5	<i>Quercus robur</i>	12
6	<i>Crataegus monogyna</i>	8
7	<i>Ilex aquifolium</i>	10
8	<i>Nerium oleander</i>	11
9	<i>Betula pubescens</i>	14
10	<i>Tilia tomentosa</i>	13
11	<i>Acer palmatum</i>	16
12	<i>Celtis sp.</i>	12
13	<i>Corylus avellana</i>	13
14	<i>Castanea sativa</i>	12
15	<i>Populus alba</i>	10
16	<i>Acer negundo</i>	10
17	<i>Taxus bacatta</i>	5

Продовження таблиці 4.2

Клас	Назва	Кількість зразків
18	<i>Papaver</i> sp.	12
19	<i>Polypodium vulgare</i>	13
20	<i>Pinus</i> sp.	12
21	<i>Fraxinus</i> sp.	10
22	<i>Primula vulgaris</i>	12
23	<i>Erodium</i> sp.	11
24	<i>Bougainvillea</i> sp.	13
25	<i>Arisarum vulgare</i>	9
26	<i>Euonymus japonicus</i>	12
27	<i>Ilex perado</i> ssp. <i>azorica</i>	11
28	<i>Magnolia soulangeana</i>	12
29	<i>Buxus sempervirens</i>	12
30	<i>Urtica dioica</i>	12
31	<i>Podocarpus</i> sp.	11
32	<i>Acca sellowiana</i>	11
33	<i>Hydrangea</i> sp.	11
34	<i>Pseudosasa japonica</i>	11
35	<i>Magnolia grandiflora</i>	11
36	<i>Geranium</i> sp.	10
37	<i>Aesculus californica</i>	10
38	<i>Chelidonium majus</i>	10
39	<i>Schinus terebinthifolius</i>	10
40	<i>Fragaria vesca</i>	11



Рисунок 4.3 – Аспекти розглянутих різних листів

Згідно складності листів в цій базі даних можна виділити дві великі групи: листи з класу 1 до 15 і 22 до 36 прості, а листи від класу від 16 до 21 і від 37 до 40 є складними. Для розробки нинішньої системи буде розглянуто лише прості листи, оскільки досліджені методики будуть неадекватними для опису форми та аналізу складних листів. Система може описувати будь-який замкнений контур, але варіативність форми та непередбачена кількість листових пластин у зразках будь-якого з цих класів дасть неточний результат. З іншого боку, при достатньому об'єму тестових даних та епох навчання нейронної мережі можна поліпшити точність результулювання розпізнавання.

У другому датасеті приведені лише прості листи, але враховуючи підвиди різних видів рослин. Їх назін та кількість листів наведені у таблиці 4.3.

Змін	Лист	№ докум.	Підпис	Дата

Таблиця 4.3 – Інформація про види рослин та кількість зразків

Клас	Назва	Кількість зразків
1	Acer_Campestre	16
2	Acer_Capillipes	16
3	Acer_Circinatum	16
4	Acer_Mono	16
5	Acer_Opalus	16
6	Acer_Palmatum	16
7	Acer_Pictum	16
8	Acer_Platanoids	16
9	Acer_Rubrum	16
10	Acer_Rufinerve	16
11	Acer_Saccharinum	16
12	Alnus_Cordata	16
13	Alnus_Maximowiczii	16
14	Alnus_Rubra	16
15	Alnus_Sieboldiana	16
16	Alnus_Viridis	16
17	Arundinaria_Simonii	16
18	Betula_Austrosinensis	16
19	Betula_Pendula	16
20	Callicarpa_Bodinieri	16
21	Castanea_Sativa	16
22	Celtis_Koraiensis	16
23	Cercis_Siliquastrum	16
24	Cornus_Chinensis	16
25	Cornus_Controversa	16
26	Cornus_Macrophylla	16
27	Cotinus_Coggygria	16
28	Crataegus_Monogyna	16
29	Cytisus_Battandieri	16
30	Eucalyptus_Glaucescens	16
31	Eucalyptus_Neglecta	16

Продовження таблиці 4.3

Клас	Назва	Кількість зразків
32	<i>Eucalyptus_Urnigera</i>	16
33	<i>Fagus_Sylvatica</i>	16
34	<i>Ginkgo_Biloba</i>	16
35	<i>Ilex_Aquifolium</i>	16
36	<i>Ilex_Cornuta</i>	16
37	<i>Liquidambar_Styraciflua</i>	16
38	<i>Liriodendron_Tulipifera</i>	16
39	<i>Lithocarpus_Cleistocarpus</i>	16
40	<i>Lithocarpus_Edulis</i>	16
41	<i>Magnolia_Heptapeta</i>	16
42	<i>Magnolia_Salicifolia</i>	16
43	<i>Morus_Nigra</i>	16
44	<i>Olea_Europaea</i>	16
45	<i>Phidelphus</i>	16
46	<i>Populus_Adenopoda</i>	16
47	<i>Populus_Grandidentata</i>	16
48	<i>Populus_Nigra</i>	16
49	<i>Prunus_Avium</i>	16
50	<i>Prunus_X_Shmittii</i>	16
51	<i>Pterocarya_Stenoptera</i>	16
52	<i>Quercus_Afares</i>	16
53	<i>Quercus_Agrifolia</i>	16
54	<i>Quercus_Alnifolia</i>	16
55	<i>Quercus_Brantii</i>	16
56	<i>Quercus_Canariensis</i>	16
57	<i>Quercus_Castaneifolia</i>	16
58	<i>Quercus_Cerris</i>	16
59	<i>Quercus_Chrysolepis</i>	16
60	<i>Quercus_Coccifera</i>	16
61	<i>Quercus_Coccinea</i>	16
62	<i>Quercus_Crassifolia</i>	16

Продовження таблиці 4.3

Клас	Назва	Кількість зразків
63	Quercus_Crassipes	16
64	Quercus_Dolicholepis	16
65	Quercus_Ellipsoidalis	16
66	Quercus_Greggii	16
67	Quercus_Hartwissiana	16
68	Quercus_Ilex	16
69	Quercus_Imbricaria	16
70	Quercus_Infectoria_sub	16
71	Quercus_Kewensis	16
72	Quercus_Nigra	16
73	Quercus_Palustris	16
74	Quercus_Phellos	16
75	Quercus_Phillyraeoides	16
76	Quercus_Pontica	16
77	Quercus_Pubescens	16
78	Quercus_Pyrenaica	16
79	Quercus_Rhysophylla	16
80	Quercus_Rubra	16
81	Quercus_Semecarpifolia	16
82	Quercus_Shumardii	16
83	Quercus_Suber	16
84	Quercus_Texana	16
85	Quercus_Trojana	16
86	Quercus_Variabilis	16
87	Quercus_Vulcanica	16
88	Quercus_x_Hispanica	16
89	Quercus_x_Turneri	16
90	Rhododendron_x_Russelianum	16
91	Salix_Fragilis	16
92	Salix_Intergra	16
93	Sorbus_Aria	16

### Продовження таблиці 4.3

Клас	Назва	Кількість зразків
94	Tilia_Oliveri	16
95	Tilia_Platyphylllos	16
96	Tilia_Tomentosa	16
97	Ulmus_Bergmanniana	16
98	Viburnum_Tinus	16
99	Viburnum_x_Rhytidophylloides	16
100	Zelkova_Serrata	16

Набіри з 340 та 1600 простих листів був випадковим чином розбитий на 70% тренувань та 30% випробувань, запевнивши, що принаймні один елемент кожного виду представлений у кожному підгрупі та відповідає структурі класу. Тестовий набір містить в середньому 3 листи відожної рослини, і він буде використовуватися для порівняння серед моделей.

### 4.3 Тестування розробленого програмного застосування на реальній тестовій виборці

Під час процесу тестування було проведено 9 тестів з малою вибіркою та 2 тести з великою. Тестування великої вибірки було проведено у розробленому мобільному застосуванні та за допомогою згорткової нейронної мережі архітектури InceptionV3 з використанням фреймворку Keras [13]. В процесі тестування була побудована таблиця 4.4

Таблиця 4.4 – Результати тестування

Розмір вибірки	25	80	25	80	25	80	1600	1600
Кількість класів	5	10	5	10	5	10	100	100
Кількість зображень у класі	5	8	5	8	5	8	16	16

Продовження таблиці 4.4

Час розпізнавання	1.3 с	2.9 с	1.2 с	3.4 с	1.6 с	3.5 с	4.2 с	2.6 хв
Час обробки зображень	87.5 с	280 с	87.5 с	280 с	87.5 с	280 с	5600 с	—
Час навчання нейронної мережі	3.7 с	5.6 с	8.9 с	22.17 с	18.6 с	42.45 с	612 с	~12 год
Кількість епох	100	100	500	500	1000	1000	200	200
Максимальна кількість токенів	308	908	308	908	308	908	607	—
Точність розпізнавання	90%	40%	80%	70%	70%	80%	70%	95%
Threshold	5	5	5	5	5	5	5	—
Distance	3	3	3	3	3	3	3	—
Min. line	3	3	3	3	3	3	3	—

Результати такого підходу можуть бути неточними внаслідок застосування одного значення параметрів Threshold, Distance та Min. Line для всієї виборки.

Точність розпізнавання (accuracy) є надійною і зрозумілою метрикою для closed-set ідентифікації, коли відомо, що шукана рослина є в базі. По суті, точність вимірює кількість разів, коли потрібна рослина була серед результатів пошуку. Усього було проведено 10 випробувань, для кожного випробування ми визначаємо зображення рослини, яку будемо шукати, і галерею, в якій будемо шукати, так, щоб галерея містила хоча б ще одне зображення цієї рослини. Ми переглядаємо перші десять результатів роботи алгоритму пошуку і перевіряємо, чи є серед них шукана рослина. Щоб отримати точність, слід підсумувати всі випробування, в яких шукана рослина була в результатах пошуку, і поділити на загальне число випробувань. Звісно при використанні згорткової нейронної мережі точність розпізнавання є набагато кращою, але час навчання і розпізнавання такої мережі незрівнянно довший ніж у розробленого застосування.

#### 4.4 Висновки до четвертого розділу

Дослідження розробленого алгоритму показало основні його переваги і недоліки за основними показниками. Розроблений алгоритм здатний за невеликий проміжок часу видати результат впливові результати показала достатньої точності. Зі збільшенням кількості вхідних даних, зменшується точність отриманого результату, тому для поліпшення потрібно збільшувати кількість епох та кількість листів в кожному класі тестової вибірки.

Під час дослідження були змодельовані ситуації з різною кількістю і набором вхідних даних. Тестування показало, що для отримання більш якісного результату потрібно вручну аналізувати якість обробки кожної фотографії листа та стежити за фільтруванням шуму. У навчальній виборці бажано використовувати фотографії зняті у ідеальних умовах (на контрастному однотонному фоні, для зелених листів це переважно червоний) та при фотографуванні тестового листа намагатися знизити рівень шуму та забезпечити однотонний фон та рівне освітлення.

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

## 5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Темою кваліфікаційного проекту є проблема створення методики нейромережової класифікації листів рослин для використання в мобільних пристроях, вирішення якої здійснюються працівниками–галузь біології та ботаніки, а саме їх здійснює агроном, біолог, біотехнолог, біофізик, біохімік, овочівник, озеленювач, рослинник, садівник, вчитель біології, флорист, квітникар, еколог тощо. Розробку методики та Android застосувань здійснює розробник програмного забезпечення.

Отже розглянемо питання охорони праці та безпеку у надзвичайних ситуаціях робоче місце розробника програмного забезпечення, яке розташоване у офісі розробників ПЗ.

### 5.1 Організація та управління охороною праці у фірми розробника програмного забезпечення у мобільних пристроях

Згідно з Законом України «Про охорону праці» на підприємстві створена система управління охороною праці для організації виконання правових, організаційно–технічних, санітарно–гігієнічних, соціально–економічних і лікувально–профілактичних заходів, спрямованих на запобігання нещасних випадків, професійних захворювань та аварій у процесі праці.

Організацію охорони праці на фірмі займається заступник директора з технічних питань. Безпосередньо охороною праці на підприємстві керує головний інженер і інженер з охорони праці. Забезпечується проведення інструктажів робітників і службовців з виробничої санітарії та техніки безпеки, а також контроль за додержанням працівниками вимог інструкцій з охорони праці. Використовуються наступні види інструктажів: вступний, первинний інструктаж на робочому місці, повторний, позаплановий та цільовий.

Усі заходи з охорони праці базуються на законодавчих і нормативних положеннях. Адміністрація для створення безпечних і нешкідливих умов праці

Змін	Лист	№ докум.	Підпис	Дата	ІС КРМ 122 036 ПЗ	Лист
						77

працівників керується переліком основних нормативно–законодавчих актів і документів з охорони праці. Відповідно проводиться аудит відповідності наявного переліку нормативно–правових актів з охорони праці існуючим законодавчим вимогам (табл. 4.1).

Таблиця 4.1 – Нормативно–правова база з питань охорони праці та безпеки в надзвичайних ситуаціях (станом на 1 листопада 2018 р.)

№	Нормативно–правові акти	Відмітка про наявність	Відповідність чинному законодавству
1.	Закон України «Про охорону праці»	+	відповідає
2.	Закон України «Про об’єкти підвищеної небезпеки»	+	відповідає
3.	Типове положення про службу охорони праці	+	відповідає
4.	Типове положення про навчання з питань охорони праці	+	відповідає
5.	Положення про розробку інструкцій з охорони праці	+	відповідає
6.	Положення про медичний огляд працівників певних категорій	+	відповідає
7.	Перелік посад, посадових осіб, які зобов’язані проходити попередню і періодичну перевірку знань з охорони праці	+	відповідає

Усю будівлю електрифіковано згідно з усіма відповідними нормами. Для швидкого та якісного виконання своїх службових обов’язків працівники користуються персональними комп’ютерами (ПК), телефонами, принтерами та іншою оргтехнікою.

## 5.2 Обґрунтування заходів з покращення умов праці

Умови праці співробітників відповідають існуючим санітарно–гігієнічним нормам. Але у зв’язку з тим, що більшу частину часу працівник займає сидячу позу та мало рухається, пропонується ввести п’ятихвилинну виробничу гімнастику, яку необхідно проводити після кожних 60 хвилин сидячої роботи, і яка буде спрямована на покращення фізичного та морального стану та самопочуття

Змін	Лист	№ докум.	Підпис	Дата

працівника. Також рекомендується забезпечувати комфортну площину робочого місця (РМ) згідно вимогам СанПіН 2.2.2./2.4.1340–03.

План і розположення РМ в офісі розробників програмного забезпечення (ПЗ) на підприємстві наведено на рисунку 4.2.

Варіант 03  
Офіс розробників ПЗ

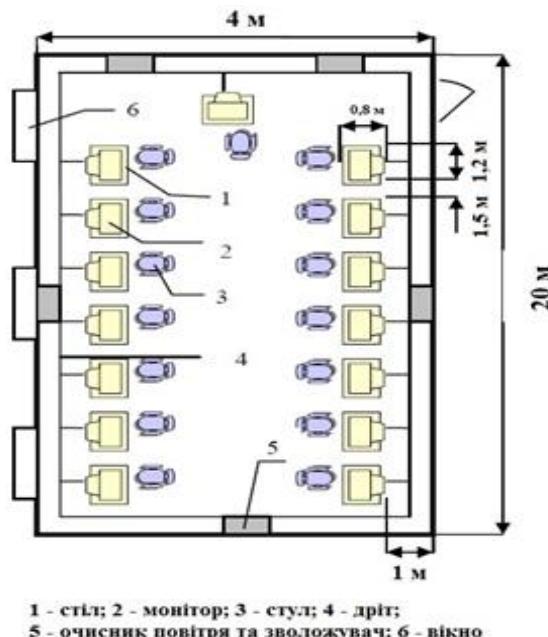


Рисунок 4.2 – План і розположення робочих місць

Проектування робочих місць, забезпечених ПК, відноситься до числа важливих проблем ергономічного проектування в області обчислювальної техніки. Проектується план і розположення РМ в офісі розробників програмного забезпечення, де встановлені ПК з рідкокристалічними моніторами.

Розраховується площа одного РМ, виходячи з площею офісного приміщення та кількості робочих місць.

Площа офісного приміщення розраховується за формулою:

$$S = a \cdot b, \quad (4.1)$$

де  $a$  – довжина приміщення;

$b$  – ширина приміщення.

Змін	Лист	№ докум.	Підпис	Дата

Отже площа офісного приміщення дорівнює:

$$S = 20 \cdot 4 = 80 \text{ м}^2$$

Кількість РМ дорівнює 15. Довжина стола дорівнює 1 м. Ширина стола дорівнює 0,8 м. Відстань між боковими сторонами екранів дорівнює 1,4 м. Так як у кожного працівника є сусід по РМ, то довжина площини його РМ по обидві сторони дорівнює 0,7 м. Відстань від стіни до тильної сторони екрану дорівнює 1 м. Довжина стула дорівнює 0,4 м.

Довжина РМ дорівнює сумі довжин відстаней між боковими сторонами екранів і довжиною стола. Таким чином довжина РМ дорівнює 2,4 м.

Ширина РМ дорівнює сумі довжин відстані від стіни до тильної сторони екрану, ширини стола та довжини крісла. Таким чином ширина РМ дорівнює 2,2 м.

Розраховується площа одного РМ за формулою:

$$S_{\text{РМ}} = a_{\text{РМ}} \cdot b_{\text{РМ}} \quad (4.2)$$

де  $a_{\text{РМ}}$  – довжина РМ;

$b_{\text{РМ}}$  – ширина РМ.

Отже площа одного РМ дорівнює:

$$S_{\text{РМ}} = 2,4 \cdot 2,2 = 5,28 \text{ м}^2$$

Загальна площа, відведенна для усіх РМ, розраховується за формулою:

$$S_{\text{загальна}} = S_{\text{РМ}} \cdot N \quad (4.3)$$

де  $S_{\text{РМ}}$  – площа одного РМ;

$N$  – кількість РМ.

Отже загальна площа усіх РМ дорівнює:

$$S_{\text{загальна}} = 5,28 \cdot 15 = 79,2 \text{ м}^2$$

Значення площини  $S_{\text{РМ}}$  використовує майже усю площу приміщення, що робить не можливим зміну параметрів РМ для збільшення плоши РМ. Так як площа на одне РМ користувача ПК на базі електронно–лучової трубки повинна мати не менше 6 м<sup>2</sup>, то значення  $S_{\text{РМ}} = 5,28 \text{ м}^2$  не відповідає вимогам СанПіН 2.2.2./2.4.1340–03. Тому рекомендується зробити зменшити кількість РМ в офісі розробників ПЗ для забезпечення комфорних умов праці.

Змін	Лист	№ докум.	Підпис	Дата

У таблиці 4.2 наведений перелік шкідливих речовин у повітрі приміщення, допустима концентрація та клас небезпеки.

Таблиця 4.2 – Допустима концентрація та клас небезпеки шкідливих речовин

Найменування речовини	Концентрація	Клас небезпеки
Вміст кислороду	21–22	—
Діоксид углерода, об.%	Не більше 0,1	IV
Озон, мг/м <sup>3</sup>	Не більше 0,03	I

### 5.3 Індивідуальне завдання. Розрахунок захисного заземлення

Розраховується захисне заземлення електроустановки потужністю до 1000В в мережі з ізольованою нейтраллю. Для здійснення заземлюючих функцій опір заземлюючого пристрою повинен бути не більше 4 Ом. Використовується вертикальний заземлювач з розміщенням по контору. Вихідні дані згідно 3 варіанту наведені у таблиці 4.3.

Таблиця 4.3 – Вихідні дані для розрахунку заземлювання

Варіант	р, Ом*м	z	Вертикальний заземлювач			Горизонтальний заземлювач		
			l1, м	d1, м	t1, м	l2, м	d2, м	t2, м
3	300,00	1,00	2,50	0,05	2,40	2,50	0,05	1,40

При використанні заземлення для забезпечення електробезпеки технічних систем повинна виконуватися умова:

$$R_{\text{гр}} \leq R_{\text{з.у.доп}} \quad (4.4)$$

де  $R_{\text{з.у.доп}}$  – допустимий опір заземлюючого пристрою в електроустановках;

$R_{\text{гр}}$  – груповий опір заземлюючих стержнів і з'єднаючої полоси, Ом.

Для розрахунку  $R_{\text{гр}}$  необхідно використовувати наступну формулу:

$$R_{\text{гр}} = \frac{R_{\text{в}} \cdot R_{\text{г}}}{R_{\text{в}} \cdot \eta_{\text{г}} + R_{\text{г}} \cdot \eta_{\text{в}} \cdot n_{\text{табл}}}, \quad (4.5)$$

де  $R_{\text{в}}, R_{\text{г}}$  – відповідно опір вертикального стержня та горизонтальної полоси;

$\eta_B$ ,  $\eta_G$  – відповідно коефіцієнти використання вертикальних стержнів і горизонтальної полоси, Ом;

$n_{\text{табл}}$  – кількість вертикальних стержнів обраних зі стандартного ряду.

Розрахунок  $R_B$ , Ом – опір заземлювача у вигляді стержня, круглого січення (трубчатого) та заглибленого в землю повністю виконується по формулі:

$$R_B = \frac{p}{2\pi \cdot l_1} \cdot \left( \ln\left(\frac{2 \cdot l_1}{d_1}\right) + \frac{1}{2} \cdot \ln\left(\frac{4 \cdot t_1 + l_1}{4 \cdot t_1 - l_1}\right) \right), \quad (4.6)$$

де  $p$  – питомий опір ґрунту, Ом · м;

$l_1$  – довжина вертикального стержня, м;

$d_1$  – діаметр перетину, м;

$t_1$  – заглиблення заземлювача (відстань від поверхні ґрунту до середини заземлювача, м).

Розрахунок  $R_G$ , Ом – опір протяжного заземлювача (стержень, труба, полоса, кабель і т.д.) заглибленого в землю виконується по формулі:

$$R_G = \frac{p}{2\pi \cdot L_{\text{с.п.}}} \cdot \ln\left(\frac{2L_{\text{с.п.}}^2}{d_2 \cdot t_2}\right), \quad (4.7)$$

де  $p$  – питомий опір ґрунту, Ом · м;

$L_{\text{с.п.}}$  – довжина горизонтальної полоси, м;

$d_2$  – ширина полоси або діаметр перетину стержня (труби), м;

$t_2$  – відстань від поверхні ґрунту до середини ширини горизонтальної полоси, м.

Довжина сполучної смуги розраховується за формулою:

$$L_{\text{с.п.}} = a \cdot n_{\text{табл}}, \quad (4.8)$$

де  $n_{\text{табл}}$  – кількість вертикальних стержнів;

$a$  – відстань між вертикальними стержнями, м.

Відстань між вертикальними стержнями, м, визначається за формулою:

$$a = z \cdot l_1, \quad (4.9)$$

де  $l_1$  – довжина вертикального стержня;

$z$  – коефіцієнт кратності, рівний 1,2,3.

Попередня кількість вертикальних заземлювачів, шт., визначається з виразу:

Змін	Лист	№ докум.	Підпис	Дата

$$n_{\text{попер}} = R_B / [4], \quad (4.10)$$

де  $R_B$  – опір розтікання струму одиночного вертикального заземлювача, Ом;

Розраховується опір вертикального заземлювача,  $R_B$ , Ом за формулою 4.6:

$$R_B = \frac{300}{2\pi \cdot 2,5} \cdot \left( \ln \frac{2 \cdot 2,5}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,4 + 2,5}{4 \cdot 2,4 - 2,5} \right) = 92,92 \text{ Ом}$$

Визначається попередня кількість вертикальних заземлювачів, шт., з формулі 4.10:

$$n_{\text{попер}} = 92,92 / [4] = 23,23 \text{ шт}$$

Таким чином  $n_{\text{табл}} = 20$  шт. зі стандартного ряду.

Визначається відстань між вертикальними стержнями за формулою 4.9:

$$a = 1 \cdot 2,5 = 2,5 \text{ м}$$

Визначається довжина горизонтального заземлювача  $L_{\text{с.п.}}$  за формулою 4.8:

$$L_{\text{с.п.}} = 2,5 \cdot 20 = 50 \text{ м}$$

Визначається опір горизонтального заземлювача  $R_g$  за формулою 4.7:

$$R_g = \frac{300}{2\pi \cdot 50} \cdot \ln \left( \frac{2 \cdot 50^2}{0,05 \cdot 1,4} \right) = 10,61 \text{ Ом}$$

Визначаються коефіцієнти використування заземлювачів  $\eta_B$ ,  $\eta_g$  в залежності від кількості заземлювачів. Відношення відстаней між електродами до їх довжини  $a / l = 2,5 / 2,5 = 1$ . Таким чином  $\eta_B = 0,47$  і  $\eta_g = 0,27$  для  $n_{\text{табл}} = 20$  шт.

Розраховується опір групового заземлювача  $R_{\text{гр}}$  за формулою 4.5:

$$R_{\text{гр}} = \frac{92,92 \cdot 10,61}{92,92 \cdot 0,27 + 10,61 \cdot 0,47 \cdot 20} = 7,9 \text{ Ом}$$

$R_{\text{гр}} > 4$  Ом. Отже необхідно збільшити кількість вертикальних стержнів до наступного табличного ряду  $n_{\text{табл}} = 40$ . Таким чином  $\eta_B = 0,41$  і  $\eta_g = 0,22$  для  $n_{\text{табл}} = 40$  шт.

$$R_{\text{гр}} = \frac{92,92 \cdot 10,61}{92,92 \cdot 0,22 + 10,61 \cdot 0,41 \cdot 40} = 5,07 \text{ Ом}$$

$R_{\text{гр}} > 4$  Ом. Отже необхідно збільшити кількість вертикальних стержнів до наступного табличного ряду  $n_{\text{табл}} = 50$ . Таким чином  $\eta_B = 0,4$  і  $\eta_g = 0,21$  для  $n_{\text{табл}} = 50$  шт.

Змін	Лист	№ докум.	Підпис	Дата

$$R_{\text{гр}} = \frac{92,92 \cdot 10,61}{92,92 \cdot 0,21 + 10,61 \cdot 0,4 \cdot 50} = 4,25 \text{ Ом}$$

$R_{\text{гр}} > 4 \text{ Ом}$ . Отже необхідно збільшити кількість вертикальних стержнів до наступного табличного ряду  $n_{\text{табл}} = 60$ . Таким чином  $\eta_{\text{в}} = 0,39$  і  $\eta_{\text{г}} = 0,2$  для  $n_{\text{табл}} = 60$  шт.

$$R_{\text{гр}} = \frac{92,92 \cdot 10,61}{92,92 \cdot 0,2 + 10,61 \cdot 0,39 \cdot 60} = \frac{985,88}{266,85} = 3,69 \text{ Ом}$$

$R_{\text{гр}} < 4 \text{ Ом}$ . Таким чином для забезпечення електробезпеки технічної системи необхідно використовувати 60 вертикальних заземлюючих стержнів.

#### 5.4 Надзвичайні ситуації та шляхи їх запобігання

Для дослідження безпеки в надзвичайних ситуаціях на ТОВ «Телекарт–прилад» охарактеризується приміщення з погляду пожежної безпеки. В приміщенні використовуються тільки негорючі речовини та матеріали у холодному стані, за ступенем вибухопожежної та пожежної небезпеки приміщення офіс розробників відноситься до категорії «Д». Пожежну небезпеку несуть у собі лише кабельні електропроводки до обладнання, що є припустимим для даної категорії приміщень.

За проходження практики на ТОВ «Телекар–прилад» було розглянуто головні можливі причини виникнення пожежі у приміщеннях такі як:

- несправна електропроводка (іскріння, перегрів провідників, пересихання електроізоляційних матеріалів);
- використання електропобутових пристрійв; попадання води на працюючі електроагрегати;
- Для покращення техногенної та пожежної безпеки пропонується наступний ряд заходів:
  - щорічне проведення повторних протипожежних інструктажів та занять за програмою пожежно–технічного мінімуму з особами, що відповідають за пожежну безпеку;

Змін	Лист	№ докум.	Підпис	Дата

- утримання в справному стані засоби протипожежного захисту та зв'язку, обладнання та інвентар, не допускати їх використання не за призначенням;
- своєчасне інформування пожежної охорони про несправність пожежної техніки, систем протипожежного захисту, водопостачання тощо.

### 5.5 Індивідуальне завдання. Безпека в надзвичайної ситуації радіоактивного забруднення

Розглядається нештатна ситуація на підприємстві, при якій створено радіоактивне забруднення. Тому визначається доза опромінення та рівень радіації в зоні радіоактивного забруднення. Вихідні дані до задачі згідно варіанту 3 наведені у таблиці 4.4.

Таблиця 4.4 – Вихідні дані до задачі

Вихідні дані	Значення
Час роботи в осередку р/з $t_{\text{роб}}$	8 годин
Час початку роботи після аварії $t_H$	3 години
Рівень радіації $P_H$	4 рад/год
Коефіцієнт послаблення $K_{\text{посл}}$	10

В ході вирішення задачі визначається час кінця роботи за формулою:

$$t_K = t_H + t_{\text{роб}}, \quad (4.11)$$

де  $t_H$  – час початку роботи після аварії;

$t_{\text{роб}}$  – час роботи у осередку радіоактивного забруднення.

Таким чином час кінця роботи дорівнює:

$$t_K = 3 + 8 = 11 \text{ годин}$$

Рівень радіації на 11 годину після аварії знаходиться за формулою:

$$D = \frac{P_H \cdot K_K}{K_t}, \quad (4.12)$$

де  $K_K = 0,385$  – коефіцієнт кінця роботи;

$K_t = 0,645$  – коефіцієнт початку роботи.

З формули 4.12 розраховується рівень радіації:

Змін	Лист	№ докум.	Підпис	Дата

$$P_K = \frac{4 \cdot 0,385}{0,645} = 2,38 \frac{\text{рад}}{\text{год}}$$

Далі визначається доза опромінення за 8 годин за формулою:

$$D = \frac{1,7 (P_K \cdot t_K - P_H \cdot t_H)}{K_{\text{посл}}} \quad (4.13)$$

$$D = \frac{1,7 (2,38 \cdot 11 - 4 \cdot 3)}{10} = 2,52 \text{ рад}$$

Таким чином людина під час 8-ми годинної праці в осередку радіоактивного забруднення отримує дозу опромінення  $D = 2,52$  рад при коефіцієнті послаблення  $K_{\text{посл}} = 10$ .

## 5.6 Заходи з охорони праці та безпеки у надзвичайних ситуаціях

Розглянуто стан охорони праці в офісі розробників програмного забезпечення. Виявлено, що приміщення підприємства відповідає як внутрішнім документам з питань охорони праці, так і нормативно–законодавчим актам. Щодо умов праці співробітника, то вони відповідають існуючим санітарно–гігієнічним нормам, за винятком ненормованого режиму праці та малорухомого характеру роботи. Приміщення офісу розробників по категорії вибухо– і пожежонебезпечності та ступеню вогнестійкості відповідає нормам, але особливу увагу потрібно звернути на утримання в справному стані засобів протипожежного захисту та своєчасне інформування пожежної охорони про несправність пожежної техніки, впровадження систем протипожежного захисту. Організаційні та технічні заходи, спрямовані на попередження виникнення пожежі, обмеження поширення вогню та успішної евакуації людей розроблені та реалізовані належним чином.

Загалом, до стану охорони праці та безпеки в надзвичайних ситуаціях відносяться відповідально. Безпосередні обов'язки із забезпечення належної охорони праці співробітників підприємства покладені на заступника директора з технічних питань за сумісництвом. На підприємстві за час його діяльності не зафіксована нещасних випадків та випадків виробничого травматизму. Санітарно–гігієнічні, будівельні, пожежні норми неухильно дотримуються.

Змін	Лист	№ докум.	Підпис	Дата

Загалом рекомендується:

- збільшити кількість перевірок стану виробничих машин, механізмів і ПК, приведених у відповідальність до сучасних вимог стандартів безпеки охорони праці для підвищення загального рівня безпеки на виробництві;
- зробити погодинний графік роботи працівника з значенням часу роботи та відпочинку для зменшення рівня втоми та монотонності праці;
- раціонально компонувати робочі місця в офісі розробників ПЗ для поліпшення естетичних показників і збільшення рівня комфорту;
- надавати премії робітникам відділів і цехів за тривалу роботу без порушень правил охорони праці, без травм або аварій. У випадку наявності небезпечних і шкідливих виробничих чинників, що постійно загрожують здоров'ю працівника, рекомендується виплачувати надбавку за підвищенну обережність.

Пропонується керівництву підприємства звернути увагу на забезпечення постійного контролю за дотриманням правил безпеки працівниками при роботі з оргтехнікою, сприяти встановленню короткочасних додаткових перерв під час роботи з комп'ютерами, забезпечити додаткову вентиляцію та теплообмін в адміністративних приміщеннях.

### 5.7 Висновки до п'ятого розділу

На досліджуваному підприємстві провели аналіз умов праці, апаратури і обладнання з точки зору можливості появи небезпечних факторів, виділення шкідливих виробничих речовин, який показав, що умови праці, в яких, знаходяться працівники, не завжди відповідають нормативним, але в цілому по підприємству певного впливу на здоров'я і працездатність робітника не надають.

У кваліфікаційній роботі були розглянуті теоретичні основи охорони праці на підприємствах електроенергетики. Для розкриття цих основ були розглянуті актуальні питання охорони праці для підприємств електроенергетики: умови праці; шкідливі і небезпечні виробничі фактори в електроенергетиці; професійна

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

захворюваність. В електроенергетиці шкідливими і небезпечними виробничими факторами є: фізичні: виробничий шум (постійний, непостійний); вібрація; інфразвук (Постійний, непостійний); ультразвук; мікроклімат (температура повітря, тиск; відносна вологість повітря; швидкість руху повітря; теплове випромінювання; і т.д.); світлове середовище; неіонізуючі електромагнітні поля і випромінювання (електромагнітні поля і випромінювання ВДТ і ПЕОМ; електростатичне поле; електромагнітні поля і випромінювання промислової частоти 50 Гц; постійне магнітне поле; електромагнітні випромінювання радіочастотного діапазону; лазерне випромінювання; ультрафіолетове випромінювання; теплове випромінювання); хімічні: хімічні речовини, суміші, хімічні речовини біологічної природи;

Але керівництву підприємства звернути увагу на забезпечення постійного контролю за дотриманням правил безпеки працівниками при роботі з оргтехнікою, сприяти встановленню короткочасних додаткових перерв під час роботи з комп'ютерами, забезпечити додаткову вентиляцію та теплообмін в адміністративних приміщеннях.

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

## ВИСНОВКИ

У даній кваліфікаційній роботі розглядається одна з проблем швидкої класифікації рослин, яка відомі здавна та повністю не вирішена до сьогоднішнього часу внаслідок великої кількості різноманітних рослин, складності та схожості листків та індивідуальними особливостями виду.

У першому розділі було з'ясовано, що найбільш повну картину о класі можливо отримати, аналізуючи форму листів рослин. Для виділення форми використовуються методи виділення контурів на зображення. Для задачі виділення контуру листів по якості виділених контурів найбільш ефективно показав себе оператор Кенни, але по швидкості обробки зображення найкраще показав себе метод Прюітт і було вирішено використовувати його.

У другому розділі розроблена методика аналізу форми листа для формування признового простору з таких номінальних признаків, як  $\sin \theta$  і  $\cos \theta$  першого гострого кута прямокутного трикутника геометрії листа, у якого лінія спрощеної геометрії листа виступає гіпотенузою, а катетами є дві лінії, проведенні паралельно краям зображення до їх перетину.

Після цього стане можливим знаходження кута між лініями форми листа. Для отримання повної картини форми листа знайдемо  $\sin \theta$  і  $\cos \theta$ , це дозволить побудувати ознаку, яка характеризує довжину лінії, напрям та кут її нахилу.

Після виділення ознак стає можливим подати ці ознаки на вхід нейронної мережі для навчання і подальшого тестування на тестовій вибірці. Вхід функції активації нейрона визначається зміщенням і сумою зважених входів. Вихід нейрона залежить як від входів нейрона, так і від виду функції активації. Один нейрон не може вирішувати складні завдання, проте кілька нейронів, об'єднаних в один або кілька шарів, мають більші можливості.

Архітектура мережі складається з опису того, скільки шарів має мережу, кількості нейронів в кожному шарі, виду функції активації кожного шару та інформації про з'єднання шарів. Архітектура мережі залежить від тієї конкретної задачі, яку повинна вирішувати мережу.

Змін	Лист	№ докум.	Підпись	Дата
------	------	----------	---------	------

На етапі реалізації було створено мобільне застосування для класифікації листів рослин. Кожна нова версія коду була створена за допомогою системи контролю версій.

Для контролю за якістю системи було проведено набір тестів, що допоміг перевірити помилки та можливість системи працювати під навантаженням.

Проведені дослідження показали, що розроблена методика має переваги перед порівнювальними. Найбільш кращих результатів методика показала при обробленому датасеті з 1600 зображень, працюючи в рази швидші, ніж аналогічна програма, працююча за допомогою згорткової нейронної мережі архітектури InceptionV3 з використанням фреймворку Keras [13]. Звісно при використанні згорткової нейронної мережі точність розпізнавання є кращою, але час навчання і розпізнавання такої мережі незрівнянно довший ніж у розробленого застосування.

Таким чином всі задачі виконані, мета роботи досягнута.

Результати, що наведені у кваліфікаційній роботі, представлено на конференції «MIT-2018» [17].

Змін	Лист	№ докум.	Підпис	Дата

## ПЕРЕЛІК ПОСИЛАНЬ

1. Г. Буч Объектно–ориентированный анализ и проектирование с примерами приложений на Android // Буч Г. – СПб: «Невский Диалект», 1998 г. – 632с.
2. Б. Страуструп Язык программирования Java, 3–е издание // Страуструп Б. – М.: «Невский Диалект» – «Издательство БИНОМ», 1999 г. – 991 с.
3. Г. Кейт Использование Android 6. Специальное издание. // Кейт Г. – К.: Издательский дом «Вильямс», 1999 г. – 864 с.
4. Доронина Ю. В. Конспект лекций по ООП
5. Алгоритм зворотного поширення помилки [Электронний ресурс] – Режим доступу: <http://www.aiportal.ru/articles/neural-networks/back-propagation.html>
6. Lasty P., Sebastiani F. Determining the Semantic Orientation of Terms through Gloss Classification // Conference of Information and Knowledge Management (Bremen). ACM, New York, 2005, pp. 617–624.
7. Карамазина М.Н. Методы обработки изображений. Карамазина М.Н., Давыдов / РОМИП 2011.
8. Пуховой Р.И. Программа анализа поиска границ объектов // Дипломная работа, М. 2013, 9 – 17.
9. UCI Machine Learning Repository [Электронний ресурс] – Режим доступу: <https://archive.ics.uci.edu/ml/datasets/leaf>
10. Алгоритм навчання багатошарової нейронної мережі методом зворотнього Поширення помилки [Электронний ресурс] – Режим доступу: <https://habr.com/post/198268/>
11. Herbert Freeman. Computer processing of line-drawing images. ACM Comput. Surv., 6(1):57–97, March 1974.
12. Rafael C. Gonzalez and Richard E. Woods. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2006.
13. Создаём нейронную сеть InceptionV3 для распознавания изображений [Электронний ресурс] – Режим доступу: <https://habr.com/post/321834/>

Змін	Лист	№ докум.	Підпис	Дата

14. Тимонин А.К. Ботаника. Том 4. Систематика высших растений. Книга 2  
Учебник для студ. высш. учеб. заведений. Издательский центр "Академия".  
2009.— 352 с
15. Ботаника. Андреева И.И., Родман Л.С. 2 – е изд., перераб. и доп. – М.:  
КолосС, 2002.— 488 с.
16. Биология. Современная иллюстрированная энциклопедия л. ред. Горкин А.  
П. – М.: Росмэн–Пресс, 2006. – 560 с. (Серия: Современная  
иллюстрированная энциклопедия.)
17. Методика нейросетевой классификации листьев растений на основе методов  
обработки и анализа изображений: матеріали восьмої міжнародної наукової  
конференції студентів та молодих вчених “Сучасні Інформаційні Технології  
2018” – Іщенко А.Д, Годовиченко Н. А. – 156-158 с.

Змін	Лист	№ докум.	Підпис	Дата
------	------	----------	--------	------

ІС КРМ 122 036 ПЗ

Лист

92

## ДОДАТОК А

Код основних класів:

```
class FindTokensTask extends AsyncTask<Void, String, Void> {
    private AbstractLeafClassifierFragment fragment;
    private View view;
    private TextView progressText;
    private ProgressBar progressBar;
    private FancyButton findTokens;
    private LeafImage leafImage;

    BiConsumer<String, String> publishProgress = (s, s2) ->
    publishProgress(s, s2);

    public FindTokensTask(AbstractLeafClassifierFragment fragment, View view,
    LeafImage leafImage) {
        this.fragment = fragment;
        this.view = view;
        this.leafImage = leafImage;

        this.findTokens = view.findViewById(R.id.findTokens);
        this.progressText = view.findViewById(R.id.progressText);
        this.progressBar = view.findViewById(R.id.progressBar);
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected Void doInBackground(Void... params) {
        // first we disable the button
        fragment.getActivity().runOnUiThread(() ->
    findTokens.setEnabled(false));
        FindTokensUtils.findTokens(view, publishProgress, leafImage,
    fragment);

        fragment.getActivity().runOnUiThread(() ->
    findTokens.setEnabled(true));

        TextView nameText = view.findViewById(R.id.nameText);
        TextView sizeText = view.findViewById(R.id.sizeText);
        TextView tokensText = view.findViewById(R.id.tokensText);
        fragment.getActivity().runOnUiThread(() -> updateLeafInfo(leafImage,
nameText, sizeText, tokensText));
        return null;
    }

    @Override
    protected void onProgressUpdate(String... values) {
        super.onProgressUpdate(values);
        progressText.setText(values[0]);
        progressBar.setProgress(Integer.parseInt(values[1]));
    }

    @Override
    protected void onPostExecute(Void result) {
        super.onPostExecute(result);
    }
}
```

```

private void updateLeafInfo(LeafImage leafImageForRecognizing, TextView
nameText, TextView sizeText, TextView tokensText){
    nameText.setText(leafImageForRecognizing.getFileName().getName());
    sizeText.setText(String.valueOf(leafImageForRecognizing.getFileName().length()
) + " " + fragment.getString(R.string.BYTES));
    tokensText.setText(String.valueOf(leafImageForRecognizing.numTokens()));
    nameText.setText(leafImageForRecognizing.getFileName().getName());
}
}

public class FindTokensUtils {

    public static void findTokens(View view, BiConsumer<String, String>
publishProgress, LeafImage leafImage, AbstractLeafClassifierFragment
fragment){

        publishProgress.accept("Load Image...", "5");
        // Now we perform the Image processing
        ImageProcessor imgProc = new ImageProcessor(leafImage.getImage(),
view.getContext());

        SeekBar threshold = view.findViewById(R.id.threshold);
        SeekBar distance = view.findViewById(R.id.distance);
        SeekBar minLine = view.findViewById(R.id.minLine);
        ImageView imageView = view.findViewById(R.id.imageView);

        publishProgress.accept("Edge detection...", "10");
        imgProc.edgeDetect(threshold.getProgress() * 10);

        publishProgress.accept("Thinning...", "20");

        imgProc.thinning();
        publishProgress.accept("Line checking...", "40");

        imgProc.checkLines(minLine.getProgress() * 10);
        publishProgress.accept("Distance points...", "60");

        imgProc.markPoints(distance.getProgress() * 10);
        publishProgress.accept("Searching tokens...", "70");

        // now we calculate the tokens of the image by calculating
        // the angles
        imgProc.calcAngels();

        // set the TextField for the amount of tokens
        ArrayList leafTokens = imgProc.getTokens();
        leafImage.setTokens(leafTokens);

        Bitmap bitmap =
ImageUtils.convertFromImageToBitmap(imgProc.getImage());
        leafImage.setImageBitmap(bitmap);
        fragment.getActivity().runOnUiThread(() ->
imageView.setImageBitmap(bitmap));
        publishProgress.accept("finished.", "100");
    }
}

public class ImageUtils {
    private ImageUtils()
    {}
}

```

```

public static Bitmap convertFromImageToBitmap(Image image)
{
    BufferedImage bufferedImage = ImageProcessor.toBufferedImage(image);

    ByteArrayOutputStream baos = new ByteArrayOutputStream();
    try {
        ImageIO.write(bufferedImage, "png", baos);
        baos.flush();
        byte[] imageInByte = baos.toByteArray();
        return BitmapFactory.decodeByteArray(imageInByte, 0,
imageInByte.length);
    } catch (IOException e) {
        e.printStackTrace();
    }
    return null;
}
}

<**
 * Created by Artem on 08.04.2018.
 */

public class LeafAdapter extends BaseExpandableListAdapter {

    private List<LeafSpaceContainer> spaces;
    private Context mContext;
    private ExpandableListView view;
    private AbstractLeafClassifierFragment fragment;

    public LeafAdapter(AbstractLeafClassifierFragment fragment,
List<LeafSpaceContainer> spaces, ExpandableListView view) {
        this.fragment = fragment;
        this.mContext = fragment.getContext();
        this.spaces = spaces;
        this.view = view;
    }

    @Override
    public int getGroupCount() {
        return spaces.size();
    }

    @Override
    public int getChildrenCount(int groupPosition) {
        return spaces.get(groupPosition).getLeafsName().size();
    }

    @Override
    public Object getGroup(int groupPosition) {
        return spaces.get(groupPosition);
    }

    @Override
    public Object getChild(int groupPosition, int childPosition) {
        return spaces.get(groupPosition).getLeafsName().get(childPosition);
    }

    @Override
    public long getGroupId(int groupPosition) {
        return groupPosition;
    }
}

```

```

@Override
public long getChildId(int groupPosition, int childPosition) {
    return childPosition;
}

@Override
public boolean hasStableIds() {
    return true;
}

@Override
public View getGroupView(int groupPosition, boolean isExpanded, View convertView,
                         ViewGroup parent) {

    if (convertView == null) {
        LayoutInflator inflater = (LayoutInflator)
mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        convertView = inflater.inflate(R.layout.group_view, null);
    }

    convertView.setOnClickListener(v -> {
        if (isExpanded) {
            view.collapseGroup(groupPosition);
        } else {
            view.expandGroup(groupPosition);
        }
    });
}

TextView textGroup = convertView.findViewById(R.id.textGroup);
textGroup.setText(spaces.get(groupPosition).getSpaceName());
TextView classText = fragment.getView().findViewById(R.id.classText);

Button addInSpace = convertView.findViewById(R.id.add_in_space);

addInSpace.setOnClickListener(e) -> {
    PickImageDialog.newInstance(new PickSetup()).setOnPickResult(r ->
{
    LeafImage leafImage = new LeafImage(r.getBitmap(),
r.getPath());
    List<String> path = Arrays.asList(r.getPath().split("/"));

    spaces.get(groupPosition).getLeafsName().add(path.get(path.size() - 1));
    LeafSpecies leafSpecies =
LeafClassifier.getProjectEnv().getLeafSpecies().get(groupPosition);
    leafSpecies.addImage(leafImage);
    leafImage.setSpecies(leafSpecies);
    new FindTokensTask(fragment, fragment.getView(),
leafImage).execute();
    classText.setText(leafImage.getSpecies().getName());
    this.notifyDataSetChanged();
    LeafClassifier.getProjectEnv().setModified();

}).show(fragment.getFragmentManager());
});

Button editSpace = convertView.findViewById(R.id.edit_space);

editSpace.setOnClickListener(e) -> {
    EditText classNameField = new EditText(fragment.getContext());
    new AlertDialog.Builder(fragment.getContext())
        .setTitle("Class name")

```

```

        .setMessage("Please enter a class name:")
        .setView(classNameField)
        .setPositiveButton("Enter", (dialog, whichButton) -> {
            String spaceName =
            classNameField.getText().toString();
            LeafSpecies lSpecies =
            LeafClassifier.getProjectEnv().getLeafSpecies().get(groupPosition);
            lSpecies.setName(spaceName);
            spaces.get(groupPosition).setSpaceName(spaceName);
            this.notifyDataSetChanged();
            LeafClassifier.getProjectEnv().setModified();
        })
        .setNegativeButton("Cancel", (dialog, whichButton) -> {
        })
        .show();
    });

    Button deleteSpace = convertView.findViewById(R.id.delete_space);

    deleteSpace.setOnClickListener((e) -> {
        new AlertDialog.Builder(fragment.getContext())
            .setTitle("Confirm")
            .setMessage("Are you sure? that want to delete " +
textGroup.getText() + "?")
            .setPositiveButton("Enter", (dialog, whichButton) -> {

LeafClassifier.getProjectEnv().getLeafSpecies().remove(groupPosition);
            spaces.remove(groupPosition);
            this.notifyDataSetChanged();
            LeafClassifier.getProjectEnv().setModified();
        })
            .setNegativeButton("Cancel", (dialog, whichButton) -> {
        })
            .show();
    });

    return convertView;
}

private static LeafImage currentLeaf = null;

@Override
public View getChildView(int groupPosition, int childPosition, boolean
isLastChild,
                        View convertView, ViewGroup parent) {
    if (convertView == null) {
        LayoutInflator inflater = (LayoutInflator)
mContext.getSystemService(Context.LAYOUT_INFLATER_SERVICE);
        convertView = inflater.inflate(R.layout.child_view, null);
    }

    TextView textChild = convertView.findViewById(R.id.textChild);
    String imageName =
spaces.get(groupPosition).getLeafsName().get(childPosition);
    textChild.setText(imageName);

    ImageView imageView =
fragment.getView().findViewById(R.id.imageView);
    TextView classText = fragment.getView().findViewById(R.id.classText);
    TextView nameText = fragment.getView().findViewById(R.id.nameText);
    TextView sizeText = fragment.getView().findViewById(R.id.sizeText);
    TextView tokensText =

```

```

fragment.getView().findViewById(R.id.tokensText);
    convertView.setOnClickListener(v -> {
        LeafImage leaf =
LeafClassifier.getProjectEnv().getLeafSpecies().get(groupPosition).getImage(c
hildPosition);
        Bitmap image = leaf.getBitmap(mContext);
        updateLeafInfo(leaf, nameText, sizeText, tokensText, classText);
        currentLeaf = leaf;
        imageView.setImageBitmap(image);
    });

Button deleteLeaf = convertView.findViewById(R.id.delete_leaf);

deleteLeaf.setOnClickListener(e) -> {
    new AlertDialog.Builder(fragment.getContext())
        .setTitle("Confirm")
        .setMessage("Are you sure? that want to delete " +
textChild.getText() + "?")
        .setPositiveButton("Enter", (dialog, whichButton) -> {

LeafClassifier.getProjectEnv().getLeafSpecies().get(groupPosition).getImages(
).remove(childPosition);

spaces.get(groupPosition).getLeafsName().remove(childPosition);
    this.notifyDataSetChanged();
    LeafClassifier.getProjectEnv().setModified();
}
        .setNegativeButton("Cancel", (dialog, whichButton) -> {
})
        .show();
});

FancyButton findTokens =
fragment.getView().findViewById(R.id.findTokens);

findTokens.setOnClickListener(e) -> {
    if (currentLeaf == null)
    {
        return;
    }
    new FindTokensTask(fragment, fragment.getView(),
currentLeaf).execute();
}

return convertView;
}

@Override
public boolean isChildSelectable(int groupPosition, int childPosition) {
    return true;
}

private void updateLeafInfo(LeafImage leafImageForRecognizing, TextView
nameText, TextView sizeText, TextView tokensText, TextView classText){
    nameText.setText(leafImageForRecognizing.getFileName().getName());

sizeText.setText(String.valueOf(leafImageForRecognizing.getFileName().length(
)) + " " + fragment.getString(R.string.BYTES));

tokensText.setText(String.valueOf(leafImageForRecognizing.numTokens()));
    nameText.setText(leafImageForRecognizing.getFileName().getName());
    classText.setText(leafImageForRecognizing.getSpecies().getName());
}

```

```

        }

    }

public class LeafLibraryFragment extends AbstractLeafClassifierFragment {

    static final String ARGUMENT_PAGE_NUMBER = "arg_page_number";
    public static final String title = "Leaf Library";

    int pageNumber;
    ExpandableListView treePanel;

    static LeafLibraryFragment newInstance(int page) {
        LeafLibraryFragment leafLibraryFragment = new LeafLibraryFragment();
        Bundle arguments = new Bundle();
        arguments.putInt(ARGUMENT_PAGE_NUMBER, page);
        leafLibraryFragment.setArguments(arguments);
        return leafLibraryFragment;
    }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        pageNumber = getArguments().getInt(ARGUMENT_PAGE_NUMBER);
    }

    @RequiresApi(api = Build.VERSION_CODES.N)
    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                           Bundle savedInstanceState) {
        View view = inflater.inflate(R.layout.leaf_library_fragment, null);

        treePanel = view.findViewById(R.id.treePanel);
        List<LeafSpaceContainer> spaces =
        LeafClassifier.getProjectEnv().getLeafSpecies().stream()
            .map(leafSpecies -> {
                List<String> leafNames =
                leafSpecies.getImages().stream().map(leafImage ->
                leafImage.getFileName().getName()).collect(Collectors.toList());
                return new LeafSpaceContainer(leafSpecies.getName(),
                    leafNames);
            }).collect(Collectors.toList());
        LeafAdapter adapter = new LeafAdapter(this, spaces, treePanel);
        treePanel.setAdapter(adapter);

        FancyButton addSpace = view.findViewById(R.id.addSpace);

        addSpace.setOnClickListener((e) -> {
            EditText classNameField = new EditText(this.getContext());
            new AlertDialog.Builder(this.getContext())
                .setTitle("Class name")
                .setMessage("Please enter a class name:")
                .setView(classNameField)
                .setPositiveButton("Enter", (dialog, whichButton) -> {
                    String spaceName =
                    classNameField.getText().toString();
                    LeafSpecies lSpecies = new LeafSpecies(spaceName);
                    spaces.add(new LeafSpaceContainer(spaceName, new
                    LinkedList<>()));
                })
                .setNegativeButton("Cancel", (dialog, whichButton) -> {
                    LeafClassifier.getProjectEnv().addLeafSpecies(lSpecies);
                    LeafClassifier.getProjectEnv().setModified();
                })
                .show();
        });
    }
}

```

```

        })
        .show();
    });

FancyButton save = view.findViewById(R.id.save);

save.setOnClickListener((e) -> {
    File directory = view.getContext().getFilesDir();
    File file = new File(directory, WelcomeActivity.CACHE_NAME);
    boolean isSaved = LeafClassifier.getProjectEnv().Save(file);
    String result;
    if (isSaved) {
        result = "Saved!";
    } else {
        result = "Error!";
    }
    Toast toast = Toast.makeText(getContext(),
        result, Toast.LENGTH_SHORT);
    toast.show();
});

FancyButton share = view.findViewById(R.id.share);

share.setOnClickListener((e) -> {

    if (checkSelfPermission(getContext(),
        android.Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
        return;
    }

    Intent intentShareFile = new Intent(Intent.ACTION_SEND);
    File directory = view.getContext().getFilesDir();
    File source = new File(directory, WelcomeActivity.CACHE_NAME);
    File destination = new
File(Environment.getExternalStorageDirectory().getAbsolutePath() +
"/leafrecog.txt");
    try {
        FileUtils.copyFile(source, destination);
    } catch (IOException e1) {
        e1.printStackTrace();
    }

    if (destination.exists()) {
        StrictMode.VmPolicy.Builder builder = new
StrictMode.VmPolicy.Builder();
        StrictMode.setVmPolicy(builder.build());
        intentShareFile.setType("application/txt");
        intentShareFile.putExtra(Intent.EXTRA_STREAM,
            Uri.parse("file://" + destination.getAbsolutePath()));
        startActivity(Intent.createChooser(intentShareFile, "Share
Library"));
    }
});
}

FancyButton loadLib = view.findViewById(R.id.loadLib);

loadLib.setOnClickListener((e) -> {
    new ChooserDialog().with(this.getContext())
        .withFilterRegex(false, false, ".*\\.(txt)")
        .withResources(R.string.title_choose_file,
R.string.title_choose, R.string.dialog_cancel)
});
```

```

        .withChosenListener( (path, pathFile) -> {
            File directory = view.getContext().getFilesDir();
            File destination = new File(directory,
WelcomeActivity.CACHE_NAME);
            try {
                FileUtils.copyFile(pathFile, destination);
                Intent i = new Intent(getContext(),
                    WelcomeActivity.class);
                startActivity(i);
            } catch (IOException e1) {
                e1.printStackTrace();
            }
        })
        .build()
        .show();
    });

    return view;
}

@Override
public String getTitle() {
    return title;
}
}

public class RecognitionTask extends AsyncTask<Void, String, Void> {
    private View view;
    private TextView progressText;
    private ProgressBar progressBar;
    LeafImage leafImageForRecognizing;
    private AbstractLeafClassifierFragment fragment;

    BiConsumer<String, String> publishProgress = (s, s2) ->
publishProgress(s, s2);

    public RecognitionTask(AbstractLeafClassifierFragment fragment, View
view, LeafImage leafImageForRecognizing) {
        this.view = view;
        progressText = view.findViewById(R.id.progressText);
        progressBar = view.findViewById(R.id.progressBar);
        this.leafImageForRecognizing = leafImageForRecognizing;
        this.fragment = fragment;
    }

    @Override
    protected Void doInBackground(Void... voids) {
        TableLayout tableLayout = view.findViewById(R.id.probabilitiesTable);
        fragment.getActivity().runOnUiThread(() -> {
            tableLayout.removeAllViews();
            TableRow row = getTableRow("Leaf Name", "Probability");
            tableLayout.addView(row, new
TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT,
TableLayout.LayoutParams.WRAP_CONTENT));
        });

        // Now we recognize the image with the trained neutronal network
        BackProp nNetwork = LeafClassifier.getProjectEnv().getNetwork();

        double[] resultID =
nNetwork.propagate(leafImageForRecognizing.getTokens(nNetwork.numInput())));
    }
}

```

```

/*
for(int i=0; i < nNetwork.numOutput(); i++)
{
    System.out.println("OUT["+i+"] : "+test[i]);
}
*/
publishProgress("classifying.", "85");

ArrayList leafSpecies =
LeafClassifier.getProjectEnv().getLeafSpecies();
double error = 0.0;

int size = leafSpecies.size();

TreeMap<String, Double> results = new TreeMap<>();

for (int i = 0; i < size; i++, error = 0.0) {
    LeafSpecies lSpecies = (LeafSpecies) leafSpecies.get(i);

    //System.out.println("LeafSpecies "+i+
    ["+lSpecies.getName()+"]:");

    double[] ID = lSpecies.getID();

    for (int j = 0; j < nNetwork.numOutput(); j++) {
        //System.out.println(" ["+j+"] : "+ID[j]);

        if (ID[j] >= resultID[j]) {
            error += ID[j] - resultID[j];
        } else error += resultID[j] - ID[j];

    }
    double probabilityValue = (100.0 - (100 / size) * error);
    results.put(lSpecies.getName(), probabilityValue);
}

results.entrySet().stream().

sorted(reverseOrder(Map.Entry.comparingByValue())).forEach((entry) -> {
    TableRow row = getTableRow(entry.getKey(), String.format("%.2f",
entry.getValue()));
    fragment.getActivity().runOnUiThread(() -> {
        tableLayout.addView(row, new
TableLayout.LayoutParams(TableLayout.LayoutParams.WRAP_CONTENT,
TableLayout.LayoutParams.WRAP_CONTENT));
    });
    publishProgress("finished.", "100");
}

return null;
}

private TableRow getTableRow(String left, String right) {
    TableRow row = new TableRow(view.getContext());
    row.setLayoutParams(new
TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT,
TableRow.LayoutParams.WRAP_CONTENT));
    TextView space = new TextView(view.getContext());
    space.setGravity(Gravity.CENTER_HORIZONTAL);
    space.setText(left);
    space.setLayoutParams(new
TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT,

```

```

        TableRow.LayoutParams.WRAP_CONTENT, 1.0f));
        row.addView(space);

        TextView probability = new TextView(view.getContext());
        probability.setText(right);
        probability.setLayoutParams(new
        TableRow.LayoutParams(TableRow.LayoutParams.WRAP_CONTENT,
        TableRow.LayoutParams.WRAP_CONTENT, 1.0f));
        probability.setGravity(Gravity.CENTER_HORIZONTAL);
        row.addView(probability);
        return row;
    }

    @Override
    protected void onProgressUpdate(String... values) {
        super.onProgressUpdate(values);
        progressText.setText(values[0]);
        progressBar.setProgress(Integer.parseInt(values[1]));
    }
}

class TrainNetworkTask extends AsyncTask<Void, String, Void> {
    private View view;
    private TextView progressText;
    private ProgressBar progressBar;
    private TextView error;
    TextView leafImagesField;
    //    TextView leafSpeciesField;
    TextView maxTokensField;
    TextView status;

    public TrainNetworkTask(View view) {
        this.view = view;
        progressText = view.findViewById(R.id.progressText);
        progressBar = view.findViewById(R.id.progressBar);
        error = view.findViewById(R.id.error);
        leafImagesField = view.findViewById(R.id.leafImages);
        //    leafSpeciesField = view.findViewById(R.id.leafSpecies);
        maxTokensField = view.findViewById(R.id.maxTokens);
    }

    @Override
    protected void onPreExecute() {
        super.onPreExecute();
    }

    @Override
    protected Void doInBackground(Void... params) {
        // get the properties of the network first

        publishProgress("Initializing...", "0");

        EditText inputNeurons = view.findViewById(R.id.inputNeurons);
        EditText hiddenNeurons = view.findViewById(R.id.hiddenNeurons);
        EditText outputNeurons = view.findViewById(R.id.outputNeurons);
        EditText learnRateField = view.findViewById(R.id.learnRate);
        EditText momentumField = view.findViewById(R.id.momentum);
        EditText maxSteps = view.findViewById(R.id.maxSteps);

        GraphView graph = view.findViewById(R.id.graph);
        graph.removeAllSeries();
        LineGraphSeries<DataPoint> series = new LineGraphSeries<>(new
        DataPoint[]{});
```

```

int inputs = Integer.parseInt(inputNeurons.getText().toString());
int hiddens = Integer.parseInt(hiddenNeurons.getText().toString());
int outputs = Integer.parseInt(outputNeurons.getText().toString());
double learnRate =
Double.parseDouble(learnRateField.getText().toString());
double momentum =
Double.parseDouble(momentumField.getText().toString());
int steps = Integer.parseInt(maxSteps.getText().toString());

// change progressbar
// netProgressBar.setMaximum(steps);

// Now we create a new BackProp object
// for doing the neuronal network stuff
BackProp nNetwork = new BackProp(inputs, hiddens, outputs);

// Set the network in the project environment
LeafClassifier.getProjectEnv().setNetwork(nNetwork);

// set learnrate & momentum to user settings
nNetwork.setAlpha(learnRate);
nNetwork.setMomentum(momentum);

ArrayList<LeafSpecies> leafSpecies =
LeafClassifier.getProjectEnv().getLeafSpecies();

for (int i = 0; i < leafSpecies.size(); i++) {
    LeafSpecies lSpecies = leafSpecies.get(i);

    double[] outputVals = new double[outputs];

    // create a random ID as long as the output neurons
    for (int j = 0; j < outputs; j++) {
        if (j <= i) {
            outputVals[j] = 1.0;
        } else outputVals[j] = 0.0;
    }

    lSpecies.setID(outputVals);
}

double sumError = 0;

// Now we create the ArrayList of all Images that
// will be shuffled later

ArrayList imageList = new ArrayList();

for (int j = 0; j < leafSpecies.size(); j++) {
    LeafSpecies lSpecies = leafSpecies.get(j);

    for (int k = 0; k < lSpecies.numImages(); k++) {
        LeafImage lImage = lSpecies.getImage(k);

        // set the Species of this image now!
        lImage.setSpecies(lSpecies);

        imageList.add(lImage);
    }
}

// steps for training

```

```

        for (int i = 0; i < steps; i++) {
            sumError = 0.0;

            int progress = (i * 100) / steps;
            publishProgress("Training... step " + i + "/" + steps,
String.valueOf(progress));

            // now we have to shuffle the tokenList to get
            // a randomness in the training process
            Collections.shuffle(imageList);

            for (int j = 0; j < imageList.size(); j++) {
                LeafImage lImage = (LeafImage) imageList.get(j);

                sumError += nNetwork.learnVector(lImage.getTokens(inputs),
lImage.getSpecies().getID());
            }

            // errorPanel.addError(sumError);
            series.appendData(new DataPoint(i, sumError), true, 100);
        }
        graph.addSeries(series);
        String textErrorOnClick =
String.valueOf(LeafClassifier.getProjectEnv().getNetwork().getAbsError());
        error.setText(textErrorOnClick);

leafImagesField.setText(String.valueOf(LeafClassifier.getProjectEnv().numLeaf
Images()));
//
leafSpeciesField.setText(String.valueOf(LeafClassifier.getProjectEnv().getLea
fSpecies()));

maxTokensField.setText(String.valueOf(LeafClassifier.getProjectEnv().getTokenCount()));
        graph.setViewport().setXAxisBoundsManual(true);
        graph.setViewport().setMaxX(steps);
        publishProgress("finished", "100");

        return null;
    }

    @Override
    protected void onProgressUpdate(String... values) {
        super.onProgressUpdate(values);
        progressText.setText(values[0]);
        progressBar.setProgress(Integer.parseInt(values[1]));
    }

    @Override
    protected void onPostExecute(Void result) {
        super.onPostExecute(result);
    }
}

public class BackProp {
    private double inputA[];           // activations input

    private double hiddenA[];          // activations hidden
    private double hiddenN[];          // sum of products for hidden units
    private double hiddenD[];          // output error
    private double hiddenW[][];         // connection weights matrix

    private double outputA[];          // activations output
}

```

```

private double outputN[];           // sum of products
private double outputD[];           // output error
private double oldD[];             // old output error
private double outputW[][];         // connection weights matrix

private double biasH[];             // bias for the hidden units
private double biasO[];             // bias for the output units

private int numInput;              // number of neurons on input layer
private int numHidden;              // number of neurons on hidden layer
private int numOutput;              // number of neurons on output layer

private int epoch;                 // number of epochs of the learn process
private double momentum;            // momentum
private double alpha;               // learnrate
@Getter
@Setter
private double absError = 0.0; // the absolute error of the learning
proc.

// create random object to get individual random numbers
private Random rand;

< /**
 * Constructor
 */
public BackProp(int input, int hidden, int output) {
    numInput = input;      // number of neurons in the input layer
    numHidden = hidden;    // number of neurons in the hidden layer
    numOutput = output;    // number of neurons in the output layer

    inputA = new double[numInput];
    hiddenW = new double[numHidden][numInput];

    hiddenA = new double[numHidden];
    hiddenN = new double[numHidden];
    hiddenD = new double[numHidden];
    biasH = new double[numHidden];

    outputW = new double[numOutput][numHidden];
    outputA = new double[numOutput];
    outputN = new double[numOutput];
    outputD = new double[numOutput];
    oldD = new double[numOutput];
    biasO = new double[numOutput];

    alpha = 0.3; // default learnrate = 0.3
    momentum = 1.0; // default momentum = 1.0

    // initialize the random object
    rand = new Random();
}

// now we initialize the network
init();
}

< /**
 * Constructor
 */
public BackProp(int input, int hidden, int output, double alpha, double
mom) {

```

```

this(input, hidden, output);

this.alpha = alpha;
this.momentum = mom;
}


$$\begin{array}{l} \text{***} \\ \text{* init()} \\ \text{* } \langle p \rangle \\ \text{* initialize the network with random numbers} \\ \text{*} \end{array}$$

private void init() {
    epoch = 0;

    // init net with small random values
    for (int i = 0; i < numInput; i++) {
        inputA[i] = frandom(-1.0, 1.0);
    }

    for (int i = 0; i < numHidden; i++) {
        hiddenA[i] = frandom(-1.0, 1.0);
        biasH[i] = frandom(-1.0, 1.0);
        for (int m = 0; m < numInput; m++) {
            hiddenW[i][m] = frandom(-1.0, 1.0);
        }
    }

    for (int i = 0; i < numOutput; i++) {
        biasO[i] = frandom(-1.0, 1.0);
        for (int m = 0; m < numHidden; m++) {
            outputW[i][m] = frandom(-1.0, 1.0);
        }
    }
}


$$\begin{array}{l} \text{***} \\ \text{* sigmoid()} \\ \text{* } \langle p \rangle \\ \text{* sigmoid activation function} \\ \text{*} \end{array}$$

private double sigmoid(double x) {
    return (1.0 / (1.0 + Math.exp(-x)));
}

private double sigmoidDeriv(double x) {
    return (sigmoid(x) * (1 - sigmoid(x)));
}


$$\begin{array}{l} \text{***} \\ \text{* feedForward()} \\ \text{* } \langle p \rangle \\ \text{* method to do the feed forward} \\ \text{*} \end{array}$$

private void feedForward() {
    double sum2 = 0.0;

    // calculate the hidden weights
    for (int i = 0; i < numHidden; i++) {
        sum2 = biasH[i];
        for (int j = 0; j < numInput; j++) {
            sum2 += hiddenW[i][j] * inputA[j];
        }
        hiddenN[i] = sum2;
    }
}

```

```

        hiddenA[i] = sigmoid(sum2);
    }

    // calculate the new output weights
    for (int i = 0; i < numOutput; i++) {
        sum2 = biasO[i];
        for (int j = 0; j < numHidden; j++) {
            sum2 += outputW[i][j] * hiddenA[j];
        }
        outputN[i] = sum2;
    }
}

/***
 * computeDelta()
 * <p>
 * method to calculate the new delta
 */
private void computeDelta(int m) {
    outputD[m] = (outputA[m] - sigmoid(outputN[m])) *
sigmoidDeriv(outputN[m]);

    // hidden layer calculation
    for (int i = 0; i < numHidden; i++) {
        outputW[m][i] += outputD[m] * hiddenA[i] * alpha;
    }

    // output layer calculation
    for (int i = 0; i < numOutput; i++) {
        biasO[i] += outputD[m] * alpha;
    }
}

/***
 * updateWeights()
 * <p>
 * method to update the other weights also
 */
private void updateWeights() {
    double sum2;

    for (int j = 0; j < numHidden; j++) {
        sum2 = 0.0;
        for (int i = 0; i < numOutput; i++) {
            sum2 += outputD[i] * outputW[i][j];
        }

        sum2 *= sigmoidDeriv(hiddenN[j]);
        biasH[j] += sum2 * alpha;

        for (int i = 0; i < numInput; i++) {
            hiddenW[j][i] += alpha * sum2 * inputA[i];
        }
    }
}

/***
 * frandom()
 * <p>
 * special random method that return a random number
 * within min and max.
 */
private double frandom(double min, double max) {

```

```

        return rand.nextDouble() * (max - min) + min;
    }

    /**
     * propagate()
     * <p>
     * method to return a array of doubles with the recognized
     * values after a double vector has been parsed.
     */
    public double[] propagate(double[] vector) {
        double sum2;

        for (int i = 0; i < numInput; i++) {
            inputA[i] = vector[i];
        }

        for (int i = 0; i < numHidden; i++) {
            sum2 = biasH[i];
            for (int j = 0; j < numInput; j++) {
                sum2 += hiddenW[i][j] * inputA[j];
                hiddenA[i] = sigmoid(sum2);
            }
        }

        for (int i = 0; i < numOutput; i++) {
            sum2 = biasO[i];
            for (int j = 0; j < numHidden; j++) {
                sum2 += outputW[i][j] * hiddenA[j];
            }
            outputN[i] = sum2;
            outputA[i] = sigmoid(sum2);
        }

        return outputA;
    }

    /**
     * learnVector()
     * <p>
     * method to let the network learn on passed input and output
     * vectors and return the error for this learning operation.
     * <p>
     * Please note that this error is not the absolute error of the
     * whole network! You have to add all error for a learning phase
     * to get the overall error.
     */
    public double learnVector(double[] in, double[] out) {
        for (int i = 0; i < numInput; i++) {
            inputA[i] = in[i];
        }

        for (int i = 0; i < numOutput; i++) {
            outputA[i] = out[i];
        }

        feedForward();

        absError = 0.0;

        // calculate the absError
        for (int j = 0; j < numOutput; j++) {
            computeDelta(j);
            absError += Math.pow(outputA[j] - sigmoid(outputN[j]), 2);
        }
    }
}

```

```

    }

    updateWeights();

    // update the learnrate
    alpha *= momentum;

    return absError;
}

public int numInput() {
    return numInput;
}

public int numHidden() {
    return numHidden;
}

public int numOutput() {
    return numOutput;
}

public double[] getBiasH() {
    return biasH;
}

public double[] getBiasO() {
    return biasO;
}

public double[][] getHiddenW() {
    return hiddenW;
}

public double[][] getOutputW() {
    return outputW;
}

public double getAlpha() {
    return alpha;
}

public double getMomentum() {
    return momentum;
}

public int getEpoch() {
    return epoch;
}

public void setAlpha(double a) {
    alpha = a;
}

public void setMomentum(double mom) {
    momentum = mom;
}

public void setHiddenW(double weight, int hidden, int input) {
    //System.out.println("SetHiddenW: ["+hidden+"] ["+input+"]="+weight);
    hiddenW[hidden][input] = weight;
}

```

```
public void setBiasH(double bias, int hidden) {
    //System.out.println("SetBiasH: ["+hidden+"]="+bias);
    biasH[hidden] = bias;
}

public void setOutputW(double weight, int output, int hidden) {
    //System.out.println("SetOutputW: ["+output+"] ["+hidden+"]="+weight);
    outputW[output][hidden] = weight;
}

public void setBiasO(double bias, int output) {
    //System.out.println("SetBiasO: ["+output+"]="+bias);
    biasO[output] = bias;
}

}
```

## ДОДАТОК Б

Вибірка, використана у кваліфікаційній роботі:

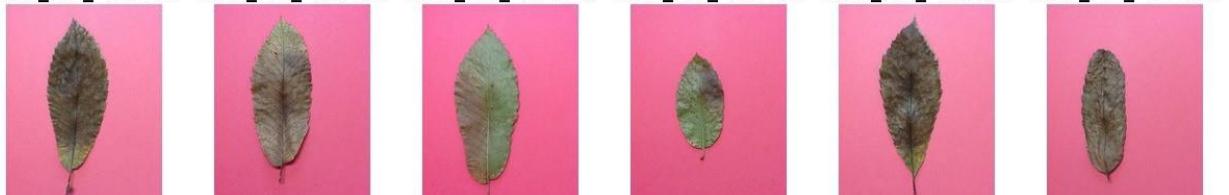








'iPAD2\_C13\_EX09.JPG' 'iPAD2\_C13\_EX10.JPG' 'iPAD2\_C13\_EX11.JPG' 'iPAD2\_C13\_EX12.JPG' 'iPAD2\_C13\_EX13.JPG' 'iPAD2\_C14\_EX01.JPG'



'iPAD2\_C14\_EX02.JPG' 'iPAD2\_C14\_EX03.JPG' 'iPAD2\_C14\_EX04.JPG' 'iPAD2\_C14\_EX05.JPG' 'iPAD2\_C14\_EX06.JPG' 'iPAD2\_C14\_EX07.JPG'



'iPAD2\_C14\_EX08.JPG' 'iPAD2\_C14\_EX09.JPG' 'iPAD2\_C14\_EX10.JPG' 'iPAD2\_C14\_EX11.JPG' 'iPAD2\_C14\_EX12.JPG' 'iPAD2\_C15\_EX01.JPG'



'iPAD2\_C15\_EX02.JPG' 'iPAD2\_C15\_EX03.JPG' 'iPAD2\_C15\_EX04.JPG' 'iPAD2\_C15\_EX05.JPG' 'iPAD2\_C15\_EX06.JPG' 'iPAD2\_C15\_EX07.JPG'



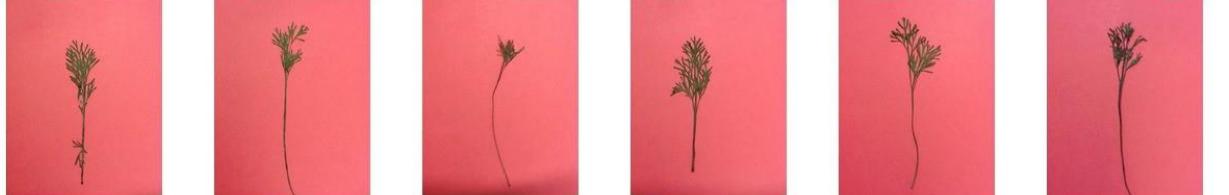
'iPAD2\_C15\_EX08.JPG' 'iPAD2\_C15\_EX09.JPG' 'iPAD2\_C15\_EX10.JPG' 'iPAD2\_C16\_EX01.JPG' 'iPAD2\_C16\_EX02.JPG' 'iPAD2\_C16\_EX03.JPG'



'iPAD2\_C16\_EX04.JPG' 'iPAD2\_C16\_EX05.JPG' 'iPAD2\_C16\_EX06.JPG' 'iPAD2\_C16\_EX07.JPG' 'iPAD2\_C16\_EX08.JPG' 'iPAD2\_C16\_EX09.JPG'

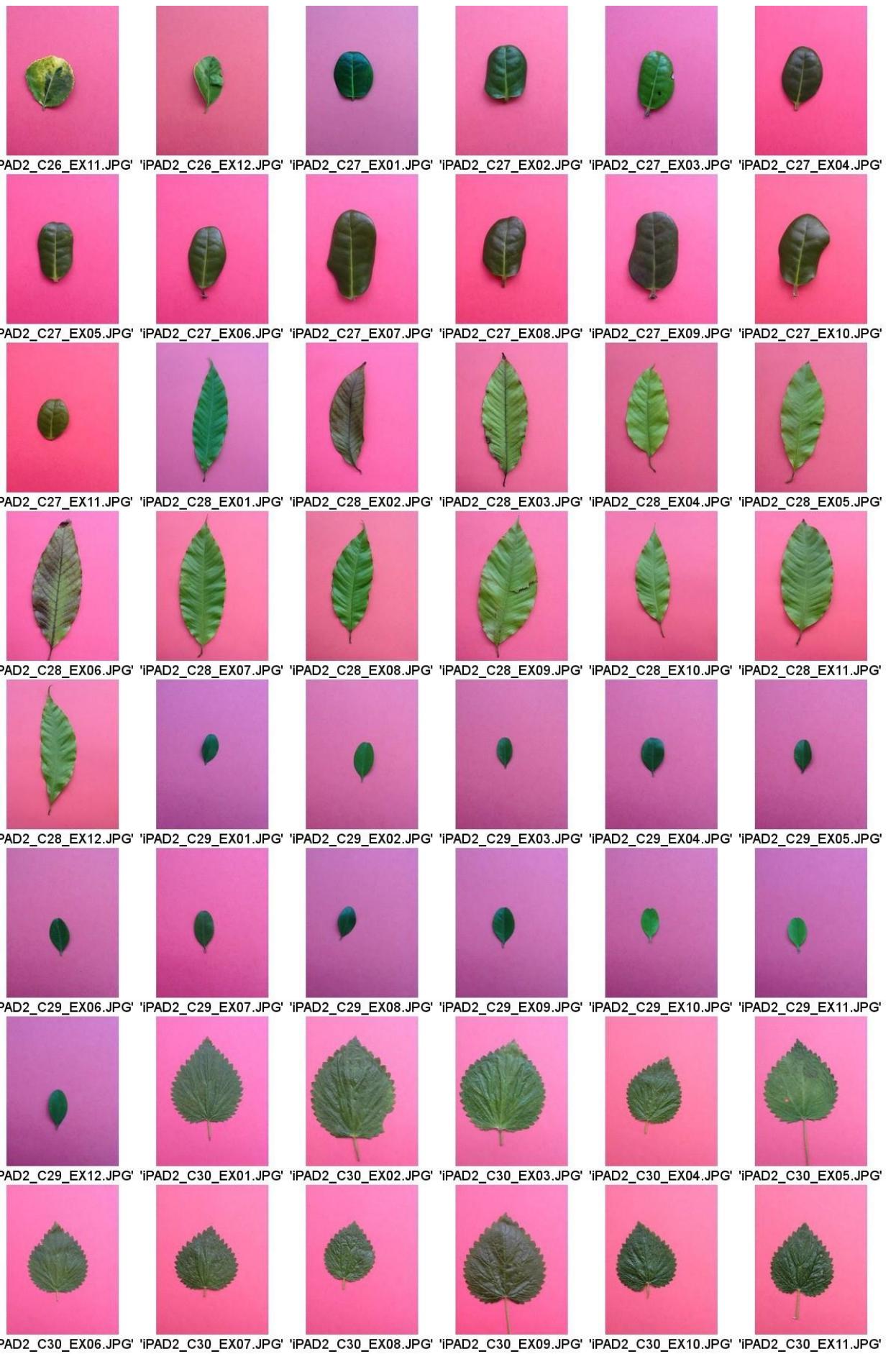


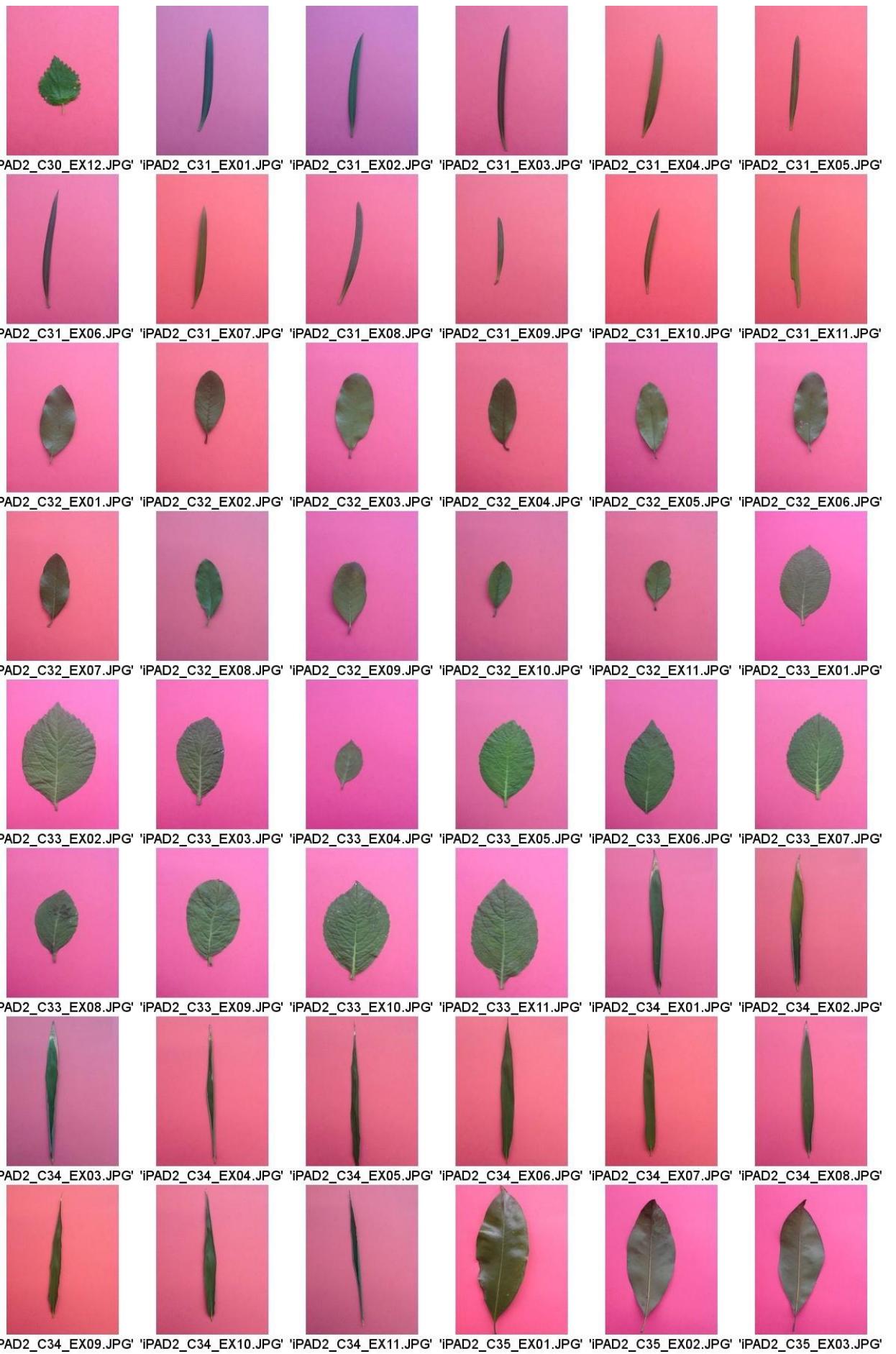
'iPAD2\_C16\_EX10.JPG' 'iPAD2\_C17\_EX01.JPG' 'iPAD2\_C17\_EX02.JPG' 'iPAD2\_C17\_EX03.JPG' 'iPAD2\_C17\_EX04.JPG' 'iPAD2\_C17\_EX05.JPG'



'iPAD2\_C18\_EX01.JPG' 'iPAD2\_C18\_EX02.JPG' 'iPAD2\_C18\_EX03.JPG' 'iPAD2\_C18\_EX04.JPG' 'iPAD2\_C18\_EX05.JPG' 'iPAD2\_C18\_EX06.JPG'







'iPAD2\_C30\_EX12.JPG' 'iPAD2\_C31\_EX01.JPG' 'iPAD2\_C31\_EX02.JPG' 'iPAD2\_C31\_EX03.JPG' 'iPAD2\_C31\_EX04.JPG' 'iPAD2\_C31\_EX05.JPG'

'iPAD2\_C31\_EX06.JPG' 'iPAD2\_C31\_EX07.JPG' 'iPAD2\_C31\_EX08.JPG' 'iPAD2\_C31\_EX09.JPG' 'iPAD2\_C31\_EX10.JPG' 'iPAD2\_C31\_EX11.JPG'

'iPAD2\_C32\_EX01.JPG' 'iPAD2\_C32\_EX02.JPG' 'iPAD2\_C32\_EX03.JPG' 'iPAD2\_C32\_EX04.JPG' 'iPAD2\_C32\_EX05.JPG' 'iPAD2\_C32\_EX06.JPG'

'iPAD2\_C32\_EX07.JPG' 'iPAD2\_C32\_EX08.JPG' 'iPAD2\_C32\_EX09.JPG' 'iPAD2\_C32\_EX10.JPG' 'iPAD2\_C32\_EX11.JPG' 'iPAD2\_C33\_EX01.JPG'

'iPAD2\_C33\_EX02.JPG' 'iPAD2\_C33\_EX03.JPG' 'iPAD2\_C33\_EX04.JPG' 'iPAD2\_C33\_EX05.JPG' 'iPAD2\_C33\_EX06.JPG' 'iPAD2\_C33\_EX07.JPG'

'iPAD2\_C33\_EX08.JPG' 'iPAD2\_C33\_EX09.JPG' 'iPAD2\_C33\_EX10.JPG' 'iPAD2\_C33\_EX11.JPG' 'iPAD2\_C34\_EX01.JPG' 'iPAD2\_C34\_EX02.JPG'

'iPAD2\_C34\_EX03.JPG' 'iPAD2\_C34\_EX04.JPG' 'iPAD2\_C34\_EX05.JPG' 'iPAD2\_C34\_EX06.JPG' 'iPAD2\_C34\_EX07.JPG' 'iPAD2\_C34\_EX08.JPG'

'iPAD2\_C34\_EX09.JPG' 'iPAD2\_C34\_EX10.JPG' 'iPAD2\_C34\_EX11.JPG' 'iPAD2\_C35\_EX01.JPG' 'iPAD2\_C35\_EX02.JPG' 'iPAD2\_C35\_EX03.JPG'

## ДОДАТОК В

Витяг з матеріалов восьмої міжнародної наукової конференції студентів та молодих вчених (MIT 2018):

### МЕТОДИКА НЕЙРОСЕТЕВОЇ КЛАССИФІКАЦІИ ЛИСТЬЕВ РАСТЕНИЙ НА ОСНОВЕ МЕТОДОВ ОБРАБОТКИ И АНАЛИЗА ИЗОБРАЖЕНИЙ

Іщенко А.Д.

ст. викладач Годовіченко М. А.

Одесский Национальный Политехнический Университет, УКРАИНА

**АННОТАЦІЯ.** Предложено мобильное приложение для Android устройств, выполняющее нейросетевую классификацию изображений листьев растений с возможностью обучения нейронной сети на новых видах растений.

**Введение.** По оценкам, в мире насчитывается почти полмиллиона видов растений. Классификация видов исторически проблематична и из-за несовершенства методов часто приводит к дублированию информации. Текущие реализации данной технологии позволяют решать задачу обнаружения и классификации листьев деревьев, потратив определенное время, но подобные технологии используют либо ресурсоёмкие алгоритмы, которые требуют дорогостоящего оборудования, которое не может использоваться в полевых условиях, либо вообще не применяют алгоритмы распознавания, требуя от пользователя ввести описание растения. В связи с этим, актуальной задачей будет использование менее затратных алгоритмов, выдающих приемлемый результат.

**Цель работы.** Целью данной работы является разработка мобильного приложения, которое позволит проводить анализ изображения, выделения на нем объекта (листа растения) и отнесение данного объекта к какому-либо классу при помощи нейронной сети.

**Основная часть работы.** Решение задачи классификации почти полмиллиона видов растений с применением автоматизации распознавания растений может иметь множество практических применений. Некоторые из них приведены ниже:

1) Наблюдение и сохранение популяций видов - изменение окружающей среды под влиянием человека, трансформация местообитаний, разрушение квазинатурального растительного покрова приводят к раздроблению и уменьшению численности популяций растений, вымиранию отдельных видов, общему и локальному обеднению флоры, невосполнимой утрате генетических ресурсов растительного мира.

Наиболее уязвимыми элементами региональных флор обычно оказываются эндемичные, реликтовые, а также некоторые полезные растения (декоративные, лекарственные, пищевые).

Для того чтобы своевременно принять меры по спасению редких и исчезающих растений, необходимо знать состояние их популяций. В ряде контрольных пунктов должны быть организованы наблюдения за популяциями наиболее интересных и важных в научном и практическом отношении видов растений. При этом необходимо учитывать плотность, численность популяций, пространственную и возрастную структуру, их динамику, реакцию на антропогенные воздействия. Особенно ценным показателем служит возрастная структура популяций: если она приобретает регрессивный характер, это уже серьезный сигнал тревоги.

2) Фармацевтические исследования на основе растений - перспективность исследований лекарственных растений для применения в современной фармации несомненна. При введении таких растений в фармацевтическую и медицинскую практику в первую очередь следует проводить целый комплекс исследований, устанавливающих их видовую принадлежность и очерченность границ вида.

3) Применение в сфере продовольственной и сельскохозяйственной промышленности - сегодня аграрии все чаще обращаются к современным технологиям,

чтобы увеличить производство в условиях ограниченных ресурсов и повысить эффективность земледелия. Так, например, автоматизация распознавания позволит ускорить отбор полезных сырьевых растений и как следствие улучшить качество производимого продукта.

Однако текущие реализации данной технологии позволяют решать задачу обнаружения и классификации листьев деревьев, потратив определенное время, но подобные технологии используют либо ресурсоёмкие алгоритмы, которые требуют дорогостоящего оборудования, которое не может использоваться в полевых условиях, либо вообще не применяют алгоритмы распознавания, требуя от пользователя ввести описание растения. В связи с этим, актуальной задачей будет использование менее затратных алгоритмов, выдающих приемлемый результат.

В качестве метода обработки изображения было принято решения использовать оператора Прюитта (англ. Prewitt operator), в дисциплине компьютерного зрения — метод выделения границ в обработке изображений (рис. 1), который вычисляет максимальный отклик на множество ядер свёртки для нахождения локальной ориентации границы в каждом пикселе.

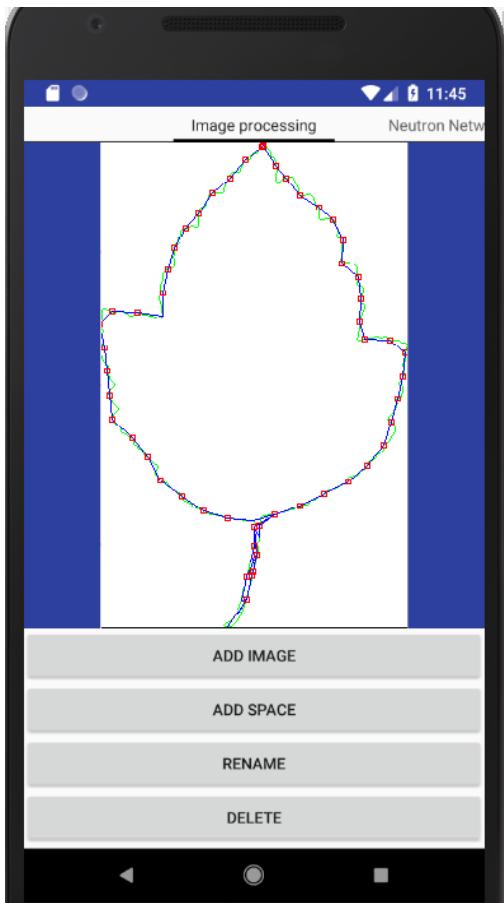


рис. 1 – Демонстрация работы оператора Прюитта в приложении для Android устройств

**Выводы.** Разработанное мобильное приложение позволяет выполнять нейросетевую классификацию изображений листьев растений с возможностью обучения нейронной сети на новых видах растений. Использование мобильного приложения имеет значительный потенциал при использовании на производстве или в ботанических исследованиях, так как в разы снижает время и стоимость классификации, при сохранении достаточного уровня точности.