Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)
Навчально-науковий інститут комп'ютерних наук
та інформаційних технологій
(повне найменування інституту, назва факультету (відділення))
Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

Магістерська кваліфікаційна робота

другий (магістерський)
(рівень вищої освіти)

на тему: **Інтелектуальна система класифікації побутових відходів методами глибокого навчання**

Виконав: студент VI курсу, групи КН-62м
спеціальності
122 – "Комп'ютерні науки"
(шифр і назва напряму підготовки, спеціальності)

Критчак В.І.
(прізвище та ініціали)

Керівник Пірко І.Б.
(прізвище та ініціали)

Рецензент Сторожук О.Л.
(прізвище та ініціали)

Львів – 2025 р.

Національний лісотехнічний університет України
(повне найменування вищого навчального закладу)

ННІ комп'ютерних наук та інформаційних технологій
Кафедра комп'ютерних наук
Рівень вищої освіти другий (магістерський)
Спеціальність 122 "Комп'ютерні науки"
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КН

Борецька І. Б.
"10" грудня 2025 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Критчак Віталій Іванович
(прізвище, ім'я, по батькові)

1. Тема роботи **Інтелектуальна система класифікації побутових відходів методами глибокого навчання**

керівник роботи Пірко І. Б, канд. фіз.-мат. наук, доцент.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від 29.04. 2025 р.№ С-288

2. Термін подання студентом роботи 10.12. 2025 р.

3. Вихідні дані до роботи:

- вивчити предметну область, проаналізувати існуючі фактори та методи моделювання систем класифікації побутових відходів
- розглянути і використати алгоритми, які лежать в основі математичної моделі інтелектуальної системи класифікації побутових відходів;
- спроектувати інтелектуальну систему з допомогою мови програмування Python та відповідних бібліотек;
- представити результати роботи інтелектуальної системи.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Розділ 1. Стан проблемної області
Розділ 2. Інформаційне забезпечення
Розділ 3. Математичне забезпечення
Розділ 4. Програмне забезпечення
Розділ 5. Розроблення стартап-проєкту

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
додаток А, додаток Б

6. Дата видачі завдання 1 травня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Примітка
1	Огляд літературних даних та інших джерел згідно досліджуваної теми	01.05-30.05.2025	виконано
2	Аналіз досліджуваної теми та вибір відповідних варіантів її розробки	01.06-30.06. 2025	виконано
3	Постановка задачі та її формалізація	01.07-30.07. 2025	виконано
4	Вибір та обґрунтування методів і засобів проведення дослідження	01.08-30.08. 2025	виконано
5	Розроблення концептуальної схеми реалізації завдання	01.09-15.09. 2025	виконано
6	Програмна реалізація завдання	16.09-30.10. 2025	виконано
7	Тестування програмного продукту та отриманих результатів	01.11-15.11. 2025	виконано
8	Розробка пояснювальної записки магістерської роботи	16.11-30.11. 2025	виконано
9	Корегування пояснювальної записки згідно вимог, розроблення презентації	01.12-09.12. 2025	виконано

Студент (підпис) Критчак В.І.
(прізвище та ініціали)

Керівник роботи (підпис) Пірко І.Б.
(прізвище та ініціали)

АНОТАЦІЯ

Магістерська робота містить 90 сторінок пояснювальної записки, 14 рисунків, 3 таблиці, 2 додатки, 15 джерел.

Розроблено та реалізовано інтелектуальну систему класифікації побутових відходів за методами глибокого навчання. Проект включає огляд літератури, аналіз існуючих систем, а також створення математичної та програмної моделей. Система використовує згорткові нейронні мережі для точного розпізнавання різних класів відходів на зображеннях. У результаті дослідження була досягнута висока точність класифікації, що підтверджено тестуванням моделі. Проект має практичне значення для покращення процесів розумного управління відходами.

Ключові слова: *інтелектуальна система, глибоке навчання, згорткові нейронні мережі, класифікація побутових відходів*.

ABSTRACT

The thesis contains 90 pages of explanatory note, 14 figures, 3 table, 2 appendix, 15 used literary sources.

An intelligent system for classifying household waste using deep learning methods has been developed and implemented. The project includes a literature review, analysis of existing systems, and the creation of mathematical and software models. The system uses convolutional neural networks to accurately recognize different classes of waste in images. As a result of the research, high classification accuracy was achieved, which was confirmed by testing the model. The project has practical significance for improving smart waste management processes.

Keywords: *intelligent system, deep learning, convolutional neural networks, household waste classification*.

ТЕХНІЧНЕ ЗАВДАННЯ

В дипломній роботі потрібно розробити інтелектуальну систему для класифікації побутових відходів, для цього потрібно вирішити такі завдання.

1. Дослідити та проаналізувати вже розроблені системи для класифікації побутових відходів, визначити їхні сильні та слабкі сторони.

2. Створити математичну модель, що описує алгоритми класифікації побутових відходів.

3. Розробити алгоритм функціонування інтелектуальної системи, що описує етапи обробки зображень і класифікації відходів.

4. Розробити та реалізувати програмну модель системи, включаючи використання згорткових нейронних мереж для автоматичного розпізнавання та класифікації відходів.

5. Провести тестування реалізованої системи для оцінки її продуктивності та точності класифікації.

.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

- ANN (Artificial Neural Network) – штучна нейронна мережа;
- CNN (Convolutional Neural Network) – згорткова нейронна мережа;
- CSV (Comma-Separated Values) – значення, розділені комами;
- DL (Deep Learning) – глибоке навчання;
- GUI (Graphical User Interface) – графічний інтерфейс користувача;
- F1-score – метрика якості класифікації, яка враховує як precision, так і повноту;
- Light Gradient Boosting (LGB) – градієнтний бустинг;
- ML – машинне навчання;
- OpenCV (Open Source Computer Vision Library) бібліотека для комп'ютерного зору, обробки зображень та машинного навчання;
- ReLU (Rectified Linear Unit) – нелінійна функція активації;
- ResNet50 (Residual Networks) – згорткова нейронна мережа, яка входить до сімейства ResNet;
- ROC (Receiver operating characteristic) – ROC-крива;
- Pillow – бібліотека для базових операцій зображення;
- Precision – метрика, яка показує, яка частка передбачених позитивних прикладів є дійсно позитивною;
- SWM (smart waste management) – розумне управління відходами;
- VGG16 – одна з найвідоміших згорткових нейронних мереж;
- БД – база даних;
- ГА – генетичний алгоритм;
- Європейське агентство з охорони навколишнього середовища (ЄАО);
- ООП – об'єктно-орієнтоване програмування;
- ПЗ – програмне забезпечення;
- ТПВ – тверді побутові відходи;
- ШНМ – штучна нейронна мережа;
- ШІ – штучний інтелект.

ВСТУП

Актуальність дипломної роботи

Забруднення навколишнього середовища через неправильне сортування та утилізацію побутових відходів стає все більшою проблемою глобального масштабу. Застосування сучасних технологій штучного інтелекту та глибокого навчання дозволяє підвищити ефективність та швидкість сортування відходів. В умовах зростаючого обсягу сміття автоматизовані системи класифікації є необхідним інструментом для управління відходами. Використання машинного навчання дозволяє створити точні рішення для розпізнавання різних типів побутових відходів за зображеннями. Розробка такої системи відповідає сучасним тенденціям цифрової трансформації екологічного сектору. Впровадження автоматичних систем сортування може значно зменшити людські помилки та підвищити швидкість обробки відходів. Враховуючи глобальні проблеми змін клімату та ресурсоощадження, автоматизація процесів сортування має стратегічне значення. Підвищення точності класифікації сприяє зменшенню навантаження на довкілля. Актуальність дослідження зумовлена потребою у сучасних, автоматизованих рішеннях для екологічної безпеки.

Предмет дослідження – розробка інтелектуальної системи для класифікації побутових відходів за допомогою методів глибокого навчання.

Об’єкт дослідження – процес автоматизованої класифікації побутових відходів з використанням технологій комп’ютерного зору.

Мета роботи – розробка та реалізація інтелектуальної системи для надійного розпізнавання побутових відходів на зображеннях.

Завдання:

1. Дослідити та проаналізувати розроблені системи для класифікації побутових відходів, визначити їхні сильні та слабкі сторони.

2. Створити математичну модель, що описує алгоритми класифікації побутових відходів.

3. Розробити алгоритм функціонування інтелектуальної системи, що описує етапи обробки зображень і класифікації відходів.

4. Розробити та реалізувати програмну модель системи, включаючи використання згорткових нейронних мереж для автоматичного розпізнавання та класифікації відходів.

5. Провести тестування реалізованої системи для оцінки її продуктивності та точності класифікації.

Наукова новизна одержаних результатів

Наукова новизна курсової роботи полягає у розробці інтелектуальної системи, яка використовує передові методи глибокого навчання для класифікації побутових відходів. У проекті реалізовано алгоритм, який оптимізує процес навчання нейронної мережі, забезпечуючи високу точність класифікації. Досліджено новий підхід до формалізації задачі класифікації, що охоплює багатокласову класифікацію зображень відходів. Крім того, система передбачає інтеграцію з роботизованими установками для автоматизації процесу роздільного збору відходів. Результати роботи можуть бути використані для підвищення ефективності управління відходами та зменшення негативного впливу на екологію.

Практичне значення одержаних результатів

Практичне значення одержаних результатів полягає у впровадженні інтелектуальної системи класифікації побутових відходів, що суттєво спростить процес їхньої переробки. Система може використовуватися муніципалітетами та підприємствами для покращення ефективності збору та утилізації відходів, сприяючи екологічним ініціативам. Результати дослідження дозволять оптимізувати роботу сміттєзвалищ, зменшуючи об’єми невитрачених ресурсів і запобігаючи забрудненню. Використання технологій глибокого навчання у розподілі відходів сприяє впровадженню інноваційних рішень в секторі управління відходами. Крім того, результати можуть стати основою для подальших досліджень і розробок у сфері екологічних технологій та штучного інтелекту.

РОЗДІЛ 1. СТАН ПРОБЛЕМНОЇ ОБЛАСТІ

1.1. Класифікація побутових відходів на основі згорткових нейронних мереж

Переробка та управління відходами є серйозною проблемою, що спричиняє проблеми для здоров'я людини, а також для атмосфери. Світ зараз більше зацікавлений в управлінні відходами з точки зору розробки технологій для зменшення їх обсягу. Переробка є одним з основних способів обмеження утворення відходів. Ключовим кроком перед переробкою є сортування відходів за їхньою природою за допомогою нових технологій. Автори роботи [1] віддають перевагу автоматичному підходу до сортування, а не ручному, через його обсяг, точність та економічну ефективність процесу. Класифікація та переробка відходів проводяться у певних фіксованих місцях за межами міста. Всі ці відходи необхідно скидати та сортувати в цьому місці, що вимагає великої трудомісткості, низької продуктивності сортування та поганих умов праці з меншою точністю. Проблему класифікації відходів можна вирішити з початкового етапу утилізації, ефективно вдома або біля джерела їх утворення на основі їх структури, матеріалу та кількох інших характеристик, включаючи їх зв'язок з іншими предметами, такими як їхнє місцезнаходження, структура та підвищення їх інтелектуальних здібностей, щоб уникнути марнування ресурсів; різні види сміття повинні фільтруватися за сегментами, щоб відповідати вимогам охорони навколишнього середовища [2]. Однак, оскільки знання людей про класифікацію відходів мінімальні, класифікація є складною, оскільки існує багато різних форм відходів. Люди, які живуть у селах, не можуть класифікувати ці відходи. Наприклад, численні дослідження в розвинених країнах повідомляють, що середній обсяг збору сміття в Європі становить 500 кг на рік, причому лише невелика його частина переробляється. За даними Агентства з охорони навколишнього середовища, 75 % сміття, що утворюється, можна використовувати повторно, але лише невелика його частина переробляється. Згідно з останньою міжнародною оцінкою, обсяг сміття у світі зросте на 70 % до 2050 року [3].

Харчові відходи призводять до екологічних проблем, таких як викиди парникових газів. Харчові відходи утворюють метан, коли їх скидають на сміттєзвалище, також вони займають місце та створюють вищий ризик забруднення ґрунтових вод. Пристрої для збору газу зі сміттєзвалищ не здатні вловлювати метан, що утворюється харчовими відходами, який виходить в атмосферу. Автори [4] зазначають, що вартість сільськогосподарських відходів в харчовому секторі становить 750 мільярдів доларів США на основі цін виробників (без урахування рибальства та морепродуктів). Світові лідери, уряди та екологічні організації зобов'язалися вирішити цю проблему шляхом зменшення харчових відходів, перенаправлення їх зі сміттєзвалищ та переробки відходів. Цю проблему також можна вирішити та зробити її більш перспективною шляхом автоматизації системи класифікації та управління відходами шляхом додавання штучного інтелекту [5].

1.2. Класифікація органічних та перероблюваних відходів

Відходи здебільшого потрапляють на звалища. Основними проблемами, пов'язаними зі звалищами, є збільшення кількості токсинів, споживання токсичних матеріалів тваринами та забруднення ґрунту та води. Основна проблема стосується токсинів та відходів домогосподарств.

Дані можна поділити на два класи: органічні та перероблювані матеріали. Розумне автоматизоване сортування призводить до меншого заповнення звалищ токсинами. Завдання класифікації перероблюваних відходів від органічних предметів та різних типів відходів вимагає автоматизованого підходу як простого, але точного рішення.

Багато авторів опублікували підходи, що базуються на машинному навчанні та глибокому навчанні. Розвиток моделей машинного навчання (ML) та глибокого навчання (DL) відіграє значну роль у вирішенні вищезгаданих проблем з використанням комп'ютерного зору. Дослідження класифікації та управління відходами були проведені на основі популярних алгоритмів ML, таких як штучна нейронна мережа (ANN), MAPMobileNet-18, Mobile-NetV3, Mobile-NetV2, а також

популярна згорткова нейронна мережа (CNN), яка широко використовується в класифікації, розпізнаванні та сегментації зображень [5].

Працюючи над покращенням існуючих результатів класифікації відходів, в роботі [6] використовують модель CNN та додають покращення на рівні шарів з використанням регуляризації. Модель експериментально випробувана з глибиною мережі та тонким налаштуванням гіперпараметрів на основі регуляризації. Показані результати є найкращими комбінаціями.

1.3. Огляд глибоких архітектур нейронних мереж

Згорткові нейронні меержі складаються з шару вилучення ознак (згорткового шару) та класифікатора (повнозв'язного шару). Шар класифікатора складається з великої кількості нейронів. Вилучення може ефективно генерувати результати класифікації. Майбутньою альтернативою цьому можуть бути згорткові шари для підвищення ефективності результатів. Спостерігалося багато покращень у доповненні даних та змінах на основі шару класифікатора, в роботі [6] було проведено в CNN з використанням покращень на основі шару активації для кращого вилучення ознак та налаштування гіперпараметрів разом з вилученням на рівні класифікатора в класифікації органічних та перероблюваних відходів.

Крім того, класифікацію досліджували за допомогою моделей на основі трансферного навчання разом із покращеною CNN. Продуктивність порівнюли для VGG16, VGG19, MobileNetV2, DenseNet121 та EfficientNetB0. Результати моделі Light MobileNetV2 демонструють кращу якість точність [7].

Використання CNN полягає у виявленні перероблюваних органічних відходів, що визначає проблему як бінарну класифікацію. Набір даних містить зображення високої роздільної здатності з різноманітним фоном.

Розділення відходів – це як ручний, так і автоматизований процес. Ручний процес – це процес, під час якого працівники або робоча сила розділяють відходи на різні типи для утилізації або переробки. Ручний процес схильний до багатьох проблем, таких як:

– людська помилка: працівники можуть допускати помилки під час сортування відходів. Це впливає на ефективність роботи.

– невідповідність в інтерпретації матеріалу: різні люди можуть мати різне розуміння класу відходів, що може призвести до невідповідного сортування;

– обмежена масштабованість: зі збільшенням кількості відходів та звалищ ручна праця може стати складною через її обмежені можливості масштабування. Великий обсяг, що призводить до обмеженої масштабованості, призводить до неефективного процесу;

– збільшення вартості робочої сили: збільшення кількості відходів призведе до збільшення робочої сили, до збільшення експлуатаційних витрат;

– відсутність зворотного зв'язку щодо процесу: ручна робота не забезпечує жодного зворотного зв'язку щодо оптимізації відходів, що, обмежує можливості оптимізації процесів переробки та вдосконалення стратегій управління відходами.

Щоб вирішити ці проблеми, багато систем управління відходами додають автоматизовані технології для підвищення продуктивності та ефективності процесів класифікації відходів. Автоматизовані процеси можуть допомогти зменшити кількість помилок та покращити масштабованість. Щоб вирішити ці проблеми, багато систем управління відходами включають автоматизовані технології, такі як робототехніка та штучний інтелект, для підвищення точності та ефективності процесів класифікації відходів. Автоматизовані системи можуть допомогти зменшити кількість помилок, покращити масштабованість та мінімізувати ризики для здоров'я та безпеки, які пов'язані з ручним сортуванням [8].

У статті [9] автори представили автоматичну техніку категоризації перероблюваного сміття, щоб відтворити старі методи обробки трьох категорій сміття: пластикових пляшок, паперу та банок з-під газованої води. Вони назвали точність з використанням різних класифікаторів як 85,70 % для тонкого дерева, 88,10 % для упакованого дерева, 78,60 % для K-найближчого сусіда (KNN), 81,00 % для зваженого KNN, 92,90 % для лінійної машини опорних векторів (SVM). Вони досягли точності 83,3 % для пляшок, 100 % точності для банок з-під газованої води та 100 % точності для паперу. Там, де розмір набору даних був незрозумілим,

категорії з постійно зростаючими даними потребували інших моделей, таких як моделі на основі глибокого навчання. В роботі [10] запропоновано інтелектуальний класифікатор відходів, який працює на моделі ResNet-50, що забезпечувала екстрактор та SVM, які класифікували сміття на кілька груп, таких як скло, метал, папір та пластик. Автори [11] навчили та порівняли кілька архітектур глибокого навчання для автоматизованих систем управління відходами, використовуючи набір даних TrashNet. Були досліджені проекти нейронних мереж (CNN), включаючи VGG, Inception та Residual Neural Networks (ResNet). Комбінований фреймворк Inception-ResNet забезпечив максимальний результат класифікації з точністю 88,6 %.

У статті [12] автори представили підхід, що базується на принципах глибокого навчання та комп'ютерного зору, для класифікації відходів за шістьма основними категоріями, такими як скло, метал, папір, пластик, картон та інші, за допомогою зображень відходів. Навчальний набір даних був підготовлений з зображень з інтернет-сайтів та застосований до багатошарового використання фреймворку CNN, зокрема моделі Inception-v3, для класифікації відходів з коефіцієнтом точності 92,5%. Крім того, було досліджено різноманітні методології для вичерпного резюме. Моделі, що були протестовані, використовували SVM з використанням ознак HOG, базову CNN та CNN із залишковими блоками [13].

Результати оцінювання довели, що базові мережі CNN із залишковими блоками або без них добре функціонують. Пізніше у [14] було представлено фреймворк ResNet18, який зосереджений на методах реалізації категоризації біорозкладних відходів. Скло, металеві елементи, пластик та папір були серед типів відходів, які програма могла автоматично витягувати для категоризації. Алгоритм мав точність прогнозування 92 % у категоризації перероблюваного сміття, що свідчить про його здатність ефективно виявляти перероблюване сміття, згідно з експериментальними даними. Дослідження [6] представило систему, яка включала добре налаштовану структуру VGG-19 для категоризації 95 різних видів великогабаритних відходів, три гібридні функції для ефективного управління проблемою незбалансованого класу, такі як вага класу VGG19, XGB VGG19 та Light Gradient Boosting (LGB) VGG19, та

великий набір даних з 95 класами. Аналіз показав, що система перевершила існуючі методи у виявленні великогабаритного сміття з ефективністю 86,19 %.

Пізніше, з ідеєю покращення CNN обговорили застосування автоматизованого навчання для вирішення реальної проблеми в реальному житті [7]. У своїй моделі вони використали функцію активації випрямленої лінійної одиниці Relu для покращення запропонованої архітектури CNN, і запропонована архітектура отримала кінцеву точність 79 %. Також в роботі [8] представили унікальну систему під назвою подвійне злиття, яка використовувала підходи злиття ознак та рівня оцінки для ефективного об'єднання двох моделей глибокого навчання. Метод подвійного злиття підтвердив, що глибокі моделі оптимально сприяють, спочатку інтегруючи свої компетенції в початкову та відкладену структуру злиття, а потім інтегруючи отриману продуктивність класифікатора за допомогою методів первинного та відкладеного злиття на рівні оцінки. Вони оцінили точність, середню прецизійність, повноту та F1-оцінку кожної глибокої моделі окремо та помітили, що ResNet-101 показав непогані результати. Автори в [9] представили гібридну технологію, що базується на багатошаровому сприйнятті та CNN, яка діє як інтелектуальний класифікатор відходів у режимі реального часу, для класифікації відходів за відповідними категоріями. CNN визначила категорію неметалевого сміття, тоді як багатошарове сприйняття визначає бінарну класифікацію, як сміття з металевими або неметалевими властивостями. Навчена модель мала точність 99 % з тестовим набором даних, що є надзвичайно надійним та послідовним. Автори в [10] представили факторний аналіз для обстеження місця та обробки даних про сприйняття мешканцями утилізації та управління та використали CNN для категоризації та розпізнавання зображень сміття, що було використано для допомоги у прийнятті рішень щодо класифікації сміття, з точністю 85,32 %. Пізніше було використано набір даних TrashNet, відомий стандартний набір даних із загальною кількістю 2527 фотографій, розділених на шість типів відходів, для оцінки ефективності згорткових нейронних мереж та запропоновано оптимізовану DenseNet121, використовуючи генетичний алгоритм (ГА) з повністю зв'язаним шаром DenseNet121 для підвищення точності DenseNet121 [11]. Порівнюючи його з

згортковими нейронними мережами, що використовувалися в попередніх експериментах, оптимізована DenseNet121 досягла високої точності 99,6 %.

1.4. Розумні контейнери на основі штучного інтелекту для ефективної класифікації відходів

Зростання утворення твердих побутових відходів (ТПВ) створює значні екологічні проблеми, що вимагає передових рішень для управління відходами. В роботі [12] представлено роботизовану систему сортування на базі штучного інтелекту, яка призначена для автоматизації та оптимізації процесів класифікації відходів. Система інтегрує передові методи глибокого навчання, зокрема модель VGG16, з надійними апаратними компонентами, такими як Raspberry Pi 4 та камера Logitech C920, для досягнення високоточного сортування відходів. Обробка зображень у режимі реального часу та точні алгоритми класифікації дозволяють системі розрізняти вологі, сухі та електронні відходи з точністю до 98 %. Мінімізуючи потребу в втручанні людини, запропонована система підвищує ефективність сортування, покращує коефіцієнти відновлення матеріалів та усуває ключові недоліки традиційних методів управління відходами. Ця іновація не лише сприяє зменшенню залежності від сміттєзвалищ, але й сприяє екологічній стійкості шляхом оптимізації використання ресурсів та зменшення екологічного сліду. Результати дослідження підкреслюють потенціал систем на базі штучного інтелекту для трансформації управління відходами, пропонуючи масштабований та ефективний підхід до пом’якшення впливу ТПВ на навколишнє середовище.

Рисунок 1.1 – Розумний контейнер

Експоненціальне зростання утворення твердих побутових відходів (ТПВ) є однією з найактуальніших екологічних проблем 21 століття. З огляду на те, що поточне світове виробництво ТПВ перевищує 2 мільярди тонн на рік, а прогнози вказують на зростання до 3,5 мільярда тонн до 2050 року, масштаби цієї кризи вимагають негайної уваги та іноваційних рішень. Це збільшення утворення відходів нерозривно пов’язане з прискоренням урбанізації, промисловим розширенням та зміною моделей споживання, особливо в країнах, що розвиваються. Ця невідповідність між утворенням відходів та можливостями управління ними підкреслює системні проблеми, з якими стикаються країни, у своєму прагненні до рішень для управління відходами.

Поточна парадигма управління відходами стикається з численними проблемами, включаючи неефективні практики сортування відходів, недоліки інфраструктури та обмежену участь громадськості. Ці недоліки призвели до нестійкої залежності від звалищ, які створюють значні екологічні ризики через забруднення ґрунту та води, а також сприяють викидам парникових газів. Оскільки утворення твердих побутових відходів продовжує зростати, необхідність ефективних рішень для управління відходами стає дедалі нагальнішою. Хоча з’явилися різні технологічні рішення для сортування та класифікації відходів, кожне з них має свої переваги та обмеження. Традиційні методи, такі як сортування за

допомогою вихрових струмів, пропонують помірну точність для кольорових металів за низькою вартістю, але їх матеріальний обсяг залишається обмеженим. Інші технології, такі як лазерно-індукована пробійна спектроскопія, досягають вищої точності для металів та пластмас, хоча високі витрати на їх впровадження створюють перешкоди для їх впровадження. Системи рентгенівського пропускання демонструють вражаючу точність для металів та певних видів пластику, таких як ПВХ, але їх середню та високу вартість слід зважувати з їхньою високою продуктивністю. Традиційне оптичне сортування забезпечує економічно ефективне рішення з помірною точністю для пластику, паперу та скла, тоді як спектральна візуалізація досягає вищої точності, але за значно вищих витрат.

Системи розумних контейнерів на основі штучного інтелекту стають переконливим рішенням, пропонуючи чудову продуктивність за кількома параметрами. Ці системи досягають найвищого діапазону точності при обробці різноманітних матеріалів завдяки налаштовуваним алгоритмам сортування. Незважаючи на середні та високі початкові інвестиції, їхня здатність обробляти помірні та великі обсяги, зберігаючи при цьому стабільну точність для різних потоків відходів, позиціонує їх як більш універсальне та ефективне рішення порівняно з традиційними методами [12].

Сучасні проблеми управління відходами вимагають іноваційних рішень, що виходять за рамки традиційних методів сортування. В основі системи розумних сміттєвих баків лежать моделі глибокого навчання, зокрема згорткові нейронні мережі, які демонструють майстерність у класифікації відходів. Ці нейронні мережі проходять навчання на різноманітних наборах даних зображень маркованих відходів, що дозволяє їм розрізняти різні матеріали з високою точністю. Завдяки постійному доступу до нових даних ці моделі демонструють поступове покращення точності класифікації, ефективно знижуючи рівень забруднення в відсортованих потоках відходів, одночасно оптимізуючи потенціал відновлення ресурсів. Автоматизація цього процесу значно мінімізує потреби в втручанні людини, що призводить до зниження експлуатаційних витрат та мінімізації помилок у рішеннях щодо класифікації.

Інтеграція технології комп'ютерного зору в архітектуру розумного контейнера є вирішальним кроком у переробці відходів у режимі реального часу. Камери високої роздільної здатності та складні датчики фіксують детальні зображення відходів, щойно вони потрапляють у систему. Ці зображення негайно обробляються за допомогою алгоритмів глибокого навчання, які аналізують характеристики матеріалу та виконують точні рішення щодо сортування на основі попередньо встановлених параметрів класифікації. Ця обробка забезпечує ефективне сортування відходів у місці утилізації, максимізуючи можливості переробки та коефіцієнти відновлення матеріалів. Окрім підвищення точності сортування, інтеграція штучного інтелекту та глибокого навчання в розумний контейнер сприяє збору та аналізу даних, пропонуючи інформацію для подальшої оптимізації. Ці технології разом дозволяють створити масштабоване, адаптивне рішення, яке може вирішувати зростаючі проблеми управління відходами, допомагаючи створити більш сталу екосистему відходів.

Роботизована система сортування відходів використовує модель класифікації зображень, навчену на наборі даних про сміття, для автоматичної класифікації та сортування відходів за категоріями: сухі, вологі та електроніка. Система класифікації відходів складається з камери Logitech C720, підключеної до Raspberry Pi 4. Камера постійно знімає кадри та запускає модель класифікації відходів у режимі реального часу.

Система сортування відходів складається з конвеєрної стрічки, яка відповідає за доставку відходів до відповідного контейнера після сортування, та системи обертових дисків з трьома контейнерами відповідно для сухих, вологих та електронних відходів. Після того, як відходи розміщені на конвеєрній стрічці, камера фіксує кадри, модель класифікації відходів класифікує їх. Залежно від типу класифікованих відходів, двигун механізму обертових дисків запускається, щоб розмістити відповідний контейнер у кінці конвеєрної стрічки. Для точного позиціонування контейнерів розміщені кінцеві вимикачі, які спрямовують відходи до правильного контейнера. Після того, як правильний контейнер розміщено в кінці

конвеєрної стрічки, запускається двигун конвеєрної стрічки, доставляючи відходи до контейнера.

Рисунок 1.2 – Прототип розумного сміттєвого бака: камера 1, баки для відходів 2, конвеєрна стрічка 3, механізм обертового диска 4, кінцеві вимикачі 5.

Ця методологія дослідження полягає в розробці та впровадженні роботизованої системи сортування твердих побутових відходів на основі штучного інтелекту. Методологія включає вибір апаратних компонентів (камер, датчиків, двигунів, контролерів) та розробку моделі глибокого навчання для класифікації відходів. Процедура сортування включає захоплення зображень відходів, їх класифікацію за допомогою моделі штучного інтелекту та спрямування відходів до призначених контейнерів за допомогою моторизованих механізмів. Ця методологія спрямована на покращення обробки відходів та виявлення можливостей переробки.

Система призначена для виконання таких функцій.

- знімання зображення відходів та його надсилання до системи;
- класифікація захопленого зображення як вологе, сухе або електронне;

- переміщення об’єкта до відповідного сміттєвого контейнера.

Для реалізації цих функцій система повинна складатися з трьох основних компонентів: система класифікації сміття, система конвеєрної стрічки, система обертових дисків. У роботі [12] для класифікації відходів використовуються моделі ResNet-50 та VGG-16, Raspberry Pi 4 керує камерою, яка знімає зображення, системою конвеєрної стрічки, яка складається з ременя та двигуна для доставки відходів до відповідного контейнера, системою обертових дисків, яка складається з двигуна, трьох контейнерів та кінцевих вимикачів для позиціонування відповідного контейнера після отримання результатів від двигуна класифікації сміття.

Набір даних, використаний у цьому дослідженні, був зібраний вручну шляхом зйомки зображень відходів, розділених на три класи: сухі, вологі та електроніка. Загалом було отримано 1646 зображень, з яких 1196 використано для навчання, а 450 зображень для тестування. Зображення були зроблені за допомогою стандартної цифрової камери та організовані за відповідними категоріями в окремих папках для кожного класу. Щоб зображення відповідали реальним умовам, вони відрізнялися освітленням, фоном та орієнтацією. Зібрані зображення були позначені відповідно до класу, до якого вони належать: сухі, вологі або електроніка. Ці зображення сформували набір даних, який згодом був використаний для навчання та оцінки моделей глибокого навчання.

Зібрані зображення були попередньо оброблені для підготовки до навчання та оцінки моделей глибокого навчання. Усі зображення були змінені на стандартний розмір вхідного даних 224 x 224 пікселів, щоб відповідати вимогам до вхідних даних моделей ResNet50 та VGG16. Значення пікселів зображень були нормалізовані до діапазону від 0 до 1 шляхом ділення кожного значення пікселя на 255. Для подальшого покращення набору даних та зменшення перенавчання були застосовані методи доповнення даних. Вони включали випадкові обертання, масштабування, горизонтальні перевороти та зміщення, що вносило мінливість у зображення.

Набір даних був розділений на навчальні та валідаційні набори, причому 70 % від загальної кількості зображень (1196 зображень) було виділено для навчання, а

решта 30 % для валідації. Такий поділ гарантував, що моделі могли добре узагальнюватися та запобігав перенавчанню. Навчальний набір використовувався для підгонки моделей, тоді як валідаційний набір для моніторингу продуктивності моделі під час навчання.

У дослідженні [13] використовувалися моделі ResNet50 та VGG16, обидві є добре зарекомендували себе як згорткові нейронні мережі (CNN) в області класифікації зображень. ResNet50 використовує фреймворк залишкового навчання, який включає пропуски з'єднань, що дозволяють створювати глибші мережі, пом'якшуючи проблему зникнення градієнта. VGG16 - це простіша архітектура з серією невеликих згорткових фільтрів 3x3 та повністю зв'язаних шарів. Обидві моделі були модифіковані шляхом видалення верхніх шарів (повністю зв'язаних шарів) та заміни їх спеціальними щільними шарами, що підходять для завдання класифікації трьох класів. Остаточний вихідний шар обох моделей містить три нейрони, що відповідають класам: Dry (сухий), Wet (мокрий) та Electronics (електроніка). Для виведення ймовірностей класів використовувалася функція активації SoftMax, оскільки вона використовується в задачах класифікації з кількома класами. Для обох моделей попередньо навчені ваги використовувалися для вилучення ознак і лише останні кілька шарів були точно налаштовані для оптимізації продуктивності для цього конкретного завдання класифікації.

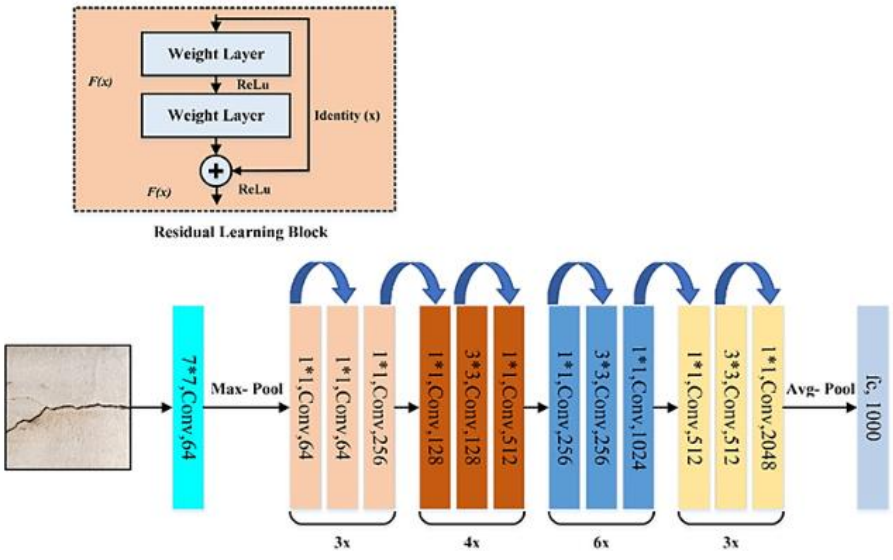

Рисунок 1.3 – Архітектура ResNet50

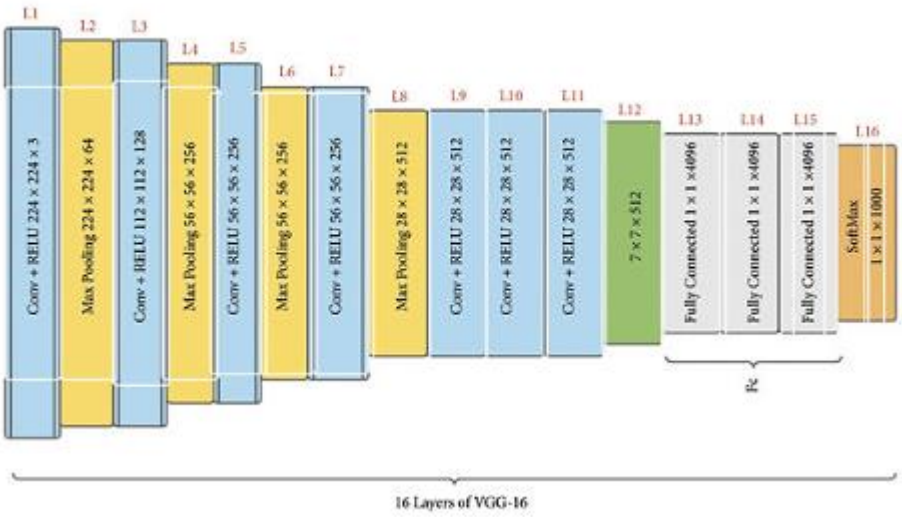

Рисунок 1.4 – Архітектура VGG16

Обидві моделі були навчені за допомогою оптимізатора Adam, відомого своїми можливостями адаптивного навчання, з початковою швидкістю навчання 0,0001. Моделі навчалися протягом 10 епох, протягом яких відстежувалися втрати та інші показники продуктивності. Окрім втрат, для оцінки продуктивності моделей контролювалися інші показники, такі як точність, повнота та F1-оцінка. Ці показники надають глибше розуміння того, наскільки добре моделі справляються з дисбалансом класів, який може бути присутнім у реальних наборах даних. Моделі навчалися на графічному процесорі для пришвидшення обчислень, а навчальні та валідаційні набори даних оброблялися пакетами, кожен пакет яких містив певну кількість зображень.

Після кожної епохи оцінювалася продуктивність валідаційного набору для перевірки на наявність перенавчання. Якщо модель демонструвала ознаки перенавчання, її було завчасно зупинено, щоб запобігти подальшому навчанню. Процес навчання мав на меті мінімізувати функцію втрат, одночасно максимізуючи точність, повноту та F1-оцінку, забезпечуючи не лише точність, але й надійність моделей у класифікації різних типів відходів.

ВИСНОВКИ ДО РОЗДІЛУ

В даному розділі представлено аналіз стану предметної області, який показує зростаючу проблему неправильної утилізації та переробки побутових відходів. Існують різноманітні методи сортування відходів, але вони часто є недостатньо автоматизованими і неконкурентоспроможними. Використання сучасних інформаційних систем і технологій штучного інтелекту в цій галузі є перспективним рішенням для підвищення ефективності. Проведений огляд сучасних технологій дозволяє зробити висновок про необхідність розробки нових систем автоматичного сортування. Недосконаліть існуючих підходів підкреслює потребу у створенні більш точних та швидких іноваційних рішень. Обгрунтовано актуальність застосування глибокого навчання для розпізнавання та класифікації побутових відходів за їх зовнішніми ознаками. Аналіз предметної області продемонстрував можливості інтеграції IoT-технологій для покращення збору та обробки даних про відходи. Важливим є врахування економічних, екологічних та соціальних аспектів сучасної системи управління відходами.

РОЗДІЛ 2. ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Загальна характеристика інтелектуальної системи класифікації побутових відходів

Інтелектуальна система класифікації побутових відходів є спеціалізованим програмним продуктом, розробленим для автоматичного розпізнавання та класифікації сміття за допомогою алгоритмів глибокого навчання. Вона належить до класу систем комп'ютерного зору, які опрацьовують зображення предметів для подальшого прийняття рішень на основі виявлених характеристик.

Метою створення системи є підвищення ефективності сортування відходів на ранніх етапах обробки, зниження навантаження на працівників сміттєсортувальних станцій, а також сприяння свідомій поведінці населення шляхом автоматизації процесу визначення категорії сміття.

Основними функціональними можливостями системи є:

- автоматичне визначення типу побутового відходу (пластик, скло, папір, метал, органічні відходи тощо) на основі фото або відеопотоку;
- виведення результату класифікації у зручному для користувача форматі (текстовий опис, іконка, голосовий супровід);
- збереження результатів класифікації до бази даних для подальшого аналізу та оптимізації процесів сортування;
- можливість масштабування для роботи з мобільними, веб- або вбудованими пристроями (смарт-контейнери, роботизовані сортувальні системи).

Інформаційна система побудована на основі згорткової нейронної мережі, яка виконує навчання на великій кількості зображень побутових відходів. Мережа навчається виділяти характерні ознаки різних категорій сміття, що дозволяє системі досягати високої точності класифікації навіть у випадках поганого освітлення, фону або часткового перекриття об'єктів на зображенні.

Система орієнтована на декілька категорій користувачів:

- оператори сортувальних станцій, яким необхідна автоматизована підтримка при сортуванні великих обсягів сміття;

- органи місцевого самоврядування та екологічні служби, зацікавлені в підвищенні рівня переробки відходів;
- широке коло користувачів, які можуть застосовувати систему в мобільному застосунку для самостійного сортування вдома.

Ключовими перевагами даної інформаційної системи є висока точність класифікації завдяки застосуванню глибоких нейронних мереж, можливість роботи в реальному часі, гнучкість у застосуванні для різних платформ (мобільних, вбудованих, десктопних), простота використання завдяки зрозумілому інтерфейсу.

2.2 Архітектура інтелектуальної системи

Архітектура інформаційної системи класифікації побутових відходів є багаторівневою та модульною, що дозволяє забезпечити гнучкість, масштабованість і ефективність її роботи. Кожен рівень виконує окремі функції та взаємодіє з іншими модулями через чітко визначені інтерфейси. Архітектура передбачає як локальне, так і віддалене розгортання (у хмарному середовищі або на вбудованих пристроях).

Загальна структура системи включає такі основні модулі.

1. Модуль введення даних. Цей модуль відповідає за отримання зображень побутових відходів. Джерелами вхідних даних можуть бути:

- камера реального часу (веб-камера або вбудована камера смарт-контейнера);
- графічний інтерфейс користувача, через який завантажуються зображення;
- база даних, що містить приклади для тестування та навчання системи.

Зображення, отримані на цьому етапі, передаються до модуля попередньої обробки.

2. Модуль попередньої обробки зображень. Для досягнення стабільної роботи нейронної мережі потрібно уніфікувати всі зображення. До типових етапів попередньої обробки належать:

- масштабування зображень до стандартного розміру (128×128 або 256×256 пікселів);
- нормалізація піксельних значень (переведення їх у діапазон [0,1]);
- перетворення кольорового простору (RGB або Grayscale);

- аугментація даних (повороти, дзеркальне відображення, шум) для покращення генералізації моделі.

Результатом є однорідний вхідний масив даних, готовий до подачі в модель глибокого навчання.

3. Модуль класифікації. Центральним елементом архітектури є згорткова нейронна мережа (Convolutional Neural Network, CNN), яка виконує основну функцію класифікацію зображень. Структура моделі включає:

- вхідний шар, який приймає зображення;
- кілька згорткових шарів Conv2D з фільтрами для виділення ознак;
- шари активації ReLU;
- шари субдискретизації для зменшення розмірності;
- повнозв'язні шари Dense на етапі класифікації;
- вихідний Softmax-шар, який повертає ймовірності належності зображення до кожного з класів.

Модель навчається на великій кількості прикладів сміття, які розмічені вручну.

4. Модуль інтерпретації результатів. Після класифікації модель повертає назву класу, до якого відноситься об'єкт. Цей модуль обробляє результати класифікації, формує відповідь для користувача, передає дані до інтерфейсу або в базу даних.

Також можливе додавання повідомлень про рекомендовану утилізацію або сортувальні поради.

5. Модуль інтерфейсу користувача. Інтерфейс забезпечує взаємодію користувача із системою. Можливі варіанти реалізації: графічний інтерфейс GUI на основі бібліотеки Tkinter або PyQt; вебінтерфейс за допомогою Flask або Streamlit.

Користувач завантажує зображення, бачить результат класифікації та отримує підказки.

6. Блок збереження та обліку даних. Для аналізу ефективності роботи системи та подальшого вдосконалення передбачено базу даних, у яку записуються зображення та результати класифікації, метадані (дата, час, пристрій, користувач), історія класифікацій, статистика помилок. На етапі прототипу використовується формат CSV.

7. Модуль моніторингу та тестування. Цей модуль призначений для перевірки точності роботи моделі на основі тестової вибірки, відстеження продуктивності системи, візуалізації метрик, таких як Accuracy, Precision, Recall, F1-score, звітування про невдалі класифікації для подальшого аналізу.

Рисунок 2.1 – Візуальна схема модуля моніторингу та тестування

Переваги обраної архітектури:

- модульність: кожен компонент можна окремо тестувати й покращувати;
- масштабованість: можливість розгортання на мобільних або вбудованих пристроях;
- гнучкість: легке оновлення моделі або зміна інтерфейсу;
- продуктивність: оптимізація обчислень завдяки згортковим шарам і попередній обробці.

У системі використовується згорткова нейронна мережа, побудована на основі класичної архітектури типу CNN. Мережа навчена класифікувати зображення розміром 128×128×3 (RGB).

Таблиця 2.1 – Структура архітектури CNN

Шар (Layer)	**Тип шару**	**Розмір виходу**	**Кількість параметрів**
Input	Вхідне зображення	(128, 128, 3)	0
Conv2D (32 фільтри)	Згортка	(126, 126, 32)	896
ReLU	Активація	(126, 126, 32)	0
MaxPooling2D (2×2)	Пулінг	(63, 63, 32)	0

Conv2D (64 фільтри)	Згортка	(61, 61, 64)	18,496
ReLU	Активація	(61, 61, 64)	0
MaxPooling2D (2×2)	Пулінг	(30, 30, 64)	0
Conv2D (128 фільтрів)	Згортка	(28, 28, 128)	73,856
ReLU	Активація	(28, 28, 128)	0
MaxPooling2D (2×2)	Пулінг	(14, 14, 128)	0
Flatten	Розгортання	(25088,)	0
Dense (128 нейронів)	Повнозв’язний	(128,)	3,211,392
ReLU	Активація	(128,)	0
Dropout (0.5)	Регуляризація	(128,)	0
Dense (6 нейронів)	Класифікація	(6,)	774
Softmax	Вихід	(6,)	0

Кількість вихідних нейронів 6 відповідає кількості класів відходів: папір, пластик, метал, скло, органіка, інше. Conv2D – згорткові шари з ядром 3×3 і збільшенням кількості фільтрів (від 32 до 128), які автоматично виділяють ознаки об'єкта (краї, контури, текстури). ReLU (Rectified Linear Unit) – нелінійна функція активації, що вводить неглибоку нелінійність у модель. MaxPooling2D зменшує розміри карти ознак, зберігаючи найважливішу інформацію. Flatten перетворює багатовимірні тензори у вектор для подачі у повнозв’язні шари. Dense – повнозв’язні шари, що здійснюють навчання на основі ознак, виділених згортками. Dropout – метод боротьби з перенавчанням, вимикає випадкову частину нейронів під час навчання. Softmax надає ймовірнісний розподіл по всіх класах.

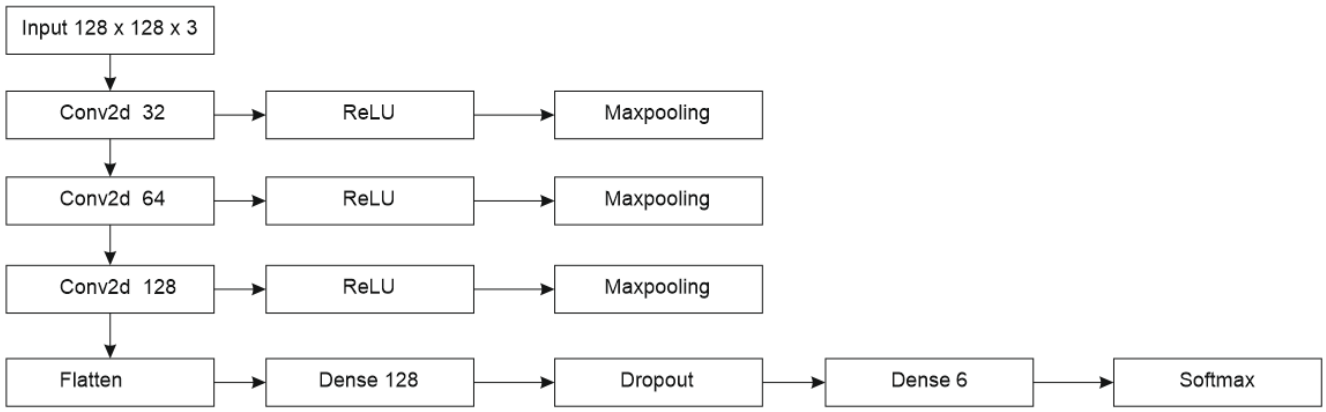

Рисунок 2.2 Структура зготкової нейронної мережі

2.3 Вибір інструментів та програмного забезпечення

У процесі розробки інтелектуальної системи класифікації побутових відходів ключову роль відіграє правильно підібране програмне забезпечення, інструменти програмування, бібліотеки та фреймворки, які забезпечують ефективне розроблення, навчання, тестування та впровадження моделі глибокого навчання. Для реалізації системи було використано такі технології:

- мова програмування Python 3.10;
- фреймворки глибокого навчання TensorFlow, Keras;
- інструменти обробки зображень OpenCV;
- інструменти роботи з даними бібліотеки NumPy, Pandas;
- інтерфейс користувача Tkinter для десктоп-додатка або Streamlit для веб-додатка;
- інструменти візуалізації бібліотеки Matplotlib, Seaborn;
- середовище розробки Google Colab.

Мова програмування: Python 3.10. Python є однією з найпопулярніших мов програмування у сфері штучного інтелекту та глибокого навчання. Її перевагами є великий набір бібліотек для роботи з даними та нейронними мережами, простий та зрозумілий синтаксис, висока швидкість розробки та тестування;

Фреймворки глибокого навчання: TensorFlow 2.x + Keras. Для побудови та тренування згорткової нейронної мережі CNN було обрано TensorFlow у зв’язці з Keras. TensorFlow - відкрита бібліотека машинного навчання від Google, що

підтримує масштабування на кластерах, ефективне виконання моделей. Keras - високорівневий API, що працює поверх TensorFlow та дозволяє швидко будувати, навчати й відлагоджувати моделі нейронних мереж.

Бібліотеки для обробки зображень OpenCV та Pillow. OpenCV (Open Source Computer Vision Library) використовується для попередньої обробки зображень: зменшення розміру, зміна кольорової моделі, фільтрація шуму. Pillow - полегшена бібліотека для базових операцій зображення (відкриття, збереження, обрізка).

Бібліотеки для роботи з даними NumPy та Pandas. NumPy забезпечує операції з багатовимірними масивами, необхідними для навчання моделей. Pandas використовується для аналізу, обробки, фільтрації, групування табличних даних.

Інструменти візуалізації бібліотеки Matplotlib, Seaborn використовуються для відображення графіків точності та втрат під час навчання, побудови матриці невизначеності confusion matrix, візуалізації прикладів класифікацій.

Середовище розробки Google Colab - хмарне середовище, що дозволяє навчати моделі без потреби у потужному локальному комп'ютері. Підтримує пряме збереження на Google Drive.

Інтерфейс користувача Tkinter для десктопу або Streamlit для вебу. Tkinter - стандартна бібліотека Python для створення графічного інтерфейсу користувача GUI. Використовується для десктопного прототипу системи. Streamlit - фреймворк для створення веб-серверів. Використовуються CSV-файли для збереження історії класифікацій, ведення журналу помилок, формування звітності.

2.4 Опис набору даних

Для реалізації інтелектуальної системи класифікації побутових відходів було обрано датасет, який містить зображення різноманітних категорій сміття, призначених для автоматичного розпізнавання та класифікації. Одним із найбільш популярних наборів даних у цій сфері є Garbage Classification Dataset, який включає зображення таких типів відходів, як пластик, метал, скло, органіка, папір, батарейки, текстиль, електроніка [15]. Датасет взято з відкритих джерел платформи Kaggle:

https://www.kaggle.com/datasets/mostafaabla/garbage-classification?utm_source=chatgpt.com

Формат даних: зображення у форматі JPG/PNG, структура організована по папках, де кожна папка відповідає певному класу відходів. Кількість зображень приблизно 5 000 зображень, рівномірно розподілених між класами. Кількість класів 6-10 категорій залежно від обраного набору: Plastic, Paper, Glass, Metal, Organic, Other або General Waste.

Розділення набору: Train навчальна вибірка 70 %, Validation валідаційна вибірка 15 %, Test (тестова вибірка) 15 %. Розмір зображень змінюється, проте для навчання моделі всі зображення були приведені до єдиного розміру 224×224 пікселів, що є стандартним для моделей глибокого навчання.

Класи позначені назвою категорії. Приклади зображень у наборі:

- пластик: пляшки, упаковка від їжі, пакети;
- метал: консервні банки, алюмінієві кришки;
- скло: розбиті та цілі пляшки, банки;
- органіка: залишки фруктів, овочів, їжа;
- папір: газети, картон, серветки;
- інше: змішані відходи, не класифіковані у попередні категорії.

Перед подачею в модель зображення проходили попередню обробку;

- зміна розміру до 224×224;
- нормалізація значень пікселів, перетворення до діапазону [0,1];
- аугментація.

Обраний набір даних дозволяє ефективно навчати та тестувати модель глибокого навчання, що виконує класифікацію сміття за типом. Він є придатним як для побудови експериментальної моделі, так і для створення інтелектуальної системи, що застосовується у сферах автоматичного сортування відходів, екологічного моніторингу та розумних сміттєвих контейнерів.

2.5 Опис логіки роботи інтелектуальної системи

Інтелектуальна система класифікації побутових відходів функціонує як система, що здійснює автоматичне розпізнавання, класифікацію та сортування сміття на основі зображень. Основна логіка роботи системи базується на послідовному виконанні етапів попередньої обробки, розпізнавання зображень за допомогою згорткової нейронної мережі, класифікації відходів за категоріями та подальшої інтеграції з виконавчими механізмами (у випадку впровадження системи на виробництві або сортувальній лінії).

Розглянемо основні етапи логіки роботи системи.

1. Захоплення зображення.

Система отримує зображення побутових відходів за допомогою встановленої камери. Зображення можуть бути отримані в режимі реального часу або завантажені з локального сховища.

2. Попередня обробка зображення:

- зміна розміру зображення до стандартного розміру 128x128 пікселів;
- нормалізація значень пікселів від 0 до 1;
- перетворення зображення до формату, що відповідає вхідному шару нейронної мережі.

3. Аналіз зображення за допомогою CNN. Попередньо навчена згорткова нейронна мережа приймає оброблене зображення як вхідні дані та здійснює виявлення характерних ознак для подальшої класифікації.

4. Класифікація об’єкта. На виході моделі система формує прогноз щодо належності об’єкта до однієї з наперед визначених категорій відходів, таких як пластик, папір, скло, метал, органічні відходи, змішані/ інше. У разі, якщо ймовірність класифікації нижча за визначений поріг, система може надіслати об’єкт на повторну обробку або вивести повідомлення про невизначеність.

5. Візуалізація результату та передача керуючих сигналів. Кінцевий результат виводиться на екран користувача або передається до механізму сортування - конвеєра, що направляє відходи у відповідні контейнери.

6. Збереження даних для статистики та подальшого навчання. Зображення, результати класифікації та часові мітки зберігаються в базі даних для моніторингу, аналізу ефективності роботи системи, а також для подальшого донавчання моделі.

У системі реалізовано механізм самонавчання: користувач може позначати правильну категорію, якщо система помилилася, ці дані потім використовуються для донавчання моделі. Система підтримує масштабування: у майбутньому можливо додати нові класи відходів або інтегрувати модель у мобільний застосунок для сортування вдома.

ВИСНОВКИ ДО РОЗДІЛУ

У другому розділі досліджено сучасний стан інформаційних систем, що використовуються для автоматизованого сортування побутових відходів. Аналіз описаних систем показав їхні переваги у швидкості обробки та точності класифікації, але й існуючі недоліки, що потребують удосконалення. Розглянуто архітектурні підходи та модулі, що забезпечують функціональність інформаційної системи в умовах реального застосування. Виявлено ключові компоненти інфраструктури, потрібні для інтеграції сенсорних пристроїв, баз даних та алгоритмів обробки даних. Особливу увагу приділено процесам збору, збереження та аналізу даних для підвищення ефективності роботи системи. Висвітлено роль класифікаційних метрик і алгоритмів машинного навчання у підвищенні точності автоматичного розпізнавання відходів. Ступінь інтеграції різних компонентів системи дозволяє забезпечити масштабованість і гнучкість у її функціонуванні. Виявлено важливість оптимізації процесів обробки даних для зменшення часу реагування та підвищення ефективності системи. Інформаційне забезпечення є ключовим аспектом успішної реалізації системи класифікації побутових відходів і визначає її перспективи подальшого розвитку.

РОЗДІЛ 3. МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Математична модель інтелектуальної системи класифікації побутових відходів

Математичне забезпечення інтелектуальної системи класифікації побутових відходів включає опис математичних моделей, алгоритмів глибокого навчання та метрик оцінювання, що використовуються для вирішення задачі автоматичної класифікації зображень побутових відходів.

Задача класифікації побутових відходів формалізується як багатокласова задача класифікації зображень. Нехай маємо множину зображень $X=\{x_1, x_2,\ldots,x_n\}$, кожне з яких належить одному з k класів $Y=\{y_1, y_2,\ldots, y_k\}$, де $y_i \in \{1, 2,\ldots, k\}$. Метою є побудова функції $f: X\rightarrow Y$, яка на основі ознак вхідного зображення x передбачає його клас y.

Згорткова нейронна мережа. Для розв'язання задачі використовується згорткова нейронна мережа CNN (Convolutional Neural Network), яка добре зарекомендувала себе в задачах комп'ютерного зору. Мережа складається з наступних основних шарів. Згортковий шар Convolutional layer виконує операцію згортки між входом та ядром згортки:

$$S(i,j) = (X * K)(i,j) = \sum_m \sum_n X(i+m, j+n) \cdot K(m,n) \tag{3.1}$$

де X - вхідне зображення, K - ядро згортки, S - вихідна активація.

Шар активації ReLU застосовується до результату згортки:

$$f(x) = \max(0, x) \tag{3.2}$$

Шар підвибірки Pooling зменшує розмірність простору ознак, найчастіше використовується max-pooling:

$$P(i,j) = \max_{(m,n)\in R_{ij}} S(m,n) \tag{3.3}$$

Повнозв'язний шар Fully connected layer обчислює ймовірність належності до кожного з класів. Softmax-шар на виході мережі забезпечує нормалізацію до ймовірностей:

$$\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \quad (3.4)$$

Функція втрат. Для тренування мережі використовується крос-ентропійна функція втрат:

$$\mathcal{L} = -\sum_{i=1}^{k} y_i \cdot \log(\hat{y}_i) \quad (3.5)$$

де y_i - істинна мітка класу (в one-hot представленні), $\hat{y}_i$ - ймовірність, передбачена мережею.

Алгоритм навчання. Навчання моделі здійснюється за допомогою методу зворотного поширення помилки у поєднанні з методом градієнтного спуску:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta \mathcal{L} \quad (3.6)$$

де θ - вектор параметрів моделі, η - швидкість навчання, ∇θ L - градієнт функції втрат.

Метрики якості моделі. Для оцінювання якості класифікації використовуються такі метрики. Accuracy (точність):

$$Accuracy = \frac{\text{Кількість правильних передбачень}}{\text{Загальна кількість прикладів}} \quad (3.7)$$

Precision, Recall, F1-score застосовуються для детального аналізу по класах, особливо коли класи незбалансовані.

Математичне забезпечення системи включає фундаментальні компоненти глибокого навчання, побудову згорткової мережі та ефективне використання оптимізаційних алгоритмів. Такий підхід дозволяє забезпечити високу точність класифікації побутових відходів за їх візуальними ознаками.

3.2. Алгоритмічне забезпечення моделі інтелектуальної системи

Алгоритмічне забезпечення інтелектуальної системи класифікації побутових відходів охоплює сукупність алгоритмів, які реалізують повний цикл обробки вхідних даних, навчання моделі глибокого навчання, а також процес класифікації нових зображень. Система класифікації побутових відходів працює за наступним алгоритмом.

- завантаження даних - імпорт зображень з датасету та відповідних міток класів;
- попередня обробка - зміна розміру зображень, нормалізація, аугментація;
- розбиття на підвибірки - поділ набору даних на навчальну, валідаційну та тестову частини;
- побудова моделі - створення архітектури згорткової нейронної мережі;
- компіляція моделі - вибір функції втрат, метрики та алгоритму оптимізації;
- навчання моделі - проведення епох тренування та оцінюванням на валідаційній вибірці;
- оцінка якості - обчислення точності, F1-міри та інших метрик на тестових даних;
- збереження моделі - збереження навченої моделі у файл для подальшого використання;
- класифікація нових зображень - завантаження моделі та класифікація нових прикладів у реальному часі або в пакетному режимі.

Алгоритм попередньої обробки зображень:

```
def preprocess_image(image):
    image = resize(image, (128, 128))
    image = image / 255.0
    return image
```

У випадку використання бібліотек TensorFlow попередню обробку можна реалізувати за допомогою спеціальних класів або генераторів.

Алгоритм побудови та навчання моделі:

```
model = Sequential([
    Conv2D(32, (3, 3), activation='relu', input_shape=(128, 128, 3)),
    MaxPooling2D(2, 2),
    Conv2D(64, (3, 3), activation='relu'),
    MaxPooling2D(2, 2),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(num_classes, activation='softmax')
])
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

```
model.fit(X_train, y_train,
          epochs=20,
          batch_size=32,
          validation_data=(X_val, y_val))
```

Алгоритм класифікації нового зображення:

```
def classify_image(image_path):
    image = load_image(image_path)
    image = preprocess_image(image)
    image = np.expand_dims(image, axis=0)
    prediction = model.predict(image)
    predicted_class = np.argmax(prediction)
    return class_labels[predicted_class]
```

Алгоритмічне забезпечення системи є ключовим елементом у реалізації всіх етапів класифікації побутових відходів. Воно поєднує сучасні методи комп'ютерного зору, алгоритми глибокого навчання та ефективні стратегії оптимізації. Завдяки цьому забезпечується висока точність класифікації при збереженні швидкодії системи.

Рисунок 3.1 – Блок-схема роботи моделі

ВИСНОВКИ ДО РОЗДІЛУ

У розділі викладено формалізацію задачі автоматичної класифікації побутових відходів як багатокласової задачі машинного навчання. Розглянуто математичні моделі, що описують процес обробки зображень та їх класифікації за допомогою нейронних мереж. Виявлено важливість застосування глибоких навчальних алгоритмів для підвищення точності і надійності системи класифікації. Описано структури моделей, що дозволяють ефективно виділяти ознаки та класифікувати зображення побутових відходів. Визначено і обґрунтовано використання метрик оцінювання якості моделей, таких як точність, повнота та F1-score. Проведено аналіз математичних моделей, що дозволяють оптимізувати процес навчання та підвищити продуктивність системи. Виявлено важливість формалізації задачі для впровадження алгоритмів машинного навчання і автоматичного аналізу даних. Створено математичну основу для реалізації системи, що забезпечує високий рівень точності класифікації в реальних умовах. Математичне забезпечення є фундаментом побудови ефективної та надійної системи автоматичної класифікації побутових відходів.

РОЗДІЛ 4. ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

4.1. Розроблення інтелектуальної системи класифікації побутових відходів

Розглянемо етапи розробки інтелектульної системи:

- перегляд класів у каталозі;
- візуалізація зображень у наборі даних з кожного класу;
- конфігурація даних;
- підготовка та завантаження даних;
- створення генератора для навчального набору;
- створення генератора для тестового набору;
- запис міток у текстовий файл;
- розроблення архітектури моделі;
- компіляція моделі;
- навчання моделі;
- тестування прогнозів;
- збереження моделі для майбутнього використання;
- розгортання моделі як веб-застосунку за допомогою бібліотеки Streamlit.

Розглянемо почергово всі етапи розроблення інтелектуальної системи для класифікації побудових відходів. Перегляд класів у каталозі.

```
import os
import matplotlib.pyplot as plt
from PIL import Image
import math
dir_example = "Data"
classes = os.listdir(dir_example)
print(classes)
```

Отримують список категорій для класифікації, які представлені як піддиректорії у головній папці Data. Кожна піддиректорія відповідає одному класу зображень (plastic, paper, glass). Імпортують модуль os для роботи з операційною системою, бібліотеку Matplotlib для візуалізації даних, клас Image з бібліотеки PIL для роботи з зображеннями, а також математичні функції math. Задають шлях до

директорії з даними, отримують список всіх піддиректорій у папці Data, виводять список знайдених класів

```
['README.md', 'Test', 'Train']
```

Результат виводу означає, що у директорії Data знаходяться три елементи. README.md – це текстовий файл, який містить опис датасету, інструкції чи метадані. Test – папка з тестовими зображеннями, яка використовується для перевірки моделі. Train – папка з тренувальними зображеннями, використовується для навчання моделі. У папках Train і Test є підпапки з назвами класів (Train/plastic, Train/paper).

```
dir_example = "Data/Train"
train_classes = os.listdir(dir_example)
print(train_classes)
```

Результат виводу цього коду покаже список піддиректорій у папці Data/Train, які відповідають категоріям відходів для навчання моделі. Кожна піддиректорія (plastic, paper) – це окремий клас для класифікації. Усередині кожної піддиректорії знаходяться зображення відходів цього типу (Data/Train/plastic/image1.jpg, Data/Train/plastic/image2.jpg). Список train_classes можна використовувати для налаштування моделі, створення міток для навчання.

```
['cardboard', 'glass', 'metal', 'paper', 'plastic', 'trash']
```

Цей результат означає, що у папки Data/Train знаходиться 6 піддиректорій, кожна з яких відповідає окремому класу побутових відходів для навчання моделі:

- cardboard – картон (коробки, пакування);
- glass – скло (пляшки, банки);
- metal – метал (бляшанки, фольга);
- paper – папір (газети, листівки);
- plastic – пластик (пляшки, пакети);
- trash – сміття, яке не належить до попередніх категорій або не підлягає переробці.

Модель матиме 6 вихідних класів, останній шар нейромережі повинен мати 6 нейронів з активацією softmax.

Клас trash часто використовується для іншого сміття, яке не можна віднести до інших категорій.

Візуалізація зображень у наборі даних з кожного класу.

```
dir_with_examples = 'visualize'
files_per_row = 6
files_in_dir = os.listdir(dir_with_examples)
number_of_cols = files_per_row
number_of_rows = int(len(files_in_dir) / number_of_cols)
fig, axs = plt.subplots(number_of_rows, number_of_cols)
fig.set_size_inches(20, 15, forward=True)
try:
  for i in range(0, len(files_in_dir)):
    file_name = files_in_dir[i]
    image = Image.open(f'{dir_with_examples}/{file_name}')
    row = math.floor(i / files_per_row)
    col = i % files_per_row
    axs[col].imshow(image)
    axs[col].axis('off')
except:
  pass
plt.show()
```

Цей код створює візуалізацію зображень із заданої директорії у вигляді сітки. Задають параметри сітки, створюють саму сітку, заповнюють сітку зображеннями. Після чого відображають результат.

Рисунок 4.1 – Приклади зображень побутових відходів із набору даних для навчання інтелектуальної системи сортування побутових відходів

Зображення ілюструють різні категорії відходів, пластик, метал, картон, скло та упаковку, що використовуються для тренування моделі глибокого навчання у системі класифікації.

Імпорт необхідних бібліотек для моделі. Імпортують ключові компоненти для побудови та навчання згорткової нейронної мережі для класифікації зображень.

```
from tensorflow.keras.models import Sequential
from keras.layers import Conv2D, Flatten, MaxPooling2D, Dense, Dropout,
SpatialDropout2D
```

```
from tensorflow.keras.losses import sparse_categorical_crossentropy,
binary_crossentropy
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
```

Sequential - клас для послідовного побудови моделі, шари додаються один за одним.

Шари нейронної мережі. Conv2D - згортковий шар для виявлення просторових ознак у зображеннях. MaxPooling2D - шар підвибірки для зменшення розмірності даних. Flatten перетворює 2D-дані у 1D-вектор для повнозв'язних шарів. Dense - стандартний повнозв'язний шар нейронної мережі. Dropout - регуляризація для запобігання перенавчанню, випадкове відключення нейронів. SpatialDropout2D - просторова версія Dropout для згорткових шарів

Функції втрат та оптимізатори. sparse_categorical_crossentropy - функція втрат для багатокласової класифікації. binary_crossentropy - функція втрат для бінарної класифікації. Adam - популярний оптимізатор з адаптивним швидкістю навчання.

Підготовка даних. ImageDataGenerator - інструмент для потокового завантаження зображень, аугментації даних, нормалізації пікселів, розділення на тренувальні/ валідаційні набори.

Ці імпорти використовуються для побудови архітектури CNN, налаштування процесу навчання, обробки вхідних зображень, застосування методів боротьби з перенавчанням.

Конфігурація даних. Визначають шляхи до директорій з даними для навчання та тестування моделі. Задають шлях до тренувальних та тестових даних.

```
train = 'Data/Train'
test = 'Data/Test'
```

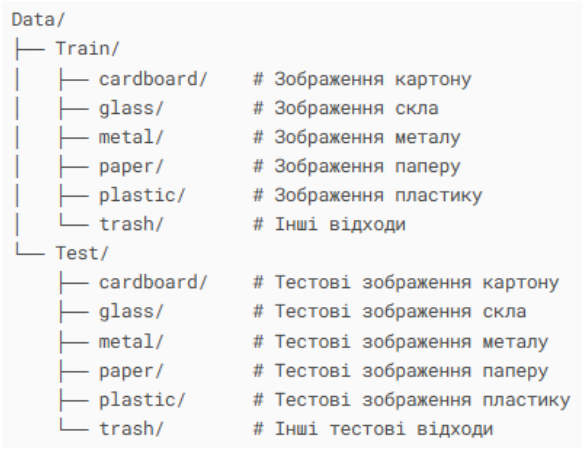

Рисунок 4.2 – Очікувана структура директорій

Ці шляхи використовуються з ImageDataGenerator для завантаження даних, функціями перевірки доступних класів, візуалізацією структури даних. Ця структура є стандартною для задач класифікації зображень і дозволяє завантажувати дані для навчання та оцінки моделі.

Створення генератора для навчального набору. Готують тренувальні дані для нейромережі, яка працює з мітками класів. Створюють генератор даних: ImageDataGenerator - інструмент для підготовки зображень, rescale=1/255 - нормалізують піксельні значення 0-255 → 0-1.

```
train_generator = ImageDataGenerator(rescale = 1/255)
train_generator = train_generator.flow_from_directory(train,
target_size = (300,300),batch_size = 32,class_mode = 'sparse')
labels = (train_generator.class_indices)
print(labels,'\n')
labels = dict((v,k) for k,v in labels.items())
print(labels)
```

Завантажують дані з директорії: train - шлях до тренувальних даних, target_size=(300,300) - зміна розміру всіх зображень до 300×300 пікселів, batch_size=32 - кількість зображень в одному пакеті, class_mode='sparse' - використовується для багатокласової класифікації.

Отримують мітки класів: виводять словник з відповідністю імен класів до числових міток. Інвертують словник міток: створюють зворотне відображення (числова мітка → ім’я класу). Перший словник class_indices використовується при роботі з передбаченнями моделі. Другий інвертований словник зручний для інтерпретації результатів.

```
Found 2184 images belonging to 6 classes.
{'cardboard': 0, 'glass': 1, 'metal': 2, 'paper': 3, 'plastic': 4, 'trash': 5}
{0: 'cardboard', 1: 'glass', 2: 'metal', 3: 'paper', 4: 'plastic', 5: 'trash'}
```

2184 images - загальна кількість тренувальних зображень у всіх категоріях, 6 classes - кількість категорій відходів. Це означає, що набір даних містить 2184 зображення, розподілених між 6 різними типами відходів.

Другий рядок (перший словник) показує відображення імен класів у числові мітки. Кожному текстовому імені класу присвоєно унікальний числовий індекс: cardboard → 0, glass → 1, metal → 2, paper → 3, plastic → 4, trash → 5. Це відображення використовується при навчанні моделі.

Третій рядок (другий словник): це інвертована версія попереднього словника. Він показує відображення числових міток назад у текстові імена класів: 0 → cardboard, 1 → glass, 2 → metal, 3 → paper, 4 → plastic, 5 → trash. Цей словник корисний для інтерпретації результатів передбачення моделі. Модель буде працювати з числами 0-5, але можна перетворювати їх назад у зрозумілі назви класів.

Створення генератора для тестового набору.

```
test_generator = ImageDataGenerator(rescale = 1./255)
test_generator = test_generator.flow_from_directory(test,
target_size = (300,300),batch_size = 32,class_mode = 'sparse')
test_labels = (test_generator.class_indices)
print(test_labels,'\n')
test_labels = dict((v,k) for k,v in test_labels.items())
print(test_labels)
```

Цей код готує тестові дані для перевірки моделі та аналізує мітки класів. 546 images - загальна кількість тестових зображень, 6 classes - кількість категорій. Тестовий набір містить 546 зображень, розподілених між 6 типами відходів.

Другий рядок (перший словник): відображення імен класів на числові мітки у тестовому наборі. Порядок і зіставлення класів повністю збігаються з тренувальним набором. Це гарантує, що модель навчається і тестується на однакових категоріях. Третій рядок (інвертований словник): зворотне відображення для інтерпретації результатів, дозволяє перетворювати числові передбачення моделі назад у текстові мітки.

```
Found 343 images belonging to 6 classes.
{'cardboard': 0, 'glass': 1, 'metal': 2, 'paper': 3, 'plastic': 4, 'trash': 5}
{0: 'cardboard', 1: 'glass', 2: 'metal', 3: 'paper', 4: 'plastic', 5: 'trash'}
```

У тестовій директорії знайдено 343 зображення, розподілених між 6 категоріями відходів. Це означає, що ваш тестовий набір містить в середньому 57 зображень на клас.

Другий рядок (словник) відповідає за відображення назв класів у числові індекси, які використовуватиме модель: cardboard → 0, glass → 1, metal → 2, paper → 3, plastic → 4, trash → 5. Це відповідає тому самому порядку, що й у тренувальних даних.

Третій рядок (інвертований словник) відповідає за зворотне відображення для інтерпретації результатів: 0 → cardboard, 1 → glass, 2 → metal, 3 → paper, 4 → plastic, 5 → trash. Він дозволяє перетворювати числові прогнози моделі назад у зрозумілі назви.

```
for image_batch, label_batch in train_generator:
  break
image_batch.shape, label_batch.shape
```

Отримують один батч даних з тренувального генератора і виводять розмірності отриманих тензорів:

```
((32, 300, 300, 3), (32,))
```

Результат виводуописує структуру одного батча даних, отриманого з train_generator. Для зображень (image_batch.shape = (32, 300, 300, 3)): 32 - кількість зображень у одному батчі (відповідає параметру batch_size=32), 300, 300 - розмір зображень у пікселях (відповідає target_size=(300, 300)), 3 - кількість кольорових каналів (RGB, кожне зображення кольорове).

Для міток label_batch.shape = (32,): 32 - кількість міток у батчі, по одній на кожне зображення. Форма (32,) означає одновимірний масив із 32 цілих чисел, де кожне число - це клас зображення (0 для cardboard, 1 для glass, згідно зі словником class_indices).

```
for image_batch, label_batch in test_generator:
  break
image_batch.shape, label_batch.shape
```

Результат виводу image_batch.shape, label_batch.shape після виконання цього коду покаже структуру одного батча тестових даних. 32 - кількість зображень у батчі. 300, 300 - розмір зображень у пікселях. 3 - кількість кольорових каналів RGB. label_batch.shape = (32,), 32 - кількість міток, що відповідає кількості зображень у батчі. Кожна мітка - це ціле число від 0 до 5 відповідно до словника test_labels.

```
((32, 300, 300, 3), (32,))
```

Результат виводу ((32, 300, 300, 3), (32,)) описує структуру одного батча тестових даних, який генерується test_generator.

Для зображень image_batch.shape = (32, 300, 300, 3): 32 - кількість зображень у батчі, 300, 300 - розмір зображень у пікселях, кожне зображення масштабується до цього розміру, 3 - кількість кольорових каналів RGB. Якщо зображення чорно-білі, тут було б 1.

Для міток label_batch.shape = (32,): 32 - кількість міток, що відповідає кількості зображень у батчі. Кожна мітка — це ціле число від 0 до 5 відповідно до словника test_labels. Наприклад: 0 = "cardboard", 1 = "glass", 2 = "metal", 3 = "paper", 4 = "plastic", 5 = "trash"

Запис міток у текстовий файл.

```
print(train_generator.class_indices)
Labels = '\n'.join(sorted(train_generator.class_indices.keys()))
with open('Labels.txt', 'w') as file:
  file.write(Labels)
```

Цей код виконує дві основні дії: виводить словник класів та зберігає назви класів у текстовий файл.

```
{'cardboard': 0, 'glass': 1, 'metal': 2, 'paper': 3, 'plastic': 4, 'trash': 5}
```

Це словник, де ключі назви класів (піддиректорій у папці Train), значення - числові мітки, які використовує модель для навчання. Це потрібно для перевірки відповідності між іменами класів та їх числовими ідентифікаторами. Після цього зберігають назви класів у файл Labels.txt.

Файл Labels.txt можна використовувати для інтерпретації результатів передбачення моделі, налаштування інтерфейсу (відображення назв класів у веб-додатку), зручного зберігання списку класів для майбутніх експериментів.

Розроблення архітектури моделі. Створюють послідовну модель нейронної мережі Sequential і додають до неї перші два шари: згортковий шар Conv2D та шар підвибірки MaxPooling2D. Sequential - модель, де шари додаються послідовно один за одним, підходить для більшості згорткових мереж.

```
model=Sequential()
model.add(Conv2D(32, kernel_size = (3,3),
padding='same',input_shape=(300,300,3),activation='relu'))
model.add(MaxPooling2D(pool_size=2))
```

Додавання згорткового шару Conv2D: цей шар аналізує зображення за допомогою 32 фільтрів 3x3, виділяє просторові ознаки, зберігає розмірність зображення 300x300. Додавання шару підвибірки MaxPooling2D: цей шар скорочує розмірність зображення з 300x300 до 150x150, вибирає максимальне значення у кожному вікні 2x2, зберігає кількість фільтрів 32.

```
model.add(Conv2D(64, kernel_size = (3,3), padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2))
```

Додають до нейромережі ще один згортковий шар і шар підвибірки. Другий згортковий шар Conv2D: мережа тепер використовує 64 фільтри для аналізу зображень. Кожен фільтр шукає більш складні шаблони. Вхідна розмірність (None, 150, 150, 32) після першого MaxPooling. Вихідна розмірність (None, 150, 150, 64).

Другий шар підвибірки MaxPooling2D: вихідна розмірність: (None, 75, 75, 64), зменшує чутливість до точного розташування ознак, знижує обчислювальну складність. Це потрібно для того, що додаткові шари допомагають виявляти ієрархічні ознаки. Перший шар (32 фільтри): прості краї, кольорові переходи. Другий шар (64 фільтри): форми об'єктів (форму пляшки, кути коробки).

```
model.add(Conv2D(32, kernel_size = (3,3), padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2))
```

Додають до моделі два ключові шари для обробки зображень. Згортковий шар Conv2D: вхід: (batch_size, 300, 300, 3) (RGB-зображення 300x300).
Вихід: (batch_size, 300, 300, 32) (32 фільтри зберегли розмір).

Шар підвибірки MaxPooling2D: вхід: (batch_size, 300, 300, 32). Вихід: (batch_size, 150, 150, 32) (розмір зменшено, кількість фільтрів не змінилась). Conv2D виявляє базові ознаки: горизонтальні/ вертикальні краї, кольорові переходи.

Приклад: для класу скло може активуватись на блискучих ділянках. MaxPooling2D зменшує чутливість до незначних зсувів об’єктів, скорочує обсяг обчислень для наступних шарів. Ця архітектура дозволяє моделі послідовно виявляти спочатку прості, а потім складніші ознаки для класифікації зображень сміття.

```
model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(32,activation='relu'))
```

Цей код завершує архітектуру CNN-моделі, додаючи повнозв’язні шари та механізми регуляризації. Flatten() - вирівнювання даних: перетворює 3D-тензор після останнього Conv2D+MaxPooling в 1D-вектор. Він готує дані для повнозв’язних шарів Dense, які працюють лише з 1D-даними.

Dense(64, activation='relu') - перший повнозв’язний шар. Параметри: 64 - кількість нейронів, relu - функція активації, яка додає нелінійність. Dropout(0.2) – регуляризація: випадково вимикає 20% нейронів (0.2) під час кожного кроку навчання. Запобігає перенавчанню, змушуючи мережу покладатися на різні комбінації ознак. Dense(32, activation='relu') - другий повнозв’язний шар. Призначений для подальшого стиснення інформації (64 → 32 нейрони), додатковий рівень абстракції перед фінальною класифікацією.

```
model.add(Dropout(0.2))
model.add(Dense(6,activation='softmax'))
model.summary()
```

Завершують архітектуру моделі та виводять її зведену інформацію. Dropout(0.2): регуляризація для запобігання перенавчанню. Параметр 0.2 - означає, що 20 % нейронів будуть випадково вимкнені під час кожного кроку навчання. Застосовується до попереднього шару Dense(32).

Dense(6, activation='softmax'): вихідний шар для класифікації. Параметри: 6 - кількість нейронів (відповідає числу класів: cardboard, glass, metal, paper, plastic, trash), softmax - перетворює вихід у ймовірності (сума всіх виходів = 1). Вихід: вектор з 6 значень (ймовірність кожного класу)

model.summary() виводить зведену інформацію про модель.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 300, 300, 32)      896

 max_pooling2d (MaxPooling2D  (None, 150, 150, 32)     0
 )

 conv2d_1 (Conv2D)           (None, 150, 150, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 75, 75, 64)       0
 2D)
conv2d_2 (Conv2D)           (None, 75, 75, 32)        18464

 max_pooling2d_2 (MaxPooling  (None, 37, 37, 32)       0
 2D)

 flatten (Flatten)           (None, 43808)             0

 dense (Dense)               (None, 64)                2803776

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 32)                2080

 dropout_1 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 6)                 198

=================================================================
Total params: 2,843,910
Trainable params: 2,843,910
Non-trainable params: 0
```

Опис моделі глибокого навчання показує повну архітектуру нейронної мережі для класифікації зображень побутових відходів. Вхідні дані. Розмір зображення на вході 300×300×3 (3 канали = RGB). Модель має 12 шарів, з яких 3 згорткові Conv2D, 3 підвибірки MaxPooling2D, 3 повнозв'язні Dense, 2 Dropout і 1 Flatten.

conv2d (Conv2D) → Output: (300, 300, 32). Параметри: 896.

Conv2D – згортковий шар:

- 32 фільтри розміром 3×3;
- виявляє базові ознаки (контури, кольорові переходи);
- кількість параметрів: (3×3×3 + 1) × 32 = 896.

max_pooling2d → Output: (150, 150, 32)

MaxPooling2D – шар підвибірки: зменшує просторові розміри вдвічі для зменшення обчислень.

conv2d_1 → Output: (150, 150, 64). Параметри: 18,496

Conv2D:

- 64 фільтри - більше ознак (текстури, форми);
- параметри: (3×3×32 + 1) × 64 = 18,496

max_pooling2d_1 → Output: (75, 75, 64)

MaxPooling2D.

conv2d_2 → Output: (75, 75, 32). Параметри: 18,464.

Conv2D:

зменшується кількість фільтрів до 32 для виділення специфічних ознак;

Параметри: (3×3×64 + 1) × 32 = 18,464.

max_pooling2d_2 → Output: (37, 37, 32).

MaxPooling2D.

flatten → Output: (43808).

Flatten: перетворює 3D тензор (37×37×32) у вектор для подачі у щільний шар.

dense → Output: (64). Параметри: 2,803,776.

Dense: повнозв’язний шар на 64 нейрони. Має найбільше параметрів, бо кожен з 43808 входів має вагу + зміщення:

(43808 + 1) × 64 = 2,803,776

dropout → Output: (64).

Dropout: Регуляризація, вимикає випадкові нейрони під час навчання для запобігання перенавчанню.

dense_1 → Output: (32). Параметри: 2,080.

Dense: ще один щільний шар для зменшення розмірності перед виходом.

dropout_1 → Output: (32).

Dropout.

dense_2 → Output: (6). Параметри: 198.

Dense (вихідний шар):

- 6 виходів — це 6 класів сміття (пластик, метал, скло, папір, органіка, інше);
- функція активації softmax, щоб отримати ймовірність кожного класу.

Компіляція моделі. Компілюють модель глибокого навчання у Keras, задаючи параметри оптимізації, функцію втрат та метрику оцінки.

```
model.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentropy',
metrics = ['accuracy'])
```

Навчання моделі. train_generator - генератор даних, який повертає батчі зображень та міток. epochs=10 - навчання триватиме 10 епох. steps_per_epoch=2184//32 означає, що на кожну епоху виконується 2184/32 = 68 кроків (батчів), тобто розмір набору даних поділено на розмір одного батчу.

```
model.fit_generator(train_generator, epochs=10, steps_per_epoch=2184//32)
Epoch 1/10
68/68 [==============================] - 118s 2s/step - loss: 1.6845 - accuracy:
0.2783
Epoch 2/10
68/68 [==============================] - 112s 2s/step - loss: 1.4399 - accuracy:
0.4154
Epoch 3/10
68/68 [==============================] - 111s 2s/step - loss: 1.3438 - accuracy:
0.4498
Epoch 4/10
68/68 [==============================] - 111s 2s/step - loss: 1.2584 - accuracy:
0.5046
Epoch 5/10
68/68 [==============================] - 112s 2s/step - loss: 1.1042 - accuracy:
0.5599
Epoch 6/10
68/68 [==============================] - 111s 2s/step - loss: 0.9881 - accuracy:
0.6199
Epoch 7/10
68/68 [==============================] - 111s 2s/step - loss: 0.8935 - accuracy:
0.6733
Epoch 8/10
68/68 [==============================] - 111s 2s/step - loss: 0.7175 - accuracy:
0.7342
```

```
Epoch 9/10
68/68 [==============================] - 111s 2s/step - loss: 0.6144 - accuracy:
0.7876
Epoch 10/10
68/68 [==============================] - 114s 2s/step - loss: 0.5502 - accuracy:
0.7941
```

Цей вивід - результат навчання моделі протягом 10 епох, він демонструє показників точності accuracy та зменшення функції втрат loss.

Таблиця 4.1 – Динаміки змін показників точності та функції втрат

Епоха	**Втрата Loss**	**Точність Accuracy**	**Прогрес**
1	1,6845	27,8 %	Початкова фаза навчання, модель вчиться відрізняти класи.
2	1,4399	41,5 %	Значне покращення, модель вловлює базові патерни.
3	1,3438	44,9 %	Модель починає класифікувати точніше, хоча все ще помиляється.
4	1,2584	50,5 %	Перейдено межу 50 %, вже краще, ніж випадкове вгадування для 6 класів.
5	1,1042	56,0 %	Покращення триває, модель добре навчається.
6	0,9881	62,0 %	Менше помилок, більше правильних класифікацій.
7	0,8935	67,3 %	Понад 2/3 зображень класифікуються правильно.
8	0,7175	73,4 %	Дуже гарний прогрес, видно ефект глибокого навчання.
9	0,6144	78,8 %	Рівень точності високий, помилок дедалі менше.
10	0,5502	79,4 %	Стабілізація, модель наближається до своєї оптимальної якості.

Loss (втрата) зменшується на кожній епосі, що свідчить про те, що модель краще передбачає цільові значення. Accuracy (точність) постійно зростає і досягає 79,4 %, це приємливий результат для задачі багатокласової класифікації.

4.2. Результати роботи інтелектуальної системи класифікації побутових відходів

```
import keras.utils as ku
import numpy as np
import matplotlib.pyplot as plt
```

Імпортують модуль utils з бібліотеки Keras. Цей модуль містить допоміжні функції, наприклад, load_img, img_to_array, які зручно використовувати для підготовки зображень перед передачею їх у модель. Імпортують бібліотеку NumPy, яка використовується для роботи з масивами та числовими операціями. Імпортують модуль pyplot з бібліотеки Matplotlib. Він використовується для побудови графіків та візуалізації, для відображення зображень, графіків точності чи втрат моделі по епохах.

```
print("Probability:",np.max(prediction[0], axis=-1))
predicted_class = labels[np.argmax(prediction[0], axis=-1)]
print("Classified:",predicted_class,'\n')
plt.axis('off')
plt.imshow(img.squeeze())
plt.title("Loaded Image")
```

Цей фрагмент коду завантажує зображення, обробляє його, передає в нейронну мережу, отримує прогноз, виводить класифікацію зображення та ймовірність передбаченого класу. Задають шлях до тестового зображення, яке потрібно класифікувати. Завантажується зображення з диску і змінюється його розмір до 300×300 пікселів, щоб відповідати вимогам моделі. Перетворюють зображення у формат NumPy масиву типу uint8. Нормалізують значення пікселів у діапазон [0, 1], що покращує роботу нейронної мережі. Виводиться найбільше значення ймовірності, яке відповідає передбаченому класу. Визначається назва класу з найбільшою ймовірністю, виводиться текстовий результат класифікації.

Для інтерпретації ймовірності є певні діапазони:

- 80 % - дуже впевнене визначення;
- 60-80 % - помірно впевнене;
- < 50 % - невпевнене, можлива помилка.

```
test_img = 'Data/Test/plastic/plastic425.jpg'
img = ku.load_img(test_img, target_size = (300,300))
```

```
img = ku.img_to_array(img, dtype=np.uint8)
img = np.array(img)/255.0
prediction = model.predict(img[np.newaxis, ...])
print("Probability:",np.max(prediction[0], axis=-1))
predicted_class = labels[np.argmax(prediction[0], axis=-1)]
print("Classified:",predicted_class,'\n')
plt.axis('off')
plt.imshow(img.squeeze())
plt.title("Loaded Image")
```

Цей код виконує класифікацію тестового зображення пластикової пляшки та візуалізує результат.

```
1/1 [==============================] - 0s 50ms/step
Probability: 0.9019071
Classified: plastic
```

Цей результат показує, що модель успішно класифікувала тестове зображення як пластик з дуже високою впевненістю. Оброблено 1 зображення за 50 мілісекунд. Висока точність (> 90 %) свідчить про те, що модель чітко ідентифікувала характерні ознаки пластику, тренувальні дані містили достатньо прикладів цього класу. Швидкість обробки 50 ms дозволяє використовувати модель у реальному часі. Модель чудово справляється з розпізнаванням пластикових предметів.

Рисунок 4.3 – Вхідне зображення пластику для класифікації системою

```
classes = []
probability = []
for i,j in enumerate(prediction[0],0):
  print(labels[i].upper(),':',round(j*100,2),'%')
```

Цей код виводить розподіл ймовірностей по всіх класах для зображення.

```
CARDBOARD : 0.73 %
GLASS : 3.26 %
METAL : 4.2 %
PAPER : 1.41 %
PLASTIC : 90.19 %
TRASH : 0.21 %
```

Цей вивід показує розподіл ймовірностей, з якими модель віднесла зображення до кожної з 6 категорій сміття. Основне передбачення. PLASTIC 90,19 %. Модель дуже впевнено на 90,19 % класифікувала об'єкт як пластик. Це дуже гарний результат, який свідчить про те, що зображення містить чіткі ознаки пластикового предмета, Модель добре навчена розпізнавати цей матеріал. Вірогідно, це типовий представник класу (пляшка, пакет).

Альтернативні гіпотези: METAL 4,2 %. Найбільша альтернативна ймовірність. Може бути пов'язано з блискучими елементами, металевими включеннями в упаковці. GLASS 3,26 %. Можлива схожість з прозорими скляними предметами.

Незначні ймовірності: CARDBOARD (0,73 %), PAPER (1,41 %), TRASH (0,21 %). Практично виключені моделлю, що логічно для пластикового об'єкта. Чітке розрізнення пластику від несхожих матеріалів (папір, картон). Висока впевненість у правильному класі. Потенційні проблеми: модель може плутати пластик з металом/ склом у 4-5 % випадків.

```
test_img = 'Data/Test/paper/paper522.jpg'
img = ku.load_img(test_img, target_size = (300,300))
img = ku.img_to_array(img, dtype=np.uint8)
img = np.array(img)/255.0
prediction = model.predict(img[np.newaxis, ...])
print("Probability:",np.max(prediction[0], axis=-1))
predicted_class = labels[np.argmax(prediction[0], axis=-1)]
print("Classified:",predicted_class,'\n')
plt.axis('off')
plt.imshow(img.squeeze())
plt.title("Loaded Image")
```

Тестують модель на зображенні паперу (шлях Data/Test/paper/paper522.jpg) та виводять результати класифікації.

```
1/1 [==============================] - 0s 25ms/step
Probability: 0.97152126
Classified: paper
```

```
Text(0.5, 1.0, 'Loaded Image')
```

Цей результат демонструє ідеальну роботу моделі при класифікації зображення паперу. Час обробки 25 мілісекунд. Ефективність: оптимально для реального використання в системі сортування.

Рисунок 4.4 – Вхідне зображення паперу для класифікації системою

Виводять детальний розподіл ймовірностей для всіх класів сміття.

```
CARDBOARD : 0.44 %
GLASS : 0.29 %
METAL : 0.61 %
PAPER : 97.15 %
PLASTIC : 1.09 %
TRASH : 0.42 %
```

Цей вивід показує детальний розподіл ймовірностей класифікації зображення моделлю. Модель надзвичайно впевнено на 97,15 % ідентифікувала об’єкт як папір. Це ідеальний результат, який свідчить про хіткі відмінні ознаки паперу на зображенні, добре навчену модель для цього класу, відсутність серйозних альтернативних гіпотез. Цей результат свідчить про відмінну роботу моделі для класифікації паперу, але варто звертати увагу на наявність інших матеріалів навіть з низькою ймовірністю, умови зйомки (освітлення, ракурс), фізичний стан об’єкта (чистота, цілісність).

Збереження моделі як model.h5: виконують збереження навченої моделі глибокого навчання у файл. Формат .h5 HDF5 (Hierarchical Data Format) - бінарний

формат, оптимізований для зберігання великих обсягів даних. Він зберігає всі компоненти моделі в одному файлі, швидкий для завантаження/ збереження.

```
model.save('weights/model.h5')
```

4.3. Розроблення графічного інтерфейсу інтелектуальної системи класифікації побутових відходів

Розроблено модель для класифікації сміття як картону, скла, металу, паперу, пластику або відходів з використанням згорткових нейронних мереж. Сортування відходів означає групування відходів за різними категоріями. Кожні відходи потрапляють до своєї категорії в місці скидання або збору. Потрібно сортувати картон, скло, метал, папір, пластик, сміття та утилізацію, якщо це не є технічно або економічно недоцільним. Потрібно запровадити ієрархію відходів: зменшення кількості відходів, повторне використання, переробка, інші види відновлення та утилізації. Ефективне сортування відходів означає, що чим менше відходів потрапляє на сміттєзвалища, що робить їх дешевшими та кращими для людей та навколишнього середовища. Небезпечні відходи можуть спричинити проблеми зі здоров'ям, тому важливо, щоб вони утилізувалися правильно та безпечно, а не змішувалися зі звичайними відходами, що надходять з дому чи офісу.

```
import streamlit as st
import tensorflow as tf
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
```

Імпортують необхідні бібліотеки для створення графічного інтерфейсу класифікації зображень побутових відходів у Streamlit. Це бібліотека Python для швидкої побудови веб-інтерфейсів до моделей машинного та глибокого навчання. Фреймворк TensorFlow використовується для завантаження та використання збереженої моделі глибокого навчання формату .h5. NumPy - бібліотека для чисельних обчислень. PIL (Python Imaging Library) або Pillow - бібліотека для роботи із зображеннями. Дозволяє відкривати, масштабувати, змінювати формат зображень. Matplotlib - бібліотека для візуалізації, потрібна для побудови графіків або виводу зображень, якщо потрібно показати зображення разом із поясненнями. Ці бібліотеки

разом дозволяють створити інтерфейс завантаження зображень, підготувати зображення до моделі, класифікувати їх за допомогою нейромережі, виводити результати та графіку.

```
@st.cache_resource
def load_model():
    return tf.keras.models.load_model("modelnew.h5")
model = load_model()
```

Виконують завантаження моделі глибокого навчання у Streamlit-додатку. Функція load_model() відповідає за завантаження навченої моделі з файлу. Використовується метод TensorFlow load_model() для завантаження збереженої моделі у форматі .h5. modelnew.h5 - файл моделі, він містить структуру нейромережі та збережені ваги. Викликається функція і збережена модель записується у змінну model. Після цього її можна використовувати для класифікації зображень,

```
labels = {
    0: 'cardboard',
    1: 'glass',
    2: 'metal',
    3: 'paper',
    4: 'plastic',
    5: 'trash'
}
```

Це словник labels, у якому числові ключі (від 0 до 5) співставлені з назвами класів сміття чи відходів. Такий словник використовують, щоб на основі числового індексу, який повертає модель класифікації, отримати назву класу у зрозумілому вигляді.

```
translations = {
    "uk": {
        "title": "🗑 Інтелектуальна система класифікації побутових відходів",
        "upload_prompt": "📂 Завантаж зображення...",
        "classify_button": "🔍 Класифікувати",
        "result": "✅ Клас:",
        "probabilities": "🔍 Ймовірності по класах:",
        "bar_chart_title": "Розподіл ймовірностей по класах",
        "no_image": "Будь ласка, завантажте зображення перед класифікацією."
    },
    "en": {
```

```
        "title": "🗑 Intelligent Household Waste Classification System",
        "upload_prompt": "📁 Upload an image...",
        "classify_button": "🔍 Classify",
        "result": "✅ Predicted class:",
        "probabilities": "🔍 Class probabilities:",
        "bar_chart_title": "Class Probability Distribution",
        "no_image": "Please upload an image before classification."
    }
}
```

Це словник translations з двома вкладеними словниками для мов: української (uk) та англійської (en). Кожен вкладений словник містить текстові рядки для інтерфейсу системи класифікації побутових відходів:

- title - заголовок системи;
- upload_prompt - підказка для завантаження зображення;
- classify_button - текст кнопки класифікації;
- result - текст перед показом класифікованого класу;
- probabilities - текст перед ймовірностями по класах;
- bar_chart_title - заголовок для графіку розподілу ймовірностей;
- no_image - повідомлення, якщо користувач не завантажив зображення.

Цей словник зручно використовувати для багатомовного інтерфейсу, вибираючи потрібні рядки за мовою користувача.

```
lang = st.selectbox("🌐 Language / Мова", ["Українська", "English"])
lang_code = "uk" if lang == "Українська" else "en"
t = translations[lang_code]
```

Створюють випадаючий список для вибору мови інтерфейсу. st.selectbox відображає список з двома варіантами: Українська та English. Змінна lang приймає вибраний користувачем варіант. В lang_code записується відповідний код мови: uk для Українська і en для English. В t завантажується відповідний словник перекладів з раніше наданого словника translations.

```
st.title(t["title"])
```

Встановлюють заголовок додатку, використовуючи текст із словника перекладів t за ключем title. Залежно від вибраної користувачем мови (uk або en) заголовок буде українською або англійською мовами.

```
uploaded_file = st.file_uploader(t["upload_prompt"], type=["jpg", "jpeg", "png"])
```

Створюють віджет для завантаження файлів file_uploader з підписом, який залежить від вибраної мови. Обмеження за типом файлу - зображення формату JPG, JPEG або PNG. Користувач зможе вибрати файл з комп'ютера і цей файл потім можна обробляти, передавати у модель класифікації.

```
if st.button(t["classify_button"]):
    if uploaded_file is not None:
        image = Image.open(uploaded_file).convert('RGB')
        st.image(image, caption="🖼", use_column_width=True)
```

Виконують класифікацію після натискання кнопки. st.button(t["classify_button"]) створює кнопку з текстом Класифікувати залежно від мови або Classify. Якщо кнопку натиснуто і при цьому користувач завантажив файл uploaded_file is not None, то відкривається зображення через Image.open(uploaded_file) і конвертується в RGB. За допомогою st.image показується завантажене зображення. Це базова логіка для показу завантаженого зображення перед подальшою класифікацією.

```
image_resized = image.resize((300, 300))
img_array = np.array(image_resized) / 255.0
input_tensor = np.expand_dims(img_array, axis=0)
```

Готують завантажене зображення для передачі в модель глибокого навчання. image.resize(300, 300) змінює розмір зображення до 300×300 пікселів, щоб відповідати очікуваному розміру моделі. np.array(image_resized) / 255.0 конвертує зображення у NumPy-масив і нормалізує піксельні значення в діапазон [0, 1]. Після цього input_tensor готовий до подачі у модель для передбачення.

```
prediction = model.predict(input_tensor)[0]
predicted_class = labels[np.argmax(prediction)]
confidence = np.max(prediction)
```

Роблять передбачення класу зображення за допомогою моделі і обробляють результат. model.predict(input_tensor)[0] передає підготовлене зображення у модель, яка повертає масив ймовірностей по кожному класу. np.argmax(prediction) знаходить індекс класу з максимальною ймовірністю (найімовірніший клас). labels[np.argmax(prediction)] отримує назву класу за індексом із словника labels. confidence = np.max(prediction) зберігає максимальне значення ймовірності, ступінь

упевненості моделі у передбаченні. predicted_class - це назва найбільш ймовірного класу, confidence - ймовірність, з якою модель це передбачила (число від 0 до 1).

```
st.success(f"{t['result']} **{predicted_class.upper()}** ({confidence:.2%})")
```

Виводять повідомлення про успішну класифікацію з результатом, st.success() показує зелене повідомлення, яке сигналізує про успішне виконання дії. predicted_class.upper() - назва передбаченого класу великими літерами. ({confidence:.2%}) - упевненість моделі у відсотках.

Користувач побачить щось на зразок клас: GLASS (85,43 %)

```
st.subheader(t["probabilities"])
for i, prob in enumerate(prediction):
st.write(f"- {labels[i].capitalize()}: **{prob:.2%}**")
```

Виводять підзаголовок та список імовірностей по кожному класу.

st.subheader(t["probabilities"]) показує підзаголовок з текстом, що залежить від мови ймовірності по класах. Виводять для кожного класу рядок у форматі:

- клас із першою великою літерою;
- ймовірність у відсотках.

В результаті користувач побачить докладний розподіл ймовірностей по всіх класах.

```
st.subheader(t["bar_chart_title"])
fig, ax = plt.subplots()
ax.bar([labels[i] for i in range(len(prediction))], prediction, color='green')
ax.set_ylabel('Probability')
ax.set_ylim([0, 1])
plt.xticks(rotation=30)
st.pyplot(fig)
    else:
st.warning(t["no_image"])
```

Додають до інтерфейсу візуалізацію ймовірностей у вигляді стовпчикової діаграми. Таким чином реалізовано інтерфейс інтелектуальної системи класифікації побутових відходів з підтримкою двох мов, завантаженням зображення, передбаченням класу, виводом текстових результатів та графіком.

Для запуску веб-додатку інтелектуальної системи класифікації побутових відходів командному рядку перейти в папку, де знаходиться скрипт системи і

набрати команду Streamlit run app.py.Після цього появиться графічний інтерфейс інтелектуальної системи, який відкриється в броузері:

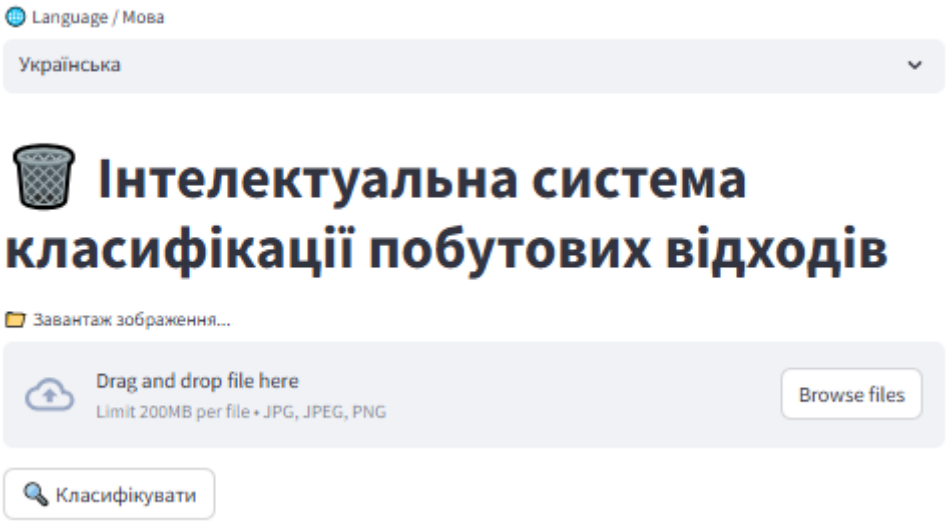

Рисунок 4.5 – Графічний інтерфейс інтелектуальної системи після запуску

Зверху у випадаючому списку можна вибрати мову інтерфейсу українську або англійську. Після цього потрібно натиснути кнопку Browse files, зайти в папкуData/Test, де знаходяться тестові зображення побутових відходів: cardboard, glass, metal, paper, plastic, trash і вибрати необхідне зображення, після чого потрібно натиснути кнопку Класифікувати.

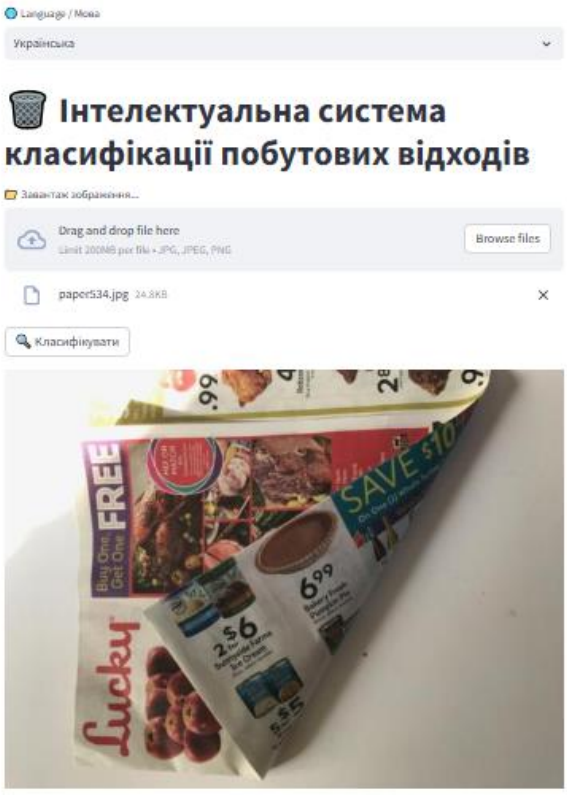

Рисунок 4.6 – Класифікація зображення паперу

Таким чином можна відкрити зображення 6 класів побутових відходів та класифікувати їх, для якого виду вони належать.

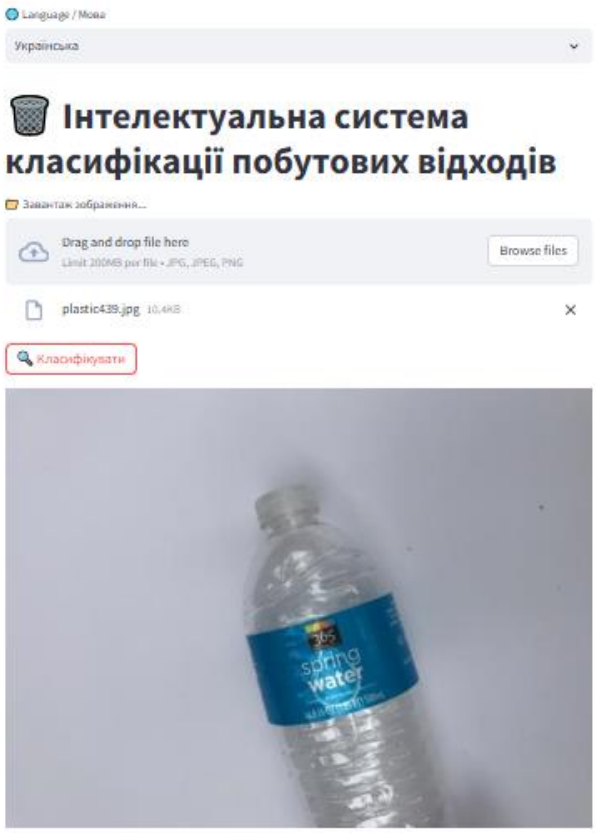

Рисунок 4.7 – Класифікація зображення пластику

ВИСНОВКИ ДО РОЗДІЛУ

Розділ присвячений детальному опису архітектури та реалізації програмного забезпечення системи класифікації побутових відходів. Використання бібліотек Keras та TensorFlow забезпечує ефективну розробку та тренування нейронної мережі. Основною частиною системи є послідовна модель Sequential, яка складається з кількох згорткових, пулінгових, повнозв'язних шарів та шарів Dropout для боротьби з перенавчанням. Для підготовки зображень застосовуються функції з модуля utils, що дозволяє швидко завантажувати і перетворювати зображення у потрібний формат. Важливою складовою є нормалізація пікселів для підвищення точності роботи моделі. Для інтерпретації результатів використовуються словники, що перетворюють числові передбачення у відповідні назви класів відходів. Візуалізація зображень допомагає у визначенні точності класифікації та аналізі роботи моделі. Вбудовані методи обробки даних, такі як аугментація та нормалізація, сприяють підвищенню стійкості системи до перекручувань та шумів. В результаті реалізована система здатна швидко та точно класифікувати побутові відходи, що важливо для автоматизації процесу сортування.

У роботі реалізовано інтерфейс користувача за допомогою бібліотеки Streamlit, що дозволяє зручно завантажувати зображення, запускати класифікацію та візуалізувати результати у реальному часі. Інтерфейс є інтуїтивно зрозумілим, забезпечуючи зручний доступ до функціоналу системи та швидкий перегляд класифікаційних результатів. Аналіз програмної реалізації підтверджує ефективність обраного підходу та переносимість отриманої моделі на практичні задачі управління відходами.

РОЗДІЛ 5. РОЗРОБЛЕННЯ СТАРТАП-ПРОЄКТУ

5.1. Структура проєкту інтелектуальної системи класифікації побутових відходів

Стартовий проект спрямований на створення базової версії інтелектуальної системи класифікації побутових відходів, яка здатна автоматично розпізнавати та класифікувати різні види сміття за допомогою глибоких нейронних мереж. Метою є запуск прототипу, який продемонструє обґрунтування технічних рішень, забезпечить високий рівень точності та стане основою для подальшого масштабування та удосконалення.

Таблиця 5.1 – Структура проєкту інтелектуальної системи класифікації побутових відходів

Назва	програма на мові Python
Назва проєкту	Інтелектуальна система класифікації побутових відходів методами глибокого навчання
Назва ВНЗ, факультету, спеціальності	НЛТУ, кафедра комп'ютерних наук, 122 «Комп'ютерні науки»
Прізвище, ім'я, по-батькові	Критчак Віталій Іванович
Цілі і задачі проєкту	Мета роботи – розробка та реалізація інтелектуальної системи для розпізнавання побутових відходів на зображеннях.
	Задачі проєкту:
	1. Дослідити та проаналізувати розроблені системи для класифікації побутових відходів, визначити їхні сильні та слабкі сторони.
	2. Створити математичну модель, що описує алгоритми класифікації побутових відходів.

Цілі і задачі проєкту	3. Розробити алгоритм функціонування інтелектуальної системи, що описує етапи обробки зображень і класифікації відходів.
	4. Розробити та реалізувати програмну модель системи, включаючи використання згорткових нейронних мереж для автоматичного розпізнавання та класифікації відходів.
	5. Провести тестування реалізованої системи для оцінки її продуктивності та точності класифікації.

5.2 Структура проекту інтелектуальної системи класифікації побутових відходів

- Проект складається з кількох ключових компонентів.
- дослідження і аналіз сучасних методів класифікації зображень;
- збір та підготовка датасету з зібраними зображеннями побутових відходів;
- розробка прототипу нейронної мережі для класифікації;
- робота з метриками якості та тестування системи;
- розгортання системи на локальній платформі з подальшою перевіркою у реальних умовах;
- робота з базою даних для збереження результатів та логів.

Вартість реалізації включає оплату праці розробників, закупівлю обладнання, необхідного для тестування і розгортання та інше.

Структура проекту інтелектуальної системи класифікації побутових відходів детально включає кілька ключових компонентів та етапів, що забезпечують ефективну розробку і реалізацію системи.

1. Ініціація проекту:
    - визначення цілей і задач проекту;
    - аналіз вимог замовників і користувачів;
    - вивчення існуючих рішень і технологій у галузі;
    - формування команди розробників і плану робіт.

2. Проектна документація:
    - розробка технічного завдання (ТЗ), що окреслює функціональні та нефункціональні вимоги системи;
    - створення архітектурної документації, включно з модульною структурою системи;
    - планування ресурсів (технічних, людських, фінансових);
    - визначення критеріїв успішності та методів тестування.

3. Структура програмного продукту.

- Модуль обробки зображень:
    - загрузка зображень користувачами або з камер;
    - попередня обробка зображень: зміна розміру, конвертація у формат RGB, нормалізація пікселів;
    - збереження оброблених зображень для історії та подальшого аналізу.
- Модуль класифікації:
    - використання згорткових нейронних мереж (CNN) для визначення типу відходів;
    - валідація та тестування моделі для забезпечення високої точності класифікації;
    - оптимізація моделі на основі метрик F1, Precision, Recall.
- Модуль управління даними:
    - застосування бази даних;
    - зберігання історії класифікацій, результатів, помилок;
    - метадані: дата, час, пристрій, користувач.
- Модуль користувацького інтерфейсу:
    - інтуїтивно зрозумілий дизайн для виконавчих користувачів;
    - відображення результатів класифікації та статистичних даних.
- Модуль моніторингу та тестування:
    - постійна перевірка роботи моделі на тестових наборах;
    - візуалізація метрик Accuracy, Precision;
    - звітування про невдалі класифікації та збої.

4. Архітектурна схема системи.

- багатокомпонентна архітектура;
- локальне або хмарне розгортання;
- комунікацію між модулями через визначені інтерфейси;
- можливість масштабування та оновлення системи без зупинки основних функцій.

5. Розробка та інтеграція.

- поетапне створення кожного модуля згідно з технічним завданням;
- інтеграція модулів у єдину систему;
- проведення модульного і системного тестування;
- впровадження механізмів автоматичного і ручного тестування для забезпечення стабільної роботи.

6. Навчання та запуск системи.

- навчання кінцевих користувачів роботі з системою;
- випробування проекту у реальних умовах;
- збір зворотного зв’язку для подальшого вдосконалення.

7. Подальше супроводження та масштабування.

- моніторинг роботи системи в експлуатації;
- збір статистичних даних для покращення моделей;
- модульне додавання нових функцій або платформ для поширення використання системи.

Ця структура проекту охоплює всі важливі етапи від початкового аналізу до підтримки та масштабування. Вона дозволяє систематично управляти процесом розробки, тестування та впровадження інтелектуальної системи класифікації побутових відходів, забезпечуючи її ефективність і надійність у реальних умовах експлуатації.

5.3 Етапи підготовки і реалізації проекту

Реалізація стартапу з розробки інтелектуальної системи сортування побутових відходів передбачає послідовне проходження ключових етапів, кожен з яких охоплює як технічні, так і організаційні аспекти.

1. Аналітичний та підготовчий етап (1 місяць): дослідження проблемної області, вивчення ринку та формування вимог до майбутньої системи.

- аналіз екологічної ситуації в регіоні, пов’язаної з неправильним сортуванням відходів;
- ідентифікація цільової аудиторії (комунальні підприємства, сміттєсортувальні заводи);
- визначення потреб користувачів і формулювання вимог;
- вибір типу відходів, які система буде класифікувати (пластик, скло, папір, органіка, метал);
- підбір відповідного відкритого набору даних для навчання моделі.

2. Проектування архітектури системи (1 місяць): розробка логічної та фізичної структури інформаційної системи.

- вибір архітектурного підходу;
- проектування базової архітектури згорткової нейронної мережі;
- підготовка прототипу інтерфейсу користувача (веб застосунок);
- вибір інструментів: Python, TensorFlow/Keras, Streamlit.

3. Збір і обробка даних (1 місяць): формування якісного навчального набору даних.

- завантаження зображень з набору даних;
- аугментація зображень для підвищення якості навчання;
- поділ даних на навчальну, тестову та валідаційну вибірки.

4. Розробка та навчання моделі (2 місяці): створення ефективної моделі глибокого навчання для класифікації відходів.

- побудова CNN-моделі;
- навчання моделі з використанням оптимізатора Adam та функції втрат;

- оцінка точності моделі за метриками precision, recall, F1-score;
- оптимізація моделі.

5. Інтеграція моделі в програмне забезпечення (1 місяць): реалізація повнофункціонального програмного рішення.

- створення користувацького інтерфейсу (вебдодаток на Streamlit);
- інтеграція моделі у вебсервіс: обробка фото → передача до CNN → відображення класу відходів;
- розробка модуля реєстрації користувачів та збереження результатів.

6. Тестування та налагодження системи (1 місяць): перевірка працездатності та стабільності продукту.

- проведення функціонального, модульного та інтеграційного тестування;
- усунення помилок і нестабільної поведінки;
- оптимізація продуктивності та зниження часу відповіді.

7. Пілотне впровадження (1 місяць): запуск інтелектуальної системи в реальних умовах.

- розгортання системи на сервері або хмарному середовищі Heroku;
- залучення перших тестових користувачів;
- збір зворотного зв’язку для майбутньої оптимізації.

8. Маркетинг і просування: залучення інвесторів і потенційних користувачів.

- створення вебсайту стартапу;
- публікація інформації у соцмережах, на екоплатформах, форумах;
- презентація проекту та пошук грантового фінансування.

9. Масштабування та супровід: розширення можливостей системи та її підтримка.

- додавання нових класів відходів.
- випуск оновлень з урахуванням зворотного зв’язку.
- підтримка користувачів та технічний супровід.

Загальна тривалість реалізації початкової версії інтелектуальної системи класифікації побутових відходів 6-7 місяців.

5.4 Орієнтовний бюджет проекту

Бюджет проекту складається з наступних статей:

- оплата праці команди (розробники, тестувальники) 60 % бюджету;
- обладнання (сервери, мобільні пристрої, обладнання для зйомки) 25 %;
- ресурси для обробки даних (хмарне сховище, ліцензії) 10 %;
- інші витрати (маркетинг, непередбачені витрати) 5 %.

Загальний бюджет проекту орієнтовно становить 100 000 грн.

5.5 Ризики і заходи щодо їх мінімізації

Основними ризиками є:

- недостатня якість зібраних даних – мінімізується шляхом проведення додаткових зйомок;
- недошкодовка моделі – контроль за процесом навчання, регулярна валідація та корекція;
- технічні проблеми з обладнанням – гарантійне обслуговування та резервні рішення.
- бюджетні обмеження – оптимізація витрат і пошук додаткового фінансування.

5.6 Очікувані результати

Очікувані результати реалізації стартап-проекту «Інтелектуальна система сортування побутових відходів на основі глибокого навчання» охоплюють як технічну, так і соціально-економічну площину.

1. Технічні результати.

- створення робчочого прототипу системи.

У межах реалізації проекту буде розроблено повнофункціональний прототип системи, який складається з програмного забезпечення на основі згорткової нейронної мережі, здатної автоматично класифікувати зображення побутових відходів за категоріями: пластик, папір, скло, метал, органіка тощо.

- інтеграція з апаратною частиною.

У перспективі можливе з’єднання програмного модуля з апаратною платформою (Arduino/Raspberry, камера, сортувальний механізм), що забезпечить автоматичне фізичне сортування в реальному часі.

- масштабованість системи.

Структура системи буде побудована з урахуванням подальшого розширення її можливостей, включаючи підтримку нових категорій відходів, багатомовного інтерфейсу, використання хмарного зберігання, тощо.

2. Соціальні результати.

- підвищення екологічної свідомості населення.

Запуск системи у пілотному режимі у школах або житлових комплексах сприятиме формуванню навичок відповідального поводження з відходами.

- зменшення кількості несортованого сміття.

Інтелектуальна система дозволить підвищити ефективність роздільного збору побутових відходів, що зменшить навантаження на полігони, підвищить коефіцієнт переробки та утилізації.

- сприяння розвитку економіки.

Сортування відходів на джерелі їхнього утворення дозволить ефективніше реалізовувати принципи переробки й повторного використання матеріалів.

3. Економічні результати.

- зменшення витрат на ручне сортування.

Автоматизація процесу сортування дозволить скоротити витрати муніципалітетів або підприємств на персонал та ресурси.

- потенціал монетизації.

У разі подальшого впровадження система може стати комерційним продуктом для муніципалітетів, екологічних компаній, сміттєпереробних підприємств.

- привабливість для інвесторів та грантодавців.

Екологічна спрямованість, потенціал впливу на сталий розвиток та зниження шкідливого впливу на довкілля робить стартап перспективним з точки зору залучення фінансування (гранти, фонди, інвестиції).

ВИСНОВКИ ДО РОЗДІЛУ

У розділі 5 розроблено структуру стартап-проєкту інтелектуальної системи класифікації побутових відходів, яка враховує основні етапи та ресурсні витрати. Вказано, що перед запуском проєкту слід провести підготовчі заходи, включаючи аналіз ринку та визначення потреб користувачів. Детально описано архітектуру системи, яка є багаторівневою і модульною, що забезпечує гнучкість та масштабованість її роботи. Визначено ключові компоненти системи, включаючи обробку зображень, класифікацію за допомогою глибоких нейронних мереж. Враховано можливість роботи системи в реальному часі та її застосування на різних платформах, що сприяє підвищенню її універсальності. Обгрунтовано необхідність тестування системи, використовувши метрики точності, прецизійності, повноти та F1-score для підтвердження її ефективності. Приведено структуру проєкту, яка включає бюджетні витрати, витрати на оплату праці та обладнання, що є важливим для планування та реалізації проєкту. Описано алгоритмічне забезпечення, яке охоплює всі етапи обробки даних від завантаження до класифікації та виведення результатів. Вказано, що система зберігатиме історію класифікацій та аналізуватиме її для покращення якості роботи та просування технологій у сфері управління відходами. Робота підтверджує перспективність та практичну реалізацію створення інтелектуальної системи для класифікації побутових відходів з високою точністю та адаптивністю.

ВИСНОВКИ

У дипломній роботі розроблено систему автоматичної класифікації побутових відходів на основі глибокого навчання. Використана архітектура згорткової нейронної мережі показала високий рівень точності при тестуванні на контрольних знімках. Процес попередньої обробки зображень та нормалізації їхнього розміру істотно підвищив ефективність навчання моделі. Застосування інструментів для передачі даних та аугментації зображень дозволило зменшити перенавчання та покращити загальну стабільність моделі.

Результати експериментів показали, що модель досягає рівня точності близько 80-95 %, що є прийнятним для задач багатокласової класифікації. Впровадження зручного інтерфейсу на базі Streamlit забезпечує швидкий і зручний доступ до системи для кінцевих користувачів. Система легко масштабується та може бути адаптована для класифікації інших типів зображень або додаткових категорій. Бібліотеки Keras та TensorFlow надійно підтримують процес створення й тренування нейронної мережі. Проведені візуалізації та аналіз результатів підтвердили коректність роботи моделі та її здатність до інтерпретації. Впровадження автоматизованої класифікації сприятиме більш ефективному сортуванню та переробці побутових відходів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Johansson N., Corvellec H. Waste policies gone soft: an analysis of European and Swedish waste prevention plans. Waste management. 2018, vol. 77, pp. 322-332.

2. Abdallah M., Abu TalibM., Feroz, S. Artificial Intelligence Applications in Solid Waste Management: a systematic research review. Waste management. 2020, 109, pp. 231-246.

3. Abdel-Shafy H.I., Mansour M.S. Solid waste issue: sources, composition, disposal, recycling, and valorization. Egyptian journal of petroleum. 2018, 27, pp. 1275-1290.

4. Gundupalli S., Hait S., Thakur A. A review on automated sorting of source-separated municipal solid waste for recycling. Waste management. 2017, 60, pp. 56-74.

5. Singh J., El-Sappagh S., Ali F., Goyal S. Smart waste management: a systematic review and scientometric analysis of artificial intelligence applications. Environment, development and sustainability. 2025, pp. 1-31.

6. Xia W., Jiang Y., Chen X. Zhao R. Application of machine learning algorithms in municipal solid waste management: a mini review. Waste management. 2022, 40, pp. 609-624.

7. Zhang Q., Yang Q., Zhang X. Waste image classification based on transfer learning and convolutional neural network. Waste management. 2021, 135, pp. 150-157.

8. Yang T., Xu J., Zhao Y., Gong T. Classification technology of domestic waste from 2000 to 2019: a bibliometrics-based review. Environmental science and pollution research. 2021, 28, pp. 26313-26324.

9. Abdu H., Mohd M. A survey on waste detection and classification using deep learning. Institute of electrical and electronics engineers. 2022, 10, pp. 128151-128165.

10. Su Y., Shang Y., Chen X., Lyu X. Garbage recognition and sorting system based on computer vision. Computer science and applications. 2023, pp. 1321-1332.

11. Yang M., Thung G. Classification of trash for recyclability status. Project report CS229. 2016, pp. 940-945.

12. Liang S., Gu Y. A deep convolutional neural network to simultaneously localize and recognize waste types in images. Waste management. 2021, 126, pp. 247-257.

13. Bobulski J., Piatkowski J. PET waste classification method and plastic waste databasewadaba. Advances in intelligent systems and computing. 2018, 681, pp. 57-64.

14. Hogan D., Chimeme Martin E., Taiwo Horsfall I. Enhanced convolutional neural network methodology for solid waste classification utilizing data augmentation techniques. Waste management. 2024, 2, pp. 184-193.

15. Waste Classification Data:

https://www.kaggle.com/datasets/techsash/waste-classification-data

ДОДАТКИ

ДОДАТОК А. ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

1.ipynb

```
import os
import matplotlib.pyplot as plt
from PIL import Image
import math
dir_example = "Data"
classes = os.listdir(dir_example)
print(classes)

dir_example = "Data/Train"
train_classes = os.listdir(dir_example)
print(train_classes)

['cardboard', 'glass', 'metal', 'paper', 'plastic', 'trash']

dir_with_examples = 'visualize'
files_per_row = 6
files_in_dir = os.listdir(dir_with_examples)
number_of_cols = files_per_row
number_of_rows = int(len(files_in_dir) / number_of_cols)
fig, axs = plt.subplots(number_of_rows, number_of_cols)
fig.set_size_inches(20, 15, forward=True)
try:
  for i in range(0, len(files_in_dir)):
    file_name = files_in_dir[i]
    image = Image.open(f'{dir_with_examples}/{file_name}')
    row = math.floor(i / files_per_row)
    col = i % files_per_row
    axs[col].imshow(image)
    axs[col].axis('off')
except:
  pass
plt.show()

from tensorflow.keras.models import Sequential
from keras.layers import Conv2D, Flatten, MaxPooling2D, Dense, Dropout,
SpatialDropout2D
```

```
from tensorflow.keras.losses import sparse_categorical_crossentropy,
binary_crossentropy
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator

train = 'Data/Train'
test = 'Data/Test'

train_generator = ImageDataGenerator(rescale = 1/255)
train_generator = train_generator.flow_from_directory(train,
                                                      target_size = (300,300),
                                                      batch_size = 32,
                                                      class_mode = 'sparse')
labels = (train_generator.class_indices)
print(labels,'\n')
labels = dict((v,k) for k,v in labels.items())
print(labels)
```

```
Found 2184 images belonging to 6 classes.
{'cardboard': 0, 'glass': 1, 'metal': 2, 'paper': 3, 'plastic': 4, 'trash': 5}
{0: 'cardboard', 1: 'glass', 2: 'metal', 3: 'paper', 4: 'plastic', 5: 'trash'}
```

```
test_generator = ImageDataGenerator(rescale = 1./255)
test_generator = test_generator.flow_from_directory(test,
                                                    target_size = (300,300),
                                                    batch_size = 32,
                                                    class_mode = 'sparse')
test_labels = (test_generator.class_indices)
print(test_labels,'\n')
test_labels = dict((v,k) for k,v in test_labels.items())
print(test_labels)
```

```
Found 343 images belonging to 6 classes.
{'cardboard': 0, 'glass': 1, 'metal': 2, 'paper': 3, 'plastic': 4, 'trash': 5}
{0: 'cardboard', 1: 'glass', 2: 'metal', 3: 'paper', 4: 'plastic', 5: 'trash'}
```

```
for image_batch, label_batch in train_generator:
  break
image_batch.shape, label_batch.shape
```

```
((32, 300, 300, 3), (32,))
```

```
for image_batch, label_batch in test_generator:
  break
image_batch.shape, label_batch.shape

((32, 300, 300, 3), (32,))

print(train_generator.class_indices)
Labels = '\n'.join(sorted(train_generator.class_indices.keys()))
with open('Labels.txt', 'w') as file:
  file.write(Labels)

{'cardboard': 0, 'glass': 1, 'metal': 2, 'paper': 3, 'plastic': 4, 'trash': 5}

model=Sequential()
model.add(Conv2D(32, kernel_size = (3,3),
padding='same',input_shape=(300,300,3),activation='relu'))
model.add(MaxPooling2D(pool_size=2))

model.add(Conv2D(64, kernel_size = (3,3), padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2))

model.add(Conv2D(32, kernel_size = (3,3), padding='same',activation='relu'))
model.add(MaxPooling2D(pool_size=2))

model.add(Flatten())
model.add(Dense(64,activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(32,activation='relu'))

model.add(Dropout(0.2))
model.add(Dense(6,activation='softmax'))

model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 300, 300, 32)      896

 max_pooling2d (MaxPooling2D  (None, 150, 150, 32)     0
 )
```

```
 conv2d_1 (Conv2D)           (None, 150, 150, 64)      18496

 max_pooling2d_1 (MaxPooling  (None, 75, 75, 64)       0
 2D)

 conv2d_2 (Conv2D)           (None, 75, 75, 32)        18464

 max_pooling2d_2 (MaxPooling  (None, 37, 37, 32)       0
 2D)

 flatten (Flatten)           (None, 43808)             0

 dense (Dense)               (None, 64)                2803776

 dropout (Dropout)           (None, 64)                0

 dense_1 (Dense)             (None, 32)                2080

 dropout_1 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 6)                 198

=================================================================
Total params: 2,843,910
Trainable params: 2,843,910
Non-trainable params: 0

model.compile(optimizer = 'Adam', loss = 'sparse_categorical_crossentropy',
metrics = ['accuracy'])

model.fit_generator(train_generator,
          epochs=10,
          steps_per_epoch=2184//32)

Epoch 1/10
68/68 [==============================] - 118s 2s/step - loss: 1.6845 - accuracy:
0.2783
Epoch 2/10
68/68 [==============================] - 112s 2s/step - loss: 1.4399 - accuracy:
0.4154
Epoch 3/10
68/68 [==============================] - 111s 2s/step - loss: 1.3438 - accuracy:
0.4498
```

```
Epoch 4/10
68/68 [==============================] - 111s 2s/step - loss: 1.2584 - accuracy:
0.5046
Epoch 5/10
68/68 [==============================] - 112s 2s/step - loss: 1.1042 - accuracy:
0.5599
Epoch 6/10
68/68 [==============================] - 111s 2s/step - loss: 0.9881 - accuracy:
0.6199
Epoch 7/10
68/68 [==============================] - 111s 2s/step - loss: 0.8935 - accuracy:
0.6733
Epoch 8/10
68/68 [==============================] - 111s 2s/step - loss: 0.7175 - accuracy:
0.7342
Epoch 9/10
68/68 [==============================] - 111s 2s/step - loss: 0.6144 - accuracy:
0.7876
Epoch 10/10
68/68 [==============================] - 114s 2s/step - loss: 0.5502 - accuracy:
0.7941
```

```
import keras.utils as ku
import numpy as np
test_img = 'Data\Test\metal\metal363.jpg'
img = ku.load_img(test_img, target_size = (300,300))
img = ku.img_to_array(img, dtype=np.uint8)
img = np.array(img)/255.0
prediction = model.predict(img[np.newaxis, ...])
print("Probability:",np.max(prediction[0], axis=-1))
predicted_class = labels[np.argmax(prediction[0], axis=-1)]
print("Classified:",predicted_class,'\n')
plt.axis('off')
plt.imshow(img.squeeze())
plt.title("Loaded Image")
```

```
1/1 [==============================] - 0s 174ms/step
Probability: 0.69203347
Classified: cardboard
```

```
classes = []
probability = []
```

```
for i,j in enumerate(prediction[0],0):
  print(labels[i].upper(),':',round(j*100,2),'%')

CARDBOARD : 69.2 %
GLASS : 6.72 %
METAL : 8.1 %
PAPER : 1.44 %
PLASTIC : 13.71 %
TRASH : 0.83 %

test_img = 'Data/Test/glass/glass421.jpg'
img = ku.load_img(test_img, target_size = (300,300))
img = ku.img_to_array(img, dtype=np.uint8)
img = np.array(img)/255.0
prediction = model.predict(img[np.newaxis, ...])
print("Probability:",np.max(prediction[0], axis=-1))
predicted_class = labels[np.argmax(prediction[0], axis=-1)]
print("Classified:",predicted_class,'\n')
plt.axis('off')
plt.imshow(img.squeeze())
plt.title("Loaded Image")

1/1 [==============================] - 0s 43ms/step
Probability: 0.6233342
Classified: glass

classes = []
probability = []
for i,j in enumerate(prediction[0],0):
  print(labels[i].upper(),':',round(j*100,2),'%')

CARDBOARD : 2.61 %
GLASS : 62.33 %
METAL : 17.81 %
PAPER : 0.73 %
PLASTIC : 15.99 %
TRASH : 0.53 %

test_img = 'Data/Test/plastic/plastic425.jpg'
img = ku.load_img(test_img, target_size = (300,300))
img = ku.img_to_array(img, dtype=np.uint8)
img = np.array(img)/255.0
prediction = model.predict(img[np.newaxis, ...])
```

```
print("Probability:",np.max(prediction[0], axis=-1))
predicted_class = labels[np.argmax(prediction[0], axis=-1)]
print("Classified:",predicted_class,'\n')
plt.axis('off')
plt.imshow(img.squeeze())
plt.title("Loaded Image")

1/1 [==============================] - 0s 50ms/step
Probability: 0.9019071
Classified: plastic

classes = []
probability = []
for i,j in enumerate(prediction[0],0):
  print(labels[i].upper(),':',round(j*100,2),'%')

CARDBOARD : 0.73 %
GLASS : 3.26 %
METAL : 4.2 %
PAPER : 1.41 %
PLASTIC : 90.19 %
TRASH : 0.21 %

model.save('weights/model.h5')
```

ДОДАТОК Б. Графічний інтерфейс інтелектуальної системи

app.py

```
import streamlit as st
import tensorflow as tf
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt

@st.cache_resource
def load_model():
    return tf.keras.models.load_model("modelnew.h5")

model = load_model()

labels = {
    0: 'cardboard',
    1: 'glass',
    2: 'metal',
    3: 'paper',
    4: 'plastic',
    5: 'trash'
}

translations = {
    "uk": {
        "title": "🗑 Інтелектуальна система класифікації побутових відходів",
        "upload_prompt": "📂 Завантаж зображення...",
        "classify_button": "🔍 Класифікувати",
        "result": "✅ Клас:",
        "probabilities": "🔍 Ймовірності по класах:",
        "bar_chart_title": "Розподіл ймовірностей по класах",
        "no_image": "Будь ласка, завантажте зображення перед класифікацією."
    },
    "en": {
        "title": "🗑 Intelligent Household Waste Classification System",
        "upload_prompt": "📂 Upload an image...",
        "classify_button": "🔍 Classify",
        "result": "✅ Predicted class:",
        "probabilities": "🔍 Class probabilities:",
```

```
            "bar_chart_title": "Class Probability Distribution",
            "no_image": "Please upload an image before classification."
        }
    }

    lang = st.selectbox("🌐 Language / Мова", ["Українська", "English"])
    lang_code = "uk" if lang == "Українська" else "en"
    t = translations[lang_code]

    st.title(t["title"])

    uploaded_file = st.file_uploader(t["upload_prompt"], type=["jpg", "jpeg",
"png"])

    if st.button(t["classify_button"]):
        if uploaded_file is not None:
            image = Image.open(uploaded_file).convert('RGB')
            st.image(image, caption="🖼", use_column_width=True)

            image_resized = image.resize((300, 300))
            img_array = np.array(image_resized) / 255.0
            input_tensor = np.expand_dims(img_array, axis=0)

            prediction = model.predict(input_tensor)[0]
            predicted_class = labels[np.argmax(prediction)]
            confidence = np.max(prediction)

            st.success(f"{t['result']} **{predicted_class.upper()}**
({confidence:.2%})")

            st.subheader(t["probabilities"])
            for i, prob in enumerate(prediction):
                st.write(f"- {labels[i].capitalize()}: **{prob:.2%}**")

            st.subheader(t["bar_chart_title"])
            fig, ax = plt.subplots()
            ax.bar([labels[i] for i in range(len(prediction))], prediction,
color='green')
            ax.set_ylabel('Probability')
            ax.set_ylim([0, 1])
            plt.xticks(rotation=30)
            st.pyplot(fig)
```

```
else:
    st.warning(t["no_image"])
```