КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет інформаційних технологій
Кафедра інтелектуальних технологій

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

НА ТЕМУ

«Програмний модуль інтелектуального розпізнавання військової техніки за супутниковими знімками»

Галузь знань **12 «Інформаційні технології»**

Спеціальність **122 «Комп'ютерні науки»**

Освітня програма «**Комп'ютерні науки»**

Освітній рівень: **бакалавр**

Виконав:
студент 4 курсу групи КН-42
Удоденко О.К.

Керівник:
асистент кафедри
к.т.н., с.д.
Андрійчук Олег Валентинович

Випускна кваліфікаційна робота бакалавра допущена до захисту
рішенням кафедри *інтелектуальних технологій*
Протокол № 11 від 06.06.2023 р.
Зав. кафедри _____ доц. Іларіонов О.Є.

Київ – 2023

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра інтелектуальних технологій

Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інтелектуальних технологій
Іларіонов О.Є.

_____

"___"_____ 2023 р.

ЗАВДАННЯ
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Удоденку Олександру Костянтиновичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи): «Програмний модуль інтелектуального розпізнавання військової техніки за супутниковими знімками»

   затверджена протоколом засідання кафедри від «11» листопада 2023 р. № 4

2. Термін здачі студентом закінченого проекту (роботи): 15 червня 2023 року
3. Вихідні дані до проекту (роботи): Розробити програмний модуль розпізнавання військової техніки за супутниковими знімками.
4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити): 1) аналіз принципів застосування нейронних мереж для розпізнавання військової техніки на зображеннях;
   2) реалізація програмного модуля інтелектуального розпізнавання військової техніки на зображеннях;
   3) програмне впровадження модуля інтелектуального розпізнавання військової
5. техніки за супутниковими знімками. Перелік презентаційного матеріалу (з точним зазначенням обов'язкових презентацій: актуальність (1 слайд), мета, об'єкт та предмет (1 слайд), дерево функцій програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками (1 слайд), (1 слайд),
   завдання на кваліфікаційну роботу (1 слайд), Основні бізнес-процеси в модулі колаборативної фільтрації (1 слайд), дерево функцій (1 слайд), діаграма роботи у форматі IDEF0 (1 слайд), діаграми рівня A1 (2 слайди), діаграма використання (1 слайд), архітектура системи рекомендацій (1 слайд), Таблиці БД (2 слайди) обробка даних (2 слайди), робочі вікна системи (2 слайди),
   висновки (1 слайди).

6. Консультанти з випускної кваліфікаційної роботи із зазначенням її розділів, що їх стосуються

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Андрійчук О.В.		
2	Андрійчук О.В.		
3	Андрійчук О.В.		

7. Дата видачі завдання 15 лютого 2023 року

Керівник випускної кваліфікаційної роботи_____ / О.В. Андрійчук /
(підпис) (ініціали та прізвище)

Завдання прийняв до виконання _____ / О.К. Удоденко /
(підпис) (ініціали та прізвище)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів	Примітка
1	Обговорення постановки завдання та змісту пояснювальної записки	15.02.2023 - 16.02.2023	
2	Вибір та формування напряму дослідження	17.02.2023 - 19.02.2023	
3	Аналіз предметної області	20.02.2023 – 11.03.2023	
4	Вибір методів рішення задачі	12.03.2023 – 21.04.2023	
5	Створення програмного модулю	22.04.2023 – 11.05. 2023	
6	Оформлення пояснювальної записки	12.05.2023 – 29.05.2023	

Студент-дипломник _____ / Удоденко О.К. /
(підпис) (ПІБ)

Керівник випускної кваліфікаційної роботи _____ / Андрійчук О. В. /
(підпис) (ПІБ)

Анотація

Удоденко Олександр Костянтинович виконав випускну кваліфікаційну роботу на тему “Програмний модуль інтелектуального розпізнавання військової техніки за супутниковими знімками” за спеціальністю 122 – «Комп’ютерні науки».

У випускній кваліфікаційній роботі проведено аналіз сучасних методів інтелектуального розпізнавання військової техніки за супутниковими знімками та розроблено програмний модуль для виконання основних функцій розпізнавання військової техніки за супутниковими знімками.

У даній роботі було проведено аналіз сучасних методів інтелектуального розпізнавання військової техніки на основі супутникових знімків. Було виявлено, що використання нейронних мереж та методів машинного навчання дозволяє досягти високої точності розпізнавання.

Розроблений програмний модуль має різні режими роботи, які дозволяють користувачеві виконувати різноманітні завдання розпізнавання військової техніки. Модуль здатний виявляти та класифікувати різні типи військової техніки на основі вхідних супутникових знімків. Крім того, модуль може виконувати аналіз інших параметрів, таких як розміри об'єктів, деталізація знімків та інші характеристики, що сприяють більш детальному аналізу технічних аспектів військової техніки.

В результаті розробки та використання програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками досягнуто значний прогрес у розпізнаванні та аналізі військової техніки. Модуль виявляється потенційно корисним для військових, розвідувальних та безпекових служб, які мають потребу у швидкому та точному розпізнаванні військової техніки на основі супутникових знімків.

Даний модуль відкриває широкі перспективи для подальшого вдосконалення та використання у військових і розвідувальних діях.

Використання інтелектуального розпізнавання військової техніки на основі супутникових знімків допоможе збільшити ефективність військових операцій, покращити збір та аналіз інформації про військову техніку, а також забезпечити швидку та точну ідентифікацію та класифікацію об'єктів на зображеннях.

Ключові слова: розпізнавання зображень, військова техніка, нейронні мережі, машинне навчання.

Abstract

Udodenko Oleksandr Kostiantynovych has carried out a graduation thesis on the topic "Software module for intelligent recognition of military equipment using satellite images" in the specialty 122 - "Computer Science".

The graduation thesis analyzes modern methods of intelligent recognition of military equipment using satellite images and develops a software module to perform basic functions of recognition of military equipment using satellite images.

This work involves an analysis of modern methods of intelligent recognition of military equipment based on satellite imagery. It has been discovered that the use of neural networks and machine learning methods enables achieving high accuracy in recognition.

The developed software module has various operating modes that allow users to perform diverse tasks related to the recognition of military equipment. The module is capable of detecting and classifying different types of military equipment based on input satellite images. Additionally, it can analyze other parameters such as object sizes, image resolution, and other characteristics, facilitating a more detailed analysis of technical aspects of military equipment.

This module opens up broad prospects for further improvement and utilization in military and intelligence operations. The use of intelligent recognition of military equipment based on satellite imagery will contribute to enhancing the efficiency of military operations, improving the collection and analysis of information about military equipment, and enabling fast and accurate identification and classification of objects in images.

Keywords: image recognition, military equipment, neural network, machine learning

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ANN – artificial neural networks (штучна нейронна мережа)

CNN – Convolutional Neural Network (конволюційна нейронна мережа)

MCC – Matthews Correlation Coefficient (коефіцієнт кореляції Метьюза)

F2 – F-beta score зі значенням beta = 2

AUC – Area Under the Curve (площа під кривою)

PR – Precision-Recall (точність-повернення)

DS – dataset (набір даних)

ReLU – Rectified Linear Unit

MaxPooling2D – шар пулінгу (зменшення розміру зображення)

ГНМ – глибинна нейронна мережа

ШНМ – штучна нейронна мережа

ВСТУП

В сучасному світі, де військова техніка та обладнання є однією з найважливіших складових безпеки країни, розробка програмних модулів для їх інтелектуального розпізнавання на супутникових знімках є дуже актуальною та важливою задачею. Це пов'язано з необхідністю оперативного та ефективного контролю за розташуванням військової техніки противника, а також забезпечення власної безпеки та обороноздатності.

Актуальність даного дослідження обумовлена рядом факторів. По-перше, розвиток технологій супутникового зондування дозволяє збирати все більше інформації про території, на яких розміщені військові об'єкти. Однак, її обробка та аналіз з часом стає все більш складною задачею. По-друге, розробка програмного модулю інтелектуального розпізнавання військової техніки за супутниковими знімками може бути важливою для захисту національної безпеки та оборони країни. Відповідно, розвиток ефективних методів та технологій для розпізнавання військової техніки на супутникових знімках є важливою науково-технічною задачею.

Однією з основних задач даної роботи є розробка програмного модулю розпізнавання військової техніки на супутникових знімках. За допомогою цього модулю буде можливо швидко та ефективно виявляти, ідентифікувати та відстежувати рух військової техніки противника на території країни.

Для досягнення поставленої мети, в роботі будуть використані сучасні методи та технології машинного навчання та обробки супутникових знімків. Крім того, в роботі буде детально описано процес розробки програмного модулю, включаючи вибір необхідних інструментів та технологій, структуру програмного модулю та його алгоритми розпізнавання.

Результати роботи можуть бути використані як підґрунтя для подальшого розвитку та вдосконалення програмного модулю, що сприятиме підвищенню ефективності військової розвідки та забезпеченню безпеки країни.

Один із головних етапів розробки такого модулю – це обробка та аналіз супутникових знімків, що вимагає використання методів комп'ютерного зору. Комп'ютерний зір (або комп'ютерний зірок) – це сукупність алгоритмів, що дозволяють комп'ютеру "бачити" та "розуміти" зображення, що були отримані за допомогою різних пристроїв, таких як камери або супутники. У даній роботі будуть використані методи комп'ютерного зору для попередньої обробки та підготовки зображень, а також для визначення розміру, форми та інших характеристик військової техніки на знімках.

Для виявлення та ідентифікації військової техніки на супутникових знімках будуть використані методи машинного навчання. Машинне навчання – це галузь штучного інтелекту, що дозволяє комп'ютеру "навчатися" розпізнавати образи та виконувати складні завдання без явного програмування. У даній роботі для машинного навчання будуть використані алгоритми нейронних мереж, що дозволяють автоматично визначати параметри для розпізнавання об'єктів на зображеннях.

Для обробки та аналізу супутникових знімків також будуть використані сучасні методи обробки зображень. Це можуть бути методи фільтрації, сегментації, аналізу зображень та розпізнавання образів. Для досягнення максимальної ефективності і точності розпізнавання, можна використовувати нейронні мережі та глибинне навчання (deep learning).

З огляду на вищенаведені дані можна визначити об’єкт та предмет дослідження:

– об'єкт дослідження – процес інтелектуального розпізнавання різних типів військової техніки на супутникових знімках;

– предмет дослідження –методи та технології машинного навчання, розпізнавання та обробки супутникових знімків.

Метою даної роботи є створення ефективного програмного модулю інтелектуального розпізнавання військової техніки на супутникових знімках, що може знайти застосування у військових цілях, зокрема у контролі за військово-стратегічним об'єктами.

РОЗДІЛ 1 АНАЛІТИЧНИЙ ОГЛЯД, ПОСТАНОВКА ЗАВДАННЯ

1.1 Аналіз проблематики обробки і розпізнавання супутникових знімків для класифікації військової техніки

1.1.1 Задачі розпізнавання об’єктів на супутникових знімках

Проблематика задач, що пов'язані з розпізнаванням військової техніки за супутниковими знімками, включає ряд складних питань, вирішення яких має велике значення для досягнення точності та ефективності розпізнавання:

1. Розпізнавання різних типів військової техніки: військова техніка може включати різноманітні типи, такі як танки, бронетранспортери, військові літаки тощо. Кожен тип має унікальні особливості та візуальні характеристики, що ускладнює їх точне розпізнавання. Проблема полягає в розробці моделі [1], яка здатна ефективно розпізнавати і класифікувати різні типи військової техніки на основі супутникових знімків.

2. Варіації у розташуванні та орієнтації техніки: техніка може бути розташована у різних позиціях та орієнтаціях на супутникових знімках [2]. Це створює виклик у визначенні точних меж об'єктів та відповідній класифікації. Потрібно вирішити питання з орієнтацією об'єктів, а також розробити методи для знаходження та виділення військової техніки незалежно від її розташування на знімку.

3. Змінні умови зйомки: умови зйомки супутникових знімків можуть різнитися, включаючи різний освітлення, тіні, атмосферні умови тощо [1]. Це може призводити до зміни візуальних характеристик об'єктів та ускладнювати їх розпізнавання. Розробка методів, що стійкі до змінних умов зйомки, є важливим аспектом в цій задачі.

4. Недостатня кількість навчальних даних: ефективне навчання нейронної мережі для розпізнавання вимагає великої кількості репрезентативних даних [3].

Однак отримання великого набору даних про військову техніку може бути складним завданням. Недостатня кількість навчальних даних може призвести до недосягнення високої точності розпізнавання та зменшення загальної ефективності моделі.

5. Ціна обробки та аналізу великих обсягів даних: супутникові знімки можуть бути великого розміру та великими обсягами даних, що вимагає значних обчислювальних ресурсів для їх обробки та аналізу [1]. Необхідність у потужних обчислювальних системах та ефективних алгоритмах обробки становить додаткову проблему у реалізації даної задачі.

Для вирішення цих проблем можна розглянути такі варіанти, як покращення архітектури нейронної мережі, вдосконалення методів обробки зображень та використання більш потужних обчислювальних ресурсів. Також можна розглянути співробітництво зі спеціалістами з військової техніки для отримання більш точних та репрезентативних даних. Це допоможе забезпечити більш точне та ефективне розпізнавання військової техніки на основі супутникових знімків.

Розпізнавання різних типів військової техніки є однією з ключових проблем у задачі інтелектуального розпізнавання військової техніки на основі супутникових знімків. Військова техніка охоплює широкий спектр об'єктів, включаючи танки, бронетранспортери, військові літаки, артилерійські установки та багато інших.

Кожен тип військової техніки має свої унікальні особливості та візуальні характеристики, що ускладнює процес їх розпізнавання. Різні типи техніки можуть мати відмінні форми, розміри, структури та деталі, що створює виклик для точного класифікації на основі зображень.

Проблема полягає в розробці моделі, яка буде здатна ефективно розпізнавати і класифікувати різні типи військової техніки на основі супутникових знімків. Для досягнення цієї мети необхідно розробити алгоритми та нейронні мережі, які зможуть виявляти та аналізувати візуальні ознаки, що характеризують кожен тип військової техніки. Це може включати в себе

виявлення специфічних форм, структурних особливостей, розмірів та інших деталей, що дозволять встановити точну класифікацію об'єктів.

Розробка моделі розпізнавання, яка здатна розрізняти різні типи військової техніки на основі супутникових знімків, вимагає великої кількості навчальних даних, що представляють різні класи техніки. Також варто звернути увагу на різноманітність умов зйомки, освітлення та інших факторів, які можуть впливати на якість зображень і ускладнювати завдання розпізнавання.

Крім того, проблема точного розпізнавання різних типів військової техніки може бути пов'язана з відсутністю репрезентативних даних для деяких класів техніки або наявністю неправильно класифікованих об'єктів у навчальному наборі даних. В таких випадках може бути необхідним використання додаткових методів підвищення якості даних.

У подальшому дослідженні проблематики розпізнавання різних типів військової техніки можуть бути розглянуті такі аспекти, як оптимізація алгоритмів розпізнавання, використання додаткових джерел даних (наприклад, додаткових супутникових знімків), підвищення точності класифікації за рахунок комбінації різних моделей та методів, а також розгляд спеціальних випадків, таких як розпізнавання військової техніки в умовах обмеженої видимості або розпізнавання камуфльованих об'єктів.

Варіації у розташуванні та орієнтації техніки на супутникових знімках представляють додаткові виклики для завдання розпізнавання військової техніки. На знімках техніка може мати різні позиції, такі як фронтальний або боковий вигляд, а також різні орієнтації, такі як обернена, нахилена або повернута.

Це створює виклик у визначенні точних меж об'єктів та правильній класифікації. Залежно від орієнтації техніки, її візуальні ознаки можуть змінюватись, що ускладнює процес розпізнавання. Наприклад, орієнтація танка може впливати на видимість його башти, гусениць або інших деталей, що може вплинути на правильність класифікації.

Для вирішення цих питань необхідно розробити методи, які зможуть ефективно виявляти та виокремлювати військову техніку незалежно від її розташування та орієнтації на знімку. Це може включати в себе використання алгоритмів детекції об'єктів, які здатні виявляти техніку незалежно від її орієнтації, а також алгоритми сегментації, які допоможуть виокремити техніку від інших об'єктів та фону.

Окрім того, для ефективного розпізнавання варіацій у розташуванні та орієнтації техніки можуть бути застосовані методи обробки зображень, які враховують геометричні перетворення, включаючи розтягування, обертання та перекладання. Такі методи можуть допомогти вирішити питання з орієнтацією об'єктів та покращити точність розпізнавання незалежно від їхнього розташування на знімку.

Змінні умови зйомки супутникових знімків представляють виклик для завдання розпізнавання військової техніки, оскільки можуть виникати різні умови освітлення, тіні, атмосферні умови та інші фактори, які можуть впливати на візуальні характеристики об'єктів.

Зміна освітлення може призводити до зміни яскравості, контрастності та кольорових відтінків на знімках. Тіні, що утворюються внаслідок освітлення, можуть приховувати деякі деталі техніки або створювати спотворення в зображенні. Крім того, атмосферні умови, такі як хмарність або туман, можуть знижувати якість зображень та ускладнювати їх аналіз.

Для розпізнавання військової техніки в умовах змінних параметрів зйомки необхідно розробити методи, що стійкі до цих змін. Це може включати застосування алгоритмів адаптивної обробки зображень, які можуть компенсувати зміни в яскравості, контрастності та кольорі на знімках. Також можуть використовуватися методи фільтрації, які допомагають зменшити вплив шуму та спотворень, що виникають внаслідок атмосферних умов.

Окрім цього, існує необхідність у використанні навчальних даних, які враховують різні умови зйомки. Це може включати збір даних з різних джерел та у різні періоди часу, щоб отримати репрезентативність різних умов зйомки.

Такі дані допоможуть моделі навчитися розпізнавати військову техніку в різних ситуаціях та покращити її стійкість до змінних умов зйомки.

Ціна обробки та аналізу великих обсягів даних є однією з важливих проблем у задачі розпізнавання військової техніки за допомогою супутникових знімків. Супутникові знімки можуть бути великого розміру та містити значну кількість даних, що вимагає великої обчислювальної потужності для їх обробки та аналізу.

Обробка великих обсягів даних вимагає потужних обчислювальних систем, таких як сервери з високою продуктивністю та значною кількістю ресурсів. Такі системи можуть забезпечити швидку обробку зображень і виконання складних алгоритмів розпізнавання. Однак, використання таких систем може бути витратним і вимагати значних фінансових витрат.

Ефективність алгоритмів обробки даних також важлива. Необхідно використовувати оптимізовані алгоритми, які можуть ефективно працювати з великими обсягами даних і забезпечувати швидку обробку знімків. Це може включати використання паралельних обчислень, розподілених систем обробки даних та інші техніки, що дозволяють ефективно використовувати обчислювальні ресурси.

Також важливо враховувати економічний аспект даної проблеми. Використання потужних обчислювальних систем та оптимізованих алгоритмів може вимагати значних витрат, які можуть стати обмеженням для впровадження розпізнавання військової техніки на практиці. Тому розробка ефективних і вартісно-ефективних рішень є важливим аспектом для успішної реалізації даної задачі.

1.1.2 Принципи обробки і розпізнавання супутникових знімків

Принципи обробки і розпізнавання супутникових знімків включають наступні етапи:

1. Підготовка даних – цей етап включає в себе обробку знімків з метою покращення якості зображення та зменшення шумів. Для цього можуть використовуватися різні методи фільтрації, такі як медіанний фільтр [1], фільтр Гауса [2] та інші.

2. Сегментація – цей етап включає в себе виділення об'єктів на зображенні та їх розділення на окремі частини. Для цього можуть використовуватися методи порогової обробки, алгоритми визначення контуру та інші.

3. Витягнення ознак – цей етап включає в себе вибір та виділення важливих ознак з об'єктів на зображенні. Це можуть бути такі ознаки, як розмір, форма, текстура, кольорова гама та інші.

4. Класифікація – цей етап включає в себе призначення класу об'єктів на зображенні відповідно до їх характеристик. Для цього можуть використовуватися нейронні мережі, алгоритми машинного навчання та інші методи.

5. Постобробка – цей етап включає в себе оцінку та покращення результатів розпізнавання об'єктів на зображенні. Для цього можуть використовуватися методи пост-класифікаційної обробки, такі як згладжування, розширення та інші.

Використання цих принципів у програмному модулі інтелектуального розпізнавання військової техніки на супутникових знімках дозволяє досягти високої точності та ефективності розпізнавання.

1.1.3 Аналіз основних задач на етапі підготовки даних

На етапі підготовки даних перед подальшим розпізнаванням військової техніки на супутникових знімках, необхідно провести обробку зображення з метою покращення якості та зменшення шумів. Для цього можна використовувати різноманітні методи фільтрації. Наприклад, медіанний фільтр дозволяє видалити шуми, що знаходяться в окрузі кожного пікселя, і відображається значенням медіани цих пікселів. Фільтр Гауса зменшує шуми [2],

застосовуючи гаусівську функцію для згладжування значень пікселів зображення.

Після фільтрації зображення необхідно підготувати для подальшої сегментації, яка полягає в розділенні зображення на окремі сегменти, що містять об'єкти військової техніки. Для цього можуть використовуватися різні методи, включаючи методи кластеризації, регіонів зростання та порогової обробки. Після сегментації об'єкти можуть бути відокремлені від фону та підготовлені для подальшої обробки та класифікації.

Також на етапі підготовки даних може бути виконано геометричну корекцію зображення, щоб скорегувати спотворення геометрії, спричинені різними чинниками, такими як перспектива та кут огляду. Це може бути зроблено за допомогою методів, таких як афінна трансформація та проективна трансформація [1].

Медіанний фільтр та фільтр Гауса є популярними методами фільтрації, які застосовуються для покращення якості зображення на супутникових знімках [2]. Медіанний фільтр видаляє шуми, які знаходяться в окрузі кожного пікселя, шляхом заміни значення кожного пікселя медіаною значень усіх пікселів в цій окрузі. Цей метод є ефективним для видалення шумів зображення, таких як "сіль" і "перець", і має високу стійкість до випадкових відхилень. Фільтр Гауса [2] використовує гаусівську функцію для згладжування значень пікселів зображення. Цей метод зменшує шуми та робить зображення більш рівномірним.

Обидва методи можуть бути використані як окремо, так і разом з іншими методами фільтрації для покращення якості зображення на супутникових знімках та підготовки даних для подальшого аналізу.

1.1.4 Аналіз основних задач на етапі сегментації зображень

Сегментація зображень – це процес виділення окремих об'єктів на зображенні та їх розділення на окремі частини. Це дуже важливий етап в обробці супутникових знімків для розпізнавання військової техніки.

Одним зі способів сегментації є бінаризація, що полягає в розділенні зображення на дві групи: області, які містять об'єкти військової техніки (які відображаються білим кольором), та області, які не містять об'єкти військової техніки (які відображаються чорним кольором). Для цього можуть використовуватися різні методи бінаризації, такі як глобальна бінаризація або адаптивна бінаризація [3].

Іншими методами сегментації можуть бути використані алгоритми кластеризації, які розділять зображення на кластери на основі схожих властивостей, таких як яскравість, текстура або форма об'єктів [2]. Зазвичай ці алгоритми використовуються в поєднанні зі знаннями про форму об'єктів військової техніки, що дозволяє знизити кількість помилок в результатах сегментації.

Одним з методів сегментації є порогова обробка. Вона полягає в тому, що на зображенні задається певний поріг, і всі пікселі зі значенням яскравості вище цього порогу вважаються об'єктами, а всі решта – фоном. Таким чином, застосування порогової обробки дозволяє виділити об'єкти на зображенні залежно від їх яскравості [2].

Інший метод сегментації – це алгоритми визначення контуру. Вони дозволяють виділити об'єкти на зображенні, виокремивши їх контури, тобто роблять каркас, який оточує кожен об'єкт [3]. Ці алгоритми можуть бути основані на математичних моделях, зокрема на основі диференціальних рівнянь або алгоритмів пошуку локальних максимумів та мінімумів на зображенні.

Також для сегментації можуть використовуватися інші методи, такі як градієнтні алгоритми, кластерний аналіз, штучні нейронні мережі та інші [1, 2, 3].

Після проведення сегментації на зображенні отримують окремі сегменти, що містять окремі об'єкти військової техніки. Ці сегменти можуть бути використані для подальшого аналізу та розпізнавання за допомогою методів машинного навчання та нейронних мереж.

1.1.5 Аналіз основних задач на етапі витягнення ознак

Етап витягнення ознак є одним з ключових етапів обробки супутникових знімків для подальшого розпізнавання військової техніки за допомогою нейронних мереж. Його основна мета полягає у виборі та виділенні найбільш важливих ознак з об'єктів на зображенні, які допоможуть класифікувати ці об'єкти на етапі розпізнавання.

Для витягнення ознак можуть використовуватися різноманітні методи та алгоритми. Один з таких методів – це метод кутового коефіцієнту, який використовується для визначення форми об'єктів на зображенні [5]. Він включає в себе обчислення кутових коефіцієнтів для кожного пікселя в області і дозволяє визначити форму об'єкта шляхом порівняння цих коефіцієнтів.

Інший метод – це метод локальних бінарних зразків (Local Binary Patterns, LBP), який використовується для опису текстурних ознак на зображенні [6]. Він полягає в порівнянні значень пікселів у певній області зі значенням центрального пікселя та визначенні бінарного коду на основі цього порівняння.

Також можуть використовуватися методи геометричної обробки, які дозволяють визначити розмір, форму та орієнтацію об'єктів на зображенні. Наприклад, метод габаритного прямокутника дозволяє визначити найменший прямокутник, який повністю охоплює об'єкт на зображенні [5].

Для витягнення ознак також можуть використовуватися нейронні мережі [4], зокрема, згорткові нейронні мережі (ЗНМ).

ЗНМ дуже схожі на звичайні нейронні мережі прямого проходу, такі як перцептрон [7]. Вони все ще складаються з нейронів з вагами, які можна дізнатися з даних. Кожен нейрон отримує певні вхідні дані і виконує точковий добуток. Вони все ще мають функцію втрат на останньому повністю підключеному шарі. Вони все ще можуть використовувати функцію нелінійності. Перцептрон отримує вхідні дані як один вектор і проходить крізь серію прихованих шарів. Кожен прихований шар складається з набору нейронів, де кожен нейрон повністю пов’язаний з усіма іншими нейронами в

попередньому шарі. У межах одного шару кожен нейрон повністю незалежний, і вони не мають спільних зв’язків.

У випадку даних реальних зображень ЗНМ працюють краще, ніж багатошарові перцептрони (БП). Для цього є дві причини:

1) для передачі зображення в БП перетворюємо вхідну матрицю в простий числовий вектор без просторової структури. Він не знає, що ці числа просторово розташовані. Отже, ЗНМ створені саме з цієї причини; тобто для з’ясування закономірностей у багатовимірних даних. На відміну від БП, ЗНМ розуміють той факт, що пікселі зображення, які знаходяться ближче один до одного, більш пов'язані між собою, ніж пікселі, які знаходяться далі один від одного. ЗНМ складається з вхідного шару, прихованого шару і повністю підключеного шару.

2) ЗНМ відрізняються від БП типами прихованих шарів, які можуть бути включені в модель. ЗНМ організовує свої нейрони в трьох вимірах: ширина, висота та глибина. Кожен шар перетворює свій тривимірний вхідний об’єм у тривимірний вихідний об’єм нейронів за допомогою функцій активації.

Основна мета згортки по відношенню до ЗНМ – витягти ознаки з вхідного зображення. Цей рівень виконує більшу частину обчислень у ЗНМ.

Щоб створити згортковий шар у Keras [12], ви повинні спочатку імпортувати необхідні модулі. Потім можна створити згортковий шар, використовуючи формат Conv2D [13].

Необхідно передати такі аргументи:

– фільтри: кількість фільтрів;

– розмір ядра: число, що вказує як висоту, так і ширину (квадратного) вікна згортки. Є також деякі додаткові аргументи, які ви можете налаштувати;

– кроки: крок згортки. Якщо нічого не вказувати, це встановлюється на один;

– прокладка: дійсний або той саме. Якщо не вказувати, буде встановлено відступ дійсний.

– активація: якщо нічого не вказувати, активація не застосовується.

Використовуючи згортковий шар як перший шар (що з’являється після вхідного шару) у моделі, необхідно надати додатковий шар input_shape (аргумент input_shape). Це кортеж, що визначає висоту, ширину та глибину (у такому порядку) введення.

Більше фільтрів збільшує розмірність згортки. Більша розмірність означає більше параметрів. Таким чином, рівень об’єднання контролює переобладнання, поступово зменшуючи просторовий розмір представлення, щоб зменшити кількість параметрів і обчислень. Рівень об’єднання часто приймає згортковий шар як вхідні дані. Найбільш поширений підхід до об’єднання – максимальне об’єднання. На додаток до максимального об’єднання, блоки об’єднання можуть виконувати й інші функції, наприклад середнє об’єднання. У ЗНМ ми можемо контролювати поведінку згорткового шару, вказуючи розмір кожного фільтра та кількість фільтрів. Щоб збільшити кількість вузлів у згортковому шарі, ми можемо збільшити кількість фільтрів, а щоб збільшити розмір шаблону, ми можемо збільшити розмір фільтра. Є також кілька інших гіперпараметрів, які можна налаштувати. Одним з них є крок згортки.

Крок 1 переміщує фільтр на 1 піксель по горизонталі та вертикалі. Тут згортка стає такою ж, як ширина та глибина вхідного зображення.

Крок 2 складає згортковий шар, що дорівнює половині ширини та висоти зображення. Якщо фільтр виходить за межі зображення, ми можемо або ігнорувати ці невідомі значення, або замінити їх нулями.

У Keras можемо встановити Крок 1 переміщуваного фільтру на 1 піксель по горизонталі та вертикалі. Тут згортка стає такою ж, як ширина та глибина вхідного зображення.

Крок 2 складає згортковий шар, що дорівнює половині ширини та висоти зображення. Якщо фільтр виходить за межі зображення, ми можемо або ігнорувати ці невідомі значення, або замінити їх нулями. Згортковий шар допомагає виявити регіональні закономірності на зображенні.

1.1.6 Аналіз основних задач на етапі класифікації

Класифікація є одним з основних етапів розпізнавання об'єктів на супутникових знімках. На цьому етапі проводиться визначення класу об'єкта на зображенні на основі виділених ознак в попередньому етапі – витягненні ознак.

Для класифікації можуть використовуватися різноманітні алгоритми машинного навчання, зокрема нейронні мережі. Наприклад, для класифікації танків, бронетранспортерів та іншої військової техніки можна використовувати звичайну зворотно зв'язану нейронну мережу з декількома прихованими шарами.

На початкових етапах навчання нейронної мережі її параметри визначаються на основі навчальних даних, які складаються зі зображень військової техніки та їх класифікаційних міток. Після цього, на основі визначених параметрів, мережа може визначати класи об'єктів на нових зображеннях військової техніки.

Звичайний БП чудово працює для невеликих зображень (наприклад, MNIST [14] або CIFAR-10 [16]). Однак він не працює для великих зображень через величезну кількість необхідних параметрів. Наприклад, зображення розміром 100 × 100 має 10 000 пікселів, і якщо перший шар має лише 1 000 нейронів (що вже серйозно обмежує обсяг інформації, що передається наступному шару), це означає 10 мільйонів з’єднань; і це лише для першого шару.

ЗНМ вирішують цю проблему за допомогою частково зв'язаних шарів. Оскільки послідовні шари пов’язані лише частково, і оскільки він повторно використовує свої ваги, ЗНМ має набагато менше параметрів, ніж повністю підключений глибинна нейронна мережа (ГНМ), що робить його набагато швидшим, зменшує ризик переобладнання та вимагає набагато менше навчальних даних. Більше того, коли ЗНМ дізнається ядро, яке може виявити певну функцію, воно може виявити цю функцію в будь-якому місці зображення. На відміну від цього, коли ГНМ дізнається функцію в одному місці, вона може виявити її лише в цьому конкретному місці.

Оскільки зображення зазвичай мають дуже повторювані функції, ЗНМ можуть узагальнювати набагато краще, ніж ГНМ, для завдань обробки зображень, таких як класифікація, використовуючи менше навчальних прикладів. Важливо, що ГНМ не має попередніх знань про те, як організовані пікселі; він не знає, що поруч розташовані пікселі. Архітектура ЗНМ закладає ці попередні знання. Нижчі шари зазвичай визначають об’єкти на невеликих ділянках зображень, тоді як вищі шари об’єднують об’єкти нижнього рівня в більші об’єкти. Це добре працює з більшістю природних зображень, надаючи ЗНМ вирішальну перевагу в порівнянні з ГНМ.

В ЗНМ кожен шар представлений у 2D, що полегшує узгодження нейронів з відповідними входами. Приклади цього ми побачимо в наступних розділах. Іншим важливим фактом є те, що всі нейрони в карті ознак мають однакові параметри, тому це різко зменшує кількість параметрів у моделі; але що важливіше, це означає, що як тільки ЗНМ навчиться розпізнавати шаблон в одному місці, він може розпізнати його в будь-якому іншому місці.

Навпаки, як тільки звичайний ГНМ навчиться розпізнавати шаблон в одному місці, він може розпізнати його лише в цьому конкретному місці. У багатошарових мережах, таких як БП або DBN, виходи всіх нейронів вхідного шару підключаються до кожного нейрона в прихованому шарі, а потім вихід знову буде діяти як вхід до повністю підключеного шару. У мережах ЗНМ схема підключення, яка визначає згортковий шар, істотно відрізняється. Згортковий шар є основним типом шару в ЗНМ, де кожен нейрон підключений до певної області вхідної області, яка називається рецептивним полем.

У типовій архітектурі ЗНМ кілька згорткових шарів з'єднані в каскадному стилі. За кожним шаром слідує шар Rectified Linear Unit (ReLU), потім шар об’єднання, потім один або кілька згорткових шарів (+ReLU), потім інший шар об’єднання і, нарешті, один або кілька повністю пов’язаних шарів. Проте, залежно від типу проблеми, мережа може бути глибинною. Результатом кожного шару згортки є набір об’єктів, які називаються картами ознак, створених одним

фільтром ядра. Потім карти об’єктів можна використовувати для визначення нового входу до наступного шару.

1.1.7 Аналіз основних задач на етапі постобробки

Постобробка є важливим етапом в процесі розпізнавання об'єктів на зображеннях за допомогою нейронних мереж. Основна мета цього етапу полягає у покращенні якості результатів розпізнавання та зниженні кількості помилкових визначень класу об'єкта.

Оцінка результатів розпізнавання є першим кроком у постобробці. На цьому етапі робиться аналіз результатів розпізнавання та оцінка їхньої точності. Цей етап може включати порівняння результатів розпізнавання з еталонними даними та видалення помилкових визначень класу об'єкта.

Далі, можуть використовуватися різні методи для покращення результатів розпізнавання. Наприклад, можна використовувати методи об'єднання та розділення об'єктів з метою видалення помилок, або методи підтримки рішень для підвищення точності класифікації. Також можуть використовуватися методи видалення шумів та інших артефактів зображення.

Постобробка також може включати в себе ручне коригування результатів розпізнавання. Цей підхід дозволяє виправити помилки та неточності, які не були виявлені автоматичними методами розпізнавання. Однак, використання ручної корекції може значно збільшити час і витрати на обробку даних.

1.2 Аналіз наукових досліджень за напрямком розпізнавання військової техніки на зображеннях

Для дослідження методів розпізнавання військової техніки за допомогою нейронних мереж на супутникових знімках було проаналізовано ряд літературних джерел, що знаходяться у вільному доступі та розкривають можливості використання НМ для визначення військової техніки.

Так робота авторів І. В. Коваленка, І. М. Жуковського та інших [1] розглядала методи нейронних мереж та їх застосування у задачах розпізнавання об'єктів на зображеннях. В основному, ця стаття досліджує можливості нейронних мереж у розв'язанні задач, які вимагають високого рівня точності та швидкості обробки даних.

В роботі [2] описується система розпізнавання військової техніки на зображеннях, яка використовує методи машинного навчання та штучних нейронних мереж. Автори статті проводять дослідження ефективності системи на відомих наборах даних та показують високі результати її роботи.

А у роботі [3] розглядається застосування нейронних мереж для автоматичної класифікації зображень, зокрема для розпізнавання військової техніки на знімках. Автори використовували зображення з бази даних техніки та здійснили їх попередню обробку, підготовку та виділення ознак.

Ще одна стаття, що стосується даної теми, це "Абрамов С.В., Лупаленко О.В., Манжай О.В. Аналіз автоматизованих систем виявленнята розпізнавання об’єктів збройних сил російської федерації". У цій статті автори розглядають задачу розпізнавання танків на знімках з дронів [4].

Стаття Мустафіна Р. та інших [5] також стосується даної теми та розкриває особливості застосування методів машинного навчання для розпізнавання військової техніки на знімках з дронів.

На основі аналізу низки статей про готові програмні рішення щодо розпізнавання зображень було отримано наступні результати:

– в статті [6] проаналізовано застосування згорткових нейронних мереж для виявлення пішоходів на зображеннях та відео, де автори статті проводять огляд різних методів з використанням згорткових нейронних мереж для виявлення пішоходів, описують їх переваги та недоліки та надають рекомендації щодо подальшого розвитку цієї галузі досліджень;

– з статті [7] отримано приклади розв’язку задачі семантичної сегментації зображень з використанням глибоких згорткових нейронних мереж. Автори пропонують нову архітектуру мережі з використанням atrous convolution та fully

connected Conditional Random Fields (CRFs), яка забезпечує більш точну сегментацію зображень. Результати, отримані з використанням запропонованої архітектури, демонструють її високу ефективність порівняно з іншими методами;

– стаття [8] надала опис методів та рішень, які використовуються для задачі виявлення об'єктів на зображеннях та відео з використанням глибоких згорткових нейронних мереж. Автори описують основні архітектури нейронних мереж, що використовуються для виявлення об'єктів, а також розглядають питання даних та підходів до навчання таких мереж. Крім того, автори порівнюють різні методи та архітектури мереж та надають рекомендації щодо подальшого розвитку цієї галузі досліджень.

Дослідження також спиралось на аналіз навчальної літератури:

– книга "Космічна фотограмметрія" авторства Железняка О.О. та Чубко Л.С. [9] описує основні методи та технології, що використовуються в космічній фотограмметрії – науці, яка займається обробкою фотографій Землі, зроблених з космосу. Книга містить розділи про космічні знімальні системи, процеси ортокорекції, триангуляції, визначення орбіт супутників;

– книга "Аерокосмічні знімальні системи" авторства Бурштинської Х.В. та Станкевича С.А. [10] описує різні технології та обладнання, що використовуються для зйомки повітряних та космічних знімків. Автори надають детальний опис різних типів камер, супутників, аеростатів та іншого обладнання, що використовується в аерокосмічній зйомці. Книга також містить інформацію про обробку та аналіз отриманих знімків.

– книга "Дистанційні дослідження Землі" [11] є навчальним посібником і містить опис методів і засобів дистанційного зондування Землі. Автори детально розглядають фізичні основи, прилади та технології збору та обробки даних з різних дистанційних джерел, таких як супутники, літаки та дрони. Книга також присвячена застосуванню дистанційного зондування для розв'язання різних завдань, зокрема, для дослідження клімату, розробки карт та геодезичних робіт;

Для побудови ЗНМ та оцінки можливостей ГМН були використані матеріали наступних статей:

– стаття [12] описує метод оптимізації показника Intersection-Over-Union (IoU) у глибоких нейронних мережах для задач семантичної сегментації зображень. Автори пропонують використовувати підхід, що базується на поєднанні IoU та binary cross-entropy, та наводять експериментальні результати на різних датасетах;

– стаття [13] присвячена архітектурі Inception, яка використовується в комп'ютерному зорі. Автори статті пропонують ряд змін до оригінальної архітектури, щоб покращити її ефективність та зменшити кількість параметрів. Автори наводять результати експериментів на декількох датасетах, що показують покращення в порівнянні з попередніми версіями Inception.

– стаття [14] описує порівняльний аналіз між системою навчання Inception-v3 та іншими системами для виявлення виразів обличчя. Результати показали, що Inception-v3 має вищу точність в порівнянні з іншими системами.

– у статті [15] описана нова архітектура глибинної нейронної мережі – Deep Residual Learning, яка дає значно кращі результати в розпізнаванні зображень порівняно з попередніми моделями;

– стаття [17] описує швидку систему виявлення об'єктів, яка базується на архітектурі MobileNetv2 та покращеному функціоналу Feature Pyramid. Дослідження показали високу точність та швидкість виявлення об'єктів на зображеннях;

– стаття [18] описує метод Multiresolution Gray-Scale і Rotation Invariant Texture Classification з використанням локальних бінарних шаблонів. Цей метод дозволяє відрізняти текстури на зображенні незалежно від їхнього розміру, під кутом або зміщенням.

– у статті [19] описані найкращі практики застосування згорткових нейронних мереж візуального аналізу документів. Автори пропонують шляхи покращення точності та швидкості розпізнавання тексту на зображеннях.

Для аналізу сучасних наукових тенденцій у розвитку систем навчання НМ було проаналізовано наступні роботи:

– стаття [20], що описує методики використання генетичних алгоритмів для проектування архітектури згорткових нейронних мереж з метою поліпшення їхньої точності. Результати дослідження показали, що запропонована методика дозволяє досягти кращих результатів порівняно з традиційними підходами до проектування архітектури.

– стаття [21], що описує метод глибокого навчання на основі глибоких гаусівських змішаних моделей для факторизації варіацій в природних зображеннях. Результати дослідження показали, що запропонований метод працює ефективніше, ніж інші аналогічні методи на різних задачах;

– стаття [22], що описує підхід до навчання нейронних мереж, який дозволяє використовувати як позначені, так і непозначені дані для навчання мережі. Результати експериментів показали, що запропонований підхід працює краще за інші методи на різних задачах, зокрема на задачі класифікації текстів.

Безпосередньо побудову НМ було зроблено на основі документацій і методик з наступних джерел:

– DeepLearning 0.1. LISA Lab [23] – це набір практичних уроків з глибокого навчання на мові програмування Python, що розроблений Лабораторією машинного навчання Лінкольнського університету (LISA Lab);

– електронна книга [24] досліджує використання практичне застосування до виявлення та класифікації дорожніх знаків згорткових нейронних мереж у візуальному розпізнаванні знаків дорожнього руху. Книга містить приклади програмного коду та детальні інструкції щодо розробки власної моделі згорткової нейронної мережі.

1.2 Стандартні алгоритми розпізнавання зображень

Найбільш ефективними методами виведення нових знань є методи розпізнавання образів на основі навчання (самонавчання) [11, 12, 13].

Стандартні алгоритми розпізнавання зображень включають наступні методи та підходи:

1. Застосування згорткових нейронних мереж (Convolutional Neural Networks - CNN): Цей підхід базується на використанні глибоких нейронних мереж зі спеціальною архітектурою, що включає згорткові шари для виявлення локальних ознак зображень та повно зв’язані шари для класифікації. CNN здатні автоматично вивчати репрезентативні функції зображень, що дозволяє їм ефективно розпізнавати об'єкти та патерни на зображеннях.

2. Метод опорних векторів (Support Vector Machines - SVM): Цей метод базується на використанні лінійних або нелінійних моделей для класифікації зображень. SVM шукає границю розділення між класами з найбільшою маржою та використовує ядерні функції для розпізнавання нелінійних залежностей.

3. Метод k-найближчих сусідів (k-Nearest Neighbors - k-NN): Цей метод використовує класифікацію на основі найближчих сусідів. Він шукає k найближчих сусідів до тестового зображення в просторі ознак та призначає йому клас, який є найбільш представленим серед цих сусідів.

4. Байєсовські класифікатори (Naive Bayes Classifiers): Ці класифікатори використовують теорему Байєса для визначення ймовірностей належності зображення до певного класу на основі вхідних ознак. Вони припускають незалежність ознак та використовують статистичні методи для призначення класу.

Щодо відображення знань при розпізнаванні зображень виділяють [14]:

– інтенсіональне відображення (Intensional Representation): Цей спосіб відображення знань передбачає представлення абстрактних понять та відношень між ними. Він фокусується на сутності зображення, його характеристиках та атрибутах, які можуть бути використані для класифікації та розпізнавання;

– екстенсіональне відображення (Extensional Representation): Цей спосіб відображення знань передбачає конкретне перерахування екземплярів об'єктів на зображенні. Він зосереджується на фактичних зображеннях та їхніх властивостях, що можуть бути використані для порівняння та класифікації.

Ці два способи відображення знань можуть використовуватись в розпізнаванні зображень для представлення та використання інформації про об'єкти на зображеннях.

Інтенсіональне відображення фіксує закономірності і зв'язки, якими пояснюється структура даних та відношення між їхніми характеристиками. В контексті розпізнавання зображень, це означає виявлення і використання абстрактних понять та характеристик об'єктів на зображенні для класифікації та розпізнавання.

Наприклад, при використанні інтенсіонального відображення у розпізнаванні обличь, можуть бути визначені закономірності, що вказують на розташування очей, носа та рота на зображенні. Ці закономірності можуть бути представлені у вигляді абстрактних понять, таких як "очі", "ніс", "рот", а також відношень між ними, таких як "розміщення очей над носом" або "розташування рота під носом".

Інтенсіональне відображення дозволяє зберегти інформацію про структуру даних та їх внутрішні зв'язки. Воно використовується для створення моделей, що описують сутність об'єктів та їхні відношення, що допомагає у розумінні та класифікації зображень.

У контексті розпізнавання зображень, екстенсіональне відображення може включати список конкретних зображень військової техніки, які розпізнаються моделлю. Кожне зображення представляє окремий екземпляр об'єкта, і вони можуть бути перелічені або збережені в базі даних для подальшого використання.

Екстенсіональне відображення може бути корисним для конкретних випадків розпізнавання, коли необхідно точно визначити, які об'єкти присутні на зображенні. Воно забезпечує деталізовану інформацію про конкретні об'єкти, їх властивості та розміри.

Однак, екстенсіональне відображення може бути обмеженим в масштабованості, оскільки вимагає конкретного перерахування всіх можливих

екземплярів об'єктів. Великі набори даних можуть бути складними для обробки і вимагати значних обчислювальних ресурсів для збереження та обробки.

Загалом, екстенсіональне відображення є важливим способом відображення знань в контексті розпізнавання зображень, зокрема для точного ідентифікування та класифікації конкретних об'єктів на зображеннях.

Аналіз перспективних напрямків розвитку методів розпізнавання показує, що для успішного досягнення мети дослідження необхідно вирішити (або обійти) такі проблеми:

– висока складність обробки великого обсягу даних, яка означає, що з ростом обсягу даних, таких як великі набори зображень, виникають проблеми з обробкою та аналізом цих даних. Необхідно розробити ефективні алгоритми та методи для обробки великих обсягів даних та забезпечення швидкості та ефективності розпізнавання;

– недостатня точність та стабільність, яка означає, що у деяких випадках методи розпізнавання можуть давати неточні результати або бути чутливими до змін у вхідних даних, таких як зміна освітлення або шуму. Потрібно розробити методи, які забезпечать високу точність та стабільність навіть у незвичайних умовах;

– обробка зображень низької якості, яка передбачає, що у реальних умовах зображення військової техніки можуть бути низької якості, змазані або містити спотворення. Це ускладнює розпізнавання та класифікацію. Необхідно розробити методи та алгоритми, які зможуть ефективно працювати з низькоякісними зображеннями та враховувати їх особливості;

– розпізнавання об'єктів у складних сценах, яка означає, що у воєнних умовах, об'єкти військової техніки можуть знаходитися в складних сценах, таких як міські вулиці або лісисті райони. Розпізнавання в цих умовах стає складним через наявність перешкод, накладання об'єктів або зміну перспективи. Потрібно розробити методи, які зможуть ефективно розпізнавати об'єкти в складних сценах та враховувати їх контекст;

– недостатня робастність до варіацій, яка означає, що розпізнавання військової техніки повинне бути робастним до варіацій у зовнішньому вигляді об'єктів, таких як зміна кольору, форми або масштабу. Необхідно розробити методи, які будуть ефективними при розпізнаванні об'єктів зі значними варіаціями та здатні враховувати ці варіації.

Вирішення цих проблем дозволить покращити точність та надійність методів розпізнавання військової техніки, що забезпечить більш ефективне використання цих методів у практичних ситуаціях.

1.3 Аналіз вимог до програмного модуля і постановка задачі на розробку

З огляду на аналіз відкритих джерел, де представлено приклади роботи подібних модулів [27, 28], сформовано функціональні і нефункціональні вимоги до програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками:

1) функціональні вимоги:

– можливість завантаження супутникових знімків: модуль повинен мати можливість завантажувати супутникові знімки військової техніки з вказаної директорії або іншого джерела даних;

– можливість проводити обробку супутникових знімків: модуль повинен здійснювати обробку зображень для попередньої підготовки даних перед процесом розпізнавання. Це може включати зміну розміру зображень, нормалізацію пікселів, видалення шуму або покращення контрастності.

– можливість проводити розпізнавання військової техніки на завантажених супутникових знімках: модуль повинен розпізнавати об'єкти військової техніки на зображеннях і класифікувати їх згідно з типом техніки. Вихідним результатом повинні бути ідентифіковані об'єкти та їх класифікація.

– візуалізація результатів: модуль повинен забезпечувати візуалізацію результатів розпізнавання, наприклад, шляхом виділення об'єктів на зображенні

або створення матриці невідповідностей (confusion matrix) для оцінки точності класифікації, виводу інформації про тип розпізнаної військової техніки;

– мати можливість навчати нейронну мережу на нових дата сетах, для чого передбачити інструменти завантаження групи файлів з розміткою і навчання нейронної мережі на цих даних (використовувати ЗНМ і відкриті бібліотеки TensorFlow, Keras).

2) нефункціональні вимоги:

– швидкодія: модуль повинен забезпечувати швидку обробку та розпізнавання зображень в реальному часі;

– точність: модуль повинен досягати високої точності розпізнавання (більше 90%), щоб забезпечити надійні результати класифікації військової техніки;

– масштабованість: модуль повинен бути гнучким та масштабованим, здатним працювати з різними розмірами зображень та оброблювати великі обсяги даних;

– надійність: модуль повинен бути надійним і стійким до помилок, забезпечуючи коректну обробку зображень та точні результати розпізнавання.

Вирішення задачі розпізнавання військової техніки на зображеннях передбачає розробку і програмну реалізацію алгоритмів обробки зображень, які повинні підтримувати вирішення наступних задач: попередня обробка, сегментація, витягнення ознак, класифікація та постобробка. Кожна з перерахованих задач передбачає використання методів, які повинні дозволити розпізнавати військову техніку на зображеннях з достатнім рівнем якості результатів.

З огляду на функціональні і нефункціональні вимоги до прогрмного модуля розпізнавання військової техніки за супутниковими знімками в результаті кваліфікаційної роботи необхідно виконати наступні завдання:

1) провести аналіз сучасних методів розпізнавання і класифікації зображень;

2) провести аналіз методів побудови і навчання штучних нейронних мереж для розпізнавання зображень та розглянути існуючи програмні системи розпізнавання зображень з використанням нейронних мереж;

3) розробити програмний модуль інтелектуального розпізнавання військової техніки за супутниковими знімками, який забезпечує завантаження, попередня обробку і сегментацію зображень, класифікацію визначених на зображеннях об’єктів та представлення результатів розпізнавання наявних на зображенні типів військової техніки через інтерфейс користувача;

4) провести навчання нейронної мережі для розпізнавання військової техніки за супутниковими знімками та оцінити якість навчання;

5) провести тестування програмного модуля та надати описання режимів його використання.

Поставлені завдання вимагають реалізації складної системи у відповідності до функціональних і нефункціональних вимог.

1.4 Висновки до першого розділу

Дослідження етапів роботи та методів, які в них використовуються, є важливими для розробки автоматизованих систем розпізнавання зображень.

На основі аналізу відомих принципів обробки зображень та наукових досліджень за подібними тематиками було визначено основні етапи розробки системи і методи, які використовуються на кожному з етапів.

Для кожного етапу було визначено основні методи та алгоритми для обробки даних, такі як порогова обробка, алгоритми визначення контуру, методи згладжування та підсилення границь, а також різні архітектури нейронних мереж для класифікації.

Дослідження показало, що практика використання глибинних нейронних мереж у поєднанні з методами обробки зображень дає найкращі результати в розпізнаванні військової техніки на зображеннях. На основі аналізу відкритих проектів з розпізнавання військової техніки сформовано функціональні і

нефункціональні вимоги до програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками та визначено завдання до кваліфікаційної роботи.

РОЗДІЛ 2 РЕАЛІЗАЦІЯ ПРОГРАМНОГО МОДУЛЯ ІНТЕЛЕКТУАЛЬНОГО РОЗПІЗНАВАННЯ ВІЙСЬКОВОЇ ТЕХНІКИ НА ЗОБРАЖЕННЯХ

2.1 Аналіз основних процесів предметного середовища

Головна задача програмного модуля – розпізнавати техніку на супутникових знімках. Головні критерії для користування програмою це максимальний простий функціонал та можливість додавати нові зображення та класи для сортування.

2.1.1 Функціональний аналіз

Для початку розробки архітектури інтелектуальної системи розпізнавання військової техніки необхідно розробити дерево функцій для відображення базових функцій модуля, серед яких можна виділити:

– функції інтерфейсу користувача;

– функції інтерфейсу інженера знань;

– функції інтерфейсу адміністратора системи.

До функцій інтерфейсу користувача відносяться:

– завантаження супутникових знімків;

– обробка супутникових знімків;

– розпізнавання та класифікація супутникових знімків з виявленим типів військової техніки;

– візуалізація результатів розпізнавання супутникових знімків.

Функції інтерфейсу інженера знань передбачають роботи з створення дата сету зображень військової техніки і навчання системи. Дані роботи можна розкласти на наступні функції:

1. Функції роботи з дата сетом зображень:

– завантаження супутникових знімків до датасету;

– видалення шуму: функція, яка використовує методи обробки зображень для видалення шуму зі зображень, таких як згладжування, фільтрація та зменшення роздільної здатності;

– обрізка та ресайз зображень: функція, яка зменшує розмір зображення та вирізає зображення військової техніки з фону, щоб полегшити процес обробки;

– класифікація супутникових знімків: функція, яка використовує методи машинного навчання, такі як нейронні мережі, для класифікації зображень військової техніки зі збору даних;

– внесення розмітки до супутникових знімків: функція, яка дозволяє визначати місцеположення та клас військової техніки на зображенні для навчання системи;

– збереження датасету: функція, яка зберігає дата сет зображень військової техніки для навчання системи.

2. Функції для навчання системи:

– підготовка даних: функція, яка підключає вже підготовлений дата сет для навчання системи;

– налаштування нейронної мережі: функція, яка створює налаштовує нейронну мережу для класифікації зображень;

– навчання нейронної мережі: функція, яка забезпечує навчання нейронної мережі на дата сеті зображень військової техніки;

– оцінка нейронної мережі: функція, яка оцінює точність розпізнавання на тестовому дата сеті;

– підготовка моделі до використання: функція, яка зберігає навчену модель для використання в майбутньому;

– моніторинг процесу навчання: функція, яка дозволяє відстежувати процес навчання моделі, щоб визначити, чи потрібно внести зміни у параметри навчання або додаткову підготовку даних.

Функції інтерфейсу адміністратора системи:

– введення нових користувачів;

– налаштування прав користувачів.

Графічно функції представлено на рис. 2.1.

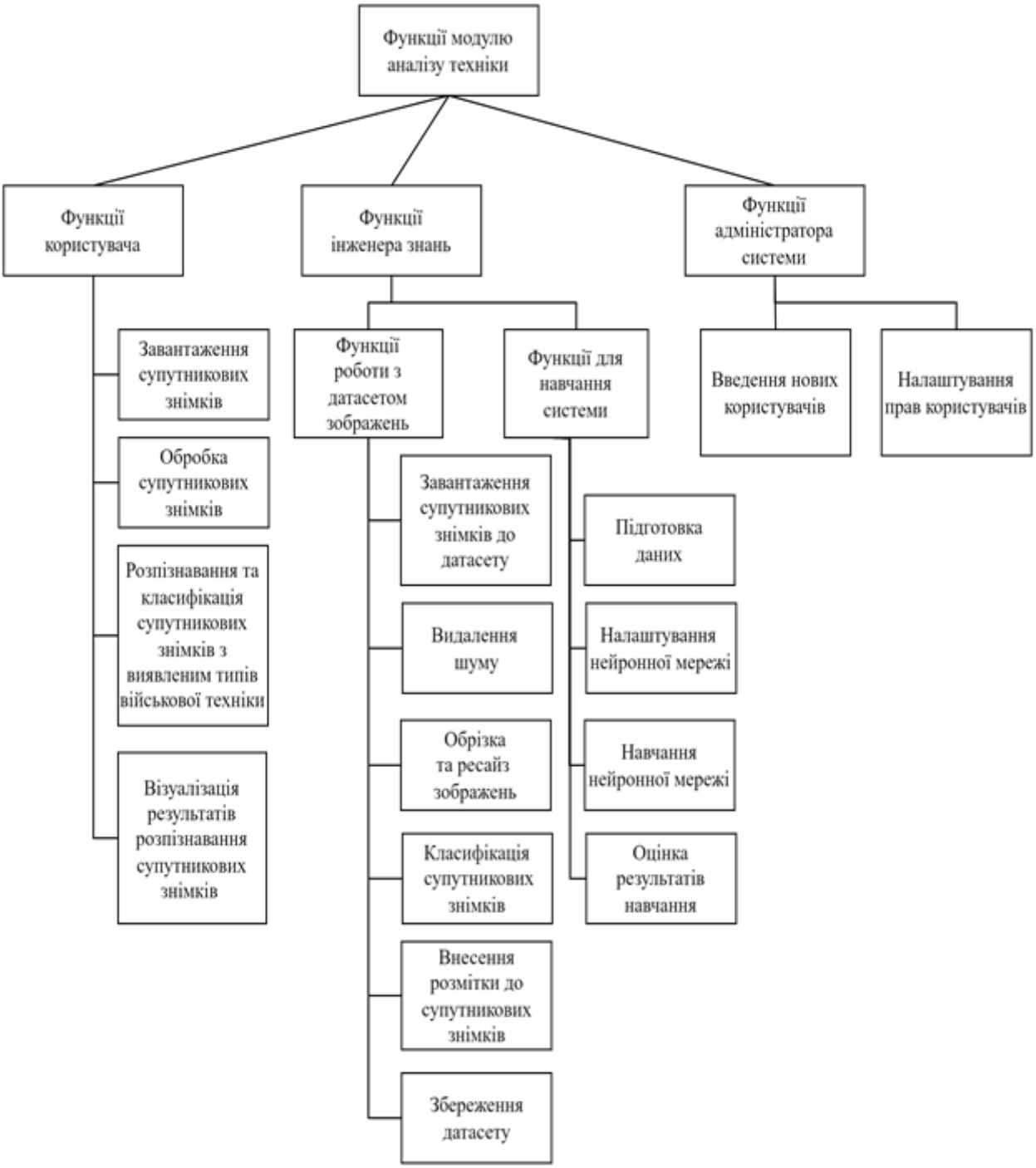

Рисунок 2.1 – Дерево функцій програмного модуля інтелектуального розпізнавання військової техніки

Наступним етапом є опис принципу функціонування застосунку для інтелектуальної системи розпізнавання військової техніки.

2.1.2 Діаграма IDEF0 програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками

Контекстна діаграма "ЯК БУДЕ" для програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками в нотації IDEF0 може бути наступною (рис. 2.2).

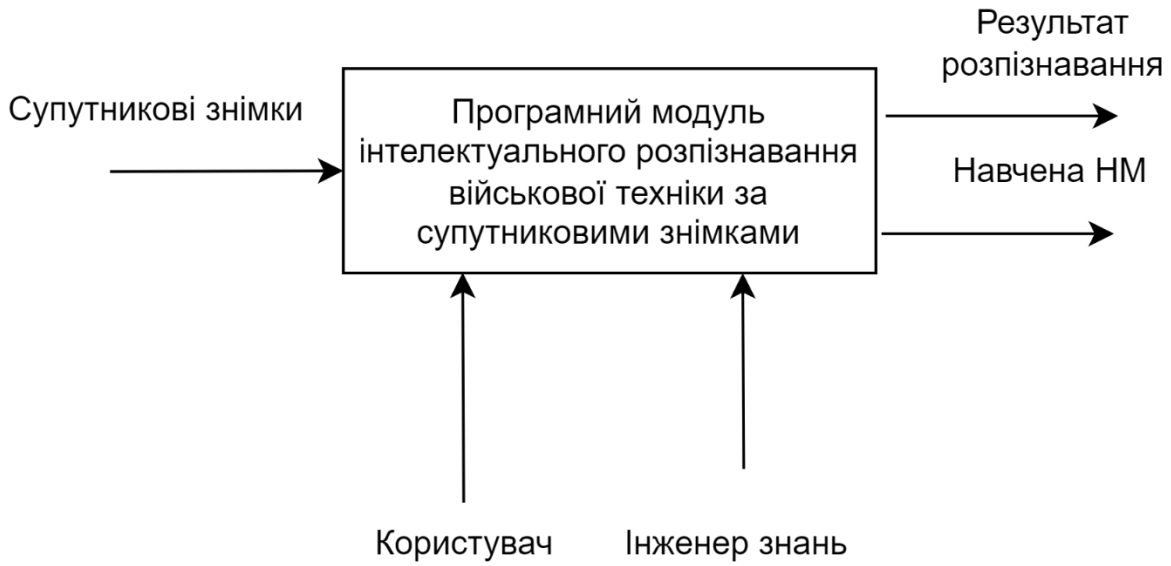

Рисунок 2.2 – Контекстна діаграма ЯК БУДЕ програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками

На контекстній діаграмі, військові знімки проходять спочатку попередню обробку зображень, щоб видалити зайву інформацію та підготувати зображення для подальшого аналізу. Після цього, зображення надходять до модуля розпізнавання військової техніки, який використовує різні методи машинного навчання та алгоритми обробки зображень для розпізнавання техніки. Результатом є інформація про техніку, яка може бути використана для подальшої обробки та аналізу.

Основними компонентами програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками є:

– супутникові знімки;

– користувачі (Користувач та Інженер знань);

– результати розпізнавання;

– навчена НМ.

На вході система отримує супутникові знімки і в залежності від інтерфейсу користувача може проводити розпізнавання супутникових знімків або навчання нейронної мережі.

Для декомпозиції контекстної діаграми «ЯК БУДЕ» за моделлю IDEF0 можна визначити два шляхи використання програмного модуля: розпізнавання супутникових знімків та навчання нейронної мережі (рис. 2.3).

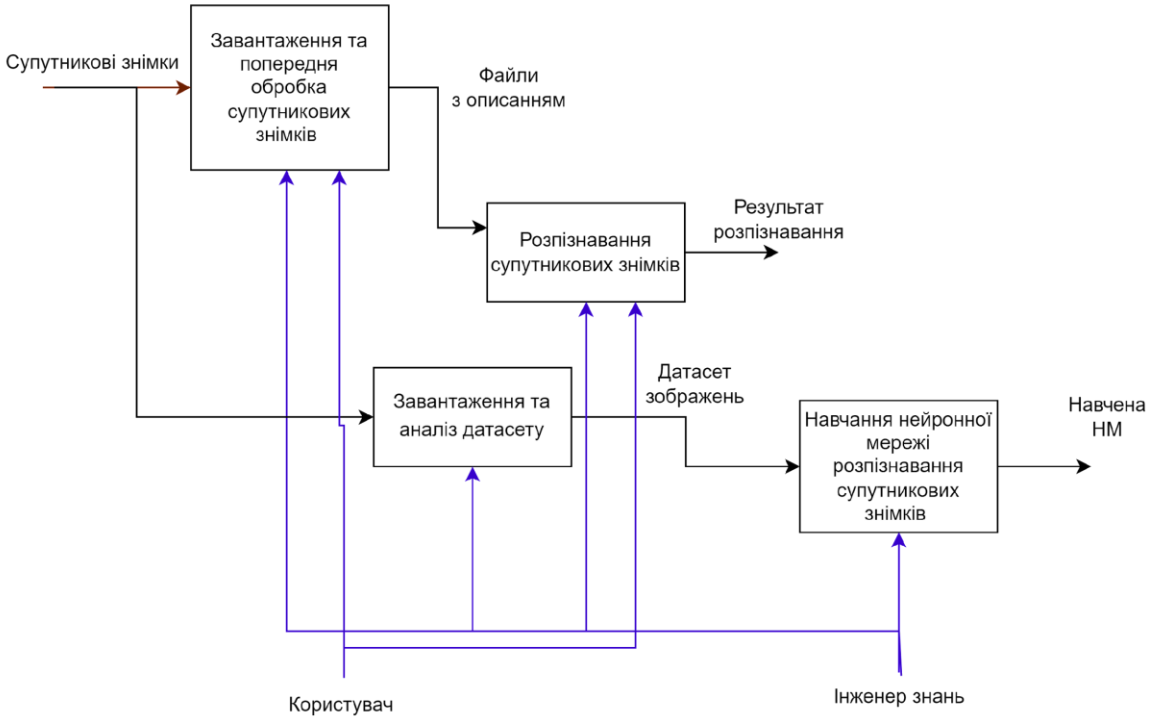

Рисунок 2.3 – Діаграма IDEF0 програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками рівня 1

Шлях розпізнавання супутникових знімків передбачає процеси:

– завантаження та попередня обробки супутникових знімків (завантаження знімків, попередня обробка супутникових знімків, декомпозиція об’єктів);

– розпізнавання знімків (класифікація техніки, генерація результатів, передача результатів).

На виході буде отримано результат розпізнавання.

Шлях навчання нейронної мережі розпізнавання супутникових знімків передбачає процеси

– завантаження та аналіз дата сету;

– навчання нейронної мережі розпізнавання супутникових знімків.

Вихід – навчена нейронна мережа.

На рис. 2.4 представлено діаграму IDEF0 рівень 1 процесу завантаження та попередньої обробки супутникових знімків.

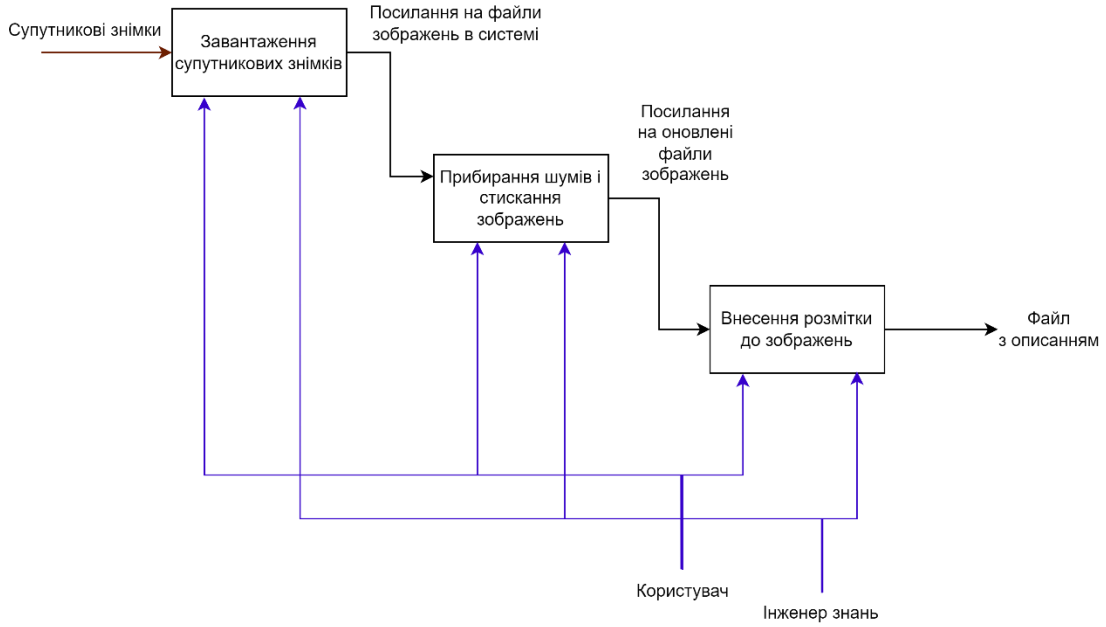

Рисунок 2.4 – Діаграма IDEF0 рівень 1 процесу завантаження та попередньої обробки супутникових знімків

На рис. 2.5 представлено діаграму IDEF0 результату декомпозиції процесу навчання нейронної мережі розпізнавання супутникових знімків.

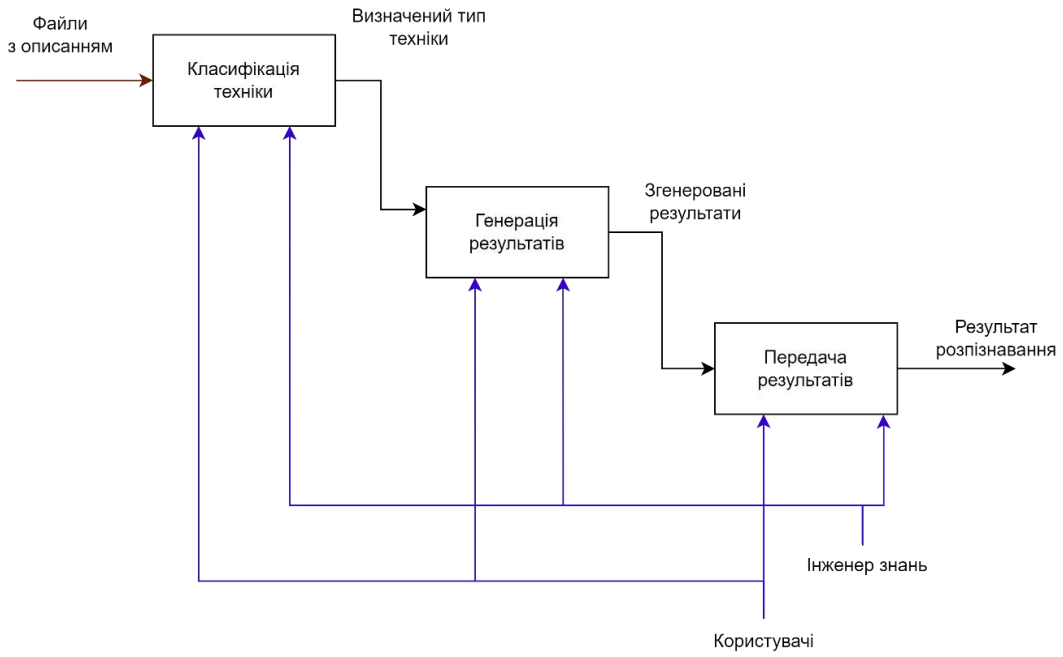

Рисунок 2.5 – Діаграма IDEF0 рівень 1 процесу розпізнавання супутникових знімків

На рис. 2.6. представлено діаграму у форматі IDEF0 рівень 1 процесу завантаження та аналізу дата сету.

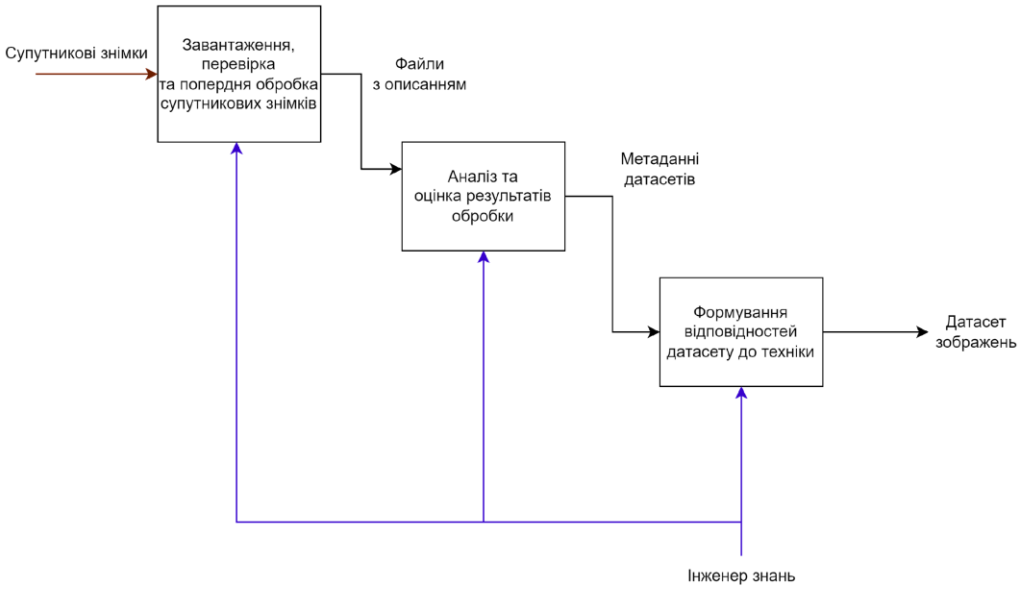

Рисунок 2.6 – Діаграма IDEF0 рівень 1 процесу завантаження та аналізу дата сету

На рис. 2.7. представлено діаграму у форматі IDEF0 рівень 1 процесу навчання нейронної мережі розпізнавання супутникових знімків.

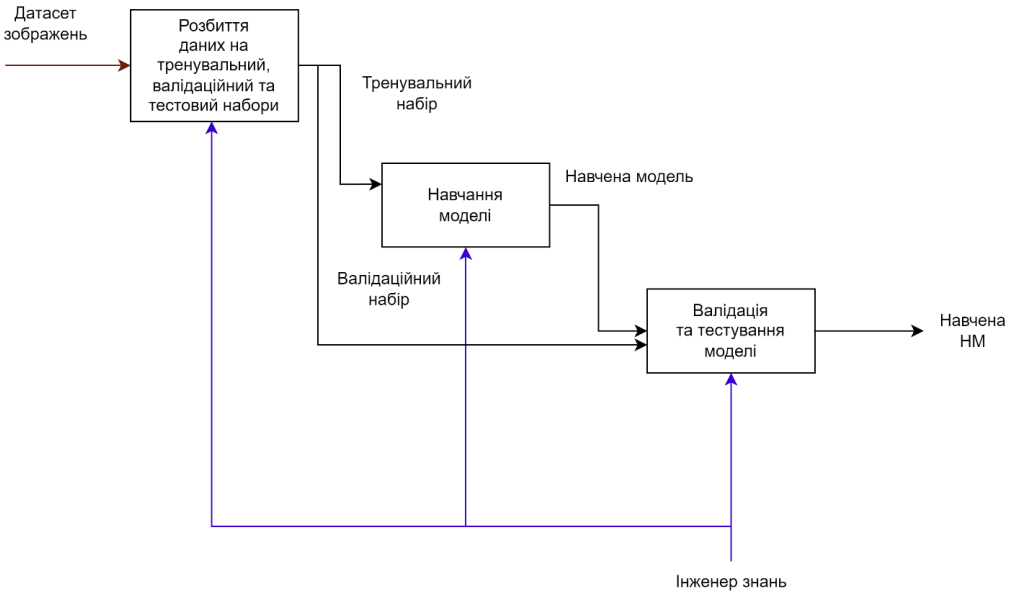

Рисунок 2.7 – Діаграма IDEF0 рівень 1 процесу навчання нейронної мережі розпізнавання супутникових знімків

На основі декомпозиції процесів і функціональних вимог до програмного модуля можна візуалізувати use-case діаграму (рис. 2.8), в якій передбачено три актори, але функції в ході розробки системи є сенс винести функції адміністратора за межі програмного моду інтелектуального розпізнавання військової техніки за супутниковими знімками.

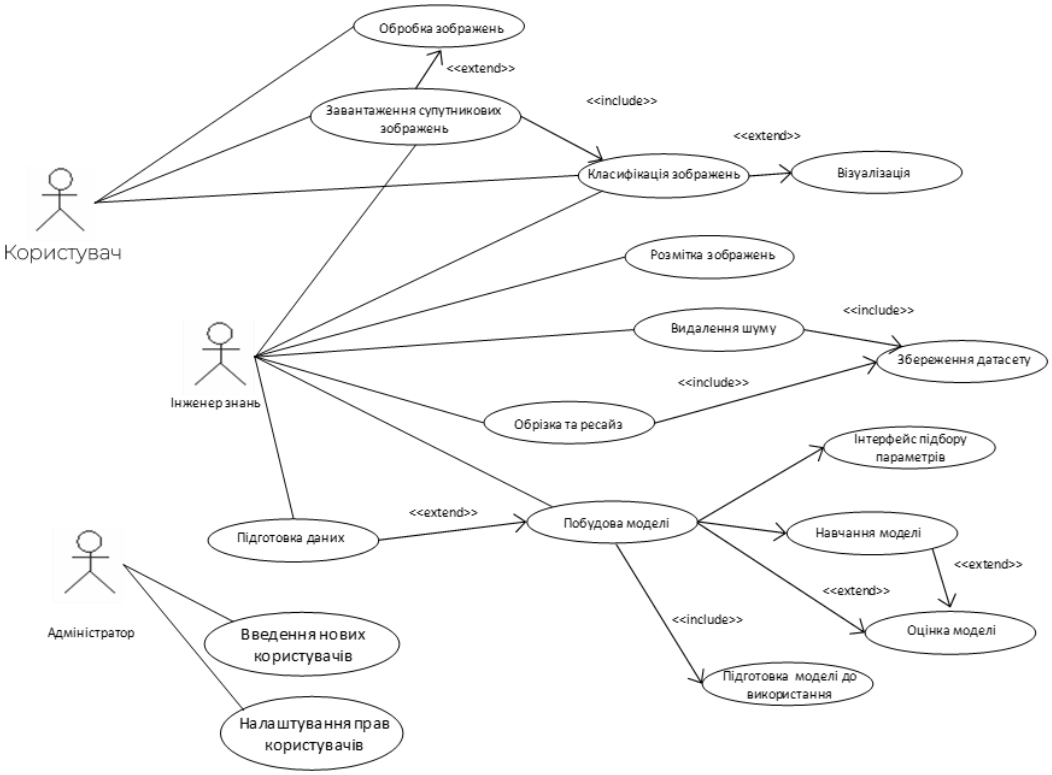

Рисунок 2.8 – Діаграма використання модуля

2.1.3 Архітектура інтелектуального програмного модуля

Наступним етапом є розробка архітектури програмного модуля (рис. 2.9). Дана схема відображає структуру і взаємозв'язок між компонентами.

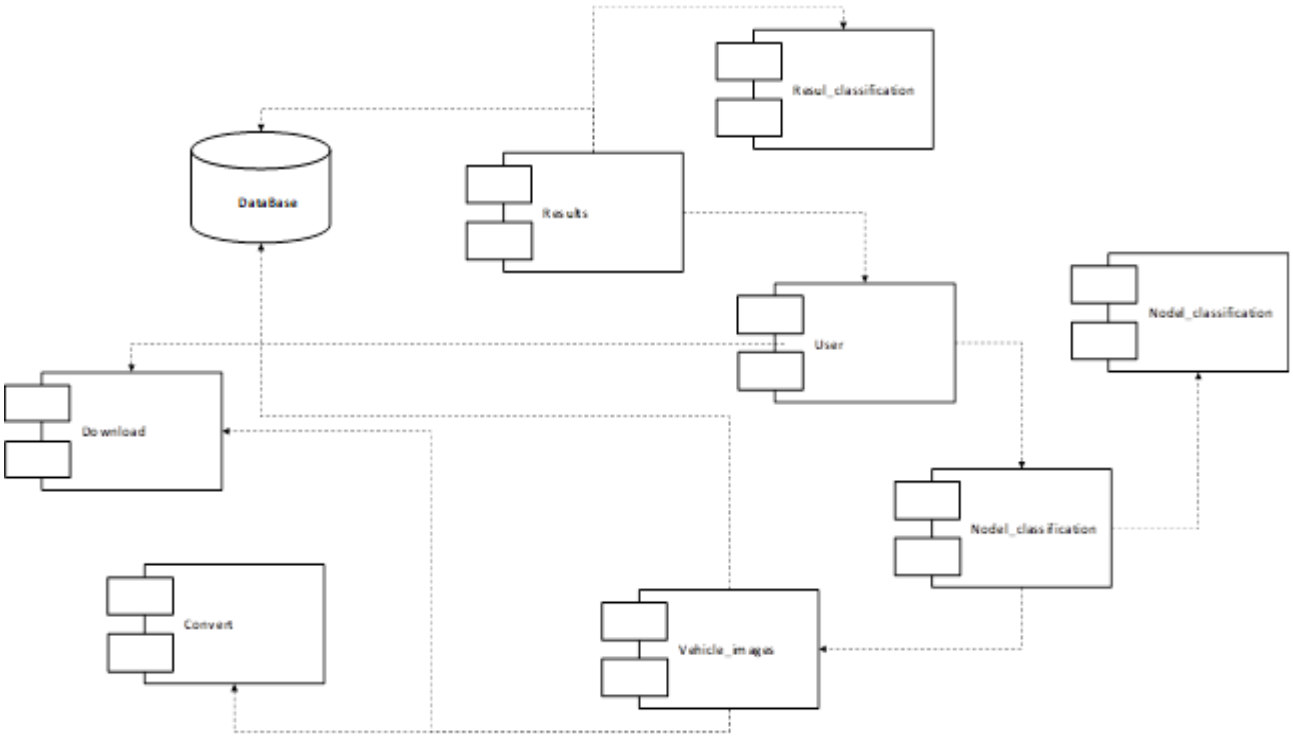

Рисунок 2.9 – Діаграма компонентів модуля інтелектуального розпізнавання об’єктів на супутникових знімках

На вершині знаходиться головний модуль розпізнавання військової техніки який включає в себе(в якості компонентів) два модулі: клієнтський та адміністративний.

До клієнтського модуля в якості компонентів входять модуль завантаження зображень та модуль формування інтерфейсу. В свою чергу модуль завантаження зображень включає в себе модуль обробки і класифікації об’єктів та модуль візуалізації.

Модуль адміністративної частини включає в себе: модуль навчання нейронної мережі, модуль обробки зображень, та модуль завантаження баз знімків.

2.2 Проектування інформаційного забезпечення програмного модуля інтелектуального розпізнавання військової техніки

База даних для роботи програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками повинна містити наступну інформацію:

1) зображення військової техніки: це можуть бути зображення танків, бронетранспортерів, військових літаків тощо, які були використані для тренування моделі розпізнавання;

2) класифікація військової техніки: це може бути інформація про тип військової техніки (наприклад, танк, бронетранспортер, військовий літак тощо);

3) інформація про параметри військової техніки: це може включати інформацію про розмір, швидкість, рівень палива;

4) інші додаткові відомості: це можуть бути будь-які додаткові відомості про військову техніку, такі як інформація про її властивості та характеристики;

5) результати попереднього аналізу знімків: можуть містити інформацію про параметри знімків, такі як роздільна здатність, кут огляду, освітлення тощо, які можуть впливати на якість розпізнавання військової техніки, місцеположення об’єктів для аналізу (розпізнані об'єкти, їх розмір, форму, орієнтацію та інші властивості). Ця інформація може бути використана для подальшої класифікації та ідентифікації військової техніки на зображеннях.

База даних повинна бути добре структурованою і організованою таким чином, щоб модуль інтелектуального розпізнавання міг швидко і ефективно звертатися до неї для отримання необхідної інформації. Концептуальна модель на рис. 2.10.

На основі концептуальної моделі можна побудувати логічну модель БД (рис. 2.11).

З урахуванням концептуальної та логічної моделі є можливість перейти до створення БД.

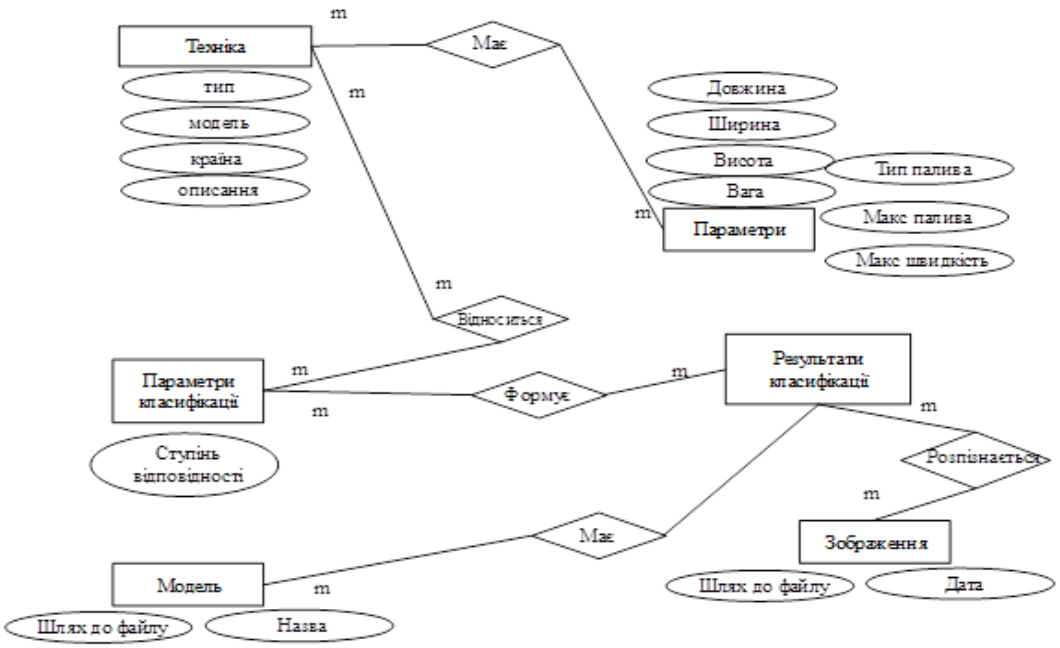

Рисунок 2.10 – Концептуальна модель БД

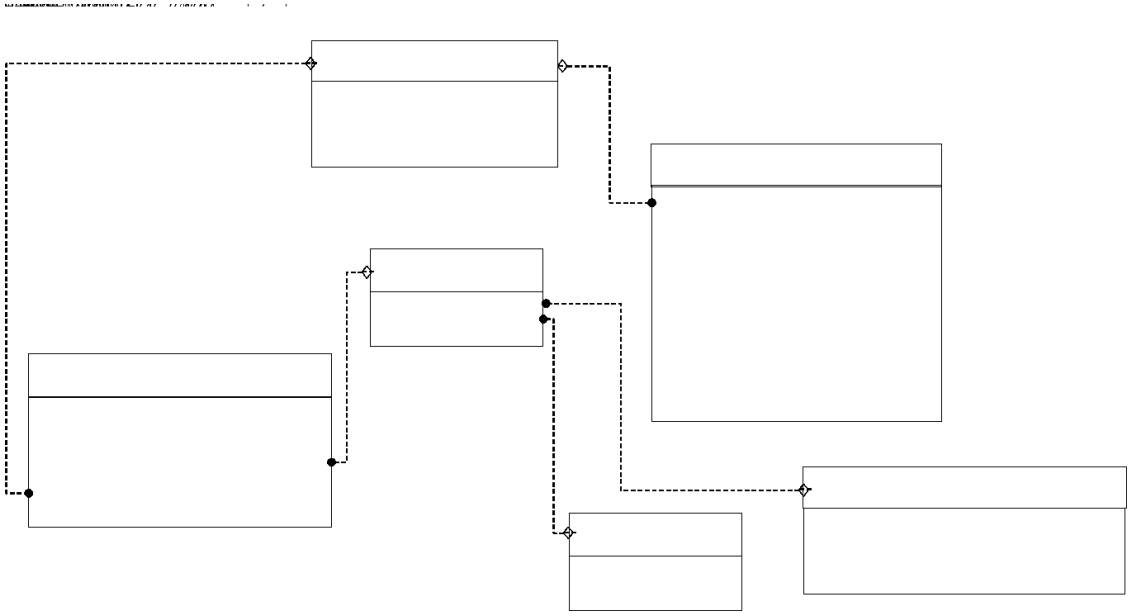

Рисунок 2.11 – Логічна модель БД

Для збереження інформації про класифікацію військової техніки можна створити таблицю "vehicles" з наступними полями:

– vehicle_id – унікальний ідентифікатор транспортного засобу (первинний ключ).

– vehicle_type – тип транспортного засобу (танк, бронетранспортер, військовий літак тощо).

– vehicle_model – модель транспортного засобу.

– country – країна виробник транспортного засобу.

– description – опис транспортного засобу.

Так для збереження інформації про параметри військової техніки можна створити таблицю "vehicle_parameters" з наступними полями:

– id_vehicle – ідентифікатор транспортного засобу (зовнішній ключ, що посилається на таблицю "vehicles").

– length – довжина транспортного засобу.

– width – ширина транспортного засобу.

– height – висота транспортного засобу.

– weight – вага транспортного засобу.

– max_speed – максимальна швидкість транспортного засобу.

– fuel_type – тип палива, який використовується для транспортного засобу.

– fuel_consumption – середнє споживання палива транспортним засобом.

Така структура таблиці дозволить зберігати інформацію про розміри, вагу, швидкість, тип палива та споживання палива транспортного засобу. Зв'язок з таблицею "vehicles" дозволяє зв'язувати параметри транспортного засобу з його ідентифікатором та іншою інформацією про транспортний засіб.

Для зберігання результатів попереднього аналізу вхідних знімків для програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками створено такі таблиці в базі даних:

– models (моделі НМ);

– clafssification_result_models (Класифікація визначеною моделлю);

– classification_Parameters (Результати класифікації);

– clafssification_result_models (Класифікація визначеною моделлю);

1. Структура таблиці models (Моделі):

– model_id: унікальний ідентифікатор;

– path: шлях до моделі;

– date: дата та час створення моделі;

2. Структура таблиці vehicle_images (Зображення техніки):

– vehicle_images_id: унікальний ідентифікатор знімку;

– path: шлях до знімку;

– date: дата та час зйомки знімку;

4. Таблиця "Результат класифікації" (Clafssification_Parameters):

– param_id: унікальний ідентифікатор характеристики;

– result_id: ID об'єкту, до якого належить характеристика – зв'язок з таблицею об'єктів;

id_vehicle – ідентифікатор транспортного засобу (зовнішній ключ, що посилається на таблицю "vehicles").

– vehicle_propability: ступінь відношення до техніки.

4. Таблиця "Класифікація за моделлю" (clafssification_result_models):

– param_id: унікальний ідентифікатор характеристики;

– result_id: ID об'єкту, до якого належить характеристика – зв'язок з таблицею об'єктів;

Ці таблиці дозволяють зберігати інформацію про кожний знімок, об'єкти, що на ньому знаходяться, та їх характеристики після розпізнавання (рис. 2.12).

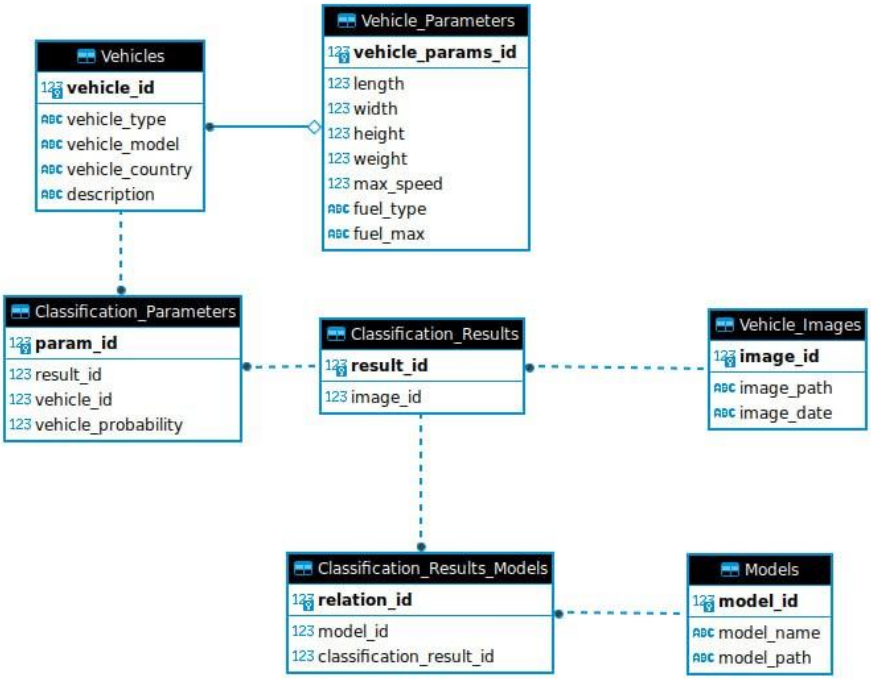

Рисунок 2.12 – Фізична модель БД

2.3 Проектування інтерфейсу програмного модуля інтелектуального розпізнавання військової техніки

Для проектування інтерфейсу визначено основні вікна програмного модуля інтелектуального розпізнавання військової техніки і розроблено для них прототипи:

– вікно завантаження супутникових знімків (рис. 2.13);

– вікно результатів розпізнавання супутникових знімків (рис. 2.14);

– вікно додавання класів військової техніки (рис. 2.15).

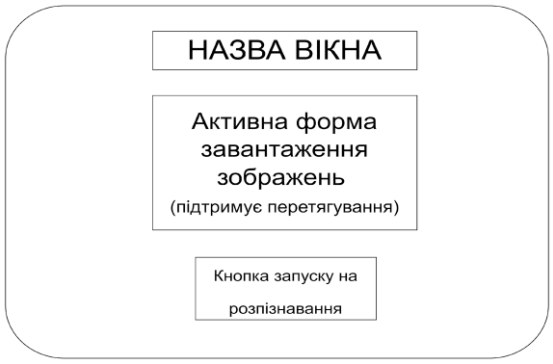

Рисунок 2.13 – Прототип вікна завантаження супутникових знімків

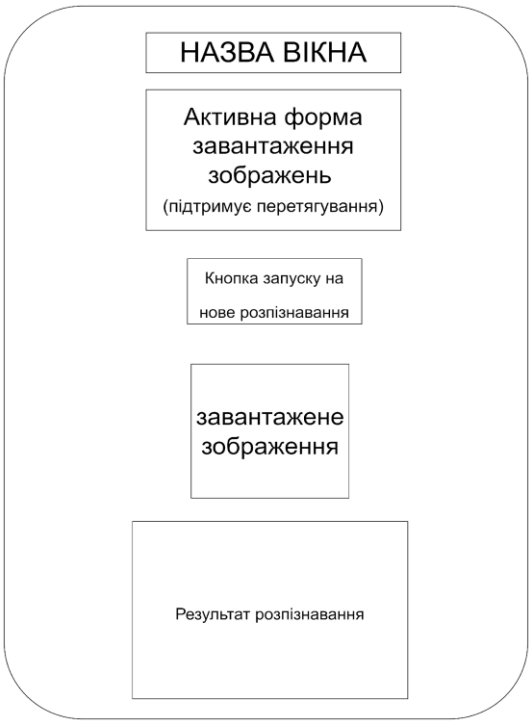

Рисунок 2.14 – Прототип вікна результатів розпізнавання супутникових знімків

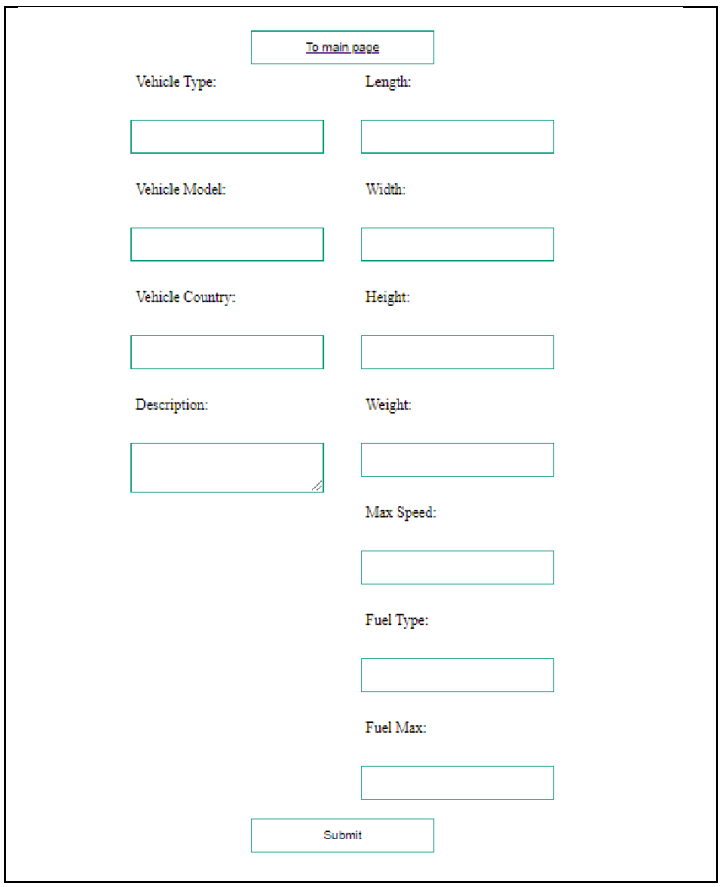

Рисунок 2.15 – Прототип вікна додавання класів військової техніки

2.4 Висновки до другого розділу

В результаті проведення етапу розробки інтелектуальної модуля розпізнавання військової техніки було проведено функціональний аналіз та побудовано дерево функцій, яке відображає бізнес-процеси інформаційної системи. Також було розроблено діаграми Event-Driven Process Chain. На даних схемах було спроектовано та описано логіку роботи боту.

Також було представлено діаграми IDEF0 "ЯК БУДЕ", які показують процеси видачі висновку за зображеннями.

Було сформовано бізнес-архітектуру модуля з описанням його внутрішніх компонентів та їх взаємозв’язків, а також розроблено базу даних для роботи програмного модуля.

РОЗДІЛ 3 РЕАЛІЗАЦІЯ, ТЕСТОВІ ПРИКЛАДИ

3.1 Вибір програмного інструментарію для розробки програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками

Для розробки модуля використано ряд інструментів та методів, за допомогою яких створено елементи штучного інтелекту для навчання та виявлення техніки на знімках.

Основна мова програмування Python з використанням бібліотек: matplotlib, NumPy, seaborn, Scikit-learn, TensorFlow та SklearnMetrics. [26].

Python (найчастіше вживане прочитання — «Па́йтон», запозичено назву з британського шоу Монті Пайтон) — інтерпретована об'єктно-орієнтована мова програмування високого рівня зі строгою динамічною типізацією. Розроблена в 1990 році Гвідо ван Россумом. Структури даних високого рівня разом із динамічною семантикою та динамічним зв'язуванням роблять її привабливою для швидкої розробки програм, а також як засіб поєднування наявних компонентів. Python підтримує модулі та пакети модулів, що сприяє модульності та повторному використанню коду. Інтерпретатор Python та стандартні бібліотеки доступні як у скомпільованій, так і у вихідній формі на всіх основних платформах. В мові програмування Python підтримується кілька парадигм програмування, зокрема: об'єктно-орієнтована, процедурна, функціональна та аспектно-орієнтована.

Scikit-learn (також відома як sklearn або scikits.learn) — це безкоштовна програмна бібліотека машинного навчання для мови програмування Python, яка надає функціональність для створення та тренування різноманітних алгоритмів класифікації, регресії та кластеризації, таких як лінійна регресія, random forest, градієнтний бустинг, і працює у зв'язці з бібліотеками NumPy та SciPy. Scikit-learn є однією з найбільш популярних бібліотек машинного навчання.

Matplotlib — бібліотека на мові програмування Python для візуалізації даних двовимірною 2D графікою (3D графіка також підтримується).

Отримувані зображення можуть бути використані як ілюстрації в публікаціях. Matplotlib написана і підтримується в основному Джоном Хантером і поширюється на умовах BSD-подібної ліцензії. Зображення, які генеруються в різних форматах, можуть бути використані в інтерактивній графіці, наукових публікаціях, графічному інтерфейсі користувача, веб-додатках, де потрібно будувати діаграми. В документації автор зізнається, що Matplotlib починався з імітування графічних команд MATLAB, але є незалежним від нього проектом.

Бібліотека Matplotlib побудована за принципами ООП, але має процедурний інтерфейс pylab, який надає аналоги команд MATLAB.

NumPy (скорочено від Numerical Python) — бібліотека з відкритим кодом для мови програмування Python. Можливості:

– підтримка багатовимірних масивів (включаючи матриці);

– підтримка високорівневих математичних функцій, призначених до роботи з багатовимірними масивами.

Для підготовки середовища на комп'ютері для розробки програмного модуля з використанням описаних бібліотек необхідно виконати наступні кроки:

1. Встановити Python: завантажити встановлювач Python з офіційного сайту (https://www.python.org/downloads/). Оберти версію, яка підходить для операційної системи, і запустити встановлювач. Відзначити опцію "Add Python to PATH" під час встановлення, щоб забезпечити доступ до Python з командного рядка.

2. Встановити бібліотеки: Після встановлення Python встановити необхідні бібліотеки за допомогою пакетного менеджера pip через командний рядок:

pip install matplotlib numpy seaborn scikit-learn tensorflow sklearn-metrics

3. Налаштувати робоче середовище: після встановлення бібліотек відкрити текстовий редактор або інтегровану середу розробки (IDE). На кожній платформі можуть бути свої особливості при встановленні та налаштуванні середовища розробки.

Після підготовки середовища розробки та встановлення необхідних бібліотек необхідно поєднати різні компоненти (бібліотеки).

Для використання бібліотеки matplotlib, знадобиться імпортувати цю бібліотеку та використати функції для створення та відображення графіку.

```
import matplotlib.pyplot as plt
# створення даних
x = [1, 2, 3, 4, 5]
y = [2, 4, 6, 8, 10]
# створення графіку
plt.plot(x, y)
# відображення графіку
plt.show()
```

У цьому коді імпортуємо бібліотеку matplotlib.pyplot та використовуємо функції plot() та show() для створення та відображення графіку.

Для використання машинного навчання для виявлення військової техніки, знадобиться імпортувати бібліотеки Scikit-learn, Tensorflow та SklearnMetrics, та використовувати їх функції та методи для побудови та тренування моделей машинного навчання.

При програмуванні з використанням бібліотек використовується модульний підхід, коли функції та класи розміщуються в окремих файлах. Тому знадобиться імпортувати відповідний модуль, щоб використовувати функції та класи, що в ньому містяться.

Для файлу my_module.py з функцією my_function() необхідно імпортувати цей модуль та викликати функцію my_function() в своєму коді наступним чином: import my_module my_module.my_function()

Також можна імпортувати лише певну функцію з модулю:

from my_module import my_function my_function()

Для імпорту модуля з іншою назвою необхідно використати ключове слово as: import my_module as mm mm.my_function()

3.2 Розроблені алгоритми для підтримки роботи модуля

Основною задачею було навчити нейронну мережу, для чого використано Keras, OpenCV та TensorFlow. Схема алгоритму навчання представлена на рис. 3.1.

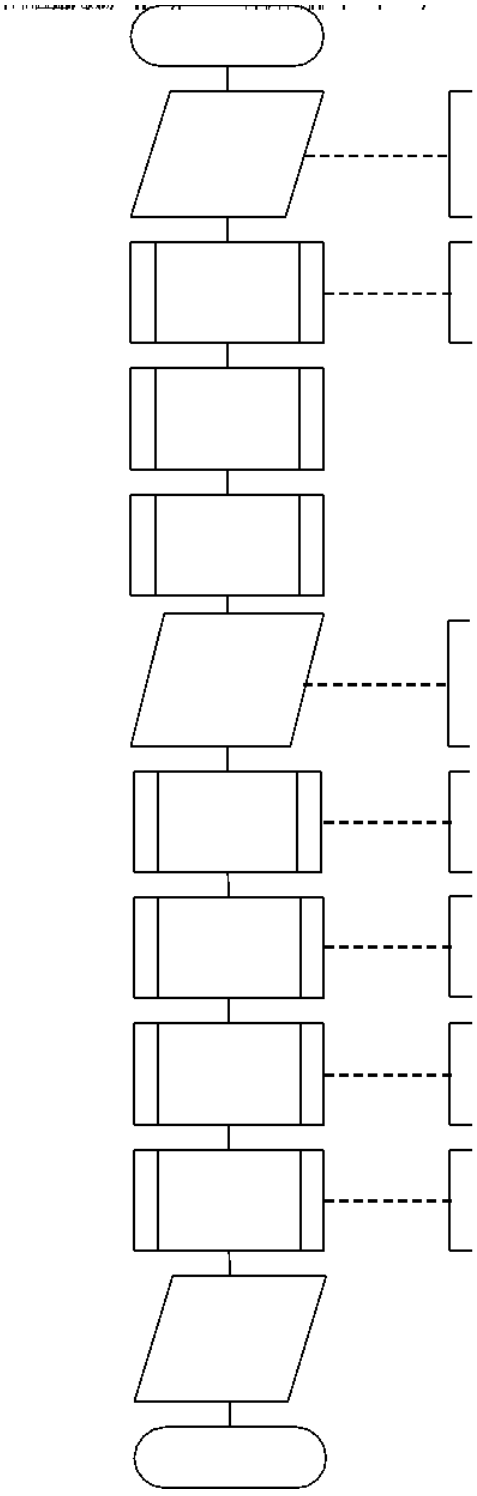

Рисунок 3.1 – Схема алгоритму попередньої обробки і тренування моделі

Для обробки зображень розроблено алгоритм попереднього аналізу зображень з можливою корекцією або відмовою у прийнятті зображення до аналізу (рис. 3.2).

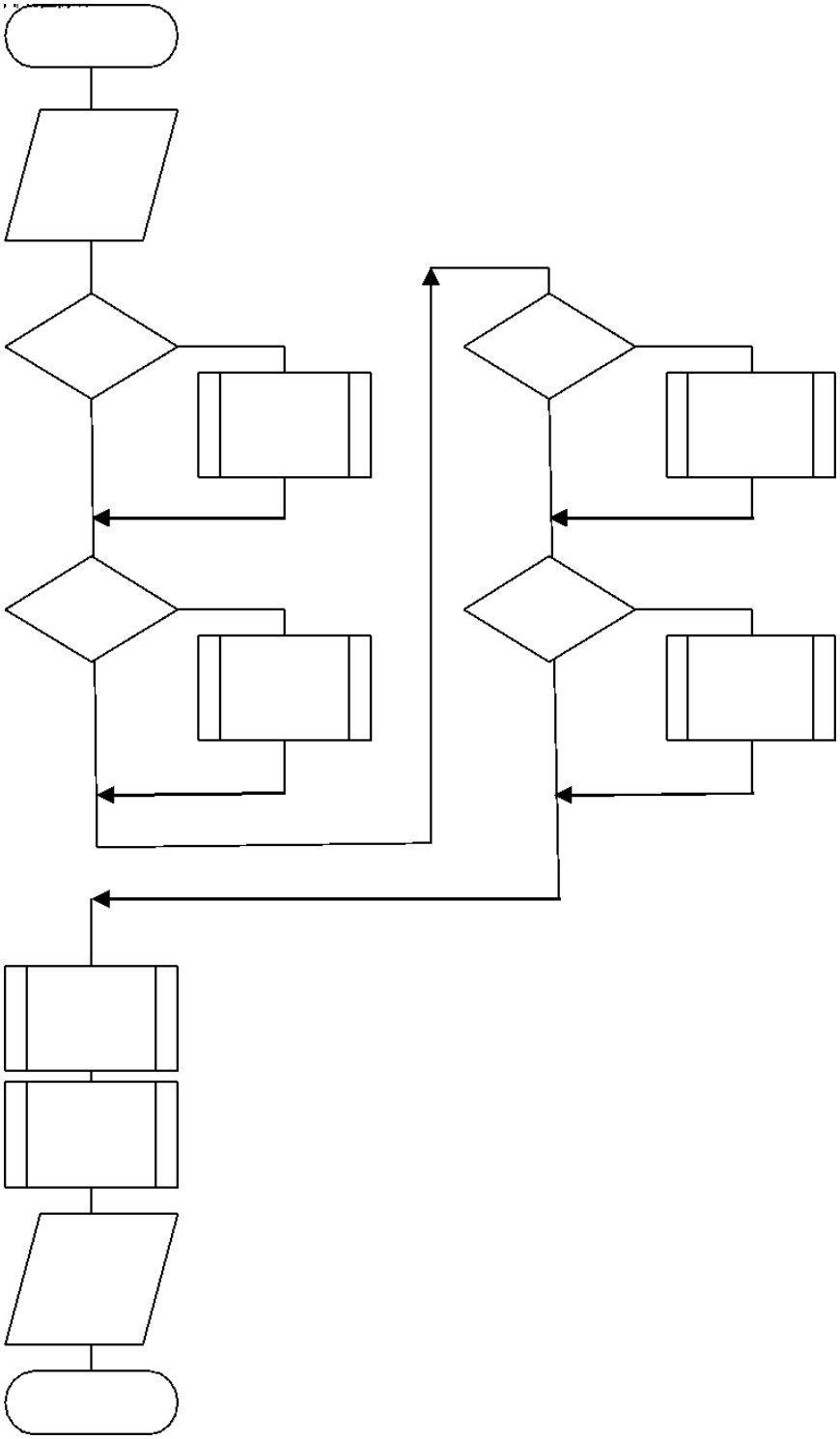

Рисунок 3.2 – Схема алгоритму роботи основного вікна програмного модуля визначення воєнної техніки на зображеннях

3.3 Навчання нейронної мережі

Першим кроком є завантаження дата сету (рис. 3.3) за класами техніки.

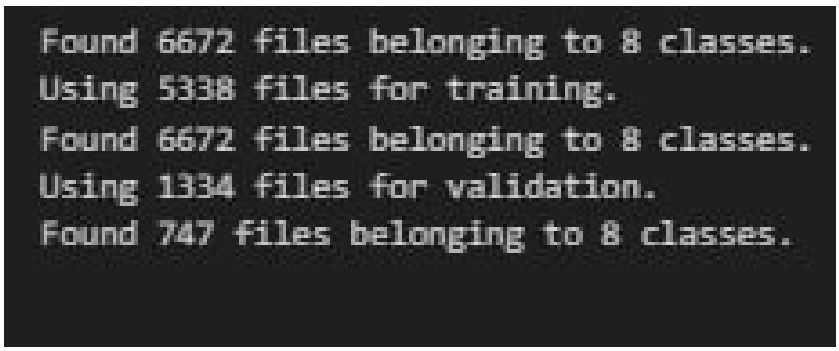

Рисунок 3.3 – Завантаження дата сету техніки

Виділено 10% вибірки як тестову (це 747 зображення, які довільним чином вибрані з вихідних 7419). Решту 90% (6672 зображень) розділено на: тренувальну вибірку та валідаційну вибірку для крос-валідації. Пропорції 72% (5338 зображень) та 18% (1334 зображень). Після чого програма класифікує військову техніку за типом. На рис. 3.4 можна побачити (зліва на право) супутникові знімки наступної техніки: танк Т-62, артилерійську установку 2с1, спец. техніка Д7, вантажівка ЗІЛ131, спец. техніка Д7, БТР-60, артилерійську установку 2с1. Код представлено в Додатку А.

3.3.1 Архітектура нейронної мережі

На основі відкритої моделі Keras проведено налаштування архітектури нейронної мережі для розпізнавання зображень (рис. 3.4).

За документацією схожої системи розпізнавання об’єктів на зображеннях [26] було підібрано наступну архітектуру:

1. Inputlayer: початковий шар мережі, який визначає формат та розмір вхідних зображень;

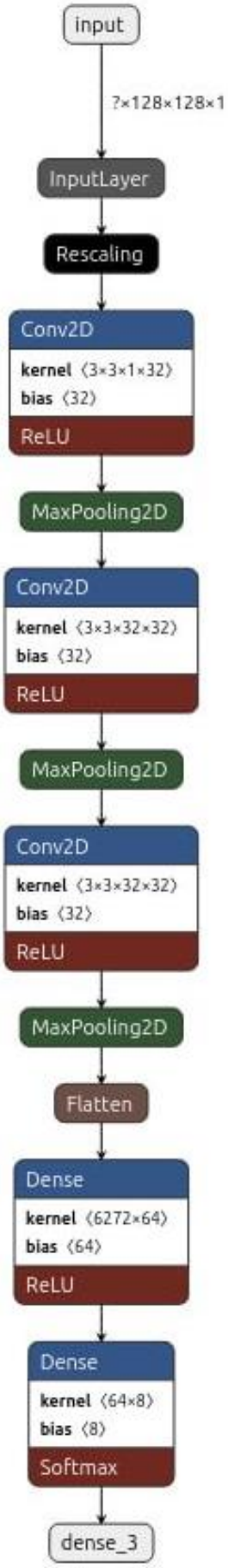

Рисунок 3.4 – Архітектура НМ для розпізнавання зображень

2. Rescaling: шар, який виконує масштабування вхідних зображень для приведення їх значень до певного діапазону;

3. Conv2D: шар виконує згортку на вхідних зображеннях з використанням фільтрів (ядер) розміром 3x3 з одним вхідним каналом і 32 вихідними каналами. Кожен фільтр проходить по зображенню і виконує обчислення, що допомагають виявити різні ознаки та шаблони;

4. MaxPooling2D: шар виконує пулінг (зменшення розміру) на виході попереднього згорткового шару. В даній архітектурі використовується максимальне значення з певної області для стиснення інформації та забезпечення інваріантності до малих зміщень та змін масштабу;

5. Conv2D: шар є аналогічним до попереднього згорткового шару, але з використанням 32 вхідних та 32 вихідних каналів. Він допомагає виявити більш складні ознаки та шаблони на зображенні;

6. MaxPooling2D: шар є аналогічним до попереднього пулінгового шару і здійснює подальше зменшення розміру вихідних даних;

7. Conv2D: шар є аналогічним до попередніх згорткових шарів, але знову використовує 32 вхідних та 32 вихідних канали і виконує згортку на виході попередніх шарів, щоб виявити ще більш складні ознаки та шаблони;

8. MaxPooling2D: шар є аналогічним до попереднього пулінгового шару;

9. Flatten: шар перетворює вихідні дані з попереднього шару в одновимірний вектор, який може бути використаний як вхід для повністю зв'язаного шару;

10. Dense: шар є повністю зв'язаним шаром з 64 нейронами. Використовує функцію активації ReLU. Його роль полягає в отриманні важливих ознак з попередніх шарів та виконанні подальшого аналізу;

11. Dense: шар є повністю зв'язаним шаром з 8 нейронами. Використовує функцію активації Softmax. Використовується для класифікації зображень та вироблення остаточних рішень щодо класу, до якого належить вхідне зображення.

В системі налаштувань використовувались наступні змінні для налаштування:

– rescaling_1 (Rescaling): цей параметр вказує на кількість нейронів у шарі Rescaling, який відповідає за масштабування вхідних зображень (значення 0 означає, що в цьому шарі використовується нульова кількість нейронів. Це може бути обумовлено тим, що Rescaling не вимагає обчислення нейронів, а лише здійснює операцію масштабування зображень);

–. conv2d_3 (Conv2D): цей параметр вказує на кількість нейронів у шарі Conv2D, який відповідає за згорткову операцію на зображеннях. Кількість нейронів у Conv2D визначається архітектурою мережі і може бути налаштована для досягнення бажаних результатів;

– max_pooling2d_3 (MaxPooling2D): цей параметр вказує на кількість нейронів у шарі MaxPooling2D, який відповідає за операцію максимального пулінгу на зображеннях. MaxPooling2D не має нейронів, а лише виконує операцію пулінгу для зменшення розміру зображень.

– conv2d_4 (Conv2D): параметр вказує на кількість нейронів у шарі Conv2D, який виконує згортку на зображеннях;

– max_pooling2d_4 (MaxPooling2D): параметр вказує на кількість нейронів у шарі MaxPooling2D, який виконує пулінг на зображеннях (значення 0 означає, що в цьому шарі не використовуються нейрони, а лише виконується операція пулінгу);

– conv2d_5 (Conv2D): параметр вказує на кількість нейронів у другому шарі Conv2D, який також виконує згортку на зображеннях;

– max_pooling2d_5 (MaxPooling2D): параметр вказує на кількість нейронів у другому шарі MaxPooling2D, який виконує пулінг на зображеннях;

– flatten_1 (Flatten): параметр вказує на кількість нейронів у шарі Flatten, який перетворює вихідні дані з попередніх шарів в одновимірний вектор;

– dense_2 (Dense): параметр вказує на кількість нейронів у шарі Dense, який є повністю зв'язаним шаром мережі. Значення 200736 вказує на кількість нейронів у цьому шарі.

3.3.2 Підібір гіперпарматрів нейронної мережі

Для того, щоб підібрати гіперпарматри НМ, використано функцію GridSearch, класу Tuner, з бібліотеки Keras.

Гіперпараметри - це налаштування моделі, які не вивчаються під час процесу навчання, але впливають на її ефективність і поведінку.

GridSearch - це метод пошуку гіперпараметрів, який перебирає всі можливі комбінації значень гіперпараметрів заздалегідь визначеного простору. Це означає, що ви можете визначити список значень для кожного гіперпараметра, і GridSearch виконає навчання та оцінку моделі для кожної комбінації цих значень. Після завершення пошуку ви отримаєте набір результатів, що допоможуть визначити оптимальні значення гіперпараметрів.

Клас Tuner в бібліотеці Keras надає зручний і автоматизований інтерфейс для підбору гіперпараметрів моделі. Він дозволяє визначити простір пошуку гіперпараметрів, включаючи можливі значення та типи параметрів. Tuner автоматично перебирає комбінації значень та навчає модель для кожної комбінації, зберігаючи і оцінюючи результати. Після завершення пошуку ви можете отримати найкращі значення гіперпараметрів та використати їх для побудови та навчання остаточної моделі.

Використання функції GridSearch та класу Tuner дозволяє систематично експериментувати з різними значеннями гіперпараметрів та знаходити оптимальні комбінації для моделі нейронної мережі.

Налаштовуються наступні гіперпараметри:

– 'conv_blocks': це кількість згорткових блоків у моделі нейронної мережі. Згорткові блоки використовуються для виявлення різних ознак та шарів зображень.

– 'filters_0', 'filters_1', 'filters_2': це кількість фільтрів у кожному згортковому блоку. Фільтри використовуються для виділення різних ознак зображень на кожному рівні.

– 'dense_units': це кількість нейронів у повністю з'єднаному шарі (dense layer) моделі. Повністю з'єднаний шар відповідає за формування рішень на основі ознак, виявлених у попередніх шарах.

– 'learning_rate': це швидкість навчання моделі, яка визначає, наскільки швидко модель адаптується до навчальних даних. Цей параметр керує оновленням ваг моделі під час процесу навчання.

В результаті експериментів за допомогою GridSearch та Tuner були знайдені оптимальні значення для цих гіперпараметрів. Найкращою комбінацією для моделі є 3 згорткових блоки з 32 фільтрами у кожному блоку, 32 нейрони у повністю з'єднаному шарі та швидкість навчання 0.0019952623149688 81.

У налаштуваннях конфігураційних файлів за ці параметри відповідають наступні змінні:

– 'conv_blocks': 3;

– 'filters_0': 32;

– 'filters_1': 32;

– 'filters_2': 32;

– 'dense_units': 32;

– 'learning_rate': 0.0019952623149688 81.

3.3.3 Використання методу перехресного затвердження

Для навчання мережі було використано підхід перехресного затвердження (cross-validation) – метод K-Fold бібліотеки Sklearn (рис. 3.5): Перехресне затвердження методом K-Fold є одним з найпоширеніших методів оцінки ефективності моделей машинного навчання. Використовуючи цей метод, доступний набір даних розбивається на K рівних частин, що називаються фолдами.

Рисунок 3.5 – Візуалізація методу перехресного затвердження (cross-validation)

Кожен фолд використовується як тренувальний набір даних, тоді як залишаючи один фолд для валідації. Процедура навчання та оцінки моделі повторюється K разів, кожного разу використовуючи інший фолд як валідаційний набір.

Перехресне затвердження дозволяє отримати більш надійну оцінку ефективності моделі, оскільки кожен фолд використовується як тренувальний і валідаційний набір даних. Це допомагає уникнути проблем, пов'язаних з випадковим вибором тренувального та валідаційного наборів. Крім того, перехресне затвердження дозволяє використовувати всі дані для тренування та валідації моделі, що робить оцінку більш об'єктивною.

Метод K-Fold полягає у тому, що доступний набір даних розбивається на K рівних частин. Зазвичай значення K обирають 5 або 10, але це може варіюватися залежно від конкретної задачі. Потім для кожної ітерації модель навчається на K-1 фолдах, а валідація проводиться на залишковому фолді. Результати кожної ітерації об'єднуються, і зазвичай використовується середнє значення метрик оцінки, щоб отримати оцінку ефективності моделі.

Бібліотека Sklearn (scikit-learn) є потужним інструментарієм для машинного навчання в мові програмування Python. Вона надає багато функцій та методів для побудови, навчання та оцінки моделей машинного навчання. Sklearn також має вбудовану підтримку перехресного затвердження з використанням методу K-Fold.

В результаті отримано наступні параметри точності:

– точність валідації для 1334 зображень – 0.987;

– середня точність валідації (кратність 5) – 0.986.

Таким чином для супутникових знімків, приклад яких представлено на рис. 3.6 налаштовано НМ з гіпермараметрами, які представлено на рис. 3.7.

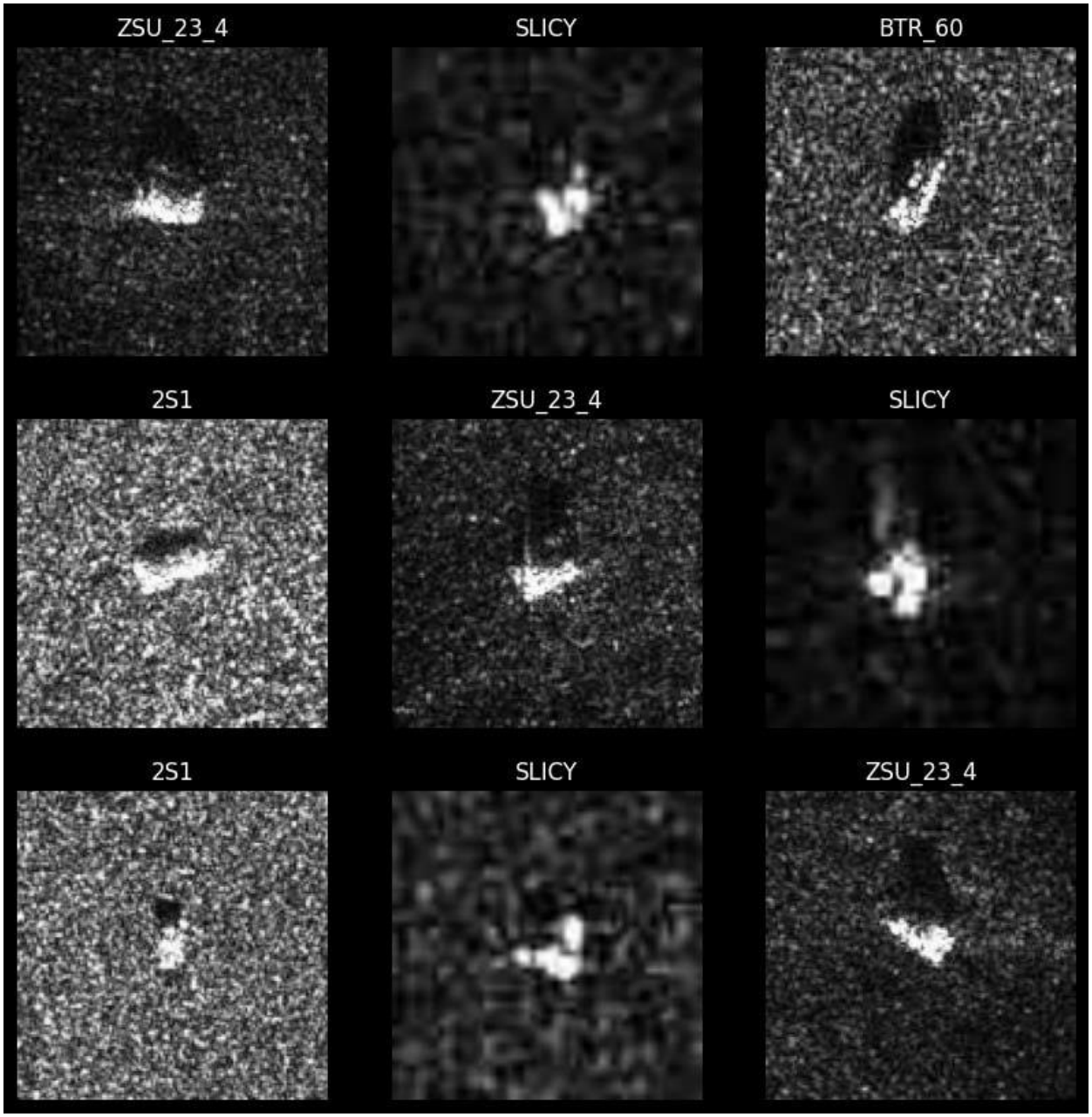

Рисунок 3.6 – Приклади супутникових знімків військової техніки

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 rescaling_1 (Rescaling)     (None, 128, 128, 1)       0

 conv2d_3 (Conv2D)           (None, 126, 126, 32)      320

 max_pooling2d_3 (MaxPooling  (None, 63, 63, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 61, 61, 32)        9248

 max_pooling2d_4 (MaxPooling  (None, 30, 30, 32)       0
 2D)

 conv2d_5 (Conv2D)           (None, 28, 28, 32)        9248

 max_pooling2d_5 (MaxPooling  (None, 14, 14, 32)       0
 2D)

 flatten_1 (Flatten)         (None, 6272)              0

 dense_2 (Dense)             (None, 64)                401472

 dense_3 (Dense)             (None, 8)                 520

=================================================================
Total params: 420,808
Trainable params: 420,808
Non-trainable params: 0
_________________________________________________________________
```

Рисунок 3.7 – Налаштування моделі НМ

Результатом навчання моделі було отримання оптимальних важелів і параметрів, які дозволяють моделі ефективно розпізнавати військову техніку на основі вхідних супутникових знімків. Це означає, що модель набуває здатності класифікувати зображення військової техніки з високою точністю і надійністю.

3.3.4 Оцінка результатів навчання нейронної мережі

Результати навчання моделі було оцінено в наступних метриках: точність класифікації, MCC (Matthews Correlation Coefficient), F-міра (F-Measure) та інші.

Ці метрики оцінюють продуктивність моделі і демонструють її здатність до правильної класифікації військової техніки на зображеннях (рис. 3.8).

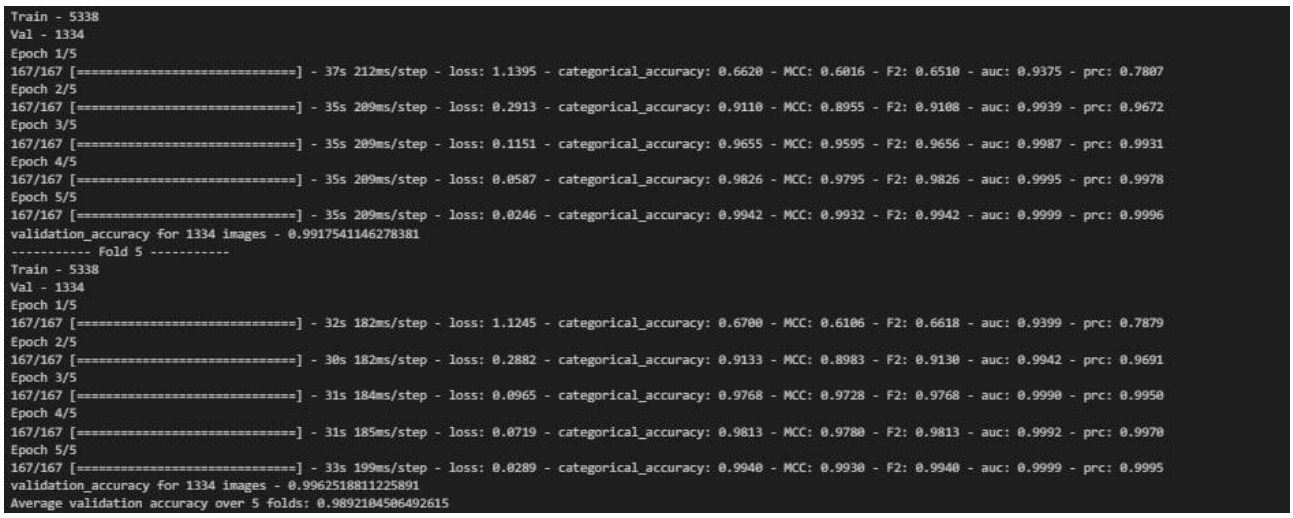

Рисунок 3.8 – Інформація про навчену нейронну мережу

Наведемо описання метрик:

– Loss (втрати) – це значення втрат або помилок моделі під час навчання. Чим менше значення втрат, тим краще модель працює на навчальних даних.

– Categorical Accuracy (категоріальна точність) – ця метрика вимірює точність класифікації моделі для категоріальних (багатокласових) задач. Вона показує відсоток правильно класифікованих прикладів від загальної кількості прикладів.

– MCC (Matthews Correlation Coefficient) – це метрика, яка оцінює якість класифікації, особливо в контексті незбалансованих даних. Вона залежить від значень true positive, false negative, false positive та true negative, і вона набуває значення в діапазоні [-1, 1], де 1 вказує на ідеальну класифікацію, 0 - на випадкову, а -1 - на найгіршу можливу.

– F2 – це метрика, яка оцінює точність моделі, зосереджуючись на розпізнаванні позитивних класів. Вона дає більший ваговий коефіцієнт для виявлення позитивних зразків, тому є корисною, коли важливо зменшити кількість хибних негативів.

– Auc (Area Under the Curve) – це метрика, яка вимірює якість моделі для задачі бінарної класифікації. Вона представляє площу під кривою ROC (Receiver Operating Characteristic) і вказує на здатність моделі правильно розподіляти приклади між класами.

– PRC (Precision-Recall Curve) – це метрика, яка оцінює точність та повноту моделі для задачі бінарної класифікації. Вона представляє криву точність-повнота і допомагає визначити оптимальний поріг розпізнавання для моделі.

– метрики з префіксом "val_", які відносяться до валідаційних даних. Вони вимірюють ті ж самі метрики, але на валідаційному наборі даних, що допомагає оцінити загальну здатність моделі до узагальнення на нові дані.

Зведемо на графіку (рис. 3.9) метрики loss та auc для тестевих і валідаціних зображень.

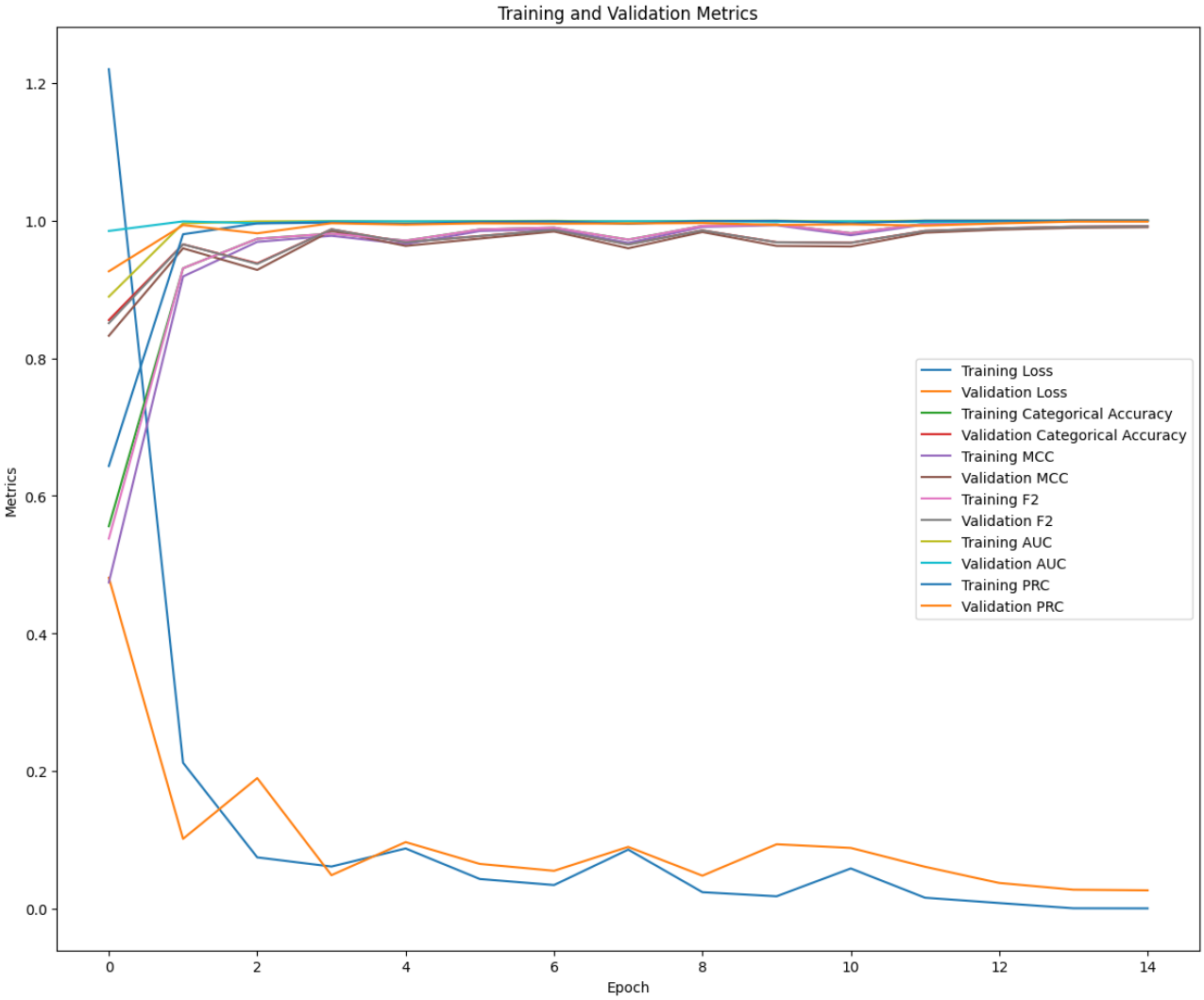

Рисунок 3.9 – Метрики навчання моделі: тестові і валідаційні зображення

Після навчання нейронної мережі програма звітує графіком та частот вдалої класифікації технік по основній діагоналі (рис. 3.10).

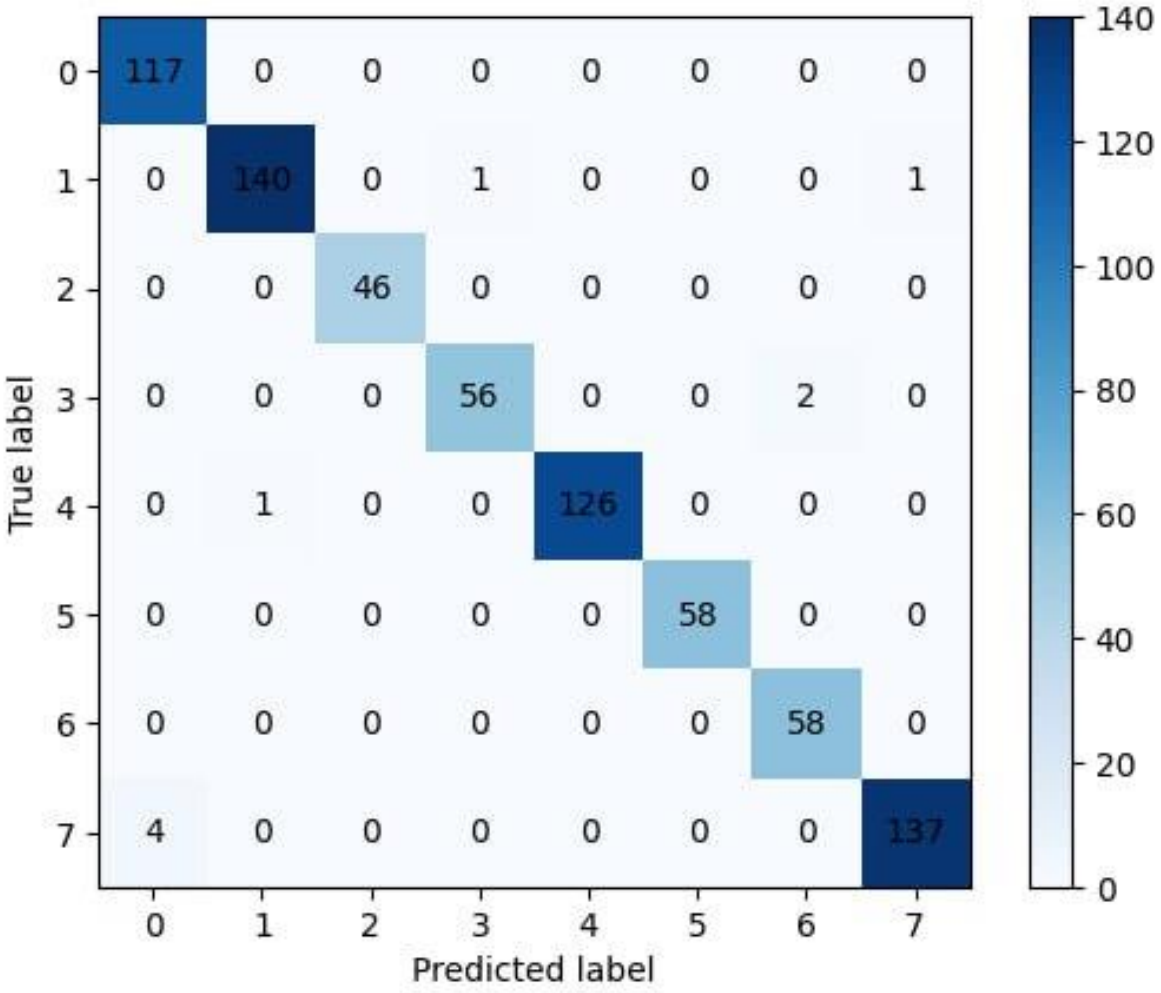

Рисунок 3.10 – Інформація про вірну класифікацію техніки

Після підрахунку, програма видає метрики нашої моделі на данні які ми ніколи не бачили, точність нашої моделі становить приблизно 98% з похибкою менше 10% (рис. 3.11).

```
24/24 [==============================] - 1s 47ms/step - loss: 0.0541 - categorical_accuracy: 0.9880 - MCC: 0.9859 - F2: 0.9879 - auc: 0.9984 - prc: 0.9970

{'loss': 0.0540560409426689l5,
 'categorical_accuracy': 0.9879518151283264,
 'MCC': 0.9859123229980469,
 'F2': 0.9879298806190491,
 'auc': 0.9984369277954102,
 'prc': 0.9969944357872009}
```

Рисунок 3.11 – Результат сканування

При навчанні моделі постійно проводився контроль того, щоб не з’явився феномен «перенавчання», який є явищем у машинному навчанні, коли модель надмірно адаптується до навчальних даних і стає недостатньо загальним для

використання на нових, невідомих даних. В результаті перенавчання модель "запам'ятовує" навчальні дані, включаючи шум і неправильності, і не вміє правильно узагальнювати для нових прикладів.

Перевірялись наступні ознаки перенавчання:

– дуже висока точність на навчальних даних, але низьку точність на тестових або невідомих даних;

– велику різницю між результатами навчання та результатами валідації або тестування;

– занадто складені або "розряжені" моделі з великою кількістю параметрів, що можуть "запам'ятовувати" навчальні дані.

3.4. Інтерфейс Користувача програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками

Для зручності використання системи було розроблено інтерфейс завантаження зображень та виведення результатів пошук (рис. 3.12).

Рисунок 3.12 – Інтерфейс завантаження зображення

В результаті роботи нейронної мережі отримується найбільш відповідна техніка з бази (рис. 3.13). В результаті роботи нейронної мережі отримується класифікація зображення військової техніки на основі навчених важелів і

параметрів моделі. Ця класифікація вказує на найбільш відповідну техніку з бази, до якої належить зображення, та видає коротку характеристику цієї техніки з бази даних.

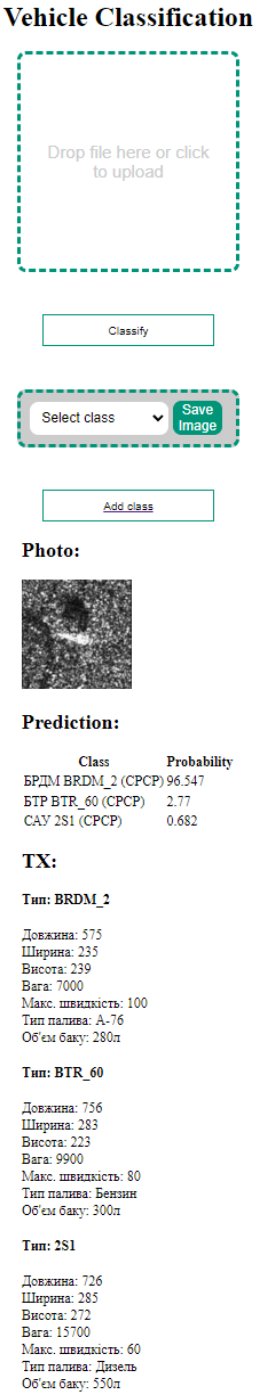

Рисунок 3.13 – Інтерфейс виведення результату розпізнавання військової техніки на зображеннях

Також була дана можливість додавання нового класу техніки та визначення його параметрів.

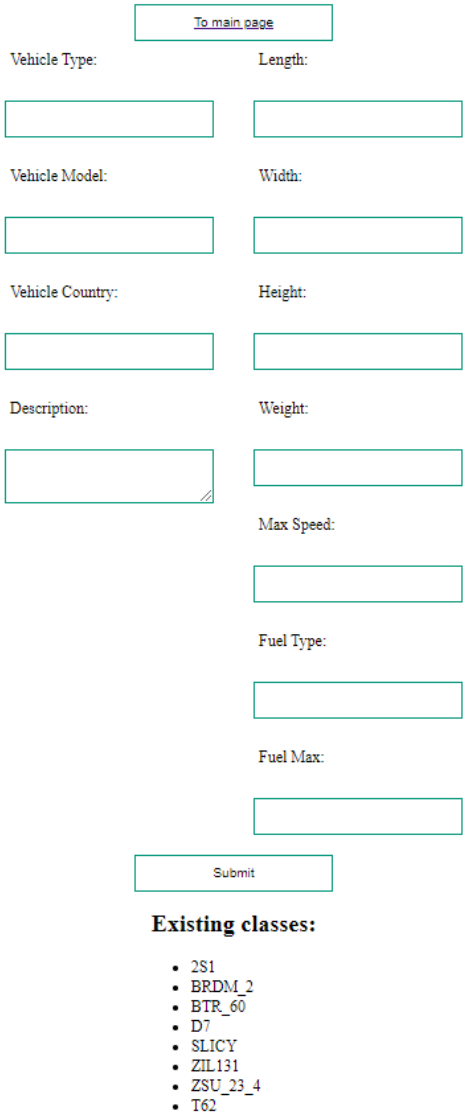

Рисунок 3.14 – Інтерфейс додавання нового класу військової техніки

У роботі можуть бути використані різні класи військової техніки, такі як танки, бронетранспортери, військові літаки тощо. Нейронна мережа навчається розпізнавати ці класи на основі тренувального набору зображень, де кожне зображення має відповідну мітку класу.

Після навчання моделі та застосування її до нового зображення військової техніки, нейронна мережа видає прогнозований клас, який найбільше відповідає зображенню. Наприклад, якщо вхідне зображення представляє танк, модель може класифікувати його як "танк" і надати відповідний результат.

Отримання найбільш відповідної техніки з бази є результатом процесу класифікації, де модель визначає найбільш ймовірний клас військової техніки

для заданого зображення. Цей результат може бути використаний для подальшого аналізу, ідентифікації та прийняття рішень в контексті військових та розвідувальних операцій.

Весь процес завантаження та розпізнавання зберігається в базі даних, структура якої детально розглядалась у Розділі 2.3.4.

На рис. 3.15 представлено приклади збережених даних в таблицях (використовується SQLite).

Vehicles — Enter a SQL expression to filter results (use Ctrl+Space)

	vehicle id	vehicle type	vehicle model	vehicle country	description
1	1	САУ	2S1	СРСР	Радянська 122-мм полкова самохідна артилерійсь
2	2	БРДМ	BRDM_2	СРСР	Основна бойова броньована машина розвідувальн
3	3	БТР	BTR_60	СРСР	Радянський бронетранспортер (БТР).¶¶БТР-60П ро
4	4	Трактор	D7	США	Середній гусеничний трактор виробництва Caterp
5	5	Танк	T62	СРСР	Т-62 - радянський середній танк, вперше предста
6	6	Вантажівка	ZIL131	СРСР	ЗІЛ-131 - 3,5-тонна армійська вантажівка загальн
7	7	ЗСУ	ZSU_23_4	СРСР	ЗСУ-23-4 "Шилка" - легкоброньована радянська са
8	8	Будівля	SLICY	-	SLICY ціль - це кілька простих геометричних форм

А)

Vehicle_Parameters — Enter a SQL expression to filter results (use Ctrl+Space)

	vehicle params id	length	width	height	weight	max speed	fuel type	fuel max
1	1	726	285	272	15,700	60	Дизель	550л
2	2	575	235	239	7,000	100	А-76	280л
3	3	756	283	223	9,900	80	Бензин	300л
4	4	410	97	96	14,460	30	Дизель	100л
5	5	934	330	240	37,000	50	Дизель	960л
6	6	704	250	248	6,700	80	Бензин	90л
7	7	653	312	2,576	19,000	50	Дизель	515л
8	8	600	900	400	5,800	0	Немає	0л

Б)

Vehicle_Images — Enter a SQL expression to filter results (use Ctrl+Space)

	image id	image path	image date
1	1	pred_images/image1.png	12/06/2023 16:12:27
2	2	pred_images/image2.png	12/06/2023 16:15:23
3	3	pred_images/image3.png	12/06/2023 16:15:50
4	4	pred_images/image4.png	12/06/2023 16:16:05
5	5	pred_images/image5.png	12/06/2023 16:16:30

В)

Рисунок 3.15 – Приклад збережених даних: А) таблиця техніки, Б) таблиця параметрів техніки, В) таблиця завантажених зображень

3.5 Висновки до третього розділу

В розділі 3 представлено процес розробки програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками.

У розділі наведено вибір програмних і технічних компонентів для розробки модуля, а також описані алгоритми, що допомагають у роботі з модулем. Було побудовано алгоритми для розпізнавання образів і машинного навчання.

Під час вибору програмних і технічних компонентів для створення програмного модуля проведено аналіз і вибір необхідних бібліотек та фреймворків, які будуть використовуватися для розробки модуля, відповідно до завдань, що поставлені у роботі.

Було сформовано архітектуру нейронної мережі, підібрано гіперпараметри та для навчання мережі використано підхід перехресного затвердження (cross-validation).

В результаті отримано наступні параметри точності:

– точність валідації для 1334 зображень – 0.987;

– середня точність валідації (кратність 5) – 0.986.

Результати навчання нейронної мережі оцінено метриками: точність класифікації, MCC, F-міра.

Також було представлено режими роботи програмного модуля і представлено приклади роботи модуля при розпізнаванні зображень.

ВИСНОВКИ

В межах розробки програмного модуля для розпізнавання військової техніки було виконано декілька кроків для його реалізації.

В результаті проведення етапу розробки інтелектуальної модуля розпізнавання військової техніки було проведено функціональний аналіз та побудовано дерево функцій, яке відображає бізнес-процеси інформаційної системи. Також було розроблено діаграми Event-Driven Process Chain. На даних схемах було спроєктовано та описано логіку роботи боту.

Також було представлено діаграми IDEF0 "ЯК БУДЕ", які показують процеси видачі предмету завдяки інтелектуальній системі.

Було сформовано бізнес-архітектуру модуля з описанням його внутрішніх компонентів та їх взаємозв’язків, а також розроблено базу даних для роботи програмного модуля.

Було визначено методи машинного навчання для розпізнавання військової техніки, також визначено структуру програмного застосунку, мова програмування, бібліотеки.

На основі аналізу відомих принципів обробки зображень та наукових досліджень за подібними тематиками було визначено основні етапи розробки системи і методи, які використовуються на кожному з етапів.

Дослідження показало, що практика використання глибинних нейронних мереж у поєднанні з методами обробки зображень дає найкращі результати в розпізнаванні військової техніки на зображеннях. Важливою складовою процесу є також постобробка результатів, яка дозволяє покращити точність класифікації та визначення об'єктів на зображенні.

В розділі 3 представлено процес розробки програмного модуля інтелектуального розпізнавання військової техніки за супутниковими знімками. У розділі наведено вибір програмних і технічних компонентів для розробки модуля, а також описані алгоритми, що допомагають у роботі з модулем. Було побудовано алгоритми для розпізнавання образів і машинного навчання.

Під час вибору програмних і технічних компонентів для створення програмного модуля проведено аналіз і вибір необхідних бібліотек та фреймворків, які будуть використовуватися для розробки модуля, відповідно до завдань, що поставлені у роботі.

Було сформовано архітектуру нейронної мережі, підібрано гіперпараметри та для навчання мережі використано підхід перехресного затвердження (cross-validation).

В результаті отримано наступні параметри точності:

– точність валідації для 1334 зображень – 0.987;

– середня точність валідації (кратність 5) – 0.986.

Результати навчання нейронної мережі оцінено метриками: точність класифікації, MCC, F-міра.

Також було представлено режими роботи програмного модуля і представлено приклади роботи модуля при розпізнаванні зображень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Коваленко Н. П., Лисенко Ю. І. та інші. Застосування нейронних мереж у задачі автоматичної класифікації зображень // Матеріали II Міжнародної науково-технічної конференції "Проблеми та перспективи розвитку сучасної техніки і технологій". – Київ, 2017. – С. 111-114.
2. Родіонов О. С., Черкаський В. Ю. Методи обробки та аналізу супутникових знімків для задач військового застосування / Родіонов О. С., Черкаський В. Ю.// Військова техніка, 2019. – № 1 (21). – С. 43-49.
3. Березін, А. В., Гудков, О. М., Кіпріянов, І. В. Автоматизована система розпізнавання військової техніки на зображеннях / Березін, А. В., Гудков, О. М., Кіпріянов, І. В. // Збірник наукових праць Харківського національного університету імені В. Н. Каразіна. Серія: Радіофізика та електроніка, випуск 23. – ХНУ, 2018. – С. 122-131.
4. Абрамов С.В., Лупаленко О.В., Манжай О.В. Аналіз автоматизованих систем виявлення та розпізнавання об’єктів збройних сил Російської Федерації. – Scientific Collection «InterConf» 95. Scientific goals and purposes in XXI century. – 2022. – С. 893–905.
5. Mustafin R., Isakov I., Kussainov T. Recognition of military equipment on aerial photographs using convolutional neural networks. – International Journal of Engineering & Technology, 7(4.6). – 2018. – P. 29-33.
6. Ouyang W., Luo P., Zeng W., Zhong Z., Li, X. Convolutional neural networks for pedestrian detection: A survey. – IEEE Transactions on Circuits and Systems for Video Technology, 28(4). – 2018. – P. 969-982.
7. Chen L.C., Papandreou G., Kokkinos I., Murphy K., Yuille, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. – IEEE Transactions on Pattern Analysis and Machine Intelligence, 40(4). – 2018. – P. 834-848.
8. Li, P., Qi, Y., Zhang, S., Wang, C. Object detection based on deep learning: A review. – Journal of Sensors, 2017. – P. 1-13.

9. Железняк О.О. Космічна фотограмметрія / Железняк О.О., Чубко Л.С. – Київ: Міністество освіти і науки, молоді та спорту України, 2012. – 216 с.
10. Бурштинська Х. В. Аерокосмічні знімальні системи / Бурштинська Х.В., Станкевич С.А. – Львів: Львівської політехніки, 2010. – 292 с.
11. Байрак Г.Р., Муха Б.П. Дистанційні дослідження Землі: навчальний посібник. – Львів: Видавничий центр ЛНУ імені Івана Франка. – 2010. – 712 с.
12. Atiqur Rahman. Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation / Atiqur Rahman, Wang Y. // Режим доступу: https://www.cs.umanitoba.ca/~ywang/papers/isvc16.pdf.
13. Christian Szegedy. Rethinking the Inception Architecture for Computer Vision //Режим доступу: https://www.cv-foundation.org/openaccess/content _cvpr_2016 /papers/Szegedy_Rethinking_the_Inception_CVPR_2016.pdf.
14. Rayed Bin Wahed. Comparative Analysis between Inception-v3 and Other Learning Systems using Facial Expressions Detection // Режим доступу: http://dspace.bracu.ac.bd/xmlui/bitstream/handle/10361/6397/12201020%20 %26%2016141024_CSE.pdf.
15. Цифрова обробка зображень [Текст] : метод, рекомендації до викон. лаборатор. робіт для студ. спеціальності 7.05080302, 8.05080302 «Аудіо-, відео- та кінотехніка» усіх форм навчання / Уклад.: В. С. Лазебний, П. В. Попович. – К.: НТУУ «КПІ», 2016. – 73 с.
16. Kaiming He. Deep Residual Learning for Image Recognition / Xiangyu Zang, Shaoqing Ren, Jian Sun // Режим доступу: https://www.cvfoundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.
17. Nguyen H. Fast object detection framework based on mobilenetv2 architecture and enhanced feature pyramid. Journal of Theoretical and Applied Information Technology, 98 (05) – 2020. – P. 145-156.
18. Ojala T., Pietikainen M., Maenpaa T. Multiresolution gray-scale and rotation

invariant texture classification with local binary patterns. – IEEE Transactions on Pattern Analysis and Machine Intelligence, 24(7). – 2002. – P. 971-987.

19. Simard P.Y. A Best practices for convolutional neural networks applied to visual document analysis / Steinkraus D., J.C. Platt // Режим доступу: https://ieeexplore.ieee.org/document/1227801.
20. Matusugu M., Katsuhiko M., Yusuke M., Yuji K. Designing Convolutional Neural Network Architecture Using Genetic Algorithms. – International Journal of Advanced Network, Monitoring and Controls, Volume & Issue: Volume 6. – 2021 – P. 555–559.
21. Van den Oord A., Dieleman S., Schrauwen B. Factoring variations in natural images with deep Gaussian mixture models. – Advances in Neural Information Processing Systems, Vol. (4). – 2014. – P. 3518-3526.
22. Collobert R., Weston J. Deep learning via semi-supervised embedding. – Proceedings of the 25th International Conference on Machine Learning. ICML '08. – New York, NY, USA: ACM. – 2008. – P. 160–167.
23. DeepLearning 0.1. LISA Lab. Посилання: https://github.com/lisa-lab/DeepLearningTutorials
24. Habibi A.H., Elnaz J.H. Guide to convolutional neural networks: a practical application to traffic-sign detection and classification. – Cham: Springer International Publishing. – 2017. – 348 P.
25. Про введення воєнного стану в Україні : Указ Президента України від 24.02.2022 № 64/2022. Посилання: https://zakon.rada.gov.ua/laws/show/64/2022.
26. Офіційний сайт Python.org. Посилання: https://www.python.org/.
27. Military Vehicles Tracking. Посилання: https://github.com/Lin-Sinorodin/Military_Vehicles_Tracking.
28. Military Vehicles Image Recognition. Посилання: https://github.com/AlexandreSajus/Military-Vehicles-Image-Recognition://github.com/Lin-Sinorodin/Military_Vehicles_Tracking.

ДОДАТОКИ

Додаток А Лістинг програмного модулю розпізнавання військової техніки

```
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.preprocessing import image_dataset_from_directory
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import *
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix
import tensorflow_addons as tfa
from sklearn.metrics import matthews_corrcoef, fbeta_score, roc_auc_score,
precision_recall_curve, auc
import os
import shutil
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
import keras_tuner as kt
import seaborn as sns
from sklearn.metrics import confusion_matrix
import matplotlib.image as mpimg

directory = './mstar_images/'
# Вказуйте шлях до ваших зображень (абсолютний або відносний)
src_dir = './mstar_images/'

# Створюємо нові папки для розбиття на тестування/тренування/валідацію
os.makedirs('./test', exist_ok=True)
```

```
os.makedirs('./train', exist_ok=True)

class_names = os.listdir(src_dir)

for class_name in class_names:
    os.makedirs(f'./test/{class_name}', exist_ok=True)
    os.makedirs(f'./train/{class_name}', exist_ok=True)

    # Список всіх файлів у цій папці класу
    image_files = os.listdir(f'{src_dir}/{class_name}')

    # Випадково перемішуємо файли
    np.random.shuffle(image_files)

    # Виділяємо 10% вибірки як тестову
    train_val_files, test_files = train_test_split(image_files, test_size=0.10,
random_state=42)

    # Копіюємо файли до відповідних папок
    for file in test_files:
        shutil.copy(f'{src_dir}/{class_name}/{file}', f'./test/{class_name}/{file}')

    for file in train_val_files:
        shutil.copy(f'{src_dir}/{class_name}/{file}', f'./train/{class_name}/{file}')

image_size = (128, 128)
batch_size = 32

datagen = tf.keras.preprocessing.image.ImageDataGenerator(
    rescale=1./255,
```

```
    validation_split=0.2,
)

# data for GridSearch

train_ds = tf.keras.utils.image_dataset_from_directory('./train/',
                                                       image_size=image_size,
                                                       label_mode='categorical',
                                                       color_mode='grayscale',
                                                       batch_size=batch_size,
                                                       validation_split=0.2,
                                                       subset='training',
                                                       seed=10)

val_ds = tf.keras.utils.image_dataset_from_directory('./train/',
                                                     image_size=image_size,
                                                     label_mode='categorical',
                                                     color_mode='grayscale',
                                                     batch_size=batch_size,
                                                     validation_split=0.2,
                                                     subset='validation',
                                                     seed=10)

test_ds = tf.keras.utils.image_dataset_from_directory('./test/',
                                                     image_size=image_size,
                                                     label_mode='categorical',
                                                     color_mode='grayscale',
                                                     batch_size=batch_size,
                                                     seed=10)
params = {"ytick.color" : "w",
```

```
        "xtick.color" : "w",
        "axes.labelcolor" : "w",
        "axes.edgecolor" : "w",
        "figure.figsize" : (10,10),
        "axes.titlecolor" : 'w',
        "axes.facecolor" : 'w',
        "figure.facecolor" : 'k'}

colors = plt.rcParams['axes.prop_cycle'].by_key()['color']

%matplotlib inline

class_names = train_ds.class_names

with plt.rc_context(params):
    plt.figure(figsize=(10, 10))
    for images, labels in train_ds.take(1):
        for i in range(9):
            ax = plt.subplot(3, 3, i + 1)
            image = images[i].numpy().astype('uint8')
            image = image[:, :, 0]
            plt.imshow(image, cmap='gray')
            plt.title(class_names[np.argmax(labels[i])], )
            plt.axis("off")

print('Number of validation batches: %d' %
tf.data.experimental.cardinality(val_ds))
print('Number of test batches: %d' % tf.data.experimental.cardinality(test_ds))

AUTOTUNE = tf.data.AUTOTUNE
```

```
train_ds = train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUNE)
val_ds = val_ds.cache().prefetch(buffer_size=AUTOTUNE)
test_ds = test_ds.cache().prefetch(buffer_size=AUTOTUNE)

num_classes = 8

metrics = [
      keras.metrics.CategoricalAccuracy(name='categorical_accuracy'),
      tfa.metrics.MatthewsCorrelationCoefficient(num_classes=8, name='MCC'),
      tfa.metrics.FBetaScore(num_classes=8, average='weighted', beta=2.0,
name='F2'),
      keras.metrics.AUC(name='auc'),
      keras.metrics.AUC(name='prc', curve='PR'),
]

def holdout_results(model):
    result = model.evaluate(test_ds)
    return dict(zip(model.metrics_names, result))

x_train = []
y_train = []

for images, labels in train_ds.as_numpy_iterator():
    x_train.append(images)
    y_train.append(labels)

x_train = np.array(x_train)
y_train = np.array(y_train)
class CNNHyperModel(kt.HyperModel):
```

```
    def __init__(self, input_shape, num_classes):
        self.input_shape = input_shape
        self.num_classes = num_classes
    def build(self, hp):
        model = tf.keras.Sequential()
        model.add(tf.keras.layers.InputLayer(input_shape=self.input_shape))
        model.add(tf.keras.layers.Rescaling(1./255))
        for i in range(hp.Int('conv_blocks', 3, 5, default=3)):
            filters = hp.Int('filters_' + str(i), 32, 256, step=32)
            model.add(tf.keras.layers.Conv2D(filters, kernel_size=(3,3),
activation='relu'))
            model.add(tf.keras.layers.MaxPooling2D())
        model.add(tf.keras.layers.Flatten())
        model.add(tf.keras.layers.Dense(units=hp.Int('dense_units', 32, 512,
step=32), activation='relu'))
        model.add(tf.keras.layers.Dense(self.num_classes, activation='softmax'))
        model.compile(optimizer=tf.keras.optimizers.Adam(hp.Float('learning_rate',
1e-4, 1e-2, sampling='log')),
                      loss='categorical_crossentropy', metrics=metrics)
        return model

hypermodel = CNNHyperModel(input_shape=image_size + (1,),
num_classes=num_classes)

tuner = kt.GridSearch(
    hypermodel,
    objective=kt.Objective("val_categorical_accuracy", direction="max"),
    max_trials=20,  # максимальна кількість спроб
    directory='./tuner_params',
    project_name='Deplomka')
```

```
tuner.search(train_ds, validation_data=val_ds)
best_hps = tuner.get_best_hyperparameters(num_trials=1)[0]
print(f"Найкращі гіперпараметри: {best_hps.values}")

# Побудова моделі з найкращими гіперпараметрами
model = tuner.hypermodel.build(best_hps)
history = model.fit(train_ds, validation_data=val_ds, epochs=15)

val_accuracy_per_fold = []
n_splits = 5
kf = KFold(n_splits=n_splits, shuffle=True, random_state=10)
num_fold = 1

train_ds = tf.keras.utils.image_dataset_from_directory('./train/',
                                                       image_size=image_size,
                                                       label_mode='categorical',
                                                       color_mode='grayscale',
                                                       batch_size=batch_size,
                                                       seed=10)

# Convert the dataset to numpy arrays for use with KFold
images, labels = [], []
for x, y in train_ds:
    images.append(x.numpy())
    labels.append(y.numpy())
images = np.concatenate(images)
labels = np.concatenate(labels)

print(f"Total images for training - {len(images)}")
```

```
from tensorflow.keras import backend as K

# K-Fold cross-validation
for train_index, val_index in kf.split(images):
    print(f"----------- Fold {num_fold} -----------")
    train_images, val_images = images[train_index], images[val_index]
    train_labels, val_labels = labels[train_index], labels[val_index]
    print(f"Train - {len(train_images)}")
    print(f"Val - {len(val_images)}")

    # Use a copy of the model to avoid overfitting
    model = tf.keras.models.clone_model(model)
    model.compile(optimizer='adam', loss='categorical_crossentropy',
metrics=metrics)
    model.fit(train_images, train_labels, epochs=15, batch_size=batch_size,
verbose=1)
    val_accuracy = model.evaluate(val_images, val_labels, verbose=0)[1]
    print(f"validation_accuracy for {len(val_images)} images - {val_accuracy}")
    val_accuracy_per_fold.append(val_accuracy)

    # Clear GPU memory
    K.clear_session()

    num_fold += 1

print(f"Average validation accuracy over {n_splits} folds:
{np.mean(val_accuracy_per_fold)}")

model.summary()
```

```
def plot_cm(model, data):
    with plt.rc_context(params):
        y_true = []
        y_pred = []
        for x, y in data:
            y = tf.argmax(y, axis=1)
            y_true.append(y)
            print (y_true)
            y_pred.append(tf.argmax(model.predict(x), axis=1))
            print (y_pred)

        y_pred = tf.concat(y_pred, axis=0)
        y_true = tf.concat(y_true, axis=0)

        cm = confusion_matrix(y_true, y_pred)
        fig = plt.figure(figsize=(10, 10))
        ax1 = fig.add_subplot(1, 1, 1)
        sns.set(font_scale=1.4)  # for label size
        sns.heatmap(cm, cmap='binary', annot=True, fmt='d',
xticklabels=class_names, yticklabels=class_names, annot_kws={"size": 10},
            cbar=False)
        ax1.set_ylabel('True Values', fontsize=14)
        ax1.set_xlabel('Predicted Values', fontsize=14)
        plt.show()

plt.figure(figsize=(12, 10))

# Loss
plt.plot(history.history['loss'], label='Training Loss')
```

```
plt.plot(history.history['val_loss'], label='Validation Loss')

# Categorical Accuracy
plt.plot(history.history['categorical_accuracy'], label='Training Categorical Accuracy')
plt.plot(history.history['val_categorical_accuracy'], label='Validation Categorical Accuracy')

# MCC
plt.plot(history.history['MCC'], label='Training MCC')
plt.plot(history.history['val_MCC'], label='Validation MCC')

# F2
plt.plot(history.history['F2'], label='Training F2')
plt.plot(history.history['val_F2'], label='Validation F2')

# AUC
plt.plot(history.history['auc'], label='Training AUC')
plt.plot(history.history['val_auc'], label='Validation AUC')

# PRC
plt.plot(history.history['prc'], label='Training PRC')
plt.plot(history.history['val_prc'], label='Validation PRC')

plt.title('Training and Validation Metrics')
plt.xlabel('Epoch')
plt.ylabel('Metrics')
plt.legend()
plt.tight_layout()
plt.show()
```

```
test_ds = tf.keras.utils.image_dataset_from_directory('./test/',
                                                      image_size=image_size,
                                                      label_mode='categorical',
                                                      color_mode='grayscale',
                                                      batch_size=batch_size,
                                                      seed=10)

# Convert the test dataset to numpy arrays
test_images, test_labels = [], []
for x, y in test_ds:
    test_images.append(x.numpy())
    test_labels.append(y.numpy())
test_images = np.concatenate(test_images)
test_labels = np.concatenate(test_labels)

K.clear_session()

# Make predictions on the test dataset
predictions = model.predict(test_images)
predicted_labels = np.argmax(predictions, axis=1)

# Create a prediction matrix using scikit-learn's confusion_matrix function
prediction_matrix = confusion_matrix(
    np.argmax(test_labels, axis=1), predicted_labels)

# Plot the prediction matrix as a heatmap using Matplotlib
plt.imshow(prediction_matrix, cmap='Blues')
plt.colorbar()
```

```
plt.xlabel('Predicted label')
plt.ylabel('True label')

# Add the counts inside the cells
for i in range(prediction_matrix.shape[0]):
    for j in range(prediction_matrix.shape[1]):
        plt.text(j, i, prediction_matrix[i, j], ha='center', va='center', color='black')

plt.show()

holdout_results(model)

import sqlite3

conn = sqlite3.connect("../vehicle_classify.db")
cursor = conn.cursor()

sql = "SELECT COUNT(model_id) AS count FROM Models"
cursor.execute(sql)

model_id = cursor.fetchall()[0][0] + 1;

model_path = f"models/model{model_id}.h5"

print(model_path)

model.save(model_path)

cursor.execute("INSERT INTO Models VALUES (?, ?, ?)",
                    [None, "wartech_classifier", model_path])
```

```
conn.commit()
```

Додаток Б Лістинг інтерфейсу програмного модулю розпізнавання військової техніки

HTML

```
<!DOCTYPE html>
<html>
  <head>
    <title>Vehicle Classification</title>
    <link
      rel="stylesheet"
      type="text/css"
      href="{{ url_for('static',filename='style.css') }}"
    />
  </head>
  <body>
    <main>
      <h1>Vehicle Classification</h1>
      <form method="POST" enctype="multipart/form-data">
        <div class="drop-zone">
          <span class="drop-zone__prompt"
            >Drop file here or click to upload</span
          >
          <input type="file" name="file" class="drop-zone__input" />
        </div>
        <button type="submit">Classify</button>
        <div class="class-selector">
          <select id="classInput">
            <option value="">Select class</option>
            {% for class_name in classes %}
```

```
      <option value="{{class_name}}">{{class_name}}</option>
      {% endfor %}
    </select>
    <button type="button" id="saveButton">Save Image</button>
  </div>
  <a href="/add_vehicle"
    ><button type="button" id="addClass">Add class</button></a
  >
</form>
<div>
  <h2>Photo:</h2>
  <img src="../{{ image_url }}" />
  <h2>Prediction:</h2>
  <table>
    <thead>
      <tr>
        <th>Class</th>
        <th>Probability</th>
      </tr>
    </thead>
    <tbody>
      {% for class_type, class_name, country, probability in predictions
      %}
      <tr>
        <td>{{ class_type }} {{ class_name }} {{ country }}</td>
        <td>{{ probability }}</td>
      </tr>
      {% endfor %}
    </tbody>
  </table>
```

```
        <div>
          <h2>TX:</h2>
          {% for vehicle_type, len, width, height, weight, max_speed, fuel_type,
fuel_max in
          vehicle_params_additional %}
          <h4>{{ vehicle_type[0] }} {{ vehicle_type[1] }}</h4>
          <div>{{ len[0] }} {{ len[1] }}</div>
          <div>{{ width[0] }} {{ width[1] }}</div>
          <div>{{ height[0] }} {{ height[1] }}</div>
          <div>{{ weight[0] }} {{ weight[1] }}</div>
          <div>{{ max_speed[0] }} {{ max_speed[1] }}</div>
          <div>{{ fuel_type[0] }} {{ fuel_type[1] }}</div>
          <div>{{ fuel_max[0] }} {{ fuel_max[1] }}</div>
          {% endfor %}
        </div>
      </div>
    </main>
    <script src="{{url_for('static', filename='index.js')}}"></script>
    <script>
      document
        .getElementById("addClassButton")
        .addEventListener("click", function () {
          var newClass = document.getElementById("newClassInput").value;
          if (newClass) {
            var select = document.getElementById("classInput");
            var option = document.createElement("option");
            option.text = newClass;
            option.value = newClass;
            select.add(option);
            document.getElementById("newClassInput").value = "";
```

```
        }
      });
    </script>
  </body>
</html>
HTML
<!DOCTYPE html>
<html>
  <head>
    <title>Vehicle Form</title>
    <link
      rel="stylesheet"
      type="text/css"
      href="{{ url_for('static',filename='style.css') }}"
    />
  </head>
  <body>
    {% if message %}
    <h4 style="width: fit-content; margin: auto;">{{ message }}</h4>
    {% endif %}
    <a href="/"><button type="button" id="toMain">To main page</button></a>
    <form method="post">
      <div class="add_class_selector" style="width: fit-content; margin: auto;">
        <div style="display: flex; gap: 40px; flex-wrap: wrap;">
          <div>
            <label for="vehicle_type">Vehicle Type:</label><br />
            <input type="text" id="vehicle_type" name="vehicle_type" /><br />

            <label for="vehicle_model">Vehicle Model:</label><br />
            <input type="text" id="vehicle_model" name="vehicle_model" /><br />
```

```
  <label for="vehicle_country">Vehicle Country:</label><br />
  <input
    type="text"
    id="vehicle_country"
    name="vehicle_country"
  /><br />

  <label for="description">Description:</label><br />
  <textarea id="description" name="description"></textarea><br />
</div>
<div>
  <label for="length">Length:</label><br />
  <input type="number" id="length" name="length" /><br />

  <label for="width">Width:</label><br />
  <input type="number" id="width" name="width" /><br />

  <label for="height">Height:</label><br />
  <input type="number" id="height" name="height" /><br />

  <label for="weight">Weight:</label><br />
  <input type="number" id="weight" name="weight" /><br />

  <label for="max_speed">Max Speed:</label><br />
  <input type="number" id="max_speed" name="max_speed" /><br />

  <label for="fuel_type">Fuel Type:</label><br />
  <input type="text" id="fuel_type" name="fuel_type" /><br />
```

```
            <label for="fuel_max">Fuel Max:</label><br />
            <input type="text" id="fuel_max" name="fuel_max" /><br />
          </div>
        </div>
        <input type="submit" value="Submit" />
      </div>
    </form>
    <div style="width: fit-content; margin: auto;">
      <h2>Existing classes:</h2>
      <ul>
        {% for class_name in classes %}
        <li>{{class_name}}</li>
        {% endfor %}
      </ul>
    </div>
  </body>
</html>
```

CSS

```
main {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
}
button {
    display: block;
    width: 200px;
    margin-top: 50px;
    margin-left: auto;
    margin-right: auto;
```

```
    border: 2px solid #009578;
    background: transparent;
    padding: 10px 0;
    cursor: pointer;
}
.drop-zone {
    max-width: 200px;
    height: 200px;
    padding: 25px;
    display: flex;
    align-items: center;
    justify-content: center;
    text-align: center;
    font-family: "Quicksand", sans-serif;
    font-weight: 500;
    font-size: 20px;
    cursor: pointer;
    color: #cccccc;
    border: 4px dashed #009578;
    border-radius: 10px;
}

.drop-zone--over {
    border-style: solid;
}

.drop-zone__input {
    display: none;
}
```

```
.drop-zone__thumb {
    width: 100%;
    height: 100%;
    border-radius: 10px;
    overflow: hidden;
    background-color: #cccccc;
    background-size: cover;
    position: relative;
}

.drop-zone__thumb::after {
    content: attr(data-label);
    position: absolute;
    bottom: 0;
    left: 0;
    width: 100%;
    padding: 5px 0;
    color: #ffffff;
    background: rgba(0, 0, 0, 0.75);
    font-size: 14px;
    text-align: center;
}
.class-selector {
    display: flex;
    justify-content: space-between;
    align-items: center;
    margin-top: 50px;
    padding: 10px;
    border: 4px dashed #009578;
    border-radius: 10px;
```

```
    font-family: "Quicksand", sans-serif;
    font-weight: 500;
    font-size: 20px;
    background-color: #cccccc;
}

.class-selector select {
    width: 70%;
    background-color: white;
    border-radius: 10px;
    padding: 10px;
    border: none;
    font-family: "Quicksand", sans-serif;
    font-weight: 500;
    font-size: 16px;
}

.class-selector button {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 25%;
    height: 40px;
    border: 2px solid #009578;
    background: #009578;
    color: white;
    cursor: pointer;
    border-radius: 10px;
    font-family: "Quicksand", sans-serif;
    font-weight: 500;
```

```
    font-size: 16px;
    margin-top: 0;
}
.add_class_selector input, textarea {
    display: block;
    width: 200px;
    margin-left: auto;
    margin-right: auto;
    border: 2px solid #009578;
    background: transparent;
    padding: 10px 4px;
}
.add_class_selector label {
    display: block;
    width: 200px;
    margin-left: auto;
    margin-right: auto;
    background: transparent;
    padding: 10px 4px;
}
```

Index.JS

```
const dropZone = document.querySelector(".drop-zone");
const dropZoneInput = document.querySelector(".drop-zone__input");
const dropZonePrompt = document.querySelector(".drop-zone__prompt");
const dropZoneThumbnail = document.querySelector(".drop-zone__thumb");

dropZone.addEventListener("click", () => dropZoneInput.click());

dropZoneInput.addEventListener("change", () => {
    const files = dropZoneInput.files;
```

```
    if (files.length) {
        dropZonePrompt.textContent = files[0].name;
        dropZone.classList.add("drop-zone--over");
        showThumbnail(files[0]);
    }
});

function showThumbnail(file) {
    let reader = new FileReader();

    reader.onload = function(event) {
        dropZoneThumbnail.style.backgroundImage = `url('${event.target.result}')`;
        dropZoneThumbnail.dataset.label = file.name;
    };

    reader.readAsDataURL(file);
}

const saveButton = document.getElementById("saveButton");
const classInput = document.getElementById("classInput");

saveButton.addEventListener("click", () => {
    if (classInput.value) {
        const xhr = new XMLHttpRequest();
        const formData = new FormData();
        formData.append("class", classInput.value);
        formData.append("file", dropZoneInput.files[0]);
        xhr.open("POST", "/save_image");
        xhr.send(formData);
        alert("Image Saved")
```

```
    }
});
```

Додаток В Лістинг бази даних

create_db.py

```
import sqlite3
from sqlite3 import Error


def create_connection(db_file):
    """ create a database connection to the SQLite database
        specified by db_file
    :param db_file: database file
    :return: Connection object or None
    """
    conn = None
    try:
        conn = sqlite3.connect(db_file)
        return conn
    except Error as e:
        print(e)

    return conn


def create_table(conn, create_table_sql):
    """ create a table from the create_table_sql statement
    :param conn: Connection object
    :param create_table_sql: a CREATE TABLE statement
    :return:
```

```
    """
    try:
        c = conn.cursor()
        c.execute(create_table_sql)
    except Error as e:
        print(e)

def main():
    database = "vehicle_classify.db"

    sql_create_models_table = """ CREATE TABLE IF NOT EXISTS Models (
                                    model_id INTEGER PRIMARY KEY,
                                    model_name VARCHAR(255) NOT NULL,
                                    model_path VARCHAR(255) NOT NULL
                                ); """

    sql_create_classification_results_models_table = """ CREATE TABLE IF
NOT EXISTS Classification_Results_Models (
                                    relation_id INTEGER PRIMARY KEY,
                                    model_id INT NOT NULL,
                                    classification_result_id INT NOT NULL,
                                    FOREIGN KEY(model_id) REFERENCES
Models(model_id),
                                    FOREIGN KEY(classification_result_id) REFERENCES
Classification_Results(result_id)
                                ); """

    sql_create_vehicle_images_table = """CREATE TABLE IF NOT EXISTS
Vehicle_Images (
                                    image_id INTEGER PRIMARY KEY,
```

```
                    image_path VARCHAR(255) NOT NULL,
                    image_date DATETIME NOT NULL
                );"""

    sql_create_classification_results_table = """CREATE TABLE IF NOT
EXISTS Classification_Results (
                    result_id INTEGER PRIMARY KEY,
                    image_id INT NOT NULL,

                    FOREIGN KEY(image_id) REFERENCES
Vehicle_Images(image_id)
                ); """

    sql_create_classification_parameters_table = """CREATE TABLE IF NOT
EXISTS Classification_Parameters (
                    param_id INTEGER PRIMARY KEY,
                    result_id INT NOT NULL,
                    vehicle_id INT NOT NULL,
                    vehicle_probability REAL NOT NULL,
                    FOREIGN KEY(result_id) REFERENCES
Classification_Results(result_id)
                    FOREIGN KEY(vehicle_id) REFERENCES
Vehicles(vehicle_id)
                ); """

    sql_create_vehicles_table = """ CREATE TABLE IF NOT EXISTS Vehicles (
                    vehicle_id INTEGER PRIMARY KEY,
                    vehicle_type VARCHAR(255),
                    vehicle_model VARCHAR(255),
                    vehicle_country VARCHAR(255),
```

```
                    description TEXT,
                    FOREIGN KEY(vehicle_id) REFERENCES
Vehicle_Parameters(vehicle_params_id)
                ); """

    sql_create_vehicle_parameters_table = """ CREATE TABLE IF NOT EXISTS
Vehicle_Parameters (
                    vehicle_params_id INTEGER PRIMARY KEY,
                    length INT NOT NULL,
                    width INT NOT NULL,
                    height INT NOT NULL,
                    weight INT NOT NULL,
                    max_speed INT NOT NULL,
                    fuel_type VARCHAR(255),
                    fuel_max VARCHAR(255)
                ); """

    # create a database connection
    conn = create_connection(database)

    # create tables
    if conn is not None:
        create_table(conn, sql_create_models_table)
        create_table(conn, sql_create_vehicles_table)
        create_table(conn, sql_create_vehicle_images_table)
        create_table(conn, sql_create_vehicle_parameters_table)
        create_table(conn, sql_create_classification_results_table)
        create_table(conn, sql_create_classification_parameters_table)
        create_table(conn, sql_create_classification_results_models_table)
    else:
```

```
        print("Error! cannot create the database connection.")



if __name__ == '__main__':
    main()
```

app.py

```
from flask import Flask, render_template, request, current_app, send_file, abort
import tensorflow as tf
import numpy as np
from PIL import Image
from pathlib import Path
import os
from werkzeug.utils import secure_filename
from PIL import Image
import base64
from io import BytesIO
from datetime import datetime

import sqlite3



class MatthewsCorrelationCoefficient(tf.keras.metrics.Metric):
    def __init__(self, name='matthews_correlation', **kwargs):
        super().__init__(name=name, **kwargs)
        self.tp = self.add_weight(name='tp', initializer='zeros')
        self.tn = self.add_weight(name='tn', initializer='zeros')
        self.fp = self.add_weight(name='fp', initializer='zeros')
        self.fn = self.add_weight(name='fn', initializer='zeros')
```

```
def update_state(self, y_true, y_pred, sample_weight=None):
    y_pred_pos = tf.round(tf.clip_by_value(y_pred, 0, 1))
    y_pred_neg = 1 - y_pred_pos

    y_pos = tf.round(tf.clip_by_value(y_true, 0, 1))
    y_neg = 1 - y_pos

    tp = tf.reduce_sum(y_pos * y_pred_pos)
    tn = tf.reduce_sum(y_neg * y_pred_neg)
    fp = tf.reduce_sum(y_neg * y_pred_pos)
    fn = tf.reduce_sum(y_pos * y_pred_neg)

    self.tp.assign_add(tp)
    self.tn.assign_add(tn)
    self.fp.assign_add(fp)
    self.fn.assign_add(fn)

def result(self):
    numerator = self.tp * self.tn - self.fp * self.fn
    denominator = tf.sqrt((self.tp + self.fp) * (self.tp + self.fn)
                          * (self.tn + self.fp) * (self.tn + self.fn))
    return numerator / (denominator + tf.keras.backend.epsilon())

def reset_states(self):
    self.tp.assign(0)
    self.tn.assign(0)
    self.fp.assign(0)
    self.fn.assign(0)
```

```
app = Flask(__name__, template_folder=Path(
    __file__).resolve().parent / 'templates')

tf.keras.utils.get_custom_objects(
)['MatthewsCorrelationCoefficient'] = MatthewsCorrelationCoefficient



@app.route('/', methods=['GET', 'POST'])
def home():
    app = current_app._get_current_object()

    conn = sqlite3.connect("vehicle_classify.db")
    cursor = conn.cursor()

    classes = []

    cursor.execute("SELECT vehicle_model FROM Vehicles ORDER BY
vehicle_model")
    for row in cursor.fetchall():
        classes.append(row[0])

    num_classes = len(classes)

    if request.method == 'POST':
        model_id = 1

        cursor.execute(
            "SELECT model_path FROM Models WHERE model_id = ?",
[model_id])
```

```
        model = tf.keras.models.load_model(
            f"training/{cursor.fetchall()[0][0]}", compile=False)
        model.summary()
        model.compile(optimizer='adam', loss='binary_crossentropy', metrics=[
                'accuracy', MatthewsCorrelationCoefficient()])

        file = request.files['file']
        img = Image.open(file)
        img = img.resize((128, 128))

        cursor.execute("SELECT COUNT(image_id) AS count FROM
Vehicle_Images")

        image_id = cursor.fetchall()[0][0] + 1

        image_path = f'pred_images/image{image_id}.png'
        img.save(image_path)

        # Save the image data to the SQL database
        current_datetime = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
        cursor.execute(f"INSERT INTO Vehicle_Images VALUES (?,?,?)", [
                None, image_path, current_datetime])
        conn.commit()

        cursor.execute(f"INSERT INTO Classification_Results VALUES (?,?)", [
                None, image_id])

        cursor.execute('SELECT max(result_id) FROM Classification_Results')
        result_id = cursor.fetchone()[0]
```

```
        cursor.execute("INSERT INTO Classification_Results_Models VALUES
(?,?,?)",
                       [None, model_id, result_id])
        conn.commit()

        # Convert the image to grayscale
        img = img.convert('L')

        # Convert the image to a numpy array
        img_array = np.array(img)

        # Expand the dimensions of the array to match the input shape of the model
        img_array = np.expand_dims(img_array, axis=2)
        img_array = np.expand_dims(img_array, axis=0)
        prediction = model.predict(img_array)
        top3 = sorted(zip(prediction[0], range(num_classes)), reverse=True)[:3]
        top3_classes = [(classes[idx], float('%.3f' % (prob*100)))
                        for (prob, idx) in top3]

        top3_classes = []

        vehicle_params_additional = []

        for (prob, idx) in top3:
            vehicle_id = idx + 1

            cursor.execute(
                "SELECT * FROM Vehicles WHERE vehicle_id = ?", [vehicle_id])

            vehicle_data = cursor.fetchall()
```

```
            top3_classes.append([
                vehicle_data[0][1],
                vehicle_data[0][2],
                f"({vehicle_data[0][3]})",
                float('%.3f' % (prob*100))
            ]
            )

            cursor.execute(
                "SELECT * FROM Vehicle_Parameters WHERE vehicle_params_id
= ?", [vehicle_id])

            for row in cursor.fetchall():
                print(row)
                temp_data = [
                    [
                    "Тип: ", vehicle_data[0][2]
                    ],
                    [
                    "Довжина: ", row[1]
                    ],
                    [
                    "Ширина: ", row[2]
                    ],
                    [
                    "Висота: ", row[3]
                    ],
                    [
                    "Вага: ", row[4]
```

```
                ],
                [
                "Макс. швидкість: ", row[5]
                ],
                [
                "Тип палива: ", row[6]
                ],
                [
                "Об'єм баку: ", row[7]
                ],
            ]
            vehicle_params_additional.append(temp_data)

        cursor.execute("INSERT INTO Classification_Parameters VALUES
(?,?,?,?)",
                       [None, result_id, vehicle_id, float('%.3f' % (prob*100))])
        conn.commit()

    return render_template('index.html', classes=classes, image_url=image_path,
predictions=top3_classes,
vehicle_params_additional=vehicle_params_additional)

    return render_template('index.html', classes=classes)


@app.route('/add_vehicle', methods=['GET'])
def add_vehicle_class():
    conn = sqlite3.connect("vehicle_classify.db")
    cursor = conn.cursor()
    classes = []
```

```
    cursor.execute("SELECT vehicle_model FROM Vehicles ORDER BY
vehicle_model")

    for row in cursor.fetchall():
        classes.append(row[0])
    return render_template('add_vehicle.html', classes=classes)


@app.route('/add_vehicle', methods=['POST'])
def add_vehicle():
    conn = sqlite3.connect("vehicle_classify.db")
    cursor = conn.cursor()
    vehicle_type = request.form['vehicle_type']
    vehicle_model = request.form['vehicle_model']
    vehicle_country = request.form['vehicle_country']
    description = request.form['description']
    length = request.form['length']
    width = request.form['width']
    height = request.form['height']
    weight = request.form['weight']
    max_speed = request.form['max_speed']
    fuel_type = request.form['fuel_type']
    fuel_max = request.form['fuel_max']

    cursor.execute("INSERT INTO Vehicle_Parameters (length, width, height,
weight, max_speed, fuel_type, fuel_max) VALUES (?, ?, ?, ?, ?, ?, ?)",
                   (length, width, height, weight, max_speed, fuel_type, fuel_max))

    last_row_id = cursor.lastrowid
```

```
    cursor.execute("INSERT INTO Vehicles (vehicle_id, vehicle_type, vehicle_model, vehicle_country, description) VALUES (?, ?, ?, ?, ?)",
                   (last_row_id, vehicle_type, vehicle_model, vehicle_country, description))

    conn.commit()

    classes = []
    cursor.execute("SELECT vehicle_model FROM Vehicles ORDER BY vehicle_model")

    for row in cursor.fetchall():
        classes.append(row[0])

    conn.close()

    return render_template('add_vehicle.html', classes=classes, message="Vehicle submitted successfully!")



@app.route('/save_image', methods=['POST'])
def save_image():
    file = request.files['file']
    class_name = request.form['class']

    # make sure the class directory exists
    directory = os.path.join('saved_images', class_name)
    os.makedirs(directory, exist_ok=True)

    # save the image to the class directory
```

```
    filename = secure_filename(file.filename)
    file.save(os.path.join(directory, filename))

    return 'Image saved successfully'


@app.route('/pred_images/<filename>')
def get_image(filename):
    image_path = f'pred_images/{filename}'
    if os.path.exists(image_path):
        return send_file(image_path, mimetype='image/png')
    else:
        abort(404)
```