Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБКА ЗАСТОСУНКУ ДЛЯ КЛАСИФІКАЦІЇ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ
(тема)

Виконав:
студент 4 курсу, групи ІТІНФ-18-2
Алевська А.І.
(прізвище, ініціали)

Спеціальності 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф. Машталір С.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Кобилін О.А.
(підпис) (прізвище, ініціали)

2022 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 Комп'ютерні науки
(код і повна назва)
Тип програми освітньо-професійна
Освітня програма Інформатика
(повна назва освітньої програми)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2022 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Алевській Анні Ігорівні
(прізвище, ім'я, по батькові)
1. Тема роботи Розробка застосунку для класифікації об'єктів на зображеннях

затверджена наказом по університету від «16» травня 2022 року № 541Ст
2. Термін подання студентом роботи до екзаменаційної комісії 30 травня 2022 р.
3. Вихідні дані до роботи: алгоритм попередньої обробки зображень, алгоритм розпізнавання образів, комп'ютерна модель розпізнавання на базі згорткової нейронної мережі.

4. Перелік питань, що потрібно опрацювати в роботі
1. Огляд методів попередньої обробки зображень.
2. Огляд методів розпізнавання об'єктів на зображенні.
3. Вирішення задачі класифікації за допомогою нейронних мереж.
4. Аналіз структури згорткових нейронних мереж.
5. Розробка архітектури згорткової нейронної мережі.
6. Програмна реалізація згорткової нейронної мережі з використанням бібліотеки OpenCV, TensorFlow та мови програмування Python.
7. Порівняння ефективності розроблених моделей.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) Розбір принципу роботи згортки, структура CNN, тестові зображення, розгляд програмного коду алгоритму, аналіз результату роботи алгоритмів, розгляд середовища програмування.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата
Консультант з дотримання діючих стандартів та норм	Доцент Бєлова Н.В.		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Отримання завдання на до атестаційної роботи	18.04.2022	
2	Аналіз завдання, аналіз літератури	18.04.22-25.04.22	
3	Огляд існуючих алгоритмів попередньої обробки зображень	26.04.22-29.04.22	
4	Огляд існуючих алгоритмів розпізнавання образів	30.04.22-02.05.22	
5	Програмна реалізація	03.05.22-10.05.22	
6	Тестування застосунку	11.05.22-15.05.22	
7	Оформлення пояснювальної записки	18.05.22-27.05.22	
8	Перевірка на плагіат	03.06.22	
9	Рецензування	03.06.22	
10	Підготовка презентації та доповіді	27.05.22-05.06.22	
11	Попередній захист атестаційної роботи	08.06.22	

Дата видачі завдання 18 квітня 2022 р.

Студент _____
(підпис)

Керівник роботи _____ проф. Машталір С.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 46 с., 2 табл., 18 рис., 1 дод., 42 джерела.

РОЗПІЗНАВАННЯ ОБ'ЄКТІВ, СИСТЕМИ КОМП'ЮТЕРНОГО ЗОРУ, КЛАСИФІКАЦІЯ ОБ'ЄКТІВ, АНАЛІЗ ОБ'ЄКТІВ, НЕЙРОННА МЕРЕЖА ЗГОРТКИ, OPENCV.

Об'єктом роботи є послідовність різноракурсних зображень об'єктів.

Метою роботи є розробка методів, що базуються на використанні нейронної мережі згортки та попередній обробці зображень.

Використано алгоритм пошуку в глибину, досліджено навчання нейронної мережі згортки, розроблено алгоритм виявлення об'єкта на попередньо обробленому зображенні з подальшим виведенням списку обмежувальних прямокутників, в яких виявлено об'єкт.

У результаті роботи здійснена програмна реалізація нейронної мережі згортки.

OBJECT DETECTION, COMPUTER VISION SYSTEMS, OBJECT CLASSIFICATION, OBJECT ANALYSIS, CONVOLUTIONAL NEURAL NETWORK, OPENCV.

The object of the research is a sequence of multidisciplinary images of objects.

The aim of the research is to develop methods based on the use of convolution neural network and image pre-processing.

Methods used depth search algorithm. The study of the convolution neural network studying, the object detection algorithm on the pre-processed image is developed with the subsequent output of the list of limiting rectangles in which the object is detected.

As a result, the software realization of a neural network of convolution is carried out.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

РО – розпізнавання об’єктів

VDF – Vector Directional Filters (векторні спрямовані фільтри)

MEEF – Multichannel Edge Enhancement Filter (багатоканальний фільтр посилення країв)

CIE – International Commission on Illumination (Міжнародна комісія з освітлення)

MRF – Markov Random Field (випадкове поле Маркова)

EM – Expectation-Maximization (алгоритм максимізації очікування)

CNN – Convolutional Neural Network (згорткова нейронна мережа)

MLN – Multilayered Neural Network (багатошарова нейронна мережа)

MLP – Multilayered Perceptrons (багатошарові персептрони)

OpenCV – Open-Source Computer Vision (відкритий ресурс комп’ютерного зору)

HDF – Hierarchical Data Format (ієрархічні формати даних)

RELU – Rectified Linear Unit (очищена лінійна одиниця)

REPL – Read-Eval-Print Loop (цикл читання, друку та оцінки)

ВСТУП

На сьогоднішній день системи комп’ютерного зору, нейронні мережі та методи глибокого навчання стають дедалі більш використовуваними, тож попит на якісні алгоритми, які будуть ефективніше виявляти та обробляти інформацію, аналогічно зростає.

Хоча моделі виявлення і класифікації об’єктів такі, як Mask R-CNN і YOLO вже існують, вони обмежені тим, що модель або вже має бути навчена на зображеннях із заздалегідь підготованим набором даних, або набір маркерів зображень повинен бути введений власноруч для точного налаштування. Тож можна зазначити, що задача розпізнавання образів досі не вирішена в повному обсязі. Потрібна розробка підходів, які б були універсальними [1-2].

Задачу створення такого підходу поставлено у якості предмету дослідження атестаційної роботи на прикладі проекту для коректного виявлення і класифікації суден на великих супутникових знімках з використанням згорткової нейронної мережі (CNN), яка була навчена на зображеннях розміром 80×80 з даними «корабель/некорабель» у поєднанні з призначеним для користувача алгоритмом виявлення об’єктів. Цей підхід дозволяє ефективно і точно виявляти кораблі на супутникових знімках різних розмірів.

1 ІСНУЮЧІ МЕТОДИ КЛАСИФІКАЦІЇ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ

1.1 Попередня обробка зображень

На сьогодні існує безліч різних застосувань обробки зображень за допомогою комп'ютерного зору. Обробка зображень включає маніпулювання цифровими зображеннями для отримання додаткової інформації. За останнє десятиліття спостерігалося багато інновацій у сфері комп'ютерного обладнання, що призвело до появи більш швидких обчислювальних та графічних процесорів. Це дозволило нам вирішувати нові постійно виникаючі проблеми за допомогою обробки зображень [3].

Її застосування варіюється від медицини до сфери розваг, включно з геологічною обробкою та дистанційним зондуванням. Мультимедійні системи, одна із опор сучасного інформаційного суспільства, значною мірою покладаються на цифрову обробку зображень.

Цифрове зображення – це двовимірна матриця пікселів різних значень. Усі зображення складаються з пікселів, які є необробленими будівельними блоками зображень. Зображення складаються з пікселів у сітці. Зображення розміром 640×480 має 640 стовпців (ширина) і 480 рядків (висота). У зображенні з такими розмірами є 640×480 = 307200 пікселів [4].

По суті, обробка зображень включає в себе наступні основні кроки:

– імпортування зображення за допомогою інструментів отримання зображень;

– попередня обробка зображень/аналіз і маніпулювання зображеннями;

– вихідні данні, у яких можна змінити зображення або провести будь-який аналіз [1].

Попередня обробка кольорового зображення та сегментація є класичними прикладами багатоканальної обробки інформації. Основними

проблемами, з якими стикаються при обробці кольорових зображень, є різноманітність і величезний діапазон інтенсивності кольорів, а також обробка спектральних характеристик різних компонентів кольору. Якщо бути точним, Завдання обробки кольорових зображень включає в себе величезні витрати на обробку, оскільки інформація про інтенсивність кольору, як правило, проявляється у вигляді домішок різних кольорових компонентів. Крім того, відносні пропорції кольорів компонентів та їх взаємозв'язки також демонструють нелінійні характеристики.

Загалом, характеристики та ефективність алгоритму визначаються областю вхідних даних, які підлягають обробці. Типові вхідні домени включають пікселі, локальні об'єкти, краї зображення, вбудовані об'єкти тощо [5]. Вихідні домени незмінно містять однорідні сегменти зображення, краї виявлених/локалізованих об'єктів, області/сегменти та різні об'єкти, що відрізняються за розміром, формою, кольором та текстурною інформацією.

З існуючих процедур покращення зображення, техніки фільтрації стали дуже популярними протягом багатьох років для вирішення проблеми видалення шуму та покращення краю. Векторні спрямовані фільтри [6] грають дуже важливу роль в обробці кольорових зображень, розглядаючи їх як векторні сигнали. Як спрямована, так і амплітудна обробка вмісту сигналу здійснюється цим класом фільтрів незалежно [7, 8]. Розглядають багатоканальний фільтр посилення країв на основі векторної медіани для покращення погіршених країв у кольорових зображеннях. У запропонованому підході вхідний багатоканальний сигнал фільтрується трьома підфільтрами. Кінцевий вихід визначається шляхом порівняння вихідних даних підфільтрів та їх векторної медіани.

Існує також адаптивний багатоканальний фільтр найближчого сусіда для вирішення проблеми ослаблення шуму для багатоканальних даних. У фільтрі використовуються адаптивно визначені залежні від даних коефіцієнти на основі нової міри відстані, що включає як векторну спрямовану фільтрацію, так і фільтрацію векторної величини.

Зовсім іншу структуру для хроматичної фільтрації кольорових зображень ввів Луччезе [9]. Підхід зосереджений на кодуванні хроматичного та ахроматичного вмісту кольорового зображення різними способами. Хроматичний вміст закодовано в CIE-координати кольоровості. Ахроматичний вміст кодується як тристимульне CIE-значення. Кольори в хроматичній частині додаються згідно з відомим законом центру ваги адитивних кольорових сумішей і відповідно фільтруються. Ахроматичний вміст обробляється за традиційними схемами лінійної або нелінійної фільтрації. Існує безліч хроматичних фільтрів, призначених для обробки кольорових зображень із зашумленням і шумами. Але більшість із цих підходів страждають від потреби вапріорі знання про розподіл шуму у вхідних зображеннях.

Що стосується сегментації кольорових зображень, можна звернутись до схеми сегментації зображення Макроджіанніса [10] з багатьма роздільною здатністю на основі теоретико-графічного підходу. У цій техніці використовується відношення міжрегіональної несхожості між сусідніми областями зображень, які розглядаються, на основі ознак. Нарешті, регіони групуються для досягнення бажаних сегментованих результатів. Однак стратегія групування залежить від вибраного міжрегіонального відношення несхожості.

Венбінг [11] розробив надійний підхід у реальному часі для сегментації кольорового зображення з використанням сегментації MS та нормалізованого вирізу (Ncut) [12] методи розподілу. Метод вдається до методу Ncut для оптимізації зображень, кластеризованих за алгоритмом MS. Однак ці методи страждають від недоліків евристичного вибору порогового власного значення для досягнення стабільних сегментів.

Моделі випадкового поля Маркова часто використовувалися для моделювання та аналізу просторових залежностей між даними мультиспектрального зображення підтримується алгоритмом максимізації очікування [13]. Однак обчислювальна складність цих методів перешкоджає

їх використанню в програмах реального часу. Алгоритм EM використовується для оцінки різних параметрів у прихованих марківських моделях з метою зменшення структури залежності в моделях. Для оцінки параметрів моделі використовують просторово обмежений алгоритм EM. Процедура оцінки використовує залежний від даних коефіцієнт штрафу, щоб максимізувати ймовірність наборів даних, тим самим зменшуючи накладні витрати на обчислення.

Для належної оцінки структурних розподілів даних зображень було запропоновано декілька моделей статистичної суміші. Приклади включають суміш Гаусса і моделі суміші Діріхле. Гауссова суміш популярна, оскільки вона ізотропна і може представляти розподіли даних за допомогою вектора середнього значення та матриці коваріації. Паналвер [14] використовував його, щоб знайти рішення з максимальною імовірністю проблеми сегментації за допомогою одного стартового ядра. Однак гауссовій суміші не вдається виявити справжню структуру негауссових та асиметричних розподілів даних. У цих ситуаціях розподіл Діріхле, який є багатовимірним узагальненням бета-розподілу, може бути дуже хорошим вибором для моделювання даних. Модель суміші Діріхле була застосована для кількох завдань обробки та сегментації зображень, а саме, оцінки гістограми, характеристики бази даних зображень та виявлення шкіри людини в мультимедійних базах даних.

1.2 Алгоритм розпізнавання об’єктів

Розпізнавання об’єктів – це завдання комп’ютерного зору, на яке нещодавно вплинув прогрес, досягнутий у машинному навчанні [3].

Розпізнавання об’єктів можна здійснити кількома способами [4]:

– розпізнавання об’єктів на основі функцій;
– розпізнавання об’єктів Віоли-Джонса;
– класифікації SVM з функціями HOG;

– розпізнавання об’єктів глибокого навчання.

У минулому, створення спеціального детектора об’єктів виглядало трудомістким і складним завданням. Тепер за допомогою таких інструментів, як TensorFlow Object Detection API, з’явилась можливість створювати надійні моделі швидко та з легкістю.

Щоб підготуватися до створення робочої моделі, необхідно зібрати таку кількість даних, яка б була достатньою для того, щоб навчити її. Зазвичай можна знайти готові датасети в загальному доступі, але це залежить від предметної області та цілей, поставлених для задачі. Також можна збирати образи з відеопотоку та зі спеціальних бібліотек.

Нижче наведено деякі приклади використання алгоритмів розпізнавання об’єктів у різних сферах [4].

Розпізнавання обличь. Групою дослідників у Facebook була розроблена система глибокого навчання розпізнавання обличь під назвою «DeepFace», яка дуже ефективно ідентифікує людські обличчя на цифровому зображенні. Google використовує власну систему розпізнавання обличчя в Google Photos, яка автоматично відокремлює всі фотографії на основі людини на зображенні. У розпізнаванні обличчя беруть участь різні компоненти, такі як очі, ніс, рот і брови [14, 15].

Підрахунок людей. Виявлення об’єктів також можна використовувати для підрахунку людей, він використовується для аналізу роботи магазину або статистики натовпу під час фестивалів. Це, як правило, складніше, оскільки люди швидко виходять із кадру. Це дуже важлива програма, оскільки під час збирання натовпу цю функцію можна використовувати для багатьох цілей.

Перевірка промислової якості. Виявлення об’єктів також використовується в промислових процесах для ідентифікації продукції. Пошук конкретного об’єкта за допомогою візуального огляду є основним завданням, яке бере участь у багатьох промислових процесах, таких як сортування, управління запасами, обробка, управління якістю, пакування тощо.

Управління запасами може бути дуже складним, оскільки предмети важко відстежити в режимі реального часу. Автоматичний підрахунок і локалізація об’єктів дозволяє підвищити точність інвентаризації.

Самокеровані автомобілі. Самокеровані автомобілі – це майбутнє, в цьому немає жодних сумнівів. Але працювати з ними дуже складно, оскільки вони поєднують різноманітні методи для сприйняття оточення, включаючи радар, лазерне світло, GPS, одометрію та комп’ютерний зір.

Розширені системи керування інтерпретують сенсорну інформацію для визначення відповідних навігаційних шляхів, а також перешкод, і як тільки датчик зображення виявляє будь-які ознаки живої істоти на своєму шляху, він автоматично зупиняється. Це відбувається дуже швидко і є великим прогресом до створення автомобілів без водіїв.

Безпека. Виявлення об’єктів відіграє дуже важливу роль у безпеці. Будь то ідентифікатор обличчя Apple або сканування сітківки ока, яке використовується в усіх науково-фантастичних фільмах.

Вони також використовується урядом для доступу до каналу безпеки та зіставлення його з наявною базою даних, щоб знайти злочинців або виявити автомобіль грабіжників.

Кожен алгоритм виявлення об’єктів працює по-різному, але всі вони працюють за одним принципом – вилучення функцій. Вони витягують функції з вхідних зображень і використовують ці функції для визначення класу зображення, будь то через MatLab, OpenCV, алгоритм Віоли-Джонса або глибоке навчання.

1.2.1 Метод розпізнавання об’єктів на основі функцій

Функція методу полягає в тому, щоб витягти з контурів зображення ознаки та зв’язки, які можуть бути представлені в базі даних. Можна вибрати ряд об’єктів бази даних, які мають високу ймовірність розміщення в знятому зображенні. Метод повинен вибрати всі зображення об’єктів, що містяться в знятому зображенні (без помилкових негативів), включаючи лише невелику частину інших (помилкові позитиви).

На даний момент існує три різні типи об’єктів:

– лінія: визначається її центральною точкою, довжиною та кутом дотичної. Її знаходять шляхом виділення плоских плям, які виникають при нульовій амплітуді на графіку відбитків пальців. Сама лінія прилягає до двох кінцевих точок на частині контуру, пов’язаного з плоскою плямою;

– дуга: визначається її середньою точкою, довжиною, радіусом і дотичним кутом. Вона схожа на лінію, але плоскі плями повинні з’являтися з більшою амплітудою. Дугу потім припасовують до двох кінцевих точок і середньої точки вздовж частини контуру, пов’язаного з плоскою плямою;

– частка: частка визначається її середньою точкою, двома кінцевими точками, довжиною, кутом дотичної та кутом контуру. Змінюється від початку до кінця. Точний контур неможливо відтворити з цього визначення, але охоплюються найважливіші частини, і це забезпечує дуже гарне наближення. До кожного шипа на графіку відбитків прикріплюється частка за допомогою відповідних кінцевих точок контуру та дотичних.

Відбиток кожного контуру потрібно перевіряти, щоб виділити відповідні ознаки. Об’єкти зберігаються разом із відповідним контуром у програмі. Загальна кількість функцій зазвичай коливається від тисячі до десяти тисяч. Усі зібрані ознаки потрібно об’єднати та зменшити, щоб знайти лише найважливіші.

Це робиться шляхом перевірки надмірності, а потім розумного об’єднання менших об’єктів у більші:

– усі типи об’єктів проходять за допомогою алгоритму групування, який знаходить відповідності за допомогою різних атрибутів для кожного типу об’єктів. Для кожної групи створюється набір середніх атрибутів об’єктів;

– усі групи перевіряються на достатню надмірність, щоб гарантувати, що об’єкт міститься на кількох контурах, тобто є частиною значного контуру. Це означає, що кількість об’єктів тепер значно зменшена, тому можна застосувати більш розумний алгоритм групування;

– лінії різної довжини об’єднуються за умови, що вони паралельні та знаходяться поблизу. Розрахунки виконуються, щоб створити нову лінію, яка охоплює обидві лінії, які її створили;

– дуги поєднуються так само, як і лінії, але також враховуються радіус, точка перетину та локальні дотичні кути. Знову проводяться розрахунки, щоб переконатися, що вони враховують обидві менші дуги;

– частки об’єднуються, якщо вони зміщені одна від одної; це перевіряється шляхом знаходження векторів між кожним із трьох наборів точок. Нова частка обчислюється шляхом переміщення точок на зважену відстань уздовж векторів.

Усі ці кроки скорочення означають, що основні контури зображення описуються лише кількома «суперфункціями», що робить набір кроків обробки набагато ефективнішим [16].

1.2.2 Метод розпізнавання об’єктів Віоли-Джонса

Розроблена Полом Віолою та Майклом Джонсом у 2001 році, система виявлення об’єктів Віоли-Джонса може швидко й точно виявляти об’єкти на зображеннях і особливо добре працює з людським обличчям. Незважаючи на свій вік, фреймворк все ще є провідним гравцем у розпізнаванні обличь поряд з багатьма частинами CNN. Фреймворк Віоли-Джонса Object Detection Framework поєднує концепції подібних до Хаара функцій, інтегральних

зображень, алгоритму AdaBoost і каскадного класифікатора, щоб створити систему для швидкого та точного виявлення об’єктів.

Існує дві стадії алгориму:

– тренування;

– розпізнавання.

Алгоритм був розроблений для фронтальних облич, тому він здатний краще розпізнати фронтальні зображення, а не обличчя, що дивляться убік, вгору чи вниз. Перш ніж розпізнати обличчя, зображення перетворюється на відтінки сірого, оскільки з ним простіше працювати та обробляти менше даних. Алгоритм Віоли-Джонса спочатку виявляє обличчя на зображенні у відтінках сірого, а потім знаходить розташування на кольоровому зображенні.

Віола-Джонс окреслює поле і шукає обличчя в ньому. Вікно переміщається на крок праворуч після проходження кожної плитки на зображенні. З меншими кроками ряд блоків виявляє риси, схожі на обличчя (риси, подібні до Хаара), а дані всіх цих блоків, разом узяті, допомагають алгоритму визначити, де знаходиться обличчя [17].

1.2.3 Метод класифікації SVM з функціями HOG

Гістограма орієнтованих градієнтів (HOG) – це дескриптор ознак, що використовується для виявлення об’єктів у комп’ютерному баченні та обробці зображень. Техніка дескриптора HOG підраховує випадки орієнтації градієнта в локалізованих частинах зображення – вікні виявлення або області інтересу.

Реалізація алгоритму дескриптора HOG виглядає наступним чином:

– зображення поділяють на невеликі зв’язані області, які називаються комірками, і для кожної клітинки розраховується гістограма напрямків градієнта або орієнтації країв для пікселів всередині комірки;

– далі розділяють кожну клітинку на кутові блоки відповідно до орієнтації градієнта;

– піксель кожної комірки вносить зважений градієнт у відповідний кутовий бік;

– групи суміжних клітинок розглядаються як просторові області, які називаються блоками. Групування комірок у блок є основою для групування та нормалізації гістограм;

– нормалізована група гістограм являє собою блокову гістограму. Набір цих блокових гістограм представляє дескриптор.

Для обчислення дескриптора HOG потрібні наступні основні параметри конфігурації:

– маски для обчислення похідних і градієнтів;

– геометрія розбиття зображення на клітинки та групування клітинок у блок;

– блокове перекриття;

– параметри нормалізації.

Рекомендовані значення параметрів HOG:

– 1D центрована похідна маска [-1, 0, +1];

– розмір вікна виявлення 64×128;

– розмір осередку 8×8;

– розмір блоку 16×16 (2×2 клітинки) [18].

1.3 Класифікація об’єктів нейронною мережею згортки

Одним із важливих аспектів вирішення будь-якої проблеми за допомогою машинного навчання є виділення функцій з набору об’єктів. У випадку обробки зображень набором функцій є по суті кожен піксель, з якого складається зображення. Набір функцій залежить від роздільної здатності та розміру зображення. Кількість пікселів в одному мегабайті залежить від колірного режиму зображення:

– у 8-бітному (256 кольорів) зображенні в одному мегабайті міститься 1048576 або 1024×1024 пікселів;

– 16-бітове (65536 кольорів) зображення – один мегабайт містить 524288 (1024×512) пікселів;

– 24-бітове зображення (16,7 мільйонів кольорів) – один мегабайт має приблизно 349920 (486×720) пікселів;

– 32-бітове зображення (16,7 мільйонів кольорів) – один мегабайт має 262144 (512×512) пікселів;

– 48-бітне зображення – один мегабайт має лише 174960 (486×360) пікселів.

CNN працює з простим припущенням, що не всі пікселі потрібні для ідентифікації деяких функцій із зображення.

Для задачі класифікації необхідно визначити межі/ребра всередині зображень, щоб класифікувати їх в одній із згаданих категорій.

Згорткові шари є основними будівельними блоками, які використовуються в згорткових нейронних мережах.

Згортка – це просте застосування фільтра до входу, що призводить до активації. Повторне застосування одного й того ж фільтра до входу призводить до створення карти активацій, яка називається картою функцій, яка вказує розташування та силу виявленої функції у вхідних даних, наприклад зображення.

Нововведенням згорткових нейронних мереж є здатність автоматично вивчати велику кількість паралельних фільтрів, специфічних для навчального набору даних, з урахуванням обмежень конкретної проблеми прогнозного моделювання, наприклад, класифікації зображень. Результатом є дуже специфічні функції, які можна виявити в будь-якому місці на вхідних зображеннях [16, 19].

Щоб усвідомити тонкощі CNN, потрібно зрозуміти, як працюють згортки. Можна взяти зображення, представлене у вигляді матриці значень 5×5, причому кожна клітинка представляє один піксель. Потім взяти матрицю

3×3 і перемістити вікно 3×3 навколо зображення. У кожній позиції, відвідування матриці 3×3 на зображенні, матриця множиться на значення в поточній позиції зображення.

Інакше, згортка працює так, як показано на рисунках 1.1, 1.2.

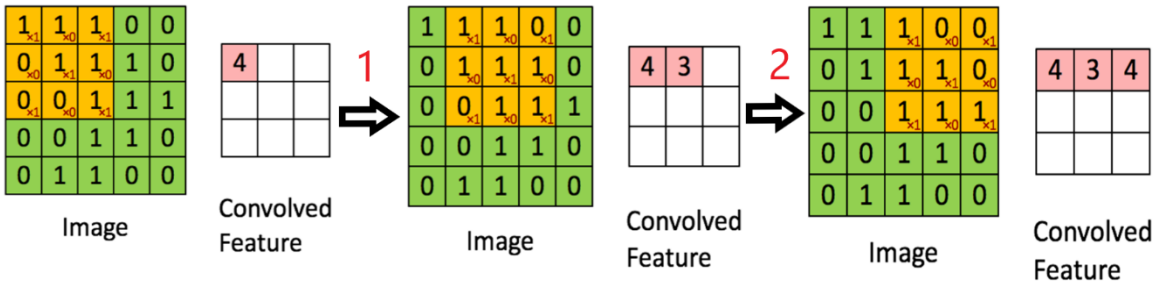

Рисунок 1.1 – Принцип роботи згортки, кроки 1–3

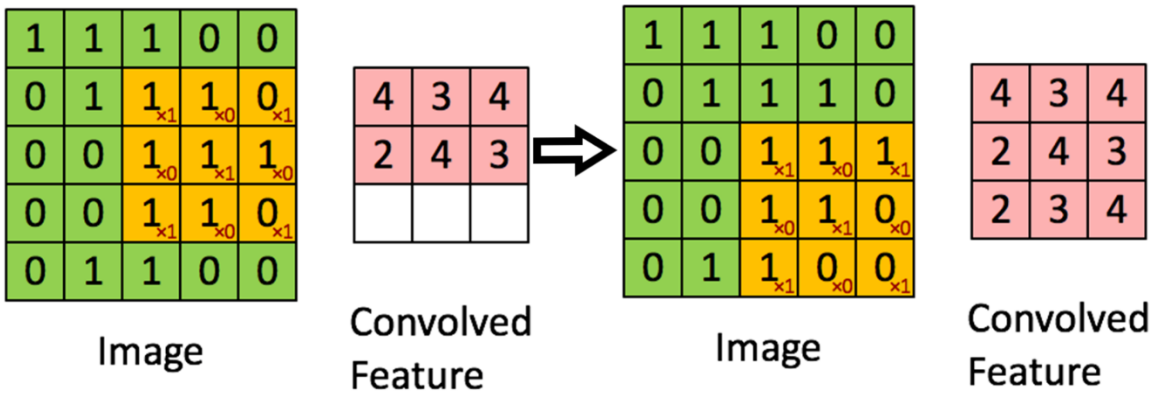

Рисунок 1.2 – Принцип роботи згортки, кроки 6 та 9

Вікно, яке переміщується, називається ядром. Відстань, на яку щоразу переміщується вікно, називається кроком.

Метою згорткового шару є фільтрація. Переміщуючись по зображенню, відбувається перевірка шаблонів в цій частині зображення. Це працює завдяки фільтрам, стекам ваг, представленим у вигляді вектора, які помножуються на значення, що виводяться за допомогою згортки.

Типова архітектура CNN включає наступні компоненти (рис. 1.3).

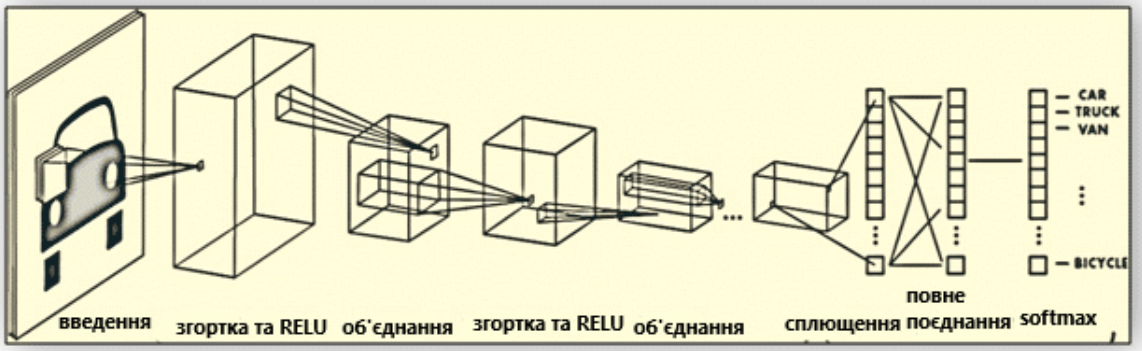

Рисунок 1.3 – Типова архітектура CNN

Об’єднання в пул працює подібно до згортки, різниця полягає у функції, яка застосовується до ядра, та у тому, що вікно зображення не є лінійним.

RELU – це функція активації, яка збиває значення в діапазон, як правило, [0,1] або [-1,1] [20].

Softmax – це ймовірнісна функція, яка дозволяє нам виразити наші вхідні дані у вигляді дискретного розподілу ймовірностей. Для обробки зображень набір функцій є суті кожного пікселя, створеного зображенням [1].

1.4 Постановка задачі

Таким чином, використання CNN для підвищення ефективності є актуальним завданням для обробки і розпізнавання зображень. Тому ставиться завдання розробки алгоритму розпізнавання образів, з використанням бібліотек OpenCV та TensorFlow.

Об’єктом роботи є послідовність різноракурсних зображень об’єктів.

Метою роботи є розробка методів, що базуються на використанні нейронної мережі згортки та попередній обробці зображень.

Для досягнення мети необхідно вирішити такі завдання:

– провести аналіз існуючих методів розпізнавання об’єктів на зображеннях;

– розробити алгоритм попередньої обробки зображень;

– реалізувати алгоритм розпізнавання, за допомогою бібліотек OpenCV та TensorFlow, на зображеннях, отриманих в результаті попередньої обробки;

– реалізувати комп’ютерну модель для знаходження кораблів на супутникових знімках.

2 АНАЛІЗ ВИКОРИСТАНИХ МЕТОДІВ КЛАСИФІКАЦІХ ОБРАЗІВ НА ЗОБРАЖЕННЯХ

2.1 Модуль попередньої обробки зображень

Кожне велике супутникове зображення має бути попередньо оброблено, перш ніж воно буде використано як вхідні дані для подальшого алгоритму виявлення об’єктів. Основною метою етапу попередньої обробки є підвищення точності та прискорення виконання завдання виявлення та класифікації зображень судна. Цей крок допомагає алгоритму виявлення об’єктів виводити менше пропозицій регіону, а також мінімізує кількість класифікацій, що виконуються в мережі класифікації об’єктів [8-10].

Оскільки зображення існують у різних форматах, наприклад, природні, штучні, у відтінках сірого тощо, ми повинні взяти це до уваги та стандартизувати їх, перш ніж передавати їх у нейронну мережу.

Існують наступні методи попередньої обробки зображень:

- перетворення відтінків сірого;
- нормалізація;
- збільшення даних;
- стандартизація зображення.

Перш за все зображення змінюються в градаціях сірого і вводяться у функцію, яка включає порогове значення відтінків сірого, щоб перетворити кожен піксель у зображенні на значення 0 або 255. Якщо значення дорівнює або перевищує заданий поріг, його значення в пікселях масштабується до 255, інакше воно встановлюється на 0 [15].

У якості засобу реалізації методу було обрано бібліотеку OpenCV для виконання всіх завдань попередньої обробки зображень. OpenCV зчитує дані з суміжної ділянки пам’яті. Для читання та запису даних зображення використовується формат HDF5.

Формат HDF5 можна розглядати як файлову систему, що міститься та описується в одному файлі. Однак у файлі HDF5 те, що можна назвати «каталогами» або «папками» на персональних комп'ютерах, називається групами, а те, що називається файлами – наборами даних [1].

Даний формат дозволяє зберігати величезну кількість числових даних і легко маніпулювати цими даними з NumPy. Наприклад, можна розділити багатотерабайтні набори даних, що зберігаються на диску, як ніби це справжні масиви NumPy. Тисячі наборів даних можна зберігати в одному файлі, класифікувати та позначати тегами.

H5py використовує прості метафори NumPy і Python, як-от словник і синтаксис масиву NumPy. Наприклад, можна перебирати набори даних у файлі або перевірити атрибути .shape або .dtype наборів даних [2].

Для застосованого випадку використання зображення будуть зберігатися у форматі HDF5, упорядковуючи у різні папки, залежно від типу та категорії, до якої належить зображення.

Для створення алгоритму використано бібліотеку OpenCV задля зміни розміру зображень і створення з них векторів ознак, що може бути досягнуто шляхом перетворення даних зображення в NumPy масиви чисел.

OpenCV, бібліотека комп'ютерного зору Python, використовується для зменшення шуму та покращення зображення шляхом закриття невеликих отворів всередині об'єктів переднього плану за допомогою серії операцій розширення та розмивання разом із кількома квадратними розмірами ядра, що наглядно показано на рисунку 2.1.

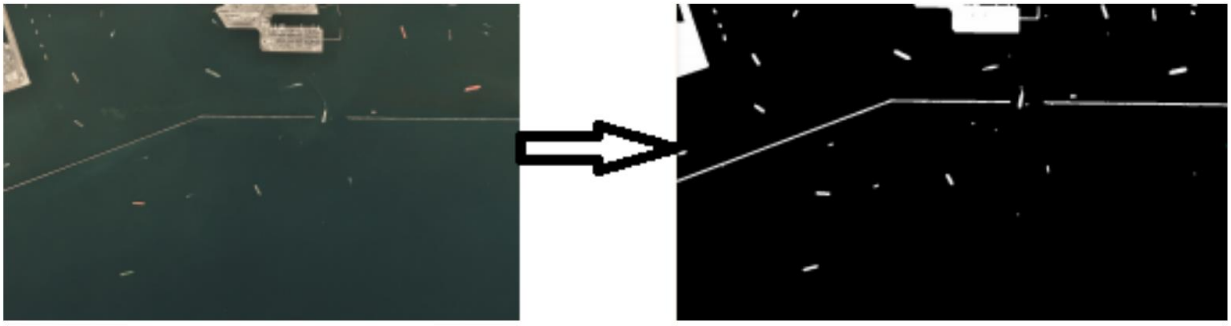

Рисунок 2.1 – Приклад попередньої обробки зображення

Потім зображення нормалізується шляхом ділення всіх значень пікселів на 255, так що матриця зображення представляє набір значень з 0 і 1.

2.2 Метод розпізнавання об’єктів

Метод розпізнавання об’єктів здійснює пошук у попередньо обробленому зображенні та виводить список обмежувальних рамок, у яких, на його думку, міститься об’єкт (рис. 2.2). Ці обмежувальні рамки потім масштабуються до ідеального квадратного розміру, намагаючись утримати об’єкти в центрі поля. Це робиться для того, щоб уникнути втрати будь-якої інформації про розміри, коли вони зменшуються для використання в якості вхідних даних для CNN [21-23].

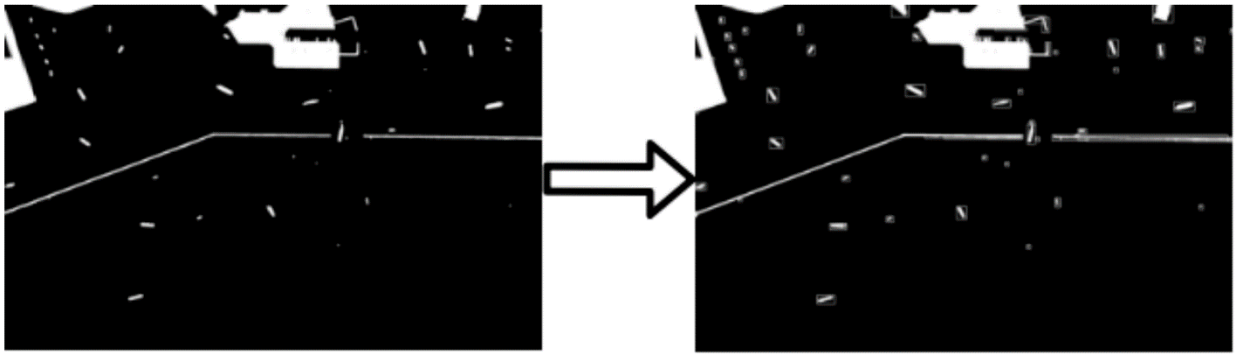

Рисунок 2.2 – Приклад розпізнавання образів на зображенні

Через відсутність надійного алгоритму виявлення об’єктів і пропозиції регіону на супутникових знімках, було прийняте рішення створити більш ефективний аналог (рис. 2.3).

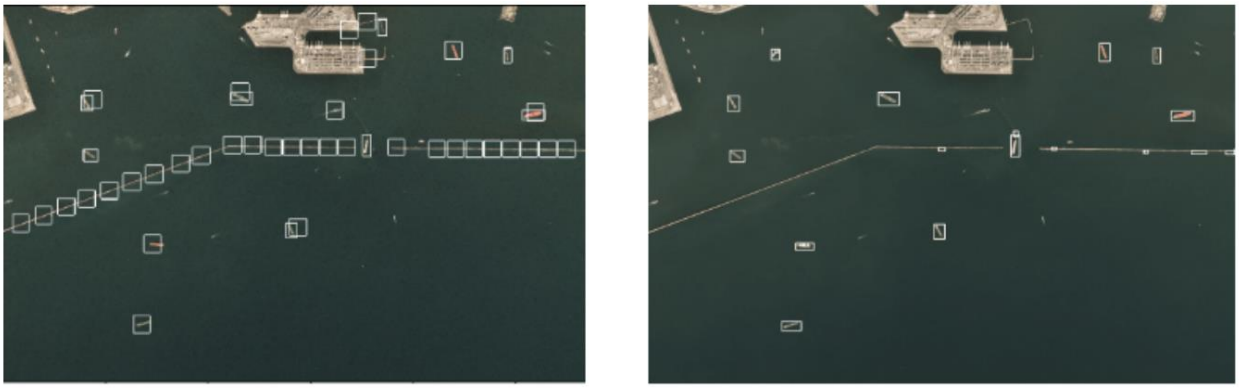

Рисунок 2.3 – Порівняння ефективності створеного алгоритму з алгоритмом розпізнавання Kaggle

Метод розпочинає роботу з того, що спочатку отримує нормалізовану матрицю зображення та переміщує вікно 100×100 зі 100 кроками по зображенню. На кожній ітерації обчислюється загальне значення ваги вікна, щоб побачити, чи воно перевищує поріг активації 3000. Якщо вага більше або дорівнює порогу, обчислюється сума кожного рядка у вікні 100×100 щоб визначити розташування пікселя, для початку пошуку глибини алгоритмом, в іншому випадку він ігнорує розташування та продовжує свої ітерації [24, 25].

Створений модифікований метод пошуку в глибину намагається знайти лише краєві пікселі, а не весь об’єкт. Як тільки він визначить, що об’єкт правильно окреслив, метод реєструє початкові (x, y) і кінцеві координати x. Далі координати використовуються для визначення загальної ваги об’єкта в створеному полі та порівнюються з пороговим значенням об’єкта 8000, щоб переконатися, що він не є частиною основної землі. Якщо вага менше або дорівнює порогу об’єкта, координати зберігаються, а якщо більше, координати викидаються (рис. 2.4). Після будь-якого сценарію область в межах координат затемнюється, щоб запобігти зайвій процедурі.

Нарешті, список усіх збережених координат масштабується і використовується для обрізання пропозицій зображення з оригінального необробленого супутникового зображення та вводиться в класифікатор CNN.

```
Initialize all variable

MAXXAXIS = imageMatrix[y-Axis]
MAXYAXIS = imageMatrix[x-Axis]

while startY < MAXYAXIS
        find endY
    while(starX < MAXYAXIS)
        find endX
        find weight
        if(weight < activationThreshhold)
                Add findObjectList to ObjectProposalList
        startX = endX
    startX = 0
    endx = 0
    startY = endY

return ObjectProposalList
```

Рисунок 2.4 – Псевдокод алгоритму

2.3 Класифікація об’єктів нейронною мережею згортки

2.3.1 Біологічні та штучні нейронні мережі

Одна з найпростіших моделей штучних нейронів полягає у відображенні вхідних даних за допомогою зваженої суми, яка згодом перетворюється на сигмовидну нелінійність відповідно до штучних нейронів.

$$a_i = b_i + \sum_{j=1}^{d} w_{i,j} x_j, \tag{2.1}$$

$$y_i = \sigma(a_i) = \frac{1}{1+e^{-a_i}}. \tag{2.2}$$

Функція сквош $\sigma(\cdot)$ сприяє поведінці, подібній до прийняття рішень, оскільки $y_i \to 1$ або $y_i \to 0$ активація розходиться ($a_i \to \pm\infty$). Тут i позначає загальний нейрон, поки $w_{i,j}$ – вага, пов’язана із з’єднанням від входу j до нейрона i [5].

Це надає можливість використовувати цей будівельний блок для побудови нейронних мереж, які обчислюють складні функції, складаючи багато нейронів.

Серед можливих способів об'єднання нейронів архітектура багатошарової нейронної мережі (Multilayered Neural Network – MLN) є однією з найпопулярніших, оскільки вона використовується у вражаючій кількості різних програмних засобів.

Ідея полягає в тому, що нейрони згруповані в шари, і що з'єднують тільки нейрони від нижнього до верхнього шарів. Отже, з'єднання j з i відбувається тоді і тільки тоді, коли $l(j) < l(i)$, де $l(k)$ позначає шар, якому належить загальна одиниця k.

Розрізняють вхідний шар $\mathcal{I}$, приховані шари $\mathcal{H}$, і вихідний шар $\mathcal{O}$. Багатошарові мережі можуть мати більше одного прихованого шару. Обчислення відбувається за схемою трубопроводів, де дані, доступні на вході, керують обчислювальним потоком. Спочатку відбувається обробка прихованих блоків, а потім, коли їх активація доступна, вони поширюються на вихід [21-23].

Нейронна мережа отримує, як вхідні дані, так і вихід модуля попередньої обробки $x = \pi(e) \epsilon \mathbb{R}^{25}$. Мережа містить 15 прихованих нейронів і 10 вихідних нейронів. Її завдання полягає в тому, щоб забезпечити вихід, який максимально наближений до цілі, зазначеної на рисунку 2.5.

Це демонструє, що, коли мережа отримує вхідний шаблон «2», очікується, що вихідний сигнал буде максимально близьким до цілі, зазначеної 1 на вершині нейрона 43 (рис. 2.5).

Тут передбачалося одноразове кодування, так як тільки мета, що відповідає класу «2», досягнута, тоді як усі інші встановлені на нуль. Вхідний символ представляється на вхідному шарі шляхом сканування відповідного рядка зображення, а потім пересилається на прихований шар, де очікується, що мережа створить функції, корисні для процесу розпізнавання.

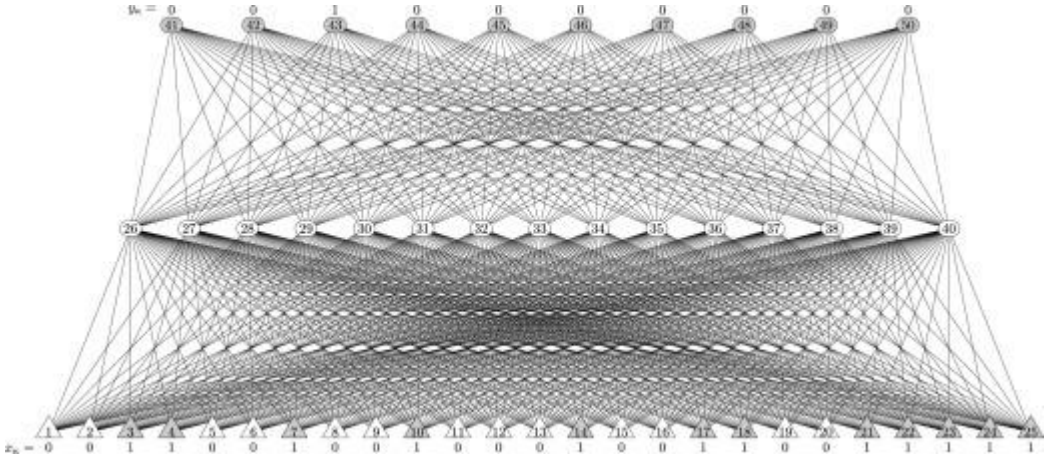

Рисунок 2.5 – MLP, призначений для розпізнавання рукописних символів

Очікується, що вибрані 15 прихованих одиниць відображатимуть відмінні властивості різних класів. Інтерпретація людиною подібних ознак призводить до характеристики геометричних і топологічних властивостей. Площа, периметр, а також елементи з геометрії мас, допомагають розрізняти символи. Барицентр і імпульс різного порядку виявляють хороші властивості шаблонів, але особливості можуть бути пов’язані з людською інтуїцією щодо того, що характеризує різні класи, наприклад, округлість і кількість отворів є додатковими характеристиками.

Є також деякі ключові моменти, такі як кути та хрестики, які сильно характеризують певний клас (наприклад, хрест класу «8»). Цікаво, що нейрони прихованого шару можна розглядати як детектори ознак, але зрозуміло, що постає два фундаментальні питання: які функції, очікувано, що MLP виявить? За умови, що MLP може виділити певний набір ознак, як можна визначити вагу прихованих нейронів, які виконують таке виявлення ознак?

По-перше, оскільки метою цієї нейронної мережі є класифікація символів, це може виявитися не так, що вищезгадані функції є найкращими для отримання в рамках MLP .

По-друге, можна відразу зрозуміти ідею навчання, яке полягає в правильному виборі ваг, таким чином, щоб звести до мінімуму помилку щодо цілей. Таким чином, набір рукописних символів разом із відповідними цілями можна використовувати для підбору даних.

Та насправді це особливий вид навчання на прикладі схеми, яка ґрунтується на інформації, наданій оракулом, який називають супервізором. Хоча загальна мета навчання полягає в тому, щоб забезпечити відповідну класифікацію, виявлення ваг нейронів призводить до побудови проміжних прихованих уявлень, які представляють особливості шаблону.

На рисунку 2.5 представлений шаблон розпізнається належним чином. Це випадок, коли очікується, що агент отримає винагороду, тоді як у випадку помилки агент буде покараний. Ця метафора кнута та пряника лежить в основі більшості алгоритмів машинного навчання.

MLN також називають багатошаровими персептронами (multilayered perceptrons – MLP) [5].

2.3.2 Архітектура створеної нейронної мережі згортки

Використана архітектура CNN була заснована на VGG-16. Оригінальна архітектура VGG-16 не може бути використана, оскільки розмір навчального зображення становить лише 80×80 пікселів, тоді як в оригінальній архітектурі є 5 шарів об’єднання [17]. Це створює проблему, оскільки 80 не можна зменшити лише вдвічі в 4 рази. В результаті один із шарів об’єднання необхідно видалити. Остаточну функцію класифікації також потрібно було змінити на сигмовидну, оскільки це проблема бінарної класифікації. Для перевірки всіх п’яти можливих випадків використовувалася 5-кратна перевірка.

У першому випадку був видалений перший шар об’єднання, у другому випадку видалено другий шар об’єднання тощо. Моделі навчалися за 100 епох (кількість проходів всього набору навчальних даних, який завершив алгоритм машинного навчання) для кожної складки. Результати, наведені нижче в таблиці 2.1, показують, що оптимальним шаром об’єднання для видалення є перший шар об’єднання.

Таблиця 2.1 – Середні втрати та точність для 5 тестових випадків

Тестовий випадок	**Середні втрати**	**Середня точність**
1	0,1206	98,20%
2	0,1562	97,84%
3	0,1493	97,80%
4	0,1233	98,12%
5	0,1269	98,16%

Модель, використовуючи оптимальну архітектуру з таблиці 2.1, була потім навчена на всіх 2500 навчальних зображеннях за 100 епох.

Ця модель була використана для класифікації запропонованих об’єктів за допомогою створеного алгоритму виявлення об’єктів на супутниковому знімку. Об’єкти, які були класифіковані моделлю як «кораблі», потім були обрамлені, і результат показаний на рисунку 2.6.

Рисунок 2.6 – Тестове зображення для першої тренованої моделі

У цьому тестовому випадку лише сім з десяти кораблів були правильно класифіковані як «корабель». Крім того, є частина зображення, яку неправильно класифікували як «корабель» (хибнопозитивні).

2.4 Розширення даних для навчання

Оскільки поточною проблемою, є обмежений розмір даних, його необхідно збільшити за домогою розширення даних [26]. Дані були доповнені за допомогою класу Keras – ImageDataGenerator. Генератор був ініціалізований, щоб включати використання діапазону обертання 45 градусів, діапазону зсуву ширини 0,2, діапазону зсуву висоти 0,2 і діапазону масштабування 0,2 і включає як горизонтальні, так і вертикальні перевороти зображень. Цей генератор доповнює зображення пакетами, які були ініціалізовані до 50, оскільки модель навчається.

Перша модель, яка була навчена за допомогою цього доповнювача даних, навчалася лише на 100 партіях за епоху, тому модель навчалася на 5000 зображеннях за епоху; більше того, дані доповнюються на льоту в кожній партії, тому вони тренуються на різних розширених зображеннях у кожну епоху. Це значно підвищило точність загального тестового зображення. Ця модель ідеально класифікувала зображення з рисунку 2.6. Однак, як видно на рисунку 2.7, під час проходження іншого тестового зображення в нижньому лівому куті зображення було хибнопозитивне.

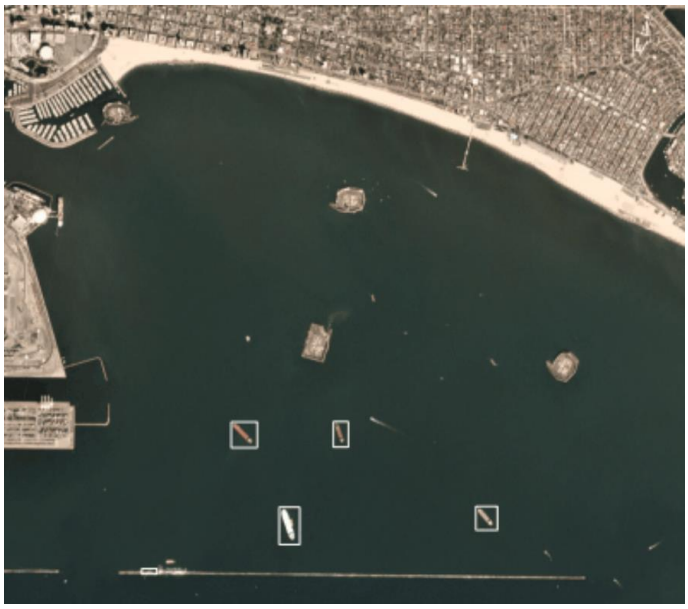

Рисунок 2.7 – Тестування для першої тренованої моделі зі збільшеним обсягом даних

Наступну модель навчено за допомогою того ж доповнювача даних, але кількість пакетів, що використовуються на епоху, було збільшено до 200, тому модель навчалася на 10 000 зображень за епоху. Ця модель ідеально класифікувала всі кораблі на обох тестових зображеннях без жодних помилкових спрацьовувань.

3 КОМП'ЮТЕРНА МОДЕЛЬ КЛАСИФІКАЦІЇ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ

3.1 Обґрунтування вибору середовища програмної реалізації

У якості середовища програмної реалізації мовою програмування Python було обрано Microsoft Visual Studio 19.

Microsoft Visual Studio – це IDE, створена Microsoft, і яка використовується для різних типів розробки програмного забезпечення, наприклад комп'ютерних програм, веб-сайтів, веб-програм, веб-сервісів та мобільних додатків. Він містить інструменти збірки, компілятори та інші функції для полегшення процесу розробки програмного забезпечення [6].

Visual Studio забезпечує підтримку мови Python з відкритим вихідним кодом через робочі навантаження Python Development і Data Science і безкоштовне розширення Python Tools for Visual Studio.

Вікно середовищ Python Visual Studio (рис. 3.1) надає єдине місце для керування глобальними середовищами Python, середовищами conda та віртуальними середовищами. Visual Studio автоматично виявляє інсталяції Python у стандартних місцях і дозволяє налаштувати власні інсталяції. З кожним середовищем можна легко керувати пакетами, відкривати інтерактивне вікно для цього середовища та отримувати доступ до папок середовища.

Для запуску Python можна використовувати як внутрішні інтерактивні засоби Visual Studio, так і окреме командне вікно в папці вибраного середовища з PowerShell, з якого можна запустити будь-який скрипт Python [27-30].

Visual Studio надає редактор Python, що включає забарвлення синтаксису, автозаповнення для коду та бібліотек, форматування коду, довідку щодо підпису, рефакторинг та підказки. Visual Studio також надає функції, такі як перегляд класу, перехід до визначення, пошук всіх посилань та фрагментів

коду. Пряма інтеграція з інтерактивним вікном допомагає швидко розробити код Python, який уже збережений у файлі [6].

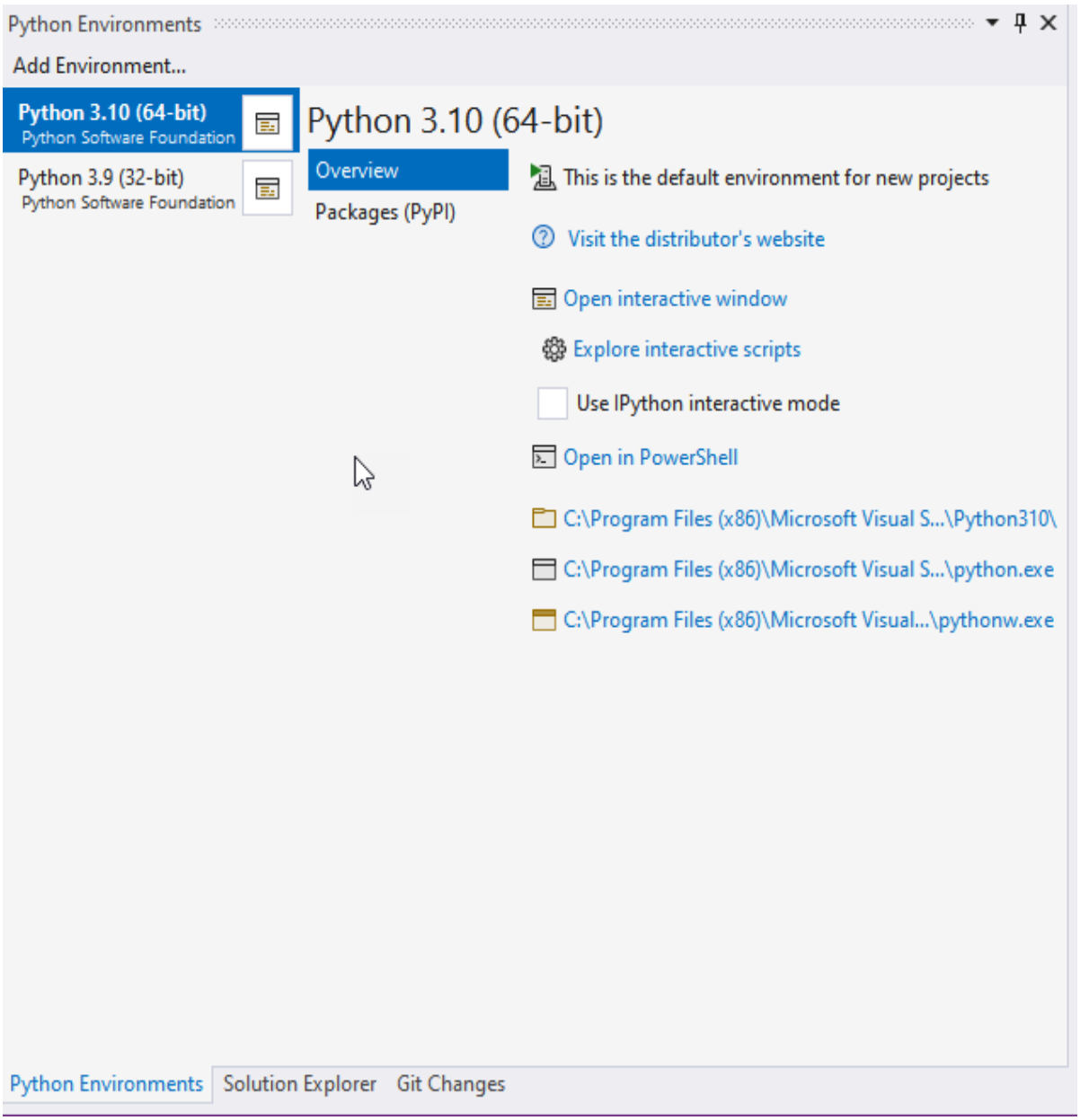

Рисунок 3.1 – Вікно середовищ Python

Visual Studio також підтримує IPython/Jupyter у REPL, включаючи вбудовані графіки, .NET і Windows Presentation Foundation (WPF).

Таким чином, вибір даного середовища програмної реалізації зумовлений великим переліком функцій та особливостей, що допомагають покращити досвід створення програмного продукту мовою Python.

3.2 Програмна реалізація

Застосунок має підготовані дані, отримані з Kaggle, що містять загалом 2800 зображеннь RGB, позначені як «корабель», або класифікація «не корабель».

PreProcessingModule.py – основний файл, який використовується для попередньої обробки зображень. Він забезпечує наступні методи:

– створення обмежувальної рамки навколо певних місць на зображенні;

– зменшення шуму зображення за допомогою методів ядра OpenCV;

– масштабування зображення таким чином, щоб їх розмірні характеристики не були втрачені;

– нормалізація даних зображення, для надання простіших даних як вхідних для алгоритму виявлення об’єктів;

– двійкове масштабування сірого з використанням порогового значення для надання простіших даних як вхідних для алгоритму виявлення об’єктів.

ObjectDetection.py – основний файл, який використовується для виведення пропозицій регіонів із великих супутникових знімків. Він виводить список пропозицій регіону, які виконуються на основі попередньо оброблених даних.

Основні файли, які використовувалися для створення, навчання, тестування та збереження CNN:

– CNN.py створений для розміщення усіх функцій, які маніпулюють самою моделлю CNN;

– Tuning_CNN.py – для навчання та тестування багатьох можливих архітектур і гіперпараметрів CNN;

– Helper.py працює з необробленими даними зображення, такими, як довільний розподіл даних на дані навчання/перевірки та тестування, та завантаження заданого файлу даних;

– Constants.py містить шляхи до збережених різних моделей CNN, а також короткий опис їх відмінностей.

3.3 Тестування розробленої моделі

Щоб викреслити всі можливості помилкового розпізнавання, оригінальна архітектура VGG-16 з розміром входу 224×244 була протестована за допомогою створеного алгоритму. Набір навчальних даних та пропозиції об’єктів були масштабовані до 224×224. Усі моделі використовували попередньо навчені ваги для згорткових шарів, і ці шари також були заморожені, щоб значення ваг не змінилися. Повністю пов’язані шари потім були навчені з нуля разом із сигмовидним класифікатором.

Шість різних моделей були навчені та перевірені, щоб знайти оптимальні гіперпараметри для навчання моделі за допомогою навчання з перенесенням. Три моделі були навчені з використанням вихідного набору навчальних даних, тоді як інші три були навчені за допомогою розширення даних. У таблиці 3.1 наведено результати тестування.

Таблиця 3.1 – Втрати перевірки та значення точності від тестування

Набір гіперпараметрів	**Без розширення даних**		**З розширеними даними**	
	Втрати	**Точність**	**Втрати**	**Точність**
1	0,1450	98,60%	3,7394	76,80%
2	0,0601	97,80%	0,0395	98,60%
3	0,1274	95,80%	0,0575	97,00%

Перший набір гіперпараметрів такий самий, як і гіперпараметри за замовчуванням з розміром епохи 10. Другий набір має швидкість навчання $1 \cdot 10^{-4}$, швидкість загасання 0,75 і розмір епохи 10. Третій набір має швидкість навчання $1 \cdot 10^{-5}$, швидкість розпаду 0,65 і розмір епохи 30.

Більшість значень у таблиці 3.1 виглядають пристойно, не враховуючи перший набір з розширеними даними. Проблема не виникає, доки ці моделі не

будуть протестовані з використанням пропозицій об’єктів від більших супутникових знімків. У кожному тестовому зображенні в середньому є близько п’яти помилкових позитивних результатів, наприклад, на рисунку 3.2.

Рисунок 3.2 – Тестова модель, підготована на другому наборі із розширеними даними

Крім того, час виконання між модифікованою архітектурою VGG-16 та оригінальною архітектурою VGG-16 під час запуску алгоритму є досить значним.

У середньому моделі, які навчаються з трансферним навчанням, приблизно на 30% повільніше, ніж моделі з модифікованою архітектурою VGG-16.

ВИСНОВКИ

У рамках кваліфікаційної роботи був розроблений і реалізований метод розпізнавання та класифікації об’єктів за допомогою бібліотек OpenCV та TensorFlow, на зображеннях, отриманих в результаті попередньої обробки.

У якості об’єкту роботи було обрано послідовність різноракурсних супутникових знімків з об’єктами типу «корабель»/«некорабель».

Результатом проведеної роботи є виконання всіх поставлених цілей:

– проведений аналіз існуючих методів розпізнавання об’єктів на зображеннях;

– створений алгоритм попередньої обробки зображень;

– створений алгоритм розпізнавання об’єктів на зображеннях, отриманих в результаті попередньої обробки;

– реалізація комп’ютерної моделі для знаходження кораблів на супутникових знімках;

– проведене тестування на експериментальних даних.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Image Processing using OpenCV, CNN, and Keras backed by Tensor Flow. URL:https://medium.com/analytics-vidhya/image-processing-using-opencv-cnn-and-keras-backed-by-tensor-flow-c9adf22bb271 (дата звернення 19.05.2022).

2. HDF5 for Python. URL:https://www.h5py.org (дата звернення 20.05.2022).

3. How to Train Your Own Object Detector Using TensorFlow Object Detection API. URL:https://neptune.ai/blog/how-to-train-your-own-object-detector-using-tensorflow-object-detection-api (дата звернення 20.05.2022).

4. Object Detection Tutorial using TensorFlow. URL:https://www.edureka.co/blog/tensorflow-object-detection-tutorial (дата звернення 20.05.2022).

5. Gori, M. (2017). Machine learning: A constraint-based approach. Morgan Kaufmann.

6. Python support in Visual Studio on Windows. URL:https://docs.microsoft.com/en-us/visualstudio/python/overview-of-python-tools-for-visual-studio?view=vs-2022 (дата звернення 25.05.2022).

7. Гороховатський, В.О., Гадецька, С.В. (2020) Статистичне оброблення та аналіз даних у структурних методах класифікації зображень (монографія), Харків, ФОП Панов А.Н., 128 с.

8. Гороховатський, В.О., Творошенко, І.С. (2021) Методи інтелектуального аналізу та оброблення даних: навч. посібник. Харків: ХНУРЕ.

9. Mashtalir, S., Stolbovyi, M., & Mikhnova, O. (2019). Multidimensional sequence clustering with adaptive iterative dynamic time warping. International Journal of Computing, 18(1), 53-59.

10. Егоров, А. С., & Машталир, С. В. (2005). Сравнительный анализ методов морфологической нормализации. Радиоэлектроника и информатика, (4), 90-95.

11. Mashtalir, S., & Mikhnova, O. (2017). Detecting significant changes in image sequences. In Multimedia Forensics and Security (pp. 161-191). Springer, Cham.

12. Bodyanskiy, Y., Grimm, P., Mashtalir, S., & Vinarski, V. (2010, July). Fast training of neural networks for image compression. In Industrial Conference on Data Mining (pp. 165-173). Springer, Berlin, Heidelberg.

13. Custom Object Detection using TensorFlow from Scratch. URL:https://towardsdatascience.com/custom-object-detection-using-tensorflow-from-scratch-e61da2e10087 (дата звернення 24.05.2022).

14. Preprocessing Module - an overview. URL:https://www.sciencedirect.com/topics/computer-science/preprocessing-module (дата звернення 24.05.2022).

15. CNN custom image classification based on tensorflow + opencv. URL: https://developpaper.com/cnn-custom-image-classification-based-on-tensorflow-opencv/ (дата звернення 24.05.2022).

16. Data Preprocessing and Network Building in CNN. URL:https://towardsdatascience.com/data-preprocessing-and-network-building-in-cnn-15624ef3a28b (дата звернення 26.05.2022).

17. Используйте VGG16 для реализации распознавания и классификации изображений и используйте VGG 19 для реализации передачи художественного стиля. URL:https://russianblogs.com/article/5912670330/ (дата звернення 27.05.2022).

18. Papageorgiou, C. P., Oren, M., & Poggio, T. (1998, January). A general framework for object detection. In Sixth International Conference on Computer Vision (IEEE Cat. No. 98CH36271) (pp. 555-562). IEEE.

19. Huang, R., Pedoeem, J., & Chen, C. (2018, December). YOLO-LITE: a real-time object detection algorithm optimized for non-GPU computers. In 2018 IEEE International Conference on Big Data (Big Data) (pp. 2503-2510). IEEE.

20. Zhiqiang, W., & Jun, L. (2017, July). A review of object detection based on convolutional neural network. In 2017 36th Chinese control conference (CCC) (pp. 11104-11109). IEEE.

21. Shin, H. C., Roth, H. R., Gao, M., Lu, L., Xu, Z., Nogues, I., ... & Summers, R. M. (2016). Deep convolutional neural networks for computer-aided detection: CNN architectures, dataset characteristics and transfer learning. IEEE transactions on medical imaging, 35(5), 1285-1298.

22. Chua, L. O., & Roska, T. (1993). The CNN paradigm. IEEE Transactions on Circuits and Systems I: Fundamental Theory and Applications, 40(3), 147-156.

23. Gross, R., & Brajovic, V. (2003, June). An image preprocessing algorithm for illumination invariant face recognition. In International Conference on Audio-and Video-Based Biometric Person Authentication (pp. 10-18). Springer, Berlin, Heidelberg.

24. Albright, J., & Alzheimer's Disease Neuroimaging Initiative. (2019). Forecasting the progression of Alzheimer's disease using neural networks and a novel preprocessing algorithm. Alzheimer's & Dementia: Translational Research & Clinical Interventions, 5, 483-491.

25. Bow, S. T. (2002). Pattern recognition and image preprocessing. CRC press.

26. Bhattacharyya, S. (2011). A brief survey of color image preprocessing and segmentation techniques. Journal of Pattern Recognition Research, 1(1), 120-129.

27. Rehman, A., & Saba, T. (2014). Neural networks for document image preprocessing: state of the art. Artificial Intelligence Review, 42(2), 253-273.

28. Agarap, A. F. (2018). Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.

29. Van Dyk, D. A., & Meng, X. L. (2001). The art of data augmentation. Journal of Computational and Graphical Statistics, 10(1), 1-50.

30. Raschka, S. (2015). Python machine learning. Packt publishing ltd.

31. Trahanias, P. E., & Venetsanopoulos, A. N. (1993). Vector directional filters-a new class of multichannel image processing filters. IEEE Transactions on Image Processing, 2(4), 528-534.

32. Machuca, R., & Phillips, K. (1983). Applications of vector fields to image processing. IEEE Transactions on Pattern Analysis and Machine Intelligence, (3), 316-329.

33. Trahanias, P. E., & Venetsanopoulos, A. N. (1993). Color edge detection using vector order statistics. IEEE Transactions on Image Processing, 2(2), 259-264.

34. Lucchese, L., & Mitra, S. K. (2004). A new class of chromatic filters for color image processing. Theory and applications. IEEE Transactions on Image Processing, 13(4), 534-548.

35. Makrogiannis, S., Economou, G., Fotopoulos, S., & Bourbakis, N. G. (2005). Segmentation of color images using multiscale clustering and graph theoretic region synthesis. IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, 35(2), 224-238.

36. Tao, W., Jin, H., & Zhang, Y. (2007). Color image segmentation based on mean shift and normalized cuts. IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 37(5), 1382-1389.

37. Shi, J., & Malik, J. (2000). Normalized cuts and image segmentation. IEEE Transactions on pattern analysis and machine intelligence, 22(8), 888-905.

38. Zhang, Y., Brady, M., & Smith, S. (2001). Segmentation of brain MR images through a hidden Markov random field model and the expectation-maximization algorithm. IEEE transactions on medical imaging, 20(1), 45-57.

39. Penalver, A., Escolano, F., & Sáez, J. M. (2006). Color image segmentation through unsupervised Gaussian mixture models. In Advances in Artificial Intelligence-IBERAMIA-SBIA 2006 (pp. 149-158). Springer, Berlin, Heidelberg.

40. Howarth, J. W., Bakker, H. H., & Flemmer, R. C. (2009, February). Feature-based object recognition. In 2009 4th International Conference on Autonomous Robots and Agents (pp. 375-379). IEEE.

41. Breaking Down Facial Recognition: The Viola-Jones Algorithm. URL:https://towardsdatascience.com/the-intuition-behind-facial-detection-the-viola-jones-algorithm-29d9106b6999 (дата звернення 27.05.2022).

42. HOG Descriptor. URL:https://www.intel.com/content/www/us/en/develop/documentation/ipp-dev-reference/top/volume-2-image-processing/computer-vision/feature-detection-functions/histogram-of-oriented-gradients-hog-descriptor.html (дата звернення 27.05.2022).