Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
(повне наймснування інституту, назва факультету (відділення))

Кафедра комп'ютерних наук
(повна назва кафедри (предметної, циклової комісії))

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Інформаційна технологія розпізнавання об'єктів на зображеннях з використанням згорткової нейронної мережі»

Виконав: студент 2-го курсу, групи 2КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
_____ Мальцев С.В.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
_____ Колесницький О.К.
(прізвище та ініціали)
«15» 12 2022 р.

Опонент: к.т.н., професор каф. КСУ
_____ Биков М. М.
(прізвище та ініціали)
«15» 12 2022 р.

Допущено до захисту
Завідувач кафедри КН
_____ д.т.н., проф. Яровий А.А.
(прізвище та ініціали)
«16» 12 2022 р.

Вінниця ВНТУ - 2022 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 122 «Комп'ютерні науки»
Освітньо-професійна програма – «Системи штучного інтелекту»

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
Д.т.н., проф. Яровий А.А.
14 09 2022 року

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Мальцеву Сергію Васильовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна технологія розпізнавання об'єктів на зображеннях з використанням згорткової нейронної мережі

керівник роботи к.т.н., доцент кафедри КН Колесницький О. К.
затверджені наказом вищого навчального закладу від "14" 09 2022 року №203
2. Строк подання студентом роботи 18 листопада 2022 року
3. Вихідні дані до роботи:
   Вхідні дані – формат вхідних зображень - jpg, jpeg або png, розмірність вхідних зображень – не менше 400x400, кількість класів розпізнавання – 80, обсяг навчальної вибірки – не менше 1000 зображень, обсяг тестової вибірки – не менше 100 зображень, мова програмування – об'єктно-орієнтована.
4. Зміст текстової частини:
   Вступ, аналіз предметної області розпізнавання об'єктів на зображеннях, розробка інформаційної технології розпізнавання об'єктів на зображеннях, програмна реалізація інформаційної технології розпізнавання об'єктів на зображеннях, економічна частина, висновки, перелік використаних джерел, додатки
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
   Алгоритм роботи програми розпізнавання об'єктів на зображеннях, структура нейронної мережі, UML діаграма класів програми розпізнавання об'єктів на зображеннях, робочі вікна програми розпізнавання об'єктів на зображеннях, результати тестування програми розпізнавання об'єктів на зображеннях та програми-аналога.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконан прийня
1-4	Колесницький О.К., к.т.н., доц. каф. КН	14.09.2022	14.12.2022
5	Бурєннікова Н. В., д. е. н., проф. каф. ЕПВМ	19.09.2022	12.12.20

7. Дата видачі завдання 14.09 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз сучасного рівня інформаційних технологій розпізнавання об'єктів на зображеннях. Постановка задач дослідження	14.09.2022р - - 1.10.2022р.
2	Побудова моделей розпізнавання об'єктів на зображеннях на основі нейронної мережі та функціонування нейронної мережі	2.10.2022р - - 16.10.2022р
3	Практичне застосування та оцінка ефективності розроблених моделей	17.10.2022р - - 02.11.2022р.
4	Підготовка економічної частини	03.11.2022р - - 21.11.2022р
5	Апробація та/або впровадження результатів дослідження	23.11.2022р - - 01.12.2022р.
6	Оформлення пояснювальної записки, графічного матеріалу та презентації	02.12.2022р - - 14.12.2022р

Студент _____ (підпис) Мальцев С.В.

Керівник роботи _____ (підпис) Колесницький О. К.

АНОТАЦІЯ

УДК 004.8

Мальцев С. В. Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі. Магістерська кваліфікаційна робота зі спеціальності 122 – комп’ютерні науки, освітня програма - комп’ютерні науки. Вінниця: ВНТУ, 2022. 109 с.

На укр. мові. Бібліогр.: 27 назв; рис.: 25; табл. 16.

Дана магістерська кваліфікаційна робота присвячена розробці програмного забезпечення для розпізнавання об’єктів на зображеннях на основі згорткової нейронної мережі. Були розглянуті та проаналізовані існуючі методи розпізнавання об’єктів на зображеннях, як найбільш перспективний, було обрано нейромережевий метод. Було проаналізовано різні парадигми штучних нейронних мереж та обґрунтовано вибір для даної задачі згорткової нейронної мережі. Було розроблено архітектуру обраного типу мережі. Було спроектовано програму розпізнавання об’єктів на зображеннях, написану мовою програмування C# у середовищі розробки Visual Studio з використанням технологій ML.NET і .NET Core. Розроблена програмна має вищу на 7,5%. достовірність розпізнавання об’єктів на зображеннях (95,4%), ніж аналогічна програма (87,9%)..

Графічна частина складається з 6 плакатів.

У економічному розділі визначено, що згідно узагальненого коефіцієнту конкурентоспроможності, науково-технічна розробка переважає існуючі аналоги приблизно в 2,06 рази. Термін окупності становить 0,56 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

Ключові слова: розпізнавання, об’єкти на зображеннях, згорткова нейронна мережа.

ABSTRACT

Maltsev S.V. Information technology of object recognition on images using a convolutional neural network. Master's thesis in the specialty 122 - computer science, educational program - computer science. Vinnytsia: VNTU, 2022. 109 p.

In Ukrainian language. Bibliogr .: 27 titles; fig . 25; table 16.

This master's thesis is dedicated to the development of software for image recognition on the basis of convolutional neural network. Existing methods of image recognition on images were considered and analyzed, and the neural network method was chosen as the most promising. Different paradigms of artificial neural networks were analyzed and the choice of a convolutional neural network for this problem was substantiated. The architecture of the selected network type was developed. An image object recognition program designed in the C # programming language in the Visual Studio development environment using ML.NET and .NET Core technologies was designed. The developed software is 7.5% higher. Accuracy of image recognition on images (95.4%) than similar program (87.9%).

The graphic part consists of 6 posters.

In the economic section, it is determined that according to the generalized coefficient of competitiveness, scientific and technical development exceeds existing analogues by approximately 2.06 times. The payback period is 0.56 years, which is less than 3 years, which indicates the commercial attractiveness of the scientific and technical development and may encourage a potential investor to finance the implementation of this development and its introduction to the market.

Keywords: recognition, objects on images, convolutional neural network.

ЗМІСТ

ВСТУП

Актуальність. У наш час роль інформаційних технологій у всіх сферах людської діяльності є надвисокою. Аналіз та обробка інформації є одними із найбільш пріоритетних завдань будь-якої сучасної установи від з приватної компанії, що виробляє продукцію та надає послуги до передових наукових лабораторій. Оскільки об’єми інформації сьогодні є надзвичайно великими, то постає задача автоматизації обробки цієї інформації.

Нині існує багато підходів до обробки візуальної інформації, тому що вона є найпоширенішою. Питання автоматизованої обробки цифрових зображень та відео стають ключовими, адже потребують розробки складніших алгоритмів та використання штучного інтелекту для отримання вагомих результатів. Сучасний світ все більше зацікавлений у таких технологіях як розпізнавання об’єктів.

Розпізнавання об'єктів – це перспективна технологія, що стоїть за розвиненими системами допомоги водіям, яка дозволяє автомобілям виявляти смуги руху або розпізнавати пішоходів для підвищення безпеки дорожнього руху. Розпізнавання об'єктів на зображеннях також важливе в таких сферах, як відеомоніторинг, виявлення аномалій, підрахунок відвідувачів, самокеровані автомобілі, робототехніка, системи пошуку зображень або розпізнавання людських обличь.

Актуальність та доцільність дослідження. Як правило, сучасні системи розпізнавання зображень є частиною хмарних сервісів, доступ до яких є платним, а алгоритми обробки зображень розроблені для більш загальних випадків, що є незадовільним у більшості ситуацій. Дана предметна область є досить актуальною у таких галузях як медичні зображення, фізика, вимірювання та контроль якості в процесах виробництва, нейробіологія тощо. Саме тому розробка власного програмного модуля розпізнавання об’єктів у зображеннях є актуальною та доцільною.

Зв'язок роботи з науковими програмами, планами, темами. Магістерська робота виконана відповідно до напрямку наукових досліджень кафедри комп'ютерних наук Вінницького національного технічного університету 22 К1 «Моделі, методи, технології та пристрої інтелектуальних інформаційних систем управління, економіки, навчання та комунікацій» та плану наукової та навчально-методичної роботи кафедри.

Мета і завдання досліджень. Метою магістерської кваліфікаційної роботи є підвищення достовірності розпізнавання об'єктів на зображеннях за рахунок використання попередньо натренованої згорткової нейронної мережі.

Для досягнення мети розробки необхідно виконати такі задачі:

- провести аналіз предметної області розпізнавання об'єктів на зображеннях;
- розглянути існуючі методи розпізнавання об'єктів на зображеннях та обрати й обґрунтувати вибір методу, який задовольняє мету даної магістерської кваліфікаційної роботи;
- обгрунтувати вибір нейронної мережі та проаналізувати математичну модель її роботи та навчання;
- сформулювати стадії інформаційної технології, розробити структуру та алгоритм роботи програмного засобу;
- виконати програмну реалізацію запропонованої інформаційної технології;
- провести тестування програмного продукту та виконати аналіз отриманих результатів.

Об'єкт дослідження – процес комп'ютеризованого розпізнавання об'єктів на зображеннях з використанням нейронних мереж.

Предмет дослідження – інформаційна технологія та програмні засоби розпізнавання об'єктів на зображеннях комп'ютерними засобами з використанням нейронних мереж та достовірність їх роботи.

Методи дослідження. У роботі використані наступні методи наукових досліджень: системного аналізу, теорії штучних нейронних мереж для

реалізації інформаційної технології, методи математичної статистики для розробки процесу розв'язання задачі нейромережевого розпізнавання об’єктів на зображеннях та обрахунків результатів експериментів із програмним засобом, об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів.

1. Набула подальшого розвитку інформаційна технологія нейромережевого розпізнавання об’єктів на зображеннях, яка відрізняється використанням згорткової нейронної мережі, що дозволило підвищити достовірність програмних засобів розпізнавання об’єктів на зображеннях.

Практичне значення одержаних результатів полягає в тому, що на основі проведених досліджень розроблено програмне забезпечення розпізнавання об’єктів на зображеннях.

Запропонована інформаційна технологія сприяє підвищенню достовірності програмних засобів розпізнавання об’єктів на зображеннях, зокрема:

- розроблено алгоритм роботи програмного забезпечення розпізнавання об’єктів на зображеннях;
- розроблено програмні засоби для розпізнавання об’єктів на зображеннях.

Достовірність теоретичних положень магістерської кваліфікаційної роботи підтверджується коректністю постановки завдання, коректністю використання математичного апарату методів дослідження, експериментальними дослідженнями тестування програмної реалізації інформаційної технології розпізнавання об’єктів на зображеннях. Адекватність розроблених математичних моделей підтверджується результатами експериментальних досліджень.

Особистий внесок здобувача. Усі результати, наведені у магістерській кваліфікаційній роботі, отримані самостійно. У працях, написаних у співавторстві, здобувачу належать: аналіз процесу розпізнавання об’єктів на

зображеннях та методів підвищення достовірності програмних засобів розпізнавання об’єктів на зображеннях [1].

Апробація результатів роботи. Результати роботи були апробовані на конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)», Вінниця, 15 листопада 2022 року - 12 травня 2023 року.

Публікації. За результатами досліджень опубліковано одні тези доповіді на науково-технічній конференції [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ

1.1 Постановка задачі розпізнавання об'єктів на зображеннях

Для розробки інформаційної технології розпізнавання об'єктів на зображеннях, було проаналізовано, з використанням літератури, особливості поставленої задачі з точки зору вимог до її розв'язання.

Вхідні дані до роботи: методи розпізнавання об'єктів у зображеннях; процес виділення ознак; алгоритми і методи штучного інтелекту; засоби комп'ютерного зору; графічний режим – TrueColor; тип нейронної мережі – згорткова нейронна мережа, використання мови програмування C#; вихідні дані для розпізнавання об'єктів – зображення у форматі .jpg, .jpeg або .png; попередньо натренована згорткова нейронна мережа YOLOv2 з можливістю розпізнавання 80 класів об'єктів; вихідні дані – кінцеве зображення з виділеними на ньому за допомогою розробленого програмного забезпечення класами об'єктів, їх назвою та достовірністю розпізнання. Виконувані функції кожного програмного модуля повинні бути зручними для розширення та відлагодження.

Отже, визначені вимоги до вхідних та вихідних даних, архітектури та виконуваних функцій програми, а також досяжних технічних параметрів. Виконання саме цих вимог дозволить розробити якісний та корисний програмний продукт.

1.2 Актуальність та галузі застосування розпізнавання об'єктів на зображеннях

У наш час системи комп'ютерного зору застосовуються у будь-яких сферах [2, 3]. Ця галузь характеризується як молода, всеохоплююча та

динамічна [4, 5]. З кінця 1970-х почалося інтенсивне дослідження цієї проблеми, коли комп'ютери стали спроможні обробляти великі набори даних, до яких відносяться зображення [6, 7]. Ці дослідження зазвичай стосувалися інших областей, а, отже, не існує стандартного формулювання проблематики комп'ютерного зору. Також немає стандартного формулювання того, як мають вирішуватися завдання комп’ютерного зору. Замість цього, існує низка методів для вирішення різних суворо визначених задач комп'ютерного зору, де методи зачасту залежать від завдань і не завжди можуть бути узагальнені для широкого кола застосування. Багато методів і засобів все ще знаходяться в стадії фундаментальних досліджень, але все більше їх число знаходять застосування в комерційних програмних продуктах, де вони складають частину більшої системи, яка вирішує складні завдання (наприклад, в галузі медичних зображень [8] або вимірювання і контролю якості в процесах виробництва). У більшості практичних застосувань комп'ютерного зору комп'ютери попередньо запрограмовані для вирішення окремих завдань, але методи, засновані на знаннях, стають все більш загальними.

Але є і нові галузі застосування комп’ютерного зору: автономні транспортні засоби, включно з підводними, наземними (роботи, автомобілі), повітряними. Рівень автономності сягає від повністю автономних (безпілотних) до транспортних засобів, де системи, засновані на комп'ютерному зорі, допомагають водію або пілоту в різних ситуаціях. Повністю автономні транспортні засоби використовують комп'ютерний зір для навігації, а саме - для отримання інформації про місце перебування, для створення карти навколишнього середовища, для виявлення завад і перешкод. Вони також можуть бути використані для певних завдань, наприклад, для виявлення лісових пожеж. Прикладами таких систем можуть бути система попереджувальної сигналізації про перешкоди на транспортних засобах і системи автономного приземлення літаків. Деякі виробники транспорту демонстрували системи автономного керування автомобілем, але ця

технологія ще не досягла того рівня, коли її можна запускати у масове виробництво [9]. Можливість ефективно та якісно обробити зображення та розпізнати на ньому пішохода чи тварину, які вибігли на дорогу – одне з головних завдань у цій сфері, тому вчені з усього світу продовжують працювати над проблемою швидкого та точного розпізнавання об’єктів у зображеннях.

1.3 Огляд відомих методів розв’язання задачі

На сьогоднішній день існує велика кількість методів (алгоритмів) інтелектуальних обчислень для розв’язання задач у сфері комп’ютерного зору, зокрема [10]:

- системи розмірковувань на основі аналогічних випадків;
- алгоритми визначення асоціацій і послідовностей;
- дерева рішень;
- нечітка логіка;
- генетичні алгоритми.
- нейронні мережі;

Системи розмірковувань на основі аналогічних випадків працюють таким чином, що для того, щоб зробити прогноз на майбутнє або вибрати правильне рішення, системи знаходять у минулому близькі аналоги наявної ситуації і вибирають ту ж відповідь, що і була для них правильною. Тому цей метод ще називають методом «найближчого сусіда». Системи розмірковувань на основі аналогічних випадків дають гарні результати в різних задачах. Головний їхній недолік полягає в тому, що вони не створюють яких-небудь моделей або правил, що узагальнюють попередній досвід, – у виборі рішення вони базуються на всьому масиві доступних історичних даних, тому неможливо сказати, на основі яких конкретно факторів ці системи будують свої відповіді.

Алгоритми визначення асоціацій знаходять правила про окремі предмети, що з’являються разом в одній транзакції, наприклад, в одній покупці.

Дерева рішень – цей метод широко застосовується в різних областях виробництва, фінансів і бізнесу, де часто зустрічаються задачі числового прогнозу. У результаті застосування цього методу, для навчальної вибірки даних створюється ієрархічна структура правил класифікації типу, «ЯКЩО... ТОДІ...», що мають вид дерева. Для того щоб вирішити, до якого класу віднести деякий об’єкт або ситуацію, треба відповісти на запитання, що стоїть у вузлах цього дерева, починаючи з його кореня. Питання можуть мати вигляд «Значення параметра А більше Х?» або «Значення змінної В належить підмножині ознак С?». Якщо відповідь позитивна, перехід до правого вузла наступного рівня, якщо негативна – то до лівого вузла; потім знову здійснюється процедура відповіді на питання, яке зв'язане з відповідним вузлом. Таким чином, зрештою, можна дійти до одного з кінцевих вузлів, де визначений клас об'єкта.

Нечітка логіка застосовується для масивів даних, у яких приналежність даних до якої-небудь групи є ймовірністю в інтервалі від 0 до 1. Чітка логіка маніпулює результатами, що можуть бути або істиною, або неправдою. Нечітка логіка застосовується в тих випадках, коли існує «може бути» у доповненні до «так» або «ні».

Генетичні алгоритми є могутнім засобом рішення різних комбінаторних задач і задач оптимізації. Особливістю генетичного алгоритму є акцент на використання оператора «схрещення», який виконує операцію рекомбінацію рішень-кандидатів, роль якої аналогічна ролі схрещення в живій природі. У результаті послідовної зміни поколінь виробляється рішення поставленої задачі, що уже не може бути далі поліпшено.

Нейронна мережа [11] – це комп’ютерна система, яка за принципом своєї роботи схожа на біологічну нейронну мережу, тобто мозок. Така мережа

складається з великої кількості вузлів, які є аналогами нейронів у мозку. Вузли поєднуються між собою штучним аналогом синапсів, що дозволяє їм обмінюватися інформаціє один між одним. Такі системи «навчаються» виконувати поставлену задачу на основі навчальних даних. Корекція результату відбувається за рахунок корекції ваг ребер, які з'єднують нейрони. Нейроні мережі реалізують непрозорий процес, тобто побудована модель, як правило, не має чіткої інтерпретації.

1.4 Обгрунтування вибору аналогу до програми розпізнавання об'єктів на зображеннях

На сьогоднішній день можна виділити декілька відомих програмних реалізацій, що здатні здійснювати розпізнавання об'єктів на зображеннях.

До таких програм можна віднести BytePace [12], що є програмним інструментом на мові Python. Спочатку алгоритм був розроблений для виявлення осіб на зображеннях, але його можна натренувати на виявлення інших об'єктів. Для своєї роботи він використовує розбиття (splitting) зображення на області, оцінку яскравості в цих областях і відсікання областей, де класифікований об'єкт однозначно не знаходиться. З цього випливає головний недолік системи – її низька достовірність розпізнавання. Це може призводити до помилок в роботі.

Вікно програми BytePace подане на рисунку 1.1.

Іншою реалізацією для розпізнавання об'єктів на зображеннях є програма Inception [13]. Принцип роботи програми полягає у відкиданні зайвих векторів. Зайві – це ті, які не ввійшли у взаємно знайдений кластер векторів. Наприклад, на зразку може бути тінь, яка розпізнається як межа, а на наступному зразку її може не бути. До недоліків можна віднести складність роботи з програмою та низьку достовірність розпізнавання [13].

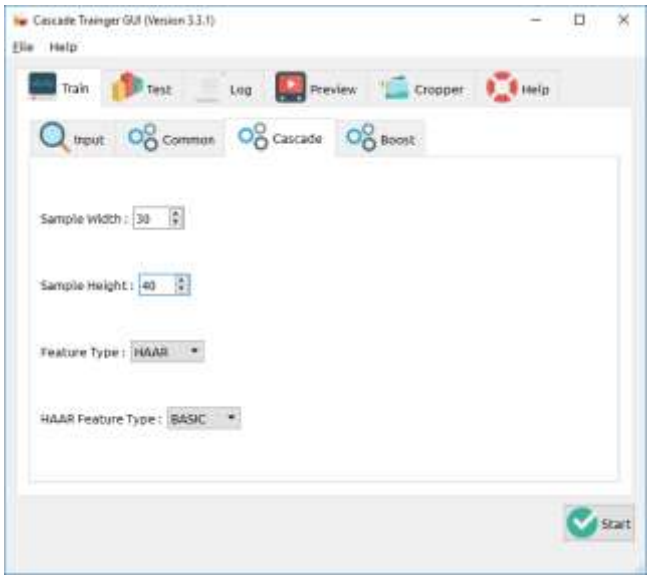


Рисунок 1.1 – Вікно програми BytePace

Вікно програми Inception подане на рисунку 1.2.

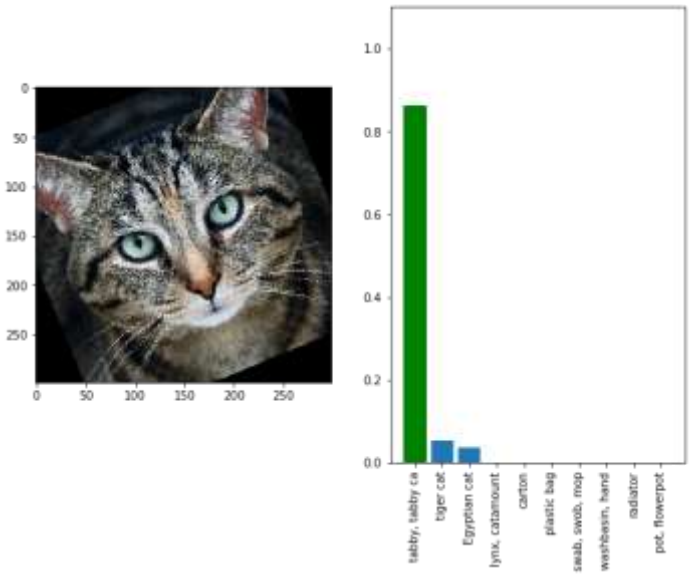


Рисунок 1.2 – Вікно програми Inception

Отже, розглянуті програмні засоби мають ряд недоліків таких як складність в освоєнні, низька швидкодія та достовірність розпізнавання. Відповідно, постає питання в розробці нових програмних засобів для розпізнавання зображень, які були б вільні від вказаних недоліків.

1.5 Висновок до розділу 1

У першому розділі було розглянуто стан питання розпізнавання об’єктів у зображеннях. Проведено огляд відомих методів для роботи з інтелектуальними обчисленнями у сфері комп’ютерного зору. В ході аналізу предметної області розглянуто основні методи розпізнавання зображень та як найбільш перспективний, було обрано нейромережевий метод. Також було здійснено аналіз відомих програмних засобів розпізнавання об’єктів на зображеннях та як аналог до розроблюваної програми було обрано програму Inception.

2 РОЗРОБКА ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ З ВИКОРИСТАННЯМ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

2.1 Обгрунтування вибору згорткової нейронної мережі для розпізнавання об'єктів на зображеннях

Проаналізувавши відомі методи для роботи з інтелектуальними обчисленнями у сфері комп'ютерного зору, було прийнято рішення використовувати нейронні мережі для вирішення задачі, адже вони є досить швидкими та точними в своїх обчисленнях. Загалом, існує багато різновидів штучних нейронних мереж. Для поставленої задачі найбільше підходять такі 2 групи нейромереж:

- Згорткові нейронні мережі (англ. Convolutional Neural Network, CNN);
- Рекурентні нейронні мережі (англ. Recurrent Neural Network, RNN).

Згорткові нейронні мережі – це клас глибинних нейронних мереж, які широко використовуються у задачах обробки зображень [14]. CNN складаються з трьох типів шарів:

- шар згортки (convolutional layer);
- шар об'єднання (pooling layer);
- шар повного об'єднання (fully-connected layer).

Шар згортки аналізує вхідні дані за допомогою набору фільтрів. У результаті аналізу отримуємо набір ознак, які подаються у вигляді n-вимірного масиву. Потім цей масив скеровується у шар об'єднання, який зменшує кількість ознак шляхом певних операцій над масивом. У результаті матимемо стиснений набір особливостей. Це необхідно для пришвидшення роботи алгоритму. Послідовність із шарів згортки та об'єднання повторюється декілька разів, в залежності від типу нейромережі. Після цього остаточний набір ознак надсилається до повнозв'язного шару. На цьому етапі

проводиться аналіз усіх ознак, отриманих у ході роботи нейронної мережі та визначається фінальний результат.

Рекурентні нейронні мережі (англ. Recurrent Neural Networks, RNN) – це клас нейромереж які дозволяють аналізувати вхідні дані в контексті попередніх або наступних даних [15]. Такий вид нейронних мереж є особливо ефективним при аналізі зв’язаних послідовностей, таких як текст, відео, аудіо, тощо.

У рекурентних нейронних мережах вхідні дані обробляються в одному шарі. Це дозволяє будувати нейронні мережі з різним співвідношенням вхідних даних до вихідних. Дані у рекурентній нейронній мережі обробляються послідовно у часі. Саме це й дозволяє проводити аналіз поточних даних у контексті попередніх. Хоча, у такого підходу є недолік, який полягає у тому, що для аналізу наступного блоку даних необхідно очікувати результат аналізу попереднього блоку, що обмежує можливість використання.

На рис. 2.1 фрагмент нейронної мережі A приймає вхідне значення x_t і повертає значення h_t.

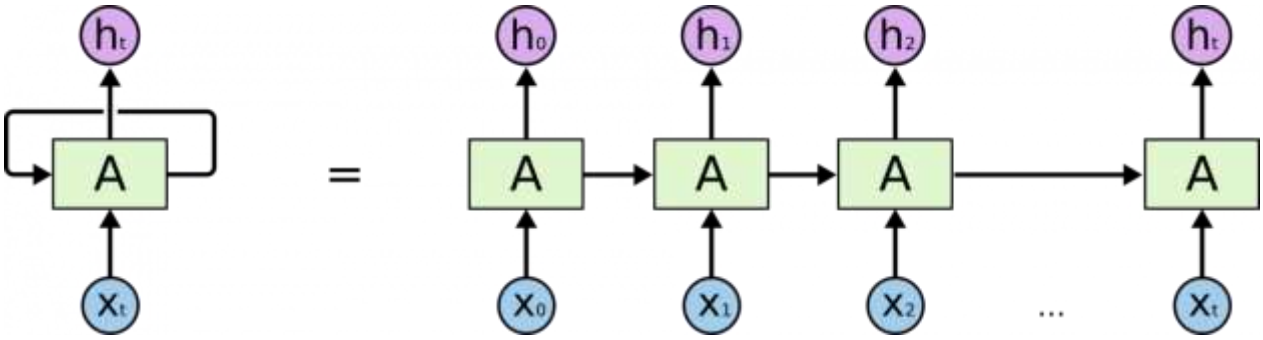


Рисунок 2.1 – Рекурентна нейромережа у розгортц

Наявність зворотного зв’язку дозволяє передавати інформацію від одного кроку навчання мережі до іншого. Одним з різновидів RNN є LSTM-мережі [15]. LSTM (англ. Long Short Term Memory) – це RNN здатні до навчання довготривалими залежностями. LSTM-мережі складаються з

повторюваних елементів. Кожен такий елемент містить чотири шари і відрізняється тим, що має комірку довгої короткочасної пам’яті [15]

Хоча обидва види нейронних мереж можна використати для вирішення поставленої задачі розпізнавання об’єктів у зображеннях, але найпопулярнішим підходом, що застосовується у наш час, є використання згорткових нейронних мереж (CNN), таких як VGG-16, R-CNN та YOLO [16], які автоматично навчаються виявляти об'єкти у зображеннях. У свою чергу, такі нейронні мережі розпізнавання об’єктів бувають двох типів – двоступеневі та одноступеневі.

Початковий етап двоступеневих мереж, таких як R-CNN, ідентифікує підмножини регіонів зображення, які можуть містити об’єкт. На другому етапі класифікуються об’єкти регіонів. Двоступеневі мережі можуть досягати дуже точних результатів розпізнавання об’єктів, однак вони, як правило, менш швидкодіючі, ніж одноступеневі мережі.

У одноступеневих мережах, таких як YOLO, CNN виробляє мережеві передбачення для регіонів у всьому зображенні за допомогою обмежувальних прямокутників, і потім прогнози декодуються для створення остаточних обмежувальних прямокутників для об’єктів [17]. Одноступеневі мережі можуть бути набагато швидкодіючими, ніж двоступеневі, але вони можуть не досягати однакового рівня достовірності, особливо для зображень, які містять невеликі об’єкти.

На сьогодні існує два способи аби розпочати роботу над розпізнаванням об’єктів у зображеннях за допомогою нейронних мереж.

З одного боку, можна почати працювати з нейромережею «з нуля», а саме - створити та навчити власний розпізнавач зображень. Для цього необхідно розробити архітектуру мережі, а також набрати величезний обсяг навчальних даних, на яких власна мережа буде тренуватися розпізнавати певні об’єкти. Звичайно, результати використання такої нейромережі можуть бути чудовими, однак з огляду на те, що потрібно в ручному режимі налаштовувати шари та ваги нейронної мережі, витрати часу на

налаштування декількох мільйонів параметрів, а також на власне тренування мережі (сотні годин роботи GPU) будуть просто величезними.

Тому з іншого боку, існує варіант використання попередньо натренованої нейронної мережі. Цей метод може забезпечити більш швидкі результати, оскільки розпізнавачі об'єктів вже пройшли навчання на тисячах, а то й мільйонах зображень, і все що залишається – точно налаштувати таку мережу для власної задачі, що не потребує великих обсягів обчислювальних ресурсів.

Отже, провівши аналіз відомих методів для роботи з інтелектуальними технологіями та обчисленнями у сфері комп’ютерного зору, для розробки програмного модуля розпізнавання об’єктів у зображеннях було обрано алгоритм, який базується на використанні попередньо натренованої згорткової нейронної мережі YOLOv2, яка, порівняно зі своїми аналогами [17], є досить швидкою та точною в обчисленнях.

2.2 Структура та порядок функціонування згорткової нейронної мережі

YOLOv2 – нейромережа з одноступеневою архітектурою, яка складається з 19 шарів згортки та 5 шарів об’єднання. Вона розглядає ціле зображення під час розпізнавання, тому її вхідний шар приймає тензор 1 x 3(RGB) x 416px x 416px. Мережа приймає пікселі вхідного зображення і пропускає їх через різні шари для отримання вихідних даних (рисунок 2.2).

Вихідний шар має розмірність 1 x 13 x 13 x (k x (1 + 4 + 80)), де 13 x 13 – кількість комірок сітки, на яке поділяється зображення; k – кількість полів прив`язки (anchor boxes); 80 – кількість класів об’єктів, які розпізнає нейромережа. Оскільки авторами мережі було визначено значення k = 5 як найбільш оптимальне, то вихідний тензор має розмірність 1 x 13 x 13 x 425 (рисунок 2.3).

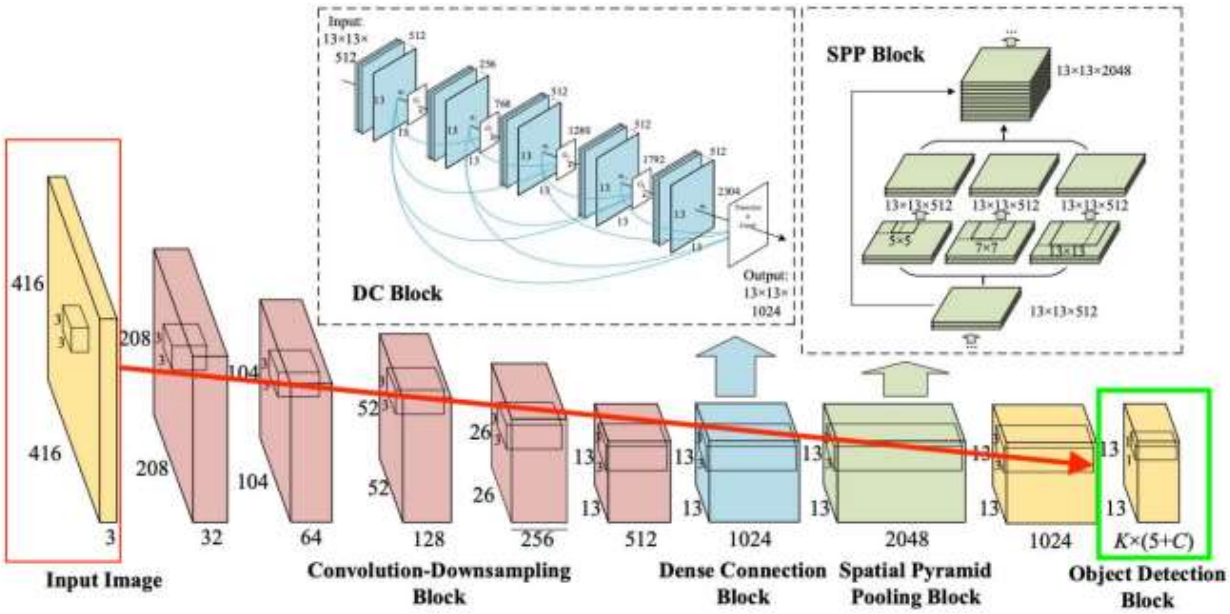


Рисунок 2.2 – Структура нейромережі YOLOv2

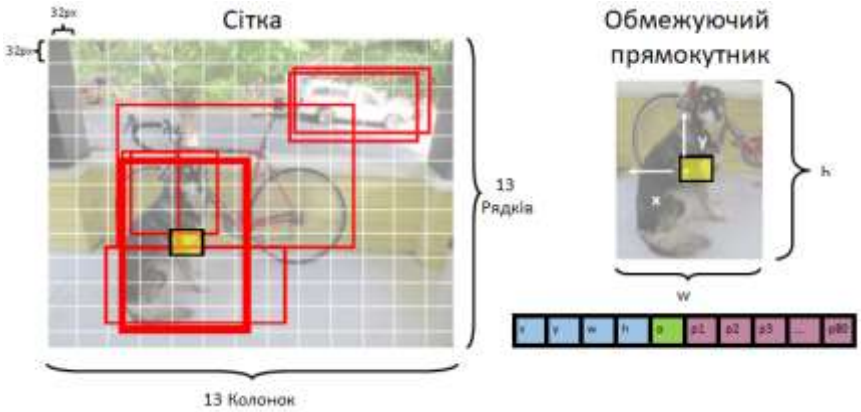


Рисунок 2.3 – Графічне відображення вихідних даних нейронної мережі YOLOv2

Отже, на виході, для кожної комірки сітки (32px x 32px) міститься 425 параметрів (див. рисунок 2.3):

- x – координата по осі X центру обмежувального прямокутника щодо комірки сітки, з якою він пов’язаний;
- y - координата по осі Y центру обмежувального прямокутника щодо комірки сітки, з якою він пов’язаний;
- w – ширина обмежувального прямокутника;

- h – висота обмежувального прямокутника;
- o – значення достовірності того, що об’єкт існує в межах обмежувального прямокутника, також відоме як оцінка присутності об’єкта (від 0 до 1);
- p1-p80 – ймовірності для кожного з 80 класів об’єктів, прогнозованих нейромережею (від 0 до 1).

Ідея роботи згорткової нейромережі є в тому, що обробка певної ділянки зображення має відбуватися незалежно від місця конкретного розташування цієї ділянки. Процес розпізнавання об'єкту на зображенні має відбуватися локально і незалежно від конкретного місця положення ділянки з об'єктом на всьому зображенні. Для цього вхідне зображення покривається невеликими вікнами (напр., 5x5 пікселів) і ознаки виділяються у кожному такому вікні невеликою нейромережею, причому будуть виділятися одні й ті ж самі ознаки, тобто невелика нейромережа буде лише одна, у неї буде 5×5=25 входів, а із кожного зображення будемо мати багато виходів.

За тим, результати цієї нейромережі знов можна буде подати у вигляді зображення, замінивши вікна розміру 5×5 на їх же центральні пікселі, яке можна подати на другий згортковий шар, з іншою уже нейромережею.

Часто зображення, які подають на вхід, подані у вигляді кількох прямокутних матриць, кожна з яких має рівень одного з колірних каналів у кожному пікселі зображення. Наприклад, зображення розміром 200×200 пікселів - це 120000 чисел, тобто три матриці яскравостей розміром 200×200 пікселів кожна. Якщо вхідне зображення є чорно-білим, то така матриця одна. Якщо вхідне зображення багатоспектральне, то таких матриць буде багато. Передбачається, що у кожному пікселі вхідного зображення знаходиться деякий тензор (вектор чисел), а його компоненти звуться каналами (channels).

Такі само матриці будуть виходити після згорткового шару. У них, як і раніше, буде просторова структура, що відповідає вихідній картині, але кількість каналів може стати більшою. Значення ознак, що виділені вікнами із

вхідного зображення, утворюють матрицю, що має назву - карта ознак (feature map).

Згортка – це лінійне перетворення вхідних даних (чисел). Якщо x^l – це карта ознак у шарі з номером $\boldsymbol{L}$, то результатом двовимірної згортки з ядром розміру 2d+1 і матрицею ваг W розміру (2d + 1) × (2d + 1) на наступному шарі буде таке

$$y_{i,j}^{l} = \sum_{-d \le a,b \le d} W_{a,b} x_{i+a,j+b}^{l}$$

де $y_{i,j}^{l}$ - результат згортки на рівні l, а $\boldsymbol{x}_{i,j}^{l}$- вхід згортки, тобто вихід попереднього шару. Отож, щоб отримати компонент *(i, j)* наступного рівня, треба застосувати лінійне перетворення до квадратного вікна попереднього рівня, тобто скалярно перемножити пікселі вікна на вектор згортки, що і зображено на рис. 2.4. Використовується згортка з ядром (матрицею ваг) *W* розміру 3×3 до матриці розміру 5×5. Приклад обчислення згортки зображено на рис. 2.4.

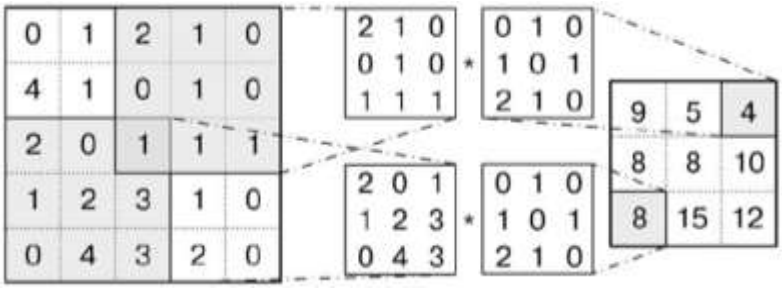


Рисунок 2.4 — Приклад обчислення згортки

Операція згортки має такі властивості:

• зберігає структуру входу (взаємне розташування пікселів), так як застосовується окремо до кожної ділянки вхідного зображення.

• має властивість розрідженості, оскільки вихідний сигнал кожного нейрона наступного шару залежить тільки від невеликої кількості вхідних нейронів.

• багаторазово перевикористовує одні і ті самі ваги, оскільки вони повторно використовуються до різних ділянок вхідного зображення.

За згорткою у більшості випадків виконується нелінійність, що записується так

$$z_{i,j}^{l} = h(y_{i,j}^{l})$$

Як варіант нелінійності h часто використовують функцію *ReLU*.

У класичному випадку після лінійної згортки і наступної за нею нелінійності, застосовують ще операцію субдискретизації (pooling). Ідея субдискретизації: наявність або відсутність ознаки є важливішою, ніж її точні координати, а тому можна узагальнити виділені ознаки, втративши при цьому частину інформації про їх місцеположення, але натомість скоротивши розмірність даних. Як операція субдискретизації застосовується функція визначення максимуму (max-pooling)

$$x_{i,j}^{l+1} = max(z_{i+a,j+b}^{l}), a \in [-d, d], b \in [-d, d]$$

де d - розмір вікна субдискретизації. При d = 2 приклад операції субдискретизації зображено на рис. 2.5.

Стандартний шар згорткової нейромережі складається з трьох частин:

• Згортка у вигляді лінійного відображення, яке виділяє локальні ознаки,

• Нелінійна функція, яка застосовкється покомпонентно до результатів згортки,

• Субдискретизація, яка зменшує геометричний розмір отримуваних тензорів.

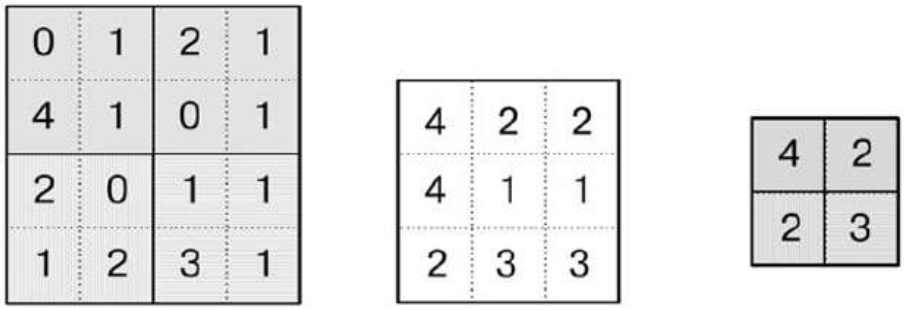


Рисунок 2.5 - Приклад субдискретизації з вікном розміру 2×2: *а* - початкова матриця; *б* - матриця після субдискретизації з кроком 1, *в* - матриця після субдискретизації з кроком 2.

На рис. 2.6 зображено такий стандартний шар. Порівняно з вхідним зображенням, на виході шару розмірність тензора збільшилася через те, що згорткову нейромережу зазвичай навчає декілька карт ознак у кожному шарі.

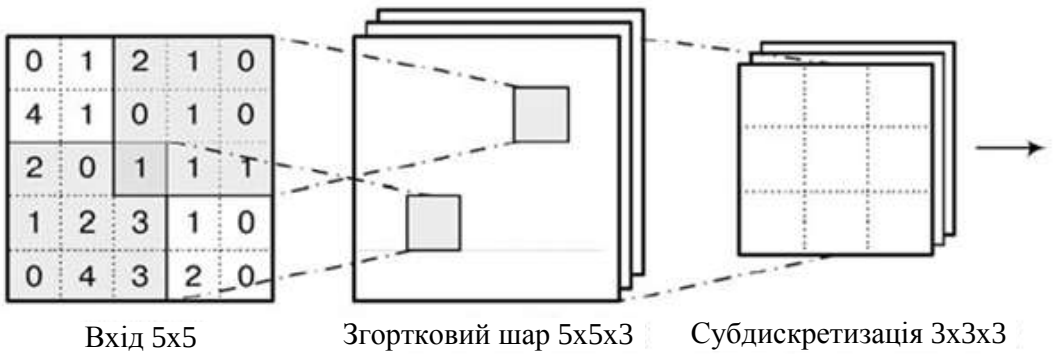


Рисунок 2.6 - Схема одного шару згорткової нейромережі

Для того, щоб навчити карти ознак, необхідно поширити градієнт помилки у зворотному напрямку через згортковий шар. Потрібно оптимізувати якусь функцію помилки E , коли відомо її значення на виході мережі. Завдяки функції знаходження максимуму помилка проходить без

змін, оскільки шар субдискретизації нічому не навчається. Однак, він, проходячі по графу обчислення, робить градієнти розрідженими, оскільки з усіх елементів вікна субдискретизації $z_{i,j}^{l}$ часткова похідна $\frac{\partial E}{\partial x_{i,j}^{l+1}}$ відноситься тільки до максимального елемента, а решта отримують нульовий градієнт. І на цьому навчальному прикладі їх тренування можна вважати закінченим. Пропустивши градієнт через нелінійність, отримаємо

$$\frac{\partial E}{\partial y_{i,j}^{l}} = \frac{\partial E}{\partial z_{i,j}^{l}} \frac{\partial z_{i,j}^{l}}{\partial y_{i,j}^{l}} = \frac{\partial E}{\partial z_{i,j}^{l}} h'(y_{i,j}^{l}),$$

де $\frac{\partial E}{\partial z_{i,j}^{l}}$ уже є відомим, а $h'(y_{i,j}^{l})$ можна обчислити.

На згортковому рівні існують ваги, які потрібно навчати:

$$\frac{\partial E}{\partial w_{a,b}^{l}} = \sum_{i} \sum_{j} \frac{\partial E}{\partial y_{i,j}^{l}} \frac{\partial y_{i,j}^{l}}{\partial w_{a,b}^{l}} = \sum_{i} \sum_{j} \frac{\partial E}{\partial z_{i+a,j+b}^{l-1}},$$

де індекси *i* та *j* пробігають усі пієселі зображення на проміжному шарі $y_{i,j}^{l}$, тобто це відбувається після згортки, але до субдискретизації.

Пропускаючи градієнти на попередній шар, отримуємо:

$$\frac{\partial E}{\partial x_{i,j}^{l}} = \sum_{a} \sum_{b} \frac{\partial E}{\partial y_{i-a,j-b}^{l}} \frac{\partial y_{i-a,j-b}^{l}}{\partial x_{i,j}^{l}} = \sum_{j} \sum_{i} \frac{\partial E}{\partial y_{i-a,j-b}^{l}} w_{a,b}.$$

Таким чином, маємо процедуру зворотного поширення помилки, яка є адаптованою для згорткового шару.

Глибока нейромережа будується шляхом подачі вихідного сигналу згорткового шару на вхід наступного шару. Розмірність шару за рахунок

операції субдискретизації буде поступово зменшуватися, що дозволить шарам, які знаходяться в кінці мережі, розпізнавати більш загальні ознаки вхідного зображення.

Архітектура згорткової нейромережі YOLO представлена у табл. 2.1:

Таблиця 2.1 - Архітектура згорткової нейромережі YOLO

Шар	ядро	крок	розмірність виходу
Вхідний			(416, 416, 3)
Згортковий 1	3×3	1	(416, 416, 16)
MaxPooling 1	2×2	2	(208, 208, 16)
Згортковий 2	3×3	1	(208, 208, 32)
MaxPooling 2	2×2	2	(104, 104, 32)
Згортковий 3	3×3	1	(104, 104, 64)
MaxPooling 3	2×2	2	(52, 52, 64)
Згортковий 4	3×3	1	(52, 52, 128)
MaxPooling 4	2×2	2	(26, 26, 128)
Згортковий 5	3×3	1	(26, 26, 256)
MaxPooling 5	2×2	2	(13, 13, 256)
Згортковий 6	3×3	1	(13, 13, 512)
MaxPooling 6	2×2	1	(13, 13, 512)
Згортковий 7	3×3	1	(13, 13, 1024)
Згортковий 8	3×3	1	(13, 13, 1024)
Згортковий 9	1×1	1	(13, 13, 425)

Ця нейронна мережа використовує лише стандартні типи шарів: згортка з ядром 3×3 і максимальне об’єднання з ядром 2×2. Ніяких вишуканих речей. У YOLOv2 немає повнозв`язного шару.

Останній згортковий шар має ядро 1 × 1 і існує, щоб зменшити дані до форми 13 × 13 × 425. Цей розмір 13 × 13 має виглядати знайомим: це розмір сітки, на яку ділиться зображення.

Таким чином ми отримуємо 425 каналів для кожної комірки сітки. Ці 425 чисел містять дані для обмежувальних рамок і прогнозів класу. Каналів 425 тому, що кожна клітинка сітки передбачає 5 обмежувальних рамок, а обмежувальна рамка описується 85 елементами даних:

- координати x, y, ширина, висота прямокутника обмежувальної рамки,
- оцінка довіри,
- розподіл ймовірностей по 80 класах.

YOLO використовується так: на вхід мережі подається вхідне зображення (розмір якого змінюється до 416 × 416 пікселів), воно проходить через згорткову нейромережу за один прохід і виходить на іншому кінці у вигляді тензора 13 × 13 × 425, що описує обмежувальні рамки для клітинки сітки. Все, що залишається зробити, це обчислити остаточні бали для обмежувальних рамок і відкинути ті, що мають імовірність нижче 30%.

2.3 Навчання згорткових нейронних мереж

Головне завдання при навчанні нейромереж – це мінімізація обраної функції помилки [11, 12]. Як правило, оптимізують апостеріорну ймовірність.

$$p(\theta|D) = p(\theta)\prod_{d\in D} p(d|\theta),$$

де $p(\theta)$ - функція правдоподібності (похибка на навчальній вибірці), $p(d \mid \theta)$ — апріорний розподіл (регуляризація).

Завданням оптимізації по заданій функції є знаходження аргументів, у яких ця функція мінімізовується. Функція помилки в нейромережах має доволі багато локальних екстремумів. Для знаходження оптимального екстремуму використовують евристичний метод оптимізації, так званий градієнтний спуск.

Маючи уявлення про поверхню функції помилки, завданням оптимізації буде обчислення градієнта. Якщо функцію, яка визначає поверхню помилки, позначити як

$$E(\boldsymbol{\theta})=E(\theta_1, \theta_2,\ldots, \theta_n),$$

де $(\theta_1, \theta_2,\ldots, \theta_n)$ – параметри функції, то градієнт цієї функції ∇E – це вектор окремих похідних функції від кількох змінних по кожному із аргументів

$$\nabla_\theta E = \begin{pmatrix} \frac{\partial E}{\partial \theta_1} \\ \vdots \\ \frac{\partial E}{\partial \theta_n} \end{pmatrix}.$$

Градієнт визначає напрям, у якому функція зростає найшвидше. А отже, напрям, у якому вона найшвидше спадає – це напрям, зворотній до градієнту: $-\nabla_\theta E$. Позначивши через $\boldsymbol{\theta}_t$ вектор параметрів моделі на кроці t, а через E - мінімізовану функцію, можемо визначити вектор відновлення параметрів на кроці t як

$$\boldsymbol{u}_t = -\eta \nabla_\theta E(\boldsymbol{\theta}_{t-1}),$$
$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} + \mathbf{u}_t,$$

де η – швидкість навчання. Вона регулює крок поновлення параметрів.

У стандартному алгоритмі градієнтного спуску швидкість навчання задають вручну. Це може викликати такі проблеми:

– Якщо швидкість навчання занадто маленька, то навчання буде занадто тривалим і підвищиться імовірність потрапляння у невдалий локальний мінімум функції помилки (див. рис. 2.7(а)),

– Якщо швидкість навчання занадто велика, то можна пропустити потрібний локальний мінімум (див. рис. 2.7(б)).

Для розв`язання цих проблем застосовується стохастичний градієнтний спуск, тобто підрахунок похибки і оновлення ваг здійснюється не після проходу всієї навчальної множини, а після кожного прикладу:

$$\boldsymbol{\theta}_t = \boldsymbol{\theta}_{t-1} - \eta \nabla E(f(\mathbf{x}_t, \boldsymbol{\theta}_{t-1}), y_t)$$

Рисунок 2.7 - Проблеми при різній швидкості градієнтного спуску:
а) занадто маленька швидкість,
б) занадто велика швидкість.

Головна перевага стохастичного градієнтного спуску – це велика швидкість обчислення похибки, оскільки оновлення ваг здійснюється після кожного кроку, що підвищує швидкість навчання. Окрім цього, стохастичний градієнтний спуск функціонує більш випадково, а це дозволяє знайти більш глибокий локальний мінімум функції похибки.

На практиці застосовується стохастичний градієнтний спуск по міні-батчах - невеличких підмножинах навчального набору. Це дозволяє зберегти позитивні риси стохастичного градієнтного спуску, і одночасно користуватися операціями матричної арифметики, що обчислюються швидко на пристроях, які підтримують паралельні обчислення.

При навчанні великої нейромережі, функція похибки - складна композиція кількох інших функцій. Представляючи складну функцію як композицію простіших функцій, можна легше обчислити її похідну, що потрібно для реалізації градієнтного спуску.

Функція похибки представляється у виді графа обчислень, тобто графа, вузлами якого є функції, а ребра пов'язують ці функції з їхніми аргументами. На рисунку 2.8 зображено приклад графа обчислень для функції

$$f(x, y) = x^2 + xy + (x + y)^2.$$

На рис. 2.8а - граф обчислень з використанням функцій ×, + і x^2, б - граф обчислень з використанням функцій × і +, в - граф обчислень у вигляді дерева з використанням функцій × і +.

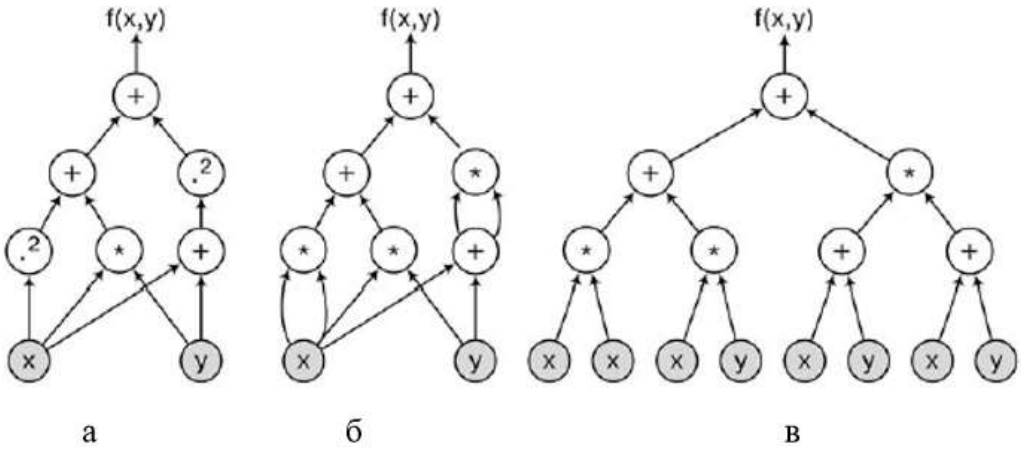


Рисунок 2.8 - Граф обчислень для функції $f(x,y) = x^2 + xy + (x+y)^2$.

У нейромережах як елементарні функції використовують функції, з яких отримуються нейрони. Наприклад, функції скалярного добутку векторів $\mathbf{x}^\top \mathbf{y}$ і функції логістичної сигмоїди σ достатньо для того, щоб побудувати довільну нейромережу, що складається з нейронів з функцією активації σ.

Так, представляючи складну функцію через прості, можна обчислити похідну складної функції, обчисливши окремі похідні її складових:

$$(f \circ g)'(x) = \big(f(g(x))\big)' = f'(g(x))g'(x)$$

Якщо x - вектор $\mathbf{x}=(x_1, ..., x_n)$, то замість окремої похідної обчислюється градієнт

$$\nabla_x f = \begin{pmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{pmatrix}.$$

Використовуючи метод для кожного компонента окремо, в результаті отримаємо

$$\nabla_x (f \cdot g) = \begin{pmatrix} \frac{\partial f \cdot g}{\partial x_1} \\ \vdots \\ \frac{\partial f \cdot g}{\partial x_n} \end{pmatrix} = \begin{pmatrix} \frac{\partial f}{\partial g}\frac{\partial g}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial g}\frac{\partial g}{\partial x_n} \end{pmatrix} = \frac{\partial f}{\partial g}\nabla_x g.$$

Якщо g- вектор, то вийде

$$\nabla_x f = \frac{\partial f}{\partial g_1}\nabla_x g_1 + \cdots + \frac{\partial f}{\partial g_k}\nabla_x g_k = \sum_{i=1}^{k} \frac{\partial f}{\partial g_i}\nabla_x g_i$$

У результаті, отримується формула матричного множення

$$\nabla_x f = \nabla_x g \nabla_g f,$$

$$\nabla_x g = \begin{pmatrix} \frac{\partial g_1}{\partial x_1} & \cdots & \frac{\partial g_k}{\partial x_1} \\ \vdots & \ddots & \vdots \\ \frac{\partial g_1}{\partial x_n} & \cdots & \frac{\partial g_k}{\partial x_n} \end{pmatrix}.$$

Тобто, можна обчислити похідні і градієнти будь-якої композиції функцій, зокрема і векторних.

Таким чином, можна використовувати формулу диференціювання композиції на графі або від початків до стоків, обчислюючи окремі похідні кожного вузла по одній і тій самій змінній $\frac{dx}{dx}$, $\frac{da}{dx}$, ... ,$\frac{df}{dx}$ або від стоків до

початків, отримуючи окремі похідні по всіх проміжних вузлах $\frac{df}{df}$, $\frac{df}{de}$, … ,$\frac{df}{dx}$. Дійшовши до початків графа - вершин x_1 , ..., x□ , отримкються окремі похідні $\frac{df}{dx1}$, ... ,$\frac{df}{dxn}$, тобто градієнт ∇xf.

Оскільки окремі похідні обчислюються в напрямку, що є протилежним до графу обчислень, то такий алгоритм називають алгоритмом зворотного поширення (backpropagation).

2.4 Структура інформаційної технології розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі

Структура інформаційної технології розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі, зображена на рис. 2.9.

В основі цієї інформаційної технології лежить згорткова нейронна мережа. Так як кожна нейромережа має два режими роботи: режим навчання та режим функціонування, то структура інформаційної технології має дві частини. Ліва частина структури на рис. 2.9 віддзеркалює режим навчання згорткової нейронної мережі, а права частина структури на рис. 2.9 віддзеркалює режим функціонування. Для роботи інформаційної технології розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі в режимі навчання вхідною є інформація у вигляді набору зображень для навчання і тестування. У цій частині спочатку відбувається процес створення (генерації) згорткової нейромережі відповідно до обраної архітектури YOLOv2. Потім здійснюється процес початкової ініціалізація ваг мережі невеликими випадковими значеннями у обраному діапазоні. Потім відбуваться процес навчання нейронної мережі на основі навчального набору зображень. Після цього згорткова нейронна мережа готова до розпізнавання об’єктів на зображеннях.

Рисунок 2.9 - Структура інформаційної технології розпізнавання об'єктів на зображеннях з використанням згорткової нейронної мережі

Для роботи інформаційної технології розпізнавання об'єктів на зображеннях на основі згорткової нейронної мережі в режимі функціонування вхідною є інформація у вигляді файлу зображення форматів .jpg, .jpeg, .png.). У цій частині спочатку відбувається процес перетворення зображення до розміру 416x416 пікселів, оскільки згорткова нейронна мережа YOLOv2 на вхід приймає зображення саме такого розміру. Далі відбуваться процес подачі зображення на вхід попередньо навченої згорткової нейронної мережі. Після цього здійснюється процес виявлення нейронною мережею на зображенні об`єктів, що належать до наперед визначених 80 класів. Далі визначаються розміри і положення рамки навколо знайдених об`єктів. І, нарешті, проходить процес накладання на зображення рамки, назви класу об'єкта та достовірності віднесення його до цього класу.

Таким чином, розроблена структура інформаційної технології розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі може бути використана для подальшої розробки програмних засобів.

2.5 Розробка структури програмного забезпечення

Для того, щоб програма працювала стабільно і підлягала масштабуванню, необхідно, щоб її структура була простою та поділялася на прості модулі, які можна додавати або вилучати. Структура програми наведена на рисунку 2.10.

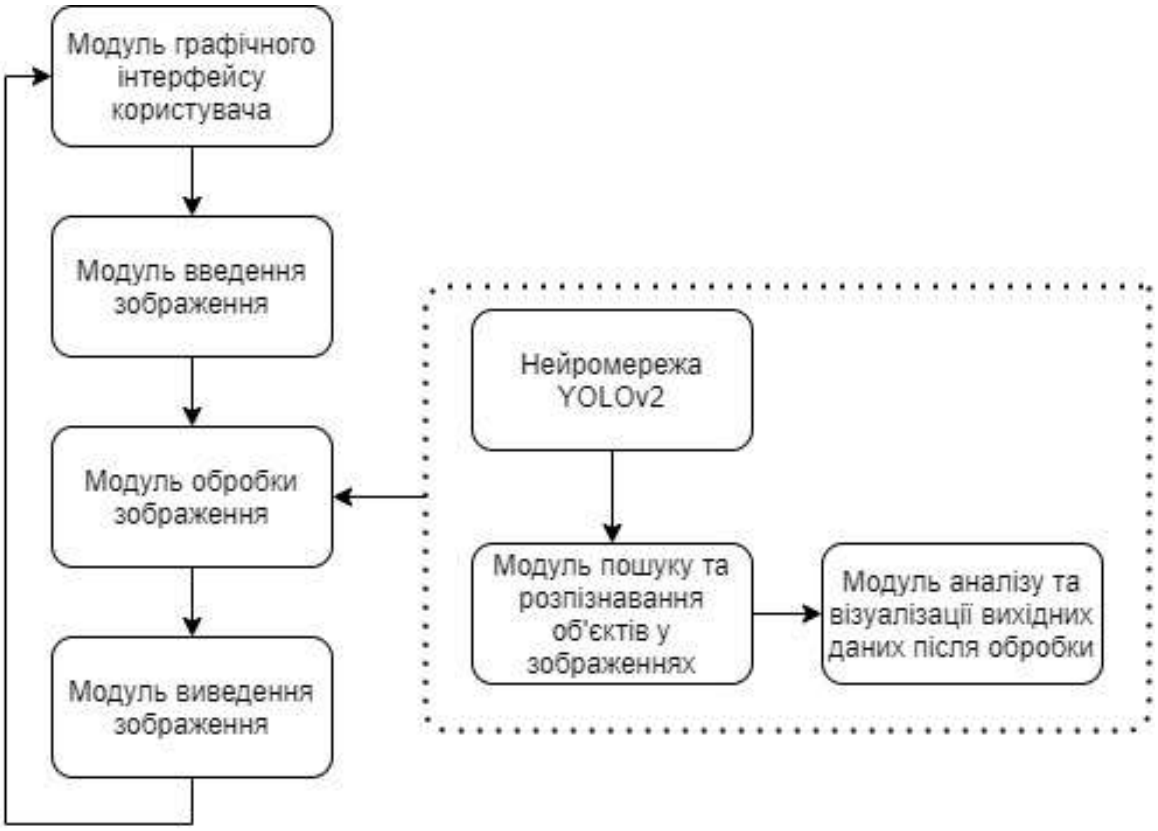


Рисунок 2.10 – Структура програмного забезпечення

Як видно з рисунку 2.10, графічний інтерфейс користувача пов’язаний з модулем введення та виведення зображення і відповідає за відображення форми завантаження нового зображення та відображення результуючого

після розпізнавання зображення користувача. Модуль введення контролює отримання вхідного зображення допустимого формату (.jpg, .jpeg, .png.), приводить його до розміру (416x416), який приймає обрана нейромережа та передає його у модуль обробки.

Модуль обробки зображення виконує пошук і розпізнавання об’єктів за допомогою нейромережі та аналіз і візуалізацію вихідних даних після обробки. Результат роботи модуля обробки зображення інтерпретується у кінцеве зображення з виділеними класами об’єктів (якщо такі є), їх назвою та достовірністю розпізнавання (рисунок 2.11). Результуюче зображення повертається користувачеві у графічний інтерфейс.

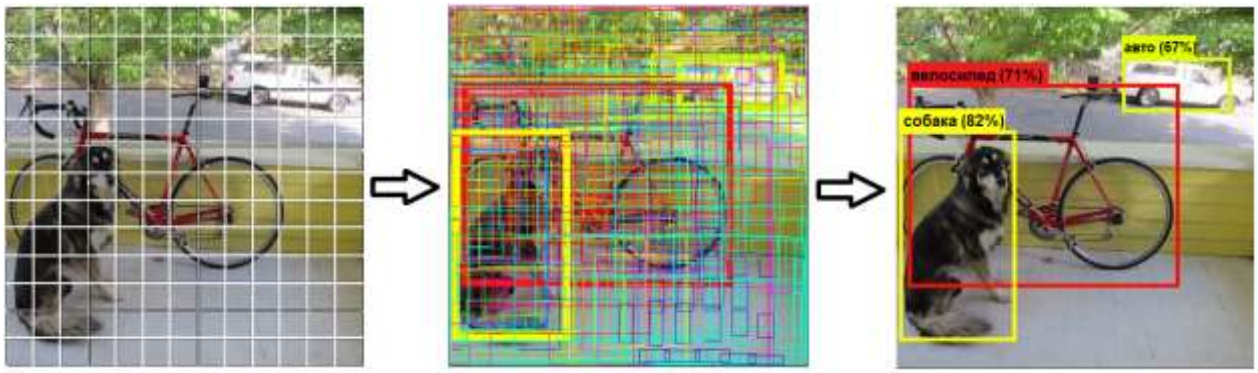


Рисунок 2.11 – Графічне відображення початкових, проміжних та кінцевих даних обробки зображення

На етапі проектування було також створено діаграму класів програмного забезпечення, наведену на рисунку 2.12.

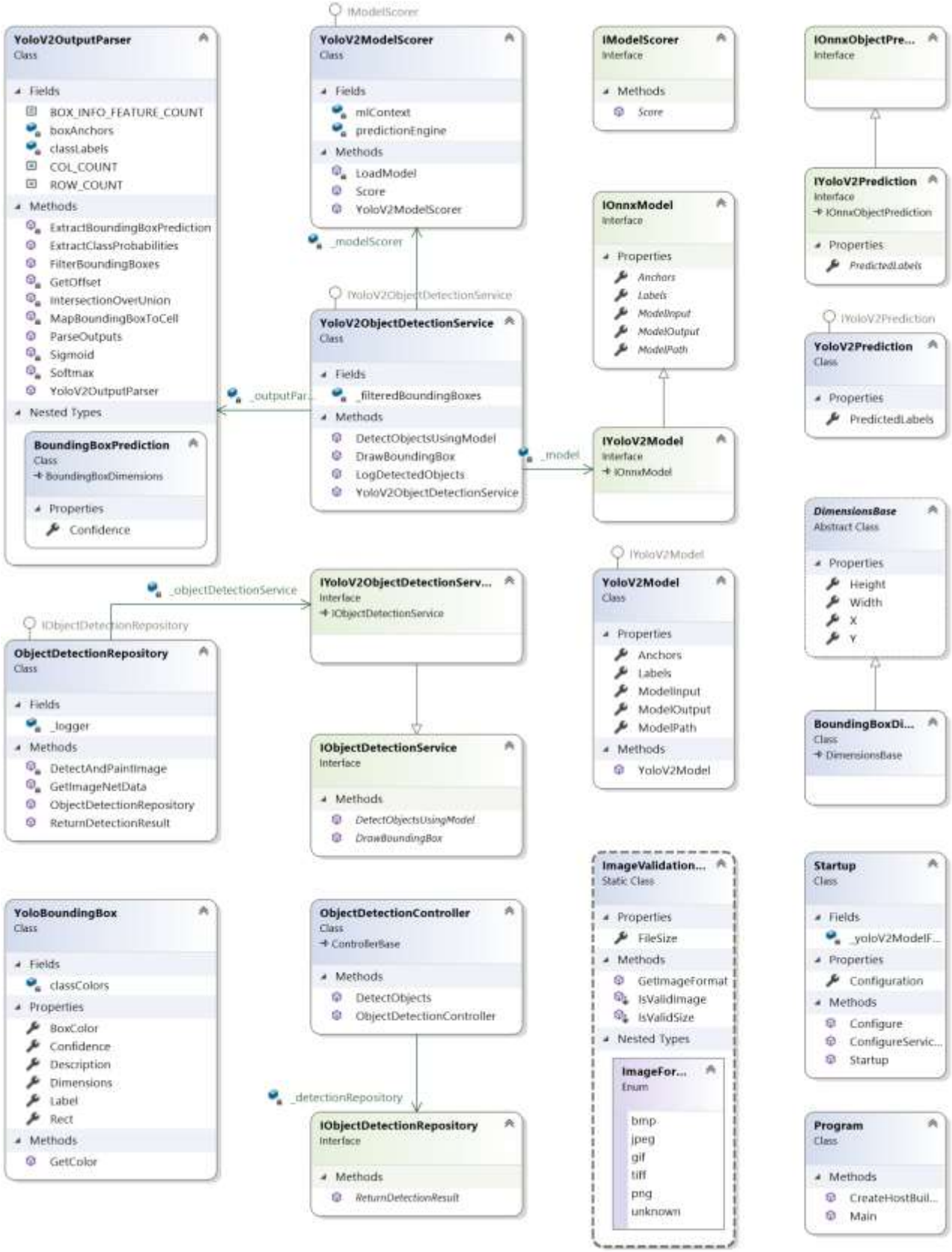


Рисунок 2.12 – Діаграма класів програмного забезпечення

2.6 Розробка алгоритму роботи програмного забезпечення розпізнавання об’єктів на зображеннях

Щоб розробити програму розпізнавання об’єктів у зображеннях, необхідно розробити його загальний алгоритм роботи. Граф-схема алгоритму зображена на рисунку 2.13.

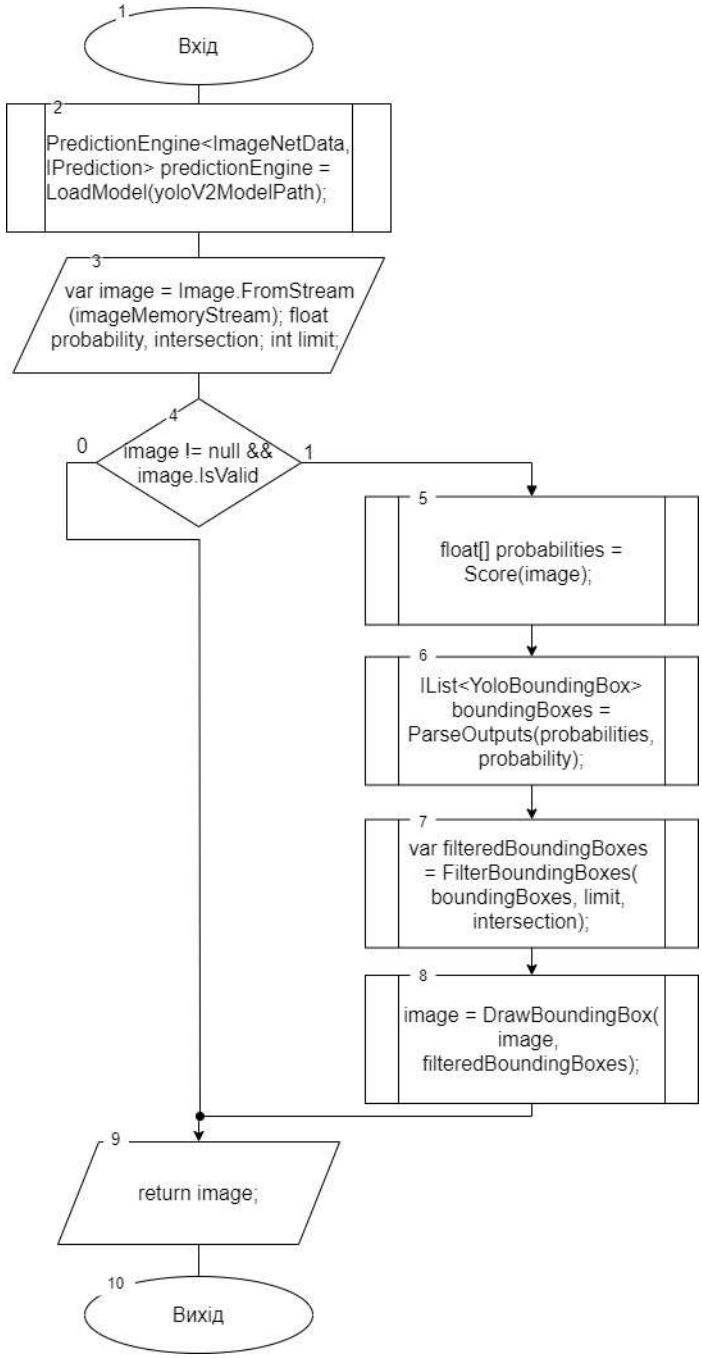


Рисунок 2.13 – Граф-схема загального алгоритму роботи програми

Загальний алгоритм роботи програми має такі кроки: завантаження попередньо натренованої нейронної мережі, отримання вхідних даних у вигляді цифрового зображення користувача, обробка вхідних даних шляхом пошуку та розпізнавання об’єктів за допомогою нейромережі, обробка масиву вихідних даних, які створила мережа, фільтрація вихідних даних, створення зображення з обмежувальними прямокутниками, з назвами розпізнаних класів та достовірністю їх розпізнання.

Вхід.

Вхідне зображення форми (3x416x416).

Вихід.

Вихід – масив (1x125x13x13).

Етапи попередньої обробки.

Змініть розмір вхідного зображення до (3x416x416) масиву типу float32.

Етапи постобробки.

Результатом є тензор (125x13x13), де 13x13 — це кількість клітинок сітки, на які ділиться зображення. Кожна клітинка сітки відповідає 125 каналам, які складаються з 5 обмежувальних рамок, передбачених клітинкою сітки, і 25 елементів даних, які описують кожну обмежувальну рамку (5x25=125).

Граф-схема алгоритму роботи програми складається з операцій, що слідують у заданій послідовності процесів, при виконанні яких наведено конкретний результат у визначеному форматі. Програма функціонує до моменту її закриття користувачем.

2.6 Висновок до розділу 2

У розділі було обґрунтовано доцільність використання для даної задачі згорткової нейронної мережі, а саме – попередньо натренованої нейронної мережі YOLOv2. Була проаналізована структура та порядок функціонування

згорткової нейромережі YOLOv2. Розроблено структуру інформаційної технології розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі Також було розроблено структуру програмного забезпечення та алгоритм роботи програми розпізнавання об’єктів на зображеннях.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ

3.1 Обгрунтування вибору засобів розробки

Значно полегшити процес реалізації будь-якого програмного додатку дозволить вибір правильних технології розробки. Розглянемо інтегровані середовища розробки (англ. Integrated development environment – IDE) для ефективної реалізації програмного модуля розпізнавання об'єктів у зображеннях. Для вибору середовища програмування розглянемо наступні IDE [18]:

1. Qt Creator.
2. Eclipse.
3. Microsoft Visual Studio.

Qt Creator – безкоштовний крос-платформний інструментарій розробки програмного забезпечення мовою C++ [19, 20]. Дозволяє запускати написане за його допомогою ПЗ на більшості сучасних ОС, просто компілюючи текст програми для кожної операційної системи без змін коду. Містить усі основні класи, які можуть бути потрібні для розробки прикладного програмного забезпечення, починаючи з елементів графічного інтерфейсу й закінчуючи класами для роботи з мережею, базами даних, OpenGL, SVG і XML. Програма створена компанією Qt Development Frameworks, поточна версія – 4.7.0.

Qt Creator також може використовуватись в багатьох інших мовах програмування: C# (Qyoto/Kimono), Ada (QtAda), Qt Jambi, Java (Qt Jambi), PHP (PHP-Qt), Pascal, Perl, Ruby (QtRuby), та Python (PyQt,PySide).

Фрагмент роботи програми Qt Creator наведено на рисунку 3.1.

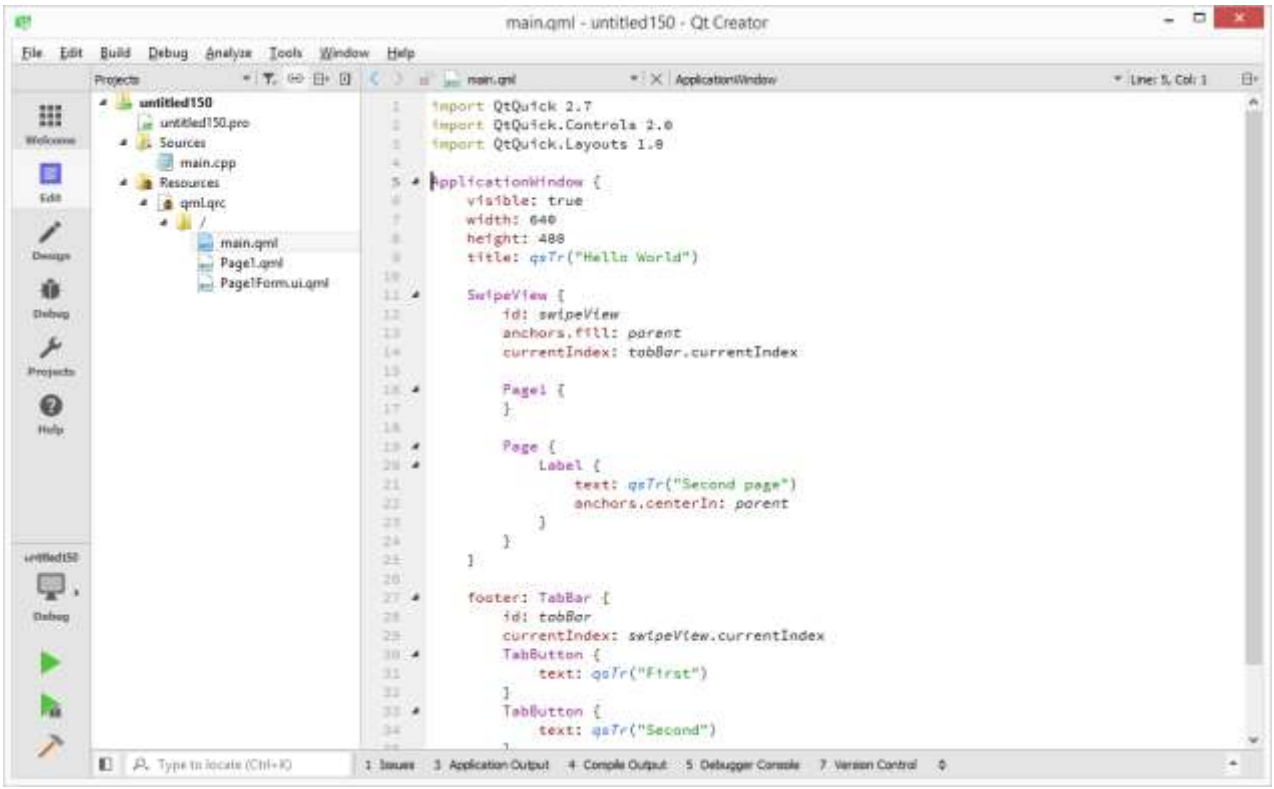


Рисунок 3.1 – Інтерфейс середовища розробки Qt Creator

Eclipse – безкоштовне інтегроване модульне середовище розробки програмного забезпечення [19]. Розробляється і підтримується Eclipse Foundation і включає проекти, такі як платформа Eclipse, набір інструментів для розробників на мові Java, візуальні конструктори GUI тощо. Написаний в основному на Java, він може бути використаний для розробки додатків на Java і, за допомогою різних плагінів, на інших мовах програмування, включаючи C, C++, Ada, Fortran, COBOL, PHP, Perl, Python, R, Ruby (включно з каркасом Ruby on Rails), Clojure, Scala та Scheme. Розробник – Eclipse Foundation., ліцензування – безкоштовне, поточна версія – 4.8.

Eclipse являє собою фреймворк із таким переліком особливостей:

- можливість розробки програмного забезпечення на багатьох мовах програмування (рідною є Java);
- модульна, призначена для подальшого розширення незалежним розробниками.
- крос-платформна;

Інтерфейс середовища розробки Eclipse наведено на рисунку 3.2.

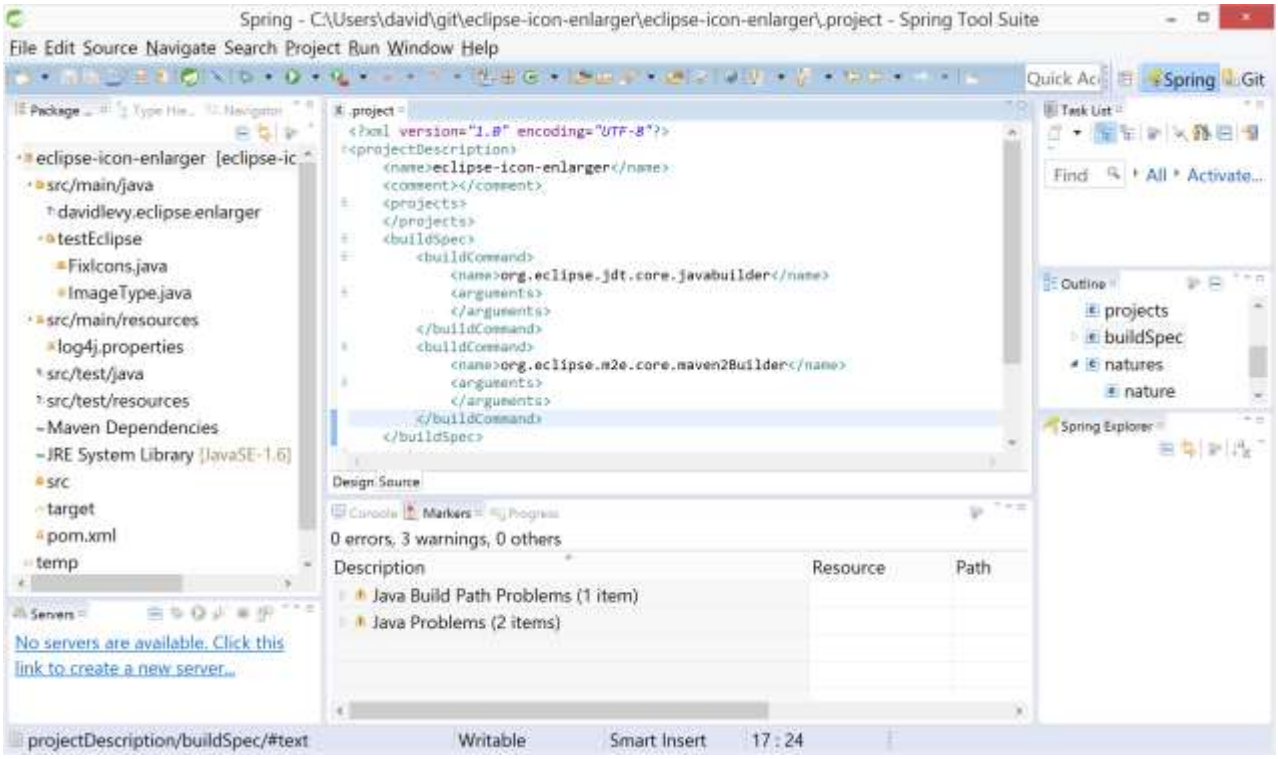


Рисунок 3.2 – Інтерфейс середовища розробки Eclipse

Visual Studio 2019 – кросплатформне середовище розробки від корпорації Microsoft, що дає змогу розробляти як консольні й графічні додатки для настільних комп’ютерів, так і веб-додатки, веб-сайти, веб-служби, а також мобільні додатки [21]. Visual Studio дає змогу розробляти графічний інтерфейс додатків за допомогою фреймворків Windows Forms та WCF (Windows Presentation Foundation). MSVS має три збірки для розробників з різним професійним рівнем, а саме: Community, Professional, Enterprise, що містять можливості створення діаграм класів та функції юніт-тестування коду. Visual Studio підтримує 36 різних мов програмування, серед яких найбільш використовуваними є: Visual Basic C, JavaScript, C++, .NET, C#, F#, XML, HTML, TypeScript та CSS. Програма створена компанією Microsoft, поточна версія – 16.6.0.

Інтерфейс середовища розробки Visual Studio 2019 Professional наведено на рисунку 3.3.

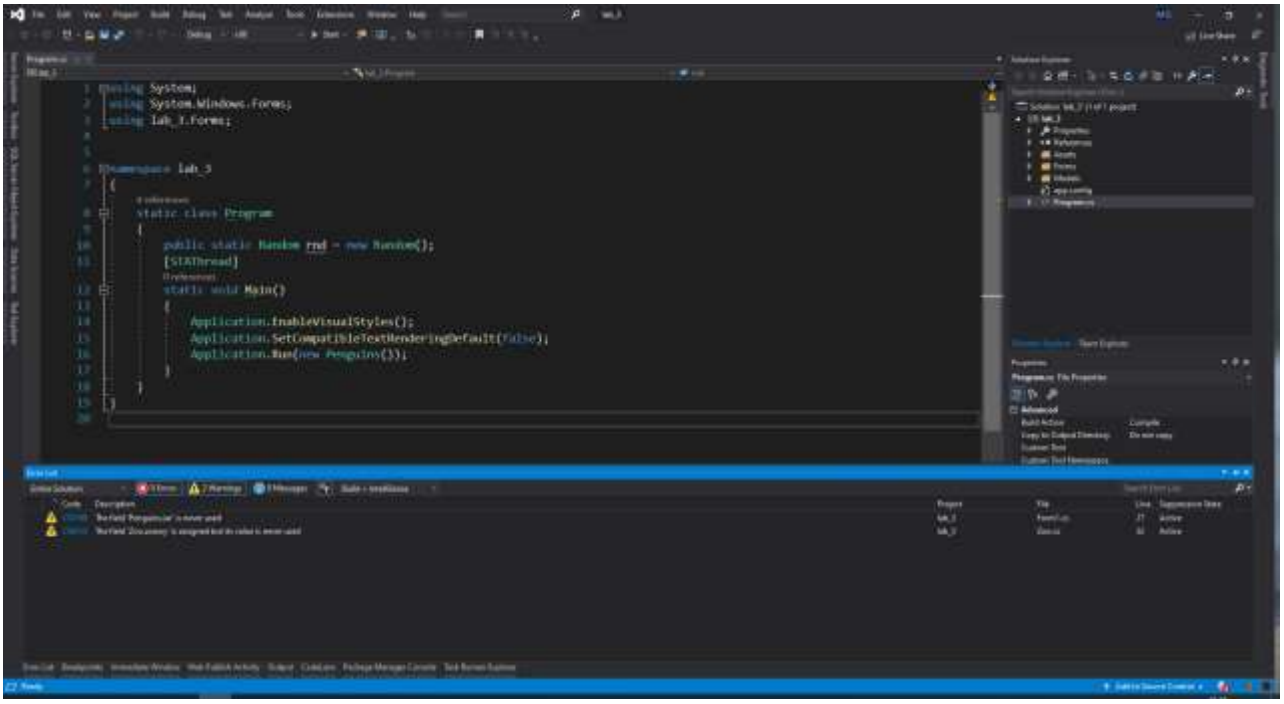


Рисунок 3.3 – Інтерфейс середовища розробки Visual Studio 2019 Professional

Для того, щоб обрати, яке саме інтегроване середовище розробки найбільше підходить для реалізації програми розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі, було складено таблицю 3.1 з порівнянням вище описаних IDE:

Таблиця 3.1 – Порівняння інструментальних засобів розробки

	Qt Creator	Eclipse	Visual Studio
Статичний аналіз коду	-	-	+
Підтримка роздільних вікон для декількох моніторів	-	+	+
Вбудовані засоби для створення та тренування нейронних мереж	-	-	+
Оптимізація коду	+	-	+
Вбудована підтримка системи контролю версій Git	-	-	+

Аналізуючи дані таблиці, середовище розробки Visual Studio краще за аналоги, тому даний засіб розробки є оптимальним для реалізації програми розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі, оскільки містить усі необхідні можливості для її реалізації.

Для вибору мови програмування було складено таблицю 3.2 з порівнянням популярних об’єктно-орієнтованих мов програмування.

Таблиця 3.2 - Порівняльний аналіз мов програмування

Мова / Характеристика	Python	Java	C#
Автоматична збірка сміття	+	+	+
Кросплатформність додатків	–	+	+
Перевантаження операторів	+	–	+
Підтримка бібліотек штучного інтелекту	+	–	+

Мова C# заслужено вважається однією з кращих мов програмування багатоцільового призначення [22]. Саме через ці переваги, наведені в таблиці 3.2, саме її було обрано для реалізації програми розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі.

.NET Core – це модульна платформа для розробки програмного забезпечення з відкритим вихідним кодом, заснована на .NET Framework. Сумісна з такими операційними системами як Windows, Linux і macOS. Модульність .NET Core працює таким чином, що кожна програма може

працювати з різними модулями і не залежить від єдиного оновлення платформи.

Важливою особливістю платформи .NET Core при розробці проекту є те, що до її складу входить бібліотека ML.NET [23]. Ця бібліотека надає можливість створювати алгоритми з використанням штучного інтелекту та підключати нейромережі, розроблені за допомогою інших технологій та мов програмування.

3.2 Програмна реалізація розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі

Модуль пошуку та розпізнавання об’єктів на зображеннях названий класом ModelScorer і містить два методи – LoadModel та Score. Метод LoadModel визначає послідовність функцій бібліотеки ML.NET для створення конвеєра розпізнавань, а метод Score – отримує вхідні дані та за допомогою створеного конвеєра розпізнавань повертає масив чисел з імовірностями для кожного з 80 класів об’єктів розпізнавання. Код класу з даними методами наведено нижче.

```
public class ModelScorer : IModelScorer
{
    private readonly MLContext mlContext;
    private readonly PredictionEngine<ImageNetData, IPrediction> predictionEngine;

    public ModelScorer(IModel onnxModel)
    {
        this.mlContext = new MLContext();
        this.predictionEngine = LoadModel(onnxModel);
    }

    public float[] Score(ImageNetData image) =>
this.predictionEngine.Predict(image).PredictedLabels;

    private PredictionEngine<ImageNetData, IPrediction> LoadModel(IModel onnxModel)
    {
```

```
        Console.WriteLine("Завантаження моделі.");
        Console.WriteLine($"Розташування моделі: {onnxModel.ModelPath}");
        var data = mlContext.Data.LoadFromEnumerable(new List<ImageNetData>());
        var pipeline = mlContext.Transforms.LoadImages(
                outputColumnName: onnxModel.ModelInput,
                imageFolder: "",
                inputColumnName: nameof(ImageNetData.ImagePath))
            .Append(mlContext.Transforms.ResizeImages(
                outputColumnName: onnxModel.ModelInput,
                imageWidth: 416,
                imageHeight: 416,
                inputColumnName: onnxModel.ModelInput,
                resizing: ImageResizingEstimator.ResizingKind.Fill))
            .Append(mlContext.Transforms.ExtractPixels(
                outputColumnName: onnxModel.ModelInput,
                inputColumnName: onnxModel.ModelInput))
            .Append(mlContext.Transforms.ApplyOnnxModel(
                outputColumnName: onnxModel.ModelOutput,
                inputColumnName: onnxModel.ModelInput,
                modelFile: onnxModel.ModelPath));

        var model = pipeline.Fit(data);

        var predictionEng = mlContext.Model.CreatePredictionEngine<ImageNetData,
IPrediction>(model);

        return predictionEng;
    }
}
```

3.3 Програмна реалізація модуля аналізу та візуалізації вихідних даних після обробки

Наступним етапом розробки програми розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі є створення класів та функцій для аналізу та візуалізації вихідних даних після обробки.

Спочатку потрібно створити клас OutputParser, який міститиме усі необхідні поля та методи для аналізу даних. Код класу наведено на рис. 3.4.

```
public class OutputParser
{
    // Кількість рядків та сповпців у сітці, на яку поділяється зображення
    public const int ROW_COUNT = 13;
    public const int COL_COUNT = 13;
    // Кількість елементів, що містяться у комірці (x, y, height, width,
confidence).
    public const int BOX_INFO_FEATURE_COUNT = 5;
    // Назви 80 класів об’єктів, які може розпізнавати модель
    private readonly string[] classLabels;
    // Заздалегідь визначені розміри обмежуючих прямокутників у комірці
    private readonly (float x, float y)[] boxAnchors;

    public OutputParser(IOnnxModel<(float, float)> onnxModel)
    {
        boxAnchors = onnxModel.Anchors;
        classLabels = new[]
        {

            "людина", "велосипед", "машина", "мотоцикл", "літак", "автобус",
"поїзд",  "вантажівка", "човен", "світлофор", "пожежний гідрант", "знак зупинки",
"паркомат", "лавка", "птах", "кіт", "собака", "кінь", "вівця", "корова", "слон",
"ведмідь", "зебра", "жирафа", "рюкзак", "парасолька", "сумочка", "краватка",
"валіза", "фрісбі", "лижі", "сноуборд", "спортивний м'яч", "повітряний змій",
"бейсбольна бита", "бейсбольна рукавиця", "скейтборд", "дошка для серфінгу",
тенісна ракетка", "пляшка", "келих", "чашка", "виделка", "ніж", "ложка", "миска",
"банан", яблуко", "сендвіч", "апельчин", "брокколі", "морква", "хот-дог", "піца",
"пончик", "пиріг", стілець", "диван", "вазон", "ліжко", "стіл", "туалет",
"телевізор", "ноутбук", "миша", "пульт", "клавіатура", "мобільник",
"мікрохвильовка", "духовка", "тостер", "раковина", "холодильник", "книга",
"годинник", "ваза", "ножиці", "плюшевий ведмедик", "фен", "зубна щітка"
        };
    }
}
```

Рисунок 3.4 – Фрагмент коду класу OutputParser

Для навчання згорткової нейронної\ мережі розпізнаванню 80 класів об`’єктів, перерахованих на рис. 3.4, використовувалась база зображень, взятих з «Wikimedia Commons Site.». Wikimedia Commons – це колекція з 88 814 867 вільно використовуваних медіа файлів.

Додамо в даний клас метод ParseOutputs, який буде обробляти масив вихідних даних, які створила мережа:

```
public IList<BoundingBox> ParseOutputs(float[] onnxModelOutputs, float
probabilityThreshold = .3f)
{
    var boxes = new List<BoundingBox>();
    for (int row = 0; row < ROW_COUNT; row++)
    {
        for (int column = 0; column < COL_COUNT; column++)
        {
            for (int box = 0; box < boxAnchors.Length; box++)
            {
                var channel = box * (classLabels.Length + BOX_INFO_FEATURE_COUNT);
                var boundingBoxPrediction =
ExtractBoundingBoxPrediction(onnxModelOutputs, row, column, channel);
                var mappedBoundingBox = MapBoundingBoxToCell(row, column, box,
boundingBoxPrediction);

                if (boundingBoxPrediction.Confidence < probabilityThreshold)
                    continue;
                var classProbabilities =
ExtractClassProbabilities(onnxModelOutputs, row, column, channel,
boundingBoxPrediction.Confidence);
                var (topProbability, topIndex) = classProbabilities
                    .Select((probability, index) => (Score: probability, Index:
index)).Max();
                if (topProbability < probabilityThreshold)
                    continue;
                boxes.Add(new BoundingBox
                {
                    Dimensions = mappedBoundingBox,
                    Confidence = topProbability,
                    Label = classLabels[topIndex],
                    BoxColor = BoundingBox.GetColor(topIndex)
                });
            }
        }
    }
    return boxes;
}
```

Далі, коли координати всіх обмежувальних прямокутників були добуті з вихідних даних мережі, потрібно провести додаткову фільтрацію для видалення прямокутників, які перекриваються. Додамо метод FilterBoundingBoxes під методом ParseOutputs, код якого наведено далі:

```
public IList<BoundingBox> FilterBoundingBoxes(IList<BoundingBox> boxes, int limit,
float threshold)
{
    var filteredBoxes = new bool[boxes.Count];
    var sortedBoxes = boxes.OrderByDescending(b => b.Confidence).ToArray();
    var results = new List<BoundingBox>();

  for (int i = 0; i < boxes.Count; i++)
    {
        if (filteredBoxes[i])
        {
            continue;
        }
        results.Add(sortedBoxes[i]);
        if (results.Count >= limit)
        {
            break;
        }

        for (var j = i + 1; j < boxes.Count; j++)
        {
            if (filteredBoxes[j])
            {
                continue;
            }
            if (IntersectionOverUnion(sortedBoxes[i].Rect, sortedBoxes[j].Rect) >
threshold)
            {
                filteredBoxes[j] = true;
            }
            if (filteredBoxes.Count(b => b) <= 0)
            {
                break;
            }
        }
    }
    return results;
}
```

І на останок, додамо метод DrawBoundingBox, який до вхідного зображення додає обмежувальні прямокутники за визначеними координатами, що виділяють об’єкти розпізнаних класів з їх назвою та достовірністю розпізнання, та повертає результуюче зображення користувачеві. Код методу наведено далі.

```
public Image DrawBoundingBox(string inputImageLocation, string outputImageLocation,
string imageName, Image inputImage = null)
{
    var image = inputImage ?? Image.FromFile(Path.Combine(inputImageLocation,
imageName));
    var originalImageHeight = image.Height;
    var originalImageWidth = image.Width;
    foreach (var box in this._filteredBoundingBoxes)
    {
        // Отримати розміри обмежуючого прямокутника
        var x = (uint)Math.Max(box.Dimensions.X, 0);
        var y = (uint)Math.Max(box.Dimensions.Y, 0);
        var width = (uint)Math.Min(originalImageWidth - x, box.Dimensions.Width);
        var height = (uint)Math.Min(originalImageHeight - y,
box.Dimensions.Height);

        // Змінити розмір зображення до початкового
        x = (uint)originalImageWidth * x / ImageNetSettings.imageWidth;

        y = (uint)originalImageHeight * y / ImageNetSettings.imageHeight;
        width = (uint)originalImageWidth * width / ImageNetSettings.imageWidth;
        height = (uint)originalImageHeight * height / ImageNetSettings.imageHeight;
        using (Graphics thumbnailGraphic = Graphics.FromImage(image))
        {
            thumbnailGraphic.CompositingQuality = CompositingQuality.HighQuality;
            thumbnailGraphic.SmoothingMode = SmoothingMode.HighQuality;
            thumbnailGraphic.InterpolationMode =
InterpolationMode.HighQualityBicubic;

            // Визначити параметри тексту
            Font drawFont = new Font("Arial", 12, FontStyle.Bold);
            SizeF size = thumbnailGraphic.MeasureString(box.Description, drawFont);
            SolidBrush fontBrush = new SolidBrush(Color.Black);
            Point atPoint = new Point((int)x, (int)y - (int)size.Height + 20);
            Pen pen = new Pen(box.BoxColor, 3.2f);
            SolidBrush colorBrush = new SolidBrush(box.BoxColor);

            // Намалювати обмежуючий прямокутник на зображенні
            thumbnailGraphic.DrawRectangle(pen, x, y, width, height);

            // Намалювати текст на зображенні
            thumbnailGraphic.FillRectangle(colorBrush, (int)x, (int)(y -
size.Height + 20), (int)size.Width, (int)size.Height);
            thumbnailGraphic.DrawString(box.Description, drawFont, fontBrush,
atPoint);
        }
    }
     if (inputImage == null)
     {
         if (!Directory.Exists(outputImageLocation))
         {
             Directory.CreateDirectory(outputImageLocation);
         }
         image.Save(Path.Combine(outputImageLocation, imageName));
     }
     return image;
}
```

3.4 Висновок до розділу 3

У третьому розділі було обґрунтовано вибір середовища розробки, мови програмування та технологій, які будуть використовуватися при розробці програми та наведено їх основні переваги. В результаті порівняльного аналізу було обрано середовище розробки Visual Studio, мову програмування C# та технології ML.NET для розробки модулів обробки зображень і .NET Core для розробки графічного інтерфейсу користувача. Було проведено реалізацію програмних модулів, а саме модуля пошуку та розпізнавання об’єктів у зображеннях та модуля аналізу та візуалізації вихідних даних після обробки.

4 ТЕСТУВАННЯ ТА АНАЛІЗ РЕЗУЛЬТАТІВ РОБОТИ ПРОГРАМИ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ

4.1 Аналіз методів тестування

Тестування програмного забезпечення – це процес технічного аналізу, призначенням якого є виявлення інформації відносно якості продукту щодо контексту його використання [24]. Техніка тестування також містить як процес пошуку помилок та дефектів, так і випробування програмних складових з метою відповідності завданню. Може оцінюватись: узгодженість з вимогами, якими керувалися проектувальники та розробники, вірна відповідь для усіх можливих вхідних даних, виконання функцій за прийнятний час, практичність, сумісність з програмним забезпеченням та операційними системами, відповідність завданням замовника [25].

Методи тестування програмного забезпечення:

- статичне тестування;
- динамічне тестування;
- тестування «чорної скриньки».
- тестування «білої скриньки»;

Статичне тестування – тестові дії, що пов'язані з аналізом результатів розробки програмного забезпечення. Воно передбачає перевірку програмних кодів, контроль та перевірку програми без запуску на комп'ютері. На етапі статичного тестування перевіряється документація, отримана як результат життєвого циклу програми.

Динамічне тестування – тестові дії, що передбачають експлуатацію програмного продукту. Динамічне та статичне тестування доповнюють одне одного. Динамічні методи застосовуються в процесі безпосереднього виконання програми на комп'ютері. Коректність програмного засобу перевіряється на множині тестів або наборів підготовлених вхідних даних.

При виконанні кожного тесту збираються та аналізуються дані про відмови та збої в роботі програми.

Тестування «чорної скриньки» – вид тестування, у якому відомі функції додатку і досліджується робота кожної функції на всій області визначення. При тестуванні «чорної скриньки» розглядаються системні характеристики програм, не враховується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе. Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^{10}. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок, а забезпечує пошук наступних категорій помилок:

- помилки інтерфейсу;
- некоректні чи відсутні функції;
- помилки у зовнішніх структурах даних або в доступі до бази даних;
- помилки характеристик (потрібна ємність пам'яті тощо).

Подібні категорії помилок способом «білої скриньки» не виявляються.

Тестування «білої скриньки» – тестування, у якому відома внутрішня структура програми, а досліджуються внутрішні елементи програми і зв’язки між ними. Об'єктом тестування тут є не зовнішня, а внутрішня поведінка програми. Перевіряється коректність побудови всіх елементів програми та вірність їхньої взаємодії один з одним. Зазвичай аналізуються керуючі зв'язки елементів, рідше – інформаційні зв'язки. Тестування «білої скриньки» характеризується ступенем, в якому тести виконують або покривають логіку (вихідний код) програми.

У цьому випадку формуються тестові варіанти, в яких:

- виконуються всі цикли (у межах їхніх кордонів та діапазонів);
- гарантується перевірка всіх незалежних маршрутів програми;
- аналізується правильність внутрішніх структур даних.
- знаходяться гілки True, False для всіх логічних рішень;

При тестуванні розробленої програми, використовується тестування за методом «чорної скриньки», так як воно найкраще підходить для тестування готових додатків, що не передбачають доступ до вихідного коду тестувальником.

4.2 Процес тестування програми

Для тестування програми за методом «чорної скриньки» було розроблено і виконано варіанти тестування для основних функцій та варіантів використання, доступних для користувача. Результати тестування наведені в таблиці 4.1.

Таблиця 4.1 – Варіанти тестування за методом «чорної скриньки»

№ п/п	Вхідні дані	Вихідні дані	Відповідність
1	Зображення без об’єктів	Зображення без виділених об’єктів	Відповідає (див. рис. 4.1)
2	Зображення з 1 об’єктом	Зображення з 1 виділеним об’єктом	Відповідає (див. рис. 4.2)
3	Зображення з 2 об’єктами	Зображення з 2 виділеними об’єктами	Відповідає (див. рис. 4.3)
4	Зображення з 3 об’єктами	Зображення з 3 виділеними об’єктами	Відповідає (див. рис. 4.4)
5	Зображення з повними та неповними об’єктами	Зображення з виділеними повними та неповними об’єктами	Відповідає (див. рис. 4.5)
6	Зображення з безліччю об’єктів на передньому і задньому плані	Зображення з виділеними об’єктами переднього плану	Відповідає (див. рис. 4.6)

Тест 1. Після запуску програми з обраним зображенням без чітко визначених об’єктів, слід натиснути кнопку «Upload File». Очікуваний результат: програма повертає зображення без виділених об’єктів. Отриманий результат відповідає очікуваному. Результат тесту – успішний. Результат цього тесту зображено на рисунку 4.1.

Рисунок 4.1 – Результат тесту 1

Тест 2. Після запуску програми з обраним зображенням з одним чітко визначеним об’єктом, слід натиснути кнопку «Upload File». Очікуваний результат: програма повертає зображення з 1 виділеним об’єктом. Отриманий результат відповідає очікуваному. Результат тесту – успішний. Результат цього тесту зображено на рисунку 4.2.

Рисунок 4.2 – Результат тесту 2

Тест 3. Після запуску програми з обраним зображенням з двома чітко визначеними об’єктами, слід натиснути кнопку «Upload File». Очікуваний результат: програма повертає зображення з 2 виділеними об’єктами. Отриманий результат відповідає очікуваному. Результат тесту – успішний. Результат цього тесту зображено на рисунку 4.3.

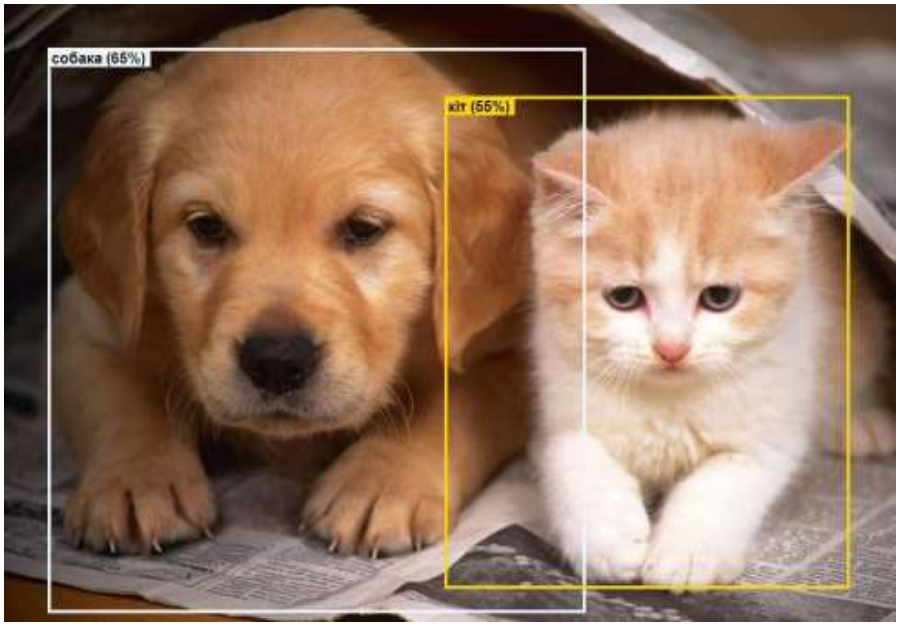


Рисунок 4.3 – Результат тесту 3

Тест 4. Після запуску програми з обраним зображенням з трьома чітко визначеними об’єктами, слід натиснути кнопку «Upload File». Очікуваний результат: програма повертає зображення з 3 виділеними об’єктами. Отриманий результат відповідає очікуваному. Результат тесту – успішний. Результат цього тесту зображено на рисунку 4.4.

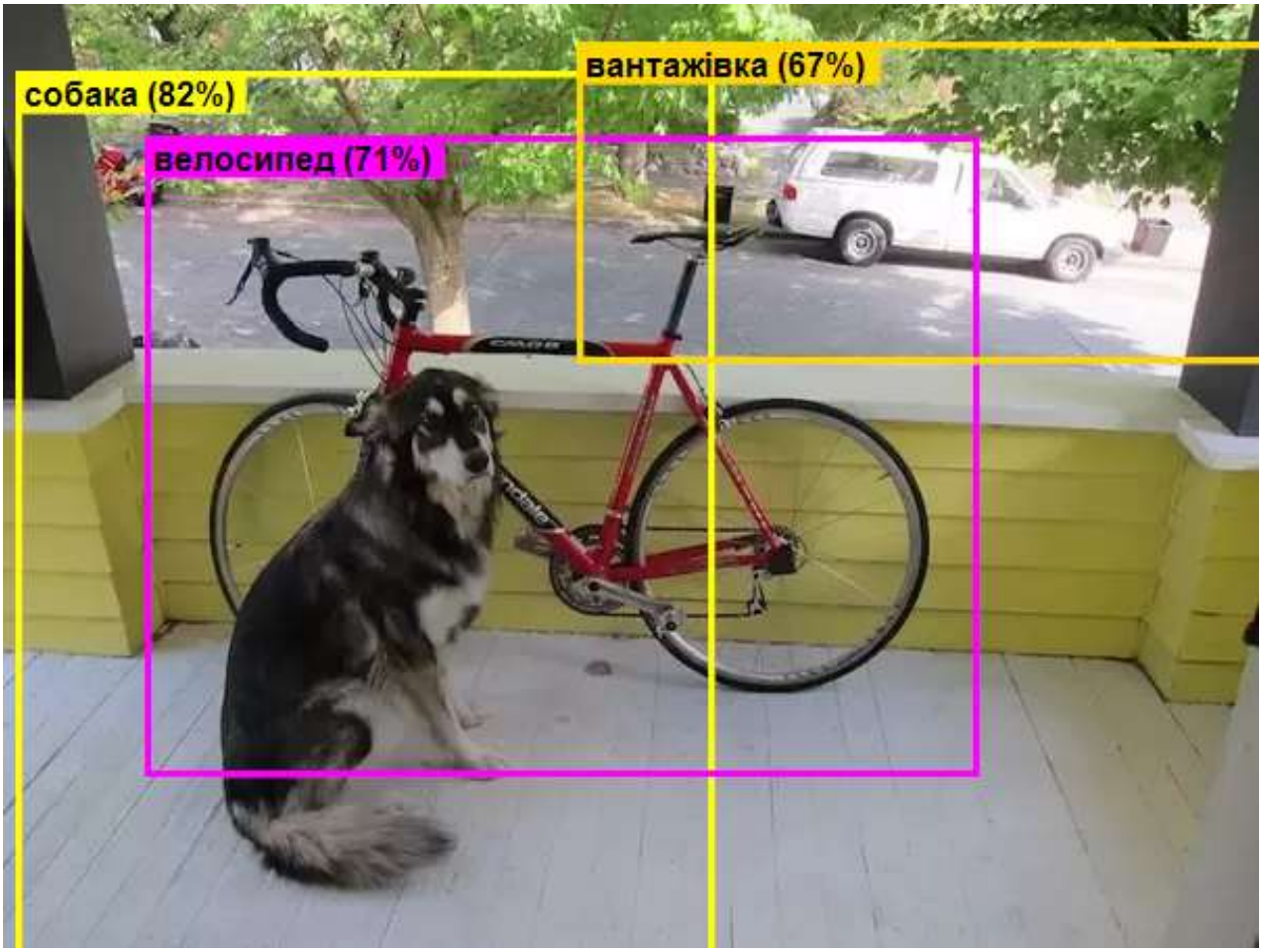


Рисунок 4.4 – Результат тесту 4

Тест 5. Після запуску програми з обраним зображенням з повними та неповними об’єктами, слід натиснути кнопку «Upload File». Очікуваний результат: програма повертає зображення з виділеними повними та неповними об’єктами. Отриманий результат відповідає очікуваному. Результат тесту – успішний. Результат цього тесту зображено на рисунку 4.5.

Рисунок 4.5 – Результат тесту 5

Тест 6. Після запуску програми з обраним зображенням з безліччю чітко визначених об’єктів на передньому і задньому плані, слід натиснути кнопку «Upload File». Очікуваний результат: програма повертає зображення з виділеними об’єктами переднього плану. Отриманий результат відповідає очікуваному. Результат тесту – успішний. Результат цього тесту зображено на рисунку 4.6.

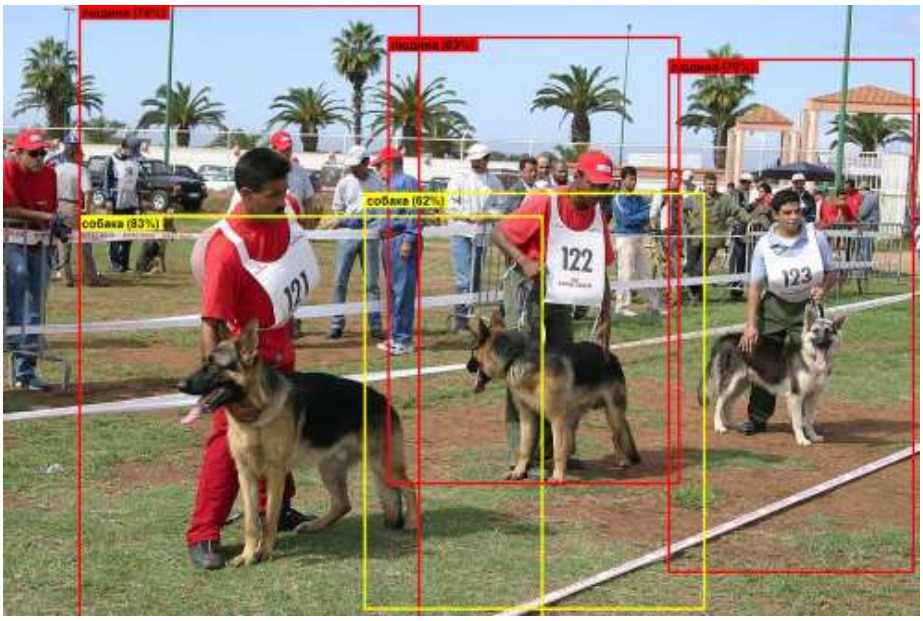


Рисунок 4.6 – Результат тесту 6

Отже, у ході тестування програми розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі було проведено 6 тестів. Згідно з результатами тестування, робота програми на 100% відповідає результатам тестів, що підтверджує її функціональність.

Для доведення досягнення поставленої в роботі мети – підвищення достовірності роботи програми розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі – було протестовано роботу розробленої програми та програми-аналога Inception [13] на 100 зображеннях із тестової вибірки. Результати тестування представлені у табл. 4.2.

Таблиця 4.2 – Результати тестування розробленої програми та програми-аналога Inception

Програмний засіб	Кількість зображень у тестовій вибірці	Сумарна кількість об’єктів на зображеннях	Кількість правильно розпізнаних об’єктів	Достовірність розпізнавання об’єктів на зображеннях, %
Програма Inception	100	240	211	87,9
Розроблений програмний модуль	100	240	229	95,4

Із табл. 4.2 видно, що розроблена програма розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі має вищу достовірність розпізнавання об’єктів на зображеннях (95,4%), ніж аналогічна програма (87,9%), а значить достовірність розпізнавання об’єктів на зображеннях покращена на 7,5%, тобто мета роботи досягнута.

Тестування розробленої програми розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі показало його надійну роботу. Програма повністю відповідає завданню.

Деякий ілюстративний матеріал до програми (у т.ч. скріншоти) подано в додатку В. Інструкцію користування розробленою програмою наведено у додатку Г.

4.3 Висновок до розділу 4

Було розглянуто різні методи тестування, та визначено, що для даної нейромережевої системи найоптимальнішим є тестування методом «чорної скриньки». Під час тестування було розглянуто декілька можливих випадків розпізнавання об’єктів у зображеннях. В результаті тестування програми було доведено її повну працездатність та відповідність поставленому завданню. Серед переваг програми можна відзначити її більшу достовірність у розпізнаванні великих об’єктів – тих, які знаходяться на передньому плані і займають певну частину зображення. До недоліків нейромережевої системи можна віднести неможливість розпізнавання дуже малих об’єктів заднього плану зображення. Розроблений програмний модуль має вищу достовірність розпізнавання об’єктів на зображеннях (95,4%), ніж аналогічна програма (87,9%), а значить достовірність розпізнавання об’єктів на зображеннях покращена на 7,5%, тобто мета роботи досягнута.

5 ЕКОНОМІЧНА ЧАСТИНА

Науково-технічна розробка має право на існування та впровадження, якщо вона відповідає вимогам часу, як в напрямку науково-технічного прогресу та і в плані економіки. Тому для науково-дослідної роботи необхідно оцінювати економічну ефективність результатів виконаної роботи.

Магістерська кваліфікаційна робота з розробки та дослідження «Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі» відноситься до науково-технічних робіт, які орієнтовані на виведення на ринок (або рішення про виведення науково-технічної розробки на ринок може бути прийнято у процесі проведення самої роботи), тобто коли відбувається так звана комерціалізація науково-технічної розробки. Цей напрямок є пріоритетним, оскільки результатами розробки можуть користуватися інші споживачі, отримуючи при цьому певний економічний ефект. Але для цього потрібно знайти потенційного інвестора, який би взявся за реалізацію цього проекту і переконати його в економічній доцільності такого кроку.

Для наведеного випадку нами мають бути виконані такі етапи робіт:

1) проведено комерційний аудит науково-технічної розробки, тобто встановлення її науково-технічного рівня та комерційного потенціалу;

2) розраховано витрати на здійснення науково-технічної розробки;

3) розрахована економічна ефективність науково-технічної розробки у випадку її впровадження і комерціалізації потенційним інвестором і проведено обґрунтування економічної доцільності комерціалізації потенційним інвестором.

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного і технологічного аудиту дослідження за темою «Інформаційна технологія розпізнавання об’єктів на зображеннях з

використанням згорткової нейронної мережі» є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Оцінювання науково-технічного рівня розробки та її комерційного потенціалу рекомендується здійснювати із застосуванням 5-ти бальної системи оцінювання за 12-ма критеріями, наведеними в табл. 5.1 [26].

Таблиця 5.1 – Рекомендовані критерії оцінювання науково-технічного рівня і комерційного потенціалу розробки та бальна оцінка

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Технічна здійсненність концепції					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено працездатність продукту в реальних умовах
Ринкові переваги (недоліки)					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в	Технічні та споживчі влас-тивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в	Технічні та споживчі властивості продукту значно кращі, ніж в
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 5.1

Бали (за 5-ти бальною шкалою)					
	0	1	2	3	4
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуютьс я у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь- які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання науково-технічного рівня та комерційного потенціалу науково-технічної розробки потрібно звести до таблиці 5.2.

Таблиця 5.2 – Результати оцінювання науково-технічного рівня і комерційного потенціалу розробки експертами

Критерії	Експерт (ПІБ, посада)		
	1	2	3
	Бали:		
1. Технічна здійсненність концепції	5	4	5
2. Ринкові переваги (наявність аналогів)	3	4	3
3. Ринкові переваги (ціна продукту)	4	3	3
4. Ринкові переваги (технічні властивості)	4	4	5
5. Ринкові переваги (експлуатаційні витрати)	1	1	1
6. Ринкові перспективи (розмір ринку)	3	3	3
7. Ринкові перспективи (конкуренція)	3	2	3
8. Практична здійсненність (наявність фахівців)	4	4	3
9. Практична здійсненність (наявність фінансів)	3	4	3
10. Практична здійсненність (необхідність нових матеріалів)	5	5	4
11. Практична здійсненність (термін реалізації)	5	4	4
12. Практична здійсненність (розробка документів)	5	4	5
Сума балів	45	42	42
Середньоарифметична сума балів $СБ_c$	43,0		

За результатами розрахунків, наведених в таблиці 5.2, зробимо висновок щодо науково-технічного рівня і рівня комерційного потенціалу розробки. При цьому використаємо рекомендації, наведені в табл. 5.3 [26].

Таблиця 5.3 – Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Науково-технічний рівень та комерційний потенціал розробки
41…48	Високий
31…40	Вище середнього
21…30	Середній
11…20	Нижче середнього
0…10	Низький

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі» становить 43,0 бала, що,

відповідно до таблиці 5.3, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий).

5.2 Визначення рівня конкурентоспроможності розробки

В процесі визначення економічної ефективності науково-технічної розробки також доцільно провести прогноз рівня її конкурентоспроможності за сукупністю параметрів, що підлягають оцінюванню.

Одиничний параметричний індекс розраховуємо за формулою [26]:

$$q_i = \frac{P_i}{P_{базі}}. \qquad (5.1)$$

де q_i – одиничний параметричний індекс, розрахований за i-м параметром;

P_i – значення i-го параметра виробу;

$P_{базі}$ – аналогічний параметр базового виробу-аналога, з яким проводиться порівняння.

Загальні технічні та економічні характеристики розробки представлено в таблиці 5.4.

Нормативні параметри оцінюємо показником, який отримує одне з двох значень: 1 – пристрій відповідає нормам і стандартам; 0 – не відповідає.

Груповий показник конкурентоспроможності за нормативними параметрами розраховуємо як добуток частинних показників за кожним параметром за формулою [26]:

$$I_{НП} = \prod_{i=1}^{n} q_i \ , \qquad (5.2)$$

де $I_{нп}$ – загальний показник конкурентоспроможності за нормативними параметрами;

q_i – одиничний (частинний) показник за i-м нормативним параметром;

n – кількість нормативних параметрів, які підлягають оцінюванню.

Таблиця 5.4 – Основні техніко-економічні показники аналога та розробки, що проектується

Показники (параметри)	Одиниця вимірю-вання	Аналог	Проектований пристрій	Відношення параметрів нової розробки до аналога	Питома вага показника
Вимоги до оперативної пам'яті	GB	4	2	2	0,15
Рекомендоване місце на диску	GB	6	3	2	0,1
Точність розпізнавання	%	87	94	1,1	0,3
Вимоги до CPU	бал	7	8	0,87	0,25
Захищеність БД від зовнішнього впливу (до 10 балів)	бал	7	7	1	0,2
Експлуатаційні витрати	грн	520	300	0,58	0,4
Ціна	грн	12500	7800	0,624	0,6

За нормативними параметрами розроблюваний пристрій відповідає вимогам ДСТУ, тому $I_{нп} = 1$.

Значення групового параметричного індексу за технічними параметрами визначаємо з урахуванням вагомості (частки) кожного параметра [26]:

$$I_{ТП} = \sum_{i=1}^{n} q_i \cdot \alpha_i \,, \tag{5.3}$$

де $I_{тп}$ – груповий параметричний індекс за технічними показниками (порівняно з виробом-аналогом);

q_i – одиничний параметричний показник i-го параметра;

α_i – вагомість i-го параметричного показника, $\sum_{i=1}^{n} \alpha_i = 1$;

n – кількість технічних параметрів, за якими оцінюється конкурентоспроможність.

Проведемо аналіз параметрів згідно даних таблиці 5.4.

$$I_{mn} = 2 \cdot 0{,}15 + 2 \cdot 0{,}1 + 1{,}1 \cdot 0{,}3 + 0{,}87 \cdot 0{,}25 + 1 \cdot 0{,}2 = 1{,}25.$$

Груповий параметричний індекс за економічними параметрами розраховуємо за формулою [26]:

$$I_{ЕП} = \sum_{i=1}^{m} q_i \cdot \beta_i \,, \tag{5.4}$$

де $I_{ЕП}$ – груповий параметричний індекс за економічними показниками;

q_i – економічний параметр i-го виду;

β_i – частка i-го економічного параметра, $\sum_{i=1}^{m} \beta_i = 1$;

m – кількість економічних параметрів, за якими здійснюється оцінювання.

Проведемо аналіз параметрів згідно даних таблиці .

$$I_{ЕП} = 0{,}58 \cdot 0{,}4 + 0{,}624 \cdot 0{,}6 = 0{,}61.$$

На основі групових параметричних індексів за нормативними, технічними та економічними показниками розрахуємо інтегральний показник конкурентоспроможності за формулою [26]:

$$K_{ІНТ} = I_{НП} \cdot \frac{I_{ТП}}{I_{ЕП}}, \tag{5.5}$$

$$K_{ІНТ} = 1 \cdot 1{,}25 / 0{,}61 = 2{,}06.$$

Інтегральний показник конкурентоспроможності $К_{ІНТ} > 1$, отже розробка переважає відомі аналоги за своїми техніко-економічними показниками.

5.3 Розрахунок витрат на проведення науково-дослідної роботи

Витрати, пов’язані з проведенням науково-дослідної роботи на тему «Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі», під час планування, обліку і калькулювання собівартості науково-дослідної роботи групуємо за відповідними статтями.

5.3.1 Витрати на оплату праці

До статті «Витрати на оплату праці» належать витрати на виплату основної та додаткової заробітної плати керівникам відділів, лабораторій, секторів і груп, науковим, інженерно-технічним працівникам, конструкторам, технологам, креслярам, копіювальникам, лаборантам, робітникам, студентам, аспірантам та іншим працівникам, безпосередньо зайнятим виконанням конкретної теми, обчисленої за посадовими окладами, відрядними розцінками, тарифними ставками згідно з чинними в організаціях системами оплати праці.

Основна заробітна плата дослідників

Витрати на основну заробітну плату дослідників ($З_о$) розраховуємо у відповідності до посадових окладів працівників, за формулою [26]:

$$З_о = \sum_{i=1}^{k} \frac{M_{ni} \cdot t_i}{T_p}, \qquad (5.6)$$

де k – кількість посад дослідників залучених до процесу досліджень;

M_{ni} – місячний посадовий оклад конкретного дослідника, грн;

t_i – число днів роботи конкретного дослідника, дн.;

T_p – середнє число робочих днів в місяці, T_p=20 дні.

$$З_o = 15220{,}00 \cdot 42 / 20 = 31962{,}00 \; грн.$$

Проведені розрахунки зведемо до таблиці 5.5.

.

Таблиця 5.5 – Витрати на заробітну плату дослідників

Найменування посади	Місячний посадовий оклад, грн	Оплата за робочий день, грн	Число днів роботи	Витрати на заробітну плату, грн
Керівник проекту	15220,00	761,00	42	31962,00
Інженер-розробник програмного забезпечення	14580,00	729,00	28	20412,00
Консультант	17800,00	890,00	5	4450,00
Лаборант	7040,00	352,00	15	5280,00
Всього				62104,00

Основна заробітна плата робітників

Витрати на основну заробітну плату робітників ($З_р$) за відповідними найменуваннями робіт НДР на тему «Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі» розраховуємо за формулою:

$$З_p = \sum_{i=1}^{n} C_i \cdot t_i , \qquad (5.7)$$

де C_i – погодинна тарифна ставка робітника відповідного розряду, за виконану відповідну роботу, грн/год;

t_i – час роботи робітника при виконанні визначеної роботи, год.

Погодинну тарифну ставку робітника відповідного розряду C_i можна визначити за формулою:

$$C_i = \frac{M_M \cdot K_i \cdot K_c}{T_p \cdot t_{зм}} , \qquad (5.8)$$

де $М_М$ – розмір прожиткового мінімуму працездатної особи, або мінімальної місячної заробітної плати (в залежності від діючого законодавства), приймемо $Мм$=6700,00 грн;

$К_і$ – коефіцієнт міжкваліфікаційного співвідношення для встановлення тарифної ставки робітнику відповідного розряду (табл. Б.2, додаток Б) [27];

$К_с$ – мінімальний коефіцієнт співвідношень місячних тарифних ставок робітників першого розряду з нормальними умовами праці виробничих об'єднань і підприємств до законодавчо встановленого розміру мінімальної заробітної плати.

$Т_р$ – середнє число робочих днів в місяці, приблизно $Т_р$ = 20 дн;

$t_{зм}$ – тривалість зміни, год.

$С_1 = 6700{,}00 \cdot 1{,}10 \cdot 1{,}7 / (20 \cdot 8) = 78{,}31$ *грн.*

$З_{р1} = 78{,}31 \cdot 12{,}00 = 939{,}68$ *грн.*

Таблиця 5.6 – Величина витрат на основну заробітну плату робітників

Найменування робіт	Тривалість роботи, год	Розряд роботи	Тарифний коефіцієнт	Погодинна тарифна ставка, грн	Величина оплати на робітника грн
Установка обладнання для розробки програмного забезпечення	12,00	2	1,10	78,31	939,68
Підготовка робочого місця розробника програмного забезпечення	8,50	3	1,35	96,10	816,88
Інсталяція програмного забезпечення для розпізнавання зображення	6,20	5	1,70	121,02	750,32
Компіляція програмних блоків	8,00	4	1,50	106,78	854,25
Налагодження програмних блоків	4,00	5	1,70	121,02	484,08
Формування бази даних нейромережі	15,00	4	1,50	106,78	1601,72
Попереднє тренування нейромережі	60,00	3	1,35	96,10	5766,19
Всього					11213,10

Додаткова заробітна плата дослідників та робітників

Додаткову заробітну плату розраховуємо як 10 … 12% від суми основної заробітної плати дослідників та робітників за формулою:

$$З_{дод} = (З_о + З_р) \cdot \frac{Н_{дод}}{100\%}, \quad (5.9)$$

де $Н_{дод}$ – норма нарахування додаткової заробітної плати. Приймемо 10%.

$З_{дод}$ = (62104,00 + 11213,10) · 10 / 100% = 7331,71 грн.

5.3.2 Відрахування на соціальні заходи

Нарахування на заробітну плату дослідників та робітників розраховуємо як 22% від суми основної та додаткової заробітної плати дослідників і робітників за формулою:

$$З_н = (З_о + З_р + З_{дод}) \cdot \frac{Н_{зп}}{100\%} \quad (5.10)$$

де $Н_{зп}$ – норма нарахування на заробітну плату. Приймаємо 22%.

Зн = (62104,00 + 11213,10 + 7331,71) · 22 / 100% = 17742,74 грн.

5.3.3 Сировина та матеріали

До статті «Сировина та матеріали» належать витрати на сировину, основні та допоміжні матеріали, інструменти, пристрої та інші засоби і предмети праці, які придбані у сторонніх підприємств, установ і організацій та витрачені на проведення досліджень за темою «Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі».

Витрати на матеріали (*М*), у вартісному вираженні розраховуються окремо по кожному виду матеріалів за формулою:

$$M = \sum_{j=1}^{n} H_j \cdot Ц_j \cdot K_j - \sum_{j=1}^{n} B_j \cdot Ц_{вj}, \qquad (5.11)$$

де H_j – норма витрат матеріалу j-го найменування, кг;

n – кількість видів матеріалів;

$Ц_j$ – вартість матеріалу j-го найменування, грн/кг;

K_j – коефіцієнт транспортних витрат, (K_j = *1,1 ... 1,15*);

B_j – маса відходів j-го найменування, кг;

$Ц_{вj}$ – вартість відходів j-го найменування, грн/кг.

$M_1 = 4,0 \cdot 295,00 \cdot 1,12 - 0,000 \cdot 0,00 = 1321,60$ *грн.*

Проведені розрахунки зведемо до таблиці 5.7.

Таблиця 5.7 – Витрати на матеріали

Найменування матеріалу, марка, тип, сорт	Ціна за 1 кг, грн	Норма витрат, кг	Величина відходів, кг	Ціна відходів, грн/кг	Вартість витраченого матеріалу, грн
Офісний папір SOFT Ultra Plus	295,00	4,0	0	0	1321,60
Папір для записів Parers SOFT Light A5	136,00	3,0	0	0	456,96
Органайзер офісний OFFICE SOFT	198,00	3,0	0	0	665,28
Канцелярське приладдя (набір офісного працівника)	185,00	4,0	0	0	828,80
Картридж для принтера Erixon EZ2500	860,00	2,0	0	0	1926,40
Диск оптичний NewOice CD-RW	22,00	3,0	0	0	73,92
Flesh-пам'ять Kingston 64 GB	623,00	1,0	0	0	697,76
Тека для паперів BOSS BOX	105,00	3,0	0	0	352,80
Всього					6323,52

5.3.4 Розрахунок витрат на комплектуючі

Витрати на комплектуючі *($К_в$)*, які використовують при проведенні НДР на тему «Інформаційна технологія розпізнавання об'єктів на зображеннях з використанням згорткової нейронної мережі» не передбачені.

5.3.5 Спецустаткування для наукових (експериментальних) робіт

До статті «Спецустаткування для наукових (експериментальних) робіт» належать витрати на виготовлення та придбання спецустаткування необхідного для проведення досліджень, також витрати на їх проектування, виготовлення, транспортування, монтаж та встановлення.

Балансову вартість спецустаткування розраховуємо за формулою:

$$B_{спец} = \sum_{i=1}^{k} Ц_i \cdot C_{пр.i} \cdot K_i \ , \qquad (5.12)$$

де $Ц_i$ – ціна придбання одиниці спецустаткування даного виду, марки, грн;

$C_{пр.i}$ –кількість одиниць устаткування відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує доставку, монтаж, налагодження устаткування тощо, ($K_i = 1,10…1,12$);

k – кількість найменувань устаткування.

$B_{спец} = 45430,00 \cdot 1 \cdot 1,11 = 50427,30$ *грн.*

Отримані результати зведемо до таблиці 5.8:

Таблиця 5.8 – Витрати на придбання спецустаткування по кожному виду

Найменування устаткування	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
Мультимедіа проектор SAMSUNG PLC-XW300	1	45430,00	50427,30
Всього			50427,30

5.3.6 Програмне забезпечення для наукових (експериментальних) робіт

До статті «Програмне забезпечення для наукових (експериментальних) робіт» належать витрати на розробку та придбання спеціальних програмних засобів і програмного забезпечення, (програм, алгоритмів, баз даних) необхідних для проведення досліджень, також витрати на їх проектування, формування та встановлення.

Балансову вартість програмного забезпечення розраховуємо за формулою:

$$B_{прг} = \sum_{i=1}^{k} Ц_{iпрг} \cdot C_{прг.i} \cdot K_i \ , \qquad (5.13)$$

де $Ц_{iпрг}$ – ціна придбання одиниці програмного засобу даного виду, грн;

$C_{прг.i}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

K_i – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо, (K_i = *1,10…1,12);*

k – кількість найменувань програмних засобів.

$B_{прг} = 8410{,}00 \cdot 1 \cdot 1{,}11 = 9335{,}10$ *грн.*

Отримані результати зведемо до таблиці:

Таблиця 5.9 – Витрати на придбання програмних засобів по кожному виду

Найменування програмного засобу	Кількість, шт	Ціна за одиницю, грн	Вартість, грн
ОС Windows	1	8410,00	9335,10
Прикладний пакет Microsoft Office	1	7770,00	8624,70
середовище розробки Visual Studio	1	11320,00	12565,20
Згорткова нейронна мережа	1	13400,00	14874,00
Програмне забезпечення програмування C# та технології ML.NET	1	5100,00	5661,00
Програмне забезпечення програмування .NET Core	1	3200,00	3552,00
Всього			54612,00

5.3.7 Амортизація обладнання, програмних засобів та приміщень

В спрощеному вигляді амортизаційні відрахування по кожному виду обладнання, приміщень та програмному забезпеченню тощо, розраховуємо з використанням прямолінійного методу амортизації за формулою:

$$A_{обл} = \frac{Ц_б}{T_в} \cdot \frac{t_{вик}}{12}, \quad (5.14)$$

де $Ц_б$ – балансова вартість обладнання, програмних засобів, приміщень тощо, які використовувались для проведення досліджень, грн;

$t_{вик}$ – термін використання обладнання, програмних засобів, приміщень під час досліджень, місяців;

$T_в$ – строк корисного використання обладнання, програмних засобів, приміщень тощо, років.

$A_{обл} = (24800{,}00 \cdot 2) / (3 \cdot 12) = 1377{,}78$ *грн.*

Проведені розрахунки зведемо до таблиці 5.10.

Таблиця 5.10 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використання, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Персональний комп'ютер розробника програмного забезпечення	24800,00	3	2	1377,78
Робоче місце інженера-програміста	8250,00	3	2	458,33
Пристрої передачі даних	6540,00	6	2	181,67
Оргтехніка	7850,00	5	2	261,67
Приміщення лабора-торії розробки інформаційних систем	205000,00	20	2	1708,33
Електронно-обчислювальний комплекс обробки зображень на базі QUBE QB i9 10900KF RTX 3060 TI 8GB 1642	75900,00	3	2	4216,67
Всього				8204,44

5.3.8 Паливо та енергія для науково-виробничих цілей

Витрати на силову електроенергію (B_e) розраховуємо за формулою:

$$B_e = \sum_{i=1}^{n} \frac{W_{yi} \cdot t_i \cdot Ц_e \cdot K_{вni}}{\eta_i}, \quad (5.15)$$

де W_{yi} – встановлена потужність обладнання на визначеному етапі розробки, кВт; t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_e$ – вартість 1 кВт-години електроенергії, грн; (вартість електроенергії визначається за даними енергопостачальної компанії), приймемо $Ц_e$ = 6,20 грн;

$K_{вni}$ – коефіцієнт, що враховує використання потужності, $K_{вni} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

$B_e = 0,45 \cdot 320,0 \cdot 6,20 \cdot 0,95 / 0,97 = 892,80$ *грн.*

Проведені розрахунки зведемо до таблиці 5.11.

Таблиця 5.11 – Витрати на електроенергію

Найменування обладнання	Встановлена потужність, кВт	Тривалість роботи, год	Сума, грн
Персональний комп'ютер розробника програмного забезпечення	0,45	320,0	892,80
Робоче місце інженера-програміста	0,25	280,0	434,00
Пристрої передачі даних	0,03	160,0	29,76
Оргтехніка	0,65	10,0	40,30
Електронно-обчислювальний комплекс обробки зображень на базі QUBE QB i9 10900KF RTX 3060 TI 8GB 1642	0,75	180,0	837,00
Мультимедіа проектор SAMSUNG PLC-XW300	0,06	100,0	37,20
Всього			2271,06

5.3.9 Службові відрядження

До статті «Службові відрядження» дослідної роботи на тему «Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов’язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з’їзди, конференції, наради, пов’язані з виконанням конкретних досліджень.

Витрати за статтею «Службові відрядження» розраховуємо як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100\%}, \qquad (5.16)$$

де $Н_{св}$ – норма нарахування за статтею «Службові відрядження», приймемо $Н_{св}$ = 20%.

$В_{св}$ = (62104,00 + 11213,10) · 20 / 100% = 14663,42 грн.

5.3.10 Витрати на роботи, які виконують сторонні підприємства, установи і організації

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуємо як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$B_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100\%}, \qquad (5.17)$$

де $Н_{сп}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації», приймемо $Н_{сп}$= 40%.

$В_{сп}$ = (62104,00 + 11213,10) · 40 / 100% = 29326,84 грн.

5.3.11 Інші витрати

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуємо як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.18)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати», приймемо $Н_{ів}$ = 75%.

$І_в$ = (62104,00 + 11213,10) · 75 / 100% = 54987,82 грн.

5.3.12 Накладні (загальновиробничі) витрати

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін.

Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуємо як 100…150% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.19)$$

де $H_{нзв}$ – норма нарахування за статтею «Накладні (загальновиробничі) витрати», приймемо $Н_{нзв}$ = 125%.

$B_{нзв} = (62104,00 + 11213,10) \cdot 125 / 100\% = 91646,37$ *грн.*

Витрати на проведення науково-дослідної роботи на тему «Інформаційна технологія розпізнавання об'єктів на зображеннях з використанням згорткової нейронної мережі» розраховуємо як суму всіх попередніх статей витрат за формулою:

$$B_{заг} = З_о + З_р + З_{дод} + З_н + М + К_в + B_{спец} + B_{прг} + A_{обл} + B_е + B_{св} + B_{сп} + I_в + B_{нзв}. \quad (5.20)$$

$B_{заг}$= *62104,00 +11213,10 +7331,71 +17742,73797 +6323,52 +0,00 +50427,30 + 54612,00 + 8204,44 +2271,06 +14663,42 +29326,84 +54987,82 +91646,37 = 410854,33 грн.*

Загальні витрати *ЗВ* на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховується за формулою:

$$ЗВ = \frac{B_{заг}}{\eta}, \quad (5.21)$$

де η - коефіцієнт, який характеризує етап (стадію) виконання науково-дослідної роботи, приймемо η=0,9.

ЗВ = 410854,33 / 0,9 = 456504,81 грн.

5.4 Розрахунок економічної ефективності науково-технічної розробки при її можливій комерціалізації потенційним інвестором

В ринкових умовах узагальнюючим позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження

результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку.

Результати дослідження проведені за темою «Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі» передбачають комерціалізацію протягом 4-х років реалізації на ринку і можуть бути характеризовані як розробка чи суттєве вдосконалення програмного засобу (програмного забезпечення, програмного продукту) для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

ΔN – збільшення кількості споживачів продукту, у періоди часу, що аналізуються, від покращення його певних характеристик;

Показник	1-й рік	2-й рік	3-й рік	4-й рік
Збільшення кількості споживачів, осіб	2500	4200	4500	1200

N – кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки, приймемо 12000 осіб;

$Ц_б$ – вартість програмного продукту у році до впровадження результатів розробки, приймемо 7800,00 грн;

$\pm\Delta Ц_o$ – зміна вартості програмного продукту від впровадження результатів науково-технічної розробки, приймемо 1010,50 грн.

Можливе збільшення чистого прибутку у потенційного інвестора $\Delta П_i$ для кожного із 4-х років, протягом яких очікується отримання позитивних результатів від можливого впровадження та комерціалізації науково-технічної розробки, розраховуємо за формулою [27]:

$$\Delta П_i = (\pm\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \qquad (5.22)$$

де λ – коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. У 2022 році ставка податку на додану вартість складає 20%, а коефіцієнт $\lambda = 0,8333$;

ρ – коефіцієнт, який враховує рентабельність інноваційного продукту). Приймемо ρ =37%;

ϑ – ставка податку на прибуток, який має сплачувати потенційний інвестор, у 2022 році $\vartheta = 18\%$;

Збільшення чистого прибутку 1-го року:

$\Delta П_1 = (1010,50 \cdot 12000,00+8810,50 \cdot 2500) \cdot 0,83 \cdot 0,37 \cdot (1-0,18/100\%)=8600287,90$ грн.

Збільшення чистого прибутку 2-го року:

$\Delta П_2 = (1010,50 \cdot 12000,00+8810,50 \cdot 6700) \cdot 0,83 \cdot 0,37 \cdot (1-0,18/100\%)=17918734,37$ грн.

Збільшення чистого прибутку 3-го року:

$\Delta П_3 = (1010,50 \cdot 12000,00+8810,50 \cdot 11200) \cdot 0,83 \cdot 0,37 \cdot (1-0,18/100\%)=27902784,16$ грн.

Збільшення чистого прибутку 4-го року:

$\Delta П_4 = (1010,50 \cdot 12000,00+8810,50 \cdot 12400) \cdot 0,83 \cdot 0,37 \cdot (1-0,18/100\%)=30565197,44$ грн.

Приведена вартість збільшення всіх чистих прибутків *ПП*, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.23)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному з років, протягом яких виявляються результати впровадження науково-технічної розробки, грн;

T – період часу, протягом якого очікується отримання позитивних результатів від впровадження та комерціалізації науково-технічної розробки, роки; τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ=0,17; t – період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = 8600287,90/(1+0,17)^1+17918734,37/(1+0,17)^2+27902784,16/(1+0,17)^3+$$
$$+30565197,44/(1+0,17)^4=7350673,42+13089878,27+17421676,87+16311119,09=$$
$$54173347,65 \text{ грн.}$$

Величина початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки:

$$PV = k_{інв} \cdot ЗВ, \tag{5.24}$$

де $k_{інв}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, приймаємо $k_{інв}$=2;

$ЗВ$ – загальні витрати на проведення науково-технічної розробки та оформлення її результатів, приймаємо 456504,81 грн.

$PV = k_{інв} \cdot ЗВ = 2 \cdot 456504,81 = 913009,62$ грн.

Абсолютний економічний ефект $E_{абс}$ для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$E_{абс} = ПП - PV \tag{5.25}$$

де $ПП$ – приведена вартість зростання всіх чистих прибутків від можливого впровадження та комерціалізації науково-технічної розробки, 54173347,65 грн;

PV – теперішня вартість початкових інвестицій, 913009,62 грн.

$E_{абс} = ПП - PV$ = 54173347,65 - 913009,62 = 53260338,03 грн.

Внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1, \quad (5.26)$$

де $E_{абс}$ – абсолютний економічний ефект вкладених інвестицій, 53260338,03 грн; PV – теперішня вартість початкових інвестицій, 913009,62 грн; $T_ж$ – життєвий цикл науково-технічної розробки, тобто час від початку її розробки до закінчення отримування позитивних результатів від її впровадження, 4 роки.

$E_в = \sqrt[Tж]{1 + \frac{E_{абс}}{PV}} - 1$ = (1+53260338,03/913009,62)$^{1/4}$= 1,78.

Мінімальна внутрішня економічна дохідність вкладених інвестицій $\tau_{мін}$:

$$\tau_{мін} = d + f, \quad (5.27)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2022 році в Україні d =0,19; f – показник, що характеризує ризикованість вкладення інвестицій, приймемо 0,45. $\tau_{мін}$ = 0,19+0,45 = 0,64 < 1,78 свідчить про те, що внутрішня економічна дохідність інвестицій $E_в$, які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки вища мінімальної внутрішньої дохідності. Тобто інвестувати в науково-дослідну роботу за темою «Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі» доцільно.

Період окупності інвестицій $T_{ок}$ які можуть бути вкладені потенційним інвестором у впровадження та комерціалізацію науково-технічної розробки:

$$T_{ок} = \frac{1}{E_в}, \quad (5.28)$$

де $E_в$ – внутрішня економічна дохідність вкладених інвестицій. $T_{ок}$ = 1 / 1,78 = 0,56 р.

$T_{ок}$ < 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

5.5 Висновок до розділу 5

Згідно проведених досліджень рівень комерційного потенціалу розробки за темою «Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі» становить 43,0 бала, що, свідчить про комерційну важливість проведення даних досліджень (рівень комерційного потенціалу розробки високий). При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,06 рази. Також термін окупності становить 0,56 р., що менше 3-х років, що свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок. Отже можна зробити висновок про доцільність проведення науково-дослідної роботи за темою «Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі».

ВИСНОВКИ

При виконанні магістерської кваліфікаційної роботи розв’язано задачу розробки інформаційної технології та програмного забезпечення для розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі.

У першому розділі було розглянуто стан питання розпізнавання об’єктів у зображеннях. Проведено огляд відомих методів для роботи з інтелектуальними обчисленнями у сфері комп’ютерного зору. В ході аналізу предметної області розглянуто основні методи розпізнавання зображень та як найбільш перспективний, було обрано нейромережевий метод. Також було здійснено аналіз відомих програмних засобів розпізнавання об’єктів на зображеннях та як аналог до розроблюваної програми було обрано програму Inception.

У другому розділі магістерської кваліфікаційної роботи було обґрунтовано доцільність використання для даної задачі згорткової нейронної мережі, а саме – попередньо натренованої нейронної мережі YOLOv2. Була проаналізована структура та порядок функціонування згорткової нейромережі YOLOv2. Розроблено структуру інформаційної технології розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі Також було розроблено структуру програмного забезпечення та алгоритм роботи програми розпізнавання об’єктів на зображеннях.

У третьому розділі було обґрунтовано вибір середовища розробки, мови програмування та технологій, які будуть використовуватися при розробці програми та наведено їх основні переваги. В результаті порівняльного аналізу було обрано середовище розробки Visual Studio, мову програмування C# та технології ML.NET для розробки модулів обробки зображень і .NET Core для розробки графічного інтерфейсу користувача. Було проведено реалізацію

програмних модулів, а саме модуля пошуку та розпізнавання об’єктів у зображеннях та модуля аналізу та візуалізації вихідних даних після обробки.

У четвертому розділі було розглянуто різні методи тестування, та визначено, що для даної нейромережевої системи найоптимальнішим є тестування методом «чорної скриньки». Під час тестування було розглянуто декілька можливих випадків розпізнавання об’єктів у зображеннях. В результаті тестування програми було доведено її повну працездатність та відповідність поставленому завданню. Серед переваг програми можна відзначити її більшу достовірність у розпізнаванні великих об’єктів – тих, які знаходяться на передньому плані і займають певну частину зображення. До недоліків нейромережевої системи можна віднести неможливість розпізнавання дуже малих об’єктів заднього плану зображення. Розроблена програма має вищу достовірність розпізнавання об’єктів на зображеннях (95,4%), ніж аналогічна програма (87,9%), а значить достовірність розпізнавання об’єктів на зображеннях покращена на 7,5%, тобто мета роботи досягнута.

У п`ятому розділі було визначено рівень комерційного потенціалу розробки, який становить 43,0 бала, що є високим і свідчить про комерційну важливість проведення даних досліджень. При оцінюванні рівня конкурентоспроможності, згідно узагальненого коефіцієнту конкурентоспроможності розробки, науково-технічна розробка переважає існуючі аналоги приблизно в 2,06 рази. Термін окупності становить 0,56 р., що менше 3-х років, а це свідчить про комерційну привабливість науково-технічної розробки і може спонукати потенційного інвестора профінансувати впровадження даної розробки та виведення її на ринок.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. С. В. Мальцев, Р. О. Давидюк, О. К. Колесницький «Інформаційна технологія розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі», в Матеріали конференції «Молодь в науці: дослідження, проблеми, перспективи (МН-2023)», Вінниця, 2023, [Електронний ресурс]. Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/viewFile/16829/14042.

2. Object Detection [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Object_detection.

3. Tang, Y.; Zhang, C.; Gu, R. Vehicle detection and recognition for intelligent traffic surveillance system. Multimed. Tools Appl. 2017, 76, 5817–5832.

4. Azure Congnitive Services [Електронний ресурс] – Режим доступу до ресурсу: https://azure.microsoft.com/en-us/services/cognitive-services/.

5. Amazon Recognition [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/rekognition/.

6. Machine vision [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Machine_vision.

7. Фисенко В.Т. / Компьютерная обработка и распознавание изображений / В.Т. Фисенко, Т.Ю. Фисенко – учеб. пособие – СПб: СПбГУ ИТМО, 2008. – 192 с.

8. Dougherty G. / Digital Image Processing for Medical Applications / G. Dougherty – Cambridge University Press. – ISBN 978-0-5218-6085-7.

9. Pan, C.; Sun, M.; Yan, Z. The Study on Vehicle Detection Based on DPM in Traffic Scenes. In Proceedings of the International Conference on Frontier Computing, Tokyo, Japan, 13–15 July 2016; pp. 19–27.

10. Куссуль Н. М. Інтелектуальні обчислення: навчальний посібник (із грифом МОН України) / Н. М. Куссуль, А. Ю. Шелестов, А. Н. Лавренюк. - К.: «Наукова думка», 2006. — 186 с. ISBN 966-00-0592-X.

11. Руденко О.В. Штучні нейронні мережі: Навчальний посібник / О.В.Руденко, Є.В.Бодянський. - Харків : ТОВ «Компанія СМІТ», 2006. — 404 с. - ISBN 966-8630-73-X.

12. Krizhevsky A. ImageNet Classification with Deep Convolutional Neural Networks [Електронний ресурс]. URL: https://papers.nips.cc/paper/4824imagenet-classification-with-deep-convolutional-neural-networks.pdf.

13. Inceptionv3 [Електронний ресурс]. URL: https://se.mathworks.com/help/deeplearning/ref/inceptionv3.html

14. Глибокі нейронні мережі для вирішення завдань розпізнавання і класифікації зображення [Електронний ресурс]. – Режим доступу: http://itcm.comp-sc.if.ua/2017/Sineglazov.pdf

15. Recurrent neural network [Електронний ресурс] – Режим доступу до ресурсу: https://en.wikipedia.org/wiki/Recurrent_neural_network.

16. What is Object Detection [Електронний ресурс] – Режим доступу до ресурсу: https://www.mathworks.com/discovery/object-detection.html/.

17. Redmon, J.; Farhadi, A. YOLO9000: Better, Faster, Stronger. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 6517–6525.

18. Бубнов И. Лучшие IDE для разработки на C# Один очевидный вариант и несколько других. [Електронний ресурс] / Илья Бубнов // geekbrains. – 2018. – Режим доступу до ресурсу: https://geekbrains.ru/posts/c_sharp_ides/.

19. Eclipse desktop & web IDEs [Електронний ресурс] – Режим доступу: https://www.eclipse.org/ide/.

20. QT Creator IDE [Електронний ресурс] – Режим доступу: https://www.qt.io/product/.

21. Visual Studio Лучшие в своем классе средства для разработчиков [Електронний ресурс] // Visual Studio – Режим доступу до ресурсу: https://visualstudio.microsoft.com/en/.

22. Joseph Albahari C# 6.0 in a Nutshell: The Definitive Reference/ Joseph Albahari, Ben Albahari – O'Reilly Media, Inc., 2015. – 1136 ст.

23. ML.NET – Machine Learning made for .NET [Електронний ресурс] – Режим доступу: https://dotnet.microsoft.com/apps/machinelearning-ai/ml-dotnet/.

24. Степанченко И.В. Методы тестирования программного обеспечения: учебное пособие / И.В. Степанченко. – Волгоград : ВолгГТУ, 2006. – 74с.

25. Software testing – Wikipedia [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Software_testing.

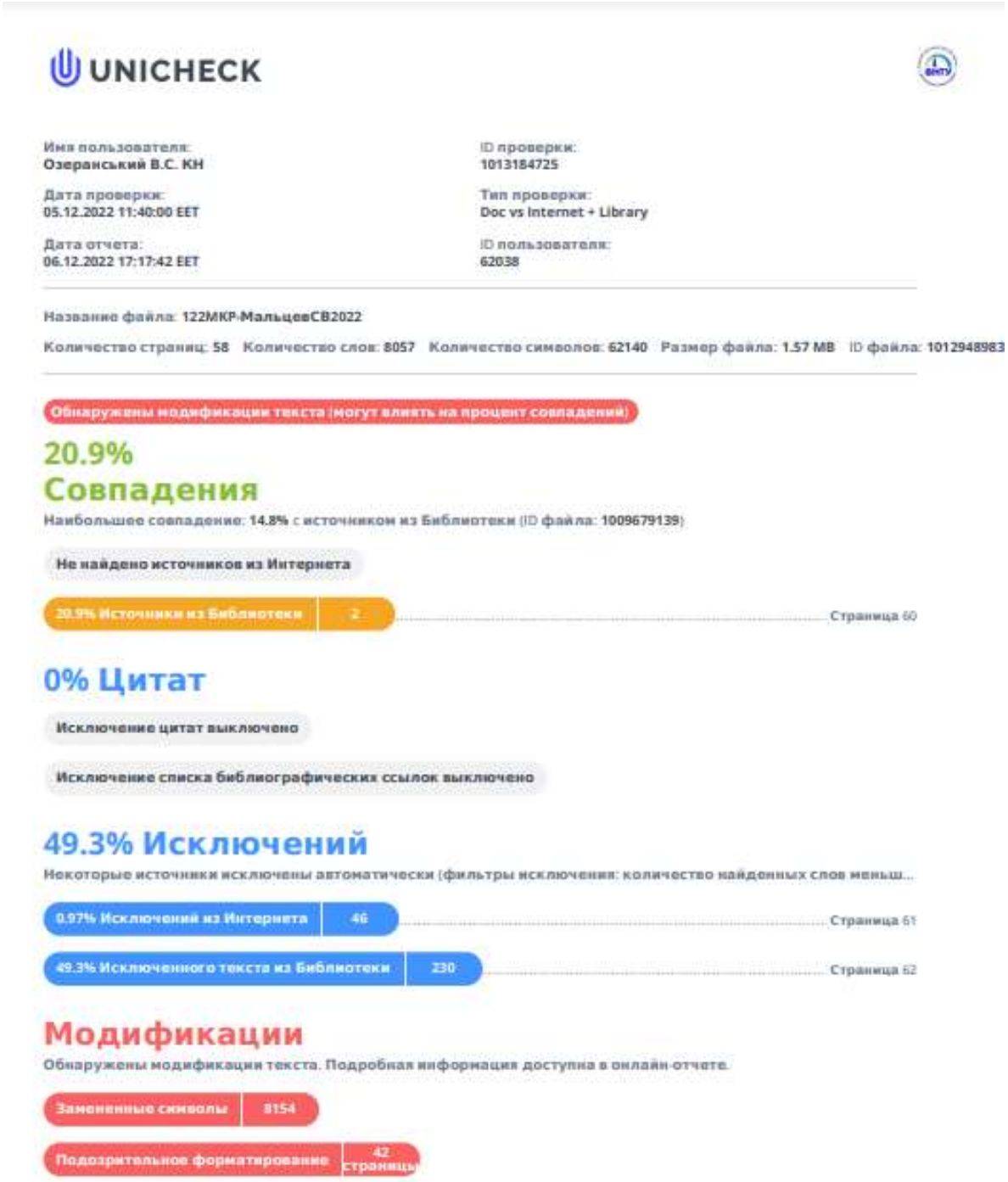
26. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

27. Кавецький В. В. Економічне обґрунтування інноваційних рішень: практикум / В. В. Кавецький, В. О. Козловський, І. В. Причепа – Вінниця : ВНТУ, 2016. – 113 с.

ДОДАТКИ

Додаток А (обов'язковий)

Результат перевірки на плагіат в онлайн-системі UNICHECK



UNICHECK

Имя пользователя:
Озеранський В.С. КН

ID проверки:
1013184725

Дата проверки:
05.12.2022 11:40:00 EET

Тип проверки:
Doc vs Internet + Library

Дата отчета:
06.12.2022 17:17:42 EET

ID пользователя:
62038

Название файла: 122МКР-МальцевСВ2022

Количество страниц: 58 Количество слов: 8057 Количество символов: 62140 Размер файла: 1.57 MB ID файла: 1012948983

Обнаружены модификации текста (могут влиять на процент совпадений)

20.9%
Совпадения

Наибольшее совпадение: 14.8% с источником из Библиотеки (ID файла: 1009679139)

Не найдено источников из Интернета

20.9% Источники из Библиотеки | 2 ..... Страница 60

0% Цитат

Исключение цитат выключено

Исключение списка библиографических ссылок выключено

49.3% Исключений

Некоторые источники исключены автоматически (фильтры исключения: количество найденных слов меньш...

0.97% Исключений из Интернета | 46 ..... Страница 61

49.3% Исключенного текста из Библиотеки | 230 ..... Страница 62

Модификации

Обнаружены модификации текста. Подробная информация доступна в онлайн-отчете.

Замененные символы | 8154

Подозрительное форматирование | 42 страницы

Додаток Б (обов'язковий)

Лістинг програми

YoloV2ObjectDetectionService.cs

```
namespace ObjectDetector.Services.V2.Services
{
    using System;
    using System.Collections.Generic;
    using System.Drawing;
    using System.Drawing.Drawing2D;
    using System.IO;
    using ObjectDetector.Infrastructure.DataStructures;
    using ObjectDetector.Infrastructure.YoloParser;
    using ObjectDetector.Services.Interfaces;
    using ObjectDetector.Services.Models;
    using ObjectDetector.Services.V2.DataModels;

    public class YoloV2ObjectDetectionService :
ITinyYoloV2ObjectDetectionService, IYoloV2ObjectDetectionService
    {
        private IList<YoloBoundingBox> _filteredBoundingBoxes;
        private readonly IYoloV2Model _model;
        private readonly YoloV2ModelScorer _modelScorer;
        private readonly YoloV2OutputParser _outputParser;

        public YoloV2ObjectDetectionService(IYoloV2Model model)
        {
            IYoloV2Model model1;
            this._model = model;
            this._modelScorer = new YoloV2ModelScorer(this._model);
            this._outputParser = new YoloV2OutputParser(this._model);
        }

        public void DetectObjectsUsingModel(ImageNetData imageInputData, float
probability = .3F, int limit = 5, float intersection = .5F)
        {
            var probabilities = _modelScorer.Score(imageInputData);
            var boundingBoxes = _outputParser.ParseOutputs(probabilities,
probability);
            this._filteredBoundingBoxes =
_outputParser.FilterBoundingBoxes(boundingBoxes, limit, intersection);
        }


        public void DrawBoundingBox(string inputImageLocation, string
outputImageLocation, string imageName)
        {
            Image image = Image.FromFile(Path.Combine(inputImageLocation,
imageName));
            var originalImageHeight = image.Height;
            var originalImageWidth = image.Width;

            foreach (var box in this._filteredBoundingBoxes)
            {
                var x = (uint)Math.Max(box.Dimensions.X, 0);
```

```
                var y = (uint)Math.Max(box.Dimensions.Y, 0);
                var width = (uint)Math.Min(originalImageWidth - x,
box.Dimensions.Width);
                var height = (uint)Math.Min(originalImageHeight - y,
box.Dimensions.Height);
                // Resize To Image
                x = (uint)originalImageWidth * x / ImageNetSettings.imageWidth;
                y = (uint)originalImageHeight * y / ImageNetSettings.imageHeight;
                width = (uint)originalImageWidth * width /
ImageNetSettings.imageWidth;
                height = (uint)originalImageHeight * height /
ImageNetSettings.imageHeight;

                using (Graphics thumbnailGraphic = Graphics.FromImage(image))
                {
                    thumbnailGraphic.CompositingQuality =
CompositingQuality.HighQuality;
                    thumbnailGraphic.SmoothingMode = SmoothingMode.HighQuality;
                    thumbnailGraphic.InterpolationMode =
InterpolationMode.HighQualityBicubic;
                    // Define Text Options
                    Font drawFont = new Font("Arial", 12, FontStyle.Bold);
                    SizeF size = thumbnailGraphic.MeasureString(box.Description,
drawFont);
                    SolidBrush fontBrush = new SolidBrush(Color.Black);
                    Point atPoint = new Point((int)x, (int)y - (int)size.Height +
20);
                    // Define BoundingBox options
                    Pen pen = new Pen(box.BoxColor, 3.2f);
                    SolidBrush colorBrush = new SolidBrush(box.BoxColor);
                    // Draw bounding box on image
                    thumbnailGraphic.DrawRectangle(pen, x, y, width, height);
                    // Draw text on image
                    thumbnailGraphic.FillRectangle(colorBrush, (int)x, (int)(y -
size.Height + 20), (int)size.Width, (int)size.Height);
                    thumbnailGraphic.DrawString(box.Description, drawFont,
fontBrush, atPoint);
                }
            }

            if (!Directory.Exists(outputImageLocation))
            {
                Directory.CreateDirectory(outputImageLocation);
            }
            image.Save(Path.Combine(outputImageLocation, imageName));
        }
    }
}
```

YoloV2ModelScorer.cs

```
namespace ObjectDetector.Services.V2.Services
{
    using System;
    using System.Collections.Generic;
    using Microsoft.ML;
    using Microsoft.ML.Transforms.Image;
```

```
    using ObjectDetector.Infrastructure.DataStructures;
    using ObjectDetector.Services.Interfaces;
    using ObjectDetector.Services.Models;
    using ObjectDetector.Services.V2.DataModels;

    public class YoloV2ModelScorer : IModelScorer
    {
        private readonly MLContext mlContext;
        private readonly PredictionEngine<ImageNetData, YoloV2Prediction>
predictionEngine;

        public YoloV2ModelScorer(IYoloV2Model onnxModel)
        {
            this.mlContext = new MLContext();
            this.predictionEngine = LoadModel(onnxModel);
        }

        public float[] Score(ImageNetData image) =>
this.predictionEngine.Predict(image).PredictedLabels;

        private PredictionEngine<ImageNetData, YoloV2Prediction>
LoadModel(IYoloV2Model onnxModel)
        {
            Console.WriteLine("Завантаження моделі.");
            Console.WriteLine($"Розташування моделі: {onnxModel.ModelPath}");
            Console.WriteLine($"Default parameters: image
size=({ImageNetSettings.imageWidth},{ImageNetSettings.imageHeight})");

            var data = mlContext.Data.LoadFromEnumerable(new
List<ImageNetData>());
            var pipeline = mlContext.Transforms.LoadImages(
                    outputColumnName: onnxModel.ModelInput,
                    imageFolder: "",
                    inputColumnName: nameof(ImageNetData.ImagePath))
                .Append(mlContext.Transforms.ResizeImages(
                    outputColumnName: onnxModel.ModelInput,
                    imageWidth: ImageNetSettings.imageWidth,
                    imageHeight: ImageNetSettings.imageHeight,
                    inputColumnName: onnxModel.ModelInput,
                    resizing: ImageResizingEstimator.ResizingKind.Fill))
                .Append(mlContext.Transforms.ExtractPixels(
                    outputColumnName: onnxModel.ModelInput,
                    inputColumnName: onnxModel.ModelInput))
                .Append(mlContext.Transforms.ApplyOnnxModel(
                    outputColumnName: onnxModel.ModelOutput,
                    inputColumnName: onnxModel.ModelInput,
                    modelFile: onnxModel.ModelPath));

            var model = pipeline.Fit(data);
            var predictionEng =
mlContext.Model.CreatePredictionEngine<ImageNetData, YoloV2Prediction>(model);

            return predictionEng;
        }
    }
}
```

YoloV2OutputParser.cs

```
namespace ObjectDetector.Infrastructure.YoloParser
{
    using System;
    using System.Collections.Generic;
    using System.Drawing;
    using System.Linq;
    using ObjectDetector.Core.Interfaces;
    using ObjectDetector.Infrastructure.DataStructures;

    public class YoloV2OutputParser
    {
        private class BoundingBoxPrediction : BoundingBoxDimensions
        {
            public float Confidence { get; set; }
        }

    public const int ROW_COUNT = 13;
    public const int COL_COUNT = 13;

    public const int BOX_INFO_FEATURE_COUNT = 5;

    private readonly string[] classLabels;

    private readonly (float x, float y)[] boxAnchors;


    public YoloV2OutputParser(IOnnxModel<(float, float)> onnxModel)
    {
        classLabels = onnxModel.Labels;
        boxAnchors = onnxModel.Anchors;
    }

        private float Sigmoid(float value)
        {
            var k = MathF.Exp(value);
            return k / (1.0f + k);
        }

        private float[] Softmax(float[] classProbabilities)
        {
            var max = classProbabilities.Max();
            var exp = classProbabilities.Select(v => MathF.Exp(v - max));
            var sum = exp.Sum();
            return exp.Select(v => v / sum).ToArray();
        }

ML.NET flattens

        private int GetOffset(int row, int column, int channel)
        {
            const int channelStride = ROW_COUNT * COL_COUNT;
            return (channel * channelStride) + (column * COL_COUNT) + row;
        }

        private BoundingBoxPrediction ExtractBoundingBoxPrediction(float[]
modelOutput, int row, int column, int channel)
```

```
        {
            return new BoundingBoxPrediction
            {
                X = modelOutput[GetOffset(row, column, channel++)],
                Y = modelOutput[GetOffset(row, column, channel++)],
                Width = modelOutput[GetOffset(row, column, channel++)],
                Height = modelOutput[GetOffset(row, column, channel++)],
                Confidence = Sigmoid(modelOutput[GetOffset(row, column,
channel++)])
            };
        }

        private BoundingBoxDimensions MapBoundingBoxToCell(int row, int column,
int box, BoundingBoxPrediction boxDimensions)
        {
            const float CELL_WIDTH = ImageNetSettings.imageWidth / COL_COUNT;
            const float CELL_HEIGHT = ImageNetSettings.imageHeight / ROW_COUNT;

            var mappedBox = new BoundingBoxDimensions
            {
                X = (row + Sigmoid(boxDimensions.X)) * CELL_WIDTH,
                Y = (column + Sigmoid(boxDimensions.Y)) * CELL_HEIGHT,
                Width = MathF.Exp(boxDimensions.Width) * CELL_WIDTH *
boxAnchors[box].x,
                Height = MathF.Exp(boxDimensions.Height) * CELL_HEIGHT *
boxAnchors[box].y,
            };

            // The x,y coordinates from the (mapped) bounding box prediction
represent the center
            // of the bounding box. We adjust them here to represent the top left
corner.
            mappedBox.X -= mappedBox.Width / 2;
            mappedBox.Y -= mappedBox.Height / 2;

            return mappedBox;
        }

        // IoU (Intersection over union) measures the overlap between 2
boundaries.
        private float IntersectionOverUnion(RectangleF boundingBoxA, RectangleF
boundingBoxB)
        {
            var areaA = boundingBoxA.Width * boundingBoxA.Height;
            var areaB = boundingBoxB.Width * boundingBoxB.Height;
            if (areaA <= 0 || areaB <= 0)
            {
                return 0;
            }
            var minX = MathF.Max(boundingBoxA.Left, boundingBoxB.Left);
            var minY = MathF.Max(boundingBoxA.Top, boundingBoxB.Top);
            var maxX = MathF.Min(boundingBoxA.Right, boundingBoxB.Right);
            var maxY = MathF.Min(boundingBoxA.Bottom, boundingBoxB.Bottom);
            var intersectionArea = MathF.Max(maxY - minY, 0) * MathF.Max(maxX -
minX, 0);
            return intersectionArea / (areaA + areaB - intersectionArea);
        }
```

```
        public IList<YoloBoundingBox> ParseOutputs(float[] onnxModelOutputs,
float probabilityThreshold = .3f)
        {
            var boxes = new List<YoloBoundingBox>();
            for (int row = 0; row < ROW_COUNT; row++)
            {
                for (int column = 0; column < COL_COUNT; column++)
                {
                    for (int box = 0; box < boxAnchors.Length; box++)
                    {
                        var channel = box * (classLabels.Length +
BOX_INFO_FEATURE_COUNT);
                        var boundingBoxPrediction =
ExtractBoundingBoxPrediction(onnxModelOutputs, row, column, channel);
                        var mappedBoundingBox = MapBoundingBoxToCell(row, column,
box, boundingBoxPrediction);

                        if (boundingBoxPrediction.Confidence <
probabilityThreshold)
                            continue;
                        var classProbabilities =
ExtractClassProbabilities(onnxModelOutputs, row, column, channel,
boundingBoxPrediction.Confidence);
                        var (topProbability, topIndex) = classProbabilities
                            .Select((probability, index) => (Score: probability,
Index: index)).Max();
                        //var (topProbability, topIndex) = classProbabilities
                        //    .Select((probability, index) => (Score:
probability, Index: index))
                        //    .OrderByDescending(result => result.Score).First();
                        if (topProbability < probabilityThreshold)
                            continue;
                        boxes.Add(new YoloBoundingBox
                        {
                            Dimensions = mappedBoundingBox,
                            Confidence = topProbability,
                            Label = classLabels[topIndex],
                            BoxColor = YoloBoundingBox.GetColor(topIndex)
                        });
                    }
                }
            }
            return boxes;
        }

        public IList<YoloBoundingBox> FilterBoundingBoxes(IList<YoloBoundingBox>
boxes, int limit, float threshold)
        {
            var filteredBoxes = new bool[boxes.Count];
            var sortedBoxes = boxes.OrderByDescending(b =>
b.Confidence).ToArray();
            var results = new List<YoloBoundingBox>();
            for (int i = 0; i < boxes.Count; i++)
            {
                if (filteredBoxes[i])
                {
                    continue;
                }
```

```
                results.Add(sortedBoxes[i]);
                if (results.Count >= limit)
                {
                    break;
                }
                for (var j = i + 1; j < boxes.Count; j++)
                {
                    if (filteredBoxes[j])
                    {
                        continue;
                    }
                    if (IntersectionOverUnion(sortedBoxes[i].Rect,
sortedBoxes[j].Rect) > threshold)
                    {
                        filteredBoxes[j] = true;
                    }

                    if (filteredBoxes.Count(b => b) <= 0)
                    {
                        break;
                    }
                }
            }
            return results;
        }
    }
}
```

Додаток В (обов'язковий)

Додаток В (обов'язковий) 100

ІЛЮСТРАТИВНА ЧАСТИНА

ІНФОРМАЦІЙНА ТЕХНОЛОГІЯ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ НА ЗОБРАЖЕННЯХ З ВИКОРИСТАННЯМ ЗГОРТКОВОЇ НЕЙРОННОЇ МЕРЕЖІ

Виконав: студент 2-го курсу,
групи 2КН-21м
спеціальності 122 «Комп'ютерні науки»
(шифр і назва напряму підготовки, спеціальності)
Мальцев С.В.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Колесницький О.К.
(прізвище та ініціали)
«15» 12 2022 р.

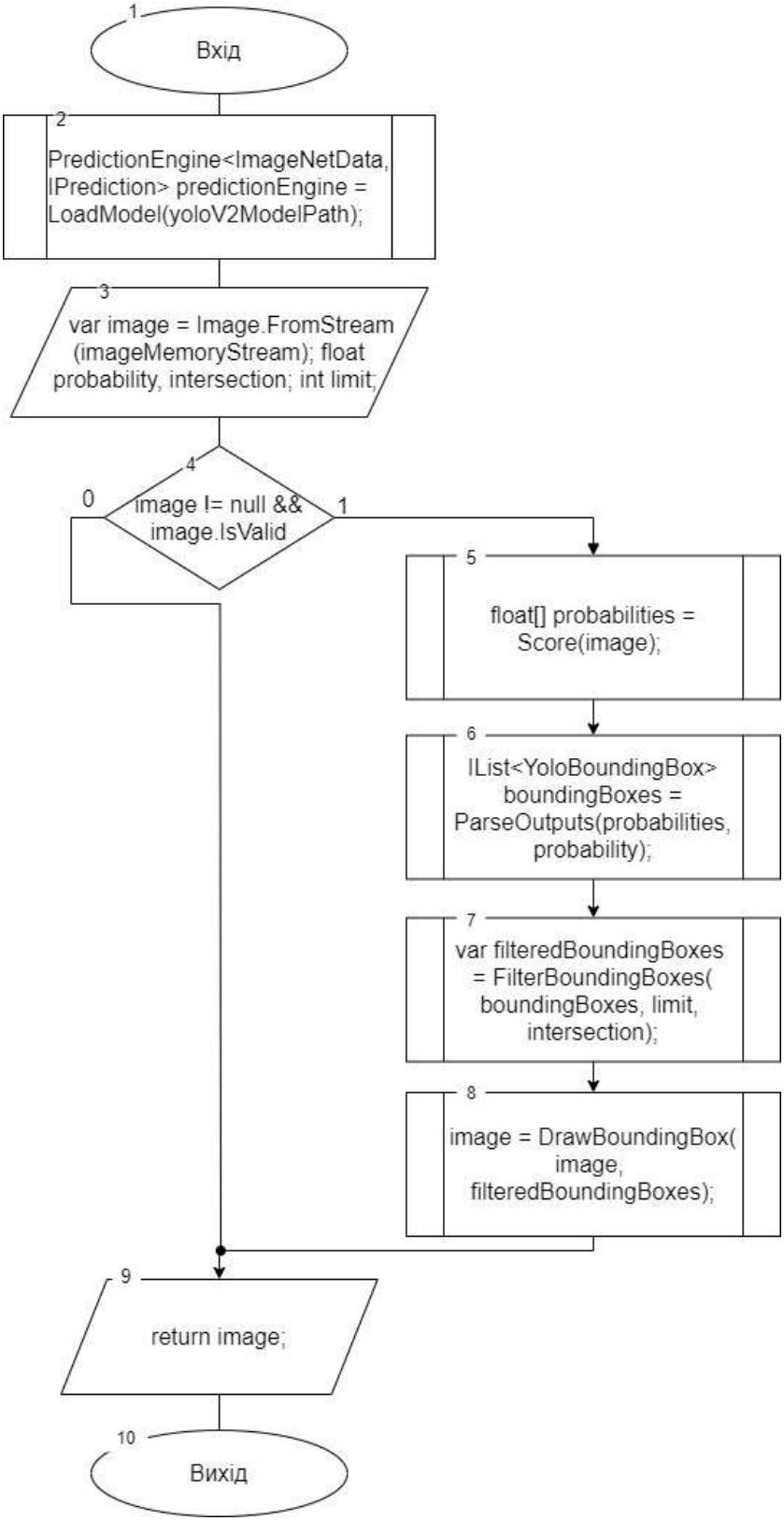


Рисунок В.1 – Граф-схема загального алгоритму роботи програми розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі

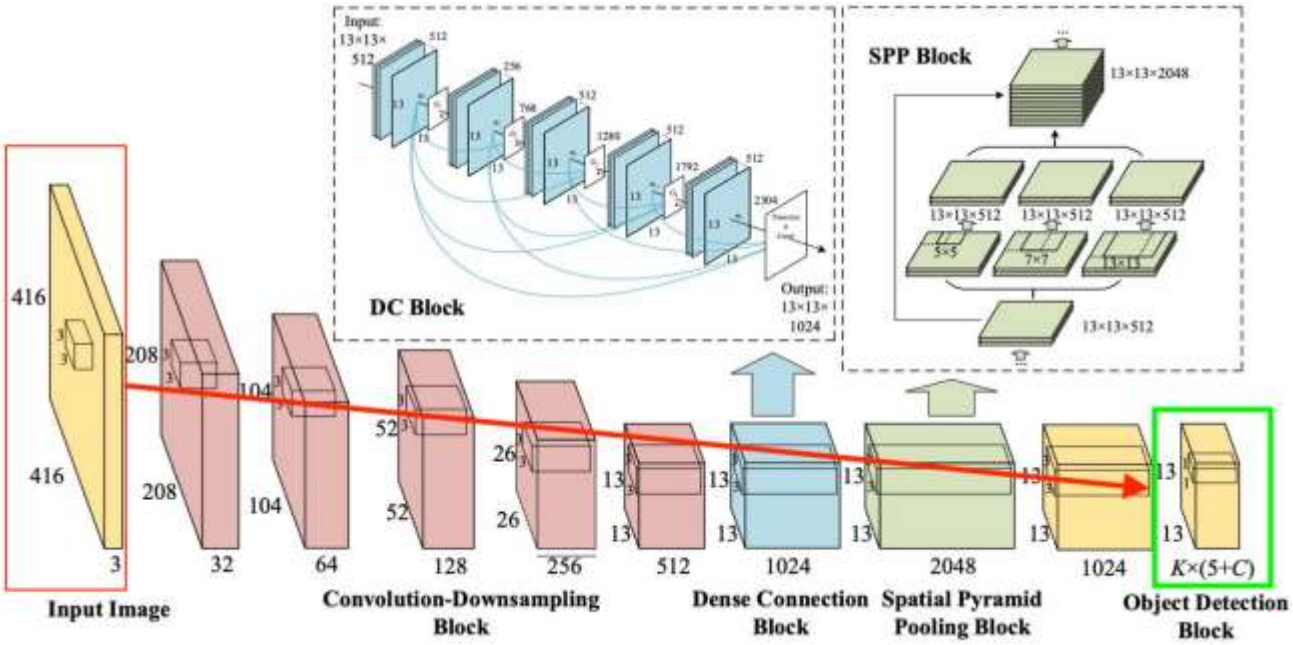


Рисунок В.2 – Структура нейронної мережі YOLOv2

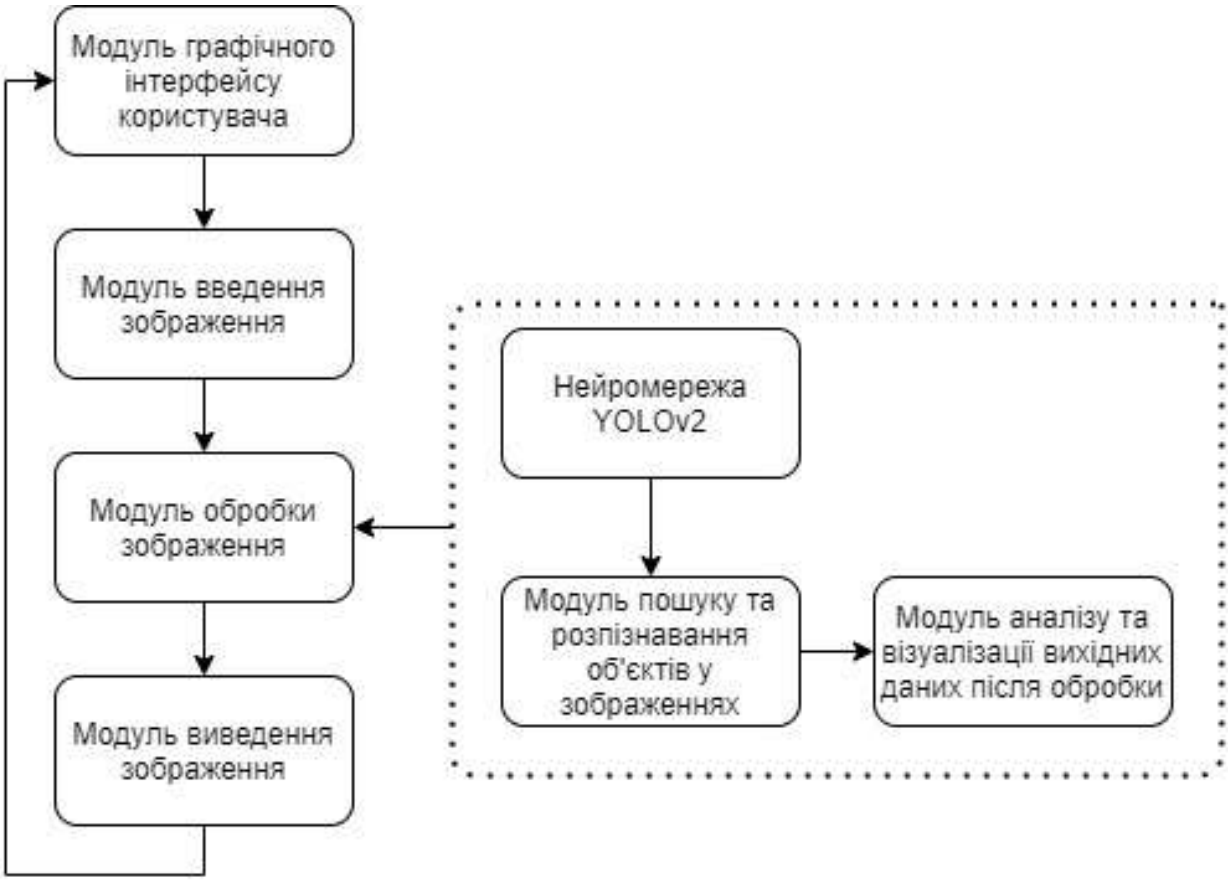


Рисунок В.3 – Структура програмного забезпечення розпізнавання об’єктів на зображеннях з використанням згорткової нейронної мережі

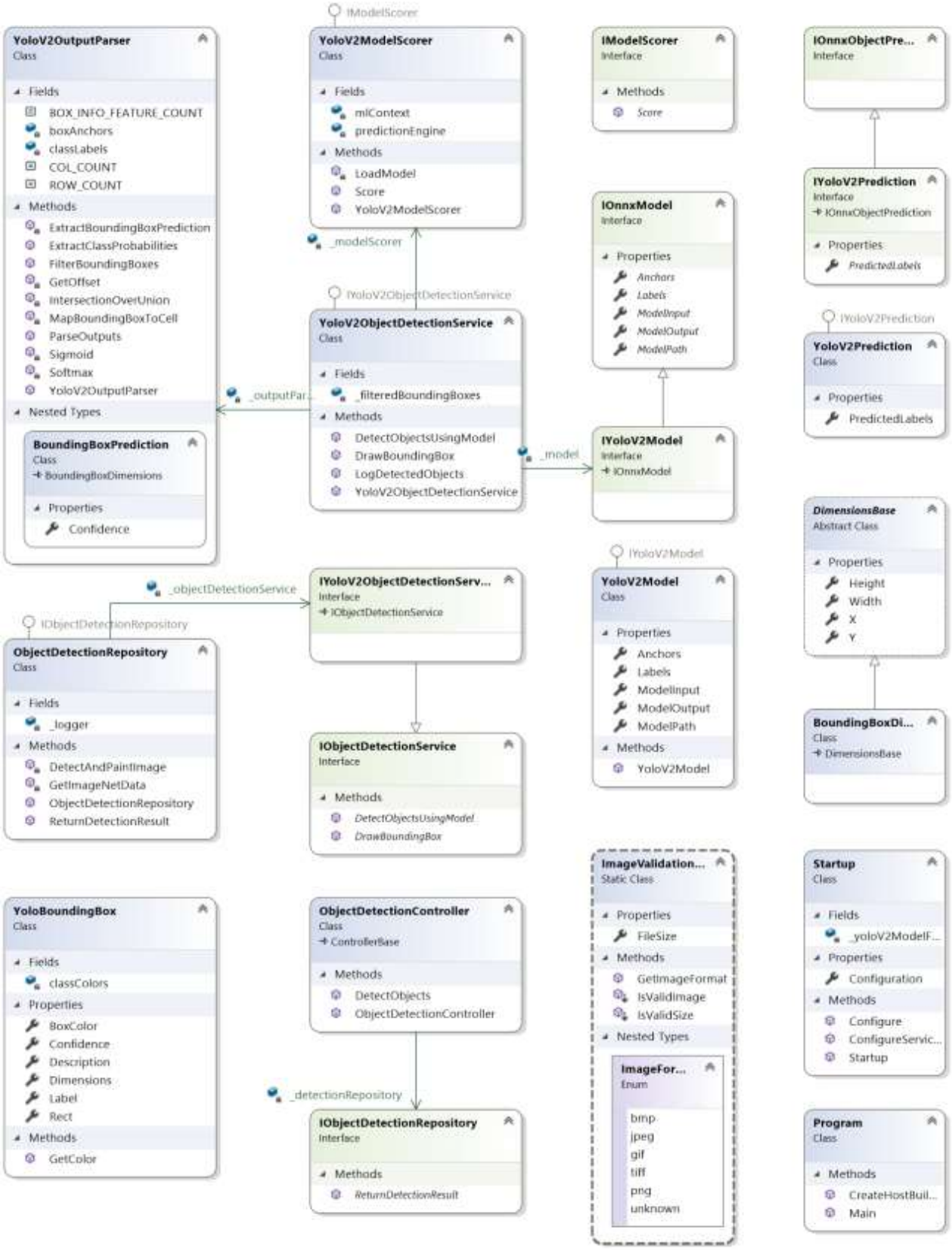
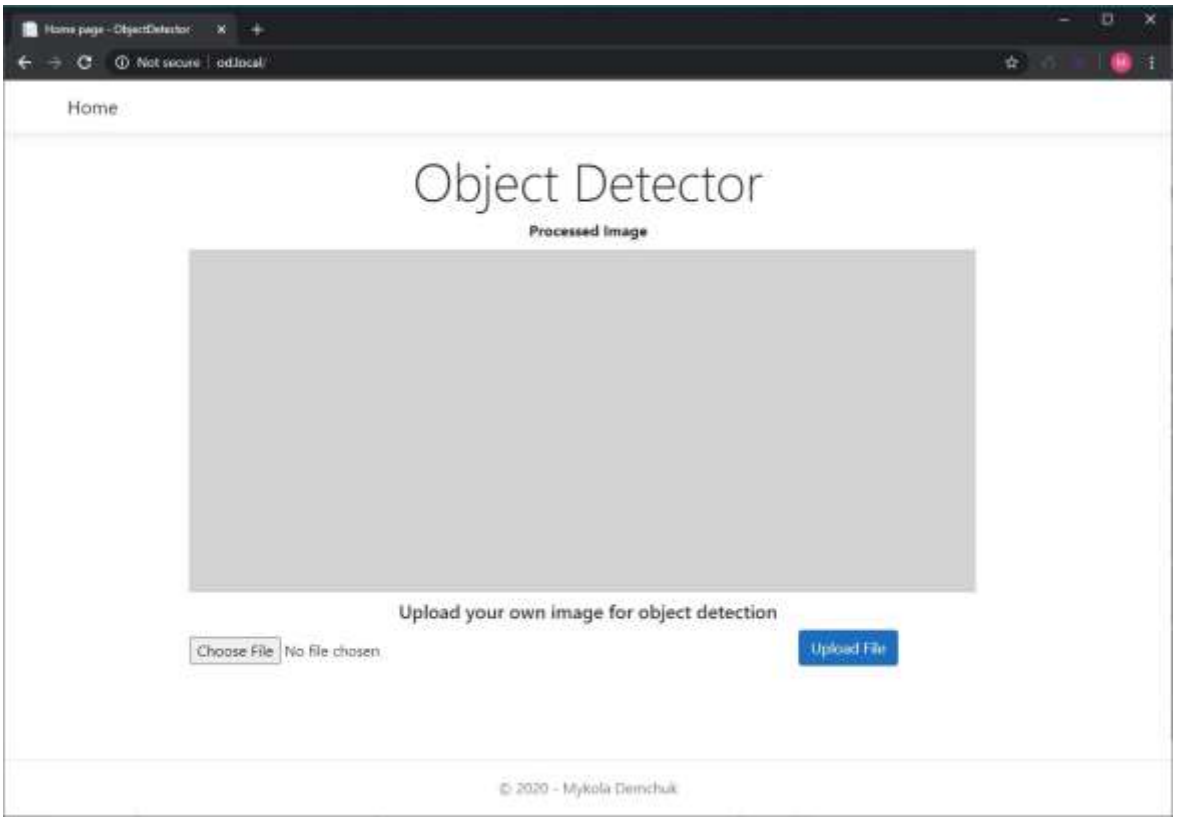


Рисунок В.4 – Діаграма класів програмного забезпечення



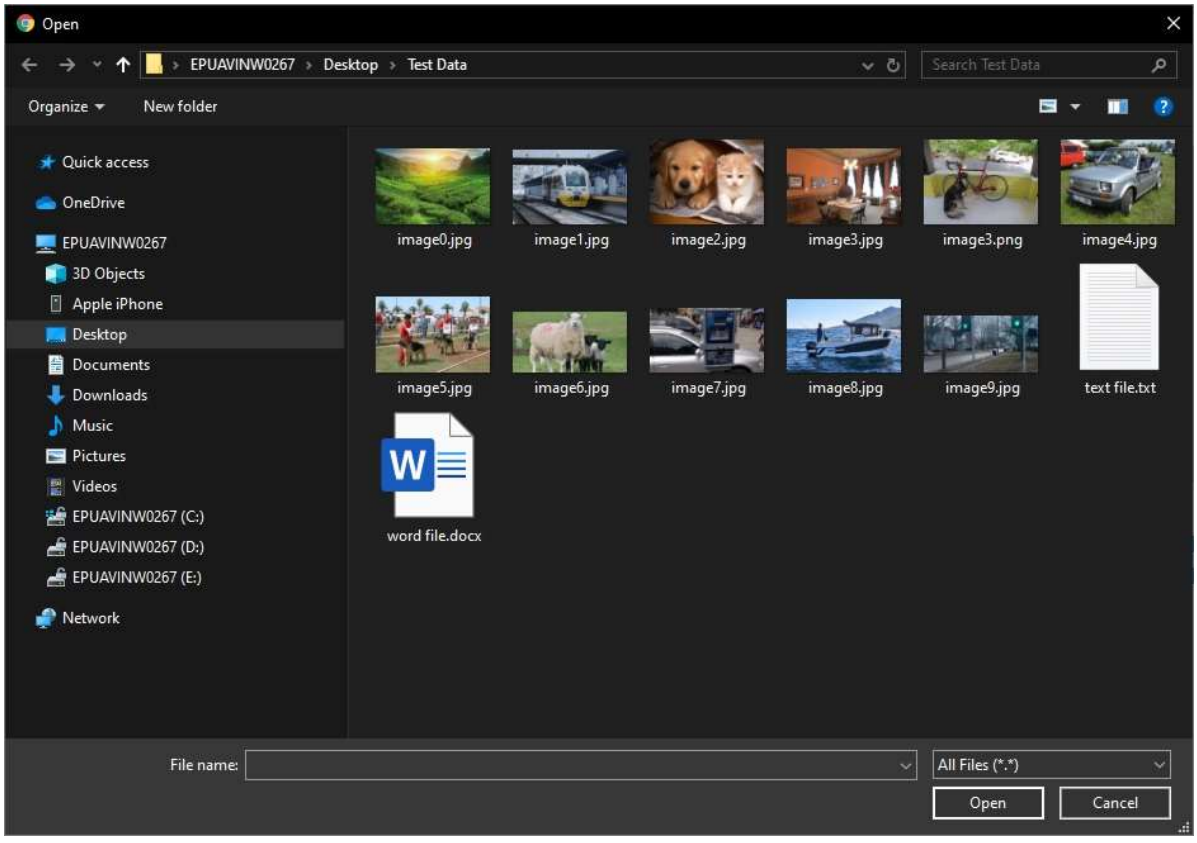


Рисунок В.5 – Стартові вікна програми розпізнавання об’єктів на зображеннях

Рисунок В.6 – Результати роботи програми розпізнавання об’єктів на зображеннях

Додаток Г (довідниковий)

Інструкція користувача

Так як додаток не вимагає інсталяції, то для початку роботи з ним достатньо запустити файл ObjectDetector.Web.exe. За замовчуванням відкривається вікно браузера, в якому можна побачити робочу область програми (рисунок Г.1).

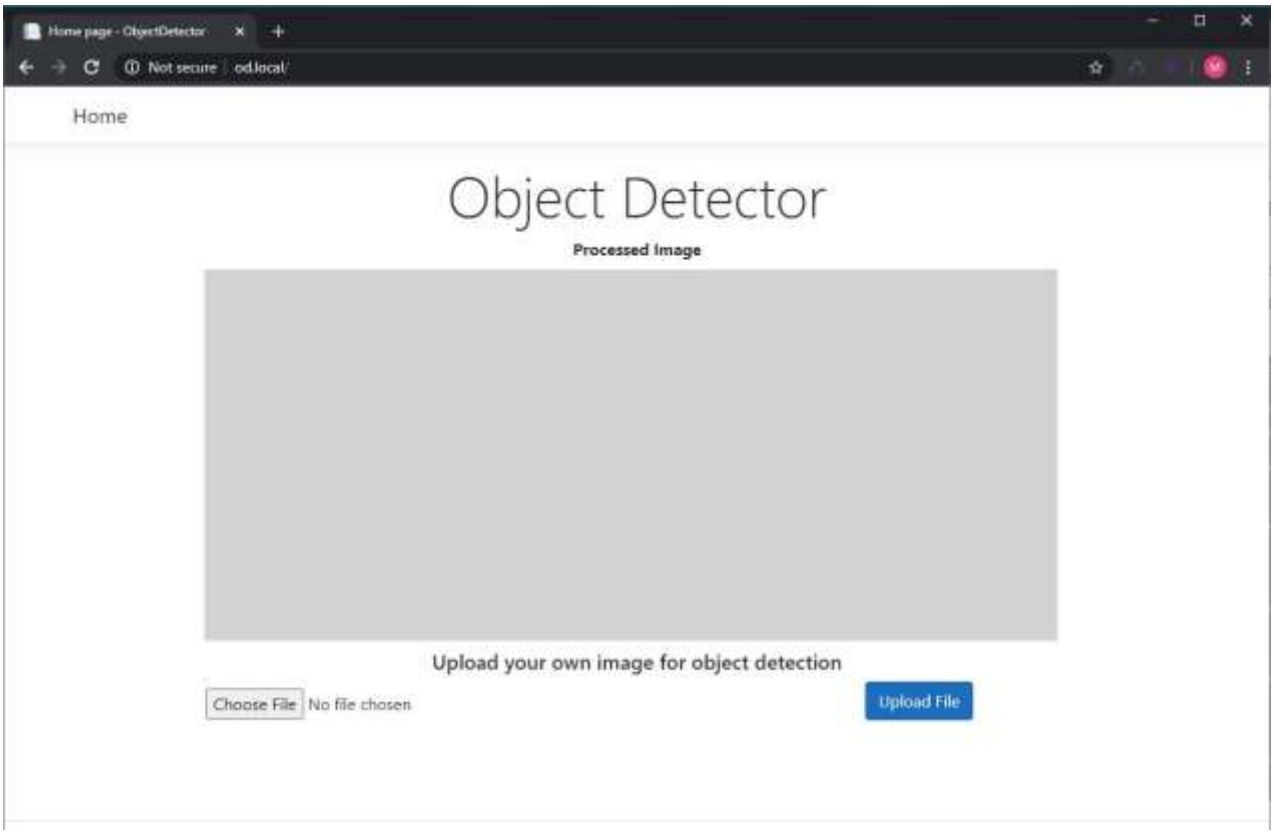


Рисунок Г.1 – Головне вікно програми

Для того, аби вибрати зображення для розпізнавання об’єктів у ньому, слід натиснути кнопку «Choose File», яка відкриває вікно вибору зображення з файлової системи користувача (рисунок Г.2). Допустимі формати файлів: .jpg, .jpeg, .png. Після обрання необхідного файлу зображення, необхідно натиснути кнопку «Open», після чого файл буде доданий у програму для обробки.

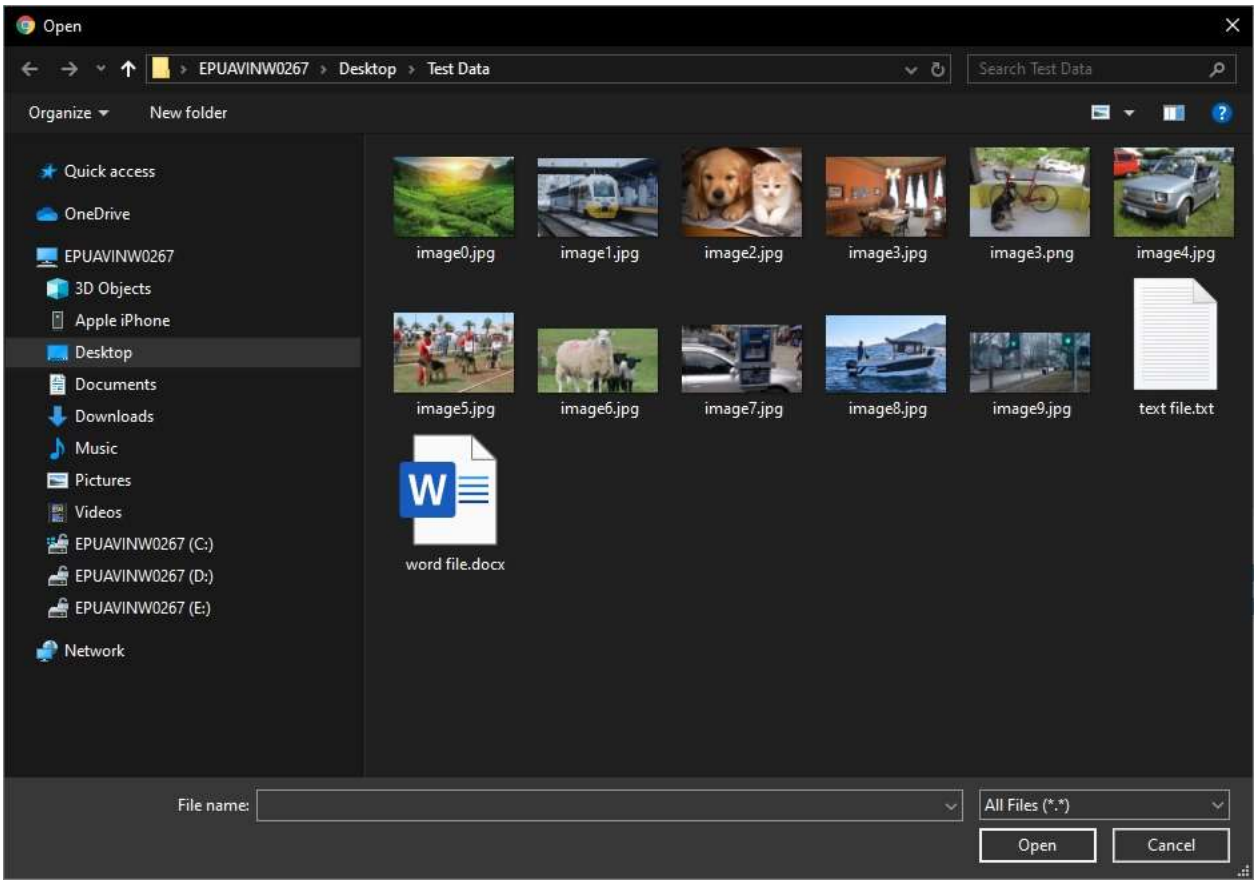


Рисунок Г.2 – Вікно вибору зображення

Після того, як зображення обрано, слід натиснути кнопку «Upload File». Додаток виконує усі необхідні обчислення та повертає результуюче зображення з виділеними класами об’єктів, їх назвою та достовірністю розпізнавання у робочу область програми. Також програма повертає кількість затраченого часу в мілісекундах на розпізнавання об’єктів у зображенні. Приклад роботи додатку наведено на рисунку Г.3.

Для того, аби повторно виконати розпізнавання об’єктів у зображенні, слід знову натиснути кнопку «Choose File» та повторити вищеописаний процес. Програма працює до повного закриття головного вікна користувачем.

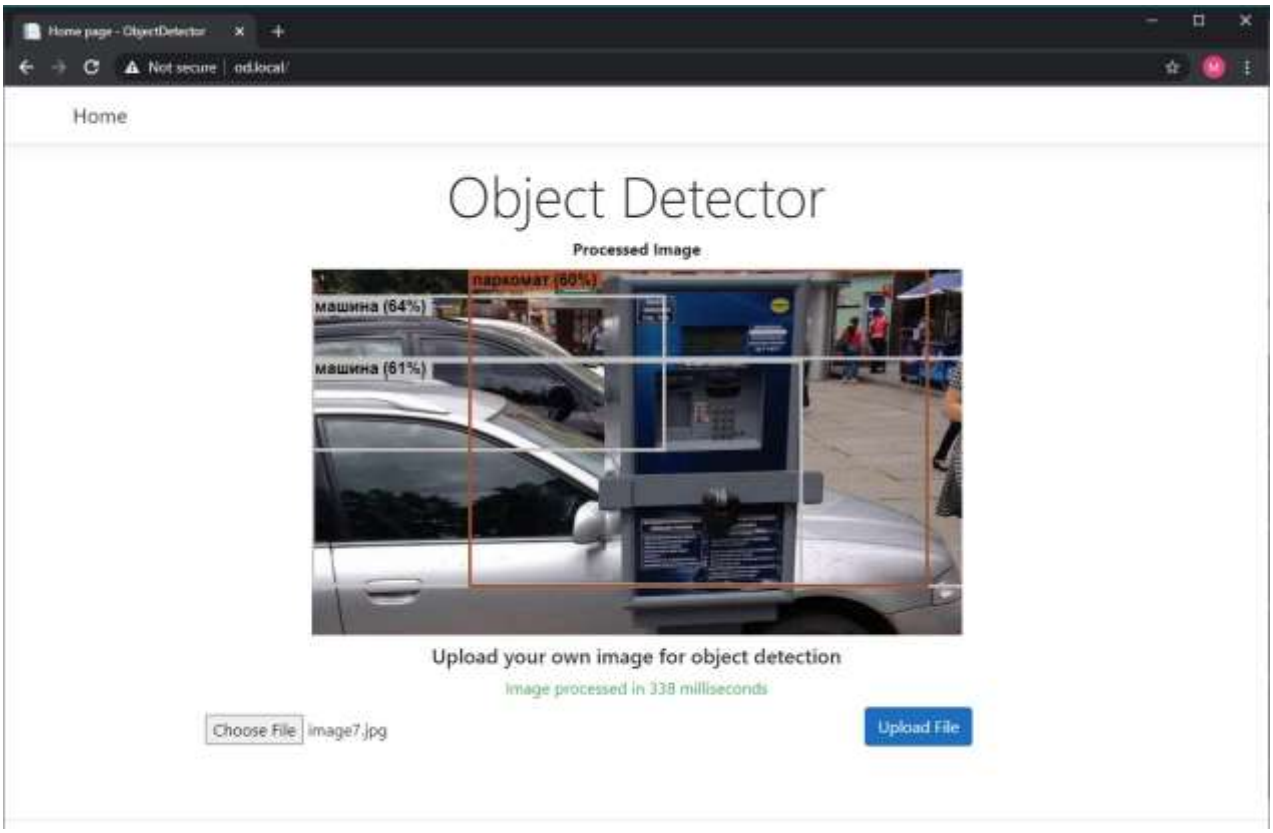


Рисунок Г.3 – Приклад роботи додатку