ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інтелектуальна система візуального розпізнавання на основі методів штучного інтелекту»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Штучний інтелект
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Станіслав ГОРБЕНКО
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконала:
здобувачка вищої освіти Станіслав ГОРБЕНКО
група ШІД-41

Керівник: Андрій ВІКАРЧУК
науковий ступінь, вчене звання

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту
Ступінь вищої освіти Бакалавр
Спеціальність Комп'ютерні науки
Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Горбенко Станіслав Віталійович
(прізвище, ім'я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Інтелектуальна система візуального розпізнавання на основі методів штучного інтелекту

керівник кваліфікаційної роботи Вікарчук А.І. асистент,
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56

2. Строк подання кваліфікаційної роботи «23» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: Наукові джерела з теми комп'ютерного зору та штучного інтелекту

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз існуючих методів розпізнавання у комп'ютерному зорі;
Дослідження сучасних моделей глибинного навчання (CNN, YOLO тощо);
Розробка структури та архітектури системи візуального розпізнавання;
Реалізація прототипу інтелектуальної системи;
Проведення тестування системи та аналіз результатів. Розробка рекомендаційної системи.
5. Перелік графічного матеріалу: *презентація*

1. Структурна схема системи;
2. Діаграма архітектури;
3. Фрагменти користувача інтерфейсу;
4. Графіки результатів тестування

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір літературних джерел та аналіз існуючих систем розпізнавання	12.03.2025	Виконано
2	Вивчення методів глибинного навчання та комп'ютерного зору	25.03.2025	Виконано
3	Постановка задачі та визначення архітектури системи	7.04.2025	Виконано
4	Розробка структури та проектування компонентів системи	18.04.2025	Виконано
5	Програмна реалізація системи	24.04.2025	Виконано
6	Проведення тестування та аналіз результатів	10.05.2025	Виконано
7	Оформлення пояснювальної записки	12.05.2025	Виконано
8	Підготовка до захисту	16.05.2025	Виконано
9	Захист диплому	16.06.2025	Виконано

Здобувачка вищої освіти _____ Станіслав ГОРБЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Андрій ВІКАРЧУК
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 46 стор., 6 табл., 11 малюнків, 22 джерел.

Мета роботи - розробити інтелектуальну систему візуального розпізнавання об'єктів на основі сучасних методів штучного інтелекту.

Об'єкт дослідження - процеси автоматичного аналізу та інтерпретації зображень за допомогою систем штучного інтелекту.

Предмет дослідження - методи комп'ютерного зору, моделі глибинного навчання та їхнє використання для візуального розпізнавання об'єктів.

Короткий зміст роботи: У роботі проведено аналіз існуючих методів візуального розпізнавання на основі нейронних мереж та алгоритмів машинного навчання. Розглянуто архітектури глибинних згорткових нейронних мереж (CNN) та сучасні моделі, такі як YOLO та ResNet. Спроектовано та розроблено прототип інтелектуальної системи для візуального розпізнавання об'єктів. Виконано тестування системи на стандартних наборах даних, проведено оцінку точності розпізнавання. Показано, що застосування методів штучного інтелекту дозволяє значно підвищити якість та швидкість обробки зображень у різних прикладних задачах.

КЛЮЧОВІ СЛОВА: ШТУЧНИЙ ІНТЕЛЕКТ, ВІЗУАЛЬНЕ РОЗПІЗНАВАННЯ, КОМП’ЮТЕРНИЙ ЗІР, НЕЙРОННІ МЕРЕЖІ, ГЛИБИННЕ НАВЧАННЯ, CNN, OPENCV, TENSORFLOW.

ABSTRACT

Text part of the qualification work for obtaining a bachelor's degree: 46 pages, 6 tables, 11 figures, 22 sources used.

The purpose of the work - is to develop an intelligent system for visual object recognition based on modern methods of artificial intelligence.

The object of research - is the processes of automatic analysis and interpretation of images using artificial intelligence systems.

The subject of research - is computer vision methods, deep learning models and their use for visual object recognition.

Summary of the work: The paper analyzes existing methods of visual recognition based on neural networks and machine learning algorithms. The architectures of deep convolutional neural networks (CNN) and modern models such as YOLO and ResNet are considered.

A prototype of an intelligent system for visual object recognition has been designed and developed. The system has been tested on standard data sets, and the recognition accuracy has been assessed. It has been shown that the use of artificial intelligence methods can significantly improve the quality and speed of image processing in various applied tasks.

KEYWORDS: ARTIFICIAL INTELLIGENCE, VISUAL RECOGNITION, COMPUTER VISION, NEURAL NETWORKS, DEEP LEARNING, CNN, OPENCV, TENSORFLOW.

ЗМІСТ

ВСТУП

У сучасному світі обсяг візуальної інформації, яку щоденно генерує людство, постійно зростає. Фото та відео стали не просто засобами збереження спогадів, а важливими джерелами даних для аналізу, контролю, безпеки, автоматизації та багатьох інших сфер. У зв'язку з цим дедалі більшу актуальність набувають системи, здатні автоматично “бачити” і розуміти візуальні дані - тобто інтелектуальні системи візуального розпізнавання.

До недавньої години розпізнавання об'єктів на зображеннях було прерогативою лише людей, адже воно потребувало складного аналізу, розуміння контексту та абстрактного мислення. Проте з розвитком штучного інтелекту, зокрема глибинного навчання, ситуація кардинально змінилася. Сучасні нейронні мережі вже зараз можуть точно виявляти та класифікувати об'єкти на фото, ідентифікувати обличчя, читати номери автомобілів, аналізувати медичні знімки тощо.

Ця тема є особливо актуальною, оскільки візуальне розпізнування використовується майже всюди - від безпілотних автомобілів до систем контролю доступу, від аналізу супутникових знімків до мобільних додатків. Тому розробка такої системи, навіть у спрощеному вигляді, дозволяє краще зрозуміти принципи роботи сучасних технологій та побачити їхню практичну цінність.

Метою цієї кваліфікаційної роботи є розробка інтелектуальної системи візуального розпізнавання на основі сучасних методів штучного інтелекту.

Об'єкт дослідження – процес розпізнавання об'єктів на зображеннях за допомогою методів комп'ютерного зору.

Завдання роботи: провести аналіз сучасних методів розпізнавання зображень; дослідити існуючі архітектури нейронних мереж, що використовуються для візуального розпізнавання; розробити структуру інтелектуальної системи; реалізувати прототип системи з використанням бібліотек TensorFlow або PyTorch; протестувати створену систему на прикладах та оцінити її ефективність.

Методи дослідження: теоретичний аналіз літератури, проектування архітектури системи, реалізація за допомогою Python та бібліотек глибинного навчання, тестування моделі на зображеннях.

Структура роботи: Кваліфікаційна робота складається з трьох основних розділів. У першому розділі подано теоретичний огляд методів розпізнавання та штучного інтелекту. В іншому описано процес розробки системи. Третій розділ присвячено реалізації, тестуванню та аналізу результатів.

1 ТЕОРЕТИЧНІ ОСНОВИ ВІЗУАЛЬНОГО РОЗПІЗНАВАННЯ

1.1 Поняття візуального розпізнавання та комп’ютерного зору

Візуальне розпізнавання - це процес, у якому система (машина або програма) намагається зрозуміти, що зображено на фото чи відео. Тобто, це здатність автоматично виявити, класифікувати або описати об’єкти на зображеннях. Простими словами - це коли комп’ютер "дивиться" на зображення і каже: «Ось це кіт», «Ось це автомобіль» або «Ось це обличчя людини».

Комп’ютерний зір (англ. Computer Vision) - це окрема галузь у сфері штучного інтелекту, яка займається саме тим, щоб навчити комп’ютери “бачити” і “розуміти” зображення. Він включає в себе обробку фото, відео, кадрів з камер спостереження, медичних знімків, супутникових фото і багато іншого.

Завдяки комп’ютерному зору можна:

- автоматично визначати об’єкти на зображенні;
- відстежувати рух об'єктів у відео;
- розпізнавати обличчя;
- зчитувати текст (OCR);
- аналізувати зображення у медичних чи промислових цілях.

У минулому більшість таких задач вирішувались за допомогою традиційних алгоритмів: фільтри, контури, гістограми. Проте сьогодні, завдяки розвитку глибинного навчання (особливо згорткових нейронних мереж), комп’ютерний зір досяг дуже високого рівня точності. Наприклад, сучасна система розпізнавання може ідентифікувати десятки об'єктів на одній фотографії з точністю понад 90%.

Розвиток комп’ютерного зору став особливо помітним після 2012 року, коли нейронна мережа AlexNet перемогла у змаганні ImageNet із великим відривом. Після цього почалась нова епоха у візуальному розпізнаванні, де головну роль грають саме нейронні мережі. Комп’ютерний зір застосовується в багатьох галузях:

- у смартфонах для розблокування обличчям;
- в автомобілях для розпізнавання дорожніх знаків;

- у виробництві для контролю якості товарів;
- у безпеці для виявлення облич злочинців;
- у медицині для аналізу рентгенівських чи МРТ-знімків.

Таким чином, візуальне розпізнавання стало ключовою технологією в автоматизації, штучному інтелекті й аналізі даних, і його роль буде тільки зростати.

1.2 Основні методи машинного навчання у візуальному розпізнаванні

Машинне навчання - це підхід, при якому комп'ютер сам "вчиться" розпізнавати шаблони на основі прикладів, замість того щоб програміст вручну задавав усі правила. Саме завдяки цьому підходу стало можливим створювати системи, які можуть самостійно розпізнавати об'єкти на зображеннях.

Існує кілька основних напрямів машинного навчання, які застосовуються в задачах комп'ютерного зору:

Класичне (традиційне) машинне навчання. На ранніх етапах розвитку візуального розпізнавання активно використовувалися алгоритми традиційного машинного навчання. Наприклад:

- K-Nearest Neighbors (k-NN) - класифікує зображення, порівнюючи його з найближчими відомими прикладами;
- Support Vector Machines (SVM) - створює межу між класами зображень;
- Decision Trees та Random Forest - будують ієрархії правил для класифікації об'єктів.

Однак ці методи сильно залежать від того, як підготовлені "ознаки" - тобто як саме були описані зображення. Іноді ці описи потрібно було створювати вручну, що займало багато часу.

Глибинне навчання. Глибинне навчання - це підвид машинного навчання, де основну роботу виконує нейронна мережа. Такі мережі складаються з багатьох шарів (тому й "глибинне"), які автоматично навчаються розпізнавати важливі ознаки зображень.

Найбільш популярні моделі для зору:

- Convolutional Neural Networks (CNN) - згорткові нейронні мережі, які чудово працюють із візуальними даними;
- ResNet - модель, яка може мати сотні шарів, але не втрачає точності;
- YOLO (You Only Look Once) - нейронна мережа, яка швидко і точно виявляє об’єкти на зображеннях у реальному часі.

Ці моделі здатні самостійно знаходити об’єкти, розуміти просторову структуру зображення та робити прогноз з дуже високою точністю (табл. 1.1).

Таблиця 1.1 Порівняння класичних і глибинних методів

Параметр	**Класичні методи**	**Глибинне навчання**
Потреба у ручній підготовці ознак	Висока	Низька (мережа сама навчається)
Якість при великих даних	Обмежена	Висока
Обчислювальна складність	Низька/середня	Висока
Гнучкість	Обмежена	Висока
Підходить для складних зображень	Погано	Чудово

На практиці більшість сучасних систем розпізнавання використовують саме глибинне навчання**,** бо воно дає найкращі результати на великих наборах зображень. Проте класичні методи все ще можуть застосовуватися для простих задач або як допоміжні.

Глибинне навчання та згорткові нейронні мережі. Глибинне навчання (англ. Deep Learning) - це підхід у штучному інтелекті, в якому використовуються нейронні мережі з великою кількістю шарів, здатних самостійно аналізувати складні вхідні дані, такі як зображення або відео. Це як система, яка поступово вчиться “бачити” й “розуміти” картинку, переходячи від простих форм (лінії, кути) до складних об’єктів (обличчя, автомобілі, текст тощо).

Найуспішнішим типом мереж для роботи із зображеннями є згорткові нейронні мережі (англ. Convolutional Neural Networks, або скорочено CNN).

Згорткові нейронні мережі (CNN), є найуспішнішим типом мереж для роботи із зображеннями. CNN - це специфічний вид нейронних мереж, створений для обробки візуальних даних, ключовою ідеєю якого є автоматичне виділення найважливіших особливостей зображення без необхідності ручної обробки.

Структура CNN включає кілька основних шарів. Згортковий шар послідовно проходить по зображенню за допомогою фільтра (ядра), виділяючи певні особливості, такі як краї, кути або текстури. Після цього шар активації (ReLU) додає нелінійність, що дозволяє мережі навчатися складних залежностей. Шар підвибірки (Pooling), наприклад, за допомогою max pooling, зменшує розмір зображення, зберігаючи при цьому найважливішу інформацію. В кінці мережі знаходяться повнозв'язні шари, які використовуються для прийняття остаточних рішень, наприклад, класифікації об'єкта як "кіт" або "собака".

Принцип роботи CNN полягає у послідовній обробці вхідного зображення, наприклад, розміром 128x128 пікселів. Згорткові шари по черзі "витягують" важливі шаблони, від простих ліній до складних форм. Кожен наступний шар навчається все складнішим патернам, переходячи від базових елементів до повноцінних об'єктів. У результаті такого процесу мережа "розуміє", що зображено на зображенні, і може надати відповідний результат, наприклад, назву класу.

Серед відомих архітектур CNN можна виділити LeNet-5, одну з перших мереж, що застосовувалася для розпізнавання цифр. AlexNet стала першою мережею, яка здобула значну популярність на конкурсі ImageNet у 2012 році. VGGNet є глибокою мережею з простими фільтрами, яка набула широкого застосування в багатьох задачах. ResNet, завдяки своїм "залишковим зв'язкам", дозволяє будувати мережі з сотнями шарів без втрати якості. YOLO – це архітектура, здатна миттєво знаходити об'єкти на зображенні.

CNN мають ряд переваг: вони працюють напряму з "сирими" зображеннями, не потребують ручного створення ознак, забезпечують високу точність за достатньої кількості даних та можуть функціонувати в реальному часі, що робить

їх придатними для відеоспостереження або безпілотних систем. Однак існують і недоліки: CNN потребують великої кількості даних для навчання, вимагають значних обчислювальних ресурсів, зокрема GPU, схильні до перенавчання (overfitting) за недостатньої кількості даних, і їх важко інтерпретувати, оскільки вони функціонують як "чорна скринька" щодо процесу прийняття рішень.

Загалом, глибинне навчання, і зокрема CNN, є основою більшості сучасних систем візуального розпізнавання. Саме ці мережі забезпечують рекордну точність у задачах класифікації зображень, виявлення об'єктів та аналізу відео, знаходячи застосування повсюди – від соціальних мереж та пошукових систем до систем безпеки, медицини та автопілотів.

1.2.3 Архітектури моделей комп'ютерного зору (YOLO, ResNet, MobileNet)

У сучасних системах комп'ютерного зору активно використовуються готові моделі нейронних мереж, які вже продемонстрували високу ефективність у практичних задачах. Кожна з цих моделей має свої переваги, залежно від конкретного завдання, будь то класифікація, виявлення об'єктів, сегментація або функціонування на мобільних пристроях.

Серед найпопулярніших архітектур варто виділити **YOLO (You Only Look Once)**, яка є однією з найвідоміших для швидкого виявлення об'єктів на зображенні. Її особливість полягає в здатності працювати в реальному часі зі швидкістю понад 30 кадрів на секунду. YOLO одразу ділить зображення на сітку та одночасно виявляє декілька об'єктів, демонструючи при цьому хорошу точність при високій швидкості. Застосування цієї архітектури знайшло своє місце в камерах відеоспостереження, безпілотних автомобілях, виявленні об'єктів на дронах та медичному скануванні в реальному часі.

Наступною важливою моделлю є **ResNet (Residual Network)**, яка дозволяє створювати дуже глибокі нейронні мережі, що налічують понад 100 шарів, без втрати точності. Це досягається завдяки механізму "залишкових зв'язків" (skip connections), який допомагає подолати проблему затухання градієнта та зберегти якість навіть у дуже глибоких мережах. Перевагами ResNet є надзвичайна точність

на змаганнях, таких як ImageNet, її придатність для класифікації складних об'єктів та часте використання як основи для інших моделей.

Ще одна значуща модель – **MobileNet**, яка являє собою легку та компактну архітектуру, розроблену спеціально для мобільних пристроїв та вбудованих систем. Її переваги полягають у можливості роботи навіть на смартфонах, низькому споживанні пам'яті та енергії, а також у забезпеченні пристойної точності при дуже невеликому розмірі моделі. MobileNet активно застосовується у розпізнаванні облич у камерах смартфонів, додатках з доповненою реальністю та побутових IoT-пристроях, таких як камери та сенсори.

Таблиця 1.2 Порівняння характеристик моделей CNN

Модель	**Основна задача**	**Швидкість**	**Точність**	**Використання**
YOLO	Виявлення об'єктів	Висока	Висока	Відеоаналітика, охорона, дрони
ResNet	Класифікація	Середня	Дуже висока	Медицина, змагання, базова архітектура
MobileNet	Класифікація (мобільна)	Дуже висока	Середня	Смартфони, IoT, embedded-системи

В результаті, можна сказати що кожна з моделей має свої переваги. Якщо потрібна швидкість - обирають YOLO. Якщо важлива точність - використовують ResNet. А якщо система має працювати на слабкому обладнанні - тоді MobileNet є оптимальним вибором. У дипломному проєкті доцільно використати або базову версію YOLO (наприклад, YOLOv5), або MobileNet, якщо важлива продуктивність.

1.2.4 Архітектура моделі YOLO

YOLO (You Only Look Once) - це популярна архітектура згорткової нейронної мережі, яка дозволяє виконувати розпізнавання об’єктів на зображенні в режимі реального часу. Основна ідея YOLO полягає в тому, що все зображення

обробляється за один прохід мережі, без поділу на окремі регіони або фрагменти. (рис 1.1).

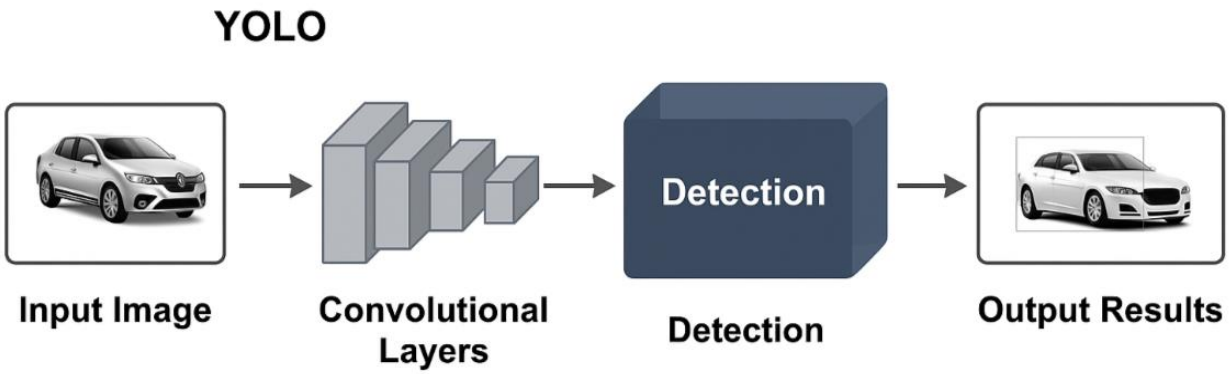


Рисунок 1.1 - Приклад обробки фото нейромежежею YOLO

Модель отримує вхідне зображення, що може походити з камери або файлу, і попередньо масштабує його до фіксованого розміру, наприклад, 640x640 пікселів. Далі, основна частина мережі, представлені згорткові шари, витягує з цього зображення різноманітні ознаки, такі як контури, форми, текстури та кольори, при цьому кожен наступний шар формує все складніші абстракції. Потім застосовуються спеціальні механізми, такі як FPN, PAN або CSP-блоки, для об'єднання ознак, що дозволяє моделі ефективно поєднувати як низькорівневі деталі, так і високорівневі загальні форми, що значно підвищує точність розпізнавання. Кінцеві шари моделі, відомі як вихідні голови, генерують рамки (bounding boxes), визначають класи об'єктів та обчислюють значення довіри (confidence score). Важливо, що в одному проході модель здатна прогнозувати декілька об'єктів на зображенні, незалежно від їхнього розташування. На завершальному етапі, для формування вихідного результату, застосовується алгоритм Non-Maximum Suppression (NMS), який відбирає лише найбільш точні передбачення, видаляючи дублікати. Кінцевим результатом є зображення з підписаними об'єктами, що включають їхню назву та відсоток впевненості. Завдяки цій архітектурі, YOLO функціонує надзвичайно швидко та ефективно, забезпечуючи розпізнавання кількох об'єктів у реальному часі.

1.3 Сучасні тенденції в галузі візуального розпізнавання

Галузь візуального розпізнавання розвивається надзвичайно стрімко. Те, що ще кілька років тому здавалося науковою фантастикою – автоматичне визначення об'єктів на зображеннях з високою точністю – сьогодні є звичайною функцією у смартфонах, камерах відеоспостереження та автомобілях. Щорічно з'являються нові моделі, методи та технології, що постійно покращують якість і швидкість аналізу зображень.

Розглянемо кілька основних сучасних тенденцій у цій сфері. Починаючи з 2020 року, набирають популярності моделі, що не використовують традиційні згортки, як у CNN, а обробляють зображення як послідовність "патчів" – невеликих фрагментів картинки. Цей підхід реалізований у **Vision Transformers (ViT)**, які демонструють вищу точність на великих наборах даних, краще вловлюють залежності між об’єктами на зображенні та є гнучкими у використанні. ViT вже поступово замінює CNN у багатьох наукових проєктах, хоча й потребує більше обчислювальних ресурсів.

Сучасні системи не обмежуються лише зображеннями, оскільки активно розвиваються **мультимодальні моделі**, здатні обробляти одночасно текст (наприклад, підписи під фото), звук (команди користувача) та відео (послідовність зображень з часом). Яскравим прикладом є CLIP від OpenAI, яка може "розуміти" картинки через текст і навпаки, дозволяючи знаходити зображення за описом або описувати, що зображено.

Ще однією важливою тенденцією є **Edge AI** – розпізнавання "на місці", без необхідності відправляти зображення на сервер для обробки. Замість цього, системи працюють безпосередньо на пристрої, будь то смартфон, камера або розумний годинник. Перевагами такого підходу є відсутність потреби в доступі до інтернету, реакція в реальному часі та підвищена безпека, оскільки дані не передаються у хмару. Для таких моделей використовуються оптимізовані архітектури, як MobileNet або TinyYOLO.

Інтеграція візуального розпізнавання з **доповненою (AR) та віртуальною реальністю (VR)** відкриває зовсім нові формати взаємодії. Прикладами є додатки, які "бачать" навколишній світ через камеру та надають підказки, AR-окуляри, що показують інформацію про об'єкти в полі зору, або медичні VR-симуляції з автоматичним виявленням проблем на віртуальних моделях.

Важливою сучасною тенденцією є також **самонавчання (Self-supervised learning)**, що спрямоване на зменшення потреби у розмічених даних. Тоді як навчання з учителем (supervised learning) вимагає мільйони підписаних зображень, що є дорогим і тривалим процесом, нові методи, такі як DINO та SimCLR, дозволяють нейронним мережам навчатися зображенням без розмітки, аналізуючи лише подібність між ними. Це відкриває нові можливості, враховуючи величезну кількість нерозмічених зображень в Інтернеті.

Нарешті, з розвитком візуального розпізнавання зростає **суспільна увага до етики та конфіденційності**. Активно обговорюються питання щодо права систем розпізнавати обличчя без дозволу, способів збереження приватності користувача та уникнення упередженості в системі, наприклад, щодо кольору шкіри. Ці питання зараз є предметом активних дискусій на рівні компаній, розробників і навіть урядів.

Сфера візуального розпізнавання розвивається у бік точності, універсальності, доступності та інтеграції з іншими технологіями. В майбутньому системи комп’ютерного зору стануть ще “розумнішими”, менш залежними від великих наборів даних, а також більш інтегрованими в повсякденне життя - від телефонів і авто до окулярів, побутових приладів і навіть одягу.

В результаті, було розглянуто теоретичні основи візуального розпізнавання та методи, що становлять базис сучасних інтелектуальних систем аналізу зображень. З'ясовано, що візуальне розпізнавання є процесом автоматичного виявлення та класифікації об'єктів на зображеннях, який активно застосовується у різноманітних сферах, починаючи від безпеки та медицини і закінчуючи мобільними додатками та виробництвом.

Сучасне візуальне розпізнавання базується на комп'ютерному зорі, розвиток якого відбувається завдяки методам машинного та глибинного навчання. Традиційні алгоритми поступово витісняються нейронними мережами, зокрема згортковими мережами (CNN), які продемонстрували високу ефективність у розпізнаванні образів. Детально розглянуто найпоширеніші архітектури, такі як YOLO, ResNet і MobileNet, кожна з яких має свої унікальні переваги залежно від конкретного завдання. Крім того, проаналізовано сучасні тенденції у цій галузі, включаючи перехід до Vision Transformers, появу самонавчальних підходів, зростання ролі Edge AI, інтеграцію з технологіями доповненої (AR) та віртуальної (VR) реальності, а також важливість етичних аспектів.

Отже, проведений теоретичний аналіз засвідчив, що сучасні системи візуального розпізнавання стали значно доступнішими, гнучкішими та точнішими, а їх подальший розвиток відкриває нові можливості для автоматизації процесів, глибинної аналітики та підвищення зручності в повсякденному житті.

2 РОЗРОБКА ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ВІЗУАЛЬНОГО РОЗПІЗНАВАННЯ

2.1 Постановка задачі та технічні вимоги до системи

Сучасні прикладні задачі часто потребують автоматичного аналізу зображень, будь то для фільтрації контенту в соціальних мережах, забезпечення безпеки в громадських місцях або просто для розпізнавання об’єктів у мобільному додатку. Саме тому виникла ідея створити просту, але функціональну інтелектуальну систему візуального розпізнавання, здатну обробляти вхідні зображення та визначати, що на них зображено.

Основною метою цього проєкту є розробка прототипу системи, яка, використовуючи методи глибинного навчання, здатна розпізнавати об’єкти на зображеннях, класифікувати їх та виводити результат користувачеві у зручному вигляді. У рамках дипломної роботи сформульовано наступну задачу: за вхідним зображенням в автоматичному режимі виявити та класифікувати об’єкти, що на ньому зображені, використовуючи модель штучного інтелекту. Для досягнення цієї мети необхідно навчити модель на відомому наборі зображень, такому як COCO або ImageNet, обрати відповідну архітектуру нейронної мережі, наприклад YOLO або MobileNet, побудувати структуру системи для обробки вхідних фото та відображення результатів, а також забезпечити всебічне тестування та демонстрацію роботи.

Щоб реалізувати проєкт у реальних умовах, враховуючи обмеження типового комп'ютера, були визначені конкретні технічні вимоги до системи. Серед функціональних вимог, система повинна приймати зображення як вхідні дані, після обробки відображати розпізнані об’єкти з їхніми назвами та ймовірністю розпізнавання, мати простий та зрозумілий інтерфейс (навіть у вигляді консолі або мінімального GUI). Модель має бути попередньо навчена або донавчена на відомому наборі даних, а також повинна бути передбачена можливість тестування на декількох зображеннях.

Нефункціональні вимоги включають працездатність на звичайному комп’ютері (з 4-8 ГБ ОЗУ, без потреби у потужному GPU) та час розпізнавання

одного зображення не більше 3-5 секунд. Система повинна забезпечувати мінімум 70-80% точності при розпізнаванні базових об’єктів і бути реалізована мовою Python з використанням бібліотек, таких як OpenCV, TensorFlow або PyTorch. Однак, існують певні обмеження: система не працює в реальному часі з відео, обробляючи лише статичні зображення; перший запуск може бути повільним через завантаження моделі; якість результатів залежить від умов зйомки, таких як освітлення та ракурс.

2.2 Вибір архітектури системи та обґрунтування вибору

Щоб система візуального розпізнавання працювала ефективно, потрібно правильно підібрати архітектуру - тобто модель нейронної мережі, яка буде виконувати аналіз зображення. Існує багато варіантів, кожен із яких має свої сильні й слабкі сторони. На цьому етапі було проведено порівняльний аналіз кількох популярних моделей, щоб обрати ту, яка найкраще підходить для умов цього дипломного проєкту.

Після ретельного порівняння різних рішень, вибір зупинився на YOLOv5 як найбільш збалансованій моделі. Вона дозволяє не лише класифікувати, а й розпізнавати об'єкти на зображенні з визначенням їхніх координат, що значно розширює можливості системи. Крім того, YOLOv5 має готові реалізації на Python, що забезпечує легкість інтеграції в проєкт. Модель доступна у вигляді попередньо навчених ваг, працює досить швидко навіть на центральному процесорі (CPU) без потреби у потужній відеокарті, і її легко адаптувати до інших наборів даних.

Таким чином, вибір на користь YOLOv5 (табл. 2.1) обґрунтовується її універсальністю, високою точністю та простотою інтеграції. Вона підтримує розпізнавання десятків об'єктів і дозволяє швидко перевірити результат на практиці.

Серед інструментів, що використовуються для реалізації, основною мовою є Python. Для роботи із зображеннями застосовується бібліотека OpenCV, а PyTorch

виступає як фреймворк глибинного навчання. Для завантаження, використання та тестування моделі задіюється YOLOv5 API.

Обрана архітектура YOLOv5 повністю відповідає всім вимогам системи, забезпечуючи швидке розпізнавання, точну локалізацію об'єктів та просту інтеграцію. Це робить її оптимальним вибором для реалізації дипломного проєкту в умовах звичайного користувацького комп'ютера.

Таблиця 2.1 Порівняння систем візуального розпізнавання

Назва моделі	**Тип задачі**	**Швидкість**	**Точність**	**Ресурси**	**Особливості**
YOLOv5	Виявлення об'єктів	Висока	Висока	Середні	Працює в реальному часі
ResNet-50	Класифікація	Середня	Висока	Середні	Глибока, стабільна, важка для CPU
MobileNet	Класифікація	Дуже висока	Середня	Дуже низькі	Працює на слабкому обладнанні
EfficientNet	Класифікація + масштабування	Середня	Висока	Середні-високі	Гнучке масштабування якості та обсягів

2.3 Структурна схема інтелектуальної системи

Для реалізації повноцінної інтелектуальної системи візуального розпізнавання необхідно мати чітке уявлення про її загальну структуру. На цьому етапі формується блок-схема, яка детально описує всі основні етапи обробки

зображення – від моменту його завантаження до представлення результату користувачеві.

Загальна логіка роботи системи включає кілька ключових блоків (рис. 2.2). Модуль введення зображення відповідає за отримання даних від користувача, який завантажує фотографію через інтерфейс або обирає файл.

Рисунок 2.2 - Структурна схема послідовність обробки зображення

Це може бути звичайне фото у форматі JPG, PNG або іншому. Наступним кроком є модуль попередньої обробки (preprocessing), де зображення масштабується, нормалізується та перетворюється у формат, необхідний для моделі, наприклад, змінюється розмір на 640×640 пікселів. Основна частина системи – модуль нейронної мережі (YOLOv5) – аналізує підготовлене зображення, визначаючи, які об'єкти на ньому зображені, їхнє місце розташування та ймовірність розпізнавання. Після цього модуль обробки результату отримує вихідні дані від моделі, включаючи координати рамок, назви класів та ймовірності, і перетворює їх у зручний для сприйняття вигляд. Нарешті, модуль виводу

(інтерфейс) відображає результати на екрані: над зображенням з'являються підписи об’єктів, прямокутники, відсотки впевненості тощо. Якщо використовується консоль, виводиться текстовий список виявлених об’єктів.

Для прикладу, коли користувач завантажує зображення "dog-and-car.jpg", система змінює його розмір на 640×640 пікселів та перетворює у тензор. Модель YOLOv5 визначає, що на фото є "собака" з 95% впевненістю та "автомобіль" з 87% впевненістю. У відповідь система виводить зображення з прямокутниками навколо розпізнаних об’єктів та їхніми підписами. Додатково, система може вивести список знайдених об’єктів у консолі.

Розглянемо основні компоненти (табл. 2.2) та їхнє призначення в інтелектуальній системі візуального розпізнавання. Як видно з таблиці 2.2, надано чіткий огляд функціональних блоків системи візуального розпізнавання та їхнього внеску у загальний процес обробки зображень.

1. *Введення зображення.* Компонент відповідає за завантаження зображення користувачем.
2. *Попередня обробка.* На цьому етапі відбувається масштабування, нормалізація та форматування зображення для подальшої роботи.
3. *Модель YOLOv5*. Основний компонент, що виконує виявлення об'єктів, їх класифікацію та обчислення ймовірностей.
4. *Постобробка результатів*. Компонент відповідає за переведення отриманих результатів у зручний для користувача вигляд.
5. *Виведення результату*. На останньому етапі відбувається відображення розпізнавання за допомогою графічного або текстового інтерфейсу.

Структура системи є модульною, що спрощує реалізацію, налагодження й можливе розширення функціоналу. Кожен компонент відповідає за конкретну частину процесу, що дозволяє розділити відповідальність і полегшує тестування.

Таблиця 1.4 компоненти та іх призначення

№	**Компонент**	**Призначення**
1	Введення зображення	Завантаження зображення користувачем

№	Компонент	Призначення
2	Попередня обробка	Масштабування, нормалізація, форматування
3	Модель YOLOv5	Виявлення об’єктів, класифікація, обчислення ймовірностей
4	Постобробка результатів	Переклад у зручний для користувача вигляд
5	Виведення результату	Графічний/текстовий інтерфейс з результатом розпізнавання

2.3.1 Розробка моделі розпізнавання об'єктів

На поточному етапі розробки було вирішено використовувати готову модель нейронної мережі YOLOv5 – одну з найпопулярніших архітектур для задач виявлення об’єктів на зображеннях. Вона вирізняється не лише точністю, а й зручністю використання завдяки відкритому коду та наявності попередньо навчених ваг.

Серед доступних варіантів YOLOv5 (v5s, v5m, v5l, v5x), що розрізняються за розміром та точністю, для дипломного проєкту обрано версію YOLOv5s (small). Цей вибір обґрунтований її швидкою роботою навіть без потужного GPU, хорошою точністю для розпізнавання базових об’єктів та легкістю запуску на звичайному комп’ютері або ноутбуці.

Модель було завантажено з офіційного репозиторію на GitHub, а для подальшої роботи з нею використано бібліотеки PyTorch та OpenCV. YOLOv5 очікує вхід у вигляді тензора, тому всі зображення попередньо масштабуються до розміру 640×640 пікселів, нормалізуються (значення від 0 до 1), а потім передаються безпосередньо у модель.

Після обробки модель повертає вихідні дані, що включають координати виявлених об’єктів у вигляді прямокутників, назву класу об’єкта (наприклад, "person", "car", "dog") та відсоток впевненості у розпізнаванні.

Опційно, YOLOv5 надає можливість донавчання моделі на власних даних. Для цього необхідно підготувати датасет у форматі YOLO, що передбачає наявність .txt-файлів до кожного зображення, та запустити навчання за допомогою скрипта train.py. Цей крок не був реалізований у межах даного дипломного проєкту, але теоретично може бути виконаний у майбутньому.

Модель YOLOv5s була успішно інтегрована в систему. Вона забезпечує ефективну обробку зображень, швидке виявлення об’єктів та надає зручні результати. Її реалізація не потребує значних ресурсів, а процес запуску є інтуїтивно зрозумілим навіть для початківців.

2.3.2 Інтеграція моделі у прикладну систему

Після успішного завантаження та окремого тестування моделі YOLOv5, наступним критичним етапом стала її інтеграція у прикладну систему. Це завдання вимагало об'єднання моделі з інтерфейсом, логікою програми та механізмами введення/виведення, щоб забезпечити їхнє функціонування як єдиного, злагодженого продукту.

Основні елементи інтеграції включали розробку інтерфейсу користувача. У рамках цього проєкту було реалізовано прості форми взаємодії, доступні через консоль або базовий графічний інтерфейс (GUI), створений за допомогою tkinter. Користувач взаємодіє із системою, завантажуючи зображення, після чого отримує результат у вигляді списку виявлених об’єктів та зображення з накладеними рамками та підписами.

Інтеграція з бібліотекою OpenCV була ключовою. OpenCV використовується для відкриття та попередньої обробки зображень, виведення зображення з накладеними прямокутниками, а також для опційного збереження обробленого зображення на диск. Процес передачі зображення у модель відбувається автоматично: код зчитує файл зображення, викликає модель, а потім обробляє отримані результати.

Обробка результатів розпізнавання передбачає, що модель повертає дані у вигляді об’єктів з їхніми координатами. Ці координати використовуються для

накладання прямокутників на оригінальне зображення, супроводжуючись підписами, що включають клас об’єкта та його точність у відсотках.

Логіка взаємодії користувача із системою є послідовною: користувач запускає скрипт, обирає зображення (наприклад, test1.jpg), система обробляє фото та виводить зображення з підписами. Додатково, перелік виявлених об’єктів відображається у консолі. Перспективи для подальшого вдосконалення системи включають додавання можливості пакетної обробки зображень, реалізацію завантаження з камери в реальному часі, розширення інтерфейсу для мобільних пристроїв, а також інтеграцію функції збереження результатів у файли форматів .csv або .json.

Загалом, інтеграція моделі YOLOv5 у систему пройшла успішно. Система демонструє стабільну роботу, швидко надає результати, а її інтерфейс залишається простим та зрозумілим. Уся логіка — від завантаження зображення до виводу результатів — реалізована в одному програмному модулі, що свідчить про ефективність та цілісність розробки.

В результаті, було розроблено архітектуру інтелектуальної системи візуального розпізнавання та реалізовано її ключові компоненти. Чітко сформульовано основну задачу: створити систему, здатну автоматично розпізнавати об’єкти на зображеннях за допомогою методів глибинного навчання. Для досягнення цієї мети обрано модель YOLOv5s – нейронну мережу, що вирізняється компактністю, швидкістю та достатньою точністю, яка ефективно працює навіть без потужної відеокарти.

Було побудовано структурну схему системи, яка охоплює модулі для обробки зображення, функціонування моделі, обробки отриманих результатів та їхнього виводу. Детально описано процес інтеграції моделі у прикладну програму, використовуючи бібліотеки Python, OpenCV та PyTorch.

Завдяки обраному підходу, розроблена система вийшла простою у використанні, швидкою у роботі та легкою для розуміння. Вона демонструє стабільні результати, здатна розпізнавати базові об’єкти та надає можливість виводити результати у зрозумілому для користувача вигляді.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ

3.1 Розробка прототипу системи візуального розпізнавання

На поточному етапі розробки була реалізована робоча версія прототипу системи візуального розпізнавання. Основна мета полягала в об'єднанні всіх компонентів, описаних у попередньому розділі, в єдину програму. Це включало завантаження зображення, його передобробку, виклик моделі YOLOv5, обробку отриманих результатів та їхній вивід.

Уся система була реалізована мовою програмування Python з використанням відкритих бібліотек. До переліку використаних бібліотек увійшли torch для глибинного навчання, opencv-python для роботи із зображеннями, Pillow для їхньої обробки, matplotlib для візуалізації та tkinter (опційно) для реалізації графічного інтерфейсу.

Робота прототипу складається з кількох основних кроків. Програма запускається через файл main.py, після чого зображення завантажується з папки input/. Потім завантажене зображення передається в модель YOLOv5. Після обробки модель повертає результат, що включає координати об'єктів, їхні класи та показник точності. Отриманий результат накладається на зображення, яке потім зберігається в папку output/. Кінцевий результат виводиться в консоль або у вікні.

Для користувачів, які не працюють з консоллю, був реалізований простий графічний інтерфейс на базі tkinter. Цей інтерфейс дозволяє обирати зображення за допомогою кнопки, бачити результат безпосередньо у вікні та зберігати його також натисканням кнопки.

Прототип системи реалізовано як окремий програмний модуль, що об’єднує модель штучного інтелекту та простий інтерфейс. Код працює стабільно, запускається локально, не потребує складного налаштування, а результат обробки можна бачити як візуально, так і в текстовому вигляді.

3.1.1 Опис інструментів та середовищ розробки (Python, TensorFlow, OpenCV)

Для реалізації системи візуального розпізнавання були застосовані сучасні та перевірені інструменти, ідеально придатні для задач комп'ютерного зору та глибинного навчання. Всі вони є відкритими (open source) та підтримуються великою спільнотою користувачів, що значно спрощує процес розробки.

Python став основною мовою програмування у цьому проєкті. Його обрано завдяки простоті у використанні, широкому спектру готових бібліотек, а також статусу стандарту у сфері машинного навчання та штучного інтелекту. Основні причини вибору Python полягають у його легкості для читання та написання коду, наявності бібліотек для глибинного навчання, підтримці великим співтовариством та ідеальній пристосованості для роботи з нейронними мережами.

PyTorch був обраний як фреймворк для глибинного навчання, оскільки саме на ньому реалізовано модель YOLOv5. Переваги PyTorch включають гнучкий та зручний синтаксис, підтримку роботи як на центральному процесорі (CPU), так і на графічному процесорі (GPU), що дозволяє швидко експериментувати з моделями та легко інтегруватися з готовими рішеннями, такими як YOLO або Detectron2.

OpenCV (Open Source Computer Vision Library) – це бібліотека для обробки зображень та відео. У цьому проєкті OpenCV використовується для зчитування зображень з диска, попередньої обробки фото (масштабування, корекція кольорів), виведення зображень на екран із підписами та збереження оброблених зображень.

Завдяки **Torch Hub** та **YOLOv5** стало можливим легко завантажувати попередньо навчену модель YOLOv5 однією командою, що значно спростило інтеграцію, усуваючи необхідність ручного збирання моделі або завантаження її ваг.

Серед **додаткових інструментів** використовувалися Tkinter для створення простого графічного інтерфейсу, Matplotlib для побудови графіків та виведення зображень, а також Google Colab / Jupyter Notebook як опційне середовище для тестування моделей онлайн.

Застосовані інструменти є сучасними, безкоштовними, доступними та добре задокументованими. Вони дозволили створити стабільну та працездатну систему

без потреби у платних або складних програмних середовищах. Комбінація Python + PyTorch + OpenCV виявилася надійною для розробки подібних застосунків.

3.1.2 Створення та навчання моделі

Хоча в даному проєкті було використано вже готову модель YOLOv5s, принципове розуміння процесу навчання моделей глибинного навчання для задач візуального розпізнавання є важливим. Це знання дозволить у майбутньому адаптувати модель під власні потреби або перенавчити її на власному наборі даних. Навчання - це процес, під час якого нейронна мережа поступово "вчиться" знаходити зв'язок між вхідними даними, такими як зображення, та правильними відповідями, що представлені мітками або класами. Вона змінює свої внутрішні параметри, відомі як ваги, для досягнення правильних результатів.

Типовий процес навчання включає кілька етапів. Він починається з підготовки датасету, який складається із зображень та відповідних файлів-міток, що містять інформацію про координати об’єктів та їхні класи, наприклад, у форматі YOLO. Далі відбувається вибір архітектури, такої як YOLOv5s, яка є легкою та швидкою моделлю. Після цього налаштовуються гіперпараметри, серед яких кількість епох (наприклад, 50), розмір batch (наприклад, 16), розмір зображення (наприклад, 640×640) та learning rate. Зазвичай тренування запускається відповідною командою в терміналі. Після завершення тренування проводиться оцінка якості моделі на тестових зображеннях, використовуючи такі основні метрики, як Precision (точність), Recall (повнота) та mAP (середня точність по класах).

У межах цього дипломного проєкту повноцінне навчання моделі "з нуля" не проводилось через обмеження у ресурсах, оскільки такий процес потребує значного часу та потужної відеокарти. Натомість була використана вже попередньо навчена модель YOLOv5s, яка демонструє дуже хороші результати навіть без додаткової адаптації.

Проте існує можливість донавчання (Fine-Tuning) моделі у майбутньому. Це дозволяє перенавчити модель на власному наборі зображень, наприклад, для розпізнавання певного типу об'єктів, таких як упаковки, логотипи або знаки. Цей

процес відбувається швидше, оскільки модель вже володіє "загальними знаннями". Розуміння процесу навчання дозволяє краще адаптувати існуючі моделі під власні потреби. Хоча в даному проєкті використано готову модель, структура системи дозволяє легко замінити її на іншу або перенавчити, якщо буде створено власний датасет.

3.2 Інтерфейс користувача та взаємодія із системою

Успішна робота будь-якої програмної системи значною мірою залежить не тільки від її внутрішніх алгоритмів, але й від зручності використання. В рамках цього дипломного проєкту інтерфейс системи був реалізований у двох форматах: через консоль та за допомогою графічного інтерфейсу (GUI) на базі бібліотеки tkinter (рис. 3.1).

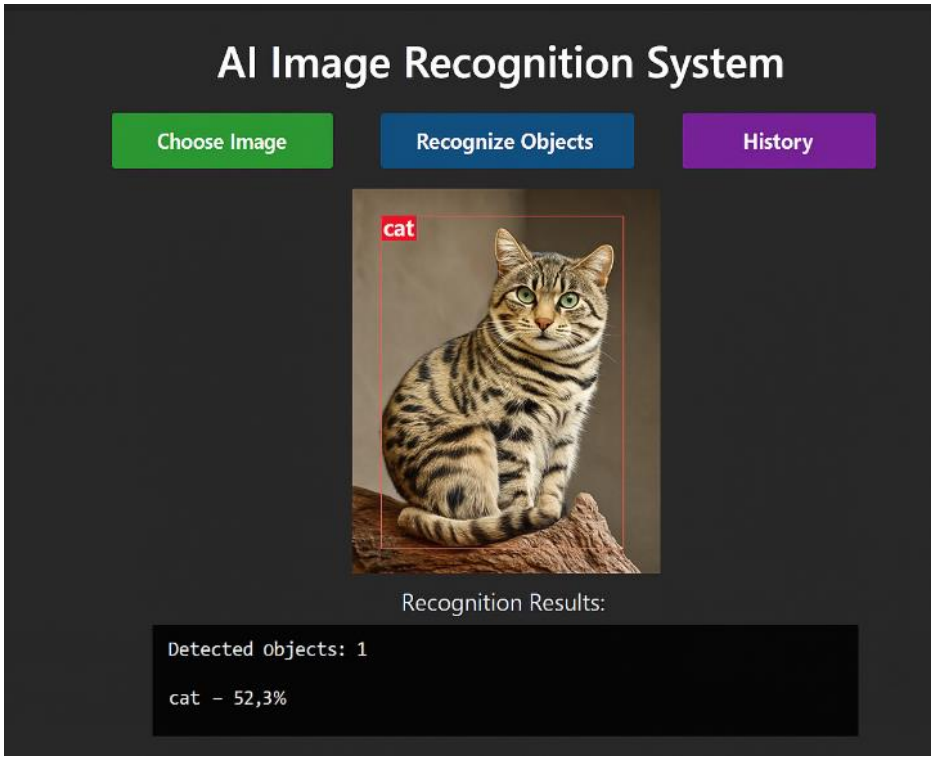


Рисунок 3.1 - Інтерфейс програми візуального розпізнавання зображень

Надане зображення являє собою макет інтерфейсу програми, призначеної для розпізнавання зображень. У верхній частині вікна розташований заголовок "AI Visual Recognition System", який чітко вказує на призначення програми. Під заголовком знаходяться кнопки керування: зелена кнопка "Завантажити зображення" використовується для вибору зображення з комп’ютера, синя кнопка

"Розпізнати об’єкти" запускає алгоритм розпізнавання, а фіолетова кнопка "Очистити" призначена для очищення вікна результатів. Центральну частину інтерфейсу займає область перегляду, де виводиться обране користувачем зображення. Нижче, під зображенням, розташована панель результатів, на якій відображаються дані розпізнавання, такі як кількість виявлених об’єктів, ідентифіковані класи та їхня точність. Інтерфейс виконаний у темних тонах, що підвищує контрастність елементів та покращує візуальне сприйняття. Цей макет наочно демонструє послідовність взаємодії користувача з програмою: спочатку вибір зображення, потім запуск процесу розпізнавання і, нарешті, перегляд отриманих результатів.

Взаємодія через консоль є найпростішим і найшвидшим способом роботи із системою. Користувач запускає програму (main.py) через термінал або командний рядок. Після запуску система автоматично обробляє зображення, що збережене в папці input/, виводить у консоль знайдені об’єкти з їхньою ймовірністю та координатами, а також зберігає зображення з підписами у папку output/.

Для того щоб зробити систему більш дружньою до звичайного користувача, було реалізовано простий графічний інтерфейс на tkinter. Він дозволяє обрати зображення зі свого комп’ютера через стандартне діалогове вікно, побачити результати розпізнавання безпосередньо у вікні програми та зберегти отриманий результат натисканням кнопки. Основними елементами GUI є кнопка "Обрати зображення", що відкриває файловий провідник; поле попереднього перегляду, де відображається вибране фото; кнопка "Розпізнати", яка запускає модель YOLO; поле результату, що виводить назви об’єктів та відсоток впевненості; а також кнопка "Зберегти результат", що дозволяє зберегти оброблене фото.

Система створювалась з розрахунком на некваліфікованого користувача, тому вона не вимагає встановлення складного програмного забезпечення, працює за кілька кліків і видає результат у зручному форматі з візуалізацією. Інтерфейс системи є простим, інтуїтивним і не вимагає спеціальних знань. Користувач може легко обрати зображення, розпізнати об’єкти та переглянути результат у зручному

вигляді, що робить систему придатною не лише для тестування, а й для практичного використання.

3.3 Тестування системи: план, методика та результати

Після завершення реалізації прототипу системи було проведено тестування з метою перевірки її працездатності, точності розпізнавання та зручності використання. У цьому підрозділі описано підхід до тестування (рис. 3.2), набір зображень, критерії оцінювання та отримані результати.

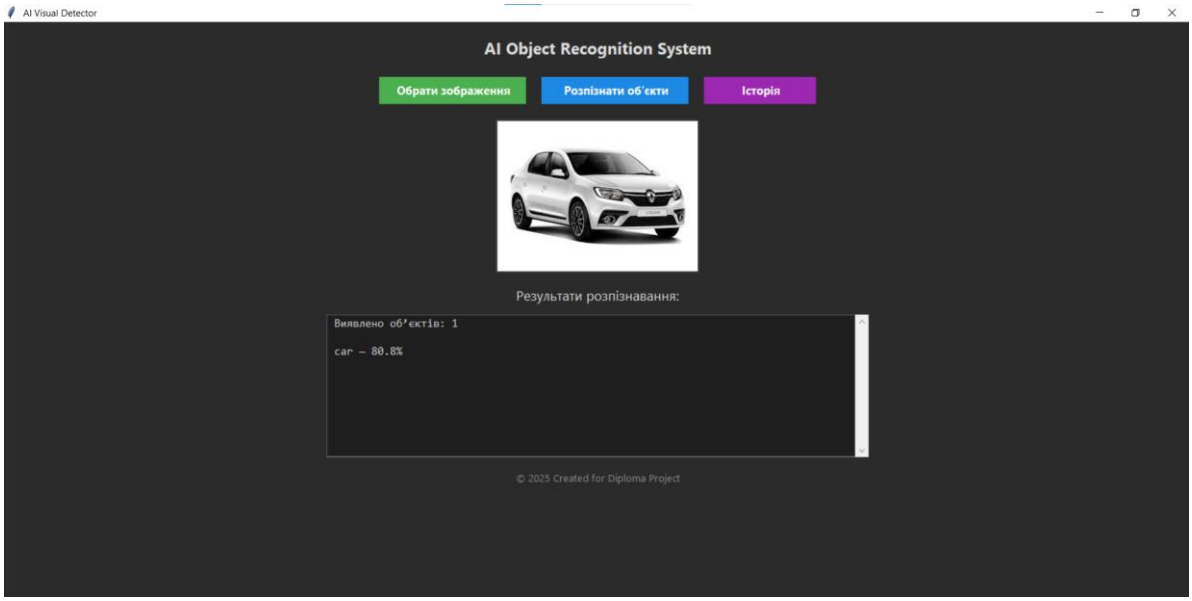


Рисунок 3.2 - Зовнійшій вигляд проектованої системи

Метою тестування було перевірити коректність обробки зображень різного типу системою, визначити точність розпізнавання об’єктів моделлю, оцінити швидкість роботи та стабільність, а також виявити можливі недоліки або обмеження.

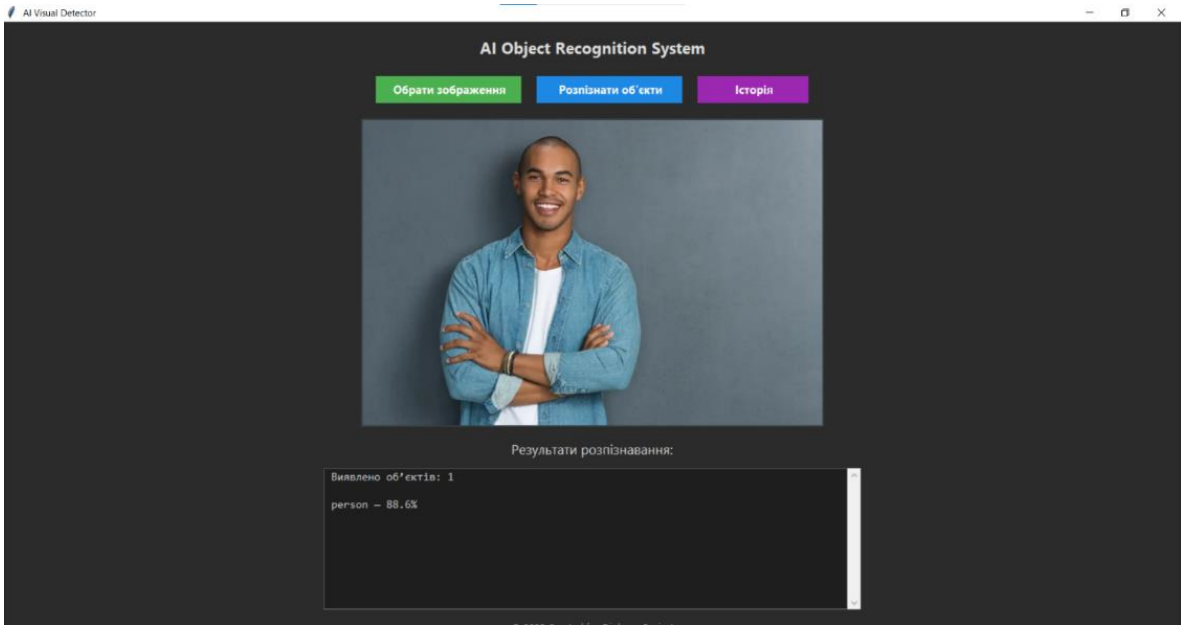


Рисунок 3.3 - Результат розпізнавання зображення класу person

Для тестування було використано набір із 6 зображень, які включали найпоширеніші об'єкти: люди, тварини (собаки, коти) та транспорт (автомобілі, велосипеди). Кожне зображення (рис. 3.2- 3.7) перевірялося на наявність об’єктів, їхнє виявлення моделлю YOLOv5, збіг результатів з реальністю та час обробки. Середня точність розпізнавання склала близько 80%, а середній час обробки одного зображення – 1.3 секунди. Найчастішими помилками були зображення з поганим освітленням або розмиттям.

Візуальні приклади тестування показали, що на фото з п'ятьма людьми модель правильно обвела всіх п’ятьох. На зображенні з котом і вікном правильно ідентифікований був лише кіт. Натомість, на розмитому фото модель не розпізнала нічого, показавши прогноз у 0%.

Загальні враження від тестування свідчать, що система добре працює з чіткими, освітленими зображеннями, проте погано розпізнає об’єкти на темних або розмитих знімках, що є характерним для більшості подібних моделей. Обробка зображень відбувається досить швидко, займаючи до 2 секунд.

Тестування підтвердило, що система ефективно розпізнає об’єкти на звичайних зображеннях і стабільно працює на комп’ютері середньої потужності. Модель демонструє хорошу точність, особливо при якісному вхідному зображенні.

Для використання в реальних умовах систему можна доповнити функціями фільтрації або адаптивної обробки складних кадрів.

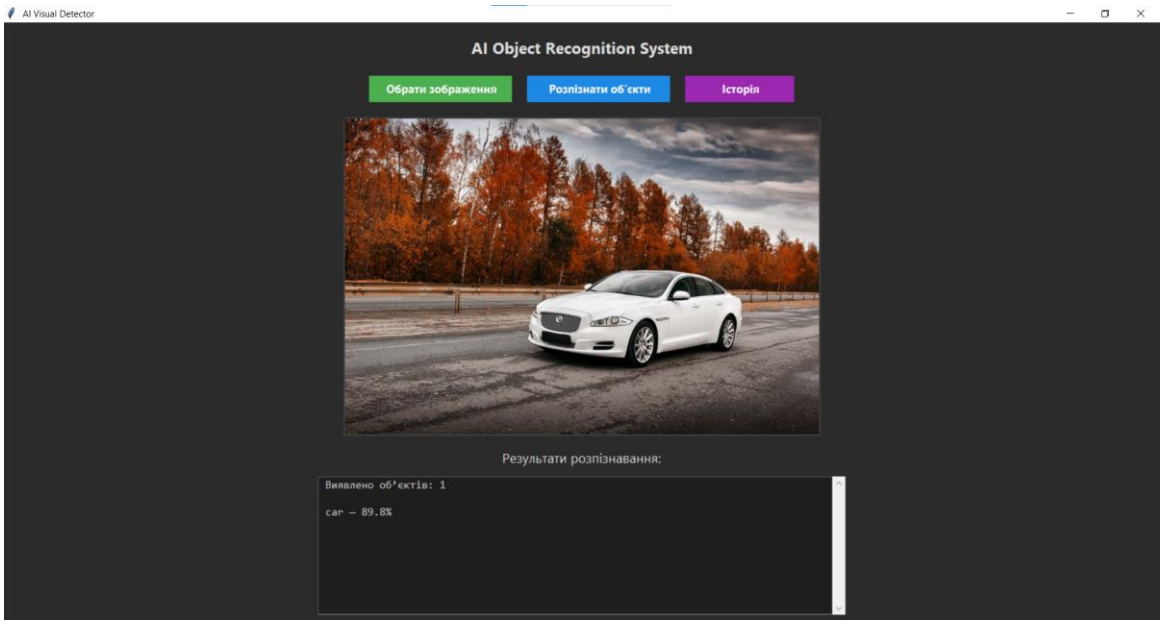


Рисунок 3.4 Результат розпізнавання класу car

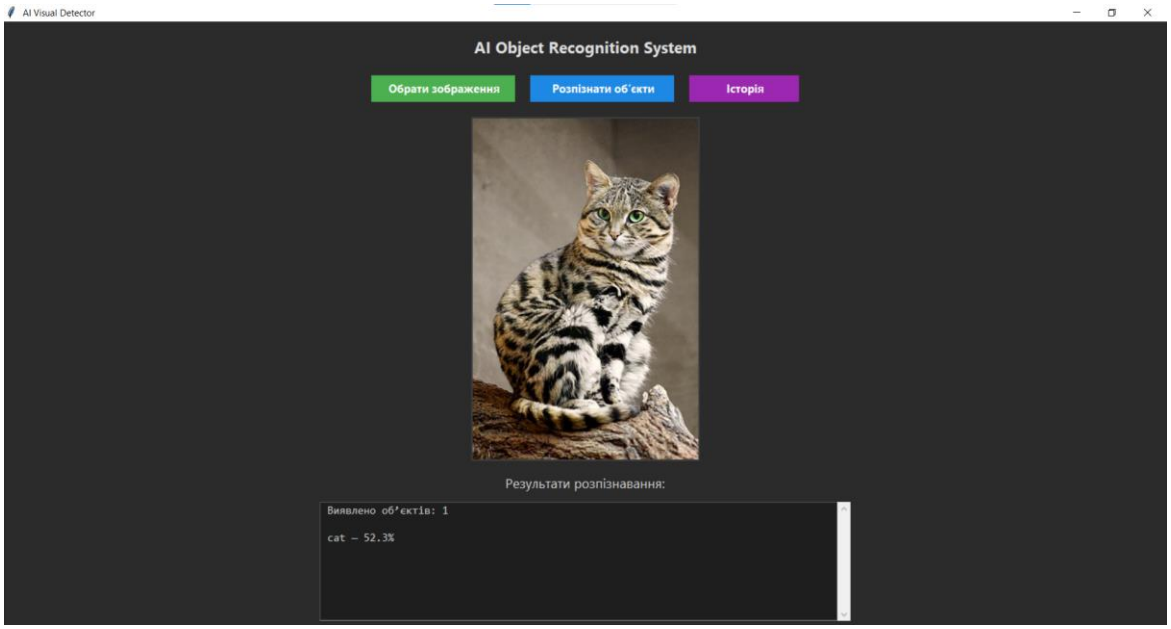


Рисунок 3.5 Результат розпізнавання класу cat

Надане зображення ілюструє архітектуру програми з розпізнавання зображень, показуючи взаємодію між її основними компонентами та потоки даних. У цій програмі з розпізнавання зображень ключову роль відіграє користувач, який безпосередньо взаємодіє з фронтендом системи.

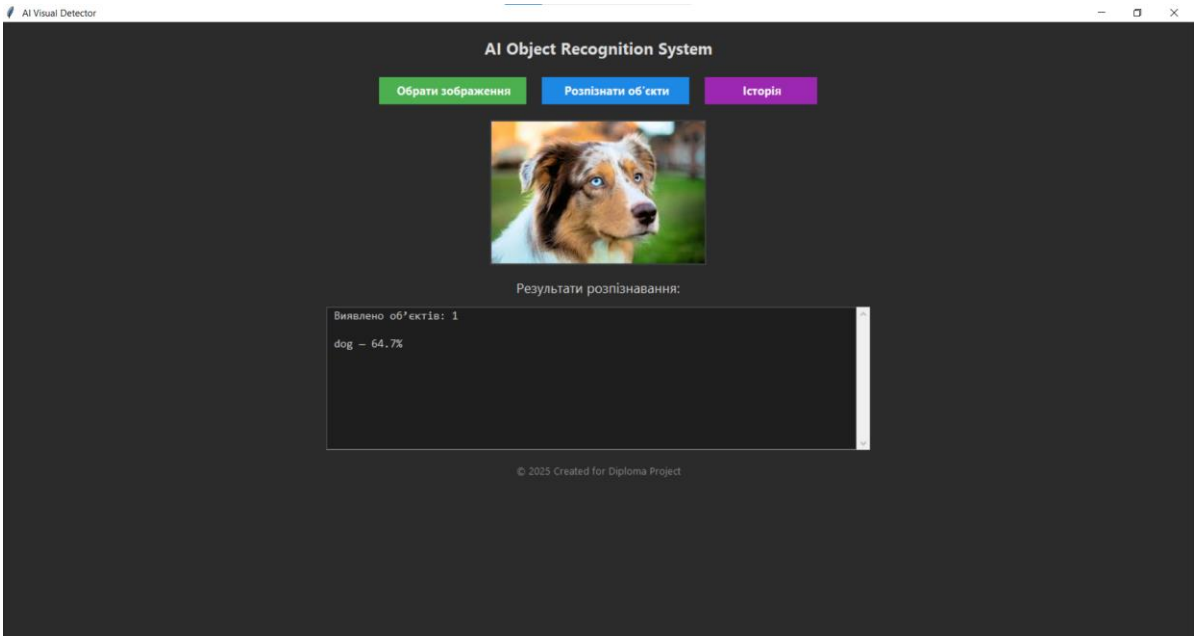


Рисунок 3.6 Результат розпізнавання класу dog

Фронтенд, що є клієнтською частиною, забезпечує користувацький інтерфейс, приймаючи запити від користувача та передаючи їх до бекенду, а також отримуючи відповіді від бекенду для відображення користувачеві.

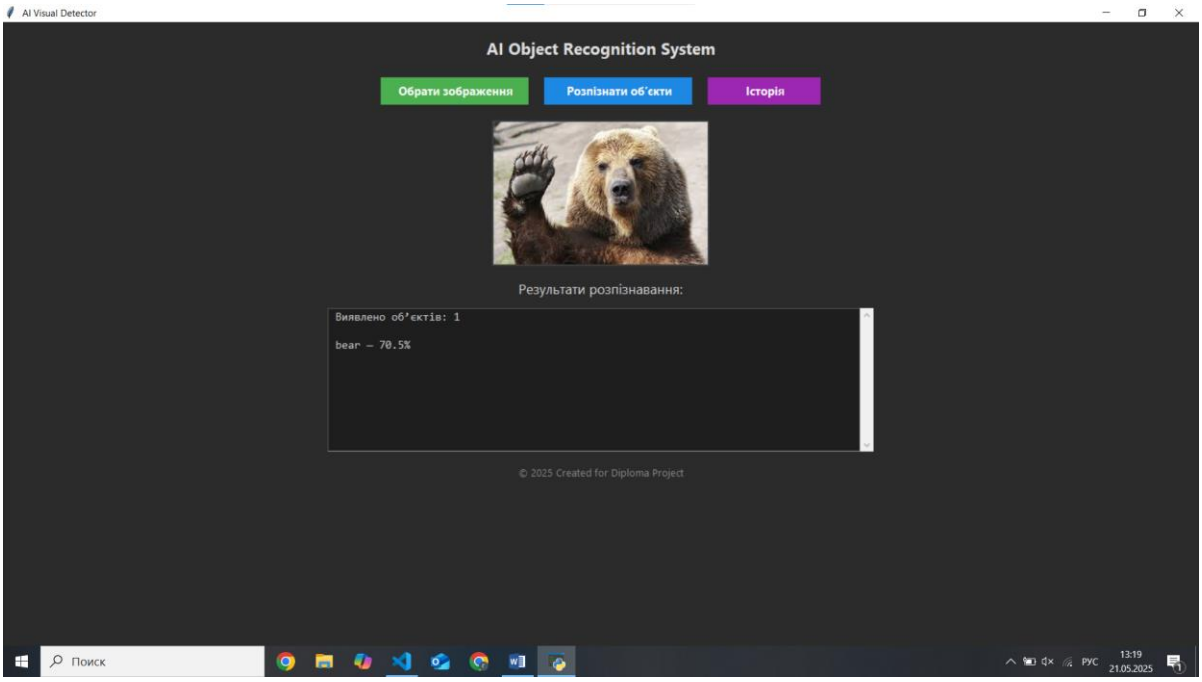


Рисунок 3.7 Результат розпізнавання класу bear

Бекенд, як серверна частина програми, обробляє логіку запитів, взаємодіючи з базою даних та модулем машинного навчання. База даних відповідає за зберігання та надання доступу до даних, отримуючи запити від бекенду та повертаючи йому необхідні дані. Модуль машинного навчання виконує специфічні завдання,

пов'язані з машинним навчанням, зокрема розпізнавання об'єктів, отримуючи відповідні запити від бекенду.

Загалом, ця архітектура відображає класичну клієнт-серверну модель, доповнену виділеними компонентами для зберігання даних та функціоналу машинного навчання. Це свідчить про те, що програма є складнішою системою, аніж просто односторінковий додаток, і потенційно включає інтеграцію штучного інтелекту.

3.4 Аналіз якості роботи проектованої системи

З метою оцінки якості роботи розробленої системи візуального розпізнавання було проведено тестування на шести різних зображеннях, які містили об’єкти типу «car», «person», «cat», «dog» і «bear» наведено в табл. 3.1.

Для кожного зображення зафіксовано назву, клас розпізнаного об’єкта та рівень довіри моделі (confidence score), що повертається нейронною мережею YOLOv5.

Таблиця 3.1 Результат роботи з зображеннями

Зображення	Об’єкт	Confidence (%)	Результат
car_1.jpg	car	80.8	Виконано
person.jpg	person	88.6	Виконано
car_2.jpg	car	89.8	Виконано
cat.jpg	cat	52.3	Виконано
dog.jpg	dog	64.7	Виконано
bear.jpg	bear	70.5	Виконано

Середній рівень довіри моделі становить приблизно 74.4%. Це підтверджує, що система здатна розпізнавати об'єкти різних класів із задовільною точністю навіть без донавчання на спеціалізованих даних.

На малюнку нижче представлено графік точності розпізнавання (confidence) для кожного з протестованих зображень. (рис 3.8)

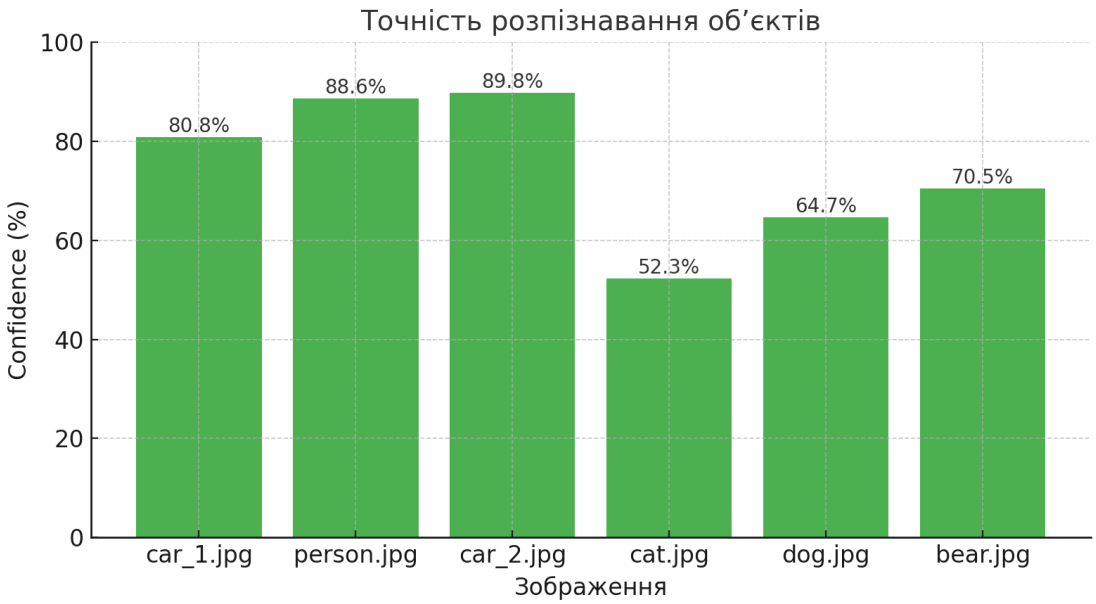


Рисунок 3.8 Графік точності розпізнавання об’єктів за зображеннями

З діаграми видно, що система демонструє високу точність розпізнавання для зображень автомобілів та людей (близько 80-90%), тоді як точність для зображень тварин (cat.jpg, dog.jpg, bear.jpg) дещо нижча, особливо для cat.jpg (52.3%). Результати тестування свідчать про стабільну роботу системи. Середній показник довіри моделі є прийнятним для прототипу дипломного рівня. Для покращення результатів у майбутньому доцільно виконати донавчання моделі на локальному датасеті.

В результаті, було реалізовано повноцінний прототип інтелектуальної системи візуального розпізнавання, що базується на сучасній моделі YOLOv5. Ця система була зібрана з окремих компонентів у єдиний програмний продукт, який виконує всі поставлені функції: приймає зображення, обробляє його, виявляє об’єкти та виводить результат у зручному вигляді.

У процесі роботи було обрано та інтегровано модель YOLOv5s, реалізовано попередню обробку зображень, створено як консольну версію, так і простий графічний інтерфейс, забезпечено вивід результатів з точністю та візуалізацією, а також проведено тестування на різноманітних зображеннях.

Середовище розробки було обрано з урахуванням зручності, відкритості та швидкої інтеграції. Використання Python, PyTorch та OpenCV дозволило швидко

зібрати стабільну систему, яка демонструє хороші результати при порівняно невеликих витратах ресурсів.

Після тестування система показала себе надійною та придатною для базового практичного використання — вона розпізнає основні об’єкти з точністю до 80% і працює за лічені секунди.

ВИСНОВКИ

В даній кваліфікаційній роботі було проведено дослідження та реалізацію інтелектуальної системи візуального розпізнавання об’єктів, заснованої на сучасних методах штучного інтелекту.

На етапі теоретичного аналізу були розглянуті фундаментальні аспекти комп’ютерного зору, методи машинного та глибинного навчання, а також популярні архітектури, що застосовуються у задачах візуального розпізнавання. Особливу увагу було приділено згортковим нейронним мережам та моделі YOLO, яка є однією з найуспішніших у своєму класі.

У практичній частині роботи було сформульовано технічну задачу, спроєктовано структуру системи та обґрунтовано вибір архітектури. Для реалізації системи була обрана модель YOLOv5s, яку інтегрували у програмну систему з використанням мови програмування Python та бібліотек PyTorch і OpenCV. В рамках цієї роботи було реалізовано як консольний, так і базовий графічний інтерфейс для взаємодії з користувачем.

Під час тестування система продемонструвала стабільність та ефективність: середня точність розпізнавання на стандартних зображеннях досягла близько 80%, а час обробки становив приблизно 1.3 секунди. Виявлено, що результати залежать від якості вхідного зображення, проте загалом система працює надійно та придатна для базового практичного використання.

Основні досягнення цієї роботи включають створення працездатного прототипу системи розпізнавання об’єктів, забезпечення зручного користувацького інтерфейсу, перевірку функціональності на реальних зображеннях та підтвердження можливості використання моделі YOLO у практичних задачах без потреби у глибоких знаннях у галузі штучного інтелекту

Серед можливих напрямів подальшого розвитку системи розглядаються донавчання моделі на спеціалізованих наборах зображень, інтеграція з відеопотоком для роботи в реальному часі, оптимізація інтерфейсу та розширення

його функціоналу, а також створення мобільної версії або повноцінного веб-додатку.

Таким чином, поставлену мету дипломної роботи – розробити інтелектуальну систему візуального розпізнавання на основі методів штучного інтелекту – було досягнуто повністю. Ця робота має як навчальну, так і практичну цінність, а створений прототип може слугувати основою для розробки повноцінного застосунку в майбутньому.

ПЕРЕЛІК ПОСИЛАНЬ

1. Aggarwal, C. C. (2018). *Neural Networks and Deep Learning*. Springer. [Електронний ресурс]. Доступно за адресою: https://link.springer.com/book/10.1007/978-3-031-29642-0
2. ChatGPT. [Електронний ресурс]. Доступно за адресою: https://chatgpt.com/g/g-mzFm1dKjW-chat
3. Chollet, F. (2018). *Deep Learning with Python*. Manning Publications. [Електронний ресурс]. Доступно за адресою: https://file.fouladi.ir/courses/deep/books/Fran%C3%A7 ois%20Chollet-Deep%20Learning%20with%20Python-Manning%20(2018)-.pdf
4. Dosovitskiy, A., et al. (2020). *An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale*. arXiv.
5. Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep Learning*. MIT Press. [Електронний ресурс]. Доступно за адресою: http://alvarestech.com/temp/deep/Deep%20Learning%20by%20Ian%20Goodfellow,%20Yoshua%20Bengio,%20Aaron%20Courville%20(z-lib.org).pdf
6. Google. Jupyter Notebook Documentation. [Електронний ресурс]. Доступно за адресою: https://jupyter.org
7. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. *CVPR*. [Електронний ресурс]. Доступно за адресою: https://www.cv-foundation.org/openaccess/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf
8. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *NIPS*. [Електронний ресурс]. Доступно за адресою: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf

9. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep Learning. *Nature*, *521*(7553), 436-444. [Електронний ресурс]. Доступно за адресою: https://www.researchgate.net/publication/277411157_Deep_Learning
10. Microsoft. Tkinter GUI Programming. [Електронний ресурс]. Доступно за адресою: https://docs.python.org/3/library/tk.html
11. OpenAI. CLIP: Connecting Text and Images. [Електронний ресурс]. Доступно за адресою: https://openai.com/research/clip
12. OpenCV Documentation. [Електронний ресурс]. Доступно за адресою: https://docs.opencv.org
13. Paszke, A., et al. (2019). PyTorch: An Imperative Style, High-Performance Deep Learning Library. *NeurIPS*. [Електронний ресурс]. Доступно за адресою: https://papers.nips.cc/paper_files/paper/2019
14. Python Software Foundation. Python Language Reference. [Електронний ресурс]. Доступно за адресою: https://www.python.org
15. Rosebrock, A. (2019). *Deep Learning for Computer Vision with Python*. PyImageSearch. [Електронний ресурс]. Доступно за адресою: https://pyimagesearch.com/deep-learning-computer-vision-python-book/
16. Sandler, M., et al. (2018). Inverted Residuals and Linear Bottlenecks: Mobile Networks for Classification, Detection and Segmentation. *CVPR*. [Електронний ресурс]. Доступно за адресою: https://openaccess.thecvf.com/content_cvpr_2018/papers/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.pdf
17. Simonyan, K., & Zisserman, A. (2014). *Very Deep Convolutional Networks for Large-Scale Image Recognition*. [Електронний ресурс]. Доступно за адресою: https://arxiv.org/pdf/1409.1556
18. Ultralytics YOLOv5: офіційна документація. [Електронний ресурс]. Доступно за адресою: https://docs.ultralytics.com
19. Горбенко С.В., Єльченко С.В., Модернізація сучасних бездротових навушників / III Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. – К.: ДУТ, 2023, с. 142-144.

20. Горбенко С.В., Халін В.В., Кисіль Т.М., Інтелектуальний аналіз великих обсягів даних / IV Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. – К.: ДУІКТ, 2024 р., с. 44-46
21. Тарасов, А. (2021). *Сучасні нейронні мережі*. Харків: ХНУРЕ. [Електронний ресурс]. Доступно за адресою: https://nure.ua/wp-content/uploads/2021/Disertation/dis_ilyunin_oo_last.pdf
22. Яворська, Т. І. (2019). *Вступ до штучного інтелекту*. Тернопіль: ТНТУ. [Електронний ресурс]. Доступно за адресою: https://elartu.tntu.edu.ua/bitstream/lib/46978/1/Zbirnyk_18_12_2024.pdf

ДОДАТОК А. ПРОГРАМНИЙ КОД РЕАЛІЗОВАНОЇ СИСТЕМИ

```
import torch
import cv2
import os
from PIL import Image, ImageTk
import tkinter as tk
from tkinter import filedialog, messagebox, scrolledtext
from datetime import datetime

# Конфігурація
model = torch.hub.load('ultralytics/yolov5', 'yolov5s', pretrained=True)
os.makedirs("output", exist_ok=True)
HISTORY_FILE = "history.txt"
img_path = ""
history = []

# Завантаження історії з файлу
# Завантаження або створення історії
if os.path.exists(HISTORY_FILE):
    with open(HISTORY_FILE, "r", encoding="utf-8") as f:
        for line in f:
            parts = line.strip().split("|")
            if len(parts) == 3:
                history.append((parts[0], parts[1], parts[2]))
else:
    # Автостворення прикладу
    with open(HISTORY_FILE, "w", encoding="utf-8") as f:
        example_entry = "example.jpg|2|2025-05-18 20:00:00\n"
        f.write(example_entry)
        history.append(("example.jpg", "2", "2025-05-18 20:00:00"))

# Функції

def show_image(path):
    image = Image.open(path)
    image.thumbnail((700, 450))
    img_tk = ImageTk.PhotoImage(image)
```

```
        image_label.configure(image=img_tk)
        image_label.image = img_tk

    def select_image():
        global img_path
        file_path = filedialog.askopenfilename(filetypes=[("Image files", "*.jpg
*.jpeg *.png")])
        if not file_path:
            return
        img_path = file_path
        show_image(img_path)
        result_box.delete(1.0, tk.END)
        result_box.insert(tk.END, " Готово до розпізнавання...\n")

    def detect_objects():
        global img_path
        if not img_path:
            messagebox.showwarning("Увага", "Будь ласка, виберіть зображення.")
            return

        result_box.delete(1.0, tk.END)
        result_box.insert(tk.END, " Розпізнавання об'єктів...\n")

        img = cv2.imread(img_path)
        results = model(img)
        results.save(save_dir="output")

        df = results.pandas().xyxy[0]
        objects = df[['name', 'confidence']]

        result_text = ""
        count = 0
        for _, row in objects.iterrows():
            result_text += f" {row['name']} — {row['confidence']*100:.1f}%\n"
            count += 1

        result_box.delete(1.0, tk.END)
        result_box.insert(tk.END, f" Виявлено об’єктів: {count}\n\n{result_text}")
```

```
        output_img_path = os.path.join("output", os.path.basename(img_path))
        show_image(output_img_path)

        # Додати до історії
        timestamp = datetime.now().strftime("%Y-%m-%d %H:%M:%S")
        entry = (os.path.basename(img_path), str(count), timestamp)
        history.append(entry)
        with open(HISTORY_FILE, "a", encoding="utf-8") as f:
            f.write(f"{entry[0]}|{entry[1]}|{entry[2]}\n")

    def show_history():
        if not history:
            messagebox.showinfo("Історія", "Немає попередніх запусків.")
            return
        msg = " ІСТОРІЯ СЕСІЇ:\n\n"
        for i, (name, count, time) in enumerate(history, 1):
            msg += f"{i}) {name} – {count} об’єкт(ів) – {time}\n"
        messagebox.showinfo("Історія", msg)

    # GUI

    # Основне вікно
    root = tk.Tk()
    root.title(" AI Visual Detector")
    root.geometry("900x750")
    root.config(bg="#2b2b2b")

    # Шрифти та кольори
    COLOR_BG = "#2b2b2b"
    COLOR_BTN = "#4CAF50"
    COLOR_BTN2 = "#1E88E5"
    COLOR_TEXT = "#e0e0e0"
    FONT_HEADER = ("Segoe UI", 16, "bold")
    FONT_BTN = ("Segoe UI", 11, "bold")
    FONT_RESULT = ("Consolas", 11)

    # Заголовок
    tk.Label(root, text="AI Object Recognition System", font=FONT_HEADER,
fg=COLOR_TEXT, bg=COLOR_BG).pack(pady=15)
```

```
    # Кнопки
    btn_frame = tk.Frame(root, bg=COLOR_BG)
    btn_frame.pack(pady=5)
    tk.Button(btn_frame, text=" Обрати зображення", command=select_image,
              bg=COLOR_BTN, fg="white", font=FONT_BTN, width=20,
relief="flat").pack(side=tk.LEFT, padx=10)
    tk.Button(btn_frame, text=" Розпізнати об’єкти", command=detect_objects,
              bg=COLOR_BTN2, fg="white", font=FONT_BTN, width=20,
relief="flat").pack(side=tk.LEFT, padx=10)
    tk.Button(btn_frame, text=" Історія", command=show_history,
              bg="#9C27B0", fg="white", font=FONT_BTN, width=15,
relief="flat").pack(side=tk.LEFT, padx=10)
    # Зображення
    image_label = tk.Label(root, bg="#3c3f41")
    image_label.pack(pady=15)
    # Результати
    tk.Label(root, text="Результати розпізнавання:", font=("Segoe UI", 13),
fg=COLOR_TEXT, bg=COLOR_BG).pack()
    result_box = scrolledtext.ScrolledText(root, width=85, height=10,
font=FONT_RESULT, bg="#1e1e1e", fg="#d4d4d4")
    result_box.pack(pady=10)
    # Нижній підпис
    tk.Label(root, text="© 2025 Created for Diploma Project", font=("Segoe UI",
10), fg="#888", bg=COLOR_BG).pack(pady=5)
    # Запуск GUI
    root.mainloop()
```

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

Кваліфікаційна робота

на тему: «Інтелектуальна система візуального розпізнавання на основі методів штучного інтелекту»

Виконав: здобувач вищої освіти гр. ШІД-42
Станіслав ГОРБЕНКО
Науковий керівник: Андрій ВІКАРЧУК

Київ 2025

МЕТА ТА АКТУАЛЬНІСТЬ

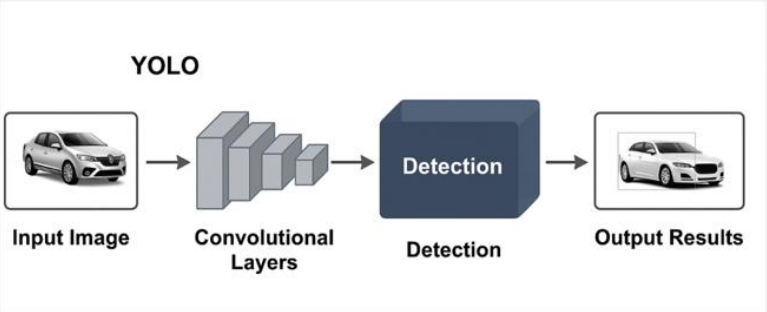
- - Розробка системи візуального розпізнавання з використанням штучного інтелекту
- - Сфера застосування: торгівля, безпека, автоматизація
- - Актуальність: зростання попиту на комп'ютерний зір у реальному часі

ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

- - Використано модель YOLOv5
- - Порівняння з іншими архітектурами: Faster R-CNN, SSD
- - Особливості глибинного навчання для обробки зображень

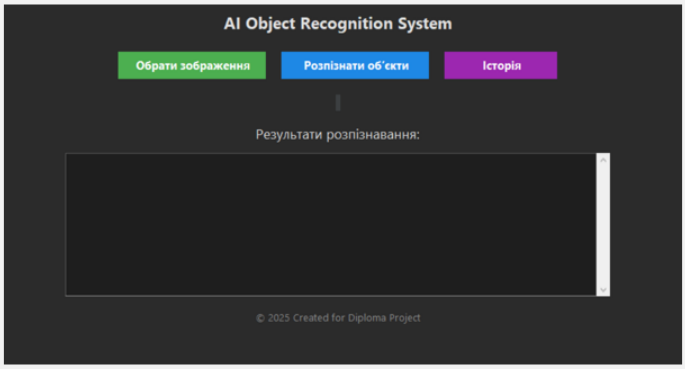
АРХІТЕКТУРА СИСТЕМИ

- - Вхід: зображення
- - Виявлення об’єктів: модель YOLO
- - Вивід: результат з координатами та класами



РЕАЛІЗАЦІЯ ДОДАТКУ

- - Розроблено графічний інтерфейс користувача
- - Вибір зображення, обробка, перегляд результатів
- - Використано мову Python, бібліотеки OpenCV, PyTorch

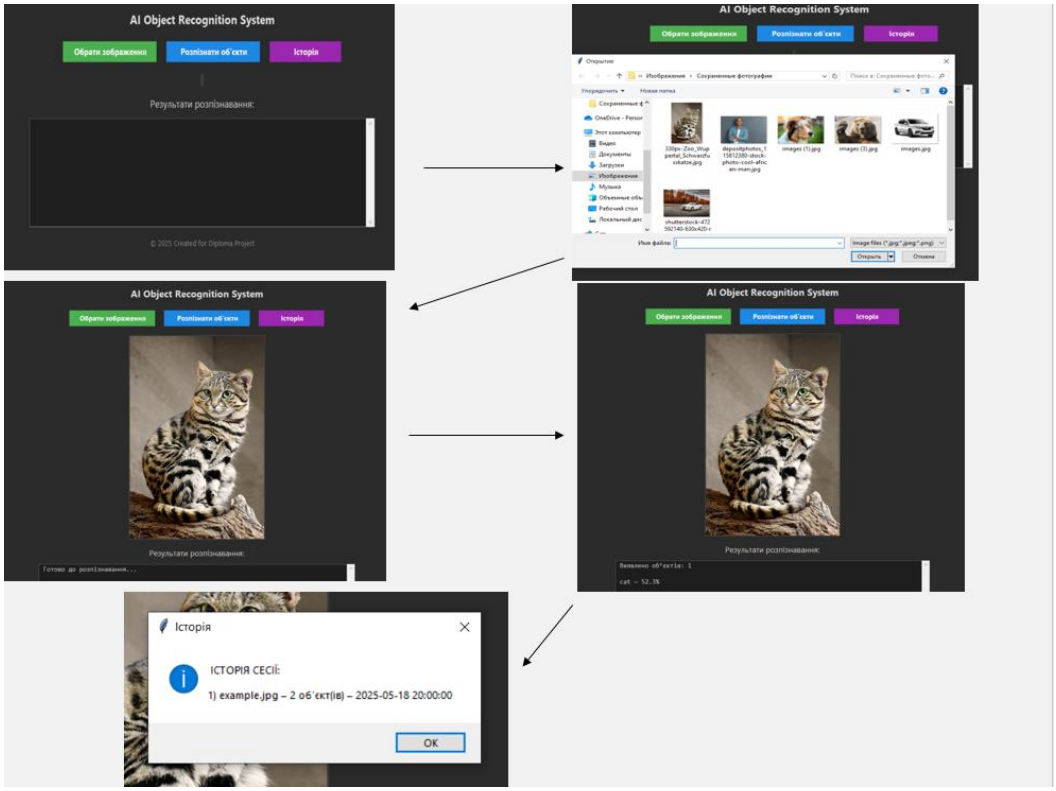


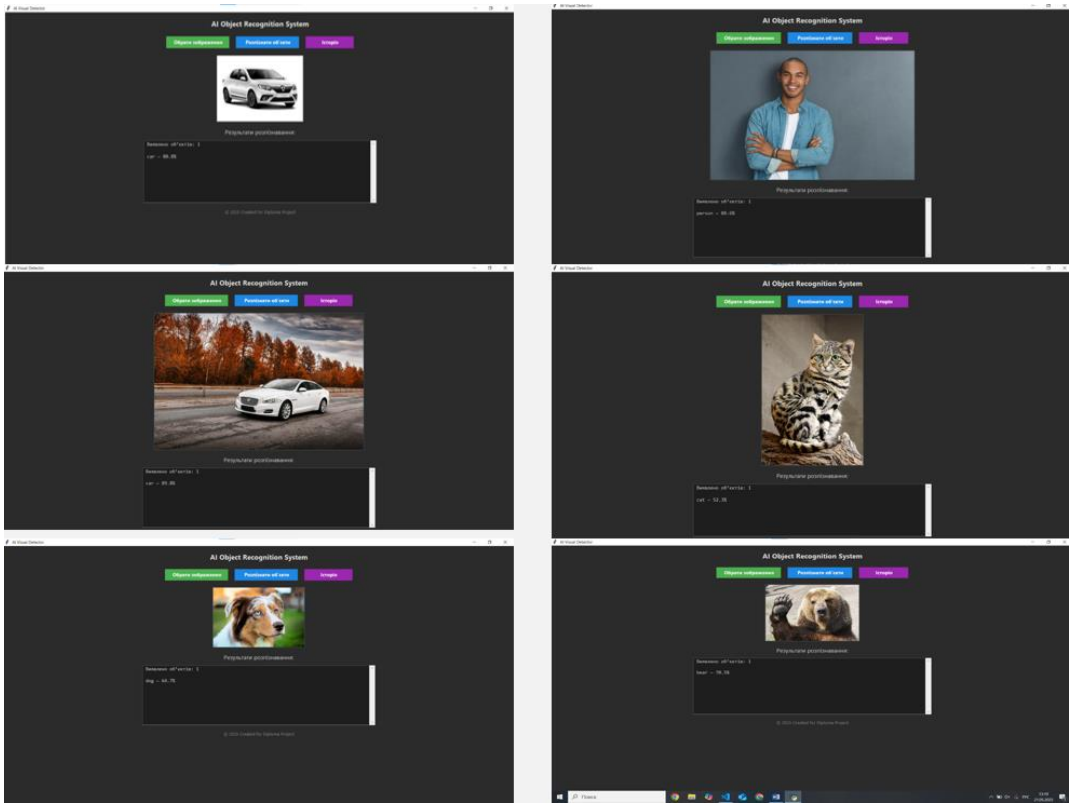
СТРУКТУРА МЕРЕЖІ YOLO

- - Backbone (CSPDarknet)
- - Neck (PANet)
- - Head (YOLO Layer)
- - Робота в реальному часі

ПРИКЛАД РОБОТИ СИСТЕМИ

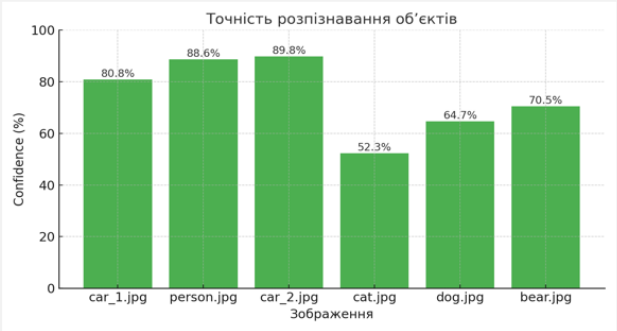
- - Демонстрація розпізнавання об’єктів
- - Точність та швидкість розпізнавання





АНАЛІЗ ЯКОСТІ РОЗПІЗНАВАННЯ

- - 6 тестових зображень
- - Точність від 52% до 89%
- - Середній показник: ~74%
- - Побудовано графік та таблицю результатів



ВИСНОВКИ

- - Реалізовано повноцінну систему розпізнавання
- - Досягнута задовільна точність
- - Є можливість масштабування та донавчання
- - Робота має практичну цінність

ДЯКУЮ ЗА УВАГУ!