МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«ЗАСТОСУВАННЯ НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ ДЛЯ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У СИСТЕМАХ БЕЗПЕКИ»**

Виконав: студент 2 курсу, групи 8.1264-з
спеціальності 126 Інформаційні системи та технології
(шифр і назва спеціальності)
освітньої програми Інформаційні системи та штучний інтелект
(назва освітньої програми)
Г.Ю. Колісник
(ініціали та прізвище)

Керівник доцент кафедри комп'ютерних наук, к.т.н., доцент Матвіїшина Н.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент декан математичного факультету, д.т.н., професор Гоменюк С.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 126 Інформаційні системи та технології
(шифр і назва)

Освітня програма Інформаційні системи та штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М..
(підпис)

" ____ _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Колісннику Геннадію Юрійовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Застосування нейромережевих моделей для розпізнавання об'єктів у системах безпеки

керівник роботи Матвіїшина Надія Вікторівна, к.т.н., доцент
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » 10 2025 року № 1457-с

2. Строк подання студентом роботи 03.12.2025 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Постановка задачі.
2. Основні теоретичні відомості.
3. Аналіз методів та засобів розпізнавання об'єктів у системах безпеки.
4. Теоретичні основи нейромережевих моделей розпізнавання об'єктів.
5. Проектування та практична реалізація модуля розпізнавання об'єктів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 21.10.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	21.10.2025	
2.	Збір вихідних даних.	27.10.2025	
3.	Обробка методичних та теоретичних	03.11.2025	
	джерел.		
4.	Розробка першого та другого розділу.	10.11.2025	
5.	Розробка третього розділу.	17.11.2025	
6.	Оформлення та нормоконтроль	26.11.2025	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	10.12.2025	

Студент _____ Г.Ю. Колісник
(підпис) (ініціали та прізвище)

Керівник роботи _____ Н.В. Матвіїшина
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Застосування нейромережевих моделей для розпізнавання об’єктів у системах безпеки»: 53 с., 23 рис., 2 табл., 19 джерел, 3 додатка.

ЗОБРАЖЕННЯ, КОМП’ЮТЕРНИЙ ЗІР, НЕЙРОННА МЕРЕЖА, РОЗПІЗНАВАННЯ, ШТУЧНИЙ ІНТЕЛЕКТ, CCTV, CNN, DVR, IMAGEAI, IMAGE CLASSIFICATION, OBJECT RECOGNITION, OPENCV, PYTHON, YOLO.

Об’єкт дослідження – процеси виявлення, локалізації та класифікації об’єктів на зображеннях і відеокадрах.

Мета роботи: підвищення ефективності функціонування систем відеоспостереження шляхом розробки алгоритмів для автоматичного розпізнавання об’єктів на основі сучасних архітектур згорткових нейронних мереж, оптимізованих для роботи в режимі реального часу.

Методи дослідження – аналіз, порівняння, моделювання, експеримент.

У магістерській кваліфікаційній роботі досліджувалися методи виявлення, локалізації та класифікації об’єктів на зображеннях і відеокадрах. Було проведено порівняльний аналіз різних алгоритмів розпізнавання об’єктів. Запропоновано архітектуру програмного засобу для автоматичного розпізнавання об’єктів, реалізованого мовою програмування Python з використанням бібліотек ImageAI, OpenCV та попередньо натренованої на наборі даних COCO архітектури YOLO.

Розроблений прототип може слугувати основою для створення модулів автоматичного розпізнавання об’єктів у реальних системах безпеки.

SUMMARY

Master’s Qualifying Theses «Application of Neural Network Models for Object Recognition in Security Systems»: 53 pages, 23 figures, 2 tables, 19 references, 3 supplements.

IMAGE, COMPUTER VISION, NEURAL NETWORK, RECOGNITION, ARTIFICIAL INTELLIGENCE, CCTV, CNN, DVR, IMAGEAI, IMAGE CLASSIFICATION, OBJECT RECOGNITION, OPENCV, PYTHON, YOLO.

Object of the study – processes of detection, localization, and classification of objects in images and video frames.

Aim of the study: improving the efficiency of video surveillance systems by developing algorithms for automatic object recognition based on modern convolutional neural network architectures optimized for real-time operation.

Methods of research – analysis, comparison, modeling, experimentation.

The master’s thesis investigated methods for detecting, localizing, and classifying objects in images and video frames. A comparative analysis of various object recognition algorithms was performed. The architecture of a software tool for automatic object recognition was proposed, implemented in the Python programming language using ImageAI and OpenCV libraries and the YOLO architecture pre-trained on the COCO dataset.

The developed prototype can serve as a basis for creating modules for automatic object recognition in real security systems.

ЗМІСТ

Завдання на кваліфікаційну роботу……..2
Реферат 4
Summary 5
Вступ..... 7
1 Аналітичний огляд методів та засобів розпізнавання об’єктів у системах безпеки 9
1.1 Еволюція та сучасний стан інтелектуальних систем відеоспостереження 9
1.1.1 Від аналогових систем (CCTV) до цифрових рішень 9
1.1.2 Класичні методи комп’ютерного зору в системах безпеки..... 10
1.1.3 Революція глибокого навчання (Deep Learning)..... 11
1.2 Порівняльний аналіз провідних комерційних рішень відеоаналітики 12
1.2.1 Hikvision..... 13
1.2.2 Dahua Technology 14
1.2.3 Axis Communications..... 14
1.2.4 Хмарні системи відеоаналітики (Verkada, Arcules)..... 15
1.2.5 Порівняння параметрів комерційних рішень відеоаналітики . 15
2 Теоретичні основи нейромережевих моделей розпізнавання об’єктів ... 17
2.1 Теоретичні основи згорткових нейронних мереж (CNN)..... 17
2.2 Класифікація та порівняльний аналіз архітектур нейронних мереж..... 22
2.2.1 Двостадійні детектори (Two-Stage Detectors) 22
2.2.2 Одностадійні детектори (One-Stage Detectors) 22
2.2.3 Трансформери (Vision Transformers) 24
2.2.4 Порівняння параметрів архітектур нейронних мереж 24

3 Проектування та практична реалізація модуля розпізнавання об’єктів для систем безпеки 26
3.1 Теоретичні основи розпізнавання об’єктів (Object Recognition) 26
3.2 Загальна концепція та архітектура програмної реалізації 27
3.3 Вибір середовища розробки програмного забезпечення 29
3.4 Програмна реалізація модуля розпізнавання об’єктів 32
3.4.1 Клас ObjectDetection для виявлення та розпізнавання об’єктів на зображеннях 33
3.4.2 Клас VideoObjectDetection для виявлення та розпізнавання об’єктів у відеофайлах і відеопотоках з відеокамер 42
Висновки 46
Перелік посилань 48
Додаток А Модулі виявлення та розпізнавання об’єктів на зображеннях 50
Додаток Б Модуль виявлення та розпізнавання об’єктів у відеофайлах 51
Додаток В Модуль виявлення та розпізнавання об’єктів у відеопотоках 52

ВСТУП

Сучасний етап розвитку суспільства характеризується стрімкою цифровізацією всіх сфер діяльності людини, зростанням ролі інформаційних технологій, автоматизації процесів та широким впровадженням інтелектуальних систем у промисловості, транспорті, медицині, сфері обслуговування та, зокрема, у системах безпеки. В умовах глобалізації, урбанізації та підвищення рівня техногенних і соціальних загроз питання забезпечення безпеки набувають особливої актуальності.

Зростання кількості правопорушень, терористичних актів, аварій, кіберзлочинів, надзвичайних ситуацій та військових загроз обумовлює необхідність створення ефективних, надійних і оперативних систем моніторингу, контролю та реагування. Традиційні системи безпеки, що базуються виключно на людському факторі, дедалі частіше виявляються недостатньо ефективними через обмеженість людських можливостей щодо обробки великих обсягів відеоданих, втомлюваність операторів та суб’єктивність оцінки ситуації. Відеоспостереження, яке традиційно розглядалося як пасивний інструмент фіксації подій для постфактум-аналізу, сьогодні трансформується у ключовий елемент проактивних систем безпеки.

У зв’язку з цим важливого значення набуває використання інтелектуальних систем відеоспостереження, які ґрунтуються на методах комп’ютерного зору та штучного інтелекту. Одним із ключових елементів таких систем є нейромережеві моделі розпізнавання об’єктів, що дозволяють автоматично виявляти, класифікувати та відстежувати об’єкти в режимі реального часу.

Особливу роль у цьому процесі відіграють згорткові нейронні мережі (Convolutional Neural Networks, CNN), які довели свою високу ефективність у задачах комп’ютерного зору, зокрема в задачах детекції об’єктів на

зображеннях і відеопослідовностях. Сучасні моделі родини YOLO (You Only Look Once), Faster R-CNN, SSD, а також трансформерні архітектури (Vision Transformer, RT-DETR) забезпечують високу точність розпізнавання при збереженні швидкої обробки інформації.

Однак, незважаючи на значні успіхи, імплементація нейромережевих алгоритмів у реальні системи безпеки пов’язана з рядом невирішених науково-технічних проблем. До них належать високі вимоги до обчислювальних ресурсів, необхідність роботи в режимі реального часу (Real-Time) з мінімальною затримкою, залежність точності розпізнавання від умов освітлення та погодних факторів. Таким чином, дослідження методів та засобів застосування нейромережевих моделей для розпізнавання об’єктів у системах безпеки є актуальним, науково й практично значущим завданням.

Окрім цього, в умовах воєнного стану та підвищених безпекових ризиків для України, задачі автоматизованого відеоспостереження, ідентифікації осіб, виявлення підозрілих об’єктів і транспортних засобів набувають стратегічного значення.

1 АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ ТА ЗАСОБІВ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ У СИСТЕМАХ БЕЗПЕКИ

1.1 Еволюція та сучасний стан інтелектуальних систем відеоспостереження

Системи відеоспостереження пройшли довгий шлях розвитку, який можна умовно поділити на кілька етапів, кожен з яких характеризується якісним стрибком у технологіях обробки та передачі інформації. Розуміння цієї еволюції є ключовим для усвідомлення місця нейромережевих технологій у сучасному ландшафті безпеки.

1.1.1 Від аналогових систем (CCTV) до цифрових рішень

Перше покоління систем відеоспостереження CCTV (Closed Circuit Television) з'явилося ще в середині XX століття. Це були повністю аналогові системи, де сигнал з камери передавався коаксіальним кабелем безпосередньо на монітор та записувався на відеомагнітофон (VCR).

Характеристики: низька роздільна здатність, необхідність постійної заміни касет, відсутність можливості швидкого пошуку в архіві.

Інтелектуальна складова відсутня. Функція "детекції" реалізовувалася виключно людиною-оператором, який повинен був невідривно дивитися на стіну моніторів.

Недоліки: людський фактор був найслабшою ланкою. Втомлюваність призводила до того, що більшість інцидентів виявлялися лише постфактум, під час перегляду архіву після скоєння злочину.

Друге покоління ознаменувалося переходом до цифрового запису DVR (Digital Video Recorder). Сигнал все ще залишався аналоговим на вході, але

оцифровувався, стискався та зберігався на жорстких дисках. Це дозволило реалізувати перші примітивні алгоритми відеоаналітики VMD (Video Motion Detection).

Механізм VMD – порівняння кадрів (Frame Differencing). Якщо значення яскравості пікселя у поточному кадрі відрізняється від попереднього на певну порогову величину, система сигналізує про рух.

Проблема VMD – алгоритм не розуміє змісту сцени. Рух хмар, коливання дерев від вітру, зміна освітлення, сніг чи дощ, перемикання камери в нічний режим – все це викликало лавину хибних спрацювань. Це призвело до явища "тривожної втоми" (alarm fatigue), коли оператори просто відключали детектори руху через їхню неефективність.

Третє покоління – це повністю мережеві системи (IP-Surveillance). Камера стала самостійним комп’ютерним пристроєм, що оцифровує, стискає та аналізує відео. Це відкрило шлях для впровадження більш складної аналітики на базі комп’ютерного зору CV (Computer Vision).

1.1.2 Класичні методи комп’ютерного зору в системах безпеки

До ери глибокого навчання (до 2012 року) домінували методи, що базувалися на ручному конструюванні ознак (hand-crafted features). Розробники намагалися математично описати, як виглядає "людина" або "автомобіль", використовуючи градієнти, текстури та форми.

Виявлення переднього плану (Background Subtraction).

Найпопулярнішим методом було моделювання фону за допомогою суміші Гауссіан – GMM (Gaussian Mixture Models). Для кожного пікселя будувалася статистична модель його поведінки у часі. Пікселі, що не вписувалися в розподіл, вважалися переднім планом (об’єктом).

Обмеження – метод погано працював у щільному натовпі та при різких змінах освітлення (наприклад, коли сонце виходить з-за хмар).

Дескриптори ознак (HOG, SIFT, SURF, Haar Cascades).

HOG (Histogram of Oriented Gradients) – метод, запропонований Dalal та Triggs у 2005 році для детекції пішоходів. Зображення розбивається на комірки, в кожній з яких обчислюється гістограма напрямків градієнтів. Це дозволяє описати форму об’єкта (силует людини) незалежно від кольору одягу.

Haar Cascades використовувалися для детекції обличь (метод Viola-Jones). Це набір простих прямокутних фільтрів, які швидко обчислюються.

Класифікація (SVM, k-NN, AdaBoost).

Ознаки, витягнуті за допомогою HOG або Haar, подавалися на вхід класифікатора, такого як метод опорних векторів SVM (Support Vector Machine). SVM будував гіперплощину, що розділяла вектори ознак "людини" та "не людини".

Поява дескрипторів ознак, таких як HOG (Histogram of Oriented Gradients), у поєднанні з класифікаторами SVM (Support Vector Machine) вже дозволяла розпізнавати людей. Хоча ці методи були значним кроком вперед порівняно з VMD, вони все ще мали суттєві обмеження та були чутливими до змін освітлення та перекриття об’єктів. Вони вимагали ретельного налаштування параметрів під кожну конкретну сцену камери. Зміна ракурсу або поява нових об’єктів (наприклад, людини на велосипеді замість пішохода) часто призводила до збоїв.

1.1.3 Революція глибокого навчання (Deep Learning)

Переломним моментом у розвитку систем безпеки став прорив у галузі розпізнавання об’єктів після появи у 2012 році згорткових нейронних мереж (CNN), зокрема моделі AlexNet, яка суттєво перевершила традиційні підходи. На відміну від класичних методів, CNN не потребують ручного конструювання ознак. Мережа самостійно навчається виділяти найбільш релевантні ознаки з великого масиву даних.

У контексті безпеки це забезпечило:

– семантичне розуміння. Система розрізняє класи об’єктів (людина, кішка, собака, автомобіль), а не просто фіксує "рухому пляму";
– інваріантність. Висока стійкість до змін освітлення, деформацій об’єкта, часткового перекриття об’єктів та різноманітності ракурсів;
– мультизадачність. Одна й та сама архітектура може використовуватися для детекції обличь, номерних знаків, пожеж, зброї тощо.

Сьогодні нейромережева аналітика є стандартом де-факто для професійних систем безпеки. Ринок вимагає рішень, що не просто записують відео, а перетворюють його на структуровані метадані (хто, де, коли, у чому вдягнений), які можна аналізувати та використовувати для прийняття рішень.

1.2 Порівняльний аналіз провідних комерційних рішень відеоаналітики

Інтелектуальні системи відеоспостереження IVSS (Intelligent Video Surveillance Systems) є ключовим компонентом сучасних комплексних систем безпеки. Вони поєднують апаратні засоби відеофіксації з програмними модулями відеоаналітики, що базуються на методах штучного інтелекту, нейромереж та комп’ютерного зору.

Основними функціями IVSS є:

– автоматичне виявлення об’єктів у відеопотоці;
– ідентифікація осіб;
– підрахунок людей;
– класифікація транспортних засобів;
– розпізнавання номерних знаків;
– аналіз поведінкових аномалій;
– виявлення залишених предметів;
– контроль периметра;

– інтеграція з іншими компонентами систем безпеки.

На відміну від традиційних систем відеоспостереження, де аналіз відео здійснюється оператором, інтелектуальні системи виконують автоматизований аналіз у режимі реального часу, суттєво знижуючи навантаження на персонал та підвищуючи ймовірність своєчасного реагування на загрози.

Сучасні системи відеоспостереження мають такі характеристики:

– висока роздільна здатність камер (HD, Full HD, 4K, 8K та більше), що дозволяє отримувати детальне зображення об’єктів;
– широкий динамічний діапазон (WDR) для роботи за складних умов освітлення;
– нічне бачення (ІЧ-підсвітка, EXIR);
– стиснення відео сучасними кодеками архівування (H.265+, H.266);
– мережева аналітика – камери здатні самостійно виконувати базові функції аналітичної обробки інформації.

На ринку систем безпеки домінують кілька глобальних виробників, які активно впроваджують технології штучного інтелекту. До них належать:

– Hikvision;
– Dahua Technology;
– Axis Communications;
– Bosch Security Systems;
– Hanwha Vision (founded as Samsung Techwin);
– сервіси на основі хмарних технологій (Verkada, Arcules).

Нижче наведено ключові функціональні характеристики кількох з цих рішень.

1.2.1 Hikvision

Компанія Hikvision – один із найбільших у світі виробників систем відеоспостереження. Основні можливості:

– Deep Learning серія камер (DeepInView);

– детекція обличчя, номерних знаків, типів транспортних засобів;
– виявлення шоломів, касок, форм;
– аналітика черг і потоків людей;
– тепловізійні системи для виявлення проникнення та температурних аномалій.

Hikvision використовує власний фреймворк HikDeepNet, оптимізований під edge-обробку – обробка даних відбувається ближче до джерела (камери).

1.2.2 Dahua Technology

Dahua (大华) – це китайська назва, яка в контексті компанії означає щось на зразок «Велика Батьківщина» або «Велика Краса». Компанія Dahua Technology активно розвиває власну платформу WizSense/WizMind, що використовує глибокі згорткові мережі для фільтрації помилкових спрацювань та підвищення точності детекції. Основні можливості:

– розпізнавання обличь, емоцій, віку, статі;
– трекінг багатьох об’єктів;
– ANPR (розпізнавання номерних знаків) з високою точністю;
– класифікація транспортних засобів;
– інтелектуальні тепловізійні рішення.

Особливість – активне застосування технологій граничних обчислень (edge-computing) та апаратних нейронних процесорів.

1.2.3 Axis Communications

Компанія Axis Communications робить ставку на відкриту архітектуру:

– підтримка сторонніх алгоритмів через відкриту платформу додатків ACAP (AXIS Camera Application Platform);
– висока надійність і кіберстійкість;

– фокус на професійних системах для критичної інфраструктури.

1.2.4 Хмарні системи відеоаналітики (Verkada, Arcules)

Сучасний тренд – перехід на хмарні (cloud) системи. Хмарні системи відеоаналітики значно спрощують роботу користувача та відкривають широкі можливості:

– централізований доступ;

– масштабованість;

– високий рівень автоматизації.

При всіх позитивних сторонах хмарних систем є у них і деякі недоліки, головний з них – залежність від інтернету. Іноді зв’язок може перериватися. Це може призвести до втрати важливої інформації.

1.2.5 Порівняння параметрів комерційних рішень відеоаналітики

У таблиці 1.1 наведено порівняльний аналіз параметрів провідних комерційних рішень відеоаналітики.

Таблиця 1.1 – Порівняння параметрів комерційних рішень відеоаналітики

Параметр	Hikvision	Dahua	Axis	Verkada
Нейромережеві моделі	Так	Так	Частково	Так
Хмарна підтримка	Частково	Частково	Частково	Повна
Реальний час	Так	Так	Так	Так
Відкритість системи	Низька	Низька	Висока	Середня
Оптимізація під AI	Висока	Висока	Середня	Висока

Компанії Hikvision та Dahua Technology є світовими лідерами у виробництві інтелектуальних систем відеоспостереження. Їхні продукти використовують вбудовані нейромережеві модулі відеоаналітики, але вони залишаються закритими. В умовах повномасштабної війни та трансформації безпекового ландшафту України у 2025 році актуальна необхідність заміщення обладнання та програмного забезпечення китайського виробництва (Hikvision, Dahua), які становлять загрозу національній безпеці через потенційну передачу даних на сервери, контрольовані виробником.

При всіх позитивних сторонах хмарних (cloud) сервісів є у них і важливі недоліки – висока вартість та залежність від каналів передачі даних, яка посилюється в даний час постійними відключеннями електроенергії.

Отримані результати аналізу параметрів комерційних рішень відеоаналітики дозволяють обґрунтувати доцільність розробки власних нейромережевих систем розпізнавання об’єктів, адаптованих до конкретних умов застосування.

2 ТЕОРЕТИЧНІ ОСНОВИ НЕЙРОМЕРЕЖЕВИХ МОДЕЛЕЙ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ

У даному розділі розглядаються математичні основи та алгоритмічні принципи, що лежать в основі функціонування обраних для дослідження архітектур нейронних мереж. Глибоке розуміння математичного апарату є необхідним для обґрунтування вибору досліджуваних архітектур під специфічні задачі систем безпеки.

2.1 Теоретичні основи згорткових нейронних мереж (CNN)

Для глибокого розуміння роботи сучасних алгоритмів розпізнавання об'єктів необхідно розглянути математичні та архітектурні принципи функціонування згорткових нейронних мереж.

Згорткові нейронні мережі, або Convolutional Neural Networks (далі CNN), є різновидом нейронних мереж для обробки зображень. CNN стали фактично стандартом у застосунках комп'ютерного зору, включаючи розпізнавання об'єктів та класифікацію зображень. Архітектура цих систем розроблена спеціально для обробки двовимірних структур даних, таких як фотографії або відеокадри. Ці системи дозволяють автоматично визначати релевантні ознаки зображення, усуваючи таким чином необхідність ручного програмування [1].

Архітектура CNN складається з послідовності функціональних блоків, кожен з яких виконує певну роль:

– згортковий шар (Convolution Layer);
– функція активації (ReLU);
– шар підвибірки (Pooling Layer);

– повнозв’язний шар (Fully Connected Layer);

– функція Softmax для класифікації.

На рисунку 2.1 наведена типова архітектура CNN, для класифікації зображень, яка складається зі згорткових шарів, шарів підвибірки та повнозв’язних шарів класичних штучних нейронів.

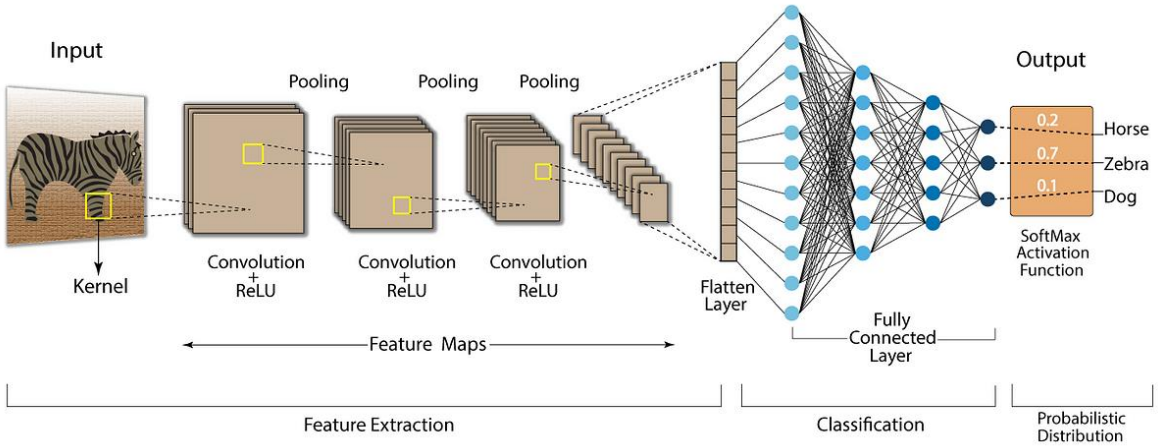


Рисунок 2.1 – Архітектура CNN для класифікації зображень

Згортка (Convolution) – це операція, що дозволяє виділити локальні особливості зображення. Згортковий шар (Convolutional Layer) – це ключовий шар мережі, який виконує операцію згортки (див. рис. 2.2), що передбачає накладення фільтра (ядра) на ділянки зображення з метою виділення характерних ознак (країв, контурів, текстур) [1].

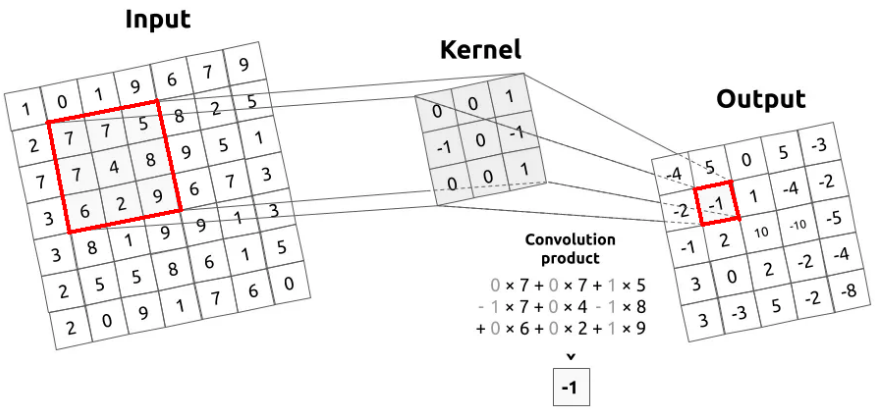


Рисунок 2.2 – Приклад операції згортки

Операція згортки відрізняє CNN від інших типів нейронних мереж, є потужний інструмент, який дозволяє мережі ефективно виявляти корисні ознаки на зображеннях незалежно від їхнього положення.

Згортка в CNN – це процес застосування фільтра (ядра згортки) до зображення. Фільтр – це невелика матриця, яку ми перемножуємо з пікселями зображення. Він рухається по зображенню, розглядаючи невеликі ділянки зображення по черзі, що дозволяє аналізувати локальні ознаки.

Ключові параметри згорткового шару:

– розмір ядра (Kernel Size): зазвичай 3x3, 5x5 або 7x7, менші ядра дозволяють виділяти дрібні деталі та зменшують кількість параметрів;
– крок (Stride): величина зсуву ядра при кожній ітерації, Stride > 1 призводить до зменшення розмірності вихідної карти ознак;
– доповнення (Padding): додавання нульових пікселів по краях зображення (Zero-padding) для збереження просторової розмірності після згортки.

Після лінійної операції згортки до результату застосовується нелінійна функція активації, яка дозволяє мережі моделювати складні залежності. Найпоширенішою є функція ReLU. Без неї нейронна мережа, незалежно від кількості шарів, перетворилася б на просту лінійну регресію. ReLU обнуляє всі від’ємні значення, залишаючи позитивні без змін. Іноді це називають активацією, оскільки в наступний шар переносяться тільки активовані ознаки. Це робить нейронну мережу здатною моделювати нелінійні залежності, які важко відобразити за допомогою лінійних перетворень.

Шар підвибірки або субдискретизації (Pooling Layer) використовується для зменшення просторової розмірності карти ознак (наприклад, перетворює 32×32 у 16×16), при цьому залишаючи найважливішу інформацію. Це знижує обчислювальне навантаження, запобігає перенавчанню та робить мережу стійкою до невеликих зміщень та деформацій об’єкта.

Найчастіше використовується максимальне об’єднання (Max Pooling). Метод Max Pooling (див. рис. 2.3) вибирає максимальне значення певної

ділянки попереднього результату (наприклад, 2x2) й знаходить її максимум, що буде виходом з поточного шару. Так робимо для кожної ділянки. Це найпопулярніший метод, оскільки він зберігає найяскравіші ознаки (наприклад, наявність чіткої межі).

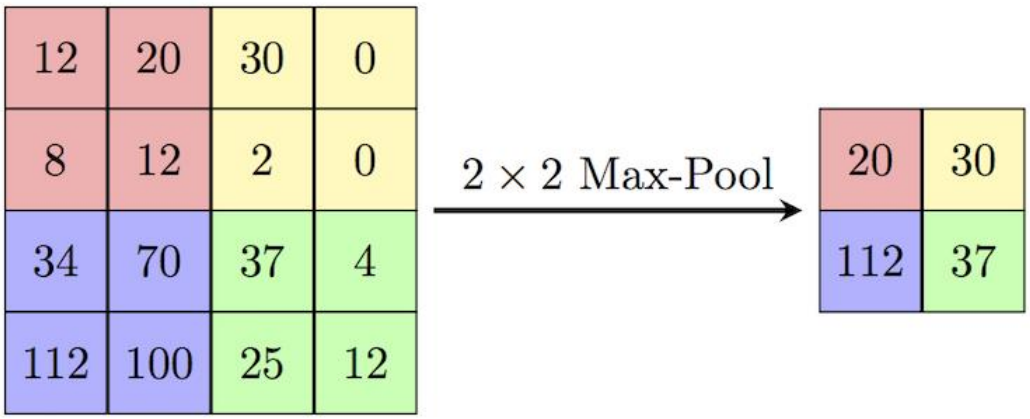


Рисунок 2.3 – Приклад операції пулінгу фільтром 2×2 з кроком два

Фактично, вище наведено опис одного шару згорткової нейронної мережі: перший крок – це згортка зображення накладанням на нього фільтру, другий крок – максимізація ділянок на отриманому виході.

На рисунку 2.4 наведений приклад того, як влаштований шар згорткової нейронної мережі.

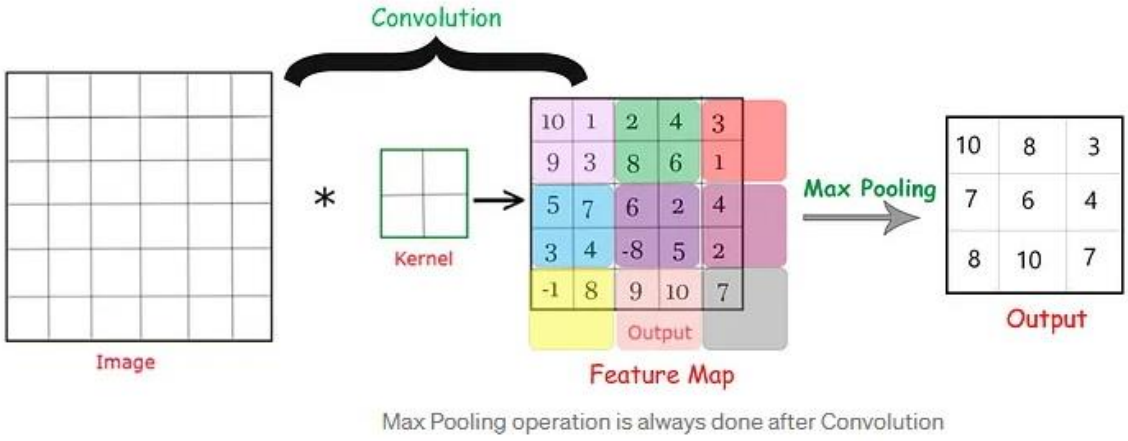


Рисунок 2.4 – Приклад одного шару згорткової нейронної мережі

Спершу зображення Image проходить згортку фільтром Kernel, в результаті чого отримується нове зображення Feature Map. Після операції

згортки, до Feature Map зображення застосовується операція Max Pooling, де обирається максимальне значення для кожної кольорової області, що і є результатом роботи згорткового шару, тобто Output.

Якщо виходи з одного шару такої мережі передати на вхід до іншого, то ми отримаємо складніші патерни. І шар за шаром, дуже прості лінії та точки перетворяться в повноцінні прототипи об’єктів, які бачить людина.

Накладаючись, шар за шаром, згорткові нейронні мережі здатні отримувати все складніші та складніші ознаки з зображення. Таким чином будуються глибокі згорткові нейронні мережі.

Після навчання ознак на багатьох шарах архітектура згорткової нейронної мережі переключається на класифікацію.

Передостанній шар – це повнозв’язний шар (FC), який видає вектор з K вимірів (де K – кількість класів, які можна передбачити) і містить ймовірності для кожного класу класифікованого зображення.

У заключній частині архітектури CNN використовується шар класифікації для отримання остаточного результату етапу класифікації. Тут відбувається логічне прийняття рішення – до якого класу належить зображення. Останній повнозв’язний шар (FC) має стільки нейронів, скільки є класів у задачі, й повертає ймовірності належності до кожного класу.

Функція втрат (Loss Function) відображає різницю між передбаченим результатом і правильним.

Процес навчання згорткових нейронних мереж полягає у мінімізації функції втрат (Loss Function) шляхом оновлення ваг мережі методом зворотного поширення помилки (Backpropagation) за допомогою класичного алгоритму градієнтного спуску.

Архітектура згорткових нейронних мереж CNN дозволяє ефективно вирішувати задачі розпізнавання складних об’єктів, зокрема у інтелектуальних системах відеоспостереження.

2.2 Класифікація та порівняльний аналіз архітектур нейронних мереж

Вибір архітектури є критичним етапом проектування системи безпеки, оскільки від нього залежить баланс між точністю та швидкістю. Архітектури детекції поділяються на дві основні парадигми: двостадійні (Two-Stage) та одностадійні (One-Stage).

2.2.1 Двостадійні детектори (Two-Stage Detectors)

Ці моделі працюють у два етапи – спочатку виділяють потенційно цікаві області (Region Proposals), а потім детально їх аналізують.

R-CNN (Region-based CNN) – родоначальник методу. Використовував зовнішній алгоритм Selective Search для генерації ~2000 регіонів, кожен з яких проганявся через CNN окремо. Це було дуже повільно (десятки секунд на кадр).

Fast R-CNN – прискорив процес, подаючи на вхід CNN все зображення одразу і використовуючи ROI Pooling для вилучення ознак з карти ознак.

Faster R-CNN – замінив Selective Search на нейромережу Region Proposal Network (RPN), яка навчається разом з основною мережею. Це стало стандартом для задач, де точність важливіша за швидкість (наприклад розпізнавання обличь на прохідній, де люди зупиняються перед камерою).

Mask R-CNN – додає гілку для сегментації об’єкта (виділення піксельної маски). Корисно для точного визначення контурів.

2.2.2 Одностадійні детектори (One-Stage Detectors)

Для відеоспостереження, де камери генерують 25 та більше кадрів на секунду, двостадійні детектори часто є занадто повільними. Одностадійні моделі виконують детекцію за один прохід мережі.

YOLO (You Only Look Once) – революційна архітектура, що розбиває зображення на сітку. Кожна комірка сітки відповідає за детекцію об’єкта, центр якого в ній знаходиться.

YOLOv1-v3 створені Joseph Redmon. YOLOv3 ввела детекцію на трьох масштабах (Feature Pyramid), що вирішило проблему дрібних об’єктів.

YOLOv4 (Alexey Bochkovskiy) – оптимізована для навчання на одній GPU, ввела поняття "Bag of Freebies" (аугментації) та "Bag of Specials" (архітектурні модулі).

YOLOv5/v8 (Ultralytics) – фокус на зручності використання (PyTorch implementation), швидкості експорту на мобільні пристрої та вбудованих інструментах для роботи з даними.

YOLOv10 (Tsinghua University) вводить революційну концепцію Consistent Dual Assignment (узгодженого подвійного призначення), що дозволяє відмовитися від ресурсомісткої операції NMS (Non-Maximum Suppression) під час інференсу, ключовому етапі в процесі застосування штучного інтелекту. Традиційні детектори використовують стратегію "один-до-багатьох" (one-to-many), де одному об’єкту Ground Truth (GT) відповідає кілька позитивних якорів. Це покращує збіжність, але створює дублікати. YOLOv10 додає гілку "один-до-одного" (one-to-one), що значно знижує затримку (latency) та обчислювальне навантаження, роблячи модель ідеальною для edge-пристроїв.

YOLO-NAS (Neural Architecture Search) – архітектура, знайдена автоматично за допомогою алгоритмів пошуку, оптимізована під конкретне апаратне рішення.

SSD (Single Shot MultiBox Detector) – використовує карти ознак з різних шарів піраміди для передбачення об’єктів різних розмірів. Раніше був основним конкурентом YOLO, зараз часто поступається в точності/швидкості новим версіям YOLO.

RetinaNet – Ввела Focal Loss, щоб вирішити проблему дисбалансу класів (велика кількість фону проти малої кількості об’єктів), що дозволило одностадійним детекторам наздогнати двостадійні за точністю.

2.2.3 Трансформери (Vision Transformers)

Останній тренд у розпізнавання складних об’єктів – використання механізму Self-Attention, запозиченого з обробки природної мови (NLP).

DETR (DEtection TRansformer) відмовляється від якорів (anchors) та NMS (Non-Maximum Suppression). Розглядає детекцію як задачу прямого передбачення множини об’єктів (set prediction) за допомогою двочасткового співставлення (bipartite matching).

Переваги – глобальний контекст, мережа "бачить" зв’язок між віддаленими частинами зображення.

Недоліки – дуже довге навчання та високі вимоги до обсягу вхідних даних. Поки що рідко застосовується в embedded системах безпеки через важкість обчислень, але перспективна для серверної аналітики.

RT-DETR (Real-Time Detection Transformer) адаптує архітектуру трансформерів для задач реального часу, вирішуючи проблему високої обчислювальної складності стандартного DETR за допомогою гібридного енкодера.

2.2.4 Порівняння параметрів архітектур нейронних мереж

У таблиці 2.1 наведено порівняльний аналіз параметрів архітектур нейронних мереж.

Із порівняння параметрів архітектур нейронних мереж видно, що для задач відеоспостереження в реальному часі (Real-time Object Detection) найбільш оптимальними є архітектури сімейства YOLO. Вони забезпечують оптимальний баланс між точністю та швидкістю. Двостадійні детектори (Faster R-CNN) залишаються нішевими рішеннями для задач надвисокої точності. Модель RT-DETR є оптимальною для серверних рішень, де критично важливою є висока точність у складних сценах з багатьма об’єктами.

Таблиця 2.1 – Порівняння параметрів архітектур нейронних мереж

Параметр порівняння	Двостадійні детектори (R-CNN family)	Одностадійні детектори (YOLO, SSD)	Трансформери (DETR, ViT)
Принцип	1. Генерація регіонів (RPN) 2. Класифікація регіонів	Пряма регресія координат та класів з зображення	Механізм уваги (Self-Attention) для зв’язків між частинами
Точність (mAP)	Висока (особливо для дрібних об’єктів)	Середня/Висока	Дуже висока (на великих датасетах)
Швидкість (FPS)	Низька (5-15 FPS)	Дуже висока (30-140+ FPS)	Повільна/Середня
Обчислювальна складність	Висока, вимагає потужних GPU	Масштабована (від мобільних версій до GPU)	Висока, потребує багато пам’яті
Типові застосування	Медична діагностика, аналіз супутникових знімків	Відео-спостереження, автономні авто, дрони	Аналіз складних сцен, детекція взаємодій

Отримані результати аналізу параметрів архітектур нейронних мереж дозволяють обґрунтувати доцільність розробки власних нейромережевих систем розпізнавання об’єктів для задач відеоспостереження в реальному часі з використанням архітектури сімейства YOLO. Вони забезпечують оптимальний баланс між точністю та швидкістю.

3 ПРОЕКТУВАННЯ ТА ПРАКТИЧНА РЕАЛІЗАЦІЯ МОДУЛЯ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ ДЛЯ СИСТЕМ БЕЗПЕКИ

3.1 Теоретичні основи розпізнавання об'єктів (Object Recognition)

Розпізнавання об'єктів (Object Recognition) у системах безпеки – це складна задача комп'ютерного зору, яка полягає у виявленні, локалізації та класифікації об'єктів на зображеннях або відеокадрах.

На відміну від простої класифікації зображень, де визначається лише клас зображення загалом, процес розпізнавання об'єктів передбачає:

– визначення координат об'єкта (bounding box);
– віднесення кожного об'єкта до певного класу;
– іноді – побудову маски сегментації.

Формально задача розпізнавання об'єктів може бути представлена як відображення:

$F: I \rightarrow \{(b_i, c_i, p_i)\}$

де:

I – вхідне зображення;

b_i – координати обмежувальної рамки i-го об'єкта;

c_i – клас об'єкта;

p_i – ймовірність належності до класу.

Основні складові підзадачі процесу розпізнавання об'єктів:

– Детекція (detection) – виявлення об'єктів на зображенні;
– Класифікація (classification) – віднесення об'єкта до певного класу;
– Локалізація (localization) – визначення положення об'єкта;
– Сегментація (segmentation) – піксельне окреслення об'єкта.

На рисунку 3.1 показані складові процесу розпізнавання об'єктів та різниця між різними задачами в комп'ютерному зорі, як для поодиноких, так і для багатьох об'єктів.

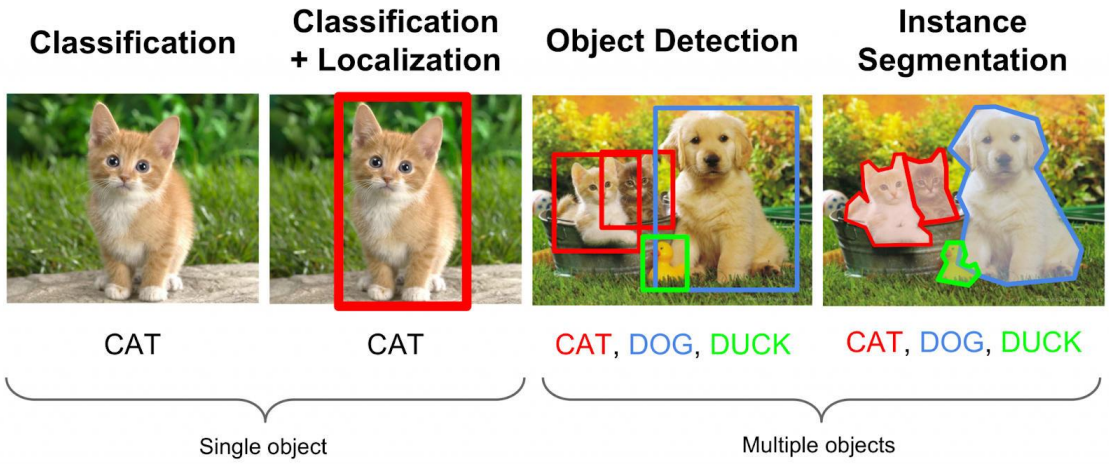


Рисунок 3.1 – Основні складові підзадачі процесу розпізнавання об’єктів

Таким чином, для вирішення нашого завдання розпізнавання об’єктів в системах безпеки нам необхідно на вихідному зображенні виявити різні об’єкти, класифікувати їх – визначити, що це за об’єкти та визначити їхнє положення у просторі зображення за допомогою обмежувальної рамки (bounding box). Застосування комп’ютерного зору дозволяє перейти від пасивного відеоспостереження до активного аналітичного контролю.

3.2 Загальна концепція та архітектура програмної реалізації

Метою практичної частини магістерської роботи є розробка та дослідження програмної системи розпізнавання об’єктів на зображенні, у відеофайлі або у відеопотоці для можливого їх подальшого застосування у системах безпеки.

Об’єктами розпізнавання в межах роботи обрано людей та транспортні засоби. Виявлення людей та транспортних засобів найчастіші завдання в різноманітних системах безпеки.

Проектування будь-якої інтелектуальної системи безпеки починається з чіткого формулювання функціональних та нефункціональних вимог. Виходячи

з аналізу, проведеного у попередніх розділах, до системи розпізнавання об’єктів висуваються наступні основні вимоги.

Функціональні вимоги:

– автоматичне виявлення об’єктів на зображенні, у відеофайлі або у відеопотоці;
– класифікація типів об’єктів (людина, транспортних засіб);
– визначення положення об’єкта (bounding box);
– збереження результатів аналізу;
– виведення результатів у реальному часі.
– Нефункціональні вимоги:
– робота в режимі реального часу (20-30 FPS);
– відмовостійкість системи;
– масштабованість під різну кількість камер.

Структурна схема архітектури системи наведена на рисунку 3.2.

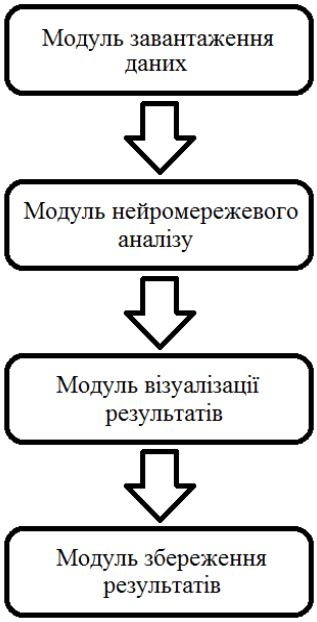


Рисунок 3.2 – Загальна архітектура програмної реалізації

Загальна архітектура програмної реалізації має модульну багаторівневу структуру та складається з таких основних компонентів:

– модуль завантаження даних (Data Acquisition Layer);
– модуль нейромережевого аналізу (AI Inference Layer);

– модуль візуалізації результатів (Result visualization Layer);
– модуль збереження результатів (Result preservation Layer).

3.3 Вибір середовища розробки програмного забезпечення

Створення систем штучного інтелекту відрізняється від розробки звичайних інформаційних систем. При роботі над ними використовується зовсім інший технологічний підхід, потрібні навички формування тренувальних наборів даних і машинного навчання. Для реалізації систем штучного інтелекту бажано мати просту і зручну мову програмування з великою кількістю готових бібліотек. Python – якраз одна з таких мов, на ній ведеться величезна кількість проектів, пов’язаних зі створенням нейронних мереж і машинним навчанням.

Величезна кількість бібліотек Python для штучного інтелекту та машинного навчання значно спрощують і прискорюють розробку. Простий синтаксис і читабельність сприяють швидкому тестуванню складних процесів і роблять мову зрозумілою для всіх. Для реалізації алгоритмів машинного навчання та штучного інтелекту необхідне добре структуроване та перевірене середовище, тільки так можна досягти найкращих результатів. Численні бібліотеки Python, призначені для машинного навчання, дозволяють значно скоротити час створення проектів, а наявність безлічі вже налаштованих бібліотек для роботи із зображеннями остаточно визначило його як основний засіб розробки.

У процесі розробки програмних модулів зручніше працювати в інтерактивному середовищі розробки (IDE), а не в текстовому редакторі. Для Python одним з найкращих варіантів буде IDE PyCharm від компанії JetBrains.

Більшість сучасних моделей розпізнавання об’єктів розробляються з використанням бібліотек глибокого навчання, таких як Keras, TensorFlow та PyTorch. Ці бібліотеки надають попередньо навчені моделі та архітектури, які

можна точно налаштувати під конкретні завдання, скоротивши таким чином необхідний час розробки.

Keras – одна з основних бібліотек Python з відкритим вихідним кодом, написана для побудови нейронних мереж і проектів машинного навчання. Keras може працювати спільно з MXNet, Microsoft Cognitive Toolkit (CNTK), Theano, TensorFlow. У цій бібліотеці реалізовані практично всі автономні модулі нейронної мережі, включаючи оптимізатори, нейронні шари, функції активації шарів, схеми ініціалізації, функції витрат і моделі регуляризації. Це дозволяє будувати нові модулі нейронних мереж, просто додаючи функції або класи. І оскільки модель вже визначена в коді, розробнику не доводиться створювати для неї окремі конфігураційні файли.

Розробка бібліотеки Keras почалася на початку 2015 року, і на сьогоднішній день вона еволюціонувала в одну з найпопулярніших та широко використовуваних бібліотек, побудованих поверх бібліотеки Theano, яка для прискорення тренування нейронних мереж дозволяє задіяти графічний процесор. Одна з її яскравих рис – інтуїтивно зрозумілий інтерфейс, що дозволяє реалізовувати нейронні мережі всього в декількох рядках вихідного коду.

Бібліотека TensorFlow працює в команді з іншими корисними бібліотеками, зокрема з Keras. Тандем TensorFlow + Keras значно спрощує конструювання нейронних мереж. У Keras реалізовані зручність і простота створення прототипів, а TensorFlow успішно використовує ці якості.

За допомогою TensorFlow можна побудувати глибокі нейронні мережі для розпізнавання зображень і рукописного тексту та рекурентні нейронні мережі для NLP (обробки природних мов). Також є модулі для векторизації слів (embedding) і вирішення диференціальних рівнянь у часткових похідних (PDE). Цей фреймворк має відмінну архітектурну підтримку, що дозволяє з легкістю виконувати обчислення на найрізноманітніших платформах, у тому числі на десктопах, серверах і мобільних пристроях.

Бібліотека OpenCV (Open Source Computer Vision Library) – одна з найпопулярніших бібліотек для додатків з комп'ютерного зору. OpenCV-Python – це Python-версія інтерфейсу для OpenCV. Наявність коду на C/C++ в бекенді гарантує швидку роботу додатків, а Python-обгортка забезпечує легкість налаштування і розгортання. Завдяки цьому OpenCV-Python є відмінним рішенням для високонавантажених обчислювальних програм з комп'ютерного зору. Це бібліотека комп'ютерного зору і машинного навчання з відкритим вихідним кодом. До неї входять понад 2500 алгоритмів, серед яких є як класичні, так і сучасні алгоритми для комп'ютерного зору і машинного навчання.

Великою популярністю у розробників користується ImageAI. Це бібліотека Python, створена для того, щоб дозволити розробникам створювати додатки та системи з можливостями глибокого навчання (deep learning) і комп'ютерного зору (computer vision) з використанням простої і невеликої кількості рядків програмного коду.

Бібліотека ImageAI призначена для створення нейронних мереж обробки зображень. Це потужна бібліотека, що має цілий набір вже навчених моделей нейронних мереж, які користувач може всього декількома рядками програмного коду підключити до своїх додатків для розпізнавання близько 80 різних об'єктів. На вхід цим моделям можна подавати або малюнки і фотографії у вигляді файлів, або відеопотік з відеокамер.

У цій бібліотеці є моделі нейронних мереж для виявлення об'єктів на зображеннях з архітектурою YOLOv3 яка, як було зазначено вище, забезпечує оптимальний баланс між точністю та швидкістю. Для таких моделей користувач може сформувати свій навчальний набір даних і, по суті, навчити модель розпізнавати будь-які об'єкти на зображеннях, у відеофайлах і потоковому відео.

Розроблений Мозесом Олафенвою і створений з урахуванням простоти використання, ImageAI в даний час підтримує ряд сучасних алгоритмів виявлення об'єктів, виявлення відео і відстеження об'єктів з використанням

моделей RetinaNet, YOLOv3 і TinyYOLOv3, навчених на наборі даних COCO (Common Objects in Context). Крім цього, ImageAI дозволяє навчати користувацькі моделі для виконання виявлення і розпізнавання нових об’єктів.

3.4 Програмна реалізація модуля розпізнавання об’єктів

Для роботи з ImageAI потрібно встановити Python версії 3.5.1 або вище, а також деякі інші бібліотеки:

– Tensorflow 1.14.0 або вище;
– OpenCV;
– Keras;
– ImageAI.

Для завантаження даних бібліотек можна використовувати наступні команди, наведені на рисунку 3.3:

```
pip install tensorflow==1.15.3
pip install opencv-python
pip install keras
pip install imageai
```

Рисунок 3.3 – Встановлення необхідних бібліотек

За допомогою ImageAI можна створювати складні додатки, пов’язані з комп’ютерним зором, з мінімальною кількістю програмного коду:

– розпізнавати близько 1000 різних об’єктів на зображеннях;
– виявляти 80 найпоширеніших предметів побуту на зображеннях;
– виявляти 80 найпоширеніших предметів побуту у відеофайлах;
– отримувати інформацію про час виявлення об’єктів у відеопотоках;
– навчати моделі нейронних мереж розпізнавати об’єкти користувачів;
– навчати нові моделі YOLOv3 для виявлення об’єктів користувачів.

Бібліотека ImageAI надає дуже потужні, але прості у використанні класи та функції цих класів (методи) для виконання завдань виявлення та розпізнавання об’єктів з різних зображень. Набір функцій в ImageAI дозволяє виконувати все це за допомогою сучасних алгоритмів глибокого навчання, таких як RetinaNet, YOLOv3 і TinyYOLOv3. Можна вирішувати найпоширеніші завдання комп’ютерного зору за допомогою коду Python, при цьому потрібно написати всього від 5 до 15 рядків програмного коду. Розглянемо опис основних класів і відповідних методів (функцій всередині класів), які дозволяють швидко і просто створювати ефективні додатки. Ці класи можуть бути інтегровані в будь-яку програму, що розробляється на Python, будь то веб-сайт, додаток для Windows, Linux або MacOS.

3.4.1 Клас ObjectDetection для виявлення та розпізнавання об’єктів на зображеннях

Клас ObjectDetection надає функції (методи) для виявлення об’єктів на будь-якому зображенні або наборі зображень за допомогою моделей, попередньо навчених на наборі даних COCO (Common Objects in Context) , який містить:

– понад 330 000 зображень;
– 80 класів об’єктів;
– понад 1,5 млн анотованих об’єктів.

Клас підтримує наступні моделі: RetinaNet , YOLOv3 і TinyYOLOv3. Це означає, що можна виявити і розпізнати 80 різних повсякденних предметів. Об’єктами розпізнавання в межах роботи обрано людей (person) та транспортні засоби (car, bus, truck).

Для початку необхідно завантажити будь-яку з цих попередньо навчених моделей, яку ми плануємо використовувати, за посиланням:

https://github.com/OlafenwaMoses/ImageAI/releases/tag/1.0/

Ми вибираємо модель YOLOv3, файл «yolo.h5», розміром 242 859 Кбайт.

Функція .detectObjectsFromImage() виконує завдання виявлення об’єктів після завантаження моделі. Її можна викликати багаторазово для виявлення об’єктів на будь-якій кількості зображень. Ініціювати процес виявлення об’єктів можна за допомогою наступного програмного коду (див. рис. 3.4):

```
detections = detector.detectObjectsFromimage(
                      input_image="image.jpg",
                      output_image_path="imagenew.jpg",
                      minimum_percentage_probability=30)
```

Рисунок 3.4 – Виклик функції .detectObjectsFromImage()

Функція .detectObjectsFromImage() має кілька параметрів:

– input_image (обов’язковий) – шлях і ім’я файлу з зображенням, яке хочемо обробляти. Цей символьний параметр має значення «file» (файл) за замовчуванням, а також може приймати значення «array» (масив) і «stream» (потік). Можливо обробляти не один файл, а групу файлів. Для цього потрібно імена оброблюваних файлів записати в масив Numpy (якщо для параметра input_type вказано значення «array» або «stream»);
– output_image_path (потрібно тільки в тому випадку, якщо input_type=«file») – шлях до файлу, в якому буде збережено оброблене зображення;
– minimum_percentage_probability (необов’язковий) – параметр для завдання ймовірності результатів виявлення об’єктів (%). Зниження цього значення призведе до показу більшої кількості об’єктів, тоді як збільшення цього значення забезпечує виявлення меншої кількості об’єктів з більш високою точністю прогнозування. Значення за замовчуванням дорівнює 50;
– output_type (необов’язковий) – параметр для встановлення формату, в якому буде отримано оброблене зображення. Доступні значення – «file» (файл; значення за замовчуванням) і «array» (масив). Якщо для

цього параметра встановлено значення «array», функція поверне масив Numpy оброблених зображень;

- display_percentage_probability (необов'язковий) – параметр для приховування об'єктів з малою ймовірністю вірності розпізнавання, виявлених в обробленому зображенні (якщо встановлено значення False). Значення за замовчуванням True;
- display_object_name (необов'язковий) – параметр для приховування імені кожного об'єкта, виявленого в обробленому зображенні (якщо встановлено значення False). Значення за замовчуванням True;
- extract_detected_objects (необов'язковий) – параметр для вилучення та збереження кожного об'єкта, виявленого в обробленому зображенні, у вигляді окремого зображення (файлу). Значення за замовчуванням False;
- thread_safe (необов'язковий) – параметр, що гарантує, що завантажена модель виявлення працює у всіх потоках, якщо встановлено значення True;
- return – набір повернутих значень, які будуть залежати від параметрів, заданих у функції detectObjectsFromImage().

Функція .CustomObjects() використовується, коли необхідно задати виявлення тільки певних об'єктів. Функція повертає словник об'єктів і їх значення True або False. Для того щоб виявити вибрані об'єкти на зображенні, слід використовувати словник, що повертається функцією this(), з функцією: detectCustomObjectsFromImage().

Моделі можна вказати, що на зображенні потрібно виявити тільки деякі з перерахованих вище об'єктів. Для цього необхідно викликати функцію CustomObjects() і задати їй в якості параметра ім'я об'єкта або список імен бажаних об'єктів. Імена цих об'єктів отримають ознаку true, всім іншим іменам об'єктів буде присвоєно ознаку false за замовчуванням.

Функція .detectCustomObjectsFromImage() має ті ж параметри і повертає ті ж значення, що і функція detectObjectsFromImage(), але у неї є невелика

відмінність. Вона дозволяє виявляти на зображенні тільки ті об’єкти, які задавав користувач. На відміну від звичайної функції detectObjectsFromImage(), для цього потрібно явно задати додатковий параметр custom_object, який приймає словник, що повертається функцією CustomObjects().

Використовуючи вищезазначені бібліотеки та функції, напишемо програмний модуль виявлення та розпізнавання об’єктів на зображеннях із застосуванням функції виявлення об’єктів .detectObjectsFromImage() (див. рис. 3.5) і подивимося, наскільки ефективно працює обрана модель нейронної мережі YOLOv3 на тестових і реальних зображеннях.

```
# Diplom_Kolisnyk
# Модуль розпізнавання об'єктів на зображенні

from imageai.Detection import ObjectDetection
import os

execution_path = os.getcwd()
im_path = os.path.abspath('D:\\Pictures\\')  # папка із зображеннями

detector = ObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
detector.loadModel()
detections = detector.detectObjectsFromImage(
    input_image=os.path.join(im_path, "test1.jpg"),
    output_image_path=os.path.join(im_path, "test1_new.jpg"),
    minimum_percentage_probability=30)

for eachObject in detections:
    print(eachObject["name"], " : ", eachObject["percentage_probability"], " : ",
eachObject["box_points"])
    print("--------------------------------")
```

Рисунок 3.5 – Модуль виявлення і розпізнавання об’єктів на зображеннях

Запускаємо програму і подаємо на вхід нейронної мережі тестове зображення test1.jpg (див. рис. 3.6). В результаті роботи програми у вихідній папці отримуємо файл обробленого зображення з іменем test1_new.jpg (див. рис. 3.7). Нейронна мережа впевнено розпізнала і людину (100%), і собаку

(100%), і кішку (91%) та виділила об’єкти кольоровими рамками. Крім обробленого зображення програма видала наступну текстову інформацію:

dog : 99,76242780685425 : [404, 393, 529, 559]
cat : 90,87273478507996 : [138, 427, 212, 508]
person : 99,92441534996033 : [232, 106, 405, 547]



Рисунок 3.6 – Тестове зображення test1.jpg

Далі в коді програми змінюємо імена файлів на "cam_01.jpg" і "cam_01_new.jpg". Запускаємо програму і подаємо на вхід нейронної мережі реальне зображення з камери відеоспостереження cam_01.jpg (див. рис. 3.8). В результаті роботи програми у вихідній папці отримуємо новий файл обробленого зображення з іменем cam_01_new.jpg (див. рис. 3.9).

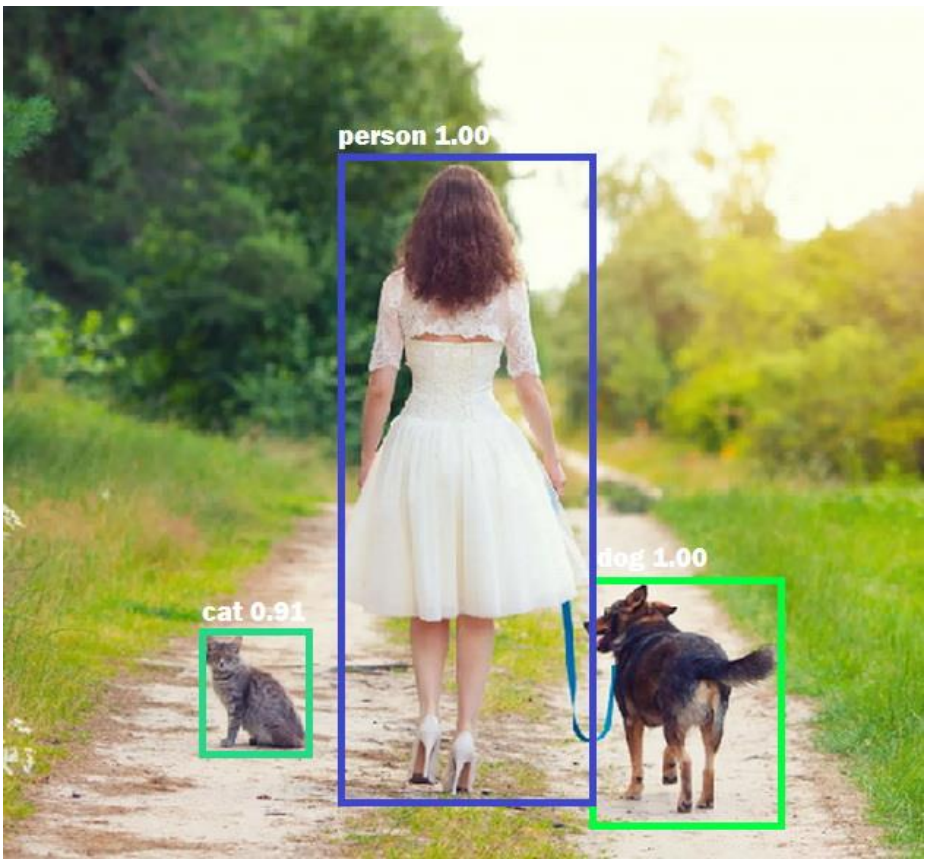


Рисунок 3.7 – Оброблене зображення test1_new.jpg



Рисунок 3.8 – Реальне зображення з камери відеоспостереження cam_01.jpg



Рисунок 3.9 – Оброблене зображення cam_01_new.jpg

Нейронна мережа на реальному зображенні також впевнено розпізнала людей (99% и 96%), автобус (100%) і автомобіль на задньому плані (76%) та виділила об’єкти кольоровими рамками.

Тепер трохи змінимо початковий програмний код і обмежимо кількість об’єктів, що розпізнаються на зображенні. Дамо команду визначити на тих самих зображеннях, наприклад, лише людину і собаку. Для цього додаємо функцію .CustomObjects() для визначення класів розпізнаваних об’єктів і застосуємо нову функцію виявлення об’єктів .detectCustomObjectsFromImage(). Програмний модуль виявлення та розпізнавання обраних об’єктів на зображеннях буде виглядати наступним чином (див. рис. 3.10).

Запускаємо програму і подаємо на вхід нейронної мережі тестове зображення test1.jpg. В результаті роботи програми у вихідній папці отримуємо новий файл обробленого зображення з іменем test1_new_select.jpg (див. рис. 3.11). Нейронна мережа у даному випадку розпізнала лише визначені в програмі класи об’єктів – людину (100%) і собаку (100%) та виділила ці об’єкти кольоровими рамками.

```
# Diplom_Kolisnyk
# Модуль розпізнавання обраних об'єктів на зображенні

from imageai.Detection import ObjectDetection
import os

execution_path = os.getcwd()
im_path = os.path.abspath('D:\\Pictures\\')  # папка із зображеннями

detector = ObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
detector.loadModel()
custom = detector.CustomObjects(person=True, dog=True)
detections = detector.detectCustomObjectsFromImage(custom_objects=custom,
    input_image=os.path.join(im_path, "test1.jpg"),
    output_image_path=os.path.join(im_path, "test1_new_select.jpg"),
    minimum_percentage_probability=30)

for eachObject in detections:
    print(eachObject["name"], " : ", eachObject["percentage_probability"], " : ",
eachObject["box_points"])
    print("--------------------------------")
```

Рисунок 3.10 – Модуль виявлення і розпізнавання обраних об’єктів на зображеннях

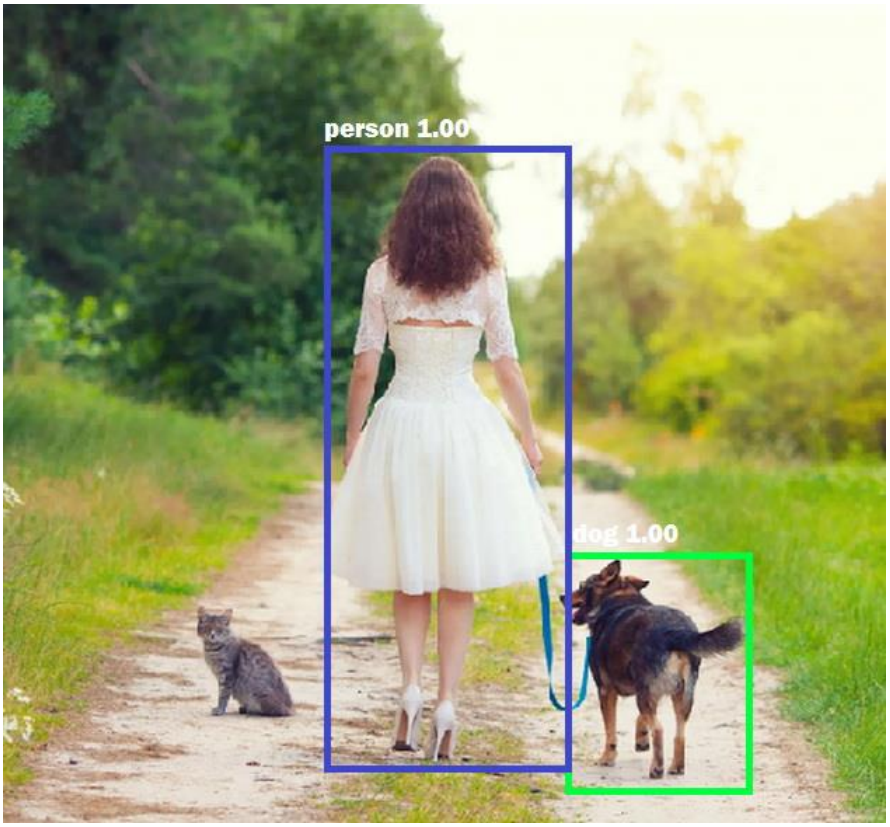


Рисунок 3.11 – Оброблене зображення test1_new_select.jpg

Далі в коді програми змінюємо імена файлів на "cam_01.jpg" і "cam_01_new_select.jpg". Запускаємо програму і подаємо на вхід нейронної мережі реальне зображення з камери відеоспостереження cam_01.jpg. В результаті роботи програми у вихідній папці отримуємо новий файл обробленого зображення з іменем cam_01_new_select.jpg (див. рис. 3.12).

Нейронна мережа на реальному зображенні у даному випадку розпізнала лише визначений в програмі клас об’єктів – людей (99% и 96%) та виділила ці об’єкти кольоровими рамками.

При цьому другий визначений для виявлення в програмі клас об’єктів (собака) на реальному зображенні відсутній.



Рисунок 3.12 – Оброблене зображення cam_01_new_select.jpg

Тестування розробленого програмного модуля виявлення та розпізнавання об’єктів на тестових і реальних зображеннях з камер відеоспостереження продемонструвало високу ефективність роботи обраної

моделі нейронної мережі YOLOv3 та здатність розпізнавання об’єктів з точністю прогнозу від 76% до 99%.

Отримані високі результати розпізнавання об’єктів свідчать про готовність розробленого програмного модуля до інтеграції в реальні існуючі системи відеомоніторингу.

3.4.2 Клас VideoObjectDetection для виявлення та розпізнавання об’єктів у відеофайлах і відеопотоках з відеокамер

За допомогою бібліотеки ImageAI можна також вирішувати завдання, пов’язані з виявленням та розпізнаванням об’єктів у відеофайлах і на потоковому відео з відеокамер у режимі реального часу.

Клас VideoObjectDetection надає можливість виявлення та розпізнавання 80 класів об’єктів у відеофайлах і на потоковому відео з відеокамер з використанням моделей RetinaNet, YOLOv3, TinyYOLOv3, попередньо навчених на наборі даних COCO.

Функція .detectObjectsFromVideo() виконує завдання виявлення об’єктів у відеофайлі або відеопотоці з відеокамер в режимі реального часу після завантаження моделі. Ініціювати процес виявлення об’єктів у відеофайлі можна за допомогою наступного програмного коду (див. рис. 3.13):

```
video_path = detector.detectObjectsFromVideo(
             input_file_path=os.path.join(execution_path, "video.mp4"),
             output_file_path=os.path.join(execution_path, "video_detected"),
             minimum_percentage_probability=30)
```

Рисунок 3.13 – Приклад виклику функції .detectObjectsFromVideo()

Функція .detectObjectsFromVideo() має кілька параметрів:

– input_file_path – шлях до відеофайлу, в якому потрібно виявляти об’єкти (вказується в тому випадку, якщо не встановлений вхідний потік з відеокамери – параметр camera_input);

- output_file_path – шлях до папки, в якій буде збережено оброблений вихідний відеофайл (вказується, якщо не встановлено параметр відмови від запису вихідного файлу save_detected_video = False), за замовчуванням ця функція зберігає відео у форматі AVI;
- frames_per_second (необов'язковий) – частота кадрів в секунду для обробленого вихідного відеофайлу (який буде збережений). За замовчуванням дорівнює 20, але є можливість встановити значення, яке найбільш підходить для вихідного відеофайлу;
- minimum_percentage_probability (необов'язковий, мінімальна процентна ймовірність) – параметр для визначення достовірності результатів виявлення об'єктів. При зниженні значення даної ймовірності буде виявлено і показано більше об'єктів, в той час як збільшення значення цієї ймовірності зменшить кількість знайдених об'єктів, але забезпечить більш високу точність виявлення. Значення за замовчуванням становить 50;
- log_progress (необов'язковий) – якщо для цього параметра встановлено значення True, то буде відображатися хід обробки відеофайлу (у вигляді прямої трансляції). Значення за замовчуванням – False;
- return_detected_frame (необов'язковий) – параметр, що дозволяє повертати виявлений об'єкт у кадрі у вигляді масиву Numpy (для кожного кадру), секунди та хвилини виявлення об'єкта на відео;
- display_percentage_probability (необов'язковий) – параметр для приховування (відображення) ймовірності відповідності об'єкта зразкам, що містяться в навченій моделі (якщо значення параметра дорівнює False, то ймовірність, виражена у відсотках, не буде відображатися). Значення за замовчуванням – True;
- display_object_name (необов'язковий) – параметр для приховування імені кожного об'єкта, виявленого у відео, якщо встановлено значення False. Значення за замовчуванням – True;

– save_detected_video (необов'язковий) – параметр, що вказує програмі, зберігати чи не зберігати виявлені об'єкти в обробленому відеофайлі. За замовчуванням встановлено значення True;
– camera_input (необов'язковий) – параметр, який можна встановити замість параметра input_file_path (якщо потрібно виявляти об'єкти в прямому ефірі з відеокамери, а не з відеофайлу). Для цього попередньо необхідно активувати відеокамеру за допомогою функції VideoCapture() з бібліотеки OpenCV.

Використовуючи вищезазначені бібліотеки та функції, напишемо програмний модуль виявлення та розпізнавання об'єктів у відеофайлах з використанням моделі нейронної мережі YOLOv3 із застосуванням функції виявлення об'єктів .detectObjectsFromVideo() (див. рис. 3.14).

```
# Diplom_Kolisnyk
# Модуль розпізнавання об'єктів у відеофайлі

from imageai.Detection import VideoObjectDetection
import os

execution_path = os.getcwd()
im_path = os.path.abspath('D:\\Video\\')  # папка з відеофайлами

detector = VideoObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
detector.loadModel()

video_path = detector.detectObjectsFromVideo(
    input_file_path=os.path.join(im_path, "video.mp4"),
    output_file_path=os.path.join(im_path, "video_detected"),
    frames_per_second=20, minimum_percentage_probability=30)
print(video_path)
```

Рисунок 3.14 – Модуль виявлення і розпізнавання об'єктів у відеофайлах

Далі, за допомогою функції VideoCapture() напишемо програмний модуль виявлення та розпізнавання об'єктів у відеопотоках (див. рис. 3.15).

```
# Diplom_Kolisnyk
# Модуль розпізнавання об'єктів у відеопотоці

from imageai.Detection import VideoObjectDetection
import os
import cv2

execution_path = os.getcwd()
im_path = os.path.abspath('D:\\Video\\')  # папка з відеофайлами

camera = cv2.VideoCapture(0)

detector = VideoObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
detector.loadModel()

video_path = detector.detectObjectsFromVideo(camera_input=camera,
    output_file_path=os.path.join(im_path, "camera_detected"),
    frames_per_second=20, minimum_percentage_probability=30)
print(video_path)
```

Рисунок 3.15 – Модуль виявлення і розпізнавання об’єктів у відеопотоках

Обидва розроблені програмні модулі виявлення та розпізнавання об’єктів у відеофайлах та у відеопотоках з камер відеоспостереження працюють аналогічно описаному раніше програмному модулю виявлення та розпізнавання об’єктів на зображеннях. На кожному кадрі вхідного відеофайлу або відеопотоку виявляються та розпізнаються до 80 класів об’єктів, всі виявлені об’єкти виділяються кольоровими рамками і оброблений кадр записується в заданий вихідний відеофайл із зазначеними параметрами частоти кадрів в секунду і мінімальної ймовірності виявлення результатів.

Висока ефективність роботи розроблених програмних модулів обумовлена використанням обраної моделі нейронної мережі YOLOv3. Отримані високі результати розпізнавання об’єктів підтверджують готовність розроблених програмних модулів до інтеграції в реальні існуючі системи автоматичного розпізнавання об’єктів, які здатні функціонувати в режимі реального часу.

ВИСНОВКИ

У магістерській кваліфікаційній роботі на тему «Застосування нейромережевих моделей для розпізнавання об’єктів у системах безпеки» було всебічно досліджено теоретичні та практичні аспекти застосування методів глибокого навчання та комп’ютерного зору для побудови інтелектуальних систем відеоаналітики. Актуальність даної теми особливо важлива в умовах воєнного стану та підвищених безпекових ризиків для України.

У ході виконання даної кваліфікаційної роботи були проведені та послідовно вирішені декілька ключових завдань. По перше був проведений порівняльний аналіз існуючих провідних комерційних рішень відеоаналітики провідних виробників Hikvision, Dahua Technology, Axis Communications та хмарних сервісів Verkada, виявлено їхні переваги, недоліки та технічні обмеження. Отримані результати аналізу дозволяють обґрунтувати доцільність розробки власних нейромережевих систем розпізнавання об’єктів, адаптованих до конкретних умов застосування.

Далі був проведений детальний огляд основних нейромережевих моделей для розпізнавання об’єктів: двостадійних детекторів сімейства R-CNN, одностадійних детекторів RetinaNet, SSD, сімейства YOLO, а також сучасні підходи на основі трансформерів Vision Transformers, DETR. Обґрунтовано вибір нейромережевої архітектури YOLO для подальшої практичної реалізації. Вона забезпечує оптимальний баланс між точністю та швидкістю для вирішення задач детекції об’єктів в умовах обмежених апаратних ресурсів.

Далі були визначені основні складові підзадачі процесу розпізнавання об’єктів: детекція, класифікація, локалізація, сегментація; запропоновані загальна концепція та архітектура програмної реалізації; обрано середовище розробки програмного забезпечення: мова програмування Python та бібліотеки

Keras, TensorFlow, OpenCV, ImageAI.

У відповідності до поставленої мети у практичний частини магістерської роботи були розроблені та досліджені програмні модулі розпізнавання об’єктів на зображеннях, у відеофайлах та у відеопотоках з камер відеоспостереження. Тестування розроблених програмних модулів на реальних зображеннях з камер відеоспостереження продемонструвало високу здатність розпізнавання об’єктів з точністю прогнозу від 76% до 99%.

Таким чином, мету магістерської роботи досягнуто, а всі поставлені завдання виконано повністю.

Практична цінність роботи полягає у створенні програмних модулів, готових до інтеграції в існуючі системи відеомоніторингу, які здатні функціонувати в режимі реального часу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Bishop C. M. Pattern Recognition and Machine Learning. New York : Springer, 2006. 738 p.

2. Jason A. The basics of information security: understanding the fundamentals of InfoSec in theory and practice. Syngress, 2011. 190 p.

3. ImageNet Large Scale Visual Recognition Challenge / O. Russakovsky, J. Deng and otc. Cornell University. 2014. 43 p. URL : https://arxiv.org/abs/1409.0575 (дата звернення : 05.10.2025).

4. Mayank Mishra. Convolutional Neural Networks, Explained. 2020. URL : https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939 (дата звернення : 10.10.2025).

5. A Comprehensive Guide to Types of Neural Networks. Digital Vidya. URL : https://www.digitalvidya.com/blog/types-of-neural-networks/ (дата звернення : 15.10.2025).

6. Erhan D., Szegedy C., Toshev A., Anguelov D. Scalable object detection using deep neural networks. *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference*. 2014. P. 2155–2162.

7. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once : Unified, Real-Time Object Detection. Cornell University. 2015. URL : https://arxiv.org/abs/1506.02640 (дата звернення : 20.10.2025).

8. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. Cornell University. 2018. URL : https://arxiv.org/abs/1804.02767 (дата звернення : 20.10.2025).

9. SSD object detection: Single Shot MultiBox Detector for real-time processing. Medium. URL : https://medium.com/@jonathan_hui/ssd-object-

detection-single-shot-multibox-detector-for-real-time-processing (дата звернення : 22.10.2025).

10. Shastry A. P. Convolutional Neural Networks. URL : https://medium.com/ @aparnack/convolutional-neural-networks-part-1-2aeb17fc208c (дата звернення : 22.10.2025).

11. Planche B., Andres E. Hands-On Computer Vision with TensorFlow 2. Packt, 2019. 363 p.

12. Паскарюк Д. О. Розпізнавання образів за допомогою нейронних мереж. Харків : ХНУРЕ, 2020. 77 с.

13. Вовк О., Шаховська Н., Камінський Р. Системи штучного інтелекту. Львів : Львівська політехніка, 2018. 392 с.

14. Пратік Д. Штучний інтелект з прикладами на Python. Київ : Вильямс, 2016. 448 с

15. Використовуємо CNN для обробки зображень. DOu. URL : https://dou.ua/forums/topic/48368/ (дата звернення : 22.10.2025).

16. Python Software Foundation. Python 3 Documentation. URL : https://docs.python.org/3/ (дата звернення : 23.10.2025).

17. Official English Documentation for ImageAI! URL : https://imageai.readthedocs.io/en/latest/ (дата звернення : 23.10.2025).

18. About OpenCV. OpenCV. URL : https://opencv.org/about/ (дата звернення : 25.10.2025).

19. OpenCV4Android SDK. opencv.org. URL : https://docs.opencv.org/2.4/doc/tutorials/introduction/android_binary_package/O4A_SDK.html (дата звернення : 27.10.2025).

ДОДАТОК А

Модулі виявлення та розпізнавання об’єктів на зображеннях

```
# Diplom_Kolisnyk
# Модуль розпізнавання об'єктів на зображенні

from imageai.Detection import ObjectDetection
import os

execution_path = os.getcwd()
im_path = os.path.abspath('D:\\Pictures\\')  # папка із зображеннями

detector = ObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
detector.loadModel()
detections = detector.detectObjectsFromImage(
    input_image=os.path.join(im_path, "test1.jpg"),
    output_image_path=os.path.join(im_path, "test1_new.jpg"),
    minimum_percentage_probability=30)

for eachObject in detections:
    print(eachObject["name"], " : ", eachObject["percentage_probability"], " : ",
eachObject["box_points"])
    print("--------------------------------")
```

Рисунок А.1 – Модуль розпізнавання об’єктів на зображеннях

```
# Diplom_Kolisnyk
# Модуль розпізнавання обраних об'єктів на зображенні

from imageai.Detection import ObjectDetection
import os

execution_path = os.getcwd()
im_path = os.path.abspath('D:\\Pictures\\')  # папка із зображеннями

detector = ObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
detector.loadModel()
custom = detector.CustomObjects(person=True, dog=True)
detections = detector.detectCustomObjectsFromImage(custom_objects=custom,
    input_image=os.path.join(im_path, "test1.jpg"),
    output_image_path=os.path.join(im_path, "test1_new_select.jpg"),
    minimum_percentage_probability=30)

for eachObject in detections:
    print(eachObject["name"], " : ", eachObject["percentage_probability"], " : ",
eachObject["box_points"])
    print("--------------------------------")
```

Рисунок А.2 – Модуль розпізнавання обраних об’єктів на зображеннях

ДОДАТОК Б

Модуль виявлення та розпізнавання об’єктів у відеофайлах

```
# Diplom_Kolisnyk
# Модуль розпізнавання об'єктів у відеофайлі

from imageai.Detection import VideoObjectDetection
import os

execution_path = os.getcwd()
im_path = os.path.abspath('D:\\Video\\')  # папка з відеофайлами

detector = VideoObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
detector.loadModel()

video_path = detector.detectObjectsFromVideo(
    input_file_path=os.path.join(im_path, "video.mp4"),
    output_file_path=os.path.join(im_path, "video_detected"),
    frames_per_second=20, minimum_percentage_probability=30)
print(video_path)
```

Рисунок Б.1 – Модуль виявлення і розпізнавання об’єктів у відеофайлах

ДОДАТОК В

Модуль виявлення та розпізнавання об’єктів у відеопотоках

```
# Diplom_Kolisnyk
# Модуль розпізнавання об'єктів у відеопотоці

from imageai.Detection import VideoObjectDetection
import os
import cv2

execution_path = os.getcwd()
im_path = os.path.abspath('D:\\Video\\')  # папка з відеофайлами

camera = cv2.VideoCapture(0)

detector = VideoObjectDetection()
detector.setModelTypeAsYOLOv3()
detector.setModelPath(os.path.join(execution_path, "yolo.h5"))
detector.loadModel()

video_path = detector.detectObjectsFromVideo(camera_input=camera,
    output_file_path=os.path.join(im_path, "camera_detected"),
    frames_per_second=20, minimum_percentage_probability=30)
print(video_path)
```

Рисунок В.1 – Модуль виявлення і розпізнавання об’єктів у відеопотоках