МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«АВТОМАТИЗАЦІЯ ІДЕНТИФІКАЦІЇ ОБ'ЄКТІВ ПРИ СОРТУВАННІ»**

Виконав: студент 2 курсу, групи 8.1212-іпз-1
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми інженерія програмного забезпечення
(назва освітньої програми)
Д.І. Куц
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф-м.н. Горбенко В.І.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Куц Дмитру Івановичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Автоматизація ідентифікації об’єктів при сортуванні

керівник роботи Горбенко Віталій Іванович, к.ф.-м.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 01 » травня 2023 року № 642-с

2. Строк подання студентом роботи 27.11.2023 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, аналіз предметної області.
2. Проєктування.
3. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

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

7. Дата видачі завдання 05.05.2023 р.

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.	12.05.2023	
2.	Збір вихідних даних.	22.05.2023	
3.	Обробка методичних та теоретичних	19.06.2023	
	джерел.		
4.	Розробка першого та другого розділу.	28.08.2023	
5.	Розробка третього розділу.	30.10.2023	
6.	Оформлення та нормоконтроль	20.11.2023	
	кваліфікаційної роботи магістра.		
7.	Захист кваліфікаційної роботи.	14.12.2023	

Студент _____ Д.І. Куц
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.І. Горбенко
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Автоматизація ідентифікації об’єктів при сортуванні»: 49 с., 17 рис., 1 табл., 28 джерел, 1 додаток.

CNN, COCO, ICP, IMAGE CLASSIFICATION, MAP, MLP, NEURAL NETWORK, OPENCV, PYTHON, RELU, YOLO.

Об’єкт дослідження – процес ідентифікації зображень у задачах промислового сортування.

Мета роботи – розробити застосунок для автоматичної ідентифікації об’єктів за зображенням з використанням нейронних мереж на мові Python.

Методи дослідження – аналіз літературних джерел, математичне моделювання, архітектурне проєктування, програмна реалізація, тестування.

У роботі проаналізовано сучасні підходи до ідентифікації об’єктів на зображеннях, розглянуто базові архітектури штучних нейронних мереж, такі як згорткові (CNN), багатошарові перцептрони (MLP), а також специфічні моделі виявлення об’єктів, зокрема YOLOv3. Описано принципи функціонування згорткових шарів, активації ReLU, пулінгу, повнозв’язних шарів та функцій втрат, таких як крос-ентропія. Запропоновано архітектуру програмного засобу, реалізованого мовою Python з використанням бібліотеки OpenCV. Модель навчено на наборі даних COCO з використанням попередньо натренованої архітектури YOLOv3.

Розроблений застосунок дозволяє здійснювати автоматизоване розпізнавання об’єктів на зображеннях у режимі реального часу та може бути інтегрований у промислові системи сортування. Використання нейромережевих підходів забезпечило високу точність розпізнавання та адаптивність до різних умов роботи.

SUMMARY

Master’s qualifying paper «Automation of the Objects Identification During Sorting»: 49 pages, 17 figures, 1 table, 28 references, 1 supplement.

CNN, COCO, ICP, IMAGE CLASSIFICATION, MAP, MLP, NEURAL NETWORK, OPENCV, PYTHON, RELU, YOLO.

The object of the study is the process of image identification in industrial sorting tasks.

The aim of the study is to develop an application for automatic object identification from images using neural networks implemented in Python.

The research methods include the analysis of literature sources, mathematical modeling, architectural design, software implementation, and testing.

The paper analyzes modern approaches to object identification in images, describes fundamental architectures of artificial neural networks such as convolutional neural networks (CNN), multilayer perceptrons (MLP), as well as object detection models including YOLOv3. It discusses the principles of convolutional layers, ReLU activation, pooling, fully connected layers, and loss functions such as categorical cross-entropy. A software architecture is proposed and implemented using the Python programming language and the OpenCV library. The model is trained on the COCO dataset using a pre-trained YOLOv3 architecture.

The developed application enables automated object recognition in images in real time and can be integrated into industrial sorting systems. The use of neural network approaches ensures high recognition accuracy and adaptability to varying operating conditions.

ЗМІСТ

Завдання на кваліфікаційну роботу……..……..2
Реферат 4
Summary 5
Скорочення та умовні познаки 8
Вступ..... 10
1 Аналіз проблематики області дослідження..... 12
1.1 Сучасні програмні комплекси для розпізнавання образів 12
1.2 Методи розпізнавання образів..... 14
1.3 Критерії якості та оцінювання результатів класифікації..... 18
1.3.1 Побудова вирішального правила 18
1.3.2 Метрики якості класифікації 19
1.3.3 Метрики для об’єктного детектування..... 20
1.4 Висновки до розділу 22
2 Нейронні мережі та способи їх навчання 23
2.1 Нейронні мережі..... 23
2.1.1 Основні типи нейронних мереж..... 24
2.2 Архітектура CNN 27
2.3 Навчання нейронних мереж..... 30
2.3.1 Типи навчання..... 31
2.3.2 Функції втрат..... 31
2.3.3 Оптимізація та оновлення ваг..... 32
2.3.4 Перенавчання та регуляризація..... 32
2.3.5 Якість навчання (generalization) 33
2.4 Висновки до розділу 33
3 Рішення для розробки системи розпізнавання зображень пляшок..... 34
3.1 Функціональні та технічні вимоги до системи 34
3.2 Використані інструменти 34
3.3 Функції системи та їх призначення..... 35

3.4 Керівництво користувача 37
3.5 Результати роботи програми 39
3.6 Висновки до розділу 40
Висновки 41
Перелік посилань 43
Додаток А Програмна реалізація додатку для пошуку пластикових пляшок 46

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

CNN	Convolutional Neural Network – згорткова нейронна мережа; підклас глибинних нейронних мереж прямої дії, що використовується переважно для аналізу візуальних зображень.
COCO	Common Objects in Context – набір зображень, призначений для виявлення об’єктів, сегментації, визначення ключових точок та генерації підписів.
FC	Fully Connected layer – повнозв’язний шар; нейронний шар, у якому кожен нейрон пов’язаний з усіма нейронами попереднього шару.
IoU	Intersection over Union – показник якості перекриття передбаченого об’єкта і справжнього, який використовується в задачах виявлення об’єктів.
ISR	Інтегроване середовище розробки – програмне забезпечення, яке забезпечує розробку, компіляцію, налагодження та тестування програмного коду.
MAP	Mean Average Precision – середнє значення точності; метрика для оцінювання ефективності алгоритмів виявлення об’єктів.
ML	Machine Learning – машинне навчання; галузь штучного інтелекту, що вивчає методи створення алгоритмів, здатних навчатися на основі даних.
MLP	Multilayer Perceptron – багатошаровий перцептрон; тип штучної нейронної мережі, що складається з одного або кількох прихованих шарів.
OpenCV	Open Source Computer Vision Library – бібліотека з відкритим кодом для комп’ютерного зору та обробки зображень.
R-CNN	Region-based Convolutional Neural Network – клас архітектур для виявлення об’єктів, які використовують попередню регіональну пропозицію.

ReLU	Rectified Linear Unit – найпоширеніша функція активації в глибоких нейронних мережах: f(x)=max⁡(0,x)f(x) = \max(0, x)f(x)=max(0,x).
ResNet	Residual Network – глибока нейронна мережа, що використовує залишкові зв’язки для запобігання затуханню градієнта.
RNN	Recurrent Neural Network – рекурентна нейронна мережа; модель, що має зворотні зв’язки та зберігає інформацію про попередні стани.
SGD	Stochastic Gradient Descent – стохастичний градієнтний спуск; базовий метод оптимізації параметрів нейронної мережі.
SSD	Single Shot MultiBox Detector – архітектура для виявлення об’єктів у режимі реального часу без регіональних пропозицій.
YOLO	You Only Look Once – архітектура нейронних мереж, яка накладає на зображення сітку, розділяючи його на локальні осередки і виконує виявлення об’єктів за один прохід.

ВСТУП

У сучасних умовах зростання обсягів споживання й, відповідно, кількості відходів, особливого значення набуває автоматизація процесів сортування сміття. Один із критичних напрямів – виявлення та відбір пластикових пляшок, які є одними з найпоширеніших джерел забруднення довкілля. Ефективне сортування таких об’єктів потребує швидкої та точної ідентифікації на конвеєрних стрічках або в потоках зображень. Наявні рішення для автоматичного розпізнавання об’єктів не завжди адаптовані під специфіку задач сортування, можуть вимагати дорогого обладнання або складного навчання моделей. Водночас поширення відкритих інструментів глибинного навчання, таких як згорткові нейронні мережі та фреймворки OpenCV чи YOLO, створює можливості для створення простих, недорогих і ефективних систем виявлення об’єктів у реальному часі.

Розробка такої системи дозволить автоматизувати процес ідентифікації пластикових пляшок, зменшити навантаження на персонал, підвищити точність сортування та зробити вагомий внесок у сферу екологічних технологій. Реалізація подібної програми може знайти застосування як у навчальних цілях, так і в пілотних виробничих проєктах. Виходячи з цього, було прийнято рішення реалізувати програмний додаток, що поєднує комп’ютерний зір, попередньо навчену модель глибинного навчання та простий інтерфейс користувача для демонстрації результатів.

Актуальність теми зумовлена зростаючою потребою в автоматизованих рішеннях для екологічного сортування відходів і широкими можливостями застосування згорткових нейронних мереж у задачах комп’ютерного зору.

Мета: розробка програми ідентифікації пластикових пляшок на зображеннях з використанням глибинного навчання.

Задачі:

– проаналізувати методи комп’ютерного зору та архітектури CNN для задач розпізнавання об’єктів;

– обґрунтувати вибір моделі та критерії якості;
– розробити програму виявлення пластикових пляшок за допомогою YOLOv3;
– реалізувати графічний інтерфейс користувача;
– протестувати програму на вибраних прикладах.

Об’єкт дослідження: процес автоматичної ідентифікації об’єктів на зображеннях.

Предмет дослідження: методи глибинного навчання та їх реалізація за допомогою бібліотек OpenCV, NumPy, Tkinter та моделі YOLOv3.

Методи дослідження: аналітичний, моделювання, алгоритмізація, програмний.

У першому розділі описано основні підходи до задач розпізнавання об’єктів, проаналізовано популярні архітектури згорткових нейронних мереж та обґрунтовано вибір YOLO як найбільш ефективної моделі.

У другому розділі розглянуто загальну структуру нейронної мережі YOLO, надано опис основних компонентів, формул і алгоритмів її роботи, а також охарактеризовано критерії якості класифікації та детекції.

У третьому розділі представлено реалізацію програмного додатку, описано інтерфейс користувача, роботу з бібліотеками, результати тестування та керівництво з використання програми для ідентифікації пластикових пляшок.

1 АНАЛІЗ ПРОБЛЕМАТИКИ ОБЛАСТІ ДОСЛІДЖЕННЯ

1.1 Сучасні програмні комплекси для розпізнавання образів

У сучасному світі, де технології стрімко розвиваються, аналіз візуальної інформації став ключовим компонентом у багатьох галузях, включаючи науку, медицину та безпеку. Комп’ютерний зір є прикладною та різноманітною галуззю. Ця технологія дозволяє автоматизувати обробку та аналіз зображень завдяки використанню спеціалізованих алгоритмів і моделей машинного навчання.

Комп’ютерний зір – це підгалузь штучного інтелекту, яка дозволяє машинам «бачити», тобто інтерпретувати візуальну інформацію аналогічно до того, як це робить людина. Основними функціями системи є виявлення об’єктів, класифікація, локалізація, сегментація сцени та виявлення аномалій даних [1]. У промисловості ці функції реалізуються в системах автоматизованого сортування сміття, контролю якості продукції, автономного управління тощо.

Цифрові зображення, призначені для обробки, отримуються з датчиків або камер і можуть бути у вигляді 2D-зображень, 3D-моделей або відеопотоків. Кожен піксель зображення несе інформацію про інтенсивність світла, колір та інші фізичні параметри, які можуть бути використані для подальшої аналітики. У більшості випадків перед розпізнаванням виконується попередня обробка зображення, яка включає фільтрацію шуму, масштабування, підвищення контрастності та перетворення системи координат [1].

Історично системи розпізнавання часто використовували інструменти, орієнтовані на оптичне розпізнавання символів (OCR), такі як CuneiForm або FineReader. Тим не менш, ефективність таких систем обмежена в контексті завдань виявлення та класифікації об’єктів у промисловому середовищі. З

огляду на це, дане дослідження зосереджується на сучасних бібліотеках і програмних фреймворках, які здатні керувати зображеннями складної структури, такими як пластикові пляшки [3].

Однією з найпоширеніших платформ розпізнавання зображень є OpenCV – бібліотека з відкритим вихідним кодом, яка надає широкий спектр функцій комп'ютерного зору, включаючи фільтрацію, обробку контурів, сегментацію та виявлення об'єктів. Ця бібліотека підтримує багато мов програмування та добре інтегрується з більшістю фреймворків глибокого навчання [6, 19].

Іншою важливою технологією є YOLO (You Only Look Once) – згорткова нейронна мережа, яка полегшує виявлення об'єктів у реальному часі. Ефективність YOLO особливо помітна в сценаріях, де швидкість і точність локалізації об1єктів мають першорядне значення. Ця архітектура знаходить широке застосування в автоматизованих системах сортування сміття, відеоспостереження та робототехніці [8].

Фреймворки, як TensorFlow та PyTorch, використовуються для побудови, навчання та виконання моделей глибокого навчання в задачах комп'ютерного зору. Вони забезпечують розробку згорткових нейронних мереж, управління гіперпараметрами, організацію процесу навчання й тестування моделей, а також підтримують апаратне прискорення обчислень за допомогою GPU [20, 21].

З огляду на вищезазначене, сучасні програмні системи розпізнавання зображень перетворилися на ключові інструменти інтеграції інтелектуальних систем. Їх впровадження у виробничі процеси, зокрема у сфері сортування твердих побутових відходів, дозволяє підвищити ефективність, зменшити потребу в людській праці, забезпечити адаптивність до змін навколишнього середовища та досягти високої точності ідентифікації об'єктів.

1.2 Методи розпізнавання образів

Сфера розпізнавання образів зазнала значної еволюції, пройшовши шлях від відносно примітивних геометричних методів до складних глибоких нейронних мереж. Ці методи можна розділити на чотири основні групи: традиційні (геометричні та евристичні), засновані на ознаках, машинне навчання та методи глибокого навчання [1].

Першими алгоритмами, які застосовувалися для роботи та розвитку комп'ютерного зору, були геометричні. Вони включаючи виявлення країв (алгоритми Собела, Лапласа і Кенні), представлення контурів, аналіз симетрії і морфологічну обробку. Їх значною перевагою є простота з точки зору процесу обчислень, але вони втрачають ефективність у складних умовах, таких як варіації шуму, зміни освітлення або варіації форми об'єкта [15].

На рисунку 1.1 зображено знаходження градієнту яскравості зображення за допомогою алгоритму Кенні.

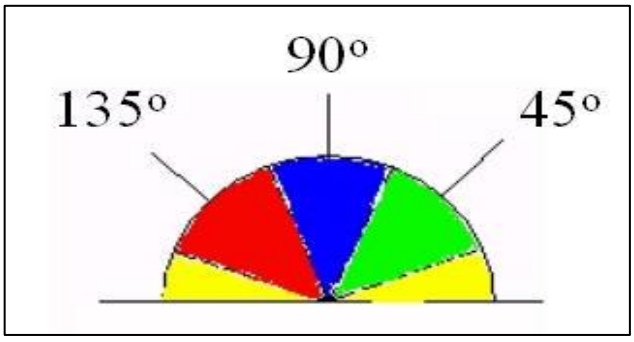


Рисунок 1.1 – Знаходження градієнту яскравості зображення [15]

Методи, в основі яких лежать дескриптори ознак, мають більшу гнучкість, про за рахунок використання моделей SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features) та ORB (Oriented FAST and Rotated BRIEF). Ці моделі полегшують ідентифікацію ключових точок, формулювання вектора дескрипторів і подальшу передачу цих дескрипторів на вхід класифікатора. Перевагою є інваріантність до масштабування, повороту та зміни освітлення, але вони не підтримують самонавчання [6].

На рисунку 1.2 наведено швидке виявлення ключових точок ORB-моделі.

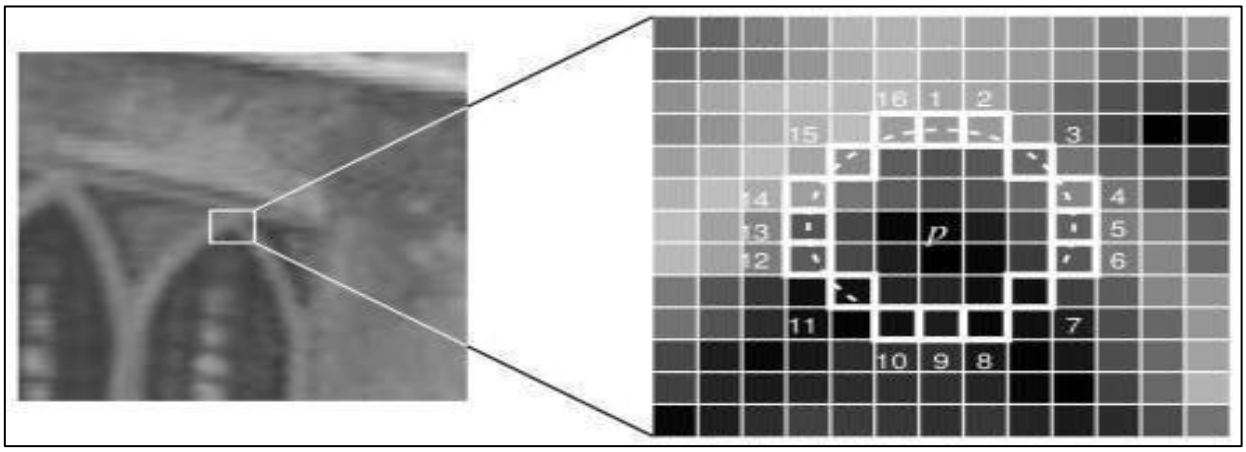


Рисунок 1.2 – Швидке виявлення ключових точок ORB-моделі [6]

Методології машинного навчання, такі як Support Vector Machine (SVM), Random Forest або k-Nearest Neighbors (kNN), використовують навчання класифікатора на маркованих даних, в якому ознаки подаються на вхід до моделі. Загальний процес включає попередню підготовку даних, вилучення ознак і подальшу класифікацію [13, 14].

На рисунку 1.3 наведено приклад класифікації за допомогою kNN = 1.

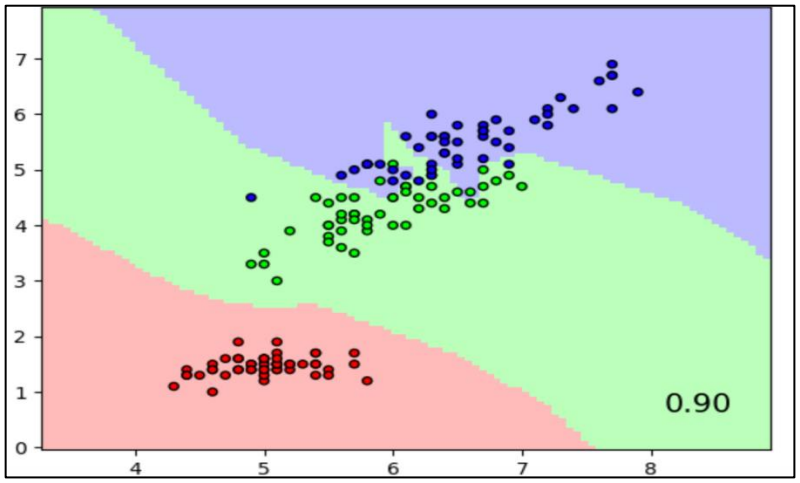


Рисунок 1.3 – Приклад класифікації за допомогою kNN [13]

Значний прорив у цій галузі стався з появою методів глибокого навчання, зокрема згорткових нейронних мереж (CNN), які надали можливість

автоматизовано вилучати важливі ознаки з необроблених зображень. У таких мережах згорткові фільтри формують багаторівневі представлення структури об’єкта, поступово переходячи від простих ліній до складних форм [2, 21].

На рисунку 1.4 наведено архітектуру CNN для виявлення об’єктів за допомогою глибокого навчання.

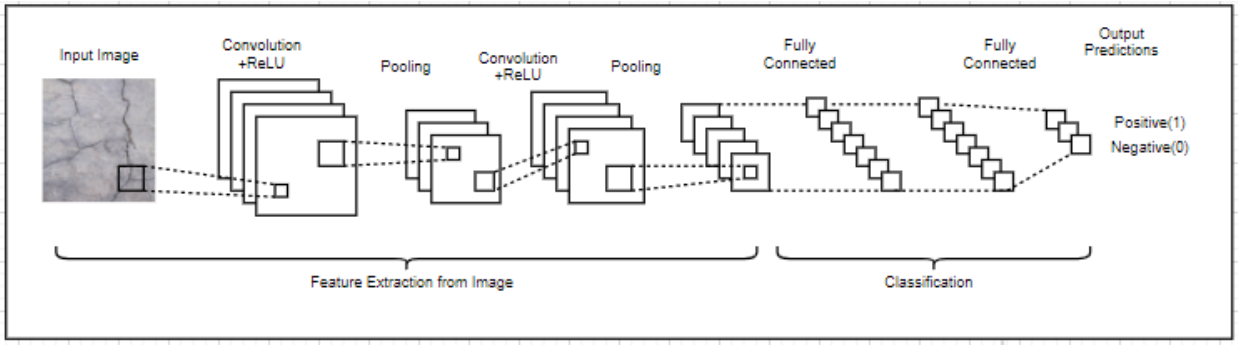


Рисунок 1.4 – Архітектура CNN для виявлення об’єктів за допомогою глибокого навчання [21]

Поява CNN прискорила поширення численних архітектур: LeNet (LeCun et al., 1998) – для простих зображень, тоді як AlexNet, VGG і ResNet – для класифікації складних об’єктів. YOLO (You Only Look Once), SSD та Faster R-CNN – для одночасної локалізації та класифікації об’єктів [7-10]. Ці архітектури поєднують етапи виявлення та класифікації, що надає можливість працювати в режимі реального часу.

Модель YOLO (You Only Look Once) виконує одночасну детекцію та класифікацію об’єктів за один прохід мережі, що забезпечує надзвичайно високу швидкодію. Вона розбиває зображення на сітку та прогнозує межі (bounding boxes) і класи об’єктів для кожної клітинки. Такий підхід забезпечує баланс між точністю та продуктивністю, що робить YOLO ідеальним для задач у реальному часі [8, 9].

Single Shot MultiBox Detector (SSD) – це ще один представник одноетапних (single-shot) детекторів, який виконує детекцію об’єктів на різних масштабах зображення. SSD використовує кілька вихідних шарів для прогнозування об’єктів різного розміру, що забезпечує кращу обробку дрібних

об’єктів порівняно з ранніми версіями YOLO. Проте SSD часто демонструє нижчу точність для великих об’єктів і є менш оптимізованим для роботи в умовах обмежених ресурсів [10].

Faster R-CNN – це двоетапний детектор, який спочатку генерує регіони інтересу (Region Proposal Network, RPN), а потім класифікує кожен запропонований регіон. Такий підхід забезпечує дуже високу точність, особливо для складних сцен із багатьма об’єктами, але вимагає значно більше часу на обробку. Через двоетапну структуру Faster R-CNN не підходить для застосувань у реальному часі, зокрема в системах сортування [7].

У таблиці 1.1 наведено аналіз трьох розглянутих моделей.

Таблиця 1.1 – Порівняння моделей детекції об’єктів

Характеристика	**YOLO**	**SSD**	**Faster R-CNN**
Швидкодія	Висока (реальний час)	Середня	Низька
Точність (mAP)	Висока	Середня	Дуже висока
Складність обчислень	Низька	Середня	Висока
Принцип роботи	Одноетапна (single pass)	Одноетапна (multi-scale)	Двоетапна (RPN + класифікатор)
Обробка малих об’єктів	Посередня	Краща, ніж у YOLO	Висока
Підходить для реального часу	Так	Частково	Ні

Порівняльний аналіз показує, що YOLO є оптимальним вибором для задач автоматичного сортування пляшок, де висока швидкість, прийнятна точність і обмежені обчислювальні ресурси мають першочергове значення. На відміну від Faster R-CNN, YOLO не потребує генерації областей інтересу.

Крім того, на відміну від SSD, YOLO демонструє підвищену стабільність при роботі з середніми і великими об’єктами у відеопотоці. Це робить YOLO особливо добре придатним для розгортання у виробничих середовищах, де швидке прийняття рішень має першорядне значення.

Перевагами глибокого навчання є автоматичне виділення ознак, масштабованість і висока точність. Водночас воно потребує значної кількості анотованих даних, сучасних обчислювальних ресурсів та спеціального часу для навчання [2].

Як наслідок, сучасні методології розпізнавання образів пропонують значний ступінь автоматизації та точності, тим самим обґрунтовуючи використання архітектури типу YOLO в системах реального часу, зокрема з метою виявлення пластикових пляшок на зображеннях.

1.3 Критерії якості та оцінювання результатів класифікації

Ефективність систем комп'ютерного зору залежить не лише від вибору моделі чи алгоритму, але й від якості навчання та здатності точно класифікувати об'єкти в нових даних. Для досягнення цієї мети необхідно встановити формальне правило прийняття рішень і критерії якості, які полегшують об'єктивну оцінку точності та надійності результатів класифікації [1, 2, 11].

1.3.1 Побудова вирішального правила

У задачах класифікації вирішальне правило визначає, до якого класу належить вхідне зображення (або об'єкт на зображенні), залежно від обчисленої ймовірності або активації на вихідному шарі нейромережі [14].

Нехай задано простір ознак $X \subset R^n$, де кожен об'єкт описується вектором ознак x, та множину класів $\{C1, C2, ..., Ck\}$. Тоді вирішальне правило формулюється як [1]:

$$\text{Присвоїти об'єкту клас } C_j, \text{ якщо } P(C_j \mid x) = \max_i P(C_j \mid x).$$

У нейронних мережах ці ймовірності зазвичай апроксимуються функцією softmax [1]:

$$P(C_j \mid x) = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}},$$

де z_i – вихід нейрона для класу C_i.

1.3.2 Метрики якості класифікації

Для оцінки якості класифікації (наприклад, чи правильно зображення містить пляшку чи ні) застосовуються такі метрики: accuracy, precision, recall, F1-score [11, 13].

Розглянемо кожну з наведених метрик більш детально.

Accuracy (загальна точність) – відображає частку всіх правильно класифікованих випадків. Ця метрика є найбільш інтуїтивно зрозумілою, але не підходить для незбалансованих наборів даних із-за великої ймовірності невідповідності значення метрики до реального результату. Наприклад, якщо більшість зображень не містять пляшок (фон), модель може досягти високого значення accuracy, ігноруючи об’єкти [13]. Ця метрика розраховується за допомогою наступної формули:

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN},$$

де TP – істинно позитивні, FP – хибнопозитивні, TN – істинно негативні, FN – хибнонегативні.

Precision (точність позитивних передбачень) – показує, яка частка об’єктів, класифікованих як «пляшка», насправді є ними. Високе значення precision означає низький рівень хибних тривог [2, 11].

Розраховується за допомогою формули:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}.$$

Recall (повнота, чутливість) – визначає, яку частку справжніх пляшок модель змогла виявити. Високе значення recall вказує на здатність не пропускати об’єкти, навіть якщо зменшується точність [2, 11]. Розраховується за наступною формулою:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

F1-score – забезпечує баланс між precision та recall. Це узагальнена метрика, що враховує як помилкові виявлення, так і пропуски [2, 11]. Розраховується за допомогою формули:

$$\text{F1} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}.$$

У контексті задачі автоматичного виявлення пластикових пляшок найдоцільніше буде використовувати комбінацію Recall та Precision: Recall допомагає оцінити здатність моделі знайти всі об’єкти, а Precision – запобігає хибному виявленню інших предметів як пляшок. Також доцільно використати F1-score щоб узагальнено оцінити якість моделі, так як обидва типи помилок однаково критичні [2, 11].

1.3.3 Метрики для об’єктного детектування

На відміну від класичної класифікації, задача виявлення об’єктів вимагає не лише точного визначення класу об’єкта, але й його точної

локалізації на зображенні, тобто окреслення його точного місцезнаходження у вигляді прямокутника (bounding box). Таким чином, якість виявлення залежить як від класу, так і від геометричної точності [3, 9].

Наступні метрики є основними інструментами, що використовуються для оцінки ефективності детектора.

Intersection over Union (IoU) – обчислює ступінь перекриття між передбаченим і справжнім (еталонним) прямокутником навколо об’єкта. Якщо IoU перевищує деяке порогове значення (зазвичай при тестуванні 0.5), вважається, що об’єкт розпізнано коректно [11]. Метрика розраховується за наступною формулою:

$$\mathrm{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}.$$

Confidence threshold – більшість моделей (наприклад, YOLO), повертають для кожної детекції коефіцієнт впевненості (confidence score), що відображає, наскільки модель впевнена у тому, що виявлений об’єкт належить до певного класу [11].

Під час тестування або роботи моделі використовується порогове значення (зазвичай 0.5), нижче якого передбачення вважаються хибними або відсіюються. Для прикладу, якщо значення коефіцієнту 0,8, то модель дуже впевнена, а якщо 0,3 – сумнівається.

Non-Maximum Suppression (NMS) – при детекції, один об’єкт може бути знайдений кілька разів із різними координатами та рівнями впевненості. Цей алгоритм дозволяє зашилити лише значення з найвищим рівнем впевненості, відкидаючи решту прямокутників із меншим значенням confidence [3, 11].

Mean Average Precision (mAP) – зведена метрика, яка поєднує точність (precision) та повноту (recall) для всіх класів і всіх передбачень. Вона враховує як правильність класифікації, так і точність локалізації (через IoU) [11]. Розрахувати можна за формулою:

$$\text{mAP} = \frac{1}{N}\sum_{i=1}^{N} AP_i \ ,$$

де AP_i – площа під кривою Precision–Recall для кожного класу i.

Вибір правильних параметрів для цих метрик і порогів безпосередньо впливає на ефективність моделі в реальному середовищі.

1.4 Висновки до розділу

У першому розділі представлено всебічний огляд галузі автоматичного розпізнавання об’єктів на зображеннях, акцентуючи увагу на задачі ідентифікації пластикових пляшок під час сортування відходів. Обґрунтовано актуальність цієї задачі в контексті автоматизації подібних процесів.

Розглянуті різні підходи до розпізнавання образів, починаючи від класичних геометричних методів і закінчуючи сучасними архітектурами глибокого навчання. Проведено порівняльний аналіз для визначення доцільності використання згорткових нейронних мереж (CNN), зокрема моделі YOLO, яка пропонує баланс точності та швидкості, необхідний для додатків, що працюють у реальному часі.

Крім того, встановлено фундаментальні принципи класифікації та окреслено критерії оцінки продуктивності моделі, включаючи precision, recall, F1-score, IoU та mAP. Такий підхід дозволяє оцінити продуктивність моделі на навчальних даних, а також її практичну ефективність.

Таким чином, перший розділ сформував теоретичне підґрунтя для реалізації практичного рішення в наступних розділах дипломної роботи, а саме – розробки програмного модуля автоматичної ідентифікації пластикових пляшок на основі моделі YOLO в середовищі Python.

2 НЕЙРОННІ МЕРЕЖІ ТА СПОСОБИ ЇХ НАВЧАННЯ

2.1 Нейронні мережі

Сучасні системи комп'ютерного зору все частіше використовують нейронні мережі – математичні моделі, що імітують роботу біологічного мозку шляхом узагальнення та обробки вхідної інформації – для автоматичного розпізнавання об'єктів. Ці алгоритми дозволяють виявляти закономірності в даних, які складно або неможливо формалізувати за допомогою звичайних методів [1-3].

Штучна нейронна мережа (ШНМ) – це обчислювальна модель, що складається з взаємопов'язаних нейронів (вузлів), які забезпечують передачу сигналів один одному через зважені зв'язки [1]. Кожен нейрон отримує вхідний сигнал, обробляє його за допомогою функції активації, а потім передає на наступний етап обробки [17].

На рисунку 2.1 наведено схему такого штучного нейрону.

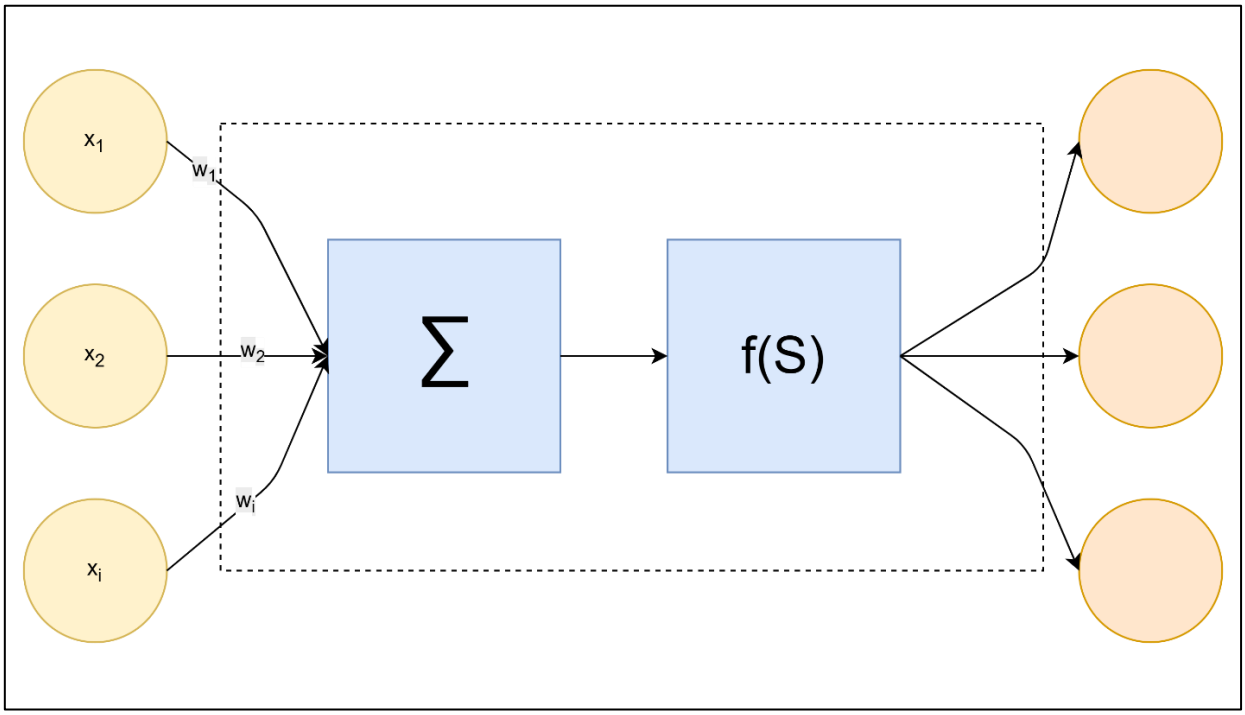


Рисунок 2.1 – Схема одного штучного нейрону

2.1.1 Основні типи нейронних мереж

Вибір архітектури нейронної мережі залежить від характеру завдання, конфігурації структури даних і необхідної обчислювальної потужності.

Розглянемо основні типи нейронних мереж.

Перцептрон та багатошаровий перцептрон (MLP) – це найпростіший тип повнозв'язної мережі, який складається з лінійних зважених сум та нелінійної активації. Всі складні обчислення відбуваються в одному або декількох прихованих шарів, кожен з яких містить різну кількість нейронів. Він підходить для невеликих наборів даних зі структурованим векторним входом, але є непридатним для обробки зображень у їх «сирому» вигляді через втрату просторових залежностей [16].

На рисунку 2.2 зображено перцептрон з одним прихованим шаром.

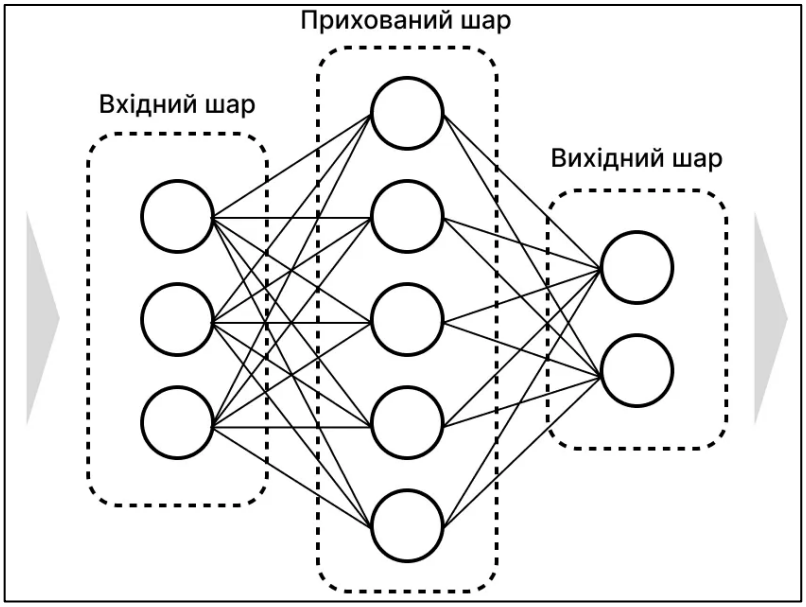


Рисунок 2.2 – Перцептрон з одним прихованим шаром [16]

Згорткова нейронна мережа (CNN) – орієнтована на обробку зображень та відео. У своїй основі використовує згорткові шари, які автоматично виділяють ознаки (контури, текстури, кольори). Серед переваг можна виділити меншу кількість параметрів у порівнянні з повнозв'язними мережами, що дозволяє масштабувати її для великих зображень [5, 7, 18].

Ця архітектура вже стала базовою в задах розпізнавання об’єктів та класифікації зображень. Тому мережі типу CNN є основним вибором для автоматизованого розпізнавання об’єктів, зокрема для задачі розпізнавання пляшок, бо вони одночасно і враховують локальні залежності пікселів, і інваріантні до масштабування, поворотів та зсувів. Також можна виділити високу точність при порівняно невеликому обсязі навчання [5, 7, 18].

На рисунку 2.3 наведена типова архітектура CNN, для класифікації зображень, яка складається зі згорткових шарів та повнозв’язних шарів класичних штучних нейронів.

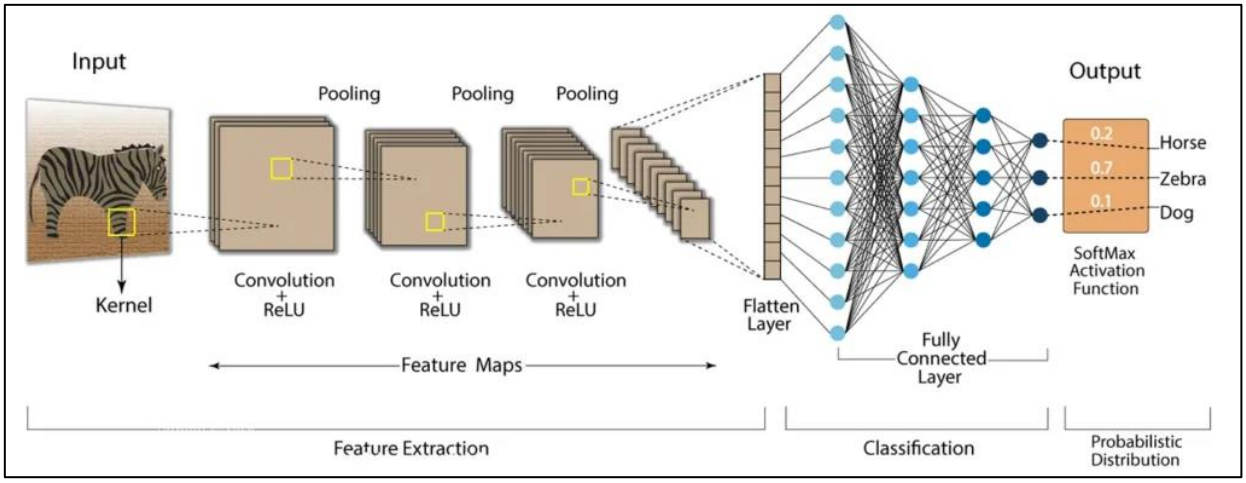


Рисунок 2.3 – Архітектура CNN, для класифікації зображень [18]

Рекурентні нейронні мережі (RNN) – розраховані на послідовності: мова, час, дії. Зберігають пам’ять про попередні стани та використовуються переважно для обробки тексту або часових рядів [14].

У задачах комп’ютерного зору застосовуються в поєднанні з CNN для розпізнавання динаміки відео або контексту в сценах.

Резидуальні та глибокі мережі (ResNet, DenseNet) – побудовані на принципі залишкових зв’язків (residual connections), які дозволяють передавати градієнт через велику кількість шарів. Використовуються в задачах глибокої класифікації зображень, коли потрібно навчити мережу з 50+ шарами [5].

На рисунку 2.4 наведено архітектуру ResNet50.

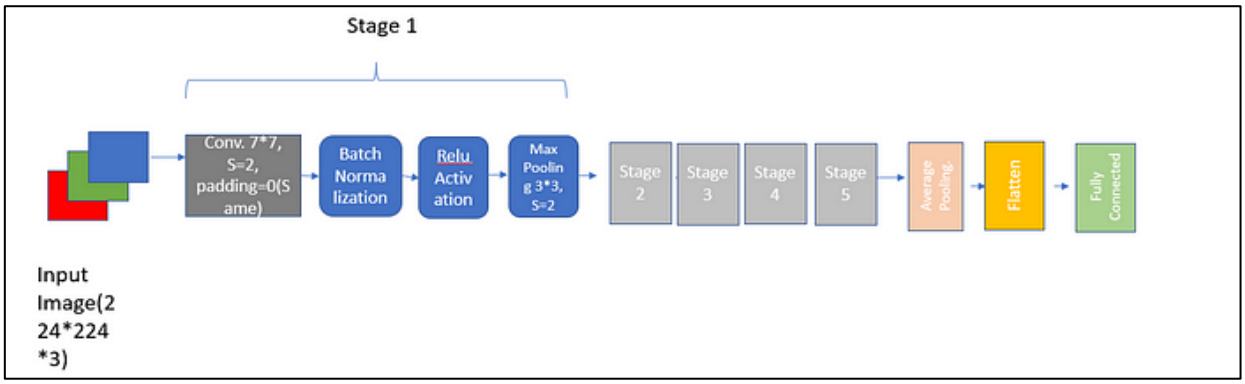


Рисунок 2.4 – Архітектура ResNet50 [5]

Сітки для детекції об’єктів (YOLO, SSD, Faster R-CNN) – дані мережі не лише класифікують зображення, а й в режимі реального часу дозволяють виявляти об’єкти (bounding boxes) та визначати класи [7-10].

Беручи за основу аналіз методів з першого розділу, оптимальною є архітектура YOLO через високу швидкодію та прийнятну точність у задачах розпізнавання об’єктів [8].

YOLO розбиває вхідне зображення на сітку ($S \times S$) і для кожної клітинки одночасно передбачає:

– координати обмежувального прямокутника (bounding box): x, y, w, h;

– впевненість (confidence score): P(object) * IoU;

– ймовірність належності до кожного класу: P(classi | object).

Це дозволяє одночасно виявляти декілька об’єктів різних класів на одному зображенні за один прохід моделі. На рисунку 2.5 наведено типову архітектуру YOLO.

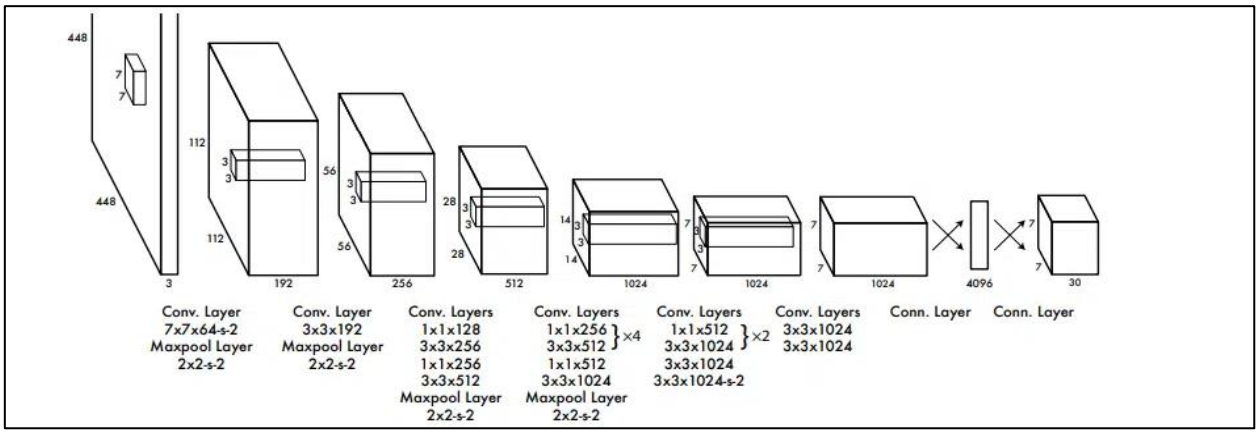


Рисунок 2.5 – Архітектура YOLO [8]

Нейронні мережі – це клас моделей, які демонструють універсальність, масштабованість та адаптивність, що дозволяє ефективно вирішувати складні завдання розпізнавання образів без необхідності програмування явних правил. CNN та YOLO найбільш оптимальне рішення для ідентифікації та сортування пляшок, оскільки вони демонструють високу здатність досягати прийнятної точності та працювати в режимі реального часу [5, 8, 18].

2.2 Архітектура CNN

Згорткові нейронні мережі стали фактично стандартом у застосунках комп'ютерного зору, включаючи розпізнавання та класифікацію зображень. Архітектура цих систем розроблена спеціально для обробки двовимірних структур даних, таких як фотографії або відеокадри. Ці системи дозволяють автоматично вивчати релевантні ознаки зображення, усуваючи таким чином необхідність ручного програмування [1, 2, 5].

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

Згортковий шар (Convolutional Layer) – це ключовий шар мережі, який виконує операцію згортки, що передбачає накладення фільтра (ядра) на ділянки зображення з метою виділення характерних ознак (країв, контурів, текстур) [1]. Обчислюється за формулою:

$$\mathrm{s(t)} = (\mathrm{x} * \mathrm{w})(\mathrm{t}) = \sum_{\tau} x(\tau) * w(t - \tau),$$

де $x(\tau)$ – значення вхідного сигналу (зображення);

$w(t - \tau)$ – ядро згортки (фільтр), що «ковзає» по зображенню;

t – поточна позиція згортки;

s(t) – результат згортки в точці t [1].

У двовимірному випадку згортка застосовується до зображення з певним кроком і може мати відступи.

Після згортки до результату застосовується нелінійна функція активації, яка дозволяє мережі моделювати складні залежності.

Найпоширенішою є функція ReLU:

$$f(x) = \max(0, x),$$

де x – вхідне значення від згорткового шару;

f(x) – вихідне значення після застосування ReLU [2].

ReLU обнуляє всі від'ємні значення, залишаючи позитивні без змін. Це робить нейронну мережу здатною моделювати нелінійні залежності, які важко відобразити за допомогою лінійних перетворень.

Шар субдискретизації (Pooling Layer) зменшує розмірність простору ознак (наприклад, перетворює 32×32 у 16×16), залишаючи найважливішу інформацію. Найчастіше використовується максимальне об'єднання (Max Pooling):

$$y = \max(x_{i,j}) - \text{ для заданої області},$$

де $x_{i,j}$ – елементи вхідної області;

y – результат операції max pooling – найбільше значення серед $x_{i,j}$ [2, 5].

Цей шар допомагає зменшити обсяг обчислень, запобігти перенавчанню та забезпечує інваріантність до невеликих зсувів і деформацій.

Повнозв'язні шари (FC) – на завершальному етапі ознаки, отримані на згорткових шарах, перетворюються у вектор і передаються в нейронну класифікаційну модель. Тут відбувається логічне прийняття рішення, а саме до якого класу належить зображення. Обчислюється за допомогою формули:

$$y = f(Wx + b),$$

де x – вхідний вектор ознак (flattened output з попереднього шару);

W – матриця ваг, яка визначає важливість кожної ознаки;

b – вектор зміщень (bias);

f – функція активації;

y – вектор вихідних значень (ймовірностей класів) [1].

Останній FC-шар зазвичай має стільки нейронів, скільки є класів у задачі, й повертає ймовірності належності до кожного класу.

Функція втрат (Loss Function) – під час навчання мережа мінімізує величину, яка відображає різницю між передбаченим результатом і правильним. Для задач класифікації використовується категоріальна крос-ентропія:

$$\mathrm{L}(y, \hat{y}) = -\sum_{i=1}^{K} y_i * \log(\hat{y}_i)$$

де y_i – істинна мітка класу (1, якщо зображення належить класу i, інакше 0);

$\hat{y}_i$ – передбачена ймовірність класу i на виході моделі;

K – кількість класів;

L – втрати, які мінімізуються під час навчання [1, 2].

Це забезпечує математичне обґрунтування оновлення ваг через зворотне поширення помилки (backpropagation).

Згорткові нейронні мережі навчаються шляхом ітеративного оновлення ваг за допомогою класичного алгоритму градієнтного спуску. Процес навчання складається з декількох послідовних етапів.

Розглянемо ці етапи.

Forward Pass – дані проходять через всі шари мережі, і модель генерує прогноз.

Обчислення втрат – за допомогою функції втрат обчислюється помилка між прогнозом та істинним класом.

Backpropagation – обчислюється градієнт втрат по кожному параметру мережі.

Оновлення ваг – ваги змінюються за формулою:

$$w_{new} = \mathrm{w}_{old} - \eta * \frac{dL}{dw},$$

де *w* – ваговий коефіцієнт;

η – швидкість навчання (learning rate);

dL/*dw* – градієнт функції втрат по параметру [1, 2].

Цикл повторюється для всіх прикладів навчальної вибірки протягом кількох епох.

Архітектура згорткових нейронних мереж дозволяє ефективно вирішувати задачі розпізнавання складних об’єктів, зокрема у промислових умовах сортування.

Ретельна побудова шарів, правильний вибір гіперпараметрів і грамотне навчання мережі забезпечують високу точність класифікації та виявлення, що критично важливо для автоматизованих ліній, де рішення приймаються без участі людини [5, 12, 18].

2.3 Навчання нейронних мереж

Навчання є ключовим етапом у розвитку нейронної мережі, оскільки безпосередньо впливає на її здатність узагальнювати дані та формулювати точні рішення, коли вона стикається з новими прикладами. У процесі навчання мережа налаштовує свої вагові коефіцієнти, щоб мінімізувати розбіжності між прогнозованими та фактичними результатами. Це досягається за допомогою оптимізації функції втрат, яка кількісно оцінює розбіжність між прогнозом мережі та фактичним результатом [1, 2].

2.3.1 Типи навчання

Залежно від характеру доступних даних і цілей розв’язуваної задачі розрізняють кілька основних типів навчання.

Навчання з учителем (supervised learning) – найбільш поширений тип, коли кожному вхідному зображенню відповідає правильна мітка (наприклад, «пляшка», «банка»). Мережа навчається зіставляти вхід і правильний вихід. Саме цей тип використовується у задачі сортування об’єктів [1, 2].

Навчання без учителя (unsupervised learning) – мережа не має доступу до міток і намагається сама знайти структуру в даних (кластеризація, зниження розмірності). Використовується рідше для розпізнавання зображень [2, 14].

Навчання з підкріпленням (reinforcement learning) – актуальне для автономних систем, де агент навчається шляхом взаємодії з навколишнім середовищем [2].

Transfer learning – метод переносу знань із попередньо навченої моделі. Наприклад, можна взяти CNN, навчений на ImageNet, і донавчити його для виявлення пластикових пляшок [5, 12].

2.3.2 Функції втрат

Навчання відбувається шляхом мінімізації функції втрат – кількісного показника помилки.

У розділі 2.2 вже була розглянута категоріальна крос-ентропія, яка є найпоширенішою у класифікації. Також застосовується бінарна крос-ентропія, яка призначена для задач із двома класами.

Ще однією функцією є MSE (Mean Squared Error), але вона більше використовується у регресії, бо менш ефективна в класифікації.

Функція втрат формує градієнт, який потім використовується для оновлення вагів під час зворотного поширення помилки [1, 2].

2.3.3 Оптимізація та оновлення ваг

Процес оновлення вагів здійснюється за допомогою методів оптимізації. Розглянемо найпоширеніші з них:

– Stochastic Gradient Descent (SGD) – класичний метод, оновлення параметрів здійснюється на основі похідної функції втрат [2, 5];
– Adam – адаптивний алгоритм, який поєднує переваги SGD та RMSprop [2];
– Learning rate schedule – дозволяє уникнути застрягання у локальних мінімумах, так як швидкість навчання може змінюватися динамічно [5].

Оновлення ваг відбувається за формулою наведеною у розділі 2.2.

2.3.4 Перенавчання та регуляризація

У процесі навчання модель може запам’ятати всі особливості тренувального набору, включно з шумами, і втратити здатність узагальнювати, що призводить до перенавчання (overfitting). Його можна ідентифікувати за низкою ознак:

– висока точність на тренувальних даних;
– погані результати на нових прикладах [2].

Для боротьби з перенавчанням використовується низка підходів:

– Dropout – випадкове відключення частини нейронів під час навчання [2];
– Early stopping – зупинка тренування, коли метрика на валідації перестає покращуватись;
– L1 / L2-регуляризація – додаткові штрафи за великі ваги в моделі [5].

2.3.5 Якість навчання (generalization)

Мета навчання полягає не лише у відтворенні правильних відповідей на заданих прикладах, а й у розвитку здатності моделі до узагальнення. Для досягнення цієї мети дані сегментуються на окремі категорії:

– Training set – для навчання;
– Validation set – для підбору гіперпараметрів;
– Test set – для підсумкової оцінки [1, 2, 12].

Хороша модель демонструє:

– стабільні показники на validation/test;
– низьке відхилення між train/test loss;
– здатність адаптуватися до нових об’єктів.

2.4 Висновки до розділу

У цьому розділі детально розглянуто архітектуру згорткових нейронних мереж (CNN), яка є основним інструментом, що використовуються для вирішення проблем комп’ютерного зору. Послідовно описано ключові структурні компоненти мережі: згорткові шари, pooling, функції активації, повнозв’язані шари, а також функції втрат і способи оптимізації. Наголошено на особливостях forward- та backward-проходів у поєднанні з алгоритмами оновлення ваг. Ці особливості полегшують навчання моделей на значних масивах даних.

Особливу увагу приділено особливостям навчання нейронних мереж, типам навчання, ролі функцій втрат, таких як крос-ентропія, та методам регуляризації. Це дає комплексне розуміння принципів, за якими нейромережеві моделі здатні узагальнювати вхідні дані, тим самим мінімізуючи похибку та підвищуючи точність класифікації.

Таким чином, було сформульовано теоретичну основу, яка слугує фундаментом для подальшої реалізації практичної частини дипломної роботи.

3 РІШЕННЯ ДЛЯ РОЗРОБКИ СИСТЕМИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ПЛЯШОК

3.1 Функціональні та технічні вимоги до системи

Розроблена система має на меті автоматизувати процес ідентифікації пластикових пляшок на зображеннях, що є частиною задачі сортування сміття.

Основні функціональні вимоги включають:

– визначення наявності пластикових пляшок на зображеннях;
– підрахунок кількості виявлених об’єктів;
– завантаження/обробка зображень із теки, вибраної користувачем;
– виведення результатів.

Нефункціональні вимоги включають:

– використання мови програмування Python;
– застосування моделей глибокого навчання (YOLO);
– інтеграцію з OpenCV, PIL, Tkinter для обробки зображень та інтерфейсу;
– можливість запуску на ПК з підтримкою Python та середовища Windows.

Програма розроблялась у середовищі PyCharm Community Edition, що забезпечує зручну навігацію, автодоповнення коду та базовий рефакторинг.

3.2 Використані інструменти

Для реалізації програмної частини системи було використано низку бібліотек мови програмування Python [22, 23], які забезпечують завантаження та застосування моделі нейронної мережі, обробку зображень, роботу з файловою системою та створення графічного інтерфейсу користувача.

Ключовим компонентом є бібліотека OpenCV [19], яка використовується для роботи з вхідними зображеннями, а також для завантаження попередньо натренованої моделі YOLO [8, 9] і виконання основних операцій комп'ютерного зору. Для ефективної обробки числових даних та масивів застосовується бібліотека NumPy [24], яка забезпечує оптимальні математичні обчислення, необхідні при аналізі результатів згорткових операцій.

Для побудови графічного інтерфейсу користувача використано Tkinter, вбудовану в Python бібліотеку, яка дозволяє створювати прості інтерактивні вікна [27]. У поєднанні з нею застосовано бібліотеку Pillow (PIL), яка відповідає за відкриття та відображення зображень у графічному вікні інтерфейсу [26].

Додатково використовувалася бібліотека os для взаємодії з файловою системою, що дозволяє обирати директорії, зчитувати зображення та передавати їх для подальшої обробки [28]. Для графічної візуалізації результатів роботи моделі, зокрема – для відображення знайдених об'єктів на зображенні та статистичної інформації, застосовується бібліотека Matplotlib [25].

3.3 Функції системи та їх призначення

Програма логічно розділена на 3 основні функції. Розглянемо їх. Детально ознайомитися з повною версією коду можна в додатку А.

Browse_directory() – відкриває вікно вибору папки з зображеннями, після чого викликає функцію обробки вмісту директорії (рис. 3.1).

```
26 def browse_directory():
27     global directory_path
28     directory_path = filedialog.askdirectory()
29     if directory_path:
30         process_images_in_directory()
```

Рисунок 3.1

Process_images_in_directory() – проходить по кожному зображенню в директорії, виводить його на екран, викликає функцію process_image() для аналізу (рис. 3.2).

```
33 def process_images_in_directory():
34     image_files = [f for f in os.listdir(directory_path) if f.lower().endswith(('.png', '.jpg', '.jpeg'
           , '.gif', '.bmp'))]
35     if not image_files:
36         messagebox.showinfo("Warning", "There are no images in the selected folder.")
37     else:
38         for image_file in image_files:
39             image_path = os.path.join(directory_path, image_file)
40             image = Image.open(image_path)
41             photo = ImageTk.PhotoImage(image)
42             label.config(image=photo)
43             label.image = photo
44             process_image(image_path)
```

Рисунок 3.2

Process_image(image_path, classes) – виконує завантаження моделі YOLO, обробку зображення та визначення об’єктів, виводить результат на графіки та в інтерфейс (рис. 3.3).

```
47 def process_image(image_path):
48     image = cv2.imread(image_path)
49     height, width, _ = image.shape
50     blob = cv2.dnn.blobFromImage(image, 1 / 255, (416, 416), swapRB=True, crop=False)
51     net.setInput(blob)
52     outs = net.forward(net.getUnconnectedOutLayersNames())
53 
54     plastic_bottles = []
55 
56     for out in outs:
57         for detection in out:
58             scores = detection[5:]
59             class_id = np.argmax(scores)
60             confidence = scores[class_id]
61 
62             if confidence > confidence_threshold and classes[class_id] == 'bottle':
63                 center_x = int(detection[0] * width)
64                 center_y = int(detection[1] * height)
65                 w = int(detection[2] * width)
66                 h = int(detection[3] * height)
67 
68                 x = int(center_x - w / 2)
69                 y = int(center_y - h / 2)
70 
71                 plastic_bottles.append((x, y, w, h))
72 
73     for box in plastic_bottles:
74         x, y, w, h = box
75         cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
76 
77     bottle_count = len(plastic_bottles)
78     result_label.config(text=f"Found plastic {bottle_count} bottles in {image_path}")
79     plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
80     plt.title('Plastic Bottles Detection')
81     plt.axis('off')
82     plt.show()
```

Рисунок 3.3

3.4 Керівництво користувача

Після запуску з’являється головне вікно з кнопкою вибору директорії (рис. 3.4).

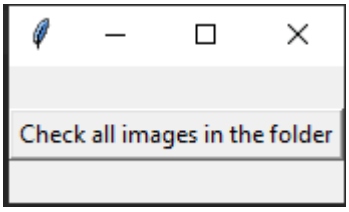


Рисунок 3.4 – Початкове вікно

Після того, як користувач натискає на кнопку «Check all images in the folder», система відображає інтерфейс вибору зображення з пристрою користувача (рис. 3.5).

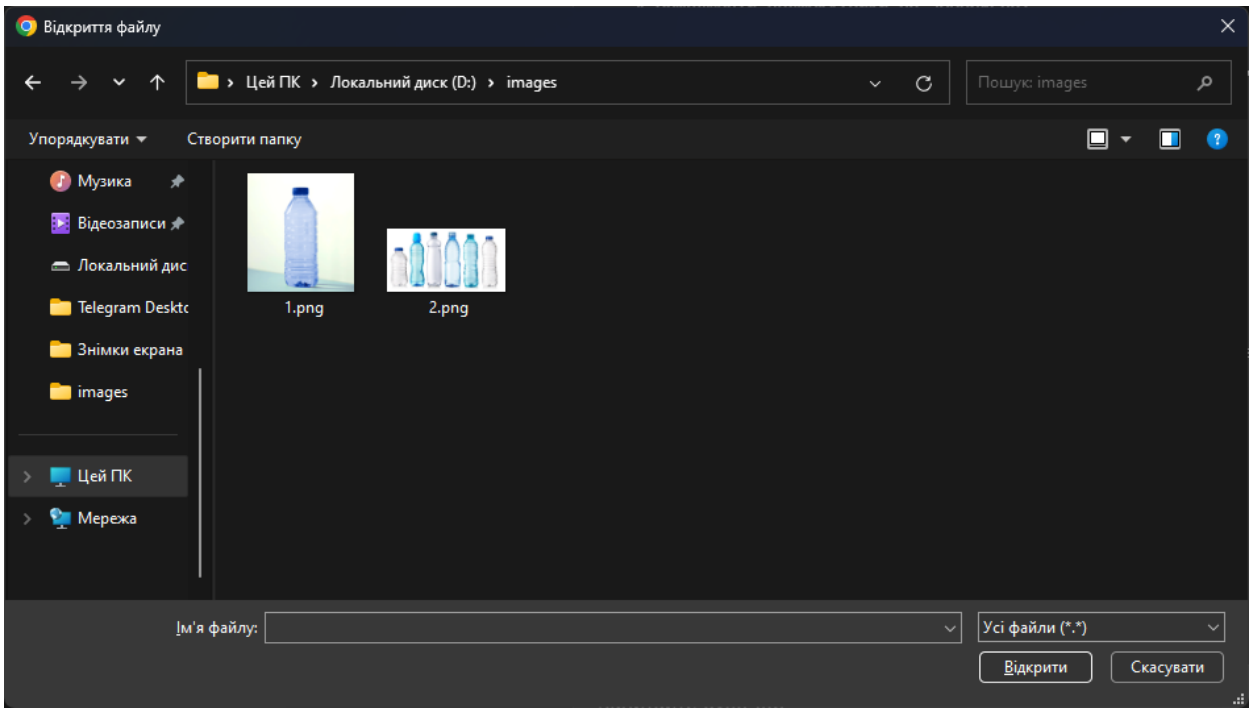


Рисунок 3.5 – Вибір зображення

Програма завантажить кожне зображення та застосує модель YOLOv3 для виявлення об’єктів класу bottle. Після обробки, виведе зображення з виділеними прямокутниками навколо пляшок та кількість знайдених пляшок

у відповідному підписі. Результат для кожного зображення буде відображено у новому вікні (рис. 3.6 та 3.7).

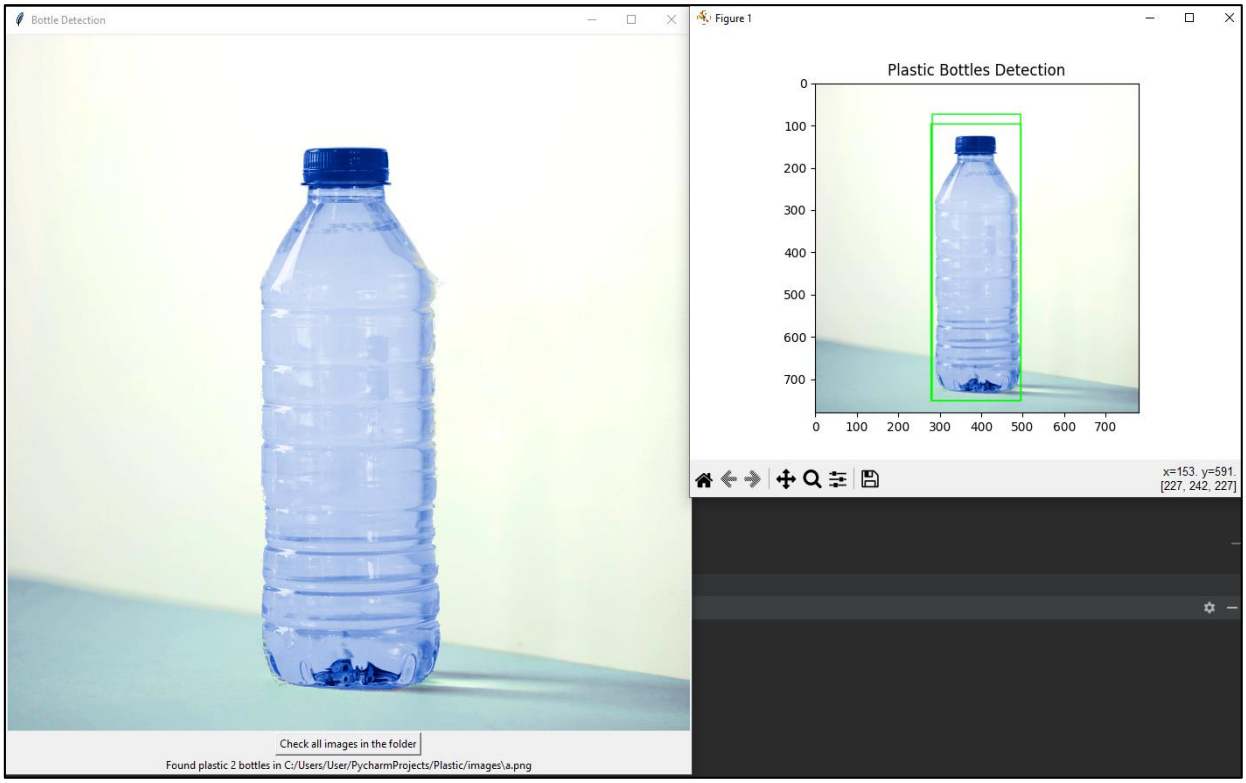


Рисунок 3.6 – Вікна з висновком щодо одного із зображень

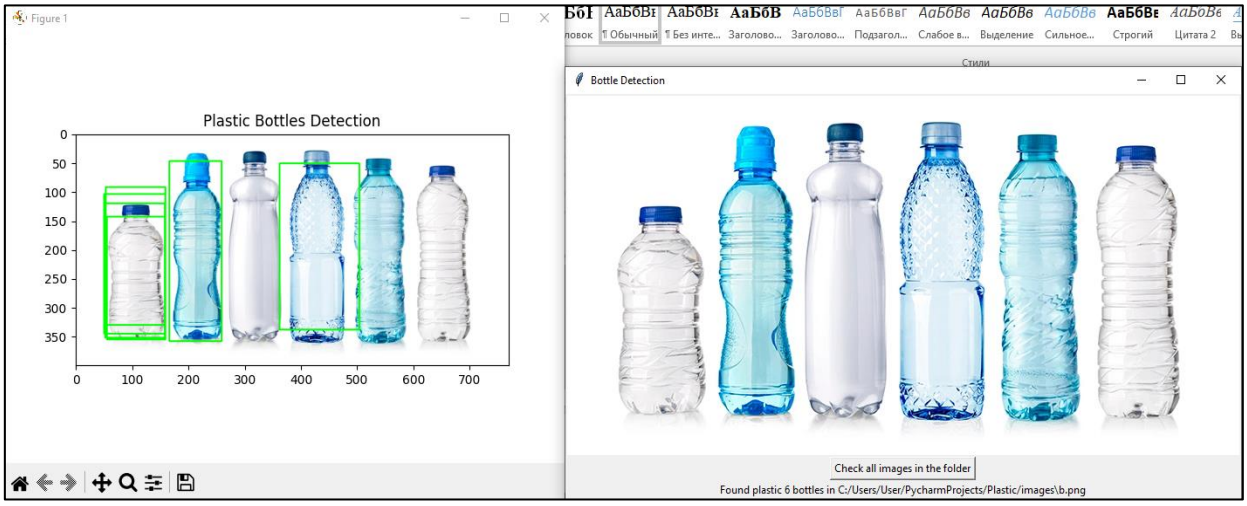


Рисунок 3.7 – Зображення типів пляшок

Під час використання програми користувачі можуть зіткнутися з помилками, характерними як для некоректних налаштувань середовища, так і для обмежень моделі.

Найпоширенішою помилкою є відсутність необхідних файлів моделі, таких як yolov3.cfg, yolov3.weights або coco.names. У такому випадку система повідомить про критичну помилку запуску, і користувач повинен переконатися, що ці файли доступні в теці yolo/.

Іншою типовою проблемою є відсутність зображень у вибраному каталозі або використання файлів у непідтримуваних форматах. У таких випадках користувачеві видається інформаційне повідомлення з пропозицією вибрати відповідну папку або перевірити розширення зображення.

Крім того, за певних обставин модель не виявляє об’єкти на зображенні, незважаючи на їхню наявність. Це явище може бути пов’язане з нездатністю моделі адаптуватися до конкретних умов, таких як неоптимальне освітлення, нестандартні ракурси або часткове перекриття об’єктів. У таких ситуаціях рекомендується протестувати інші приклади зображень або вручну відрегулювати параметри у коді, щоб підвищити чутливість.

Помилки, пов’язані з відображенням зображень або графіків, можуть виникати, якщо на комп’ютері не встановлені відповідні бібліотеки Python. Для усунення таких помилок рекомендується перевірити всі залежності та/або встановити їх за допомогою pip install.

3.5 Результати роботи програми

Під час тестування було оброблено різні типи зображень: як із пляшками, так і без них. У середньому, модель виявляла 80 % пластикових пляшок, присутніх у кадрі. Зображення без пляшок не давали хибних спрацювань (рис. 3.8).

Виявлено, що у випадках великої кількості об’єктів або накладень модель може розпізнавати одну пляшку кілька разів. Це пов’язано з особливостями налаштувань порогу confidence і threshold [8, 9].

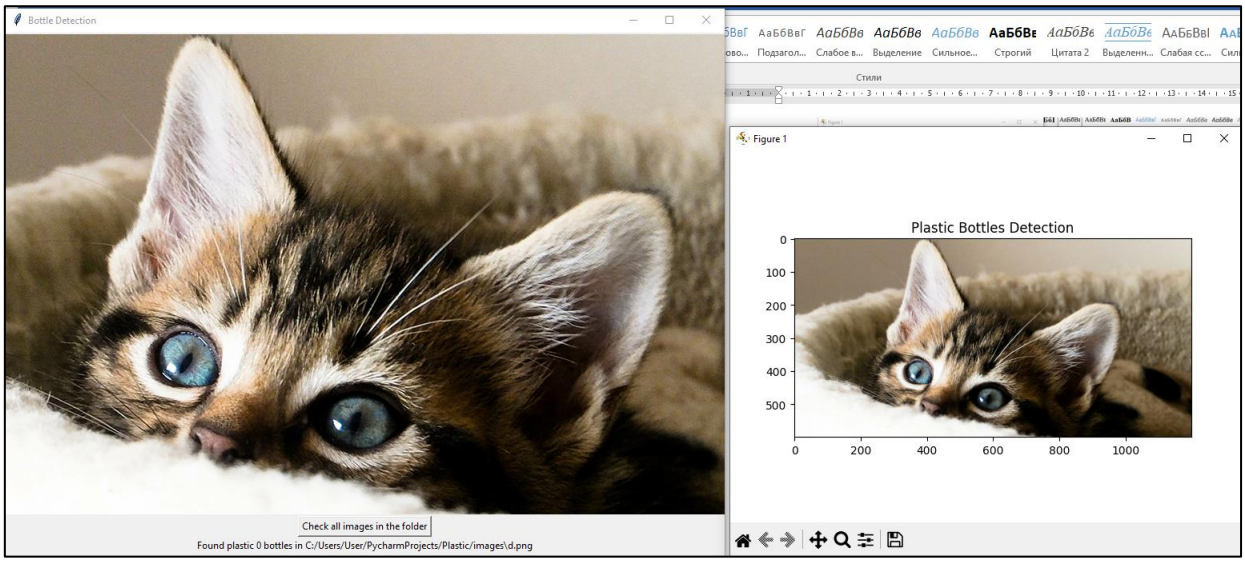


Рисунок 3.8 – Приклад обробки зображення без виявлених об’єктів

3.6 Висновки до розділу

У даному розділі реалізовано програмну систему для виявлення пластикових пляшок на зображеннях, що зчитуються з вибраної директорії. Основні функції програми включають обробку серії зображень, застосування згорткової нейронної мережі YOLOv3 для детекції об’єктів класу bottle, а також виведення результатів у зручному для користувача вигляді. Інтерфейс побудовано на базі бібліотеки Tkinter, що дозволяє навіть недосвідченому користувачу легко взаємодіяти з системою.

Результати виводяться як і число (кількість знайдених пляшок), так і графічно – зображення з виділеними об’єктами подається у новому вікні. Це дозволяє швидко оцінити точність роботи моделі та її відповідність реальному змісту зображення. Завдяки використанню попередньо натренованої моделі YOLOv3 відпала потреба в повторному навчанні, що значно спрощує запуск системи в умовах обмежених ресурсів.

Загалом, реалізоване програмне забезпечення демонструє ефективну роботу при детекції об’єктів у тестових прикладах і може бути використане для навчальних цілей, прототипування або пілотного впровадження в автоматизовані системи сортування відходів.

ВИСНОВКИ

У ході виконання даної кваліфікаційної роботи було розроблено програмну систему автоматичної ідентифікації пластикових пляшок на зображеннях із використанням методів глибинного навчання. Система реалізована на мові програмування Python з використанням моделі YOLOv3 та бібліотек OpenCV, NumPy, Pillow, Matplotlib і Tkinter. Основний функціонал програми включає можливість обробки зображень із вибраної директорії, виявлення пластикових пляшок, виведення кількості знайдених об’єктів і візуалізацію результатів через графічний інтерфейс користувача.

Розроблений застосунок відповідає вимогам до навчального або пілотного індустріального програмного продукту з візуальною інтерпретацією результатів роботи.

У відповідності до поставленої мети було реалізовано програму виявлення пластикових пляшок, що поєднує ефективну архітектуру згорткової нейронної мережі та простий інтерфейс для користувача.

У відповідності до поставлених задач були виконані наступні етапи роботи:

- проведено огляд методів комп’ютерного зору та аналіз існуючих моделей об’єктного розпізнавання (YOLO, SSD, Faster R-CNN), обґрунтовано доцільність використання YOLO для задачі виявлення пляшок;
- визначено метрики якості класифікації та сформульовано критерії ефективності моделі;
- реалізовано програмну систему, яка використовує попередньо натреновану модель YOLOv3, обробляє зображення, виявляє об’єкти, виводить графічний результат і текстову інформацію;
- створено графічний інтерфейс, який забезпечує взаємодію користувача із системою, та надано інструкцію з використання;

– проведено тестування на підготовлених прикладах та проаналізовано отримані результати.

Результати, отримані в процесі реалізації даного проєкту, демонструють ефективність поєднання глибинного навчання з інструментами Python для вирішення прикладних задач комп’ютерного зору, що підтверджує можливість практичного використання розробленої системи в умовах сортування твердих побутових відходів.

ПЕРЕЛІК ПОСИЛАНЬ

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

2. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 775 p.

3. Zhang Z. A Survey on Object Detection in the Age of Deep Learning. *arXiv preprint*. URL: https://arxiv.org/pdf/1506.02640 (дата звернення: 21.09.2023).

4. A Comprehensive Guide to Types of Neural Networks. *Digital Vidya*. URL: https://www.digitalvidya.com/blog/types-of-neural-networks/ (дата звернення: 10.09.2023).

5. Sharma T. Detailed explanation of Residual Network ResNet50 CNN Model. *Medium*. URL: https://medium.com/@sharma.tanish096/detailed-explanation-of-residual-network-resnet50-cnn-model-106e0ab9fa9e (дата звернення: 02.08.2023).

6. A Comparison of SIFT, SURF and ORB on OpenCV. *Medium*. URL: https://mikhail-kennerley.medium.com/a-comparison-of-sift-surf-and-orb-on-opencv-59119b9ec3d0 (дата звернення: 11.10.2023).

7. Getting Started with R-CNN, Fast R-CNN, and Faster R-CNN. *MathWorks*. URL: https://www.mathworks.com/help/vision/ug/getting-started-with-r-cnn-fast-r-cnn-and-faster-r-cnn.html (дата звернення: 24.09.2023).

8. Redmon J., Divvala S., Girshick R., Farhadi A. You Only Look Once: Unified, Real-Time Object Detection. 2016. *arXiv preprint*. https://doi.org/10.48550/arXiv.1506.02640 (дата звернення: 17.08.2023).

9. Redmon J. YOLOv3: An Incremental Improvement. URL: https://www.researchgate.net/publication/324387691_YOLOv3_An_Incremental_Improvement (дата звернення: 15.10.2023).

10. 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: 17.08.2023).

11. Understanding the mAP Evaluation Metric for Object Detection. *Medium*. URL: https://medium.com/@timothycarlen/understanding-the-map-evaluation-metric-for-object-detection-a07fe6962cf3 (дата звернення: 10.07.2023).

12. Object Detection with CNNs. *EasyODM*. URL: https://easyodm.tech/object-detection-with-cnns/ (дата звернення: 10.07.2023).

13. Machine Learning: K-Nearest Neighbors Classifier. *Medium*. URL: https://medium.com/@hareesha1906/ml-k-nearest-neighbors-classifier-e20783f30f19 (дата звернення: 11.09.2023).

14. Classical Machine Learning Algorithms. *Evergreens*. URL: https://evergreens.com.ua/ua/articles/classical-machine-learning.html (дата звернення: 11.09.2023).

15. Algorithm Canny. *Wikipedia*. URL: https://uk.wikipedia.org/wiki/Алгоритм_Кенні (дата звернення: 17.08.2023).

16. Множинний персептрон (MLP) у машинному навчанні. *The Transmitted*. URL: https://thetransmitted.com/adlucem/shho-take-mlp-u-mashynnomu-navchanni/ (дата звернення: 10.09.2023).

17. Штучний нейрон. *Wikipedia*. URL: https://uk.wikipedia.org/wiki/Штучний_нейрон (дата звернення: 10.09.2023).

18. Використовуємо CNN для обробки зображень. *DOu*. URL: https://dou.ua/forums/topic/48368/ (дата звернення: 11.08.2023).

19. About OpenCV. *OpenCV*. URL: https://opencv.org/about/ (дата звернення: 14.10.2023).

20. Koul A., Ganju S., Kasam M. Practical Deep Learning for Cloud, Mobile, and Edge: Real-World AI & Computer-Vision Projects Using Python, Keras & TensorFlow. Beijing : O'Reilly Media Inc., 2019. 620 p.

21. Nguyen M. T., Nguyen Q. N., Pham H. V., Nguyen, H. V. Automatic detection of sewer defects via hierarchical deep learning. *ResearchGate*. URL:

https://www.researchgate.net/publication/367253099_Automatic_detection_of_sewer_defects_via_hierarchical_deep_learning (дата звернення: 14.10.2023)

22. Python Programming Language. *Wikipedia*. URL: https://uk.wikipedia.org/wiki/Python (дата звернення: 11.09.2023).

23. Python Software Foundation. *Python 3 Documentation*. URL: https://docs.python.org/3/ (дата звернення: 14.10.2023).

24. NumPy Documentation. *NumPy Developers*. URL: https://numpy.org/doc/ (дата звернення: 14.10.2023).

25. Matplotlib Documentation. *Matplotlib Developers*. URL: https://matplotlib.org/stable/contents.html (дата звернення: 14.10.2023).

26. Python Imaging Library (Pillow). URL: https://pillow.readthedocs.io/en/stable/ (дата звернення: 14.10.2023).

27. Tkinter Documentation. *Python Docs*. URL: https://docs.python.org/3/library/tkinter.html (дата звернення: 14.10.2023).

28. OS – Miscellaneous operating system interfaces. *Python Docs*. URL: https://docs.python.org/3/library/os.html (дата звернення: 14.10.2023).

ДОДАТОК А

Програмна реалізація додатку для пошуку пластикових пляшок

```
import cv2
import numpy as np
import tkinter as tk
from tkinter import filedialog, messagebox
from PIL import Image, ImageTk
import os
import matplotlib.pyplot as plt

# === Глобальні змінні ===
directory_path = None
model_cfg_path = 'yolo/yolov3.cfg'
model_weights_path = 'yolo/yolov3.weights'
class_names_path = 'yolo/coco.names'
confidence_threshold = 0.5

# === Перевірка наявності необхідних файлів ===
if not os.path.exists(model_cfg_path) or not os.path.exists(model_weights_path) or not os.path.exists(class_names_path):
    raise FileNotFoundError("Не знайдено файли моделі YOLO (cfg, weights або coco.names). Перевірте шляхи.")

# === Завантаження моделі один раз ===
net = cv2.dnn.readNet(model_cfg_path, model_weights_path)
with open(class_names_path, 'r') as f:
    classes = f.read().strip().split('\n')
```

```
# === Функція перегляду директорії ===
def browse_directory():
    global directory_path
    directory_path = filedialog.askdirectory()
    if directory_path:
        process_images_in_directory()

# === Обробка всіх зображень у директорії ===
def process_images_in_directory():
    image_files = [f for f in os.listdir(directory_path) if f.lower().endswith(('.png', '.jpg', '.jpeg', '.gif', '.bmp'))]
    if not image_files:
        messagebox.showinfo("Warning", "There are no images in the selected folder.")
    else:
        for image_file in image_files:
            image_path = os.path.join(directory_path, image_file)
            image = Image.open(image_path)
            photo = ImageTk.PhotoImage(image)
            label.config(image=photo)
            label.image = photo
            process_image(image_path)

# === Обробка одного зображення ===
def process_image(image_path):
    image = cv2.imread(image_path)
    height, width, _ = image.shape

    # Підготовка зображення для нейронної мережі
    blob = cv2.dnn.blobFromImage(image, 1 / 255, (416, 416), swapRB=True,
```

```
crop=False)
    net.setInput(blob)
    outs = net.forward(net.getUnconnectedOutLayersNames())

    plastic_bottles = []

    # Аналіз виводу нейромережі
    for out in outs:
        for detection in out:
            scores = detection[5:]
            class_id = np.argmax(scores)
            confidence = scores[class_id]

            # Фільтрація за класом 'bottle' з порогом довіри
            if confidence > confidence_threshold and classes[class_id] == 'bottle':
                center_x = int(detection[0] * width)
                center_y = int(detection[1] * height)
                w = int(detection[2] * width)
                h = int(detection[3] * height)

                x = int(center_x - w / 2)
                y = int(center_y - h / 2)

                plastic_bottles.append((x, y, w, h))

    # Відображення знайдених пляшок на зображенні
    for box in plastic_bottles:
        x, y, w, h = box
        cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
```

```
    # Виведення результату
    bottle_count = len(plastic_bottles)
    result_label.config(text=f"Found plastic {bottle_count} bottles in {image_path}")
    plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
    plt.title('Plastic Bottles Detection')
    plt.axis('off')
    plt.show()

# === Побудова графічного інтерфейсу ===
root = tk.Tk()
root.title("Bottle Detection")

label = tk.Label(root)
label.pack()

browse_directory_button = tk.Button(root, text="Check all images in the folder", command=browse_directory)
browse_directory_button.pack()

result_label = tk.Label(root, text="")
result_label.pack()

root.mainloop()
```