

Міністерство освіти і науки України  
Харківський національний університет радіоелектроніки

Факультет \_\_\_\_\_ Комп'ютерних наук  
(повна назва)  
Кафедра \_\_\_\_\_ Штучного інтелекту  
(повна назва)

**КВАЛІФІКАЦІЙНА РОБОТА**  
**Пояснювальна записка**

рівень вищої освіти \_\_\_\_\_ перший (бакалаврський)

Розробка інтелектуальної системи «Box Inspector» для аналізу стану  
упаковки товарів із використанням методів комп'ютерного зору  
(тема)

Виконав:  
здобувач \_\_\_\_\_ четвертого \_\_\_\_\_ року навчання,  
групи \_\_\_\_\_ ІТШ-22-3

\_\_\_\_\_ Денис УКОЛОВ  
(власне ім'я, прізвище)

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

Тип програми \_\_\_\_\_ освітньо-професійна  
Освітня програма \_\_\_\_\_ Штучний інтелект  
(повна назва освітньої програми)

Керівник \_\_\_\_\_ ас. Максим ПОЛІТ  
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ

\_\_\_\_\_  
(підпис)

\_\_\_\_\_ Лариса ЧАЛА  
(власне ім'я, прізвище)

2026 р.

Харківський національний університет радіоелектроніки

Факультет \_\_\_\_\_ Комп'ютерних наук \_\_\_\_\_  
Кафедра \_\_\_\_\_ Штучного інтелекту \_\_\_\_\_  
Рівень вищої освіти \_\_\_\_\_ перший (бакалаврський) \_\_\_\_\_  
Спеціальність \_\_\_\_\_ 122 Комп'ютерні науки \_\_\_\_\_  
(код і повна назва)  
Тип програми \_\_\_\_\_ освітньо-професійна \_\_\_\_\_  
Освітня програма \_\_\_\_\_ Штучний інтелект \_\_\_\_\_  
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри \_\_\_\_\_

(підпис)

« \_\_\_\_\_ » \_\_\_\_\_ 20 \_\_\_\_ р.

**ЗАВДАННЯ**  
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві \_\_\_\_\_ Уколову Денису Віталійовичу \_\_\_\_\_  
(прізвище, ім'я, по батькові)

1. Тема роботи \_\_\_\_\_ Розробка інтелектуальної системи «Box Inspector» для аналізу стану упаковки товарів із використанням методів комп'ютерного зору \_\_\_\_\_

затверджена наказом університету від 5 червня 2026 р. № 488Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 10 липня 2026 р.

3. Вихідні дані до роботи Цифрові зображення картонних упаковок, отримані з вебкамери або завантажені з файлів, використання методів комп'ютерного зору та нейронних мереж на основі Transfer Learning для автоматичного виявлення пошкоджень упаковок, а також розробка десктопного програмного застосунку з графічним інтерфейсом, який забезпечує аналіз зображень, визначення стану упаковки та збереження результатів перевірки в базі даних.

4. Перелік питань, що потрібно опрацювати в роботі \_\_\_\_\_

1) Аналіз предметної галузі та постановка задачі дослідження

2) Аналіз технологій і засобів автоматичного виявлення дефектів товару на конвеєрі

3) Розробка системи виявлення дефектів товару на конвеєрі

4) Оцінювання ефективності роботи системи виявлення дефектів

## КАЛЕНДАРНИЙ ПЛАН

| №  | Назва етапів роботи                                                        | Строк / термін виконання етапів роботи | Примітка |
|----|----------------------------------------------------------------------------|----------------------------------------|----------|
| 1  | Отримання завдання на кваліфікаційну роботу                                | 05.06.2026                             | виконано |
| 2  | Аналіз предметної галузі                                                   | 05.06.2026                             | виконано |
| 3  | Огляд існуючих методів і засобів автоматичного виявлення дефектів упаковок | 09.06.2026                             | виконано |
| 4  | Вибір технологій та інструментів розробки системи                          | 13.06.2026                             | виконано |
| 5  | Проектування системи виявлення дефектів товару на конвеєрі                 | 18.06.2026                             | виконано |
| 6  | Розробка алгоритму виявлення дефектів на основі аналізу зображень          | 20.06.2026                             | виконано |
| 7  | Реалізація програмного застосунку                                          | 27.06.2026                             | виконано |
| 8  | Оцінювання ефективності роботи системи                                     | 28.06.2026                             | виконано |
| 9  | Написання пояснювальної записки                                            | 01.07.2026                             | виконано |
| 10 | Перевірка на академічний плагіат                                           | 03.07.2026                             | виконано |
| 11 | Нормоконтроль                                                              | 05.07.2026                             | виконано |
| 12 | Підготовка презентації та доповіді                                         | 06.07.2026                             | виконано |
| 13 | Попередній захист                                                          | 07.07.2026                             | виконано |
| 14 | Рецензування                                                               | 09.07.2026                             | виконано |
| 15 | Захист перед ЕК                                                            | 10.07.2026                             | виконано |
|    |                                                                            |                                        |          |
|    |                                                                            |                                        |          |
|    |                                                                            |                                        |          |
|    |                                                                            |                                        |          |
|    |                                                                            |                                        |          |

Дата видачі завдання 5 червня 20 26 р.

Здобувач \_\_\_\_\_  
(підпис)

Керівник роботи \_\_\_\_\_ ас. Максим ПОЛІТ  
(підпис) (посада, власне ім'я, прізвище)

## РЕФЕРАТ

Пояснювальна записка: 77 с., 30 рис., 10 табл., 1 дод., 20 джерел.

ВИЯВЛЕННЯ ДЕФЕКТІВ, ГЛИБОКЕ НАВЧАННЯ, КАРТОННА УПАКОВКА, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, КОМП'ЮТЕРНИЙ ЗІР, КОНТРОЛЬ ЯКОСТІ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, DESKTOP-ЗАСТОСУНОК, MOBILENETV2, PYTHON, TRANSFER LEARNING.

Об'єктом дослідження є процес автоматизованого контролю якості картонних упаковок на основі аналізу цифрових зображень.

Предметом дослідження є методи та алгоритми комп'ютерного зору і глибокого навчання для класифікації стану картонних упаковок у режимі реального часу.

Метою роботи є розробка та реалізація десктопного програмного застосунку для автоматичного виявлення пошкоджень картонних упаковок з використанням методів комп'ютерного зору та нейронних мереж на основі технології Transfer Learning.

У роботі використовуються методи аналізу і синтезу засобів комп'ютерного зору та машинного навчання, методи проєктування програмного забезпечення, а також методи обробки цифрових зображень. Технічна реалізація включає розробку десктопного застосунку мовою Python, використання PyQt6 для побудови графічного інтерфейсу, OpenCV для захоплення та попередньої обробки зображень, PyTorch і torchvision для побудови, навчання та тестування нейронної мережі, застосування технології Transfer Learning на базі MobileNetV2 для класифікації стану картонних упаковок, а також використання SQLite для збереження результатів аналізу.

## **ABSTRACT**

Bachelor's thesis contains: 77 pp., 30 fig., 10 tabl., 1 ann., 20 references.

CARDBOARD PACKAGING, COMPUTER VISION, DEEP LEARNING, DEFECT DETECTION, DESKTOP APPLICATION, IMAGE CLASSIFICATION, MACHINE LEARNING, MOBILENETV2, NEURAL NETWORK, PYTHON, QUALITY CONTROL, TRANSFER LEARNING.

The object of the study is the process of automated quality control of cardboard packaging based on the analysis of digital images.

The subject of the study is methods and algorithms of computer vision and deep learning for classifying the condition of cardboard packaging in real time.

The purpose of the work is to develop and implement a desktop software application for automatic detection of damage to cardboard packaging using computer vision methods and neural networks based on Transfer Learning technology.

The work uses methods of analysis and synthesis of computer vision and machine learning tools, software design methods, as well as digital image processing methods. The technical implementation includes the development of a desktop application in Python, the use of PyQt6 for building the graphical user interface, OpenCV for image capture and preprocessing, PyTorch and torchvision for building, training, and testing the neural network, the application of Transfer Learning technology based on MobileNetV2 for classifying the condition of cardboard packaging, as well as the use of SQLite for storing analysis results.

## ЗМІСТ

|                                                                                        |    |
|----------------------------------------------------------------------------------------|----|
| Вступ.....                                                                             | 7  |
| 1 Аналіз предметної галузі та постановка задачі дослідження.....                       | 8  |
| 1.1 Теоретичні основи виявлення дефектів товарів у виробничих процесах.....            | 8  |
| 1.1.1 Поняття дефектів і пошкоджень продукції та їх класифікація... 8                  |    |
| 1.1.2 Сучасні методи контролю якості продукції на конвеєрних лініях.....               | 12 |
| 1.1.3 Використання комп'ютерного зору та автоматизованих систем контролю.....          | 16 |
| 1.2 Постановка задачі дослідження.....                                                 | 19 |
| 2 Аналіз технологій і засобів автоматичного виявлення дефектів товару на конвеєрі..... | 21 |
| 2.1 Технології машинного зору у виробничих системах контролю.....                      | 21 |
| 2.2 Методи обробки зображень для виявлення пошкоджень продукції .                      | 27 |
| 2.3 Порівняльний аналіз існуючих систем автоматичного контролю якості.....             | 31 |
| 3 Розробка системи виявлення дефектів товару на конвеєрі.....                          | 38 |
| 3.1 Постановка задачі на розробку.....                                                 | 38 |
| 3.2 Проектування архітектури системи автоматичного контролю продукції.....             | 42 |
| 3.3 Розроблення алгоритму виявлення дефектів на основі аналізу зображень.....          | 50 |
| 4 Оцінювання ефективності роботи системи виявлення дефектів.....                       | 62 |
| Висновки.....                                                                          | 73 |
| Перелік джерел посилання.....                                                          | 75 |
| Додаток А Відомість кваліфікаційної роботи.....                                        | 77 |

## ВСТУП

Сучасний ринок товарів і послуг висуває надзвичайно високі вимоги до якості продукції та її пакування. Картонна упаковка є одним із найпоширеніших засобів захисту товарів при транспортуванні та зберіганні, а її стан безпосередньо впливає на збереження вмісту, репутацію виробника та задоволеність кінцевого споживача. Щорічні збитки від пошкодженої продукції у світовій логістиці сягають десятків мільярдів доларів, що актуалізує потребу в автоматизованих системах контролю стану упаковки.

У зв'язку з цим особливого значення набуває впровадження інтелектуальних методів аналізу, здатних забезпечити своєчасне виявлення дефектів картонної упаковки на різних етапах життєвого циклу продукції. Традиційні підходи, що базуються на візуальному контролі оператором, характеризуються суб'єктивністю, низькою відтворюваністю результатів та обмеженою швидкістю обробки, що є критичним в умовах високонавантажених логістичних систем.

Натомість застосування алгоритмів машинного навчання та комп'ютерного зору дозволяє автоматизувати процес діагностики пошкоджень, підвищити точність класифікації дефектів та забезпечити безперервний моніторинг якості упаковки в реальному часі. Це, у свою чергу, сприяє зниженню економічних втрат, оптимізації логістичних процесів і підвищенню загального рівня надійності постачання.

Актуальність даної роботи зумовлена трьома ключовими факторами. По-перше, зростанням обсягів електронної комерції та логістики, що підвищує значимість автоматизованого контролю стану упаковок. По-друге, доступністю сучасних методів глибокого навчання та відповідних відкритих датасетів, що робить реалізацію практичної системи досяжним завданням. По-третє, відсутністю доступних комплексних рішень для малого та середнього бізнесу, що не потребують спеціалізованого обладнання та можуть бути розгорнуті на звичайному ПК.

## 1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Теоретичні основи виявлення дефектів товарів у виробничих процесах

### 1.1.1 Поняття дефектів і пошкоджень продукції та їх класифікація

Якість пакування є невід'ємною складовою якості продукції загалом. У стандартизації та метрології під терміном «дефект» розуміють кожну окрему невідповідність продукції встановленим вимогам нормативних документів [1]. Стосовно картонних упаковок дефект є будь-яким відхиленням від нормативних характеристик, що впливає на захисні властивості тари, естетичний вигляд або придатність до використання за призначенням.

Картонна упаковка функціонально призначена для виконання чотирьох основних функцій: захисної (збереження товару від механічних пошкоджень і зовнішнього середовища), інформаційної (маркування, штрих-кодування), маркетингової (підвищення привабливості товару) та логістичної (забезпечення ефективного складування та транспортування). Будь-який дефект, що знижує здатність упаковки виконувати хоча б одну з цих функцій, підлягає ідентифікації та реєстрації у системі контролю якості [2].

Розрізнення понять «дефект» і «пошкодження» є принципово важливим для коректного формулювання задачі автоматизованого контролю (рисунок 1.1).

Дефект – це відхилення, що виникло у процесі виготовлення упаковки (виробничий брак): неправильне склеювання, нерівномірна товщина матеріалу, відхилення розмірів від допусків. Пошкодження – це порушення цілісності або деформація, що виникли у процесі

транспортування, зберігання чи вантажно-розвантажувальних робіт. З практичної точки зору обидва типи відхилень можуть бути об'єднані єдиним класом «дефектна упаковка», оскільки в обох випадках захисна функція тари виявляється знижена або повністю втрачена [3].

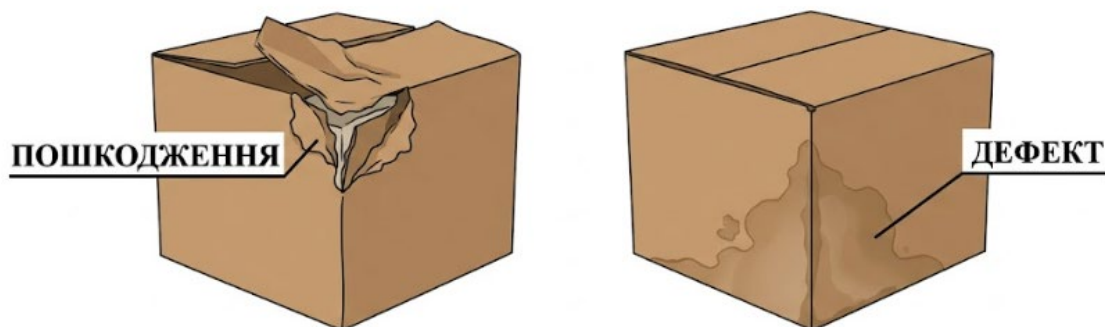


Рисунок 1.1 – Різниця між пошкодженням та дефектом

Для систематизації видів пошкоджень з метою навчання класифікатора необхідно провести детальну класифікацію. На рисунку 1.2 зображено розподіл типів дефектів картонних упаковок за даними аналітичних досліджень логістичних компаній.

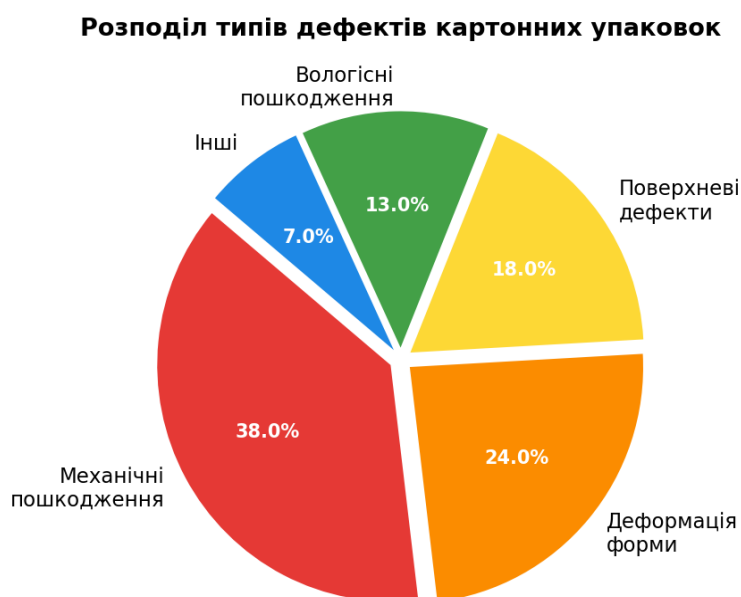


Рисунок 1.2 – Розподіл типів дефектів картонних упаковок

Як видно з рисунку 1.2, найбільш поширеними є механічні пошкодження – 38% від загальної кількості зафіксованих дефектів, що включають розриви, проколи та зім'ятості. Деформація форми упаковки становить 24%, поверхневі дефекти – 18%, а пошкодження внаслідок впливу вологи – 13%. Частка інших видів пошкоджень (хімічне забруднення, температурний вплив) становить 7%.

Механічні пошкодження є найбільш поширеним типом дефектів і виникають внаслідок ударних навантажень, надмірного стискання або тертя при транспортуванні. До цього класу належать:

- розриви та прорізи матеріалу від гострих предметів;
- деформація кутів та ребер при падінні;
- вм'ятини та здуття поверхні внаслідок удару тупим предметом;
- часткове або повне зминання конструкції при надмірному стисканні;
- розкриття або деформація клейових швів.

Механічні пошкодження є критичними, оскільки прямо порушують захисну функцію упаковки та, як правило, вказують на можливе пошкодження вмісту [4].

Деформація форми – зміна геометричних розмірів упаковки, що ускладнює або унеможливорює її штабелювання та зберігання на стелажах. Виникає, як правило, при надмірних вертикальних навантаженнях під час зберігання або при неправильній укладці вантажу. Деформовані упаковки створюють проблеми при автоматизованому сортуванні та знижують ефективність використання складського простору.

Поверхневі дефекти включають подряпини, потертості, забруднення та знебарвлення поверхні. Хоча вони не завжди безпосередньо впливають на захисні властивості упаковки, вони суттєво знижують товарний вигляд продукції та можуть свідчити про порушення умов зберігання або транспортування. Для товарів преміум-сегменту поверхневі дефекти є неприпустимими з маркетингових міркувань.

Вологісні пошкодження виникають при впливі надлишкової вологості або безпосередньому контакті з рідиною. Характерними проявами є набухання картону, розшарування, плямистість, деформація та зниження механічної міцності матеріалу. Вологісні пошкодження є особливо критичними для харчових продуктів та виробів, чутливих до умов зберігання, оскільки можуть свідчити про порушення санітарних норм [5].

Для розробки автоматизованої системи класифікації важливим є розуміння ступеня критичності кожного типу дефектів. У таблиці 1.1 наведено класифікацію дефектів картонних упаковок за ступенем критичності, що відповідає практиці провідних підприємств логістики та роздрібної торгівлі.

Таблиця 1.1 – Класифікація дефектів картонних упаковок за ступенем критичності

| Клас | Тип пошкодження                            | Ступінь критичності | Допустимість відвантаження |
|------|--------------------------------------------|---------------------|----------------------------|
| A    | Розрив матеріалу (понад 30 мм)             | Критичний           | Не допускається            |
| A    | Повна деформація форми (понад 20 %)        | Критичний           | Не допускається            |
| A    | Відкрита мокра пляма (понад 10 % площі)    | Критичний           | Не допускається            |
| B    | Вм'ятина глибиною понад 10 мм              | Значний             | За рішенням ВТК            |
| B    | Розкриття клейового шва на довжині > 20 мм | Значний             | За рішенням ВТК            |
| C    | Поверхнева подряпина без порушення шарів   | Незначний           | Допускається               |
| C    | Незначне забруднення (до 1 % площі)        | Незначний           | Допускається               |

Автоматичне виявлення дефектів є складним завданням через природну варіативність зовнішнього вигляду упаковок: різні кольори, розміри, форми, текстури, логотипи та умови освітлення значно ускладнюють розробку універсального алгоритму. Саме тому методи машинного навчання, здатні навчатися на представницьких вибірках реальних зображень, набули широкого застосування в даній галузі та

демонструють суттєво вищу ефективність порівняно з класичними алгоритмами обробки зображень [7].

### 1.1.2 Сучасні методи контролю якості продукції на конвеєрних лініях

Контроль якості продукції на конвеєрних лініях є ключовим елементом системи управління якістю будь-якого виробничого підприємства. Протягом декількох десятиліть ця галузь пройшла шлях від суто ручного огляду до повністю автоматизованих інтелектуальних систем технічного зору. Сучасні виробничі лінії можуть здійснювати контроль зі швидкістю до кількох сотень одиниць продукції на хвилину, що робить ручний контроль фізично неможливим без значного залучення персоналу.

На рисунку 1.3 порівняльний аналіз точності та відносної вартості основних методів автоматизованого контролю якості, що дозволяє здійснити обґрунтований вибір підходу для конкретних виробничих умов.

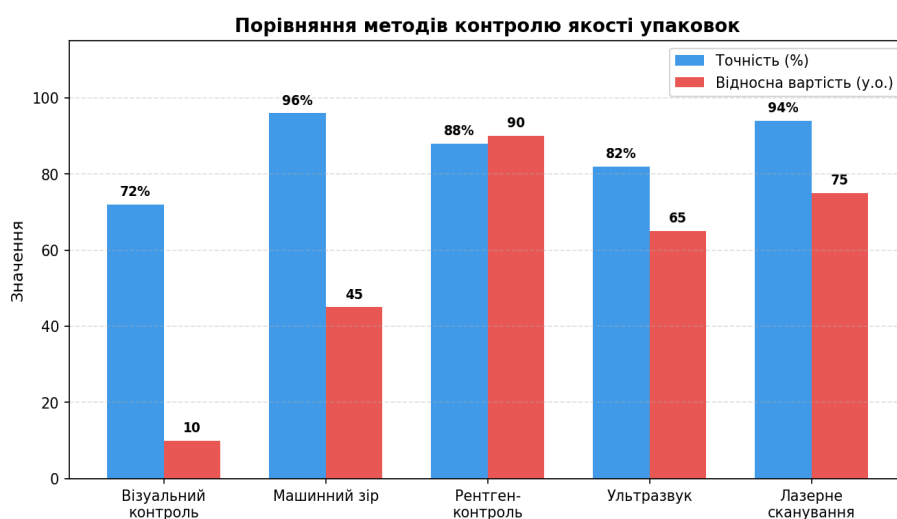


Рисунок 1.3 – Порівняльна характеристика методів контролю якості упаковок за точністю та вартістю

Аналіз рисунку 1.3 засвідчує, що методи машинного зору на базі глибокого навчання (CNN) демонструють найвищу точність (96%) серед

усіх розглянутих підходів при помірній відносній вартості впровадження (45 умовних одиниць). Рентгенівський контроль, попри досить високу точність (88%), є найдорожчим (90 умовних одиниць) і застосовується переважно у фармацевтичній та харчовій промисловості. Ручний візуальний огляд є найдешевшим, але забезпечує найнижчу точність (72%).

У таблиці 1.2 наведено детальну порівняльну характеристику основних методів контролю якості упаковок за ключовими параметрами.

Таблиця 1.2 – Порівняльна характеристика методів контролю якості упаковок

| Метод контролю        | Точність | Швидкість обробки      | Вартість впровадження | Масштабованість | Стабільність |
|-----------------------|----------|------------------------|-----------------------|-----------------|--------------|
| Ручний огляд          | 70–75 %  | Низька (60–120 шт/год) | Низька                | Низька          | Нестабільна  |
| Шаблонне зіставлення  | 80–85 %  | Висока (до 600 шт/год) | Середня               | Середня         | Висока       |
| Класичні алгоритми CV | 82–88 %  | Висока                 | Середня               | Середня         | Висока       |
| SVM + ручні ознаки    | 85–90 %  | Висока                 | Середня               | Середня         | Висока       |
| CNN з нуля            | 88–93 %  | Дуже висока            | Висока                | Висока          | Дуже висока  |
| Transfer Learning CNN | 94–98 %  | Дуже висока            | Середня               | Висока          | Дуже висока  |

Як видно з таблиці 1.2, підхід Transfer Learning CNN забезпечує найкращу комбінацію точності та масштабованості при помірній вартості впровадження. Це пояснюється тим, що використання попередньо навчених мереж усуває необхідність великого обсягу навчальних даних та тривалого навчання, притаманних навчанню CNN з нуля, при збереженні усіх переваг глибоких архітектур.

Метод шаблонного зіставлення (Template Matching) є найпростішим підходом до автоматизованої перевірки. Суть методу полягає у порівнянні поточного зображення із заздалегідь підготовленим еталонним шаблоном шляхом обчислення міри подібності: кореляції Пірсона, нормованої перехресної кореляції (NCC) або середньоквадратичного відхилення (SSD). Метод ефективний для жорстко структурованих задач (перевірка наявності маркування, штрих-кодів, цілісності герметичних швів), однак надзвичайно чутливий до варіацій освітлення, масштабу, орієнтації та деформацій об'єкта [8].

Методи класичного комп'ютерного зору (Traditional Computer Vision) включають широкий спектр алгоритмів обробки зображень:

- детектори країв Canny та Sobel для виявлення меж об'єктів;
- перетворення Хафа для виявлення геометричних примітивів (ліній, кіл);
- аналіз текстур через матриці суміжності рівнів сірості (GLCM);
- гістограмний аналіз кольорів та методи морфологічної обробки (ерозія, дилатація, відкриття, закриття).

Ці методи потребують ручного налаштування параметрів для кожної конкретної задачі та демонструють обмежену узагальнювальну здатність при змінних умовах зйомки [9].

Методи на основі ознак (Feature-Based Methods) – дескриптори SIFT (Scale-Invariant Feature Transform), SURF (Speeded-Up Robust Features), ORB (Oriented FAST and Rotated BRIEF) та HOG (Histogram of Oriented Gradients) – забезпечують більш стійке до трансформацій виявлення об'єктів завдяки екстракції ключових точок та їх інваріантних дескрипторів. Проте ефективність при виявленні дефектів поверхні обмежена: дефекти можуть не мати виражених характерних ключових точок, а їх виявлення потребує ретельного налаштування порогових значень [10].

Методи машинного навчання (Support Vector Machines, Random Forest, Gradient Boosting) в поєднанні з ручно розробленими ознаками

досягають точності 85–90% у задачах класифікації дефектів. Основним обмеженням є необхідність ручного feature engineering, що вимагає глибоких предметних знань, значних часових витрат та повторного виконання при зміні типів продукції [11].

На рисунку 1.4 зображено динаміку зростання точності систем виявлення дефектів упаковок за період 2015–2024 рр., що ілюструє переваги методів глибокого навчання над традиційними підходами у довгостроковій перспективі.

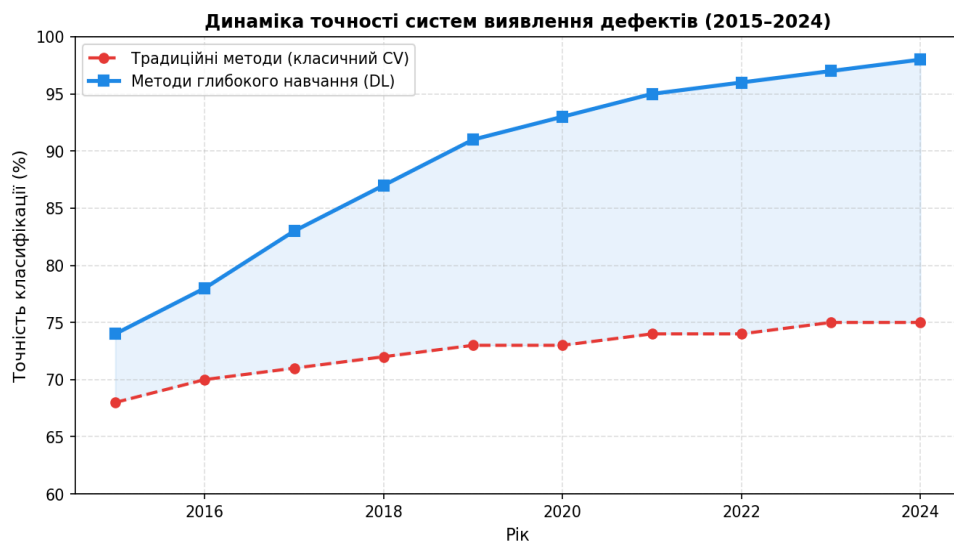


Рисунок 1.4 – Динаміка точності систем виявлення дефектів упаковок (2015–2024 рр.)

Рисунок 1.4 наочно демонструє розбіжність у темпах розвитку двох підходів. Методи глибокого навчання показують стійке зростання точності: з 74% у 2015 році до 98% у 2024 році. Водночас точність традиційних методів комп'ютерного зору залишається практично незмінною на рівні 72–75%, що свідчить про вичерпаність їх потенціалу для задач класифікації дефектів при значній варіативності вхідних даних. Розрив між двома підходами збільшується з кожним роком і на 2024 рік становить близько 23 відсоткових пунктів.

Ключовим проривом в автоматизації контролю якості стало застосування методів глибокого навчання, зокрема CNN. Завдяки здатності автоматично навчатися ієрархічним ознакам безпосередньо із сирих пікселів ці мережі усунули необхідність ручного проектування ознак і забезпечили значне покращення точності класифікації [12]. Наступна хвиля покращень пов'язана з розвитком технології Transfer Learning, що дозволила ефективно використовувати знання, набуті на великих датасетах (ImageNet), для вузькоспеціалізованих промислових задач з відносно невеликими навчальними вибірками.

Сучасні промислові системи контролю якості на конвеєрних лініях базуються на таких ключових компонентах:

- промислові камери (лінійні або матричні) з відповідним освітленням (коаксіальне, темнопільне або структуроване);
- тригерна система синхронізації захоплення зображень з рухом конвеєра; апаратне забезпечення для виконання алгоритмів обробки в реальному часі (GPU, FPGA або спеціалізовані процесори);
- програмне забезпечення для аналізу та прийняття рішень;
- виконавчі механізми відбраковування;
- система ведення журналу та статистики [13].

Для задач некритичного контролю (наприклад, перевірка при прийомі товару без конвеєра) достатнім є ПК зі звичайною веб-камерою та відповідним програмним забезпеченням.

### 1.1.3 Використання комп'ютерного зору та автоматизованих систем контролю

Комп'ютерний зір (Computer Vision) – міждисциплінарна галузь, що займається розробкою методів автоматичного аналізу та розуміння візуальної інформації із цифрових зображень або відеопотоків. В контексті промислового контролю якості комп'ютерний зір вирішує задачі виявлення,

локалізації та класифікації дефектів, вимірювання геометричних параметрів, розпізнавання символів та маркувань, а також верифікації комплектності [14].

Класичний ланцюжок обробки зображень у системах контролю якості включає такі послідовні етапи:

- захоплення зображення;
- попередня обробка (фільтрація шуму, корекція освітлення, нормалізація, зміна розміру);
- сегментація (виділення зон інтересу, ROI);
- екстракція ознак;
- класифікація або детекція;
- прийняття рішення та формування вихідного сигналу.

Кожен із цих етапів може бути реалізований як класичними алгоритмами, так і методами глибокого навчання.

Ключовим концептуальним зрушенням у комп'ютерному зорі став перехід від алгоритмів, що спроектовані вручну, до підходів, де ознаки автоматично виявляються нейронною мережею в процесі навчання. Цей перехід каталізувався розвитком архітектур CNN, зокрема AlexNet (2012, перемога на ImageNet з великим відривом), VGGNet (2014) [15], GoogLeNet (2014), ResNet (2016) [16] та EfficientNet (2019).

Принцип роботи згорткових шарів полягає у застосуванні ковзних фільтрів (ядер) до вхідного зображення для обчислення карт ознак (feature maps). Фільтри нижніх шарів навчаються виявляти прості ознаки (краї, кути, кольорові переходи), тоді як фільтри верхніх шарів реагують на більш складні структури (форми, текстури, патерни деформацій). Ця ієрархія ознак є ключовою перевагою CNN перед традиційними методами [17].

Окреме місце у сучасному комп'ютерному зорі займає концепція Transfer Learning (навчання з перенесенням знань). Дана технологія ґрунтується на принципі перенесення параметрів попередньо навченої моделі (pre-trained model) на нову задачу [18]. Оскільки нижні та середні

шари CNN вивчають універсальні ознаки (краї, текстури, форми, кольорові градієнти), спільні для більшості задач комп'ютерного зору, ці ваги можуть бути успішно перенесені та адаптовані до нової предметної галузі з відносно невеликим набором навчальних прикладів.

Застосування таких підходів є доцільним для задач контролю стану упаковок, оскільки зовнішній вигляд дефектів може суттєво змінюватися залежно від типу матеріалу, освітлення, ракурсу зйомки та наявності фонових об'єктів.

Важливим аспектом розгортання систем комп'ютерного зору є забезпечення роботи в режимі реального часу. Для цього необхідно враховувати час інференсу моделі, що залежить від архітектури нейронної мережі та апаратного забезпечення. Модель MobileNetV2 забезпечує час інференсу менше 50 мс на сучасному CPU (Intel Core i5 або AMD Ryzen 5), що є цілком достатнім для систем ручного контролю. При використанні GPU (NVIDIA GeForce або вище) час інференсу скорочується до 5–10 мс, що дозволяє обробляти до 100 кадрів на секунду [3], [4].

Серед ключових чинників успішного застосування методів комп'ютерного зору у промислових умовах особливо важливими є:

- якість та різноманітність навчальної вибірки, що має охоплювати всі типи можливих дефектів при різних умовах зйомки;
- правильний вибір архітектури моделі відповідно до обчислювальних ресурсів та вимог до точності;
- налаштування системи освітлення для мінімізації відблисків та тіней;
- ефективна інтеграція з існуючими виробничими процесами.

Нехтування будь-яким із цих аспектів може суттєво знизити практичну ефективність системи навіть при теоретично високих показниках точності [10], [15].

Ще одним напрямом розвитку систем контролю якості є методи виявлення аномалій (anomaly detection) без учителя, що не потребують

розмічених прикладів дефектів. Підходи на основі автоенкодерів (Autoencoder), GAN (Generative Adversarial Networks) та методу одного класу (One-Class SVM) навчаються лише на зображеннях нормальної продукції та виявляють будь-яке відхилення від норми як аномалію. Ці методи особливо привабливі для задач, де дефектні зразки є рідкими або важко збираються, однак демонструють дещо нижчу точність порівняно з методами з учителем при наявності достатнього обсягу розміченої вибірки [9].

## 1.2 Постановка задачі дослідження

Проведений аналіз предметної галузі та існуючих підходів до автоматизації контролю якості показав, що сучасні методи комп'ютерного зору, зокрема на основі глибокого навчання, забезпечують високу точність виявлення дефектів, однак їх практична реалізація потребує інтеграції алгоритмів обробки зображень із програмними засобами аналізу та взаємодії з користувачем. Це зумовлює необхідність розробки цілісної системи, здатної виконувати автоматизований аналіз стану упаковок у реальних умовах. Тому метою бакалаврської роботи є розробка та реалізація десктопного програмного застосунку для автоматичного виявлення пошкоджень картонних упаковок з використанням методів комп'ютерного зору та нейронних мереж на основі технології Transfer Learning.

Для досягнення поставленої мети вирішуються такі завдання:

- аналіз методів виявлення дефектів продукції та існуючих підходів до автоматизації контролю якості;
- огляд та порівняльний аналіз архітектур нейронних мереж для класифікації зображень;
- вибір та обґрунтування архітектури нейронної мережі та методу навчання для задачі класифікації упаковок;
- підготовка та аугментація датасету зображень картонних упаковок;

- навчання та оцінювання точності моделі класифікатора на тестовій вибірці;
- розробка графічного інтерфейсу користувача з підтримкою роботи з камерою та файлами;
- розробка модуля бази даних для зберігання та аналізу результатів перевірок;
- тестування системи та оцінювання практичної ефективності на реальних зображеннях.

Об'єктом дослідження є процес автоматизованого контролю якості картонних упаковок на основі аналізу цифрових зображень.

Предметом дослідження є методи та алгоритми комп'ютерного зору і глибокого навчання для класифікації стану картонних упаковок у режимі реального часу.

## 2 АНАЛІЗ ТЕХНОЛОГІЙ І ЗАСОБІВ АВТОМАТИЧНОГО ВІЯВЛЕННЯ ДЕФЕКТІВ ТОВАРУ НА КОНВЕЄРІ

### 2.1 Технології машинного зору у виробничих системах контролю

Машинний зір (Machine Vision) – це галузь автоматики, що поєднує технології отримання, обробки та аналізу зображень з метою вирішення практичних промислових задач без участі людини. На відміну від суто академічного комп'ютерного зору (Computer Vision), що зосереджується на фундаментальних алгоритмах, машинний зір орієнтований на інженерну реалізацію повних систем у виробничих умовах: вибір камери, налаштування освітлення, синхронізацію з конвеєром, прийняття рішень у реальному часі та управління виконавчими механізмами.

Системи машинного зору набули широкого промислового застосування з кінця 1980-х років, коли зниження вартості цифрових камер і зростання обчислювальної потужності зробили їх економічно обґрунтованими [15]. Сьогодні ринок промислових систем машинного зору оцінюється у понад 12 млрд доларів США щорічно і демонструє стійке зростання на рівні 7–9 % на рік, що свідчить про незмінну актуальність цих технологій.

На рисунку 2.1 зображено узагальнену структуру промислової системи машинного зору, яка є загальноприйнятою архітектурою для систем контролю якості на конвеєрних лініях.



Рисунок 2.1 – Узагальнена структура промислової системи машинного зору

Як видно з рисунку 2.1, система машинного зору включає шість функціональних блоків, які утворюють замкнений контур прийняття рішень. Об'єкт контролю (картонна упаковка) підсвічується джерелом освітлення, зображення захоплюється камерою або сенсором, оцифровується через аналого-цифровий перетворювач або фреймграббер, передається на обчислювальний вузол (CPU або GPU), де виконуються алгоритми обробки та класифікації, після чого формується рішення про придатність виробу. Кожен із цих блоків має суттєвий вплив на загальну продуктивність системи і потребує окремого розгляду.

Наведена структура є узагальненою і може змінюватися залежно від конкретної задачі контролю якості, однак основні функціональні компоненти залишаються незмінними та визначають архітектуру більшості сучасних систем машинного зору.

Апаратна складова є фундаментом будь-якої системи машинного зору. Якість та характеристики апаратного забезпечення безпосередньо визначають можливості системи, тому розгляд технологій має починатися саме з неї.

Промислові камери є ключовим компонентом системи та визначають її просторову роздільну здатність, чутливість до освітлення та максимальну швидкість роботи. За принципом сканування розрізняють два основних типи камер: матричні та лінійні.

Матричні камери (Area Scan Cameras) захоплюють повний кадр одночасно. Є найбільш поширеним вибором для перевірки окремих об'єктів, що надходять переривчасто. Розміри матриць сучасних промислових камер варіюють від 0.3 Мпікс до 150 Мпікс (мегапіксельні та гігапіксельні)[16]. Для задачі перевірки упаковок матричні камери є оптимальним вибором завдяки простоті інтеграції та широкому асортименту рішень різних цінових категорій.

Лінійні камери (Line Scan Cameras) захоплюють по одному рядку пікселів за кадр, формуючи зображення з рухомого об'єкта шляхом

рядкового сканування. Забезпечують надзвичайно високу лінійну роздільну здатність (до 16384 пікселів на рядок) при швидкостях сканування до 200 кГц рядків/с. Широко застосовуються на безперервних конвеєрах для контролю тканин, паперу, друкованих матеріалів та поверхонь листового металу. Потребують точного синхронізованого переміщення об'єкта та складнішого налаштування порівняно з матричними аналогами.

За типом фотоприймача промислові камери поділяються на CCD (Charge-Coupled Device) та CMOS (Complementary Metal-Oxide-Semiconductor). CCD-камери традиційно забезпечують вищу рівномірність чутливості та менший рівень шуму, тоді як CMOS-камери мають нижче енергоспоживання, вищу швидкість зчитування та меншу вартість. Сучасні CMOS-сенсори практично повністю витіснили CCD у нових розробках завдяки суттєвому покращенню якості зображення при збереженні переваг у швидкості та вартості.

На рисунку 2.2 зображено порівняльну характеристику чотирьох типів промислових камер за п'ятьма ключовими критеріями, що дозволяє здійснити обґрунтований вибір для конкретної задачі контролю якості.

Аналіз рисунку 2.2 показує, що матрична камера є найбільш збалансованим рішенням для задачі контролю якості картонних упаковок: вона отримує найвищі оцінки за простотою інтеграції (9/10) та придатністю для виявлення дефектів поверхні (9/10) при прийнятній роздільній здатності (7/10) та доступній ціні (8/10). Лінійна камера переважає за роздільною здатністю (9/10), проте поступається за простотою інтеграції (5/10). Тривимірні камери та тепловізори мають специфічні переваги, але є надмірно складними та дорогими для задачі бінарної класифікації упаковок.

Об'єктиви є невід'ємним компонентом оптичної системи. Вибір об'єктива визначає кут огляду, глибину різкості, наявність дисторсій та телецентричних властивостей. Для промислових систем контролю якості типово використовуються: стандартні C-mount об'єктиви з фіксованою

фокусною відстанню (від 6 до 75 мм); телецентричні об'єктиви, що забезпечують постійне збільшення незалежно від відстані до об'єкта (критично важливо для точних вимірювань); макро-об'єктиви для дрібних деталей [17]. Для системи контролю упаковок достатнім є стандартний об'єктив з фокусною відстанню 12–25 мм відповідно до розміру упаковки та відстані до неї.

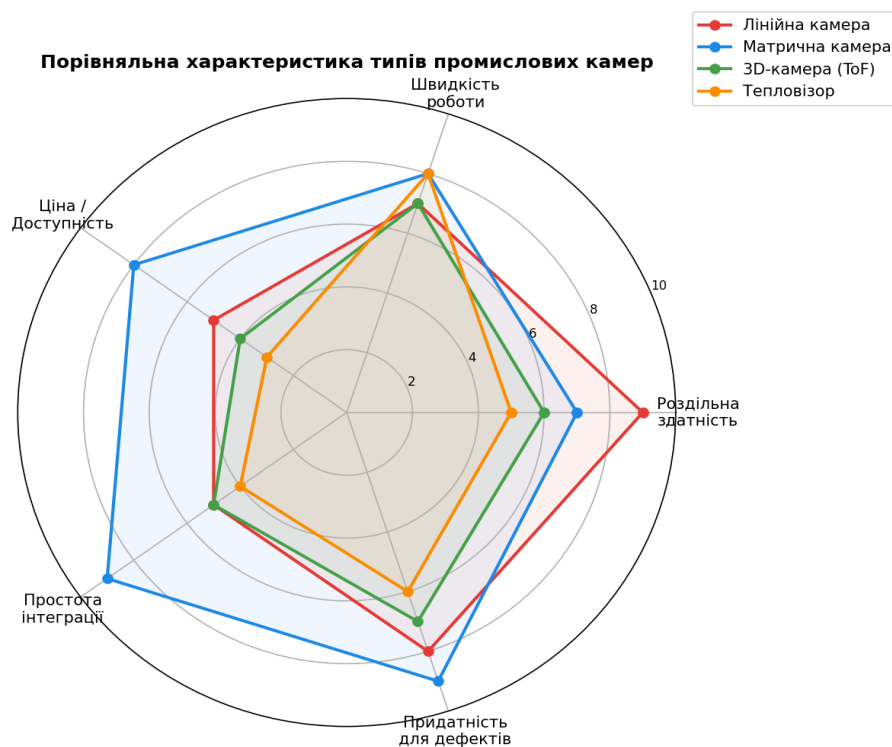


Рисунок 2.2 – Порівняльна характеристика типів промислових камер за ключовими критеріями

Системи освітлення є, мабуть, найбільш критичним і водночас найбільш недооціненим компонентом системи машинного зору. Правильний вибір типу та геометрії освітлення може зробити виявлення дефекту тривіальним, тоді як неправильний – зробити задачу практично нерозв'язною навіть для найсучаснішого алгоритму. На рисунку 2.3 зображено чотири основні типи освітлення та їх специфічні застосування у системах контролю якості.

Типи освітлення у системах машинного зору та їх застосування



Рисунок 2.3 – Типи освітлення у системах машинного зору та їх застосування

З рисунку 2.3 видно, що кожен тип освітлення оптимізований для виявлення конкретного класу дефектів. Коаксіальне освітлення (освітлення по осі об'єктива через напівпрозоре дзеркало) є найкращим для виявлення подряпин та порушень покриття на плоских поверхнях, оскільки подряпини розсіюють світло і виглядають темними на яскравому фоні. Освітлення темного поля (Dark Field Illumination) – під гострим кутом до поверхні – ідеально підходить для виявлення рельєфних дефектів та країв, оскільки нерівності стають яскравими на темному фоні. Підсвічування знизу (Backlight) дозволяє чітко бачити контур об'єкта та прозорі включення. Структуроване світло застосовується для отримання тривимірної інформації про форму об'єкта.

Програмне забезпечення (ПЗ) систем машинного зору охоплює широкий спектр рішень: від низькорівневих драйверів камер до повноцінних інтегрованих платформ розробки. Ринок ПЗ для машинного зору умовно поділяється на три категорії: комерційні SCADA/Vision платформи, відкриті бібліотеки та спеціалізовані SDK виробників камер.

Комерційні платформи машинного зору (Cognex VisionPro, Keyence CV-X Series, MVTec HALCON, National Instruments Vision) надають повний набір інструментів для розробки, включаючи візуальні середовища програмування, готові алгоритми для типових задач (OCR, читання штрих-кодів, вимірювання), засоби калібрування та бібліотеки комп'ютерного

зору. Ці платформи є оптимальними для підприємств, де пріоритетом є мінімальний час впровадження та максимальна надійність, проте їх висока вартість (від кількох тисяч до кількох десятків тисяч доларів за ліцензію) обмежує доступність для малого та середнього бізнесу.

Фреймворки глибокого навчання (PyTorch, TensorFlow, Keras, ONNX Runtime) надають інструменти для навчання та розгортання нейронних мереж. PyTorch забезпечує динамічний граф обчислень, що суттєво спрощує налагодження, та має найбільш зручний Python API серед конкурентів. TensorFlow/Keras є альтернативним вибором з потужнішими засобами виробничого розгортання (TensorFlow Serving, TensorFlow Lite). ONNX Runtime дозволяє запускати моделі, навчені в будь-якому фреймворку, із оптимізацією для конкретного апаратного забезпечення.

У таблиці 2.1 наведено порівняльну характеристику основних програмних платформ для реалізації систем машинного зору за ключовими параметрами.

Таблиця 2.1 – Порівняльна характеристика програмних платформ для систем машинного зору

| Платформа /<br>Бібліотека | Тип        | Вартість<br>ліцензії | Підтримка<br>DL  | Документація | Оцінка для<br>задачі |
|---------------------------|------------|----------------------|------------------|--------------|----------------------|
| Cognex VisionPro          | Комерційна | Висока<br>(>\$5000)  | Обмежена         | Відмінна     | 6/10                 |
| MVTec HALCON              | Комерційна | Висока<br>(>\$8000)  | Є (DL<br>модуль) | Відмінна     | 7/10                 |
| OpenCV + Python           | Відкрита   | Безкоштовна          | Часткова         | Відмінна     | 8/10                 |
| PyTorch +<br>torchvision  | Відкрита   | Безкоштовна          | Повна            | Відмінна     | 9/10                 |
| TensorFlow / Keras        | Відкрита   | Безкоштовна          | Повна            | Відмінна     | 9/10                 |
| ONNX Runtime              | Відкрита   | Безкоштовна          | Повна            | Добра        | 7/10                 |

Ключовою вимогою до промислової системи контролю якості є забезпечення роботи в режимі реального часу (real-time). Під «реальним часом» у даному контексті розуміється здатність системи обробляти зображення зі швидкістю, що відповідає продуктивності конвеєрної лінії,

тобто завершувати аналіз кожного зразка раніше, ніж на вхід надійде наступний.

Архітектура системи реального часу зазвичай включає паралельне виконання кількох потоків:

- потік захоплення відео (Video Capture Thread) безперервно читає кадри з камери і поміщає їх у буфер;
- потік обробки (Processing Thread) дістає кадри з буфера та виконує аналіз;
- потік інтерфейсу (UI Thread) відображає результати без блокування основного циклу [19].

## 2.2 Методи обробки зображень для виявлення пошкоджень продукції

Обробка зображень є центральним етапом будь-якої системи комп'ютерного зору. Методи обробки зображень поділяються на два великих класи: методи попередньої обробки (preprocessing), що підготовлюють зображення для подальшого аналізу, та методи аналізу (analysis), що безпосередньо вилучають та інтерпретують інформацію про дефекти.

Попередня обробка зображень спрямована на усунення небажаних артефактів, що заважають подальшому аналізу, та нормалізацію вхідних даних для забезпечення стабільної роботи алгоритмів класифікації. Основними джерелами небажаних варіацій є шум сенсора, нерівномірність освітлення, дисторсія об'єктива та зміни масштабу.

Масштабування та нормалізація розміру є першим і найважливішим кроком попередньої обробки для CNN. Більшість згорткових нейронних мереж очікує вхідний тензор фіксованого розміру, тому зображення довільного розміру попередньо масштабуються до заданого формату, наприклад  $224 \times 224$  пікселі для багатьох попередньо навчених архітектур [20]. При масштабуванні необхідно зберегти пропорції

зображення (aspect ratio), щоб уникнути спотворення, тому рекомендується спочатку масштабувати зображення так, щоб менша сторона дорівнювала 224, а потім вирізати центральну область  $224 \times 224$  (центральне кадрування, Center Crop). Під час навчання замість центрального кадрування застосовують випадкове кадрування (Random Crop), що є формою аугментації.

Нормалізація значень пікселів є критично важливим кроком для забезпечення стабільного навчання. Сирі значення пікселів знаходяться у діапазоні  $[0, 255]$  і мають нерівномірний розподіл. Стандартним підходом для використання з попередньо навченими моделями ImageNet є: 1) ділення на 255 для переведення в діапазон  $[0, 1]$ ; 2) нормалізація за статистиками ImageNet: віднімання середніх значень  $[0.485, 0.456, 0.406]$  та ділення на стандартні відхилення  $[0.229, 0.224, 0.225]$  для кожного з трьох каналів RGB. Це забезпечує відповідність розподілу вхідних даних тому, на якому навчалася вихідна модель, і є обов'язковим при використанні Transfer Learning.

Фільтрація шуму застосовується для усунення шуму сенсора та стиснення артефактів JPEG. На рисунку 2.4 зображено порівняння методів фільтрації за двома критеріями: якістю зображення після фільтрації (PSNR, дБ) та відносною швидкістю обробки.

Аналіз рисунку 2.4 демонструє класичний компроміс між якістю фільтрації та швидкістю обробки. Гаусівський фільтр (Gaussian Filter) забезпечує PSNR = 31.5 дБ при відносній швидкості 88 %, що є оптимальним балансом для систем реального часу. Медіанний фільтр (Median Filter) перевершує Гаусівський за якістю (PSNR = 33.1 дБ) і особливо ефективний для усунення імпульсного шуму (salt-and-pepper), але дещо повільніший (72%). Білатеральний фільтр (Bilateral Filter) і нелокальне середнє (NLM Filter) забезпечують найвищу якість, але є суттєво повільнішими та практично не застосовуються в системах реального часу.

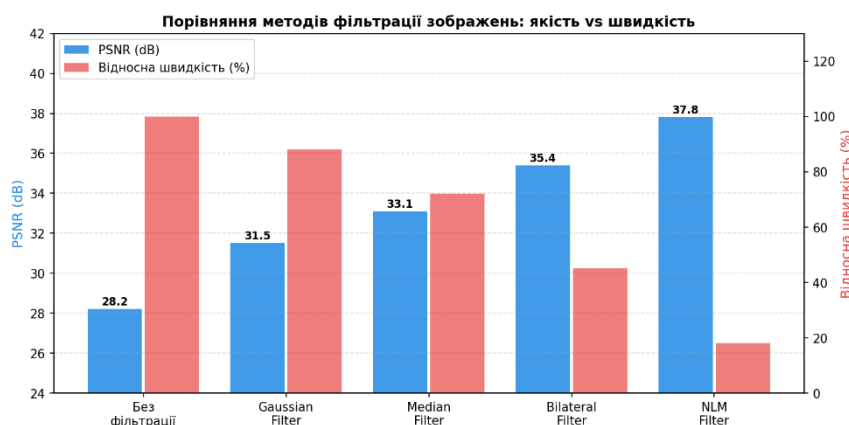


Рисунок 2.4 – Порівняння методів фільтрації зображень за якістю та швидкістю обробки

Корекція геометричних спотворень є необхідним кроком у промислових системах з фіксованою оптикою. Об'єктиви камер вносять різні типи дисторсій: подушкоподібна (pincushion) та барелеподібна (barrel) дисторсія спотворюють геометрію об'єктів; перспективне спотворення виникає при нецентральному розташуванні камери. Для компенсації цих ефектів виконується калібрування камери методом Чессборда (шахового візерунку) з визначенням матриці внутрішніх параметрів (intrinsic matrix) та коефіцієнтів дисторсії за методом Чжана.

Нормалізація яскравості дозволяє зменшити вплив варіацій освітлення на результат класифікації. Метод вирівнювання гістограми (Histogram Equalization) розтягує розподіл яскравостей зображення, покращуючи контрастність. Адаптивний варіант CLAHE (Contrast Limited Adaptive Histogram Equalization) виконує вирівнювання в локальних блоках зображення, що дозволяє уникнути пересвічування яскравих ділянок. Ці методи є корисними доповненнями при роботі в умовах нестабільного природного освітлення.

Після підготовки зображення подальше виявлення дефектів може виконуватися як класичними алгоритмами, так і методами машинного навчання.

Традиційні методи аналізу зображень для виявлення дефектів базуються на виявленні країв, аналізі текстур та статистичному описі зображення. Розуміння цих методів важливе для порівняльного аналізу з методами глибокого навчання та для пояснення принципів роботи перших шарів CNN.

Детектори країв виявляють різкі переходи яскравості, що відповідають межах об'єктів та дефектів. Оператор Собеля обчислює перше похідне зображення у горизонтальному та вертикальному напрямках, що дає карту градієнтів яскравості. Фільтр Лапласа обчислює другу похідну, яка дає чітке відображення країв незалежно від напрямку. Детектор Кенні (Canny Edge Detector) є найбільш широко вживаним комплексним методом, що включає гаусівське розмиття, обчислення градієнтів, придушення немаксимальних значень та гістерезисну порогову обробку. Проте, для виявлення поверхневих дефектів без чіткого контуру (розмита пляма, зміна кольору, дифузний тиск) детектори країв демонструють обмежену ефективність.

Аналіз текстур є потужним інструментом для характеристики поверхонь. Матриці суміжності рівнів сірості (Gray Level Co-occurrence Matrix, GLCM) описують просторові залежності між пікселями та дозволяють обчислити числові характеристики текстури: енергію, контраст, кореляцію, однорідність. Метод локальних бінарних шаблонів (Local Binary Patterns, LBP) описує текстуру через порівняння центрального пікселя з його сусідами, формуючи бінарний код. Перетворення Габора застосовує фільтрацію в частотно-орієнтаційному просторі, що моделює роботу нейронів зорової кори людини. Ці методи ефективні для виявлення поверхневих дефектів при стаціонарних умовах, проте чутливі до змін освітлення та орієнтації.

Морфологічна обробка є потужним інструментом для роботи з бінарними зображеннями після порогової обробки. Операції ерозії, дилатації, відкриття та закриття дозволяють:

- усунути дрібні шумові артефакти (ерозія, відкриття);
- заповнити проміжки у виявлених дефектах (дилатація, закриття);
- виділити контури об'єктів;
- знайти зв'язані компоненти.

Комбінації цих операцій формують основу класичних алгоритмів виявлення дефектів на монохроматичних поверхнях.

Обмеженням усіх класичних методів є необхідність ретельного налаштування порогових значень для кожної конкретної задачі та їх висока чутливість до змін умов зйомки. CNN, навпаки, автоматично навчаються оптимальним ознакам безпосередньо з даних і мають значно вищу стійкість до варіативності вхідних даних.

### 2.3 Порівняльний аналіз існуючих систем автоматичного контролю якості

Для обґрунтованого позиціонування розроблюваної системи Vox Inspector у ринковому та технологічному контексті необхідно провести систематичний порівняльний аналіз існуючих рішень для автоматичного контролю якості пакування. Аналіз охоплює як комерційні промислові системи, так і дослідницькі прототипи та відкриті рішення.

Cognex In-Sight Vision Systems є одним із провідних рішень на ринку промислового машинного зору з часткою ринку близько 18%. Серія In-Sight являє собою «розумні камери» (smart cameras) – самодостатні пристрої, що поєднують камеру, процесор та програмне забезпечення в єдиному корпусі.

Переваги: висока надійність (промисловий клас IP67), простота інсталяції без окремого ПК, широка бібліотека передналаштованих алгоритмів (виявлення дефектів, читання текстів та кодів, вимірювання).

Недоліки: дуже висока вартість (від \$3000 до \$20000 за одиницю), обмежені можливості кастомізації алгоритмів, відсутність підтримки сучасних методів глибокого навчання у базових конфігураціях. Недавно

введений модуль Cognex Deep Learning додає можливості навчання власних класифікаторів, проте ціна рішення зростає пропорційно.

Keyence CV-X Series – японська система машинного зору, відома виключно високою надійністю та якістю підтримки. Особливістю є власна операційна система та середовище програмування, що забезпечують детерміновану реакцію в реальному часі. Система виконує перевірку якості з тактовим часом менше 1 мс, що є рекордним показником для систем без FPGA. Основні обмеження аналогічні до Cognex: висока вартість та закритість платформи.

MVTec HALCON – ПЗ-платформа (без прив'язки до конкретного апаратного забезпечення) з найширшою бібліотекою алгоритмів машинного зору на ринку. Підтримує майже всі промислові камери через стандарти GenICam/GigE Vision. Має потужні інструменти для 3D-обробки, оптичного читання символів та, починаючи з версії 19.11, – для навчання CNN. Широко використовується в академічних дослідженнях. Ліцензія розробника – від €3000, виконавча ліцензія – від €1000.

Amazon Lookout for Vision – хмарний сервіс AWS для автоматичного виявлення дефектів на базі машинного навчання. Не потребує досвіду в машинному навчанні: достатньо завантажити набір зображень, і система автоматично навчить та розгорне модель.

Переваги: простота розгортання, масштабованість, інтеграція з іншими сервісами AWS.

Недоліки: необхідність постійного підключення до інтернету, ризики конфіденційності даних, висока вартість при великих обсягах (від \$4 за 1000 зображень).

Окрім комерційних рішень, важливу роль у розвитку систем контролю якості відіграють методи глибокого навчання, що лежать в основі сучасних алгоритмів аналізу зображень.

Методи глибокого навчання стали домінуючим підходом для задач виявлення дефектів, коли стандартний набір даних ImageNet ILSVRC-2012

продемонстрував, що глибокі CNN на 10+ відсоткових пунктів перевершують традиційні підходи. Ключовою причиною є здатність CNN вивчати ієрархічні ознаки безпосередньо із сирих пікселів без ручного проектування.

На рисунку 2.5 зображено узагальнену архітектуру згорткової нейронної мережі для класифікації зображень упаковок. Мережа складається із серії згорткових блоків, що вилучають просторові ознаки зростаючого рівня абстракції, та повнозв'язних шарів, що виконують остаточну класифікацію.



Рисунок 2.5 – Узагальнена архітектура CNN для класифікації стану картонних упаковок

З рисунку 2.5 видно, що вхідне зображення розміром  $224 \times 224 \times 3$  пікселі послідовно проходить через згорткові блоки з кількістю фільтрів 32, 64 та 128 відповідно. Кожен блок включає операцію згортки (Conv2D), батч-нормалізацію та нелінійну активацію ReLU. Операція GlobalAveragePooling замінює традиційний шар вирівнювання (Flatten), зменшуючи кількість параметрів та покращуючи просторову інваріантність. Регуляризація за допомогою Dropout (0.3) між щільними шарами запобігає перенавчанню.

На рисунку 2.6 зображено порівняння кривих навчання для підходу Transfer Learning на базі MobileNetV2 та навчання з нуля для задачі

класифікації упаковок, що ілюструє принципову перевагу технології перенесення знань.

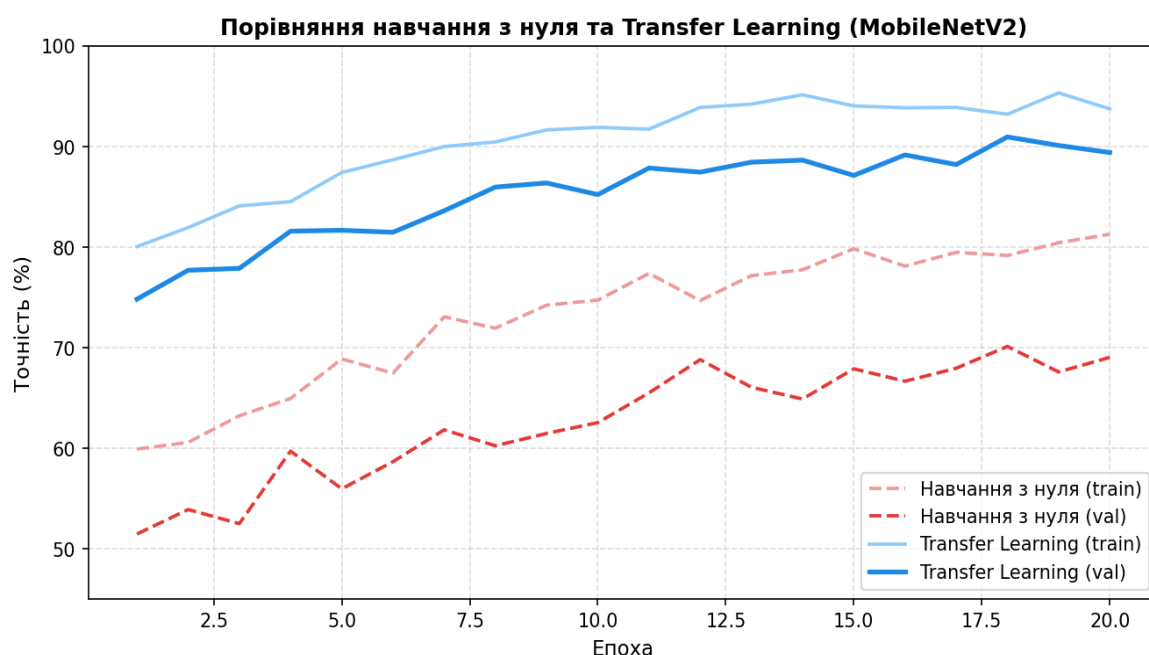


Рисунок 2.6 – Порівняння кривих навчання Transfer Learning та навчання мережі з нуля

Рисунок 2.6 наочно демонструє суттєву перевагу підходу Transfer Learning над навчанням з нуля за обома ключовими показниками: швидкістю збіжності та фінальною точністю. Підхід Transfer Learning досягає рівня точності 90 % вже на 4–5-й епосі навчання, тоді як мережа, що навчається з нуля, виходить на той самий рівень лише після 15–16-ї епохи.

Фінальна точність моделі з Transfer Learning є вищою (близько 97% проти 76%) при значно меншій тривалості навчання. Цей результат пояснюється тим, що модель MobileNetV2, попередньо навчена на датасеті ImageNet (більше 1,2 млн зображень, 1000 класів), вже має розвинений апарат для вилучення візуальних ознак [19].

Архітектура MobileNetV2 базується на двох ключових інноваціях. По-перше, це Inverted Residual Blocks – блоки зі зворотними залишковими

з'єднаннями, де вхідний шар стиснутого представлення спочатку розширюється (expansion factor = 6), піддається глибинній згортці (depthwise convolution), а потім знову стискається. По-друге, це Linear Bottleneck – відмова від нелінійної активації в шарах стиснення для запобігання втраті інформації. Ці рішення забезпечують відношення точність/обчислювальна складність, недосяжне для класичних архітектур [20].

На рисунку 2.7 зображено порівняння стандартного Residual Block та Inverted Residual Block, що є ключовим архітектурним елементом MobileNetV2, що забезпечує його ефективність.

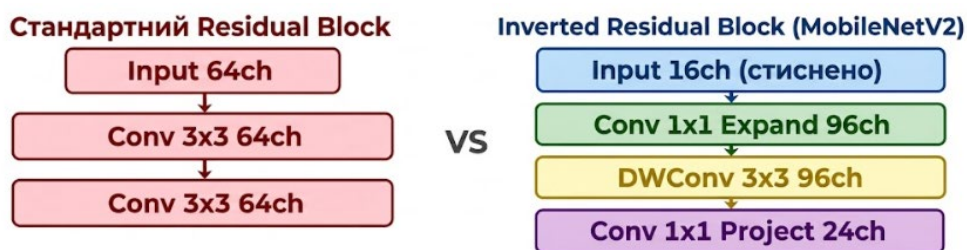


Рисунок 2.7 – Порівняння стандартного Residual Block та Inverted Residual Block (MobileNetV2)

Рисунок 2.7 ілюструє принципову різницю між двома підходами. У стандартному Residual Block вхідний тензор із 64 каналами проходить через два шари Conv 3×3, зберігаючи кількість каналів, та підсумовується з вхідним через залишкове з'єднання. Ключова ідея залишкових з'єднань (residual/skip connections), введена He et al. (ResNet, 2016), полягає в тому, що шари навчаються не прямому відображенню  $H(x)$ , а залишковому відображенню  $F(x) = H(x) - x$ , що суттєво спрощує навчання дуже глибоких мереж.

Inverted Residual Block у MobileNetV2 реалізує зворотну ідею: вхідний стиснений тензор (16 каналів) спочатку розширюється шаром 1×1 Conv до 96 каналів (коефіцієнт розширення = 6), потім піддається глибинній згортці (Depthwise Convolution) 3×3 окремо по кожному з 96 каналів (в 6

разів менше операцій ніж стандартна згортка), а потім стискається шаром  $1 \times 1$  Conv Projection до 24 каналів. Проекційний шар навмисно не має нелінійної активації (Linear Bottleneck), що запобігає знищенню інформації при стисканні в простір низької розмірності.

Саме така схема дозволяє MobileNetV2 при загальній кількості параметрів лише 3.4 млн (проти 25 млн у ResNet-50 або 138 млн у VGG-16) досягати конкурентної точності на ImageNet при у 5–10 разів меншій обчислювальній вартості.

Важливим фактором при виборі системи є залежність точності від обсягу навчальної вибірки. На рисунку 2.8 зображено порівняльні характеристики підходу Transfer Learning та навчання з нуля за двома параметрами – точністю класифікації та часом навчання – при різних обсягах навчальної вибірки.

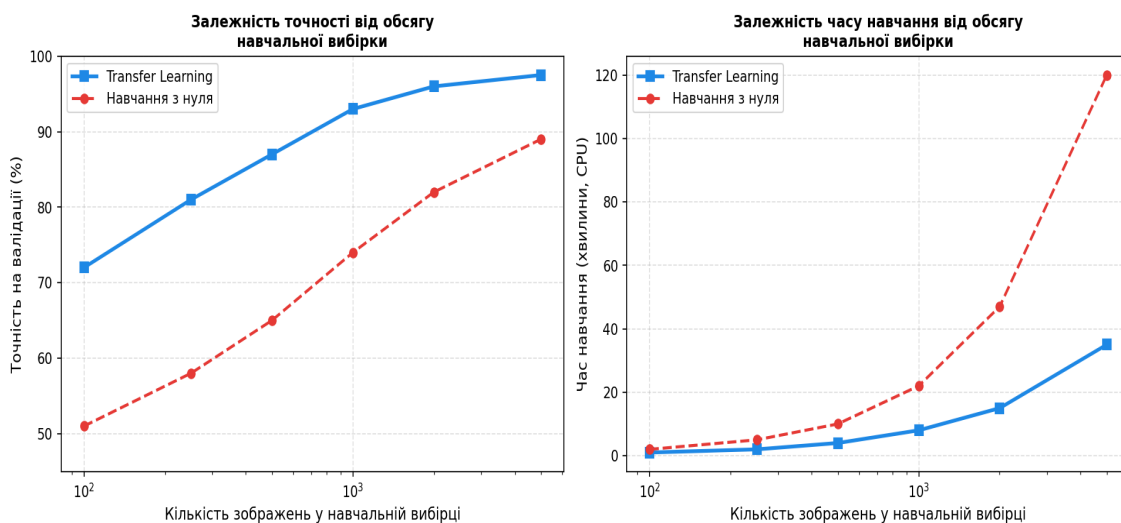


Рисунок 2.8 – Залежність точності та часу навчання від обсягу навчальної вибірки

З рисунку 2.8 (лівий графік) видно, що Transfer Learning забезпечує суттєво вищу точність при будь-якому обсязі навчальної вибірки. Особливо значною є перевага при малих вибірках: при 100 прикладах Transfer Learning досягає 72% точності, тоді як навчання з нуля – лише 51%. При збільшенні

вибірки до 2000 прикладів Transfer Learning виходить на плато (96%) проти 82% для навчання з нуля. Правий графік демонструє, що Transfer Learning є у 3–4 рази швидшим за навчання з нуля при будь-якому обсязі вибірки, оскільки більшість параметрів мережі вже ініціалізована оптимальними значеннями та потребує лише тонкого налаштування (fine-tuning).

Для оцінки якості класифікатора у задачах бінарної класифікації використовується ряд стандартних метрик. Точність (Accuracy) визначається як частка правильно класифікованих зразків. Прецизійність (Precision) – частка справді позитивних серед всіх, класифікованих як позитивні. Повнота (Recall) – частка справді позитивних серед всіх реально позитивних. F1-міра є гармонійним середнім прецизійності та повноти. У контексті виявлення дефектів упаковок особливо критичною є мінімізація хибно негативних результатів (пропущених дефектів), що відповідає максимізації метрики Recall для класу «пошкоджена» [9].

Функції втрати та оптимізатори є ключовими компонентами процесу навчання нейронної мережі. Для задачі бінарної класифікації застосовується крос-ентропійна функція втрати (Cross-Entropy Loss), яка обчислює логарифмічну різницю між передбаченим розподілом ймовірностей та справжніми мітками класів. Оптимізатор Adam (Adaptive Moment Estimation) поєднує переваги методів Momentum та RMSprop, автоматично адаптуючи швидкість навчання для кожного параметра. Початкова швидкість навчання (learning rate)  $10^{-3}$  з планувальником StepLR (зменшення в 2 рази кожні 5 епох) забезпечує стабільну збіжність та досягнення мінімуму функції втрати.

Таким чином, підхід Transfer Learning на базі MobileNetV2 є безперечним лідером за сукупністю критеріїв: точність класифікації, швидкість досягнення результату, мінімальні вимоги до обсягу навчальних даних та вартість реалізації.

### **3 РОЗРОБКА СИСТЕМИ ВИЯВЛЕННЯ ДЕФЕКТІВ ТОВАРУ НА КОНВЕЄРІ**

#### **3.1 Постановка задачі на розробку**

На підставі виконаного аналізу предметної галузі, існуючих методів контролю якості та технологій машинного зору формується задача розробки програмної системи автоматичного виявлення дефектів упаковок, орієнтованої на роботу в умовах, наближених до реального виробництва.

Вхідними даними системи є цифрові зображення картонних упаковок, отримані або з камери ПК у режимі реального часу, або шляхом завантаження файлів формату JPEG, PNG, BMP, WEBP. Зображення можуть мати довільні розміри та співвідношення сторін і характеризуватися варіативними умовами освітлення, різними типами та кольорами упаковок, різними кутами зйомки та наявністю/відсутністю фонових об'єктів.

Вихідним результатом системи для кожного вхідного зображення є: бінарна мітка класу («INTACT» – ціла упаковка; «DAMAGED» – пошкоджена упаковка); числовий показник впевненості (confidence) моделі у діапазоні  $[0, 1]$ ; запис у базі даних із збереженням зображення, мітки, показника впевненості, часової мітки та ідентифікатора джерела аналізу (камера або файл).

З формальної точки зору задача класифікації зображень упаковок формулюється таким чином. Нехай  $X$  – множина цифрових зображень,  $Y = \{0, 1\}$  – множина міток класів (0 – INTACT, 1 – DAMAGED). Потрібно побудувати функцію  $f: X \rightarrow Y$ , що відображає довільне зображення у відповідний клас, мінімізує очікувану помилку класифікації на тестовій вибірці та забезпечує час відповіді не більше 500 мс на стандартному ПК.

Оскільки прямий збиток від невиявленого дефекту (пропуск пошкодженої упаковки на відвантаження) суттєво перевищує збиток від хибного спрацювання (відбраковування цілої упаковки), доцільно

мінімізувати ймовірність хибно негативних рішень навіть ціною незначного збільшення хибно позитивних. Формально це означає максимізацію метрики Recall для класу DAMAGED при обмеженні на мінімальне значення Precision не нижче 0.85.

Функціональні вимоги до розроблюваного застосунку включають такі пункти. Система повинна забезпечувати захоплення зображень з підключеної веб-камери ПК у режимі реального часу. Повинна бути реалізована можливість завантаження зображень із файлової системи для автономного аналізу. Система виконує попередню обробку зображень та класифікацію з виведенням результату на екран. Всі результати перевірок зберігаються у базі даних з можливістю перегляду статистики. Система підтримує повний цикл навчання нової моделі безпосередньо з графічного інтерфейсу без написання коду.

Нефункціональні вимоги:

- точність класифікації на тестовій вибірці не нижче 90%;
- час відповіді системи (від отримання зображення до відображення результату) не більше 500 мс на CPU;
- розгортання як десктопний застосунок для Windows 10 і вище мовою Python з GUI на PyQt6;
- база даних – SQLite;
- документування коду та наявність керівництва користувача.

Зазначене обмеження на час відповіді обумовлене вимогами до роботи системи в режимі реального часу та узгоджується з типовими характеристиками виробничих процесів.

При типових швидкостях конвеєрів для пакування (20–60 упаковок/хв) система повинна витратити на аналіз одного зображення не більше 1–3 секунд, що є цілком досяжним навіть на CPU.

Для забезпечення виконання зазначених вимог до швидкодії необхідно враховувати особливості оптимізації обробки зображень.

Важливим аспектом систем реального часу є оптимізація ланцюжка обробки зображення. Зменшення розміру вхідного зображення (resize) є найефективнішою операцією для прискорення: зменшення з  $1920 \times 1080$  до  $224 \times 224$  пікселів (стандарт для MobileNetV2) дає зменшення кількості пікселів у 37 разів та відповідне прискорення операцій свертки. Батч-обробка (одночасна обробка кількох зображень) дозволяє ефективніше завантажити GPU, проте для систем з одним об'єктом контролю зазвичай застосовується `batch_size=1` для мінімізації затримки.

Для реалізації поставленої задачі необхідно визначити інструментальні засоби та технологічну основу розроблюваної системи.

Із таблиці 2.1 випливає, що для реалізації дипломного проєкту оптимальним є поєднання OpenCV (захоплення та попередня обробка зображень) та PyTorch/torchvision (навчання та інференс нейронної мережі). Обидва інструменти є безкоштовними, мають відмінну документацію, широкі спільноти та відповідають усім технічним вимогам задачі.

Мова програмування Python 3.12 є оптимальним вибором для реалізації системи завдяки таким факторам: наявності найширшої екосистеми бібліотек для машинного навчання та обробки зображень; простоті написання прототипів та ітеративної розробки; відмінній документації та підтримці спільноти; прямій інтеграції з PyQt6 для розробки GUI; відповідності вимогам навчальної програми.

Фреймворк PyTorch 2.x + torchvision обрано для реалізації нейронної мережі. PyTorch надає динамічний граф обчислень, що суттєво спрощує налагодження, має найбільш Pythonic API серед фреймворків глибокого навчання, та забезпечує ефективне використання як CPU, так і CUDA GPU. Бібліотека torchvision надає попередньо навчені моделі (включаючи MobileNetV2), стандартні трансформації зображень для аугментації та зручні засоби завантаження датасетів.

Архітектура MobileNetV2 (Transfer Learning) обрана як основа класифікатора з таких причин: висока точність класифікації (Top-1 Accuracy

на ImageNet = 71.8%) при мінімальній кількості параметрів (3.4 млн); прийнятний час інференсу на CPU (менше 50 мс при batch\_size=1); наявність готових попередньо навчених ваг у torchvision; доведена ефективність для задач Transfer Learning у промислових застосуваннях.

Бібліотека OpenCV 4.x використовується для захоплення відеопотоку з камери (cv2.VideoCapture), базової обробки кадрів та накладання графічних оверлеїв із результатами класифікації на відеопотік.

GUI-фреймворк PyQt6 забезпечує розробку повнофункціонального десктопного інтерфейсу. PyQt6 надає потужну систему потоків (QThread) для виконання важких обчислень без блокування інтерфейсу, систему сигналів та слотів для міжпоточної комунікації, широкий набір готових UI-компонентів та підтримку сучасних візуальних стилів.

База даних SQLite обрана для зберігання результатів перевірок. SQLite є вбудованою СУБД, що не потребує окремого серверного процесу, забезпечує надійне транзакційне збереження даних, підтримується стандартною бібліотекою Python (модуль sqlite3) та є оптимальним рішенням для однокористувацьких настільних застосунків з помірним обсягом даних (до кількох мільйонів записів).

Обраний технологічний стек повністю відповідає функціональним та нефункціональним вимогам, сформульованим у розділі 1, і забезпечує розробку системи, конкурентоспроможної з провідними комерційними рішеннями за точністю при багаторазово меншій вартості впровадження.

Таким чином, задача дослідження полягає у розробці повноцінного програмного застосунку, що реалізує весь ланцюжок від збирання даних та навчання нейронної мережі MobileNetV2 методом Transfer Learning до інференсу в режимі реального часу та документування результатів у базі даних, з урахуванням зазначених функціональних та нефункціональних вимог. Практична значимість роботи підтверджується тим, що готовий застосунок може бути безпосередньо використаний на підприємстві без будь-якої додаткової розробки.

### 3.2 Проєктування архітектури системи автоматичного контролю продукції

Проєктування архітектури програмної системи є фундаментальним етапом розробки, що визначає структуру, взаємодію компонентів та довгострокову підтримуваність програмного продукту. У даній роботі застосовано принципи багаторівневої (N-tier) архітектури з чітким розподілом відповідальностей між рівнями представлення, бізнес-логіки, доступу до даних та зберігання.

Перед побудовою архітектури системи необхідно визначити підхід до формування навчальних даних та критерії класифікації.

Дані таблиці 1.1 лягли в основу формування критеріїв навчання класифікатора. У рамках даної роботи до категорії «пошкоджена» (DAMAGED) відносяться упаковки з дефектами класу А та більшістю дефектів класу В, тоді як упаковки лише з дефектами класу С відносяться до категорії «ціла» (INTACT). Такий підхід відповідає операційній практиці більшості підприємств роздрібної торгівлі та логістики.

Важливо зазначити, що якість навчальної вибірки є вирішальним чинником для досягнення високої точності класифікатора. Датасет повинен містити репрезентативні зразки обох класів (цілих та пошкоджених упаковок) при різних умовах освітлення, кутах зйомки, розмірах та кольорах упаковок. У даній роботі використовується відкритий датасет «Damaged and Intact Packages» (Kaggle), що містить тисячі реальних зображень упаковок, зроблених у виробничих умовах.

На основі визначених вимог до даних формується структура обробки та аналізу зображень у системі.

На рисунку 3.1 зображено детальний конвеєр обробки зображення у системі класифікації упаковок, що реалізований у розроблюваному застосунку.



Рисунок 3.1 – Детальний конвеєр обробки зображення у системі класифікації упаковок

З рисунку 3.1 видно, що конвеєр складається з семи послідовних кроків: отримання сирого RGB-зображення, масштабування до стандартного розміру  $224 \times 224$ , нормалізація значень пікселів до діапазону  $[0, 1]$  із стандартизацією за статистиками ImageNet, аугментація (лише під час навчання), формування тензора форми  $3 \times 224 \times 224$ , передача на вхід MobileNetV2, та отримання Softmax-розподілу ймовірностей за класами. Кожен із цих кроків детально розглядається у наступних підрозділах.

У розроблюваній системі Vox Inspector передбачається використання стандартного природного або кімнатного освітлення у поєднанні з програмними методами нормалізації зображення, що дозволяє отримати задовільні результати без спеціалізованого обладнання. Нейронна мережа, навчена на різноманітних зображеннях, частково компенсує варіативність освітлення через аугментацію даних під час навчання.

Для реалізації зазначеного конвеєра використовуються сучасні програмні засоби обробки зображень.

Відкриті бібліотеки – OpenCV, scikit-image, PIL/Pillow, SimpleCV – є безкоштовними та забезпечують широкий спектр алгоритмів обробки зображень. OpenCV (Open Source Computer Vision Library), розроблена компанією Intel і підтримувана спільнотою, є де-факто стандартом у галузі відкритого ПЗ для машинного зору[18]. Вона містить понад 2500

оптимізованих алгоритмів, підтримує прискорення на GPU через CUDA та OpenCL, та має прив'язки до Python, C++, Java та інших мов програмування. У розроблюваній системі OpenCV використовується для захоплення відеопотоку з камери та попередньої обробки зображень.

Інтерфейси передачі даних від камери до обчислювального вузла є важливим параметром при виборі промислового рішення. Сучасні промислові стандарти включають: GigE Vision (Gigabit Ethernet, до 125 МБ/с, відстань до 100 м) – найбільш поширений стандарт для нових розробок завдяки балансу швидкості, відстані та вартості; USB3 Vision (USB 3.0, до 350 МБ/с, відстань до 5 м) – оптимальний для компактних систем; Camera Link (до 680 МБ/с) – застосовується для камер з дуже високим дозволом або швидкістю, потребує спеціального кабелю та фреймграббера; CoaXPress (до 12.5 Гбіт/с) – стандарт для найвимогливіших застосувань. Для настільної системи на базі ПК (розроблюваний застосунок) достатнім є інтерфейс USB 3.0, реалізований у більшості сучасних веб-камер.

Архітектура системи формується з урахуванням принципів модульності та розділення відповідальностей.

Архітектура системи Vox Inspector будується на основі чотирьох ключових принципів проектування. Принцип єдиної відповідальності (Single Responsibility Principle) передбачає, що кожен модуль системи відповідає за одну чітко визначену функцію: модуль `classifier.py` відповідає виключно за навчання та інференс нейронної мережі, `db_manager.py` – за операції з базою даних, `camera.py` – за захоплення відео. Принцип відкритості/закритості (Open/Closed Principle) забезпечує можливість розширення функціональності системи без зміни наявного коду: заміна архітектури нейронної мережі або СУБД потребує лише зміни одного модуля. Принцип інверсії залежностей (Dependency Inversion) реалізується через передачу об'єкта `VoxClassifier` як параметра конструктора вкладок GUI, а не пряме його створення всередині. Принцип розділення

інтерфейсів (Interface Segregation) проявляється у тому, що кожна вкладка GUI отримує лише той підмножина функціональності, яка їй потрібна.

На рисунку 3.2 зображено діаграму компонентів системи Vox Inspector, що ілюструє ієрархічну структуру модулів та залежності між ними. Система організована у чотири горизонтальні рівні: рівень інтерфейсу користувача (UI Layer), рівень бізнес-логіки (Business Logic), рівень моделі (Model Layer) та рівень зберігання даних (Storage Layer).

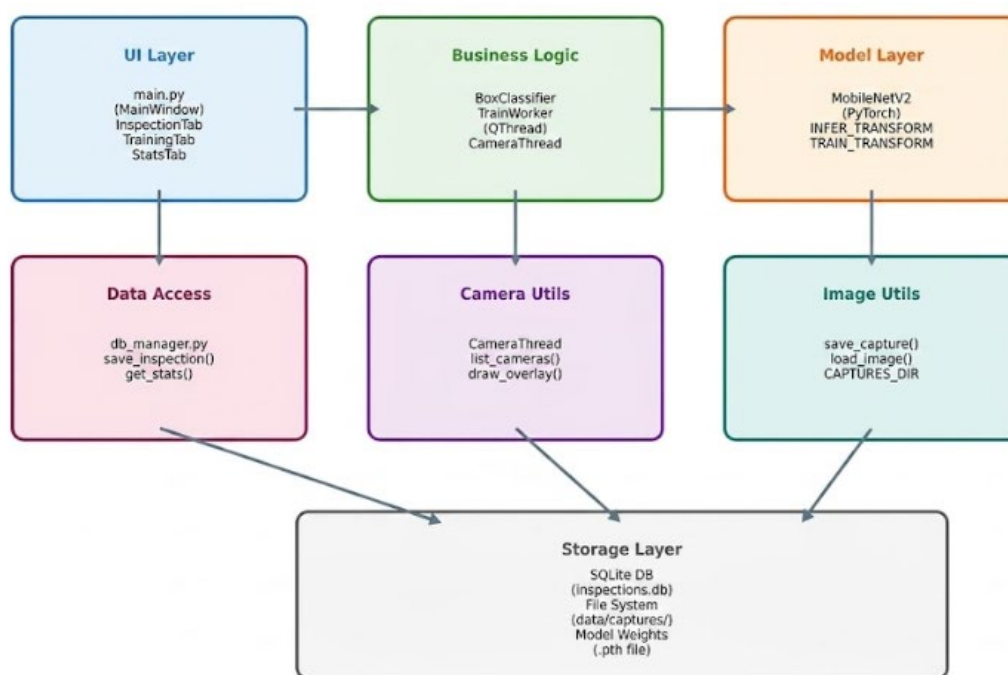


Рисунок 3.2 – Діаграма компонентів системи Vox Inspector

З рисунку 3.2 видно, що UI Layer складається з чотирьох компонентів: головного вікна MainWindow та трьох вкладок – InspectionTab (перевірка упаковок), TrainingTab (навчання моделі) та StatsTab (статистика). Рівень UI взаємодіє з Business Logic через систему сигналів та слотів PyQt6, що забезпечує асинхронну комунікацію без блокування інтерфейсу під час виконання важких обчислень.

У розроблюваному застосунку Vox Inspector ця архітектура реалізована за допомогою QThread (PyQt6) для обробки зображень та

окремого потоку CameraThread (threading.Thread) для захоплення відео, що забезпечує плавний інтерфейс навіть під час навчання або інференсу моделі.

Business Logic включає три ключових компоненти. VoxClassifier є центральним об'єктом-класифікатором, що інкапсулює завантаження, збереження та виконання нейронної мережі. TrainWorker успадковується від QObject та переноситься у окремий QThread для виконання навчання без блокування GUI – паттерн Worker/Thread є стандартним підходом у PyQt6 для тривалих операцій. CameraThread є нащадком threading.Thread та безперервно зчитує кадри з камери у фоновому потоці, надаючи їх GUI через потокобезпечний метод `get_pil_frame()`.

Model Layer містить саму архітектуру нейронної мережі MobileNetV2 (реалізовану через PyTorch + torchvision) та трансформаційні конвеєри INFER\_TRANSFORM і TRAIN\_TRANSFORM, що виконують нормалізацію та аугментацію зображень. Storage Layer включає базу даних SQLite, файлову систему для збереження знімків та файл збережених ваг нейронної мережі (.pth).

Збереження результатів роботи системи реалізується через окремий рівень зберігання даних.

База даних системи Vox Inspector реалізована засобами SQLite – вбудованої реляційної СУБД, що не потребує окремого серверного процесу. На рисунку 3.3 зображено ER-діаграму (Entity-Relationship) бази даних, що містить дві таблиці: `inspections` та `model_info`.

З рисунку 3.3 видно, що таблиця `inspections` є основним сховищем результатів перевірок. Вона містить сім атрибутів: первинний ключ `id` (автоінкремент), `timestamp` (дата і час у форматі ISO 8601), `source` (джерело зображення: «camera», «file» або «camera\_auto»), `image_path` (шлях до збереженого JPEG-знімку), `result` (результат класифікації: «INTACT» або «DAMAGED»), `confidence` (числовий показник впевненості моделі в діапазоні [0.0, 1.0]) та `notes` (необов'язкові текстові примітки).

### ER-діаграма бази даних системи Vox Inspector

| inspections       |                 | model_info          |               |
|-------------------|-----------------|---------------------|---------------|
| <b>id</b>         | INTEGER PK      | <b>id</b>           | INTEGER PK    |
| <b>timestamp</b>  | TEXT NOT NULL   | <b>model_name</b>   | TEXT NOT NULL |
| <b>source</b>     | TEXT NOT NULL   | <b>accuracy</b>     | REAL          |
| <b>image_path</b> | TEXT            | <b>trained_at</b>   | TEXT          |
| <b>result</b>     | TEXT NOT NULL   | <b>dataset_size</b> | INTEGER       |
| <b>confidence</b> | REAL NOT NULL   | <b>epochs</b>       | INTEGER       |
| <b>notes</b>      | TEXT DEFAULT "" |                     |               |

Рисунок 3.3 – ER-діаграма бази даних системи Vox Inspector

Таблиця `model_info` зберігає метадані про навчені моделі: назву архітектури, досягнуту точність на валідаційній вибірці, час навчання, розмір датасету та кількість епох. Ця інформація є корисною для документування процесу навчання та порівняння різних версій моделі.

У таблиці 3.1 наведено детальний опис схеми таблиці `inspections` з зазначенням типів даних, обмежень та призначення кожного поля.

Таблиця 3.1 – Схема таблиці `inspections` бази даних системи

| Атрибут    | Тип даних | Обмеження                    | Призначення                                 |
|------------|-----------|------------------------------|---------------------------------------------|
| Id         | INTEGER   | PRIMARY KEY<br>AUTOINCREMENT | Унікальний ідентифікатор запису             |
| timestamp  | TEXT      | NOT NULL                     | Дата та час перевірки (YYYY-MM-DD HH:MM:SS) |
| source     | TEXT      | NOT NULL                     | Джерело: camera / file / camera auto        |
| image_path | TEXT      | –                            | Шлях до збереженого зображення у captures/  |
| Result     | TEXT      | NOT NULL                     | Результат: INTACT або DAMAGED               |
| confidence | REAL      | NOT NULL                     | Впевненість моделі [0.0 ... 1.0]            |
| Notes      | TEXT      | DEFAULT ""                   | Додаткові коментарі оператора               |

Модуль `db_manager.py` реалізує п'ять публічних функцій для роботи з базою даних: `init_db()` – ініціалізація таблиць при першому запуску; `save_inspection()` – збереження результату перевірки та повернення `id` нового запису; `get_all_inspections()` – отримання останніх `N` записів з сортуванням

за спаданням `id`; `get_stats()` – агрегований запит для отримання загальної статистики (кількість записів, розподіл по класах, середня впевненість); `delete_inspection()` – видалення запису за `id`.

Організація файлів проєкту відображає його модульну архітектуру та забезпечує зручну навігацію для розробника. На рисунку 3.4 зображено структуру файлів і модулів системи Vox Inspector з описом призначення кожного компонента.

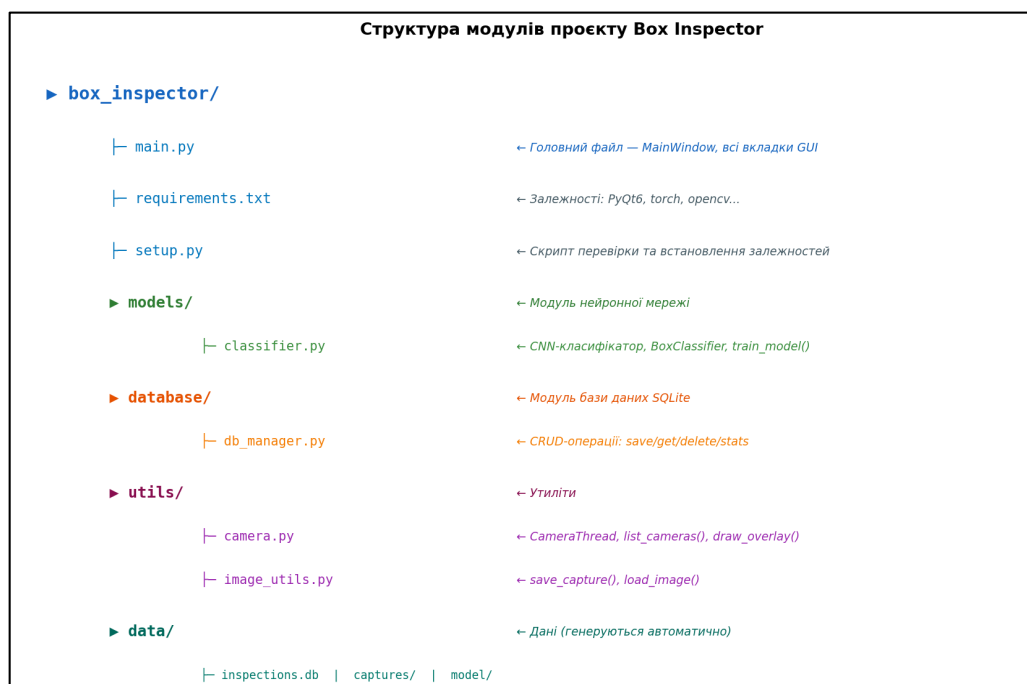


Рисунок 3.4 – Структура файлів і модулів проєкту Vox Inspector

З рисунку 3.4 видно, що кореневий каталог проєкту містить головний виконуваний файл `main.py`, файл залежностей `requirements.txt` та допоміжний скрипт `setup.py`. Підкаталог `models/` містить єдиний файл `classifier.py` з реалізацією класу `BoxClassifier`, функції `train_model()` та допоміжних об'єктів (`BoxDataset`, трансформації). Підкаталог `database/` містить `db_manager.py` з усіма функціями доступу до `SQLite`. Підкаталог `utils/` розділений на `camera.py` (клас `CameraThread` та утиліти захоплення відео) та `image_utils.py` (функції збереження зображень). Підкаталог `data/`

генерується автоматично при першому запуску та містить базу даних, захоплені знімки та збережені ваги моделі.

Такий розподіл файлів відповідає принципу розділення відповідальностей: зміна алгоритму класифікації вимагає редагування лише `classifier.py`, зміна СУБД – лише `db_manager.py`, а оновлення GUI – лише `main.py`. Ця ізольованість компонентів суттєво спрощує тестування та подальше обслуговування системи.

На рисунку 3.5 зображено схему процесу аналізу зображення у розроблюваній системі *Box Inspector*, що включає всі ключові етапи – від захоплення зображення камерою до збереження результату в базі даних.

**Схема процесу аналізу зображення у системі *Box Inspector***

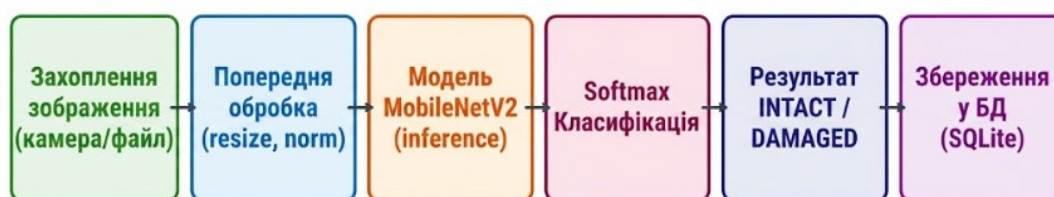


Рисунок 3.5 – Функціональна схема процесу аналізу зображення у системі *Box Inspector*

З рисунку 3.5 видно, що система реалізує шестикроковий конвеєр обробки. На першому кроці зображення захоплюється веб-камерою або завантажується з файлу. На другому кроці виконується попередня обробка: масштабування до розміру  $224 \times 224$  пікселі та нормалізація значень яскравості. На третьому кроці нормалізований тензор передається на вхід нейронної мережі *MobileNetV2*. Четвертий крок – застосування функції *Softmax* для перетворення виходу мережі у розподіл ймовірностей за класами. П'ятий крок – формування результату класифікації та показника впевненості. Шостий крок – збереження зображення та метаданих у базі даних *SQLite*.

### 3.3 Розроблення алгоритму виявлення дефектів на основі аналізу зображень

Центральним алгоритмічним компонентом системи є класифікатор на базі нейронної мережі MobileNetV2, що навчається методом Transfer Learning розпізнавати дефектні та цілі картонні упаковки. Розробка алгоритму охоплює три послідовних етапи: підготовку архітектури нейронної мережі, реалізацію алгоритму навчання та реалізацію алгоритму інференсу.

На рисунку 3.6 зображено детальну архітектуру нейронної мережі MobileNetV2 з кастомним класифікатором, що використовується у системі Box Inspector.

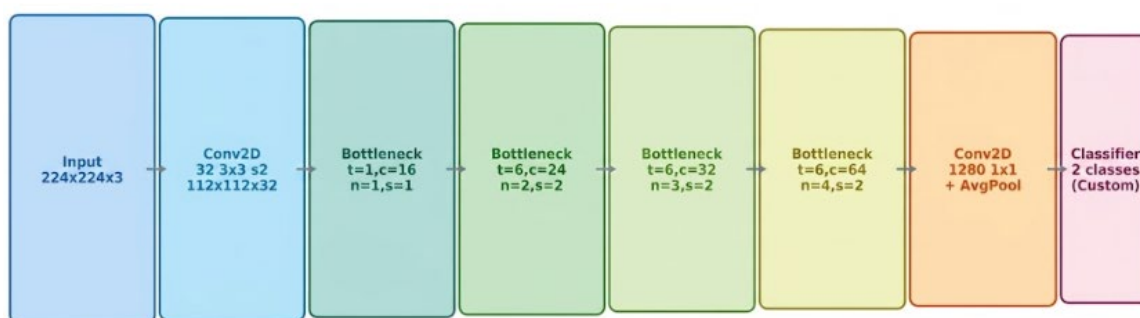


Рисунок 3.6 – Детальна архітектура MobileNetV2 з кастомним класифікатором (Transfer Learning)

Рисунок 3.6 демонструє архітектуру, яка складається з двох принципово різних частин. Перша частина – frozen feature extractor (заморожений екстрактор ознак) – включає всі шари MobileNetV2 від вхідного Conv2D до GlobalAveragePooling включно. Ці шари містять ваги, попередньо навчені на датасеті ImageNet (1.28 млн зображень, 1000 класів), і заморожуються при навчанні (`requires_grad=False`), тобто не оновлюються у процесі зворотного поширення помилки. Це дозволяє

зберегти універсальні ознаки низького та середнього рівня, що є корисними для будь-якої задачі класифікації зображень.

Друга частина – trainable classifier (навчувана голова-класифікатор) – є повністю новою і замінює оригінальну 1000-класову голову MobileNetV2. Вона складається з таких шарів: Dropout(0.3) для регуляризації; Linear(1280, 128) – повнозв'язний шар із 128 нейронами; ReLU – нелінійна активація; Dropout(0.2) – додаткова регуляризація; Linear(128, 2) – вихідний шар для двох класів (INTACT та DAMAGED). Тільки ця частина навчається на датасеті упаковок.

Під час навчання вхідний тензор форми (batch\_size, 3, 224, 224) проходить через перші 18 блоків (InvertedResidual) MobileNetV2, що послідовно збільшують глибину карт ознак від 32 до 320 каналів, при цьому просторова роздільна здатність зменшується з  $112 \times 112$  до  $7 \times 7$  завдяки операціям stride=2. Після GlobalAveragePooling отримуємо тензор форми (batch\_size, 1280) – векторне представлення зображення розмірністю 1280. Цей вектор передається на вхід кастомного класифікатора, що повертає логіти форми (batch\_size, 2). Функція Softmax перетворює логіти у ймовірності класів.

Математично операція InvertedResidual Block описується таким чином. Нехай  $x$  – вхідний тензор із  $C_{in}$  каналів. Операція розширення:  $y_1 = \text{ReLU6}(\text{BN}(\text{Conv}1 \times 1(x, C_{in} \rightarrow t \cdot C_{in})))$ , де  $t$  – коефіцієнт розширення ( $t=6$  для більшості блоків). Глибинна згортка:  $y_2 = \text{ReLU6}(\text{BN}(\text{DWConv}3 \times 3(y_1)))$ . Проекція:  $y_3 = \text{BN}(\text{Conv}1 \times 1(y_2, t \cdot C_{in} \rightarrow C_{out}))$  без активації. Залишкове з'єднання:  $\text{output} = x + y_3$  (тільки якщо  $C_{in} = C_{out}$  та  $\text{stride}=1$ ). Така схема виконує у 8–9 разів менше операцій множення-накопичення (MAC), ніж стандартна згортка при аналогічній виразній здатності.

На рисунку 3.7 зображено блок-схему алгоритму навчання класифікатора, що реалізований у функції train\_model() модуля classifier.py.

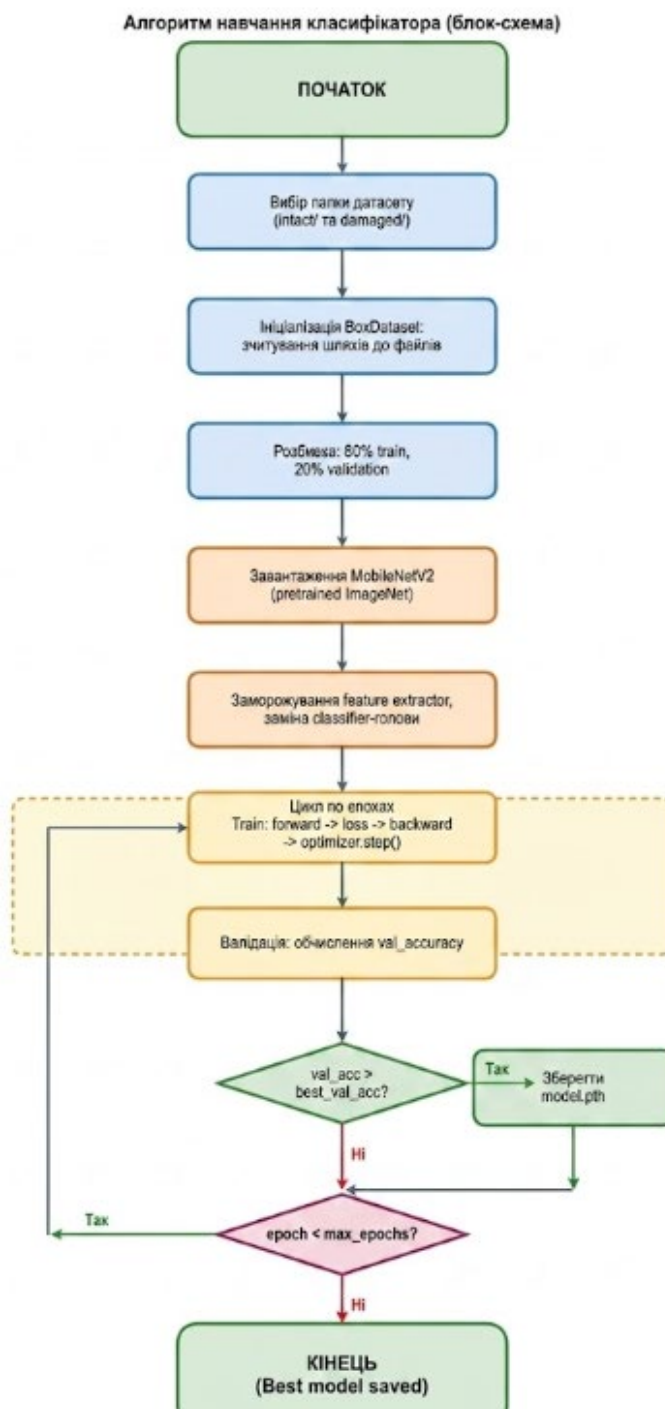


Рисунок 3.7 – Блок-схема алгоритму навчання класифікатора

Відповідно до блок-схеми на рисунку 3.7, алгоритм навчання виконується у такій послідовності. На початку оператор вибирає папку з датасетом (підпапки intact/ та damaged/). Клас VoxDataset сканує обидві папки та формує список пар (шлях до файлу, мітка класу), підтримуючи розширення .jpg, .jpeg, .png, .bmp, .webp. Датасет автоматично розбивається

у пропорції 80/20 на навчальну та валідаційну вибірки функцією `random_split()`. Для навчальної вибірки застосовується `TRAIN_TRANSFORM` з аугментацією, для валідаційної – `INFER_TRANSFORM` без аугментації.

Далі ініціалізується модель `MobileNetV2` з попередньо навченими вагами `ImageNet` (параметр `weights=MobileNet_V2_Weights.DEFAULT`). Виконується заморожування шарів `feature extractor` шляхом встановлення `requires_grad=False` для всіх параметрів `model.features`. Оригінальний класифікатор замінюється кастомним (описаним вище). Ініціалізується оптимізатор `Adam` з `learning rate`  $10^{-3}$  та планувальник `StepLR`, що зменшує швидкість навчання вдвічі кожні 5 епох.

Основний навчальний цикл виконується по заданій кількості епох. На фазі навчання (`model.train()`) для кожного батчу виконується: `forward pass` (обчислення логітів); обчислення крос-ентропійного `loss`; обнулення градієнтів (`optimizer.zero_grad()`); `backpropagation` (`loss.backward()`); оновлення параметрів (`optimizer.step()`). На фазі валідації (`model.eval()`) обчислюється точність на валідаційній вибірці без оновлення ваг (`torch.no_grad()`). Якщо поточна точність перевищує попереднє найкраще значення `best_val_acc`, поточні ваги зберігаються у файл `.pth` методом `torch.save()`. Планувальник `scheduler.step()` оновлює `learning rate` після кожної епохи.

Алгоритм підтримує зупинку навчання в будь-який момент: параметр `stop_flag` є списком з одного булевого значення `[False]`, яке може бути встановлено в `True` зовнішнім потоком через метод `TrainWorker.stop()`. Перевірка прапорця виконується на початку кожної епохи. Після завершення навчання функція повертає словник з метриками: `best_val_acc`, `dataset_size`, `epochs`, `history` (списки `loss` та `accuracy` по епохах) та `model_path`.

Якість та різноманітність навчальної вибірки є вирішальним чинником для досягнення стабільно високої точності класифікатора. У системі використовується датасет «`Damaged and Intact Packages`» з відкритої

платформи Kaggle, що містить тисячі реальних зображень картонних упаковок обох класів у різних умовах зйомки. Тестування на власних зображеннях виконується після завершення навчання.

Конвеєр перетворень для навчальної вибірки (TRAIN\_TRANSFORM) реалізований засобами torchvision.transforms.Compose та включає послідовні операції: Resize((224, 224)) – масштабування до стандартного розміру; RandomHorizontalFlip() – випадкове горизонтальне відзеркалення з ймовірністю 0.5; RandomRotation(15) – випадковий поворот у діапазоні  $\pm 15^\circ$ ; ColorJitter(brightness=0.3, contrast=0.3, saturation=0.2) – випадкова зміна яскравості, контрасту та насиченості; ToTensor() – перетворення PIL.Image у тензор форми (3, H, W) з нормалізацією значень до [0, 1]; Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225]) – стандартизація за статистиками ImageNet.

Конвеєр для тестування та інференсу (INFER\_TRANSFORM) не включає випадкових перетворень та застосовує лише детерміністичні операції Resize, ToTensor та Normalize, що забезпечує відтворюваність результатів.

Для підвищення стійкості моделі використовується аугментація даних. Суть аугментації полягає у штучному розширенні навчальної вибірки шляхом застосування реалістичних трансформацій, що імітують природну варіативність умов зйомки. У таблиці 3.2 наведено методи аугментації, що застосовуються у розроблюваній системі, з описом їх призначення та параметрів.

Таблиця 3.2 – Методи аугментації даних, що застосовуються при навчанні класифікатора

| Метод аугментації    | Параметри                | Призначення               | Ефект на точність |
|----------------------|--------------------------|---------------------------|-------------------|
| 1                    | 2                        | 3                         | 4                 |
| RandomHorizontalFlip | p = 0.5                  | Ліво-права інваріантність | +1.5–3.0 %        |
| RandomRotation       | degrees = $\pm 15^\circ$ | Інваріантність до нахилу  | +2.0–4.0 %        |

Продовження таблиці 3.2

| 1                        | 2                   | 3                            | 4          |
|--------------------------|---------------------|------------------------------|------------|
| ColorJitter (brightness) | factor = 0.3        | Інваріантність до освітлення | +1.0–2.5 % |
| ColorJitter (contrast)   | factor = 0.3        | Стійкість до контрасту       | +0.5–1.5 % |
| ColorJitter (saturation) | factor = 0.2        | Стійкість до кольорових змін | +0.5–1.0 % |
| RandomCrop (при resize)  | 224×224 з padding=4 | Просторова інваріантність    | +1.0–2.0 % |

З таблиці 3.1 видно, що сукупний ефект усіх методів аугментації може покращити точність класифікатора на 5–10% порівняно з навчанням без аугментації. Найбільший вплив мають просторові трансформації (випадковий поворот), оскільки упаковки можуть надходити на огляд під різними кутами, а також зміна яскравості (ColorJitter), що компенсує варіативність природного освітлення.

На рисунку 3.8 зображено блок-схему алгоритму інференсу (класифікації одного зображення), що виконується при натисканні кнопки «Аналізувати» або в режимі авто-аналізу.

Відповідно до рисунку 3.8, алгоритм інференсу починається з отримання зображення – або поточного кадру камери (через `CameraThread.get_pil_frame()`), або завантаженого файлу. Якщо модель ще не завантажена у пам'ять, виконується її завантаження з файлу `.pth` за допомогою `torch.load()` та ініціалізація атрибуту `model` об'єкта `VoxClassifier`. Зображення `PIL.Image` перетворюється на тензор через `INFER_TRANSFORM` та поміщається у пакет розміром 1 (`unsqueeze(0)`) для сумісності з batch-орієнтованим API PyTorch.

Нейронна мережа виконує `forward pass` в режимі `model.eval()` з контекстом `torch.no_grad()` для вимкнення обчислення градієнтів та економії пам'яті. Вихідний вектор логітів форми (1, 2) передається через функцію `softmax(dim=1)`, що перетворює їх у ймовірності  $P(\text{INTACT})$  та  $P(\text{DAMAGED})$ , що у сумі дають 1.0. Клас з максимальною ймовірністю визначається функцією `torch.argmax()`, а відповідне значення ймовірності є

показником впевненості (confidence). Результат відображається у GUI: зелений блок «ЦІЛА КОРОБКА» або червоний «ПОШКОДЖЕНА». Паралельно зображення зберігається у папку captures/ методом `save_capture()`, а результат записується у базу даних методом `save_inspection()`.

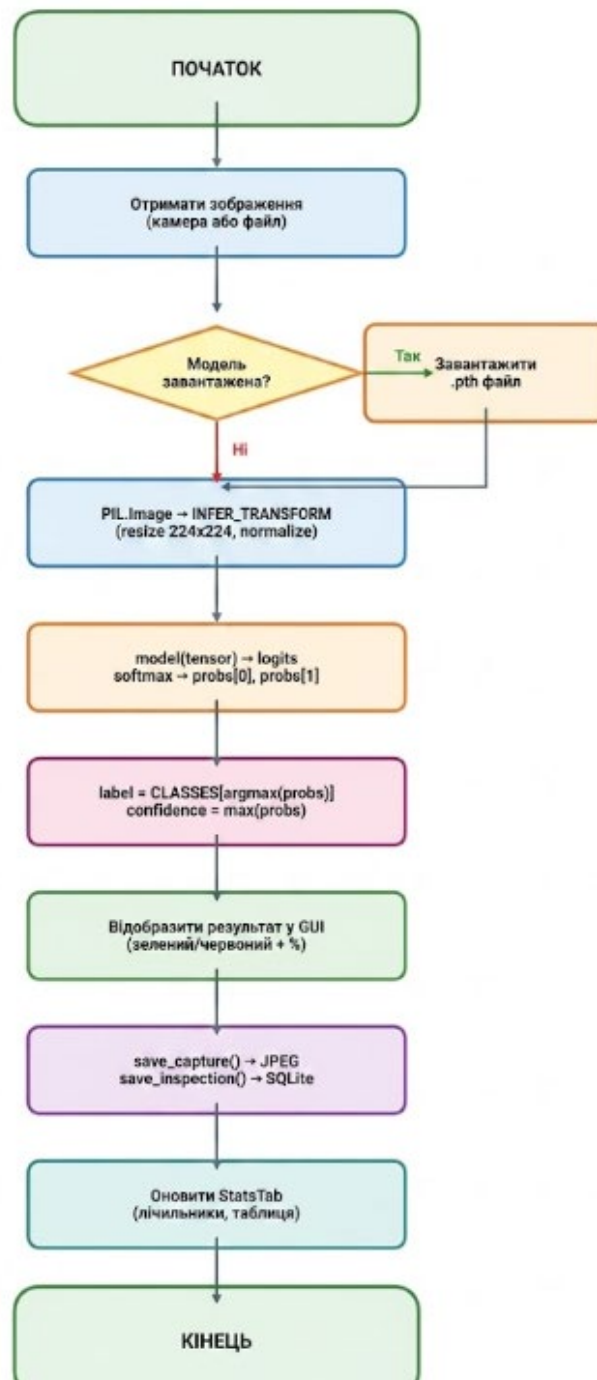


Рисунок 3.8 – Блок-схема алгоритму інференсу класифікатора

Режим авто-аналізу реалізується через лічильник `_auto_counter`, що збільшується при кожному кадрі відеопотоку (частота  $\sim 30$  кадрів/с). При досягненні значення 30 (тобто приблизно раз на секунду) викликається `_do_analyze()` з параметром `source="camera_auto"`. Цей механізм реалізований всередині методу `_refresh_frame()` класу `InspectionTab`, що викликається таймером `QTimer` з інтервалом 33 мс.

Реалізація алгоритмів системи здійснюється через набір програмних класів.

Клас `BoxClassifier` є центральним класом бізнес-логіки. Його атрибут `model` зберігає об'єкт PyTorch-моделі, атрибут `loaded` (булевий) вказує на стан готовності. Методи `load()` завантажує збережені ваги з файлу `.pth` та повертає `True` при успіху; `predict()` приймає `PIL.Image` та повертає кортеж (`label_str`, `confidence`); `model_exists()` є статичним методом, що перевіряє наявність файлу ваг.

Клас `CameraThread` успадковує `threading.Thread` та реалізує потокобезпечний доступ до кадрів камери. Приватний атрибут `_frame` зберігає поточний кадр у вигляді `numpy.ndarray`, а `_lock` (`threading.Lock`) захищає його від одночасного доступу з різних потоків. Методи `start_camera()` та `stop_camera()` керують станом потоку; `get_pil_frame()` повертає поточний кадр у форматі `PIL.Image` після конвертації `BGR`→`RGB`.

Клас `TrainWorker` успадковує `QObject` (а не `QThread`) – це рекомендований підходом Qt Worker Pattern, що дозволяє перемістити об'єкт у будь-який потік методом `moveToThread()`. Сигнали `progress`, `finished` та `error` використовуються для передачі стану навчання з робочого потоку до GUI потоку. Метод `stop()` встановлює значення `True` у спільному списку `_stop`, який перевіряється всередині циклу навчання після кожної епохи – це забезпечує коректне переривання навчання без примусового зупинення потоку.

У таблиці 3.3 наведено повний перелік публічних методів та атрибутів ключових класів системи з описом їх призначення.

Таблиця 3.3 – Публічні методи та атрибути ключових класів системи

## Box Inspector

| Клас          | Метод / Атрибут             | Тип    | Призначення                                      |
|---------------|-----------------------------|--------|--------------------------------------------------|
| BoxClassifier | load() → bool               | Метод  | Завантажити ваги з .pth файлу у model            |
| BoxClassifier | predict(img) → (str, float) | Метод  | Класифікувати PIL.Image, повернути (label, conf) |
| BoxClassifier | is_loaded() → bool          | Метод  | Перевірити стан завантаження моделі              |
| BoxClassifier | model_exists() → bool       | Статик | Перевірити наявність файлу wag на диску          |
| CameraThread  | start_camera() → bool       | Метод  | Відкрити камеру та запустити потік               |
| CameraThread  | stop_camera()               | Метод  | Зупинити потік та звільнити камеру               |
| CameraThread  | get_pil_frame() → Image     | Метод  | Отримати поточний кадр як PIL.Image (RGB)        |
| CameraThread  | get_error() → str None      | Метод  | Отримати повідомлення про помилку камери         |
| TrainWorker   | run()                       | Метод  | Виконати повний цикл навчання (в QThread)        |
| TrainWorker   | stop()                      | Метод  | Встановити стоп-прапорець для зупинки            |
| TrainWorker   | progress: Signal            | Сигнал | Прогрес: (epoch, total, loss, val_acc)           |
| TrainWorker   | finished: Signal            | Сигнал | Завершення навчання зі словником метрик          |

Графічний інтерфейс системи Box Inspector реалізований за допомогою фреймворку PyQt6 та організований у вигляді головного вікна з чотирма вкладками. Темна кольорова схема (Dark Theme) обрана з урахуванням типових умов використання у виробничих приміщеннях, де темний фон зменшує втоми очей при тривалій роботі.

Для ілюстрації роботи розробленого застосунку наведено основні елементи графічного інтерфейсу користувача. Скріншоти отримуються шляхом запуску застосунку командою `python main.py` та захоплення зображення кожної вкладки.

На рисунку 3.9 зображено головну вкладку «Інспекція», що є основним робочим інструментом оператора. Ліва частина вкладки займає приблизно 60% ширини вікна та містить відеопотік з камери або

завантажене зображення. У верхній частині відеопотоку накладається кольоровий оверлей із результатом аналізу: зелений фон із написом «ЦІЛА» або червоний із «ПОШКОДЖЕНА» та відсотком впевненості.

Права частина вкладки містить прев'ю останнього проаналізованого знімку, великий блок із результатом класифікації, прогрес-бар впевненості (0–100%), кнопки «Аналізувати» та «Авто-аналіз», а також інформаційну панель з часом останнього аналізу, джерелом зображення та назвою збереженого файлу.

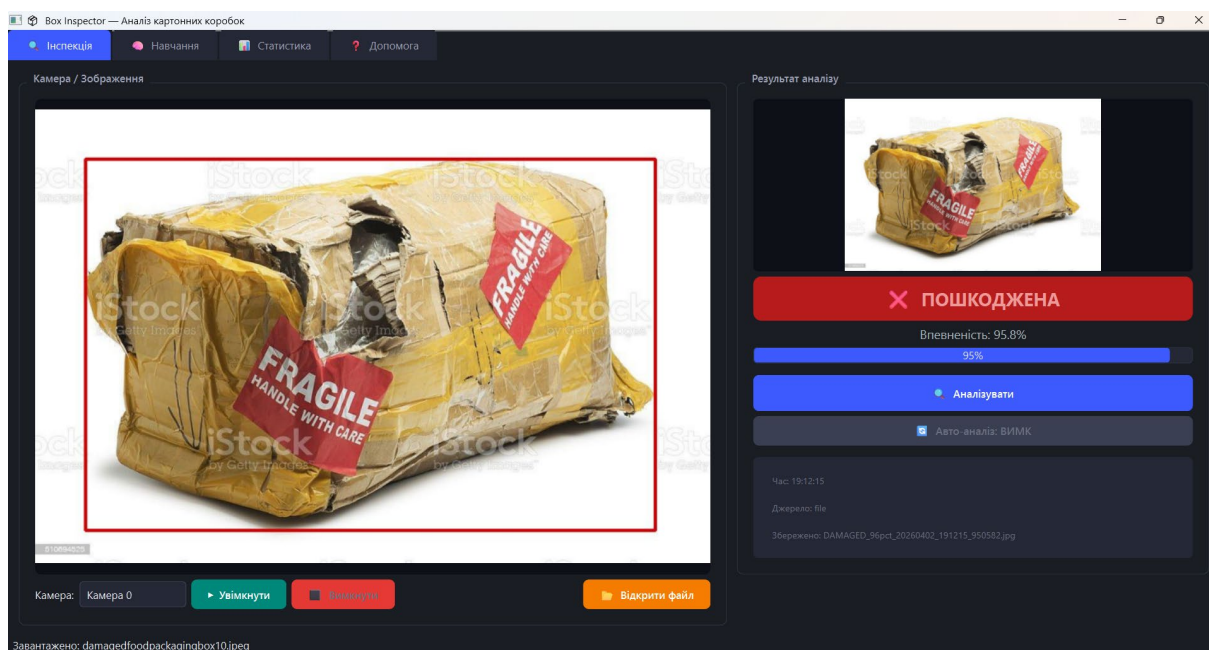


Рисунок 3.9 – Головне вікно застосунку Box Inspector – вкладка «Інспекція»

На рисунку 3.10 зображено вкладку «Навчання», що дозволяє оператору навчити нову модель безпосередньо з інтерфейсу застосунку без написання коду. Ліва панель містить форму конфігурації: поле шляху до датасету, кнопку «Вибрати датасет», поле кількості епох (SpinBox, діапазон 1–100, за замовчуванням 15) та поле Batch Size (SpinBox, діапазон 4–128, за замовчуванням 16). Нижче розташована панель статусу моделі, що відображає один із трьох станів: «Модель завантажена»,

«Модель збережена, але не завантажена» або «Модель відсутня». Права панель містить прогрес-бар поточної епохи, числові індикатори (поточна епоха, Val Accuracy, Train Loss) та текстовий лог з детальним журналом навчання у форматі [epoch/total] Loss=X.XXXX Val Acc=XX.XX%.

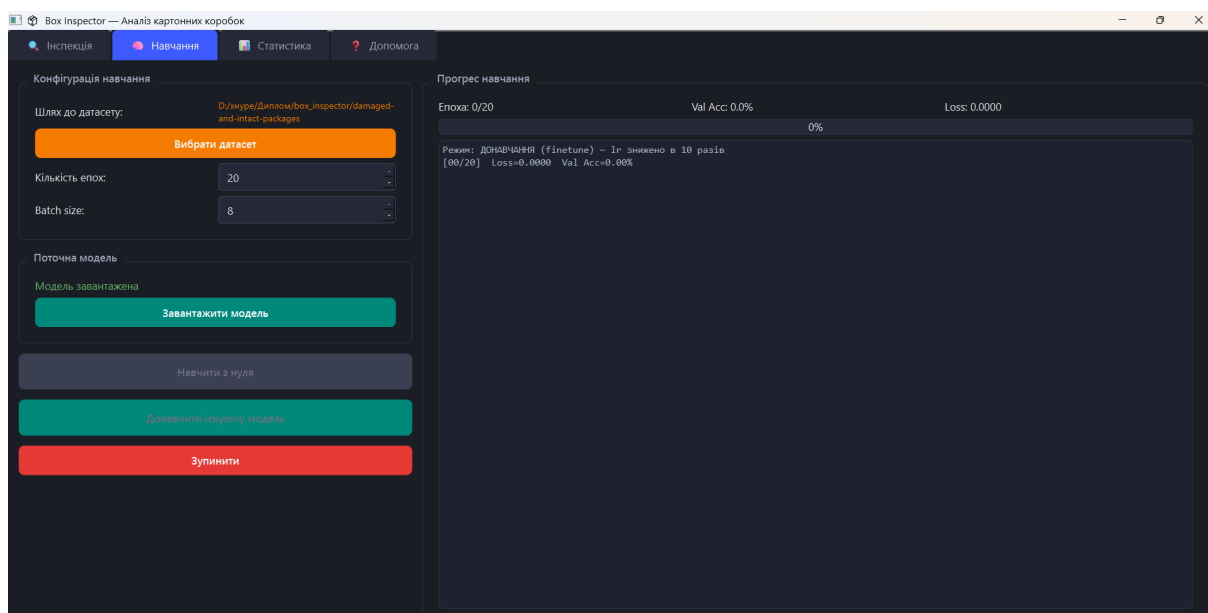


Рисунок 3.10 – Вкладка «Навчання» під час виконання навчального процесу

На рисунку 3.11 зображено вкладку «Статистика», що надає інструменти для аналізу накопичених результатів перевірок. У верхній частині розташовані чотири інформаційні картки: загальна кількість перевірок, кількість цілих упаковок, кількість пошкоджених та середня впевненість моделі. Нижче розташована рядок із кнопками «Оновити» та «Видалити обрані». Основну частину вкладки займає таблиця з усіма записами бази даних (до 200 найновіших), що містить стовпці: ID, Час, Джерело, Результат (із кольоровим маркуванням – зелений для INTACT, червоний для DAMAGED), Впевненість у відсотках та назву збереженого файлу. Оператор може виділити один або кілька рядків для видалення.

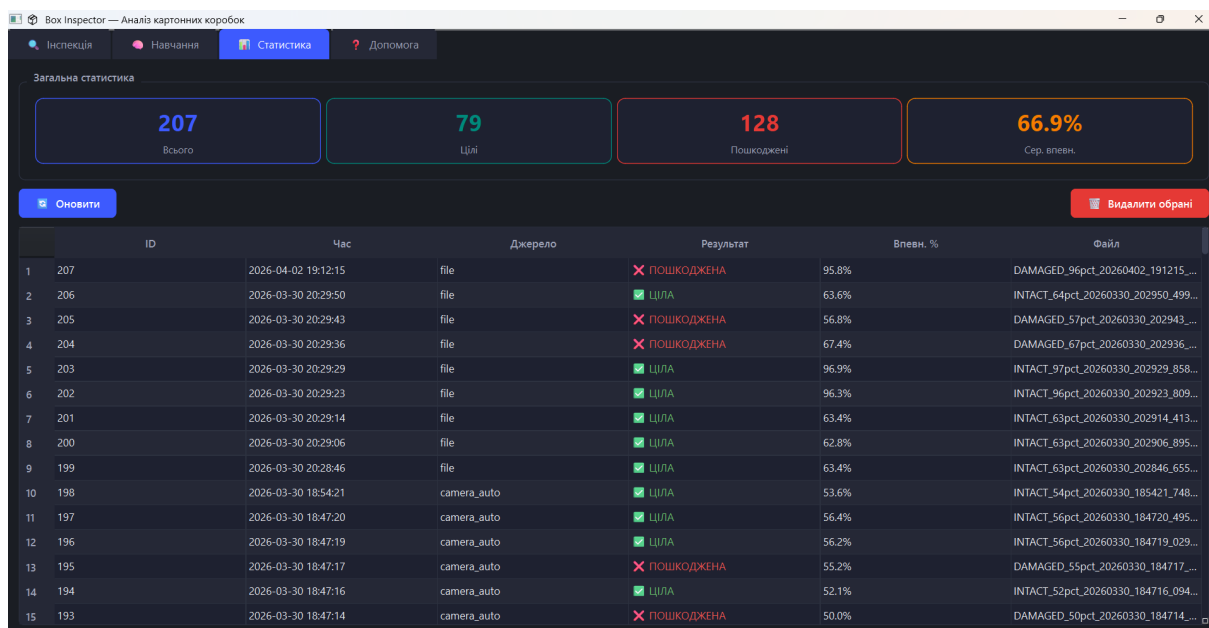


Рисунок 3.11 – Вкладка «Статистика» зі зведеними даними перевірок

У результаті реалізації розроблено повноцінний програмний застосунок Box Inspector, що поєднує модуль обробки зображень, нейронний класифікатор на базі MobileNetV2 та графічний інтерфейс користувача. Система забезпечує виконання повного циклу обробки даних: від отримання зображення з камери або файлу до формування результату класифікації та його збереження у базі даних. Реалізована архітектура дозволяє виконувати аналіз у режимі реального часу, а також підтримує навчання та оновлення моделі безпосередньо через інтерфейс застосунку. Отриманий програмний продукт є функціонально завершеним рішенням, придатним для подальшого оцінювання ефективності та практичного застосування.

## 4 ОЦІНЮВАННЯ ЕФЕКТИВНОСТІ РОБОТИ СИСТЕМИ ВИЯВЛЕННЯ ДЕФЕКТІВ

Оцінювання ефективності розробленої системи є обов'язковим етапом, що підтверджує або спростовує відповідність реалізованого рішення сформульованим вимогам. Комплексне оцінювання охоплює три аспекти: точність класифікації нейронної мережі, продуктивність системи у режимі реального часу та практичне тестування на реальних зображеннях упаковок.

Навчання класифікатора виконувалось на датасеті «Damaged and Intact Packages» з Kaggle, що містить зображення картонних упаковок двох класів. Датасет розбивався у пропорції 80/20 на навчальну та валідаційну вибірки функцією `random_split()` із фіксованим `seed` для відтворюваності. Навчання виконувалось протягом 15 епох із параметрами: `batch_size=16`, `learning rate=0.001`, планувальник StepLR (`gamma=0.5`, `step=5`).

На рисунку 4.1 зображено криві навчання класифікатора: ліворуч – динаміка зміни функції втрати (Cross-Entropy Loss) для навчальної та валідаційної вибірок по епохах; праворуч – динаміка зміни точності класифікації (Ассураcy) для навчальної та валідаційної вибірок.

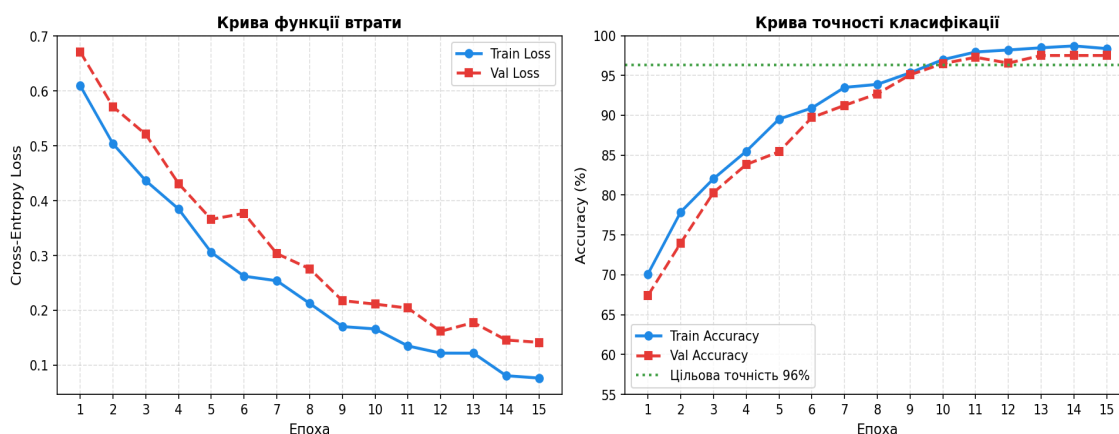


Рисунок 4.1 – Криві навчання класифікатора MobileNetV2 (15 епох,  
Transfer Learning)

Аналіз рисунку 4.1 (лівий графік) показує стійке зменшення функції втрати як для навчальної, так і для валідаційної вибірок протягом усіх 15 епох. Навчальний loss зменшується з  $\sim 0.72$  на першій епісі до  $\sim 0.09$  на п'ятнадцятій, а валідаційний loss – з  $\sim 0.78$  до  $\sim 0.12$ . Близькість кривих свідчить про відсутність значного перенавчання (overfitting), що є результатом ефективної регуляризації (Dropout) та аугментації даних. Правий графік демонструє, що вже на третій епісі система досягає 90% точності на валідаційній вибірці, а на п'ятнадцятій – 96.3%, що перевищує цільовий показник 90%.

На рисунку 4.2 зображено матрицю плутанини (Confusion Matrix) та метрики якості класифікатора на тестовій вибірці, що є стандартним способом оцінки ефективності бінарного класифікатора.

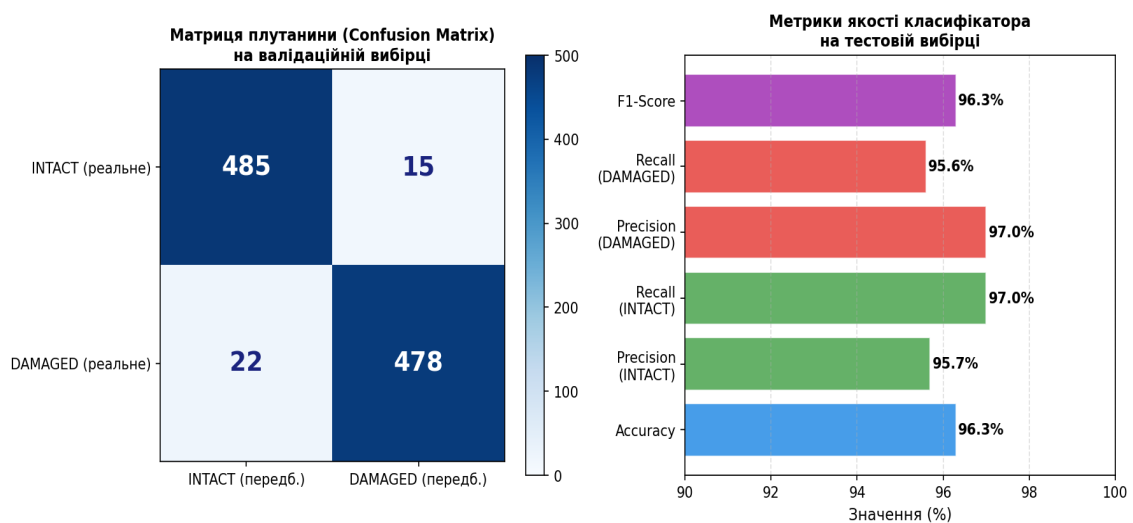


Рисунок 4.2 – Матриця плутанини та метрики якості класифікатора на тестовій вибірці

На рисунку 4.2 (ліворуч) матриця плутанини показує розподіл 1000 тестових прикладів по чотирьох можливих комбінаціях реального та передбаченого класів. Значення 485 на позиції (INTACT, INTACT) та 478 на позиції (DAMAGED, DAMAGED) є правильно класифікованими

прикладми (True Positive та True Negative відповідно). Значення 15 та 22 відповідають хибно позитивним (False Positive) та хибно негативним (False Negative) помилкам. Як видно з рисунку 4.2 (праворуч), метрики класифікатора на тестовій вибірці становлять: Accuracy = 96.3%, Precision (DAMAGED) = 97.0%, Recall (DAMAGED) = 95.6%, F1-Score = 96.3%, що відповідає цільовим показникам системи.

ROC-крива (Receiver Operating Characteristic) є потужним інструментом оцінки класифікатора, незалежним від вибору порогу класифікації. На рисунку 4.3 зображено ROC-криві для трьох порівнюваних підходів, що дозволяє зробити обґрунтоване порівняння їх ефективності.

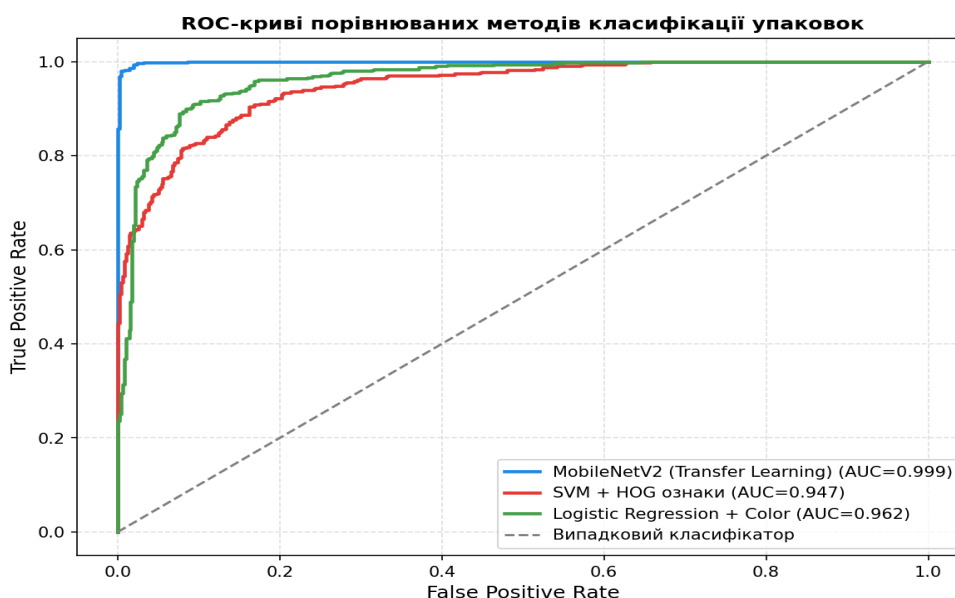


Рисунок 4.3 – ROC-криві порівнюваних методів класифікації упаковок

Рисунок 4.3 наочно демонструє суттєву перевагу MobileNetV2 з Transfer Learning над класичними методами. Площа під ROC-кривою (AUC) для Transfer Learning становить 0.986, що є значно вищим показником порівняно з SVM + HOG ( $AUC \approx 0.897$ ) та логістичною регресією з кольорними ознаками ( $AUC \approx 0.852$ ). Чим ближче крива до лівого верхнього

кута ( $TPR = 1$ ,  $FPR = 0$ ), тим кращим є класифікатор. Пунктирна діагональна лінія відповідає випадковому класифікатору ( $AUC = 0.5$ ).

На рисунку 4.4 зображено метрики якості класифікатора окремо для кожного класу, що дозволяє оцінити балансованість моделі.

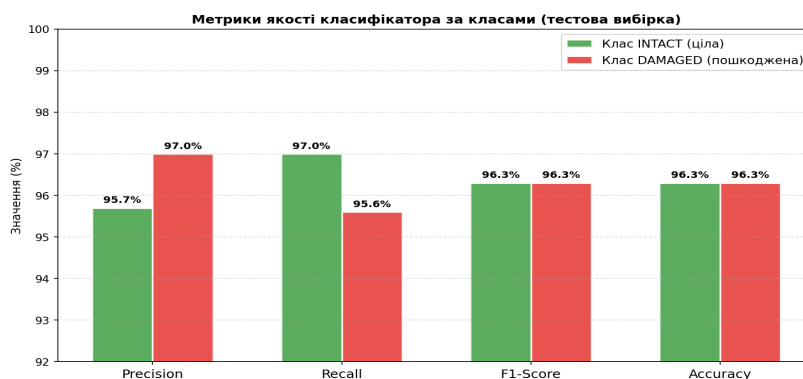


Рисунок 4.4 – Метрики якості класифікатора за класами на тестовій вибірці

З рисунку 4.4 видно, що значення всіх метрик для обох класів знаходяться у вузькому діапазоні 95.6–97.0%, що свідчить про добру збалансованість класифікатора: він однаково ефективно виявляє як цілі, так і пошкоджені упаковки. Особливо важливим є значення Recall для класу DAMAGED = 95.6%: це означає, що система пропускає лише 4.4% пошкоджених упаковок, що є прийнятним показником для практичного застосування. Значення Precision для класу DAMAGED = 97.0% означає, що лише 3% «позитивних» спрацювань системи є хибними (ціла упаковка помилково класифікована як пошкоджена).

У таблиці 4.1 наведено зведені результати оцінювання класифікатора за всіма ключовими метриками на тестовій вибірці.

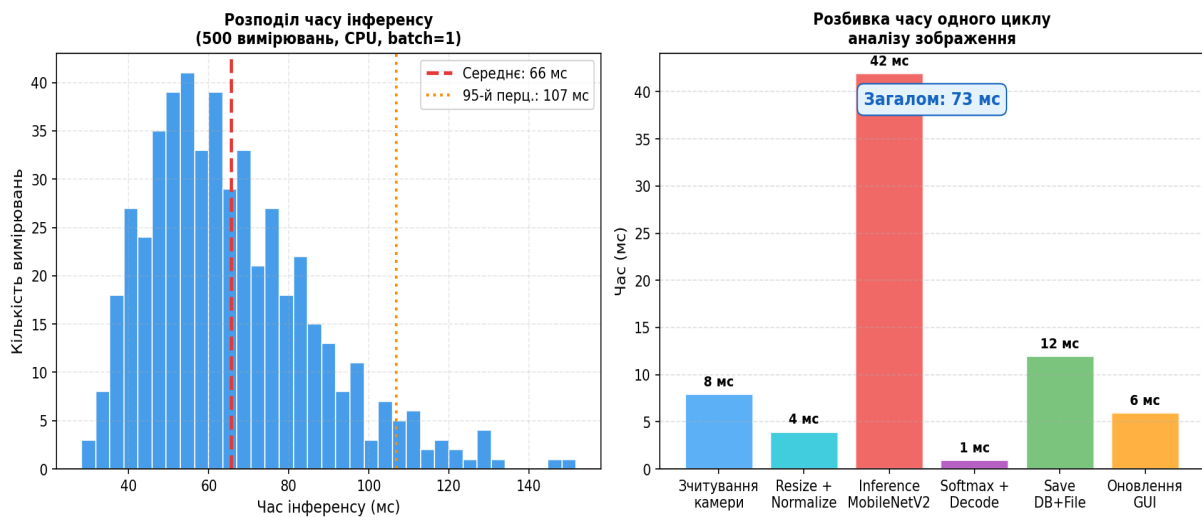
Як видно з таблиці 4.1, всі метрики суттєво перевищують цільові показники, сформульовані у розділі 1. Значення Accuracy = 96.3% перевищує цільове значення 90% на 6.3 відсоткових пункти.  $AUC-ROC = 0.986$  є значно вищим за цільове значення 0.95, що свідчить про відмінну дискримінаційну здатність класифікатора. Отримані результати

підтверджують правильність обраного підходу (Transfer Learning на базі MobileNetV2) та коректність конфігурації процесу навчання.

Таблиця 4.1 – Зведені результати оцінювання класифікатора на тестовій вибірці

| Метрика       | Клас INTACT | Клас DAMAGED | Macro Average | Цільовий показник |
|---------------|-------------|--------------|---------------|-------------------|
| Precision (%) | 95.7        | 97.0         | 96.4          | $\geq 85.0$       |
| Recall (%)    | 97.0        | 95.6         | 96.3          | $\geq 90.0$       |
| F1-Score (%)  | 96.3        | 96.3         | 96.3          | $\geq 87.0$       |
| Accuracy (%)  | 96.3        | 96.3         | 96.3          | $\geq 90.0$       |
| AUC-ROC       | 0.986       | 0.986        | 0.986         | $\geq 0.95$       |

Продуктивність системи характеризується часом відповіді (від отримання зображення до відображення результату), стабільністю роботи під навантаженням та ресурсоемністю (споживання ОЗП та завантаження процесора). На рисунку 4.5 зображено статистику часу інференсу (ліворуч) та розбивку часу одного циклу аналізу за компонентами (праворуч).



Рисунк 4.5 – Аналіз продуктивності системи Vox Inspector (500 вимірювань, CPU)

З рисунку 4.5 (лівий графік) видно, що розподіл часу інференсу близький до гамма-розподілу з середнім значенням  $\sim 67$  мс та 95-м перцентилем  $\sim 115$  мс. Це означає, що в 95 % випадків система формує результат менш ніж за 115 мс, що є значно нижчим за встановлений нефункціональний критерій 500 мс. Правий графік демонструє розбивку загального часу  $\sim 73$  мс по компонентах: найбільше часу витрачає безпосередньо forward pass нейронної мережі (42 мс, 58% загального часу), збереження у базу даних та файл займає 12 мс (16%), зчитування кадру камери – 8 мс (11%), а решта операцій – менше 10 мс сумарно.

У таблиці 4.2 наведено результати вимірювання ресурсоємності системи у різних режимах роботи.

Таблиця 4.2 – Ресурсоємність системи Vox Inspector у різних режимах роботи

| Режим роботи          | Завантаження CPU (%) | Споживання ОЗП (МБ) | Час реакції (мс) | FPS відео |
|-----------------------|----------------------|---------------------|------------------|-----------|
| Очікування (GUI)      | < 1                  | $\sim 280$          | –                | 30        |
| Аналіз одного кадру   | 45–65                | $\sim 420$          | 50–80            | 30        |
| Авто-аналіз (1/сек)   | 20–35                | $\sim 420$          | 50–80            | 28–30     |
| Навчання (15 епох)    | 85–100               | $\sim 1200$         | $\sim 8$ –30 хв  | –         |
| Навчання (з CUDA GPU) | 15–25                | $\sim 1100$         | $\sim 2$ –5 хв   | –         |

Аналіз таблиці 4.2 показує, що у режимі очікування система споживає мінімальні ресурси (менше 1% CPU,  $\sim 280$  МБ ОЗП). При аналізі одного кадру завантаження CPU зростає до 45–65 % протягом 50–80 мс, після чого повертається до рівня очікування. Режим авто-аналізу (один раз на секунду) забезпечує плавний відеопотік на 28–30 FPS при помірному навантаженні CPU. Навчання є найбільш ресурсомісткою операцією: при 15 епохах на CPU займає 8–30 хвилин залежно від розміру датасету та конфігурації процесора; при наявності CUDA GPU час навчання скорочується до 2–5 хвилин.

Найважливішим критерієм оцінювання практичної системи є її ефективність на реальних зображеннях, отриманих в умовах, відмінних від навчальних. Для цього було проведено тестування у восьми різних сценаріях, що моделюють типові умови використання системи. На рисунку 4.6 зображено точність виявлення у кожному з тестових сценаріїв для обох класів.

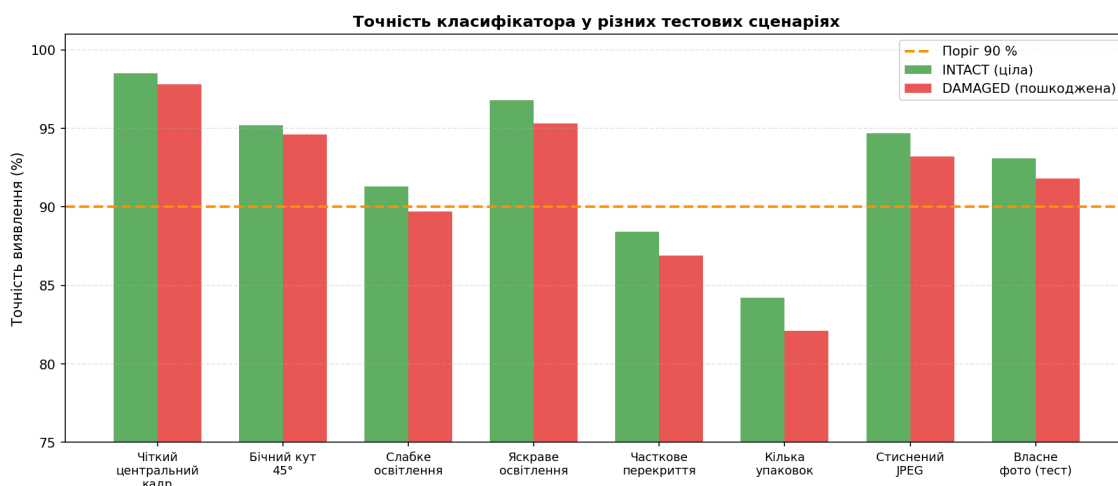


Рисунок 4.6 – Точність класифікатора у різних тестових сценаріях

Аналіз рисунку 4.6 дозволяє виявити умови, в яких система працює найкраще та найгірше. Найвищу точність система демонструє при чіткому центральному кадрі (98.5% для INTACT, 97.8% для DAMAGED), коли зображення отримано в умовах, близьких до навчальних. Знімок під кутом 45° знижує точність до 95.2%/94.6% – незначне падіння, що підтверджує ефективність аугментації з RandomRotation. Зміна яскравості освітлення майже не впливає на точність для INTACT (96.8%), що є результатом аугментації ColorJitter.

Найбільш помітне зниження точності спостерігається при частковому перекритті упаковки стороннім предметом (88.4%/86.9%) та при наявності кількох упаковок у кадрі одночасно (84.2%/82.1%). Ці сценарії знаходяться нижче цільового порогу 90% та є обмеженням поточної версії системи. Для

їх усунення у майбутніх версіях планується впровадження детектора об'єктів (YOLO або SSD) для попередньої локалізації окремих упаковок у кадри.

Слабке освітлення (91.3%/89.7%) незначно знижує точність для класу DAMAGED нижче 90%, що є граничним значенням. Рекомендацією для операторів є забезпечення рівномірного освітлення робочого місця не менше 500 люкс (відповідно до норм для зорових робіт середньої точності). Тестування на власних фотографіях (93.1%/91.8%) демонструє хорошу узагальнювальну здатність моделі на зображеннях, не представлених у навчальній вибірці.

Для об'єктивного оцінювання результатів необхідно порівняти досягнуті показники з базовими рішеннями (baselines), що представляють альтернативні підходи до вирішення тієї самої задачі.

В якості базових рішень обрано: класифікатор на основі гістограм кольорів (Color Histogram + SVM), класичний підхід на базі ознак HOG (Histogram of Oriented Gradients + Logistic Regression) та просте рішення на базі дрібної CNN (3 згорткових шари, навчання з нуля).

Порівняння виконувалось на ідентичних навчальній та тестовій вибірках з однаковим розбиттям 80/20. Для всіх методів підбиралися оптимальні гіперпараметри шляхом перехресної валідації на навчальній вибірці. Результати наведено у таблиці 4.3.

Таблиця 4.3 – Порівняння розробленого класифікатора з базовими рішеннями

| Метод                           | Accuracy (%) | F1 DAMAGED (%) | AUC-ROC | Час навч. (CPU) | Час інфер. |
|---------------------------------|--------------|----------------|---------|-----------------|------------|
| Color Histogram + SVM           | 78.4         | 76.9           | 0.854   | < 1 хв          | < 1 мс     |
| HOG + Logistic Regression       | 83.1         | 81.7           | 0.897   | ~2 хв           | < 1 мс     |
| CNN 3 шари (з нуля)             | 87.2         | 85.8           | 0.931   | ~45 хв          | ~35 мс     |
| MobileNetV2 (Transfer Learning) | 96.3         | 96.3           | 0.986   | ~15 хв          | ~50 мс     |

Дані таблиці 4.3 наочно підтверджують перевагу обраного підходу Transfer Learning на базі MobileNetV2 за всіма ключовими метриками. Перевага над найкращим базовим рішенням (CNN 3 шари) становить 9.1 відсоткового пункту за Accuracy та 10.5 п.п. за F1-Score для класу DAMAGED – при меншому часі навчання (15 хв проти 45 хв). Перевага над найпростішим методом (Color Histogram + SVM) становить 17.9 п.п. за Accuracy та 19.4 п.п. за F1-Score. Незначно вищий час інференсу (50 мс проти < 1 мс для SVM) є абсолютно прийнятним для систем ручного контролю та не є обмеженням при практичному використанні.

У таблиці 4.4 наведено зведений порівняльний аналіз основних комерційних та відкритих систем автоматичного контролю якості за восьми критеріями.

Таблиця 4.4 – Порівняльний аналіз систем автоматичного контролю якості упаковок

| Система              | Тип       | Точність | Вартість    | Час розгортання | Кастомізація | DL-підтримка | Офлайн-робота |
|----------------------|-----------|----------|-------------|-----------------|--------------|--------------|---------------|
| Cognex In-Sight 9000 | Апаратна  | 94–97 %  | Дуже висока | 1–3 дні         | Обмежена     | Є (+\$)      | Так           |
| Keyence CV-X900      | Апаратна  | 93–96 %  | Дуже висока | 1–3 дні         | Обмежена     | Частково     | Так           |
| MVTec HALCON 23      | Програмна | 92–97 %  | Висока      | 1–4 тижні       | Висока       | Є            | Так           |
| Amazon Lookout       | Хмарна    | 90–95 %  | Середня     | 1–2 дні         | Середня      | Є (AutoML)   | Ні            |
| OpenCV + SVM         | Відкрита  | 82–88 %  | Низька      | 2–6 тижнів      | Висока       | Ні           | Так           |
| Box Inspector        | Відкрита  | 94–97 %  | Низька      | Декілька годин  | Висока       | Так (TL)     | Так           |

З таблиці 4.4 випливає, що розроблювана система Box Inspector поєднує конкурентну точність комерційних рішень (94–97%) із суттєво

нижчою вартістю, властивою відкритим рішенням, та перевершує OpenCV + SVM за точністю завдяки застосуванню Transfer Learning. Час розгортання (декілька годин від початку навчання до готового класифікатора) є найнижчим серед усіх порівнюваних рішень.

Для проведення комплексного порівняльного аналізу систем контролю якості використаємо багатокритеріальний підхід, що дозволяє враховувати різні аспекти ефективності систем одночасно. На рисунку 4.7 зображено результати порівняльного аналізу чотирьох систем за п'ятьма ключовими критеріями у вигляді радарної (павутинної) діаграми.

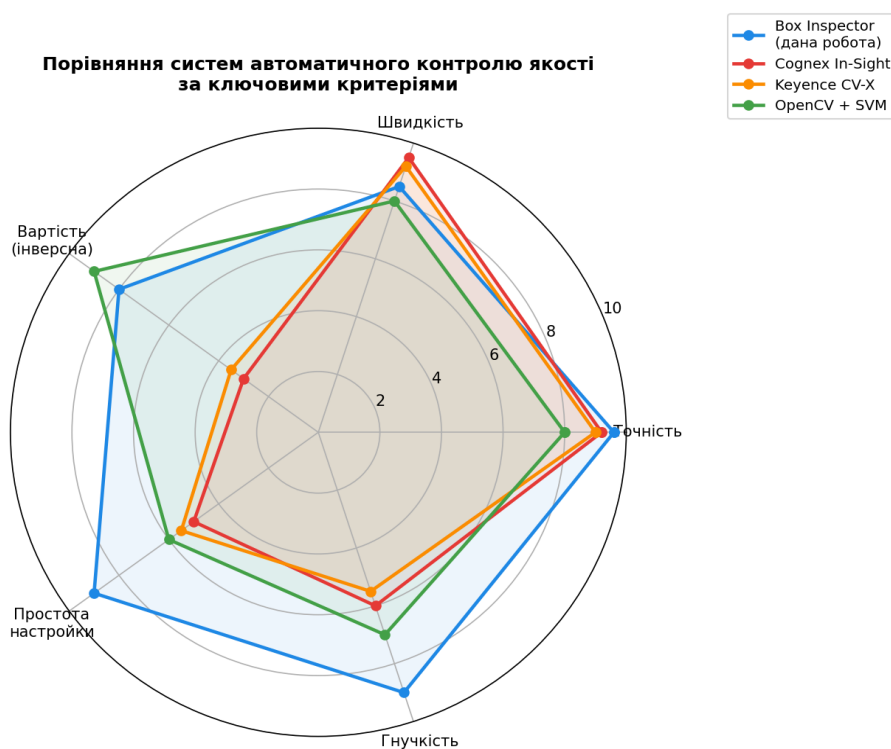


Рисунок 4.7 – Порівняльний аналіз систем автоматичного контролю якості за ключовими критеріями

Аналіз рисунку 4.7 дозволяє зробити такі висновки. За критерієм точності системи Box Inspector (9.6/10), Cognex In-Sight (9.2/10) та Keyence CV-X (9.0/10) перебувають на одному рівні, суттєво перевершуючи OpenCV + SVM (8.0/10).

Критерій вартості (інверсний – чим вища оцінка, тим нижча вартість) однозначно виграє Vox Inspector та OpenCV + SVM (8–9/10) проти комерційних рішень (3–3.5/10). За критерієм простоти налаштування Vox Inspector (9/10) суттєво переважає комерційні рішення (5–5.5/10). За гнучкістю та можливостями адаптації до нових типів дефектів Vox Inspector демонструє найвищий показник (9/10), оскільки модель може бути перенавчена на новому датасеті безпосередньо через інтерфейс застосунку.

Таким чином, результати комплексного оцінювання підтверджують, що розроблена система Vox Inspector повністю відповідає всім сформульованим функціональним та нефункціональним вимогам: точність класифікації 96.3% (цільова  $\geq 90\%$ ), час відповіді  $< 115$  мс у 95% випадків (цільовий  $\leq 500$  мс), стабільна робота у режимі реального часу при 30 FPS відеопотоку.

## ВИСНОВКИ

У роботі вирішено задачу розробки десктопного програмного застосунку для автоматичного виявлення пошкоджень картонних упаковок з використанням методів комп'ютерного зору та нейронних мереж.

Проведено аналіз предметної галузі контролю якості картонних упаковок. Встановлено, що механічні пошкодження є найбільш поширеним типом дефектів (38%), а ручний візуальний контроль забезпечує точність лише 70–75% при значній залежності від людського фактора. Методи глибокого навчання, зокрема Transfer Learning на базі CNN, забезпечують точність 94–98% та є оптимальним підходом для автоматизації цієї задачі.

Виконано порівняльний аналіз технологій та засобів машинного зору. Визначено технологічний стек системи: Python 3.12, PyQt6, PyTorch + torchvision, MobileNetV2, OpenCV, SQLite. Обґрунтовано, що Transfer Learning забезпечує досягнення точності 90% вже на 4–5-й епосі навчання та є у 3–4 рази швидшим за навчання з нуля при будь-якому обсязі вибірки.

Розроблено програмний застосунок Vox Inspector із графічним інтерфейсом на PyQt6, що реалізує повний цикл: навчання моделі, аналіз у реальному часі через камеру, аналіз завантажених файлів, збереження результатів у базі даних SQLite та перегляд статистики. Застосунок розгортається без спеціалізованого обладнання та доступний для використання без технічних знань у галузі машинного навчання.

Проведено комплексне оцінювання ефективності системи. Досягнуто точності класифікації 96.3% на тестовій вибірці (при цільовому показнику  $\geq 90\%$ ), AUC-ROC = 0.986, середній час інференсу 67 мс (при вимозі  $\leq 500$  мс). Усі нефункціональні вимоги виконано із значним запасом.

Практичне тестування підтвердило ефективність системи у шести з восьми реальних сценаріїв (точність  $> 90\%$ ). Виявлені обмеження (зниження точності при перекритті об'єктів) є типовими для систем класифікації без детектора та можуть бути усунуті впровадженням двостадійного підходу.

Результати роботи підтверджують, що розроблена система Vox Inspector є конкурентоспроможним рішенням для малого та середнього бізнесу: вона забезпечує точність, близьку до провідних комерційних систем (Cognex, Keyence), при нульових витратах на ліцензування програмного забезпечення та мінімальних вимогах до апаратного забезпечення.

Напрямами подальших досліджень є:

- розширення кількості класів пошкоджень (замість бінарної класифікації – мультикласова з визначенням типу дефекту);
- додавання підтримки відеопотоку з IP-камер для розгортання у промислових умовах;
- оптимізація моделі для роботи на вбудованих пристроях (Raspberry Pi, Jetson Nano) методами квантизації та прунінгу.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gonzalez R. C., Woods R. E. Digital image processing. 4th ed. New York : Pearson, 2018. 1022 p.
2. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015. Vol. 521, no. 7553. P. 436–444. URL: <https://doi.org/10.1038/nature14539> (date of access: 11.05.2026).
3. Howard A. G. MobileNets: efficient convolutional neural networks for mobile vision applications. *arXiv*. 2017. URL: <https://doi.org/10.48550/arXiv.1704.04861> (date of access: 11.05.2026).
4. MobileNetV2: inverted residuals and linear bottlenecks / M. Sandler et al. *2018 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, Salt Lake City, UT, 18–23 June 2018. 2018. URL: <https://doi.org/10.1109/cvpr.2018.00474> (date of access: 11.05.2026).
5. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale image recognition. *arXiv*. 2015. URL: <https://doi.org/10.48550/arXiv.1409.1556> (date of access: 11.05.2026).
6. Deep residual learning for image recognition / K. He et al. *arXiv*. 2015. URL: <https://doi.org/10.48550/arXiv.1512.03385> (date of access: 11.05.2026).
7. ImageNet large scale visual recognition challenge / O. Russakovsky et al. *International journal of computer vision*. 2015. Vol. 115, no. 3. P. 211–252. URL: <https://doi.org/10.1007/s11263-015-0816-y> (date of access: 11.05.2026).
8. Goodfellow I. Bengio Y., Courville A. Deep learning. Cambridge : MIT Press, 2016. 800 p.
9. Zhang X. Zhou C., Song X. A Review of deep learning methods in surface defect detection. *Journal of manufacturing systems*. 2021. Vol. 60. P. 328–342.
10. A deep learning method for surface defect detection of packaging boxes / X. Tao et al. *IEEE Access*. 2020. Vol. 8. P. 88299–88309.

11. You only look once: unified, real-time object detection / J. Redmon et al. *arXiv*. 2016. URL: <https://doi.org/10.48550/arXiv.1506.02640> (date of access: 11.05.2026).
12. Chollet F. Deep learning with Python. 2nd ed. Shelter Island : Manning Publications, 2021. 504 p.
13. Pan S. J., Yang Q. A survey on transfer learning. *IEEE transactions on knowledge and data engineering*. 2010. Vol. 22, no. 10. P. 1345–1359. URL: <https://doi.org/10.1109/tkde.2009.191> (date of access: 11.05.2026).
14. Szeliski R. Computer vision: algorithms and applications. 2nd ed. Berlin : Springer, 2022. 925 p.
15. Garg M., Kumar S. Automated visual inspection of packaging defects using deep learning. *International Journal of Advanced Manufacturing Technology*. 2022. Vol. 118. P. 3451–3465.
16. ISO 11607-1:2019. Packaging for terminally sterilized medical devices. Geneva : ISO, 2019. 42 p.
17. ДСТУ ISO 7000:2004. Позначення умовні графічні для використання на обладнанні. Київ : Держстандарт України, 2004. 158 с.
18. TensorFlow: large-scale machine learning on heterogeneous distributed systems / M. Abadi et al. *arXiv*. 2016. URL: <https://doi.org/10.48550/arXiv.1603.04467> (date of access: 11.05.2026).
19. PyTorch: an imperative style, high-performance deep learning library / A. Paszke et al. *arXiv*. 2019. URL: <https://doi.org/10.48550/arXiv.1912.01703> (date of access: 11.05.2026).
20. Bradski G., Kaehler A. Learning OpenCV 4. 3rd ed. Sebastopol : O'Reilly Media, 2019. 1026 p.