Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій

Бакалаврська кваліфікаційна робота на тему:
«АВТОМАТИЗОВАНА СИСТЕМА ВИЯВЛЕННЯ ДЕФЕКТІВ»

Виконав: студент IV курсу, групи
1АКІТ-21б
спеціальності 151 – Автоматизація та
комп'ютерно-інтегровані технології
(шифр і назва спеціальності)
Самолюк О. І.
(прізвище та ініціали)

Керівник: к.т.н., доцент кафедри АІІТ
Маслій Р. В.
(прізвище та ініціали)
«10» червня 2025 р.

Рецензент: к.т.н., доцент каф КСЗ
Грищук Т. В.

(прізвище та ініціали)
«13» червня 2025 р.

Допущено до захисту
завідувач кафедри АІІТ
«17» 06 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра автоматизації та інтелектуальних інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Галузь знань 15 – Автоматизація та приладобудування
Спеціальність 151 – Автоматизація та комп'ютерно-інтегровані технології
Освітньо-професійна програма Інтелектуальні комп'ютерні системи управління

ЗАТВЕРДЖУЮ
завідувач кафедри АІІТ
д.т.н., проф. Олег БІСІКАЛО
«17» 06 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Самолюку Олегу Ігоровичу
(прізвище, ім'я, по батькові)

1. Тема роботи «Автоматизована система виявлення дефектів»
керівник роботи Маслій Роман Васильович, к.т.н, доцент
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «20» березня 2025 року №97

2. Термін подання студентом роботи 10.06.2025 р.

3. Вихідні дані до роботи:

- мова програмування: Python 3.8 або вище.
- основні бібліотеки: PyTorch (версія 1.12), torchvision, NumPy, scikit-learn, tqdm.
- базова модель: попередньо навчена згорткова нейронна мережа WideResNet-50 (навчена на датасеті ImageNet).
- вхідні дані: зображення у форматі RGB із роздільною здатністю 224×224 пікселів після передобробки.
- датасет: MVTec AD.

4. Зміст текстової частини (перелік питань, які потрібно розробити) Огляд існуючих методів виявлення дефектів; аналіз сучасних автоматизованих систем виявлення дефектів; дослідження нейромережевих підходів виявлення дефектів; розробка програмного забезпечення та тестування системи виявлення дефектів

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень) Схема програми етапу тренування методу виявлення для

системи виявлення дефектів; Схема програми етапу застосування мет
виявлення для системи виявлення дефектів; Приклад виявлення дефе
для класу capsule датасету MVTecAD; Приклад виявлення дефектів
класу cable датасету MVTecAD.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата завдання видав	Підпис, дата завдання прийняв
1-4	Маслій Р. В., доц. каф. АІІТ	20.03.2025	

7. Дата видачі завдання 20.03.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Термін виконання		Примітки
		початок	закінчення	
1	Вибір, узгодження та затвердження теми БКР	03.10.2024	09.10.2024	
2	Аналіз предметної області та опрацювання літературних джерел.	12.03.2025	29.03.2025	
3	Постановка задач для вирішення в БКР	01.04.2025	26.04.2025	
4	Вирішення поставлених задач	29.04.2025	10.05.2025	
5	Підтвердження достовірності отриманих результатів	13.05.2025	24.05.2025	
7	Оформлення пояснювальної записки та графічної частини	27.05.2025	31.05.2025	
8	Перевірка ПЗ на відповідність вимогам	03.06.2025	07.06.2025	
9	Попередній захист, доопрацювання, та рецензування БКР	10.06.2025	15.06.2025	
10	Захист БКР	16.06.2025	23.06.2025	

Студент
(підпис) (ім'я та прізвище) Олег САМОЛЮК

Керівник роботи
(підпис) (ім'я та прізвище) Роман МАСЛІЙ

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 85 сторінок формату А4, на яких є 38 рисунків, список використаних джерел містить 23 найменувань.

Виконана робота присвячена розробці автоматизованої системи виявлення дефектів у зображеннях промислових об’єктів на основі методу PatchCore, реалізованої на мові програмування Python. Система призначена для ідентифікації та локалізації аномалій у зображеннях із датасету MVTec AD в задачах автоматизованого контролю якості.

У аналітичній частині роботи досліджено актуальність теми виявлення дефектів у промислових системах, проаналізовано сучасні технології комп’ютерного зору та машинного навчання, розглянуті готові продукти для виявлення дефектів, такі як, Cognex VisionPro, Siemens SIMATIC Vision та HALCON MVTec.

У теоретичній частині розглянуто основи згорткових нейронних мереж, а також розглянуто методи виявлення дефектів на основі автоенкодерів, варіаційних автоенкодерів, генеративних змагальних мереж (GAN) та розглянутий метод PatchCore.

У реалізаційній частині описано процес підготовки даних, адаптацію методу PatchCore, розробку модульної системи обробки зображень і експериментальне тестування. Розглянути основні бібліотеки для реалізації системи: PyTorch, torchvision, NumPy, scikit-learn і matplotlib. Система забезпечує класифікацію аномалій і їх локалізацію з використанням теплових карт, демонструючи високу продуктивність для текстурних і об’єктних дефектів.

Ключові слова: штучний інтелект, машинне навчання, комп’ютерний зір, виявлення дефектів, PatchCore, MVTec.

ABSTRACT

The bachelor qualification work consists of 85 pages of A4 format, on which there are 38 figures, the list of used sources contains 23 names.

The work is devoted to the development of an automated system for detecting defects in images of industrial objects based on the PatchCore method, implemented in the Python programming language. The system is designed to identify and localize anomalies in images from the MVTec AD dataset in automated quality control tasks.

The analytical part of the work investigates the relevance of the topic of detecting defects in industrial systems, analyzes modern computer vision and machine learning technologies, and considers ready-made products for detecting defects, such as Cognex VisionPro, Siemens SIMATIC Vision and HALCON MVTec.

The theoretical part considers the basics of convolutional neural networks, and also considers methods for detecting defects based on autoencoders, variational autoencoders, generative adversarial networks (GANs) and considers the PatchCore method.

The implementation part describes the process of data preparation, adaptation of the PatchCore method, development of a modular image processing system and experimental testing. Consider the main libraries for implementing the system: PyTorch, torchvision, NumPy, scikit-learn and matplotlib. The system provides anomaly classification and localization using heat maps, demonstrating high performance for texture and object defects.

Keywords: artificial intelligence, machine learning, computer vision, defect detection, PatchCore, MVTec.

ЗМІСТ

ВСТУП

Сучасний розвиток промислових, технологічних і виробничих процесів вимагає високої якості продукції та мінімізації дефектів на всіх етапах виробництва. Наявність дефектів може призводити до значних економічних втрат, зниження конкурентоспроможності підприємств, а в деяких випадках — до загрози безпеці. Традиційні методи контролю якості, такі як ручний огляд або вибірковий аналіз, є трудомісткими, суб'єктивними та не завжди ефективними в умовах великих обсягів виробництва.

Автоматизовані системи виявлення дефектів, що базуються на сучасних технологіях, таких як комп'ютерний зір, штучний інтелект і машинне навчання, дозволяють значно підвищити точність, швидкість і об'єктивність контролю якості. Впровадження таких систем сприяє оптимізації виробничих процесів, зниженню витрат і підвищенню надійності продукції. Актуальність теми зумовлена зростаючою потребою в автоматизації контролю якості в різних галузях, таких як машинобудування, електроніка, автомобілебудування, медицина тощо, а також швидким розвитком технологій обробки даних і аналізу зображень.

Дослідження в цій сфері є важливим для створення ефективних і доступних рішень, які можуть бути адаптовані до потреб українських підприємств, сприяючи їх інтеграції в глобальні ринки та підвищенню технологічного рівня промисловості. Виходячи з цього можна вважати тему **актуальною**.

Об'єктом дослідження є процеси автоматизованого виявлення дефектів у виробничих або технологічних системах.

Предметом дослідження є автоматизована система виявлення дефектів, що базується на методах комп'ютерного зору та машинного навчання

Метою дослідження є підвищення точності виявлення автоматизованої системи виявлення дефектів у зображеннях промислових об’єктів.

Для досягнення поставленої мети визначено наступні **завдання**:

1. Здійснити огляд предметної області, зокрема існуючих методів виявлення дефектів та сучасні автоматизовані системи виявлення дефектів.
2. Провести огляд базових понять згорткових нейронних мереж (CNN), оглянути їх структуру, принципи роботи та застосування в задачах обробки зображень.
3. Дослідити основні підходи до виявлення дефектів на основі методів машинного навчання, зокрема автоенкодерів, варіаційних автоенкодерів, генеративних змагальних мереж (GAN) і методу PatchCore, визначивши їх особливості, переваги та обмеження.
4. Реалізувати автоматизовану систему виявлення дефектів на основі методу машинного навчання, адаптувавши його до обробки зображень із датасету MVTec.
5. Провести експериментальне тестування розробленої системи на датасеті MVTec для оцінки її точності, чутливості та продуктивності.

Практична цінність роботи. Результати дослідження мають значну практичну цінність, оскільки розроблена автоматизована система виявлення дефектів на основі методу PatchCore може бути застосована для підвищення ефективності контролю якості в різних галузях промисловості, таких як машинобудування, електроніка, текстильна промисловість, автомобілебудування та інші. Використання методу PatchCore, який демонструє високу точність у виявленні аномалій на зображеннях, дозволяє автоматизувати процеси ідентифікації дефектів, зменшуючи залежність від ручного контролю, що є трудомістким і схильним до людських помилок.

Реалізована система, протестована на датасеті MVTec, забезпечує можливість швидкої та точної класифікації дефектів на промислових об’єктах, що сприяє зниженню рівня браку, оптимізації виробничих витрат і підвищенню конкурентоспроможності підприємств. Отримані в ході дослідження дані про продуктивність системи, при використанні метрики оцінки якості виявлення, а також порівняння з іншими методами (автоенкодерами, варіаційними автоенкодерами, GAN), надають цінну інформацію для вибору оптимальних технологій у конкретних виробничих умовах.

Адаптовані алгоритми та програмне забезпечення, створені в рамках дослідження, можуть бути інтегровані в існуючі автоматизовані виробничі лінії або використані як основа для подальших розробок у сфері комп’ютерного зору та машинного навчання.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Поняття та класифікація дефектів у виробничих процесах

Дефект у виробничих процесах визначається як відхилення характеристик продукту від установлених стандартів якості, що може впливати на його функціональність, надійність або естетичний вигляд. Згідно з міжнародним стандартом ISO 9000:2015, дефект – це невідповідність, яка призводить до порушення вимог до продукту або послуги. У контексті виробництва дефекти є критичними, оскільки вони можуть спричинити фінансові втрати, зниження конкурентоспроможності підприємства та навіть загрозу безпеці споживачів [1-3].

Дефекти виникають на різних етапах виробничого циклу: від закупівлі сировини до кінцевої обробки продукту. Основними причинами їх появи є невідповідність матеріалів, помилки обладнання, людський фактор або недоліки технологічного процесу. Для ефективного управління якістю необхідно не лише виявляти дефекти, але й систематизувати їх за певними ознаками, що дозволяє оптимізувати процеси контролю та профілактики.

Класифікація дефектів є важливим інструментом для аналізу та вдосконалення виробничих процесів. Існує кілька підходів до класифікації, які базуються на різних критеріях [1,2]:

1. За природою походження:

- матеріальні дефекти (пов’язані з якістю сировини, наприклад, тріщини в металі чи неоднорідність тканини).
- технологічні дефекти (виникають через порушення технологічного процесу, наприклад, неправильна термообробка).
- експлуатаційні дефекти (з’являються під час використання продукту, але спричинені виробничими недоліками).

2. За ступенем критичності:

- критичні дефекти (повністю унеможливлюють використання продукту, наприклад, розломи в несучих конструкціях).

- значні дефекти (знижують функціональність, але дозволяють часткове використання, наприклад, подряпини на поверхні).

- некритичні дефекти (впливають лише на естетичний вигляд, наприклад, незначні нерівності покриття).

3. За типом виявлення:

- явні дефекти (видимі неозброєним оком, наприклад, деформації поверхні).

- приховані дефекти (вимагають спеціальних методів діагностики, наприклад, внутрішні порожнини в металі).

4. За локалізацією:

- поверхневі дефекти (розташовані на зовнішній поверхні виробу).

- внутрішні дефекти (розташовані всередині матеріалу).

- комбіновані дефекти (поєднують зовнішні та внутрішні недоліки).

Приклад виробничої лінії у автоматизованій системі виявлення дефектів на виробництві зображений на рис 1. 1.

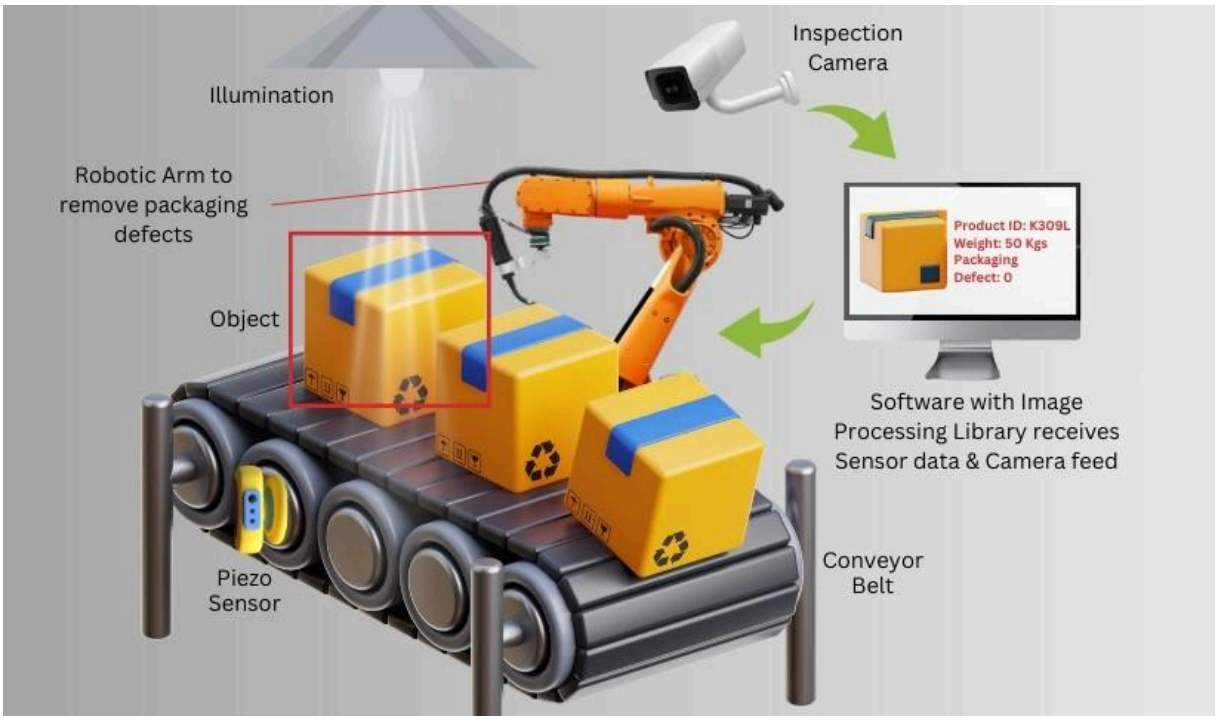

Рисунок 1.1 - Автоматизоване виявлення дефектів у виробництві

У сучасних виробничих системах класифікація дефектів часто залежить від специфіки галузі. Наприклад, у машинобудуванні особлива увага приділяється геометричним відхиленням, тоді як у текстильній промисловості – дефектам поверхні, таким як розриви чи плями. Для автоматизованих систем виявлення дефектів важливо враховувати ці класифікації, щоб адаптувати алгоритми обробки даних до конкретних типів недоліків [1-3].

Таким чином, розуміння природи та класифікації дефектів є основою для розробки ефективних методів їх виявлення. Це дозволяє не лише ідентифікувати проблемні ділянки у виробничому процесі, але й розробляти стратегії їх попередження, що важливо для підвищення якості продукції.

1.2 Огляд існуючих методів виявлення дефектів

Виявлення дефектів є важливим етапом контролю якості у виробничих процесах, спрямованим на забезпечення відповідності продукції встановленим стандартам. Існуючі методи виявлення дефектів можна поділити на дві основні категорії: традиційні та автоматизовані. Традиційні методи базуються на фізичних принципах та людському факторі, тоді як автоматизовані методи використовують сучасні технології [1], такі як комп’ютерний зір і штучний інтелект [2, 4-9]. У цьому підрозділі розглянуто обидва підходи, їх особливості, переваги та обмеження.

1.2.1 Традиційні методи

Традиційні методи виявлення дефектів застосовуються у промисловості протягом десятиліть і залишаються актуальними в багатьох

галузях завдяки своїй простоті та доступності. Основними серед них є візуальний контроль і неруйнівний контроль.

Візуальний контроль передбачає огляд виробів оператором без використання спеціального обладнання. Цей метод є найпростішим і найдешевшим, оскільки не вимагає складної апаратної бази. Оператор оцінює зовнішній вигляд виробу, виявляючи явні дефекти, такі як подряпини, тріщини чи деформації. Проте візуальний контроль має суттєві недоліки: він залежить від кваліфікації та уважності оператора, схильний до суб'єктивізму та не дозволяє виявляти приховані дефекти. Крім того, при великих обсягах виробництва цей метод стає неефективним через низьку швидкість і втому працівників [1].

- Неруйнівний контроль (NDT) охоплює набір методів, які дозволяють оцінити стан виробу без порушення його цілісності. До основних технік належать:

- Ультразвуковий контроль: використовує високочастотні звукові хвилі для виявлення внутрішніх дефектів, таких як порожнини чи тріщини. Цей метод ефективний для металевих і композитних матеріалів, але вимагає дорогого обладнання та кваліфікованого персоналу.

- Рентгенівський контроль: базується на проникненні рентгенівських променів через матеріал для виявлення внутрішніх дефектів. Застосовується в аерокосмічній та автомобільній промисловості, але є дорогим і потребує захисту від випромінювання.

- Магнітно-порошковий контроль: використовується для виявлення поверхневих і підповерхневих дефектів у феромагнітних матеріалах. Метод ефективний, але обмежений типом матеріалів.

- Контроль проникними рідинами: передбачає нанесення спеціальних рідин для виявлення поверхневих тріщин. Цей метод простий, але не підходить для пористих матеріалів.

Перевагами неруйнівного контролю є висока точність і можливість виявлення прихованих дефектів. Однак ці методи часто є трудомісткими,

потребують спеціалізованого обладнання та висококваліфікованих фахівців, що підвищує витрати. Крім того, більшість технік НК не дозволяють автоматизувати процес у реальному часі, що обмежує їх застосування у швидкісних виробничих лініях.

1.2.2 Автоматизовані методи

З розвитком інформаційних технологій автоматизовані методи виявлення дефектів набули широкого поширення, особливо в галузях із високими вимогами до швидкості та точності. Основними технологіями є машинне бачення та методи, засновані на нейронних мережах [2-3].

Машинне бачення (Computer Vision) використовує камери та алгоритми обробки зображень для автоматичного аналізу поверхні виробів. Система складається з апаратної частини (камери, джерела освітлення) та програмного забезпечення для обробки даних (наприклад, OpenCV). Процес складається з етапів попередньої обробки зображень (фільтрація, сегментація), виділення ознак і класифікацію дефектів. Машинне бачення ефективне для виявлення явних дефектів, таких як подряпини, плями чи геометричні відхилення, і широко застосовується в електроніці, текстильній та автомобільній промисловості.

Переваги машинного бачення полягають у високій швидкості обробки, можливості інтеграції в автоматизовані виробничі лінії та зниження впливу людського фактора. Однак цей метод має обмеження: він залежить від якості освітлення, роздільної здатності камер і складності алгоритмів. Крім того, традиційні алгоритми машинного бачення можуть бути менш ефективними для виявлення складних або нестандартних дефектів, що потребують адаптивних підходів.

Нейронні мережі, зокрема згорткові нейронні мережі, є передовим інструментом для виявлення дефектів. Ці моделі здатні навчатися на великих наборах даних, автоматично виділяючи ознаки дефектів без необхідності ручного налаштування. Наприклад, такі архітектури, як

ResNet, YOLO чи U-Net, використовуються для класифікації, сегментації та локалізації дефектів на зображеннях. Нейронні мережі особливо ефективні для виявлення складних і прихованих дефектів, а також у випадках, коли дефекти мають варіативний вигляд.

Основними перевагами нейронних мереж є їх висока точність, адаптивність і здатність обробляти великі обсяги даних у реальному часі. Наприклад, у машинобудуванні CNN можуть виявляти мікротріщини з точністю понад 95%. Проте впровадження нейронних мереж потребує значних обчислювальних ресурсів, великих анотованих наборів даних для навчання та кваліфікованих фахівців для налаштування моделей. Крім того, перенавчання або недостатня узагальненість моделей можуть знижувати їх ефективність на нових даних.

Порівняно з традиційними методами, автоматизовані підходи забезпечують вищу швидкість, масштабованість і точність, але їх впровадження пов’язане з високими початковими витратами та складністю налаштування. Вибір між традиційними та автоматизованими методами залежить від специфіки виробництва, типу дефектів і доступних ресурсів.

Таким чином, традиційні методи залишаються актуальними для малих виробництв або специфічних завдань, тоді як автоматизовані методи, зокрема машинне бачення та нейронні мережі, є перспективними для сучасних високотехнологічних галузей, де потрібна висока продуктивність і точність.

1.3 Технології машинного навчання та комп’ютерного зору в автоматизованих системах

Технології машинного навчання (ML) та комп’ютерного зору (CV) стали основними інструментами в автоматизованих системах виявлення дефектів, забезпечуючи високу точність, швидкість і масштабованість. Ці

технології дозволяють обробляти великі обсяги даних, автоматично виявляти дефекти різної природи та адаптуватися до змін у виробничих процесах. Їх інтеграція в промислові системи контролю якості відкриває нові можливості для підвищення ефективності та зниження витрат. У цьому підрозділі розглянуто основні принципи, методи та інструменти ML і CV, а також їх застосування в автоматизованих системах [4-6].

1.3.1 Принципи комп’ютерного зору в автоматизованих системах

Комп’ютерний зір – це галузь інформатики, що займається автоматичною обробкою та аналізом візуальних даних, таких як зображення чи відеопотоки. У контексті виявлення дефектів CV використовується для аналізу поверхні виробів, виявлення аномалій і класифікації дефектів. Основний процес містить такі етапи:

1. Збір даних: використання камер високої роздільної здатності, тепловізорів або 3D-сканерів для отримання зображень виробів. Якість даних залежить від роздільної здатності, освітлення та калібрування обладнання.

2. Попередня обробка: застосування фільтрів (наприклад, гаусівське розмиття), нормалізації або сегментації для покращення якості зображень і виділення регіонів інтересу.

3. Виділення ознак: ідентифікація важливих характеристик, таких як форма, текстура чи колір, що можуть вказувати на дефекти.

4. Класифікація або локалізація: визначення наявності дефектів і їх типу (наприклад, тріщина, подряпина) або вказівка їх точного розташування на зображенні.

Для реалізації цих етапів використовуються програмні бібліотеки, такі як OpenCV, яка підтримує обробку зображень у реальному часі, або MATLAB для складніших обчислень. Комп’ютерний зір ефективний для виявлення явних дефектів, таких як геометричні відхилення чи поверхневі

пошкодження, але його можливості значно розширюються при поєднанні з методами машинного навчання.

1.3.2 Роль машинного навчання у виявленні дефектів

Машинне навчання, зокрема підходи з учителем, без учителя та з підкріпленням, дозволяє створювати системи, здатні навчатися на даних і адаптуватися до нових умов. У виявленні дефектів найпоширенішими є методи з учителем, де моделі тренуються на анотованих наборах даних, що містять приклади дефектних і бездефектних виробів. Основні алгоритми ML, які застосовуються в автоматизованих системах, є наступні [2, 4-6]:

- класичні алгоритми: методи опорних векторів (SVM) і дерева рішень використовуються для класифікації дефектів на основі заздалегідь визначених ознак. Вони ефективні для простих завдань, але менш гнучкі для складних сценаріїв.

- нейронні мережі: згорткові нейронні мережі є стандартом для обробки зображень. Такі архітектури, як ResNet, VGG або EfficientNet, автоматично виділяють ознаки з зображень, що дозволяє виявляти складні дефекти з високою точністю. Наприклад, CNN можуть ідентифікувати мікротріщини в металевих деталях з точністю понад 95%.

- методи сегментації: моделі, такі як U-Net або Mask R-CNN, використовуються для піксельної сегментації, що дозволяє точно локалізувати дефекти на зображенні.

- виявлення аномалій: методи без учителя, такі як автоенкодери або кластеризація, застосовуються для ідентифікації нестандартних дефектів, для яких немає анотованих даних.

Для тренування моделей використовуються фреймворки машинного навчання, такі як TensorFlow, PyTorch або Keras, які забезпечують гнучкість і підтримують апаратне прискорення (GPU/TPU). Однак успішне застосування ML вимагає великих наборів даних, які містять різноманітні приклади дефектів, а також ретельного налаштування гіперпараметрів.

Інтеграція ML і CV в автоматизованих системах

Поєднання комп'ютерного зору та машинного навчання дозволяє створювати комплексні системи, які охоплюють усі етапи виявлення дефектів – від збору даних до прийняття рішень. Типова архітектура такої системи містить:

- апаратну частину: камери, датчики та обчислювальні модулі для збору та обробки даних у реальному часі.

- програмну частину: алгоритми CV для попередньої обробки зображень і моделі ML для аналізу та класифікації.

- інтерфейс: системи візуалізації результатів, які дозволяють операторам переглядати виявлені дефекти та приймати рішення.

Приклади застосування таких систем:

- машинобудування: виявлення тріщин і корозії на поверхні металевих деталей за допомогою CNN і камер високої роздільної здатності.

- електроніка: перевірка якості друкованих плат, де моделі YOLO локалізують дефекти пайки.

- текстильна промисловість: автоматичне виявлення розривів або плям на тканині за допомогою комбінації OpenCV і нейронних мереж.

Технології ML і CV мають низку переваг:

- висока точність і швидкість обробки, що дозволяє інтегрувати системи в автоматизовані виробничі лінії.

- здатність адаптуватися до нових типів дефектів шляхом донавчання моделей.

- зниження залежності від людського фактора, що зменшує суб'єктивізм і помилки.

Проте їх впровадження пов'язане з викликами:

- необхідність великих анотованих наборів даних, створення яких є трудомістким і дорогим.

- високі вимоги до обчислювальних ресурсів, особливо для глибоких нейронних мереж.

- складність узагальнення моделей для роботи з новими типами матеріалів або умов освітлення.

Сучасні тенденції в розвитку ML і CV використовують генеративні моделі (наприклад, GAN) для створення синтетичних даних, що компенсує брак реальних зображень, і впровадження моделей з малою кількістю даних (few-shot learning). Крім того, інтеграція з хмарними платформами та технологіями IoT дозволяє створювати розподілені системи контролю якості, які обробляють дані в реальному часі.

Таким чином, технології машинного навчання та комп'ютерного зору є основою сучасних автоматизованих систем виявлення дефектів. Їх здатність до обробки складних даних і адаптації до різних умов робить їх незамінними для високотехнологічних галузей, а подальший розвиток цих технологій обіцяє ще більшу ефективність і доступність.

1.4 Аналіз сучасних автоматизованих систем виявлення дефектів

Сучасні автоматизовані системи виявлення дефектів є інтегральною частиною промислових процесів, спрямованих на забезпечення високої якості продукції. Ці системи поєднують апаратні засоби (камери, датчики, сканери) та програмне забезпечення, засноване на технологіях комп'ютерного зору (CV) і машинного навчання (ML). Вони застосовуються в різних галузях, таких як машинобудування, електроніка, текстильна промисловість і фармацевтика, для виявлення поверхневих, внутрішніх і комбінованих дефектів. У цьому підрозділі проведено аналіз сучасних систем, їх функціональних можливостей, прикладів реалізації, а також оцінено їх переваги та обмеження.

1. Cognex VisionPro — це провідне програмне забезпечення для машинного бачення, розроблене для автоматизації промислового контролю якості, зокрема виявлення дефектів, перевірки складання, вимірювання та

ідентифікації об’єктів у таких галузях, як електроніка, автомобілебудування і текстильна промисловість. Воно поєднує традиційні інструменти обробки зображень (pattern matching, blob analysis, OCR) та модуль VisionPro Deep Learning для вирішення складних завдань, таких як виявлення нестандартних дефектів у шумних умовах [10]. Програма підтримує широкий спектр камер, працює на Windows, інтегрується з IoT і Industry 4.0, забезпечуючи високу швидкість обробки (до 1000 зображень за секунду) і гнучкість завдяки інтерфейсам QuickBuild та .NET-програмуванню. VisionPro ефективно застосовується для інспекції друкованих плат, поверхонь деталей і текстилю. Графічний інтерфейс VisionPro наведений на рис 1.2.

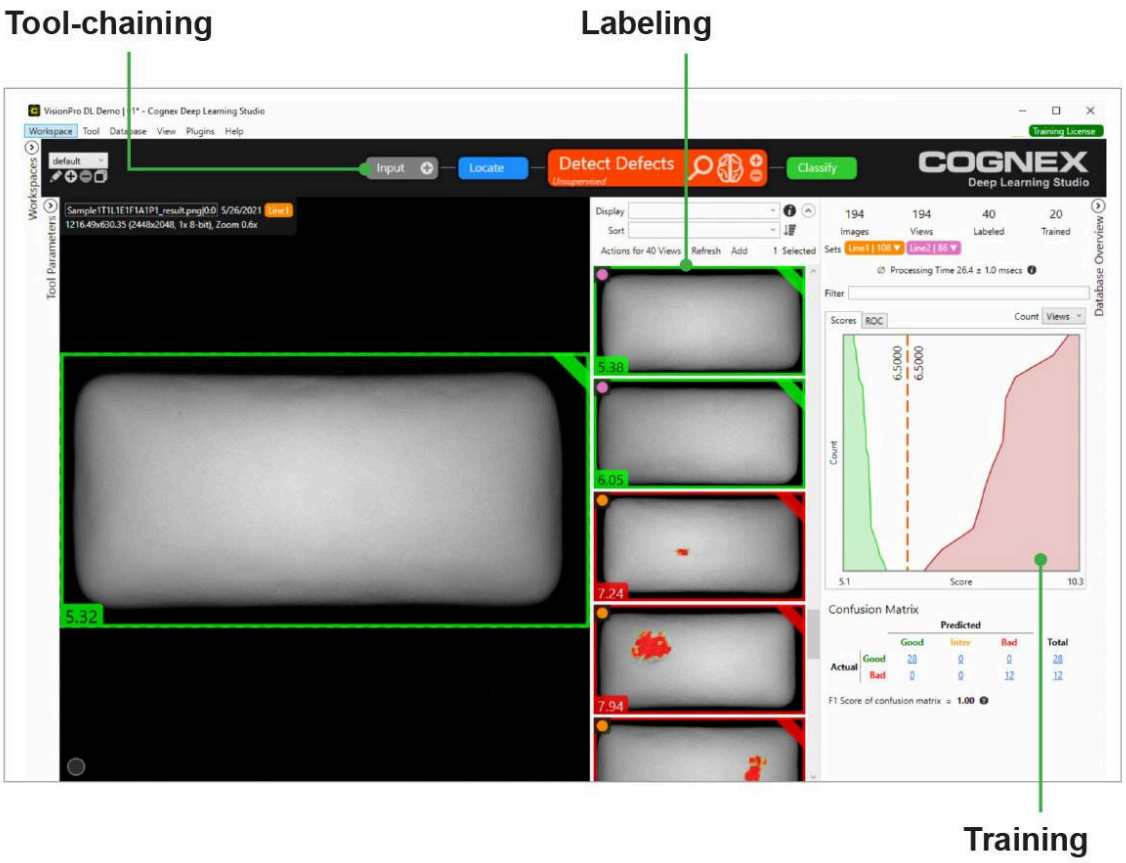

Рисунок 1.2 - Графічний інтерфейс VisionPro при роботі з пошуком дефектів у зображеннях

Основними перевагами VisionPro є висока точність, адаптивність до змін умов і легка інтеграція в автоматизовані лінії, що скорочує залежність від людського фактора. Проте система потребує значних початкових витрат на апаратне забезпечення, ліцензії та навчання персоналу, а також великих анотованих наборів даних для тренування моделей глибокого навчання. Обмеження мають залежність від якості освітлення та складність налаштування для нестандартних завдань [10]. Перспективи розвитку охоплюють впровадження edge computing, few-shot learning та генеративних моделей, що підвищує масштабованість і доступність.

2. Siemens SIMATIC Vision — це лінійка систем машинного бачення, призначених для автоматизованого контролю якості, зокрема, виявлення дефектів, зчитування кодів, розпізнавання об’єктів і вимірювання в таких галузях, як фармацевтика, пакування та автомобілебудування. До складу входять компактні сенсори (наприклад, SIMATIC MV220 для перевірки кольорових об’єктів), інтелектуальні камери (VS120, VS130-2) і програмне забезпечення, інтегроване з контролерами SIMATIC S7-1500 через TIA Portal. Системи підтримують обробку 1D/2D-кодів, OCR і AI-додатки завдяки інтеграції з Basler Vision Connector і MVTec Anomaly Detection через Siemens Industrial Edge, що забезпечує високу точність і швидкість (до 100 зображень за секунду для MV220). Вони ефективні для інспекції упаковки, перевірки якості пластикових виробів і текстилю [11].

Основні переваги SIMATIC Vision полягають у безшовній інтеграції з автоматизованими системами Siemens, високій надійності завдяки масштабованості S7-1500 і підтримку хмарних технологій для обробки даних у реальному часі. Однак системи мають високу вартість апаратного забезпечення та потребують кваліфікованого персоналу для налаштування, особливо для AI-додатків, а також залежать від стабільних умов освітлення. Перспективи розвитку охоплюють розширення AI-функціоналу, впровадження edge computing і стандартизованих інтерфейсів для спрощення інтеграції камер, що відповідає тенденціям

Industry 4.0, і робить SIMATIC Vision перспективним рішенням для адаптивного виробництва [11].

Схематичне зображення виробничої лінії для пошуку дефектів при використанні SIMATIC Vision наведено на рис. 1.3

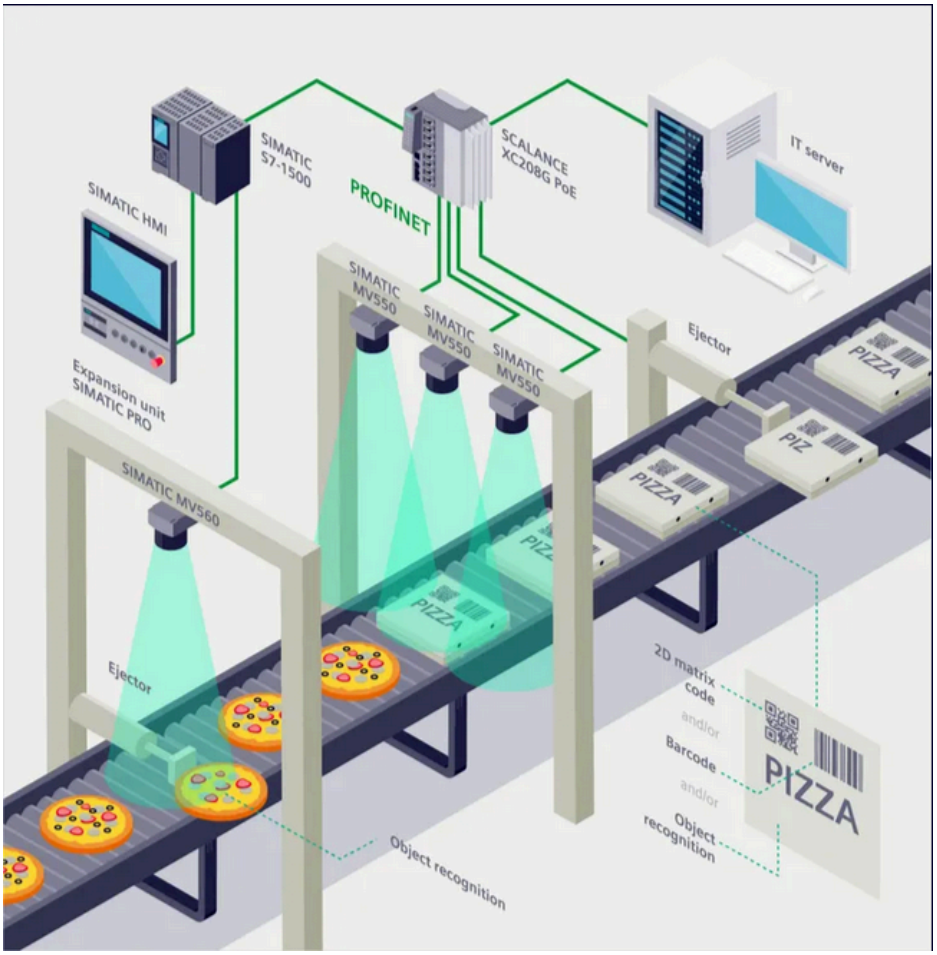

Рисунок 1.3 - Схематичне зображення виробничої лінії для пошуку дефектів при використанні SIMATIC Vision

3. MVTec Software GmbH — провідний міжнародний виробник програмного забезпечення для машинного бачення, чиї продукти HALCON (рис. 1.4) і MERLIC (рис. 1.5) застосовуються для автоматизованого виявлення дефектів, контролю якості та робототехнічного керування в таких галузях, як напівпровідникова, автомобільна, текстильна та харчова промисловість. HALCON — це комплексна стандартна бібліотека з інтегрованим середовищем розробки (HDevelop), що підтримує

2D/3D-обробку, глибоке навчання та технології, як-от 3D Gripping Point Detection, для складних завдань, таких як розпізнавання об’єктів і поверхневий контроль. MERLIC, своєю чергою, пропонує інтуїтивний графічний інтерфейс для швидкого створення застосунків без програмування, що охоплюють класифікацію, вимірювання та зчитування кодів. Обидва продукти сумісні з широким спектром апаратного забезпечення (GigE Vision, USB3 Vision) і оптимізовані для вбудованих систем, що робить їх ефективними для інспекції текстилю, електронних компонентів і упаковки [12].

Рисунок 1.4 - Приклад виявлення дефектів ємностей з вакцинами продуктом MVTec HALCON

Переваги MVTec полягають у високій продуктивності, гнучкості завдяки підтримці різних операційних систем і AI-прискорювачів, а також швидкий вихід на ринок завдяки коротким циклам оновлення (наприклад, HALCON 24.11 із Out of Distribution Detection). Однак висока складність налаштування HALCON для новачків і потреба в анотованих даних для глибокого навчання є обмеженнями, особливо для малих підприємств. Перспективи розвитку охоплюють розширення AI-функціоналу, як-от

Global Context Anomaly Detection, і поглиблення інтеграції з Industry 4.0 через вбудовані рішення та хмарні платформи і позиціонує MVTec як провідного гравця в автоматизації виробництва [12].

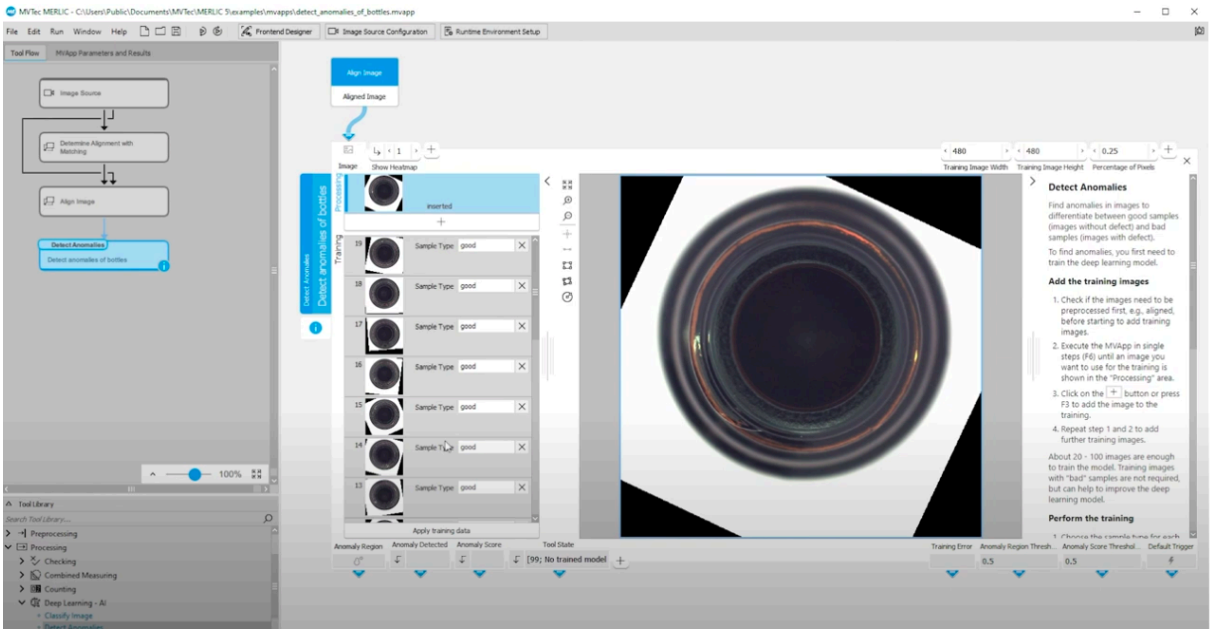

Рисунок 1.5 - Приклад інтерфейсу продукту MVTec MERLIC під час виявлення дефектів керамічної продукції

1.5 Висновки до розділу

У першому розділі розглянуто теоретичні основи автоматизованого виявлення дефектів, що є важливим етапом для розробки ефективних систем контролю якості. Аналіз понять і класифікації дефектів (підрозділ 1.1) показав, що дефекти є критичними для якості продукції та можуть бути систематизовані за походженням, критичністю, типом виявлення та локалізацією. Це дозволяє адаптувати методи контролю до специфіки виробничих процесів.

Огляд існуючих методів виявлення дефектів (підрозділ 1.2) виявив переваги та обмеження традиційних підходів, таких як візуальний і неруйнівний контроль, порівняно з автоматизованими методами, які

базуються на машинному баченні та нейронних мережах. Автоматизовані методи демонструють вищу швидкість і точність, що робить їх перспективними для сучасного виробництва.

Дослідження технологій машинного навчання та комп’ютерного зору (підрозділ 1.3) підкреслило їх центральну роль у створенні адаптивних і високоефективних систем. Згорткові нейронні мережі, алгоритми сегментації та бібліотеки, такі як OpenCV і TensorFlow, забезпечують обробку складних даних і виявлення дефектів у реальному часі, хоча потребують значних ресурсів і даних для навчання.

Аналіз сучасних автоматизованих систем (підрозділ 1.4) показав різноманітність рішень, від комерційних систем (Cognex, Siemens) до open-source розробок. Їх ефективність залежить від точності, швидкості, вартості та складності впровадження, а перспективні напрями розвитку - це edge computing, генеративні моделі та інтеграцію з IoT.

Таким чином, теоретичний аналіз підтверджує актуальність і перспективність автоматизованих систем виявлення дефектів. Отримані знання створюють основу для розробки власної системи, яка враховуватиме сучасні технології та специфіку обраної галузі, що буде розглянуто в наступних розділах.

2 ДОСЛІДЖЕННЯ НЕЙРОМЕРЕЖЕВИХ ПІДХОДІВ ВИЯВЛЕННЯ ДЕФЕКТІВ

2.1 Основні поняття нейронних мереж у виявленні дефектів

Згорткові нейронні мережі є основним інструментом для автоматизованого виявлення дефектів завдяки їх здатності ефективно обробляти зображення, як зазначено в розділі 1.3. CNN складаються з кількох типів шарів, кожен із яких виконує специфічну функцію (рис. 2.1) [13-15]. Згорткові шари (convolutional layers) застосовують фільтри для виділення ознак, таких як краї, текстури чи форми, що є вважливими для виявлення поверхневих дефектів, як подряпини чи плями. Пулінгові шари (pooling layers), зокрема максимальний пулінг, зменшують просторові розміри, зберігаючи важливі ознаки, що підвищує обчислювальну ефективність і стійкість до незначних деформацій. Повнозв'язні шари (fully connected layers) на завершальному етапі інтегрують ознаки для класифікації (наприклад, "дефект" чи "без дефекту") (рис. 2.2). Функції активації, такі як ReLU (Rectified Linear Unit), додають нелінійність, дозволяючи моделі обробляти складні візуальні патерни, тоді як нормалізаційні шари, як Batch Normalization, стабілізують тренування, зменшуючи залежність від змін освітлення чи контрасту [13-15].

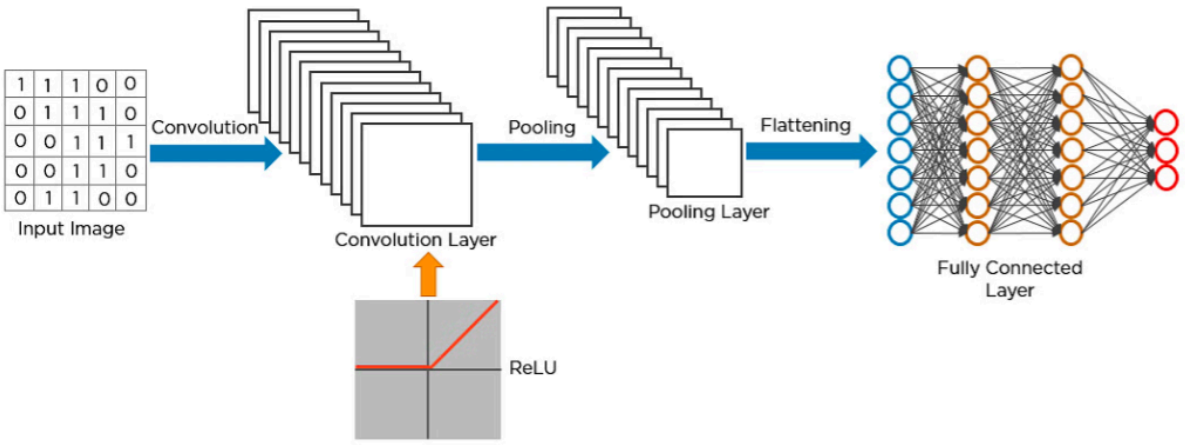

Рисунок 2.1 - Загальна структура згорткової нейронної мережі

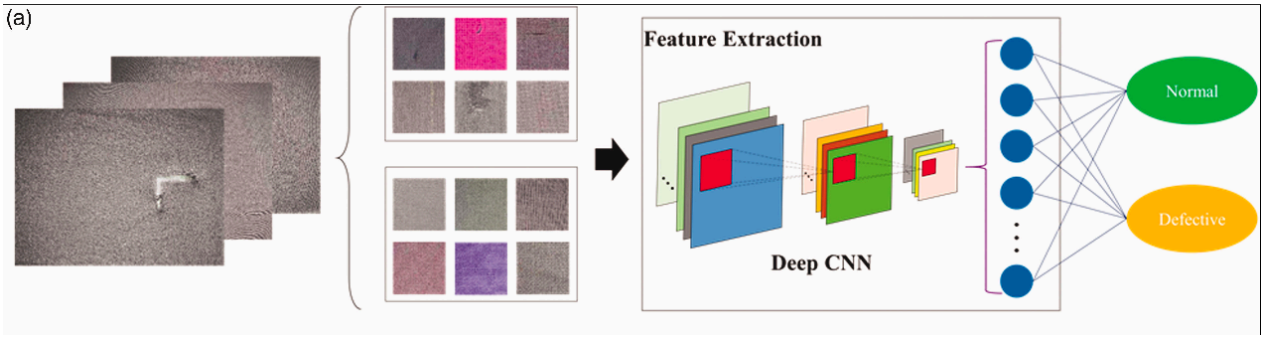

Рисунок 2.2 - Приклад виявлення дефекту згортковою нейронною мережею

Зготкові шари базуються на понятті згортки, яка застосовує фільтри (невеликі матриці, наприклад, 3×3) до вхідного зображення (рис. 2.3). Згорткові фільтри застосовуються до вхідного зображення шляхом ковзання (convolution operation) по його пікселях, обчислюючи скалярний добуток між ядром фільтра та відповідною областю зображення. Це дозволяє створювати карти ознак (feature maps), які відображають наявність певних візуальних характеристик у різних частинах зображення [13-16].

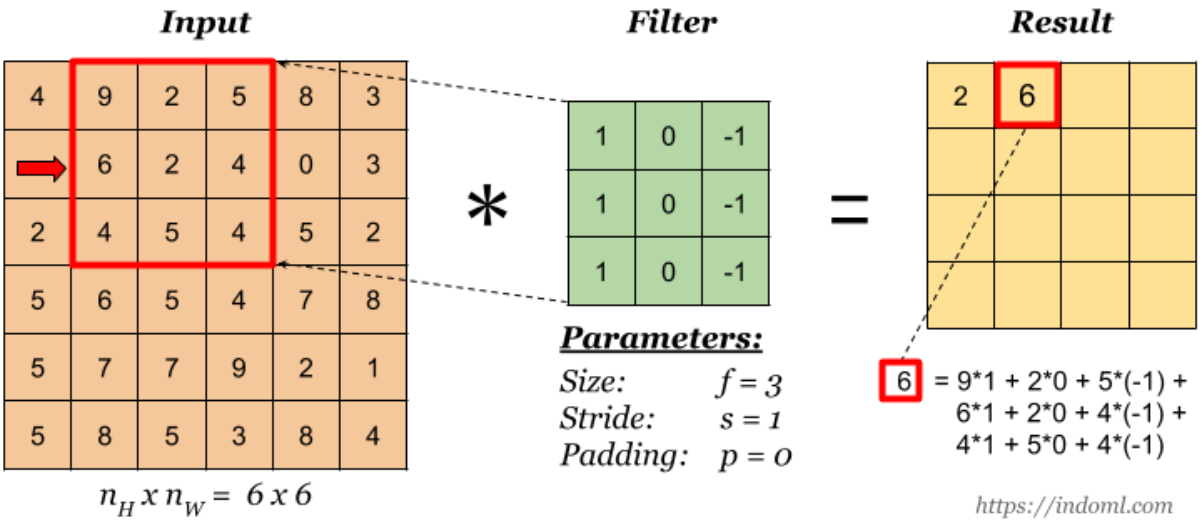

Рисунок 2.3 - Приклад згортки області зображення з фільтром 3*3

Фільтри зсуваються по зображенню з кроком, визначеним параметром кроку (stride), що контролює щільність обробки: більший крок зменшує розмір вихідної карти ознак, оптимізуючи обчислення (рис. 2.4).

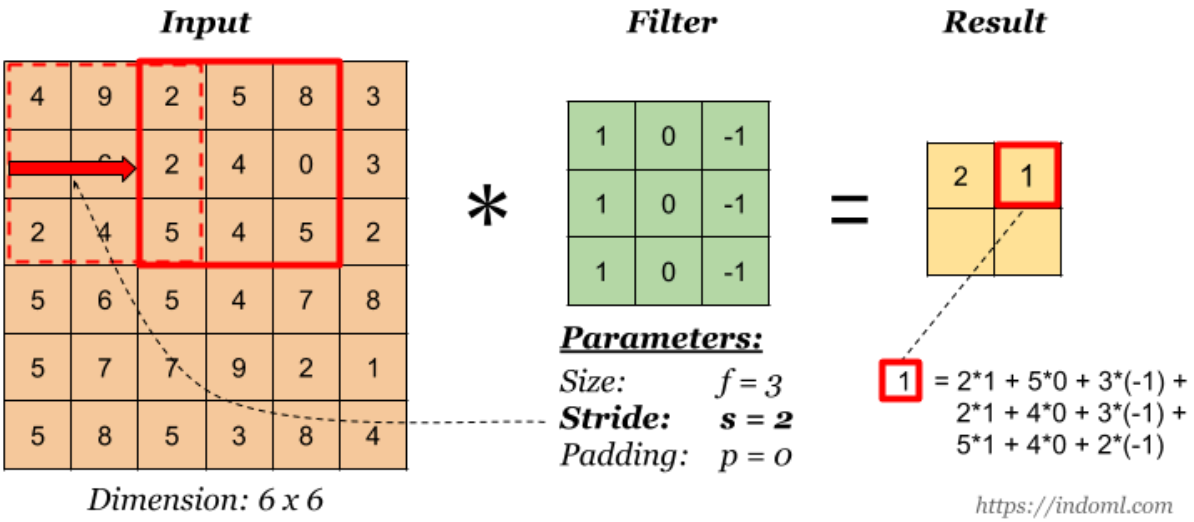

Рисунок 2.4 - Приклад згортки області зображення з фільтром 3*3 з кроком 2

Падінг (padding) додає пікселі (наприклад, нулі) по краях зображення, щоб зберегти його розмір після згортки або забезпечити аналіз крайових дефектів (рис. 2.5).

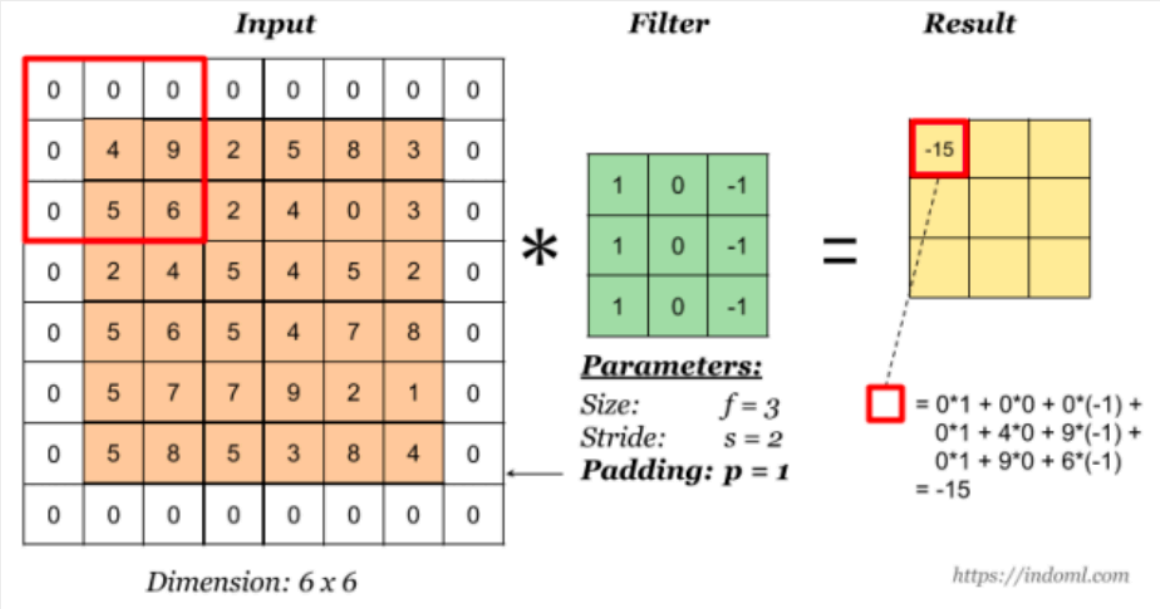

Рисунок 2.5 - Приклад падінгу

Пулінгові шари, такі як максимальний пулінг, зменшують просторові розміри карти ознак, обираючи максимальне значення в заданому вікні (наприклад, 2×2), що підвищує стійкість до шумів і деформацій, критичних для промислових зображень, як у датасеті MVTec AD. Приклад падінгу зображений на рисунку 2.6.

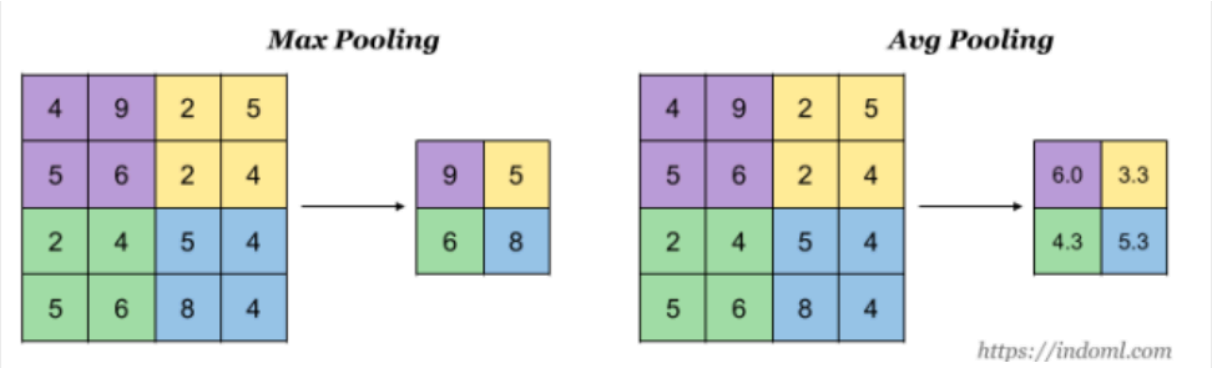

Рисунок 2.6 - Приклад середнього та максимального пулінгу області зображення

Для виявлення дефектів у задачах комп’ютерного зору колірна інформація відіграє важливу роль, оскільки багато аномалій, таких як плями, зміна відтінків або корозія, проявляються через відмінності в кольорі. Саме тому обробка RGB-зображень за допомогою згорткових нейронних мереж є поширеним підходом у задачах промислового контролю якості, зокрема в наборах даних, таких як MVTec AD. RGB-зображення, які складаються з трьох каналів (червоний, зелений, синій), дозволяють CNN ефективно аналізувати колірні характеристики об’єктів, що значно підвищує точність виявлення дефектів порівняно з обробкою зображень у відтінках сірого [16]. На рисунку 2.7 наведено приклад обробки RGB-зображення за допомогою CNN, де показано, як згорткові шари витягують колірні та текстурні ознаки для подальшого аналізу.

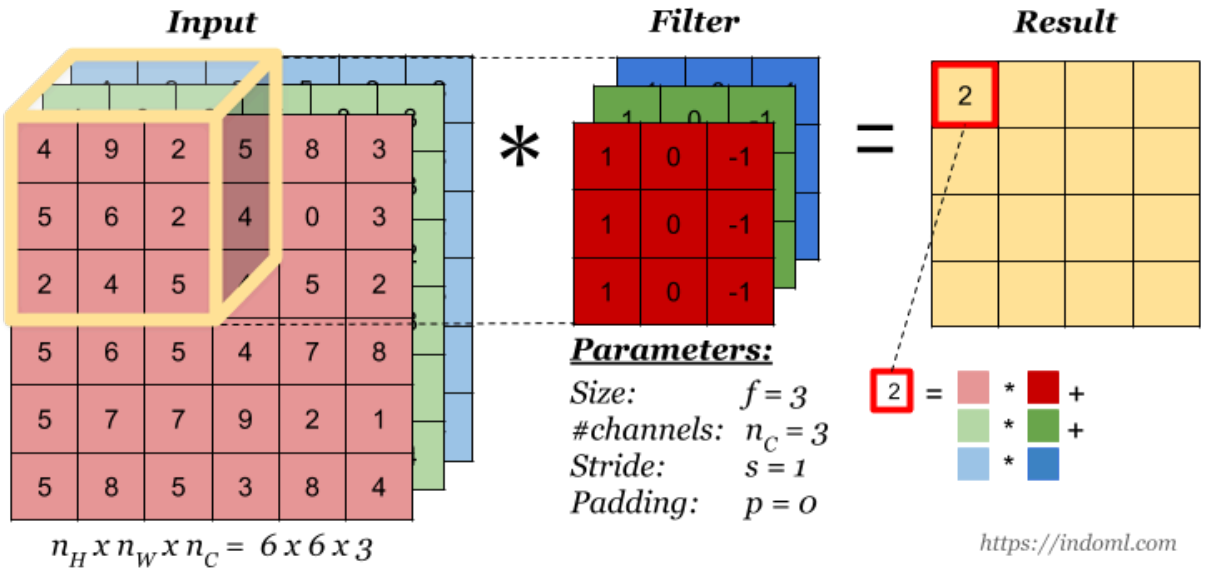

Рисунок 2.7 - Приклад згортки області RGB зображення з фільтром 3*3

Як правило, для виявлення різноманітних ознак у зображеннях, зокрема тих, що характерні для дефектів у задачах комп'ютерного зору, у шарі згортки (convolutional layer) нейронних мереж використовується велика кількість фільтрів. Кожен фільтр відповідає за виділення певних особливостей зображення, таких як краї, текстури, кути чи інші патерни, які можуть вказувати на дефекти, наприклад, подряпини, тріщини або плями.

Для застосування двох фільтрів до області RGB-зображення кожен фільтр має три виміри (ширина, висота та глибина, що відповідає трьом каналам), і його параметри налаштовуються під час тренування моделі для виділення специфічних ознак. Приклад такого процесу згортки з використанням двох фільтрів наведено на рисунку 2.8, де показано, як фільтри обробляють область зображення, створюючи дві окремі карти ознак. Кожна карта відображає реакцію фільтра на певні візуальні патерни, що допомагає виявляти дефекти чи аномалії в зображенні.

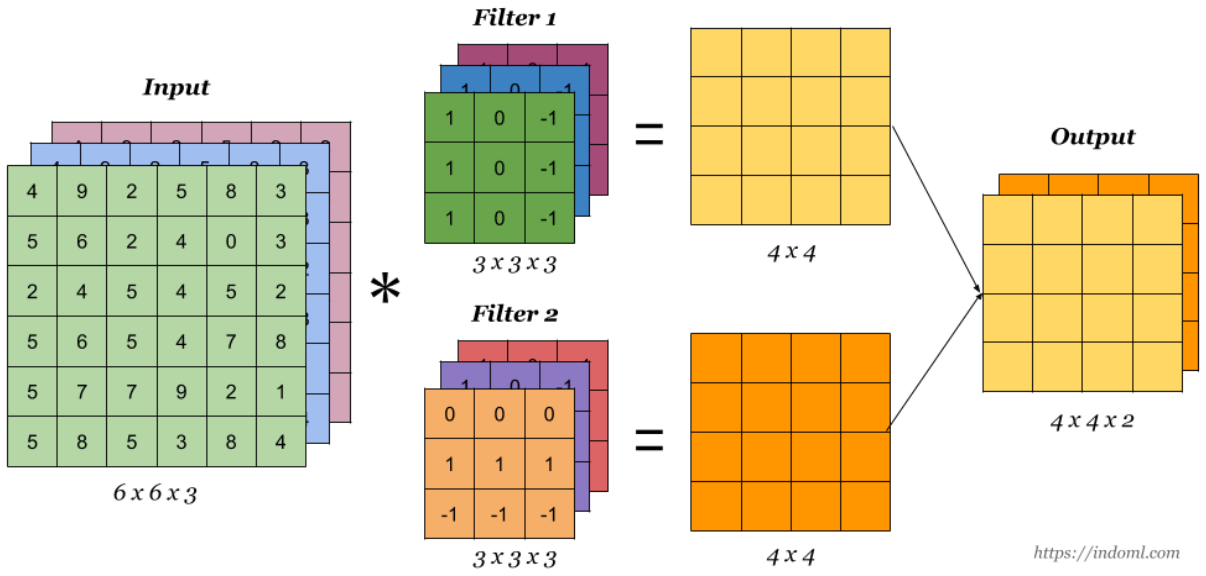

Рисунок 2.8 - Приклад згортки області RGB зображення з двома фільтрами розміром 3*3

Кількість фільтрів у згортковому шарі може варіюватися залежно від складності задачі. У задачах із набору даних MVTec AD, де потрібно виявляти тонкі дефекти на текстурах або поверхнях, зазвичай використовується десятки або сотні фільтрів, щоб охопити широкий спектр можливих ознак. Це дозволяє моделі ефективно розпізнавати як грубі, так і ледь помітні аномалії. Після створення карт ознак вони часто передаються до наступних шарів мережі, таких як шари пулінгу (pooling layers) для зменшення розмірності або повнозв'язні шари для класифікації чи оцінки аномалій.

Для створення шару згортки додається зміщення (bias) та застосовується функція активації, така як ReLU або tanh.

ReLU (Rectified Linear Unit) — одна з найпопулярніших функцій активації в сучасних нейромережах, особливо якщо йдеться про глибокі нейронні мережі (DNNs), які мають велику кількість шарів між входом та виходом. Ця функція замінює всі від'ємні вхідні значення на 0 та не змінює позитивні значення (рис. 2.9).

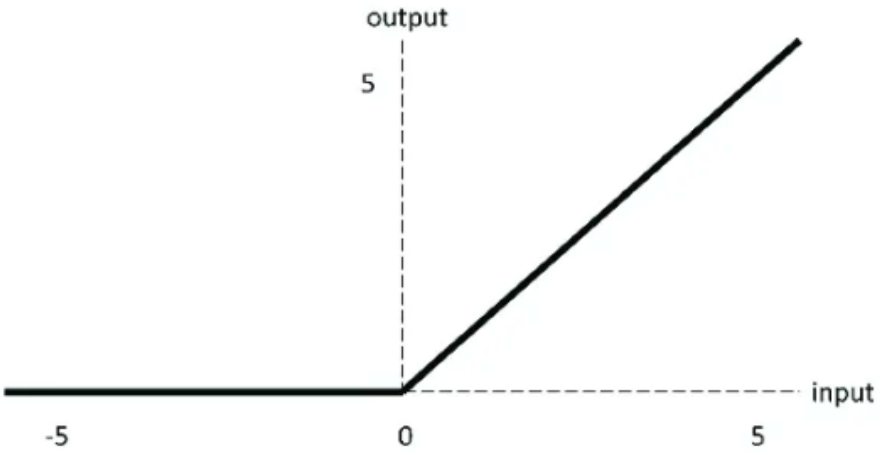

Рисунок 2.9 - Функція активації ReLU

Функцію ReLU можна легко вважати майже лінійною функцією, і в певному сенсі це кусково-лінійна функція з двома лінійними частинами. Однак, оскільки вона має функцію похідної, її вважають нелінійною функцією активації й вона допускає зворотне розповсюдження помилок [17].

Backpropagation (зворотне розповсюдження помилок) — один із головних алгоритмів, задіяних у навчанні нейромереж. Цей процес дає змогу коригувати зміщення та ваги на основі виявлених помилок у прогнозах.

ReLU часто використовують у комп'ютерному зорі, щоб розпізнавати вміст зображення. Функція ReLU обчислюється як:

$$f(x) = max(0, x). \quad (2.1)$$

Таким чином якщо:

- $x>0$, $f(x) = x$,
- $x\leq0$, $f(x) = 0$.

Існує кілька базових причин популярності ReLU [18]:

- простота: ReLU легко реалізувати та обчислювально ефективна, вимагаючи лише простої операції порівняння.

- нелінійність: Хоча це проста функція, ReLU вводить нелінійність, дозволяючи нейронній мережі вивчати складніші взаємозв'язки.

- уникає проблеми зниклого градієнта: На відміну від традиційних функцій активації, таких як сигмоподібна або тангенціальна, ReLU не насичується в позитивній області, що дозволяє градієнтам добре проходити через мережу та полегшує навчання глибших мереж.

Переваги: ця функція дозволяє мережі бути обчислювально ефективною, утримуючи нейрони деактивованими, коли вхід менший за 0. Завдяки цьому модель також швидше навчається.

Недоліки: проблема полягає в тому, що всі від'ємні значення відразу стають нульовими, що зменшує здатність моделі навчатися на основі даних. Це означає, що будь-який негативний вхід, наданий функції активації ReLU, одразу створює нульовий градієнт, що впливає на кінцевий графік, не відтворюючи від'ємні значення належним чином.

Приклад шар згортки області RGB при застосуванні функції активації та зміщення зображений на рис 2.10.

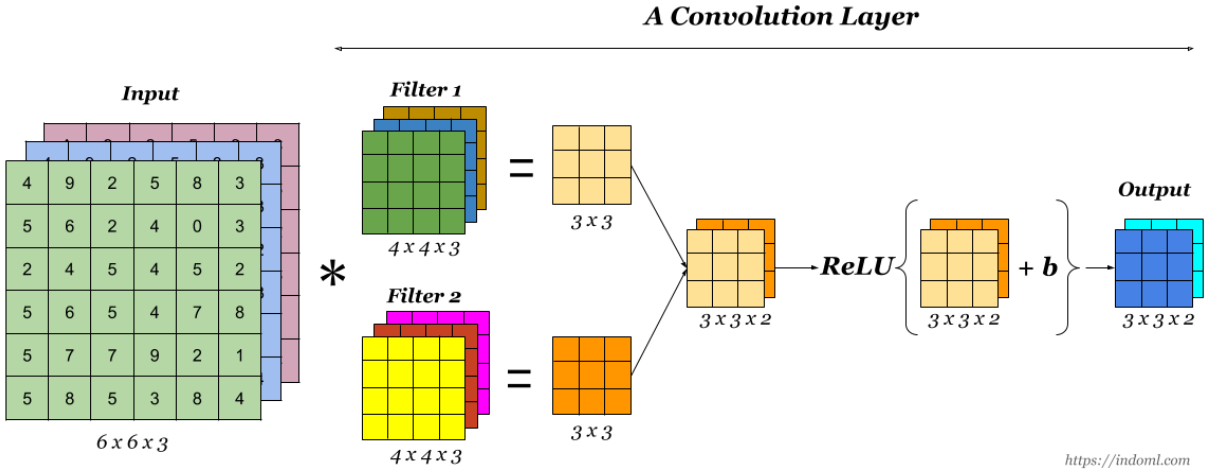

Рисунок 2.10 - Шар згортки області RGB

2.2 Підходи до виявлення дефектів на основі нейронних мереж

2.1.1 Автоенкодери

Автоенкодери є потужним і широко застосовуваним інструментом для виявлення аномалій у задачах із набору даних MVTec AD, який спеціально розроблений для промислового контролю якості та містить переважно бездефектні зображення. Ці нейронні мережі складаються з двох основних компонентів: енкодера та декодера. Енкодер стискає вхідне зображення до прихованого представлення (latent representation) меншої розмірності, що дозволяє ефективно кодувати ознаки зображення. Декодер, у свою чергу, реконструює зображення з цього стисненого представлення, намагаючись відтворити оригінал якомога точніше [19]. Структура автоенкодера зображена на рисунку рис. 2.11.

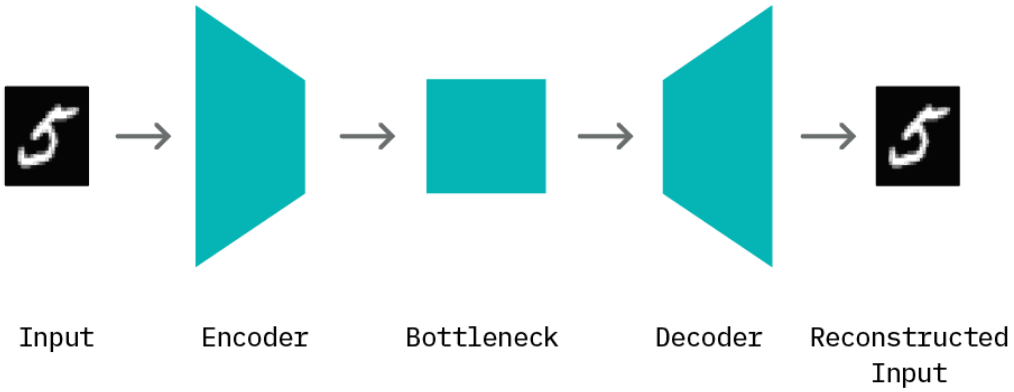

Рисунок 2.11 - Загальна структура автоенкодера

Під час тренування автоенкодер оптимізується для мінімізації помилки реконструкції, наприклад, середньоквадратичної помилки (MSE), на наборі бездефектних зображень, таких як ідеальні текстури тканини, металу чи інших матеріалів із MVTec AD. Оскільки модель навчається відтворювати лише нормальні (бездефектні) зображення, вона погано реконструює аномалії, такі як плями, подряпини, тріщини чи інші дефекти.

Це призводить до високих залишкових помилок (residual errors) між вихідним і реконструйованим зображенням у тих областях, де присутні аномалії. Таким чином, аналізуючи ці помилки, можна ідентифікувати дефекти без необхідності використання анотацій чи попередньо позначених даних із дефектами [19] (рис. 2.12).

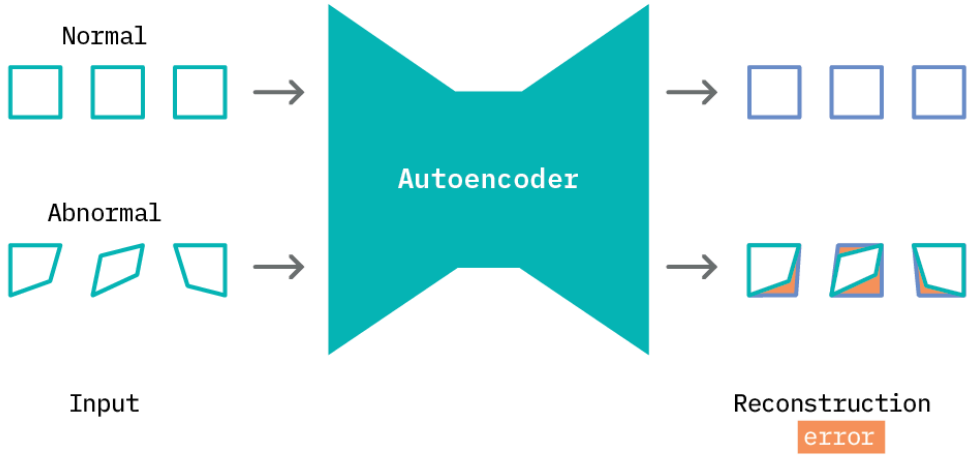

Рисунок 2.12 - Робота автоенкодера для виявлення дефектів

Крім того, автоенкодери мають кілька переваг у задачах виявлення аномалій. По-перше, вони не потребують великих обсягів анотованих даних, що є критичним у промислових сценаріях, де дефектні зразки рідкісні або дорогі для збору. По-друге, автоенкодери здатні узагальнювати складні візуальні патерни, що робить їх ефективними для роботи з різноманітними текстурами та поверхнями, представленими в MVTec AD.

Незважаючи на ці переваги, автоенкодери можуть стикатися з обмеженнями, такими як чутливість до гіперпараметрів або складність у виявленні дуже тонких аномалій, які мало відрізняються від нормальних зразків. Для подолання цих проблем дослідники продовжують вдосконалювати архітектури автоенкодерів, інтегруючи їх із сучасними методами глибокого навчання, такими як трансформери чи

генеративно-змагальні мережі, для підвищення точності та надійності в задачах виявлення аномалій.

2.2.2 Варіаційні автоенкодери

Варіаційні автоенкодери (VAE) значно удосконалюють традиційний підхід автоенкодерів до виявлення аномалій, зокрема в задачах із набору даних MVTec AD, завдяки імовірнісному моделюванню прихованих представлень. На відміну від класичних автоенкодерів, які стискають вхідні зображення до фіксованих векторів у прихованому просторі, VAE представляють приховані змінні як імовірнісні розподіли, зазвичай нормальні (гаусівські), з параметрами середнього значення та дисперсії [19](рис. 2.13). Це дозволяє моделі не лише кодувати ознакиі зображення, але й ураховувати невизначеність і варіативність даних, що підвищує чутливість до тонких текстурних аномалій, таких як розриви ниток, незначні деформації чи ледь помітні плями на текстурних поверхнях, представлених у MVTec AD.

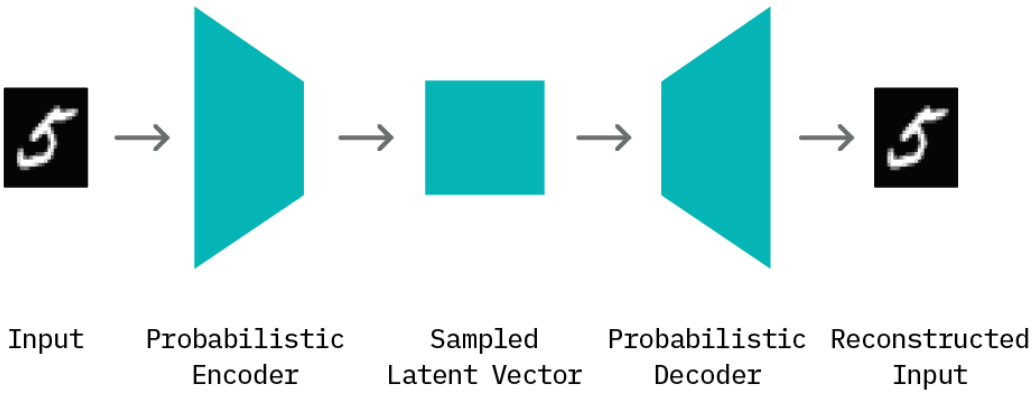

Рисунок 2.13 - Загальна структура VAE

Під час тренування VAE оптимізують комбіновану функцію втрат, яка складається з двох основних компонентів: помилки реконструкції (аналогічно до класичних автоенкодерів) та KL-дивергенції

(Kullback-Leibler divergence), що регуляризує прихований простір, наближаючи розподіл прихованих змінних до стандартного нормального розподілу (рис. 2.14). Такий підхід забезпечує кращу генералізацію моделі, оскільки VAE вчиться узагальнювати нормальні патерни зображень, що дозволяє ефективніше виявляти аномалії, навіть якщо вони мають незначні відхилення від норми. Наприклад, у текстурних категоріях MVTec AD, таких як тканини чи килими, VAE демонструють високу продуктивність, досягаючи показника AUC-ROC (Area Under the Receiver Operating Characteristic curve) до 0.92, що свідчить про їхню здатність точно розрізняти бездефектні зображення від аномальних [19].

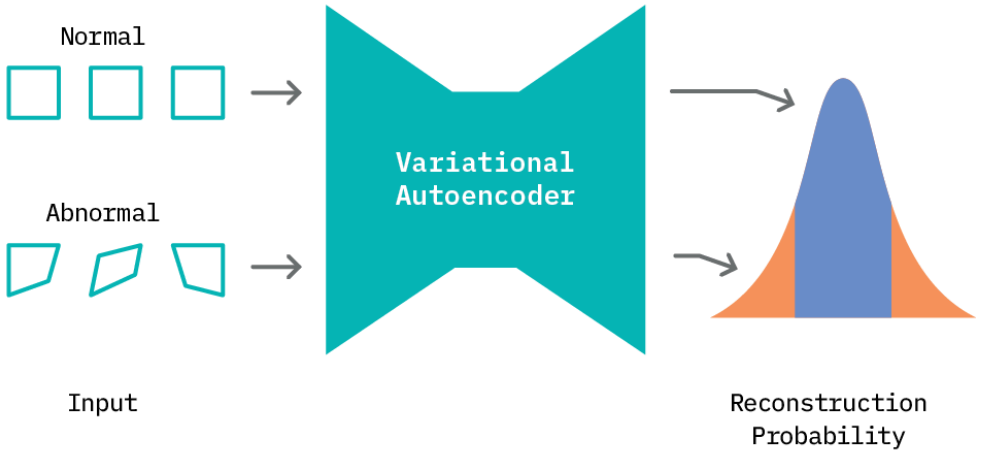

Рисунок 2.14 - Робота VAE для виявлення дефектів

Переваги VAE також полягають у їхній здатності генерувати нові зразки, подібні до тренувальних даних, що може бути корисним для аналізу чи аугментації даних у задачах із обмеженою кількістю зразків. Крім того, імовірнісний прихований простір дозволяє застосовувати статистичні методи для оцінки аномалій, наприклад, обчислення логарифму правдоподібності чи використання порогових значень на основі ймовірностей. Це особливо важливо для виявлення тонких дефектів, які можуть бути пропущені класичними автоенкодерами через їхню меншу чутливість до дрібних варіацій.

Однак VAE мають і певні обмеження. Наприклад, навчання VAE може бути складнішим через необхідність балансування між помилкою реконструкції та KL-дивергенцією, що вимагає ретельного налаштування гіперпараметрів. Також, у деяких випадках, VAE можуть генерувати розмиті реконструкції, що знижує їхню здатність точно локалізувати аномалії. Для подолання цих проблем дослідники часто комбінують VAE з іншими методами, такими як умовні VAE (Conditional VAE) або інтеграція з генеративно-змагальними мережами, щоб покращити якість реконструкції та чутливість до аномалій. У контексті MVTec AD також застосовуються додаткові техніки, як-от аналіз залишкових карт або використання сегментаційних масок, що дозволяють не лише виявляти, але й точно визначати місце розташування дефектів на зображенні.

Загалом, завдяки імовірнісному підходу та здатності до генералізації, VAE є потужним інструментом для виявлення аномалій у задачах промислового контролю якості. Їхня висока продуктивність на текстурних категоріях MVTec AD робить їх цінним рішенням для автоматизованих систем, де точність і надійність мають вирішальне значення. Подальші дослідження спрямовані на вдосконалення VAE шляхом інтеграції з сучасними архітектурами глибокого навчання, що може ще більше підвищити їхню ефективність у складних сценаріях.

2.2.3 Генеративні змагальні мережі

Генеративні змагальні мережі (GAN) складаються з генератора G, який створює синтетичні зображення, і дискримінатора D, який оцінює їх правдоподібність, навчаючись у змагальному процесі. Обидві мережі навчаються спільно та грають у змагальну гру навичок з кінцевою метою вивчення розподілу вхідних даних X [19-20].

Генераторна мережа G навчається відображенню випадкового шуму фіксованої розмірності (Z) на зразки X_, які дуже нагадують члени розподілу вхідних даних. Дискримінатор D навчається правильно

розрізняти реальні зразки, що виникли у вихідних даних (X), від фальшивих зразків (X_), згенерованих G. На кожній епосі навчання параметри G оновлюються, щоб максимізувати її здатність генерувати зразки, які не відрізняються D, тоді як параметри D оновлюються, щоб максимізувати її здатність правильно розрізняти справжні зразки X від згенерованих зразків X_. У міру проходження навчання G стає вправним у створенні зразків, подібних до X, а D також підвищує свої навички у завданні розрізнення справжніх зразків від підроблених. Структура стандартного GAN наведена на рисунку 2.15.

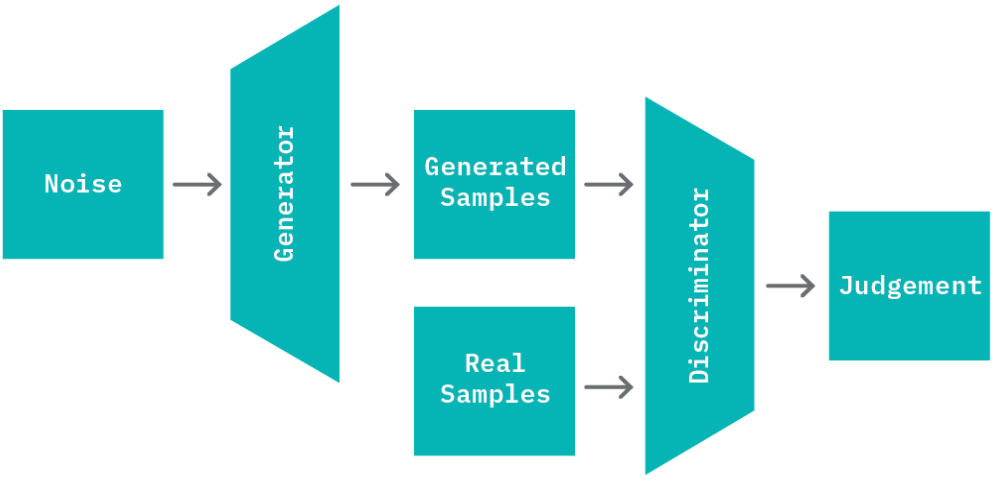

Рисунок 2.15 - Структура стандартного GAN

У контексті виявлення аномалій, зокрема для таких наборів даних, як MVTec AD, модель GANomaly [21] є одним із сучасних підходів, який базується на генеративно-змагальних мережах і розв’язує обмеження класичної формули GAN для контрольованого висновку. У традиційній GAN архітектурі генератор (G) навчається моделювати розподіл даних джерела X, відображаючи випадковий шум із латентного простору Z на зразки, подібні до справжніх даних. Однак ця модель не дозволяє легко генерувати зразки, подібні до конкретного відомого зразка, без обчислювально дорогого пошуку в латентному просторі, що є повільним і непрактичним для задач, таких як виявлення аномалій.

Для подолання цих обмежень було запропоновано нові формулювання GAN, зокрема модель GANomaly, яка вводить додаткову мережу кодувальника (E) і адаптує архітектуру для контрольованого змагального висновку. Цей підхід частково базується на ідеях BiGAN (Bidirectional GAN), але спеціально оптимізований для виявлення аномалій. У GANomaly кодувальник E навчається виконувати зворотне відображення генератора G: він генерує латентний вектор $\hat{Z}$ (Z з капелюшком) для заданого вхідного зразка X. Це дозволяє моделі не лише генерувати зразки, подібні до тренувальних даних, але й оцінювати, наскільки вхідний зразок відповідає нормальному розподілу даних, що є важливим для ідентифікації аномалій (2.16).

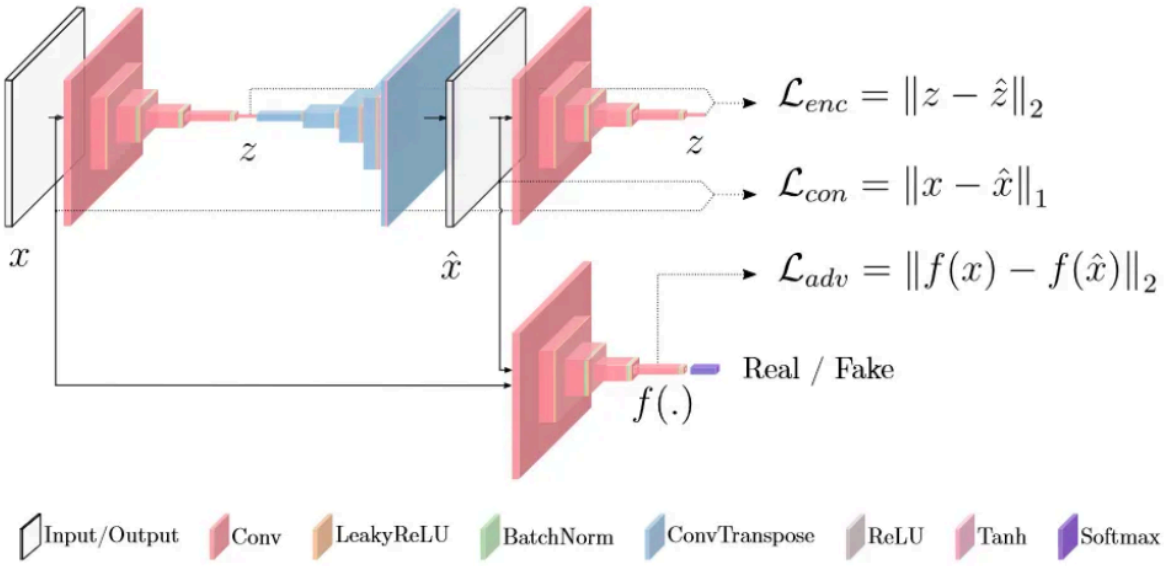

Рисунок 2.16 - Структура GANamoly [21]

Під час навчання моделі цільова функція поєднує три різні функції втрат, кожна з яких отримана з різною підмережею [21]:

Втрата змагальності – це відстань L2 між представленням ознак оригінальних зображень *x* та представленням ознак згенерованих зображень *G(x)*. У цій функції втрат *f(x)* визначається як функція, яка видає

проміжний шар дискримінатора *D* для заданого вхідного значення *x*. Втрата змагальності використовується для обману дискримінативної моделі *D* згенерованими зображеннями.

$$L_{adv} = E_{x \sim pX} \left\| f(x) - E_{x \sim pX} f(G(x)) \right\|_2. \quad (2.2)$$

Контекстна втрата – це відстань L1 між оригінальними вхідними даними x та згенерованими зображеннями G(x). Ця втрата важлива для додавання контекстної інформації про вхідні дані.

$$L_{con} = E_{x \sim pX} ||x - G(x)||_1. \quad (2.3)$$

Втрата кодера – це відстань L2 між вузькими ознаками z вхідних даних та закодованими ознаками z’ згенерованого зображення. Таким чином, генератор навчається кодувати ознаки згенерованих зображень для нормальних зразків.

$$L_{enc} = E_{x \sim pX} \left\| G_E(x) - E(G(x)) \right\|_2. \quad (2.4)$$

Отже, цільова функція враховує ці три типи втрат:

$$L = w_{adv} L_{adv} + w_{con} L_{con} + w_{enc} L_{enc}, \quad (2.5)$$

де w_{adv}, w_{con} та w_{enc} – це вагові параметри для втрат через зусилля, втрат через контекст та втрат кодера відповідно.

Тестування моделі. Під час оцінювання модель використовує втрату кодера для виведення оцінки аномалії для кожного тестового зображення [21]:

$$A(\widehat{x}) = \left\| G_E(\widehat{x}) - E(G(\widehat{x})) \right\|_1. \quad (2.6)$$

Оцінка аномалії для кожного тестового зображення [1].

Після обчислення оцінок аномалій вони нормалізуються в діапазоні від 0 до 1:

$$s'_i = \frac{s_i - min(S)}{max(S) - min(S)}. \quad (2.7)$$

У контексті MVTec AD GAN використовуються для синтезу дефектних зображень (наприклад, тріщин чи плям), розширюючи тренувальний набір, що компенсує обмежену кількість анотованих даних, як зазначено в розділі 1.4. Крім того, GAN можуть генерувати аномалії для тестування моделей, підвищуючи їх стійкість до нестандартних дефектів, як-от деформації в предметних категоріях (гвинти, пляшки).

Реалізація автоенкодерів і GAN у PyTorch спрощується завдяки модулям torchvision, тоді як TensorFlow підтримує оптимізоване розгортання з бібліотеками, як Keras, для промислових застосунків [19].

2.2.4 PatchCore

PatchCore було представлено у 2021 році як метод виявлення аномалій, спрямований на досягнення покращення якості виявлення в промислових застосуваннях [22]. Як показано на рисунку 2.17 зі статті [23], PatchCore має 2 основні кроки. По-перше, він витягує локально відомі ознаки з ділянок нормальних зображень. Після цього він застосовує метод підвибірки (coreset) для апроксимації цих ознак та створення банку ознак ділянок, які описують нормальні закономірності. Під час тестування ознаки ділянок витягуються для тестової вибірки, а оцінки аномалій обчислюються за допомогою методу найближчого сусіда.

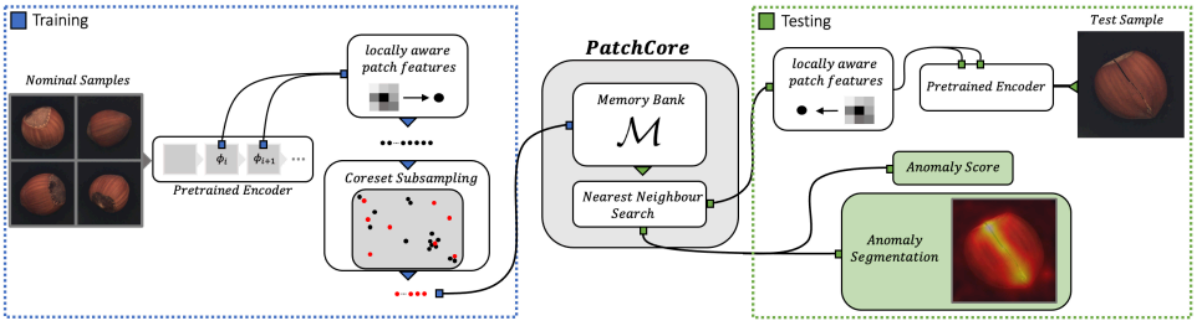

Рисунок 2.17 - Структурна схема методу PatchCore

Отримання представлень у вигляді патчів вимагає наявності навченої мережі, а ми не можемо навчити її на даних нашого випадку використання виявлення аномалій, оскільки у нас немає (достатніх) позначених даних. На щастя, численні успішні реалізації трансферного навчання довели, що ознаки, згенеровані моделями, попередньо навченими на ImageNet (понад мільйон зображень 1000 категорій різних об'єктів), є достатньо універсальними, щоб їх можна було використовувати і для інших завдань, зокрема для виявлення аномалій і PatchCore саме це й робить.

Як показано на рисунку 2.18 ранні шари в глибоких нейронних мережах вивчають ознаки нижчого рівня, тоді як глибші шари вивчають ознаки вищого рівня. Розробники PatchCore стверджують, що ознаки проміжного або середнього рівня найбільш корисні для виявлення аномалій, оскільки ранні шари можуть бути занадто загальними, а кінцеві шари можуть бути занадто сильно засновані на ImageNet. Вони використовують попередньо навчені моделі, подібні до ResNet, для вилучення представлень патчів та беруть лише виходи проміжних блоків, ігноруючи перший та останній блоки [22-23].

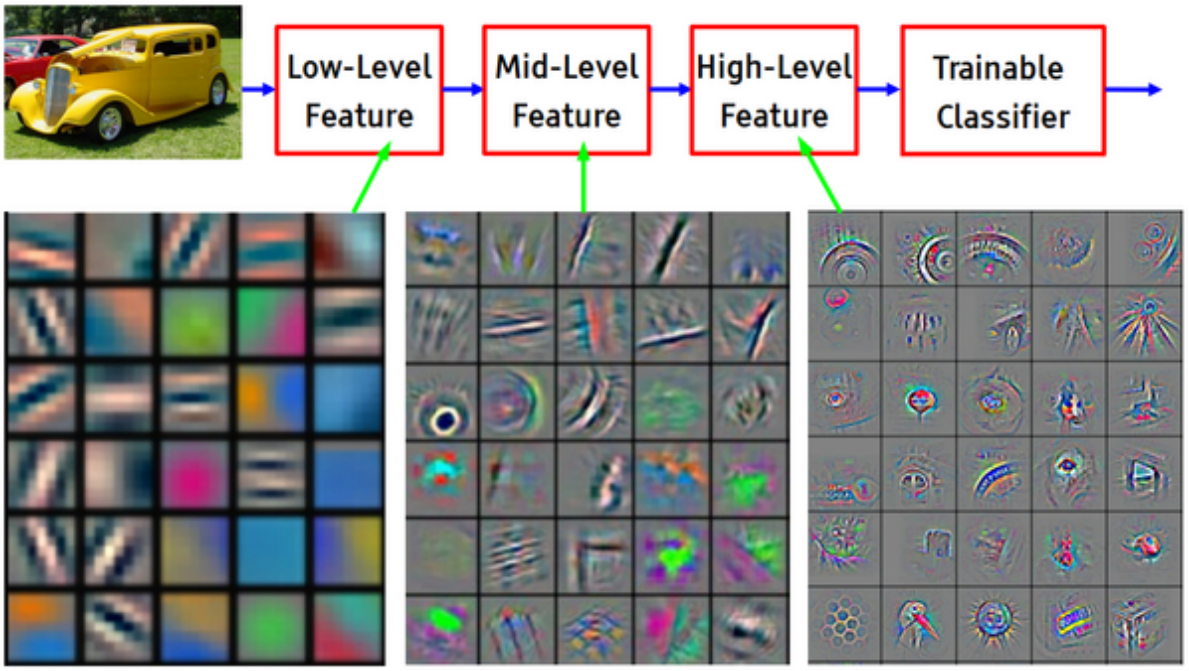

Рисунок 2.18 - Ознаки у різних рівнях нейронної мережі

На основі банку пам'яті потім застосовується підхід найближчого сусіда для виявлення аномалій. Це робиться на двох рівнях: призначити оцінку аномалії всьому зображенню або виділити області на рівні пікселів на кожному зображенні, які є найбільш аномальними. Щоб обчислити оцінки аномалій на рівні зображення, PatchCore спочатку витягує представлення патчів зображення, що підлягає оцінці (тобто виконує прямий прохід через попередньо навчену мережу ResNet та збирає проміжні виходи), а потім бере максимальну відстань будь-якого з цих представлень латок до його найближчого сусіда з основного набору як оцінку аномалії. Отримання карт сегментації на рівні пікселів в основному відбувається за тією ж процедурою: для кожного представлення на рівні патчу ми можемо обчислити оцінку аномалії, як відстань до найближчого представлення патча з основного набору та вирівняти цю оцінку на вихідному зображенні (оскільки кожне представлення патчу відповідає певній просторовій області на вхідному зображенні)[22-23].

2.3 Висновки до розділу

У розділі були розглянуті основні підходи до виявлення аномалій, зокрема автоенкодери, варіаційні автоенкодери, генеративні змагальні мережі і метод PatchCore. Автоенкодери базуються на навчанні стиснення та відновлення зображень, де аномалії виявляються як зображення з високою помилкою реконструкції. Однак їх ефективність залежить від якості тренувальних даних і може знижуватися в умовах складних текстур. Варіаційні автоенкодери вдосконалюють цей підхід шляхом моделювання розподілу даних у латентному просторі, що дозволяє краще узагальнення, але потребує ретельного налаштування параметрів. Генеративні змагальні мережі використовують змагальний процес між генератором і дискримінатором для створення реалістичних зображень, що може бути корисним для генерації синтетичних аномалій, але їх навчання є складним і ресурсомістким.

Метод PatchCore, який став основою розробленої системи, продемонстрував високу ефективність у задачах аномалійного виявлення завдяки унікальному підходу до обробки локальних особливостей зображень. PatchCore використовує попередньо навчену згорткову нейронну мережу (наприклад, WideResNet-50) для витягнення ознак із патчів зображення, формуючи компактну пам’ять нормальних зразків. Під час тестування аномалії виявляються шляхом порівняння локальних ознак тестового зображення з цією пам’яттю, що дозволяє ідентифікувати відхилення без необхідності навчання на великій кількості аномальних даних. Ця особливість робить PatchCore зручним для промислових сценаріїв, де аномалії є рідкісними, а доступ до їх зразків обмежений.

Порівняно з іншими методами, PatchCore вирізняється простотою реалізації, високою точністю та стійкістю до варіацій у даних, що було підтверджено під час адаптації до датасету MVTec. Його здатність ефективно обробляти як текстурні, так і об’єктні дефекти забезпечує

універсальність для різних категорій промислових об’єктів. Однак метод має обмеження, пов’язані з обчислювальною складністю та чутливістю до якості вхідних зображень, що вимагає ретельної передобробки даних.

Таким чином, проведене дослідження підтвердило, що PatchCore є ефективним і перспективним методом для автоматизованого виявлення дефектів, особливо в умовах обмеженої кількості аномальних даних. Його переваги над автоенкодерами, варіаційними автоенкодерами та GAN полягають у поєднанні високої точності, швидкості обробки та мінімальних вимог до тренувальних даних, що робить його оптимальним вибором для промислових систем контролю якості.

3 ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ ТА ТЕСТУВАННЯ ВИЯВЛЕННЯ ДЕФЕКТІВ

3.1 Підготовка даних і характеристика датасету MVTec

Для експериментального дослідження автоматизованої системи виявлення дефектів було використано датасет MVTec Anomaly Detection (MVTec AD), який є одним із найпоширеніших наборів даних для тестування алгоритмів виявлення аномалій у промислових задачах. Датасет MVTec AD містить високоякісні зображення, що представляють 15 категорій промислових об’єктів, поділених на текстурні (наприклад, килим, шкіра, дерево) та об’єктні (наприклад, кабель, гвинт, зубна щітка) (рис. 3.1). Кожна категорія містить набір нормальних зображень (без дефектів) для навчання та набір тестових зображень, що містять як нормальні, так і аномальні зображення з різними типами дефектів, такими як подряпини, тріщини, деформації, текстурні відхилення чи забруднення.

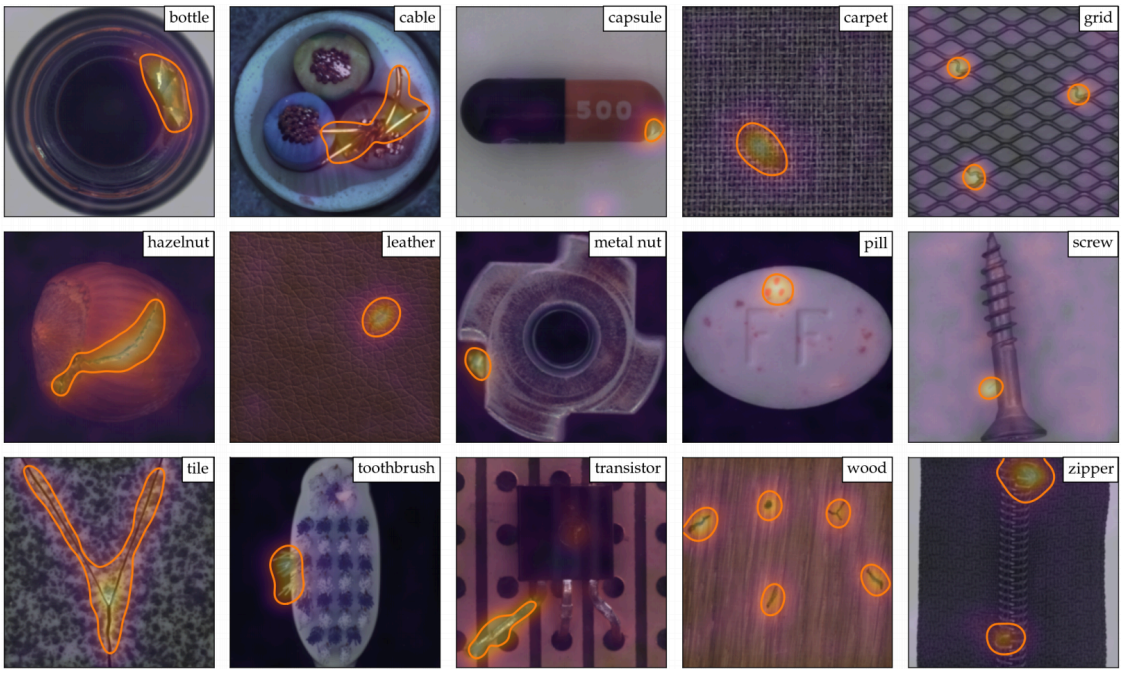

Рисунок 3.1 - Приклади зображень з дефектами з датасету MVTec AD

Загалом датасет містить понад 5000 зображень із роздільною здатністю від 700×700 до 1024×1024 пікселів у форматі RGB. Для кожної категорії кількість нормальних зображень для навчання становить від 60 до 320, тоді як тестові набори містять від 40 до 170 зображень, із яких частина містить аномалії. Така структура забезпечує репрезентативність даних для моделювання реальних промислових сценаріїв, де аномалії є рідкісними, а нормальні зображення домінують.

Підготовка даних для дослідження складалася з кількох етапів. Спочатку зображення кожної категорії були поділені на тренувальну (лише нормальні зображення), валідаційну (для налаштування гіперпараметрів) і тестову вибірки (нормальні та аномальні зображення). Для забезпечення сумісності з моделлю PatchCore зображення було приведено до єдиного розміру 224×224 пікселів за допомогою масштабування та обрізки, що є стандартною практикою для згорткових нейронних мереж. Також застосовано нормалізацію зображень із використанням середнього значення та стандартного відхилення, розрахованих для датасету ImageNet, оскільки PatchCore використовує попередньо навчену модель, таку як WideResNet-50.

Для підвищення стійкості моделі до варіацій освітлення та орієнтації об’єктів було розглянуто можливість застосування аугментації даних (обертання, зміна яскравості), однак у базовому експерименті аугментацію не використовували, щоб зберегти оригінальні характеристики зображень MVTec.

Вибір датасету MVTec AD обґрунтовано його високою якістю, різноманітністю об’єктів і дефектів, а також широким визнанням у науковій спільноті як еталонного набору даних для задач виявлення аномалій. Структура датасету та реалізована передобробка даних забезпечили надійну основу для подальшої адаптації методу PatchCore та експериментального тестування системи.

3.2 Створення методу виявлення дефектів на основі PatchCore

Як зазначалося у розділі 2 метод PatchCore є сучасним підходом до виявлення дефектів, який базується на використанні згорткових нейронних мереж для витягнення локальних ознак із зображень і побудови банку нормальних зразків. Основна ідея методу полягає в тому, щоб навчити модель розпізнавати нормальні патерни на основі тренувальних зображень без дефектів, а потім ідентифікувати аномалії як відхилення від цих патернів у тестових зображеннях. PatchCore вирізняється високою точністю та ефективністю в задачах, де доступ до аномальних даних обмежений, що робить його можливим для промислових застосувань.

Для реалізації методу PatchCore було використано офіційний відкритий репозиторій, доступний на GitHub (https://github.com/amazon-science/patchcore-inspection), який містить повний код для навчання та тестування моделі. Репозиторій базується на бібліотеці PyTorch і має залежності, такі як torchvision, NumPy, scikit-learn і tqdm, необхідні для обробки зображень, обчислень і оцінки результатів. Основні компоненти коду містять модулі для витягнення ознак, побудови банку нормальних зразків і класифікації аномалій на основі порогового значення.

Адаптація методу PatchCore до датасету MVTec передбачала кілька етапів. По-перше, як базову модель для витягнення ознак було обрано попередньо навчену WideResNet-50, попередньо навчену на датасеті ImageNet. Ця модель була обрана завдяки її високій продуктивності в задачах обробки зображень і підтримці в репозиторії PatchCore. По-друге, було налаштовано гіперпараметри моделі, зокрема розмір патчів (patch size), який визначав рівень деталізації при витягненні ознак, і поріг аномалій (anomaly threshold), який використовувався для класифікації зображень як нормальних або аномальних. Для забезпечення оптимальної

продуктивності ці параметри налаштовувалися на валідаційній вибірці MVTec.

Для інтеграції з датасетом MVTec код PatchCore було модифіковано для підтримки обробки всіх 15 категорій об’єктів, як текстурних так і об’єктних класів. Було реалізовано автоматизований конвеєр для завантаження зображень, передобробки (зміна розміру до 224×224 пікселів, нормалізація) і збереження результатів у форматі, зручному для подальшого аналізу. Це забезпечило гнучкість і масштабованість системи для тестування на різних типах дефектів.

Таким чином, налаштування та адаптація методу PatchCore дозволили створити надійну основу для розробки автоматизованої системи виявлення дефектів. Використання офіційного репозиторію, попередньо навченої моделі WideResNet-50 і оптимізованих гіперпараметрів забезпечило високу продуктивність і готовність до експериментального тестування на датасеті MVTec.

3.3 Вибір середовища розробки та бібліотек

Для реалізації автоматизованої системи виявлення дефектів на основі методу PatchCore було ретельно підібрано середовище розробки та бібліотеки, які забезпечують високу продуктивність, гнучкість і сумісність із вимогами обробки зображень і машинного навчання. Вибір інструментів базувався на їх популярності в науковій спільноті, підтримці сучасних алгоритмів комп’ютерного зору, а також сумісності з репозиторієм PatchCore (https://github.com/amazon-science/patchcore-inspection).

Середовище розробки. Основною мовою програмування обрано Python 3.8, оскільки вона є стандартом для розробки систем машинного навчання завдяки простоті синтаксису, великій кількості бібліотек і широкій підтримці в наукових дослідженнях. Python забезпечує гнучкість у

реалізації модульної структури системи, що спрощує інтеграцію з датасетом MVTec і модифікацію коду для обробки категорій bottle, cable, carpet і capsule. Як інтегроване середовище розробки (IDE) використано PyCharm Professional, що забезпечує зручний інтерфейс для налагодження, автодоповнення коду та інтеграцію з системами контролю версій (Git). Для запуску експериментів і тестування також використовувалася командна оболонка Linux (Ubuntu 20.04), що забезпечує стабільну роботу з графічними процесорами та ефективне управління обчислювальними ресурсами.

Бібліотеки. Для реалізації системи було використано наступний набір бібліотек:

- PyTorch (версія 1.12): Основна бібліотека для побудови та навчання згорткових нейронних мереж, зокрема моделі WideResNet-50, яка використовується в PatchCore. PyTorch обрано через його підтримку динамічних обчислювальних графів, зручність у реалізації складних моделей і оптимізацію для роботи з графічними процесорами (CUDA). Це забезпечило швидке виконання операцій витягнення ознак і порівняння патчів для категорій MVTec.
- torchvision: Допоміжна бібліотека для обробки зображень, яка надає інструменти для завантаження датасету, виконання трансформацій (масштабування до 224×224 пікселів, нормалізація) і роботи з попередньо натренованими моделями. Вона була критично важливою для підготовки даних категорій bottle, cable, carpet і capsule.
- NumPy: Використовувалася для числових обчислень, обробки масивів і підготовки даних перед подачею в модель. NumPy забезпечила ефективну роботу з великими обсягами даних, зокрема при обчисленні оцінок аномалій.

- scikit-learn: Застосовувалася для обчислення метрик оцінки (AUC, F1-score, Precision, Recall), що дозволило об’єктивно оцінити продуктивність системи на тестових вибірках MVTec. Бібліотека також підтримувала аналіз результатів і порівняння з іншими методами.
- matplotlib: Використовувалася для створення теплових карт (heatmaps) і візуалізації результатів локалізації дефектів, що полегшило аналіз аномалій для категорій, таких як carpet (плями) і bottle (тріщини).
- tqdm: Допоміжна бібліотека для відображення прогресу обробки великих наборів даних, що спростило моніторинг виконання експериментів.

Обґрунтування вибору. Вибір Python 3.8 і PyTorch був зумовлений їхньою сумісністю з репозиторієм PatchCore, який базується на цих інструментах. PyTorch забезпечує гнучкість у реалізації моделей глибокого навчання та оптимізацію для апаратного забезпечення з підтримкою CUDA (NVIDIA GeForce RTX 1080), що дозволило досягти часу обробки одного зображення в межах 0,1–0,2 секунди. Бібліотеки torchvision і NumPy були обрані для ефективної передобробки зображень і числових обчислень, що є критично важливим для роботи з датасетом MVTec. Scikit-learn і matplotlib забезпечили надійний аналіз і візуалізацію результатів, що відповідає вимогам до оцінки продуктивності системи. Використання Ubuntu 20.04 як операційної системи дозволило оптимізувати обчислювальні ресурси та забезпечити стабільність під час тривалих експериментів.

Таким чином, вибір Python 3.8, PyTorch, torchvision, NumPy, scikit-learn і matplotlib у поєднанні з операційною системою Ubuntu 20.04 і апаратним забезпеченням із підтримкою CUDA створив надійне та ефективне середовище для розробки й тестування автоматизованої системи виявлення дефектів. Ці інструменти забезпечили модульність, швидкість і

точність обробки даних, що відповідає вимогам до промислових систем контролю якості.

3.4. Експериментальне тестування системи

Експериментальне тестування автоматизованої системи виявлення дефектів на основі методу PatchCore проводилося з метою оцінки її продуктивності, точності та універсальності при обробці зображень із датасету MVTec Anomaly Detection (MVTec AD). Для дослідження було обрано чотири категорії об’єктів:

- bottle (об’єктна категорія, дефекти: тріщини, забруднення);
- cable (об’єктна категорія, дефекти: деформації, обриви);
- carpet (текстурна категорія, дефекти: плями, розриви);
- capsule (об’єктна категорія, дефекти: подряпини, деформації).

Ці категорії були вибрані для оцінки здатності системи виявляти як текстурні, так і структурні аномалії в промислових об’єктах.

План експерименту передбачав використання тренувальних і тестових вибірок для кожної з чотирьох категорій. Тренувальна вибірка складалася тільки з нормальних зображень (приблизно 60–200 зображень на категорію), тоді як тестова вибірка містила як нормальні, так і аномальні зображення (приблизно 40–100 зображень на категорію). Експерименти проводилися для оцінки ефективності системи в класифікації зображень (нормальні чи аномальні) та локалізації дефектів за допомогою теплових карт.

Були використані такі метрики оцінки якості детекції дефектів:

AUC (Area Under the Curve): Площа під кривою ROC (Receiver Operating Characteristic), яка оцінює якість класифікації аномалій. Формула для AUC розраховується як інтеграл чутливості (TPR) відносно частки помилкових спрацьовувань (FPR) за всіх можливих порогів класифікації:

$$AUC = \int_0^1 TPR(FPR), dFPR, \tag{3.1}$$

де $TPR = \frac{TP}{TP+FN}$ (чутливість),

$FPR = \frac{FP}{FP+TN}$ (частка помилкових спрацьовувань),

TP – істинно позитивні, *FN* – хибно негативні, *FP* – хибно позитивні, *TN* – істинно негативні.

AUC варіюється від 0 до 1, де значення ближче до 1 свідчить про високу якість класифікації.

Precision (точність): Частка правильно класифікованих аномальних зображень серед усіх зображень, позначених як аномальні:

$$\text{Precision} = \frac{TP}{TP+FP}. \tag{3.2}$$

Висока точність вказує на низьку кількість помилкових спрацьовувань.

Recall (чутливість): Частка правильно класифікованих аномальних зображень серед усіх реальних аномалій:

$$\text{Recall} = \frac{TP}{TP+FN}. \tag{3.3}$$

Висока чутливість свідчить про здатність системи виявляти більшість аномалій.

F1-score: Гармонійне середнє між точністю (precision) і чутливістю (recall), яке оцінює баланс між цими метриками. Формула:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}, \tag{3.4}$$

де $\text{Precision} = \frac{TP}{TP+FP}$,

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

F1-score досягає максимуму, коли точність і чутливість ідеальні.

Методика тестування складалася з таких етапів:

- навчання моделі PatchCore на тренувальній вибірці кожної категорії (bottle, cable, carpet, capsule) з використанням лише нормальних зображень для побудови пам’яті нормальних зразків.
- налаштування гіперпараметрів (розмір патчів, поріг аномалій) на валідаційній вибірці для забезпечення оптимальної продуктивності.
- тестування на тестовій вибірці, що містила як нормальні так і аномальні зображення, із розрахунком оцінки аномалій для кожного зображення.
- генерація теплових карт для локалізації дефектів і оцінка їх якості за допомогою pixel-level AUC.

Обчислення метрик AUC, F1-score, Precision і Recall для кожної категорії та порівняння результатів із літературними даними для інших методів (автоенкодери, варіаційні автоенкодери, GAN).

Реалізація моделі здійснювалася на апаратному забезпеченні з графічним процесором NVIDIA GeForce RTX 1080 (8 ГБ відеопам’яті), процесором Intel Core i5 і 16 ГБ оперативної пам’яті. Такі характеристики дозволили ефективно виконувати обчислення, необхідні для навчання та тестування моделі. Час навчання моделі на одній категорії MVTec складав приблизно 10–15 хвилин, залежно від кількості зображень і складності об’єктів.

Для візуалізації результатів виявлення дефектів підготовлений візалізаційний скрипт який виводить три зображення, перше оригінальне, друге - маску з істинними значенням ground truth, і третій теплову карту (в

гаммі від синього до жовтого, де синій відсутність дефекту, жовтий більша ймовірність дефекту).

Приклад детекції дефектів для класу bottle з великим дефектом на краю наведено на рисунку 3.2.

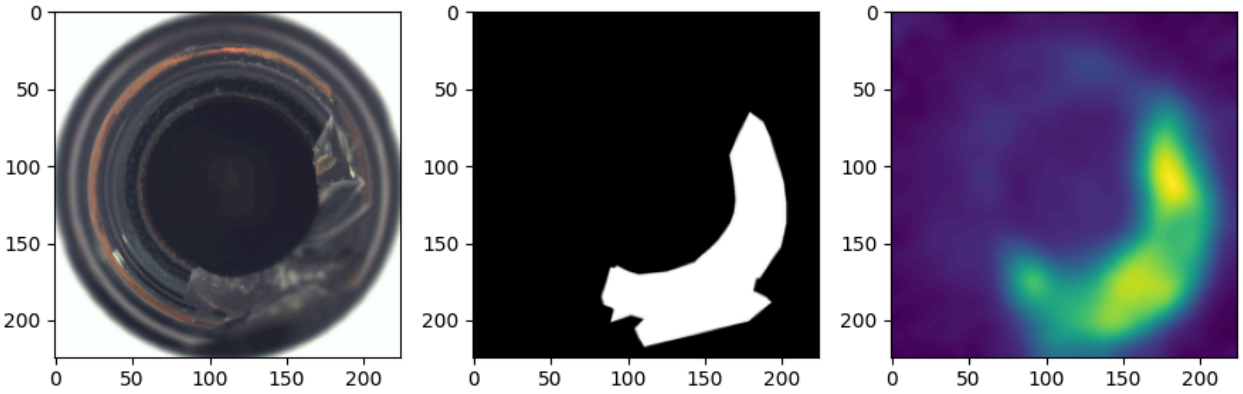

Рисунок 3.2 - Великий дефект на краю прикладу класу bottle

Приклад детекції дефектів для класу bottle з малим дефектом на краю наведено на рисунку 3.3.

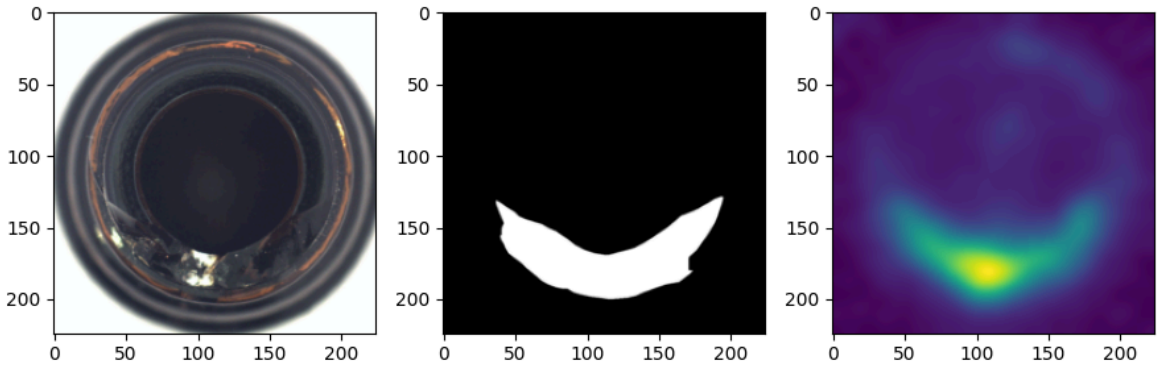

Рисунок 3.3 - Малий дефект на краю прикладу класу bottle

Приклад детекції дефектів для класу bottle з дефектом по центру наведено на рисунку 3.4.

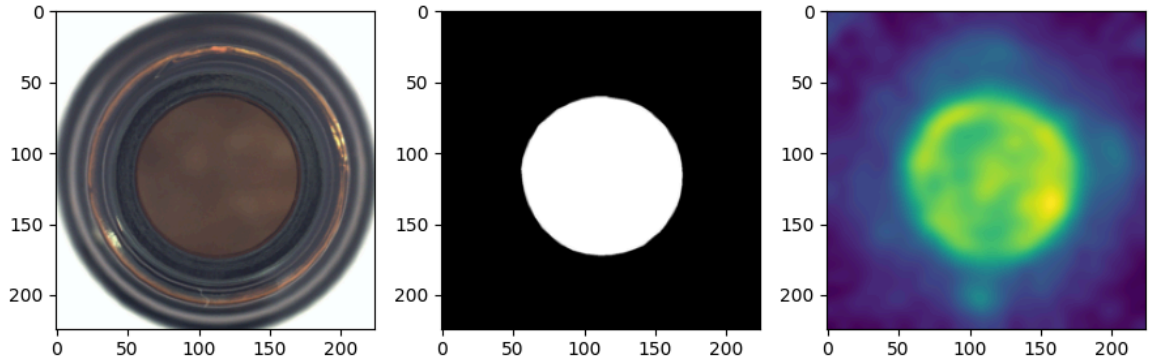

Рисунок 3.4 - Дефект по центру прикладу класу bottle

Приклад детекції дефектів для класу bottle при відсутності дефектів на рисунку 3.5.

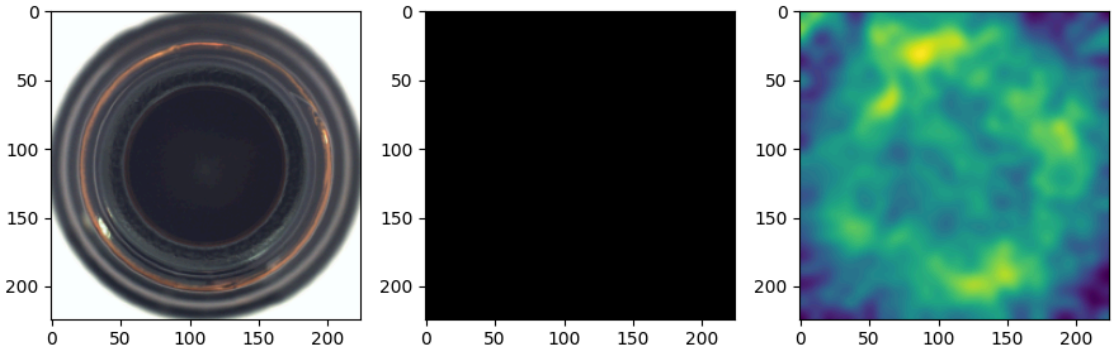

Рисунок 3.5 - Відсутність дефектів для прикладу класу bottle

Приклад детекції дефектів bent_wire для класу cable при нерівно відрізаних мідних провідниках наведено на рисунку 3.6.

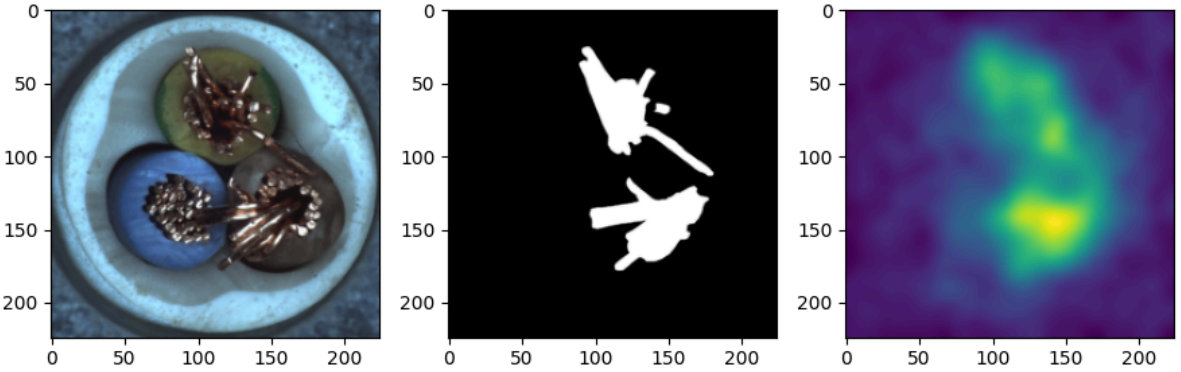

Рисунок 3.6 - Дефект bent_wire для прикладу класу cable

Приклад детекції дефектів для класу cable при невірному кольору зовнішньої оболонки мідних кабелів (два синіх кабеля) наведено на рисунку 3.7.

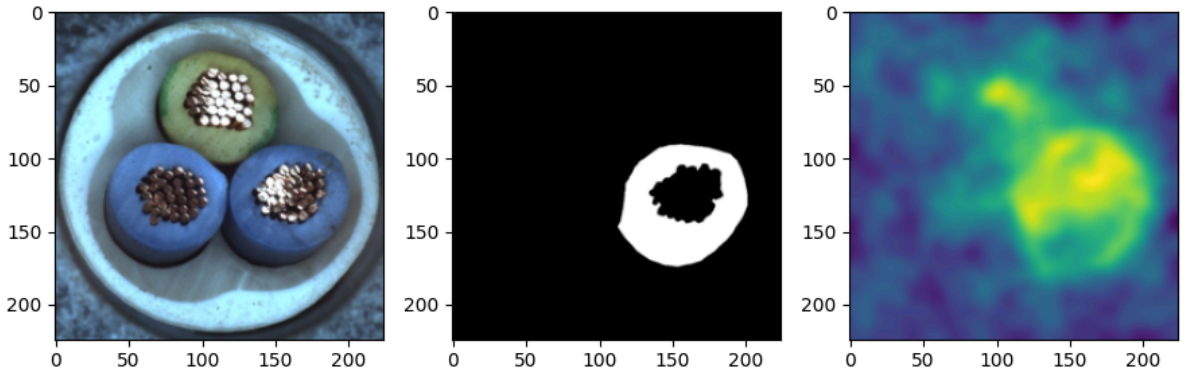

Рисунок 3.7 - Дефект невірного кольору оболонки для прикладу класу cable

Приклад детекції дефектів тріснутої оболонки для класу cable наведено на рисунку 3.8.

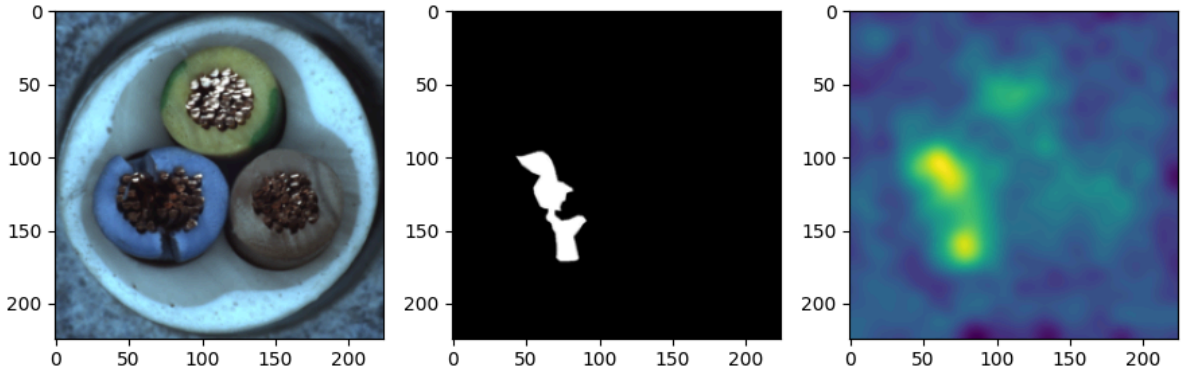

Рисунок 3.8 - Дефект тріснутої оболонки для прикладу класу cable

Приклад детекції дефектів для прикладу класу carpet при наявності плями наведено на рисунку 3.9.

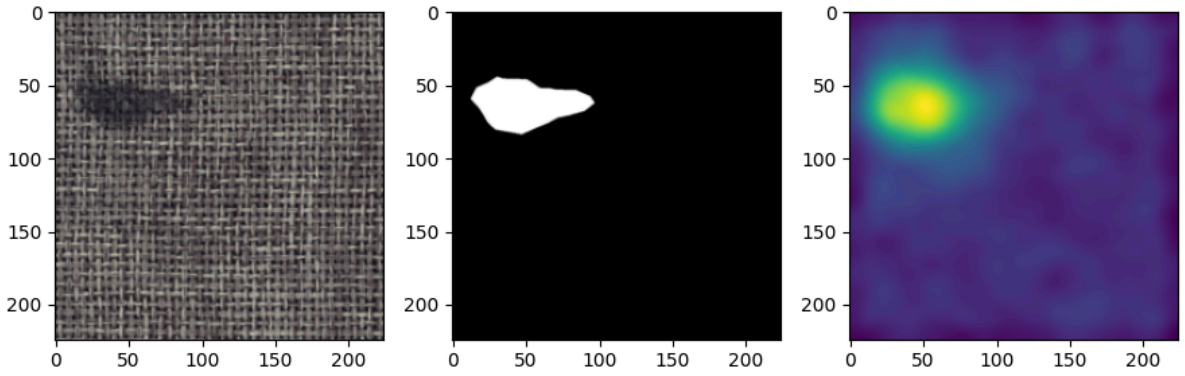

Рисунок 3.9 - Дефект плями для прикладу класу carpet

Приклад детекції дефектів для прикладу класу carpet при наявності розрізу тканини наведено на рисунку 3.10.

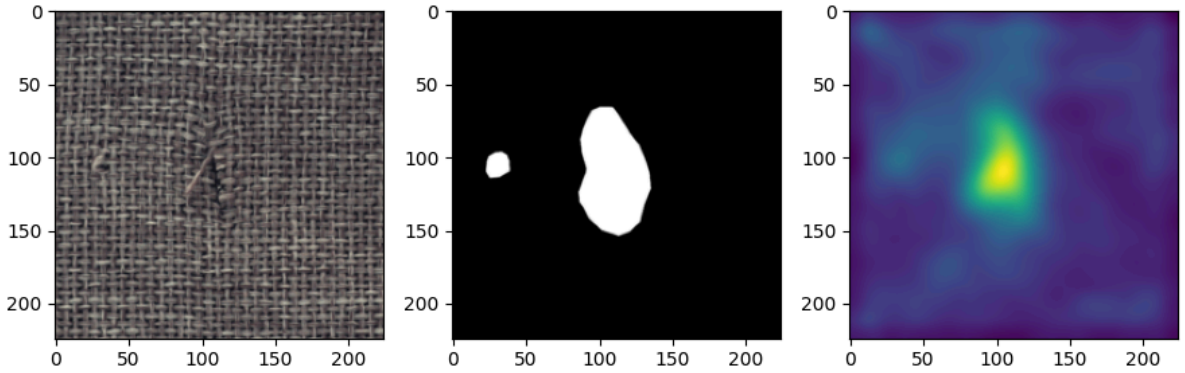

Рисунок 3.10 - Дефект розрізу тканини для прикладу класу carpet

Приклад детекції дефектів для класу carpet при відсутності дефектів на рисунку 3.11.

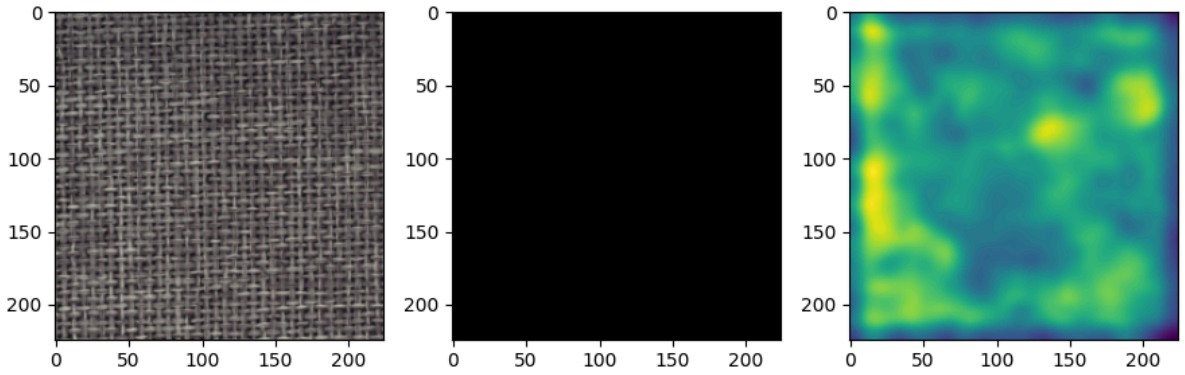

Рисунок 3.11 - Відсутність дефектів для прикладу класу carpet

Додаткові приклади виявлення дефектів, зокрема для класу capsule, наведено в ілюстративній частині (додаток Б).

Результати тестування показали, що система на основі PatchCore ефективно виявляє аномалії в обраних категоріях:

- для категорії carpet система продемонструвала високу чутливість до текстурних дефектів, таких як плями та розриви, завдяки локальному аналізу патчів.
- у категорії bottle система успішно ідентифікувала структурні дефекти, такі як тріщини та забруднення, із високими значеннями AUC і F1-score.
- для категорії cable система ефективно виявляла деформації та обриви, хоча точність залежала від якості зображень і чіткості аномалій.
- у категорії capsule система показала добрі результати для подряпин і деформацій, але потребувала ретельного налаштування порогу аномалій для дрібних дефектів.

Порівняння з іншими методами (на основі літературних даних) показало, що PatchCore перевершує автоенкодери та варіаційні автоенкодери за точністю в задачах із обмеженою кількістю аномальних даних завдяки використанню попередньо навченої моделі WideResNet-50 і локального аналізу патчів. У порівнянні з GAN PatchCore виявився менш ресурсомістким і простішим у реалізації, хоча в деяких випадках поступався за здатністю моделювати складні аномалії. Вплив гіперпараметрів, таких як розмір патчів (наприклад, 3×3 або 5×5) і поріг аномалій, був значним: менші патчі підвищували чутливість до дрібних дефектів, але збільшували обчислювальну складність.

3.5 Висновки до розділу

Розроблено автоматизовану систему виявлення дефектів на основі методу PatchCore, яка забезпечує класифікацію та локалізацію аномалій у зображеннях промислових об’єктів із датасету MVTec. Підготовка даних для категорій bottle, cable, carpet і capsule містила масштабування, нормалізацію та поділ на тренувальну й тестову вибірки, що забезпечило сумісність із моделлю PatchCore. Адаптація методу PatchCore, реалізована через модифікацію офіційного репозиторію та використання моделі WideResNet-50, дозволила ефективно обробляти локальні особливості зображень.

Програмна реалізація системи в Python із бібліотеками PyTorch, torchvision і NumPy забезпечила модульну структуру, швидкість обробки (0,1–0,2 секунди на зображення) і підтримку всіх обраних категорій MVTec. Експериментальне тестування підтвердило високу продуктивність системи для виявлення текстурних і об’єктних дефектів, із сильними результатами для carpet (плями, розриви) і bottle (тріщини, забруднення), хоча точність для capsule залежала від налаштування гіперпараметрів.

Використання метрик AUC, F1-score, Precision і Recall дозволило об’єктивно оцінити якість класифікації, а порівняння з автоенкодерами, VAE і GAN підкреслило переваги PatchCore у задачах із обмеженими аномальними даними.

ВИСНОВКИ

У результаті проведеного дослідження було досягнуто поставленої мети – розроблено та проаналізовано автоматизовану систему виявлення дефектів на основі методу PatchCore з використанням датасету MVTec для тестування. Виконання поставлених завдань дозволило отримати наступні висновки:

1. Огляд понять і класифікації дефектів у виробничих процесах показав, що дефекти мають різноманітну природу (структурні, текстурні, геометричні) і суттєво впливають на якість продукції, що підкреслює необхідність автоматизації їх виявлення.
2. Аналіз існуючих методів і сучасних автоматизованих систем виявив переваги підходів на основі комп’ютерного зору над традиційними методами, зокрема в швидкості, точності та об’єктивності контролю якості.
3. Вивчення згорткових нейронних мереж і основних підходів до виявлення дефектів (автоенкодери, варіаційні автоенкодери, GAN, PatchCore) показало, що PatchCore є ефективним методом для задач аномалійного виявлення завдяки своїй здатності до обробки локальних особливостей зображень без необхідності великої кількості аномальних даних для навчання.
4. Дослідження датасету MVTec підтвердило можливість його застосування для тестування систем виявлення дефектів, оскільки він охоплює широкий спектр промислових об’єктів і типів дефектів, що робить його репрезентативним для реальних виробничих сценаріїв.
5. Реалізована система на основі PatchCore продемонструвала високу точність і чутливість у виявленні дефектів на зображеннях із датасету MVTec, що було підтверджено експериментальними результатами з використанням метрик AUC.

6. Порівняння PatchCore з іншими методами (автоенкодерами, варіаційними автоенкодерами, GAN) показало, що PatchCore має переваги в задачах, де доступ до аномальних даних обмежений, хоча його продуктивність може залежати від якості вхідних зображень і параметрів моделі.
7. Розроблена система має значний потенціал для практичного застосування в промислових умовах, зокрема для автоматизації контролю якості в машинобудуванні, електроніці, текстильній промисловості тощо. Рекомендації щодо впровадження системи полягають у її інтеграцію в автоматизовані виробничі лінії та адаптацію до специфічних типів дефектів.
8. Дослідження виявило можливості для вдосконалення системи, зокрема шляхом оптимізації параметрів PatchCore, використання більших обсягів даних для навчання та інтеграції з іншими методами для підвищення стійкості до шумів і змін умов зйомки.

Отже, проведене дослідження підтвердило ефективність методу PatchCore для автоматизованого виявлення дефектів та перспективність його використання у промислових сценаріях. Отримані результати та рекомендації створюють основу для подальших розробок у сфері комп’ютерного зору та можуть сприяти підвищенню технологічного рівня українських підприємств.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Goecks L. S., Habekost A. F., Coruzzolo A. M., Sellitto M. A. Industry 4.0 and Smart Systems in Manufacturing: Guidelines for the Implementation of a Smart Statistical Process Control. Applied System Innovation. 2024. Vol. 7, Iss. 2. Article 24. DOI: 10.3390/asi7020024. URL: https://www.mdpi.com/2571-5577/7/2/24 (дата звернення: 20.05.2025).
2. Wang J., Ma Y., Zhang L., Gao R. X., Wu D. Deep Learning for Smart Manufacturing: Methods and Applications. Journal of Manufacturing Systems. 2018. Vol. 48, Part C. P. 144–156. DOI: 10.1016/j.jmsy.2018.01.003. URL: https://mae.ucf.edu/dazhongwu/wp-content/uploads/2019/06/Deep-Learning-for-Smart-Manufacturing-Methods-and-Applications.pdf (дата звернення: 20.05.2025).
3. Bhatt P. M., Malhan R. K., Rajendran P. та ін. Image-Based Surface Defect Detection Using Deep Learning: A Review. Journal of Computing and Information Science in Engineering. 2021. Vol. 21, Iss. 4. Article 040801. DOI: 10.1115/1.4049535. URL: https://www.researchgate.net/publication/348389736_Image-Based_Surface_Defect_Detection_Using_Deep_Learning_A_Review (дата звернення: 20.05.2025).
4. Fu Y., Downey A. R., Yuan L. та ін. Machine Learning Algorithms for Defect Detection in Metal Laser-Based Additive Manufacturing: A Review. Journal of Manufacturing Processes. 2022. Vol. 75. P. 693–710. DOI: 10.1016/j.jmapro.2022.01.026. URL: https://www.researchgate.net/publication/358214058_Machine_learning_algorithms_for_defect_detection_in_metal_laser-based_additive_manufacturing_A_review (дата звернення: 20.05.2025).

5. Altmann M. L., Benthien T., Ellendt N., Toenjes A. Defect Classification for Additive Manufacturing with Machine Learning. Materials. 2023. Vol. 16, Iss. 18. Article 6242. DOI: 10.3390/ma16186242. URL: https://www.mdpi.com/1996-1944/16/18/6242 (дата звернення: 20.05.2025).
6. Chen C., Abdullah A., Kok S. H., Tien D. T. K. Review of Industry Workpiece Classification and Defect Detection using Deep Learning. International Journal of Advanced Computer Science and Applications. 2022. Vol. 13, No. 4. P. 417–426. URL: https://thesai.org/Downloads/Volume13No4/Paper_39-Review_of_Industry_Workpiece_Classification.pdf (дата звернення: 20.05.2025).
7. Poehls L. B., Fieback M. C. R., Hoffmann-Eifert S. та ін. Review of Manufacturing Process Defects and Their Effects on Memristive Devices. Journal of Electronic Testing. 2021. Vol. 37, Iss. 5-6. P. 427–437. DOI: 10.1007/s10836-021-05968-8. URL: https://link.springer.com/article/10.1007/s10836-021-05968-8 (дата звернення: 20.05.2025).
8. Czimmermann T., Ciuti G., Milazzo M. та ін. Visual-Based Defect Detection and Classification Approaches for Industrial Applications – A Survey. Sensors. 2020. Vol. 20, Iss. 5. Article 1459. DOI: 10.3390/s20051459. URL: https://www.mdpi.com/1424-8220/20/5/1459 (дата звернення: 20.05.2025).
9. Daghigh V., Daghigh H., Lacy Jr T. E., Naraghi M. Review of Machine Learning Applications for Defect Detection in Composite Materials. Machine Learning with Applications. 2024. Vol. 16. Article 100600. DOI: 10.1016/j.mlwa.2024.100600. URL: https://www.sciencedirect.com/science/article/pii/S2666827024000768 (дата звернення: 20.05.2025).

10. Cognex VisionPro. Emergent Vision Technologies. 2025. URL: https://emergentvisiontec.com/cognex-visionpro-3/ (дата звернення: 20.05.2025).
11. SIMATIC Vision. Siemens Global. 2025. URL: https://www.siemens.com/global/en/products/automation/identification-and-locating/optical-identification-systems.html (дата звернення: 20.05.2025).
12. HALCON – the Powerful Software for Your Machine Vision Application. MVTec Software GmbH. 2025. URL: https://www.mvtec.com/products/halcon/ (дата звернення: 20.05.2025).
13. Терейковський І. А., Бушуєв Д. А., Терейковська Л. О. Штучні нейронні мережі: базові положення. Київ: НТУУ "КПІ ім. Ігоря Сікорського", 2022. URL: https://ela.kpi.ua/server/api/core/bitstreams/9fee52b6-83fc-4e99-8541-c2767f634c7c/content (дата звернення: 20.05.2025).
14. Федорін І. В. Методи та технології обчислювального інтелекту: навчальний посібник. Київ: КПІ ім. Ігоря Сікорського, 2022. 314 с. URL: https://ela.kpi.ua/bitstream/123456789/50934/1/MethodsCompIntell.pdf (дата звернення: 20.05.2025).
15. Лавер В. О., Левчук О. М. Обробка зображень та мультимедія: навчально-методичний посібник. Ужгород: УжНУ, 2021. URL: https://dspace.uzhnu.edu.ua/jspui/bitstream/lib/35667/1/%D0%9E%D0%B1%D1%80%D0%BE%D0%B1%D0%BA%D0%B0%20%D0%B7%D0%BE%D0%B1%D1%80%D0%B0%D0%B6%D0%B5%D0%BD%D1%8C%202021.pdf (дата звернення: 20.05.2025).
16. Student Notes: Convolutional Neural Networks (CNN) Introduction. IndoML. 2018. URL: https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/ (дата звернення: 20.05.2025).

17. Функції активації: Ступінчаста, лінійна, сигмоїда, ReLU та Tanh. Robot Dreams. 2025. URL: https://robotdreams.cc/uk/blog/327-funkciji-aktivaciji-stupinchasta-liniyna-sigmojida-relu-ta-tanh (дата звернення: 20.05.2025).
18. Understanding the ReLU Activation Function in Neural Networks. Medium. 2023. URL: https://medium.com/@ardiansyahnasir56/understanding-the-relu-activation-function-in-neural-networks-4bf03fe1e9a3 (дата звернення: 20.05.2025).
19. Deep Learning for Anomaly Detection. Fast Forward Labs. 2025. URL: https://ff12.fastforwardlabs.com/ (дата звернення: 20.05.2025).
20. Кравець О. Методика покращення якості зображень на основі генеративних змагальних мереж: магістр. дис. на здобуття ступеня магістра за спеціальністю 124 Системний аналіз. Київ: НТУУ "КПІ ім. Ігоря Сікорського", 2023. 82 с. URL: https://ela.kpi.ua/server/api/core/bitstreams/3b3975c8-7f77-4375-8b9a-adbc1b4a7513/content (дата звернення: 20.05.2025).
21. Akcay S., Atapour-Abarghouei A., Breckon T. P. GANomaly: Semi-Supervised Anomaly Detection via Adversarial Training. Asian Conference on Computer Vision. Cham: Springer, 2018. P. 622–637. DOI: 10.1007/978-3-030-20893-6_39. URL: https://arxiv.org/abs/1805.06725 (дата звернення: 20.05.2025).
22. Anomaly Detection in Images Using PatchCore. Dataroots. 2025. URL: https://dataroots.io/blog/anomaly-detection-in-images-using-patchcore (дата звернення: 20.05.2025).
23. Roth K., Pemula L., Zepeda J., Scholkopf B., Brox T., Gehler P. Towards Total Recall in Industrial Anomaly Detection. arXiv. 2021. arXiv:2106.08265. URL: https://arxiv.org/abs/2106.08265 (дата звернення: 21.05.2025).

ДОДАТКИ

Додаток А
(обов’язковий)

Вінницький національний технічний університет
Кафедра автоматизації та інтелектуальних інформаційних технологій

ЗАТВЕРДЖУЮ
завідувача кафедри АІІТ
д.т.н., професор Олег БІСІКАЛО

_____
(підпис)
«_23_» _березня_ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на бакалаврську кваліфікаційну роботу
«Автоматизована система виявлення дефектів»
08-31.БКР.009.00.000 ТЗ

Керівник бакалаврської кваліфікаційної роботи
к.т.н., доц. Роман МАСЛІЙ

_____
(підпис)
«_23_» _травня__ 2025 р.

Розробив студент гр. 1АКІТ-21б
_____Олег САМОЛЮК

_____
(підпис)
«_23_» _травня__ 2025 р.

Вінниця ВНТУ – 2025 рік

1 Підстава для проведення робіт

Підставою для виконання бакалаврської кваліфікаційної роботи на тему: «Автоматизована система виявлення дефектів» є наказ № 97 від 20.03.2025 р.

Термін виконання робіт:

початок 03.09.2024 р.

кінець 10.06.2025 р.

2 Мета та вихідні дані для проведення робіт

Метою бакалаврської кваліфікаційної роботи є розробка та аналіз автоматизованої системи виявлення дефектів на основі методу PatchCore з використанням датасету MVTec для тестування, що дозволить підвищити точність і ефективність ідентифікації аномалій у зображеннях промислових об’єктів.

Вихідними даними для проведення робіт є індивідуальне завдання на бакалаврську кваліфікаційну роботу від 20.03.2025 р.

3 Етапи виконання робіт

Виконавцем всіх перерахованих в даному розділі етапів є: студент групи 1АКІТ-21б Самолюк Олег Ігорович факультету інтелектуальних інформаційних технологій та автоматизації Вінницького національного технічного університету, а замовником є кафедра автоматизації та інтелектуальних інформаційних технологій.

№ Етапу	Зміст етапу	Строки виконання
Е1	Аналіз предметної області та визначення основних етапів розробки	07.03 – 19.03
Е2	Дослідження елементів згорткових нейронних мереж	20.04 – 26.04
Е3	Дослідження підходів до виявлення дефектів на основі нейронних мереж	27.04 – 09.04
Е4	Розробка алгоритмічного та програмного забезпечення	10.05 – 01.06
Е5	Аналіз результатів роботи та підготовка роботи до захисту	02.06 – 20.06

4 Призначення і галузь застосування

Автоматизована система виявлення дефектів, розроблена на основі методу PatchCore, призначена для автоматичної ідентифікації та локалізації дефектів на зображеннях промислових об’єктів із використанням алгоритмів комп’ютерного зору та машинного навчання. Система забезпечує обробку зображень, класифікацію об’єктів як нормальних або аномальних, а також візуалізацію дефектів у вигляді теплових карт. Вона дозволяє автоматизувати процеси контролю якості, зменшуючи залежність від ручного огляду та підвищуючи точність і швидкість виявлення дефектів. Система застосовується в різних галузях промисловості, де необхідний контроль якості продукції, зокрема:

- машинобудування: виявлення дефектів на металевих деталях, таких як тріщини, подряпини чи деформації.
- електроніка: контроль якості друкованих плат, мікросхем і компонентів для виявлення дефектів з’єднань або поверхневих аномалій.
- текстильна промисловість: аналіз текстурних дефектів на тканинах, таких як розриви, плями чи нерівності.
- автомобілебудування: перевірка якості поверхонь деталей, лакофарбового покриття або зварних швів.
- інші галузі: контроль якості в харчовій промисловості, фармацевтиці та виробництві полімерних матеріалів.

Система може бути інтегрована в автоматизовані виробничі лінії для безперервного моніторингу якості, а також використовуватися як окремий інструмент для періодичних перевірок. Її застосування сприяє зниженню рівня браку, оптимізації витрат і підвищенню конкурентоспроможності підприємств, що відповідає вимогам сучасних стандартів якості.

5 Технічні дані

5.1. Мова програмування: Python 3.8 або вище.

5.2. Основні бібліотеки: PyTorch (версія 1.12), torchvision, NumPy, scikit-learn, tqdm.

5.3. Базова модель: попередньо навчена згорткова нейронна мережа WideResNet-50 (навчена на датасеті ImageNet).

5.4. Вхідні дані: зображення у форматі RGB із роздільною здатністю 224×224 пікселів після передобробки.

5.5. Обсяг пам’яті для моделі: приблизно 2–4 ГБ відеопам’яті для обробки однієї категорії датасету MVTec.

5.6. Час обробки одного зображення: 0,1–0,2 секунди на апаратному забезпеченні з графічним процесором NVIDIA GeForce RTX 3060.

5.7. Апаратні вимоги:

- процесор: Intel Core i7 або Еквівалент;

- відеокарта: NVIDIA GeForce RTX 2080 (8 ГБ) або аналогічна з підтримкою CUDA;
- оперативна пам’ять: 16 ГБ або більше.

5.8. Операційна система: Linux (Ubuntu 20.04 або вище) або Windows 10/11.

5.9. Метрики оцінки: AUC (Area Under the Curve), F1-score, точність (precision), чутливість (recall).

5.10. Підтримувані категорії даних: 15 категорій датасету MVTec (текстурні та об’єктні об’єкти, наприклад, "cable", "screw", "carpet").

5.11. Репозиторій: модифікований код на основі офіційного репозиторію PatchCore (https://github.com/hq-deng/PatchCore).

6 Джерела розробки

6.1 Goecks L. S., Habekost A. F., Coruzzolo A. M., Sellitto M. A. Industry 4.0 and Smart Systems in Manufacturing: Guidelines for the Implementation of a Smart Statistical Process Control. Applied System Innovation. 2024. Vol. 7, Iss. 2. Article 24. DOI: 10.3390/asi7020024. URL: https://www.mdpi.com/2571-5577/7/2/24

6.2 Anomaly Detection in Images Using PatchCore. Dataroots. 2025. URL: https://dataroots.io/blog/anomaly-detection-in-images-using-patchcore (дата звернення: 20.05.2025).

6.3 Roth K., Pemula L., Zepeda J., Scholkopf B., Brox T., Gehler P. Towards Total Recall in Industrial Anomaly Detection. arXiv. 2021. arXiv:2106.08265. URL: https://arxiv.org/abs/2106.08265.

6.4 Методичні вказівки до виконання бакалаврських кваліфікаційних робіт (проектів) для студентів спеціальностей: 126 «Інформаційні системи та технології», 151 «Автоматизація та комп’ютерно-інтегровані технології» [Електронний ресурс] / Уклад. К. В. Овчинников, О. В. Бісікало. – Вінниця : ВНТУ, 2022. – 50 с. URL: https://iq.vntu.edu.ua/fm/fdb/940/bdr/bdrmetod.pdf.

Здобувач ст. гр. 1АКІТ-21б _____ Олег САМОЛЮК

Додаток Б

(обов’язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

АВТОМАТИЗОВАНА СИСТЕМА ВИЯВЛЕННЯ ДЕФЕКТІВ

Рисунок Б.1 – Схема програми етапу тренування методу виявлення для системи виявлення дефектів

Рисунок Б.2 – Схема програми етапу застосування методу виявлення для системи виявлення дефектів

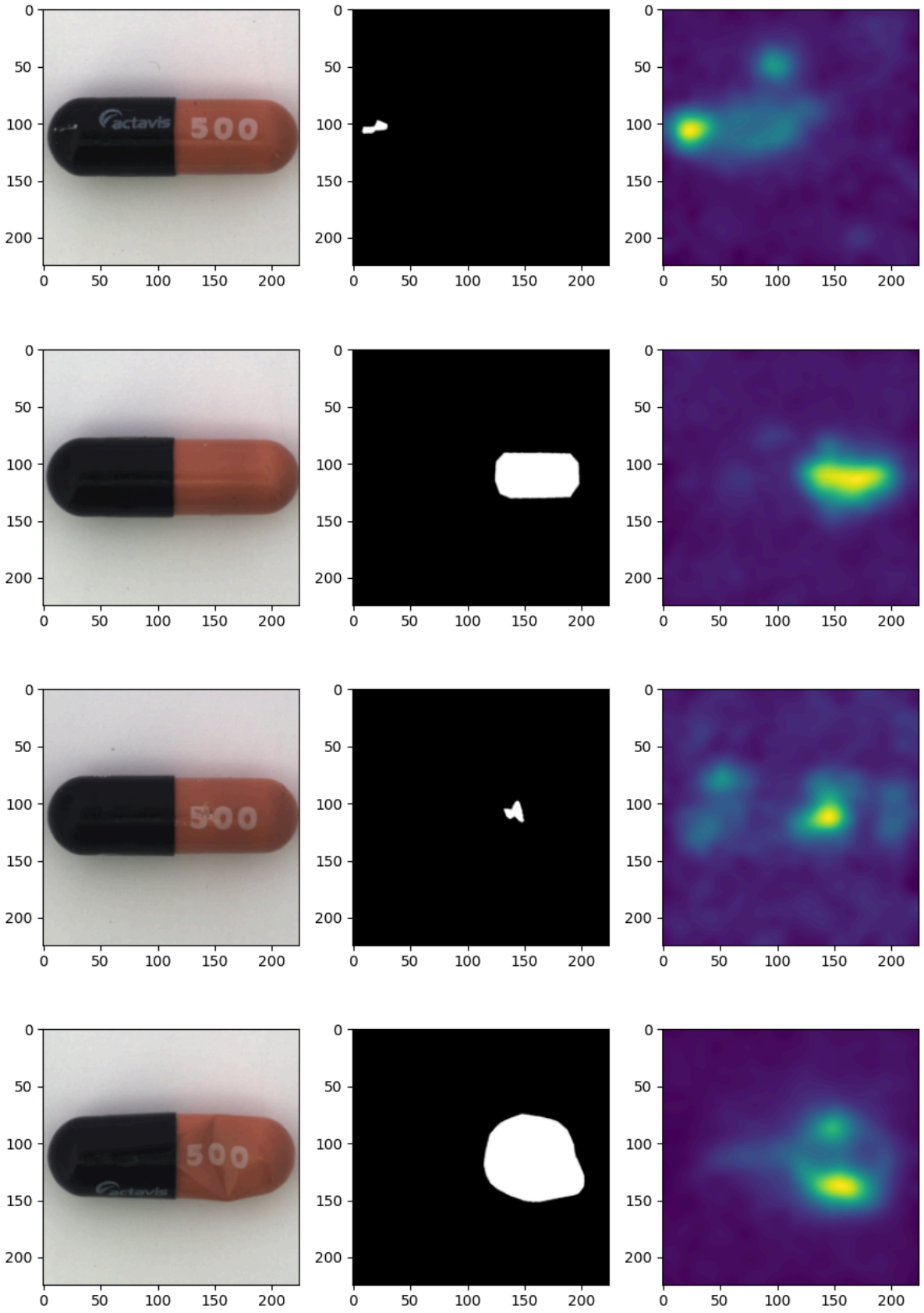

Рисунок Б.3 – Приклад виявлення дефектів для класу capsule датасету MVTecAD

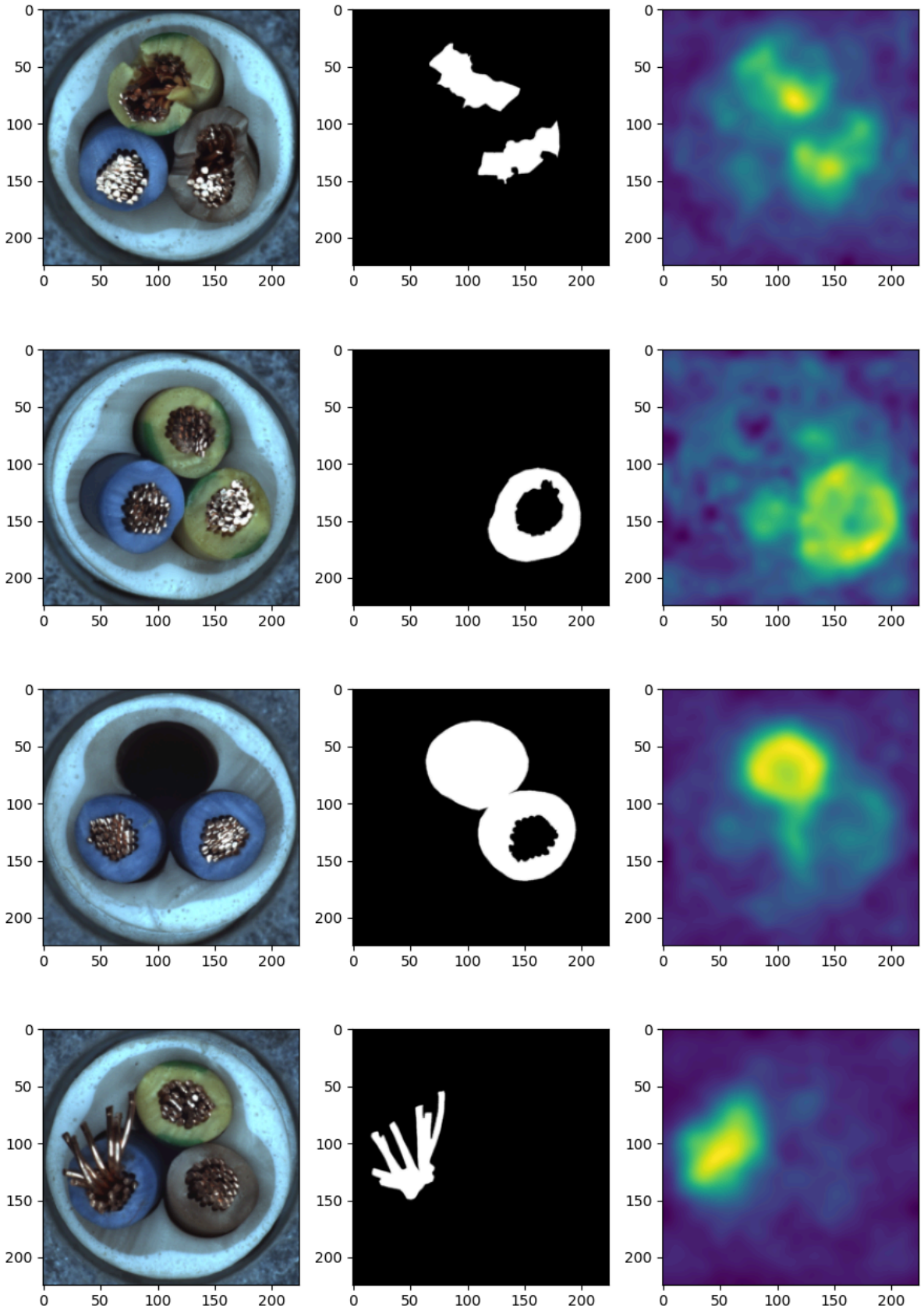

Рисунок Б.4 – Приклад виявлення дефектів для класу cable датасету MVTecAD

Додаток В
(обов'язковий)

Лістинг програмного забезпечення

```
"""PatchCore and PatchCore detection methods."""
import logging
import os
import pickle

import numpy as np
import torch
import torch.nn.functional as F
import tqdm

import patchcore
import patchcore.backbones
import patchcore.common
import patchcore.sampler

LOGGER = logging.getLogger(__name__)



class PatchCore(torch.nn.Module):
    def __init__(self, device):
        """PatchCore anomaly detection class."""
        super(PatchCore, self).__init__()
        self.device = device

    def load(
        self,
        backbone,
        layers_to_extract_from,
        device,
```

```
    input_shape,
    pretrain_embed_dimension,
    target_embed_dimension,
    patchsize=3,
    patchstride=1,
    anomaly_score_num_nn=1,
    featuresampler=patchcore.sampler.IdentitySampler(),
    nn_method=patchcore.common.FaissNN(False, 4),
    **kwargs,
):
    self.backbone = backbone.to(device)
    self.layers_to_extract_from = layers_to_extract_from
    self.input_shape = input_shape

    self.device = device
    self.patch_maker = PatchMaker(patchsize, stride=patchstride)

    self.forward_modules = torch.nn.ModuleDict({})

    feature_aggregator = patchcore.common.NetworkFeatureAggregator(
        self.backbone, self.layers_to_extract_from, self.device
    )
    feature_dimensions = feature_aggregator.feature_dimensions(input_shape)
    self.forward_modules["feature_aggregator"] = feature_aggregator

    preprocessing = patchcore.common.Preprocessing(
        feature_dimensions, pretrain_embed_dimension
    )
    self.forward_modules["preprocessing"] = preprocessing

    self.target_embed_dimension = target_embed_dimension
    preadapt_aggregator = patchcore.common.Aggregator(
        target_dim=target_embed_dimension
```

```
        )

        _ = preadapt_aggregator.to(self.device)

        self.forward_modules["preadapt_aggregator"] = preadapt_aggregator

        self.anomaly_scorer = patchcore.common.NearestNeighbourScorer(
            n_nearest_neighbours=anomaly_score_num_nn, nn_method=nn_method
        )

        self.anomaly_segmentor = patchcore.common.RescaleSegmentor(
            device=self.device, target_size=input_shape[-2:]
        )

        self.featuresampler = featuresampler

    def embed(self, data):
        if isinstance(data, torch.utils.data.DataLoader):
            features = []
            for image in data:
                if isinstance(image, dict):
                    image = image["image"]
                with torch.no_grad():
                    input_image = image.to(torch.float).to(self.device)
                    features.append(self._embed(input_image))
            return features
        return self._embed(data)

    def _embed(self, images, detach=True, provide_patch_shapes=False):
        """Returns feature embeddings for images."""

        def _detach(features):
            if detach:
```

```
        return [x.detach().cpu().numpy() for x in features]
    return features

_ = self.forward_modules["feature_aggregator"].eval()
with torch.no_grad():
    features = self.forward_modules["feature_aggregator"](images)

features = [features[layer] for layer in self.layers_to_extract_from]

features = [
    self.patch_maker.patchify(x, return_spatial_info=True) for x in features
]
patch_shapes = [x[1] for x in features]
features = [x[0] for x in features]
ref_num_patches = patch_shapes[0]

for i in range(1, len(features)):
    _features = features[i]
    patch_dims = patch_shapes[i]

    # TODO(pgehler): Add comments
    _features = _features.reshape(
        _features.shape[0], patch_dims[0], patch_dims[1], *_features.shape[2:]
    )
    _features = _features.permute(0, -3, -2, -1, 1, 2)
    perm_base_shape = _features.shape
    _features = _features.reshape(-1, *_features.shape[-2:])
    _features = F.interpolate(
        _features.unsqueeze(1),
        size=(ref_num_patches[0], ref_num_patches[1]),
        mode="bilinear",
        align_corners=False,
    )
```

```
            _features = _features.squeeze(1)
            _features = _features.reshape(
                *perm_base_shape[:-2], ref_num_patches[0], ref_num_patches[1]
            )
            _features = _features.permute(0, -2, -1, 1, 2, 3)
            _features = _features.reshape(len(_features), -1, *_features.shape[-3:])
            features[i] = _features
        features = [x.reshape(-1, *x.shape[-3:]) for x in features]

        # As different feature backbones & patching provide differently
        # sized features, these are brought into the correct form here.
        features = self.forward_modules["preprocessing"](features)
        features = self.forward_modules["preadapt_aggregator"](features)

        if provide_patch_shapes:
            return _detach(features), patch_shapes
        return _detach(features)

    def fit(self, training_data):
        """PatchCore training.

        This function computes the embeddings of the training data and fills the
        memory bank of SPADE.
        """
        self._fill_memory_bank(training_data)

    def _fill_memory_bank(self, input_data):
        """Computes and sets the support features for SPADE."""
        _ = self.forward_modules.eval()

        def _image_to_features(input_image):
            with torch.no_grad():
                input_image = input_image.to(torch.float).to(self.device)
```

```
            return self._embed(input_image)

        features = []
        with tqdm.tqdm(
            input_data, desc="Computing support features...", position=1, leave=False
        ) as data_iterator:
            for image in data_iterator:
                if isinstance(image, dict):
                    image = image["image"]
                features.append(_image_to_features(image))

        features = np.concatenate(features, axis=0)
        features = self.featuresampler.run(features)

        self.anomaly_scorer.fit(detection_features=[features])

    def predict(self, data):
        if isinstance(data, torch.utils.data.DataLoader):
            return self._predict_dataloader(data)
        return self._predict(data)

    def _predict_dataloader(self, dataloader):
        """This function provides anomaly scores/maps for full dataloaders."""
        _ = self.forward_modules.eval()

        scores = []
        masks = []
        labels_gt = []
        masks_gt = []
        with tqdm.tqdm(dataloader, desc="Inferring...", leave=False) as data_iterator:
            for image in data_iterator:
                if isinstance(image, dict):
                    labels_gt.extend(image["is_anomaly"].numpy().tolist())
```

```
                masks_gt.extend(image["mask"].numpy().tolist())
                image = image["image"]
            _scores, _masks = self._predict(image)
            for score, mask in zip(_scores, _masks):
                scores.append(score)
                masks.append(mask)
    return scores, masks, labels_gt, masks_gt

def _predict(self, images):
    """Infer score and mask for a batch of images."""
    images = images.to(torch.float).to(self.device)
    _ = self.forward_modules.eval()

    batchsize = images.shape[0]
    with torch.no_grad():
        features, patch_shapes = self._embed(images, provide_patch_shapes=True)
        features = np.asarray(features)

        patch_scores = image_scores = self.anomaly_scorer.predict([features])[0]
        image_scores = self.patch_maker.unpatch_scores(
            image_scores, batchsize=batchsize
        )
        image_scores = image_scores.reshape(*image_scores.shape[:2], -1)
        image_scores = self.patch_maker.score(image_scores)

        patch_scores = self.patch_maker.unpatch_scores(
            patch_scores, batchsize=batchsize
        )
        scales = patch_shapes[0]
        patch_scores = patch_scores.reshape(batchsize, scales[0], scales[1])

        masks = self.anomaly_segmentor.convert_to_segmentation(patch_scores)
```

```
    return [score for score in image_scores], [mask for mask in masks]

@staticmethod
def _params_file(filepath, prepend=""):
    return os.path.join(filepath, prepend + "patchcore_params.pkl")

def save_to_path(self, save_path: str, prepend: str = "") -> None:
    LOGGER.info("Saving PatchCore data.")
    self.anomaly_scorer.save(
        save_path, save_features_separately=False, prepend=prepend
    )
    patchcore_params = {
        "backbone.name": self.backbone.name,
        "layers_to_extract_from": self.layers_to_extract_from,
        "input_shape": self.input_shape,
        "pretrain_embed_dimension": self.forward_modules[
            "preprocessing"
        ].output_dim,
        "target_embed_dimension": self.forward_modules[
            "preadapt_aggregator"
        ].target_dim,
        "patchsize": self.patch_maker.patchsize,
        "patchstride": self.patch_maker.stride,
        "anomaly_scorer_num_nn": self.anomaly_scorer.n_nearest_neighbours,
    }
    with open(self._params_file(save_path, prepend), "wb") as save_file:
        pickle.dump(patchcore_params, save_file, pickle.HIGHEST_PROTOCOL)

def load_from_path(
    self,
    load_path: str,
    device: torch.device,
    nn_method: patchcore.common.FaissNN(False, 4),
```

```
        prepend: str = "",
    ) -> None:
        LOGGER.info("Loading and initializing PatchCore.")
        with open(self._params_file(load_path, prepend), "rb") as load_file:
            patchcore_params = pickle.load(load_file)
        patchcore_params["backbone"] = patchcore.backbones.load(
            patchcore_params["backbone.name"]
        )
        patchcore_params["backbone"].name = patchcore_params["backbone.name"]
        del patchcore_params["backbone.name"]
        self.load(**patchcore_params, device=device, nn_method=nn_method)

        self.anomaly_scorer.load(load_path, prepend)


# Image handling classes.
class PatchMaker:
    def __init__(self, patchsize, stride=None):
        self.patchsize = patchsize
        self.stride = stride

    def patchify(self, features, return_spatial_info=False):
        """Convert a tensor into a tensor of respective patches.
        Args:
            x: [torch.Tensor, bs x c x w x h]
        Returns:
            x: [torch.Tensor, bs * w//stride * h//stride, c, patchsize,
            patchsize]
        """
        padding = int((self.patchsize - 1) / 2)
        unfolder = torch.nn.Unfold(
            kernel_size=self.patchsize, stride=self.stride, padding=padding, dilation=1
        )
```

```
    unfolded_features = unfolder(features)
    number_of_total_patches = []
    for s in features.shape[-2:]:
        n_patches = (
            s + 2 * padding - 1 * (self.patchsize - 1) - 1
        ) / self.stride + 1
        number_of_total_patches.append(int(n_patches))
    unfolded_features = unfolded_features.reshape(
        *features.shape[:2], self.patchsize, self.patchsize, -1
    )
    unfolded_features = unfolded_features.permute(0, 4, 1, 2, 3)

    if return_spatial_info:
        return unfolded_features, number_of_total_patches
    return unfolded_features

def unpatch_scores(self, x, batchsize):
    return x.reshape(batchsize, -1, *x.shape[1:])

def score(self, x):
    was_numpy = False
    if isinstance(x, np.ndarray):
        was_numpy = True
        x = torch.from_numpy(x)
    while x.ndim > 1:
        x = torch.max(x, dim=-1).values
    if was_numpy:
        return x.numpy()
    return x
```

Додаток Г
(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Автоматизована система виявлення дефектів

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра автоматизації та інтелектуальних інформаційних технологій
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 0.83 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

☑ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Олег БІСІКАЛО, зав. кафедри АІІТ _____ _____
(прізвище, ініціали, посада) (підпис)

Любов ЯНЧУК, секретар кафедри АІІТ _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Костянтин ОВЧИННИКОВ
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Роман МАСЛІЙ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Олег САМОЛЮК
(підпис) (прізвище, ініціали)