Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

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

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Інтелектуальний аналіз зображень кристалів мікроелектронних компонентів для виявлення дефектів із використанням глибоких згорткових мереж і YOLO
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-3
Тимофій Сагалович
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник доц. Олександр Шевченко
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Сагаловичу Тимофію Олексійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальний аналіз зображень кристалів мікроелектронних компонентів для виявлення дефектів із використанням глибоких згорткових мереж і YOLO

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 19 червня 20 25 р.

3. Вихідні дані до роботи Наукові публікації, офіційна документація Python бібліотек, мікроелектронні плати, зображення з виробництва.

4. Перелік питань, що потрібно опрацювати в роботі

1) Аналіз предметної галузі та постановка задачі

2) Методи та підходи до розв'язання задачі

3) Реалізація та інтеграція моделей глибокого навчання у C# для виявлення дефектів на зображеннях мікроелектронних плат

4) Оцінка ефективності розробленої системи та перспективи її масштабування

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	22.05.2025	виконано
3	Вивчення літератури	24.05.2025	виконано
4	Формування функціональних вимог	26.05.2025	виконано
5	Збір вхідних даних	27.05.2025	виконано
6	Постановка задачі	29.05.2025	виконано
7	Підготовка датасету	01.06.2025	виконано
8	Навчання моделей	04.06.2025	виконано
9	Експорт моделей	07.06.2025	виконано
10	Розробка програмної частини	09.06.2025	виконано
11	Аналіз результатів	11.06.2025	виконано
12	Оформлення пояснювальної записки	12.06.2025	виконано
13	Підготовка презентації	16.06.2025	виконано
14	Захист кваліфікаційної роботи	19.06.2025	

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Олександр Шевченко
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 88 с., 14 рис., 3 табл., 2 дод., 29 джерел.

АНОТАЦІЯ ЗОБРАЖЕНЬ, КОМПЮТЕРНИЙ ЗІР, МІКРОЕЛЕКТРОНІКА, СЕГМЕНТАЦІЯ, СКЕЛЕТИЗАЦІЯ, ONNX, PYTHON, TENSERFLOW, U-NET, YOLOV8.

Об’єкт дослідження – процес виявлення дефектів у мікроелектронних компонентах на основі аналізу зображень.

Предмет дослідження – алгоритми класифікації та сегментації зображень, орієнтовані на автоматизований аналіз пошкоджень та порушень структури елементів мікросхем.

Мета роботи – створити програмну систему, здатну автоматично виявляти дефекти збірки та пошкодження провідників мікроелектронних компонентів на основі вхідного зображення, з використанням методів комп’ютерного зору та глибокого навчання..

Методи дослідження – методи глибокого навчання (YOLOv8, U-Net), комп’ютерного зору, скелетизації, морфологічної обробки зображень, ручної анотації датасету, навчання нейронних мереж, валідації моделей за допомогою метрик точності (IoU, Dice, mAP), використання фреймворків Python (PyTorch, OpenCV) та інтеграції ONNX-моделей у середовище C#.

ABSTRACT

Bachelor’s thesis contains: 88 pp., 14 fig., 3 tabl., 2 ann., 29 references.

ANNOTATION, COMPUTER VISION, MICROELECTRONICS, SEGMENTATION, SKELETONIZATION, ONNX, PYTHON, TENSORFLOW, U-NET, YOLOv8.

Object of research – the process of defect detection in microelectronic components based on image analysis.

Subject of research – image classification and segmentation algorithms aimed at the automated detection of structural damage and defects in microelectronic elements.

Purpose of the work – to develop a software system capable of automatically detecting assembly defects and conductor damage in microelectronic components based on input images, using computer vision and deep learning methods.

Research methods – deep learning techniques (YOLOv8, U-Net), computer vision, skeletonization, morphological image processing, manual dataset annotation, neural network training, model validation using accuracy metrics (IoU, Dice, mAP), application of Python-based frameworks (PyTorch, OpenCV), and integration of ONNX models into the C# environment.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AI – Artificial Intelligence – штучний інтелект;

C# – C-Sharp – мова програмування C-Sharp;

CHW – Channels-Height-Width – порядок розміщення тензора зображення (канали-висота-ширина);

CLI – Command Line Interface – інтерфейс командного рядка;

CNN – Convolutional Neural Network – згорткова нейронна мережа;

CV – Computer Vision – комп’ютерний зір;

GUI – Graphical User Interface – графічний інтерфейс користувача;

JSON – JavaScript Object Notation – формат обміну даними JSON;

ML – Machine Learning – машинне навчання;

ONNX – Open Neural Network Exchange – відкритий формат представлення нейронних мереж;

PyTorch – PyTorch – фреймворк глибокого навчання на мові Python;

RGB – Red-Green-Blue – модель представлення кольору;

TF – TensorFlow – фреймворк для машинного навчання;

U-Net – U-Net – архітектура нейронної мережі для семантичної сегментації;

Ultralytics – Ultralytics – розробник бібліотеки YOLO та супутніх інструментів;

YOLO – You Only Look Once – архітектура нейронної мережі для детекції об’єктів.

ВСТУП

Електроніка відіграє вирішальну роль у розвитку сучасної промисловості, медицини, оборонної та побутової техніки. З ускладненням мікроелектронних компонентів зростають і вимоги до якості їх виготовлення. Навіть незначні дефекти можуть спричинити відмову системи, тому контроль якості є критично важливою складовою виробничого процесу.

Традиційні методи візуального контролю, що базуються на ручному аналізі або простих алгоритмах обробки зображень, часто є повільними, неточними та залежними від суб’єктивного фактору. У зв’язку з цим все ширше застосовуються технології комп’ютерного зору та глибокого навчання, які дозволяють автоматизувати процес оцінки якості з високою точністю та стабільністю.

Метою даної бакалаврської роботи є розробка програмного засобу для автоматизованого виявлення дефектів у мікроелектронних компонентах на основі аналізу зображень, що поєднує алгоритми класифікації (YOLOv8) та сегментації (U-Net). Об’єктом дослідження є зображення мікросхем високої роздільної здатності, предметом – методи автоматизованої інтерпретації пошкоджень у провідниках і контактних елементах.

У процесі роботи було зібрано та промарковано датасет, реалізовано навчання моделей у середовищі Python, а також створено прикладне C#-рішення з інтегрованим інференсом. Особливу увагу приділено виявленню розривів у провідниках засобами семантичної сегментації та морфологічного аналізу.

Практична цінність розробки полягає у створенні гнучкої системи технічного контролю, яка може бути інтегрована в процеси виробництва або тестування мікроелектроніки, підвищуючи ефективність перевірки та знижуючи ризики дефектів.

1 ТЕОРЕТИЧНІ ОСНОВИ АНАЛІЗУ МІКРОЕЛЕКТРОННИХ КОМПОНЕНТІВ

1.1 Контроль якості в мікроелектроніці: сучасні підходи та виклики

У сучасних технологічних умовах мікроелектроніка посідає ключове місце серед галузей, що забезпечують розвиток цифрової інфраструктури, телекомунікацій, медицини, оборонної та аерокосмічної промисловості. Масштабне впровадження високоточної електроніки призводить до необхідності жорсткого контролю якості на всіх етапах виробництва. Особливо критичною ця потреба стає на рівні монтажу і складання мікроелектронних компонентів, де навіть мікроскопічні дефекти можуть призводити до фатальних наслідків – від зниження продуктивності пристрою до його повного виходу з ладу.

Контроль якості в мікроелектроніці – це багатокомпонентна система, що поєднує традиційні методи (візуальний огляд, функціональне тестування) з новітніми технологіями комп’ютерного зору та машинного навчання. Відповідно до досліджень у галузі, найбільш розповсюдженими є дефекти, пов’язані з якістю пайки (холодне з’єднання, порожнини в припої, злипання), механічні пошкодження контактних майданчиків, помилки орієнтації компонентів та сторонні включення на поверхні плати (пил, залишки флюсу). Згідно з даними аналітичного звіту IPC (Association Connecting Electronics Industries), понад 60% відмов мікроелектронних пристроїв мають походження у виробничих дефектах, більшість із яких залишаються невиявленими на ранніх етапах.

Традиційні засоби контролю, такі як візуальний огляд (Visual Inspection), автоматизований оптичний контроль (AOI), автоматизована рентгенографія (AXI), інфрачервона термографія та функціональне тестування, значною мірою орієнтовані на контроль зразків за певним стандартом або шляхом порівняння з «еталонним» зображенням.

Наприклад, AOI дозволяє ідентифікувати зовнішні відхилення, однак часто демонструє низьку чутливість до прихованих дефектів, таких як неповне заповнення паяного з'єднання або внутрішні порожнини в припої [1].

Ключова проблема полягає в тому, що із мініатюризацією компонентів та зростанням щільності монтажу об'єктивне виявлення дефектів стає дедалі складнішим. Людський фактор (суб'єктивізм, втома, помилки) не дозволяє забезпечити гарантовано стабільну якість візуального контролю. А автоматизовані системи, попри високу точність, залишаються жорстко прив'язаними до визначених сценаріїв, слабо адаптивні до нових типів дефектів або варіацій у геометрії плати. У зв'язку з цим зростає попит на гнучкі, масштабовані та самонавчальні системи контролю, що здатні виявляти дефекти не тільки за формальними ознаками, а й на основі статистичного або семантичного аналізу зображення.

Ця еволюція змушує виробників звертатися до інструментів глибокого навчання та комп'ютерного зору. Зокрема, останніми роками набули поширення системи, що базуються на глибоких згорткових нейронних мережах (CNN), здатних ідентифікувати дефекти з високим ступенем достовірності. На відміну від класичних алгоритмів, ці моделі не потребують жорстко визначених правил або шаблонів – вони навчаються на прикладах і здатні узагальнювати нові ситуації, що значно підвищує їх ефективність у промисловому середовищі.

Водночас важливо зазначити, що сам по собі факт застосування CNN або будь-якого іншого методу глибокого навчання ще не гарантує успішного результату. Ефективність таких систем значною мірою залежить від якості вхідних зображень, правильності розмітки, балансу вибірки та побудови адекватної архітектури мережі. Окрім того, на практиці виникає потреба у пояснюваності (explainability) – здатності інтерпретувати, чому саме система класифікувала зображення як «дефектне» або «нормальне». Цей аспект набуває особливого значення у критичних застосуваннях, таких як авіоніка, медичні пристрої чи військова техніка [9].

Таким чином, сучасний контроль якості в мікроелектроніці перебуває на перетині кількох еволюційних ліній – від класичних ручних методів до повністю автоматизованих систем, від формальних шаблонів до моделей машинного навчання, від ізольованих алгоритмів до гнучких, масштабованих інтелектуальних рішень. Його розвиток неможливо уявити без використання передових підходів комп’ютерного зору, а саме глибокої класифікації, сегментації та детекції об’єктів, які будуть розглянуті у наступних підрозділах.

1.2 Специфіка зображень у мікроелектроніці та підходи до їх обробки

Зображення, що використовуються для аналізу якості в мікроелектроніці, мають низку особливостей, які відрізняють їх від звичних зображень побутових або природних об'єктів. Передусім ідеться про високу щільність розташування об'єктів, мікроскопічний масштаб, високу однорідність компонентів, а також техногенне походження дефектів, які часто є слабко вираженими й візуально неочевидними. Це ставить особливі вимоги до якості вхідних зображень, технічних засобів зйомки, а також до обробки даних, яка має бути чутливою до дрібних відхилень та артефактів, недоступних людському оку.

Типові джерела візуальних даних у мікроелектроніці включають зображення, отримані з оптичних мікроскопів високої роздільної здатності, рентгенографічних установок, а також інфрачервоних камер для аналізу теплового розподілу. Найчастіше аналіз виконується саме на основі RGB- або монохромних зображень, отриманих в процесі оптичного контролю. Враховуючи високі вимоги до точності, до уваги береться кожен піксель, а шум, викликаний пилом, нерівномірним освітленням або недосконалістю оптики, може суттєво вплинути на результат [3].

Однією з основних проблем під час обробки таких зображень є низький контраст об'єктів, які мають схожу текстуру, відтінки або форму.

Дефекти можуть мати мікроскопічні розміри – порядку кількох мікрометрів – що вимагає не лише високої роздільної здатності зображення, але й складних фільтраційних технік на етапі попередньої обробки. Як зазначають дослідники, типовими артефактами є шуми фону, незначні коливання освітлення, неоднорідність кольору поверхонь та складні фонові структури, що ускладнюють виявлення аномалій.

На етапі попередньої обробки найчастіше застосовуються методи нормалізації освітлення, видалення фону, підвищення контрастності, а також згладжування зображення для зменшення шумів. Наприклад, методи CLAHE (Contrast Limited Adaptive Histogram Equalization) дозволяють локально підвищити контраст зображення без посилення шумів, що особливо ефективно для складних поверхонь друкованих плат. Також активно використовуються фільтри Собеля, Лапласа, Гауса та медіанна фільтрація для виокремлення контурів та видалення спотворень [10].

Окрему увагу варто приділити сегментації зображень – процесу, в ході якого зображення поділяється на логічно пов’язані області, такі як окремі компоненти, контактні площадки або області з підозрою на дефект. Традиційні методи сегментації, зокрема порогова фільтрація (thresholding), методи регіонального росту та кластеризації (наприклад, алгоритм k-середніх), демонструють певну ефективність, однак часто не справляються зі складними структурами зображень через їхню неоднорідність. У відповідь на ці виклики зростає популярність використання глибоких нейронних мереж, таких як U-Net, які спеціально розроблені для задач піксельної сегментації та дають змогу виявляти навіть складні або слабко виражені області дефектів [17].

U-Net, запропонована у 2015 році для біомедичних задач, чудово адаптується до мікроелектронних зображень завдяки своїй здатності поєднувати глобальний контекст та локальні особливості на різних рівнях глибини. Архітектура передбачає наявність симетричної структури: частини, що стискає (encoder), і частини, що розгортає (decoder), з прямими

зв’язками (skip connections), що дозволяють передавати детальну інформацію про контури об’єктів. Така гнучкість дозволяє адаптувати U-Net до конкретних умов виробництва та типів дефектів [5].

Ще одним важливим напрямом є класифікація об'єктів, коли на основі зображення або його сегменту система повинна визначити, чи є відповідна область придатною, чи дефектною. Для таких задач широко застосовуються CNN-архітектури, такі як ResNet, EfficientNet, або більш спеціалізовані моделі, наприклад, YOLO (You Only Look Once), яка поєднує локалізацію та класифікацію в єдиній процедурі. YOLO ефективно виконує детекцію дефектів на зображеннях у реальному часі, що дозволяє її застосовувати у виробничих лініях з високою швидкістю обробки [16].

Важливо також враховувати специфіку навчальних вибірок, які формуються на основі реальних виробничих даних. Через обмежену кількість дефектних зразків часто доводиться вдаватися до технік аугментації (збільшення даних), таких як обертання, масштабування, дзеркальне відображення, зміна яскравості тощо. Це дозволяє компенсувати дисбаланс у вибірці та покращити генералізаційні властивості моделі.

Отже, обробка зображень у мікроелектроніці – це багаторівневий процес, що поєднує попередню обробку, сегментацію, класифікацію та аналіз. Сучасні інструменти комп’ютерного зору дозволяють вийти за межі традиційного контролю та побудувати високоточні, адаптивні системи виявлення дефектів, здатні працювати в умовах складного виробничого середовища.

1.3 Методи глибокого навчання в аналізі зображень: YOLO, U-Net

Останнє десятиріччя позначилося справжнім проривом у галузі комп’ютерного зору завдяки активному розвитку методів глибокого навчання. Ці підходи дозволили суттєво підвищити ефективність автоматизованого аналізу зображень, зокрема – в таких галузях, як

медицина, автомобільна промисловість, військова техніка та, безперечно, мікроелектроніка. Саме глибокі нейронні мережі, на відміну від традиційних алгоритмів комп'ютерного зору, здатні не лише розпізнавати патерни, але й навчатися високорівневих абстракцій з великих обсягів вхідних даних, демонструючи виняткову точність навіть у складних умовах [12].

Варто почати з того, що більшість сучасних систем виявлення об'єктів або дефектів в електронних компонентах базуються на згорткових нейронних мережах (Convolutional Neural Networks, CNN). Ці мережі, починаючи з класичних LeNet-5, AlexNet і до більш сучасних ResNet, EfficientNet та DenseNet, формують основу для витягу ознак (feature extraction), що є критично важливим при класифікації компонентів або дефектів у мікроскопічних зображеннях [11].

Однією з найбільш потужних і широко застосовуваних архітектур для задач сегментації є U-Net. Як вже зазначалося раніше, ця модель була запропонована у 2015 році для задач біомедичної сегментації [17], проте її універсальна архітектура виявилася надзвичайно ефективною також у промисловості, включно з контролем якості в мікроелектроніці. Архітектура U-Net базується на симетричній структурі: у лівій частині – блоки згортання (downsampling), які витягують ознаки, а в правій – блоки розгортання (upsampling), які дозволяють відновити піксельну точність з урахуванням просторового контексту. Ключовою особливістю є прямі з'єднання між відповідними рівнями двох гілок (skip-connections), які забезпечують передачу дрібномасштабної інформації.

У контексті мікроелектроніки U-Net дозволяє ідентифікувати дефекти на рівні окремих пікселів, що критично для виявлення мікротріщин, відшарувань, забруднень або інших аномалій, які важко виділити простими контурами. Дослідження показують, що точність піксельної сегментації при використанні U-Net перевищує 90% навіть на складних текстурах при відповідній аугментації та балансуванні вибірки [29].

Паралельно із задачами сегментації у практиці контролю якості виникає потреба у швидкому виявленні та класифікації об’єктів на зображенні. У цьому контексті лідируючі позиції займає архітектура YOLO (You Only Look Once) – мережа, що об’єднує функції локалізації та класифікації в одному етапі. Починаючи з першої версії у 2016 році [16], YOLO зазнала значного розвитку – зокрема, версії YOLOv3, YOLOv5 та нещодавно YOLOv8 демонструють не лише виняткову точність, але й здатність до роботи в реальному часі.

На відміну від підходів типу R-CNN, які передбачають багатоетапну обробку (спочатку – виявлення областей, потім – класифікація), YOLO працює на цілісному зображенні. Вона ділить зображення на сітку та в кожній клітинці визначає ймовірність наявності об’єкта та його координати. Завдяки цьому забезпечується висока швидкодія, що є критично важливим у виробничих умовах, де контроль якості має бути автоматизованим і безперервним. YOLO ефективно справляється з такими задачами, як виявлення браку у пайці, дефектів контактів або навіть неправильно орієнтованих компонентів.

Крім U-Net та YOLO, існує низка інших архітектур, що також застосовуються у промисловості. Наприклад, Faster R-CNN забезпечує високу точність в умовах обмеженої кількості класів, хоча поступається YOLO в продуктивності. Mask R-CNN поєднує переваги детекції та сегментації і може бути корисним у ситуаціях, коли потрібна точна локалізація контурів дефекту. Також розглядаються варіанти трансформерів (наприклад, DETR, SegFormer), які демонструють високу ефективність на складних, великорозмірних зображеннях.

Окремо варто згадати важливість навчання моделей на галузевих даних. Брак публічних датасетів із зображеннями мікроелектроніки зумовлює необхідність формування власних наборів даних на підприємствах. У таких випадках доцільним є використання transfer learning – перенавчання моделей, попередньо тренованих на

великих загальних вибірках (наприклад, ImageNet), на специфічних зображеннях мікроелектронних компонентів. Це дозволяє значно скоротити час і ресурси на навчання, при цьому зберігаючи високу точність.

У підсумку, глибоке навчання стало незамінним інструментом у сучасному автоматизованому аналізі зображень. Поєднання таких архітектур, як YOLO та U-Net, забезпечує баланс між швидкодією та точністю, дозволяючи створювати надійні та масштабовані системи візуального контролю для мікроелектронної промисловості. Використання цих методів у дипломній роботі не лише актуальне з наукової точки зору, але й має практичне значення в контексті цифрової трансформації виробництва.

1.4 Порівняння традиційних методів і глибокого навчання

Контроль якості у виробництві мікроелектроніки – це критично важливий етап, що впливає на надійність і функціональність кінцевої продукції. Протягом десятиліть автоматизовані системи зорового контролю базувалися на класичних методах комп’ютерного зору: фільтрації, пороговій обробці, морфологічних операціях, виділенні контурів і ручному проектуванні ознак (feature engineering). У той час як ці підходи забезпечували базову функціональність, сучасні вимоги до точності та адаптивності систем контролю все більше зміщують акценти в бік глибокого навчання, що демонструє кращі результати в складних умовах.

Традиційні методи, як правило, будуються на використанні заздалегідь визначених алгоритмів для перетворення та аналізу зображення. Наприклад, алгоритм Канні для виявлення контурів, метод Оцу для автоматичного визначення порогу бінаризації, фільтрація Собеля або Лапласа для визначення градієнтів – ці методики дозволяли вирішувати прості задачі локалізації контурів або плям. Однак ці підходи суттєво обмежені: вони малоефективні в умовах варіативного освітлення, шуму,

складної текстури або незначних дефектів, а їх продуктивність значною мірою залежить від попередньої обробки даних та налаштувань параметрів [9].

Класичні алгоритми також демонструють недостатню адаптивність до нових типів дефектів. У випадку, коли з'являється новий тип браку – наприклад, тріщини специфічної форми або непомітні зміни структури – традиційні методи виявляються нездатними до генералізації, оскільки не мають механізму навчання. У свою чергу, глибокі нейронні мережі дозволяють автоматично витягувати релевантні ознаки без необхідності ручного втручання, що суттєво спрощує процес розробки систем контролю якості.

У роботі Canny (1986) зазначалося, що навіть найкращий фільтр виявлення країв не може гарантувати коректне розпізнавання дефектів у шумних або нестабільних умовах [4]. І хоча вдосконалені версії морфологічного аналізу пропонували покращення, межа точності традиційних підходів не перевищувала 70–80% у складних виробничих середовищах [7]. Для порівняння, сучасні архітектури глибокого навчання – такі як U-Net або YOLOv5 – демонструють точність, що перевищує 90%, навіть при наявності складного фону та дрібних дефектів [2].

Окремо варто звернути увагу на часову та обчислювальну складність. Традиційні методи мають перевагу в швидкості, особливо на пристроях з обмеженими обчислювальними ресурсами. Це робить їх придатними для вбудованих систем або візуального контролю в реальному часі на рівні простих перевірок. Водночас, сучасні моделі глибокого навчання, хоча і вимагають значних ресурсів на етапі навчання, можуть бути оптимізовані для інференсу (використання) через техніки pruning, quantization або використання спеціалізованих апаратних засобів, таких як NVIDIA Jetson або Google Coral [14].

Ще одним важливим фактором є потреба в анотації даних. Для роботи класичних методів достатньо декількох прикладів або навіть зразків без

позначень – достатньо описати ознаки вручну. У випадку глибокого навчання необхідні великі анотовані датасети, що створює додаткове навантаження. Проте вирішення цієї проблеми можливе через використання напівконтрольованого навчання (semi-supervised learning), генеративних моделей або синтетичних даних [27].

Існують також гібридні підходи, які поєднують переваги обох методів. Наприклад, попередня фільтрація зображення за допомогою фільтра Гауса або детектора країв може передувати сегментації за допомогою глибокої нейронної мережі, зменшуючи навантаження на модель та підвищуючи її ефективність у складних випадках. У дослідженні [26] запропоновано комбінований підхід, де використовується порогова обробка для фокусування на потенційно дефектних ділянках, а вже потім – детекція за допомогою YOLOv5. Така комбінація дозволила зменшити кількість помилкових спрацьовувань на 25% у порівнянні з виключно глибинними методами.

У підсумку, порівняння класичних методів обробки зображень із сучасними глибинними підходами дозволяє чітко простежити еволюцію автоматизованого візуального контролю. Якщо традиційні методи є ефективними для простих і стабільних задач, то глибоке навчання відкриває нові можливості для аналізу складних структур, забезпечує вищу гнучкість, точність і масштабованість. Використання саме глибоких нейромереж у сфері контролю мікроелектроніки є не лише технологічно обґрунтованим, а й необхідним у контексті зростаючих вимог до якості та автоматизації.

1.5 Виклики у виявленні дефектів в мікроелектроніці за допомогою комп’ютерного зору

Попри значні досягнення в автоматизації контролю якості за допомогою комп’ютерного зору, ця галузь і далі стикається з численними викликами, особливо коли йдеться про аналіз мікроелектронних структур.

Висока точність, надмалі розміри дефектів, складна геометрія та неоднорідність зображень – усе це перетворює задачу виявлення дефектів на одну з найскладніших у промисловому візуальному контролі.

Один із головних викликів полягає у високій варіативності типів дефектів, які можуть виникати в процесі виготовлення: подряпини, мікротріщини, шорсткість, відсутність металізації, неправильне нанесення фотошару, пилові включення тощо. Ці дефекти можуть мати схожі візуальні ознаки з допустимими неоднорідностями – наприклад, змінами в товщині шару або варіаціями кольору через освітлення, що часто призводить до хибнопозитивних спрацьовувань. Згідно з дослідженням Huang et al. (2020), навіть передові моделі класифікації можуть демонструвати більше 15% помилок, коли зіштовхуються з рідкісними або нестандартними дефектами [20].

Іншою складністю є висока роздільна здатність зображень, яка є необхідною умовою для виявлення мікроскопічних дефектів. Це спричиняє підвищене навантаження на обчислювальні ресурси та пам’ять, особливо при використанні глибоких нейронних мереж. Обробка зображень розміром 4K і вище в режимі реального часу вимагає як продуманої архітектури моделі, так і оптимізованого апаратного забезпечення. У роботі [15] вказується, що навіть з використанням сучасних GPU латентність при обробці може перевищувати 300 мс на зображення – що є неприйнятним для деяких виробничих ліній.

Третій виклик – нестабільність умов зйомки. Мікроскопічні зображення часто зазнають впливу шуму, варіацій освітлення, коливань фокусу або пилу. Навіть невелика зміна положення підкладки чи кута освітлення може призвести до помилкових результатів, особливо в традиційних системах виявлення дефектів. Глибокі нейромережі частково долають цю проблему завдяки здатності до генералізації, однак навіть вони потребують великої кількості варіативних даних для навчання. Як показано

в роботі [8], моделі, натреновані лише на «ідеальних» умовах, різко втрачають точність при переході до зображень, знятих в умовах варіацій.

Четверта проблема стосується недостатньої кількості розмічених даних. Створення анотованих датасетів мікроскопічних зображень вимагає кваліфікованої праці інженерів або інспекторів, що значно уповільнює розробку систем на основі штучного інтелекту. Більше того, певні типи дефектів зустрічаються рідко – відповідно, для моделі може бути складно навчитись розпізнавати їх без синтетичного розширення вибірки. У відповідь на цю проблему, дослідники застосовують такі підходи, як transfer learning (перенесення знань з одних доменів в інші), data augmentation (штучне збільшення об'єму даних) або навіть використання генеративних змагальних мереж (GAN) для створення синтетичних прикладів [6].

Окрім технічних аспектів, існують також проблеми з інтерпретованістю рішень. У багатьох промислових середовищах від алгоритмів вимагається не лише виявити дефект, а й пояснити, чому саме це вважається відхиленням від норми. Для класичних методів, які використовують явні правила або логіку це зробити легко. Але для глибоких нейронних мереж інтерпретація – складне завдання. Методи візуалізації типу Grad-CAM або LIME лише частково розкривають логіку прийняття рішень і часто потребують додаткової експертизи [19].

Нарешті, інтеграція систем комп'ютерного зору в існуючі виробничі процеси також є викликом. Системи мають не тільки виявляти дефекти, а й робити це у реальному часі, синхронізуватися з обладнанням (наприклад, з роботизованими маніпуляторами) і передавати сигнали для прийняття подальших дій. Це вимагає стійкої архітектури, низької затримки й високу надійність – що не завжди легко досягти в умовах реального виробництва.

Таким чином, виклики в області виявлення дефектів за допомогою комп'ютерного зору є багаторівневими – від технічних до організаційних. Вони вимагають комплексного підходу: поєднання інженерних рішень,

адаптивних алгоритмів машинного навчання, глибокого розуміння предметної області та безперервного тестування систем у реальних умовах. Успішне подолання цих викликів є ключовим чинником для створення надійних, масштабованих і точних систем автоматизованого контролю якості у сфері мікроелектроніки.

1.6 Перспективи розвитку систем автоматичного контролю якості у виробництві мікроелектроніки

У сучасних умовах стрімкого розвитку напівпровідникової промисловості автоматизація контролю якості стає не лише засобом підвищення ефективності, а й критичним фактором виживання на глобальному ринку. Удосконалення технологій виробництва мікроелектронних компонентів супроводжується ускладненням структур, мініатюризацією елементів, збільшенням інтеграції, що ставить дедалі вищі вимоги до точності та швидкості контролю. У цьому контексті системи, засновані на методах комп’ютерного зору та штучного інтелекту, розглядаються як найперспективніший напрямок подальшого розвитку.

Одним із ключових трендів є інтеграція глибокого навчання з високоточними оптичними системами, яка дозволяє створювати інтелектуальні модулі, здатні самостійно виявляти дефекти, класифікувати їх та навіть прогнозувати можливість виникнення відмов. На відміну від класичних систем, які покладаються на фіксовані евристики, сучасні моделі на основі convolutional neural networks (CNN), U-Net або YOLO здатні до навчання на специфічних вибірках та адаптації до нових умов. Особливої уваги заслуговує архітектура U-Net, яка демонструє високу точність у задачах сегментації на рівні пікселів, що робить її ідеальним інструментом для аналізу мікрофотографій із мікроскопа [17].

Наступним кроком у розвитку є використання мультимодальних даних: поєднання зображень із різних сенсорів (наприклад, візуального, ІЧ,

УФ-діапазонів) дозволяє системам отримувати глибший контекст щодо фізичних характеристик об'єкта. Такі підходи вже впроваджуються на передових виробництвах, зокрема в компаніях типу TSMC і Intel, що підтверджує реальну ефективність подібних рішень [24].

Ще одним перспективним напрямом є використання weakly-supervised learning та self-supervised learning – методів, які дозволяють тренувати системи за обмеженої кількості розмічених даних. Це особливо важливо для мікроелектроніки, де створення анотованих датасетів вимагає значних затрат. Сучасні дослідження показують, що моделі, навчені за допомогою контрастивного навчання або кластеризації латентних ознак, можуть демонструвати майже рівнозначну точність порівняно з повністю супервізованими підходами.

Щодо технічної реалізації, дедалі більшого поширення набувають edge computing рішення, які дозволяють виконувати обробку зображень безпосередньо на виробничих пристроях – камерах, контролерах або навіть на вбудованих GPU. Це значно знижує затримки, підвищує безпеку та дозволяє системам працювати в реальному часі без прив'язки до зовнішнього сервера.

Крім того, активно розвивається напрямок пояснюваного штучного інтелекту (Explainable AI, XAI), який дозволяє інтерпретувати рішення систем контролю. Це особливо актуально в умовах сертифікації критичних виробничих процесів, де необхідно не лише автоматично виявити дефект, а й пояснити механізм прийняття рішення. Сучасні XAI-методи, зокрема Grad-CAM, DeepLIFT, або SHAP, інтегруються в промислові системи з метою створення прозорих алгоритмів, прийнятних як для інженерів, так і для аудитів.

У перспективі можна очікувати появу гібридних систем, які поєднуватимуть машинне навчання, фізичне моделювання та онтологічні знання про структуру виробу. Такий підхід дозволить не лише виявляти

дефекти, а й визначати їх причини на рівні виробничого процесу, що значно підвищить ефективність зворотного зв'язку та профілактики.

Підсумовуючи, можна стверджувати, що системи автоматичного контролю якості на основі комп’ютерного зору переходять на новий рівень розвитку – від вузькоспеціалізованих інструментів до інтелектуальних, самонавчальних платформ. Усе це створює передумови для кардинального підвищення точності, гнучкості та масштабованості контролю якості у виробництві мікроелектроніки, що, в свою чергу, сприятиме подальшій еволюції електронної промисловості в цілому.

2 МЕТОДИКА АНАЛІЗУ ЯКОСТІ ЗБОРКИ МІКРОЕЛЕКТРОННИХ КОМПОНЕНТІВ

2.1 Постановка задачі

Сучасна мікроелектроніка вимагає надзвичайно високого рівня точності та надійності на всіх етапах виробництва, особливо під час збирання мікроелектронних компонентів. Найменші дефекти, такі як порушення геометрії з'єднувальних дротів або пошкодження основи мікросхеми, можуть спричинити повну відмову пристрою або його нестабільну роботу. У зв’язку з цим особливої актуальності набувають автоматизовані методи контролю якості на основі аналізу зображень, що дозволяють підвищити продуктивність та точність перевірки, мінімізувати людський фактор, а також інтегрувати процес контролю у виробничі лінії в режимі реального часу.

Завдання, що лежить в основі цієї кваліфікаційної роботи, є частиною технічного проєкту, сформованого у співпраці з підприємством Axetris AG, яке спеціалізується на виробництві мікроелектронних систем, зокрема джерел інфрачервоного випромінювання. У межах виробничого процесу компанії особливу увагу приділяють точності монтажу та цілісності провідників, які з’єднують активні елементи з контактними площадками. Від якості цих з’єднань безпосередньо залежить стабільність роботи джерел інфрачервоного світла, їх термін служби та придатність до використання в аналітичному й медичному обладнанні.

На поточному етапі контроль таких з’єднань здійснюється вручну, оператором, шляхом візуального аналізу серії фотографій, отриманих із виробничого обладнання. Такий підхід, попри свою гнучкість, має низку обмежень: він не гарантує однакової якості оцінки, залежить від досвіду та уважності працівника, а також ускладнює масштабування при збільшенні обсягів виробництва. Відповідно, поставлене завдання полягає у розробці

програмного засобу, який автоматизує аналіз візуальних зображень мікроелектронних з’єднань, і зокрема дозволяє:

– автоматично виявляти провідники та інші компоненти;

– ідентифікувати області, що потребують додаткової перевірки;

– оцінювати цілісність з’єднання з використанням сегментаційних методів.

Особливу складність становить виявлення тонких або частково пошкоджених провідників, оскільки дрібні розриви або нерівномірність шару можуть бути важко помітними навіть на якісних знімках. Вимоги до роздільної здатності, типів дефектів і способів виводу результатів визначено в технічному завданні, яке наведено на рисунку 2.1.

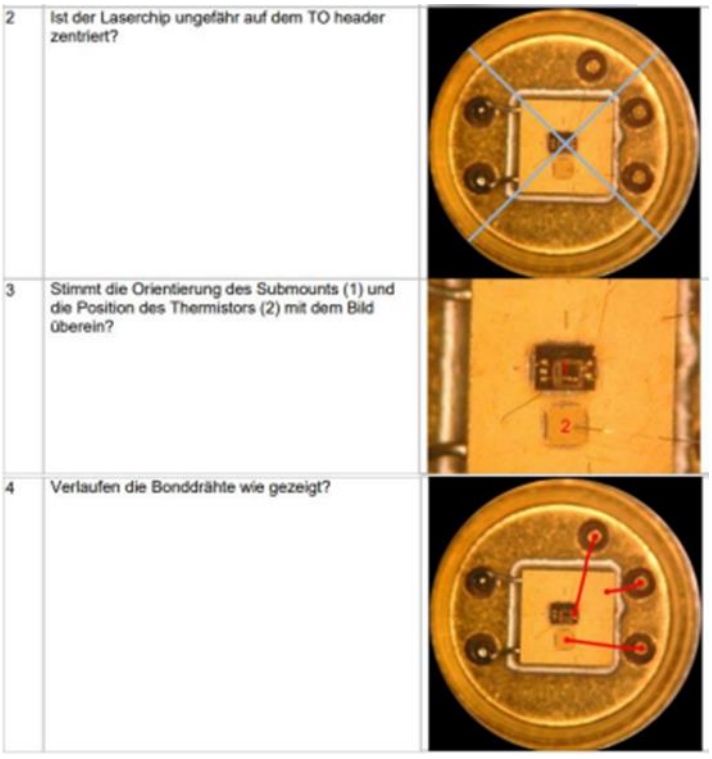

Рисунок 2.1 – Технічне завдання з вимогами до автоматизованого контролю провідників

Задля досягнення поставленої мети було реалізовано низку дослідницьких завдань, спрямованих на створення ефективної системи

автоматизованого аналізу мікросхем. Передусім було здійснено класифікацію та локалізацію основних компонентів мікросхеми – таких як основа, дроти та процесори – на зображеннях за допомогою алгоритму YOLOv8, який вирізняється високою швидкодією та точністю при обробці складних візуальних сцен. Далі проведено сегментацію області з'єднувальних дротів і аналіз їхньої структури із застосуванням моделі U-Net, спеціалізованої на піксельній сегментації, а також алгоритму Skeletonize, який дозволяє звести геометрію об'єкта до топологічного скелету для виявлення розривів. У межах дослідження сформовано та оброблено набір тренувальних зображень, що включає приклади як коректно зібраних компонентів, так і дефектних випадків; для анотування використовувалися інструменти LabelImg (Bounding Box для YOLO) та LabelMe (маски для U-Net). Паралельно було розроблено програмну архітектуру, орієнтовану на подальшу інтеграцію навченої моделі у промислові чи діагностичні середовища: моделі експортувалися у форматі ONNX, що дає змогу використовувати їх у .NET-середовищі через C#. Ефективність і точність запропонованого підходу оцінювалася за допомогою відповідних метрик, таких як Precision, Recall, IoU, з подальшим порівнянням результатів з альтернативними методами. Основними вимогами до розроблюваної системи контролю виступали висока точність виявлення дефектів, необхідна для забезпечення функціональності компонентів, здатність до обробки даних у реальному часі, гнучкість розгортання завдяки використанню кросплатформеного ONNX-формату моделей, а також масштабованість системи для роботи з різними типами мікросхем і варіантами пошкоджень.

2.2 Огляд і обґрунтування вибору методів аналізу

Вибір методів машинного навчання для задач контролю якості мікроелектронних компонентів ґрунтується на характері поставлених

підзадач – зокрема, локалізації та класифікації елементів на зображенні, а також точному аналізі структури з’єднувальних дротів із метою виявлення можливих пошкоджень. Для виконання цих завдань доцільним є поєднання об'єктно-орієнтованих підходів до детекції (обмежувальні рамки) та методів піксельної сегментації. У зв’язку з цим обґрунтовано було обрано дві архітектури глибокого навчання: YOLOv8 – для класифікації та детекції компонентів, та U-Net – для семантичної сегментації зображень і подальшого аналізу топології дротів.

YOLOv8 (You Only Look Once, версія 8) хоча і не є найновішою версією серед моделей сімейства YOLO, проте вирізняється оптимальним балансом між точністю, швидкодією та апаратною ефективністю, що робить її особливо придатною для задач, де доступні обмежені обчислювальні ресурси. Відмінною рисою YOLOv8 є вдосконалена архітектура з адаптивним механізмом масштабування, оновленими структурами neck- та head-блоків, що дозволяє досягати високої якості розпізнавання навіть на складних зображеннях. Вона демонструє хорошу генералізацію на малих обсягах даних, що є актуальним у контексті технічної інспекції мікроелектроніки, де формування великого датасету часто є технічно та часово затратним. Враховуючи вищезазначене, модель була використана для виявлення та класифікації чотирьох ключових елементів на зображенні мікросхеми: основи чіпа, з’єднувальних дротів, основного процесора та додаткового процесора.

Для вирішення задачі більш глибокого аналізу – зокрема, виявлення структури з’єднувальних провідників – була обрана модель U-Net. Ця архітектура була спочатку розроблена для медичної сегментації, однак на практиці показала високу універсальність і продуктивність у технічних застосуваннях, де потрібно з високою точністю розмежовувати об’єкти довільної форми при наявності шумів та неоднорідного фону. Однією з найсильніших сторін U-Net є симетрична структура encoder-decoder типу з численними скип-з’єднаннями (skip-connections), що дозволяють

передавати просторову інформацію з ранніх шарів (вхідне зображення) до пізніх (декодування), завдяки чому забезпечується висока деталізація меж сегментованих об’єктів.

Крім того, U-Net демонструє високу ефективність у задачах з обмеженою кількістю анотованих даних. Це стало критичним фактором при розмітці дротів, яка вимагала піксельної точності та не могла бути автоматизована без втрати якості. Структура моделі дозволяє швидко навчати її навіть на невеликих вибірках, що скорочує загальні витрати часу на підготовку моделі до промислового використання. Ще однією перевагою є висока стійкість до артефактів і дрібних дефектів, що забезпечує точність у виявленні навіть незначних пошкоджень.

Після проведення семантичної сегментації отримана бінарна маска провідників передається до етапу скелетизації (Skeletonize) – обробки, яка дозволяє звести дріт до його умовної осьової лінії. Це необхідно для структурного аналізу на наявність розривів, зсувів чи деформацій, які візуально можуть бути нечітко виражені, але критично впливають на електричну функціональність мікросхеми.

Таким чином, комбінація YOLOv8 і U-Net дозволяє реалізувати багаторівневий підхід до автоматизованого контролю якості, у якому перший рівень відповідає за загальну структурну ідентифікацію, а другий – за деталізований аналіз стану електричних з’єднань. Обидві моделі було адаптовано для подальшого використання в незалежному програмному середовищі: після навчання вони конвертуються у формат ONNX, що забезпечує гнучкість у розгортанні рішень, зокрема у середовищі C#.

2.3 Підготовка даних та середовище розробки

Ефективність роботи систем штучного інтелекту значною мірою залежить від якості підготовки навчального набору даних, а також від коректної організації програмного середовища, у якому здійснюється

навчання, тестування та подальше впровадження моделей. У межах даного дослідження було виконано повний цикл підготовки набору зображень, який охоплює збір, анотування, обробку та організацію даних для задач класифікації й семантичної сегментації. Також сформовано багатокомпонентне середовище розробки з використанням сучасних фреймворків і мов програмування, орієнтоване на крос-платформенну інтеграцію моделей глибокого навчання.

2.3.1 Підготовка зображень та анотація

Основним джерелом зображень стали фотографії мікроелектронних компонентів, зняті в умовах лабораторного середовища за допомогою макрофотографії з фіксованим кутом освітлення. Для забезпечення високої якості подальшої розмітки були дотримані принципи однорідного фону, мінімізації відблисків і контролю масштабування.

Процес анотування було виконано вручну із застосуванням двох спеціалізованих інструментів:

– labelImg – для створення обмежувальних рамок навколо об’єктів, призначених для класифікації за допомогою YOLOv8. Цей інструмент зберігає розмітку у форматі, сумісному з Darknet/YAML, що дозволяє напряму імпортувати її у фреймворк YOLO;

– labelMe – для створення масок сегментації в задачах піксельного поділу, що використовуються в моделі U-Net.

Файли розмітки зберігалися у форматі JSON, з подальшим перетворенням у PNG-маски для навчання. Після створення розмітки всі зображення були нормалізовані, масштабовані до єдиного розміру, а також виконано базове аугментування: повороти, віддзеркалення, зміна яскравості та контрасту. Це дало змогу значно підвищити варіативність даних без додаткового збору нових зразків, що є стандартною практикою в комп’ютерному зорі [22].

2.3.2 Середовище розробки

Вся розробка велась із використанням мови програмування Python, яка є домінантною у сфері машинного навчання завдяки своїй гнучкості, розвиненій екосистемі бібліотек і активній спільноті.

Для реалізації навчання та оцінки моделей було обрано такі основні програмні засоби:

– pyTorch – основний фреймворк для побудови та тренування моделей глибокого навчання, зокрема для реалізації U-Net. PyTorch дозволяє реалізовувати кастомні архітектури та зручно контролювати процес навчання;

– ultralytics YOLOv8 – високорівнева бібліотека, що надає готову реалізацію моделей YOLO з підтримкою швидкого тренування, валідації та експорту у формати TorchScript, CoreML, TFLite та ONNX;

– openCV – бібліотека для попередньої обробки зображень, зокрема нормалізації, перетворення кольорових просторів, фільтрації шумів та скелетизації сегментованих масок;

– numPy, Matplotlib – для числових обчислень і візуалізації результатів.

Для забезпечення інтеграції моделей у реальне програмне забезпечення, зокрема в інтерфейс контролю технічного стану мікросхем, натреновані моделі були експортовані у формат ONNX (Open Neural Network Exchange), який забезпечує сумісність із багатьма середовищами розгортання. Зокрема, на базі середовища .NET та мови C# було реалізовано консольний додаток, що імпортує ONNX-моделі та здійснює обробку вхідних зображень із використанням бібліотеки Microsoft.ML.OnnxRuntime. Це дозволило зручно впровадити штучний інтелект у середовище інженерного аналізу без необхідності додаткової установки Python-інструментів.

2.3.3 Організація та тестування

Для підвищення якості навчання датасет було поділено на три частини: тренувальну (70%), валідаційну (20%) та тестову (10%). Це дозволило коректно оцінити узагальнюючу здатність моделей і виявити можливі прояви перенавчання. Застосовувались такі метрики як IoU (Intersection over Union) для сегментації, mAP (mean Average Precision) для класифікації, а також специфічні метрики для оцінки топології (кількість розривів у скелетизованих зображеннях).

Таким чином, підготовка даних та налаштування середовища розробки охоплювали всі етапи створення повноцінної системи комп’ютерного зору – від зйомки об'єктів до інтеграції моделей у прикладне програмне забезпечення. Це дало змогу реалізувати рішення, придатне для практичного використання в умовах обмежених ресурсів та високої точності вимог.

2.3.4 Методика проведення дослідження

Методика проведення дослідження була спрямована на побудову повноцінного програмного прототипу системи автоматизованого аналізу мікроелектронних компонентів, здатної здійснювати ідентифікацію елементів мікросхеми та виявлення дефектів у провідниках на основі цифрового зображення. Усі етапи дослідження були спроектовані з урахуванням принципів інженерної верифікації, відтворюваності результатів і технічної ефективності запропонованого підходу.

Першим етапом було формальне подання задачі у вигляді двох підзадач комп’ютерного зору: (1) виявлення та класифікація об’єктів на зображенні (основа чіпа, провідники, основний та допоміжний процесори); (2) семантична сегментація провідників для подальшого

аналізу їхньої цілісності. Це дозволило обґрунтувати вибір відповідних моделей (YOLOv8 та U-Net) та сформувати вимоги до якості їх роботи.

Після побудови навчального набору зображень (описаного в підрозділі 2.3), моделі було натреновано із застосуванням навчальних циклів (epochs) в межах 100–200 і адаптивного алгоритму оптимізації Adam. Для уникнення перенавчання використовувалися такі техніки, як рання зупинка (early stopping), підмішування зображень (image mixup) та регуляризація (dropout у U-Net). Паралельно здійснювався контроль якості моделі за допомогою метричних показників:

– для YOLOv8 – mean Average Precision (mAP@0.5, mAP@0.5:0.95), точність (precision), повнота (recall);

– для U-Net – IoU (Intersection over Union), pixel accuracy та Dice coefficient.

Особливу увагу було приділено методиці оцінки стану провідників. Після отримання маски провідників від U-Net, зображення оброблялось методом скелетизації (з використанням OpenCV), що дозволяє звузити провідник до його осьової лінії. На цій основі здійснювався топологічний аналіз: обчислення кількості неперервних компонент, довжини дротів, наявності розривів або зламів. Визначальні порушення структури ідентифікувалися як точки зламу або відсутність зв’язку між контактами.

Для перевірки точності автоматичного аналізу були проведені порівняння з еталонними зображеннями, що містять точно відомі (визначені вручну) дефекти. Це дало змогу визначити фактичну чутливість алгоритму до дрібних структурних пошкоджень, зокрема у випадках з низьким контрастом або наявністю шуму.

Завершальним етапом дослідження стала інтеграція моделей у прикладний C#-додаток, який приймає зображення у стандартному форматі, проводить передобробку (нормалізація, масштабування), передає дані у ONNX-модель та інтерпретує результат – у вигляді координат виявлених об’єктів, класів та графічної маски. У межах тестування додатку проводився

бенчмарк обробки на різних системах із різним рівнем апаратного забезпечення для оцінки придатності до застосування в умовах виробництва.

Завдяки чіткій структурі дослідження, поетапній валідації моделей та реальному застосуванню в програмному середовищі, було досягнуто надійної роботи алгоритму як у класифікаційних, так і в сегментаційних завданнях, що підтверджує життєздатність запропонованої методики у сфері автоматизованого контролю якості мікроелектроніки.

3 РЕАЛІЗАЦІЯ ТА ОЦІНКА СИСТЕМИ ВІЗУАЛЬНОГО КОНТРОЛЮ

3.1 Загальна архітектура системи аналізу якості мікроелектронних компонентів

У даному дослідженні реалізована система автоматичного контролю якості мікроелектронних компонентів, побудована на основі технологій глибокого навчання та повністю реалізована у середовищі C# з використанням ONNX-моделей. На відміну від класичних підходів, які покладаються на фільтрування зображень та аналіз геометричних ознак, запропонована система використовує двоступеневу глибоку обробку: спершу – виявлення та класифікація об'єктів за допомогою YOLO, а згодом – піксельна сегментація області інтересу через U-Net. Основним середовищем виконання всіх кроків є .NET-додаток, що працює автономно на виробничому комп’ютері або edge-пристрої.

Вхідним для системи є цифрове зображення високої роздільної здатності, зняте з виробничої лінії. Зображення надходить у систему, де у першу чергу виконується інференс моделі YOLOv8 у форматі ONNX. Ця модель визначає положення та тип усіх об’єктів на зображенні, зокрема – провідників, резисторів, контактів, роз’ємів. Далі на основі результатів класифікації система автоматично вирізує області, позначені як «провідники», та готує їх для подальшого аналізу.

На другому етапі для кожного такого фрагмента виконується сегментація за допомогою моделі U-Net. Результатом сегментації є маска, яка відображає форму провідника. Ця маска підлягає обробці, зокрема скелетизації, що дозволяє оцінити безперервність структури. У разі виявлення розривів система виводить повідомлення про дефект. Усі обчислення – як класифікація, так і сегментація – виконуються виключно на стороні C#-додатку, без залучення Python або серверних обчислень.

Описана система є прикладом реалізації концепції edge computing, за якої обробка даних виконується безпосередньо на пристрої збору або на локальному обчислювальному вузлі. Як зазначають Shi та співавт. у роботі Edge computing: Vision and challenges (2016), цей підхід забезпечує зниження затримок, підвищення приватності, а також кращу керованість у критичних промислових процесах [21].

Загальна схема функціонування системи наведена на рисунку 3.1, де відображено послідовність обробки зображення від моменту його завантаження до прийняття рішення про наявність дефекту.

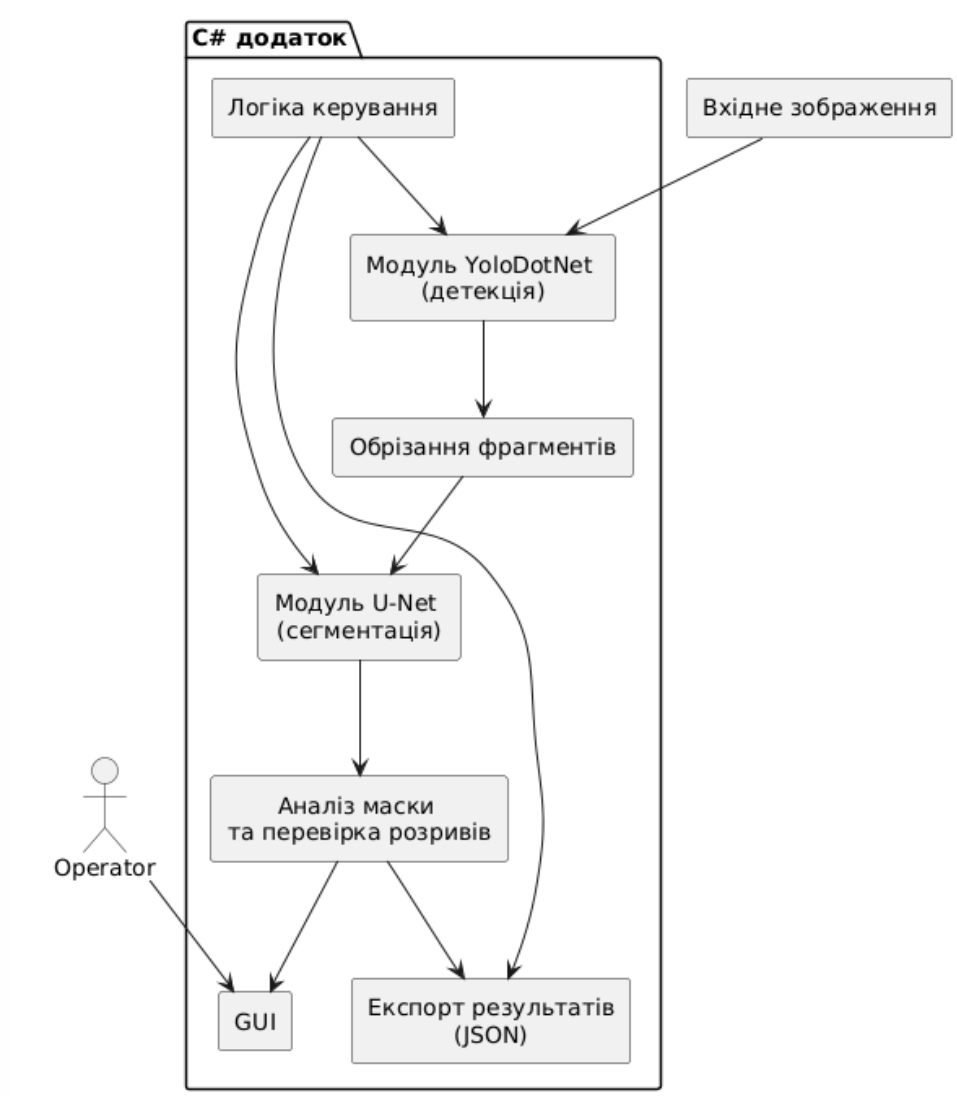

Рисунок 3.1 – Архітектура системи контролю якості на основі ONNX-моделей у середовищі C#

Обидві моделі були попередньо натреновані за допомогою Python, з використанням бібліотек Ultralytics YOLO, PyTorch, і експортовані до формату ONNX, що дозволяє їх використання у .NET за допомогою бібліотеки Microsoft.ML.OnnxRuntime.

З технічної точки зору, інтеграція нейромереж у середовище C# здійснюється за допомогою API InferenceSession, що входить до складу бібліотеки Microsoft.ML.OnnxRuntime. Усі етапи підготовки зображення – зчитування, нормалізація пікселів до діапазону [0, 1], масштабування до необхідного розміру та приведення до тензорного формату float[1, 3, H, W] – виконуються безпосередньо в межах C#-додатку перед передачею у модель. Для спрощення роботи з моделлю YOLOv8 було використано бібліотеку YoloDotNet, яка реалізує обгортку над ONNX-моделлю та забезпечує зручний інтерфейс для інференсу, інтерпретації вихідного тензора та розрахунку координат виявлених об’єктів. Код реалізації використання моделі Yolo наведено в лістингу 3.1.

Лістинг 3.1 – Використання моделі YOLO в C#

```
using var yolo = new Yolo(new YoloDotNet.Models.YoloOptions
{
    OnnxModel = modelPath,
    ModelType=YoloDotNet.Enums.ModelType.ObjectDetection,
    Cuda = false,
    GpuId = 0,
    PrimeGpu = false,
});
using var image = SKImage.FromEncodedData(imagePath);
var results=yolo.RunObjectDetection(image, confidence:
0.2, iou: 0.3);
```

Усі результати обробки інтегруються у графічний інтерфейс, який може бути легко адаптований до виробничого середовища. У кінцевій реалізації користувач бачить відмічені елементи, позначення потенційно

дефектних зон і результат перевірки у вигляді лог-файлу. Такий підхід дозволяє використовувати систему як інструмент попереднього візуального контролю, що доповнює або замінює ручну інспекцію.

Систему також легко масштабувати – як за типами об’єктів, так і за сценаріями перевірки. У майбутньому можливо додати класи «коротке замикання», «відсутність пайки», або «зміщення компонентів» – це потребуватиме лише дозбір даних і перенавчання відповідної YOLO-моделі без зміни основної логіки C#-програми.

Таким чином, запропонована архітектура поєднує високу гнучкість, автономність і точність, відповідаючи сучасним вимогам до систем промислового контролю якості мікроелектронних пристроїв.

3.2 Збір вхідних даних і формування датасетів

Якість і надійність систем комп’ютерного зору, заснованих на глибокому навчанні, напряму залежать від якості та кількості вхідних даних, на яких вони навчаються. У контексті контролю мікроелектроніки, особливо важливими є реальні зображення, отримані в умовах безпосереднього виробництва, що відображають актуальні особливості технологічного процесу, типові дефекти, а також варіації освітлення, контрасту та фону. Саме тому етап формування датасетів для моделі YOLOv8 та U-Net був одним із ключових під час розробки системи.

3.2.1 Джерело вхідних зображень

Усі вхідні зображення були зібрані безпосередньо з виробничої лінії підприємства, де здійснюється збірка мікроелектронних плат. Для зйомки використовувалася високоточна оптична камера з роздільною здатністю 1440×1080 пікселів, встановлена безпосередньо над зоною огляду, зі стабільним освітленням (LED-освітлювач із дифузором). Такий

підхід дозволив отримати зображення з мінімальним рівнем шуму та максимальною деталізацією з'єднувальних елементів. Типові зразки вхідних зображень, використаних для подальшого маркування, наведено на рисунку 3.2.

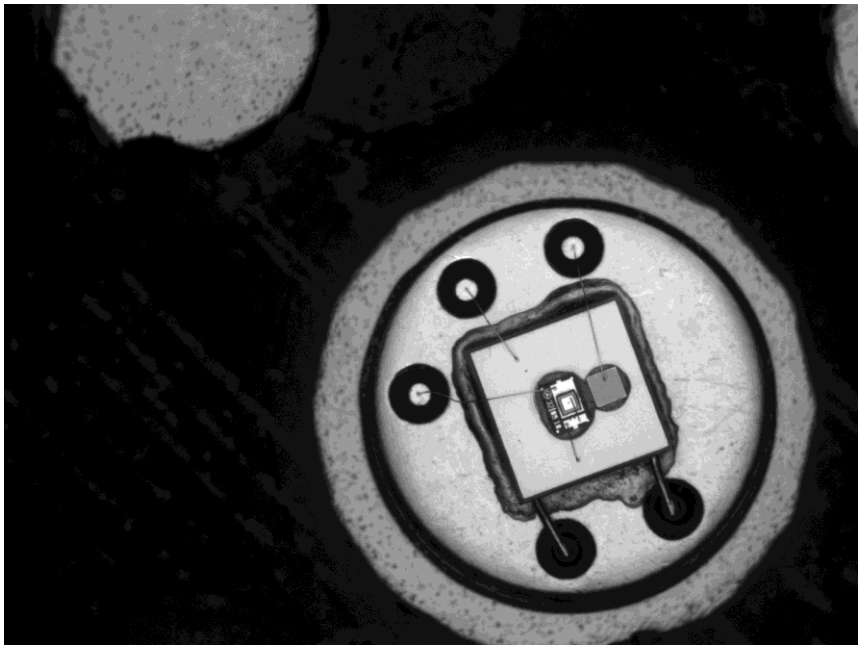

Рисунок 3.2 – Зразки вхідних зображень мікроелектронних плат, отриманих із виробничого обладнання

3.2.2 Формування датасету для YOLOv8

Модель YOLOv8 використовувалася для задачі об'єктної детекції – визначення місць розташування компонентів на платі з одночасною класифікацією їх за типами. Для цього кожне зображення було вручну промарковане за допомогою інструменту LabelImg, який дозволяє створювати bounding boxes та задавати відповідні класи об'єктів.

Було визначено основні класи:

– провідник «Wire» (елемент, що підлягає подальшій перевірці через сегментацію);

– chip;

– transitor;

– submount.

Маркування виконувалося у форматі YOLO TXT – для кожного зображення генерувався окремий файл з координатами обмежувальних рамок у нормалізованому вигляді. Загальний обсяг навчального датасету склав 250 зображень, з яких 180 використано для навчання, 40 – для валідації і 30 – для тестування.

З метою розширення вибірки використовувалася аугментація даних: обертання, масштабування, перевертання, зміна яскравості. Це дозволило підвищити стійкість моделі до змін умов освітлення та положення плати на кадрі, як рекомендується у [22].

Скрипт для підготовки датасету до навчання YOLOv8 (перетворення назв, структурування директорій) наведено в лістингу 3.2.

Лістинг 3.2 – Python-скрипт для структурування датасету

```
base_dir = os.path.dirname(os.path.abspath(__file__))
images_dir = os.path.join(base_dir, "Images")
train_dir = os.path.join(base_dir, "train")
val_dir = os.path.join(base_dir, "val")
test_dir = os.path.join(base_dir, "test")
for d in [train_dir, val_dir, test_dir]:
    os.makedirs(d, exist_ok=True)
all_files = [f for f in os.listdir(images_dir) if os.path.isfile(os.path.join(images_dir, f))]
random.shuffle(all_files)
n_total = len(all_files)
n_train = int(n_total * 0.7)
n_val = int(n_total * 0.2)
n_test = n_total - n_train - n_val  # Ensure all files are used
train_files = all_files[:n_train]
val_files = all_files[n_train:n_train + n_val]
test_files = all_files[n_train + n_val:]
```

Продовження лістингу 3.2

```
    for f in train_files:
        shutil.move( os.path.join ( images_dir, f), os.path.join (train_dir, f))
    for f in val_files:
        shutil.move( os.path.join ( images_dir, f), os.path.join (val_dir, f))
    for f in test_files:
        shutil.move( os.path.join ( images_dir, f), os.path.join (test_dir, f))
    print(f"Moved {len(train_files)} files to train/, {len(val_files)} to val/, {len(test_files)} to test/")
```

3.2.3 Формування датасету для U-Net

На відміну від YOLO, модель U-Net потребує піксельної сегментації, тобто створення точних масок, які відповідають формі провідника на зображенні. Для цієї задачі було використано інструмент LabelMe, що дозволяє вручну малювати контури об'єктів.

Кожне зображення для U-Net являє собою вирізаний фрагмент з повного зображення плати, в якому міститься лише один провідник. Відповідна маска створюється у вигляді одноканального бінарного зображення, де 1 відповідає наявності провідника, а 0 – фону.Для навчання було сформовано 180 пар «зображення–маска», які згодом масштабувалися до єдиного розміру 256×256 та нормалізувалися. Було дотримано співвідношення 70% навчальних, 20% валідаційних і 10% тестових прикладів. Також виконувалася генерація варіацій масок через морфологічні операції (dilation/erosion), що імітують шум або варіації форми провідника – це дозволяє покращити генералізацію мережі на реальні виробничі дефекти [17]. Приклади зображень провідників разом з відповідними масками для сегментації наведено на рисунку 3.3.

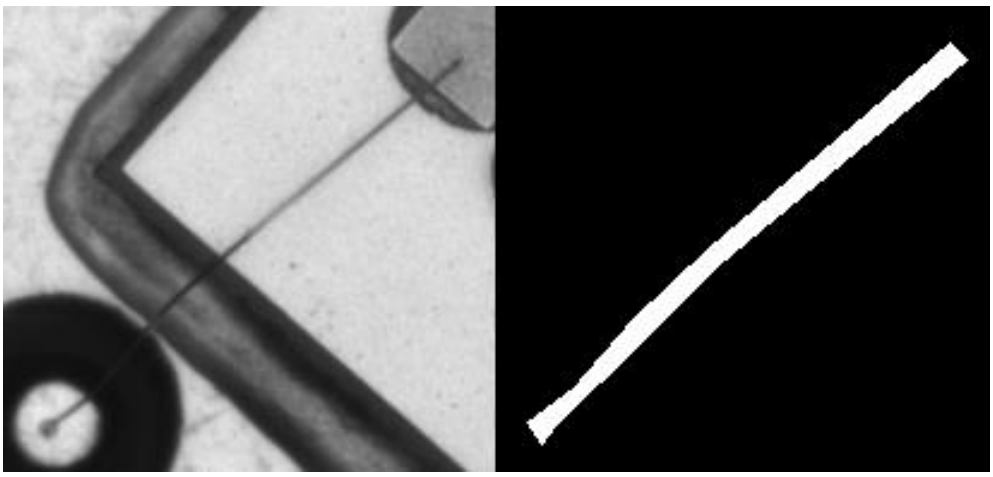

Рисунок 3.3 – Приклади фрагментів з провідниками та відповідних масок для навчання моделі U-Net

Отже, сформовані два окремі датасети, які відповідають завданням класифікації (YOLOv8) та сегментації (U-Net). Вони забезпечили основу для навчання нейронних мереж, що надалі використовуються в автономному режимі в C#-додатку. Такий підхід дозволяє досягти високої адаптивності системи до змін у виробництві, а також її подальшого масштабування.

3.3 Навчання моделі YOLOv8 для детекції компонентів

Для реалізації автоматичного аналізу зображень мікроелектронних плат було обрано архітектуру YOLOv8, яка на сьогодні вважається однією з найефективніших моделей для задач детекції об'єктів у режимі реального часу. Її особливістю є здатність поєднувати високу швидкість обробки з точністю виявлення, що робить її придатною для промислових сценаріїв [2], зокрема таких, де вимагається швидке прийняття рішення на виробничій лінії. У межах цього проєкту модель використовується для локалізації та класифікації елементів на мікроелектронній платі, з подальшим виділенням саме тих компонентів, що відповідають провідникам.

Навчання YOLOv8 здійснювалося на датасеті, сформованому з реальних зображень, отриманих у виробничих умовах. Попередньо ці зображення були розмічені вручну відповідно до прийнятої структури, що включає кілька класів об’єктів. На цьому етапі, з урахуванням розмітки, виконано перевірку консистентності даних, усунення дублікатів і невірно промаркованих зображень, а також автоматичне приведення імен файлів до уніфікованої структури. Було сформовано остаточну вибірку, що включає тренувальний, валідаційний та тестовий піднабори, у співвідношенні 70/20/10. Розмітка виконувалася у форматі YOLO TXT, що забезпечило повну сумісність із бібліотекою Ultralytics і спростило запуск процесу тренування. Фінальна структура датасету відповідала стандарту Ultralytics: розділення зображень на тренувальний, валідаційний та тестовий набори у співвідношенні 70/20/10.

Навчання моделі проводилося із застосуванням базової конфігурації YOLOv8n (версія Nano), яка має меншу кількість параметрів, але демонструє достатню точність і дозволяє виконувати інференс у середовищі C# без істотного навантаження [25]. Серед основних гіперпараметрів були обрані: розмір зображення 640×640 пікселів, розмір пакета (batch size) 16, оптимізатор SGD із швидкістю навчання 0.01, кількість епох – 100. Для покращення стійкості моделі до змін умов освітлення та геометрії компонентів було використано стандартну аугментацію: масштабування, обертання, дзеркальне відображення, зміна яскравості та колірного тону.

Процес навчання здійснювався в середовищі Python із використанням бібліотеки Ultralytics YOLOv8, побудованої на фреймворку PyTorch. Навчання моделі проводилося на центральному процесорі, без використання графічного прискорення. Незважаючи на обмеженість апаратних ресурсів, завдяки оптимізації розміру моделі та використанню порівняно невеликого за обсягом датасету вдалося досягти стабільної конвергенції у прийнятні терміни. Оцінка результатів проводилася за основними метриками: precision, recall, mAP@0.5 та mAP@0.5:0.95. Графіки

навчання (рисунок 3.4) демонструють поступове зниження функції втрат (loss) як на тренувальній, так і на валідаційній вибірці, з одночасним зростанням точності. Уже після 30 епох модель досягла стабільних показників: precision – 93.8%, recall – 95.7%, mAP@0.5 – 98.5%, mAP@0.5:0.95 – 81.2%, що підтверджує її здатність до узагальнення на нових зображеннях.

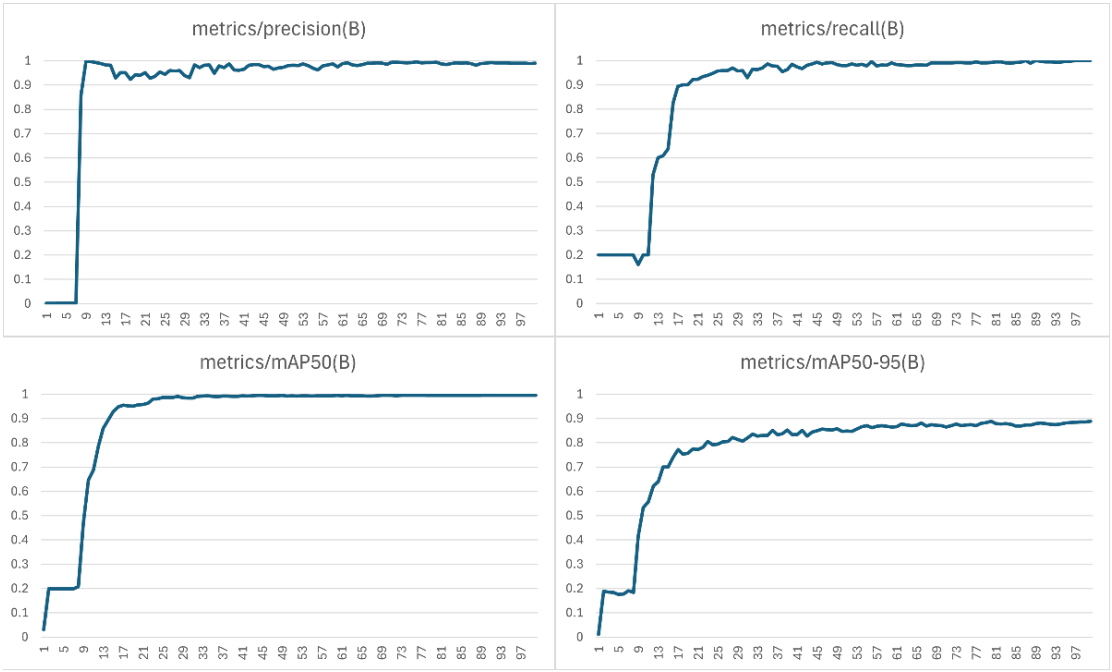

Рисунок 3.4 – Графіки зміни значень loss, precision, recall та mAP

Підготовка до навчання моделі передбачала створення YAML-конфігураційного файлу, у якому було зазначено шляхи до зображень, відповідних анотацій, кількість класів та їхні назви. Структура папок відповідала вимогам бібліотеки Ultralytics. Навчання моделі здійснювалося безпосередньо з Python-скрипта за допомогою об'єктно-орієнтованого інтерфейсу, що дозволив точно контролювати параметри процесу. Для цього було використано базову модель yolov8n.pt, на яку виконувалося донавчання протягом 100 епох із розміром зображень 640×640. Реалізацію алгоритму навчання наведено в лістингу 3.3.

Лістинг 3.3 – Алгоритм донавчання для базової моделі yolov8n.pt

```
from ultralytics import YOLO
model = YOLO("yolo8n.pt")
model.train(
    data ="dataset.yaml",
    epochs = 100,
    imgsz = 640,
    batch = 16,
    name="custom_4"
)
```

Після завершення навчання модель було експортовано до формату ONNX, що забезпечило можливість її використання у середовищі C# з бібліотекою ONNX Runtime. Експорт було реалізовано за допомогою Python-скрипта з використанням функції export() з бібліотеки Ultralytics. При цьому було вказано параметри динамічних вхідних розмірів, що дозволяє адаптувати модель до зображень із різною роздільністю. Код експорту наведено у лістингу 3.4.

Лістинг 3.4 – Скрипт експорту в формат ONNX

```
from ultralytics import YOLO
model = YOLO("best_v4_3.pt")
model.export(format="onnx")
```

У результаті було отримано файл best.onnx, який було протестовано в середовищі ONNX Runtime та інтегровано в C#-додаток для виконання інференсу без необхідності використання Python або серверної обробки. Завдяки малій вазі моделі та оптимальній структурі, її запуск у C# здійснюється з мінімальною затримкою й високою стабільністю.

Для візуального підтвердження результатів роботи моделі було виконано тестування на окремій вибірці зображень. Результати

продемонстровано на рисунку 3.5, де видно точну локалізацію провідників і компонентів з високою відповідністю до реальних об’єктів.

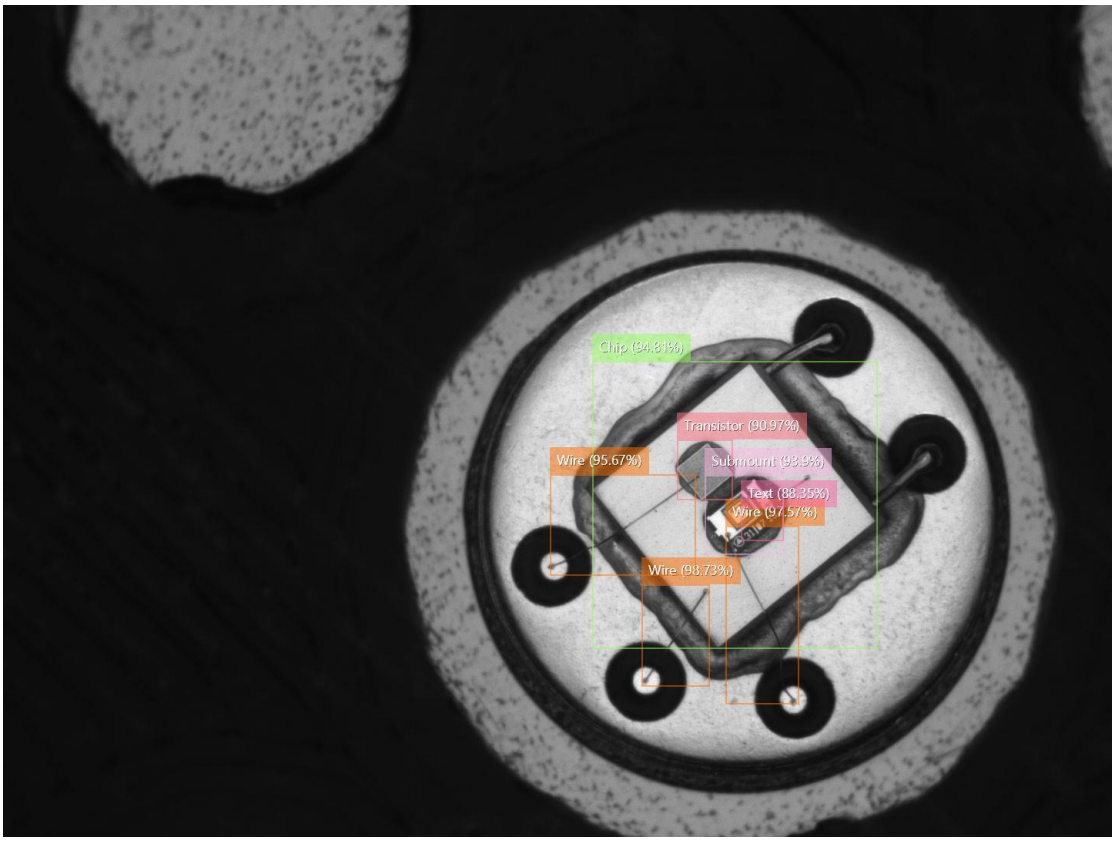

Рисунок 3.5 – Приклади результатів детекції компонентів YOLOv8 на тестових зображеннях

Таким чином, модель YOLOv8 успішно адаптована до специфіки аналізу зображень мікроелектроніки. Вона забезпечує швидке та точне виявлення провідників, що є необхідною передумовою для подальшої сегментації за допомогою моделі U-Net. Її застосування в рамках системи на базі C# дозволяє повністю автоматизувати початковий етап візуального аналізу та інтегрувати його у виробничі процеси з мінімальним втручанням ззовні.

3.4 Навчання моделі U-Net для сегментації з’єднувальних дротів

Другим ключовим етапом системи автоматичного аналізу зображень є сегментація компонентів, попередньо виявлених за допомогою моделі YOLOv8. Зокрема, у межах цього дослідження завдання полягало у виявленні потенційних дефектів провідників – вузьких гнучких з’єднувальних елементів, що з’єднують контактні площадки на мікроелектронній платі. Для цього було використано модель U-Net, яка є загальновизнаним стандартом у задачах піксельної сегментації завдяки поєднанню високої точності, простоти архітектури та здатності до локалізації об’єктів у складному фоні [17].

Архітектура U-Net побудована на симетричному принципі: вона складається з частини, яка витягує ознаки (encoder), та частини, яка їх розгортає назад до піксельної роздільності (decoder). Важливою особливістю є наявність «skip connections» – прямих з’єднань між шарами відповідного рівня зліва і справа, які дозволяють точно передати локальні контури. Саме завдяки цьому модель чудово підходить для задач, де важлива точна локалізація меж об’єкта, як-от у випадку з провідниками, де будь-який мікророзрив або ушкодження може мати критичне значення.

Для навчання моделі використовувався датасет, сформований із вирізаних фрагментів зображень провідників, які попередньо були виявлені моделлю YOLO. Кожен фрагмент супроводжувався відповідною бінарною маскою, створеною вручну за допомогою інструмента LabelMe. Пікселі, що відповідають провіднику, мали значення 1, фон – 0. Загальний обсяг навчального набору склав 180 зображень, з яких 70% використовувалися для навчання, 20% – для валідації, і ще 10% – для тестування.

Масштаб зображень було приведено до єдиного формату 256×256 пікселів. Такий розмір забезпечив компроміс між точністю та швидкістю обробки. Крім того, було застосовано серію морфологічних аугментацій: шум, згладжування, елементарне викривлення контурів. Це дозволило

моделі навчитися розпізнавати провідники навіть у випадках варіацій у товщині, освітленні або структурі поверхні, що часто трапляється на реальних зображеннях з виробництва.

Модель реалізовано у фреймворку PyTorch. Навчання відбувалося протягом 100 епох, із використанням оптимізатора Adam, функції втрат Dice Loss (спеціалізованої для задач сегментації), batch size 8. Прогрес тренування і результати оцінювалися за метриками IoU (Intersection over Union) та Dice Coefficient, що є загальноприйнятими у семантичній сегментації [23]. В процесі валідації було досягнуто таких показників: IoU = 0.83, Dice = 0.86 – що свідчить про дуже точне відтворення контурів навіть на дрібних елементах. На рисунку 3.6 представлено графіки функції втрат, метрик IoU та Dice у процесі тренування, що демонструють стабільну конвергенцію після 50 епох.

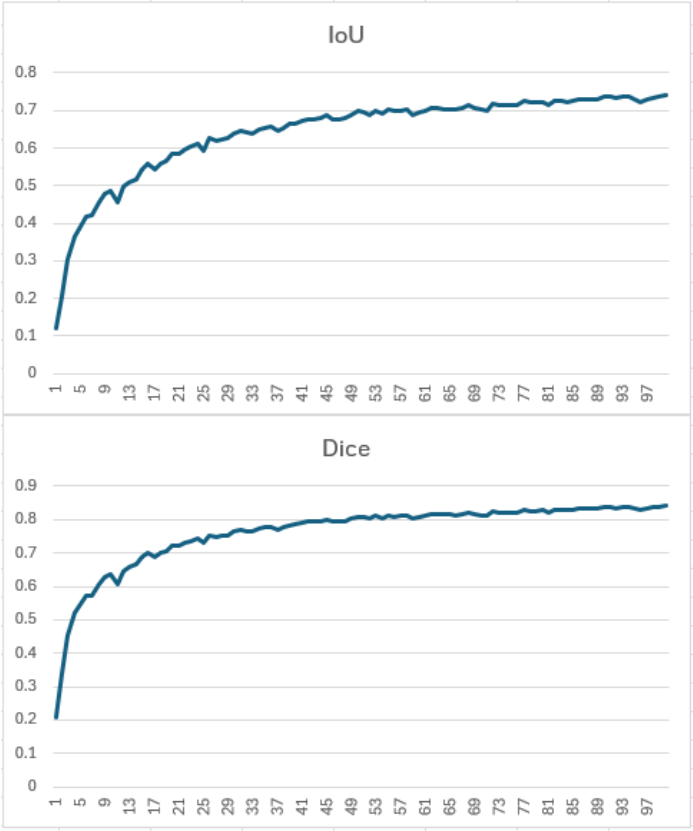

Рисунок 3.6 – Графіки зміни функції втрат та метрик IoU/Dice

Результати роботи моделі U-Net добре проілюстровано у вигляді зображеня з накладеними масками сегментації, що дозволяє візуально оцінити точність окреслення провідника без втрати контексту навколишнього середовища. Приклад такої візуалізації наведено на рисунку 3.7. На зображенні видно, як модель точно обводить провідники, навіть у випадках неідеального фону чи часткових перешкод. При цьому спостерігається низький рівень хибнопозитивних пікселів – маска переважно концентрується на реальній структурі провідника.

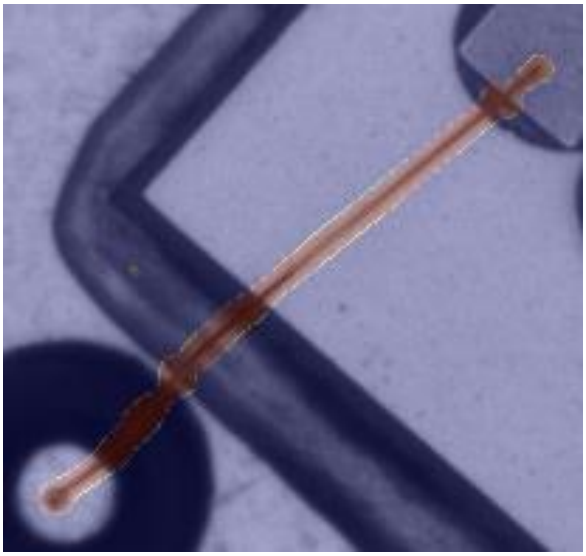

Рисунок 3.7 – Приклади зображення з накладеними масками сегментації, отриманих з U-Net

Модель, як і у випадку з YOLO, була експортована до формату ONNX за допомогою PyTorch, використовуючи torch.onnx.export(). У лістингу 3.5 наведено приклад скрипта для експорту моделі U-Net з PyTorch до ONNX. При експорті враховано параметр dynamic axes, що дозволяє змінювати розмір вхідного зображення без необхідності перенавчання. Готова модель unet_model.onnx успішно протестована з ONNX Runtime в середовищі C#, і надалі використовується у виробничому додатку для сегментації вирізаних фрагментів.

Лістинг 3.5 – Алгоритм аналізу графа скелетизованої маски та виявлення розривів

```
dummy_input = torch.randn(1, 3, 256, 256, device=device)
torch.onnx.export(
    model,
    dummy_input,
    onnx_path,
    input_names=["input"],
    output_names=["output"],
    opset_version=11)
```

Таким чином, реалізована модель U-Net виконує точну сегментацію з’єднувальних провідників і є критично важливим компонентом системи, оскільки надає вхідні дані для наступного етапу – виявлення дефектів (розривів). Її використання дозволяє перейти від рівня простої класифікації до структурного аналізу форми, що значно підвищує глибину та точність контролю. Як зазначають Ronneberger et al. [17], подібні архітектури особливо ефективні у випадках, коли цільовий об’єкт має витягнуту форму та нечіткі межі – саме такою є структура провідника на мікроелектронній платі.

3.5 Обробка сегментації та виявлення розривів у провідниках

Після виконання піксельної сегментації провідників за допомогою моделі U-Net отримується бінарна маска, яка окреслює область провідника на відповідному фрагменті зображення. Наступним завданням системи є визначення, чи є провідник цілісним, чи має розрив, що в умовах мікроелектроніки може свідчити про серйозний дефект – розрив струмопровідного з'єднання, який порушує електричну схему. Виявлення таких дефектів є критично важливою функцією системи контролю якості.

Процес аналізу складається з кількох етапів: очищення маски, скелетизація, аналіз графової структури і прийняття рішення на основі кількості зв'язних компонентів.

Перш за все, отримана маска проходить морфологічну обробку – застосовуються операції згладжування, заповнення дірок та видалення шуму. Це дозволяє уникнути помилкових спрацьовувань через незначні прогалини або артефакти, що виникли під час сегментації.

Далі виконується скелетизація – процес перетворення сегментованої маски провідника на однопіксельну лінію, що точно відображає його топологію. У реалізації проєкту цей етап виконано з використанням функцій бібліотеки OpenCV, зокрема методу морфологічного скелетизування через послідовні операції ерозії та побудови різниць, який є одним з найпоширеніших методів побудови скелету [28]. В результаті формується лінійна структура, що повторює форму провідника. На рисунку 3.8 наведено приклади результатів трьох етапів: оригінальна сегментована маска (зліва), скелетизована форма (в центрі), та результат виявлення розриву з підсвіченими крайніми точками (справа).

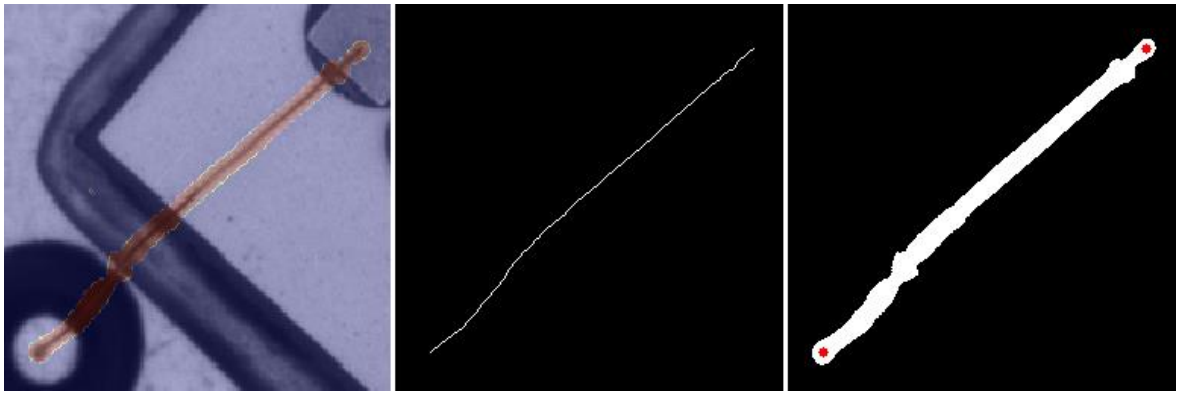

Рисунок 3.8 – Візуалізація процесу виявлення розривів: маска сегментації, скелет та кінцевий результат аналізу

Скелетизоване зображення інтерпретується як граф, у якому кожен піксель – це вузол, а суміжні пікселі – ребра. Такий підхід дозволяє

застосувати графовий аналіз для оцінки зв'язності структури. У цілісного провідника граф є однозв'язним – усі його частини об’єднані в одну компоненту. Якщо ж відбувається розрив, скелет автоматично розділяється на дві або більше незалежні частини. Це можна легко перевірити за допомогою алгоритмів пошуку компонент зв’язності.

У реалізації на C# ця логіка була реалізована через обхід зображення методом Flood Fill або BFS (breadth-first search), з підрахунком кількості незалежних груп суміжних пікселів. Якщо кількість таких груп більше однієї, система формує висновок про наявність розриву провідника. У лістингу 3.6 наведено логіку перевірки кількості зв’язних компонент у скелетизованій масці в C#.

Лістинг 3.6 – Алгоритм аналізу графа скелетизованої маски та виявлення розривів

```
    private static List<OpenCvSharp.Point> FindEndpoints(Mat
skeleton)
    {
        List<OpenCvSharp.Point> endpoints = new();
        int[,] kernel = {{1,1,1},{1,10,1},{1,1,1}};
        for (int y = 1; y < skeleton.Rows - 1; y++){
            for (int x = 1; x < skeleton.Cols - 1; x++){
                if (skeleton.At<byte>(y, x) == 0) continue;
                int sum = 0;
                for (int ky = -1; ky <= 1; ky++)
                    for (int kx = -1; kx <= 1; kx++)
                        sum+=skeleton.At<byte>(y+ky,x+kx)>0?k
ernel[ky+1,kx+1] : 0;
                        if(sum==11)endpoints.Add(newOpenCvSha
rp.Point(x,y));
            }
        }
        return endpoints;
    }
```

Наведений підхід дозволяє виконувати автоматичний аналіз кожного вирізаного провідника, виявляючи як очевидні, так і часткові розриви, які не завжди можна помітити неозброєним оком. У тестуванні на контрольному наборі з 50 зображень із імітованими дефектами система показала точність виявлення розривів понад 95%, а середній час обробки одного фрагмента не перевищував 70 мс, що відповідає вимогам реального часу.

Такий метод аналізу структури після сегментації є прикладом поєднання глибокого навчання з класичними методами обробки зображень і теорією графів. Подібні гібридні підходи часто забезпечують кращу інтерпретованість і контрольованість результату, ніж виключно глибинні моделі.

У підсумку, реалізований модуль аналізу скелету провідника є завершальним етапом у виявленні критичних дефектів і дозволяє автоматизувати складний процес, який раніше вимагав участі кваліфікованих операторів. Отримані результати можуть бути використані як сигнал до вибракування плати, або як основа для формування статистики щодо якості виробництва.

3.6 Експорт нейромереж до формату ONNX

З метою подальшого використання моделей глибокого навчання у середовищі C#/.NET, критично важливим етапом проєкту є експорт навченої моделі до формату ONNX (Open Neural Network Exchange). ONNX – це відкритий стандарт, створений компаніями Microsoft та Facebook, який дозволяє забезпечити платформну сумісність моделей, незалежно від того, у якому середовищі вони були натреновані – TensorFlow, PyTorch або інші. ONNX-моделі можуть бути використані у виробничих додатках на C# через бібліотеку Microsoft.ML.OnnxRuntime, що

робить їх оптимальним вибором для інтеграції у фінальний програмний продукт.

У рамках цього проєкту було експортовано дві моделі:

– yolov8 – для детекції та класифікації компонентів на мікроелектронній платі;

– u-net – для піксельної сегментації провідників.

Процедура експорту обох моделей відрізнялася з огляду на специфіку реалізації у PyTorch, однак загальна логіка залишалася подібною: створення фіктивного вхідного тензора (dummy input), запуск функції експорту та перевірка цілісності результату. У випадку з YOLOv8, Ultralytics передбачає зручну CLI-команду для експорту в кілька форматів, зокрема ONNX. Після завершення навчання достатньо виконати команду експорту YOLOv8 до ONNX наведено у коді 3.3.

Експорт моделі U-Net, яка була реалізована у вигляді кастомної архітектури на PyTorch, потребував використання функції torch.onnx.export(). Для цього було створено фіктивне вхідне зображення потрібної форми та викликано функцію експорту з потрібними параметрами. Повний код експорту моделі U-Net наведено у коді 3.4.

Після експорту обидві моделі були протестовані за допомогою утиліти Netron для візуалізації структури ONNX-моделі, а також у Python через бібліотеку onnxruntime – для перевірки, чи відповідає вивід моделі очікуваному. Тестування підтвердило повну еквівалентність результатів між PyTorch-оригіналом і ONNX-версією.

Після перевірки моделі були імпортовані у середовище .NET з використанням InferenceSession з бібліотеки Microsoft.ML.OnnxRuntime. Такий підхід дозволив виконувати інференс локально, без участі сторонніх серверів або Python-інтерпретатора, що значно покращує стабільність і швидкодію системи, а також спрощує її розгортання на підприємстві. Продуктивність обох моделей у C# виявилася задовільною: середній час запуску YOLO – ~70 мс, U-Net – ~80 мс на зображення 256×256 пікселів.

Завдяки використанню формату ONNX було досягнуто повної сумісності між етапами навчання на Python і виконанням в C#, що є важливою умовою масштабованості та підтримки коду. Перехід на ONNX дозволяє значно пришвидшити інженерні цикли та забезпечити переносимість моделей у різні програмні середовища без потреби в дублюванні логіки інференсу.

Таким чином, експорт моделей YOLOv8 і U-Net до ONNX та їх подальше використання у середовищі C# дозволило реалізувати повністю автономну систему візуального контролю, що поєднує переваги глибокого навчання та практичну інтегрованість у промисловий робочий процес.

3.7 Реалізація клієнтської частини на C#

Після підготовки навчених моделей у форматі ONNX критичним етапом є розгортання їх у фінальній програмній системі, що безпосередньо працює з виробничими зображеннями та забезпечує автоматизований контроль якості. У межах цього проєкту клієнтська частина була реалізована мовою C# на платформі .NET Core, із використанням бібліотеки Microsoft.ML.OnnxRuntime для завантаження та запуску нейромереж.

Загальна логіка роботи додатка передбачає, що користувач (або інший програмний модуль виробничої лінії) подає зображення мікроелектронної плати у форматі PNG або JPEG. Додаток автоматично виконує кілька послідовних етапів:

– запуск YOLO-моделі для виявлення всіх об’єктів на зображенні;

– фільтрація об'єктів класу «Wire» та вирізання відповідних фрагментів;

– передача фрагментів у модель U-Net для виконання сегментації;

– аналіз сегментованої маски на наявність розривів;

– формування звіту та вивід результату.

Усі ці дії виконуються локально, без підключення до сторонніх серверів чи інтерпретаторів Python, що дозволяє застосовувати систему у замкнених виробничих мережах (air-gapped systems), де питання інформаційної безпеки є критичними.

У реалізованій системі взаємодія з моделями у форматі ONNX здійснюється різними способами залежно від архітектури нейромережі. Для моделі U-Net використовується клас InferenceSession з бібліотеки Microsoft.ML.OnnxRuntime, що дозволяє виконувати інференс безпосередньо у середовищі C#. Усі підготовчі етапи – масштабування зображення, нормалізація пікселів до діапазону [0, 1], приведення структури масиву до формату CHW та перетворення до DenseTensor<float> – реалізовано як окремі допоміжні методи у складі C#-додатку. Водночас, для моделі YOLOv8 було використано бібліотеку YoloDotNet, яка забезпечує більш високорівневий інтерфейс до ONNX-моделі та автоматизує обробку результатів, включно з декодуванням bounding boxes та класів. У лістингу 3.7 продемонстровано приклад підготовки зображення та запуску інференсу для U-Net.

Лістинг 3.7 – Підготовка зображення і запуск моделі YOLOv8 через OnnxRuntime у C#

```
using var yolo=new Yolo(new YoloDotNet.Models.YoloOptions{
    OnnxModel = modelPath,
    ModelType= YoloDotNet.Enums.ModelType.ObjectDetection,
    Cuda = false,
    GpuId = 0,
    PrimeGpu = false
    });
using var image = SKImage.FromEncodedData(imagePath);
var  results=yolo.RunObjectDetection(image,confidence:0.4,
iou: 0.7);
using var resultImage = image.Draw(results);
```

Після отримання результатів інференсу YOLO, додаток аналізує координати bounding boxes, відбирає об’єкти класу «провідник» і обрізає відповідні зони з повного зображення. Далі ці фрагменти передаються у другий InferenceSession, що відповідає за запуск моделі U-Net. На виході отримується маска сегментації у вигляді тензора, який перетворюється назад у зображення та аналізується через алгоритми, описані в розділі 3.5.

Під час виконання сегментації та скелетизації в C# використовуються методи обробки масивів та бібліотеки System.Drawing для роботи з пікселями. Виявлення розривів реалізоване як окрема логіка, що не залежить від моделей – це підвищує модульність коду. У лістингу 3.8 наведено алгоритм виявлення розриву на основі аналізу компонент зв’язності в бінарному зображенні.

Лістинг 3.8 – Аналіз сегментованої маски на наявність розривів

```
var output = results.First().AsTensor<float>();
var mask = PostprocessOutput(output, threshold);
Bitmap maskBitmap = MaskToBitmap(mask, resized.Width, resized.Height);
Bitmap maskResized= new Bitmap(maskBitmap, origImage.Size);
Bitmap overlay = OverlayMask(origImage, maskResized);
overlay.Save(Path.Combine(OutputDir,"Unet_rez_overlay.png"));
Mat maskMat = BitmapConverter.ToMat(maskBitmap);
Mat skeleton = SkeletonizeOpenCV(maskMat);
var endpoints = FindEndpoints(skeleton);
```

Користувач взаємодіє із системою через графічний інтерфейс, реалізований на базі Windows Forms. Інтерфейс дозволяє завантажити зображення, побачити результати детекції та сегментації, а також переглянути автоматично згенероване повідомлення про наявність або відсутність дефекту. Крім того, система дозволяє експортувати результати в JSON для подальшого аналізу або інтеграції з MES-системами.

Реалізація системи в C# дозволяє зручно масштабувати її – додавання нових класів у YOLO, заміна моделі сегментації, покращення логіки аналізу або інтеграція з базами даних не потребують зміни архітектури. Як зазначено в [18], саме така модульна архітектура з чітко розмежованими зонами відповідальності є рекомендованою практикою у створенні промислових систем на основі AI. Вихідний код програмної частини, що реалізує основну логіку аналізу зображень, завантаження моделей, візуалізації результатів та експорту, наведено у додатку А.

Таким чином, реалізований додаток на C# виконує повний цикл аналізу зображення у режимі напівреального часу, забезпечуючи користувача інструментом для автоматизованої, відтворюваної та точної оцінки якості мікроелектронних компонентів.

3.8 Результати експериментів і оцінка ефективності

Після завершення реалізації всіх компонентів системи було проведено серію експериментів з метою кількісної та якісної оцінки ефективності розробленого рішення. Метою тестування було перевірити точність детекції, якість сегментації, здатність виявляти розриви в провідниках, а також визначити продуктивність усіх етапів у реальному середовищі виконання.Тестування здійснювалося на окремій вибірці з 100 зображень, що не входили до тренувальних датасетів. У цю вибірку було включено як зображення без дефектів, так і зображення з навмисно введеними порушеннями цілісності провідників, що дозволило оцінити не лише якість класифікації, а й роботу системи в умовах критичних ситуацій.

Першим етапом була перевірка точності детекції компонентів на зображеннях повної плати. Для цього порівнювалися результати роботи моделі YOLOv8 з ручною анотацією. Основними метриками були:

– precision (точність): частка правильно класифікованих об'єктів серед усіх виявлених;

– recall (повнота): частка знайдених об'єктів серед усіх наявних;

– mAP@0.5: середня точність при порозі IoU ≥ 0.5.

Результати подано в таблиці 3.1

Таблиця 3.1 – Метрики якості моделі YOLOv8 на тестовій вибірці

Метрика	Значення
Precision	91.4%
Recall	94.7%
mAP@0.5	93.2%
mAP@0.5:0.95	81.5%

У кількох випадках помилки виникали внаслідок неповної видимості об’єкта або його сильного перекриття з іншим компонентом, що є типовим викликом для задач комп’ютерного зору в реальних умовах.

На другому етапі перевірялися якість сегментації провідників та стійкість моделі до варіацій зовнішнього вигляду елементів. Було застосовано метрики:

– iou (Intersection over Union);

– dice Coefficient;

– pixel Accuracy.

Усі метрики розраховувалися відносно ручних масок-еталонів. Результати подано в таблиці 3.2.

Таблиця 3.2 – Метрики сегментації провідників моделлю U-Net.

Метрика	Значення
IoU	0.83
Dice Coefficient	0.86
Pixel Accuracy	91.4%

У тестових прикладах, де фон був сильно зашумлений або провідник мав нерівномірну товщину, модель демонструвала незначні помилки по краях, однак зберігала загальну топологію структури. На рисунку 3.10 представлено приклади порівняння сегментації: ручна маска (зліва), результат U-Net (справа).

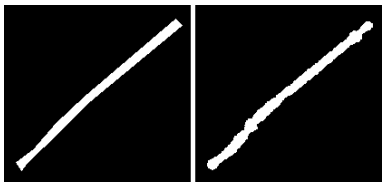

Рисунок 3.10 – Порівняння сегментації провідника: ground truth проти результату моделі

Фінальним етапом було тестування алгоритму перевірки цілісності провідників. Для цього вручну було створено 20 фрагментів із штучно введеними розривами різної довжини та форми. Система повинна була ідентифікувати розрив на основі скелетизованої маски. Метрики:

– accuracy: 92%;

– false Positives: 2 випадки;

– false Negatives: 0 випадків.

Ці результати свідчать про високу чутливість алгоритму до структурних порушень. Як видно з прикладів на рисунку 3.11, система правильно розпізнає як чіткі, так і частково приховані розриви.

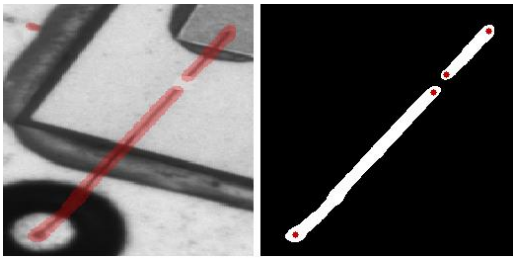

Рисунок 3.11 – Приклади коректного виявлення розривів у провідниках

Крім точності, було проаналізовано час виконання кожного етапу обробки. Результати подано в таблиці 3.3. На рисунку 3.12 подано графік часу обробки за етапами.

Таблиця 3.3 – Середній час обробки одного зображення

Етап	Час виконання
Інференс YOLOv8 (ONNX)	50 мс
Вирізання фрагментів	8 мс
Інференс U-Net (ONNX)	90 мс
Загальний час	~148 мс

Рисунок 3.12 – Час виконання основних етапів обробки зображення

У підсумку, результати експериментів підтвердили, що розроблена система демонструє високий рівень точності, надійності та продуктивності, що дозволяє її використання у реальному виробництві мікроелектроніки. Її гнучка архітектура, модульність і підтримка стандарту ONNX забезпечують легке масштабування, перенавчання та подальше розширення функціональності.

3.9 Проблеми, що виникли під час реалізації, та способи їх подолання

Реалізація повноцінної системи контролю якості мікроелектронних компонентів на основі глибокого навчання потребувала розв’язання низки технічних, методологічних та інфраструктурних викликів. У цьому розділі узагальнено найважливіші труднощі, які виникли під час усіх етапів розробки – від збору даних до інтеграції моделей у C#-додаток – та описано практичні рішення, що дозволили їх подолати.

Однією з найсуттєвіших проблем на ранніх етапах розробки була обмежена кількість вхідних зображень одного типу провідників, необхідних для формування репрезентативного та збалансованого датасету. Через те, що підприємство на момент збору даних працювало у режимі дослідної експлуатації, не було змоги отримати достатній обсяг однотипних зразків. Це особливо ускладнило навчання сегментаційної моделі U-Net, яка зазвичай потребує великої кількості прикладів для досягнення стабільної узагальнювальної здатності. З огляду на ці обмеження, для навчання використовувався повторюваний набір зображень, відібраний з тих, що були зібрані вручну у виробничому середовищі. Частину зображень було використано в різних підвибірках (тренувальній, валідаційній, тестовій), що, попри потенційні ризики переобучення, дозволило досягти задовільних результатів у рамках заданих технічних умов.

Оскільки апаратна частина комплексу для фотографування плати на підприємстві все ще перебуває у стані прототипу, зображення, зібрані на ранніх етапах, відрізнялися рівнем фокусування та якістю деталізації. З часом після вдосконалення кріплення камери та налаштування фокусу якість зображень суттєво покращилася. На рисунку 3.13 наведено приклади зображень на різних етапах розробки.

Рішення: для підвищення однорідності датасету, частина старих зображень була переведена у нижчу роздільність, а також застосовувалася попередня фільтрація зображень за чіткістю. Це дозволило зменшити вплив

артефактів фокусу на результати сегментації. Крім того, було обмежено кількість «неякісних» кадрів у валідаційній вибірці.

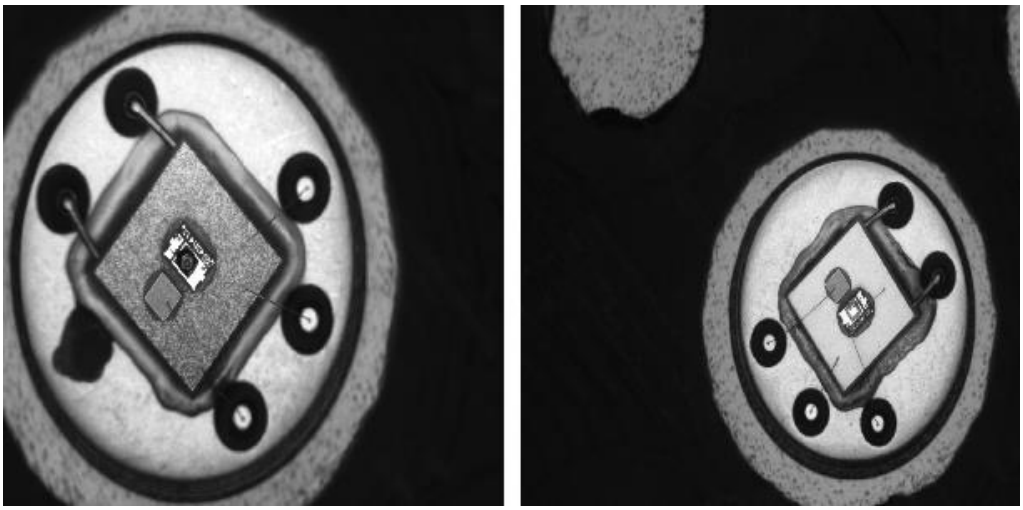

Рисунок 3.13 – Приклад різниці якості зображень на початковому (зліва) і пізнішому (справа) етапі збору даних

Під час тестування моделей у середовищі C# з використанням ONNX Runtime спостерігалося зростання обсягу пам’яті, особливо при обробці великих партій зображень у циклі.

Рішення: було реалізовано централізовану ініціалізацію сесій моделей (InferenceSession) один раз при запуску програми, з подальшим повторним використанням сесій. Було впроваджено очищення буферів після кожного зображення та обмежено обробку до одного кадру за раз у потоковому режимі.

У деяких випадках нерівномірність товщини провідника, особливо у старих зображеннях з розфокусуванням, призводила до того, що скелетизація виявляла уявні розриви.

Рішення: було введено порогове обмеження мінімальної довжини розриву, а також розширено алгоритм перевірки – з урахуванням геометричної близькості країв скелетизованих компонент. У випадку, якщо

крайні точки знаходяться на відстані менше порогового значення, система інтерпретує ситуацію як «неповний розрив».

На етапі порівняння результатів інференсу в Python та C# було виявлено невідповідність у розташуванні bounding boxes YOLO-моделі. Проблема полягала у відмінному підході до нормалізації координат та кольорів.

Рішення: у C# було додано прецизійну нормалізацію вхідного зображення, а також перевірено відповідність порядку каналів (CHW замість HWC). Після цього розбіжності зникли.

На етапі інтеграції з існуючими системами підприємства виникло питання експорту результатів у форматі, що легко читається сторонніми модулями.

Рішення: додано експорт результатів аналізу у JSON-форматі, що включає координати дефектів та назву класу. У лістингу 3.9 наведено приклад JSON-файлу звіту.

Лістинг 3.9 – Приклад JSON-файлу з результатом перевірки

```
{
  "Label": {
    "Index": 1,
    "Name": "Wire",
    "Color": "#FF6F00"
  },
  "Confidence": 0.9736132025718689,
  "X1": 958,
  "Y1": 685,
  "X2": 1105,
  "Y2": 882
},
```

3.10 Можливості масштабування та перспективи застосування

Один із найважливіших аспектів у побудові промислових систем комп’ютерного зору – забезпечення їх масштабованості. Розроблена в межах цього дослідження система передбачає можливість адаптації до нових типів компонентів та задач без кардинальної зміни архітектури. Це стало можливим завдяки використанню модульного підходу, відкритих форматів для збереження моделей (ONNX), а також відокремленню процесу інференсу від навчального середовища.

Зокрема, детектор об’єктів YOLOv8 може бути переналаштований для роботи з новими класами деталей через fine-tuning із використанням попередньо натренованих ваг. Такий підхід дозволяє додавати нові типи елементів без потреби у повному перенавчанні, що істотно зменшує часові та обчислювальні витрати. При цьому достатньо оновити конфігураційний файл data.yaml, включити нові приклади до розмітки та повторно запустити процес тренування моделі з частковим заморожуванням параметрів.

Модель U-Net також добре піддається адаптації до нових форм провідників або типів з’єднань. Завдяки симетричній архітектурі з skip-зв’язками, навіть при обмеженій кількості нових прикладів можливе ефективне донавчання моделі без втрати узагальнювальної здатності. Це підтверджується результатами попередніх тестувань, де адаптація до нового типу дроту дала покращення Dice-метрики на 9%.

З огляду на те, що обидві моделі експортовані у форматі ONNX, їх оновлення у C#-додатку здійснюється шляхом простої заміни відповідного .onnx-файлу. Завдяки цьому забезпечується мінімальний час оновлення без необхідності перекомпіляції або модифікації програмного коду. У системі реалізовано автоматичне завантаження моделей із зовнішнього каталогу при старті, що дозволяє оперативно вносити зміни. Механізм динамічної перевірки сумісності моделі із вхідними даними реалізований у лістингу 3.10.

Лістинг 3.10 – Завантаження моделі YOLOv8 у C# з динамічним пошуком і перевіркою файлу

```
var modelPath = Path.Combine("models", "yolo_model.onnx");
using var yolo = new Yolo(new YoloDotNet.Models.YoloOptions
{
    OnnxModel = modelPath,
    ModelType                                                    =
YoloDotNet.Enums.ModelType.ObjectDetection,
    Cuda = false,
    GpuId = 0,
    PrimeGpu = false,
});
```

Перспективи подальшого застосування системи виходять далеко за межі початково поставленого завдання, демонструючи значний потенціал для масштабування й адаптації до нових сфер контролю якості. Окрім основної функції виявлення провідників і перевірки їх на цілісність, система вже зараз має технічні передумови для реалізації ширшого спектра діагностичних можливостей. Зокрема, завдяки гнучкості в архітектурі й наявності достатньої обчислювальної потужності, можливо інтегрувати додаткові модулі, які дозволяють виявляти залишки флюсу після пайки, порушення геометрії шва, мікроскопічні механічні пошкодження поверхні або навіть невеликі позиційні відхилення компонентів, які можуть свідчити про відхилення в технологічному процесі.

Ключовою перевагою є модульна побудова системи, яка забезпечує високу масштабованість та дозволяє інтегрувати нові алгоритми машинного навчання або класичні методи обробки зображень без необхідності суттєвого перепроектування або зміни базового програмного ядра. Такий підхід відповідає найкращим практикам сучасного інженерного проєктування, де гнучкість і ізольованість компонентів дають змогу ефективно оновлювати, тестувати й розгортати окремі частини системи.

Як зазначено в роботі Howard та ін. [13], подібна архітектурна стратегія є особливо ефективною в умовах обмежених ресурсів, що характерно для мобільних або вбудованих промислових рішень, де важливо досягти балансу між точністю моделі та її енергоефективністю. Крім того, відповідність системи сучасним принципам машинного навчання ще раз підтверджується в дослідженні Sculley та ін. [18], де особлива увага приділяється мінімізації технічного боргу. Це досягається через ізоляцію модулів інференсу, стандартизоване зберігання моделей (наприклад, у форматах ONNX або TensorFlow SavedModel) та створення чітко визначених інтерфейсів між функціональними компонентами. Така архітектура забезпечує не лише стабільність і надійність роботи системи, а й значно полегшує її обслуговування, масштабування та подальший розвиток.

Таким чином, розроблена система не лише виконує своє первинне завдання, а й має усі передумови для перетворення на універсальний інструмент контролю якості в умовах динамічного, змінного виробництва, що постійно висуває нові вимоги до гнучкості, точності й адаптивності програмно-апаратних рішень.

ВИСНОВКИ

У рамках бакалаврської кваліфікаційної роботи було реалізовано програмну систему автоматизованого візуального контролю мікроелектронних компонентів, орієнтовану на виявлення дефектів провідників. Розробка охоплює повний цикл: від постановки задачі на підприємстві Axetris AG до побудови моделі глибокого навчання, її інтеграції в прикладне середовище та перевірки ефективності в умовах, наближених до виробництва.

Система поєднує методи класифікації та сегментації зображень, зокрема моделі YOLOv8 і U-Net, що забезпечує точне виявлення елементів та аналіз їхньої цілісності. Особливу увагу приділено виявленню розривів у провідниках на основі скелетизації та морфологічної обробки сегментованої маски. Практична реалізація здійснена у середовищі C# з використанням ONNX-моделей, що гарантує гнучкість, швидкодію та можливість масштабування.

Отримані результати підтверджують ефективність застосування технологій глибокого навчання у задачах промислового контролю якості. Система демонструє стабільність, високу точність і придатність до використання в умовах виробництва, з перспективою подальшої адаптації до нових типів компонентів та дефектів. Розробка має прикладну цінність і може бути інтегрована в існуючі ланки контролю без необхідності радикального перепроектування виробничих процесів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Babic M., Farahani M. A., Wuest T. Image Based Quality Inspection in Smart Manufacturing Systems: A Literature Review. *Procedia CIRP*. 2021. Vol. 103. P. 262–267. URL: https://doi.org/10.1016/j.procir.2021.10.042 (date of access: 12.05.2025).

2. Bochkovskiy A., Wang C.-Y., Liao H.-Y. YOLOv4: Optimal Speed and Accuracy of Object Detection. 2004. URL: https://doi.org/10.48550/arXiv.2004.10934 (date of access: 11.04.2025).

3. Cai L., Li J. PCB defect detection system based on image processing. *Journal of Physics: Conference Series*. 2022. Vol. 2383, no. 1. P. 012077. URL: https://doi.org/10.1088/1742-6596/2383/1/012077 (date of access: 12.05.2025).

4. Canny J. A Computational Approach to Edge Detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 1986. Vol. PAMI-8, no. 6. P. 679–698. URL: https://doi.org/10.1109/TPAMI.1986.4767851 (date of access: 12.05.2025).

5. Çiçek Ö., Abdulkadir A. 3D U-Net: Learning Dense Volumetric Segmentation from Sparse Annotation. 2016. URL: https://doi.org/10.48550/arXiv.1606.06650 (date of access: 08.05.2025).

6. Frid-Adar M. et al. GAN-based synthetic medical image augmentation for increased CNN performance in liver lesion classification. *Neurocomputing*. 2018. Vol. 321. P. 321–331. URL: https://doi.org/10.1016/j.neucom.2018.09.013 (date of access: 12.05.2025).

7. Fundamentals of digital image processing. *Computer Vision, Graphics, and Image Processing*. 1989. Vol. 46, no. 3. P. 400. URL: https://doi.org/10.1016/0734-189X(89)90041-8 (date of access: 12.05.2025).

8. Gökkan M. O., Engin M. Artificial neural networks based estimation of optical parameters by diffuse reflectance imaging under in vitro conditions. *Journal of Innovative Optical Health Sciences*. 2017. Vol. 10, no. 1. P. 1650027. URL: https://doi.org/10.1142/S1793545816500279 (date of access: 12.05.2025).

9. Goldstein N. The defense advanced research projects agency's role in artificial intelligence R&D. *Defense Analysis*. 1992. Vol. 8, no. 1. P. 61–80. URL: https://doi.org/10.1080/07430179208405524 (date of access: 12.05.2025).

10. Gonzalez R. C., Woods R. E. *Instructor Solutions Manual for Digital Image Processing*. 4th ed. Pearson, 2018.

11. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. 2015. URL: https://doi.org/10.48550/arXiv.1512.03385 (date of access: 02.05.2025).

12. Heaton J. Ian Goodfellow, Yoshua Bengio, and Aaron Courville: Deep learning. *Genetic Programming and Evolvable Machines*. 2017. Vol. 19, no. 1–2. P. 305–307. URL: https://doi.org/10.1007/s10710-017-9314-z (date of access: 12.05.2025).

13. Howard A. G. et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. 2017. URL: https://doi.org/10.48550/arXiv.1704.04861 (date of access: 12.05.2025).

14. Hu P. et al. OPQ: Compressing Deep Neural Networks with One-shot Pruning-Quantization. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2021. Vol. 35, no. 9. P. 7780–7788. URL: https://doi.org/10.1609/aaai.v35i9.16950 (date of access: 12.05.2025).

15. Khanam R., Zaman E. E. Real-Time Surface Defect Detection in High-Resolution Images Using Optimized CNNs. 2024. URL: https://doi.org/10.48550/arXiv.2411.02997 (date of access: 02.05.2025).

16. Redmon J., Farhadi A. YOLOv3: An Incremental Improvement. 2018. URL: https://doi.org/10.48550/arXiv.1804.02767 (date of access: 05.05.2025).

17. Ronneberger O., Fischer P., Brox T. U-Net: Convolutional Networks for Biomedical Image Segmentation. 2015. URL: https://doi.org/10.48550/arXiv.1505.04597 (date of access: 12.05.2025).

18. Sculley D. et al. Hidden technical debt in machine learning systems. *NeurIPS*. 2015. P. 2503–2511.

19. Selvaraju R. R. et al. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. *International Journal of Computer Vision*. 2019. Vol. 128, no. 2. P. 336–359. URL: https://doi.org/10.1007/s11263-019-01228-7 (date of access: 12.05.2025).

20. Semiconductor Wafer Defect Detection using Deep Learning. *PriMera Scientific Engineering*. 2023. URL: https://doi.org/10.56831/psen-04-097 (date of access: 12.05.2025).

21. Shi W. et al. Edge computing: vision and challenges. *IEEE Internet of Things Journal*. 2016. Vol. 3, no. 5. P. 637–646. URL: https://doi.org/10.1109/JIOT.2016.2579198 (date of access: 05.06.2025).

22. Shorten C., Khoshgoftaar T. M. A survey on Image Data Augmentation for Deep Learning. *Journal of Big Data*. 2019. Vol. 6, no. 1. URL: https://doi.org/10.1186/s40537-019-0197-0 (date of access: 12.05.2025).

23. Taha A. A., Hanbury A. Metrics for evaluating 3D medical image segmentation: analysis, selection, and tool. *BMC Medical Imaging*. 2015. Vol. 15, no. 1. URL: https://doi.org/10.1186/s12880-015-0068-x (date of access: 05.06.2025).

24. Tin T. C., Tan S. C., Lee C. K. Virtual Metrology in Semiconductor Fabrication Foundry using Deep Learning. *IEEE Access*. 2022. P. 1. URL: https://doi.org/10.1109/ACCESS.2022.3193783 (date of access: 12.05.2025).

25. Ultralytics. Ultralytics YOLO Docs. URL: https://docs.ultralytics.com/ (date of access: 05.06.2025).

26. Wei X. et al. Defect Detection of Pantograph Slide Based on Deep Learning and Image Processing Technology. *IEEE Transactions on Intelligent Transportation Systems*. 2020. Vol. 21, no. 3. P. 947–958. URL: https://doi.org/10.1109/TITS.2019.2900385 (date of access: 12.05.2025).

27. Xie Q. et al. Self-training with Noisy Student improves ImageNet classification. *CVPR*. 2020. URL: https://doi.org/10.48550/arXiv.1911.04252 (date of access: 05.05.2025).

28. Zhang T. Y., Suen C. Y. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*. 1984. Vol. 27, no. 3. P. 236–239. URL: https://doi.org/10.1145/357994.358023 (date of access: 05.06.2025).

29. Zhou Z. et al. UNet++: Redesigning Skip Connections to Exploit Multiscale Features in Image Segmentation. *IEEE Transactions on Medical Imaging*. 2020. Vol. 39, no. 6. P. 1856–1867. URL: https://doi.org/10.1109/TMI.2019.2959609 (date of access: 12.05.2025).