ПОЛТАВСЬКИЙ ДЕРЖАВНИЙ АГРАРНИЙ УНІВЕРСИТЕТ
Навчально-науковий інститут економіки, управління, права та інформаційних технологій
Кафедра інформаційних систем та технологій

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти магістр

на тему: **«Автоматизація візуального контролю продукції на конвеєрі за допомогою комп'ютерного зору»**

Виконав: здобувач вищої освіти
за освітньою програмою
Інформаційні управляючі системи та технології
спеціальності 126 Інформаційні системи та технології
ступеня вищої освіти магістр
групи 126ІСТ_мз_2023[1](л. н.)
Мухін Валентин Олегович
Керівник: Слюсарь Ігор Іванович
Рецензент: Муравльов Володимир В'ячеславович

Полтава – 2024 року

ВСТУП

Актуальність теми кваліфікаційної роботи підтверджується необхідністю застосування алгоритмів машинного навчання, зокрема нейронних мереж, у системах сортування продукції на конвеєрах є ключовим кроком до автоматизації виробничих процесів. Технології комп'ютерного зору та сучасні архітектури нейронних мереж, такі як YOLO, забезпечують високу точність, швидкість і адаптивність, що дозволяє ефективно вирішувати завдання сортування продукції в реальному часі.

Інтеграція штучного інтелекту у виробничі процеси сприяє значному підвищенню продуктивності, скороченню дефектів і втрат, оптимізації витрат, а також забезпечує гнучкість і масштабованість рішень. Однак, питання розробки рекомендацій щодо використання комп'ютерного зору і нейронних мереж архітектури YOLO11 в системі автоматизованого візуального контролю продукції на конвеєрі потребують додаткових досліджень. Все це свідчить про актуальність теми роботи

Зв'язок роботи з науковими програмами, темами. Робота відповідає дослідженням в межах науково-дослідної ініціативної тематики «Організаційно-методологічні аспекти впровадження інформаційно-комунікаційних систем і технологій в управлінні діяльністю сучасних організацій та підприємств за умов переходу до цифрової економіки» (ДРН 0123U105060, 2023-2028 рр.), що реалізується на кафедрі інформаційних систем та технологій, тематиці досліджень навчально-дослідної лабораторії інтелектуальних систем, комп'ютерних мереж та інтернет речей кафедри інформаційних систем та технологій Полтавського державного аграрного університету.

Метою кваліфікаційної роботи є підвищення ефективності візуального контролю продукції на конвеєрі за рахунок комп'ютерного зору.

Завданнями кваліфікаційної роботи є:

– аналіз напрямів використання штучного інтелекту для систем сортування на конвеєрах;

– дослідження особливостей архітектури та структури коду YOLO11;

– розробка рекомендацій щодо використання комп'ютерного зору і нейронних мереж архітектури YOLO11 в системі автоматизованого візуального контролю продукції.

Об'єктом дослідження є процес візуального контролю продукції на конвеєрі.

Предметом дослідження є методи та засоби автоматизації візуального контролю агропродукції за допомогою комп'ютерного зору на основі нейронних мереж.

Методами дослідження для створення моделі класифікації та виявлення об'єктів і техніко-економічного обґрунтування прийнятих рішень використовувався аналітичний метод досліджень, а для розробки архітектури нейронних мережі і формування датасету – моделювання.

Інформаційна база кваліфікаційної роботи сформована з ресурсів, що містять інформацію про архітектури YOLO11, а також інструментарій для розробки та дослідження нейронних мереж.

Елементи наукової новизни роботи полягають в розробці моделі глибокого навчання нейронної мережі класифікації якості агропродукції.

Практична значущість роботи полягає в розробці рекомендацій щодо використання комп'ютерного зору і нейронних мереж в системі автоматизованого візуального контролю продукції на конвеєрі, які можуть бути використані для подальших досліджень за даною тематикою та при реалізації обробки сільськогосподарської продукції.

Апробація результатів відбувалася в рамках VII Міжнародної студентської наукової конференції «Наука сьогодення: від досліджень до стратегічних рішень» (листопад 2024 р., м. Кривий Ріг) та VII Міжнародної студентської наукової конференції «Розвиток суспільства та науки в умовах цифрової трансформації» (листопад 2024 р., м. Тернопіль).

Структура кваліфікаційної роботи логічно пов'язана з завданнями досліджень і містить вступ, три розділи основної частини, висновки, список використаних джерел, додатки. Загальний обсяг пояснювальної записки кваліфікаційної роботи складає 74 сторінки формату А4. Вона містить 30 рисунків і 3 таблиці.

РОЗДІЛ 1
АНАЛІЗ ОСОБЛИВОСТЕЙ ВИКОРИСТАННЯ ШТУЧНОГО ІНТЕЛЕКТУ В СИСТЕМАХ СОРТУВАННЯ ПРОДУКЦІЇ

1.1 Застосування алгоритмів машинного навчання в системах сортування продукції на конвеєрах

Штучний інтелект (Artificial Intelligence, AI) [1] активно впроваджується у системи сортування продукції на конвеєрах [2]. Використання AI у системах сортування на конвеєрах значно покращує процеси за рахунок впровадження передових технологій, таких як комп'ютерний зір (Computer Vision,CV) [3], робототехніка та машинне навчання [4]. Це призводить до підвищення продуктивності, поліпшення контролю якості та зниження витрат у різних галузях промисловості, наприклад:

– харчова промисловість – сортування фруктів, овочів та інших продуктів за розміром, ступенем зрілості чи наявністю дефектів;

– логістика та складування – автоматизація сортування посилок за пунктами призначення, розміром або пріоритетом доставки;

– виробництво – сортування компонентів для складальних ліній та видалення дефектних деталей для підтримки контролю якості;

– переробка відходів – ідентифікація та розподіл різних типів перероблюваних матеріалів.

До основних переваг такого підходу можна віднести [5]:

– AI-системи сортування працюють швидше, ніж ручні методи, збільшуючи пропускну спроможність.

– зниження помилок у сортуванні призводить до більш високої якості продукції та задоволеності клієнтів.

– автоматизація скорочує витрати на працю та мінімізує втрати від бракованої продукції.

– AI-системи можуть бути легко масштабовані відповідно до потреб виробництва без істотних змін інфраструктури.

Алгоритми машинного навчання (MachineLearning, ML) відіграють ключову роль сучасних системах сортування продукції на конвеєрах. Вони дозволяють обробляти зображення, отримані з високошвидкісних камер, і приймати рішення в режимі реального часу для ідентифікації, класифікації та сортування продукції за різними характеристиками, такими як розмір, форма, колір або текстура. Розглянемо більш детально, яким чином використовуються нейронні мережі.

Під час аналізу зображень нейронні мережі виділяють релевантні ознаки продукції, таких як контури, колірні градієнти, текстурні характеристики тощо. На основі одержаних ознак алгоритми навчаються розпізнавати певні патерни, що відповідають різним категоріям продукції або дефектам. Надалі реалізуються класифікатори, які використовуються для віднесення кожного об'єкта до певної категорії (наприклад, сорт продукції, ступінь зрілості, дефектів). Тобто вони можуть виявляти відхилення від норми, що допомагає виявляти дефектні чи некондиційні продукти. Потім забезпечується прийняття рішення у реальному часі без затримок у виробничому процесі. Результати аналізу передаються на виконавчі механізми (наприклад, сортувальні пристрої, роботизовані комплекси), які виконують відповідні дії. На даний час, для виконання завдань по сортуванню продукції може випростовуватись кілька архітектур нейронних мереж. Згорткові нейронні мережі (Convolutional Neural Network, CNN) [6] – це тип глибоких нейронних мереж, особливо ефективних у завданнях обробки зображень (рис. 1.1). Використовуються для розпізнавання об'єктів, класифікації зображень та виявлення дефектів. До їх переваг слід віднести: високу точність, здатність автоматично отримувати релевантні ознаки без необхідності ручного програмування.

До використання нейронних мереж для класифікації використовували кілька алгоритмів класифікації:

– Support Vector Machines (SVM) – ефективні для класифікації, особливо за наявності великої кількості ознак;

– K-найближчих сусідів (KNN) – простий та зрозумілий алгоритм для класифікації на основі подібності до сусідніх зразків;

– Дерева рішень та випадкові ліси –для побудови моделей, які легко інтерпретувати та можуть обробляти як числові, так і категоріальні дані.

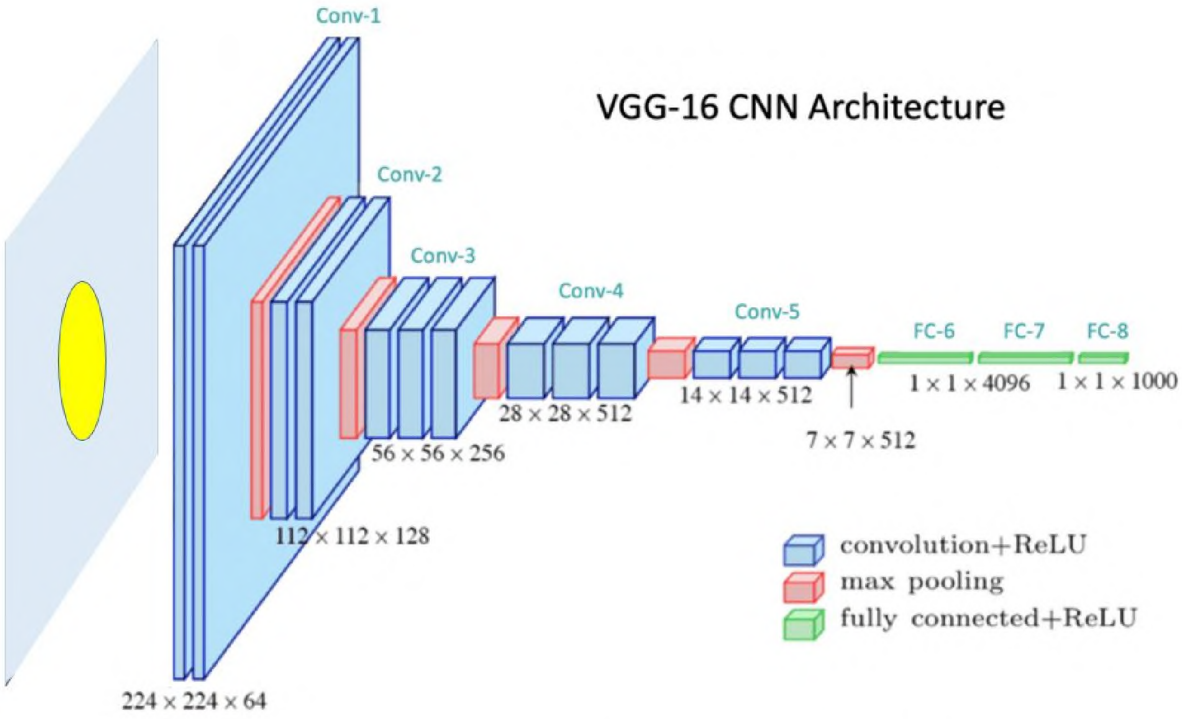

Рисунок 1.1 – Архітектура CNN VGG-16 [7]

В свою чергу, для сортування потрібно виконувати процедури групування – кластеризацію. Базовими алгоритмами кластеризації є k-середня (допомагає групувати об'єкти на основі подібності ознак, що корисно для сегментації продукції) та ієрархічна кластеризація (створює ієрархію груп, що може бути корисним при сортуванні за декількома рівнями якості) [8]. Щоб виявляти аномалії можуть використовуватись автокодувальники та однокласові SVM. Автокодувальники – тип нейронних мереж, навчених відновлювати вхідні дані, що дозволяє виявляти аномалії через високі помилки відновлення [9]. Однокласові SVM навчаються на

нормальних даних та виявляють відхилення від них. Одним з завдань CV, що використовується при сортуванні є сегментація. Її вирішення має на меті кілька результатів. По-перше, сегментація дозволяє точно виділити об'єкти на зображенні, відокремивши їх від фону та інших об'єктів, забезпечує ізоляція об'єктів на зображені, а також розглядає кожну частину продукту окремо для детального аналізу характеристик та виявлення дефектів. По-друге, забезпечення точності вимірювань розмірів та форми (площу, периметр та інші геометричні характеристики), а також порівняння сегментованих областей із еталонними моделями для виявлення деформацій чи відхилень від норми. У випадку використання класифікаторів, виключення фонових елементів та перешкод з аналізу знижує шуми даних, а також обробка лише тих частин зображення, які містять корисну інформацію, підвищує ефективність та точність алгоритмів ML. Для її реалізації на основі нейронних мереж датасет повинен мати велику кількість зображень у різних умовах та станах. Крім того, для навчання моделей необхідно включити до його складу маски сегментації, що створюються вручну або за допомогою напівавтоматичних інструментів. В даному випадку, модель навчається на розмічених даних, мінімізуючи функцію втрат. Варто також мати на увазі, залежно від виду сегментації залежить вибір відповідної архітектури нейронної мережі (наприклад, U-Net для семантичної сегментації або Mask R-CNN для інстансної [10]).

Щоб інтегрувати у системи сортування нейронні мережі потрібно дотримуватись певної послідовності кроків. Спочатку проводиться збір та підготовка даних. Необхідно зібрати великий обсяг зображень продукції в різних станах (нормальних та дефектів). Надалі фахівці відзначають на зображеннях відповідні класи або дефекти, створюючи навчальну вибірку. Зображення можуть бути нормалізовані, відфільтровані та перетворені для покращення якості даних. Потім виконується глибоке навчання AI-моделей. Дані поділяються на навчальну, валідаційну та тестову вибірки. Визначаються параметри моделі, такі як глибина нейронної мережі, розмір

ядра згортки, швидкість навчання тощо. Модель навчається на вибірці, мінімізуючи функцію втрат. В даному випадку, для нейронної мережі виконується налаштування гіперпараметрів. Під час валідації Перевіряється якість моделі на валідаційній вибірці, коригуються гіперпараметри за потреби. Нарешті, проводиться тестування та оптимізація – модель тестується на нові дані для оцінки її продуктивності. За потреби модель оптимізується для підвищення точності чи швидкості роботи. Після цього виконується інтеграція з виробничою системою, тобто модель впроваджується в систему, де вона отримує зображення від камер та передає результати на виконавчі механізми. Робота моделі контролюється, збираються дані для її подальшого покращення. При цьому доцільно розглянути застосування алгоритмів ML у сортуванні за певними характеристиками.

При сортуванні за розміром продукція класифікується та сортується залежно від виміряних розмірів. Алгоритми визначають габарити продукції на зображенні.

Крім розміру може оцінюватись форма, наприклад, виділяються та аналізуються форми об'єкта на зображенні з метою розпізнавання контурів. Також визначаються відповідності форми заданим стандартам або шаблонам. Виявляється відхилення форми, які можуть свідчити про дефекти з метою виявлення деформацій.

Також основною характеристикою є колір. Використовуються колірні простори (RGB або HSV) для отримання колірних ознак. Потім виконується розподіл (класифікація) продукції за відтінками, насиченістю або яскравістю кольору. Виявлення при цьому зміни кольору можуть вказувати на дефекти або псування продукції.

Нарешті, є ще одна характеристика – текстура. Використання методів, таких як фільтри Габора, гістограми орієнтованих градієнтів (HOG) дозволяє аналізувати текстури поверхні в інтересах вилучення текстурних ознак. Розрізнення продукції на основі текстурних особливостей, що є важливим

для матеріалів з характерною поверхнею, також дозволяє виконувати класифікацію за текстурою, а виявлення порушень однорідності текстури, таких як подряпини, плями або пошкодження, може вказувати на наявність дефектів текстури.

1.2 Аналіз напрямів використання штучного інтелекту для систем сортування на конвеєрах

Розглянемо більш детально основні напрями застосування AI для сортування продукції на конвеєрі.

Першим напрямом є CV та розпізнавання зображень. AI може інтегруватись з відеокамерами (рис. 1.2) [11]. Однак, самі відеокамери повинні бути високошвидкісними (рис. 1.3).

Рисунок 1.2 – Raspberry Pi AI Camera, 12MP, IMX500 Intelligent Vision Sensor

Вони встановлюються над конвеєром для захоплення зображень або формування відеопотоку продукції в русі для подальшого аналізу та прийняття рішень на основі AI. Вони є невід'ємною частиною систем сортування продукції на конвеєрах із використанням AI. Вони виконують функцію «очей» системи, дозволяючи захоплювати зображення або відео продукції. При цьому можуть бути використані різні спектри зйомки,

наприклад, для виявлення візуально помітних дефектів, таких як подряпини, тріщини або зміна кольору – видимий спектр; інфрачервона зйомка виявляє дефекти, невидимі у видимому спектрі (внутрішні пошкодження або температурні аномалії); ультрафіолетова зйомка використовується для виявлення поверхневих дефектів та забруднень; рентгенівська зйомка дозволяє бачити внутрішні структури та виявляти приховані дефекти. Таким чином, мультиспектральна та гіперспектральна зйомка використовується для аналізу характеристик матеріалів, включаючи склад, рівень вологості або наявність забруднень.

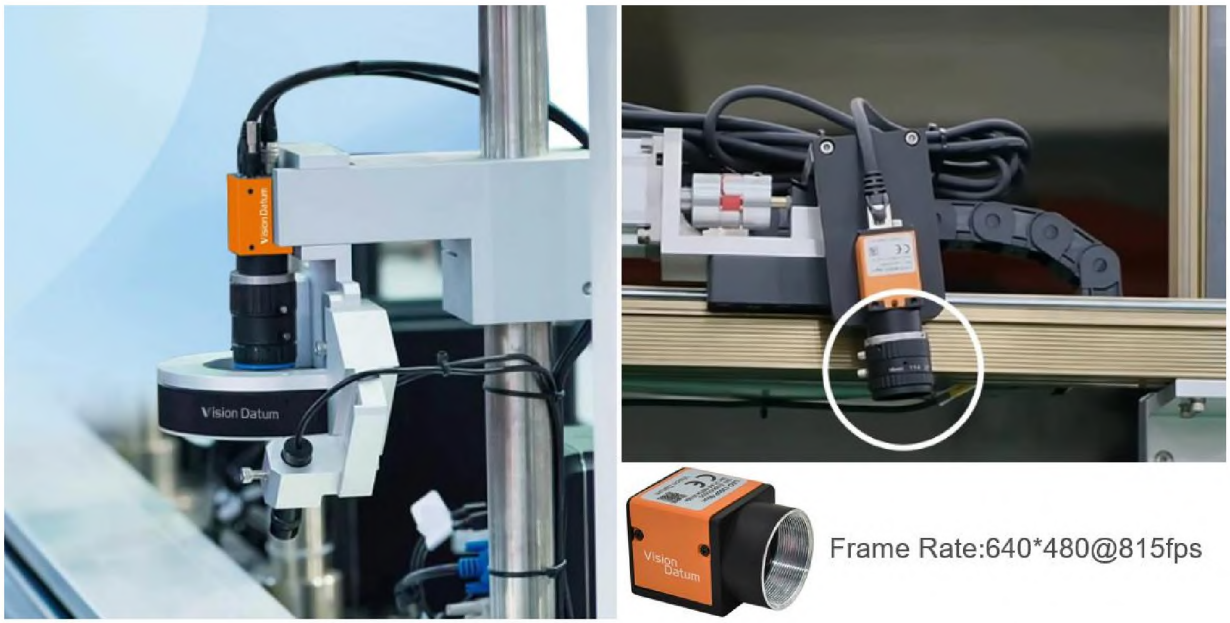

Рисунок 1.3 – Високошвидкісна камера промислового призначення

На виробничих лініях продукція рухається із високою швидкістю. Високошвидкісні камери здатні захоплювати чіткі зображення навіть за швидкого руху об'єктів. Камери можуть знімати з частотою від кількох сотень до кількох тисяч кадрів за секунду, що необхідно для точного аналізу об'єктів, що швидко рухаються. Достатнє розрізнення дозволяє отримувати зображення з високим рівнем деталізації, що важливо для виявлення дрібних дефектів або відхилень. Висока якість та чіткість зображень підвищують точність роботи AI-моделей.

Під час пересування продукції по конвеєру камери синхронізуються з конвеєрною системою та AI-алгоритмами для точного визначення положення кожного об'єкта в момент зйомки їх роботи забезпечується синхронізація із системою AI [12]. Швидка передача та обробка даних забезпечують прийняття рішень у реальному часі. Для таких камер використовуються різні типи сенсорів (CCD або CMOS), кожен з яких має свої переваги щодо чутливості, швидкості та якості зображення. Глобальний затвор запобігає спотворенню зображення під час зйомки об'єктів, що швидко рухаються. Високошвидкісні інтерфейси для передачі великих об'ємів даних із камери на обробку забезпечуючи низьку затримку, яка важлива для своєчасної обробки та прийняття рішень. Камери встановлюються для огляду верхньої поверхні продукції, або під кутом або збоку для огляду бічних поверхонь або специфічних ділянок продукції. Також велику роль відіграє освітлення. Воно повинно бути контрольованим, щоб забезпечити стабільне та рівномірне освітлення зони зйомки. Для цього застосовується LED-підсвічування, яке популярне через довговічність та можливість точного налаштування. Стробоскопічне освітлення синхронізується з камерою для короткочасного освітлення в момент зйомки, що підвищує чіткість зображень. Використання дифузних джерел світла та поляризованих фільтрів усуває відбиття та тіні. В умовах пилового або вологого виробництва потрібно використовувати пило- та вологозахищені корпуси. Для запобігання розмиттю зображень через вібрації обладнання є антивібраційні кріплення. В цілому, виконується синхронізація з іншими системами, наприклад, тригери та датчики використовуються для визначення моменту захоплення зображення, ґрунтуючись на положенні продукції на конвеєрі. До технічних особливостей високошвидкісних камер відноситься: висока частота кадрів (Frames Per Second, FPS) – необхідна для запобігання розмиттю рухомих об'єктів та отримання чітких зображень, а її регулювання дозволяє налаштовувати камеру під конкретні вимоги до виробництва. Мегапіксельні сенсори забезпечують високу деталізацію, необхідну для виявлення дрібних дефектів.

При виборі камери важливо знайти оптимальне співвідношення між частотою кадрів та роздільною здатністю залежно від завдання. На етапі попередньої обробки застосовуються алгоритми для видалення шуму та підвищення контрастності, а також виправляються спотворення, що спричинені оптикою або кутом зйомки. AI-алгоритмів безпосередньо на пристрої, що знижує затримки та навантаження на мережу. Стереокамери та 3D-сканери дозволяють отримувати 3-вимірні зображення для більш детального аналізу.

Разом з тим, висока частота кадрів і роздільна здатність потребують потужних систем обробки [13]. Зовнішні фактори можуть впливати на якість зображення. Різні форми, розміри та матеріали потребують гнучкого налаштування системи. Камери із вбудованими процесорами для виконання. Високошвидкісні камери є ключовим компонентом у сучасних системах сортування та контролю якості на конвеєрах із використанням штучного інтелекту. Їх здатність швидко і точно захоплювати зображення продукції, що швидко рухається, дозволяє AI-системам ефективно аналізувати і приймати рішення в реальному часі. Це призводить до підвищення ефективності виробничих процесів, поліпшення якості продукції та зниження витрат. Камери працюють в єдиній системі з AI-алгоритмами та механізмами сортування або відбраковування. Під час розпізнавання об'єктів виконується ідентифікація продукції та її особливостей, визначається категорія або якість продукції для подальшого сортування, а також виявляються дефекти (пошук відхилень від еталонних зразків, виявлення подряпин, тріщин, зміни кольору та ін. дефектів). Алгоритми ML обробляють отримані зображення в режимі реального часу для ідентифікації, класифікації та сортування продукції за різними характеристиками (розмір, форма, колір або текстура) [14].

Другий напрям стосується предиктивної аналітики [15]. Моделі AI аналізують дані на льоту для миттєвого прийняття рішень про сортування, перенаправлення або відбракування продукції, передбачати вузькі місця або неефективність у процесі сортування та вносити корективи для покращення

продуктивності. Система збирає дані про дії щодо сортування та їх результати для постійного покращення точності моделей AI [16].

Третій напрям пов'язаний з роботизацією та автоматизацією. Якщо застосовувати скерування на основі AI, то виконавчі механізми здатні швидко та точно переміщувати продукцію, а також проводити динамічне коригування [17]. Воно дозволяє роботам адаптуватися до змін у положенні чи орієнтації продукції без втручання людини.

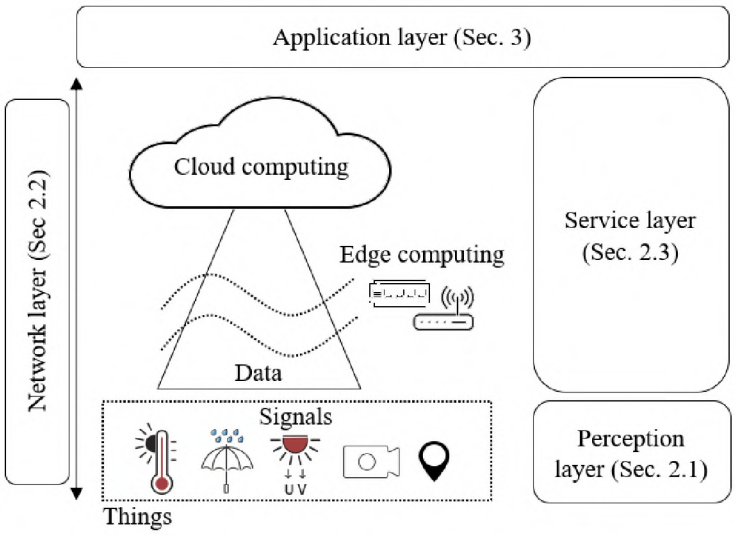

Рисунок 1.4 – Архітектура системи IoT

Четвертий напрям орієнтований на інтеграцію з пристроями Інтернет речей (IoT) для моніторингу стану обладнання (рис. 1.4) [18], прогнозування потреб в обслуговуванні та забезпечення безперебійної роботи та ін. [19].

1.3 Вибір архітектури нейронної мережі

Згідно п. 1.2, для реалізації сортування продукції на конвеєрі на основі AI треба визначитись з архітектурою нейронної мережі. Проведений аналіз дозволяє зробити висновки про доцільність застосування You Only Look Once (YOLO) [20-24], який вважається одним із найвідоміших алгоритмів

виявлення об'єктів. Серед одноетапних методів виявлення об'єктів YOLO виділяється своєю надійністю та ефективністю. Вперше представлений у 2015 р., YOLO переосмислив виявлення об'єктів шляхом прогнозування обмежувальних рамок і ймовірностей класу безпосередньо з повних зображень в одній оцінці. Найновіші версії YOLO, включаючи YOLOv8 [25], YOLOv9, YOLOv10 і YOLO11, є авангардом розвитку моделі (рис. 1.5).

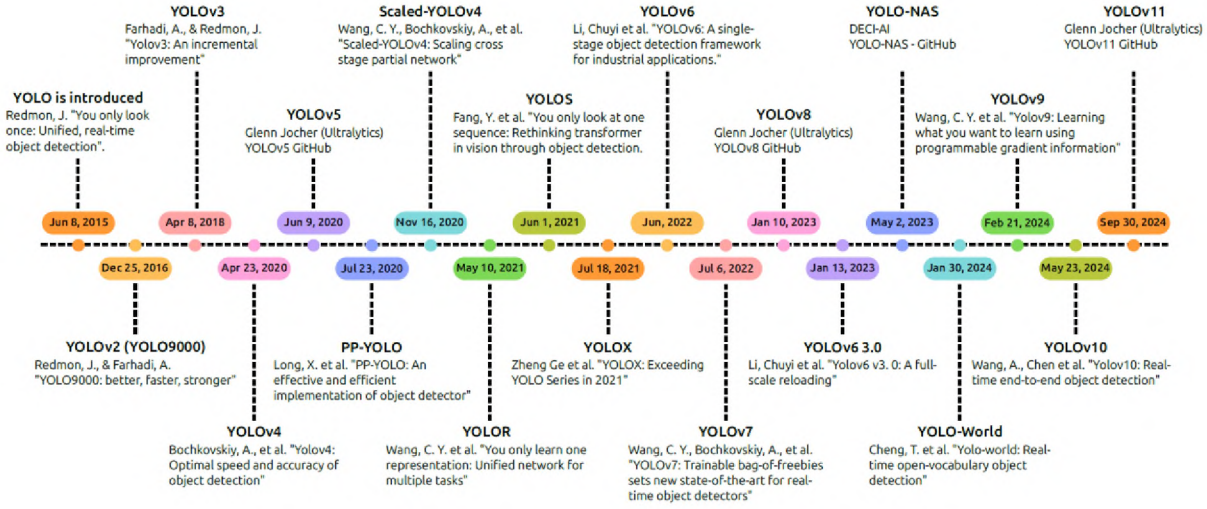

Рисунок 1.5 – Еволюція алгоритмів YOLO

Він забезпечує найсучаснішу швидкість і точність, а різноманітні застосування зробили його незамінним у багатьох сферах і галузях. Численні дослідники показали інтерес до цього алгоритму виявлення об'єктів, опублікувавши статті, в яких описується його еволюція, точне налаштування його моделей і порівняння його продуктивності з іншими алгоритмами CV. Цей широкий інтерес підкреслює важливу роль YOLO у розвитку CV [26]. YOLO є одноетапним алгоритмом детекції об'єктів, який обробляє зображення за один прохід через нейронну мережу. Це забезпечує високу швидкість роботи, що є особливо важливим для систем реального часу на конвеєрі, де продукція рухається з високою швидкістю. YOLOv8 (рис. 1.6), випущений Ultralytics [27], представив можливості семантичної сегментації, дозволяючи моделі класифікувати кожен піксель зображення, і надав масштабовані версії для задоволення різноманітних потреб додатків, від

середовищ з обмеженими ресурсами до високопродуктивних систем разом з іншими такі завдання, як оцінка пози, класифікація зображень і виявлення орієнтованого об'єкта (OOB).

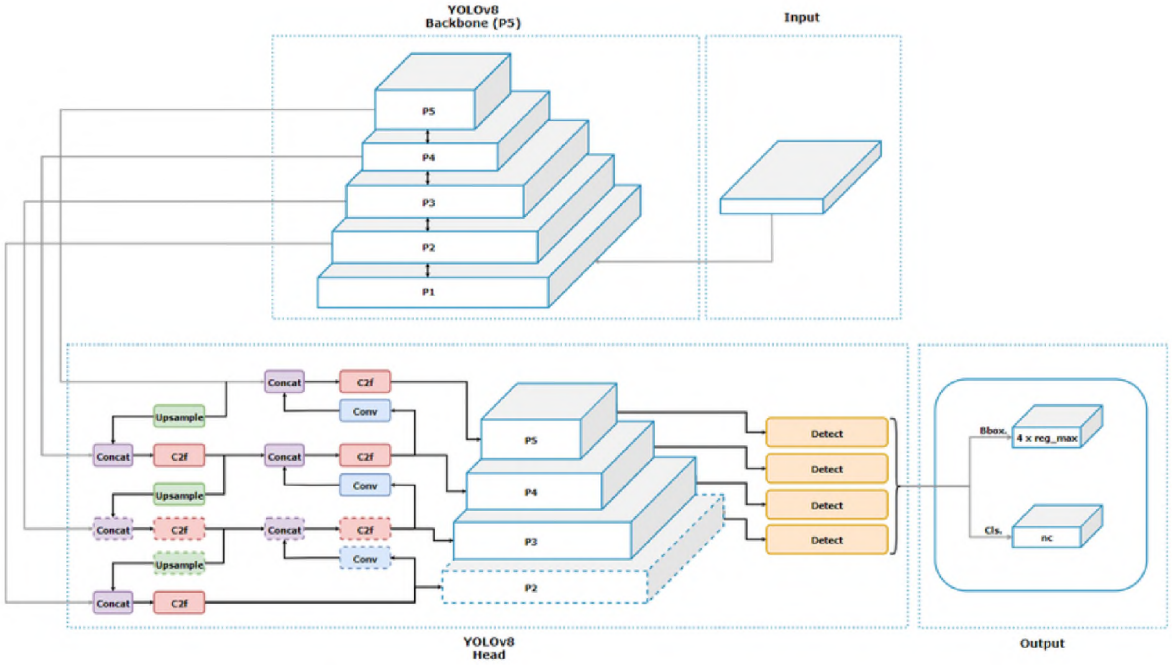

Рисунок 1.6 – Архітектура YOLOv8

Архітектура YOLOv8 демонструє кілька згорткових рівнів магістральної мережі для вилучення ієрархічних функцій, мережу піраміди функцій (FPN) для покращення виявлення в різних масштабах і головну мережу для виконання остаточних прогнозів, включаючи згорткові блоки та блоки підвищення дискретизації для уточнення функцій. YOLOv8 для сегментації використовує нову архітектуру, яка складається з екстрактора функцій CSPDarknet53, модуля C2f і модуля SegHead. CSPDarknet53 – це магістральна мережа, яка витягує функції високого рівня з вхідного зображення. Модуль C2f – це міжмасштабний модуль злиття, який поєднує функції з різних масштабів і покращує представлення функцій. Модуль SegHead – це голова сегментації, яка передбачає маску сегментації для кожного класу та кожного масштабу.

Нарешті, YOLO11, також представлений Ultralytics, зберігає

можливості YOLOv8 із такими програмами, як сегментація екземплярів, оцінка пози та виявлення орієнтованого об'єкта, надаючи при цьому 5 масштабованих версій для різних випадків використання [28, 29]. YOLO11 є останньою інновацією в серії YOLO, розробленою Ultralytics, спираючись на розробки своїх попередників, особливо YOLOv8 [30]. Ця ітерація пропонує п'ять масштабованих моделей від нано до надвеликих, призначених для різних застосувань. Як і YOLOv8, YOLO11 містить численні програми, такі як виявлення об'єктів, сегментація екземплярів, класифікація зображень, оцінка пози та орієнтоване виявлення об'єктів (OBB). Ключові вдосконалення в YOLO11 включають впровадження модуля C2PSA (Cross-Stage Partial with Self-Attention) – рис. 1.7, який поєднує в собі переваги міжступеневих часткових мереж із механізмами самоуваги [31].

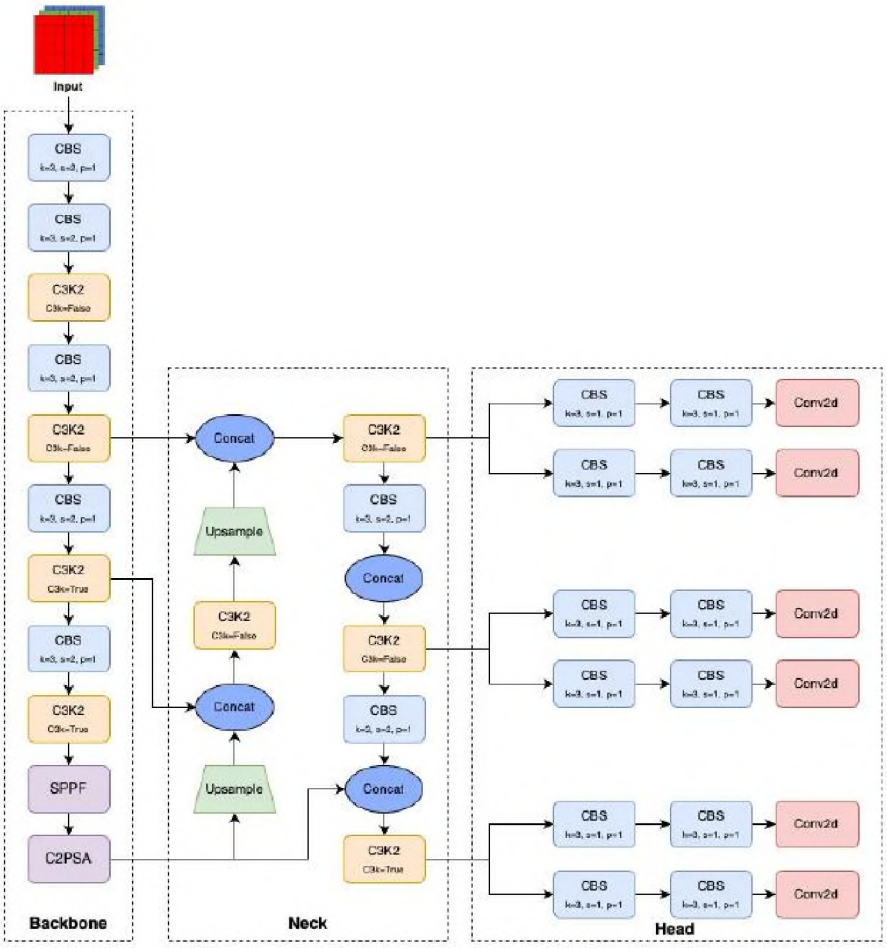

Рисунок 1.7 – Архітектура YOLO11,
що демонструє нові блоки C3k2 і модуль C2PSA

Це дає змогу моделі ефективніше фіксувати контекстну інформацію на кількох рівнях, підвищуючи точність виявлення об’єктів, особливо для невеликих і змовлених об’єктів (рис. 1.8). Крім того, у YOLO11 блок C2f було замінено на C3k2, спеціальну реалізацію CSP Bottleneck, яка використовує дві згортки, на відміну від використання однієї великої згортки в YOLOv8. Цей блок використовує менше ядро, зберігаючи точність, покращуючи ефективність і швидкість. Зокрема, сімейство YOLO11 виявилося найбільш послідовним, причому, YOLO11m досягає оптимального балансу між точністю, ефективністю та розміром моделі (рис. 1.9).

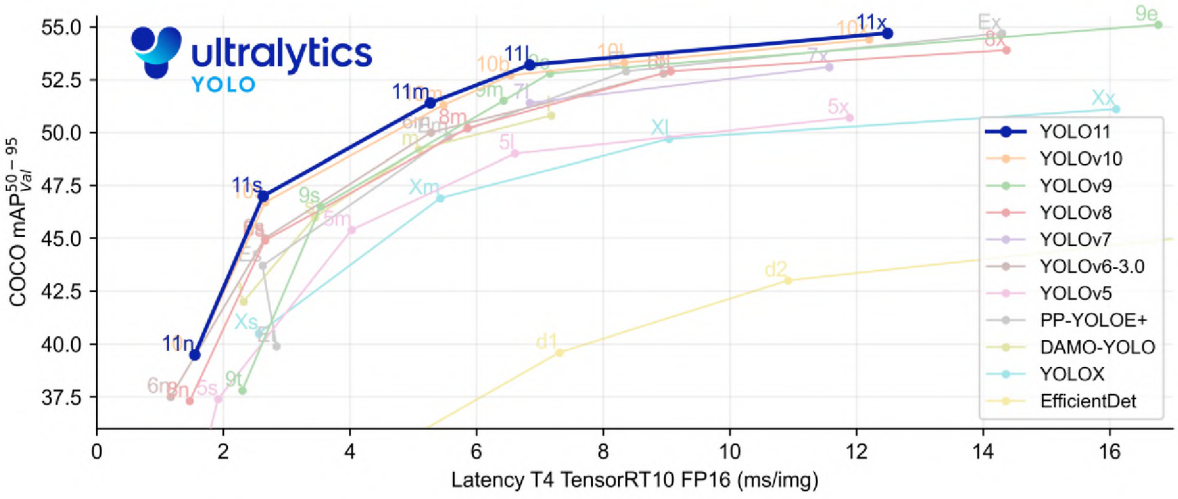

Рисунок 1.8 – Порівняння продуктивності архітектур нейронних мереж

Виявлення	Сегментація екземплярів	Поза/ключові точки	Класифікація	Орієнтоване виявлення
YOLO11n	YOLO11n-seg	YOLO11n-pose	YOLO11n-cls	YOLO11n-obb
YOLO11s	YOLO11s-seg	YOLO11s-pose	YOLO11s-cls	YOLO11s-obb
YOLO11m	YOLO11m-seg	YOLO11m-pose	YOLO11m-cls	YOLO11m-obb
YOLO11l	YOLO11l-seg	YOLO11l-pose	YOLO11l-cls	YOLO11l-obb
YOLO11x	YOLO11x-seg	YOLO11x-pose	YOLO11x-cls	YOLO11x-obb

Рисунок 1.9 – Сімейство YOLO11

Тут також варто звернути увагу на наявність орієнтованого виявлення. Воне йде на крок далі, ніж виявлення об'єктів, і вводить додатковий кут, щоб точніше знаходити об'єкти на зображенні. Результатом роботи орієнтованого

детектора об'єктів є поворотний набір обмежувальних боксів, які точно оточують об'єкти на зображенні, а також мітки класів і бали довірі для кожного боксу (рис. 1.10).

Рисунок 1.10 – Приклад використання OBB

Хоча YOLOv10 забезпечував дещо нижчу точність, ніж YOLO11, він вирізнявся швидкістю та ефективністю, що робить його сильним вибором для додатків, які вимагають ефективності та швидкої обробки. Крім того, YOLOv9 показав хороші результати в цілому і особливо виділявся в менших наборах даних [32, 33]. Ці висновки надають цінну інформацію для промисловості та наукових кіл, керуючи вибором найбільш підходящих алгоритмів YOLO та інформуючи про майбутні розробки та вдосконалення. Хоча розглянуті алгоритми демонструють багатообіцяючу продуктивність, все ще є місце для вдосконалення. Майбутні дослідження можуть бути зосереджені на оптимізації YOLOv10, щоб підвищити його точність, зберігаючи його переваги у швидкості та ефективності. Крім того, постійний прогрес в архітектурному дизайні може прокласти шлях для ще більш новаторських алгоритмів YOLO.

В цілому, YOLO11 виділяється кількома властивостями. Дана версія використовує сильні сторони своїх попередників і посилює їх, запроваджуючи кілька ключових удосконалень. Модель використовує актуальну архітектуру, яка чудово вловлює складні деталі навіть у складних сценаріях. Це покращення означає, що YOLO11 може виявляти менші

об’єкти та тонкі деталі з більшою точністю. 11-та версія досягає вищих середніх показників середньої точності (mAP) на стандартних наборах даних (COCO), використовуючи менше параметрів. Зокрема, YOLO11m використовує на 22 % менше параметрів, ніж YOLOv8m, але забезпечує гарну продуктивність. Ця ефективність робить цей алгоритм легшим і швидшим без шкоди для результатів. У додатках у реальному часі швидкість має першорядне значення. YOLO11 обробляє дані приблизно на 2 % швидше, ніж YOLOv10, завдяки оптимізованому часу висновку. Це підвищення підвищує продуктивність у чутливих до часу середовищах, від автономного водіння до аналітики відео в реальному часі. YOLO11 розроблено для розгортання на різних платформах, як на потужних хмарних серверах, так і на периферійних пристроях з обмеженими ресурсами. Його адаптивність гарантує, що він може задовольнити вимоги різноманітних галузей промисловості та застосувань. Процес навчання отримує переваги від покращеного конвеєра доповнення, що дозволяє моделі краще узагальнювати різні завдання та набори даних. Це вдосконалення спрощує адаптацію YOLO11 як до невеликих проектів, так і до великих, складних програм. Розуміючи важливість гнучкості, YOLO11 створено для легкої інтеграції з існуючими системами та платформами (з GPU NVIDIA, периферійними пристроями та основними хмарними платформами).

Сучасні версії YOLO (наприклад, YOLOv8-11) забезпечують високу точність при детекції об'єктів навіть на складних зображеннях. Це дозволяє:

- розрізняти продукцію за категоріями;
- визначати дефекти чи шлюб;
- виявляти нестандартні форми або пошкодження.

YOLO підтримує мультиоб'єктну детекцію. Це дозволяє одночасно: сортувати продукцію за кількома категоріями; обробляти продукцію різного розміру та форми; враховувати розташування об'єктів на конвеєрі для подальшої маршрутизації. YOLO може бути навчена для детекції об'єктів будь-якої природи.

У контексті сортування продукції це означає, що нейронну мережу можна налаштувати на розпізнавання різних типів фруктів, овочів, упаковок тощо; виявлення нестандартних об'єктів чи забруднень; контроль якості продукції (рис. 1.11).

Рисунок 1.11 – Приклад використання «CV+AI» для сортування

Сучасні продажі YOLO оптимізовані для роботи на пристроях з обмеженими обчислювальними ресурсами, таких як GPU або навіть одноплатні комп'ютери (наприклад, Raspberry Pi). Це знижує вартість застосування системи та спрощує експлуатацію.

YOLO можна легко інтегрувати в існуючі системи, використовуючи камери для відеоспостереження та стандартні програмні середовища (Python, C++, TensorFlow, PyTorch) [34, 35]. Це знижує витрати на розробку та прискорює впровадження. Як приклад, на конвеєрі сортування фруктів система з використанням YOLO може: визначати тип фрукта (наприклад, яблуко, груша, апельсин); виявляти пошкоджені або дефектні екземпляри; надсилати продукцію у відповідні контейнери на основі категорії. Таким чином, використання YOLO у системах сортування продукції на конвеєрі

виправдане завдяки її високій продуктивності, точності, компактності та універсальності. Вона дозволяє автоматизувати процеси, мінімізувати людську працю та підвищувати якість продукції.

Висновки до розділу 1

Застосування алгоритмів ML, зокрема нейронних мереж, у системах сортування продукції на конвеєрах є важливим кроком у напрямку автоматизації виробничих процесів. Завдяки використанню технологій CV, CNN та ін. методів класифікації і кластеризації, забезпечується значне підвищення точності та швидкості сортування, зниження кількості дефектної продукції, а також оптимізація витрат.

Ці досягнення підкреслюють важливість інтеграції AI в сучасні промислові процеси, сприяючи інноваціям та підвищенню конкурентоспроможності підприємств.

Застосування AI в системах сортування продукції на конвеєрах відкриває широкі можливості для підвищення ефективності, зниження витрат і покращення якості виробничих процесів. Аналіз основних напрямів використання AI свідчить, що технології CV є ключовим компонентом таких систем, забезпечуючи автоматичне розпізнавання та класифікацію продукції за розміром, формою, кольором чи текстурою. Використання високошвидкісних камер, мультиспектральної та гіперспектральної зйомки дозволяє ідентифікувати навіть дрібні дефекти продукції.

Додатково, предиктивна аналітика на основі AI дозволяє прогнозувати можливі вузькі місця у процесі сортування та оперативно їх усувати, що підвищує загальну продуктивність. Інтеграція систем сортування з пристроями IoT забезпечує моніторинг стану обладнання, що сприяє зменшенню простоїв і забезпеченню безперебійної роботи. Впровадження AI в системи сортування продукції дозволяє оптимізувати процеси виробництва,

забезпечити гнучкість і масштабованість рішень, а також досягти високої точності й швидкості виконання завдань у режимі реального часу. Особливої уваги заслуговують алгоритми YOLO, які демонструють високу продуктивність у реальному часі, що є критичним для швидких виробничих ліній. Впровадження таких технологій дозволяє не лише підвищити ефективність, але й адаптувати системи до змін у виробництві, зберігаючи гнучкість і масштабованість. Аналіз архітектур нейронних мереж для сортування продукції на основі AI демонструє високу ефективність використання алгоритмів YOLO. Цей підхід забезпечує швидкість і точність, що є критично важливими для реального часу у виробничих середовищах. Різні версії YOLO, включаючи новітні архітектури YOLOv8 та YOLO11, показують здатність адаптуватися до складних умов аналізу зображень, таких як семантична сегментація, класифікація та виявлення орієнтованих об'єктів.

Еволюція алгоритмів YOLO підкреслює прогрес у скороченні параметрів моделей і підвищенні швидкості роботи без втрати точності. Особливо виділяється YOLO11, яка інтегрує новітні технології обробки, такі як модулі самоуваги (C2PSA) та оптимізовані блоки (C3k2), що покращує виявлення дрібних і складних об'єктів.

Використання YOLO спрощує інтеграцію AI в існуючі виробничі системи, дозволяє працювати з обмеженими ресурсами (наприклад, на базі Raspberry Pi) та знижує вартість впровадження технологій. Завдяки цьому, YOLO забезпечує універсальність у розв'язанні широкого спектра задач, таких як контроль якості, сортування продукції за категоріями, виявлення дефектів і забруднень. Це підтверджує доцільність застосування сучасних архітектур YOLO для автоматизації виробничих процесів, що сприяє підвищенню продуктивності, скороченню витрат і забезпеченню конкурентоспроможності підприємств.

РОЗДІЛ 2
РОЗРОБКА МОДЕЛІ ГЛИБОКОГО НАВЧАННЯ НЕЙРОННОЇ МЕРЕЖІ КЛАСИФІКАЦІЇ ЯКОСТІ АГРОПРОДУКЦІЇ

2.1 Дослідження конвеєру обробки зображення YOLO11

Згідно п. 1.3, YOLO11 від к. Ultralytics поставляється з надлегкими моделями, які набагато швидші та ефективніші, ніж попередні YOLO, які здатний виконувати більш широкий спектр завдань з CV. YOLO11 побудований на основі кодової бази Ultralytics YOLOv8 з деякими архітектурними модифікаціями. Він також інтегрує нові функції (удосконалюючи ці функції) з попередніх YOLO (таких як YOLOv9 і YOLOv10) для покращення продуктивності. Серія YOLO11 є найсучаснішою (SOTA), найлегшою та найефективнішою моделлю в сімействі YOLO, яка перевершує своїх попередників. Алгоритм був створений Ultralytics, яка випустила YOLOv8, найстабільніший і широко використовуваний варіант YOLO на даний момент. YOLO, в основному, відома своїми моделями виявлення об'єктів. Однак YOLO11 може виконувати кілька завдань CV, таких як YOLOv8 (виявлення об'єктів, сегментація екземплярів, класифікація зображень, оцінка пози, виявлення орієнтованих об'єктів (OBB)). Ultralytics випустила 5 моделей YOLO11 за розміром і 25 моделей для всіх завдань:

- YOLO11n – Nano для невеликих і легких завдань;
- YOLO11s – невелике оновлення Nano з додатковою точністю;
- YOLO11m – середовище для загального використання;
- YOLO11l– великий для вищої точності з вищими обчисленнями;
- YOLO11x– надзвичайно великий для максимальної точності та продуктивності.

Надалі розглянемо наскрізний процес загального конвеєра виявлення об’єктів, який використовується в моделях YOLO. Процес починається з попередньої обробки зображення, де зображення змінюється до фіксованого

розміру (наприклад, 640×640 пікселів) відповідно до вхідних специфікацій моделі YOLO, нормалізується відповідно до шкали значень пікселів, яка використовується під час навчання, і перетворюється на тензорний формат для обробки моделі.

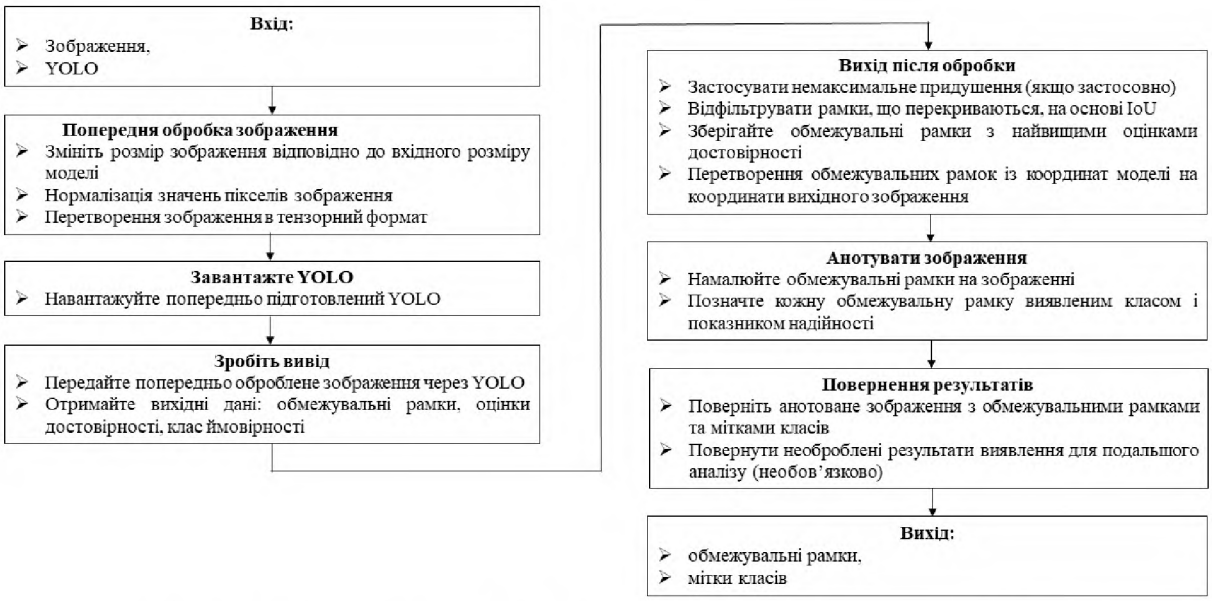

Рисунок 2.1 – Конвеєр обробки зображення алгоритму YOLO11

Потім попередньо навчена модель YOLO завантажується в пам'ять, використовуючи її навчання на великих наборах даних для виявлення різних об'єктів. Попередньо оброблене зображення потім пропускається через модель, створюючи необроблені результати, які включають обмежувальні прямокутники, показники достовірності та ймовірності класу. На етапі постобробки застосовується немаксимальне придушення (NMS) [36], щоб відфільтрувати надлишкові обмежувальні рамки, зберігаючи лише ті з найвищими балами достовірності, тоді як координати обмежувальної рамки перетворюються назад до вихідного розміру зображення для точної анотації. Потім зображення анотується обмежувальними рамками та мітками класів для візуалізації виявлених об'єктів, а кінцевий результат, який включає анотований образ і необроблені результати виявлення, повертається для візуалізації або подальшого аналізу. Увесь процес розроблений таким чином,

щоб бути ефективним і гарантувати можливе виявлення в реальному часі, особливо при розгортанні систем сортування продукції на лініях промислового конвеєру. Відповідно, деталізуємо його до завдань CV.

Виявлення об'єктів (рис. 2.2) [37]. YOLO11 виконує виявлення об'єктів, передаючи вхідне зображення в CNN для вилучення об'єктів. Потім мережа прогнозує обмежувальні рамки та ймовірності класів для об'єктів у межах цих сіток. Для обробки багатомасштабного виявлення використовуються шари, які забезпечують виявлення об'єктів різного розміру [38]. Потім ці прогнози підлягають уточненню за допомогою немаксимального придушення (NMS) для фільтрації дублікатів або коробок з низькою впевненістю, що призводить до більш точного виявлення об'єктів. YOLO11 тренується на наборі даних MS-COCO [32] для виявлення об'єктів, який включає 80 попередньо підготовлених класів.

Рисунок 2.2 – Виявлення об'єктів YOLO11

Сегментація екземплярів (рис. 2.3). Крім виявлення об'єктів, YOLO11 розширюється до сегментації екземплярів, додаючи гілку прогнозування маски. Ці моделі тренуються на наборі даних MS-COCO, який включає 80 попередньо навчених класів. Ця гілка генерує маски сегментації за пікселями для кожного виявленого об'єкта, що дозволяє моделі розрізняти об'єкти, що перекриваються, і надавати точні контури їх форм. Гілка маски в головній

частині обробляє функції відображення та виведення масок об'єктів, забезпечуючи точність на рівні пікселів у розпізнаванні та диференціюванні об'єктів на зображенні.

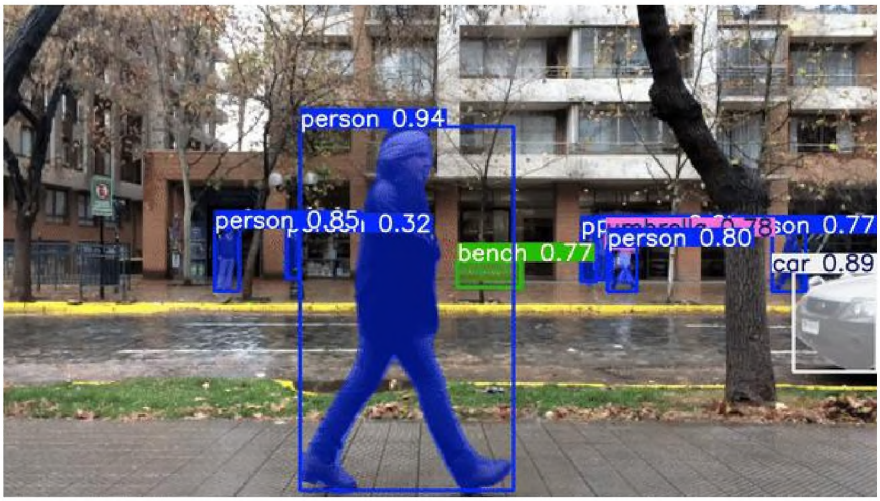

Рисунок 2.3 – Сегментація екземплярів

Рисунок 2.4 – Оцінка пози

Оцінка пози (рис. 2.4). YOLO11 виконує оцінку пози, виявляючи та прогнозуючи ключові точки на об'єкті, такі як суглоби в тілі людини. Ключові точки з'єднуються, утворюючи структуру скелета, яка представляє позу. Ці моделі тренуються на COCO, який включає в себе один попередньо навчений клас – «людина». Шари оцінки пози додаються в голові, а мережа навчена передбачати координати ключових точок. Етап постобробки з'єднує точки, щоб сформувати структуру скелета, що дозволяє розпізнавати позу в режимі реального часу.

Класифікація зображень (рис. 2.5). Для класифікації зображень YOLO11 використовує свою глибоку нейронну мережу для вилучення високорівневих функцій із вхідного зображення та призначення його одній із кількох попередньо визначених категорій. Ці моделі тренуються на платформі ImageNet [33], яка включає в себе 1000 попередньо підготовлених класів. Мережа обробляє зображення через кілька шарів звивин і об'єднання, зменшуючи просторові розміри при одночасному посиленні основних функцій. Класифікаційна головка у верхній частині мережі виводить прогнозований клас, що робить її придатною для завдань, де потрібно визначити загальну категорію зображення.

Рисунок 2.5 – Класифікація зображень

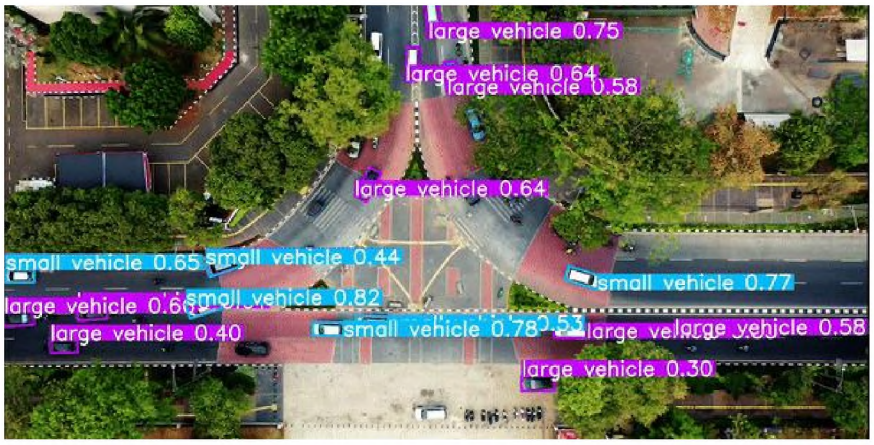

Рисунок 2.6 – Виявлення орієнтованих об'єктів

Виявлення орієнтованих об'єктів (OBB) – рис. 2.6. YOLO11 розширює можливість регулярного виявлення об'єктів за рахунок включення OBB, що

дозволяє моделі виявляти та класифікувати об'єкти, які обертаються або мають неправильну орієнтацію [39]. Це особливо корисно для таких програм, як аналіз аерофотознімків та ін. Ці моделі тренуються на DOTAv1, що включає 15 попередньо підготовлених класів. Модель OBB виводить не тільки координати обмежувальної коробки, але й кут повороту (θ) або чотири кутові точки (рис.2.7).

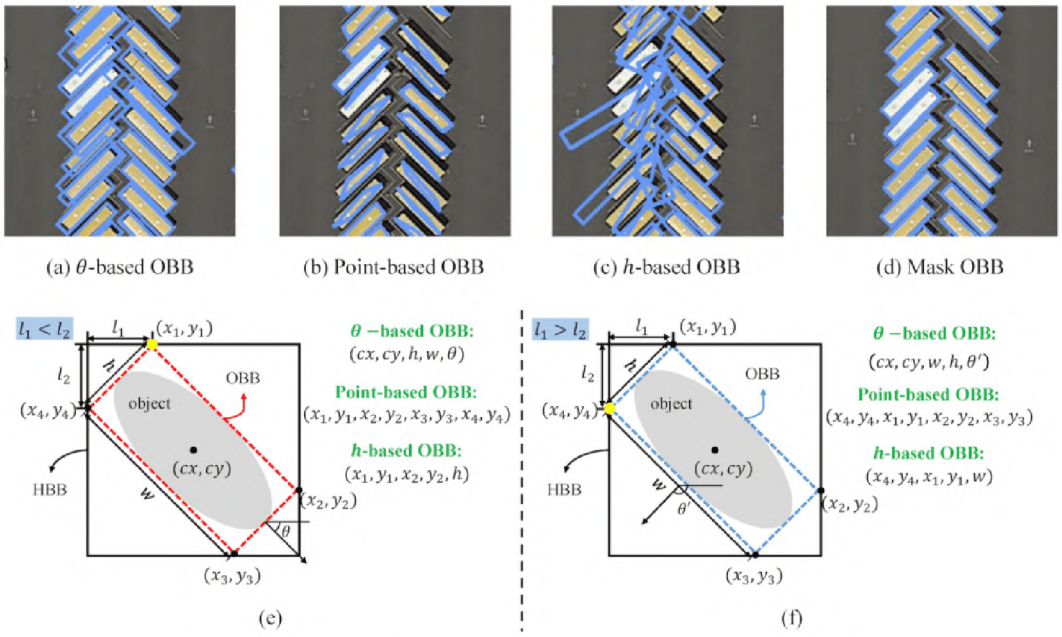

Рисунок 2.7 – параметри виводу моделі OBB

Ці координати використовуються для створення обмежувальних рамок, які вирівнюються з орієнтацією об'єкта, підвищуючи точність виявлення об'єктів, що обертаються.

2.2 Дослідження архітектури та структури коду YOLO11

Архітектура YOLO11 (див. рис. 1.7) – це оновлення в порівнянні з архітектурою YOLOv8 з деякими новими інтеграціями та налаштуванням параметрів. Якщо подивитись на конфігураційний файл YOLO11 [36], то можливо визначати введені зміни у порівнянні з YOLOv8 (рис. 2.8).

По-перше розглянемо Backbone (зустрічаються синоніми: магістраль, каркас, основа). Це основна частина нейронної мережі, яка відповідає за вилучення ознак із зображення. Вона виконує функції обробки та перетворення вхідних даних (наприклад, зображення) у високорівневі уявлення (feature maps), які потім використовуються іншими компонентами моделі, такими як голова (head) або шия (neck).

```
# Parameters
nc: 80 # number of classes
scales: # model compound scaling constants, i.e. 'model=yolo11n.yaml' will call yolo11.yaml with
scale 'n'
  # [depth, width, max_channels]
  n: [0.50, 0.25, 1024] # summary: 319 layers, 2624080 parameters, 2624064 gradients, 6.6 GFLOPs
  s: [0.50, 0.50, 1024] # summary: 319 layers, 9458752 parameters, 9458736 gradients, 21.7 GFLOPs
  m: [0.50, 1.00, 512] # summary: 409 layers, 20114688 parameters, 20114672 gradients, 68.5
GFLOPs
  l: [1.00, 1.00, 512] # summary: 631 layers, 25372160 parameters, 25372144 gradients, 87.6
GFLOPs
  x: [1.00, 1.50, 512] # summary: 631 layers, 56966176 parameters, 56966160 gradients, 196.0
GFLOPs

# YOLO11n backbone
backbone:
  # [from, repeats, module, args]
  - [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
  - [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
  - [-1, 2, C3k2, [256, False, 0.25]]
  - [-1, 1, Conv, [256, 3, 2]] # 3-P3/8
  - [-1, 2, C3k2, [512, False, 0.25]]
  - [-1, 1, Conv, [512, 3, 2]] # 5-P4/16
  - [-1, 2, C3k2, [512, True]]
  - [-1, 1, Conv, [1024, 3, 2]] # 7-P5/32
  - [-1, 2, C3k2, [1024, True]]
  - [-1, 1, SPPF, [1024, 5]] # 9
  - [-1, 2, C2PSA, [1024]] # 10

# YOLO11n head
head:
  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 6], 1, Concat, [1]] # cat backbone P4
  - [-1, 2, C3k2, [512, False]] # 13

  - [-1, 1, nn.Upsample, [None, 2, "nearest"]]
  - [[-1, 4], 1, Concat, [1]] # cat backbone P3
  - [-1, 2, C3k2, [256, False]] # 16 (P3/8-small)

  - [-1, 1, Conv, [256, 3, 2]]
  - [[-1, 13], 1, Concat, [1]] # cat head P4
  - [-1, 2, C3k2, [512, False]] # 19 (P4/16-medium)

  - [-1, 1, Conv, [512, 3, 2]]
  - [[-1, 10], 1, Concat, [1]] # cat head P5
  - [-1, 2, C3k2, [1024, True]] # 22 (P5/32-large)

  - [[16, 19, 22], 1, Detect, [nc]] # Detect(P3, P4, P5)
```

Рисунок 2.8 – Файл конфігурації YOLO11

Для YOLO11 вона витягує особливості з вхідного зображення в декількох масштабах. Зазвичай, вона включає в себе накопичення згорткових шарів (Convolutional Layers) і блоків для створення карт функцій з різною роздільною здатністю. В Backbone YOLO11 Convolutional Layers мають

аналогічну структуру за YOLOv8 з початковими шарами згортки для зменшення дискретизації зображення:

```
- [-1, 1, Conv, [64, 3, 2]] # 0-P1/2
- [-1, 1, Conv, [128, 3, 2]] # 1-P2/4
```

Замість блоку C2f (рис. 2.9) в YOLO11 представлений блок C3k2, який є більш ефективним з точки зору обчислень. Коли параметр c3k вимкнено, модуль зберігає свою оригінальну структуру типу bottleneck, функціонуючи як попередник. Однак при активації параметра модуль перетворюється на просунуту конфігурацію, засновану на модулі C3, забезпечуючи поліпшені можливості обробки ознак.

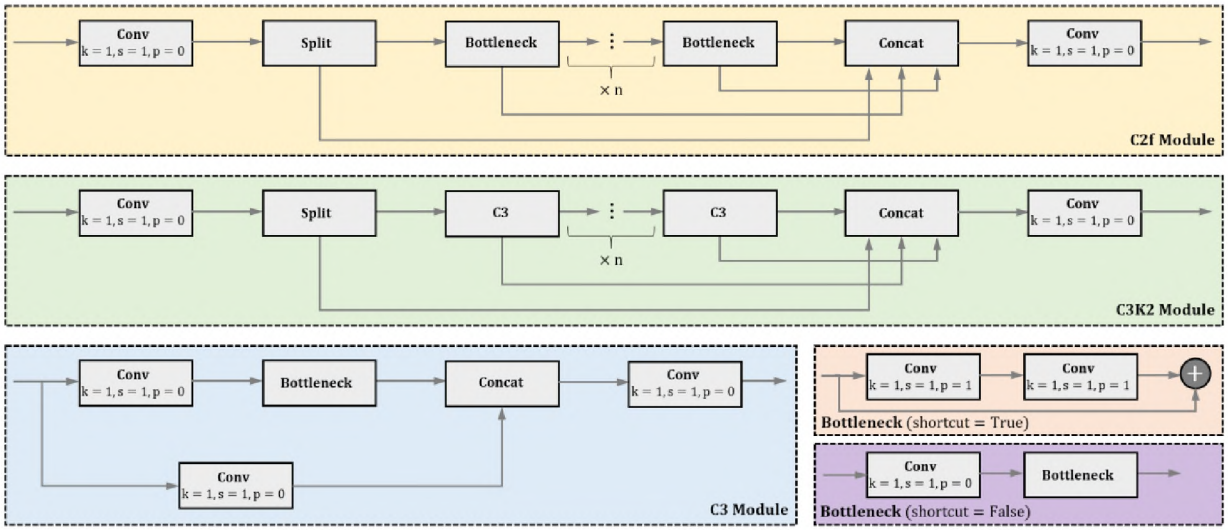

Рисунок 2.9 – Деталізована архітектура модуля C3K2 з акцентом на параметр c3k і його структурну адаптацію від модуля C2F в YOLOv8 для поліпшеної обробки ознак

C3k2 є користувацькою реалізацією CSP Bottleneck, яка використовує дві згортки замість однієї великої згортки як у YOLOv8. CSP (перехресна часткова стадія) – мережі CSP розбивають карту об'єктів і обробляють одну частину через шар вузького місця, одночасно об'єднуючи іншу частину з виходом вузького місця. Це зменшує обчислювальне навантаження та покращує представлення функцій:

```
- [-1, 2, C3k2, [256, False, 0.25]]
```

Блок C3k2 також використовує менший розмір ядра (позначається як k2), що робить його швидшим, зберігаючи продуктивність.

YOLO11 зберігає блок SPPF, але додає новий блок C2PSA після SPPF:

```
- [-1, 1, SPPF, [1024, 5]]
- [-1, 2, C2PSA, [1024]
```

В даному випадку, блок C2PSA (Cross Stage Partial with Spatial Attention) посилює просторову увагу на картах об'єктів, покращуючи фокус моделі на важливих частинах зображення. Це дає моделі можливість більш ефективно зосереджуватися на конкретних областях інтересів, просторово об'єднуючи функції.

Надалі розглянемо Neck (шия). Ця частина відповідає за агрегування ознак з різною роздільною здатністю та передачу їх голові (Head) для прогнозування. Зазвичай, вона включає збільшення вибірки та конкатенацію карт об'єктів з різних рівнів. В Neck YOLO11 замінений блок C2f на C3K2. Як обговорювалося раніше, C3k2 є більш швидким і ефективним блоком. Наприклад, після апсемплінгу та конкатенації шия у YOLO11 виглядає так: `- [-1, 2, C3k2, [512, False]] # P4/16-medium`.

Ця зміна покращує швидкість і продуктивність процесу агрегації функцій. В YOLO11 механізм уваги більше фокусується на просторовій увазі за допомогою C2PSA, що допомагає моделі зосередитися на ключових областях зображення для кращого виявлення. Цього немає в YOLOv8, що робить YOLO11 потенційно більш точною у виявленні менших або закритих об'єктів.

Надалі розглянемо Head (голова). Це частина моделі, яка відповідає за генерацію остаточних прогнозів. У виявленні об'єктів це, зазвичай, означає створення обмежувальних рамок (Bounding Boxes) і класифікацію об'єктів усередині цих рамок. Блок Аналогічно шиї, YOLO11 замінює блок C2f в голові на C3K2: `- [-1, 2, C3k2, [512, False]] # P4/16-medium`.

Кінцевий шар Detect (детектування, виявлення) такий самий, як і в початковій архітектурі – YOLOv8: `- [[16, 19, 22], 1, Detect, [nc]] # Detect(P3, P4, P5)`. Використання блоків C3k2 робить модель більш

швидкою з точки зору висновків і більш ефективною за параметрами. Отже, подивимося, як виглядають нові блоки (шари) в коді. C3k2 є швидшим та ефективнішим варіантом вузького місця CSP. У ньому використовуються дві згортки замість однієї великої згортки, що прискорює вилучення ознак.

```
class C3k2(C2f):
    def __init__(self, c1, c2, n=1, c3k=False, e=0.5, g=1,
shortcut=True):
        super().__init__(c1, c2, n, shortcut, g, e)
        self.m = nn.ModuleList(
            C3k(self.c, self.c, 2, shortcut, g) if c3k else
Bottleneck(self.c, self.c, shortcut, g) for _ in range(n)
        )
```

Блок C3k (від blocks.py) – це більш гнучкий модуль вузького місця, який дозволяє налаштовувати розміри ядра. Це корисно для отримання більш детальних об'єктів на зображеннях:

```
class C3k(C3):
    def __init__(self, c1, c2, n=1, shortcut=True, g=1, e=0.5, k=3):
        super().__init__(c1, c2, n, shortcut, g, e)
        c_ = int(c2 * e)  # hidden channels
        self.m = nn.Sequential(*(Bottleneck(c_, c_, shortcut, g, k=(k,
k), e=1.0) for _ in range(n)))
```

Блок C2PSA (Cross Stage Partial with Spatial Attention) з blocks.py підвищує здатність моделі до просторової уваги. Цей блок додає уваги до карт об'єктів, допомагаючи моделі зосередитися на важливих областях зображення:

```
class C2PSA(nn.Module):
    def __init__(self, c1, c2, e=0.5):
        super().__init__()
        c_ = int(c2 * e)
        self.cv1 = Conv(c1, c_, 1, 1)
        self.cv2 = Conv(c1, c_, 1, 1)
        self.cv3 = Conv(2 * c_, c2, 1)

    def forward(self, x):
        return self.cv3(torch.cat((self.cv1(x), self.cv2(x)), 1))
```

Надалі розглянемо основну частину кодової бази YOLO11, до якої слід віднести файл tasks.py та відповідні модулі. Останні визначають різні компоненти нейронної мережі, що використовуються в моделі YOLO11. Ці компоненти організовані в різні файли в директорії nn/modules/: block.py – визначає різні будівельні блоки (модулі), що використовуються в моделі, такі як вузькі місця, модулі CSP та механізми уваги; conv.py – містить згорткові модулі, включаючи стандартні згортки, згортки по глибині та інші варіації;

head.py – реалізує головку моделі, відповідальну за створення остаточних прогнозів (наприклад, обмежувальні рамки, ймовірності класу); transformer.py – включає модулі на основі трансформаторів, які використовуються для механізмів уваги та розширеного вилучення функцій; utils.py – надає допоміжні функції та допоміжні класи, що використовуються в усіх модулях.

Файл nn/tasks.py визначає різні моделі для конкретного завдання (наприклад, виявлення, сегментація, класифікація), які об'єднують ці модулі для формування повних архітектур. Скрипт файлу tasks.py (Додаток А) є основною частиною конвеєра коду і він використовує методи та логіку YOLOv8. Цей скрипт призначений для різних завдань CV, таких як виявлення об'єктів, сегментація, класифікація, оцінка пози, OBB. Він визначає базові моделі, моделі для конкретного завдання та корисні функції для навчання, висновків та управління моделями. Скрипт імпортує основні модулі, такі як PyTorch, шари нейронної мережі (torch.nn) та утиліти з Ultralytics.

Деякі ключові імпортні компоненти містять:

– блоки архітектури моделі: C3k2, C2PSA, C3, SPPF, Concat та ін.;

– функції втрати такі як v8DetectionLoss, v8SegmentationLoss, v8ClassificationLoss, v8OBBLoss;

– різноманітні утиліті функції, такі як model_info, fuse_conv_and_bn, scale_img, time_sync для допомоги в обробці моделі, профілюванні та оцінці.

Модель передбачає два базові класи: BaseModel та DetectionModel. По суті, цей клас призначений для розширення моделями, специфічними для конкретного завдання, наприклад, виявленням, сегментацією та класифікацією.

Безпосередньо клас BaseModel служить базовим класом для всіх моделей сімейства Ultralytics YOLO і реалізує фундаментальні методи:

– forward() обробляє навчання та вивід залежно від вхідних даних;

– predict() обробляє прямий прохід для виводу;

– fuse() – об'єднує шари Conv2d та BatchNorm2d для підвищення ефективності;

– info()надає детальну інформацію про модель.

Клас DetectionModel розширює BaseModel спеціально для завдань виявлення об'єктів. Він завантажує конфігурації моделей, ініціалізує голови виявлення (наприклад, блок Detect) і встановлює кроки моделі. Він підтримує завдання виявлення з використанням таких архітектур, як YOLOv8, і може виконувати розширене висновування через _predict_augment().

При цьому, код містить кілька моделей, що призначені для виконання конкретного завдання. Модель сегментації ініціалізує специфічні для сегментації функції втрат (v8SegmentationLoss) та підклас DetectionModel, що спеціалізується на завданнях сегментації (наприклад, сегментація YOLOv8). Модель положення вирішує завдання оцінювання поз, ініціалізуючи моделі зі специфічними конфігураціями для виявлення ключових точок (kpt_shape), використовує v8PoseLoss для розрахунку втрат для конкретної пози. Модель класифікації призначена для завдань класифікації зображень з використанням архітектури класифікації YOLOv8 та ініціалізує і керує специфічними для класифікації втратами (v8ClassificationLoss). Також підтримує зміну форми попередньо навчених моделей TorchVision для завдань класифікації. Модель OBB використовується для завдань виявлення орієнтованої обмежувальної рамки (OBB) і реалізує специфічні функції втрати (v8OBBLoss) для обробки обернених обмежувальних рамок. Модель світу справляється з такими завданнями, як підписи до зображень і розпізнавання тексту. Вона використовує функції тексту з моделей CLIP для завдань візуального розпізнавання на основі тексту та включає спеціальну обробку для вбудовування тексту (txt_feats) для створення субтитрів і завдань, пов'язаних зі світом. Модель ансамблю дозволяє усереднювати або об'єднувати виходи з різних моделей для покращення загальної продуктивності (ансамбль – простий ансамблевий клас, який об'єднує кілька моделей в одну.). Це корисно для завдань, де комбінація виходів з кількох моделей забезпечує

кращі прогнози. Таким чином, для створення моделі під конкретне контекстне завдання необхідні наступні процедури. Застосування parse_model() – аналізує конфігурацію моделі з файлу YAML у модель PyTorch. Це забезпечує обробку архітектури магістралі та голови, інтерпретує кожен тип шару (наприклад, Conv, SPPF, Detect) і будує остаточну модель. Вона підтримує різні архітектури, включаючи компоненти YOLO11, такі як C3k2, C2PSA та ін. Завантаження моделі YAML: yaml_model_load() – завантажує конфігурацію моделі з файлу YAML, виявляючи масштаб моделі (наприклад, n, s, m, l, x) і відповідним чином коригуючи параметри; guess_model_scale() та guess_model_task() – допоміжні функції для визначення масштабу моделі та завдання на основі файлової структури YAML. При цьому підтримуються наступні функції. Завантаження та управління моделлю: attempt_load_weights(), attempt_load_one_weight() – для завантаження моделей, керування ансамблевими моделями та вирішення проблем сумісності під час завантаження попередньо натренованих ваг. Ці функції гарантують, що моделі завантажуються правильно з правильними кроками, шарами та конфігураціями. Тимчасове перенаправлення модуля – temporary_modules(). Контекстний менеджер для тимчасового перенаправлення шляхів модулів, забезпечуючи зворотну сумісність при зміні розташування модулів. Це корисно для підтримки сумісності зі старішими версіями моделей. Обробка безпеки SafeUnpickler – клас користувача для безпечного завантаження контрольних точок моделі (checkpoints). Він замінює невідомі класи безпечними заповнювачами (SafeClass) для запобігання збоям під час завантаження даних.

2.3 Деталізація оновлення модулів YOLO11

Надалі варто провести дослідження ключові особливості оновлення модулів YOLO11 у порівнянні з YOLOv8: block.py, head.py, conv.py.

Файл block.py визначає різні будівельні блоки, що використовуються в моделі YOLO11. Ці блоки є важливими компонентами, які формують шари нейронної мережі.

1. Модуль Bottleneck (вузловий модуль або модуль вузькового місця) у нейронних мережах використовуються для зменшення обчислювальних витрат та підвищення ефективності. Такі модулі знижують кількість параметрів, пропускаючи інформацію через стислі уявлення (зазвичай із зменшеним кількістю каналів), та був відновлюють її розмір. Такі модулі відіграють ключову роль оптимізації глибоких мереж, зберігаючи при цьому здатність отримувати важливі ознаки. Модулі Bottleneck часто включають комбінації згорткових шарів, операцій нормалізації та активації, що робить їх легкими та швидкими для обчислень: безпосередньо Bottleneck – стандартний модуль вузького місця з додатковими з'єднаннями швидкого доступу; Res – залишковий блок, який використовує серію який використовує серію згорток та ідентичний прямий прохід (Forward Pass). У класі Bottleneck реалізований модуль bottleneck, який зменшує кількість каналів (зменшення розмірності), а потім знову їх розширює. Тут реалізовані такі складові:

– self.cv1 – Згортка 1×1, яка зменшує кількість каналів;

– self.cv2 – Згортка 3×3, яка збільшує кількість каналів назад до початкового;

– self.add – логічний символ, що вказує, чи слід додавати з'єднання ярликів;

– Forward Pass – вхід X пропускається через cv1 і cv2 (якщо self.add має значення True, вихідний вхід X додається до виходу (залишкове з'єднання)).

Сам код модулю виглядає наступним чином:

```
class Bottleneck(nn.Module):
    def __init__(self, c1, c2, shortcut=True, g=1, e=0.5):
        super().__init__()
        c_ = int(c2 * e)
        self.cv1 = Conv(c1, c_, 1, 1)
        self.cv2 = Conv(c_, c2, 3, 1, g=g)
        self.add = shortcut and c1 == c2
    def forward(self, x):
```

```
        return x + self.cv2(self.cv1(x)) if self.add else
self.cv2(self.cv1(x))
```

2. Модулі CSP (Cross Stage Partial) містить два варіанти. BottleneckCSP – це CSP версія модуля bottleneck. Модуль CSPBottleneck ділить карту об'єктів на дві частини. Одна частина проходить через серію шарів вузьких місць, а інша частина підключається безпосередньо до виходу, знижуючи обчислювальні витрати та покращуючи градієнтний потік. CSPBlock більш складний модуль CSP з кількома шарами вузьких місць. Тут використовуються складові:

– self.cv1 – зменшує кількість каналів;

– self.cv2 – послідовність шарів вузького місця;

– self.cv3 – поєднує функції та регулює кількість каналів;

– self.add – визначає, чи додано з'єднання ярликів.

Код має такий вигляд:

```
class BottleneckCSP(nn.Module):
    def __init__(self, c1, c2, n=1, shortcut=True, g=1, e=0.5):
        super().__init__()
        c_ = int(c2 * e)
        self.cv1 = Conv(c1, c_, 1, 1)
        self.cv2 = nn.Sequential(
            *[Bottleneck(c_, c_, shortcut, g, e=1.0) for _ in range(n)]
        )
        self.cv3 = Conv(2 * c_, c2, 1)
        self.add = c1 == c2

    def forward(self, x):
        y1 = self.cv2(self.cv1(x))
        y2 = x if self.add else None
             return self.cv3(torch.cat((y1, y2), 1)) if y2 is not None else
self.cv3(y1)
```

Також сюди відноситься SPPF (Spatial Pyramid Pooling Fast) – виконує об'єднання в різних масштабах, а також Concat – об'єднує кілька тензорів уздовж заданої розмірності. Модуль SPPF виконує максимальне об'єднання в різних масштабах і об'єднує результати для фіксації об'єктів у кількох просторових масштабах. В даному випадку, використовуються такі складові:

– self.cv1 зменшує кількість каналів;

– self.cv2 регулює кількість каналів після об'єднання;

– self.m максимальний шар об'єднання.

Forward Pass: вхід x передається через cv1, а потім через три послідовні максимальні шари об'єднання (y1, y2, y3). Результати об'єднуються і передаються через cv2:

```
class SPPF(nn.Module):
    def __init__(self, c1, c2, k=5):
        super().__init__()
        c_ = c1 // 2
        self.cv1 = Conv(c1, c_, 1, 1)
        self.cv2 = Conv(c_ * 4, c2, 1, 1)
        self.m = nn.MaxPool2d(kernel_size=k, stride=1, padding=k // 2)

    def forward(self, x):
        x = self.cv1(x)
        y1 = self.m(x)
        y2 = self.m(y1)
        y3 = self.m(y2)
        return self.cv2(torch.cat([x, y1, y2, y3], 1))
```

Таким чином базовими компонентами слід вважати наступні складові. Шари вузького місця використовуються для зменшення складності обчислень шляхом зменшення кількості каналів перед дорогими операціями та їх подальшого збільшення. Залишкові з'єднання – допомога в тренуванні більш глибоких мереж, пом'якшуючи проблему зникаючого градієнта. Архітектура CSP ділить карту об'єктів на дві частини, одна частина зазнає трансформації, тоді як інша залишається недоторканою, що покращує здатність до навчання та зменшує кількість обчислень.

Файл conv.py містить різні згорткові модулі, включаючи стандартні та спеціалізовані згортки. Він містить кілька ключових компонентів. Стандартний модуль згортки (Conv) реалізує стандартний згортковий шар з пакетною нормалізацією та активацією:

```
class Conv(nn.Module):
    default_act = nn.SiLU()  # default activation

    def __init__(self, c1, c2, k=1, s=1, p=None, g=1, d=1, act=True):
        super().__init__()
        self.conv = nn.Conv2d(c1, c2, k, s, autopad(k, p, d), groups=g,
dilation=d, bias=False)
        self.bn = nn.BatchNorm2d(c2)
        self.act = self.default_act if act is True else act if
isinstance(act, nn.Module) else nn.Identity()

    def forward(self, x):
        return self.act(self.bn(self.conv(x)))
```

В якості складових використовуються:

– self.conv – згортковий шар;

– self.bn – нормалізація партії;

– self.act – функція активації (за замовчуванням nn.SiLU());

– forward – застосовує згортку з подальшою пакетною нормалізацією та активацією.

Згортка по глибині (DWConv) виконує згортку по глибині, де кожен вхідний канал згорнутий окремо:

```
class DWConv(Conv):
    def __init__(self, c1, c2, k=1, s=1, d=1, act=True):
        super().__init__(c1, c2, k, s, g=math.gcd(c1, c2), d=d, act=act)
```

Використовуються кілька компонентів, один наслідує від Conv і встановлює параметр groups рівним найбільшому загальному дільнику від c1 і c2, що ефективно групує згортку каналами. Conv2 –це спрощена версія RepConv, яка використовується для стиснення та прискорення моделі. GhostConv реалізує фантомний модуль GhostNet, який зменшує надмірність у картах функцій. RepConv – перепараметризований згортковий шар, який може бути перетворений з тренування в режим виводу. Таким чином, Автододавання (автопанель) автоматично обчислює кількість відступів, потрібних для забезпечення однакових розмірів виводу. Згортки по глибині та точці використовуються в архітектурах MobileNet для зменшення обчислень при збереженні точності. Повторна параметризація – такі методи, як RepConv, дозволяють ефективно тренуватися та швидше робити виводи шляхом злиття шарів.

У файлі head.py реалізовані головні модулі, що відповідають за вироблення кінцевих прогнозів моделі. Голова детектора (Detect) виглядає наступним чином:

```
class Detect(nn.Module):
    def __init__(self, nc=80, ch=()):
        super().__init__()
        self.nc = nc  # number of classes
        self.nl = len(ch)  # number of detection layers
        self.reg_max = 16  # DFL channels
        self.no = nc + self.reg_max * 4  # number of outputs per anchor
        self.stride = torch.zeros(self.nl)  # strides computed during build
        # Define layers
        self.cv2 = nn.ModuleList(
            nn.Sequential(Conv(x, c2, 3), Conv(c2, c2, 3), nn.Conv2d(c2, 4 *
self.reg_max, 1)) for x in ch
        )
```

```
self.cv3 = nn.ModuleList(
    nn.Sequential(
        nn.Sequential(DWConv(x, x, 3), Conv(x, c3, 1)),
        nn.Sequential(DWConv(c3, c3, 3), Conv(c3, c3, 1)),
        nn.Conv2d(c3, self.nc, 1),
    )
    for x in ch
)
self.dfl = DFL(self.reg_max) if self.reg_max > 1 else nn.Identity()
```

Клас Detect визначає головку детектування, яка виводить координати обмежувальної коробки та ймовірності класу і використовує такі складові:

– self.cv2 – згорткові шари для регресії обмежувальної коробки;

– self.cv3 – згорткові шари для класифікації;

– self.dfl – розподільний модуль фокусних втрат для уточнення обмежувальної рамки;

– Forward Pass обробляє карти вхідних функцій і виводить прогнози для обмежувальних рамок і класів.

Голова для завдання сегментації (Segment) виглядає наступним чином:

```
class Segment(Detect):
    def __init__(self, nc=80, nm=32, npr=256, ch=()):
        super().__init__(nc, ch)
        self.nm = nm  # number of masks
        self.npr = npr  # number of prototypes
        self.proto = Proto(ch[0], self.npr, self.nm)  # protos

        c4 = max(ch[0] // 4, self.nm)
        self.cv4 = nn.ModuleList(nn.Sequential(Conv(x, c4, 3), Conv(c4, c4,
3), nn.Conv2d(c4, self.nm, 1)) for x in ch)
```

Розширює клас Detect, включаючи можливості сегментації:

– self.proto генерує прототипи масок;

– self.cv4 – згорткові шари для коефіцієнтів маски;

– Forward Pass виводить обмежувальні рамки, ймовірності класів та коефіцієнти маски.

Голова для завдання оцінки пози (Pose):

```
class Pose(Detect):
    def __init__(self, nc=80, kpt_shape=(17, 3), ch=()):
        super().__init__(nc, ch)
        self.kpt_shape = kpt_shape  # number of keypoints, number of
dimensions
        self.nk = kpt_shape[0] * kpt_shape[1]  # total number of keypoint
outputs

        c4 = max(ch[0] // 4, self.nk)
        self.cv4 = nn.ModuleList(nn.Sequential(Conv(x, c4, 3), Conv(c4, c4,
3), nn.Conv2d(c4, self.nk, 1)) for x in ch)
```

Вона розширює клас Detect для завдань з оцінки пози людини і використовує складові:

– self.kpt_shape – форма ключових точок (кількість ключових точок, розміри на одну ключову точку);

– self.cv4 – згорткові шари для регресії ключових точок;

– Forward Pass – виводить обмежувальні рамки, ймовірності класів та координати ключових точок.

Таким чином, розширюючи клас Detect, можна створювати спеціалізовані голови для різних завдань, повторно використовуючи загальну функціональність. сучасні детектори об'єктів часто використовують підходи без якоря, передбачаючи обмежувальні рамки безпосередньо. При оцінці пози модель передбачає ключові точки, що представляють суглоби або орієнтири.

2.4 Синтез моделі глибокого навчання нейронної мережі класифікації якості агропродукції

YOLO11 є найлегшою та найшвидшою моделлю в сімействі YOLO. Вона має п'ять різних розмірів для різних випадків використання, від легких завдань до високопродуктивних програм. YOLO11 впроваджує нові архітектурні вдосконалення, такі як блок C3k2, SPPF та C2PSA, що робить модель більш ефективною у вилученні та обробці функцій та покращує увагу до ключових областей зображення. На додаток до виявлення об'єктів, YOLO11 може обробляти сегментацію екземплярів, класифікацію зображень, оцінку пози та виявлення орієнтованих об'єктів (OBB), що робить його дуже універсальним у завданнях комп'ютерного зору. Інтеграція механізмів просторової уваги, таких як C2PSA, в архітектуру допомагає YOLO11 ефективніше фокусуватися на важливих областях зображення, підвищуючи точність його виявлення, особливо для складних або закритих об'єктів. У прямому порівнянні з YOLOv10, YOLO11 демонструє чудову продуктивність,

особливо в модельному ряду Nano. Незважаючи на наявність більшої кількості параметрів, YOLO11n перевершує YOLOv10n за швидкістю висновків і FPS, що робить його високоефективною моделлю для додатків у реальному часі без шкоди для точності чи обчислювальної ефективності. В загальному випадку, система сортування продукції на конвеєрі необхідне вирішення таких завдань CV, як виявлення (сегментація), трекінг і математична реалізація підрахунку. Однак, до виконання цієї послідовності треба розробити модель глибокого навчання нейронної мережі класифікації якості агропродукції, яка буде інтегруватись до системи сортування. Її основним завданням є виявлення дефектів продукції з метою відбраковки некондиційних екземплярів. Як наслідок, розглянемо опис функціоналу коду необхідної моделі глибокого навчання нейронної мережі, який можна розділіти на кілька фрагментів.

Перший фрагмент забезпечує завантаження датасета та налаштування середовища. Код починається з підготовки середовища для аналізу даних. Завантажуються необхідні бібліотеки, такі як gdown для отримання файлів з Google Drive, і Ultralytics, яка надає сучасні інструменти для комп'ютерного зору, включаючи моделі YOLO. Після цього виконується перевірка середовища на сумісність. На цьому етапі забезпечується підготовка середовища для роботи з даними та моделями. Завантажений датасет у вигляді ZIP-архіву розпаковується у визначену папку. Бібліотека ultralytics виконує перевірку сумісності обладнання, встановлених бібліотек і драйверів для забезпечення коректної роботи. Консоль очищується після виконання команд, щоб уникнути перевантаження текстом.

Другий фрагмент визначає шляхи до зображень, а саме, окремо формується список шляхів до зображень, що входять до складу навчальної вибірки. Шляхи використовуються для завантаження зображень у подальшій обробці. Цей список є основою для роботи функції візуалізації, яка зчитує зображення та відповідну розмітку. Вони будуть використовуватися для подальшої обробки та візуалізації.

Третій фрагмент забезпечує візуалізацію зображень з розміткою. Код зчитує розмітку до кожного зображення, отримуючи координати об'єктів та їхні класи. На основі цієї інформації наноситься графічна розмітка на зображення у вигляді кольорових полігонів, причому кожен клас отримує свій колір. Таким чином, основною буде функція `visualize_images_with_annotation`, що призначена для читання розмітки (зчитується файл, пов'язаний із кожним зображенням, який містить дані про координати об'єктів і їхні класи; координати зчитуються у вигляді відносних значень (0…1), які пізніше масштабуються до пікселів); призначення кольорів – об'єкти різних класів (наприклад, «свіже яблуко» і «зіпсоване яблуко») відображаються різними кольорами; відображення результатів – Зображення із нанесеною розміткою відображаються у підграфіках, кожне у своєму вікні.

Четвертий фрагмент призначений для конфігурації. Для виконується робота з YAML-файлом, використовується для зберігання метаданих про датасет. Функція зчитує конфігураційний файл, який містить опис датасету (Зчитуються шляхи до навчального, валідаційного та тестового наборів, Вказується кількість класів (nc) і їхні назви (names) тощо). Секція roboflow містить метаінформацію про датасет, наприклад назву проєкту, робочий простір, версію, ліцензію тощо. Функція read_and_print_yaml Зчитує YAML-файл і виводить його вміст у зручному текстовому форматі.

П'ятий фрагмент реалізує функцію, яка аналізує відео з використанням моделі YOLO11. Вона виконує трекінг об'єктів і підраховує кількість об'єктів у кадрі, враховуючи задану зону інтересу. Таким чином, основною функцією буде predict_apple, що буде виконувати кілька завдань: завантажується попередньо натренована модель (наприклад, YOLO11, яка виконує детекцію об'єктів на кожному кадрі відео); визначається зона інтересу (використовується багатокутник, що обмежує область, у якій ведеться аналіз); обробляється кожен кадр (детектуються об'єкти агропродукції, для яких обчислюються обмежувальні рамки, маски та трекери), ведеться

підрахунок об'єктів, які потрапили в зону інтересу, а на кадри накладаються обмежувальні рамки, полігони та текстові підписи з класами та ID об'єктів.

Шостий фрагмент на основі оброблених кадрів створює нове відео, що відображає результати детекції та підрахунку. В даному випадку, реалізуємо функцію frames_to_video, яка виконає Об'єднання кадрів у відео (Оброблені кадри, які зберігаються у визначеній папці, з'єднуються у відео з вибраною частотою кадрів); Зменшення частоти кадрів (У разі потреби кадри у відео можуть бути зменшені за частотою (наприклад, з 30 FPS до 15 FPS), нарешті – форматування відео (вибирається кодек (наприклад, MP4), розмір кадру відповідає розміру вихідних зображень).

Сьомий фрагмент здійснює візуалізацію отриманих результатів, включаючи зображення з нанесеною розміткою, графіки тренування та відео з передбаченнями – відображаються ключові результати роботи моделі, включаючи графіки навчання, розподіл класів і приклади передбачених масок, а створене відео із зазначеними об'єктами та зоною інтересу відображається у вбудованому плеєрі.

На основі сформованого опису реалізований програмний код на мові Python для використання у хмарному Google Colaboratory (Colab), який наведений у Додатку Б.

Висновки до розділу 2

Дослідження архітектури YOLO11 демонструє її значні переваги у вирішенні завдань CV, що охоплюють широкий спектр застосувань. Завдяки інноваційним модифікаціям та інтеграції передових технологій, таких як оцінка пози, сегментація екземплярів, класифікація зображень та OBB, YOLO11 перевершує попередні версії за точністю, швидкістю та ефективністю. Висока масштабованість забезпечується 5-ма варіантами моделі, що дозволяє адаптувати її до завдань різної складності та

ресурсоємності. Конвеєр обробки зображень YOLO11 характеризується оптимізованими етапами передобробки, виявлення, постобробки та візуалізації результатів, що робить модель особливо придатною для застосувань у реальному часі, таких як автоматизоване сортування продукції на промислових конвеєрах. Використання великих наборів даних забезпечує широку попередню підготовку, що сприяє високій точності та універсальності моделей. Завдяки можливостям виявлення та класифікації навіть у складних умовах, YOLO11 є ефективним інструментом для реалізації інноваційних рішень у промисловості та наукових дослідженнях. Використання модульного підходу та вдосконалених алгоритмів, YOLO11 забезпечує не лише високу швидкість і точність виявлення об'єктів, але й ефективну сегментацію екземплярів, класифікацію зображень, оцінку пози та роботу з OBB. Удосконалення архітектури, включаючи підтримку багатомасштабного виявлення, постпроцесинг з NMS та інтеграцію інноваційних модулів, дозволяють YOLO11 забезпечувати точні результати навіть у складних умовах. Можливість розрізняти об'єкти на рівні пікселів, прогнозувати ключові точки для оцінки пози та працювати з об'єктами нестандартної орієнтації розширює діапазон застосувань для автоматизації промислових процесів.

Деталізація оновлень модулів YOLO11 показує значне вдосконалення ключових компонентів архітектури, спрямоване на підвищення ефективності, точності та гнучкості моделі. Зміни у базових модулях, таких як Bottleneck та CSP, забезпечують оптимізацію обчислювальних витрат і покращують передавання градієнтів, що сприяє глибшому та більш ефективному навчанню нейронної мережі. Інтеграція модулів, таких як SPPF для обробки багатомасштабної інформації та RepConv для покращення продуктивності, підтверджує інноваційний підхід до вдосконалення структури.

Модулі згорток, зокрема GhostConv та DWConv, оптимізують обробку даних, зменшуючи надмірність у функціях, що дозволяє зберегти точність при меншій складності обчислень. Оновлення в головних модулях, таких як

Detect, Segment і Pose, розширюють функціональність YOLO11, дозволяючи моделі виконувати завдання виявлення об'єктів, сегментації, оцінки пози та інші завдання комп'ютерного зору. Особливо важливим є використання підходів без якоря, які забезпечують точніше передбачення обмежувальних рамок та ключових точок.

Синтез моделі глибокого навчання для класифікації якості агропродукції на основі YOLO11 демонструє високий потенціал для автоматизації систем сортування на конвеєрах. Інтеграція YOLO11 з її оптимізованими компонентами, такими як блоки C3k2, SPPF та C2PSA, дозволяє ефективно вирішувати завдання CV, включаючи сегментацію, класифікацію, трекінг і підрахунок об'єктів. Висока швидкість обробки, точність детекції та гнучкість моделі забезпечують відповідність сучасним вимогам до систем реального часу. Процес побудови моделі включає чітко структуровані етапи: підготовку середовища, обробку датасетів, візуалізацію, конфігурацію параметрів, аналіз відео та створення результатів у вигляді зображень, графіків і відео. Особливу увагу приділено функції predict_apple, яка реалізує виявлення та підрахунок об'єктів у зоні інтересу з використанням попередньо навченої моделі YOLO11, забезпечуючи адаптацію до завдань класифікації якості продукції.

Розроблений програмний код, що виконується в хмарному середовищі Google Colaboratory, дозволяє ефективно аналізувати відеодані, надаючи інструменти для глибокого аналізу та оцінки якості агропродукції.

Таким чином, версія YOLO11 забезпечує підвищену продуктивність і адаптивність для вирішення складних завдань, зберігаючи простоту інтеграції та масштабованість, що робить цю архітектуру універсальним інструментом для різноманітних застосувань у сфері CV. Це підтверджує практичну значущість і універсальність моделі, сприяючи підвищенню точності сортування та зниженню витрат у агропромисловості.

РОЗДІЛ 3
РЕКОМЕНДАЦІЇ ЩОДО ВИКОРИСТАННЯ КОМП'ЮТЕРНОГО ЗОРУ І НЕЙРОННИХ МЕРЕЖ В СИСТЕМІ АВТОМАТИЗОВАНОГО ВІЗУАЛЬНОГО КОНТРОЛЮ ПРОДУКЦІЇ НА КОНВЕЄРІ

3.1 Визначення метрик продуктивності синтезованої моделі

Щоб всебічно оцінити продуктивність моделі в задачах виявлення сортування агропродукції, в роботі були прийняті загальноприйняті показники оцінки в області виявлення об’єктів, включаючи точність (Precision), чутливості (Recall), mAP50 і mAP50-95. Розглянемо більш детально їх особливості.

Precision оцінює, наскільки точні передбачення моделі, тобто частка коректно передбачених позитивних класів серед усіх передбачених як позитивні:

$$\text{Precision}=\frac{TP}{TP+FP}, \tag{3.1}$$

де TP (True Positives) – кількість правильно виявлених об'єктів (передбачення збіглося з реальністю);

FP (False Positives) – кількість хибно виявлених об'єктів (модель виявила об'єкт там, де його насправді немає).

Recall оцінює, наскільки повно модель виявляє всі об'єкти, тобто частка коректно виявлених позитивних класів серед усіх реальних позитивних класів:

$$\text{Recall}=\frac{TP}{TP+FN}, \tag{3.2}$$

де FN (False Negatives) – кількість пропущених об'єктів (модель не виявила об'єкт, який є в реальності).

Висока Precision означає, що модель робить мало хибних тривог. Використовується, коли важливіше уникнути хибних спрацювань.

Висока Recall означає, що модель знаходить більшість об’єктів, навіть якщо вона робить більше хибних спрацювань. Використовується, коли важливіше уникнути пропусків. Часто Precision і Recall знаходяться в компромісному співвідношенні: підвищення одного може знижувати інше.

Для збалансованої оцінки використовують метрику F1, яка є гармонійним середнім Precision та Recall:

$$F1=2\cdot\frac{\text{Precision}\cdot\text{Recall}}{\text{Precision}+\text{Recall}}. \tag{3.3}$$

У задачах виявлення об’єктів Precision і Recall розраховуються на основі порівняння передбачених Bounding Boxes із реальними рамками (Ground Truth Boxes). Параметр IoU (Intersection over Union) – показує, наскільки передбачена обмежувальна рамка збігається з реальною:

$$IoU=\frac{\text{Area}_{\text{Overlap}}}{\text{Area}_{\text{Union}}}. \tag{3.4}$$

Тобто IoU використовується для визначення, чи є передбачення правильним (зазвичай $IoU\geq 0{,}5$), а TP, FP і FN розраховуються на основі збігів рамок і IoU.

Також оцінка продуктивності моделі YOLO часто базується на метриках mAP50 і mAP50-95 (Mean Average Precision), які є стандартами в задачах виявлення об'єктів, та узагальнює ефективність моделі для всіх класів. AP – це середнє значення Precision для різних рівнів Recall. AP враховує, наскільки добре модель виявляє об'єкти по всій шкалі чутливості. AP розраховується як площа під кривою Precision-Recall (PR-крива):

$$AP=\int_0^1 \text{Precession}(r)dr, \tag{3.5}$$

де r – значення Recall.

Наприклад, mAP50 розраховується за фіксованого порогу IoU=0,5. Якщо $IoU\geq 0{,}5$, передбачення вважається успішним (якщо mAP50 = 0,75, це означає, що, в середньому, 75 % передбачених об'єктів з $IoU\geq 0{,}5$ є коректними). При цьому, mAP50 вимірює середню точність (AP) для кожного

класу при цьому порозі. Таким чином, mAP – це середнє значення AP для всіх класів в задачі виявлення об’єктів. Якщо у вас є N класів, то:

$$\text{mAP}=\frac{1}{\text{N}}\sum_{\text{i=1}}^{\text{N}}\text{AP}_{\text{i}}\,. \tag{3.6}$$

Метрика mAP50-95 – це середнє значення mAP для декількох порогів IoU: від 0.50 до 0.95 із кроком 0.05. Іншими словами, розраховується середня AP для порогів: IoU = 0,50, 0,55, 0,60, ..., 0,95. Наприклад, якщо mAP50-95 = 0,65, це означає, що модель досягає в середньому 65 % точності при різних рівнях збігів обмежувальних рамок.

В цілому, у контексті YOLO mAP50 часто використовується як базовий показник, тому що IoU = 0,50 є більш «щадним», mAP50-95 є жорсткішим критерієм, який краще відображає реальну продуктивність моделі, особливо для задач, де потрібна висока точність локалізації об'єктів.

3.2 Вихідні дані для синтезованої моделі глибокого навчання нейронної мережі

З метою зменшення витрат на реалізацію розробленої в роботі моделі використовується відкритий датасет з ресурсу Roboflow [40] (рис. 3.1) та хмарний сервіс Google Colaboratory [41].

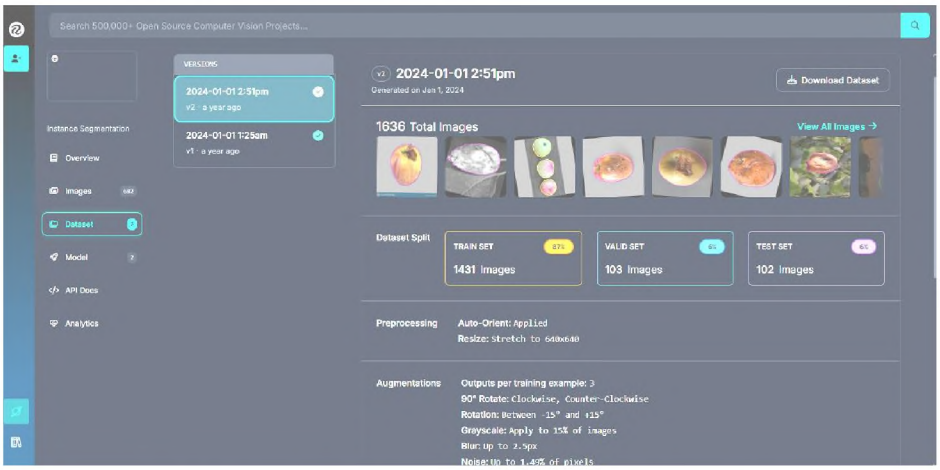

Рисунок 3.1 – Датасет з ресурсу Roboflow

Він має наступні властивості. Розмір вихідної бази зображень – 682 шт. Після аугментації датасет містить 1636 зображень, з яких 1431 – навчальна вибірка, 103 – валідаційна, 102 – тестова. В ньому анотовано два класи: свіже яблоко (fresh_apple) та зіпсоване яблоко (rotten_apple). Під час попередньої обробки виконана автоматична орієнтація та зміна розміру (розтягування до 640×640). В свою чергу, аугментація виконана на основі процедур: обертання на 90^0 (за годинниковою стрілкою, проти годинникової стрілки); обертання: від -15 до $+15^0$; відтінки сірого застосовується до 15 % зображень; розмиття: до 2,5 пікселів; рівень шуму: до 1,49 % пікселів. На рис. 3.2 наведено приклад зображень з датасету.

Рисунок 3.2 – Приклад зображень

3.3 Оцінка продуктивності розробленої моделі глибокого навчання нейронної мережі класифікації якості агропродукції

В якості базової моделі використовується YOLO11s-seg, параметри якої наведено на рис. 3.3. Приклад виконання розмітки зображень (завдання сегментації) наведено на рис. 3.4.

```
Overriding model.yaml nc=80 with nc=2
from                  n    params  module                                       arguments
  0                  -1  1       928  ultralytics.nn.modules.conv.Conv             [3, 32, 3, 2]
  1                  -1  1     18560  ultralytics.nn.modules.conv.Conv             [32, 64, 3, 2]
  2                  -1  1     29056  ultralytics.nn.modules.block.C2f             [64, 64, 1, True]
  3                  -1  1     73984  ultralytics.nn.modules.conv.Conv             [64, 128, 3, 2]
  4                  -1  2    197632  ultralytics.nn.modules.block.C2f             [128, 128, 2, True]
  5                  -1  1    295424  ultralytics.nn.modules.conv.Conv             [128, 256, 3, 2]
  6                  -1  2    788480  ultralytics.nn.modules.block.C2f             [256, 256, 2, True]
  7                  -1  1   1180672  ultralytics.nn.modules.conv.Conv             [256, 512, 3, 2]
  8                  -1  1   1838080  ultralytics.nn.modules.block.C2f             [512, 512, 1, True]
  9                  -1  1    656896  ultralytics.nn.modules.block.SPPF            [512, 512, 5]
 10                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 11             [-1, 6]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 12                  -1  1    591360  ultralytics.nn.modules.block.C2f             [768, 256, 1]
 13                  -1  1         0  torch.nn.modules.upsampling.Upsample         [None, 2, 'nearest']
 14             [-1, 4]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 15                  -1  1    148224  ultralytics.nn.modules.block.C2f             [384, 128, 1]
 16                  -1  1    147712  ultralytics.nn.modules.conv.Conv             [128, 128, 3, 2]
 17            [-1, 12]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 18                  -1  1    493056  ultralytics.nn.modules.block.C2f             [384, 256, 1]
 19                  -1  1    590336  ultralytics.nn.modules.conv.Conv             [256, 256, 3, 2]
 20             [-1, 9]  1         0  ultralytics.nn.modules.conv.Concat           [1]
 21                  -1  1   1969152  ultralytics.nn.modules.block.C2f             [768, 512, 1]
 22    [15, 18, 21]  1   2771318  ultralytics.nn.modules.head.Segment           [2, 32, 128, [128, 256, 512]]
YOLO11s-seg summary: 261 layers, 11790870 parameters, 11790854 gradients, 42.7 GFLOPs
```

Рисунок 3.3 – Параметрі використаної YOLO11s-seg

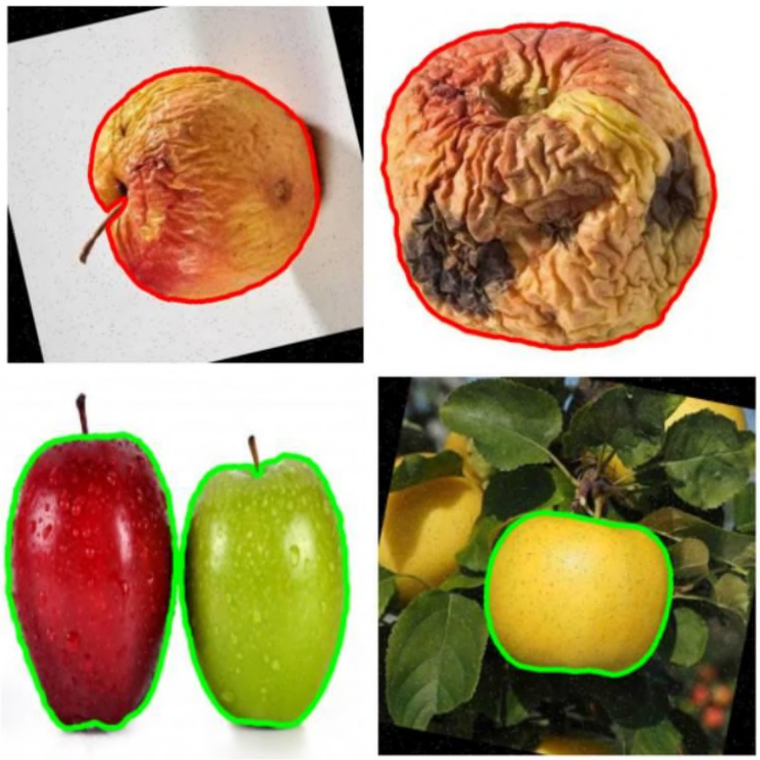

Рисунок 3.4 – Виконання розмітки зображень (завдання сегментації)

Після реалізації сегментації в інтересах сортування агропродукції (в даному випадку, це яблуки) отримаємо нормалізовану (значення від 0 до 1) матрицю помилок, що наведена на рис. 3.5.

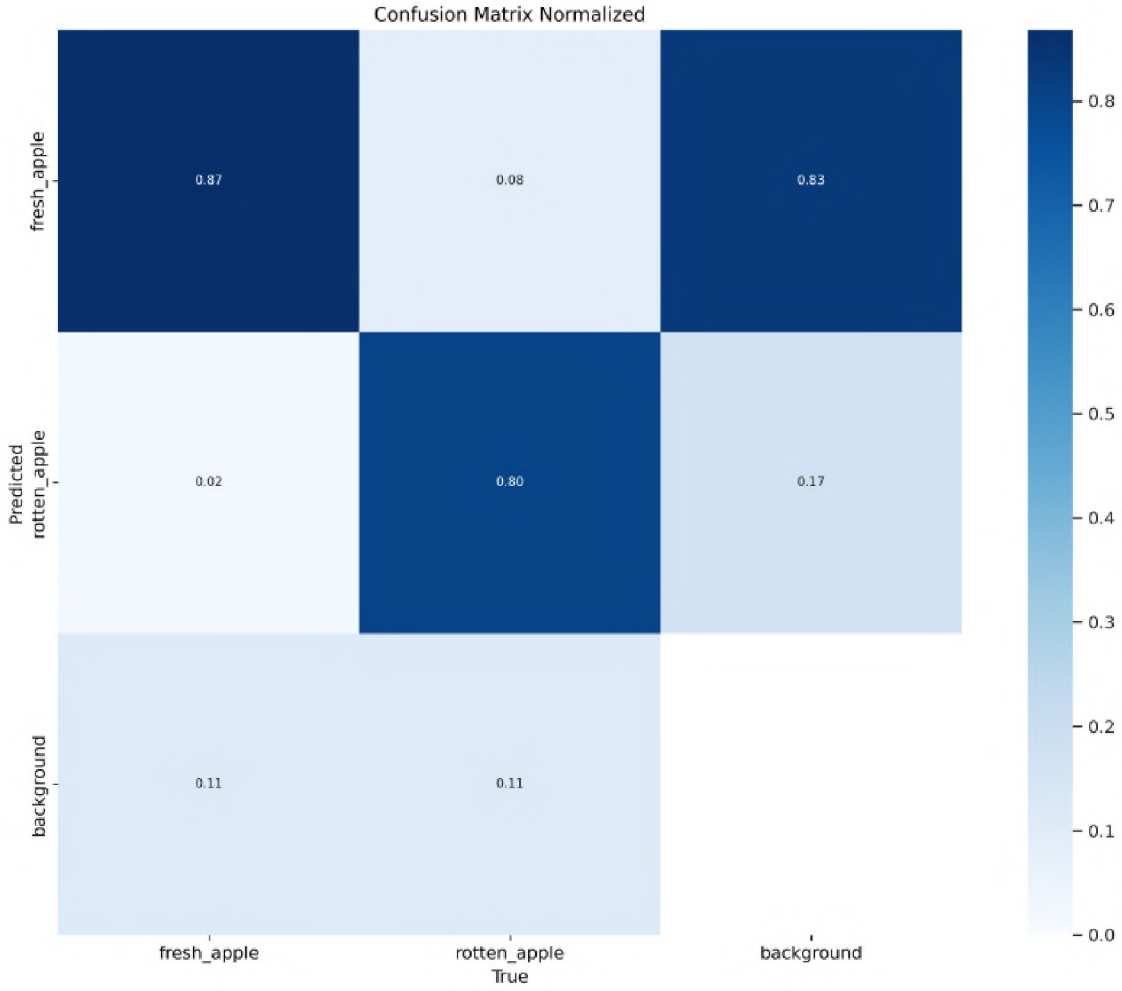

Рисунок 3.5 – Нормалізована матрицю помилок сортування агропродукції

Згідно п.3.1, в моделі розраховуються відповідні метрики оцінки продуктивності моделі глибокого навчання нейронної мережі класифікації якості агропродукції (Precision, Recall, mAP50 і mAP50-95) – рис. 3.6. На всіх графіках показники поступово покращуються зі збільшенням кількості ітерацій, що свідчить про навчання моделі. На метриках видно коливання, які можуть свідчити про складнощі з оптимізацією або нестабільність навчання. Графіки mAP50 та mAP50-95 виглядають більш гладкими, що свідчить про стабільне покращення роботи моделі.

В цілому, показник map50 = 0,87 свідчить про працездатність та адекватність запропонованої моделі. Оскільки у даному випадку використовується модель YOLO11s-seg, стратегія покращення продуктивності може бути адаптована до специфіки саме цієї архітектури. YOLO11s-seg є легкою моделлю з сегментацією, тому фокус варто зробити на тонкому налаштуванні її параметрів та якісній підготовці даних.

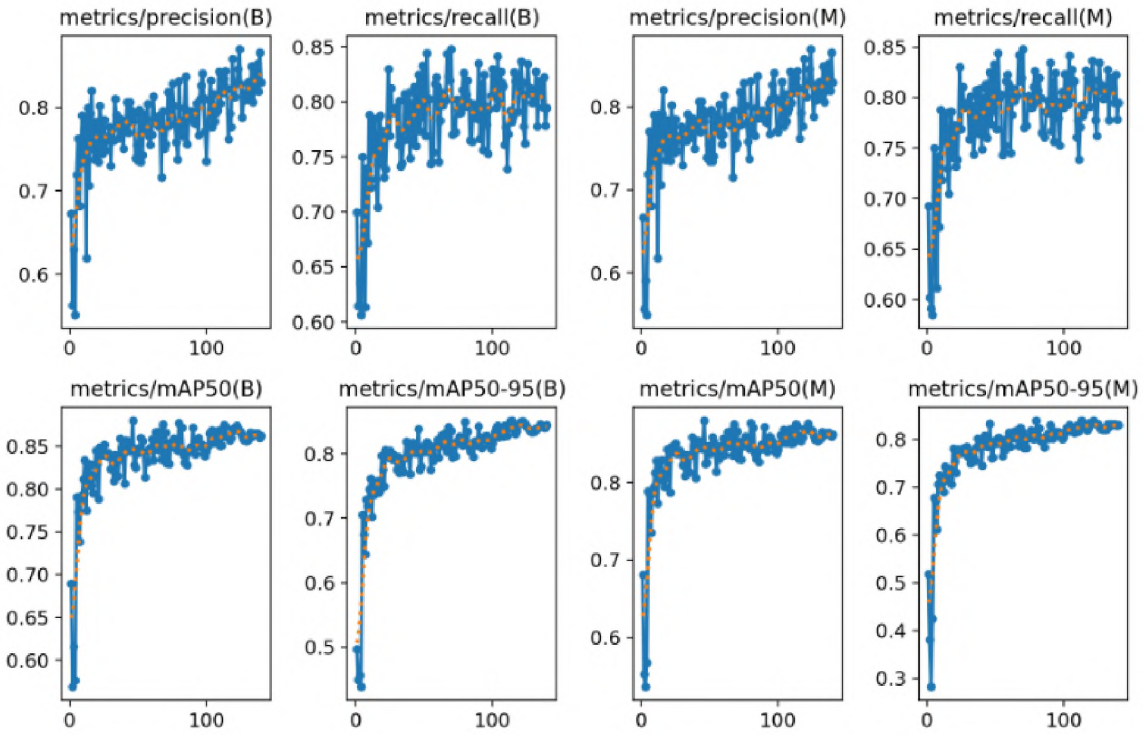

Рисунок 3.6 – Метрики продуктивності моделі глибокого навчання нейронної мережі класифікації якості агропродукції

Покращення якості та кількості даних. В даному випадку, крім проведених процедур аугментації (див. п. 3.2) можна ще застосувати специфічні техніки наприклад зміщення та, обрізання (crop). Треба зібрати більше даних з анотаціями для завдання сегментації. Для зручного створення масок крім Roboflow треба використовувати інструменти, такі як Labelme та ін., а також виконувати ручне виправлення анотацій. Треба переконатись, що маски є точними, оскільки помилки в анотаціях можуть суттєво знизити ефективність моделі.

Може допомогти тонке налаштування (Fine-Tuning) YOLO11s-seg з врахуванням донавчання на власних даних. Можна почати з заморожування базових шарів і поступово розморожувати їх для більш адаптивного навчання. Крім того, можна налаштувати mask_ratio для оптимального розподілу масок. Якщо потрібно більше деталей для сегментації, збільшують розмір вихідної маски.

Оптимізація процесу навчання. Якщо дозволяють апаратні ресурси, для більш стабільного градієнтного оновлення збільшують batch size. Якщо модель не стабільна, пробують зменшити learning rate, або збільшити цей параметр, якщо навчання надто повільне. Щоб уникнути стрибків на початкових епохах налаштовують warmup epochs. Щоб досягти стабілізації метрик збільшують кількість епох, але треба стежити за перенавчанням.

Перевірка результатів сегментації – на яких об'єктах модель часто помиляється. За необхідністю додають більше таких прикладів до навчального набору. Якщо є незбалансовані дані то варто розглянути використання Focal Loss.

Покращення архітектури. Якщо YOLO11s-seg не досягає бажаного результату, варто розглянути перехід на більшу модель (наприклад, YOLO11m-seg або YOLO11l-seg), якщо дозволяють апаратні ресурси. Щоб підвищити точність для складних зображень додають шари Attention (наприклад, Squeeze-and-Excitation або CBAM).

Покращення процесу оцінювання. Треба переконатися, що валідаційні дані покривають усі можливі сценарії.

Використання апаратного прискорення. Для прискорення навчання можна використовувати FP16. А також може виконуватись оптимізація виводу – застосування TensorRT або ONNX для прискорення роботи моделі в реальному часі на засобах, що мають невеликі обчислювальні ресурси.

Приклад відпрацювання моделі по тестовій виборці наведено на рис. 3.7.

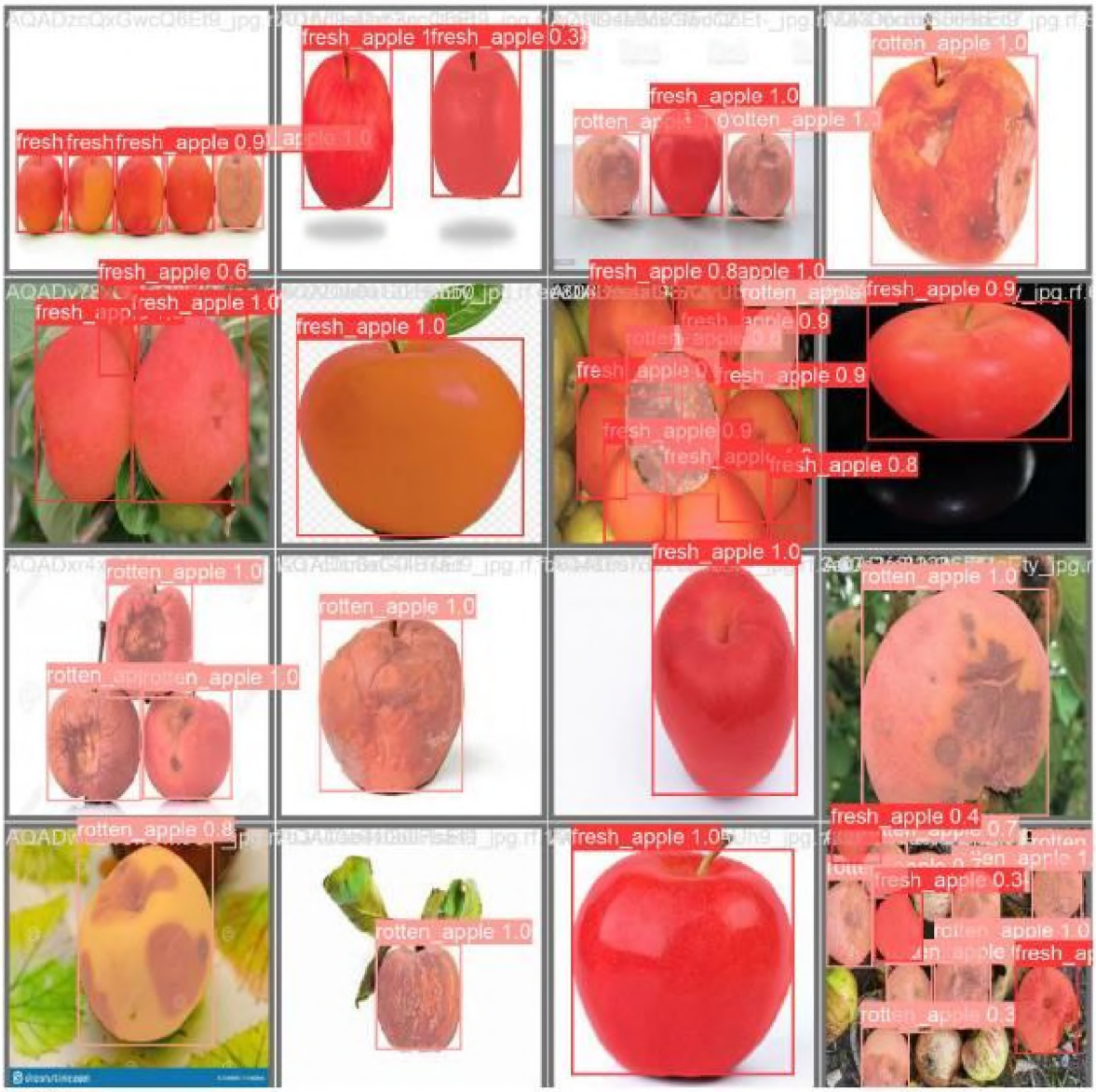

Рисунок 3.7 – Приклад класифікації агропродукції

Друга частина стосується оцінки частини коду, що відповідає за підрахунок екземплярів агропродукції. В якості даних використовується тестовий відеоряд, кадри якого з відображенням результатів роботи моделі наведено на рис. 3.8. Однак, отримана точність – 90 % є недостатньою (тобто детектується з 52-ох яблук – лише 47).

Проведений аналіз дозволив визначити напрями підвищення продуктивності моделі. Основна проблема полягає в тому, неповністю розмічені яблука на зображеннях, мало маленьких яблук, мало частково видимих яблук.

Для цього можна використовувати додаткову розмітка в Roboflow, наприклад, інтелектуальну маркування багатокутників. Крім того, введення розмиття зображення у розмірі 5 px дозволяє більш якісно обробляти зображення, що відповідають кадрам відеоряду (рис. 3.9). Проведені заходи дозволили отримати трекинг без ошибок (botsort) та детектування екземплярів агропродукції (детектується 52 із 52 яблук) – рис. 3.10.

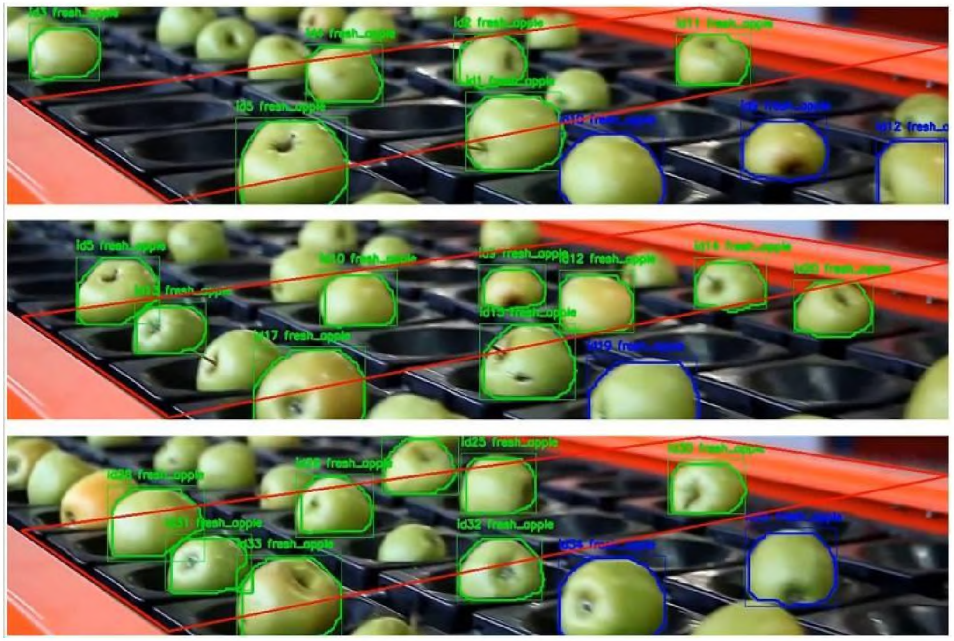

Рисунок 3.8 – Точність детектування складає 90 %

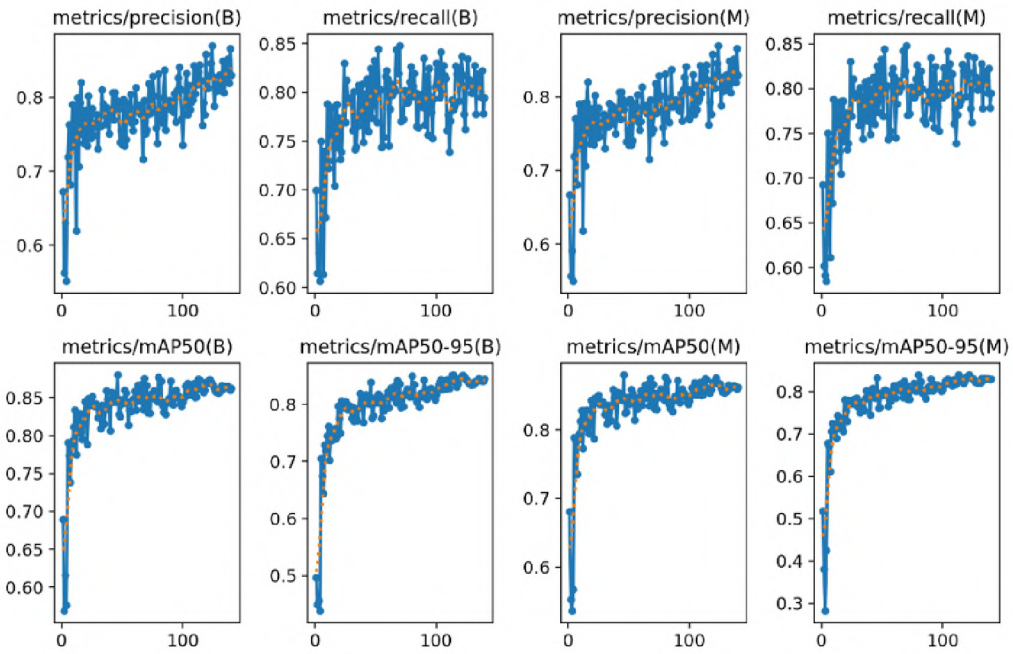

Рисунок 3.9 – Результати донавчання запропонованої моделі

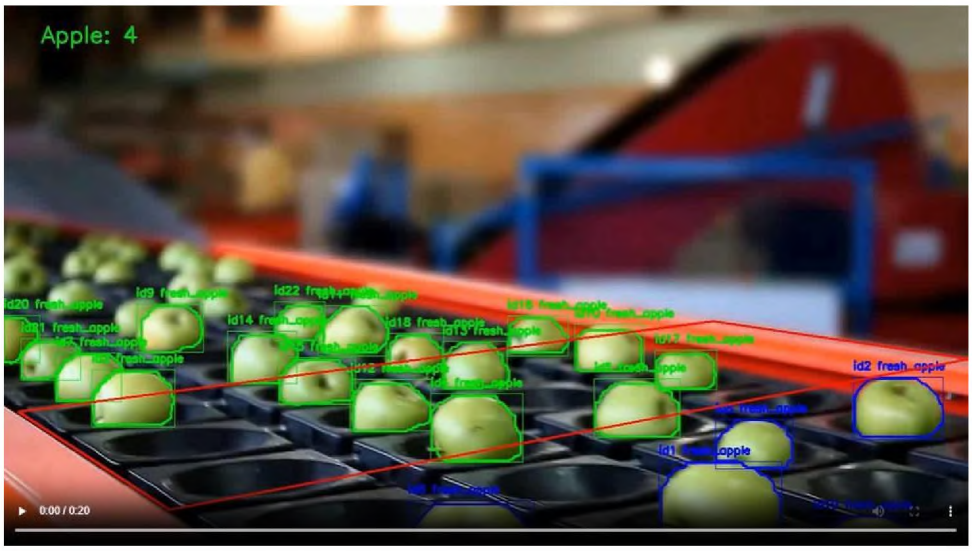

Рисунок 3.10 – Результат детектування яблук на конвеєрі

Таким чином, запропонована модель глибокого навчання нейронної мережі класифікації якості агропродукції показала працездатність та адекватність.

3.4 Економічне обґрунтування прийнятих рішень

Якщо замовляти розглянутий в роботі програмний продукт з подібним функціоналом у ІТ-компанії, то доцільно визначити орієнтовну його вартість. Розглянемо типові роботи, які проводяться при розробці та інтеграції нейронної мережі.

1. Створення ПЗ для розмітки навчальної вибірки.
2. Збір матеріалу для навчальної вибірки.
3. Розмітка навчальної вибірки.
4. Розробка архітектури нейронної мережі.
5. Навчання нейронної мережі, формування ваг.
6. Створення інфраструктури вводу/виводу даних.
7. Інтеграція нейронної мережі у продукт або програмний стек замовника.

В оцінці трудомісткості наведених пунктів враховувалися середні статичні дані, що отримані з відкритих джерел. Фонд заробітної плати та податки, що обчислюються виходячи з його розміру, є основною статтею витрат, їх розмір повинен бути визначальним у структурі ціни розробки. Розмір зазначеного фонду безпосередньо залежить від рівня заробітної плати розробників та трудомісткості роботи. У розробці ПЗ головними виробничими витратами є оплата праці програмістів, тому насамперед сформуємо перелік робіт, які необхідно виконати для розробки та визначимо їхню трудомісткість у людино-місяцях (табл. 3.1).

Таблиця 3.1 – Трудомісткість за видами робіт при використанні готового датасету

Вид робіт	Трудомісткість, люд./міс.	Виконавець, посада
Проектування	0,3	Ведучий проектувальник
Розмітка навчальної вибірки	0,1	Технік
Розробка та реалізація архітектури нейронної мережі	0,8	Ведучий інженер-програміст
Навчання нейронної мережі	0,5	Інженер-програміст
Розробка модуля отримання зображення від джерел	0,8	Інженер-програміст
Розробка графічного інтерфейсу модуля спостереження, збирання, налагодження, тестування	1,0	Інженер-програміст

Значення середньомісячного рівня «чистої» суми заробітної плати (S_1) поставимо з урахуванням середніх зарплат фахівців достатньої кваліфікації для виконання даних робіт. В розрахунку заробітної плати (S_2) використовуються діючі на поточний момент коефіцієнти зборів: (18 + 1,5) %:

$$S_2 = 1{,}195 \cdot S_1. \tag{3.1}$$

Далі порахуємо витрати на оплату праці програмістів. У табл. 3.2 у колонці «витрати часу» наведено сумарні трудовитрати у людино-місяцях кожного фахівця. S_3 відповідає фактичній сумі витрат для кожного фахівця:

$$S_3 = K \cdot S_2. \tag{3.2}$$

Склавши суми основної заробітної плати всіх фахівців, отримаємо загальні витрати на оплату праці, що необхідні для виконання роботи по створенню і впровадженню моделі глибокого навчання нейронної мережі класифікації якості агропродукції.

Таблиця 3.2 – Витрати на заробітну плату при використанні готового датасету

Виконавець	Витрати часу, K, люд./міс.	S_1, грн	S_2, грн	S_3, грн
Ведучий проектувальник	0,3	46000	54970	16491
Ведучий інженер-програміст	0,8	42000	50190	40152
Інженер-програміст	2,3	37000	44215	101694,5
Технік	0,1	20000	23900	2390
Разом:	6,6			160727,5

Стосовно проведених досліджень, слід враховувати кілька припущень. а саме: використовувався готовий датасет [40], за пунктом 6 не проводилися дослідження та розробки.

Пункт 7 може як не коштувати нічого (якщо замовник має достатню компетенцію для інтеграції та планує використання програмного модуля самостійно), так і перевищити в ціні всі інші пункти разом узяті. У табл. 3.3 наведено безпосередньо розрахунок ціни розробки ПЗ. В якості допущення статті «матеріали», «спецобладнання», єдиний соціальний внесок та «відрядження» приймаємо рівними нулю. Накладні витрати приймаємо на рівні 15 % від витрат на заробітну платню.

Витрати на оплату праці робітників беремо з табл. 3.2. Сума перерахованих вище статей становить внутрішні витрати підприємства, що виконує роботи, разом із витратами залучених сторонніх організацій (в даному випадку, дорівнюють нулю) вони становлять собівартість розробки. Прибуток організації встановимо лише на рівні 20 % собівартості.

Сума прибутку та собівартості утворюють ціну розробки. Далі до ціни додається ПДВ 20 % і отримуємо ціну розробки ПЗ з ПДВ – 266165 грн.

Таблиця 3.3 – Орієнтовна оцінка вартості розробки ПЗ при використанні готового датасету

Найменування статей витрат	Сумма, грн
Матеріали	-
Спеціальне обладнання	-
Витрати на оплату праці виконавців	160727,5
Додаткова зарплатня	-
Єдиний соціальний внесок	-
Витрати на відрядження	-
Накладні витрати	24109,13
Разом внутрішні витрати	184836,6
Витрати сторонніх організацій	-
Усього собівартість	184836,6
Прибуток	36967,33
Ціна без ПДВ	221803,9
ПДВ, 20 %	44360,79
Ціна з ПДВ	266164,7

В цілому, загальна ціна роботи може зрости тільки за рахунок робіт, що не стосуються безпосередньо синтезу нейронної мережі – розробки ПЗ, частиною якого є нейронна мережа (аналітика, звіти, бази даних, портали, інфраструктура та робочі місця користувачів та ін.).

Висновки до розділу 3

На основі аналізу метрик, представлених у розділі 3.1, можна зробити такі висновки. Застосування загальноприйнятих показників оцінки продуктивності, таких як Precision, Recall, mAP50 і mAP50-95, дозволяє всебічно характеризувати ефективність моделі для задач виявлення об’єктів.

Кожна з розглянутих метрик має свою специфіку і сферу застосування. Наприклад, Precision фокусується на точності передбачень, що є важливим для задач, де критичне значення мають помилкові спрацювання, тоді як Recall підходить для задач, де необхідно зменшити кількість пропущених об’єктів. Гармонійне поєднання цих характеристик забезпечує F1-метрика. Крім того, використання IoU для оцінки точності рамок дозволяє більш об'єктивно

визначати правильність передбачень. Метрики mAP50 і mAP50-95 забезпечують комплексну оцінку продуктивності моделі для різних порогів IoU. При цьому mAP50 є більш гнучким критерієм, який демонструє базову ефективність моделі, тоді як mAP50-95 забезпечує більш точне і суворе оцінювання. Ці метрики є ключовими для задач, де важлива висока точність локалізації об'єктів.

Таким чином, правильний вибір і інтерпретація метрик є необхідними для об’єктивної оцінки і подальшої оптимізації моделей, спрямованих на виконання специфічних задач виявлення об'єктів.

На основі аналізу вихідних даних можна зробити такі висновки. Використання відкритого датасету Roboflow для створення синтезованої моделі глибокого навчання забезпечило економію ресурсів при підготовці даних. Завдяки структурованому підходу до формування вибірки (зокрема розподілу на навчальну, валідаційну та тестову підмножини) та анотації класів, забезпечено необхідну базу для тренування нейронної мережі.

Аугментація даних, що включала обертання, зміни відтінків, розмиття та додавання шуму, значно розширила вихідну базу, що сприяє підвищенню стійкості моделі до різноманітних варіацій у вхідних даних. Застосовані методи попередньої обробки, такі як автоматична орієнтація та уніфікація розмірів, дозволили стандартизувати вхідні зображення, що є важливим для забезпечення ефективного навчання моделі. Підготовка та використання датасету Roboflow відповідають сучасним підходам до роботи з великими обсягами даних у задачах CV, створюючи міцну основу для тренування та подальшого вдосконалення моделі глибокого навчання.

На основі аналізу результатів продуктивності моделі глибокого навчання YOLO11s-seg можна зробити такі висновки. Використання нормалізованої матриці помилок та ключових метрик (Precision, Recall, mAP50, mAP50-95) дозволило оцінити ефективність моделі у задачі класифікації якості агропродукції. Отримані значення, зокрема mAP50 = 0,87, свідчать про високу точність і працездатність моделі в умовах, що були задані

для експерименту. Разом з тим аналіз виявив кілька можливостей для подальшого вдосконалення моделі, таких як збільшення та покращення якості даних через донабір прикладів і точну анотацію, використання спеціальних технік обробки зображень (наприклад, розмиття або зміщення), а також застосування додаткових архітектурних елементів (наприклад, шарів Attention). Для покращення результатів було запропоновано оптимізацію параметрів навчання, зокрема налаштування learning rate, batch size, warmup epochs та використання сучасних інструментів апаратного прискорення (FP16, TensorRT, ONNX).

Проведені експерименти з донавчання та уточнення анотацій дали змогу досягти точності детектування у 100 % на тестовій вибірці, що підтверджує високу ефективність запропонованих заходів. Зокрема, впровадження додаткової розмітки та оптимізації трекінгу дозволило усунути попередньо виявлені проблеми з класифікацією частково видимих об'єктів та малих яблук.

Таким чином, запропонована модель продемонструвала адекватність і високий потенціал для практичного застосування у задачах автоматизованого сортування агропродукції. Подальше вдосконалення моделі може бути спрямоване на її адаптацію до нових умов роботи та розширення функціональних можливостей.

ВИСНОВКИ

Інтеграція AI, зокрема алгоритмів YOLO, у системи сортування продукції є раціональним підходом, який сприяє інноваціям у виробництві, покращує якість продукції, знижує витрати та забезпечує конкурентні переваги підприємств. Особливо важливим є використання сучасних версій YOLO (YOLOv8, YOLO11), які демонструють високу ефективність у складних умовах аналізу зображень, підтримують мультиоб'єктну детекцію та інтеграцію з ресурсозберігаючими пристроями. Новітні вдосконалення, такі як модулі C2PSA та оптимізовані блоки C3k2, дозволяють розпізнавати навіть найдрібніші дефекти, забезпечуючи точність без компромісів зі швидкістю. Дослідження архітектури YOLO11 демонструє її переваги як високоефективної моделі для вирішення завдань CV. Завдяки інноваційним модифікаціям, таким як блоки C3k2, SPPF та C2PSA, модель забезпечує значне покращення у вилученні функцій, сегментації, класифікації, оцінці пози та роботі з OBB. Гнучкість архітектури з 5-ма масштабованими варіантами дозволяє адаптувати її до різних завдань та ресурсних обмежень, забезпечуючи точність та швидкість у реальному часі. Оптимізація обчислень через використання модулів Bottleneck, CSP, GhostConv та RepConv підвищує продуктивність без втрати точності. YOLO11 підтверджує свою релевантність та універсальність у сучасних задачах CV, забезпечуючи оптимальне співвідношення точності, швидкості та ефективності для промислових і наукових застосувань.

Синтез моделі глибокого навчання для класифікації якості агропродукції засвідчує значний потенціал YOLO11 для автоматизації сортувальних процесів на промислових конвеєрах. Розроблений програмний код у хмарному середовищі Google Colaboratory дозволяє ефективно аналізувати відеодані, здійснювати трекінг, підрахунок об'єктів і візуалізацію результатів, що сприяє підвищенню точності сортування та зниженню витрат у агропромисловості. На основі проведеного аналізу було зроблено висновки

щодо ефективності підходів і методів, застосованих для побудови та оцінки моделі глибокого навчання в задачах автоматизованого сортування агропродукції. Вибір і застосування метрик Precision, Recall, mAP50 та mAP50-95, забезпечили комплексну характеристику продуктивності моделі. Врахування специфіки кожної метрики дозволило оцінити модель як з точки зору точності передбачень, так і з точки зору повноти виявлення. Використання відкритого датасету Roboflow у поєднанні з процедурами аугментації та попередньої обробки даних значно покращило стійкість моделі до різноманітних варіацій у даних. Структурований підхід до формування вибірок і стандартизація зображень забезпечили міцну основу для навчання та валідації моделі. Запропонована модель на основі YOLO11s-seg продемонструвала високу точність (mAP50 = 0,87), що свідчить про її працездатність і відповідність завданням класифікації якості агропродукції. Разом з тим аналіз продуктивності виявив можливості для подальшого вдосконалення, зокрема через розширення датасету, уточнення анотацій, застосування сучасних методів оптимізації навчання та апаратного прискорення. Завдяки впровадженню запропонованих модифікацій було досягнуто значного покращення результатів — точність детектування на тестовій вибірці досягла 100 %, що підтверджує ефективність проведених заходів. Це дозволяє говорити про високу перспективність запропонованої моделі для використання в автоматизованих системах сортування агропродукції. В цілому, розроблена модель глибокого навчання є практичною, ефективною та може бути адаптована до нових умов та задач, що робить її перспективною для впровадження в реальні виробничі процеси.

Таким чином, результатами роботи є модель глибокого навчання нейронної мережі класифікації якості агропродукції; рекомендації щодо використання комп'ютерного зору і нейронних мереж в системі автоматизованого візуального контролю продукції на конвеєрі. Вони можуть бути використані для подальших досліджень за даною тематикою та при реалізації обробки сільськогосподарської продукції.