Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ ЗАСТОСУНКУ ДЛЯ ДЕТЕКЦІЇ ТА НОРМАЛІЗАЦІЇ ЧЕКІВ НА ОСНОВІ ДОНАВЧАННЯ YOLO-МОДЕЛЕЙ ДЛЯ ПІДВИЩЕННЯ ТОЧНОСТІ РОЗПІЗНАВАННЯ ТЕКСТУ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-2
Зуйко І. В.
(прізвище, ініціали)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник доц. Любченко В. А.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

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

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Зуйко Ірині Валеріївні
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення застосунку для детекції та нормалізації чеків на основі донавчання YOLO-моделей для підвищення точності розпізнавання тексту

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 19 червня 2026 р.
3. Вихідні дані до роботи науково-методична та науково-технічна література, матеріали конференцій, дані інтернет-мережі, бібліотека комп'ютерного зору з відкритим кодом OpenCV, інтегроване середовище розробки Android Studio, інструменти комп'ютерного зору Roboflow, інструмент для створення діаграм і схем Draw.io, мультимодальні моделі Gemini 2.5 Flash, Gemini 2.5 Flash Lite, Gemini 3.1 Flash Lite, мова програмування Kotlin, бібліотека для створення інтерфейсу Gradio, нейромережі для розпізнавання об'єктів YOLO.

4. Перелік питань, що потрібно опрацювати в роботі
1. Аналіз проблеми точності розпізнавання тексту засобами OCR на зображеннях із геометричними спотвореннями.
2. Реалізація набору даних та тренування моделей.
3. Аналіз та вибір методу перспективної корекції чеків із застосуванням YOLO-моделей.
4. Розроблення мобільного застосунку Android для нормалізації чеків.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) робота OCR, проблеми розпізнавання тексту, розпізнавання тексту YOLO-моделями, діаграма IDEF3, діаграма послідовностей, діаграма діяльності, діаграма класу, тестові ілюстрації роботи програми.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-16.04.26	
3	Аналіз літератури з проблеми, що вирішується	17.04.26-19.04.26	
4	Аналіз існуючих методів розпізнавання	20.04.26-22.04.26	
5	Формування кроків вирішення проблеми	23.04.26-05.05.26	
6	Програмна реалізація	26.04.26-20.05.26	
7	Аналіз отриманих результатів	21.05.26-04.06.26	
8	Оформлення пояснювальної записки	05.06.26-10.06.26	
9	Перевірка на нормоконтроль	10.06.26-19.06.26	
10	Перевірка на плагіат	11.06.26-30.06.26	
11	Рецензування	20.06.26-30.06.26	
12	Підготовка презентації та доповіді	20.06.26-30.06.26	
13	Занесення роботи в електронний архів	30.06.26-09.07.26	
14	Попередній захист кваліфікаційної роботи	30.06.26-09.07.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ _____доц. Любченко В. А._____
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 79 с., 4 табл., 58 рис., 64 джерел.

ДЕТЕКЦІЯ, СПОТВОРЕННЯ ЗОБРАЖЕНЬ, ТОВАРНИЙ ЧЕК, ANDROID ЗАСТОСУНОК, KOTLIN, LLM, OCR, OPENCV, YOLO.

Об’єктом роботи є використання YOLO-моделей для детекції зображень чеків та їх подальшого вирівнювання.

Метою роботи є розроблення андроїд застосунку, здатного визначати чеки на фотографіях за допомогою донавченої YOLO-моделі та вирівнювати зображення з використанням бібліотеки комп’ютерного зору OpenCV.

Під час роботи використано: методи комп’ютерного зору та ШІ (розпізнавання документів, визначення кутових точок, перспективна корекція зображення, OCR та LLM-розпізнавання), методи Android-розробки.

Практична цінність застосунку полягає в підвищенні точності розпізнавання тексту на чеку, шляхом автоматичної перспективної корекції зображень.

Розроблено Android застосунок, що розпізнає межі чеку та перспективно вирівнює зображення за 4 кутами.

Область застосування: мобільні OCR застосунки, автоматизація обліку витрат, ведення бухгалтерії, банківські системи.

ANDROID APP, DETECTION, IMAGE DISTORTION, KOTLIN, LLM, OCR, OPENCV, RECEIPT, YOLO.

The objective of this work is to use of YOLO models for detecting images of receipts and subsequently aligning them.

The goal of this work is to develop an Android app capable of detecting receipts in photos using a pre-trained YOLO model and aligning images using the OpenCV computer vision library.

The following were used during the work: computer vision and AI techniques (document recognition, corner point detection, perspective image correction, OCR, and LLM recognition), as well as Android development techniques.

The practical value of the app lies in improving the accuracy of text recognition on receipts through automatic perspective correction of images.

An Android app has been developed that detects the edges of a receipt and, in the future, will align the image using the four corners.

Applications: mobile OCR apps, expense tracking automation, accounting, banking systems.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ШІ – штучний інтелект

CNN – Convolutional Neural Network (згорткова нейронна мережа)

CPU – Central Prossesing Unit (центральний процесор)

CRAFT – Character Region Awareness for Text Detection (детекція тексту з аналізом символьних регіонів)

CTPN – Connectionist Text Proposal Network (мережа пропозицій текстів на основі зв’язків)

DB – Differentiable Binarization (диференційована бінаризація)

DFL – Distribution Focal Loss (розподіл центральних втрат)

GPU – Graphical Processing Unit (графічний процесор)

Ground truth – еталонні дані для навчання й оцінки моделей

IDE – Integrated Development Enviroment (інтегроване середовище розробки)

IDEF3 – Integration Definition for Process Description Capture Method (метод опису і моделювання процесів)

IoU – Intersection over Union (перетин над об’єднанням)

JSON – JavaScript Object Notation (текстовий формат обміну даними)

LLM – Large Language Model (велика мовна модель)

MSER – Maximally Stable Extremal Regions (максимально стабільні екстремальні регіони)

NLP – Natural Language Processing (обробка природної мови)

OBB – Oriented Bounding Boxes (орієнтовані обмежувальні бокси)

OCR – Optical Character Recognition (оптичне розпізнавання символів)

SAM – Segment Anything Playground (поле для сегментування будь-чого)

SWT – Stroke Width Transform (перетворення товщини штриху)

TF Lite – TensorFlow Lite(спрощенна версія TensorFlow для мобільних та вбудованих пристроїв)

UI – User Interface (інтерфейс користувача)

Visual Transformer – це трансформер, призначений для задач зорової обробки

YOLO – You Only Look Once (ви дивитесь тільки раз)

ВСТУП

Сфера застосування технологій розпізнавання тексту постійно розширюється. Для цього застосовують, як звичайні OCR системи, так і сучасні покращені LLM моделі. Чеки є складними для розпізнавання системами. Чимало факторів, а особливо перспективні спотворення зображення безпосередньо впливатимуть на якість і точність розпізнавання тексту на чеках. Не завжди хочеться витрачати час для пошуку ідеального кута документу на фото і часто на зображеннях, навіть намагаючись уникнути викривлень, вони залишаються. Тому для пришвидшення та автоматизації процесів вирівнювання чеків пропонується нове вирішення із застосуванням YOLO-моделей для виявлення меж чеку та засобів комп’ютерного зору OpenCV для перспективної корекції.

Актуальність роботи полягає у перспективі застосування рішення в реальному світі. Одне з них це можливість інтеграції застосунку в існуючи системи розпізнавання тексту для покращення точності розпізнавання тексту. Не всі системи можуть впоратися із перспективно спотвореним текстом, а застосунок гарантує швидке вирішення проблеми і покращення вихідного результату розпізнавання.

1 АНАЛІЗ ПРОБЛЕМИ ТОЧНОСТІ РОЗПІЗНАВАННЯ ТЕКСТУ ЗАСОБАМИ OCR НА ЗОБРАЖЕННЯХ ІЗ ГЕОМЕТРИЧНИМИ СПОТВОРЕННЯМИ

1.1 Системи розпізнавання тексту

З поширенням використання цифрових технологій, життя людей в усіх галузях за останні роки кардинально змінилося. Все актуальнішим і масовим стає оцифровування текстових даних. Переваг існування тексту в електронному вигляді суттєва кількість: підвищення швидкості пошуку необхідних даних, можливість повторного відтворення якісних фізичних примірників, можливість розповсюдження і наявність цифрових копій документів у багатьох людей, зручність редагування документів, компактність, збільшення кількості вільного місця та покращення структурованості зберігання даних [1]. Для розпізнавання і перенесення тексту з фізичного в цифровий формат, замість введення даних вручну, давно застосовують системи OCR. Вигода використання OCR в оптимізації витраченого час завдяки автоматичному перенесенню даних, мінімізуванні допущених помилок, пов’язаних з ручною обробкою даних та підвищенні продуктивності [2, 3].

Застосування систем OCR є важливими в багатьох сферах життя людей. Його використання корисно в бібліотеках та музеях для збереження рідких примірників архівних документів, книг, газет та інших фізичних примірників від втрат, знищення або погіршення стану через вплив часу [1]. Попереднє розпізнавання тексту вбудовано в багатьох сучасних перекладачах при роботі з фото. Ця функція корисна як для швидкого розуміння іншомовного тексту з книги, так і під час подорожі – наприклад, для перекладу написів з афіш, меню та інших матеріалах за кордоном [2]. В бізнесах, банках, сфері здоров’я, освіті, державних установках, інших закладах та сферах працюють з великою кількістю документів, книг або чеків і потребують переносу копій в цифровий

формат для ефективного та компактного їх зберігання та зручного корегування [3].

Загальний опис роботи систем OCR представлений на рисунку 1.1 [3]. Після того як документ завантажується для розпізнавання тексту, часто система виконує попередню обробку зображення для покращення точності вихідного тексту. Наступним кроком відбувається розпізнавання тексту, де зображення розділяється на темні та світлі образи. Якраз темні образи і розпізнаються як текст для розшифрування, що надалі аналізуються для локалізації літер, світлі визначаються як фон. В ході цього етапу виконуються процеси розпізнавання шаблонів та ознак. Розпізнавання шаблонів визначає символи, порівнюючи зі схожими шрифтами на яких OCR було навчено, а розпізнавання ознак розбиває символи на певні ознаки: лінії, точки та інше і по ним визначає, яка саме це літера або цифра. Після обробка фото включає в себе корекцію помилково виявлених слів з отриманого тексту на основі наявного словника [4, 5].

Рисунок 1.1 – Ілюстрація роботи систем OCR

Останнім часом все популярнішим для розпізнавання та оцифровування тексту стає використання сучасних покращених LLM моделей: ChatGPT, Copilot, Gemini [6]. На відміну від систем OCR мовні моделі не мають чіткого розподілення на етапи при розпізнаванні тексту з зображень, а об’єднують їх в один. Якщо в класичних системах спочатку виконується детекція області тексту, наступним етапом розпізнавання конкретних символів, то в LLM використовуються мультимодальні моделі, які одразу «розуміють» зміст, досягається це вбудовуванням технологій комп’ютерного зору з наявними функціями обробки NLP. Після отримання моделями зображень документів та текстового промпту, фото проходить через візуальний енкодер CNN, Visual Transformer, або подібні, який переводить пікселі в послідовність «візуальних токенів», враховуючи фрагменти з текстом. Деякі технології інтегрують системи OCR для безпосереднього розпізнавання тексту з LLM моделями для корекції, заповнення пропусків і загального покращення тексту [7–9]. Незважаючи на переваги LLM в точності розпізнавання тексту, навіть на спотворених зображеннях, все ж таки перспективні викривлення теж впливають на моделі, хоча не так сильно як на OCR. Також недоліком є швидкість розпізнавання тексту, мовні моделі витрачають набагато більше часу для зчитування за звичайні OCR, які це роблять миттєво.

1.2 Методи детекції текстових областей на геометрично спотвореному зображенні

При виконанні етапу розпізнавання тексту важливим є детекція текстових областей на зображенні. Правильне визначення розташування безпосередньо впливатиме на подальшу точність розпізнавання системами окремих символів [10, 11].

Оператор SWT перетворює зображення з того, що складається з кольорових значень на піксель на те що містить масив для кожного пікселів з

найбільш ймовірним значенням ширини штрихів. В підсумку системи можуть вилучити текст вимірюючи в кожному компоненті значення ширини, незважаючи на шрифт або масштаб, оскільки вважає, що текст схильний містити сталу ширину штриха [11–14].

MSER для детекції тексту визначає області, які називаються максимально стабільними екстремумами інтенсивності. Метод очікує, що текст складається з стабільного кольору та високого контрасту, тому він погано працюватиме зі світлим, неконтрастним текстом, методи сполучених компонентів: SWT та MSER також застосовують разом для покращення результату [15–17].

В складних умовах детекції тексту на реальних зображеннях з перспективними спотвореннями, шумами, неякісним освітленням та іншими недоліками, класичні методи, як SWT, MSER показують себе гірше ніж сучасні методи моделей заснованих на глибокому навчанні. Так як SWT метод розраховує ширину штрихів для кожного пікселя, зміна перспективи безпосередньо впливає на ширину, що позначиться на коректності розпізнавання текстових рядків і їх цілісності. Для MSER потрібні стабільні області інтенсивності, з геометричними спотвореннями однорідність порушиться і області тексту можуть розмиватися: зливатися між собою або розширюватися.

Для пристосування до реальних зображень зі складними сценами, шумами, та іншими ускладненнями розробили моделі на основі глибокого навчання, зокрема CNN, навчених на великих наборах даних, наприклад ICDAR або TotalText. Методи на основі згуртованих нейронних мереж (CNN) є швидкими та точними, оптимально підходить для детекції тексту [18–22].

CRAFT працює чітко виділяючи розміщення кожного тестового символу присутнього на зображенні. Метод використовує мережу VGG16 з U-net подібним декодером. За оцінюванням вірогідності середини символу передбачаються розташування конкретно символів, а за оцінюванням спорідненості символів з окремих символів формуються слова може

пропустити етап пост обробки так як в ньому немає необхідності. Текст виділяється посимвольно і метод відмінно адаптується до геометричних спотворень зображення [22–24].

DBnet – це метод на основі сегментації, що дозволяє йому краще локалізувати текст різних форм, наприклад викривлений або під кутом, використовує диференційовану бінарізацію (DB), адаптивний поріг, який спрощує пост обробку і покращує продуктивність виявлення тексту на зображеннях, яка не втрачається навіть після видалення DB. Отже, завдяки використанню сегментації, перспективні спотворення не повинні впливати на детекцію тексту, адже вона може підлаштовуватися під будь-які форми спотворення [25, 26].

CTPN використовує рекурентну нейронну мережу, модель виявляє текст, сегментуючи область на вертикальні пропозиції з рядками, враховуючи послідовності символів. Погано працює з геометрично спотвореним текстом, адже архітектура розрахована більше на рядки з горизонтальним текстом з фіксованими пропорціями [27].

Не зважаючи на те, що YOLO – універсальна архітектура для розпізнавання об’єктів, на базі покращеного CNN, не розроблена конкретно для детекції текстових областей, але може бути адаптована для цієї задачі, шляхом навчання на зібраному наборі даних. Працює модель синхронно на всіх областях зображення, для кожної прогнозуючи обмежувальні рамки та впевненість. Для пристосування до геометрично спотвореного тексту на зображеннях можна застосовувати OBB або сегментацію [28].

Якщо розглядати сучасні LLM, то вони не мають в собі класичних алгоритмів для детекції областей тексту, а поєднують усі процеси обробки в одному.

За кутом нахилу рядків і обмежувальних боксів оцінюється викривлення тексту на зображенні. Для кожної області з текстом розраховується кут головної осі, якщо кут текст на площині не дорівнює нулю, тоді знаходиться чотирикутник для усього тексту і застосовується проєктивна трансформація.

1.3 Проблема розпізнавання тексту на геометрично спотвореному зображені

На точність та швидкість розпізнавання тексту системами OCR впливає чимало факторів. Початкова якість зображення безпосередньо відображається на результаті: контрастність тексту, роздільна здатність, якість освітлення, шуми та нахил зображення [29–36].

Некоректна детекція текстових областей безпосередньо впливатиме на якість розпізнавання тексту. Тому для багатьох систем OCR, критичним є горизонтально або вертикально вирівняний текст для кращих і точніших результатів [35]. Деякі системи OCR можуть впоратися з незначними нахилами, або вирівнювати кути в 90°, 180°, 270°, але не можуть впоратися з іншими кутами, що зменшує точність виявленого тексту [35, 37]. В багатьох сучасних системи, як Google Cloud Vision, Azure Computer Vision API наявні вбудовані функції попередньої обробки фото, які автоматично виявляють нахил тексту або попередньо обрізають зображення [38, 39]. Google Cloud Vision використовує багатокутники для виявлення тексту, які допомагають йому краще обробляти нахилений текст і мінімізує недолік відсутності функції розпізнавання орієнтації. Azure використовує прямокутники для визначення меж тексту і по ним намагається визначити нахил зображення [38]. Якщо розглядати Tesseract і OCRopus, який заснований на його базі, то вони можуть самостійно виправити нахил зображення на 90°, 180°, 270°, але не інші, тому для покращення точності радять вирівнювати зображення самостійно або зазвичай застосовують сторонні інструменти, як Leptonica, OpenCV, ImageMagick, unpaper, Gimp [40–48] (рис. 1.2). Крім того в Tesseract вказано вплив великих країв на зображенні і підкреслено необхідность їх обрізання перед наданням для розпізнавання [40]. EasyOCR також не має вбудованих функцій для вирівнювання зображень, але він використовує модель для виявлення тексту CRAFT, яка автоматично виявляє обмежувальні блоки кожного слова і використовує їх як зображення по яким передбачає літери,

тому на систему не повинен сильно впливати нахил, хоча все одно рекомендується вирівнювати зображення [49, 50]. PaddleOCR також може виявляти та вирівнювати зображення повернуте на 90°, 180°, 270°, передбачати нахил ліній тексту, але не інші кути [51].

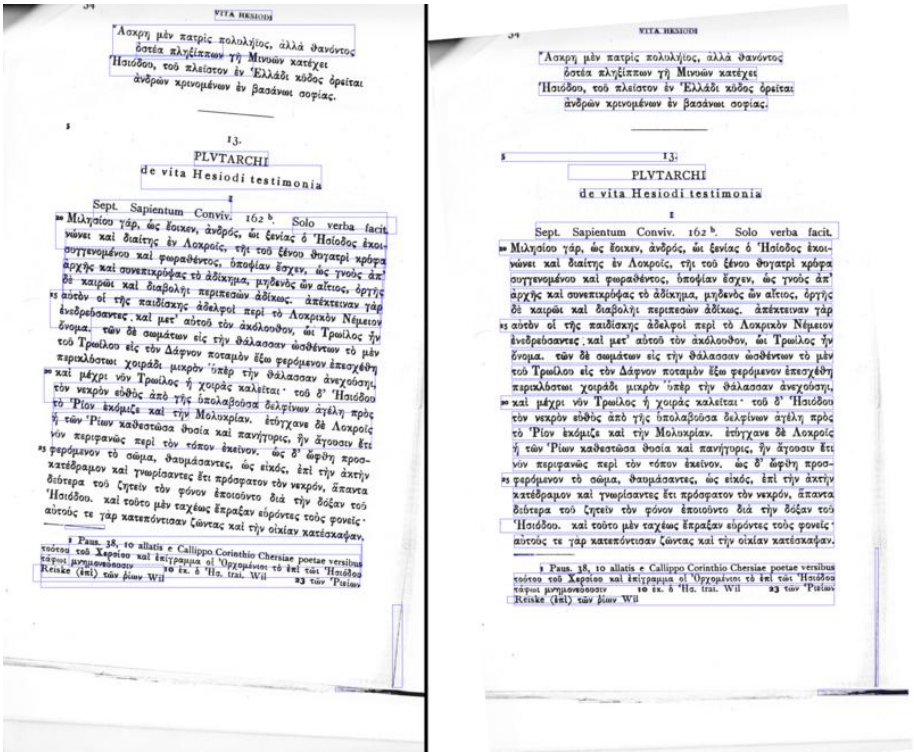
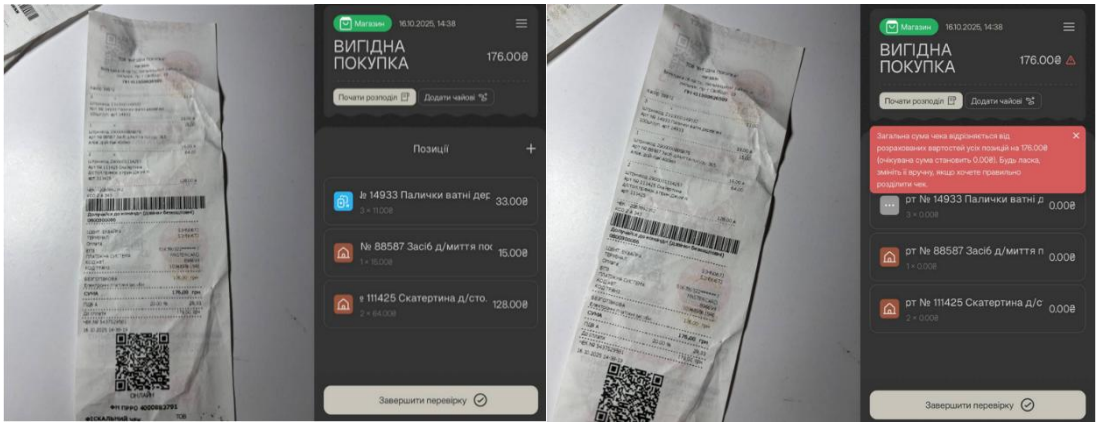


Рисунок 1.2 – Приклад роботи OCR з повернутим та рівним текстом в Tesseract

Хоча в більшості випадків сучасні OCR та LLM добре справляються з розпізнаванням тексту, але дослідження показують, що на повернутих зображеннях вони частіше помиляються [52] (рис. 1.3 б)). Особливо складною задачею є чеки, так як існує безліч їх варіантів та часом текст на них є низькоякісним, дрібним або вигорівшим [29, 53]. На рисунках показано результати роботи LLM в застосунку з рівним та спотвореним зображенням. В першому випадку на рисунку 1.3 а) текст на фото є горизонтально вирівняним, тому система точно та без помилок розпізнала і перенесла дані.

На рисунку 1.3 б) чек знаходиться нахиленим під кутом, що безпосередньо відобразилося на результаті роботи LLM: частина тексту, ціна,

не перенеслася коректно. Це показує, що навіть на розпізнавання тексту LLM моделями можуть впливати перспективні спотворення зображення, хоча значно меншою мірою, ніж на класичні OCR.



а) б)

Рисунок 1.3 – Приклади впливу нахилу чеку на розпізнавання тексту:
а) чек вирівняний; б) чек знаходиться під кутом

1.4 Огляд наявних програмних застосунків

Для підготовки фото до розпізнавання системами OCR і безпосередньо вирішення проблеми нахилу тексту розроблено різні технології, наприклад, Leptonica, OpenCV, ImageMagick, unpaper [48].

Leptonica – це відкрита бібліотека, що широко застосовується для попередньої обробки зображень перед подаванням його в системи OCR. Вирівнювання документа виконується із застосуванням алгоритму Постл, який виявляє кути нахилу ліній тексту. Проекції пікселів, одержані під різними кутами, є сумами ліній. Різниця між проекціями пікселів на суміжних лініях растра, піднесена до квадрата та підсумована, набуває максимального значення тоді, коли вибраний кут збігається з кутом нахилу [54].

В OpenCV застосовується алгоритм Кенні для знаходження контурів на зображені, використовуючи функцію для знаходження темних контурів і за

ними виділяє найбільший чотирикутник, ймовірний документ, із застосуванням Hough Line Transform, який виявляє рівні лінії [42–44]. Після знаходження 4 кутів застосовуються функції для перспективного перетворення зображення [45].

ImageMagick для корекції кута зображення застосовує вбудовану функцію, параметр визначає поріг допустимого викривлення. Щоб встановити перекіс аналізуються вершини і спади схожих на літер об’єктів [46].

Unpaper автоматично вирівнює прямокутні зони з текстом на зображені, ефективніше працює на зображеннях з рівномірними і чітко виділеними краями. По всім краям документа, прямокутній області, від зовнішнього до внутрішнього існує так звана «віртуальна лінія». Лінія поступово обертається на різні кути і рухається всередину області, по периметру цієї лінії на кутах алгоритмом підраховується кількість темних пікселів і обирається кут з найбільшою різницею між кількість темних пікселів з сусідніми позиціями [47]. Результати з порівнянням технологій для вирівнювання документів подано в таблиці 1.1.

Таблиця 1.1 – Порівняння технологій для вирівнювання документів

Технологія	**Основні алгоритми**	**Гнучкість**	**Призначення**	**Точність**
1	**2**	**3**	**4**	**5**
Leptonica	алгоритму Постл	Середня	OCR-підготовка	Висока для тексту
OpenCV	Кенні + корегування перспективи	Дуже висока	Комп’ютерний зір	Дуже висока
ImageMagick	Аналіз вершин та спадів символів	Висока	Масова конвертація	Середня

Продовження таблиці 1.1

1	2	3	4	5
Unpaper	«віртуальна лінія»	Низька	Очищення та вирівнювання сканів	Висока для сканованого тексту, не універсальна

Розглянемо існуючі застосунки, які розпізнають кути документа і вирівнюють зображення по ним, зосередимося на найпопулярніших – це Microsoft Lens, сканування документів в Нотатках IOS, Adobe Scan. Перевагою обраних програм є те, що вони безкоштовні і якісно вирівнюють зображення.

Microsoft Lens використовує покращений алгоритм Кенні поєднаний зі стандартним Pb методом для виявлення кутів документів, визначаючи на фото межі із високою контрастністю [55, 56]. Програма працює найгірше серед представлених: виділяє чеки та документи, але тільки на темному контрастному фоні, загалом не дуже неякісно виділяє межі, тому не виходить одразу якісно вирівняти зображення і доводиться самостійно редагувати кути на фото.

Сканування документів є частиною функціоналу Нотаток в IOS і в них так само використовуються технології пошуку контурів, які працюють на різних документах, чеках та іншому. Виявлення кутів iOS базується на алгоритмі Кенні і використовується Apple Vision та Core Image у поєднанні з пошуком контурів на основі машинного навчання [57]. Недоліком програми є доступність виключно на пристроях з системою IOS. Ще однією проблемою для цього застосунку можуть стати фото документів на білому фоні при слабких тінях або фото на фоні інших документів, тоді самостійно потрібно буде скорегувати кути на фото.

Точно не відомо які алгоритми для розпізнавання кутів застосовує Adobe Scan, але, ймовірно, це комбінація розповсюдженого Кенні та підсилена моделлю сегментації на основі глибокого навчання для розпізнавання

документів в реальному часі. Наскільки відомо вони використовують власний ШІ Adobe Sensei для ідентифікації типу документа та виділення кутів на фото [58]. Застосунок дуже якісно працює, як на чеках, так і на інших документах, хоча все ж стикається з проблемою, коли на фоні з ним розташовані інші документи, тоді алгоритм не може чітко виділити необхідний об’єкт і потребує втручання людини. Adobe Scan може підлаштовувати детекцію та вирівнювати не тільки документ, а і додатково інші об’єкти, такі як: книги, бізнес картки, ID картки та дошки – але це все доступно лише в платній версії застосунку. В таблиці 1.2. наведені результати порівнянь програм для вирівнювання документів.

Таблиця 1.2 – Порівняння програм для вирівнювання документів

Програми	**Основні алгоритми**	**Якість**	**Зручність**	**Гнучкість**
1	**2**	**3**	**4**	**5**
Microsoft Lens	Кенні + Pb метод	Середня	Висока, простий інтерфейс, швидке сканування	Підтримка сканування документів, чеків, дошок, вбудоване розпізнавання OCR
сканування документів в Нотатках IOS	Кенні + вбудовані бібліотеки (Apple Vision та Core Image)	Висока	Висока, одразу інтегровано в IOS	Підтримка сканування документів, чеків, фото, але наявна тільки на пристроях IOS

Продовження таблиці 1.2

1	2	3	4	5
Adobe Scan	Кенні + ШІ Adobe Sensei	Дуже висока	Висока, сучасний зрозумілий інтерфейс	Документи, інше: книги, бізнес і ID картки та дошки – платно

Отже, перед розпізнаванням тексту системами OCR необхідно вирівняти зображення для покращення якості результату, якщо попередньо програми не мають цього як вбудованої функції. Тому проаналізувавши наявні альтернативи для вирішення проблеми нахилу тексту на зображенні пропонується застосувати моделі YOLO для виявлення меж чеків, з них виділити кути по яким виконати перспективне вирівнювання зображення.

1.5 Постановка задачі

Розпізнавання і вирівнювання чеків та інших документів є актуальним як частина процесу попередньої обробки зображень перед застосуванням систем OCR. Це автоматизує та пришвидшить роботу для користувачів, замість самостійного вирівнювання зображень та повинно покращити точність розпізнавання тексту.

Об’єктом роботи є використання YOLO-моделей для детекції зображень чеків та подальшого вирівнювання.

Метою роботи є розроблення андроїд застосунку, здатного визначати чеки на фотографіях за допомогою донавченої YOLO-моделі та вирівнювати зображення з використанням бібліотеки комп’ютерного зору OpenCV.

Для досягнення мети необхідно вирішити такі завдання:

− проаналізувати сучасні підходи для розпізнавання чеків;

− сформулювати вимоги до вхідних даних і критерії вибору моделей;

− сформувати набір даних;

− натренувати моделі;

− порівняти різні техніки розпізнавання;

− порівняти різні моделі YOLO;

− застосувати OpenCV для вирівнювання чеку;

− проаналізувати вплив вирівнювання на точність розпізнавання тексту;

− створити андроїд застосунок, який інтегрує блок розпізнавання чеків та блок вирівнювання чеків;

− протестувати та оцінити роботу застосунку.

2 АНАЛІЗ ТА ВИБІР МЕТОДУ ПЕРСПЕКТИВНОЇ КОРЕКЦІЇ ЧЕКІВ ІЗ ЗАСТОСУВАННЯМ YOLO-МОДЕЛЕЙ

2.1 Хід виконання аналізу

Головною задачею є підвищення точності розпізнавання тексту системами OCR та сучасними LLM шляхом зменшення впливу нахилу тексту завдяки перспективній корекції зображень. Для цього в роботі проаналізується ефективність та доцільність застосування YOLO-моделей для автоматичного виявлення меж чеку. Оціниться якість вирівнювання зображення із застосуванням поєднання YOLO-моделей та OpenCV. Перевіриться істотність впливу нормалізованого зображення на результат роботи систем розпізнавання тексту [59].

Вирішувати задачу пропонуються за наступними кроками:

Крок 1. Виявлення меж чеку з використанням YOLO-моделей, зокрема OBB та сегментації.

Крок 2. З виявлених меж чеку визначити 4 кути та впорядкувати їх.

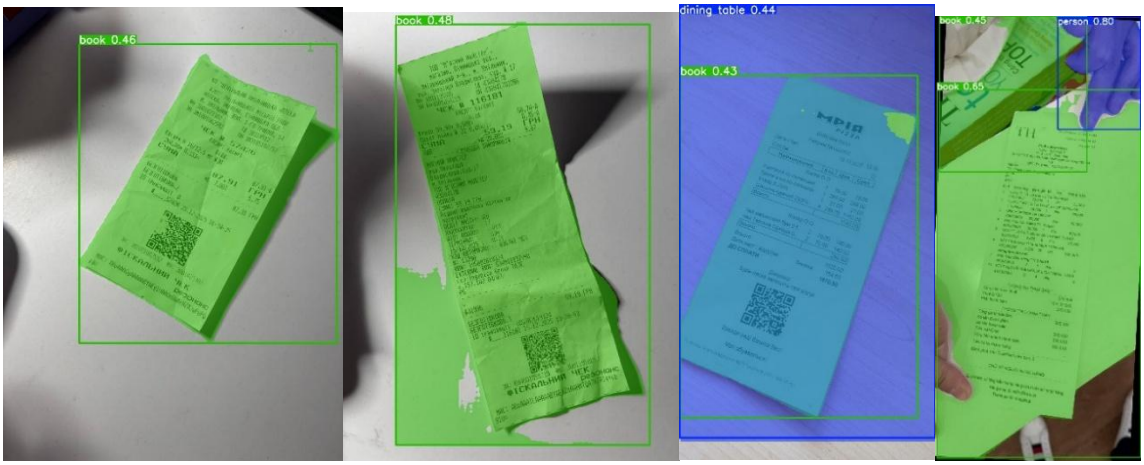
Крок 3. Перспективно скорегувати зображення чеку по знайденим 4 кутам з OpenCV.

Крок 4. Отримати нормалізований чек та перевірити вплив на системи OCR та LLM.

2.2 Реалізація набору даних та тренування моделей

Протестувавши наявні моделі було виявлено, що вони не натреновані для розпізнавання безпосередньо чеків. Зазвичай моделі сегментації розпізнають об’єкт як книги, але не виділяють точні межі чеку, додатково виділяючи тіні, або зайвий фон (рис. 2.1 а), б)). В випадках коли на фоні розташовано багато предметів: книги, стіл, руки – модель розпізнає і їх

(рис. 2.1 в), г)). Хоча бувають винятки з майже коректним розпізнаванням меж чеку, все таки це не є закономірністю і на це не потрібно покладатися. Впевненість моделі в таких ситуаціях здебільшого не є високою. При застосуванні OBB моделі на зображеннях абсолютно нічого не розпізнавалося, що часом траплялося і з використанням моделей сегментації. Виходячи з отриманих даних для початку роботи необхідно зібрати набір даних та розмітити його.



а) б) в) г)

Рисунок 2.1 – Розпізнавання чеку YOLO-моделями сегментації без донавчання:
а) виділяє межі чеку, але трішки захоплює тінь; б) виділяє білий стіл як чек; в), г) не розпізнає межі чеку взагалі, виділяє зайві об’єкти

Збір і розмітку набору даних виконуватимемо в програмі Roboflow. Необхідно знайти достатньо якісних і різноманітних зображень чеків для точнішої роботи моделі, приклади наведені на рисунку 2.2. Набір даних повинен містити чеки під різним нахилом, фоном, освітленням та від різних магазинів. Це необхідно щоб не перенавчити модель, де вона запам’ятовуватиме конкретні навчальні приклади, замість того щоб засвоїть загальні закономірності для хорошої роботи на реальних чеках. Тому для якіснішого розпізнавання рекомендується більша кількість чеків. Загальна кількість зібраних фото становить 2694 штук.

Рисунок 2.2 – Приклад зображень чеку з набору даних

Наступним етапом є анотування даних (рис. 2.3). Назва класу анотації receipt, обраний тип розмітки, полігон, гарантує свободу в розмітці та охоплення точніших форм чеку. Для пришвидшення роботи застосовувався інструмент «Smart Select», який при наведенні на чек самостійно виділяв його межі, при неточностях анотація корегувалася вручну. Для комп’ютера анотовані дані зберігаються в форматі .json.

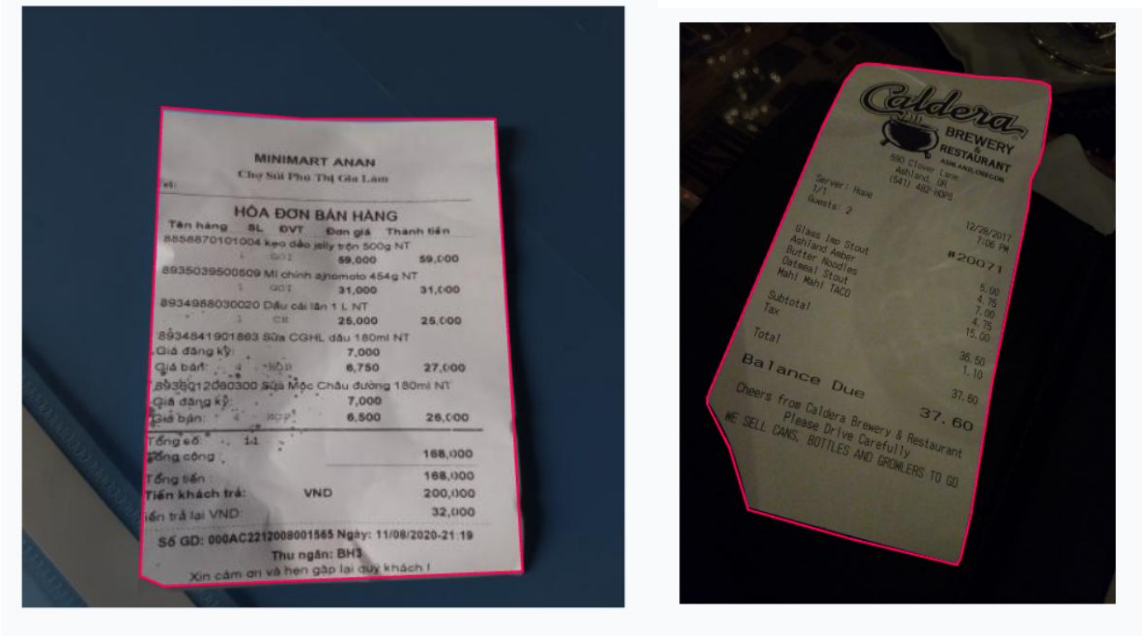


Рисунок 2.3 – Приклад анотації чеків полігонами

Лістинг 2.1 Приклад реалізації анотації для зображення:


```
{ "boxes": [
      {"id": "1",
        "type": "polygon",
        "label": "receipt",
        "x": "200.4117",
        "y": "179.9827",
        "width": "342.2535",
        "height": "217.7774",
        "points": [ [ 371.538, 285.794 ],
          [ 363.18, 71.094 ],
          [ 31.343, 74.141 ],
          [ 29.285, 75.156 ],
          [ 30.615, 288.871 ] ] }],
    "height": 416,
    "key":
"mcocr_public_145013clfsp_jpg.rf.8864b6789a986ea90a40194b425a8e3d.jpg",
    "width": 416}
```


Для пришвидшення розмітки застосовувалася функція автоматичної анотації зображень із застосуванням моделі SAM3 та промпт. В ньому випробовувався різний текст, як наприклад «rectangular or square white papers with text on them» або «reciept, rectangular or square piece of paper with text on it». Модель якісно виділяла контур чеків, що показано на рисунку 2.4, але з набору зображень менше половини були розмічені, зміна тексту запитів не покращила результат. Не зважаючи на цей недолік, функція значно пришвидшила роботу розмітки даних.

Усі зображення з набору даних були розділені на тренувальні, валідаційні та тестові набори у співвідношенні 70:20:10 відповідно (рис. 2.5).

Тренувальний набір з 1888 зображень буде використовуватися для навчання моделей. Валідаційний з 540 зображеннями використовуватиметься моделлю на етапі формування та доопрацювання для допомоги визначення кращої версії та результатів. Дані з тестового набору, 266 зображень, не надаються моделі протягом створення і потрібні для чесної, об’єктивної оцінки роботи моделі в реальних умовах.

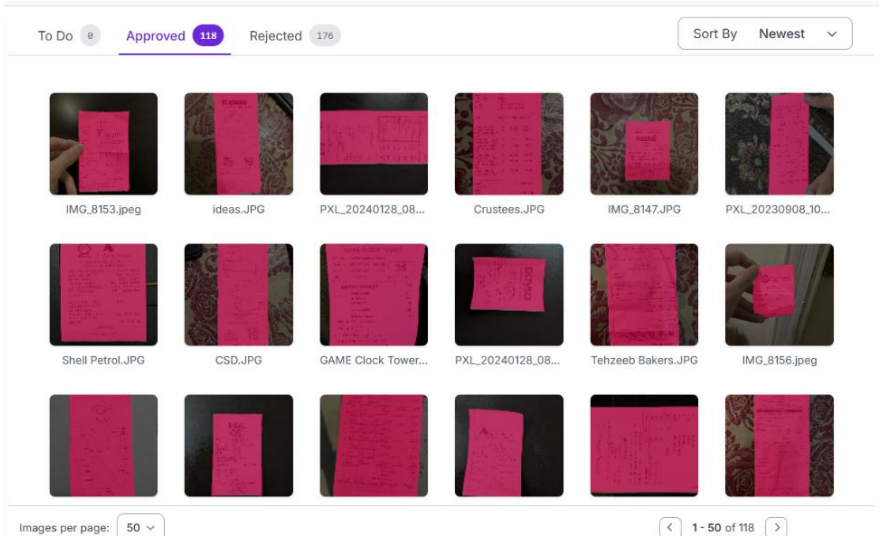


Рисунок 2.4 – Приклад автоматичної розмітки чеків на зображенні

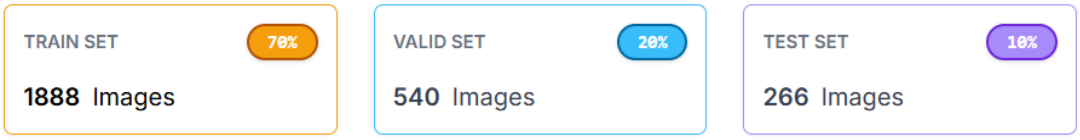


Рисунок 2.5 – Візуалізація розділення зображень на набори в наборі даних

Наступним до даних застосовується аугментація – штучне збільшення об’єму зображень тренувального набору даних шляхом застосування модифікацій на існуючих даних. Конкретно для цього було застосовано поворот на 90 по часовій стрілці і проти, нахил на -15° та +15°, вертикальний та горизонтальний зсув на ±10°, шум до 0,1% пікселей (рис. 2.6). Після застосування процесу кількість зображень в тренувальному наборі збільшилася в 3 рази з 1888 до 5664 (рис. 2.7).

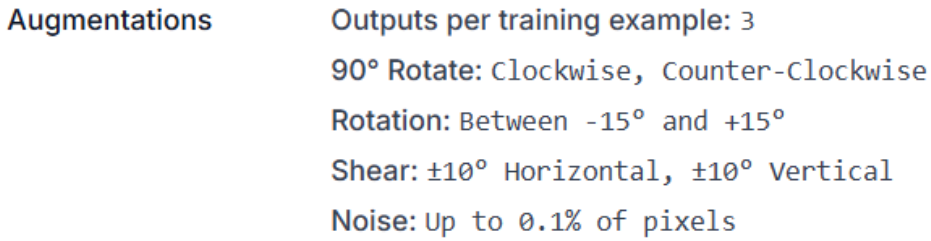


Рисунок 2.6 – Застосовані параметри аугментації для набору даних

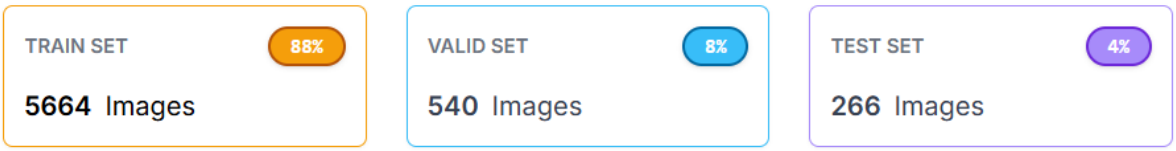


Рисунок 2.7 – Відношення між різними наборами після аугментації

Подальшим етапом є навчання YOLO-моделей на зібраному наборі даних. Структура файлів набору даних для тренування зображена на рисунку 2.8. Для навчання моделей використовувалося 35 епох та виставлено терпіння, кількість епох без прогресу в навчанні, 10 епох. Після завершення тренування отримали файл в форматі best.pt та інші дані з навчання.

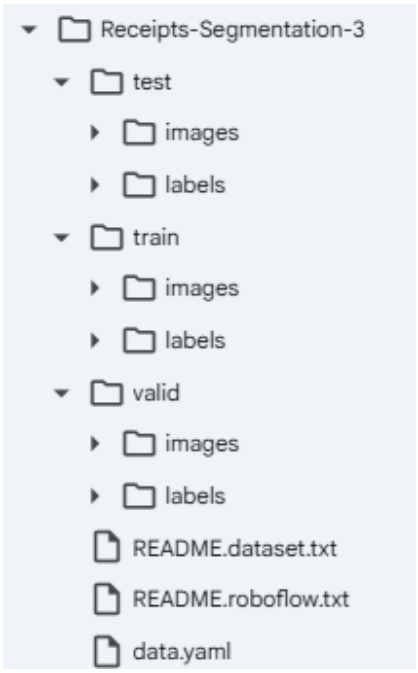


Рисунок 2.8 – Структура файлів набору даних для тренування моделей

2.3 Порівняння способів виділення чеків

Необхідно проаналізувати різні методи виділення: традиційна детекція, OBB та сегментація і обрати ті, що найкраще виділять межі чеку.

Традиційна детекція використовує для виділення об’єкта форму прямокутника вирівняного по осі з параметрами *x*, *y*, *w*, *h*, добре підходить для виявлення об’єктів, але не в нашому випадку. Вона не вирішує проблему нахилу, так як виділяється вся область з чеком і обмежувальний бокс не підлаштовується під нахил чеку, приклад представлено на рисунку 2.9.

Рисунок 2.9 – Приклад роботи традиційної детекції для виявлення чеків

Проблему нахилу вирішує OBB, який для виводу застосовує повернутий прямокутник і крім параметрів *x*, *y*, *w*, *h* має п’ятий, який визначає кут нахилу виявленого об’єкту. Використовуючи OBB можна істотно зменшити нахил чеку на фото, але бокс часом не зовсім правильно підбирає кут відповідно до реального нахилу на фото, тому незначний кут може залишитися (рис. 2.10).

Рисунок 2.10 – Приклад роботи OBB для виявлення чеків

Сегментація використовує форму полігону на рівні пікселів з параметрами у вигляді вершин полігону, що дозволяє найчіткіше окреслити контур чеків (рис. 2.11). З добре виявленими межами чеку, особливо кутами, вийде найкраще перспективно скорегувати нахил чеку, не тільки по осі, але і виправити горизонтальний та вертикальний нахил.

Рисунок 2.11 – Приклад роботи сегментації для виявлення чеків

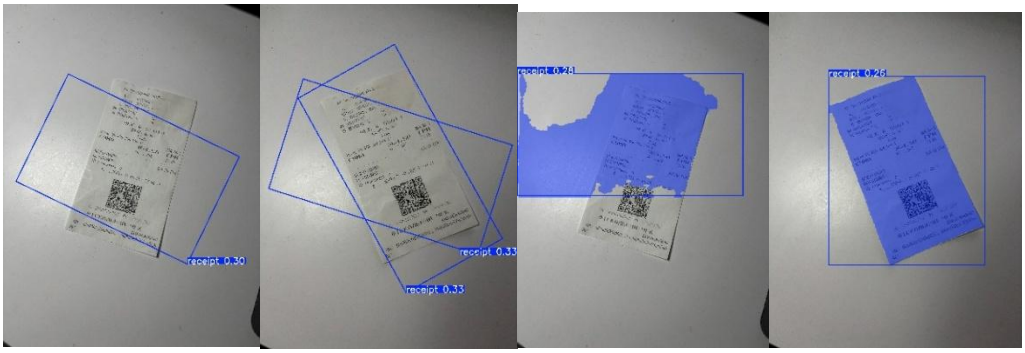
2.4 Вплив фону на розпізнавання чеку

При розпізнаванні чеків проблемою міг ставав різний фон. Найкраще себе показали моделі на темному, контрастному фоні, або будь-якому не білому (рис. 2.12). Сегментація найкраще виділяла межі чеку, а OBB найточніше вписувала чек в обмежувальний бокс і найкраще підбирала кут. Впевненість моделей зазвичай дуже висока і перевищує 90%.

а) б) в) г)

Рисунок 2.12 – Приклад розпізнавання на контрастному до чека фоні: а), в) obb; б), г) сегментація

Білий фон найбільше негативно вплинув саме на OBB моделі. Обмежувальний бокс не міг зрозуміти де саме розташований чек на фото, тому виділяв фон або видавав неправильний кут нахилу (рис. 2.13 а), б)). При сегментації моделі зазвичай не стикалися з такими проблемами і коректно виділяли межі чеку (рис. 2.13 г)), але бували випадки з помилковим виявленням чеків (рис. 2.13 в)). Впевненість моделей на білому фоні значно знижається і може становити менше 50% (рис. 2.13). Після додавання в набір даних фото без чеків, але зі схожими об’єктами точність виділення чеків моделями OBB та сегментації значно зросла, або це пов’язано з перетренуванням моделей (рис. 2.14). На збільшення впевненості моделі впливає розмір, чим він більше, тим вище впевненість.



а) б) в) г)

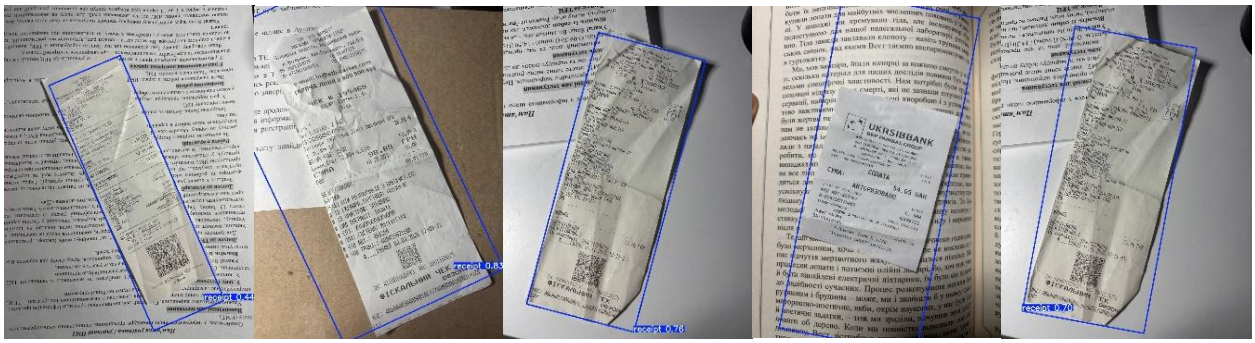
Рисунок 2.13 – Приклади помилкового визначення меж чеків на білому фоні: а), б) obb; в), г) сегментація

а) б) в) г)

Рисунок 2.14 – Приклади успішного розпізнавання меж чеків на білому фоні: а), б) obb; в), г) сегментація

Інші документи на фоні чеку створюють сильну проблему, особливо якщо вони повністю охоплюють фон поза чеком (рис. 2.15 г), рис. 2.16 а)) і мають схожий шрифт. Тоді вони сприймаються моделями частиною чеку та виділяються разом з ним. В залежності від документа на фоні та фото, модель може і вдало виділити чек (рис. 2.15 а), 2.16 г)).

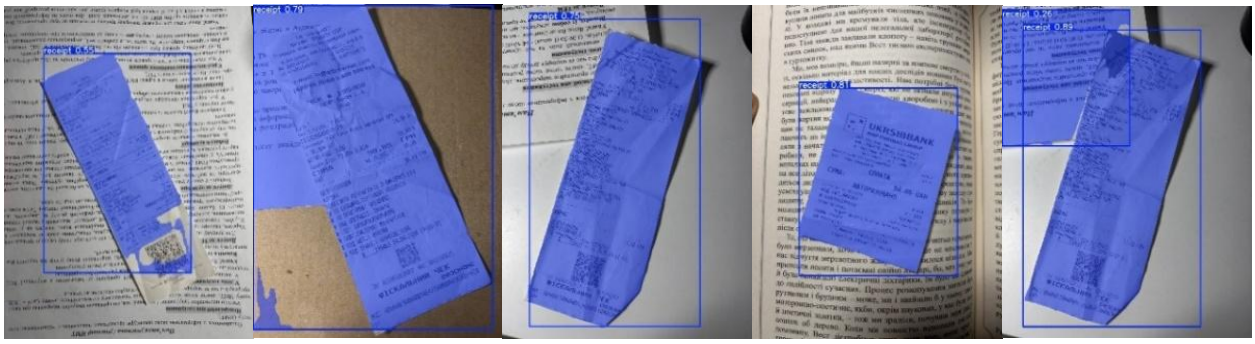
Якщо документи присутні частково, моделі можуть їх виділяти як інший чек або не виділяти і на точність виділення меж чеку це не впливатиме (рис. 2.15 в), д), 2.16 в), д)), але бувають ситуації коли модель сприймає частину іншого документа як продовження чеку (2.15 б), 2.16 б)).



а) б) в) г) д)

Рисунок 2.15 – Приклади розпізнавання чеків ОВВ моделями з документами на фоні:

а), г) повністю на фоні документа; б), в), д) частково на фоні документа



а) б) в) г) д)

Рисунок 2.16 – Приклади розпізнавання чеків моделями сегментації з документами на фоні:

а), г) повністю на фоні документа; б), в), д) частково на фоні документа

2.5 Порівняння різних YOLO-моделей

Результат аналізу порівняння різних моделей YOLO занесено в таблицю 2.1 та рисунок 2.17.

Таблиця 2.1 – Порівняння данних різних YOLO-моделей

Модель	Втрати боксів	DFL втрати	Втрати класу	Втрати сегментації	Точність тестування	Кількість параметрів	Розмір моделі(MB)	Час на передбачення
1	2	3	4	5	6	7	8	9
11n-obb	0,3	0,986	0,29	-	98,98	2653918	5,7	0,19
11s-obb	0,30	0,988	0,29	-	98,84	9699174	19,2	0,46
26n-obb	0,29	0,174	0,24	-	98,36	2446602	5,7	0,2
26s-obb	0,37	0,202	0,36	-	98,69	9751554	20,8	0,48
11n-seg	0,2	0,705	0,21	0,19	98,33	2834763	5,7	0,27
11s-seg	0,19	0,715	0,19	0,27	98,39	10067203	19,5	0,66
26n-seg	0,22	0,008	0,14	0,34	98,25	2689079	6,2	0,25
26s-seg	0,25	0,009	0,19	0,28	98,49	10365727	22,2	0,67
26m-seg	0,23	0,008	0,17	0,26	98,27	23508239	51,9	2,02

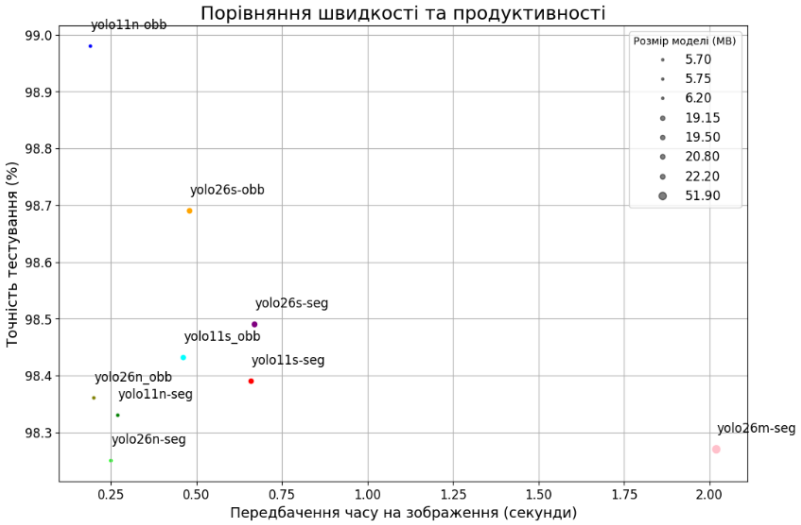


Рисунок 2.17 – Порівняння швидкості, продуктивності та розміру моделей

Втрати боксів відображають похибку в прогнозуванні розташування обмежувального бокса порівняно з вихідними даними. DFL втрати вимірюють точність прогнозу координат меж чеку. Втрати класу визначають наскільки точно модель визначає категорію об'єкта. Втрати сегментації відображають точність моделі визначати контур чека за допомогою маски сегментації. Зі збільшенням розміру моделі зростає кількість параметрів та час необхідний для виявлення чеку. Сегментація довша тому що виявляє межі з використанням полігону, а OBB з 4 кутами та кутом нахилу об'єкта. Точність всіх проаналізованих моделей становить вище 98%, що сигналізує про якісний набір даних.

$$Box\ Loss = 1 - IoU + \frac{\rho^2(b_{p1}, b_g)}{c^2} + \alpha v, \tag{2.1}$$

де IoU – це відсоток накладання;

α – це позитивний параметр трасування;

v – вимірює узгодженість співвідношення сторін.

$$DFL\ Loss(S_i, S_{i+1}) = -((y_{i+1} - y)\log(S_i) + (y - y_i)\log(S_{i+1})), \tag{2.2}$$

де y – це цільова координата;

S_i, S_{i+1} – це прогнозовані ймовірності для двох найближчих інтервалів.

$$Cls\ Loss = -[y\ \log p + (1 - y)\log(1 - p)], \tag{2.3}$$

де y – це істинне значення 0 або 1;

p – це ймовірність для якогось з класів.

$$Seg\ Loss = 1 - \frac{2|X \cap Y|}{|X| + |Y|}, \tag{2.4}$$

де X – передбачення;

Y – істинна.

В подальших підглавах для кожної моделі буде представлено матрицю невідповідностей та графіки результатів тренування.

2.5.1 YOLO 11n obb

На рисунку 2.18 зображена матриця невідповідності, де з усіх чеків 97% розпізналося правильно, а хибно – усього 3%. На рисунку 2.19 розташовано графік результатів навчання для моделі YOLO 11 nano obb.

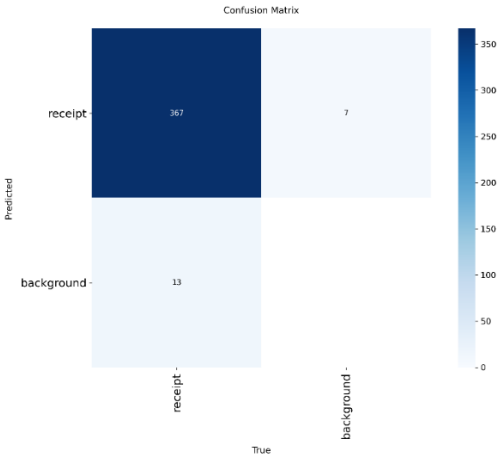


Рисунок 2.18 – Матриця невідповідності для YOLO 11 nano obb

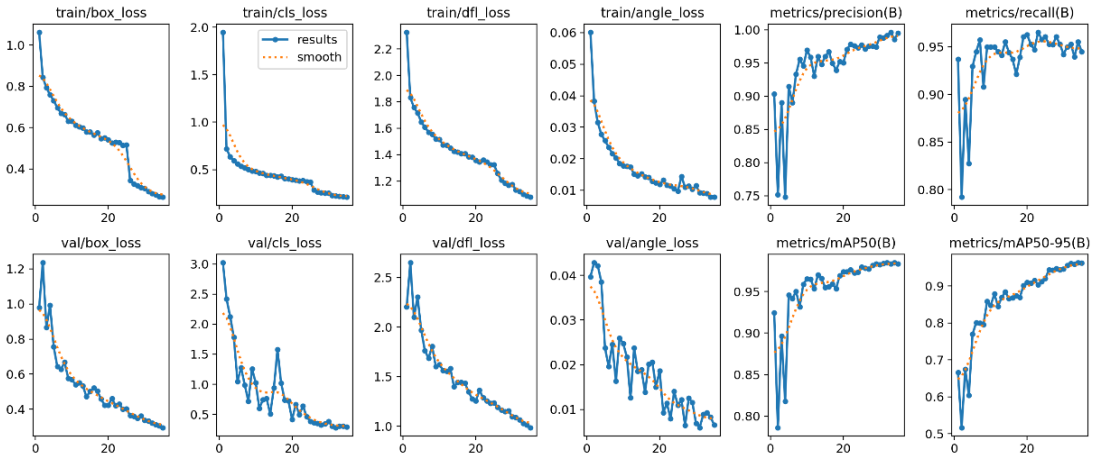


Рисунок 2.19 – Графіки результатів навчання YOLO 11 nano obb

2.5.2 YOLO 11s obb

На рисунку 2.20 зображена матриця невідповідності, де з усіх чеків 97% розпізналося правильно, а хибно – усього 3%. На рисунку 2.21 розташовано графік результатів навчання для моделі YOLO 11 small obb з втратами, точністю та іншими метриками.

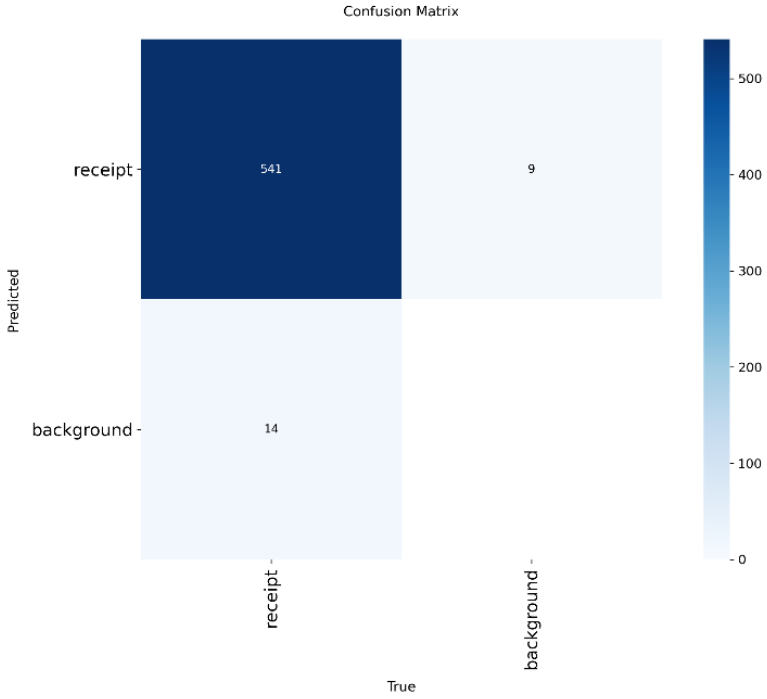


Рисунок 2.20 – Матриця невідповідності для YOLO 11 small obb

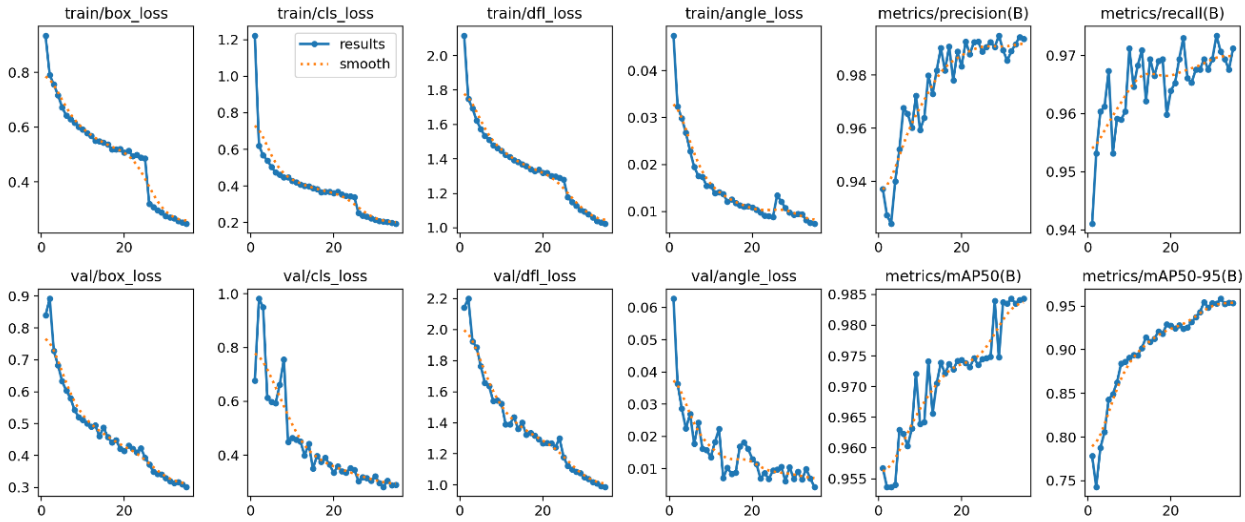


Рисунок 2.21 – Графіки результатів навчання YOLO 11 small obb

2.5.3 YOLO 26n obb

На рисунку 2.22 зображена матриця невідповідності, де з усіх чеків 97% розпізналося правильно, а хибно – усього 3%. На рисунку 2.23 розташовано графік результатів навчання для моделі YOLO 26 nano obb з втратами, точністю та іншими метриками.

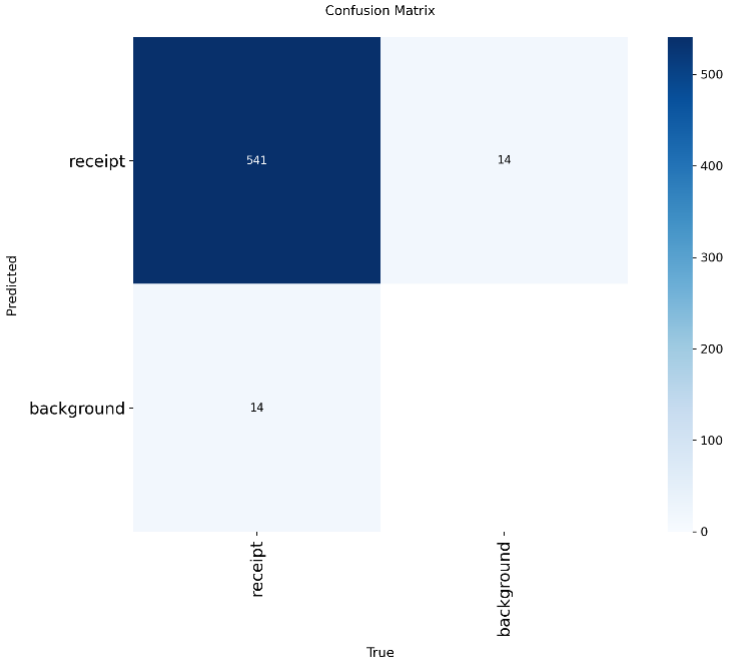


Рисунок 2.22 – Матриця невідповідності для YOLO 26 nano obb

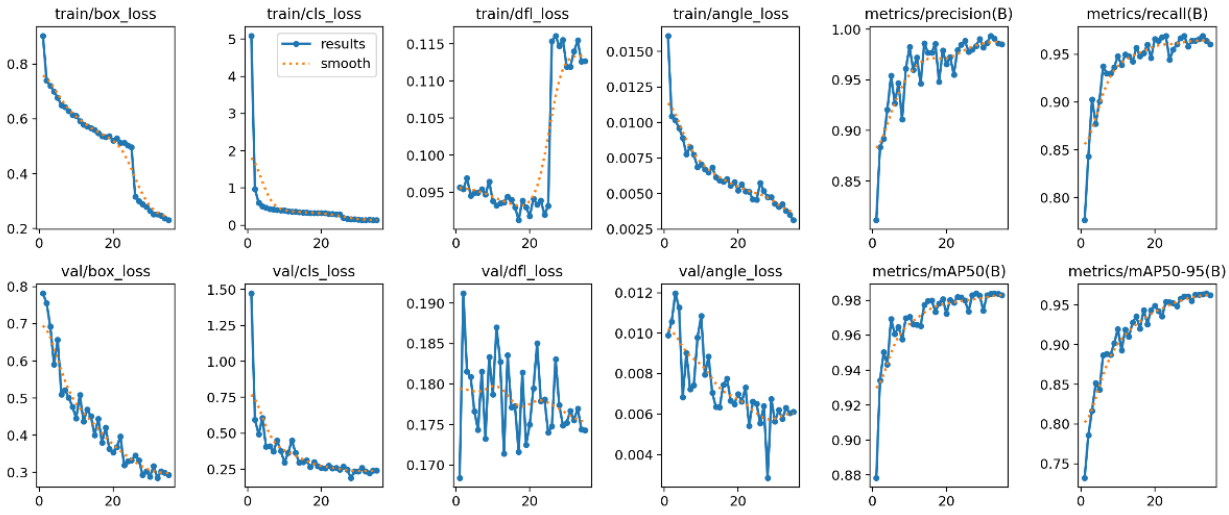


Рисунок 2.23 – Графіки результатів навчання YOLO 26 nano obb

2.5.4 YOLO 26s obb

На рисунку 2.24 зображена матриця невідповідності, де з усіх чеків 96% розпізналося правильно, а хибно – усього 4%. На рисунку 2.25 розташовано графік результатів навчання для моделі YOLO 26 small obb з втратами, точністю та іншими метриками.

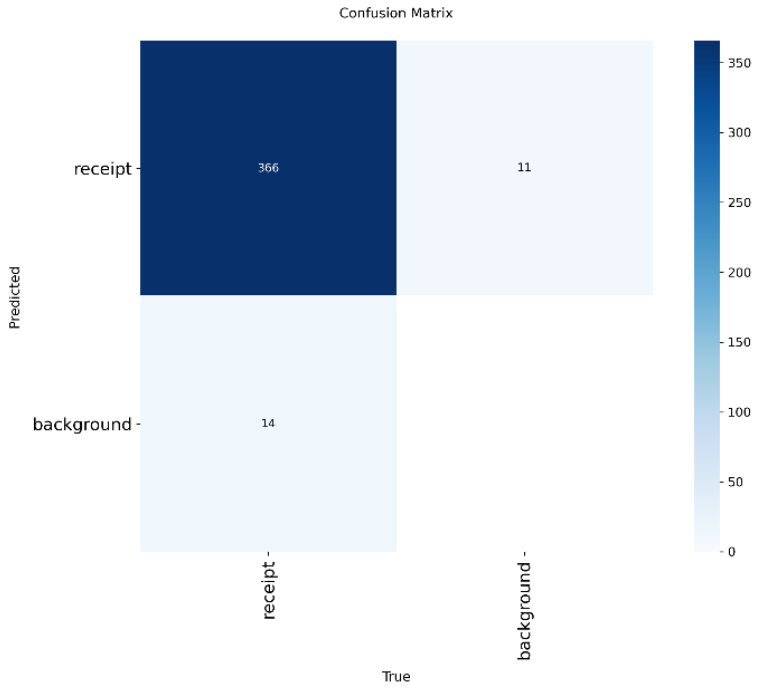


Рисунок 2.24 – Матриця невідповідності для YOLO 26 small obb

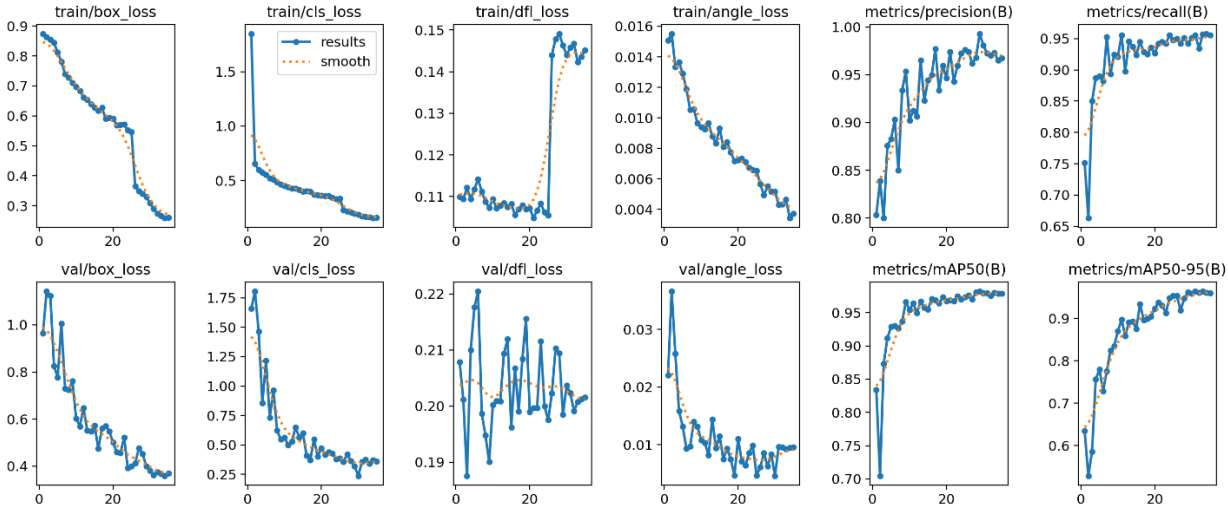


Рисунок 2.25 – Графіки результатів навчання YOLO 26 small obb

2.5.5 YOLO 11n seg

На рисунку 2.26 зображена матриця невідповідності, де з усіх чеків 97% розпізналося правильно, а хибно – усього 3%. На рисунку 2.27 розташовано графік результатів навчання для моделі YOLO 11 nano seg з втратами, точністю та іншими метриками.

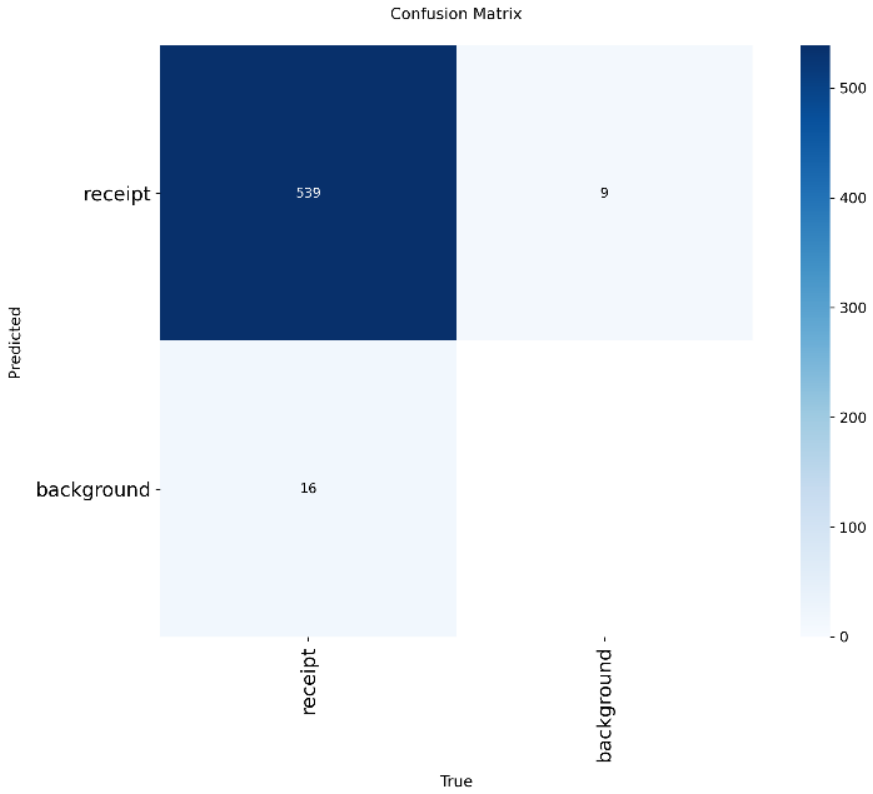


Рисунок 2.26 – Матриця невідповідності для YOLO 11 nano seg

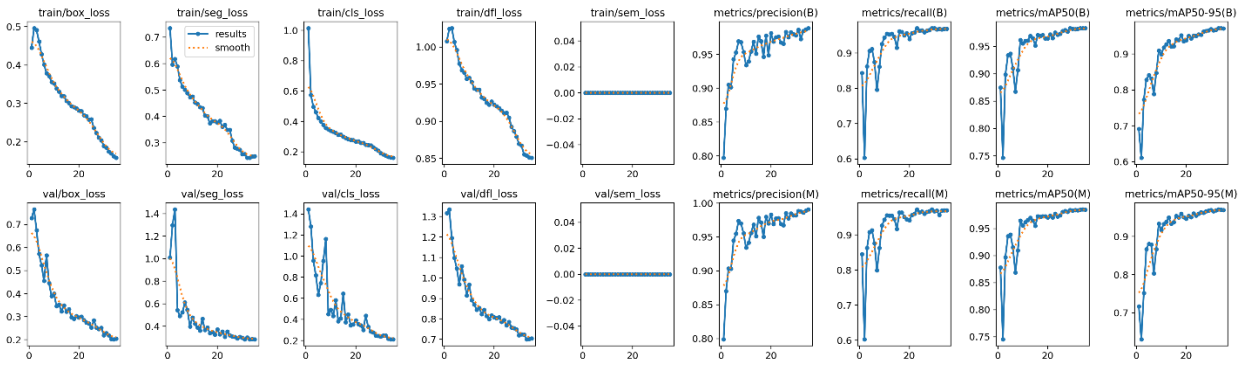


Рисунок 2.27 – Графіки результатів навчання YOLO 11 nano seg

2.5.6 YOLO 11s seg

На рисунку 2.28 зображена матриця невідповідності, де з усіх чеків 98% розпізналося правильно, а хибно – усього 2%. На рисунку 2.29 розташовано графік результатів навчання для моделі YOLO 11 small seg з втратами, точністю та іншими метриками.

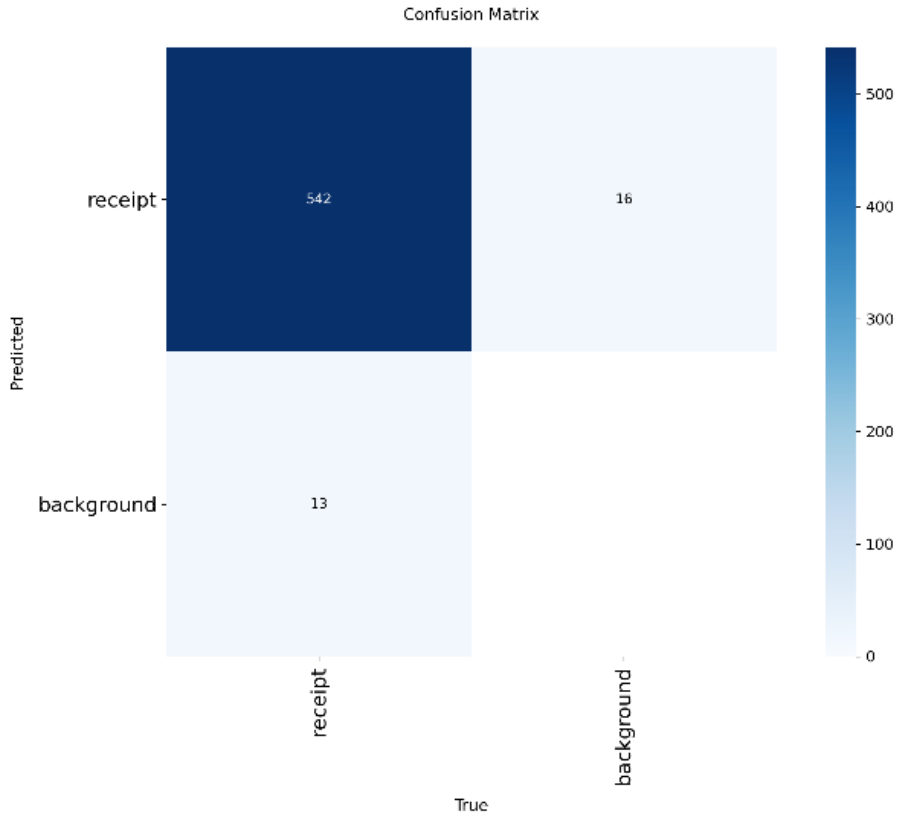


Рисунок 2.28 – Матриця невідповідності для YOLO 11 small seg

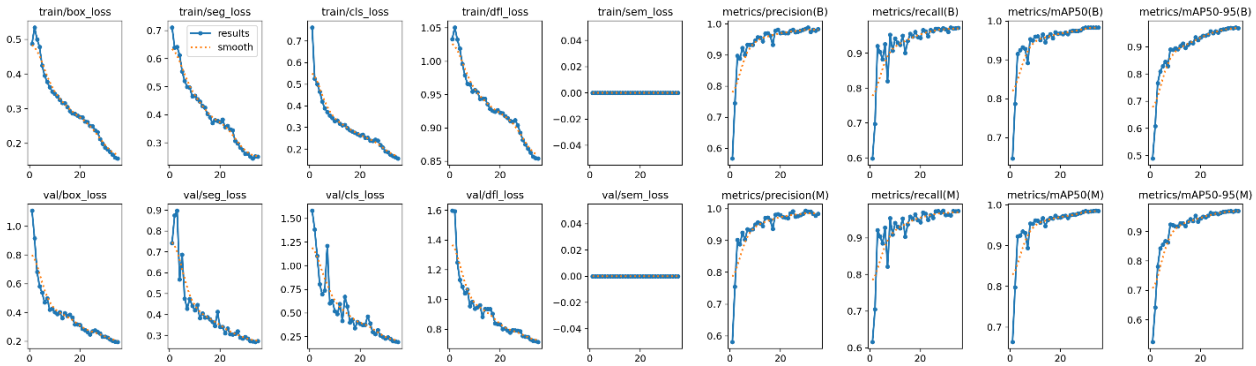


Рисунок 2.29 – Графіки результатів навчання YOLO 11 small seg

2.5.7 YOLO 26n seg

На рисунку 2.30 зображена матриця невідповідності, де з усіх чеків 98% розпізналося правильно, а хибно – усього 2%. На рисунку 2.31 розташовано графік результатів навчання для моделі YOLO 26 nano seg.

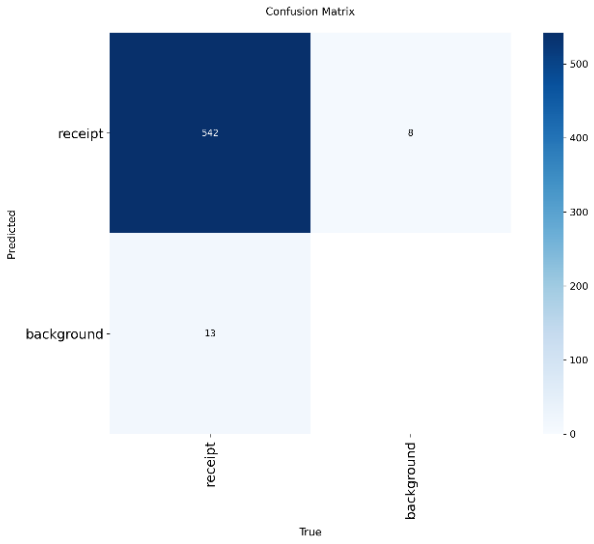


Рисунок 2.30 – Матриця невідповідності для YOLO 26 nano seg

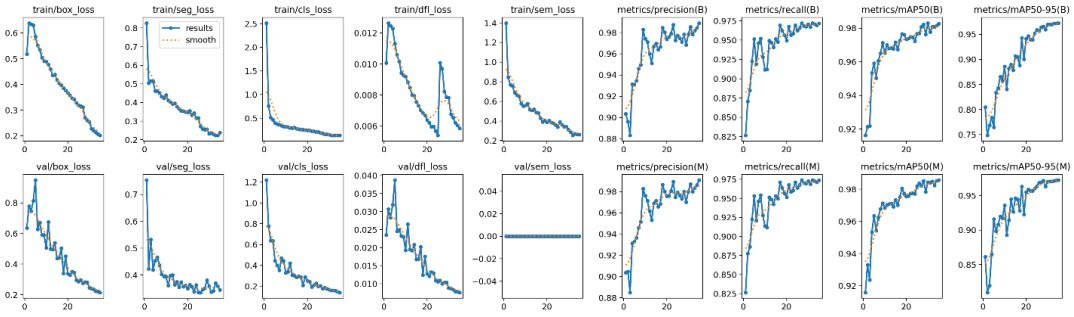


Рисунок 2.31 – Графіки результатів навчання YOLO 26 nano seg

2.5.8 YOLO 26s seg

На рисунку 2.32 зображена матриця невідповідності, де з усіх чеків 97% розпізналося правильно, а хибно – усього 3%. На рисунку 2.33 розташовано графік результатів навчання для моделі YOLO 26 small seg.

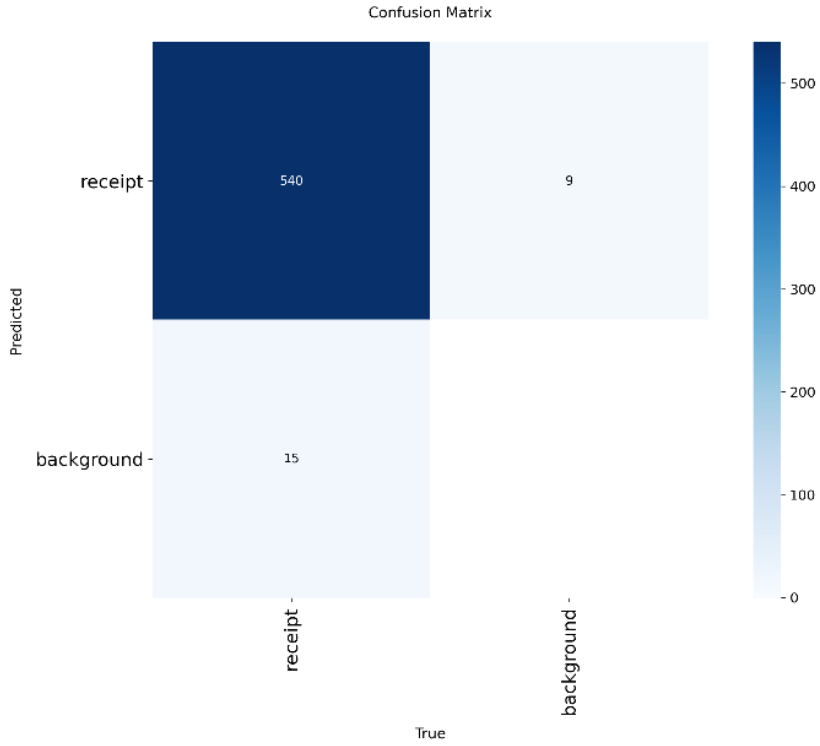


Рисунок 2.32 – Матриця невідповідності для YOLO 26 small seg

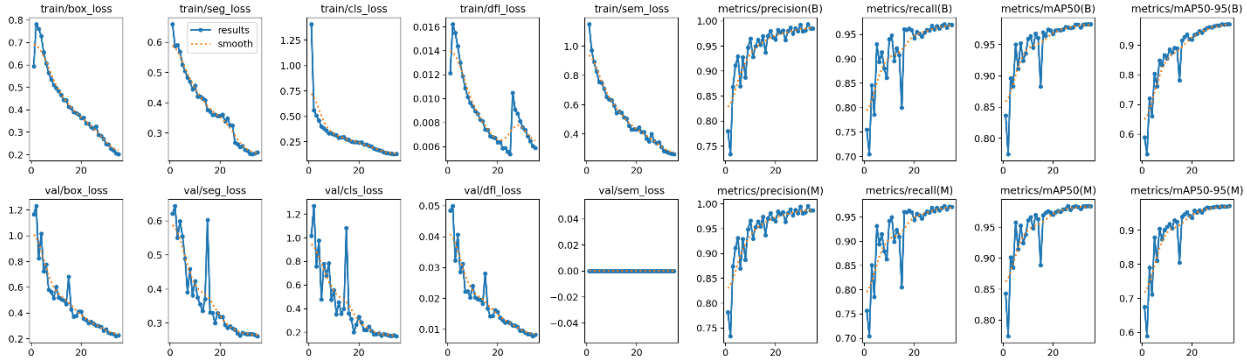


Рисунок 2.33 – Графіки результатів навчання YOLO 26 small seg

2.5.9 YOLO 26m seg

На рисунку 2.34 зображена матриця невідповідності, де з усіх чеків 97% розпізналося правильно, а хибно – усього 3%. а на рисунку 2.35 розташовано графік результатів навчання для моделі YOLO 26 medium seg з втратами, точністю та іншими метриками.

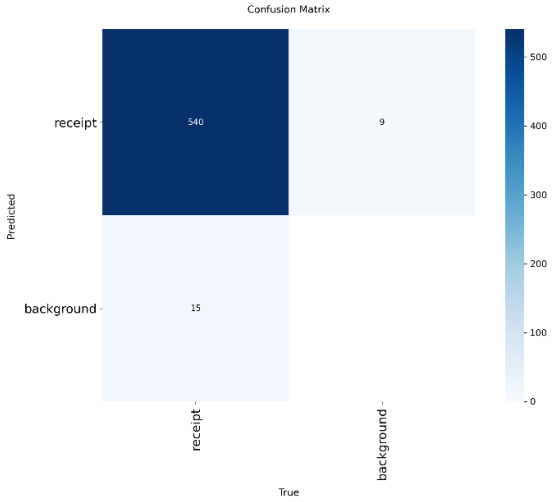


Рисунок 2.34 – Матриця невідповідності для YOLO 26 medium seg

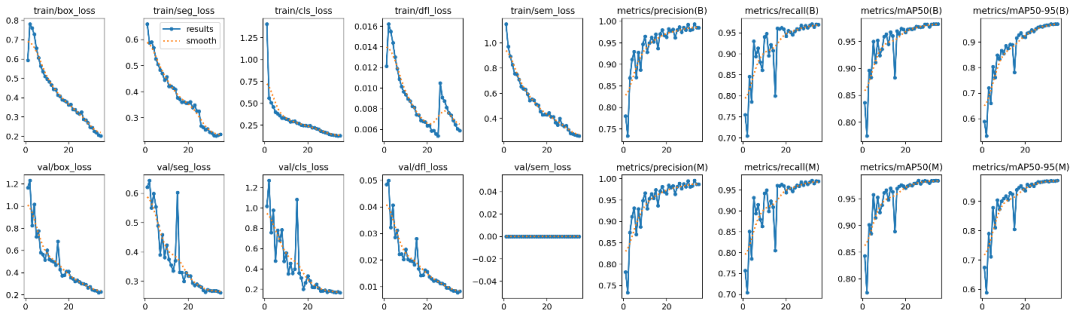


Рисунок 2.35 – Графіки результатів навчання YOLO 26 medium seg

2.6 Вирівнювання чеку

Для остаточної нормалізації чеку, необхідно знайти 4 кути, впорядкувати їх та перспективно скорегувати чек за кутами. Для реалізації використовуватимемо OpenCV [60].

Найлегше знайти 4 кути в OBB боксах, адже маємо готовий прямокутник з чіткими кутами у вигляді обмежувального бокса. По ньому залишається лише впорядкувати точки: права верхня, права нижня, ліва верхня та ліва нижня (рис. 2.36).

Для отримання 4 кутів з маски сегментації береться бінарна маска і з неї отримується індикаторна функція області чеку. Далі виконується

морфологічне замикання, що з’єднує розірвані краї та стабілізує контур. Контур розглядається як геометричний об’єкт і апроксимується полігоном. Якщо контур ідеально рівний отримуємо 4 вершини, якщо ні – більше. Якщо вершин більше знаходиться прямокутник мінімальної площі, який покриватиме контур, що гарантує 4 необхідні точки (рис. 2.37).

а) б) в) г)

Рисунок 2.36 – Вирівнювання чеку при обмежувальних боксах:
а), в) знайдені 4 точки; б) г) вирівняне за точками зображення

а) б) в) г)

Рисунок 2.37 – Вирівнювання чеку при сегментаційних масках:
а), в) знайдені 4 точки; б) г) вирівняне за точками зображення

За знайденими 4 точками відбувається проєктивне перетворення площини згідно з формулою (2.5), що стискає далекі частини і розтягує ближні [61].

Якщо порівняти вирівнювання при застосуванні сегментації та OBB, то саме сегментація трішки краще корегує спотворення чеку. Хоча бувають

випадки неточного виявлення меж при сегментації на краях, що може призводити до обрізання сторін самого чеку з необхідною інформацією.

$$\begin{bmatrix} tx' \\ ty' \\ t \end{bmatrix} = \begin{bmatrix} M_{11} & M_{12} & M_{13} \\ M_{21} & M_{22} & M_{23} \\ M_{31} & M_{32} & M_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}, \quad (2.5)$$

де x, y – координати точки на вхідному зображенні;

x', y' - координати відповідної точки в результаті після вирівнювання;

t – це масштабний коефіцієнт;

M_{ij} – це елементи матриці перспективного перетворення розміру 3×3.

2.7 Аналіз впливу вирівнювання на розпізнавання тексту

Практично перевіримо вплив повернутих зображень чеків на розпізнавання тексту системами OCR та LLM. Для оцінки точності результату LLM застосовуватимемо формулу 2.6. Для порівняння візьмемо різні моделі gemini, 50 зображень чеків з ground truth і результати занесемо в таблицю 2.2.

$$S = \frac{\frac{0,3}{H} + \frac{0,5}{I} + \frac{0,2}{T}}{100\%}, \quad (2.6)$$

де H – точність розпізнавання інформації з заголовка;

I – точність розпізнавання товарних позицій;

T – точність розпізнавання загальної суми.

Загалом помітно незначне загальне покращення точності від 0,77% до 5,46%. В деяких випадках підвищення точності сягало 20%, але бували випадки і погіршення точності після вирівнювання. Також протестовано вплив вирівнювання на точність розпізнавання тексту системами OCR, такими як Tesseract та EasyOCR. Для першої при наявності нахилу текст не розпізнавався взагалі, в другій – значно знижувалася точність (рис. 2.38).

Таблиця 2.2 – Порівняння точності розпізнавання чеку різними моделями

Модель	Рівне			Спотворене			Різниця
	avg	min	max	avg	min	max	
2.5 flash	86,32	64,61	99,87	83,67	65,99	98,72	2,65
2.5 flash lite	85,11	67,77	98,72	79,65	61,36	98,43	5,46
3.1 flash lite	87,83	73,31	98,53	87,06	63,45	98,5	0,77

а) б) в) г)

Рисунок 2.38 – Розпізнавання тексту Tesseract:
а) перспективно скорегований чек; б) результат розпізнавання рівного чеку;
в) нахилений чек; г) результат розпізнавання нахиленого чеку

2.8 Критерії обрання YOLO-моделі

Отже, аналіз показав, що YOLO-моделі дуже добре виділяють межі чеків, особливо це стосується зображень на контрастному фоні. З меншою впевненістю, але також високою точністю виділяються чеки на білому. З сильними проблемами, помилковими виявленням меж, модель стикається на фоні інших документів. Якщо в кадрі лише частина іншого схожого на чек об’єкта на модель це не впливає значною мірою, але за умови того, що документи займають більшу частину простору або весь та накладаються на чек, система може помилково визначити їх як частину чеку, або не виділити

взагалі, що в подальшому призведе до помилкової корекції зображення або відсутності зображення.

Якщо порівнювати за швидкістю, то незначною мірою OBB моделі переважають над сегментаційними. Але час неістотний, до того OBB моделі можуть гірше працювати на світлому фоні, а сегментація точніше виділяє межі, що найголовніше кути чеків, що частіше призводить до кращого результату вирівнювання. Хоча в обох випадках бувають помилкові розпізнавання меж, OBB частіше захоплює зайвий фон з чеком та часом неточно підбирає кут нахилу обмежувального боксу для чеку, а також не вирішує питання інших перспективних спотворень, як наприклад горизонтальне та вертикальне. Натомість сегментація вирішує усі види викривлень, але в залежності від фону або освітлення, також може некоректно виділяти межі обрізаючи необхідну інформацію з нього. Загалом сегментація помиляється в знаходженні меж чеку рідше за OBB.

Збільшення розміру моделі не сильно впливає на подальше покращення точності розпізнавання меж чеків, але на складному фоні з документами або білим кольором, може підвищити впевненість. Проте при збільшенні розміру з nano до small час збільшується в середньому в 2,5 рази для усіх моделей, а з small до medium час необхідний моделям для сегментації зі знаходженням усіх вершин полігонів підвищується вже приблизно в 3 рази. В мобільному застосунку з вбудованою YOLO-моделлю розмір особливо критично впливатиме на швидкість роботи, а ефективність є важливим фактором для вибору застосунку користувачами.

Враховуючи усі фактори серед протестованих моделей, вибір зупинився саме на YOLO 26n seg. Хоча модель займає трохи більше місця і часу ніж тотожна їй OBB, вона все ще достатньо компактна для мобільного застосунку і займає 6,2 MB, допустимо дещо поступитися цими параметрами на користь підвищення точності розпізнавання меж чеку, враховуючи, що втрати у обраної моделі набагато нижчі ніж у інших моделей.

3 РОЗРОБЛЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ANDROID ДЛЯ НОРМАЛІЗАЦІЇ ЧЕКІВ

3.1 Обґрунтування вибору інструментальних засобів для розробки застосунку

Особливу роль в розробці застосунку відіграє правильний вибір інструментальних засобів, що може пришвидшити роботу створення застосунку, підвищити його якість та допомогти реалізувати запланований функціонал. Для реалізації застосунку обрано наступні інструментальні засоби: Roboflow для збору і розмітки набору даних, чеків, YOLO-моделі для детекції меж чеків, моделі натреновано в нотатках Colab та розробку виконано в IDE середовищі Android Studio.

Для якісного набору даних потрібна велика кількість різноманітних зображень, а Roboflow має в собі готові розмічені набори даних, створенні користувачами, що пришвидшить процес пошуку і розмітки частини зображень, не доведеться шукати або в крайньому випадку робити фото чеків самостійно. Програмне забезпечення має вбудовані функції, які легко дозволяють поєднувати набори даних від різних користувачів у себе на сторінці в один. Також присутні додаткові інструменти для автоматичної розмітки даних завдяки моделі SAM3 і відповідному промпту, що значно зберігає час. Зручний та інтуїтивно зрозумілий інтерфейс дозволяє легко навчитися користуватися програмою і вносити зміни в набори даних на будь-якому етапі. Платформа підтримує різноманітні формати експорту анотацій зображень, особливо важливою є підтримка різних версій YOLO-моделей. Після остаточного створення набору даних можна згенерувати скрипт коду, який вставляється в блокнот для швидкого розпакування набору даних.

YOLO-моделі є популярним вибором для виділення об’єктів, так як вони гарантують дуже важливу, особливо для мобільного застосунку критерій ефективності – швидкість розпізнавання. Моделі дозволяють виявляти об’єкти

миттєво, в режимі реального часу. Час залежить від розміру моделі. Обрана модель YOLO 26 seg nano – компактна, займає 6 MB, що є важливим для мобільного застосунку. Крім звичайної детекції, присутні сегментація та OBB, які краще пристосовані до нахилу об’єктів і допоможуть точніше виділити межі чеків, вирішивши наявну проблему з перспективним спотворенням документів.

На відміну від інших моделей YOLO в передбаченнях розглядає зображення в цілому, а не різні частини, окремо одна від одної. Такий підхід сприяє зменшенню кількості помилок моделі в результаті. YOLO є дуже простим та ефективним в використанні й імплементації, в Python існують бібліотеки з якими легко і швидко можна навчитися працювати, також створенні готові шаблони в Colab з запуском й інструкціями для користування. Платформа надає безкоштовний доступ до середовища виконання GPU, значно прискорюючи виконання глибокого навчання моделей, порівняно з навчанням на CPU персонального комп’ютера. Хоча він має обмеження в часі використання, його вистачає для тренування моделей включно до medium розмірів на підготовленому наборі даних.

Для розробки застосунку обрано IDE середовище Android Studio, який використовує поєднання мов Kotlin та Java. Вирішено розробляти саме мобільний застосунок, тому що саме він є основним засобом для фотографування чеків і гарантує швидкий доступ до функцій із застосуванням камери і фотогалереї. Застосування Jetpack Compose в Android Studio спрощує розробку UI-компонентів, пропонує гнучкість і багатофункціональність, підтримує TF Lite, який забезпечує виконання попередньо натренованої YOLO-моделі на мобільному пристрої. Бібліотека OpenCV надала ефективні засоби для обробки зображень та перспективного вирівнювання чеків.

Прототип застосунку було розгорнуто в мережі Hugging Face розроблено на мові програмування Python, використовуючи пакет Gradio, який дозволяє легко і швидко створити вебзастосунок з моделями ШІ або його тестову версію без потреби реалізації складної frontend частини [62].

Отже, обрані інструментальні засоби дозволили реалізувати мобільний застосунок для автоматичної перспективної корекції чеків та їх подальшої підготовки до OCR або LLM розпізнавання. Інструменти добре поєднуються між собою та дозволяють поєднати реалізацію задачі комп’ютерного зору й розробки мобільного застосунку в межах одного проєкту. Усе це забезпечило достатню гнучкість, підвищило продуктивність розробки та надало можливість подальшого розвитку застосунку.

3.2 Функціональна специфікація застосунку

Розробляється застосунок, який буде автоматично перспективно вирівнювати зображення чеків з метою підвищення точності розпізнавання тексту і можливістю вбудовування його в системи OCR або поєднання з LLM моделями.

Основні функції застосунку:

– відображення головного екрана застосунку;

– завантаження фото для нормалізації двома способами: через камеру і через галерею;

– виконання попередньої обробки зображення;

– знаходження меж чеку моделлю yolo 26 seg nano;

– вирахування функціями бібліотеки OpenCV 4 кутів за знайденими межами чеку;

– впорядкування 4 кутових точок;

– перспективне вирівнювання зображення чеку за 4 кутами засобами OpenCV;

– відображення нормалізованого зображення чеку;

– можливість повернення користувача назад до головного екрану застосунку.

Вхідні та вихідні дані:

– вхідними даними застосунку є цифрове зображення чеку в форматах jpg, png або jpeg;

– вихідними даними є перспективно вирівняне зображення чеку для його подальшої обробки системами OCR;

Функціональні вимоги:

– висока швидкість роботи. необхідно, щоб система забезпечувала швидке вирівнювання чеку на фото в режимі реального часу;

– забезпечити в застосунку простий та інтуїтивно зрозумілий інтерфейс користувача;

– автоматичне і якісне виявлення меж чеку і перспективна корекція зображення.

Обробка помилок:

– якщо на фото немає жодного чеку, виводити повідомлення «чек на фото відсутній»;

– при випадках, коли на фото присутнє більше одного чеку, обирати той, чия впевненість більша, якщо впевненість однакова обирати будь-який з наявних;

– в ситуаціях, коли після сегментації на чеку знайдено менше 4 точок для перспективного вирівнювання, знайти додаткову точку відносно наявних і скорегувати фігуру;

– якщо після сегментації на чеку знайдено більше 4 точок для перспективного вирівнювання знайти чотирикутник, який максимально покриватиме площину полігону.

Не функціональні вимоги:

– час обробки зображення програмою повинен тривати не більше 3 секунд;

– можливість використання застосунку без необхідності доступу до інтернету;

– підтримка роботи на android версії 7 та більше;

– стійкість до помилок;

– якість нормалізованого зображення повинна зберігати якість початкового фото і читабельність тексту на ньому.

3.3 Проєктування та реалізація архітектури застосунку

Перед початком розробки застосунку важливим є створення діаграми потоків для кращого розуміння порядку і взаємозв’язків процесів у застосунку. Було створено дві діаграми у нотації IDEF3. Діаграма відображає сценарій роботи застосунку від моменту отримання зображення чека до формування перспективно скорегованого зображення. Перша, контекстна діаграма на рисунку 3.1 відображає вхідні дані, зображення чеку, та вихідні дані, вирівняне зображення чеку, і демонструє ключовий процес системи. Друга діаграма на рисунку 3.2 є декомпозицією першого рівня системи «Вирівнювання чеків для покращення точності розпізнавання тексту», включає в себе виконання 5 дій і відображає взаємодію процесів обробки інформації в системі та об’єктів, які є важливою частиною визначених процесів. Кожен процес системи виконується відповідно до нумерації зазначеній на діаграмах.

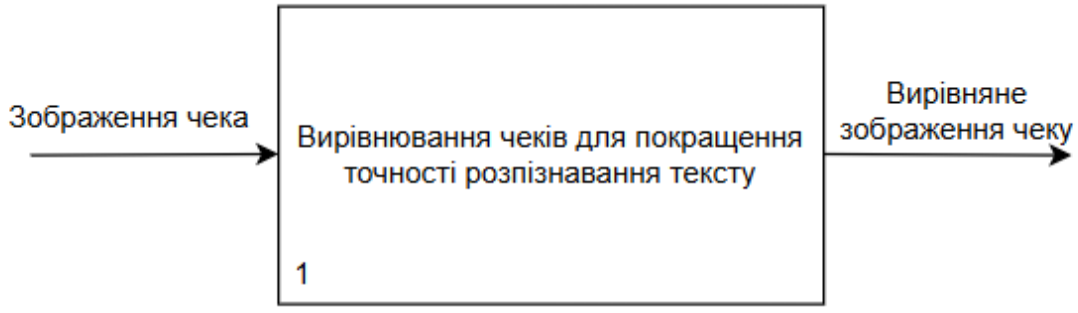


Рисунок 3.1 – Контекстна діаграма системи «Вирівнювання чеків для покращення точності розпізнавання тексту» у нотації IDEF3

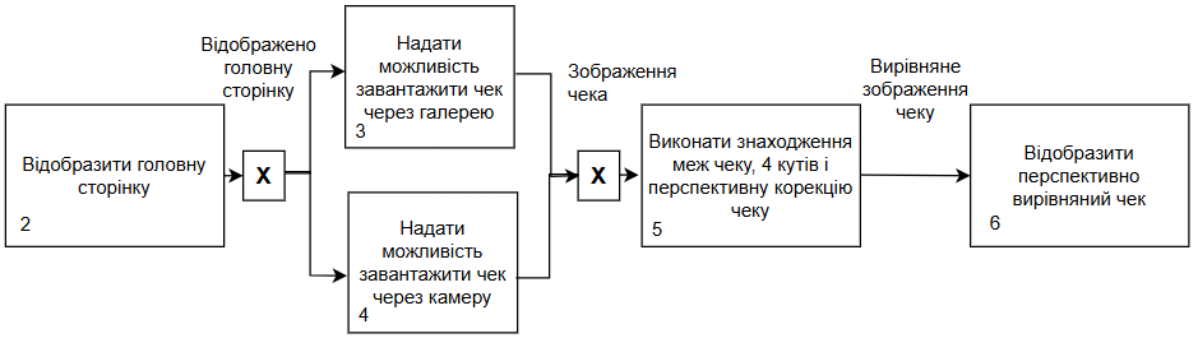


Рисунок 3.2 – Діаграма декомпозиції першого рівня системи «Вирівнювання чеків для покращення точності розпізнавання тексту»

Для вбудування в застосунок YOLO-моделей і роботи над ними створимо клас YoloModel в якому реалізуємо основну логіку роботи системи з необхідними методами. В класі забезпечимо завантаження моделі TenserFlow Lite, побудову маски сегментації, пошук контурів та перспективне вирівнювання зображення. Детальніше розглянемо методи представлені в діаграмі класу на рисунку 3.3 і їхню логіку.

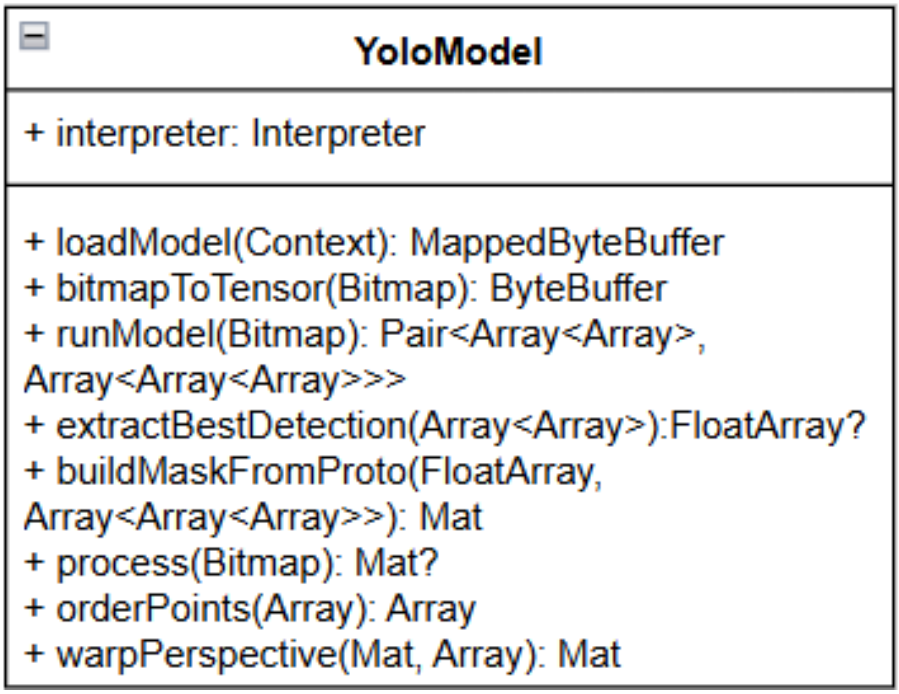


Рисунок 3.3 – Діаграма класу YoloModel

Призначення метода loadModel() в завантаженні моделі формату TensorFlow Lite з asset-директорії застосунку. На вхід метод отримує контекст Android-застосунку, а на вихід видає буфер з завантаженою моделлю. Метод відкриває файл моделі, завантаженої в середовище у форматі .tflite, створює файловий канал та виводить модель у пам’ять для наступного використання TenserFlow Interpreter.

Метод bitmapToTensor() трансформує зображення з типу Bitmap у формат Tensor, що є необхідним для подачі фото в модель машинного навчання. Вхідним параметром є зображення чека, а вихідним – Tensor моделі TensorFlow Lite. В межах методу на початку виконується масштабування зображення до розміру 640×640 пікселів, зчитування значень пікселів, нормалізація каналів RGB у межах [0;1] та записування значень у ByteBuffer.

Метод runModel() відповідає за виконання interference TensorFlow Lite моделі. Так само як в попередньому методі вхідним параметром є зображення чека, а вихідними параметрами виступають пара масивів: результати детекції та маски сегментації. Метод формує вхідний тензор, реалізовує структури для виводу тензора, розгортає TenserFlow Interpreter та повертає результат виконання роботи моделі.

Призначення методу extractBestDetection() полягає в виборі найбільш точної детекції чека. На вхід методу надходить масив результатів детекції, а повертається – найкраща знайдена детекція або null. Алгоритм проходить по всім детекціям, аналізуючи оцінку впевненості і обираючи максимальне значення з наявних.

Метод buildMaskFromProto() створює сегментаційну маску чека. Дані отримані на вході – це коефіцієнти маски та proto-маски сегментації, а на вихід вертається матриця маски сегментації. В тілі методу виконується лінійна комбінація proto-масок, використовується сігмоїдна функція та утворюється фінальна маска області меж чека.

Метод orderPoints() впорядковує кутові точки чека перед перспективним перетворенням для коректного формування матриці перетворень. На вхід

отримує масив кутових точок, а в результаті повертає впорядкований масив точок: верхня ліва, верхня права, нижня права та нижня ліва.

В методі warpPerspective() виконується перспективне перетворення зображення чека за оригінальним зображенням та отриманим масивом кутових точок, в результаті повертається перспективно вирівняне зображення. Метод виконує обчислення ширини та висоти нового зображення, формує матрицю перспективного перетворення, реалізує метод warpPerspective з OpenCV та масштабує фінальний результат.

Основним методом конвеєрної обробки зображення є process(). Вхідним параметром є зображення чеку, а вихідним – перспективно скореговане зображення чеку або null. В цьому методі поєднується робота попередньо описаних методів: масштабування зображення, запуск YOLO-моделі, побудова маски, бінарізація, пошук контурів, апроксимація контуру, знаходження чотирьох кутів та перспективне вирівнювання зображення – він є центральним елементом обробки чеків.

Клас MainActivity відповідає за реалізацію основної логіки застосунку для завантаження, обробки та відображення результату з перспективно скорегованими чеками. Розглянемо кожен метод і його роботу в межах застосунку.

Метод в якому реалізована навігація – ReceiptApp. В ньому виконується перехід між сторінками і зберігається поточний стан, де main означає головну сторінку з кнопками для завантаження зображення, після обрання фото застосунок перенаправляє на другу сторінку – result, яка відображає оброблене вирівняне зображення і кнопку «Назад», яка повертає користувача до початкового екрану. Також в цьому методі зберігається посилання на обране зображення для передавання його в обробку.

В методі MainPage реалізована робота головної сторінки мобільного застосунку, яка відображається після запуску з заголовком про завантаження фото чека та двома кнопками: обрання фото з галереї або створення фото камерою.

Друга сторінка з обробкою і відображенням результату виконано в методі ReceiptCorrectedPage. Спочатку метод ініціалізує OpenCV, завантажує Bitmap і за використанням класу YoloModel запускає модель у фоновому режиму. На сторінці відображається поточний стан згідно з виконанням роботи: обробка, готове вирівняне відображення чека або його відсутність на зображенні.

Для кожної з кнопок головної сторінки реалізовано метод. CreateSelectFromAGalleryBtn виконує вибір зображення з галереї, а CreateTakeAPhotoBtn відповідає за фотографування чека, запит дозволів і запуск камери. Метод CreateBtn універсальний метод створення кнопки з текстом і іконкою, який повторно використовується в різних частинах застосунку.

В результаті створений застосунок займає 259 МБ з яких 215 розмір програми, де 5,52 МБ становить натренована модель, а 44 МБ – дані користувача (рис. 3.4).

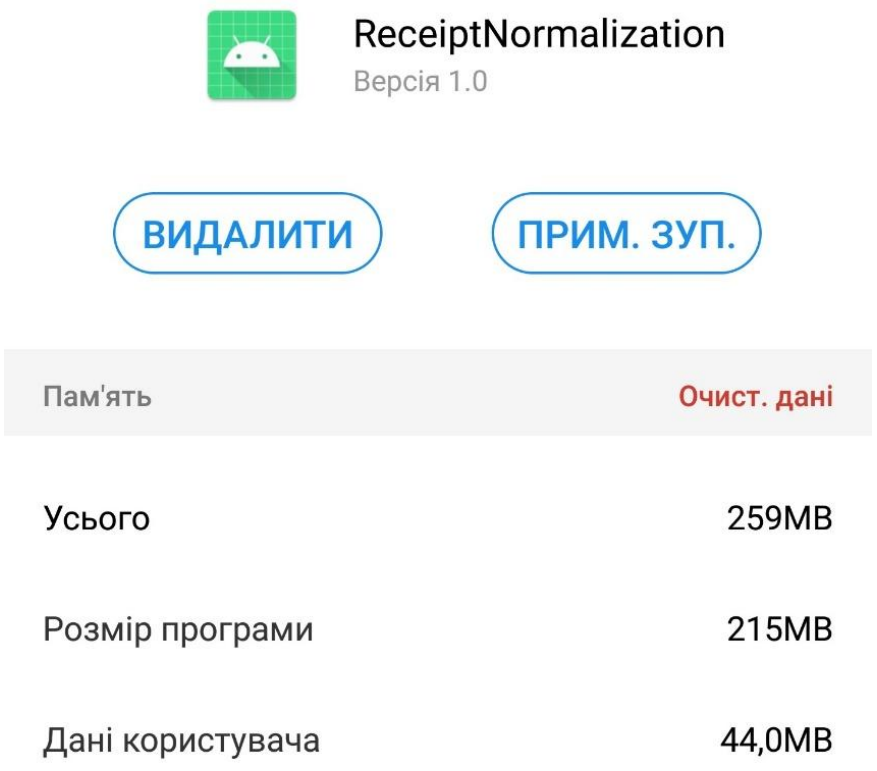


Рисунок 3.4 – Кількість пам’яті, що займає застосунок

Для роботи додаткових, сторонніх програм в мобільному застосунку Android до відповідного модулю було додано різні бібліотеки: для TensorFlow, OpenCV, доступу до камери та завантаження фото. Підключені бібліотеки розписано в лістингу 3.1.

Лістинг 3.1 Додані бібліотеки:

```
    dependencies {
  implementation("org.tensorflow:tensorflow-lite:2.17.0")
  implementation("org.tensorflow:tensorflow-lite-support:0.5.0")
  implementation(project(":opencv"))
  implementation("androidx.camera:camera-core:1.6.0")
  implementation("io.coil-kt:coil-compose:2.7.0")
}
```

Було побудовано діаграму послідовностей для кращого розуміння черговості процесів в програмі, взаємодію компонентів мобільного застосунку під час обробки фотографій чека (рис. 3.5). На діаграмі відображено послідовність викликів між користувачем, інтерфейсом застосунку, камерою або галереєю, YOLO-моделлю та бібліотекою OpenCV.

В діаграмі діяльності на рисунку 3.6 відображено послідовність роботи мобільного застосунку для перспективної корекції чеків. Вона наглядно описує логіку і послідовність виконання етапів обробки зображення. Після того як користувач відкриває застосунок і опиняється на головній сторінці застосунку на діаграмі розглядається два можливі сценарії завантаження зображення чеку: через камеру і через галерею. Обидва сценарії приводять до отримання і подальшої обробки зображення.

При здійсненні усіх етапі обробки зображення на екрані відображається перспективно скорегований результат. На останньому етапі стрілка показує можливість повернення до головної сторінки після виконання перспективної корекції.

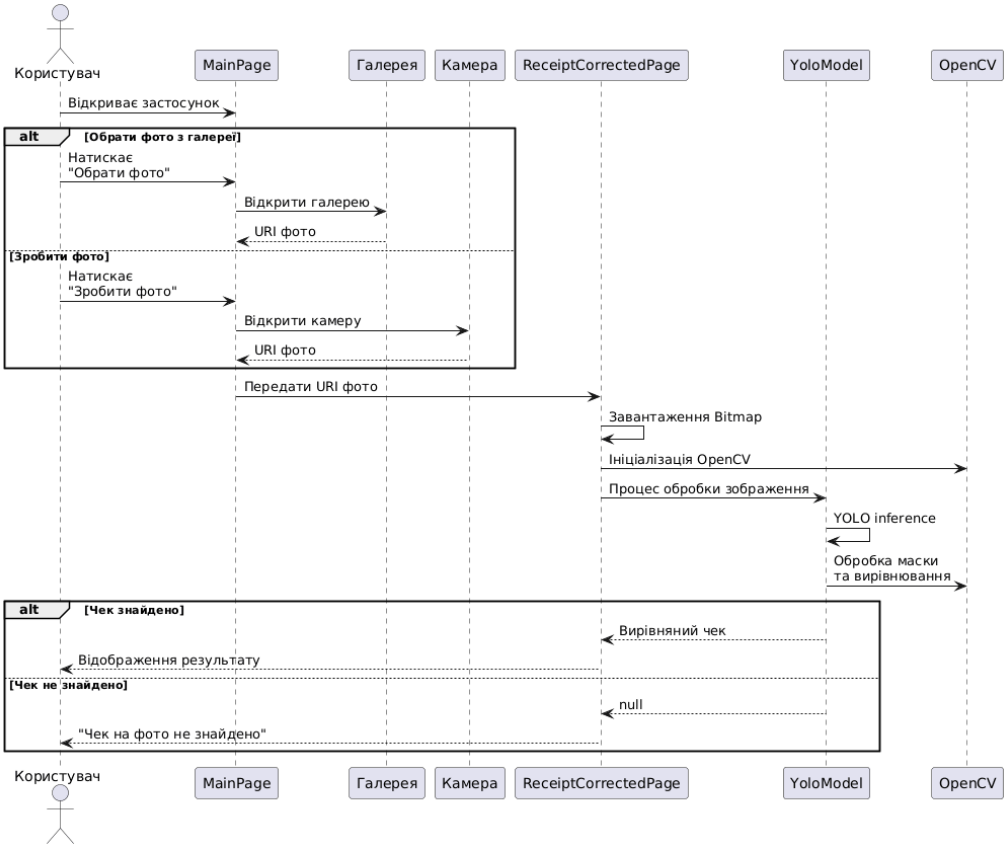


Рисунок 3.5 – Діаграма послідовностей застосунку

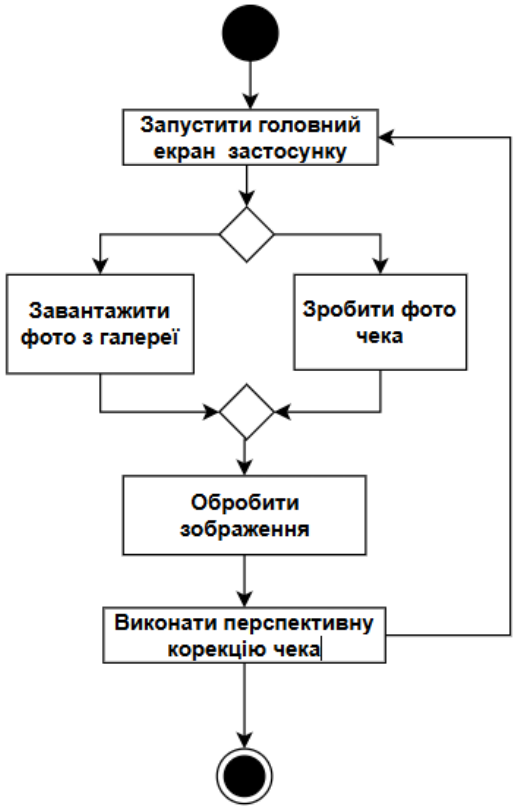


Рисунок 3.6 – Діаграма діяльності програми

Першим було створено прототип в форматі веб-застосунку на платформі Hugging Face в якому наглядно продемонстровано етапи роботи фінального застосунку. На сторінці застосунку на рисунку 3.7 присутнє одне вікно для вводу даних, де можна завантажити зображення і три вікна виведення для візуалізації кожного етапу нормалізації фото. Також знизу присутнє спливаюче вікно з вибором одної з чотирьох моделей: одна OBB і три сегментації (рис. 3.8).

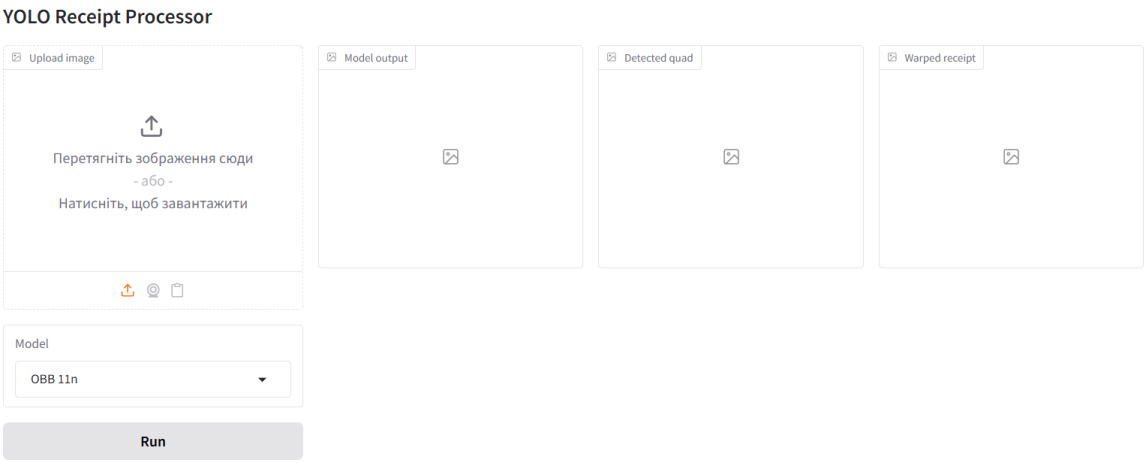


Рисунок 3.7 – Сторінка вебзастосунку на початку роботи

Рисунок 3.8 – Варіанти вибору з випадаючого списку

Після натискання на кнопку «Run» запускається робота програми і виводиться результати. Час безпосередньо залежить від обраної моделі і зазвичай швидкість виконання складає приблизно 5-10 секунд.

Перше вікно відображає роботу моделі з виділеними межами чеку, використовуючи сегментацію або OBB з обмежувальним боксом і відсоток впевненості для класу receipt. Друге вікно виводить 4 кутові точки: для OBB – це відповідно кути прямокутного обмежувального боксу, а для сегментації кути вираховуються з маски. В останньому вікні виводиться перспективно скореговане зображення чеку. На рисунку 3.9 продемонстровано роботу вебзастосунку з моделлю сегментацією, SEG 26 nano, а на рисунку 3.10 – з OBB 11 nano.

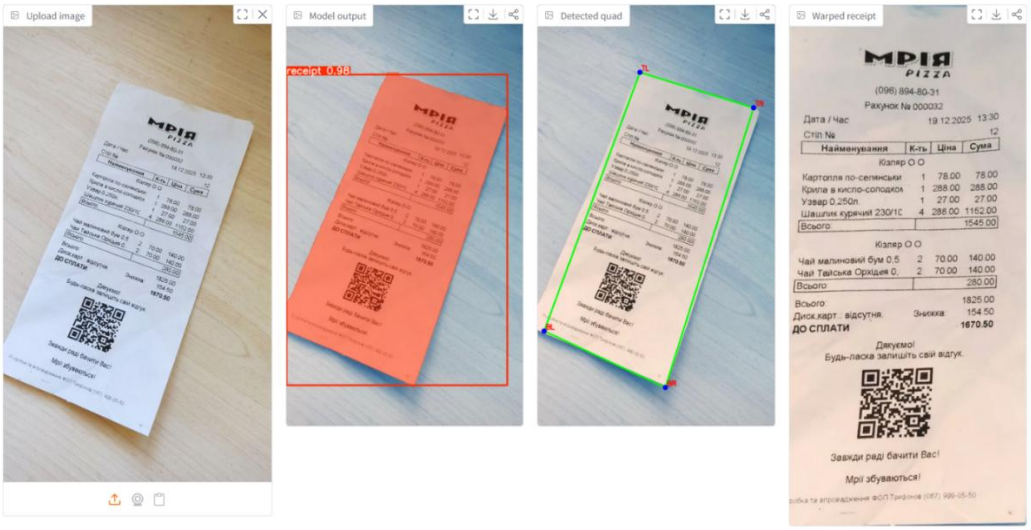


Рисунок 3.9 – Результат роботи вебзастосунку для моделі сегментації

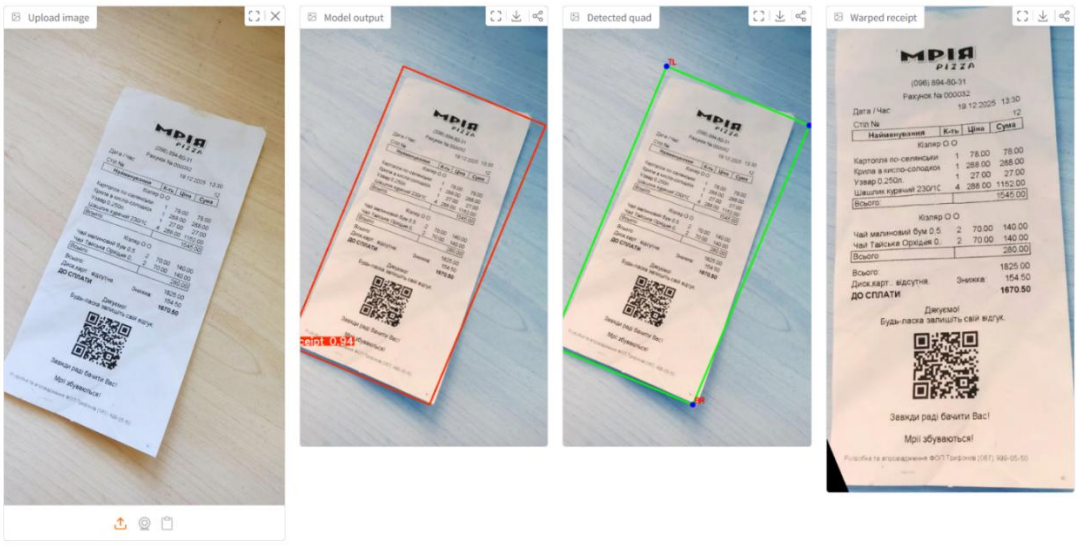


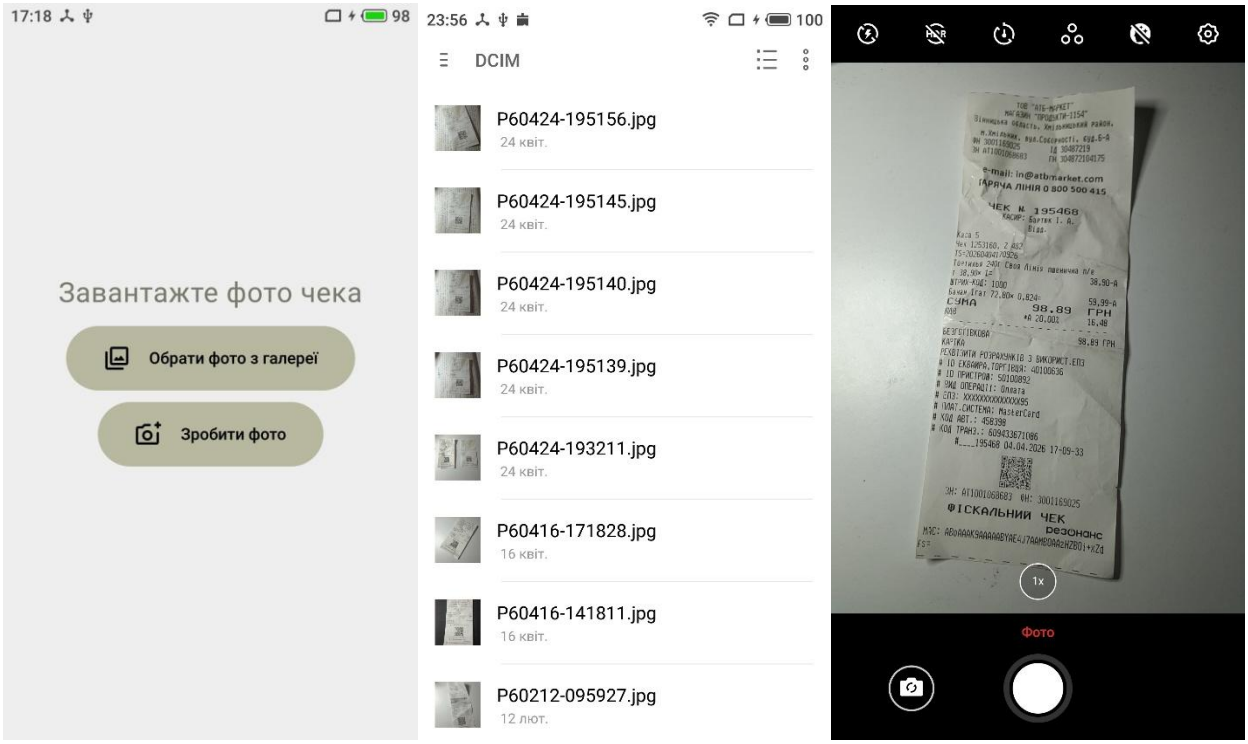
Рисунок 3.10 – Результат роботи вебзастосунку для моделі OBB

3.4 Ілюстрація роботи застосунку

Було розроблено застосунок з базовими необхідними функціями. При його запуску на екрані відображається текст з інструкцією «Завантажте фото чека» і дві кнопки, що виконують однакову функцію різними методами, завантаження фото, що відображено на рисунку 3.11 а).

Після натискання першої кнопки «Обрати фото з галереї» система перенаправляє до галереї телефону, приклад представлено на рисунку 3.11 б) з якої обирається необхідне фото.

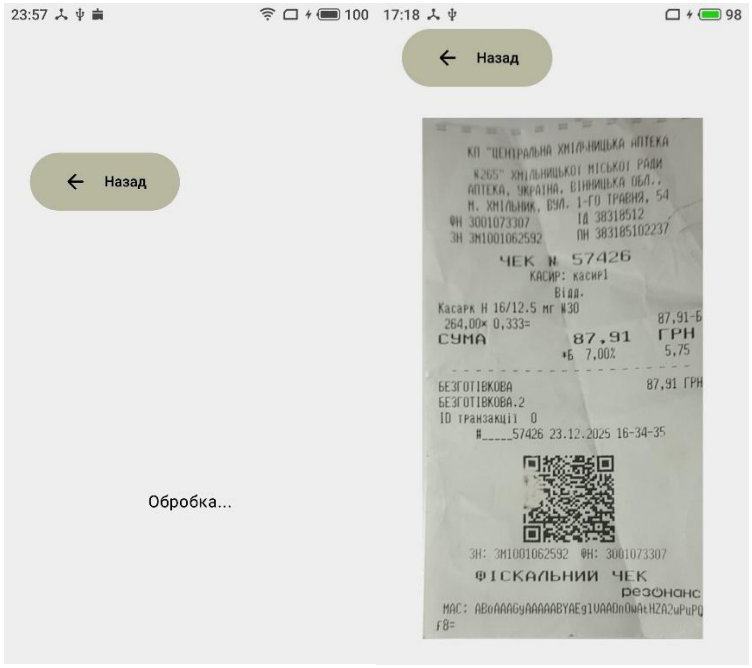
При натисканні на другу кнопку «Зробити фото» система перенаправляє користувача до камери, приклад на рисунку 3.11 в), де можна зробити нове фото через камеру.



а) б) в)

Рисунок 3.11 – Відображення роботи головної сторінки застосунку: а) головна сторінка з головним функціоналом застосунку; б) перехід до галереї через кнопку «Обрати фото з галереї»; в) перехід до камери через кнопку «Зробити фото»

Після обрання фото чеку програма виконує обробку фото, що включає знаходження меж чеку, 4 кутових точок, їх впорядкування та перспективне вирівнювання зображення чека. Поки відбувається обробка, яка зазвичай триває пару секунд, для користувача на екрані відображається відповідне повідомлення «Обробка…» (рис. 3.12 а)). Після закінчення виконання роботи системою на новій сторінці відображається результат з перспективно скорегованим чеком (рис. 3.12 б)). На цій сторінці крім зображення чеку також присутня кнопка «Назад», яка дозволяє повернутися на головну сторінку застосунку.



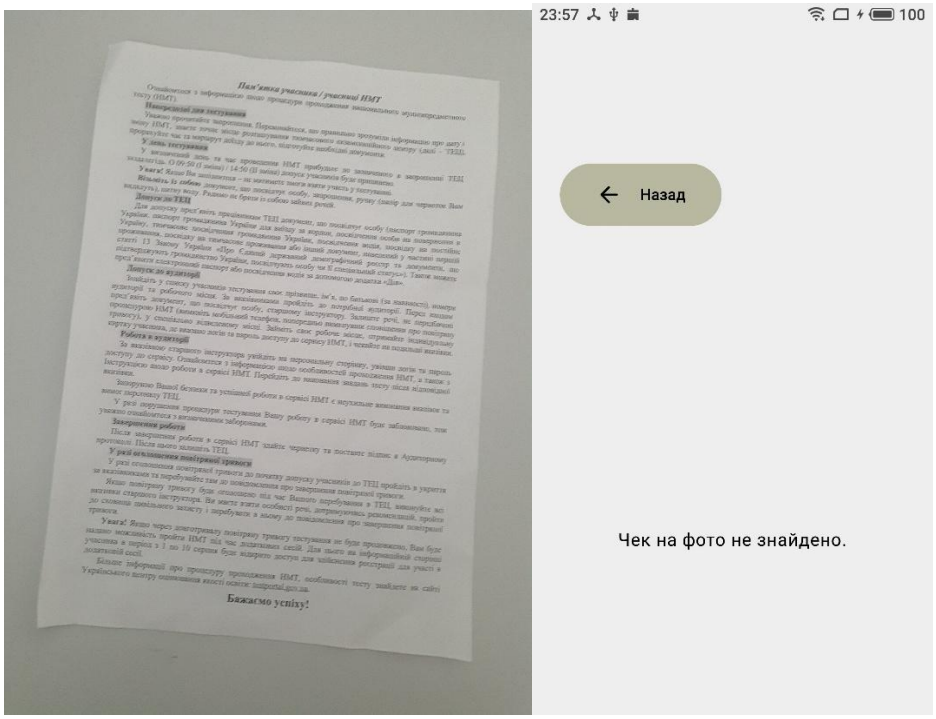
а) б)

Рисунок 3.12 – Відображення роботи другої сторінки застосунку:
а) повідомлення про обробку зображення; б) відображення результату роботи у вигляді перспективно вирівняного чека

3.5 Аналіз та тестування роботи застосунку

В застосунку враховано обробку випадків, коли на фото відсутні чеки, або впевненість системи в тому, що на фото є чеки нижча 40%. Такий поріг

впевненості встановлено, щоб уникнути розпізнавання фону, тому що не завжди на фото присутній чек і може знаходитися інший схожий документ (3.13 а)). В таких ситуаціях система виводить текстове повідомлення «Чек на фото не знайдено», що відображено на рисунку 3.13 б).



а) б)

Рисунок 3.13 – Робота програми при відсутності чеку на фото:
а) вихідний документ, який не є чеком; б) відображення повідомлення про відсутність знайденого чека на фото

В деяких випадках на фото може знаходитися не один чек. У ситуаціях коли на фото присутні два або більше чеків, програма автоматично обирає для подальшої обробки лише один з них. Для розв’язання цієї проблеми було вирішено обирати той з них, що має більшу впевненість і з більшою ймовірністю з запропонованих документів є чеком. На рисунку 3.14 а), б) показано приклад такої ситуації: на оригінальному зображенні присутні два чека, проте після обробки відображається лише другий, оскільки програма порахувала вирахувані YOLO-моделлю впевненості і визначила, що у другого чека вона більше. У випадках коли на зображені у декількох чеків однакова

впевненість для нормалізації обирається рандомний з них. Хоча при наявності більше двох чеків на фото після нормалізації розмір знижується і якість зображення чека значно знижується, що впливатиме на подальше розпізнавання тексту системами OCR.

а) б)

Рисунок 3.14 – Робота програми при наявності двох чеків на фото: а) вихідне фото з двома чеками; б) результат роботи програми з одним вирівняним чеком

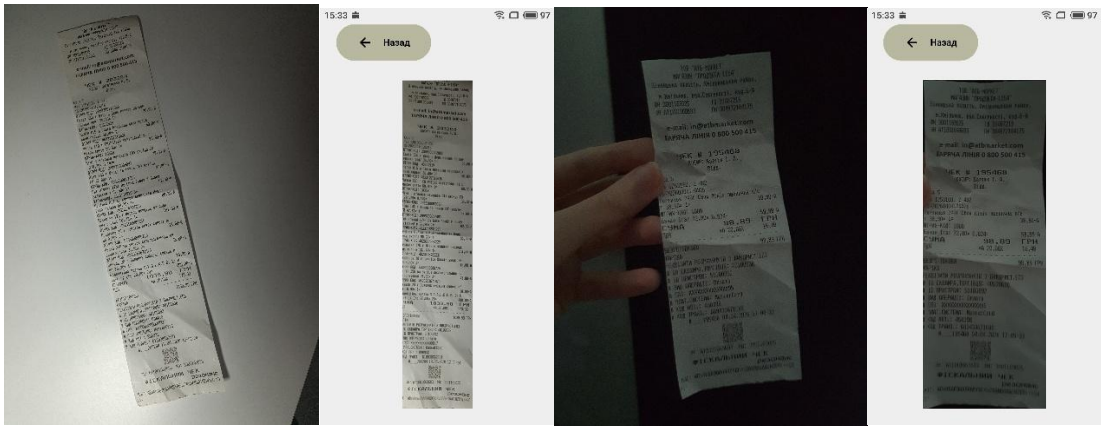
На рисунку 3.15 а), б) показано виправлення інших перспективних спотворень, крім простого нахилу, що програма також успішно виправляє. Перспективне вирівнювання також залежить від стану чеку, якщо на ньому присутні фізичні згини або зім’ятості це вплине на фінальний результат і не може бути повністю усунуто програмою після обробки.

В програмі також протестовано обробки чеків в інших різних можливих умовах зйомки та сценаріях використання застосунку. Наприклад, на рисунку 3.16 а), б) перевірено роботу на довгих чеках і їх відображення в результаті показало, що алгоритм коректно відображає їх після нормалізації. На рисунку 3.16 в), г) перевірено розпізнавання чеків на фотографіях з поганим

освітленням. Незважаючи на освітлення межі чеків успішно виявляються і вирівнюються, що свідчить про стійкість до змін умов освітлення.

а) б)

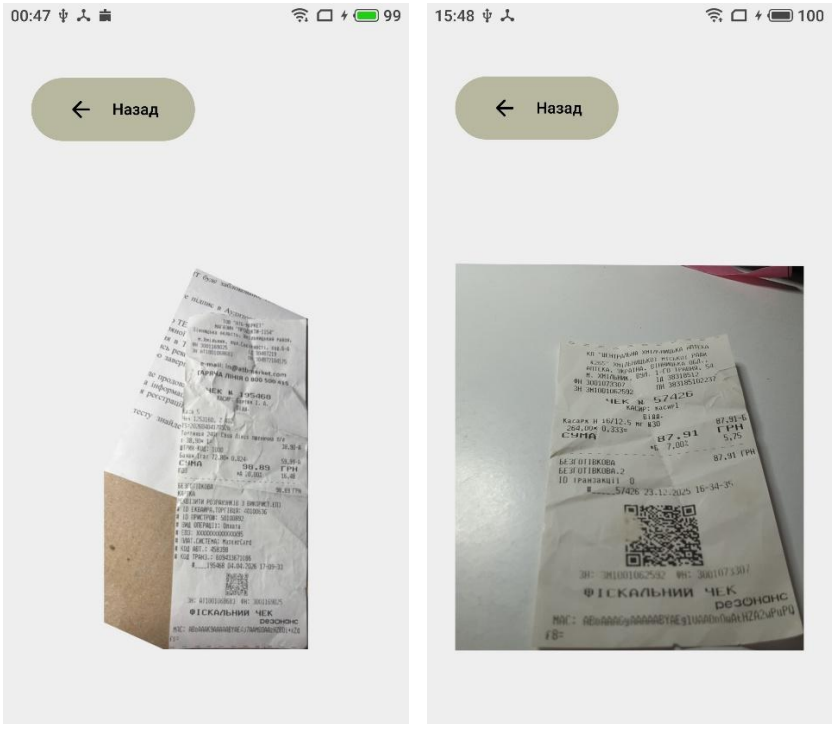
Рисунок 3.15 – Робота програми при інших перспективних спотвореннях: а) чек нахилений по вертикальній осі; б) результат зі перспективно скорегованим чек



а) б) в) г)

Рисунок 3.16 – Виявлення чеків в різних ситуаціях: а) довгий чек; б) вигляд обробки довгого чеку в програмі; в) чек з поганим освітленням; г) вигляд обробки чеку з поганим освітленням

Система стикається з помилками, коли на фоні присутні інші документи документи. YOLO-модель сприймає і виділяє сегментацією різні документи, як частину одного чека, що впливає на подальшу роботу вирівнювання і через що в результаті відображається вирівняне зображення чеку за некоректно знайденими кутами, як видно на рисунку 3.17 а). Фон значно впливає на розпізнавання і вирівнювання зображення чека. Найкраще застосунок працює на темному, контрастному до чеку фоні, але добре справляється з чисто білим фоном. При наявності документів на фоні не завжди може розпізнати межі чека, через що виводить повідомлення «Чек на фото не знайдено», навіть якщо він присутній. Також часом присутні випадки некоректного вирівнювання в інших ситуаціях, наприклад на рисунку 3.17 б), хоча на фото немає інших документів, такий результат може бути пов’язаний з сильним перспективним спотворенням чека або іншим недоліком.



а) б)

Рисунок 3.17 – Складність виділення меж чеку застосунком через документи на фоні:

а) помилкове вирівнювання чеку на фоні іншого документу; б) помилкове вирівнювання сильно перспективно спотвореного чеку.

Одним з найважливіших критеріїв оцінки застосунку є швидкість роботи, тому під час тестування значну увагу було приділено продуктивності та аналізу швидкодії системи. Було протестовано швидкість перспективного вирівнювання застосунку на різних чеках, застосовуючи вбудовані функції Android Studio. Створено дві змінні одна з яких зберігає час початку обробки зображення і вставляється перед створенням моделі, а друга – кінець і вставляється після відображення скорегованого фото на сторінці, віднявши від останньої першу отримаємо час необхідний системі для обробки зображення. Виявлено, що час за який модель обробляє фото чеку виконуючи усі етапи складає в районі 1000 мілісекунд, що становить 1 секунду. Для повної роботи в проміжку після вибору фото і до відображення результату застосунок може витрачати до 2 секунд, що створює незначну затримку для користувача, проте є прийнятним результатом для мобільного застосунку.

Було перевірено можливість зниження часу шляхом зменшення розмір вхідного зображення чеку перед початком його обробки системою. В результаті виявлено, що час, який витрачається для роботи не сильно залежить від початкового розміру зображення і зменшується відповідно до його зменшення. Перед виявленням меж YOLO-моделлю зображення вже зменшується до сталого 640 на 640, тому на цей етап обробки розмір зображення не впливає. При зменшенні фото в 2 рази перед усіма етапами роботи програми швидкість зменшилася приблизно на 0,4 секунди з 1,9 на оригіналі до 1,5 секунд на зменшеному. Таке зменшення є не значним і майже не відчувається при користуванні.

3.6 Подальший розвиток застосунку

На даному етапі розробки реалізовано виключно базові функції застосунку, як завантаження зображення і автоматичне вирівнювання чеку.

Для подальшого розвитку пропонується розширення його функціональних можливостей та вдосконалення наявних алгоритмів обробки зображень.

Одними перспективних варіантів майбутнього розширення застосунку є розширення його функціональності для реалізації повноцінної системи для відслідковування витрат, або універсального сканера для різноманітних документів з можливістю їх подальшого оцифрування системами OCR або LLM [63]. Також систему можна інтегрувати в уже існуючи програми обліку витрат або системи розпізнавання тексту.

Важливим є підвищити стійкості системи до розпізнавання моделями чеків на складних фонах з іншими документами на задньому плані. Необхідним є розгляд можливостей підвищення швидкості обробки зображення до миттєвого розпізнавання і вирівнювання.

Також перспективним напрямком є розширення можливостей і додавання в систему моделей натренованих для розпізнавання та нормалізації інших типів документів. Завдяки цьому застосунок можна буде використовувати не лише з чеками, а і як універсальний інструмент для підготовки документів до подальшої обробки в інших системах. Важливими покращенням є реалізація функції попереднього перегляду результату і ручного корегування кутів чеків користувачами при неточностях роботи системи, що дозволить користувачу оцінити якість вирівнювання і виправити. До того ж можна вбудувати роботу виявлення меж чеку в режимі реального часу через камеру.

Ще одним можливим розширенням може бути одночасна обробка кількох чеків на зображенні і занесення їх одразу в профіль користувача зі збереженими чеками. Для ще кращого виявлення тексту на чеках було б корисним додавання додаткової обробки зображень з переведенням в чорно-біле, підвищенням контрастності та обробкою шумів на чеку.

Для реалізації можливостей довготривалого збереження результатів вирівнювання та оцифрованих даних з чеків доцільним є впровадження систем користувацьких акаунтів і підключення бази даних до застосунку.

Таким чином, впровадивши запропоновані удосконалення і додатковий функціонал підвищиться ефективність системи і вдасться створити повноцінний, якісний мобільний застосунок для автоматизованої обробки чеків.

ВИСНОВКИ

У рамках кваліфікаційної роботи було розроблено і розгорнуто стенд застосунку на Hugging Face для перспективного вирівнювання зображень чеків з відображенням усіх етапів обробки і можливістю обрати різні YOLO-моделі, мобільний застосунок та вирішено наступні завдання:

– проаналізовано літературні джерела і порівняно між собою сучасні наявні застосунки для перспективного вирівнювання документів, особливу увагу приділено якості розпізнавання ними меж чеків і їх вирівнюванню, що сприяло кращому розумінню поточного стану проблеми;

– сформовано вимоги до вхідних даних і критерії вибору моделі, що забезпечило коректний вибір моделі з урахуванням швидкості і точності;

– сформовано власний набір даних з якісних і різноманітних зображень чеків в застосунку Roboflow і натреновано ним YOLO-моделі використовуючи Colab, що дозволило моделям якісно розпізнавати межі чеків;

– порівняно різні техніки і YOLO-моделі, що дозволило оцінити їхню ефективність і сприяло вибору найбільш придатної моделі для поставленої задачі;

– застосовано OpenCV для вирівнювання чеків, що сприяло покращенню якості зображень перед розпізнаванням тексту;

– створено Android-застосунок, який інтегрує блоки розпізнавання та вирівнювання чеків, що забезпечило зручність використання системи;

– протестовано та оцінено роботу застосунку на різних зображеннях чеків з різними фонами, що дозволило підтвердити працездатність застосунку та оцінити точність роботи.

В роботі порівняно різні техніки розпізнавання, найкраще вирішили задачу OBB та сегментація. Проаналізовано конкретні YOLO-моделі і з них обрано YOLO26 nano за гарною якістю виділення меж і швидкістю розпізнавання чеків.

Для нормалізації чеку застосовано бібліотеку OpenCV. Перспективно вирівняне зображення покращило розпізнавання тексту LLM в середньому від 1% до 5% і суттєво покращило розпізнавання тексту OCR моделями.

Мобільний застосунок реалізовано в середовищі Android Studio в якому розроблено базовий функціонал вирівнювання чеків з можливістю завантажити зображення двома способами: через камеру і через галерею. Для написання логіки застосовано поєднанням мов програмування Kotlin і Java, середовище дозволило реалізувати якісний, інтуїтивно зрозумілий застосунок.

Подальший розвиток застосунку може включати в себе додавання особистого кабінету користувача з можливістю зберігати в ньому чеки, сучасних систем розпізнавання тексту та поліпшення роботи наявних.

Результати можуть бути корисні для автоматизації і пришвидшення вирівнювання чеків в банківських системах, бухгалтерії, системах ведення витрат. Результати роботи апробовано у вигляді тези доповіді під час Міжнародного молодіжного форуму «РАДІОЕЛЕКТРОНІКА І МОЛОДЬ У XXI СТОЛІТТІ» та здобуття диплому III ступеня за участь у форумі [64].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Qidenus Mastered Book Scanner A2 — швидке й дбайливе оцифрування книг і документів. URL: https://panorama.uran.ua/tsyfruvannia-knyg-i-dokumentiv/ (дата звернення 17.05.2026).

2. ILOVEPDF. (2023). What is OCR and why your business needs it. URL: https://www.ilovepdf.com/uk/blog/what-is-ocr (дата звернення 17.05.2026).

3. FPT.AI. (2025). What is Optical Character Recognition? The importance of OCR. URL: https://fpt.ai/blogs/optical-character-recognition/ (дата звернення 10.06.26).

4. Optical Character Recognition (OCR) – What is it & how does it work? URL: https://www.hyperscience.ai/resource/optical-character-recognition-ocr/ (дата звернення 17.05.2026).

5. Gorokhovatskyi, O., & Yakovleva, O. (2024). Medoids as a packing of orb image descriptors. Advanced Information Systems, 8(2), 5-11.

6. Bohdan N., Tvoroshenko I., Gorokhovatskyi V., and Kobylin O. (2025) Development of a hybrid method to enhance context memory for a chatbot application based on large language models, *International Journal of Academic Information Systems Research*, 9(10), pp. 7-18.

7. Gorokhovatskyi V., Tvoroshenko I., and Yakovleva O. (2024) Transforming image descriptions as a set of descriptors to construct classification features, *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 33, no. 1, pp. 113-125.

8. V7. (2024). Visual ChatGPT: Multimodal Capabilities and Use Cases for 2024. URL: https://www.v7labs.com/blog/chatgpt-with-vision-guide (дата звернення 17.05.2026).

9. Фурсов А. Д. Розробка та реалізація методу розпізнавання документів для автоматичної реалізації із застосуванням моделей штучного інтелекту : кваліфікаційна робота першого (бакалаврського) рівня вищої освіти: Комп’ютерні науки. Харків, 2025. 67 с.

10. Харченко, В. В., & Любченко, В. А. (2025). Порівняльний аналіз продуктивності сучасних моделей детекції та розпізнавання тексту на зображеннях. In Innovations of modern science and education. Proceedings of the 2nd International Scientific and Practical Conference pp. 376–383.

11. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2026) Feature space quantization and application of metrics in structural methods of image recognition, *IEEE Access*, vol. 14, pp. 17812-17824.

12. Gorokhovatskyi V., Tvoroshenko I., Kobylin O., and Vlasenko N. (2023) Search for visual objects by request in the form of a cluster representation for the structural image description, *Advances in Electrical and Electronic Engineering*, 21(1), pp. 19-27.

13. Epshtein, B., Ofek, E., & Wexler, Y. (2010, June). Detecting text in natural scenes with stroke width transform. In *2010 IEEE computer society conference on computer vision and pattern recognition* (pp. 2963-2970). IEEE.

14. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., & Vlasenko, N. (2023). Explanation of CNN image classifiers with hiding parts. In Explainable Deep Learning Artificial Intelligence pp. 125–146. Academic Press.

15. Jiang, M., Cheng, J., Chen, M., & Ku, X. (2018, January). An improved text localization method for natural scene images. In *Journal of physics: conference series* (Vol. 960, No. 1, p. 012027). IOP Publishing.

16. Gorokhovatskyi, O., Gorokhovatskyi, V., & Peredrii, O. (2018). Analysis of application of cluster descriptions in space of characteristic image features. Data, 3(4), pp. 52.

17. Segment and Read Text in Image. URL: https://www.mathworks.com/help/vision/ug/automatically-detect-and-recognize-text-in-natural-images.html (дата звернення 17.05.2026).

18. Стрельцов О. А. (2024). Застосування глибокого навчання до виявлення об’єктів. Радіоелектроніка та молодь у XXI столітті : матеріали 28-го Міжнародного молодіжного форуму, (Харків,16-18 квітня 2024 р.). Харків : ХНУРЕ, 2024. Т. 7. С. 112–113.

19. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., Hudáková M., and Gorokhovatskyi O. (2024) Application a committee of Kohonen neural networks to training of image classifier based on description of descriptors set, *IEEE Access*, vol. 12, pp. 73376-73385.

20. Tvoroshenko I., Gorokhovatskyi V., Kobylin O., and Tvoroshenko A. (2023) Application of deep learning methods for recognizing and classifying culinary dishes in images, *International Journal of Academic and Applied Research*, 7(9), pp. 57-70.

21. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Development of an application for recognizing emotions using convolutional neural networks, *International Journal of Academic Information Systems Research*, 7(7), pp. 25-36.

22. Pomazan V., Tvoroshenko I., and Gorokhovatskyi V. (2023) Handwritten character recognition models based on convolutional neural networks, *International Journal of Academic Engineering Research*, 7(9), pp. 64-72.

23. Gorokhovatskyi V., Tvoroshenko I. (2024) An effective method for transforming an image description into a compact vector for classification. *Information Technology and Implementation (Satellite): Conference Proceedings, November 21, 2024, Kyiv, Ukraine / Ministry of Education and Science of Ukraine, Taras Shevchenko National University of Kyiv and [etc]; Vitaliy Snytyuk (Editor). – Kyiv: Publishing House «Caravela»*, pp. 25-28.

24. Ковтуненко, А., Яковлева, О., & Любченко, В. (2020). Дослідження сумісного використання математичної морфології та згорткових нейронних мереж для розпізнавання цінників. *Вісник НТУ «ХПІ»*, 1(3), pp. 24–31.

25. Lyubchenko, V., Veretelnyk, K., Kots, P., & Lyashenko, V. (2024). Digital image segmentation procedure as an example of an NP problem. *Multidisciplinary Journal of Science and Technology*, 4(4), 170-177.

26. DBNet and DBNet++. URL: https://github.com/mindspore-lab/mindocr/blob/main/configs/det/dbnet/README.md (дата звернення 17.05.2026).

27. Tian, Z., Huang, W., He, T., He, P., & Qiao, Y. (2016, September). Detecting text in natural image with connectionist text proposal network. In *European conference on computer vision* (pp. 56-72). Cham: Springer International Publishing.

28. Гороховатський В., Творошенко І. (2026) Продуктивні моделі аналізу даних у методах розпізнавання зображень: монографія. Харків: ХНУРЕ. 153 с.

29. Nanonets. (2023). Automating Receipt Digitization with OCR and Deep Learning. URL: https://nanonets.com/blog/receipt-ocr/ (дата звернення 17.05.2026).

30. Kobylin, O.; Putiatina, O. (2025). Some aspects of real-time image denoising influenced by shot noise and compound Poisson noise. *CEUR Workshop Proceedings*, volume = 3943, 109-117

31. Кобилін, О. А., & Путятіна, О. Є. (2024). Real-time image denoising corrupted by shot-noise. *Системи обробки інформації*, 1(176), pp. 46-51.

32. Daradkeh Y.I., Gorokhovatskyi V., Tvoroshenko I., and Zeghid M. (2024) Improving the effectiveness of image classification structural methods by compressing the description according to the information content criterion, *Computers, Materials & Continua*, vol. 80, no. 2, pp. 3085-3106.

33. Gorokhovatskyi V., Tvoroshenko I., Yakovleva O., and Hudáková M. (2025) Image description compression in classification structural methods, *IEEE Access*, vol. 13, pp. 43631-43641.

34. Кобилін О.А., Творошенко І.С. (2021). Методи цифрової обробки зображень: навч. посібник. *Харків: ХНУРЕ*.

35. Conexiom. (2024). The 6 Biggest OCR Problems and How to Overcome Them. URL: https://conexiom.com/blog/the-6-biggest-ocr-problems-and-how-to-overcome-them (дата звернення 17.05.2026).

36. Yevstratov, M., Lyubchenko, V., Amer, A. J., & Lyashenko, V. (2024). Color correction of the input image as an element of improving the quality of its visualization. *Technical science research in Uzbekistan*, 2(4), 79-88.

37. Pyimagesearch. (2022). Correcting Text Orientation with Tesseract and Python. URL: https://pyimagesearch.com/2022/01/31/correcting-text-orientation-with-tesseract-and-python/ (дата звернення 17.05.2026).

38. Architecture and Technology Observations. (2017). Quick look at Text Detection in the Google Vision API. URL: https://blog.silvercode.co.nz/2017/02/21/quick-look-at-text-detection-in-the-google-vision-api/ (дата звернення 17.05.2026).

39. Microsoft. (2026). Inquiry about Preprocessing Steps in Azure AI Document Intelligence (Prebuilt Receipt Model). URL: https://learn.microsoft.com/en-us/answers/questions/2260723/inquiry-about-preprocessing-steps-in-azure-ai-docu (дата звернення 17.05.2026).

40. Improving the quality of the output. URL: https://tesseract-ocr.github.io/tessdoc/ImproveQuality.html (дата звернення 17.05.2026).

41. OCRopus. URL: https://uk.wikipedia.org/wiki/OCRopus (дата звернення 17.05.2026)

42. OpenCV. Canny Edge Detection. URL: https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html (дата звернення 17.05.2026).

43. OpenCV. Contours : Getting Started. URL: https://docs.opencv.org/4.x/d4/d73/tutorial_py_contours_begin.html (дата звернення 17.05.2026).

44. OpenCV. Hough Line Transform. URL: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html (дата звернення 17.05.2026).

45. OpenCV. Geometric Image Transformations. URL: https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html#:~:text=Applies%20a%20perspective%20transformation%20to%20an%20image.,the%20source%20image%20using%20the%20specified%20matrix: (дата звернення 17.05.2026).

46. Opensource. (2022). Fix scanned images with ImageMagick. URL: https://opensource.com/article/22/11/fixing-scanned-images-imagemagick (дата звернення 17.05.2026).

47. Github. (2021). Image Processing Features. URL: https://github.com/unpaper/unpaper/blob/main/doc/image-processing.md (дата звернення 17.05.2026).

48. Docparser. (2025). Improve OCR Accuracy With Advanced Image Preprocessing. URL: https://docparser.com/blog/improve-ocr-accuracy/ (дата звернення 17.05.2026).

49. Medium. (2023). EasyOCR pipeline from A to Z. URL: https://medium.com/@mohamed5elyousfi/-277e9c685578 (дата звернення 17.05.2026).

50. Medium. (2020). PyTorch: Scene Text Detection and Recognition by CRAFT and a Four-Stage Network. URL: https://medium.com/data-science/pytorch-scene-text-detection-and-recognition-by-craft-and-a-four-stage-network-ec814d39db05 (дата звернення 17.05.2026).

51. Paddle. General OCR Pipeline Tutorial. URL: https://paddlepaddle.github.io/PaddleX/3.2/en/pipeline_usage/tutorials/ocr_pipelines/OCR.html#222-python-script-integration (дата звернення 17.05.2026).

52. Jirasuwankul, N. (2011, July). Effect of text orientation to OCR error and anti-skew of text using projective transform technique. In *2011 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM)* (pp. 856-861). IEEE.

53. Yakovleva O., Nebeský L, Liakhov P. (2023). Research methods of texture image analysis to solve the texture search problem. Proceedings of the IV International Scientific and Practical Conference «The world of modern technologies and inventions». Vienna, Austria. 2023. pp. 252-261.

54. Leptonica. (2022). Measuring the Skew of Document Images. URL: http://www.leptonica.org/skew-measurement.html (Дата звернення: 17.05.2026)

55. Wang, J., Wang, C., & Huang, T. (2013, July). Efficient image contour detection using edge prior. In *2013 IEEE International Conference on Multimedia and Expo (ICME)* (pp. 1-6). IEEE.

56. Ababneh, J., Abu-Jassar, A., Abuowaida, S., Liubchenko, V,. Lyashenko, V. (204). Evaluation of Three Different Operators for Object Highlighting in Medical RGB Images: Canny, Roberts, and LoG in Independent Color Spaces, 2024 25th International Arab Conference on Information Technology (ACIT), 1-7.

57. Apple. Documentation. URL: https://developer.apple.com/documentation/coreimage/cifilter-swift.class/cannyedgedetector() (дата звернення 17.05.2026).

58. Adobe. (2018). The Science Behind Adobe Scan. URL: https://blog.adobe.com/en/publish/2018/06/18/science-behind-adobe-scan (дата звернення 17.05.2026).

59. Larin I., Tvoroshenko I., Gorokhovatskyi V., and Liubchenko V. (2025) Research and comparison of facial color normalization methods relative to an etalon image, *International Journal of Academic and Applied Research*, 9(11), pp. 36-47.

60. Lyashenko, V., Mohammad, A., Lyubchenko, V., & Kobylin, O. (2015). Methodology of Correlation Analysis in Solution of a Problem of Normalization of Projective Image Transformations. *International Journal of Scientific & Engineering Research*, 6(9), pp. 249-255.

61. OpenCV. Geometric Image Transdormations. URL: https://docs.opencv.org/4.x/da/d54/group__imgproc__transform.html#ga20f62aa3235d869c9956436c870893ae (дата звернення 17.05.2026).

62. Yakovleva, O., Matúšová, S., & Talakh, V. (2025, February 14). Gradio and Hugging capabilities for developing research AI applications. Proceedings of the VII International Scientific and Practical Conference «Scientific practice: modern and classical research methods», Boston, USA, pp. 202-205.

63. Yakovleva O., Matúšová S., Koshel V. Implementation of AI approaches in current tools for managing image collections to improve the search capabilities. Proceedings of the IV Correspondence International Scientific and Practical Conference «Science in motion: classic and modern tools and methods in scientific investigations» in Periodical International scientific journal «Grail of science». (February 21, 2025). Vinnytsia, Ukraine – Vienna, Austria. 2025. Vol. 49. P. 752 755. DOI: 10.36074/grail-of-science.21.02.2025.096.

64. Зуйко І.В. (2026). Аналіз та навчання YOLO-моделей для детекції та нормалізації чеків з метою підвищення точності розпізнавання тексту. Радіоелектроніка і молодь у XXI столітті: тези доповідей 30-го Міжнародного молодіжного форуму (Харків, 22–24 квітня 2026 р.). Харків: ХНУРЕ, 2026. Т. 7. С. 65-67.