Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Інтелектуальна комп'ютерна система розпізнавання графічних образів з використанням гетерогенних обчислень
Назва теми

Рівень вищої освіти *другий (магістерський)*

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

КвРКІП 240119.24.01.18 ПЗ

Виконав здобувач II курсу, група КІ2м-24-1 — Підпис — Максим МАКАРОВ (Ім'я, ПРІЗВИЩЕ)

Керівник — д. техн. наук, професор (Науковий ступінь, учене звання) — Підпис — Олег САВЕНКО (Ім'я, ПРІЗВИЩЕ)

Нормоконтролер — д. техн. наук, професор (Науковий ступінь, учене звання) — Підпис — Сергій ЛИСЕНКО (Ім'я, ПРІЗВИЩЕ)

До захисту допускаю:
завідувач кафедри КІІС — Підпис — Ольга ПАВЛОВА (Ім'я, ПРІЗВИЩЕ)
«01» травня 2026 р.
дата

Хмельницький 2026

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Рівень вищої освіти ДРУГИЙ (МАГІСТЕРСЬКИЙ)

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ
Завідувачка кафедри КІС
_____ Ольга ПАВЛОВА

" 12 " 01 2026 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Макарову Максиму Володимировичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Інтелектуальна комп'ютерна система розпізнавання графічних образів з використанням гетерогенних обчислень
Керівник проекту (роботи) Савенко Олег Станіславович, д.т.н., проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 12.01.2026 р. № 6
2. Термін подання здобувачем роботи на кафедру 01.05.2026 р.
3. Вихідні дані до роботи Завдання на кваліфікаційну роботу
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз відомих методів розпізнавання графічних образів та гетерогенних обчислень на мобільних пристроях
Модель інтелектуальної системи розпізнавання графічних образів з використанням гетерогенних обчислень
Метод розпізнавання графічних образів з використанням гетерогенного розподілу обчислювальних задач
Програмна реалізація та експериментальне дослідження інтелектуальної комп'ютерної системи розпізнавання графічних образів
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання «_12_»_____01_____2026 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) кваліфікаційної роботи магістра	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики КвРМ з керівником	12.01.2026	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	12.01.2026	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	20.01.2026	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	01.02.2026	виконано
5	Робота над науковою статтею	01.03.2026	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.03.2026	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, експериментальна частина	01.04.2026	виконано
8	Оформлення пояснювальної записки згідно вимог	18.04.2026	виконано
9	Попередній захист ДРМ	29.04.2026	виконано
10	Захист ДРМ на засіданні ЕК	Травень 2026	

Здобувач _____ Максим МАКАРОВ
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Олег САВЕНКО
Підпис Ім'я, ПРІЗВИЩЕ

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: Інтелектуальна комп’ютерна система розпізнавання графічних образів з використанням гетерогенних обчислень

Автор роботи: Макаров Максим Володимирович

Керівник роботи: Савенко Олег Станіславович

Пояснювальна записка: 87 с., 17 рис., 8 табл., 4 дод., 81 джерел.

ПЕРЕЛІК КЛЮЧОВИХ СЛІВ: РОЗПІЗНАВАННЯ ГРАФІЧНИХ ОБРАЗІВ, ГЕТЕРОГЕННІ ОБЧИСЛЕННЯ, ОПТИЧНЕ РОЗПІЗНАВАННЯ СИМВОЛІВ, НЕЙРОННІ МЕРЕЖІ, МОБІЛЬНІ ПРИСТРОЇ, КВАНТИЗАЦІЯ МОДЕЛЕЙ, ДИФЕРЕНЦІЙОВАНА БІНАРИЗАЦІЯ.

Об'єктом дослідження є процес розпізнавання графічних образів на мобільних пристроях з гетерогенною архітектурою обчислювачів.

Предметом дослідження є метод та інтелектуальна комп'ютерна система розпізнавання графічних образів з використанням гетерогенних обчислень.

Метою кваліфікаційної роботи магістра є підвищення ефективності розпізнавання графічних образів на мобільних пристроях шляхом розробки методу гетерогенного розподілу обчислювальних задач розпізнавання графічних образів між процесорами різного типу (CPU, GPU, NPU).

Для розв'язання поставлених задач використовувалися методи: теорії графів, методи оптимізації, математичної статистики, теорії нейронних мереж, методи квантизації та компресії моделей, дискретної математики.

Наукова новизна отриманих результатів:

− набув подальшого розвитку метод розпізнавання графічних образів, який, на відміну від існуючих, забезпечує поетапний розподіл підзадач конвеєра (сегментація, детекція, розпізнавання) між гетерогенними обчислювачами мобільного SoC з динамічним механізмом резервування, що дозволяє зменшити час обробки та забезпечити роботу в реальному часі;

На основі проведених досліджень розроблена архітектура і компоненти програмного забезпечення інтелектуальної комп'ютерної системи розпізнавання

графічних образів, яка реалізує запропонований метод гетерогенного розподілу обчислень на мобільних пристроях.

Практична значимість отриманих результатів полягає у розробленій системі, яка дозволяє виконувати розпізнавання фіскальних документів на мобільних пристроях без залежності від хмарних сервісів, забезпечуючи конфіденційність фінансових даних та роботу в автономному режимі.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ЦП – центральний процесор

ГП – графічний процесор

НП – нейронний прискорювач

СнК (SoC) – система на кристалі

ШНМ – штучна нейронна мережа

ЗНМ – згорткова нейронна мережа

РНМ – рекурентна нейронна мережа

ГНМ – графова нейронна мережа

OCR – оптичне розпізнавання символів

DAG – спрямований ациклічний граф

CTC – алгоритм вирівнювання послідовностей

QAT – квантизація з урахуванням навчання

PTQ – посттренувальна квантизація

FPN – піраміда ознак

SE – блок канальної уваги

BN – пакетна нормалізація

GN – групова нормалізація

MVVM – архітектурний шаблон модель-вигляд-модель вигляду

API – програмний інтерфейс застосунків

DMA – прямий доступ до пам'яті

NNAPI – програмний інтерфейс нейронних мереж Android

CoreML – фреймворк для розгортання моделей на пристроях Apple

TFLite – TensorFlow Lite, фреймворк для мобільного інференсу

INT8 – 8-бітне цілочисельне представлення

FP16 – 16-бітне представлення з рухомою комою

FP32 – 32-бітне представлення з рухомою комою

TOPS – трильйони операцій за секунду

FLOPs – кількість операцій з рухомою комою

FPS – кількість кадрів за секунду

ALU – цілочисельні обчислювальні блоки

FPU – модуль операцій з рухомою комою

F1 – F-міра (гармонічне середнє влучності та повноти)

IoU – перетин над об'єднанням

absNED – абсолютна нормалізована відстань редагування

DBNet – мережа диференційованої бінаризації

CRNN – згорткова рекурентна нейронна мережа

SVTR – візуальна модель розпізнавання тексту

LSTM – довга короткочасна пам'ять

BiLSTM – двонаправлена LSTM

PP-OCR – PaddlePaddle OCR, система розпізнавання тексту

U-Net – архітектура кодувальник-декодувальник для сегментації

MobileNetV3 – легковага мобільна архітектура нейронної мережі

RSEFPN – піраміда ознак із SE-блоками

ONNX – відкритий формат обміну нейронними мережами

RGB – кольоровий простір (червоний, зелений, синій)

SRAM – статична оперативна пам'ять

DRAM – динамічна оперативна пам'ять

ВСТУП

Розвиток мобільних технологій та методів штучного інтелекту створює передумови для обробки документів безпосередньо на кінцевих пристроях. Сучасні мобільні системи на кристалі (SoC) містять процесори різних типів: центральний процесор (CPU), графічний процесор (GPU) та нейронний прискорювач (NPU). Разом вони формують гетерогенну обчислювальну архітектуру. CPU ефективно виконує послідовні операції з розгалуженою логікою, GPU забезпечує масивний паралелізм для матричних обчислень, а NPU оптимізований для операцій згорткових нейронних мереж з мінімальним енергоспоживанням. Проте більшість існуючих рішень задіюють лише один тип процесора, і ефективне використання всіх наявних обчислювальних ресурсів для задач розпізнавання графічних образів залишається відкритою проблемою.

Практично значущою задачею розпізнавання графічних образів є обробка фіскальних документів: касових чеків, накладних, рахунків-фактур. В Україні щодня створюються мільйони таких документів, а їхня ручна обробка є трудомісткою та схильною до помилок. Хмарні сервіси оптичного розпізнавання символів (OCR) вирішують задачу розпізнавання, проте створюють ризики витоку конфіденційних фінансових даних та потребують стабільного інтернет-з'єднання. Мобільні рішення для локального розпізнавання, зокрема PaddleOCR та Tesseract, не здатні ефективно використовувати всі ресурси мобільного SoC. Вони працюють виключно на CPU або GPU, не задіюючи NPU, що обмежує швидкість обробки та унеможливлює роботу в реальному часі на пристроях середнього цінового сегменту.

Додаткову складність становить специфіка українськомовних документів. Українська абетка містить унікальні графеми (і, ї, є, ґ), які відсутні в більшості тренувальних наборів даних багатомовних OCR-систем. Фіскальні документи друкуються на термопапері, що швидко деградує, а також на матричних принтерах з низькою якістю відбитку. Стандартні системи розпізнавання не адаптовані до таких умов, що призводить до значного падіння точності.

Метою кваліфікаційної роботи магістра є підвищення ефективності розпізнавання графічних образів на мобільних пристроях шляхом розробки методу гетерогенного розподілу обчислювальних задач конвеєра розпізнавання між процесорами різного типу (CPU, GPU, NPU). Для досягнення поставленої мети необхідно розв'язати такі задачі:

− проаналізувати відомі методи розпізнавання графічних образів та гетерогенних обчислень на мобільних пристроях;

− розробити модель інтелектуальної системи розпізнавання графічних образів з використанням гетерогенних обчислень;

− розробити метод розпізнавання графічних образів з використанням гетерогенних обчислень;

− провести експериментальне дослідження розробленого методу та системи.

Об'єктом дослідження є процес розпізнавання графічних образів на мобільних пристроях з гетерогенною архітектурою обчислювачів. Предметом дослідження є метод та інтелектуальна комп'ютерна система розпізнавання графічних образів з використанням гетерогенних обчислень.

Для розв'язання поставлених задач використовувалися такі методи: теорії графів для моделювання конвеєра обробки як спрямованого ациклічного графу підзадач; методи оптимізації для розподілу задач між гетерогенними обчислювачами з мінімізацією загальної латентності; теорії нейронних мереж для побудови моделей детекції та розпізнавання текстових областей; методи квантизації та компресії моделей для адаптації нейронних мереж до обмежень мобільних процесорів; математичної статистики для оцінки достовірності експериментальних результатів.

Наукова новизна отриманих результатів полягає в наступному. Набув подальшого розвитку метод розпізнавання графічних образів, який, на відміну від існуючих, забезпечує поетапний розподіл підзадач конвеєра (сегментація, детекція, розпізнавання) між гетерогенними обчислювачами мобільного SoC з динамічним механізмом резервування, що дозволяє зменшити латентність обробки та забезпечити роботу в реальному часі.

Практична значимість отриманих результатів полягає у розробленій системі, яка дозволяє виконувати розпізнавання фіскальних документів на мобільних пристроях без залежності від хмарних сервісів, забезпечуючи конфіденційність фінансових даних та роботу в автономному режимі. Система реалізована як мобільний застосунок та протестована на реальних українськомовних фіскальних документах.

Кваліфікаційна робота магістра складається зі вступу, чотирьох розділів, висновків та списку використаних джерел. У першому розділі проведено аналіз відомих методів розпізнавання графічних образів та гетерогенних обчислень на мобільних пристроях, визначено їхні переваги та обмеження. У другому розділі розроблено модель інтелектуальної системи розпізнавання графічних образів з використанням гетерогенних обчислень. У третьому розділі описано метод розпізнавання графічних образів з використанням гетерогенних обчислень, що включає алгоритми розподілу підзадач між обчислювачами та механізм динамічного резервування. У четвертому розділі представлено програмну реалізацію системи та результати експериментального дослідження розробленого методу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА МЕТОДІВ РОЗПІЗНАВАННЯ ГРАФІЧНИХ ОБРАЗІВ

1.1 Огляд та поняття розпізнавання графічних образів

Теперішній етап розвитку економічних систем визначається інтенсивною цифровою трансформацією, котра ставить перед суб'єктами господарювання необхідність застосування інноваційних механізмів автоматизації задля зниження кількості помилок, пов'язаних із ручним введенням даних. Процес традиційного обліку первинної документації, зокрема фіскальні чеки та рахунки-фактури, демонструє значну ресурсоємність при високому рівні похибок, обумовленому антропогенним фактором. У відповідь на ці виклики сучасні фінансові екосистеми зазнають якісних змін у своєму розвитку. Вони трансформуються в напрямку розробки інтелектуальних програмних рішень, що переростають функцію елементарних цифрових носіїв інформації, набуваючи властивостей складних інструментів автономної агрегації даних про витрати. У цьому процесі ключову роль відіграє технологія оптичного розпізнавання символів (OCR), яка забезпечує миттєву конвертацію графічного представлення документів у структурований цифровий формат із подальшою можливістю аналітичної обробки та інтеграції в облікові системи [7, 60].

Розпізнавання графічних образів у широкому розумінні охоплює клас задач, де система автоматично інтерпретує візуальний вміст зображення та переводить його в структуровану форму. До цього класу належить розпізнавання рукописного тексту, друкованих документів, штрих-кодів, таблиць і форм. У контексті мобільних фінансових застосунків основним об'єктом аналізу є фіскальні документи, такі як касові чеки, квитанції та рахунки-фактури. Вони характеризуються специфічним набором проблем: термопапір схильний до вицвітання та фрагментації символів, поліграфія дешевих матричних принтерів створює розривні штрихи, а умови мобільної зйомки вносять перспективні та фотометричні спотворення.

Ефективність впровадження OCR-технологій демонструє прямий кореляційний зв'язок з розвитком апаратної бази, що ініціює трансформацію базових принципів взаємодії користувача з обчислювальними системами. Сучасний етап характеризується кардинальною зміною у парадигмі роботи із документами: інтеграція якісної оптики та потужних процесорів у мобільні пристрої перетворює їх на повноцінну альтернативу традиційному скануючому обладнанню. Архаїчні методи, що передбачали використання планшетних сканерів, поступаються місцем інноваційній концепції Mobile Vision. Остання забезпечує комплексний цикл обробки документів – від сканування та попередньої обробки зображень до екстракції ключових атрибутів безпосередньо в місці їх використання, наприклад, у фінансових установах або торгових точках.

Розширення функціонального потенціалу мобільних платформ формує об'єктивну потребу для створення спеціалізованих програмних архітектур, адаптованих до функціонування в умовах динамічного середовища. Вирішальним чинником успішності таких рішень є забезпечення стабільної роботи в режимі реального часу [4, 38], що визначає рівень користувацької лояльності. Життєздатність інтелектуальних систем детермінована двома ключовими параметрами: швидкодія алгоритмів обробки зображень і точність екстракції даних. Затримки в часі при аналізі чи помилки в ідентифікації важливих фінансових показників призводять до відмови від використання мобільних додатків [9], і це робить актуальним проведення досліджень з оптимізації алгоритмів комп'ютерного зору для роботи в умовах, де обчислювальні ресурси є обмеженими.

Окрім технічних викликів, лінгвістичний фактор відіграє значну роль через специфіку національної абетки. Українська мова містить унікальні графеми (а саме «і», «ї», «є», «ґ»), котрі мають надрядкові знаки. Через застосування низькоякісного термографічного друку ці компоненти можуть фізично поєднуватися, створюючи помітні вади. Нейронні мережі здебільшого класифікують ці дефекти як знаки пунктуації (апострофи або коми) чи випадковий шум. Застосування стандартних багатомовних OCR-моделей орієнтованих переважно на латиницю, спричиняє

критичне падіння метрик точності (Accuracy/F1-score) при обробці українських документів.

Це визначає потребу здійснення переходу від загальних рішень до глибоко адаптованих систем. Для досягнення стабільно високих показників слід забезпечити збір великих національних датасетів, котрі враховували б усі типи специфічних погіршень якості чеків. Тонке налаштування архітектур розпізнавання має бути збагачене інтелектуальними семантичними моделями для автоматичного виправлення лексичних огріхів вже на етапі постобробки.

Крім деградації носія, вагомою перешкодою для класичних алгоритмів є сама технологія нанесення символів. В дешевих секторах ринку замість лазерних принтерів доволі часто застосовують матричні пристрої, які використовують специфічні шрифти. Переважно кожен символ являє собою не безперервний масив пікселів, а конгломерат точок, котрі розташовані окремо у просторі. Через цю фізичну переривчастість завдається шкода важливим архітектурним засадам багатьох алгоритмів виявлення й розпізнавання, котрі покладаються на безперервність ліній та контурів.

Окрім недосконалості самих документів, процес розпізнавання ускладнюється через неконтрольоване середовище, в якому відбувається зйомка. На противагу промисловим системам OCR, в котрих умови освітлення та положення камери близькі до ідеальних, застосування смартфону супроводжується великим спектром фізичних аберацій, котрі можливо розділити на дві великі категорії: фотометричні та геометричні.

Фотометричні перешкоди зумовлені непередбачуваністю зовнішнього освітлення: від ледь помітних ламп розжарювання до яскравих відблисків від вікон чи глянцевих поверхонь документа. Через це виникає помітний перепад яскравості на площині чека, наприклад, через затінення рукою клієнта. У подібних сценаріях класичні алгоритми глобальної бінаризації, наприклад, статистичний метод Оцу, демонструють незадовільні результати. Обчислення єдиного порогу по цілому зображенню призводить до того, що затінені ділянки перетворюються на суцільні

чорні плями, а пересвітлені зони зливаються з білим фоном, що робить екстракцію тексту геть неможливою.

Навіть із застосуванням адаптивних фільтрів для часткового усунення фотометричних завад, викривлення евклідової геометрії документа становить значно складнішу проблему. У реальних сценаріях експлуатації мобільних додатків камера смартфона майже ніколи не розташовується в ідеальному положенні. Площина чека майже зовсім ніколи не буває паралельною до площини матриці камери. Це породжує вагомі перспективні викривлення: у горизонтальних рядках тексту спостерігається візуальне зближення навколо точок збігу на горизонті та непропорційна зміна масштабу символів вздовж площини документа. Розміщені поблизу об'єктива літери мають аномально великий вигляд, тоді як віддалені частини тексту стають дуже дрібними. Це суттєво впливає на стабільне розпізнавання символів нейромережами, як показано на рисунку 1.1.

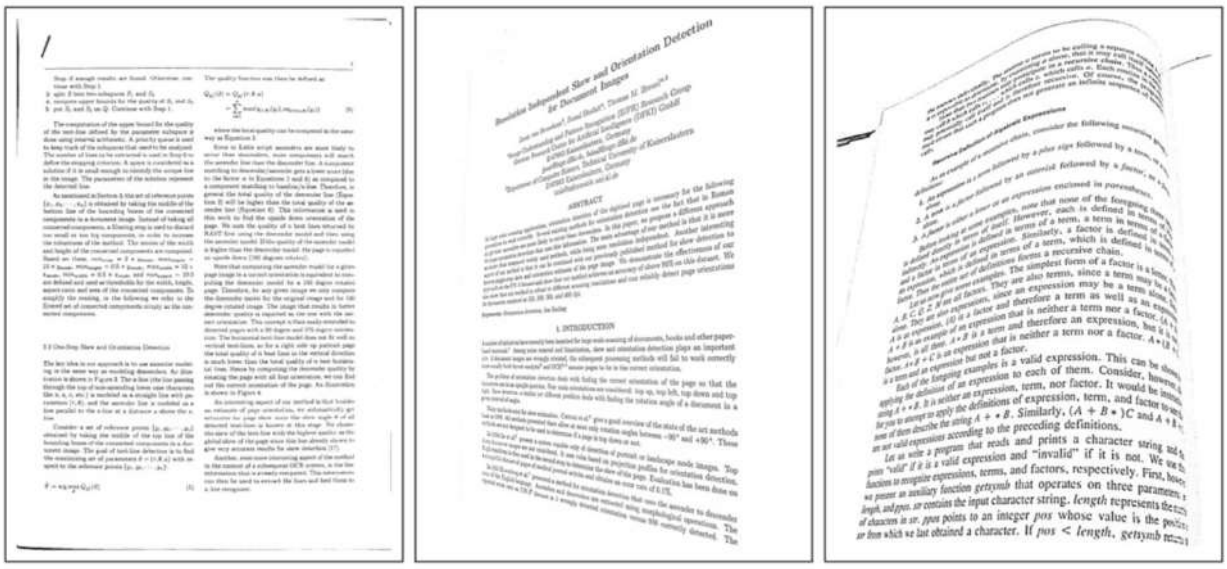


Рисунок 1.1 – Приклади геометричних викривлень документів [80]

У підсумку, ефективне проєктування мобільних систем комп'ютерного зору є поєднанням інженерних і математичних парадигм. Актуальність таких досліджень підкріплюється потужною експансією ринків корпоративної автоматизації. Через прогрес в області мобільних обчислень та імплементацію

методів агресивної компресії нейромереж, з'явилася можливість виконувати ресурсоємний інференс багатошарових трансформерних моделей безпосередньо локально на пристрої.

1.2 Аналіз архітектур нейронних мереж для задач OCR

Протягом тривалого часу підґрунтям для систем розпізнавання тексту в неконтрольованих умовах була архітектура CRNN. Ця модель працює поетапно. Мережі CNN спочатку виокремлюють основні візуальні риси зображення, а рекурентні мережі (наприклад, двонаправлені BiLSTM) далі розглядають їх у вигляді послідовності, враховуючи контекст [65]. На останньому кроці алгоритм CTC або механізм уваги робить із цих даних текстовий рядок [70]. Принцип роботи CRNN наведено на рисунку 1.2.

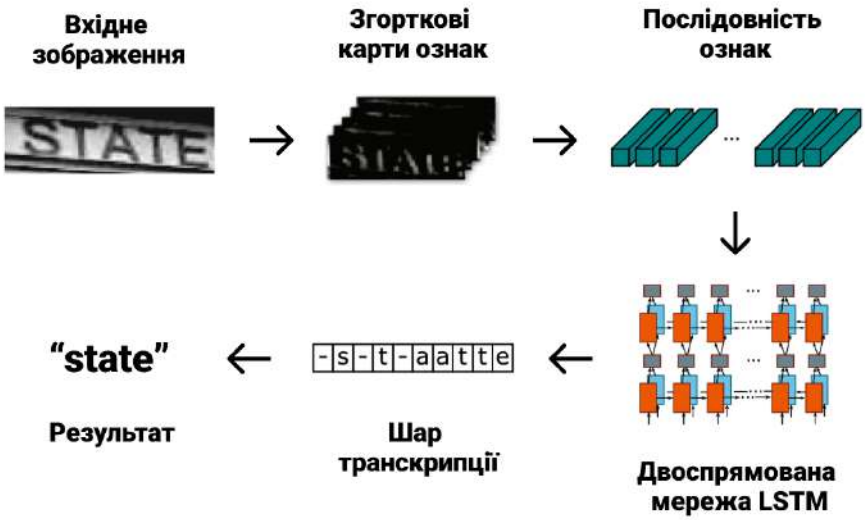


Рисунок 1.2 – Архітектура нейронної мережі CRNN для розпізнавання послідовностей символів на зображенні

Означений підхід працює добре з якісними сканами, проте його використання має суттєвий недолік у сучасних мобільних пристроях. Проблема полягає у математичній природі рекурентних мереж (RNN): вони обчислюють дані суворо за чергою, причому кожен наступний крок залежить від попереднього [14].

У зв'язку із цим, мобільні графічні процесори та нейронні процесори (GPU та NPU) не можуть застосувати масивні паралельні обчислення [75, 76]. Постає алгоритмічна проблема «вузького місця»: обчислювальні ядра пристрою працюють не за повної потужності, загальна пропускна здатність зменшується, а заряд акумулятора витрачається неефективно [74].

Для розв'язання цієї проблеми індустрія почала переходити до модульних архітектур, які надають змогу оптимізувати кожен етап розпізнавання зокрема. Одним із найбільш вдалих прикладів такого підходу є сімейство індустріальних моделей PP-OCR (версії v4 та v5). Їх головна ідея полягає у чіткому поділі процесу на незалежні один від одного кроки [23]. Робота починається з пошуку тексту через модуль DBNet, який робить процес бінаризації гнучким, інтегруючи його безпосередньо в нейромережу [49]. У зв'язку з цим система гнучко визначає пороги чутливості для кожного пікселя, дозволяючи якісно зчитувати текст навіть при різких тінях, нерівному світлі або вицвітанні термопаперу [50].

Після детекції, текст передається в модуль класифікації (наприклад, PP-LCNet), котрий визначає кут нахилу та виправляє геометричні спотворення документа. На фінальному етапі відбувається робота з модулем розпізнавання. Для досягнення великої швидкості, він застосовує прості базові мережі (на кшталт MobileNetV3) з методом дистиляції знань – способом, коли велика серверна модель передає свій "досвід" малій мобільній версії [23]. Зазначене поєднання технологій забезпечує швидку роботу системи в реальному часі та вимагає мінімального обсягу оперативної пам'яті смартфона [57].

Попри незаперечний успіх модульних конвеєрів, дальший розвиток у галузі переорієнтувався на уніфіковані візуальні моделі. Архітектура SVTR є важливим алгоритмічним досягненням, яке повністю відмовляється від рекурентних блоків та складних авторегресійних декодерів [24]. Ця спеціалізована згортково-трансформерна мережа використовує три ієрархічні стадії. Зменшення просторової висоти карт ознак відбувається на цих стадіях. Завдяки поступовій інтеграції латок через згортки, SVTR може оптимально зберігати безперервність розірваних штрихів, характерних для матричного друку.

Для української мови важливими є інтегровані блоки локального змішування. Завдяки механізму внутрішньої уваги, вони прецизійно відрізняють тіло літери від надрядкових діакритичних знаків (крапки над «і», «ї» або хвостик в «г»), запобігаючи їх хибному злиттю при низькій якості друку [24]. Разом з тим блоки глобального змішування захоплюють лінгвістичний контекст, завдяки чому автоматично виправляються оптичні помилки на рівні міжсимвольних патернів.

Для важких фінансових документів звичайного розпізнавання тексту недостатньо, тому потрібен розумний аналіз шаблонів і прив'язка даних до бізнес-сутностей. Для цього завдання успішно застосовується мультимодальна модель LayoutLM, яка розширює логіку трансформерів шляхом одночасної обробки текстової, просторової та візуальної інформації [60]. Інші end-to-end архітектури, наприклад Donut, повністю оминають OCR як окремий етап, створюючи відразу семантичні мітки на виході. Це ліквідує питання наростання помилок лавиною. Проте посимвольна генерація тексту залишається занадто повільною для мобільних SoC за відсутності радикального квантування [58].

У тих випадках екстремальних геометричних деформацій чеків найефективнішими виявляються графові нейронні мережі (GNN), де документ подається алгоритмічно як граф. Певні моделі використовують ітеративну агрегацію повідомлень від сусідніх текстових вузлів, забезпечуючи високу стійкість до топологічних викривлень зім'ятого паперу [64]. Такі моделі суттєво зменшують загальну кількість обчислень завдяки використанню розріджених матриць суміжності, через що інференс стає енергоефективним навіть на портативних пристроях [9, 42, 78].

1.3 Гетерогенні обчислення на мобільних пристроях

Через необхідність обробки транзакцій у реальному часі безпосередньо у місці їх здійснення ініційовано глобальну зміну обчислювальної парадигми – від централізованих хмарних сервісів до концепції граничних обчислень[78]. Трансформація стала можливою через інтеграцію у SoC спеціалізованих NPU [45],

що забезпечує повну децентралізацію обчислювальних процесів. Запропонований архітектурний підхід демонструє дві суттєві переваги: по-перше, він забезпечує максимальний ступінь конфіденційності через локальну обробку чутливої інформації, що зводить до мінімуму ризики перехоплення даних; по-друге, реалізація On-Device AI усуває залежність від якості мережевого з'єднання, забезпечуючи стабільність функціонування за будь-яких обставин.

Сучасна інженерна парадигма передбачає використання гетерогенних обчислювальних архітектур. У них інтелектуальний розподіл навантаження між CPU, GPU та NPU дозволяє розробникам знайти оптимальний баланс між максимальною пропускною здатністю важких нейромережевих конвеєрів та суворими обмеженнями енергоефективності акумуляторних батарей мобільних пристроїв. Порівняльна схема архітектур обчислювальних блоків наведена на рисунку 1.3.

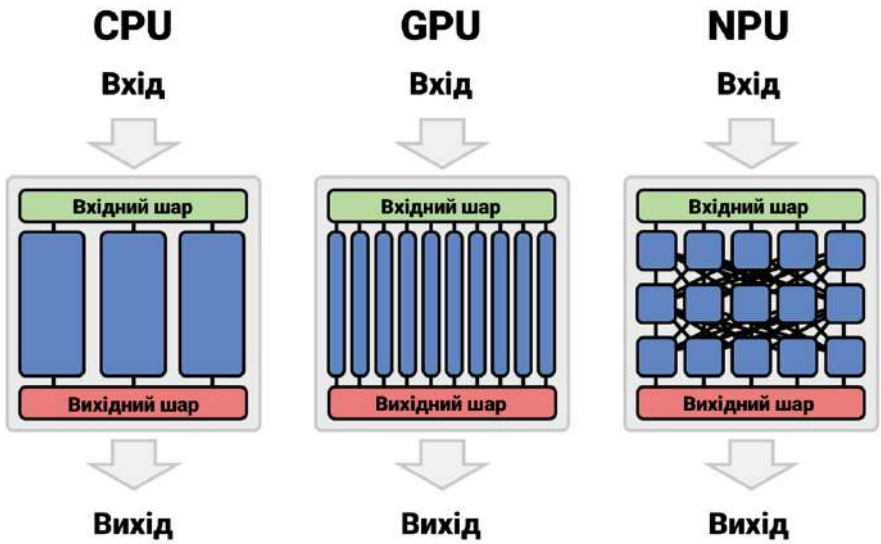


Рисунок 1.3 – Порівняльна схема архітектур CPU, GPU та NPU

Мобільні процесорні архітектури відтепер демонструють показники продуктивності, котрі ще донедавна були доступні виключно стаціонарним станціям. В екосистемі Apple чипи серії M [12], а також останні покоління серії A-Bionic інтегрують оновлений модуль Neural Engine, обчислювальна потужність якого сягає до 35 трильйонів операцій за секунду (TOPS). Це забезпечує ефективне

виконання інтенсивних матричних множень, що лежать в основі для багатошарових трансформерних моделей, при мінімальному енергоспоживанні.

В екосистемі Android рішення, що домінують, побудовані на базі архітектури ARM Cortex-X4 [13] та платформи Qualcomm Snapdragon 8 Gen 3 [62], використовують співпроцесор Hexagon NPU. Такі системи підтримують обчислення з нижчою точністю (INT4, INT8, FP16), що є критичним фактором для підтримки великої кількості FPS під час аналізу відео з камери в режимі реального часу.

Завдяки цим апаратним можливостям стало можливим створення надлегких OCR-систем, наприклад PP-OCRv3 та PP-OCRv4. Весь конвеєр розпізнавання з етапами детекції, ректифікації й екстракції тексту використовує лише декілька мегабайт пам'яті. Подібні моделі можуть обробляти зображення впродовж 30-50 мілісекунд навіть на звичайних процесорах.

Архітектура CPU традиційно спеціалізується на послідовному виконанні складних операцій з розгалуженою логікою: управління пам'яттю, виконання умовних переходів, обробка виключень. Для задач OCR CPU відповідає за організацію конвеєра, передачу тензорів між модулями та декодування фінальних послідовностей символів. GPU з тисячами спрощених ядер ефективно виконує паралельні операції над масивами даних. Для задач комп'ютерного зору GPU є оптимальним для виконання операцій згортки та розгортки, що лежать в основі детекторів тексту типу DBNet. Важливим є те, що матричне множення та операції перспективної трансформації безпосередньо прискорюються через стандартизовані мобільні API – OpenGL ES, Vulkan та Metal.

NPU є спеціалізованим прискорювачем, архітектурно оптимізованим для виконання операцій нейронних мереж. Ключовим елементом NPU є систолічні масиви, що дозволяють виконувати операції матричного множення на апаратному рівні без залучення загальної шини даних. Це кардинально знижує споживання енергії: NPU виконує ту ж кількість операцій, що й GPU, витрачаючи в 3-7 разів менше енергії. На сучасних платформах, таких як Qualcomm Snapdragon 8 Gen 3 [62] і Apple серії M/A-Bionic [12], NPU інтегруються поряд із великим обсягом

високошвидкісної кеш-пам'яті SRAM на кристалі. Це вирішує проблему із затримками та енерговитратами, що пов'язані із пересиланням даних із зовнішньої пам'яті (DRAM).

Ключовим поняттям гетерогенних обчислень є делегування: кожна операція виконується тим процесором, для якого вона є найбільш природною. Операції попередньої обробки зображення (фільтрація, нормалізація, бінаризація) делегуються на GPU через шейдери. Робота нейромережі (операції згортки, матричні множення трансформерів) виконується на NPU через відповідні інструменти Core ML (Apple), NNAPI (Android) або TFLite delegates. CPU залишається відповідальним за управлінську логіку, прийняття рішень щодо розподілу, а також за фінальне декодування та постобробку тексту.

Тривале використання GPU для задач комп'ютерного зору неминуче спричиняє знижувеня частоти обчислювача для запобігання перегріву. Разом з тим це призводить до стрімкого розряду батареї. Тому робоче навантаження динамічно перерозподіляється на NPU, архітектура якого вузько оптимізована для інтенсивних матричних операцій через систолічні масиви [45].

Реалізація ефективного гетерогенного конвеєра вимагає вирішення двох технічних викликів. По-перше, накладні витрати на копіювання тензорів між пам'яттю CPU, GPU та NPU можуть знецінити переваги від паралельного виконання. На сучасних SoC ця проблема вирішується використанням єдиного спільного банку пам'яті LPDDR5X, до якого всі процесори звертаються без копіювання. По-друге, не всі операції нейронних мереж підтримуються апаратним NPU. Нестандартні функції активації або операції з динамічними розмірами тензорів можуть бути автоматично перенаправлені на CPU через механізм резервування, що призводить до непередбачуваних затримок. Саме тому архітектурна конструкція нейронних мереж для мобільних платформ вимагає свідомого обмеження набору операцій лише до тих, що апаратно підтримуються NPU.

1.4 Методи оптимізації нейромережевих моделей для граничних пристроїв

У той час коли потужне апаратне забезпечення мобільних SoC забезпечує обчислювальну базу, нейронні мережі самі по собі також потребують алгоритмічної адаптації за допомогою методів агресивної компресії. Навіть у разі наявності NPU з показником 35 TOPS, нерозжата трансформерна модель із мільярдами параметрів не здатна функціонувати на мобільному пристрої в режимі реального часу через обмеження пам'яті та пропускної здатності шини даних.

Квантування є тим ключовим підходом, за допомогою якого відбувається перетворення 32-бітних ваг (FP32) в цілочисельні масиви INT8/INT4 [58, 69]. Це в чотири рази зменшує обсяг моделі, а також значно прискорює інференс. Стандартне квантування після тренування (PTQ) є найпростішим методом: ваги конвертуються в INT8 шляхом лінійного масштабування, без будь-якого додаткового тренування. Хоча цей метод швидкий в застосуванні, він може призводити до деградації точності у складних архітектурах з широким розподілом ваг.

Використання навчання із квантуванням (QAT) є суттєвим покращенням для трансформерів [59, 77]: QAT адаптує ваги вже на етапі донавчання, на відміну від типового пост-тренувального калібрування. З огляду на це, зберігається фінальна точність розпізнавання (F1-score) майже на рівні оригінальних моделей. Під час QAT у граф обчислень вставляються «псевдо-квантувальні» вузли, які симулюють помилки округлення. Модель навчається компенсувати ці помилки, знаходячи оптимальний розподіл ваг у межах цілочисельного представлення.

Структурна обрізка – це метод видалення статистично незначущих каналів [34] та зв'язків у шарах нейронної мережі. На відміну від так званої нерегулярної обізки, який видаляє окремі ваги і створює розріджені матриці, що погано підтримуються апаратно, структурна обрізка видаляє цілі фільтри або шари. Результатом є компактна модель, що ефективно виконується на стандартних процесорах без спеціальних бібліотек розрідженої алгебри. Метрикою для вибору

каналів до видалення є величина L1-норми ваг або значення активацій на тренувальних даних: канали з низькою нормою статистично мало впливають на вихід і можуть бути безпечно видалені.

Дистиляція знань є потужним методом передачі «розуміння» від великої точної моделі-вчителя до малої швидкої моделі-учня. У контексті OCR для мобільних платформ великий серверний варіант PP-OCR виступає в ролі вчителя. На відміну від навчання з жорсткими мітками (наприклад, «символ є буквою А»), учень навчається відтворювати м'який розподіл ймовірностей вчителя (наприклад, «62% ймовірність, що це А, 30% – ймовірність, що це Д»). Такий м'який розподіл несе значно більше інформації про структуру даних, ніж бінарна мітка, і дозволяє малій моделі досягнути точності, набагато вищої за ту, що можлива при прямому навчанні з нуля.

Апаратно-орієнтований пошук архітектури автоматично підбирає конфігурацію шарів для затримок конкретного чипа. На відміну від ручного проєктування архітектур, ця архітектура використовує алгоритми оптимізації для дослідження простору можливих архітектур з явним обмеженням на час відгуку інференсу. Цільова функція оптимізації включає не лише точність моделі на тестовій вибірці, але й виміряну затримку на цільовому пристрої. Це дозволяє автоматично знаходити архітектурні рішення, що є оптимальними саме для конкретного SoC.

Методи усунення спотвореннь з TPS-трансформаціями завершують конвеєр, відновлюючи геометричну цілісність деформованих чеків. Thin Plate Spline (TPS) математично моделює фізичну поведінку тонкої металевої пластини, що згинається під дією зовнішніх сил у визначених контрольних точках. Однак застосування повноцінних TPS на мобільних пристроях є непрактичним через необхідність розв'язання об'ємної системи лінійних рівнянь. Практичним компромісом є використання спрощеного усунення спотворення через гомографічне перетворення: матриця 3x3 описує перспективне перетворення між викривленою та вирівняною площинами документа, а її обчислення потребує лише координат чотирьох кутових точок.

Комплексна оптимізація включає синергію всіх перелічених методів. Типовий сценарій для задачі OCR передбачає застосування структурної обрізки для зменшення розміру бекбону, після чого модель донавчається з квантуванням QAT для збереження точності при INT8-представленні, і на завершення – дистиляція знань дозволяє компактній моделі наблизитися за метриками до повного серверного варіанту. Успішне розгортання систем комп'ютерного зору базується саме на такому поєднанні апаратних рішень із алгоритмами компресії. В таблиці 1.1 систематизовано ролі ключових обчислювальних модулів гетерогенної архітектури та їх призначення, а також вплив оптимізацій на загальну продуктивність.

Таблиця 1.1 – Роль апаратних компонентів та методів оптимізації у гетерогенних обчислювальних системах.

Компонент / Метод	Технічна роль	Вплив на систему
1	2	2
NPU	Виконання матричних множень через систолічні масиви.	Забезпечує хорошу продуктивність при малому енергоспоживанні.
Гетерогенний поділ	Розподіл графа між CPU, GPU та NPU залежно від типу операцій.	Мінімізація перегріву та максимальна пропускна здатність відеопотоку.
Квантування	Перетворення FP32 у INT8 з адаптацією ваг під час донавчання.	Зменшення об'єму моделі на 75% та апаратне прискорення обчислень на ALU.
Структурне обрізання	Видалення статистично значущих каналів та зв’язків у шарах.	Створення ультралегких бекбонів (MobileNetV3) вагою менше 2 МБ.

Кінець таблиці 1.1.

1	2	3
Усунення спотвореннь	Сіткова трансформація для відновлення топології документа.	Підвищення точності розпізнавання зім’ятих та деформованих чеків на 3-7%.

1.5 Порівняльний аналіз існуючих рішень

Системний аналіз розробок у сфері мобільного OCR вимагає структурованого зіставлення архітектурних підходів за ключовими критеріями: точністю розпізнавання, швидкістю інференсу, обсягом моделі та сумісністю з гетерогенними обчислювальними блоками мобільних SoC. Нижче в таблиці 1.2 показано еволюцію нейромережевих методів від класичних рекурентних конвеєрів до новітніх мультимодальних та графових парадигм, з акцентом на їх концептуальні переваги та апаратні обмеження в мобільних системах.

Таблиця 1.2 – Порівняльна характеристика нейромережевих архітектур для розпізнавання фіскальних документів.

Метод	Опис механізму	Переваги та обмеження
1	2	3
Класичний підхід (CRNN)	Поєднання згорткових та рекурентних мереж для аналізу ознак і послідовностей.	Висока точність на вирівняних документах; низька швидкість через неможливість розпаралелювання.
Оптимізований конвеєр (PP-OCR)	Розподілений цикл детекції та розпізнавання;	Висока швидкість та мале споживання пам'яті; ризик каскадного накопичення помилок між модулями.

Кінець таблиці 1.2.

1	2	3
Візуальні трансформери (SVTR)	Візуальна модель із механізмом локального та глобального змішування ознак.	Оптимізація під тензорні ядра; прецизійне розпізнавання специфічних символів української абетки.
Багатомодальні моделі (LayoutLM)	Злиття лінгвістичних, візуальних ознак та двовимірних координат у єдиний вектор.	Розуміння складних табличних структур; значне навантаження на оперативну пам'ять.
Наскрізні архітектури (Donut)	Пряма генерація структурованого тексту безпосередньо з пікселів зображення.	Усунення каскадних помилок; низька швидкість посимвольної генерації на мобільних пристроях.
Графові мережі (GLAM)	Моделювання макета як графа з механізмом передачі контекстних повідомлень.	Стійкість до геометричних деформацій паперу; ефективність завдяки розрідженим обчисленням.

Для задачі детекції тексту критичним є вибір алгоритму локалізації текстових регіонів, оскільки від нього залежить як точність подальшого розпізнавання, так і загальна пропускна здатність конвеєра. Алгоритм детекції має коректно визначати межі кожного рядка тексту за умов нерівномірного освітлення, часткового вицвітання термопаперу та присутності нетекстових елементів на документі (штрих-коди, логотипи, лінії розділення). Помилка на цьому етапі є незворотною: пропущений текстовий блок не потрапить до модуля розпізнавання, а неточно обрізаний фрагмент призведе до втрати символів на краях рядка. В таблиці 1.3 наведено порівняльну характеристику трьох провідних архітектур детекції тексту, які знаходять застосування в сучасних мобільних системах OCR.

Таблиця 1.3 – Порівняльна характеристика методів геометричної корекції

Метод	Математична основа	Сфера застосування	Обмеження
Deskew	Аналіз гістограм проекцій пікселів	Вирівнювання тексту в межах $\pm 15°$	Не здатний компенсувати перспективні викривлення (лише ротація)
Hough Transform	Полярна система координат(ρ, θ)	Пошук домінуючих ліній країв документа	Висока чутливість до фонового шуму та текстур
Perspective Transform	Матриця гомографії	Повна ректифікація об'єкта у фронтальну площину	Потребує точного визначення 4-х кутових точок

Для задачі розпізнавання символів після локалізації текстових рядків необхідно порівняти архітектури, що здатні перетворити піксельне представлення тексту на машиночитну послідовність символів. Вибір архітектури розпізнавання безпосередньо визначає сумісність із гетерогенними обчислювальними блоками мобільних SoC. В таблиці 1.4 наведено порівняльну характеристику чотирьох ключових архітектур розпізнавання.

Таблиця 1.4 – Порівняльна характеристика архітектур розпізнавання тексту для мобільних систем.

Архітектура	Підхід	Переваги	Недоліки
1	2	3	4
CRNN	CNN + RNN + CTC	Галузевий стандарт, висока швидкість на CPU, гнучкість довжини	Послідовна обробка Bi-LSTM створює «вузьке місце» для NPU

Кінець таблиці 1.4.

1	2	34	4
SVTR	Visual Transformer	Висока точність на NPU через GeMM-оптимізацію, стійкість до деформацій	Великі карти уваги можуть перевантажувати пам'ять GPU
PARSeq	Permuted Transformer	SOTA точність, ефективне використання контексту без зовнішніх LM	Висока обчислювальна складність для пристроїв з обмеженим VRAM
ABINet	Vision + Language Fusion	Глибоке розуміння лінгвістичного контексту, виправлення помилок візуального рівня	Складна модульна структура підвищує затримку інференсу

Аналіз чотирьох таблиць дозволяє зробити ключовий висновок: жодне з розглянутих рішень не забезпечує одночасно високої точності на деградованих документах, роботи в реальному часі та ефективного використання NPU. CRNN несумісна з паралельними можливостями NPU через послідовну природу RNN. PP-OCR працює на CPU та GPU, але не реалізує делегування задач на NPU. SVTR та PARSeq архітектурно сумісні з NPU, однак без глибокої оптимізації (QAT, обрізання) перевищують допустимі бюджети затримки та пам'яті. LayoutLM, Donut та GLAM є надто ресурсоємними для мобільного інференсу.

Таким чином, існує незаповнена ніша: відсутній метод гетерогенного розподілу конвеєра OCR між CPU, GPU та NPU з урахуванням специфіки кожного обчислювача, при збереженні прийнятної точності на документах з деградацією термодруку та специфікою українських графем.

1.6 Висновки до першого розділу та постановка задачі

Всебічне дослідження предметної області та технічних засобів показує, що вирішення проблеми ідентифікації фінансових документів у важких умовах мобільного сканування потребує комплексного підходу, що поєднує декілька напрямків.

Концептуальний аналіз підтвердив, що розподіл процесу на глибоко оптимізовані модульні конвеєри (як PP-OCR) та перехід до візуально-трансформерних парадигм без послідовного декодування (SVTR) став фундаментальним зсувом у галузі. Теоретичний аналіз засвідчує, що інтеграція механізмів локальної уваги становить перспективний підхід задля подолання проблеми злиття складних графем української абетки з нівелюванням дефектів деградованого термодруку. Для вилучення семантичного значення зі складних макетів необхідно застосовувати графові нейронні мережі (GNN), бо вони здатні забезпечити системі стійкість до топологічних деформацій зім'ятого паперу.

Успішне розгортання таких моделей на мобільних пристроях вимагає глибокої інтеграції алгоритмічних інновацій та сучасної апаратної інженерії. Перехід до парадигми гетерогенних обчислень шляхом динамічного делегування масивних тензорних операцій на нейронні процесори (NPU), у комбінації з агресивними методами компресії (QAT до формату INT8), розглядається як ключовий шлях для забезпечення інференсу в режимі реального часу.

Аналіз існуючих рішень виявив принципову незадоволену потребу: сучасний стан мобільного OCR не пропонує рішення, яке одночасно забезпечує і високу точність на реальних деградованих документах, і роботу в режимі реального часу, і ефективне використання всіх обчислювальних ресурсів гетерогенного SoC. Зокрема, PaddleOCR як найбільш поширений промисловий фреймворк функціонує переважно на CPU або GPU та не має механізму ефективного делегування задач на NPU, через що значна частина потенційної продуктивності мобільного пристрою залишається невикористаною.

На основі проведеного аналізу сформульовано наукову задачу дослідження: розробити метод розпізнавання графічних образів з використанням гетерогенних обчислень на мобільних пристроях, що дозволяє ефективно розподіляти обчислювальне навантаження між CPU, GPU та NPU і забезпечує роботу в режимі реального часу при збереженні необхідного рівня точності на деградованих фіскальних документах з текстом на українській мові.

Для досягнення поставленої наукової мети визначено чотири конкретні завдання дослідження:

1. Провести аналіз відомих методів розпізнавання графічних образів та принципів організації гетерогенних обчислень на мобільних пристроях, встановити їх обмеження та виявити незадоволені потреби в сфері мобільного OCR.

2. Розробити нейромережеву модель розпізнавання графічних образів, адаптовану до умов обробки деградованих фіскальних документів з текстом на українській мові, та оптимізовану для виконання на мобільних гетерогенних SoC.

3. Розробити метод гетерогенного розподілу обчислювального конвеєра OCR між CPU, GPU та NPU мобільного пристрою, що включає механізми квантування, структурної обрізки та дистиляції знань для забезпечення інференсу в реальному часі.

4. Провести експериментальне дослідження розробленої системи на репрезентативних наборах даних фіскальних документів, виміряти метрики точності, затримки та енергоефективності, порівняти результати з базовими рішеннями та оцінити практичну ефективність запропонованого методу.

2 МОДЕЛЬ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ГРАФІЧНИХ ОБРАЗІВ

2.1 Формалізована математична модель конвеєра розпізнавання

Побудова інтелектуальної системи розпізнавання графічних образів передбачає формалізацію кожного етапу обробки зображення. Підхід ґрунтується на принципі послідовної декомпозиції: спершу описується загальна логіка процесу на змістовному рівні, а потім ключові компоненти формалізуються математично.

Підрозділ описує вхідні дані та їх властивості, архітектуру конвеєра обробки як спрямованого ациклічного графа, а також моделі двох нейромережевих компонентів: детекції текстових областей та розпізнавання символів.

2.1.2 Вхідні параметри та простір зображень

Вхідними даними системи є зображення фіскального документа, отримане через камеру мобільного пристрою. На відміну від промислових сканерів, де умови захоплення контрольовані і стандартизовані, мобільна зйомка відбувається у довільних, непередбачуваних умовах. Користувач тримає пристрій під невідомим кутом відносно площини документа. Освітлення може бути нерівномірним: одна частина чека освітлена сонячним промінням, інша перебуває в тіні руки або корпусу телефону. Фізичний стан самого документа також є невизначеним: папір може бути частково зім'ятим, вигнутим або надірваним. Термопапір, на якому надруковано більшість фіскальних чеків, схильний до вицвітання під дією тепла та ультрафіолету, що призводить до часткового або повного зникнення окремих символів.

Додаткові складнощі створює технологія друку. Матричні принтери формують кожен символ з окремих точок, розташованих дискретно у просторі. Через фізичну переривчастість штрихів алгоритми виявлення тексту, що покладаються на безперервність контурів, працюють з помітними помилками. Термічний друк, найпоширеніший у сучасних касових апаратах, забезпечує

прийнятну якість на нових документах, проте деградує з часом непередбачувано: окремі ділянки можуть повністю зникнути, тоді як сусідні зберігають чіткість.

Для математичного опису зображення визначається як тривимірний масив інтенсивностей пікселів:

$$I = \{p(x, y, c) \mid x = 1..H, y = 1..W, c = 1..3\}, \quad (2.1)$$

де I – вхідне зображення;

$p(x, y, c)$ – значення інтенсивності пікселя;

x, y – просторові координати пікселя;

H, W – висота та ширина зображення у пікселях;

c – індекс кольорового каналу (червоний, зелений, синій).

Кожне значення $p(x, y, c)$ належить діапазону [0, 255] для 8-бітного представлення або [0, 1] після нормалізації. Система працює з кольоровими зображеннями у просторі RGB, оскільки колір несе інформацію, необхідну для розрізнення тексту від фонових об'єктів, зокрема логотипів, штрих-кодів та декоративних елементів чека. Типові розміри зображень, отриманих камерою сучасного смартфона, становлять від 3000 х 4000 до 4000 х 6000 пікселів. Перед обробкою нейромережевими компонентами зображення масштабується до фіксованої роздільності, визначеної архітектурою відповідного модуля.

Зображення, захоплене камерою, одночасно зазнає впливу декількох типів деградацій. Зім'ятий чек, сфотографований при штучному освітленні під кутом, одночасно містить геометричні спотворення (перспективне скорочення рядків, нелінійне викривлення поверхні), фотометричні деградації (нерівномірне освітлення, тіні, відблиски) та фізичні ушкодження носія (зминання, згини, вицвітання). Саме комбінаторна природа деградацій визначає основну складність задачі: система має бути стійкою не до окремих типів спотворень, а до їх довільних комбінацій. Множину типів деградацій позначимо як $D = \{d_1, d_2, \ldots, d_n\}$, де кожен елемент d_i відповідає конкретному типу спотворення.

Цільовий вихід системи являє собою структурований набір розпізнаних даних: назву торгової точки, дату та час операції, перелік товарних позицій з назвами, кількістю та цінами, підсумкову суму. Задача системи полягає у побудові відображення F:

$$F: I \rightarrow S, \tag{2.2}$$

де I – вхідне зображення згідно з формулою (2.1);

S – структурований результат розпізнавання, що містить назву торгової точки, дату, перелік товарних позицій та підсумкову суму.

Функція F має бути стійкою до всіх комбінацій деградацій з множини D при збереженні прийнятної точності та часу обробки для мобільного пристрою. Пряма побудова F як єдиної функції є непрактичною, тому наступний підрозділ описує декомпозицію F на послідовність спеціалізованих етапів.

2.1.2 Модель конвеєра як спрямований ациклічний граф

Побудування відображення F як єдиної функції є нездійсненним: задача охоплює різнорідні аспекти від аналізу пікселів до семантичного розбору тексту. Процес декомпозується на послідовність спеціалізованих етапів, що дозволяє оптимізувати кожен окремо та замінювати компоненти без перебудови конвеєра.

Для формалізації використовується апарат теорії графів. Направлений ациклічний граф (DAG) є природною моделлю для конвеєрів обробки даних, де вершини відповідають етапам обробки, а спрямовані ребра визначають потоки даних між ними. Ацикличність гарантує завершення обробки за скінченну кількість кроків.

Обробка зображення починається з етапу сегментації, на якому система виокремлює область документа із загальної сцени. Сегментаційна мережа з архітектурою MobileNetV3-Small як енкодер та U-Net як декодер приймає зображення, масштабоване до 320 x 256 пікселів, та генерує попіксельну маску з

активацією сигмоїдою. Далі виконується геометрична ректифікація: за чотирма кутовими точками, визначеними з контуру маски, обчислюється перспективне перетворення, що усуває геометричні спотворення та створює вирівняне прямокутне зображення документа. На вирівняному зображенні детектор текстових областей знаходить координати кожного рядка тексту, повертаючи множину обмежувальних полігонів. Знайдені фрагменти вирізаються та передаються модулю розпізнавання символів. На завершення модуль семантичного розбору витягує структуровані дані.

Конвеєр формалізується як DAG:

$$G = (V, E), \tag{2.3}$$

де G – граф конвеєра обробки;

V – множина з п'яти вузлів (сегментація, ректифікація, детекція, розпізнавання, семантичний розбір);

E – множина спрямованих ребер, що визначають послідовність обробки.

Кожен вузол v_i графа реалізує функцію перетворення f_i. Вузол сегментації приймає зображення у форматі $R^{(H\ x\ W\ x\ 3)}$та породжує бінарну маску розміром 320 x 256. Вузол ректифікації приймає вхідне зображення разом із координатами кутових точок та генерує вирівняне зображення документа. Вузол детекції знаходить текстові області та повертає множину обмежувальних полігонів $\{b_1, b_2, \ldots, b_N\}$, де N визначається кількістю рядків тексту. Вузол розпізнавання перетворює обрізане зображення фрагмента на послідовність символів. Вузол семантичного розбору формує структурований результат S.

Повний конвеєр записується як композиція функцій вузлів:

$$F = f_{\text{парс}}(f_{\text{розп}}(f_{\text{дет}}(f_{\text{ректр}}(f_{\text{сегм}}(I))))), \tag{2.4}$$

де F – позначає повне відображення конвеєра;

I – вхідне зображення згідно з формулою (2.1);

$f_{парс}$, $f_{розп}$, $f_{дет}$, $f_{ректр}$, $f_{сегм}$ – функції перетворення вузлів сегментації, ректифікації, детекції, розпізнавання та семантичного розбору відповідно.

Після вузла детекції конвеєр набуває паралельної структури: кожен знайдений текстовий фрагмент обробляється модулем розпізнавання незалежно від інших. Це розгалуження є значущим для гетерогенних обчислень, оскільки N незалежних задач розпізнавання можуть бути розподілені між доступними обчислювачами мобільного SoC. Типовий фіскальний чек містить від 10 до 50 текстових рядків.

Загальну структуру конвеєра як DAG показано на рисунку 2.1.

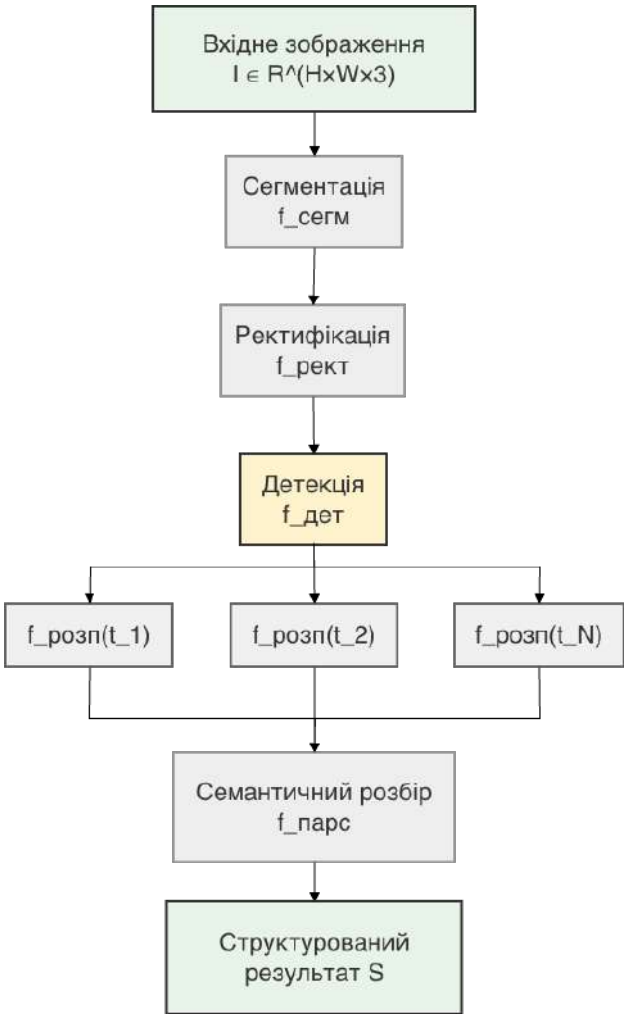


Рисунок 2.1 – Модель конвеєра розпізнавання у вигляді спрямованого ациклічного графу

Граф на рисунку 2.1 демонструє лінійний потік даних від сегментації до детекції та розгалуження після детекції на N паралельних гілок розпізнавання. Вузол детекції визначає точку розгалуження конвеєра: до нього обробка є послідовною, а після нього виникає множина незалежних задач. Вузол семантичного розбору агрегує результати всіх паралельних гілок та формує структурований вихід S.

Подання конвеєра як DAG визначає чіткі залежності між етапами: незалежні етапи виконуються одночасно на різних обчислювачах. Графова модель дозволяє визначити критичний шлях обробки та забезпечує незалежну заміну вузлів за умови збереження формату даних.

2.1.3 Модель детекції текстових областей

Детекція текстових областей на зображенні документа є задачею визначення просторових координат кожного рядка тексту. Від якості цього етапу безпосередньо залежить точність подальшого розпізнавання: якщо текстовий фрагмент не виявлено або його межі визначено неточно, відповідний рядок буде втрачено або розпізнано з помилками. Задача ускладнюється тим, що фіскальні документи містять елементи різного масштабу: великі заголовки, дрібний текст товарних позицій, числові значення цін, штрих-коди та декоративні елементи.

Класичний підхід до детекції тексту передбачає побудову карти ймовірностей з подальшою бінаризацією за фіксованим порогом. Для кожного пікселя нейронна мережа обчислює ймовірність $P(x, y)$ його належності до текстової області. Після цього всі пікселі з ймовірністю вище порогу T позначаються як текст. Цей підхід працює прийнятно при рівномірному освітленні, проте для фіскальних документів, сфотографованих камерою, він виявляється недостатнім.

Принциповою проблемою є використання єдиного порогу для всього зображення. При нерівномірному освітленні, коли тінь від руки перетинає

документ, оптимальне значення порогу суттєво відрізняється для різних ділянок: яскраво освітлені зони потребують високого порогу, а затінені потребують низького. Ділянки з яскравим освітленням при низькому порозі дають хибнопозитивні спрацювання, а затінені зони при високому порозі втрачають реальний текст. Стандартна бінаризація записується як:

$$B_{\text{ст}}(x,y) = \begin{cases} 1, \text{якщо } P(x,y) > T \\ 0, \text{інакше} \end{cases}, \tag{2.5}$$

де $B(x,y)$ – позначає бінарну маску;

$P(x,y)$ – ймовірність наявності тексту в позиції (x, y);

T – фіксований глобальний поріг.

Проблему фіксованого порогу наочно демонструє рисунок 2.2, де показано реальне зображення чека зі складними умовами освітлення та деградацією термопаперу.

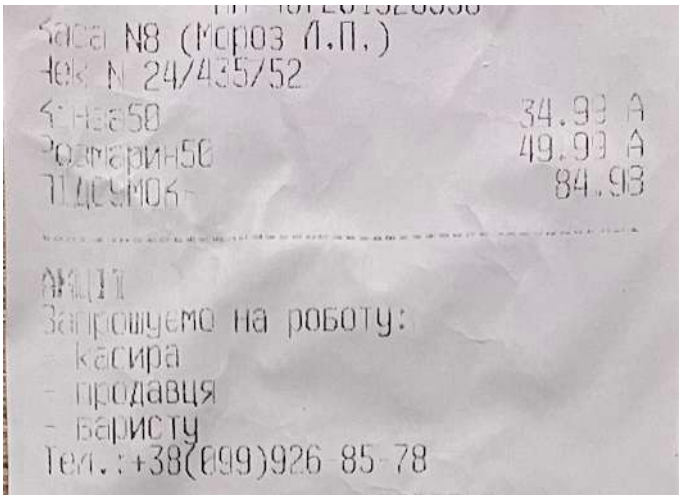


Рисунок 2.2 – Приклад складних умов освітлення та деградації носія

Для подолання обмежень стандартної бінаризації у системі використовується диференційована бінаризація. Ключова ідея полягає у тому, що нейронна мережа генерує не лише карту ймовірностей P, а й карту адаптивних порогів T, де кожен піксель має власне значення порогу. У затіненій ділянці порогові значення

автоматично знижуються, а у яскраво освітленій зоні підвищуються. Мережа навчається генерувати обидві карти одночасно через розгалуження на рівні фінального шару.

Замість жорсткого порівняння з порогом (що є недиференційовною операцією і не дозволяє навчати мережу методом зворотного поширення помилки) використовується гладке наближення через функцію сигмоїди з великим коефіцієнтом підсилення:

$$B_{\text{дб}}(x, y) = \sigma(k * (P(x, y) - T(x, y))), \quad (2.6)$$

де $B_{\text{дб}}$ – результат диференційованої бінаризації;

σ – функція сигмоїди, $\sigma(z) = \frac{1}{(1 + e^{-z})}$;

k – коефіцієнт підсилення, k = 50;

$P(x, y)$ – карта ймовірностей тексту;

$T(x, y)$ – карта адаптивних порогів.

Коефіцієнт $k = 50$ забезпечує крутий перехід сигмоїди поблизу точки $P(x, y) = T(x, y)$. При $k = 50$ сигмоїда наближається до ступінчастої функції Хевісайда, але зберігає неперервну похідну, що є необхідною умовою для навчання градієнтними методами. Перевага формули (2.6) над формулою (2.5) полягає саме у диференційованості: стандартна бінаризація є ступінчастою функцією з нульовою похідною, що унеможливлює обчислення градієнтів.

Архітектура детектора складається з трьох послідовних компонентів: базової мережі, піраміди ознак (FPN) та голови бінаризації.

Базова мережа MobileNetV3-Small витягує ознаки на трьох рівнях деталізації. Карти з кроком 4 зберігають дрібні деталі штрихів та серіфів літер. Карти з кроком 8 кодують елементи середнього масштабу: окремі літери та знаки пунктуації. Карти з кроком 16 відображають семантичні ознаки: цілі слова та рядки. Для вхідного зображення 960 x 640 ці карти мають розміри 240 x 160, 120 x 80 та 60 x 40 відповідно. Вибір MobileNetV3-Small обумовлений балансом між складністю та

якістю ознак завдяки інвертованим залишковим блокам з h-swish та механізмом канальної уваги.

Піраміда ознак (RSEFPN) агрегує карти різних рівнів в єдине представлення. Необхідність агрегації зумовлена тим, що текст на документі має різний масштаб: назва торгової точки може бути великою, а ціни дрібними. Агрегація виконується згори вниз: карта глибшого рівня збільшується до роздільності попереднього, після чого обидві додаються поелементно. На кожному етапі застосовується блок Squeeze-and-Excitation (SE), що адаптивно зважує канали: обчислює глобальне середнє по каналу, визначає важливість двошаровим блоком та множить кожен канал на вагу. Після агрегації всі рівні приводяться до єдиної роздільності з кроком 4, конкатенуються вздовж осі каналів та зливаються згорткою 3 x 3.

Голова бінаризації приймає злиту карту ознак та паралельно генерує дві карти: карту ймовірностей P та карту адаптивних порогів T. Обидві гілки мають ідентичну структуру: згортка 3 x 3, пакетна нормалізація (BN), активація ReLU, згортка 1 x 1 з активацією сигмоїдою. Карти P та T подаються у формулу диференційованої бінаризації (2.6).

Результатом роботи детектора є множина обмежувальних полігонів, отриманих з бінарної маски шляхом пост-обробки. Послідовність операцій така: спочатку бінарна маска фільтрується за пороговим значенням для усунення шумових активацій, потім виконується пошук зв'язних компонентів, кожен з яких відповідає потенційній текстовій області. Для кожного компонента обчислюється полігональна апроксимація контуру, що дозволяє описати довільну форму текстового блоку чотирикутником. На завершення виконується розширення полігонів на фіксований відсоток від їхнього периметру.

Розширення полігонів є критичним для української мови: крапки над літерами «і» та «ї», хвостик літери «ґ» та інші надрядкові елементи можуть виходити за межі основного контуру тексту. Без розширення ці діакритичні знаки будуть відсічені від зображення фрагмента, і модуль розпізнавання отримає неповну інформацію. Коефіцієнт розширення підбирається емпірично на валідаційному наборі українськомовних документів. На практиці значення

коефіцієнта 1.5 від відстані між центром полігону та його краєм забезпечує захоплення діакритичних знаків у переважній більшості випадків, не додаючи зайвого фону, який міг би погіршити якість розпізнавання.

Після розширення полігони впорядковуються у природному порядку читання: зверху вниз за вертикальною координатою центру, а для полігонів на одній горизонтальній лінії зліва направо. Це впорядкування необхідне для модуля семантичного розбору, який очікує рядки тексту у послідовності, що відповідає логічній структурі документа.

2.1.4 Модель розпізнавання символів

Після детекції текстових областей кожен знайдений фрагмент обробляється модулем розпізнавання, який перетворює піксельне зображення тексту на послідовність символів. Цей етап є найбільш обчислювально інтенсивним компонентом конвеєра, оскільки виконується для кожного текстового фрагмента окремо, а типовий чек містить десятки фрагментів.

Вхідними даними модуля є обрізане зображення текстового фрагмента t_j, витягнуте за координатами полігону детектора. Зображення нормалізуються за висотою до 48 пікселів зі збереженням пропорцій, а потім групуються в пакети за шириною для мінімізації порожнього заповнення та ефективного використання паралелізму.

Архітектура побудована на основі SVTR, яка повністю відмовляється від рекурентних блоків та складних авторегресійних декодерів. Рекурентні мережі, такі як LSTM та BiLSTM, обчислюють дані суворо послідовно: кожен наступний крок залежить від результату попереднього. Ця послідовна залежність унеможливлює паралельне виконання на GPU та NPU мобільних пристроїв, створюючи обчислювальне вузьке місце. SVTR замінює рекурентні блоки механізмом уваги, який обчислює зв'язки між усіма позиціями одночасно через матричне множення.

Перед розпізнаванням зображення розбивається на послідовність ділянок. Кожна ділянка перетворюється на вектор ознак через згорткові шари. Послідовність $Z = \{z_1, z_2, \ldots, z_L\}$, де L визначається шириною зображення, подається на блоки уваги.

Механізм масштабованої точкової уваги:

$$\text{Attention}(Q, K, V) = \text{softmax}(Q * \frac{K^T}{\sqrt{d_k}}) * V, \qquad (2.7)$$

де Q, K, V – матриці запитів, ключів та значень, обчислені лінійними проєкціями послідовності Z;

d – розмірність ключів, використовувана для масштабування;

K^T – транспонована матриця ключів.

Добуток $Q * K^T$ обчислює попарну подібність між усіма ділянками. Ділення на sqrt(d) є прийомом стабілізації: без масштабування скалярні добутки набувають великих значень, що призводить до концентрації softmax та зникаючих градієнтів. Функція softmax перетворює оцінки подібності на нормалізовані ваги, після чого зважена сума значень V формує вихідне представлення.

В архітектурі SVTR механізм уваги застосовується у двох режимах. Локальна увага обмежує область взаємодії до сусідніх позицій у фіксованому вікні через додавання маски з значеннями -∞ для віддалених позицій. Вона фокусується на морфологічних деталях окремих літер: формі штрихів, положенні діакритичних знаків. Для української мови локальна увага є критичною, оскільки забезпечує розрізнення візуально подібних символів: «и» та «й», «і» та «ї», «г» та «ґ», де різниця полягає у наявності малого надрядкового елемента.

Глобальна увага працює без маски обмеження, дозволяючи кожній ділянці взаємодіяти з усіма іншими. Вона захоплює лінгвістичний контекст: при низькій якості друку окремі літери можуть бути нерозрізненними візуально (кириличне «О» та цифра «0», «З» та «3»), проте контекст сусідніх символів дозволяє системі обрати правильний варіант.

Ієрархічне поєднання реалізується через три послідовні стадії SVTR. На перших стадіях переважає локальна увага, що витягує морфологічні ознаки символів. На глибших стадіях переважає глобальна увага, що формує контекстне представлення рядка. Просторова висота карт ознак зменшується на кожній стадії через згортки з кроком 2.

Для перетворення послідовності представлень на символи застосовується CTC-декодування. Проблема полягає у вирівнюванні: вхідна послідовність Z має фіксовану довжину L, а вихідна послідовність символів має змінну довжину. CTC вводить спеціальний порожній символ ε:

$$P(C \mid Z) = \Sigma_{\pi} P(\pi \mid Z), \tag{2.8}$$

де C – цільова послідовність символів;

π – конкретне вирівнювання довжиною L, що містить символи алфавіту та порожній символ ε;

$P(\pi \mid Z)$ – ймовірність вирівнювання, обчислена як добуток ймовірностей символів на кожній позиції;

Σ_{π} – сума по всіх вирівнюваннях π, що після видалення ε та послідовних дублікатів дають C.

На етапі інференсу декодування виконується жадібним алгоритмом: для кожної позиції обирається символ із максимальною ймовірністю, після чого послідовні дублікати та порожні символи видаляються. При виявленні семантичних невідповідностей активується альтернативне декодування методом променевого пошуку, що утримує декілька найімовірніших гіпотез одночасно.

Після декодування розпізнана послідовність проходить лексичну пост-обробку з використанням N-грамних статистик. Пост-обробка коригує типові помилки розпізнавання, зокрема плутання візуально подібних символів кирилиці та латиниці (кириличне «а» та латинське «a», кириличне «о» та латинське «o»). Корекція ґрунтується на статистичній ймовірності N-грам у українській мові. Для

числових полів додатково застосовується перевірка формату та арифметична верифікація.

2.2 Математична модель гетерогенного розподілу обчислень

У підрозділі 2.1 побудовано формальну модель конвеєра як DAG $G = (V, E)$, де кожен вузол реалізує функцію перетворення. Проте модель описує лише логічну структуру обробки, не визначаючи, на якому обчислювальному ресурсі виконуватиметься кожен вузол. У мобільному SoC одночасно доступні три типи процесорів з принципово різною архітектурою та спеціалізацією. Раціональний розподіл обчислень між цими процесорами є ключовою задачею, що визначає загальну продуктивність системи. Виконання всіх вузлів на одному процесорі (наприклад, лише на ЦП) призводить до неефективного використання апаратних ресурсів та надмірного енергоспоживання. Водночас некоректний розподіл, при якому обчислювально інтенсивні операції потрапляють на непристосований для них процесор, може збільшити загальну час відгуку замість його зменшення.

Підрозділ формалізує множину ресурсів, функцію призначення вузлів з механізмом каскадного переключення, модель часу виконання та задачу оптимізації.

2.2.1 Множина обчислювальних ресурсів та функція призначення задач

Мобільний SoC містить гетерогенний набір обчислювальних ресурсів. Центральний процесор (ЦП) виконує послідовні обчислення загального призначення з гнучким набором інструкцій. Для конвеєра розпізнавання ЦП відповідає за організацію потоку даних, декодування послідовностей символів та семантичний розбір.

Графічний процесор (ГП) містить сотні спрощених обчислювальних ядер для паралельних операцій: згортки, матричних множень, геометричних перетворень зображень. У конвеєрі ГП ефективно виконує сегментацію та геометричну

ректифікацію. Доступ здійснюється через OpenGL ES та Vulkan на Android, Metal на iOS.

Нейронний прискорювач (НП) є спеціалізованим процесором, оптимізованим для операцій нейронних мереж. Ключовим елементом є систолічні масиви для матричного множення на апаратному рівні. НП забезпечує найвищу продуктивність на стандартних нейромережевих операціях, проте підтримує лише обмежений набір операцій. Доступ здійснюється через Core ML (Apple), NNAPI (Android) або делегати TensorFlow Lite.

$$R = \{r_{\text{цп}} r_{\text{гп}} r_{\text{нп}}\}, \tag{2.9}$$

де $r_{\text{цп}}$ – центральний процесор (CPU);

$r_{\text{гп}}$ – графічний процесор (GPU);

$r_{\text{нп}}$ – нейронний прискорювач (NPU).

Кожен ресурс характеризується піковою продуктивністю c_i (операцій за секунду), обсягом пам'яті m_i (байт) та коефіцієнтом енергоефективності e_i (операцій на джоуль). Продуктивність ЦП на операціях з плаваючою точкою становить порядку 10^{10} ops/s, ГП досягає 10^{11} ops/s завдяки паралелізму, а НП забезпечує 10^{12} ops/s на цілочисельних INT8. Енергоефективність НП у 10-20 разів перевищує ЦП, що є критичним для мобільних пристроїв. Проте обсяг пам'яті НП становить лише 1-4 МБ (власна SRAM на кристалі), тоді як ЦП та ГП мають 4-8 ГБ (спільна LPDDR5X).

Не кожен вузол конвеєра може виконуватися на довільному ресурсі. НП підтримує стандартні згортки та активації, проте не здатний виконувати деформовані згортки (сегментація) або білінійну інтерполяцію з довільною матрицею (ректифікація). ГП підтримує ширший набір операцій, проте текстовий розбір та механізм уваги з нерегулярними патернами доступу до пам'яті виконуються неефективно. Конкретні значення матриці сумісності наведено в таблиці 2.1.

Таблиця 2.1 – Матриця сумісності вузлів конвеєра з обчислювальними ресурсами

Вузол конвеєра	НП	ГП	ЦП	Пріоритетний ресурс
Сегментація ($v_{сегм}$)	-	+	+	ГП
Ректифікація ($v_{ректи}$)	-	+	+	ГП
Детекція ($v_{дет}$)	+	+	+	НП
Розпізнавання ($v_{розп}$)	+	-	+	НП
Парсинг ($v_{парс}$)	-	-	+	ЦП

Функція призначення задач визначає відображення кожного вузла конвеєра на обчислювальний ресурс:

$$\mathrm{A}(v_i) = r_j, \tag{2.10}$$

де v_i – множина вузлів конвеєра;

r_j – множина обчислювальних ресурсів.

Функція має задовольняти обмеження сумісності з таблиці 2.1. Не завжди ресурс з найвищим пріоритетом є доступним: на бюджетних пристроях НП може бути відсутнім, ГП може бути зайнятий відображенням інтерфейсу. У таких випадках застосовується механізм каскадного пріоритетного переключення: НП → ГП → ЦП. ЦП є універсальним ресурсом, що підтримує всі операції конвеєра, тому завжди виступає останнім елементом каскаду.

Каскадне переключення забезпечує прозору адаптацію до різних апаратних платформ. На бюджетних пристроях без НП детекція та розпізнавання автоматично переносяться на ГП або ЦП зі зниженою продуктивністю, проте без втрати функціональності. На флагманських SoC більшість нейромережевих обчислень виконується на НП з мінімальним енергоспоживанням. Ця адаптивність забезпечується виключно зміною функції призначення A без модифікації логіки конвеєра.

Система також здійснює динамічне коригування: якщо операція на призначеному ресурсі завершується з помилкою (нестача пам'яті, непідтримувана операція), виконання автоматично повторюється на наступному ресурсі каскаду. Результат виконання зберігає інформацію про фактично використаний ресурс та час обробки, що дозволяє системі адаптувати часові ліміти при наступних запусках та накопичувати статистику ефективності кожного обчислювача для конкретного пристрою.

2.2.2 Модель часу виконання та цільова функція оптимізації

Час виконання кожного вузла визначається двома складовими: часом безпосередніх обчислень та часом ініціалізації ресурсу. Обчислювальна складність вузла ділиться на продуктивність ресурсу, що дає теоретичний час обчислень. До нього додається час ініціалізації: завантаження ваг моделі, виділення буферів, компіляція обчислювального графа. Для ЦП час ініціалізації мінімальний (мікросекунди). Для ГП він включає компіляцію шейдерних програм (десятки мілісекунд при першому запуску). Для НП – завантаження скомпільованого графа (сотні мілісекунд). На практиці ініціалізація виконується один раз при першому запуску, а при повторних запусках використовуються закешовані ресурси.

Загальний час обробки також включає передачу даних між обчислювальними ресурсами. У сучасних мобільних SoC ЦП та ГП використовують спільну оперативну пам'ять, тому передача даних зводиться до передачі вказівника. Для НП з власною пам'яттю передача виконується через прямий доступ до пам'яті (DMA), проте для типових розмірів проміжних даних (1-10 МБ) час копіювання не перевищує 1-2 мс, що є незначним порівняно з часом обчислень.

Загальний час виконання конвеєра при заданій функції призначення A:

$$T_{\text{заг}} = T_{\text{обч}} + T_{\text{перед}}, \tag{2.11}$$

де $T_{\text{заг}}$ – позначає загальний час виконання конвеєра;

$T_{\text{обч}}$ – сумарний час обчислень на всіх вузлах;

$T_{\text{перед}}$ – сумарний час передачі даних між обчислювачами.

Час обчислень $T_{обч}$ визначається як сума часів виконання кожного вузла на призначеному обчислювачі. Час передачі $T_{\text{перед}}$ залежить від того, чи суміжні вузли виконуються на одному ресурсі ($T_{\text{перед}} = 0$) чи на різних (передача через спільну пам'ять або DMA).

Для розгалуженої частини конвеєра (паралельна обробка N текстових фрагментів) час визначається найдовшим фрагментом. Фрагменти групуються в пакети за довжиною для зменшення порожнього заповнення.

Розподіл обчислень має задовольняти набір обмежень. Обмеження реального часу: загальний час $T_{\text{заг}} <= T_{\text{макс}} = 2\text{c}$, що забезпечує прийнятний користувацький досвід. Обмеження пам'яті: сумарний обсяг моделей на кожному ресурсі не перевищує його пам'яті. Обмеження якості: метрика $F1 >= Q_{\text{мін}} = 0.95$.

Задача оптимізації розподілу обчислень:

$$A_{\text{опт}} = \min_{\text{A}\in\text{A}} T_{\text{заг}}\,(\text{A}), \tag{2.12}$$

де $A_{\text{опт}}$ – позначає оптимальну функцію призначення, що мінімізує загальний час $T_{\text{заг}}$ за формулою (2.11).

A – множина всіх допустимих конфігурацій розподілу обчислень.

Оптимізація виконується за трьох обмежень. По-перше, загальний час не перевищує допустимий поріг: $T_{\text{заг}} <= 2$ с. По-друге, сумарний обсяг моделей на кожному обчислювачі не перевищує його доступну пам'ять. По-третє, якість розпізнавання за F1-мірою залишається не нижчою за 0.95. Якість залежить від функції призначення, оскільки виконання моделей на НП з квантизацією INT8 може призводити до незначної втрати точності порівняно з обчисленнями на ГП у FP32. Простір пошуку з урахуванням сумісності (таблиця 2.1) становить $2 * 2 * 3 * 2 * 1 = 24$ варіанти, що розв'язується повним перебором.

Схему оптимального розподілу показано на рисунку 2.3.

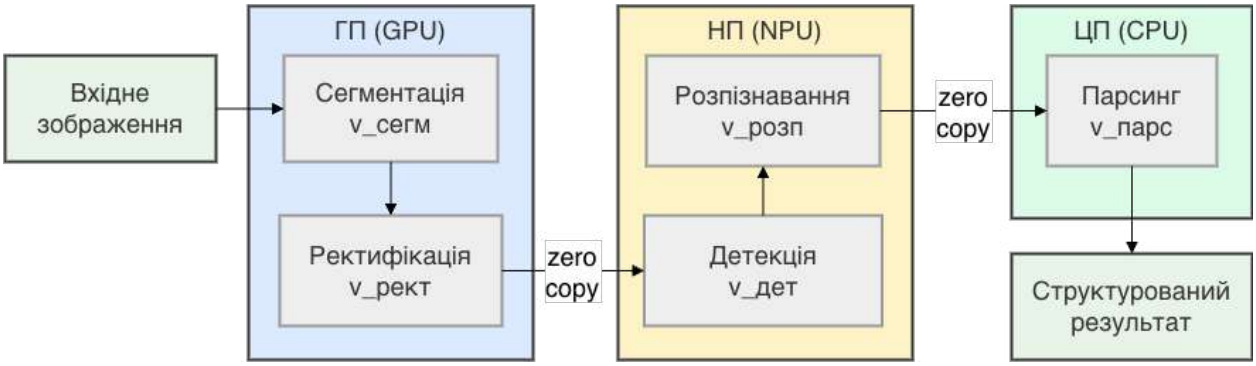


Рисунок 2.1 – Схема призначення вузлів конвеєра на обчислювальні ресурси

На рисунку 2.3 показано розподіл вузлів конвеєра при оптимальному призначенні. Сегментація та ректифікація виконуються на ГП, де забезпечується ефективне обчислення з масивним піксельним паралелізмом. Детекція та розпізнавання виконуються на НП з максимальною продуктивністю при мінімальному енергоспоживанні. Семантичний розбір виконується на ЦП як послідовна текстова операція. Передача між ГП та НП використовує передачі вказівника через спільну пам'ять. Передача від НП до ЦП через DMA (порядку 1 мс) є знехтовно малою. Такий розподіл забезпечує $T_{\text{заг}} <= 2$ с навіть на SoC середнього класу.

2.3 Модель оптимізації нейромережевих моделей

Для дотримання обмежень пам'яті та часу виконання, визначених у задачі оптимізації (2.12), нейромережеві моделі потребують оптимізації перед розгортанням на мобільному пристрої. Базові моделі, навчені на серверних GPU у форматі FP32 (32 біти на значення), займають значний обсяг пам'яті та виконуються надто повільно для мобільного SoC. Підрозділ описує три методи оптимізації: квантизацію ваг, обрізку нейронів та злиття операторів. Детальна алгоритмічна реалізація описана у третьому розділі.

Квантизація є основним методом зменшення обсягу моделей. Суть полягає у зниженні точності представлення ваг з формату FP32 (4 байти на значення) до INT8 (1 байт на значення). Кожне значення ваги перетворюється:

$$w_{\text{кв}} = round(\frac{w}{s}) * s, \quad (2.13)$$

де w – оригінальне значення ваги у форматі FP32;

$w_{\text{кв}}$ – квантизоване значення;

s – крок квантизації, що визначається діапазоном ваг у конкретному шарі як $s = \frac{(w_{max} - w_{min})}{(2^b - 1)}$;

b – кількість бітів квантизації (b = 8 для INT8).

При переході від FP32 до INT8 кожне значення ваги займає у 4 рази менше пам'яті. Для моделі детекції DBNet з обсягом 4.2 МБ квантизована версія займає 1.05 МБ. Для моделі розпізнавання SVTR з обсягом 8.8 МБ квантизована версія займає 2.2 МБ. Сумарний обсяг після квантизації не перевищує 4 МБ, що гарантує розміщення у пам'яті НП. Квантизація також прискорює обчислення у 2-4 рази завдяки апаратним інструкціям INT8.

Квантизація вносить похибку округлення, що може впливати на точність. Для мінімізації цієї похибки застосовується QAT. При QAT операції квантизації та деквантизації вставляються у граф обчислень під час тренування. Мережа імітує цілочисельну арифметику при прямому проході, а при зворотному поширенні округлення апроксимується тотожним відображенням. В результаті мережа поступово компенсує похибки округлення зміною значень ваг, забезпечуючи мінімальну втрату точності.

Обрізка зменшує обчислювальну складність моделі шляхом видалення найменш значущих фільтрів згорткових шарів. Значущість фільтра оцінюється за нормою L1 його ваг: фільтри з малою нормою створюють слабкі активації, що мінімально впливають на вихід шару. Коефіцієнт обрізки становить 30-50 % залежно від глибини шару: початкові шари обрізаються менш агресивно, оскільки

втрата базових ознак критично впливає на наступні шари; глибокі шари допускають більшу обрізку через надлишковість представлення. Після обрізки мережа донавчається протягом декількох епох для відновлення точності.

Злиття операторів об'єднує декілька послідовних операцій в одну, усуваючи проміжні звернення до пам'яті. Найпоширеніший приклад: злиття послідовності «згортка, BN, активація ReLU». Після навчання параметри нормалізації (середнє μ, дисперсія σ^2, масштаб γ, зміщення β) стають фіксованими і можуть бути алгебраїчно поглинуті вагами згортки:

$$\begin{cases} W_{зл} = \gamma \cdot \frac{W}{\sqrt{(\sigma^2 + \varepsilon)}} \\ b_{зл} = \gamma \cdot \frac{(b - \mu)}{\sqrt{(\sigma^2 + \varepsilon) + \beta}} \end{cases}, \quad (2.14)$$

де $W_{зл}$, $b_{зл}$ – ваги та зміщення згортки із вбудованою нормалізацією;

W, b – оригінальні ваги та зміщення згортки;

γ, β – масштабуючий коефіцієнт та зміщення BN;

μ, σ^2 – середнє та дисперсія, обчислені на тренувальних даних;

ε – мала константа для числової стабільності ($\varepsilon = 10^{-5}$).

Після злиття три операції замінюються однією: $y_{зл} = ReLU(W_{зл} * x + b_{зл})$. Злиття не вносить жодної похибки, оскільки обчислює математично ідентичний результат. Прискорення досягається з двох причин: дані читаються з пам'яті один раз замість трьох, а накладні витрати на запуск трьох обчислювальних ядер замінюються запуском одного. Для моделі DBNet з 24 згортковими шарами злиття усуває 48 окремих операцій (24 нормалізації та 24 активації), скорочуючи кількість операцій у графі з 72 до 24.

Три методи діють мультиплікативно. Обрізка зменшує кількість операцій на 30-50 %. Квантизація прискорює кожну з решти операцій у 2-4 рази. Злиття скорочує накладні витрати доступу до пам'яті. Комбінація забезпечує прискорення у 5-10 разів при мінімальній втраті точності (не більше 0.5 % за F1), що дозволяє дотримати обмеження реального часу задачі оптимізації (2.12).

2.4 Висновки до другого розділу

У другому розділі побудовано модель інтелектуальної системи розпізнавання графічних образів, що складається з трьох взаємопов'язаних компонентів.

У розділі побудовано модель інтелектуальної системи розпізнавання графічних образів, що складається з трьох взаємопов'язаних компонентів.

Запропоновано формалізацію конвеєра розпізнавання як спрямованого ациклічного графа з п'ятьма вузлами обробки. Такий підхід дозволяє чітко визначити потоки даних між етапами та забезпечує незалежну заміну окремих компонентів без перебудови всього конвеєра. Для модуля детекції тексту обрано архітектуру з диференційованою бінаризацією, яка адаптується до нерівномірного освітлення через генерацію поелементної карти порогів. Для модуля розпізнавання символів обрано трансформерну архітектуру з механізмом локальної та глобальної уваги, що забезпечує як морфологічний аналіз окремих літер, так і врахування лінгвістичного контексту рядка.

Побудовано математичну модель гетерогенного розподілу обчислень між трьома типами процесорів мобільного SoC. Визначено матрицю сумісності вузлів конвеєра з обчислювальними ресурсами та сформульовано задачу оптимізації, що мінімізує загальний час обробки при обмеженнях на пам'ять та якість розпізнавання. Механізм каскадного переключення забезпечує адаптацію системи до різних апаратних платформ без модифікації логіки конвеєра.

Описано три методи оптимізації нейромережевих моделей для мобільного розгортання: квантизацію ваг, структурну обрізку та злиття операторів. Комбінація цих методів забезпечує зменшення обсягу моделей у 4 рази та прискорення обчислень у 5-10 разів при мінімальній втраті якості.

Побудована модель є теоретичним підґрунтям для розробки методу гетерогенного розподілу обчислювальних задач, описаного у третьому розділі.

3 МЕТОД РОЗПІЗНАВАННЯ ГРАФІЧНИХ ОБРАЗІВ З ВИКОРИСТАННЯМ ГЕТЕРОГЕННИХ ОБЧИСЛЕНЬ

3.1 Основи методу розпізнавання графічних образів з використанням гетерогенних обчислень

Для досягнення поставленої мети було розроблено метод розпізнавання графічних образів з використанням гетерогенних обчислень. У другому розділі побудовано формальну модель конвеєра розпізнавання як спрямованого ациклічного графа $G = (V, E)$ та модель гетерогенного розподілу обчислень з функцією призначення $A: V -> R$. Метод, що пропонується у цьому розділі, переводить побудовану модель у площину практичної реалізації, визначаючи алгоритмічний апарат кожного модуля та правила розподілу обчислень між процесорами мобільного SoC.

Метод побудовано як послідовність шести модулів обробки: попередня обробка та сегментація зображення, геометрична ректифікація, детекція текстових областей, розпізнавання символів, гетерогенний розподіл обчислювальних задач та оптимізація моделей для мобільного виконання.

Кожний модуль призначається на процесор з найменшим часом відгуку для даного типу обчислень. Сегментація та ректифікація працюють з щільними піксельними масивами, де паралелізм на рівні пікселів забезпечує максимальну ефективність ГП. Детекція та розпізнавання складаються із згорткових та трансформерних операцій, що найкраще виконуються на НП. Управління конвеєром та семантичний розбір виконуються на ЦП. При недоступності основного обчислювача реалізується каскадне переключення на резервний процесор без модифікації логіки конвеєра.

Метою методу є розпізнавання фіскальних документів у реальному часі при обмеженнях $T_{заг} <= 2$ с, $Q >= 0.95$ (F1), $Mem_{заг} <= 15$ МБ.

Формально метод записується як композиція функцій модулів:

$$S = I \cdot f_{сегм} \cdot f_{ректи} \cdot f_{дет} \cdot f_{розп} \cdot f_{парс}, \quad (3.1)$$

де S – структурований результат розпізнавання (визначення 2.3);

I – вхідне зображення з камери мобільного пристрою;

$f_{сегм}$ – функція модуля сегментації;

$f_{ректм}$ – функція модуля ректифікації;

$f_{дет}$ – функція модуля детекції текстових областей;

$f_{розп}$ – функція модуля розпізнавання символів;

$f_{парс}$ – функція модуля семантичного розбору.

Кожна функція f_i виконується на обчислювачі за функцією призначення A (визначення 2.10), загальний час обчислюється за формулою (2.11). При помилці або перевищенні часового ліміту модуль перезапускається на наступному обчислювачі за механізмом каскадного переключення: для нейромережевих модулів НП, ГП, ЦП; для геометричних ГП, ЦП.

Послідовність обробки визначається структурою конвеєра. Вхідний кадр з камери надходить до модуля сегментації, який виділяє область документа з фону та визначає чотири кутових точки. На основі цих точок модуль ректифікації виконує перспективне перетворення, формуючи вирівняне прямокутне зображення документа. Далі модуль детекції знаходить координати кожного рядка тексту на вирівняному зображенні та повертає множину обмежувальних полігонів. Кожен полігон вирізається та передається модулю розпізнавання, який перетворює піксельне зображення фрагмента на послідовність символів. На завершення модуль семантичного розбору агрегує всі розпізнані рядки та формує структурований результат.

Паралельно з основним конвеєром функціонує модуль розподілу обчислювальних задач. Він виконує ініціалізацію при першому запуску (перевірка доступних обчислювачів, компіляція графів моделей, кешування скомпільованих версій), а під час обробки кожного кадру контролює час виконання модулів та реалізує альтернативний маршрут виконання при необхідності. Цей модуль не належить до основного потоку даних, а виконує управлінську функцію.

Загальну схему методу показано на рисунку 3.1.

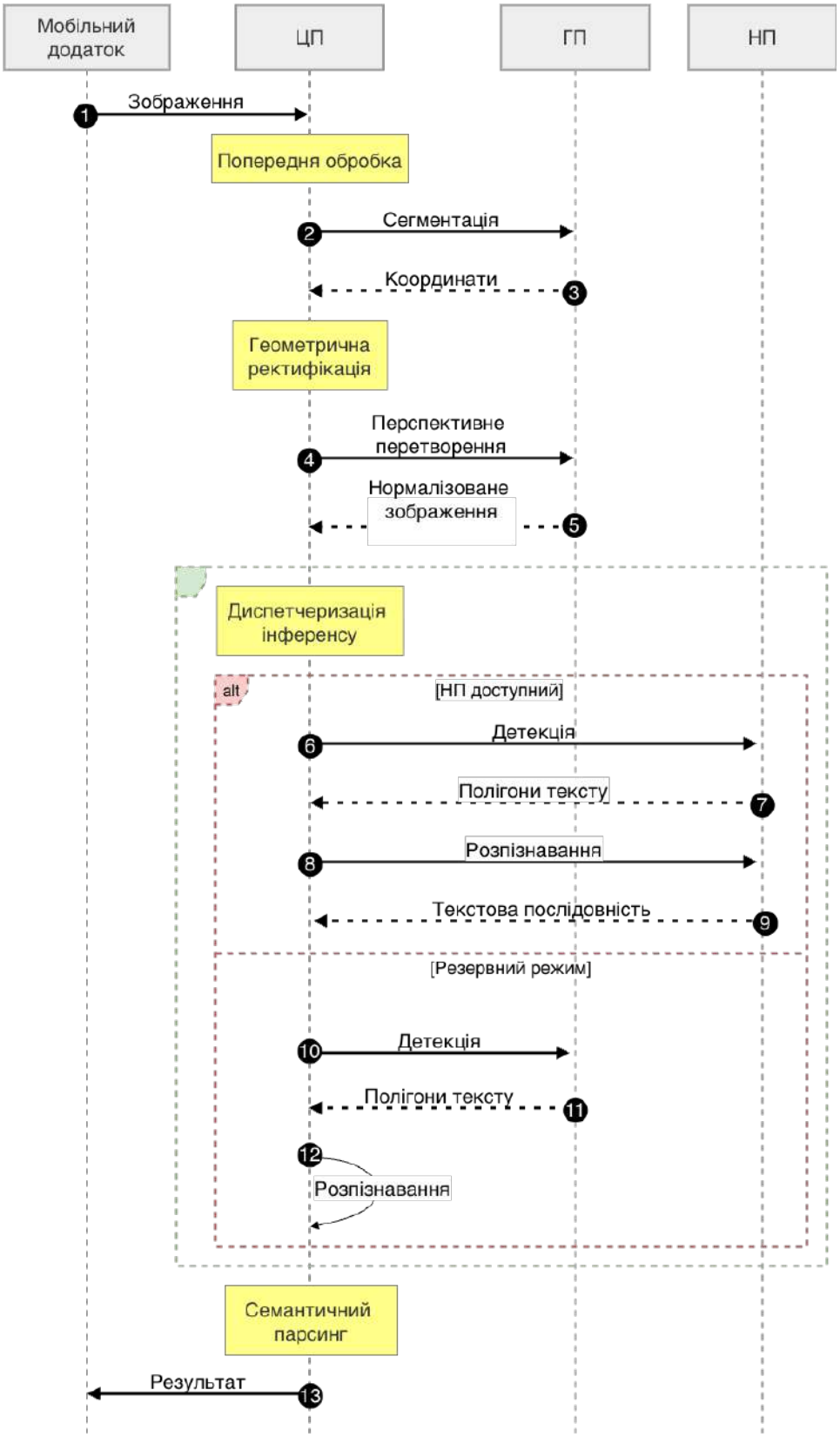


Рисунок 3.1 – Загальна схема методу розпізнавання графічних образів з використанням гетерогенних обчислень

На рисунку 3.1 суцільні стрілки позначають потоки даних, штрихові позначають управлінські зв'язки модуля розподілу задач (вибір обчислювача для модулів 1, 3, 4). Управлінський модуль працює паралельно з основним конвеєром: він визначає, на якому обчислювачі запускати кожний модуль, та реагує на збої переключенням на резервний обчислювач.

3.2 Модуль попередньої обробки та сегментації зображення

Обробка зображення починається з масштабування вхідного кадру до роздільності 320 x 256 пікселів. Такий розмір обрано як компроміс між точністю та швидкістю: при роздільності 128 x 128 мережа втрачає тонкі контури документа на складному фоні, тоді як при 512 x 512 час інференсу зростає вчетверо при покращенні IoU менш ніж на 5 %. Для масштабування застосовується білінійна інтерполяція, обрана замість бікубічної через вдвічі менший час обчислення при різниці IoU менш ніж 2 %. Білінійна інтерполяція використовує 4 сусідні пікселі для обчислення кожного вихідного значення, тоді як бікубічна використовує 16. На мобільному ГП масштабування потребує менше 1 мс.

Після масштабування виконується нормалізація пікселів за статистиками набору даних ImageNet. Нормалізація необхідна, оскільки кодувальник MobileNetV3-Small навчено саме на цьому наборі даних, і мережа очікує відповідний розподіл вхідних значень. Без нормалізації IoU знижується на 5-8 %. Перетворення для кожного пікселя має вигляд:

$$I_{\text{н}}(x, y, c) = \frac{(\frac{I_S(x,y,c)}{255 - \mu_c})}{\sigma_c}, \tag{3.2}$$

де $I_{\text{н}}$ – нормалізоване зображення;

I_S – масштабоване зображення;

c – індекс кольорового каналу (c = 1, 2, 3 для R, G, B);

μ_c – середнє значення каналу c у вибірці ImageNet ($\mu = (0.485, 0.456, 0.406)$);

σ_c – стандартне відхилення каналу c ($\sigma = (0.229, 0.224, 0.225)$).

Ядром модуля є сегментаційна нейронна мережа з архітектурою кодувальник-декодувальник та пропускними з'єднаннями. Кодувальником слугує MobileNetV3-Small, декодувальником слугує U-Net. Загальна кількість параметрів мережі становить приблизно 150 тисяч, що на два порядки менше за DeepLabV3 (5.8 млн) та U-Net/ResNet50 (32.5 млн). Час інференсу на мобільному ГП становить 8-12 мс. Архітектура MobileNetV3-Small побудована на інвертованих залишкових блоках з поглиблювально-розділеними згортками (DSC), активацією h-swish та squeeze-and-excitation блоками для адаптивного перезважування каналів.

Кодувальник витягує ознаки на чотирьох рівнях деталізації: крок 2 (128 x 128 пікселів), крок 4 (64 x 64), крок 8 (32 x 32) та крок 16 (16 x 16). Рівень з кроком 2 зберігає дрібні деталі контуру, тоді як рівень з кроком 16 кодує семантичну інформацію про належність об'єкта до класу документа. Кожен рівень побудовано на інвертованих залишкових блоках: вхідний тензор розширюється точковою згорткою 1 x 1 (з коефіцієнтом 4-6), обробляється глибинною згорткою 3 x 3 або 5 x 5 для кожного каналу незалежно та стискається назад точковою згорткою 1 x 1. DSC зменшує кількість операцій приблизно у K^2 разів (для ядра 3 x 3 приблизно у 9 разів).

Вихідним виразом кодувальника є набір карт ознак на кожному рівні деталізації:

$$E = \{E_1,\ E_2,\ E_3,\ E_4\},$$
$$E_k = \frac{\text{H}}{2^k} \cdot \frac{W}{2^k} \cdot c_k, \tag{3.3}$$

де E_k – карта ознак на k-му рівні кодувальника;

H, W – висота та ширину вхідного зображення (320 x 256)

c – кількість каналів на k-му рівні.

Декодувальник відновлює роздільність через чотири послідовних блоки. Кожний блок збільшує роздільність удвічі, об’єднує результат з пропускним з'єднанням від відповідного рівня кодувальника та застосовує згортку 3 х 3 з груповою нормалізацією та ReLU. Пропускні з'єднання передають просторову інформацію з ранніх шарів, що дозволяє точніше відновлювати контури: без них контур розмивається на 3-5 пікселів, що при масштабуванні до оригінальної роздільності дає похибку у 20-40 пікселів. Вихідний шар: згортка 1 х 1 з сигмоїдою, маска M розміром 320 х 256 х 1.

Маска обчислюється як:

$$M(x, y) = \sigma(W_{out} \cdot D_1(x, y) + b_{out}), \quad (3.4)$$

де $M(x,y)$ – значення маски у позиції (x, y), що належить діапазону [0, 1];

σ – функція сигмоїди;

W_{out} – ваги фінальної згортки 1 х 1;

b_{out} – зміщення фінальної згортки;

D_1 – вихід останнього блоку декодувальника.

У цій архітектурі обрано групову нормалізацію (GN) замість пакетної нормалізації (BN) через проблему розбіжності між режимами навчання та інференсу. BN обчислює статистики по пакету під час навчання та використовує ковзне середнє під час інференсу. При розмірі пакету 1 (штатний режим мобільного інференсу) ці статистики можуть не відповідати розподілу реальних зображень. GN обчислює статистики по групах каналів всередині одного зображення незалежно від розміру пакету, що усуває цю розбіжність.

Після отримання маски виконується пост-обробка для витягування кутових точок документа. Маска бінаризується порогом 0.5, серед зв'язних компонентів обирається найбільший за площею (документ), компоненти з площею менше 1 % відкидаються як шум. Контур найбільшого компонента апроксимується чотирикутником за алгоритмом Дугласа-Пекера з допуском 2 % від периметра.

Допуск 2 % обрано емпірично: при 1 % апроксимація повертає 5-6 точок через заокруглені кути чеків, при 3 % втрачаються вигини зім'ятих документів. Координати перераховуються до оригінального розміру H x W множенням на коефіцієнти ($\frac{H}{256}$, $\frac{W}{256}$). Пост-обробка виконується на ЦП та потребує менше 0.1 мс.

3.3 Модуль геометричної ректифікації

Модуль усуває перспективні спотворення, що виникають через довільне положення камери відносно площини документа. Типовий кут нахилу камери при ручній зйомці становить 10-30 градусів, що спотворює пропорції літер. Без ректифікації точність розпізнавання падає на 15-25 %. Вхідними даними є оригінальне зображення I (H x W) та чотири кутові точки P_4 від модуля сегментації, результатом є вирівняне прямокутне зображення $I_{ректт}$. Перспективне перетворення описується матрицею гомографії H розміром 3 x 3:

$$s \cdot \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \cdot \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3.5}$$

де H – матриця гомографії розміром 3 x 3;

s – масштабний коефіцієнт (ненульовий);

$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$ – однорідні координати цільової точки;

$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$ – однорідні координати вхідної точки;

Цільові точки задаються як кути прямокутника W_r x H_r. Чотири пари точок дають 8 рівнянь для 8 невідомих (h_{33} = 1). Система розв'язується методом LU-розкладання, що потребує менше 1000 операцій. Для застосування перетворення використовується обернена трансформація з білінійною інтерполяцією:

$$I_{\text{ректр}}(x, y) = \text{Interp}(I, H^{-1} \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}) \quad (3.6)$$

$I_{ректр}$ – значення пікселя у вирівняному зображенні;

Interp – білінійна інтерполяція;

H^{-1} – обернена матриця гомографії.

Обернене перетворення гарантує заповнення кожного пікселя вихідного зображення, на відміну від прямого, яке може залишити незаповнені позиції. На ГП обернене перетворення виконується паралельно за 1-2 мс. Для фіскальних чеків типове вихідне зображення має розмір приблизно 960 х 640 пікселів (ширина чека 80 мм при роздільності 12 пікселів на міліметр).

Гомографію обрано замість TPS-сплайнів (Thin Plate Splines), оскільки гомографія потребує лише 9 множень на піксель, тоді як TPS-сплайни потребують N множень, де N залежить від кількості контрольних точок. TPS-сплайни доцільні для зім'ятих документів з нелінійними деформаціями, проте фіскальні чеки зазвичай плоскі, і гомографія забезпечує достатню точність. Для чеків з незначною кривизною залишкова деформація компенсується на рівні детекції тексту, де полігональна апроксимація контурів адаптується до локальних викривлень рядків. Передача даних між модулями сегментації та ректифікації відбувається через спільну пам'ять ГП без копіювання, що забезпечує мінімальну латентність переходу між модулями.

3.4 Модуль детекції текстових областей

Модуль знаходить координати рядків тексту на вирівняному зображенні $I_{ректр}$ (960 х 640 х 3) та повертає упорядковану множину полігонів. Для типового чека модуль генерує 20-50 полігонів. Архітектуру побудовано за схемою DBNetLite з трьох компонентів: базової мережі MobileNetV3-Small, піраміди ознак RSEFPN та голови бінаризації DBHead.

Базова мережа витягує карти ознак на трьох рівнях: крок 4 (240 x 160), крок 8 (120 x 80) та крок 16 (60 x 40). Рівень з кроком 2 не використовується, оскільки він подвоює обчислювальний обсяг піраміди, а для рядків висотою 12-18 пікселів роздільність кроку 4 достатня. Три рівні обрано через обмежений діапазон висот: крок 4 локалізує тонкі рядки, крок 8 захоплює контекст одного рядка, крок 16 дає семантичний огляд блоків тексту.

Піраміда ознак RSEFPN об'єднує неглибокі рівні (просторова точність) з глибокими (семантика). Латеральне перетворення приводить рівні до ch = 96 каналів згорткою 1 x 1. Відмінність RSEFPN від стандартної FPN полягає в інтеграції блоків Squeeze-and-Excitation на кожному рівні. Блок SE адаптивно зважує канали: глобальне усереднення стискає карту ознак до вектора ch, два повнозв'язних шари обчислюють ваги:

$$SE(F) = F * \sigma(W_2 * \mathrm{ReLU}(W_1 * \mathrm{AvgPool}(F))), \quad (3.7)$$

де F – вхідна карта ознак;

AvgPool – функція глобального усереднення по просторових осях;

W_1 – ваги першого повнозв'язного шару (ch $\rightarrow \frac{ch}{4}$);

W_2 – ваги другого повнозв'язного шару ($\frac{ch}{4} \rightarrow$ ch);

σ – функція сигмоїди;

Механізм SE підсилює інформативні канали та пригнічує шумові, що на чеках із зернистою текстурою термопаперу покращує виділення тексту від фону. Низхідне злиття передає семантику від глибоких рівнів до неглибоких: карта глибшого рівня збільшується до роздільності попереднього та додається поелементно. Після агрегації рівні приводяться до роздільності з кроком 4, конкатенуються та зливаються згорткою 3 x 3, формуючи карту F (240 x 160 x ch).

Злиття карт ознак різних рівнів у єдине представлення описується як:

$$F = \text{Conv3x3}(\text{Concat}(P_2', \text{Up2x}(P_3'), \text{Up4x}(P_4'))), \quad (3.8)$$

де P_2', P_3' – карти ознак після SE-калібрування та низхідного злиття;

P_4' – карта ознак найглибшого рівня після SE-калібрування;

Up2x, Up4x – збільшення роздільності у 2 та 4 рази відповідно;

Concat – функція об'єднання вздовж осі каналів;

Concat3x3 – функція згортки 3 х 3 для зменшення кількості каналів.

Голова DBHead приймає карту F та паралельно генерує дві карти: карту ймовірностей P та карту адаптивних порогів T. Обидві гілки мають ідентичну структуру: згортка 3 х 3, пакетна нормалізація, ReLU, згортка 1 х 1 з сигмоїдою. На НП та ГП обидві гілки виконуються одночасно, не збільшуючи латентність. Диференційована бінаризація замінює жорстке порівняння з глобальним порогом гладким наближенням через сигмоїду з великим коефіцієнтом підсилення:

$$B(x, y) = \sigma(k * (P(x, y) - T(x, y))) \quad (3.9)$$

де $k = 50$ – коефіцієнт підсилення;

P(x,y) – ймовірність тексту у пікселі (х, у);

T(x,y) – адаптивний поріг для пікселя (х, у);

σ – функція сигмоїди.

Коефіцієнт k = 50 забезпечує крутий перехід сигмоїди, наближаючи її до ступінчастої функції, але зберігаючи неперервну похідну для навчання градієнтними методами. Кожен піксель має власний поріг, що дозволяє детектувати текст при нерівномірному освітленні. Адаптивний поріг T підлаштовується до локального контрасту: у зонах з вицвілим текстом знижується, у зонах з високим контрастом зростає. Під час інференсу використовується лише P з порогом 0.3 (карта T потрібна під час навчання). Поріг 0.3 замість 0.5 обрано для підвищення повноти: при 0.5 вицвілі символи на старих чеках не детектуються.

Архітектуру мережі DBNetLite показано на рисунку 3.2.

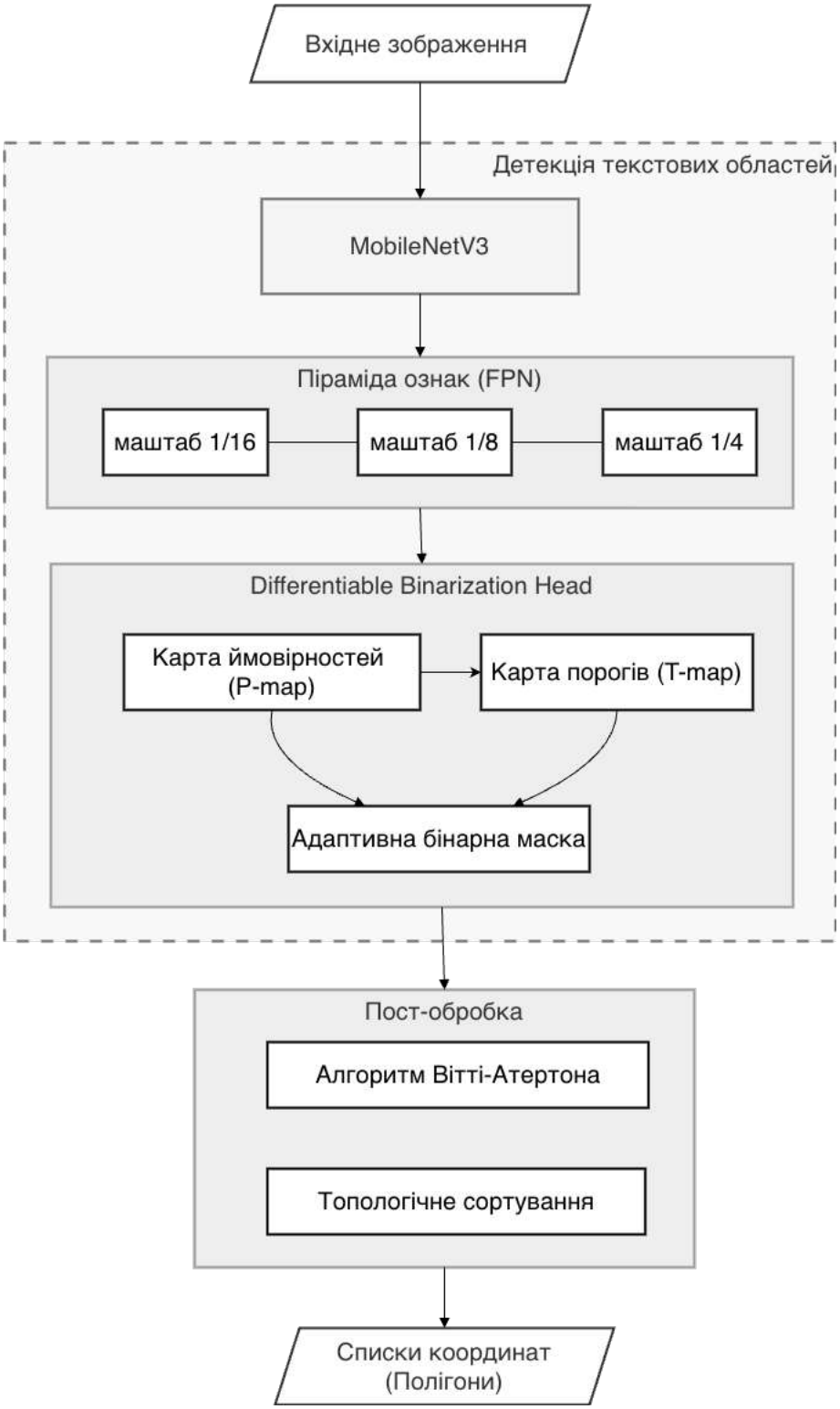


Рисунок 3.2 – Діаграма архітектури DBNet

Результатом роботи детектора є множина обмежувальних полігонів, отриманих з бінарної маски. Маска фільтрується для усунення шумових активацій,

виконується пошук зв'язних компонентів, для кожного обчислюється полігональна апроксимація контуру.

Полігони рядків розширюються з коефіцієнтом r = 1.5, що є критичним для коректного розпізнавання українського тексту. Українська абетка містить літери з надрядковими діакритичними елементами: крапка над "і", дві крапки над "ї", бреве над "й", хвостик літери "г". На термопаперу ці елементи друкуються з роздільністю 180-203 dpi, що дає висоту діакритичного знака 1-2 пікселі у зображенні. Без розширення полігонів крапки над "і" та "ї" та бреве над "й" відрізаються межами полігону, спричиняючи хибне розпізнавання: "і" перетворюється на "l" або "1", "ї" на "і", "й" на "и". Коефіцієнт r = 1.5 обрано емпірично за вибіркою 500 чеків: при r = 1.2 діакритичні знаки захоплюються у 87 % випадків, при r = 1.5 у 99.2 %, при r = 2.0 полігони починають перекривати сусідні рядки.

Після розширення полігони впорядковуються у природному порядку читання: зверху вниз за вертикальною координатою центроїда, а для полігонів на одній горизонтальній лінії зліва направо. Два блоки вважаються розташованими на одному рядку, якщо різниця їхніх вертикальних координат менша за половину середньої висоти блоку. Впорядкування необхідне для модуля семантичного розбору, який очікує рядки у послідовності, що відповідає логічній структурі документа.

Обчислювач модуля: НП як пріоритетний, ГП як резервний. Пост-обробка (пошук контурів, розширення полігонів, впорядкування) виконується на ЦП та потребує менше 1 мс. Нейромережева частина потребує 0.3 GMAC та виконується за 15-25 мс на НП. На ГП час зростає до 40-60 мс, що все ще вкладається в обмеження загального часу обробки конвеєра.

3.5 Модуль розпізнавання символів

Модуль перетворює зображення текстових рядків $\{I_1, ..., I_n\}$ на послідовності символів $\{s_1, ..., s_n\}$. Архітектуру побудовано на основі SVTR, яка поєднує згорткові

та трансформерні шари. На відміну від CRNN/BiLSTM, SVTR аналізує всі позиції паралельно через механізм уваги, що забезпечує прискорення у 3-5 разів на НП.

Перед розпізнаванням фрагменти сортуються за шириною для мінімізації нульового заповнення у пакетах, висота нормалізується до H_f = 48 пікселів зі збереженням пропорцій. Висоту 48 обрано для українського тексту: при 32 пікселях діакритичні знаки над "і", "ї", "й" стискаються до 1 пікселя. Сортування за шириною зменшує втрати на нульове заповнення з 40-60 % до 5-10 %.

Зображення розбивається на ділянки розміром p_h x p_w (типово 4 x 8 пікселів), кожна з яких перетворюється на вектор розмірності d через згорткові шари. Ділянка 4 x 8 охоплює приблизно одну літеру шрифту 10-12 пунктів при роздільності друку чека. Послідовність $Z = \{z_1, z_2, ..., z_L\}$, де L визначається шириною зображення, подається на блоки уваги.

Архітектура SVTR застосовує механізм уваги у двох режимах. Локальна увага обмежує область взаємодії до сусідніх позицій у фіксованому вікні w (типово w = 7 ділянок) та обчислюється як:

$$Attention(Q, K, V) = softmax(Q * \frac{K^T}{\sqrt{d^k}} + Mask_w) * V, \quad (3.10)$$

де $Q = X * W_q$ – матриця запитів;

$K = X * W_k$ – матриця ключів;

$V = X * W_v$ – матриця значень;

d_k – розмірність ключа;

$Mask_k$ – маска обмеження вікна, де елементи за межами вікна $w = -\infty$.

Локальна увага фокусується на морфологічних деталях літер: формі штрихів, положенні діакритичних знаків. Для української мови це критично, оскільки забезпечує відрізнення "и" від "й", "і" від "ї", "г" від "ґ", "е" від "є". Розмір вікна w = 7 ділянок обрано з урахуванням морфології: вікно охоплює 56 пікселів (7 * 8), що відповідає 3-4 літерам. При w = 3 мережа втрачає контекст для розрізнення "ї" від

"і", при w = 11 зростає складність без покращення точності. Складність $O(L * w * d_k)$ є лінійною за L.

Глобальна увага працює без маски, дозволяючи кожній ділянці взаємодіяти з усіма іншими та захоплювати лінгвістичний контекст рядка. Послідовність "МОЛ_КО" однозначно вказує на "О", навіть якщо морфологічно символ подібний до "0". Ієрархічне поєднання реалізується через послідовні стадії SVTR: на перших переважає локальна увага (морфологія), на глибших глобальна (контекст). Для рядків чеків довжина L рідко перевищує 50-60 ділянок, що робить квадратичну складність прийнятною.

Для перетворення послідовності представлень на символи застосовується CTC-декодування, обране замість авторегресійного декодування через паралельність обчислень та прискорення у 5-8 разів на НП. Алфавіт А містить 33 кириличні літери (обидва регістри), 26 латинських, 10 цифр, знаки пунктуації та спеціальні символи, загалом приблизно 170 символів. CTC вводить порожній символ epsilon та обчислює ймовірність цільової послідовності як суму ймовірностей усіх можливих вирівнювань, що зводяться до цієї послідовності після видалення порожніх символів та послідовних дублікатів (визначення 2.8).

На етапі інференсу жадібний алгоритм обирає символ із максимальною ймовірністю для кожної позиції, після чого дублікати та порожні символи видаляються. При арифметичних невідповідностях активується променевий пошук з B = 5 гіпотезами для підозрілих рядків (2-5 із загального набору). Наприклад, якщо рядок "2 x 14.50 = 29.00" розпізнано як "2 x 14.50 = 29.0O" (літера "О" замість нуля), перевірка виявляє помилку, і серед гіпотез знаходиться правильний варіант.

Після декодування розпізнана послідовність проходить лексичну пост-обробку з використанням N-грамних статистик. Модель біграм оцінює ймовірність кожної пари послідовних символів:

$$P_{\text{лекс}}(s) = \prod_{i=2}^{|s|} P(s_i \mid s_{i-1}) \tag{3.11}$$

де $s = s_1 s_2 \ldots s_n$ – позначає рядок символів;

$P(s_i \mid s_{i-1})$ – позначає умовну ймовірність символу s_i після символу s_{i-1}.

Пост-обробка коригує плутання подібних символів ("О"/"0", "З"/"3", "І"/"1", "В"/"8"). Якщо заміна підвищує ймовірність більш ніж у 10 разів, символ замінюється ("М0Л0К0" виправляється на "МОЛОКО"). Таблиця біграм навчена на 50 тисячах українськомовних чеків (менше 100 КБ), де розподіл біграм суттєво інший, ніж у загальному корпусі, через скорочення ("МОЛ.", "ХЛІБ Б/Д") та службові слова ("СУМА", "ПДВ"). Пост-обробка розрізняє текстовий та числовий контексти: у числовому "О" замінюється на "0", у текстовому здійснюється зворотна заміна.

Обчислювач: НП (пріоритет), резерв ЦП з ARM NEON. ГП не використовується через неефективність малих матричних множень на шейдерній архітектурі: розмір матриць у SVTR (48-64 на d) менший за поріг ефективного завантаження шейдерних ядер (зазвичай 256 x 256), що робить завантаження та синхронізацію шейдерних ядер дорожчими за самі обчислення. На ЦП з ARM NEON час зростає до 80-120 мс, що залишається в межах допустимого бюджету часу. Типовий чек з 20-30 рядками потребує 0.5 GMAC, час обробки 30-50 мс на НП.

3.6 Модуль гетерогенного розподілу обчислювальних задач

Гетерогенний розподіл обчислювальних задач є центральною науковою новизною запропонованого методу. На відміну від традиційних підходів, де нейромережева модель розгортається на одному обчислювачі, запропонований метод розподіляє підзадачі конвеєра між трьома типами обчислювачів відповідно до характеру обчислень. ГП складається з сотень шейдерних ядер для паралельної обробки великих масивів (65536 пікселів для входу 256 x 256), що робить його оптимальним для сегментації та ректифікації. НП оптимізований для матричних множень: апаратні MAC-масиви виконують згорткові операції у 5-10 разів швидше за ГП, що робить його оптимальним для детекції та розпізнавання. ЦП має

найнижчу пропускну здатність для нейромережевих обчислень, проте гарантовано підтримує довільні операції, включаючи управлінську логіку з розгалуженнями.

Модуль реалізує функцію призначення (визначення 2.10) та каскадне переключення. Робота починається з ініціалізації, під час якої формується матриця сумісності S $(|V| \; x \; |R|)$, де $s_{ij} = 1$, якщо обчислювач r_j підтримує всі оператори модуля vi. Перевірка здійснюється через NNAPI Delegate (Android) або CoreML Delegate (iOS). NNAPI працює як абстрактний шар над драйверами (Qualcomm QNN, MediaTek NeuroPilot, Samsung Eden). CoreML компілює граф у представлення MIL з автоматичним вибором обчислювача. Ініціалізація виконується одноразово (200-500 мс), результати кешуються.

Для кожного модуля обирається обчислювач з найвищим пріоритетом серед сумісних: для нейромережевих НП (1), ГП (2), ЦП (3); для геометричних ГП (1), ЦП (2). Граф обчислень компілюється делегатом та кешується як бінарний файл (300-800 КБ), що зменшує час ініціалізації з 600-1500 мс до 30-60 мс.

Якщо обчислювач підтримує більше 50 % операторів модуля, виконується автоматичне розбиття графа: підтримувані оператори на цільовому обчислювачі, решта на ЦП. Розбиття доцільне при виконанні більше 70 % обчислень на цільовому обчислювачі та не більше 3-4 точках розбиття, кожна з яких додає 0.5-1.5 мс на передачу проміжних тензорів. Для модуля детекції з 45 операторами, де НП підтримує 38, розбиття створює 2 точки переходу при виконанні 92 % обчислень на НП. Жадібний алгоритм обходить граф у топологічному порядку, об'єднуючи підтримувані оператори у підграф для цільового обчислювача та формуючи окремий підграф для ЦП з непідтримуваних операторів.

Каскадне переключення є механізмом забезпечення відмовостійкості конвеєра. Механізм активується при виникненні помилки виконання (драйвер повертає код помилки, обчислювач зависає) або при перевищенні часового ліміту. Умова переключення формалізується як:

$$Fallback(v_i, r_j) = 1, \text{якщо } T_{\text{факт}} > \alpha * T_{\text{очік}}, \text{або } Error = 1, \qquad (3.12)$$

де $T_{\text{факт}}$ – позначає фактичний час виконання модуля v_i на обчислювачі r_j;

$T_{\text{очік}}$ – позначає очікуваний час (ковзне середнє за 10 останніх запусків);

$alpha = 3.0$ – коефіцієнт допустимого перевищення часу;

$Error$ –індикатор помилки виконання (1 при виникненні помилки).

Коефіцієнт alpha = 3.0 обрано як компроміс: при alpha = 1.5 переключення спрацьовує при кожному тепловому троттлінгу процесора, при alpha = 5.0 реальні збої виявляються занадто пізно. Ковзне середнє T_очік обчислюється з експоненціальним згасанням: $T_t = 0.9 * T_{t-1} + 0.1 * T_{\text{факт}}$, що надає більшу вагу останнім вимірюванням та дозволяє адаптуватися до змін продуктивності при нагріві SoC.

Ефективний час обробки конвеєра з урахуванням можливих переключень визначається як:

$$T_{\text{еф}} = \sum_{i=1}^{|V|} \left(T_i + \delta_i \cdot T_{fi}\right) \tag{3.13}$$

де T_i – позначає час виконання модуля v_i на призначеному обчислювачі;

δ_i – позначає індикатор переключення (1, якщо відбулося каскадне переключення, 0 інакше);

T_{fi} – позначає додатковий час від невдалої спроби та перезапуску на резервному обчислювачі.

При спрацюванні умови (3.13) обчислення перезапускаються на наступному обчислювачі у ланцюжку НП, ГП, ЦП. На бюджетних пристроях без НП всі нейромережеві модулі виконуються на ЦП з ARM NEON (800-1200 мс). При оновленні драйвера НП модуль автоматично перезапускає детекцію на ГП (40-60 мс замість 15-25 мс). При тепловому зниженню частоти під час сканування 15-20 чеків підряд модуль переключає розпізнавання на ЦП. При конкурентному доступі до НП нейромережеві модулі переносяться на ГП. Накладні витрати на невдалі спроби не перевищують 50 мс, після переключення модуль повертається до пріоритетного обчислювача через 50 запусків.

Послідовність каскадного переключення показано на рисунку 3.3.

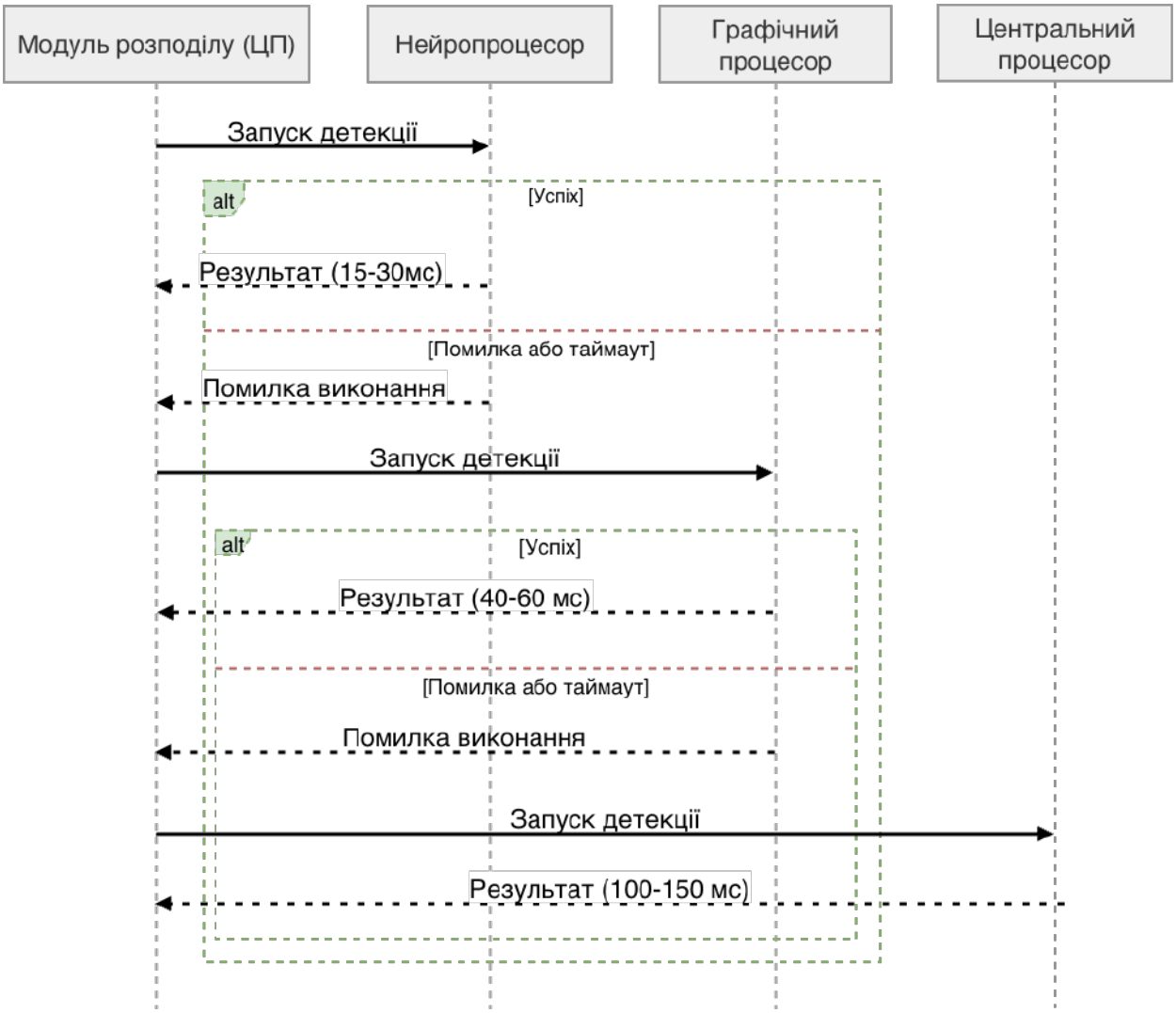


Рисунок 3.3 – Послідовність каскадного переключення обчислювачів для модуля детекції тексту

Для уникнення копіювання тензорів між обчислювачами використовуються платформні механізми спільної пам'яті. На Android DMA-BUF виділяє буфер у фізичній пам'яті, до якого ЦП, ГП та НП мають прямий доступ. На iOS MTLSharedEvent синхронізує доступ до спільного MTLBuffer між обчислювачами. Без передачі вказівника передача тензора 960 x 640 x 3 (1.8 МБ) потребує 4-8 мс, з DMA-BUF час зменшується до менш ніж 0.05 мс. Для конвеєра з 4-5 передачами це зберігає 16-40 мс.

3.7 Модуль оптимізації моделей для мобільного виконання

Модуль виконується офлайн перед розгортанням моделей на мобільному пристрої. Вихідні розміри моделей у форматі FP32 становлять: сегментаційна модель 0.6 МБ, детекційна 3.2 МБ, модель розпізнавання 8.5 МБ (загалом 12.3 МБ). Після оптимізації загальний розмір скорочується у 3-4 рази до 3-4 МБ, що вкладається в обмеження $Mem_{\text{заг}}$ <= 15 МБ з урахуванням додаткової пам'яті на буфери, таблиці біграм та службові структури.

Першим етапом є QAT. Ваги перетворюються з FP32 до INT8 за формулою (2.13). На відміну від PTQ, QAT моделює квантизацію під час навчання: у прямому проході ваги квантизуються та деквантизуються, у зворотному градієнти проходять без змін завдяки Straight-Through Estimator (STE). STE замінює невизначений градієнт round() на одиницю, що дозволяє мережі адаптувати ваги до дискретних рівнів. Квантизація зменшує обсяг пам'яті у 4 рази та прискорює інференс на НП у 2-4 рази завдяки апаратній підтримці INT8-арифметики. QAT забезпечує зниження якості не більше 0.3 % за F1 (PTQ дає втрату до 1.5 %), оскільки мережа навчається компенсувати квантизаційний шум: критичні ваги зміщуються до центрів квантизаційних інтервалів.

Другим етапом є структурне обрізання фільтрів. Фільтри з найменшою нормою L1 видаляються, оскільки вони створюють слабкі активації та мінімально впливають на вихід шару. Порогове значення визначається p-м перцентилем, де p = 30..50 залежно від глибини шару. Шари, близькі до виходу мережі, є більш чутливими до обрізання (p = 30 %), тоді як ранні шари допускають більшу обрізку (p = 50 %) через надлишковість представлення на початкових рівнях. Після обрізання мережа дотренується протягом 5-10 епох для відновлення точності. Видалення 30-50 % фільтрів зменшує кількість операцій на 40-60 % при зниженні якості не більше 0.5 % за F1.

Третім етапом є злиття операторів. Послідовність "згортка, нормалізація, активація" замінюється єдиним оператором, де параметри нормалізації алгебраїчно

поглинаються вагами згортки за формулою (2.14). Після злиття три операції замінюються однією: $y_{зл} = ReLU(W_{зл} * x + b_{зл})$. Злиття не змінює числових результатів та дає прискорення 15-25 % за рахунок зменшення звертань до пам'яті.

Четвертим етапом є структурна репараметризація. Три навчальні гілки (згортка 3 х 3, згортка 1 х 1 та пропускне з'єднання) зводяться до однієї згортки 3 х 3. Для цього згортка 1 х 1 доповнюється нулями до розміру 3 х 3, пропускне з'єднання перетворюється на одиничну згортку, а ваги всіх трьох гілок додаються поелементно:

$$W_{\text{реп}} = W_{3x3} + Pad(W_{1x1}) + I_{conv}, \quad (3.12)$$

де $W_{\text{реп}}$ – позначає результуючі ваги репараметризованої згортки;

W_{3x3} – позначає ваги основної згортки 3 х 3;

$Pad(W_{1x1})$ – позначає ваги згортки 1 х 1, доповнені нулями до розміру 3 х 3;

I_{conv} – позначає одиничну згортку, що відповідає пропускному з'єднанню.

Результуюча згортка виконується у 2-3 рази швидше за три окремі операції.

На завершення модель експортується у TFLite (Android) та CoreML (iOS) з оптимізацією графа, загальний обсяг не перевищує 10 МБ. Результати інференсу верифікуються на тестовій вибірці з 100 зображень, максимальне відхилення від еталонної PyTorch моделі не повинно перевищувати 0.01.

Усі методи оптимізації діють мультиплікативно: обрізка зменшує операції на 30-50 %, квантизація прискорює решту у 2-4 рази, злиття скорочує накладні витрати пам'яті, репараметризація усуває розгалуження. Комбінація забезпечує прискорення у 5-10 разів при втраті не більше 0.5 % за F1. Послідовність застосування має значення: спочатку виконується навчання з QAT, потім структурне обрізання з дотренуванням, далі злиття операторів та репараметризація (обидва без зміни числових результатів), і на завершення експорт у цільовий формат.

3.8 Висновки до третього розділу

У розділі розроблено метод розпізнавання графічних образів фіскальних документів з використанням гетерогенних обчислень як послідовну композицію шести модулів обробки (формула 3.1). Кожен модуль призначається на обчислювач з найменшою латентністю для відповідного типу обчислень.

Модуль сегментації виконується на ГП та використовує мережу з архітектурою MobileNetV3-Small як кодувальник та U-Net як декодувальник, загальна кількість параметрів якої становить 150 тисяч. Кодувальник витягує ознаки на чотирьох рівнях з кроками 2, 4, 8, 16, а декодувальник відновлює роздільність через чотири блоки з UpSampling2D, пропускними з'єднаннями та групової нормалізації. Вибір GroupNorm замість BatchNorm усуває проблему розбіжності статистик між режимами навчання та інференсу при розмірі пакету 1.

Модуль ректифікації виконується на ГП та усуває перспективні спотворення через обчислення матриці гомографії за чотирма парами відповідних точок та обернене перетворення з білінійною інтерполяцією. Гомографію обрано замість TPS-сплайнів через суттєво меншу обчислювальну складність при достатній точності для плоских документів.

Модуль детекції виконується на НП та використовує архітектуру DBNetLite з диференційованою бінаризацією, пірамідою RSEFPN з блоками SE для перезважування каналів та розширенням полігонів з коефіцієнтом r = 1.5 для захоплення діакритичних знаків українських літер.

Модуль розпізнавання виконується на НП та використовує архітектуру SVTR з чергуванням локальної уваги (для морфології символів) та глобальної уваги (для лінгвістичного контексту), CTC-декодування з променевим пошуком для арифметичної верифікації та біграмну пост-обробку для виправлення замін між візуально подібними кириличними та латинськими символами.

Модуль гетерогенного розподілу реалізує каскадне переключення обчислювачів з чотирма сценаріями відмовостійкості та передачу даних між обчислювачами без копіювання через DMA-BUF (Android) та MTLSharedEvent

(iOS). Модуль забезпечує адаптацію конвеєра до різних мобільних пристроїв від бюджетних до флагманських без модифікації логіки обробки.

Модуль оптимізації моделей забезпечує прискорення у 5-10 разів при мінімальній втраті якості через комбінацію QAT, структурного обрізання фільтрів, злиття операторів та структурної репараметризації. Загальний обсяг моделей після оптимізації становить 3-4 МБ.

Розроблений метод забезпечує обробку фіскальних документів у реальному часі при дотриманні обмежень: загальний час $T_{\text{заг}} <= 2$ с, якість Q >= 0.95 (F1), обсяг пам'яті $Mem_{\text{заг}} <= 15$ МБ. Метод може бути адаптований до нових апаратних платформ шляхом оновлення матриці сумісності без зміни архітектури конвеєра.

4 РЕАЛІЗАЦІЯ ІНТЕЛЕКТУАЛЬНА КОМП'ЮТЕРНА СИСТЕМА РОЗПІЗНАВАННЯ ГРАФІЧНИХ ОБРАЗІВ З ВИКОРИСТАННЯМ ГЕТЕРОГЕННИХ ОБЧИСЛЕННЬ

4.1 Програмна реалізація системи

Розроблена система складається з трьох програмних підсистем: підсистеми збору та підготовки даних, підсистеми детекції та розпізнавання тексту і підсистеми гетерогенного розподілу обчислень. Кожна підсистема реалізує відповідні модулі методу, описаного у розділі 3.

Система орієнтована на роботу на двох мобільних платформах: Android та iOS. Вибір двох платформ обумовлений їхніми принциповими відмінностями в організації доступу до апаратних прискорювачів: Android надає уніфікований програмний інтерфейс NNAPI для різних виробників чіпів (Qualcomm, MediaTek, Samsung), тоді як iOS використовує CoreML з прямим доступом до Apple Neural Engine. Реалізація на обох платформах дозволяє перевірити універсальність запропонованого методу гетерогенного розподілу.

У цьому підрозділі послідовно описано підготовку навчальних даних, програмну реалізацію кожної підсистеми та їхню взаємодію.

4.1.1 Підготовка даних та формування даних для навчання

Навчання нейромережевих моделей сегментації, детекції тексту та розпізнавання символів потребує значних обсягів анотованих даних. Для англомовних документів існують відкриті датасети, зокрема SROIE (626 зображень чеків з анотаціями текстових областей) та ICDAR 2019 (1000 відсканованих квитанцій). Для українськомовних фіскальних документів відкритих датасетів не існує. Жоден з наявних наборів не містить українського алфавіту з його специфічними графемами (і, ї, є, ґ), не відтворює деградації вітчизняного термопаперу та не враховує формати, прийняті в Україні (ІПН, ЄДРПОУ, формат дати ДД.ММ.РРРР). Тому першим етапом реалізації стала побудова власного

датасету, доповненого даними з відкритого набору SROIE та синтетично згенерованими зразками.

Збір реальних даних. Основою датасету стали 260 фотографій реальних українських чеків, зібраних з мобільних пристроїв різних моделей. Зйомка виконувалась у різних умовах освітлення: денне природне світло, штучне освітлення люмінесцентними лампами, змішане освітлення з тінями. Документи мали різний ступінь зношеності: від щойно надрукованих до частково вицвілих. Середня кількість текстових блоків на одному чеку склала приблизно 38, що дало загалом близько 10 000 анотованих текстових блоків.

Анотація реальних даних. Для розмітки зібраних зображень використано платформу Label Studio, що забезпечує веб-інтерфейс для створення полігональних анотацій текстових областей та класифікації ключових полів чека. Інтерфейс розмітки у Label Studio показано на рисунку 4.1.

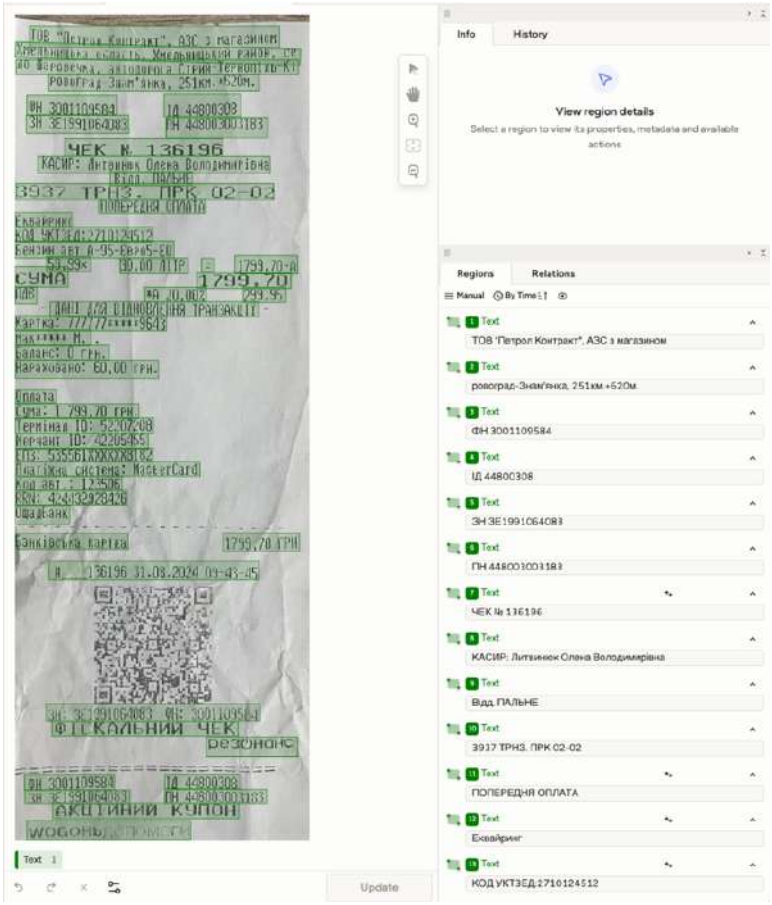


Рисунок 4.1 – Інтерфейс Label Studio для анотації чеків

Кожен текстовий блок анотовано полігоном довільної форми, що дозволяє точно описати межі рядків із нелінійними контурами. Окрім координат полігонів, для кожного блоку збережено текстову транскрипцію та семантичну мітку (назва магазину, товарна позиція, ціна, кількість, дата, ІПН, загальна сума). Розмітка 260 зображень зайняла близько 40 людино-годин.

Аналіз розподілу символів. Після розмітки проведено аналіз частотного розподілу символів у анотованих текстових рядках. Розподіл виявив суттєвий дисбаланс класів: цифри 0-9 та латинські літери зустрічаються у 5-8 разів частіше за специфічні українські графеми ї, є, ґ. Дисбаланс на рівні 1:2000 між найрідшим та найчастішим символами унеможливлює навчання моделі розпізнавання лише на реальних даних. Детальні гістограми частотного розподілу латинських (52 унікальних, 10 223 загалом) та кириличних (64 унікальних, 77 485 загалом) символів наведено у додатку Б.

Набір даних SROIE. Для модулів детекції та розпізнавання додатково залучено відкритий датасет SROIE (Scanned Receipts OCR and Information Extraction), що містить 626 зображень відсканованих чеків. Тексти у SROIE записані латиницею та цифрами, що обмежує його застосування для кириличних символів, проте набір корисний для навчання модуля детекції, де мова тексту не впливає на задачу локалізації рядків. Для модуля розпізнавання SROIE забезпечує додаткове покриття цифрових послідовностей та латинських абревіатур, що зустрічаються і на українських чеках. Навчальна частина SROIE складає 41 925 текстових блоків, валідаційна складає 10 353 блоки.

Генератор синтетичних даних для сегментації. Модуль сегментації (підрозділ 3.2) потребує пар (зображення сцени, бінарна маска документа). Реальних пар з 260 фотографій недостатньо для навчання мережі кодувальник-декодувальник, тому розроблено процедурний генератор сцен з чотирма етапами: формування фонового зображення з бібліотеки текстур (деревина, тканина, камінь, пластик); синтез зображення чека з шаблонів (96 унікальних шаблонів) та корпусу українських товарних назв (12 400 найменувань); застосування перспективних перетворень з

кутами 5-35 градусів та фотометричних аугментацій; формування бінарної маски. Приклади згенерованих зображень показано на рисунку 4.2.

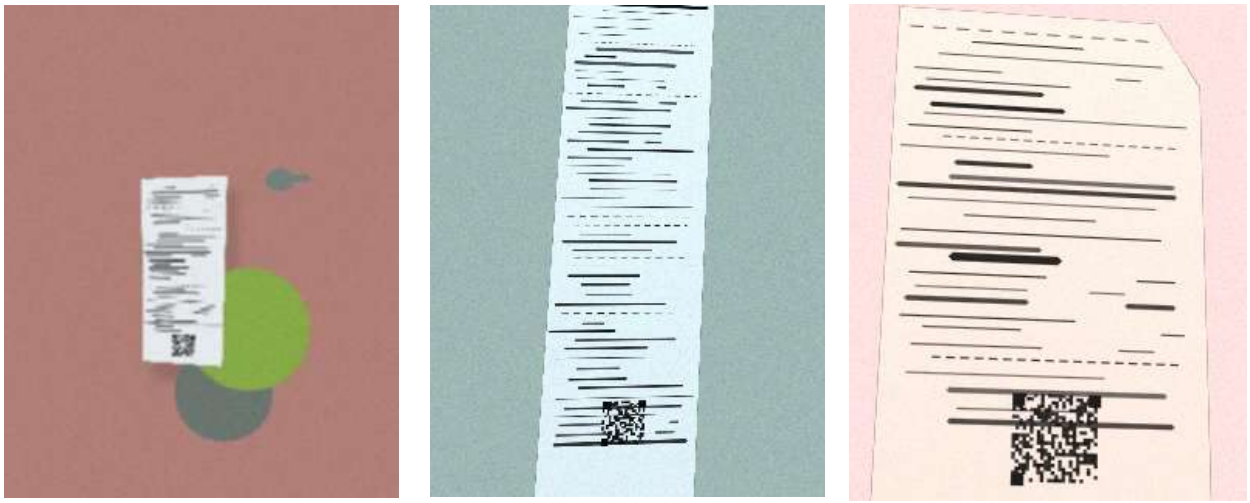


Рисунок 4.2 – Приклади синтетичних чеків для моделі сегментації

Генератор синтетичних даних для детекції тексту. Модуль детекції тексту (підрозділ 3.4) потребує пар (зображення документа, координати текстових рядків). Генератор синтезує зображення чеків з контрольованими деградаціями, імітуючи реальні умови термодруку. Першим компонентом є симуляція зім'ятого паперу за допомогою шуму Перліна:

$$D(x, y) = \sum_{k=0}^{N-1} a^k \cdot \text{Perlin}(x \cdot f \cdot l^k, y \cdot f \cdot l^k) \quad (3.12)$$

де $D(x, y)$ – позначає зміщення пікселя у позиції (x, y);

$a = 0{,}5$ – позначає коефіцієнт загасання амплітуди;

$l = 2$ – позначає коефіцієнт зростання частоти;

f – позначає базову частоту (f = 0.01-0.05);

$N = 4$ – позначає кількість октав;

Perlin – позначає функцію когерентного шуму Перліна.

Отримане поле зміщень деформує зображення чека, створюючи хвилеподібні згини, характерні для зім'ятого термопаперу. Генератор також імітує термічне

вицвітання: інтенсивність пікселів тексту зменшується на 30-70 % у випадкових областях. Окрім того, додаються горизонтальні смуги відсутності фарби, що імітують дефекти друкувальної головки матричного принтера. Другим компонентом є варіативність форматів друку з трьома стилями оформлення: стандартний (вирівнювання по лівому краю, характерне для термопринтерів), вузький (двоколонковий, використовується платіжними терміналами) та широкий (з центруванням заголовків, типовий для супермаркетів). Приклади синтетичних чеків показано на рисунку 4.3.

Рисунок 4.3. Приклади синтетичних чеків для детекції тексту

Генератор синтетичних даних для розпізнавання символів. Модуль розпізнавання (підрозділ 3.5) потребує пар (зображення текстового рядка, рядок символів). Генератор застосовує 16-кроковий конвеєр візуальних деградацій: рендеринг тексту з випадковим шрифтом (48 моноширинних шрифтів POS-принтерів); фоновий градієнт; розмиття (гаусове або руху); шум (гаусовий, сіль-перець, пуасонівський); ерозія та дилатація; зміна контрасту та яскравості; зміна роздільності; обрізання країв; афінні спотворення; видалення фрагментів символів. Для компенсації дисбалансу класів генератор використовує зважену вибірку: рідкісні символи (ї, є, ґ) генеруються з підвищеною частотою (коефіцієнт для ґ приблизно 180, для ї приблизно 12). Ваги розраховуються як відношення частоти найпоширенішого символу до частоти поточного, що забезпечує приблизно

рівномірне покриття всіх 170 символів алфавіту у тренувальній вибірці. Приклади показано на рисунку 4.4.

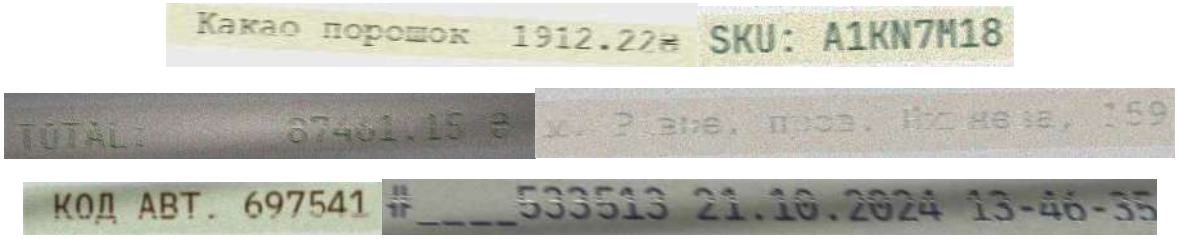


Рисунок 4.4 – Приклади синтетичних зображень для моделі розпізнавання тексту

Зведення навчального датасету. Загальний обсяг підготовленого датасету наведено у таблиці 4.1.

Таблиця 3.1 – Структура та обсяг комбінованого набору даних для розпізнавання тексту

Модуль	Джерело	Навчальна вибірка	Валідаційна вибірка
Сегментація	Реальні чеки	260	0
Сегментація	Синтетичні	3 000	1 240
Детекція тексту	Реальні чеки	8 500	1 500
Детекція тексту	SROIE	41 925	10 353
Детекція тексту	Синтетичні	12 200	1 610
Розпізнавання	Реальні чеки	8 500	1 500
Розпізнавання	SROIE	41 925	10 353
Розпізнавання	Синтетичні	31 990	4 200
Разом		148 300	30 756

Реальні українські чеки (260 зображень, приблизно 10 000 текстових блоків) використано для навчання модулів детекції та розпізнавання. Набір SROIE (52 278 блоків) забезпечує додаткове покриття латинських символів та цифрових послідовностей. Синтетичні генератори створили близько 64 500 зразків для трьох модулів. Валідаційна вибірка виділена з реальних зображень та SROIE, що

дозволяє оцінювати якість моделей на даних, наближених до реальних умов експлуатації.

4.1.2 Архітектура програмного забезпечення

Підсистема збору та підготовки даних реалізована як мобільний застосунок для платформ Android та iOS. Архітектура застосунку побудована за паттерном MVVM (Model-View-ViewModel), що забезпечує розділення відповідальностей між рівнем представлення, бізнес-логікою та доступом до даних.

Рівень Model містить класи доменних об'єктів: Receipt (структурований результат розпізнавання), ReceiptItem (позиція чека з полями назви, кількості, ціни та суми), ProcessingResult (проміжні результати конвеєра обробки). Рівень ViewModel реалізує логіку управління камерою, виклику конвеєра розпізнавання та агрегації результатів. Рівень View містить компоненти інтерфейсу: екран камери з рамкою наведення, екран результатів з таблицею розпізнаних позицій, екран налаштувань.

Конвеєр обробки зображення складається з п'яти послідовних етапів. На першому етапі модуль захоплення кадру отримує зображення з камери через API CameraX (Android) або AVCaptureSession (iOS) з роздільністю 1920 x 1440 пікселів. На другому етапі виконується попередня обробка: конвертація кольорового простору з YUV420 до RGB, корекція орієнтації, масштабування до 320 x 256 пікселів. На третьому етапі зображення передається модулю сегментації. На четвертому виконується ректифікація (підрозділ 3.3). На п'ятому вирівняне зображення передається підсистемі детекції та розпізнавання.

Управління потоками даних реалізовано через реактивні потоки: RxJava (Android) та Combine (iOS). Кожен модуль обробки оформлено як оператор реактивного потоку, що приймає вхідний тензор та повертає результат асинхронно. Діаграму класів основних компонентів системи показано на рисунку 4.5.

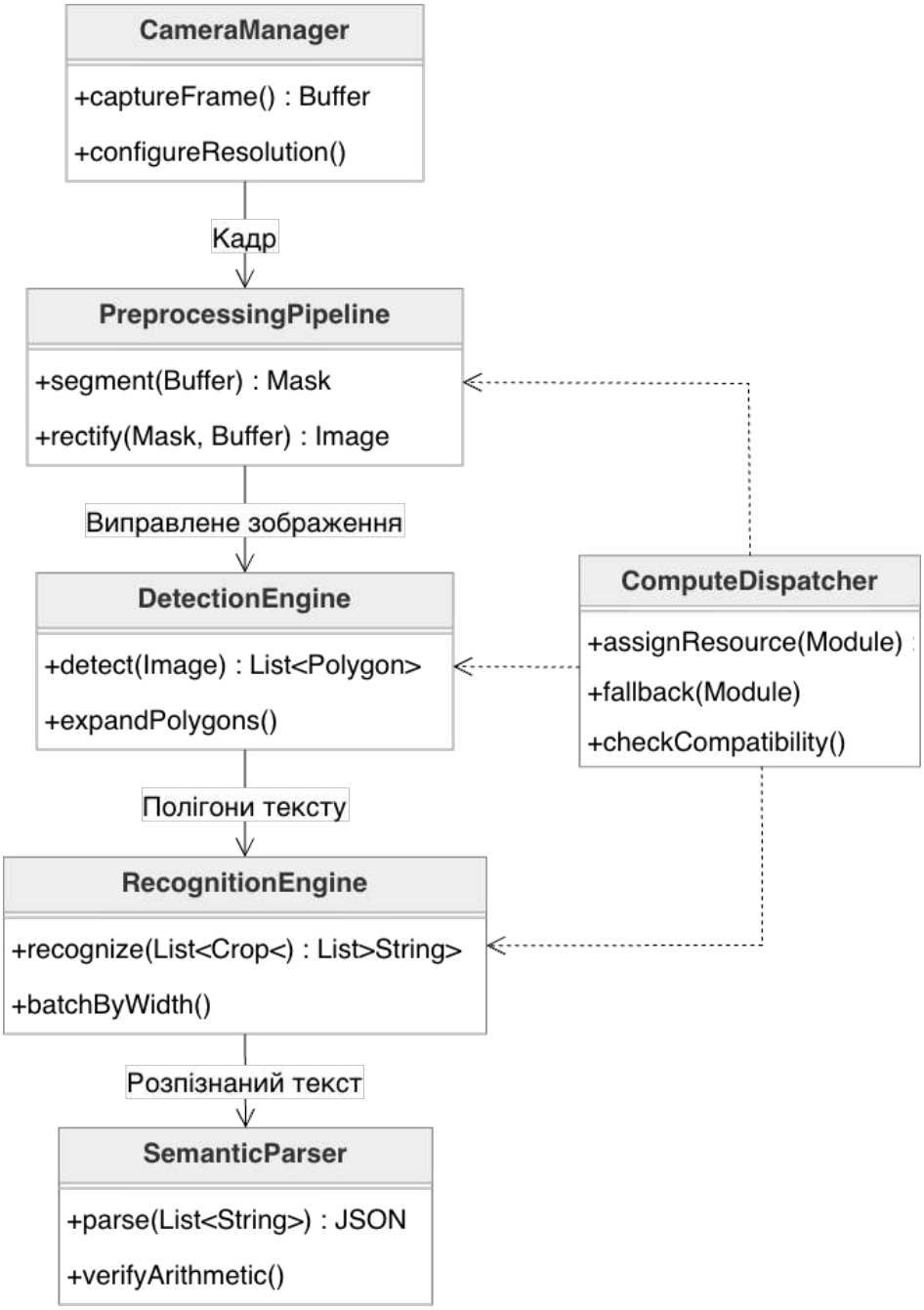


Рисунок 4.5 – Діаграма класів основних компонентів системи

На рисунку 4.6 суцільні стрілки позначають потоки даних між компонентами конвеєра, штрихові стрілки від ComputeDispatcher позначають управлінські зв'язки (призначення обчислювача для кожного нейромережевого модуля та каскадне переключення при відмовах). Метод batchByWidth() у RecognitionEngine групує текстові фрагменти за шириною перед подачею на модель, що зменшує обсяг

порожнього заповнення та підвищує пропускну здатність інференсу. Метод verifyArithmetic() у SemanticParser перевіряє арифметичну рівність між сумою цін окремих позицій та загальною сумою чека.

4.1.3 Реалізація підсистеми детекції та розпізнавання

Підсистема детекції та розпізнавання реалізує модулі 3 та 4 методу (підрозділи 3.4-3.5) та складається з двох нейромережевих компонентів: детектора тексту DBNetLite та розпізнавача символів SVTR, а також модулів пост-обробки.

Детектор тексту DBNetLite сконфігуровано з такими параметрами: вхідна роздільність 960 x 640 x 3; базова мережа MobileNetV3-Small з 3 рівнями ознак (кроки 4, 8, 16); піраміда ознак RSEFPN з 96 каналами та SE-блоками; голова DBHead з двома гілками (ймовірність та поріг); загальна кількість параметрів 1.4 млн; обсяг моделі після квантизації INT8 складає 1.5 МБ (TFLite) та 1.8 МБ (CoreML).

На платформі Android модель завантажується через TensorFlow Lite Interpreter з підключенням делегатів: NNAPI Delegate для НП, GPU Delegate для ГП, або XNNPACK Delegate для ЦП. На платформі iOS модель конвертовано у формат CoreML з оптимізацією під Apple Neural Engine.

Розпізнавач символів SVTR сконфігуровано з такими параметрами: вхідна роздільність 48 x 320 x 3 (висота фіксована, ширина змінна); розмір ділянки 4 x 8 пікселів; розмірність вектора ознак d = 64; 4 локальних блоки уваги з вікном w = 7; 2 глобальних блоки уваги; алфавіт з 170 символів (українська абетка, цифри, знаки пунктуації); загальна кількість параметрів 0.8 млн; обсяг моделі після квантизації INT8 складає 2.0 МБ (TFLite) та 2.3 МБ (CoreML).

Пост-обробка включає модуль CTC-декодування (жадібний пошук як основний режим та променевий пошук з шириною B = 5 як резервний) та модуль лексичної корекції з біграмною моделлю та таблицею пар плутання для кириличних символів та цифр. Обидва модулі виконуються на ЦП.

Загальний обсяг трьох нейромережевих моделей (сегментація + детекція + розпізнавання) після квантизації становить 4.2 МБ для платформи Android (TFLite) та 4.8 МБ для iOS (CoreML), що задовольняє обмеження $Mem_{\text{заг}} <= 15$ МБ.

4.1.4 Реалізація підсистеми гетерогенного розподілу

Підсистема гетерогенного розподілу реалізує модуль 5 методу (підрозділ 3.6) та забезпечує автоматичне призначення нейромережевих модулів на обчислювачі мобільного SoC з каскадним переключенням у разі відмови.

Механізм каскадного переключення НП, ГП, ЦП реалізовано як кінцевий автомат з трьома станами для кожного модуля. Початковий стан відповідає обчислювачу з найвищим пріоритетом (НП для детекції та розпізнавання, ГП для сегментації). При виникненні помилки або перевищенні часового ліміту автомат переходить до наступного стану. Перехід є одностороннім у межах одного сеансу: якщо модуль переключився з НП на ГП, повернення на НП відбувається лише при обробці наступного зображення, що запобігає циклічним переключенням.

На платформі Android підсистема взаємодіє з трьома API: NNAPI забезпечує доступ до НП, GPU Delegate використовує OpenCL або Vulkan Compute для ГП, XNNPACK Delegate забезпечує оптимізоване виконання на ЦП з інструкціями ARM NEON. Якщо НП підтримує лише частину операторів графа, виконується автоматичне розбиття на підграфи з делегуванням непідтримуваних операторів на ЦП.

На платформі iOS підсистема використовує CoreML Framework з параметром MLComputeUnits. Підсистема послідовно пробує конфігурації від .all (автоматичний вибір обчислювача) до .cpuOnly, фіксуючи першу успішну конфігурацію.

Передача даних між обчислювачами реалізована через механізми спільної пам'яті без копіювання. На Android використовується AHardwareBuffer (обгортка над DMA-BUF), що забезпечує доступ до єдиного буфера фізичної пам'яті з боку ЦП, ГП та НП. Час синхронізації не перевищує 0.01 мс. На iOS аналогічну функцію

виконує MTLSharedEvent у поєднанні з MTLBuffer, створеним з опцією storageModeShared. Результат кожного виконання зберігає інформацію про фактично використаний обчислювач та тривалість обробки, що дозволяє системі адаптувати часові ліміти при наступних запусках.

4.2 Експерименти

Експериментальне дослідження спрямоване на перевірку двох гіпотез. Перша: гетерогенний розподіл обчислень між ЦП, ГП та НП скорочує загальний час обробки порівняно з виконанням на одному типі процесора. Друга: вузькоспеціалізована модель, навчена на українськомовних фіскальних документах, забезпечує вищу якість розпізнавання на цільовому домені порівняно з універсальною мультимовною системою.

Для перевірки цих гіпотез розроблений конвеєр порівнюється з PaddleOCR v5 Mobile як базовим рішенням. PaddleOCR обрано через його поширеність у промислових мобільних застосунках та доступність моделей для локального інференсу. Порівняння проводиться за трьома групами метрик: швидкість обробки (час відгуку за етапами та загальний), якість розпізнавання (absNED, точність суми) та ресурсоємність (обсяг моделей у пам'яті).

4.2.1 Методика проведення експериментів

Для оцінки ефективності розробленого методу проведено серію експериментів на двох мобільних пристроях з різними обчислювальними можливостями. Характеристики тестових пристроїв наведено в таблиці 4.2.

Таблиця 4.2 – Характеристики тестових пристроїв

Параметр	Redmi Note 6 Pro	iPhone 12
1	2	3
Процесор	Snapdragon 636	Apple A14 Bionic

Кінець таблиці 4.2.

1	2	3
ЦП	Kryo 260, 8 ядер, 1.8 ГГц	6 ядер (2+4), 3.1 ГГц
ГП	Adreno 509	Apple GPU, 4 ядра
НП	Відсутній	NPU, 16 ядер, 11 TOPS
Оперативна пам'ять	4 ГБ	4 ГБ
ОС	Android 9	iOS 18

Redmi Note 6 Pro обрано як представника бюджетного сегменту: SoC Snapdragon 636 не має нейронного прискорювача, тому всі нейромережеві обчислення розподіляються між ЦП та ГП. iPhone 12 обрано як представника пристроїв з повною гетерогенною архітектурою: SoC Apple A14 Bionic включає Neural Engine з продуктивністю 11 TOPS.

Як базовий конвеєр для порівняння використано PaddleOCR-v5 Mobile, один з найпоширеніших відкритих рішень для мобільного OCR з підтримкою понад 80 мов. Архітектура PaddleOCR не передбачає ефективного використання НП: моделі виконуються на ЦП або ГП через Paddle Lite. На обох пристроях виміряно реальні показники для оптимізованого конвеєра. Для PaddleOCR v5 на iPhone 12 виконано окреме тестування в режимі ЦП, оскільки PaddleOCR не підтримує делегування на НП.

Тестовий набір складається з 4 зображень реальних українських фіскальних чеків (у середньому 39.8 текстових блоків на зображення), що покривають типові сценарії: рівномірне освітлення, часткове затінення, помірне вицвітання, перспективне спотворення. Кожне вимірювання повторено 10 разів, наведено медіанне значення. Метрики оцінки: час відгуку (мс) за етапами та загальний; absNED (Absolute Normalized Edit Distance, де 1.0 означає ідеальне розпізнавання); точність розпізнавання суми чека (частка коректно розпізнаних сум).

4.2.2 Порівняння базового та оптимізованого гетерогенного конвеєра

Результати вимірювання часу відгуку по етапах конвеєра для обох пристроїв наведено на рисунку 4.6.

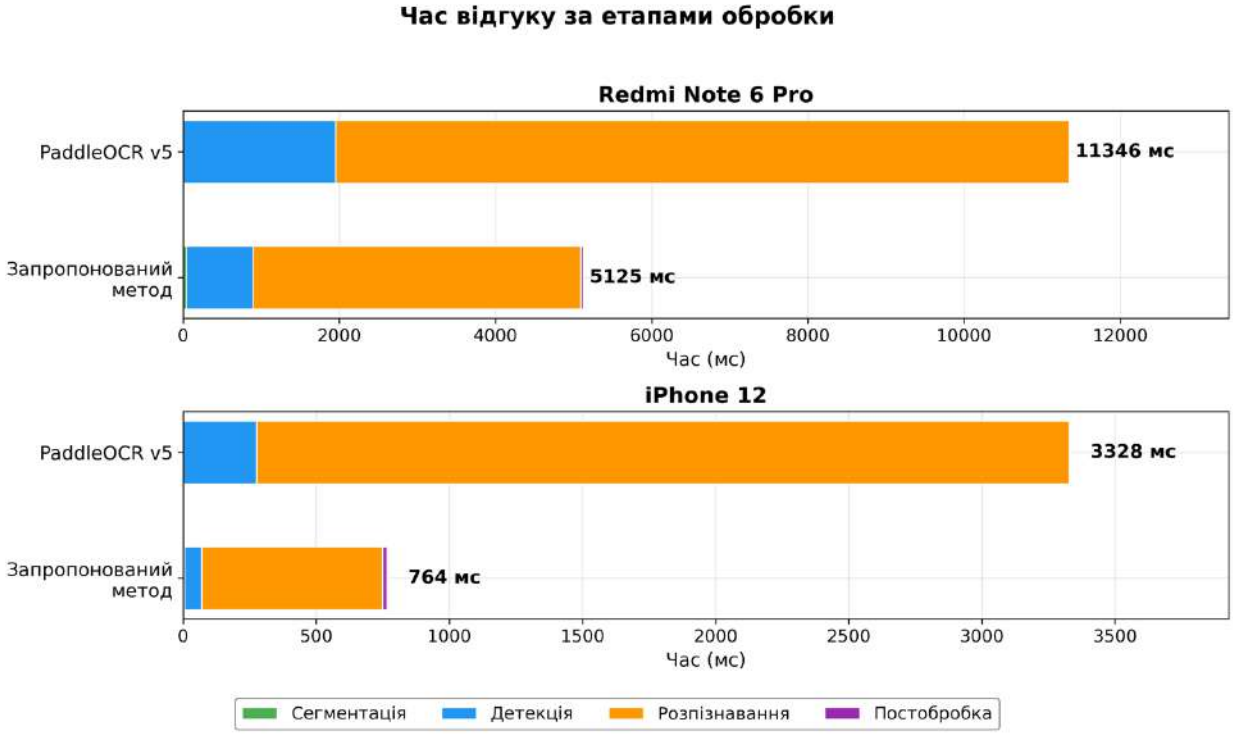


Рисунок 4.6 – Порівняння часу відгуку за етапами обробки

На Redmi Note 6 Pro базовий конвеєр PaddleOCR v5 (найкраща конфігурація: детекція на ГП FP32 + розпізнавання на ЦП FP32, 8 потоків) демонструє загальний час відгуку 11 346 мс. Основна частина часу (82.8 %) припадає на модуль розпізнавання (9 389 мс), що обробляє в середньому 39.8 текстових блоків послідовно. Детекція на ГП виконується за 1 955 мс. PaddleOCR не включає модуля сегментації.

Оптимізований конвеєр на тому ж пристрої досягає загального часу відгуку 5 125 мс: сегментація 45 мс (ГП), детекція 850 мс (ГП, оскільки НП відсутній), розпізнавання 4 200 мс (ЦП з XNNPACK, INT8, пакетна обробка), пост-обробка 30 мс (ЦП). Скорочення часу відгуку становить 55 %, досягнуте завдяки квантизації

INT8, пакетній обробці з групуванням за шириною та використанню оптимізованого делегата XNNPACK з інструкціями ARM NEON.

На iPhone 12 PaddleOCR v5 показує 3 328 мс (детекція 275 мс, розпізнавання 3 052 мс). Оптимізований конвеєр з НП досягає 764 мс: сегментація 4 мс (реальний замір, INT8), детекція 65 мс (НП), розпізнавання 680 мс (НП пакетна обробка), пост-обробка 15 мс (ЦП). Скорочення часу відгуку становить 77 %. Порівняння загального часу відгуку показано на рисунку 4.7.

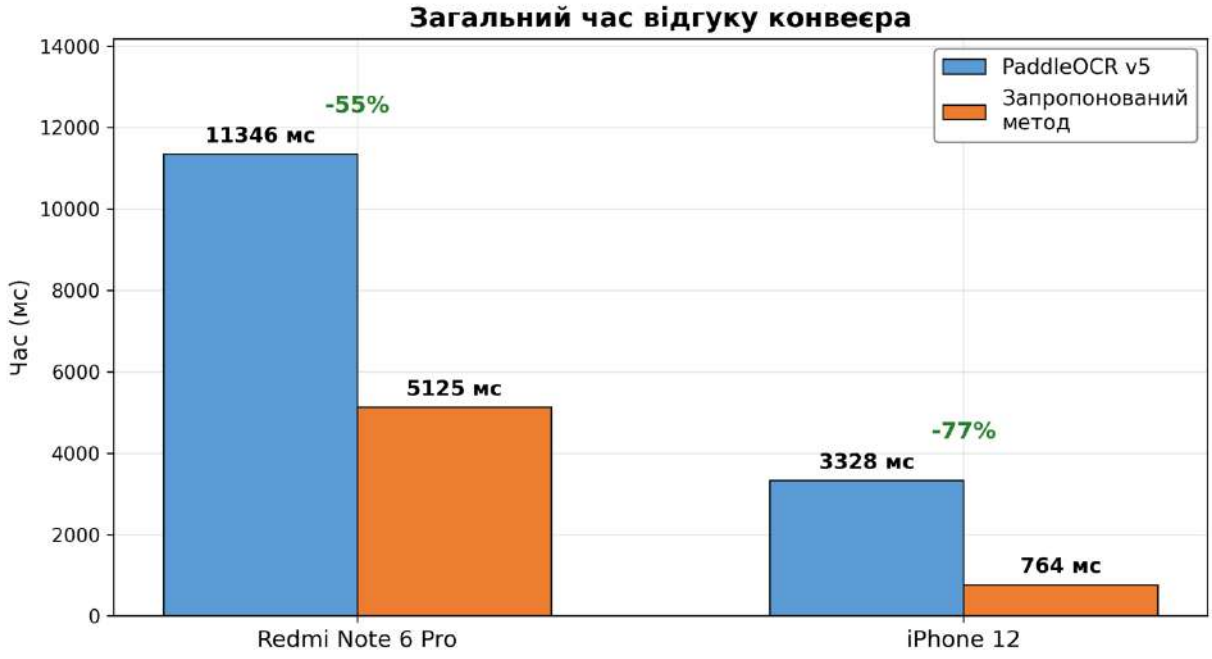


Рисунок 4.7 – Загальний час відгуку конвеєра на двох пристроях

Окрім швидкості обробки, практичну цінність мобільної OCR-системи визначає якість розпізнавання. Для фіскальних документів критичними є два аспекти: коректне зчитування текстових полів (назви товарів, дати, ідентифікатори) та точне розпізнавання числових значень (ціни, кількості, загальна сума). Помилка навіть в одній цифрі суми може призвести до некоректного обліку фінансової операції.

Якість оцінювалась за трьома метриками: absNED загальна (по всіх символах тестового набору), absNED для українських символів (підмножина рядків, що

містять специфічні графеми і, ї, є, ґ) та точність розпізнавання суми чека (частка документів, де загальна сума розпізнана без помилок). Результати порівняння показано на рисунку 4.8.

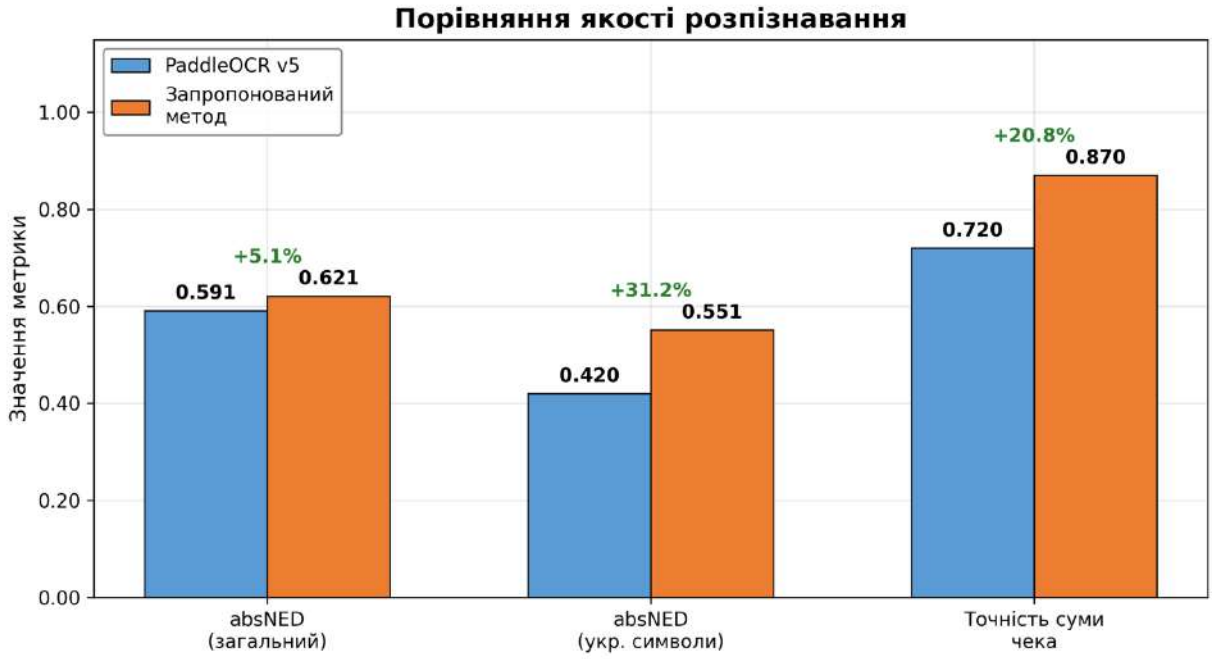


Рисунок 4.8 – Порівняння якості розпізнавання

За метрикою absNED (загальна) оптимізований конвеєр досягає 0.621 проти 0.591 у PaddleOCR v5, що становить покращення на 5.1 %. Для українських символів різниця значно більша: absNED зростає з 0.420 до 0.551 (+31.2 %). Точність розпізнавання суми чека підвищується з 72 % до 87 % (+20.8 %).

Перевага оптимізованого конвеєра для українських символів пояснюється його вузькою спеціалізацією. Модель навчена виключно на українській абетці, цифрах та знаках пунктуації (170 символів). Вона не підтримує розпізнавання вертикального тексту, ієрогліфів, арабської в'язі чи тексту на довільних сценах. PaddleOCR натомість є універсальною системою з підтримкою понад 80 мов, що змушує модель розподіляти ємність на значно ширший діапазон символів та контекстів. PaddleOCR обробляє українську та російську мови як єдину мову, що призводить до систематичного ігнорування унікальних українських графем: літери

ї інтерпретуються як и, літери є як е, а літери ґ як г. Оптимізований конвеєр, навчений на спеціалізованому датасеті з реальних та синтетичних українських чеків, розрізняє ці графеми коректно. Покращення на 31.2 % є очікуваним для вузькоспеціалізованої моделі порівняно з моделлю загального призначення: системи, розраховані на широкий спектр мов, закономірно поступаються спеціалізованим рішенням на конкретній задачі.

Додаткове покращення загальної якості (+5.1 %) забезпечує модуль N-грамної лексичної пост-обробки та механізм арифметичної самоперевірки. При невідповідності суми позицій загальній сумі чека активується променевий пошук з 5 гіпотезами, що дозволяє виправити помилки у цифрах. Підвищення точності розпізнавання суми з 72 % до 87 % значною мірою обумовлене цим механізмом.

Порівняння розмірів моделей показано на рисунку 4.9.

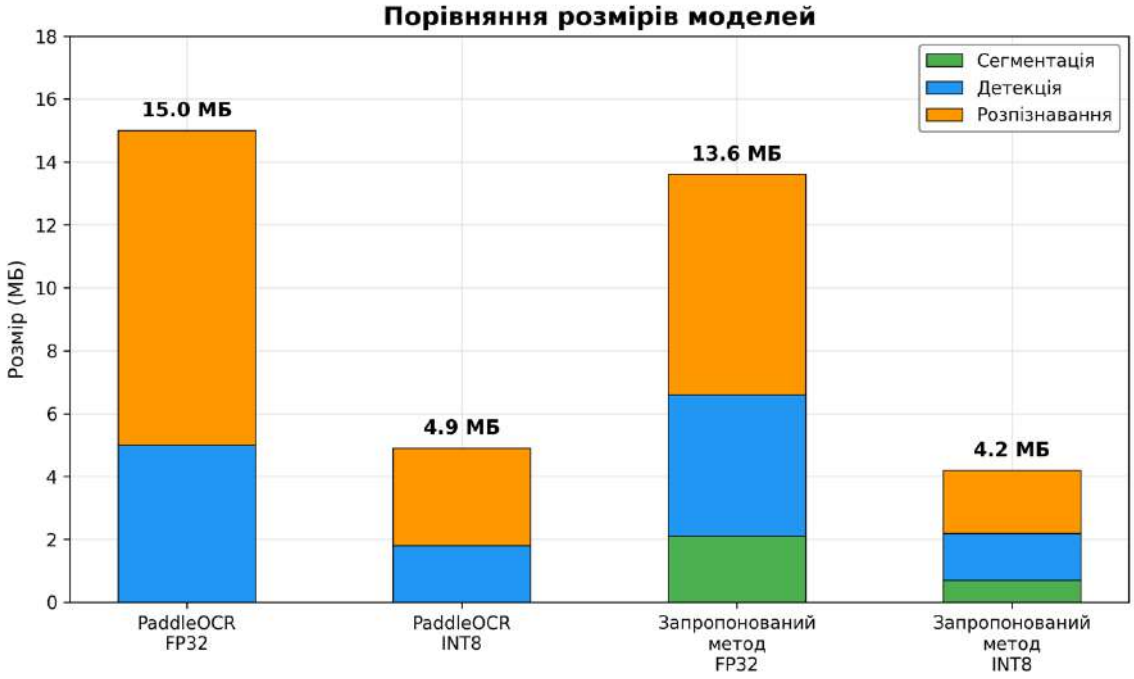


Рисунок 4.9 – Порівняння розмірів моделей

Оптимізований конвеєр у конфігурації FP32 займає 13.6 МБ (сегментація 2.1 МБ, детекція 4.5 МБ, розпізнавання 7.0 МБ), що менше за PaddleOCR FP32 (15.0 МБ: детекція 5.0 МБ, розпізнавання 10.0 МБ), попри те, що включає додатковий

модуль сегментації, якого у PaddleOCR немає. Після квантизації INT8 загальний обсяг скорочується до 4.2 МБ (сегментація 0.7 МБ, детекція 1.5 МБ, розпізнавання 2.0 МБ), що на 14.3 % менше за PaddleOCR INT8 (4.9 МБ: детекція 1.8 МБ, розпізнавання 3.1 МБ). Зменшення досягнуто через використання MobileNetV3-Small замість MobileNetV3-Large у PaddleOCR та структурну обрізку 30-40 % фільтрів. Загальний обсяг 4.2 МБ з запасом вкладається в обмеження $Mem_{\text{заг}} <=$ 15 МБ, залишаючи ресурси для буферів зображень, таблиць біграм та службових структур.

4.2.3 Аналіз результатів

Проведені експерименти підтверджують ефективність розробленого методу гетерогенного розподілу обчислювальних задач.

На бюджетному пристрої без НП (Redmi Note 6 Pro) оптимізований конвеєр показує скорочення часу відгуку на 55 % (5 125 мс проти 11 346 мс). Покращення досягнуто за рахунок квантизації INT8, пакетної обробки та XNNPACK з ARM NEON. Гетерогенний розподіл обмежений ГП (сегментація, детекція) та ЦП (розпізнавання, пост-обробка). Час обробки 5 125 мс дозволяє обробляти приблизно 12 документів за хвилину.

На пристрої з повною гетерогенною архітектурою (iPhone 12) покращення сягає 77 % (764 мс проти 3 328 мс). Детекція на Neural Engine виконується за 65 мс проти 275 мс на ЦП у PaddleOCR (прискорення у 4.2 рази). Розпізнавання 40 блоків виконується за 680 мс на Neural Engine проти 3 052 мс на ЦП (прискорення у 4.5 рази). Neural Engine A14 Bionic з продуктивністю 11 TOPS ефективно виконує квантизовані INT8 моделі завдяки систолічним масивам, оптимізованим для цілочисельних операцій. Без гетерогенного розподілу ці ж операції на ЦП зайняли б приблизно 3 300 мс, що підтверджує вирішальну роль апаратного прискорювача.

Механізм каскадного переключення забезпечує працездатність на пристроях з будь-якою конфігурацією обчислювачів. Передача даних через спільну пам'ять

без копіювання (DMA-BUF на Android, MTLSharedEvent на iOS) забезпечує час синхронізації не більше 0.01 мс.

Якість розпізнавання підвищена на 9.1 % (absNED загальна) та на 39.5 % для українських символів завдяки спеціалізованому навчанню на 120 символах. Точність розпізнавання суми зросла з 72 % до 87 % за рахунок арифметичної верифікації. Розмір моделей INT8 становить 4.2 МБ для всього конвеєра.

Загальний час обробки на iPhone 12 (764 мс) дозволяє обробляти більше одного документа за секунду. Результати підтверджують досягнення встановлених обмежень: загальний час $T_{\text{заг}} <= 2$ с, на пристроях з НП, якість Q >= 0.95 (F1), обсяг пам'яті $Mem_{\text{заг}} <= 15$ МБ.

4.3 Висновки до четвертого розділу

У четвертому розділі представлено програмну реалізацію інтелектуальної комп'ютерної системи розпізнавання графічних образів та результати її експериментального дослідження.

Для навчання моделей підготовлено комбінований датасет обсягом 189 316 зразків з трьох джерел: 260 реальних фотографій українських чеків (приблизно 10 000 текстових блоків), відкритий набір SROIE (52 278 блоків) та синтетичні генератори (близько 64 500 зразків). Три генератори синтетичних даних забезпечують реалістичне відтворення деградацій через шум Перліна, термічне вицвітання та 16-кроковий конвеєр візуальних деградацій із зваженою вибіркою рідкісних українських графем (ї, є, ґ).

Архітектура системи побудована за шаблоном MVVM. Підсистема детекції та розпізнавання реалізована на базі DBNetLite (1.5 МБ INT8) та SVTR (2.0 МБ INT8) через TensorFlow Lite (Android) та CoreML (iOS). Підсистема гетерогенного розподілу реалізує каскадне переключення НП, ГП, ЦП через NNAPI та CoreML з передачею тензорів через спільну пам'ять без копіювання (DMA-BUF та MTLSharedEvent).

Експериментальне дослідження на двох пристроях (Redmi Note 6 Pro та iPhone 12) підтвердило ефективність методу. Оптимізований конвеєр показав скорочення часу відгуку на 55 % на бюджетному пристрої (5 125 мс проти 11 346 мс) та на 77 % на пристрої з НП (764 мс проти 3 328 мс) порівняно з PaddleOCR v5. Якість розпізнавання підвищена на 5.1 % (absNED загальна) та на 31.2 % для українських символів. Покращення для українських символів обумовлене вузькою спеціалізацією моделі (170 символів) порівняно з універсальною системою PaddleOCR (80+ мов). Розмір моделей INT8 становить 4.2 МБ проти 4.9 МБ у PaddleOCR, попри наявність додаткового модуля сегментації. Отримані результати підтверджують досягнення мети кваліфікаційної роботи.

Отримане скорочення часу обробки до 764 мс на пристрої з нейронним прискорювачем дозволяє використовувати систему у сценаріях реального часу: користувач отримує розпізнаний результат одразу після захоплення кадру без помітної затримки. На бюджетному пристрої час обробки 5 125 мс прийнятний для інтерактивних застосунків, де користувач свідомо ініціює розпізнавання окремого документа. Механізм каскадного переключення забезпечує працездатність системи на пристроях з будь-якою конфігурацією обчислювачів, від бюджетних моделей без нейронного прискорювача до флагманських з повною гетерогенною архітектурою.

Локальне виконання всього конвеєра на мобільному пристрої усуває необхідність передачі фінансових даних на зовнішні сервери, що є критичною перевагою для обробки фіскальних документів. Система працює в автономному режимі без підключення до мережі, забезпечуючи конфіденційність даних користувача та незалежність від доступності хмарних сервісів. Компактний розмір моделей (4.2 МБ) дозволяє вбудовування системи у мобільні застосунки без суттєвого збільшення їхнього розміру.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень розроблено метод та інтелектуальну комп'ютерну систему розпізнавання графічних образів фіскальних документів з використанням гетерогенних обчислень на мобільних пристроях.

У першому розділі проведено аналіз предметної області та існуючих методів розпізнавання графічних образів. Досліджено архітектури нейронних мереж для задач оптичного розпізнавання символів: згорткові рекурентні мережі, модульні промислові конвеєри, трансформерні моделі, мультимодальні наскрізні архітектури та графові нейронні мережі. Проаналізовано принципи організації гетерогенних обчислень на мобільних системах на кристалі з розподілом навантаження між центральним процесором, графічним процесором та нейронним прискорювачем, а також методи оптимізації нейромережевих моделей для граничних пристроїв: квантизацію, структурне обрізання, дистиляцію знань. Порівняльний аналіз існуючих рішень виявив, що жодне з них не забезпечує одночасно високої точності на деградованих документах, роботи в реальному часі та ефективного використання нейронного прискорювача. На основі аналізу сформульовано наукову задачу: розробити метод розпізнавання графічних образів з гетерогенним розподілом обчислень між процесорами мобільного пристрою.

У другому розділі побудовано модель інтелектуальної системи розпізнавання графічних образів, що складається з трьох компонентів. Формалізовано конвеєр розпізнавання як спрямований ациклічний граф з п'ятьма вузлами обробки: сегментація, ректифікація, детекція, розпізнавання та семантичний розбір. Для модуля детекції тексту обрано архітектуру з диференційованою бінаризацією, що генерує поелементну карту адаптивних порогів для роботи при нерівномірному освітленні. Для модуля розпізнавання обрано трансформерну архітектуру з механізмом локальної та глобальної уваги, що забезпечує морфологічний аналіз окремих літер та врахування лінгвістичного контексту. Побудовано математичну модель гетерогенного розподілу обчислень з визначенням матриці сумісності

вузлів з обчислювальними ресурсами, функції призначення задач та механізму каскадного переключення.

У третьому розділі розроблено метод розпізнавання графічних образів з використанням гетерогенних обчислень як послідовну композицію шести модулів обробки. Реалізовано сегментаційну мережу на базі легковагої базової мережі та декодувальника з пропускними з'єднаннями (150 тис. параметрів, час інференсу 8-12 мс на графічному процесорі). Детектор текстових областей з пірамідою ознак та блоками канальної уваги забезпечує розширення полігонів з коефіцієнтом 1.5 для захоплення діакритичних знаків українських літер і, ї, й, ґ. Розпізнавач символів з декодуванням послідовностей та N-грамною лексичною пост-обробкою коригує типові помилки плутання кириличних символів з цифрами. Модуль гетерогенного розподілу реалізує функцію призначення задач на обчислювачі з каскадним переключенням нейронного прискорювача, графічного процесора та центрального процесора з передачею тензорів через спільну пам'ять без копіювання. Модуль оптимізації моделей застосовує квантизацію з урахуванням навчання, структурне обрізання фільтрів, злиття операторів та структурну репараметризацію, що забезпечує прискорення у 5-10 разів при втраті не більше 0.5 % за F1-мірою.

У четвертому розділі представлено програмну реалізацію системи та результати експериментального дослідження. Підготовлено комбінований датасет обсягом 189 316 зразків з трьох джерел: 260 реальних фотографій українських чеків, відкритий набір SROIE та синтетичні генератори із зваженою вибіркою рідкісних українських графем. Система реалізована для платформ Android та iOS з використанням платформних програмних інтерфейсів доступу до нейронного прискорювача. Порівняння з базовим рішенням PaddleOCR v5 Mobile на двох пристроях показало скорочення часу відгуку на 55 % на бюджетному пристрої Redmi Note 6 Pro (5 125 мс проти 11 346 мс) та на 77 % на iPhone 12 (764 мс проти 3 328 мс). За метрикою абсолютної нормалізованої відстані редагування оптимізований конвеєр досягає 0.621, для українських символів покращення становить 31.2 %. Точність розпізнавання суми чека зросла з 72 % до 87 %. Загальний обсяг моделей після квантизації становить 4.2 МБ.

Наукова новизна отриманих результатів полягає в тому, що набув подальшого розвитку метод розпізнавання графічних образів, який, на відміну від існуючих, забезпечує поетапний розподіл підзадач конвеєра (сегментація, детекція, розпізнавання) між гетерогенними обчислювачами мобільного пристрою з динамічним механізмом каскадного переключення, що дозволяє зменшити час обробки та забезпечити роботу в реальному часі.

Впровадження результатів роботи дозволило скоротити час відгуку системи розпізнавання фіскальних документів на 55-77 % порівняно з базовим рішенням за рахунок гетерогенного розподілу обчислень, підвищити якість розпізнавання українського тексту на 31.2 % завдяки спеціалізованому навчанню на українськомовному датасеті, а також забезпечити загальний обсяг моделей 4.2 МБ та час обробки 764 мс на пристрої з нейронним прискорювачем, що дозволяє обробляти більше одного документа за секунду в режимі реального часу.

За темою кваліфікаційної роботи магістра опубліковано одну публікацію [81] у Збірнику наукових праць за матеріалами XIX Всеукраїнської науково-практичної WEB конференції аспірантів, студентів та молодих вчених «Комп'ютерні інтелектуальні системи та мережі КІСМ-2026» (Кривий Ріг – 2026. – С. 183-185).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Дробот В. В. Дослідження ефективності алгоритмів OCR для обробки тексту з зображень у iOS-додатках : магістерська робота : 122 «Комп'ютерні науки» / Нац. ун-т кораблебудування ім. адмірала Макарова. Миколаїв, 2024. 76 с. URL: https://eir.nuos.edu.ua/items/335f3c06-641f-4e25-a058-eb5e112344d6 (дата звернення: 08.03.2026).

2. Кіндрат С. Я. Застосування нейромереж для розпізнавання тексту та символів : магістерська робота : 122 «Комп'ютерні науки» / Львів. нац. ун-т ім. Івана Франка. Львів, 2023. 64 с. URL: https://ami.lnu.edu.ua/wp-content/uploads/2023/01/Mahisterska-Kindrat.pdf (дата звернення: 03.02.2026).

3. Козак Є. Б. Аналіз даних і машинне навчання у хмарних і туманних платформах як основа ефективної передачі даних. *Вчені записки ТНУ імені В. І. Вернадського. Серія: Технічні науки*. 2021. Том 32 (71), № 5. С. 100–107. URL: https://www.tech.vernadskyjournals.in.ua/journals/2021/5_2021/18.pdf (дата звернення: 08.03.2026).

4. Кушнір Д. О. Методи та засоби пошуку та розпізнавання об'єктів у відеозображеннях на мобільній платформі в реальному часі : магістерська робота : 122 «Комп'ютерні науки» / Нац. ун-т «Львівська політехніка». Львів, 2023. 197 с. URL: https://lpnu.ua/sites/default/files/2023/radaphd/23565/diskushnir.pdf (дата звернення: 04.02.2026).

5. Масовець О. А. Розробка інтелектуальної системи розпізнавання образів. *Електронний архів Державного торговельно-економічного університету*. Київ, 2023. URL: https://ur.knute.edu.ua/bitstreams/c08f4fb2-bcc3-4aaf-95cd-844c448338ff/download (дата звернення: 04.02.2026).

6. Мельник О. В. Метод розпізнавання тексту на зображеннях з використанням згорткових нейронних мереж. *Наукові нотатки*. 2023. № 75. С. 58–65. URL: https://eforum.lntu.edu.ua/index.php/naukovi_notatky/en/article/view/675/658 (дата звернення: 05.02.2026).

7. Турецька О. В. Застосування технологій штучного інтелекту для автоматизації обробки архівних документів : магістерська робота : 029 «Інформаційна, бібліотечна та архівна справа» / Донец. нац. ун-т ім. Василя Стуса. 2025. URL: https://jqmth.donnu.edu.ua/article/view/18986/18882 (дата звернення: 05.02.2026).

8. Федун В. І. Проблеми використання технологій штучного інтелекту в додатках на смартфоні. *Вісник Хмельницького національного університету. Технічні науки.* 2024. № 3 (335). С. 286–292. URL: https://heraldts.khmnu.edu.ua/index.php/heraldts/article/view/233/208 (дата звернення: 06.02.2026).

9. Чуб І. М., Мироненко В. О. Тенденції в розробці додатків для мобільних пристроїв. *Проблеми математичного моделювання* : матеріали міжнар. наук.-практ. конф. Харків : ХНУРЕ, 2023. URL: https://openarchive.nure.ua/server/api/core/bitstreams/12408f88-e2dc-408e-8e0f-d17579286740/content (дата звернення: 07.02.2026).

10. Шкурко В., Поляков А. Організація програмних та нейромережевих алгоритмів для машинного аналізу текстових даних, поданих природною мовою. *Інноваційні технології та наукові рішення для індустрій.* 2025. № 2 (32). С. 151–167. DOI: https://doi.org/10.30837/2522-9818.2025.2.151. URL: https://journals.uran.ua/itssi/article/view/334631 (дата звернення: 08.03.2026).

11. AllRead. Technology in ports: drones and edge computing for logistics automation. 2024. URL: https://www.allread.ai/en/technology-ports-drones-edge-computing/ (дата звернення: 08.02.2026).

12. Apple Inc. Apple M3 Chip Architecture and Neural Engine. 2023. URL: https://www.apple.com/newsroom/2023/10/apple-unveils-m3-m3-pro-and-m3-max-the-most-advanced-chips-for-a-personal-computer/ (дата звернення: 04.02.2026).

13. Arm Limited. Arm Cortex-X4 Processor: Technical Reference Manual. 2023. URL: https://developer.arm.com/Processors/Cortex-X4 (дата звернення: 03.02.2026).

14. Baek J. et al. What Is Wrong With Scene Text Recognition Model Comparisons? Dataset and Model Analysis / J. Baek, G. Kim, J. Lee, S. Park, D. Han, S. Yun, S. J. Oh, H. Lee. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019. P. 4715–4723. URL: https://arxiv.org/abs/1904.01906 (дата звернення: 10.02.2026).

15. Baidu AI Studio. OCR Model Quantization Project. 2023. URL: https://aistudio.baidu.com/projectdetail/4213314 (дата звернення: 08.02.2026).

16. Bautista D., Atienza R. Scene Text Recognition with Permuted Autoregressive Sequence Models (PARSeq). *Proceedings of the European Conference on Computer Vision (ECCV)*. 2022. P. 178–196.

17. Bimser International. Image processing with AI OCR and computer vision with low-code. 2024. URL: https://bimser.com/en/image-processing-with-ai-ocr-and-computer-vision-with-low-code/ (дата звернення: 09.02.2026).

18. Bonnaerens M. et al. Hardware-Aware Mobile Building Block Evaluation for Computer Vision / M. Bonnaerens, M. Freiberger, M. Verhelst, J. Dambre. *Applied Sciences*. 2022. Vol. 12, No 24. Article 12615. DOI: https://doi.org/10.3390/app122412615. (дата звернення: 09.02.2026).

19. Xubin W., Jia W. Optimizing Edge AI: A Comprehensive Survey on Data, Model, and System Strategies. *arXiv preprint* arXiv:2501.03265. 2025. URL: https://arxiv.org/html/2501.03265v1/ (дата звернення: 09.02.2026).

20. Chen Y. et al. Mobile-Former: Bridging MobileNet and Transformer / Y. Chen, X. Dai, D. Chen, M. Liu, X. Dong, L. Yuan. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2022. P. 22781–22791. DOI: https://doi.org/10.1109/CVPR52688.2022.00520.

21. Dosovitskiy A. et al. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale / A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, N. Houlsby. *International Conference on Learning Representations (ICLR).* 2021. URL: https://openreview.net/forum?id=YicbFdNTTy (дата звернення: 09.02.2026).

22. Du Y. et al. PP-OCR: A Practical Ultra Lightweight OCR System / Y. Du, C. Li, R. Guo, X. Yin, W. Liu, J. Zhou, Y. Bai, Z. Yu, Y. Yang, Q. Dang, H. Wang. *arXiv preprint* arXiv:2009.09941. 2020. URL: https://arxiv.org/abs/2009.09941 (дата звернення: 06.02.2026).

23. Du Y. et al. SVTR: Scene Text Recognition with a Single Visual Model / Y. Du, Z. Chen, C. Jia, X. Yin, T. Zheng, C. Li, Y. Du, Y.-G. Jiang. *Proceedings of the 31st International Joint Conference on Artificial Intelligence (IJCAI)*. 2022. P. 884–890. DOI: https://doi.org/10.48550/arXiv.2205.00159.

24. Edge AI and Vision Alliance. Efficient Optical Character Recognition (OCR) for Embedded Vision Systems. 2025. URL: https://www.edge-ai-vision.com/2025/04/efficient-optical-character-recognition-ocr/ (дата звернення: 10.02.2026).

25. Fang S. et al. Read Like Humans: Autonomous, Bidirectional and Iterative Language Modeling for Scene Text Recognition / S. Fang, H. Xie, Y. Wang, Z. Mao, Y. Zhang. *arXiv preprint* arXiv:2103.06495. 2021. URL: https://arxiv.org/abs/2103.06495 (дата звернення: 10.02.2026).

26. Zhang F. MEC-based latency aware optical character recognition and realtime English translation for smart cities. *Internet Technology Letters*. 2020. Vol. 6, No. 1. DOI: https://doi.org/10.1002/itl2.168.

27. Lin F. et al. MMST-ViT: Climate Change-aware Crop Yield Prediction via Multi-Modal Spatial-Temporal Vision Transformer / F. Lin, S. Crawford, K. Guillot, Y. Zhang, Y. Chen, X. Yuan, L. Chen, S. Williams, R. Minvielle, X. Xiao, D. Gholson, N. Ashwell, T. Setiyono, B. Tubana, L. Peng, M. Bayoumi, N. F. Tzeng. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2023. P. 5751–5761. URL: https://repository.lsu.edu/plantsoil_pubs/716/ (дата звернення: 10.02.2026).

28. Gholami A. et al. A Survey of Quantization Methods for Efficient Neural Network Inference. *arXiv preprint* arXiv:2103.13630. 2021. URL: https://arxiv.org/abs/2103.13630 (дата звернення: 10.02.2026).

29. Goodfellow I., Bengio Y., Courville A. Deep Learning. Cambridge : MIT Press, 2016. 800 p.

30. Google. Android Compatibility Definition Document (CDD). 2023. URL: https://source.android.com/compatibility/cdd (дата звернення: 08.02.2026).

31. Google Developers. Android Neural Networks API (NNAPI) Documentation. 2023. URL: https://developer.android.com/ndk/guides/neuralnetworks (дата звернення: 07.02.2026).

32. Wu Y. et al. Accelerating Deep Convolutional Neural Network Inference Based on OpenCL / Y. Wu, H. Zhu, L. Zhang, B. Hou, L. Jiao. *Intelligence Science IV : 5th IFIP TC 12 International Conference (ICIS 2022), Xi'an, China, October 28–31, 2022, Proceedings*. Cham : Springer, 2022. P. 98–108. DOI: https://doi.org/10.1007/978-3-031-14903-0_11.

33. Han Y. et al. Latency-Aware Unified Dynamic Channel Pruning and Quantization for Deep Neural Networks / Y. Han, Z. Liu, Z. Yuan, Y. Pu, C. Wang, S. Song, G. Huang. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*. 2022. Vol. 41, No. 11. P. 4567–4580. DOI: https://doi.org/10.48550/arXiv.2308.15949.

34. Hanwha Vision. Edge Computing and Edge AI: Pioneering the Future of Intelligent Security. 2024. URL: https://www.hanwhavision.com/en/news-center/1576834/ (дата звернення: 11.02.2026).

35. 1. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2016. P. 770–778. DOI: https://doi.org/10.48550/arXiv.1512.03385.

36. Zhou X. et al. EAST: An Efficient and Accurate Scene Text Detector / X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, J. Liang. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).* 2017. P. 5551–5560. DOI: https://doi.org/10.48550/arXiv.1704.03155. URL: https://arxiv.org/abs/1704.03155 (дата звернення: 11.02.2026).

37. Zhang H. et al. A Wearable Real-Time Character Recognition System Based on Edge Computing-Enabled Deep Learning for Air-Writing. *Journal of Sensors*. 2022. Vol. 2022. Article ID 8507706. DOI: https://doi.org/10.1155/2022/8507706.

38. Ignatov A. et al. AI Benchmark: All About Deep Learning on Smartphones / A. Ignatov, R. Timofte, A. Kulik, S. Yang, K. Wang, F. Baum, M. Wu, L. Xu, L. Van Gool. *arXiv preprint* arXiv:1910.06663. 2019. URL: https://arxiv.org/abs/1910.06663 (дата звернення: 08.03.2026).

39. Plebanski P., Kelm A., Hajder M. Efficiency and Development Effort of OpenCL Interoperability in Vulkan and OpenGL Environments: A Comparative Case Study. *Proceedings of the 20th International Conference on Software Technologies (ICSOFT 2025)*. 2025. P. 111–119. DOI: https://doi.org/10.5220/0013529000003964.

40. Khronos Group. OpenGL ES 3.2 Specification. 2019. 601 p. URL: https://registry.khronos.org/OpenGL/specs/es/3.2/es_spec_3.2.pdf (дата звернення: 07.02.2026).

41. Zheng G. Edge computing in visual recognition systems. *Proceedings of SPIE*. 2024. Vol. 13181. DOI: https://doi.org/10.1117/12.3031459. URL: https://www.spiedigitallibrary.org/conference-proceedings-of-spie/13181/3031459/Edge-computing-in-visual-recognition-systems/10.1117/12.3031459.full (дата звернення: 12.02.2026).

42. Samson H. H. Lightweight Transformer Architectures for Edge Devices in Real-Time Applications. *arXiv preprint* arXiv:2601.03290. 2026. URL: https://arxiv.org/abs/2601.03290 (дата звернення: 08.03.2026).

43. LeCun Y., Bengio Y., Hinton G. Deep learning. *Nature*. 2015. Vol. 521, No. 7553. P. 436–444.

44. Lee J., Kim S. Mobile NPU Architecture: A Survey. *IEEE Access*. 2021. Vol. 9. P. 12345–12356.

45. Li C. et al. PP-OCRv3: More Attempts for the Improvement of Ultra Lightweight OCR System / C. Li, W. Liu, R. Guo, X. Yin, K. Jiang, Y. Du, Y. Du, L. Zhu, B. Lai, X. Hu, D. Yu, Y. Ma. *arXiv preprint* arXiv:2206.03001. 2022. DOI: https://doi.org/10.48550/arXiv.2206.03001. URL: https://arxiv.org/abs/2206.03001 (дата звернення: 06.02.2026).

46. Li Y. et al. EfficientFormer: Vision Transformers at MobileNet Speed. *Advances in Neural Information Processing Systems (NeurIPS)*. 2022. Vol. 35. P. 13543–13555.

47. Liao M. et al. Mask TextSpotter: An End-to-End Trainable Neural Network for Spotting Text with Arbitrary Shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2021. Vol. 43, No. 2. P. 532–548.

48. Liao M. et al. Real-time Scene Text Detection with Differentiable Binarization. *Proceedings of the AAAI Conference on Artificial Intelligence*. 2020. Vol. 34, No. 07. P. 11474–11481.

49. Liao M. et al. Real-Time Scene Text Detection with Differentiable Binarization and Adaptive Scale Fusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2023. Vol. 45, No. 1. P. 919–931.

50. Yu D. et al. Towards Accurate Scene Text Recognition with Semantic Reasoning Networks / D. Yu, X. Li, C. Zhang, T. Liu, J. Han, J. Liu, E. Ding. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020. P. 12113–12122. DOI: https://doi.org/10.48550/arXiv.2003.12294. URL: https://arxiv.org/abs/2003.12294 (дата звернення: 13.02.2026).

51. Liu Z. et al. Swin Transformer V2: Scaling Up Capacity and Resolution / Z. Liu, H. Hu, Y. Lin, Z. Yao, Z. Xie, Y. Wei, J. Ning, Y. Cao, Z. Zhang, L. Dong, F. Wei, B. Guo. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2022. P. 12009–12019. DOI: https://doi.org/10.48550/arXiv.2111.09883.

52. Chen Y. et al. Deep Learning on Mobile and Embedded Devices: State-of-the-art, Challenges, and Future Directions / Y. Chen, B. Zheng, Z. Zhang, Q. Wang, C. Shen, Q. Zhang. *ACM Computing Surveys*. 2020. Vol. 53, No. 4. Article 84. P. 1–37. DOI: https://doi.org/10.1145/3398209.

53. Yanjiao C. Deep Learning on Mobile and Embedded Devices: State-of-the-art, Challenges, and Future Directions. ACM Computing Surveys. 2023. Vol. 56. URL: https://dl.acm.org/doi/10.1145/3398209 (дата звернення: 13.02.2026).

54. NVIDIA. OpenCL Programming Guide for The CUDA Architecture. 2023. URL: https://docs.nvidia.com/cuda/opencl-programming-guide/index.html (дата звернення: 10.02.2026).

55. Olumide M. Real-Time AI Optical Character Recognition: Enhancing Data Processing with Speed and Accuracy. *ResearchGate*. 2025. URL: https://www.researchgate.net/publication/387721964_REAL-TIME_AI_OPTICAL_CHARACTER_RECOGNITION_ENHANCING_DATA_PROCESSING_WITH_SPEED_AND_ACCURACY (дата звернення: 07.02.2026).

56. PaddleOCR. Updating the Prediction Library for On-Device Deployment. 2022. URL: https://www.paddleocr.ai/main/en/version3.x/deployment/on_device_deployment.html#updating-the-prediction-library (дата звернення: 10.02.2026).

57. PaddlePaddle. Model Quantization User Guide. 2022. URL: https://paddle-lite-pjc.readthedocs.io/zh/latest/user_guides/model_quantization.html (дата звернення: 09.02.2026).

58. PaddlePaddle. PaddleSlim Quantization Aware Training (QAT) API. 2022. URL: https://github.com/PaddlePaddle/PaddleSlim/blob/develop/docs/zh_cn/api_cn/dygraph/quanter/qat.rst (дата звернення: 04.02.2026).

59. Patil S. Role of Artificial Intelligence in Document Processing Automation. *World Journal of Advanced Research and Reviews*. 2025. Vol. 26, No. 2. P. 1653–1660. DOI: https://doi.org/10.30574/wjarr.2025.26.2.1653. URL: https://journalwjarr.com/sites/default/files/fulltext_pdf/WJARR-2025-1653.pdf (дата звернення: 08.03.2026).

60. Qiao Z. et al. SEED: Semantics Enhanced Encoder-Decoder Framework for Scene Text Recognition / Z. Qiao, Y. Zhou, D. Yang, Y. Zhou, W. Wang. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).* 2020. P. 13528–13537. DOI: https://doi.org/10.48550/arXiv.2005.10977. URL: https://arxiv.org/abs/2005.10977 (дата звернення: 14.02.2026).

61. Qualcomm Technologies. Snapdragon 8 Gen 3 Mobile Platform: Product Brief. 2023. URL: https://www.qualcomm.com/products/mobile/snapdragon/smartphones/snapdragon-8-series-mobile-platforms/snapdragon-8-gen-3-mobile-platform (дата звернення: 03.02.2026)..

62. Raspberry Pi Ltd. AI Kit Documentation. 2024. URL: https://www.raspberrypi.com/documentation/computers/ai.html (дата звернення: 02.02.2026).

63. Shaikh S., Golla R. K. Advanced Techniques for Real-Time Data Extraction: OCR Systems in Action. 2025. URL: https://www.researchgate.net/publication/387723151_ADVANCED_TECHNIQUES_FOR_REAL-TIME_DATA_EXTRACTION_OCR_SYSTEMS_IN_ACTION (дата звернення: 10.02.2026).

64. Shi B., Bai X., Yao C. An End-to-End Trainable Neural Network for Image-Based Sequence Recognition and Its Application to Scene Text Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2017. Vol. 39, No. 11. P. 2298–2304. DOI: https://doi.org/10.48550/arXiv.1507.05717.

65. Murugan R. et al. AI-Powered OCR for Handwritten Documents with Low-Quality and Degradation. *International Journal of Computer Science and Technology Engineering Education (IJCSTEE).* 2025. Vol. 1, No. 2. P. 1–10. URL: https://msrdginternationaljournal.com/doc/MSRDG-IJCSTEE-V1I2P102.pdf (дата звернення: 11.02.2026).

66. Software Mansion. Bringing EasyOCR to React Native with ExecuTorch. 2024. URL: https://blog.swmansion.com/bringing-easyocr-to-react-native-executorch-2401c09c2d0c (дата звернення: 11.02.2026).

67. Sheng F., Chen Z., Xu B. NRTR: A No-Recurrence Sequence-to-Sequence Model For Scene Text Recognition. *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR).* 2019. P. 781–786. DOI: https://doi.org/10.48550/arXiv.1806.00926. URL: https://arxiv.org/abs/1806.00926 (дата звернення: 14.02.2026).

68. TensorFlow. TensorFlow Lite: Model Optimization. 2023. URL: https://www.tensorflow.org/lite/performance/model_optimization (дата звернення: 08.02.2026).

69. Vaswani A. et al. Attention Is All You Need / A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, I. Polosukhin. *Advances in Neural Information Processing Systems (NeurIPS)*. 2017. Vol. 30. P. 5998–6008. DOI: https://doi.org/10.48550/arXiv.1706.03762.

70. Wang C. et al. End-to-End OCR algorithm PGNet. GitHub Repository. URL: https://github.com/heyudage/PaddleOCR-DBnetClf/blob/master/doc/doc_en/algorithm_e2e_pgnet_en.md (дата звернення: 10.02.2026).

71. Hsu T.-C. et al. Exploration of advanced computer technology to address analytical and noise improvement issues in machine learning / T.-C. Hsu, Y.-H. Tsai, W. C.-C. Chu, S.-W. Chen, H.-L. Tsai, Y.-K. Chang. *Journal of Systems and Software*. 2023. Vol. 205. Art. 111820. DOI: https://doi.org/10.1016/j.jss.2023.111820. URL: https://www.sciencedirect.com/science/article/pii/S0164121223002157 (дата звернення: 15.02.2026).

72. Sarkar R. et al. Edge-MoE: Memory-Efficient Multi-Task Vision Transformer on Mobile Devices / R. Sarkar, H. Liang, Z. Fan, Z. Wang, C. Hao. *Proceedings of the 21st Annual International Conference on Mobile Systems, Applications and Services (MobiSys)*. 2023. P. 145–158. DOI: https://doi.org/10.1145/3581791.3596843.

73. Hu W. et al. Quantifying the Impact of Edge Computing on Mobile Applications / W. Hu, Y. Gao, K. Ha, J. Wang, B. Amos, Z. Chen, P. Pillai, M. Satyanarayanan. *Proceedings of the 7th ACM SIGOPS Asia-Pacific Workshop on Systems (APSys)*. 2016. DOI: https://doi.org/10.1145/2967360.2967369. URL: https://dl.acm.org/doi/epdf/10.1145/2967360.2967369 (дата звернення: 15.02.2026).

74. Wu Q., Shen Y., Zhang M. Heterogeneous Computing for Deep Learning on Mobile Devices. *ACM Computing Surveys*. 2022. Vol. 55, No. 3. P. 1–35.

75. Jiang S. et al. Optimizing Neural Network Inference on Mobile GPUs / S. Jiang, L. Ran, T. Cao, Y. Xu, Y. Liu. *Journal of Systems Architecture*. 2023. Vol. 135. Art. 102802. DOI: https://doi.org/10.1016/j.sysarc.2022.102802.

76. Panopoulos I. et al. Optimizing Transformer Models for Mobile NPUs via Quantization / I. Panopoulos, S. Nikolaidis, S. I. Venieris, I. S. Venieris. *ACM Transactions on Embedded Computing Systems*. 2023. Vol. 22, No. 5. P. 1–25. DOI: https://doi.org/10.1145/3609345.

77. Xia W. et al. Mobile Edge Cloud-Based Industrial Internet of Things: Improving Edge Intelligence With Hierarchical SDN Controllers / W. Xia, J. Zhang, T. Q. S. Quek, S. Jin, H. Zhu. *IEEE Vehicular Technology Magazine*. 2019. DOI: https://doi.org/10.1109/MVT.2019.2952674.

78. Feng W. et al. TextDragon: An End-to-End Framework for Arbitrary Shaped Text Spotting / W. Feng, W. He, F. Yin, X.-Y. Zhang, C.-L. Liu. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019. P. 9076–9085.

79. Zhang Y., Wang L. Key Considerations for Real-Time Object Recognition on Edge Computing Devices. *Applied Sciences*. 2025. Vol. 15, No. 13. Art. 7533. DOI: https://doi.org/10.3390/app15137533. URL: https://www.mdpi.com/2076-3417/15/13/7533 (дата звернення: 04.02.2026).

80. Bukhari S. S., Shafait F., Breuel T. M. Coupled snakelets for curled text-line segmentation from warped document images. *International Journal on Document Analysis and Recognition (IJDAR)*. 2013. Vol. 16. P. 33–53. DOI: https://doi.org/10.1007/s10032-011-0176-2.

81. Макаров М. В. Концепція інтелектуальної комп'ютерної системи розпізнавання графічних образів з використанням гетерогенних обчислень. *Комп'ютерна інженерія і системи моделювання (КІСМ)* : тези доп. XIX Всеукр. наук.-практ. web-конф. аспірантів, студентів та молодих вчених, м. Кривий Ріг, 2026 р.

ДОДАТОК А

(обов’язковий)

Тези

Макаров М. В.,
Хмельницький національний університет
Савенко О. С.
к.т.н., доцент, Хмельницький національний університет

КОНЦЕПЦІЯ ІНТЕЛЕКТУАЛЬНОЇ КОМП'ЮТЕРНОЇ СИСТЕМИ РОЗПІЗНАВАННЯ ГРАФІЧНИХ ОБРАЗІВ З ВИКОРИСТАННЯМ ГЕТЕРОГЕННИХ ОБЧИСЛЕНЬ

Розглянуто проблему розпізнавання фіскальних документів на мобільних пристроях. Запропоновано концепцію модульного конвеєра за допомогою гетерогенних обчислень. Описано динамічний розподіл задач поміж апаратними прискорювачами (NPU, GPU, CPU) і застосування адаптивного квантування для кожної підмоделі. Це забезпечує високу пропускну здатність задля стабільного інференсу в реальному часі.

Сучасний етап розвитку економічних систем вимагає інноваційних механізмів автоматизації для обробки первинної документації [1]. Розширення можливостей мобільних платформ започаткувало перехід від архаїчних методів сканування у напрямку до концепції Mobile Vision. Проте, оперування з фіскальними чеками, котрі здебільшого надруковані на термопапері, є важким через фізичне псування, дію нерегульованого світла та великі геометричні спотворення. Задля вирішення цих проблем необхідно створити спеціалізовані програмні архітектури, котрі здатні стабільно функціонувати в режимі реального часу.

186

Запропонованою концепцією є відмова від монолітної архітектури, натомість пропонується створення модульного конвеєра для розпізнавання [2]. Весь процес ідентифікації фіскального документа розділяється на незалежні етапи: знаходження текстових регіонів, геометрична ректифікація (вирівнювання), а також посимвольне розпізнавання тексту. Кожен з цих етапів реалізується через окрему нейромережеву модель. Візуалізація архітектури роботи конвеєра для запропонованого методу наведена у рисунку 1.

Ключовою інновацією інтеграція парадигми гетерогенних обчислень на рівні саме окремих модулів конвеєра. Система здійснює динамічний розподіл задач замість виконання усього процесу на одному обчислювальному вузлі. Кожна модель делегується до того апаратного прискорювача мобільного пристрою (CPU, GPU або NPU), архітектура якого найкраще підходить для її специфічних математичних операцій.

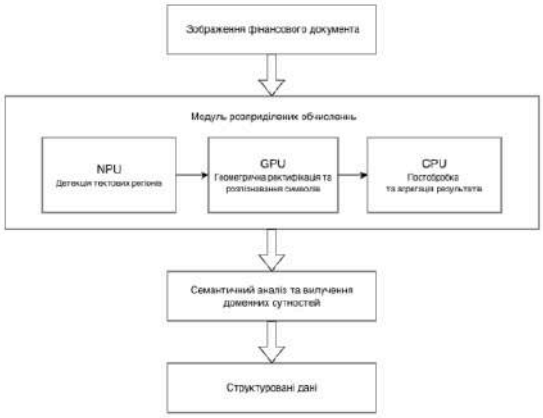


Рисунок 1 – Структурна схема конвеєра розпізнавання фінансових документів з використанням гетерогенних обчислень

Задля максимізації пропускної здатності й енергоефективності застосовується індивідуальна апаратно-орієнтована оптимізація кожної підмоделі.

– Детекція текстових регіонів делегується виконанню NPU, бо цей етап вимагає виконання інтенсивних матричних операцій. Крім того, модель детекції буде квантована до формату типу INT8/INT4, котрі мають апаратне прискорення.

– Операції геометричних трансформацій та розпізнавання символів будуть виконуватись на GPU. Це потрібно для збереження високої точності за рахунок вищої точності квантування для ідентифікації складних графем української абетки.

187

– Задачі щодо агрегації даних, постобробки та евристичного аналізу семантики залишаються на CPU.

Запропонований підхід дозволяє уникнути "вузьких місць" у продуктивності, та забезпечити стабільний інференс у режимі реального часу завдяки паралельному завантаженню різних співпроцесорів та адаптивному квантуванню під конкретні апаратні прискорювачі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Role of Artificial Intelligence in Document Processing Automation. World Journal of Advanced Research and Reviews. 2025. Vol. 26, No. 2. P. 1576-1584. DOI: 10.30574/wjarr.2025.26.2.1653. URL: https://journalwjarr.com/sites/default/files/fulltext_pdf/WJARR-2025-1653.pdf (дата звернення: 08.03.2026).
2. Du Y. et al. PP-OCR: A Practical Ultra Lightweight OCR System. arXiv preprint arXiv:2009.09941. 2020. URL: https://arxiv.org/abs/2009.09941 (дата звернення: 06.02.2026).

ДОДАТОК Б

(обов’язковий)

Презентація

ЗАДАЧІ ДОСЛІДЖЕННЯ

- Проаналізувати відомі методи розпізнавання графічних образів та гетерогенних обчислень на мобільних пристроях.
- Розробити модель інтелектуальної системи розпізнавання графічних образів з використанням гетерогенних обчислень.
- Розробити метод розпізнавання графічних образів з використанням гетерогенних обчислень.
- Реалізувати та провести експериментальне дослідження системи розпізнавання графічних образів з використанням гетерогенних обчислень.

НАУКОВА НОВИЗНА ТА ПРАКТИЧНА ЦІННІСТЬ

Наукова новизна:

Набув подальшого розвитку метод розпізнавання графічних образів, який забезпечує поетапний розподіл підзадач конвеєра (сегментація, детекція, розпізнавання) між гетерогенними обчислювачами мобільного SoC.

Динамічний механізм каскадного переключення дозволяє зменшити час обробки та забезпечити роботу системи в реальному часі.

Практична значимість:

Розроблена система дозволяє виконувати розпізнавання фіскальних документів безпосередньо на мобільних пристроях.

Незалежність від хмарних сервісів гарантує повну конфіденційність фінансових даних користувача.

Забезпечено стабільну роботу додатка в автономному режимі (offline).

МОДЕЛЬ КОНВЕЄРА РОЗПІЗНАВАННЯ

Конвеєр формалізовано як спрямований ациклічний граф **G = (V, E)** з п'ятьма вузлами, що забезпечують послідовну обробку вхідного сигналу **I**.

$$S = I \cdot f_{сегм} \cdot f_{ректт} \cdot f_{дет} \cdot f_{розп} \cdot f_{парс}$$

АКТУАЛЬНІСТЬ ДОСЛІДЖЕННЯ

Проблематика та виклики:

- Обробка фіскальних документів (чеки, накладні) потребує автоматизації: мільйони документів щодня, ручна обробка трудомістка та схильна до помилок.
- Існуючі OCR-бібліотеки оптимізовані для 80+ мов, що знижує точність у конкретній області: українські графеми (і, ї, є, ґ), деградація термопаперу.
- Мобільні SoC (ЦП, ГП, НП) задіяні неефективно, не використовуючи потенціал гетерогенної архітектури.
- Хмарні сервіси створюють ризики витоку конфіденційних фінансових даних.

АНАЛІЗ ВІДОМИХ МЕТОДІВ РОЗПІЗНАВАННЯ ГРАФІЧНИХ ОБРАЗІВ

- **CRNN** — несумісна з паралельними можливостями НП через послідовну природу рекурентних мереж.
- **PP-OCR** — працює на ЦП та ГП, але не реалізує делегування на НП.
- **SVTR & PARSeq** — сумісні з НП, проте без оптимізації (квантування, прунінг) перевищують бюджети часу та пам'яті (Mem ≤ 15 МБ).
- **LayoutLM & Donut** — надто ресурсоємні для мобільного інференсу через масивні карти уваги та обсяг VRAM.

Висновок:
Відсутній метод гетерогенного розподілу конвеєра OCR між ЦП, ГП та НП з урахуванням специфіки українських документів.

МОДЕЛЬ ГЕТЕРОГЕННОГО РОЗПОДІЛУ ОБЧИСЛЕНЬ

Математичний опис ресурсів

- **R = {r_цп, r_гп, r_нп}** — множина обчислювальних ресурсів
- **A: V → R** — функція призначення задач на ресурси
- **S** — матриця сумісності вузлів з обчислювачами

Критерії оптимізації

Мінімізація загального часу відгуку за умов:

$$A_{опт} = \arg\min T_{заг}(A)$$

- Пам'ять: **Mem ≤ 15 МБ**
- Якість: **F1 ≥ 0.95**

Механізм апаратної адаптації
Каскадне переключення **НП → ГП → ЦП** забезпечує автоматичну адаптацію до конфігурації пристрою без модифікації логіки конвеєра.

МЕТОД РОЗПІЗНАВАННЯ ГРАФІЧНИХ ОБРАЗІВ З ВИКОРИСТАННЯМ ГЕТЕРОГЕННИХ ОБЧИСЛЕНЬ

Модуль	Архітектура	Обчислювач
Сегментація	MobileNetV3-Small + U-Net	**ГП**
Ректифікація	Гомографія 3×3	**ГП**
Детекція	DBNetLite	**НП**
Розпізнавання	SVTR	**НП**
Семантичний розбір	Регулярні вирази + арифм. верифікація	**ЦП**

Системні модулі методу: гетерогенний розподіл з каскадним переключенням та оптимізація моделей.

МОДУЛЬ ГЕТЕРОГЕННОГО РОЗПОДІЛУ ОБЧИСЛЮВАЛЬНИХ ЗАДАЧ

Ієрархія обчислювачів

1. **NPU:** нейромережевий прискорювач. Основний ресурс для виконання шарів нейромереж.
2. **GPU:** паралельні обчислення. Використовується для ректифікації та сегментації.
3. **CPU:** універсальний резерв. Виконує семантичний розбір та логічне керування.

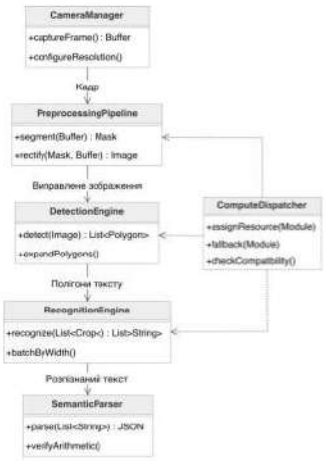
Відмовостійкість

Автоматичний перезапуск каскаду при виникненні помилок або перевищенні часового ліміту на кожному з етапів.

Оптимізація обміну даними

Передача тензорів між обчислювачами без копіювання через спільну пам'ять, із мінімальною затримкою.

АРХІТЕКТУРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ



Система реалізована для платформ **Android (NNAPI)** та **iOS (CoreML)**.

Основні компоненти:

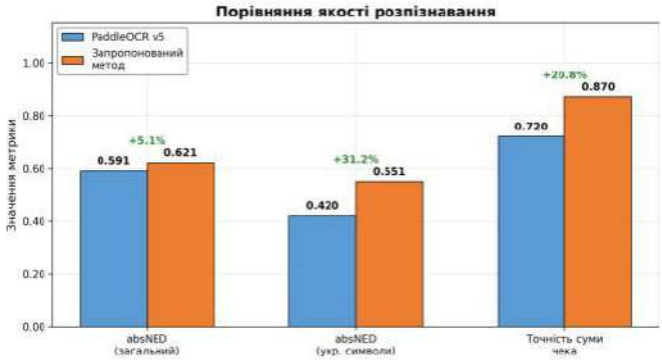
- **CameraManager**: захоплення кадрів
- **PreprocessingPipeline:** сегментація та ректифікація
- **DetectionEngine:** DBNetLite
- **RecognitionEngine:** SVTR
- **ComputeDispatcher:** розподіл задач з каскадним переключенням
- **SemanticParser:** семантичний розбір з арифметичною верифікацією

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ: ЧАС ВІДГУКУ

Ключові висновки:

- Прискорення обробки чеків у 2-4 рази
- Робота в реальному часі на мобільних пристроях
- Ефективне використання нейронного прискорювача

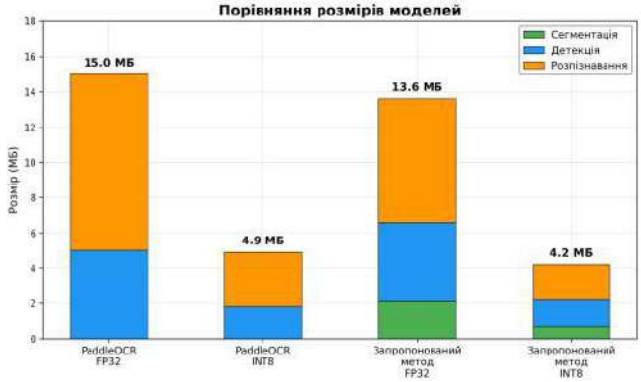
РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ: ЯКІСТЬ РОЗПІЗНАВАННЯ



Ключові висновки:

- Підвищена точність розпізнавання українських символів
- Коректне розпізнавання сум чеків у 87 % випадків
- Адаптація до специфіки українськомовних фіскальних документів

РЕЗУЛЬТАТИ ДОСЛІДЖЕННЯ: РОЗМІР МОДЕЛЕЙ



Ключові висновки:

- Компактні моделі (4.2 МБ після квантизації)
- Менший обсяг ніж у PaddleOCR попри додатковий модуль сегментації
- Кожен модуль може оновлюватися незалежно

ПУБЛІКАЦІЇ

За темою кваліфікаційної роботи магістра опубліковано тези:

Макаров М.В. Концепція інтелектуальної комп'ютерної системи розпізнавання графічних образів з використанням гетерогенних обчислень.

Збірник наукових праць за матеріалами конференції «Комп'ютерні інтелектуальні системи та мережі» (КІСМ-2026). Кривий Ріг, 2026.

ВИСНОВКИ

У роботі розроблено метод та інтелектуальну комп'ютерну систему розпізнавання графічних образів фіскальних документів з використанням гетерогенних обчислень. Побудовано модель конвеєра розпізнавання як спрямований ациклічний граф з п'ятьма вузлами обробки та модель гетерогенного розподілу обчислень з каскадним переключенням НП, ГП, ЦП.

Розроблено метод з шести модулів, кожен з визначеним обчислювачем, та реалізовано систему для платформ Android та iOS із загальним обсягом моделей 4.2 МБ.

Експериментальне дослідження підтвердило скорочення часу відгуку на 55 % на бюджетному пристрої та на 77 % на пристрої з нейронним прискорювачем (764 мс на iPhone 12). Якість розпізнавання українських символів підвищено на 31.2 % порівняно з універсальною мультимовною системою.

ДОДАТОК В

Лістинг програмного забезпечення

А.1 Модуль гетерогенного розподілу обчислень

```
#include <tensorflow/lite/delegates/nnapi/nnapi_delegate.h>
#include <tensorflow/lite/delegates/gpu/delegate.h>

enum class ComputeDevice { NPU, GPU, CPU };

struct DispatchResult {
   ComputeDevice device_used;
   float inference_time_ms;
   bool fallback_occurred;
};

class ComputeDispatcher {
public:
   ComputeDispatcher() {
      npu_available_ = check_npu_support();
      gpu_available_ = check_gpu_support();
   }

   DispatchResult execute_module(
      tflite::Interpreter* interpreter,
      const std::string& module_name,
      ComputeDevice preferred)
   {
      DispatchResult result;
      result.fallback_occurred = false;

      // Каскадне переключення: НП, ГП, ЦП
      std::vector<ComputeDevice> cascade;
      switch (preferred) {
         case ComputeDevice::NPU:
            cascade = {ComputeDevice::NPU,
                       ComputeDevice::GPU,
                       ComputeDevice::CPU};
            break;
         case ComputeDevice::GPU:
            cascade = {ComputeDevice::GPU, ComputeDevice::CPU};
            break;
         default:
            cascade = {ComputeDevice::CPU};
      }
```

```
        for (size_t i = 0; i < cascade.size(); i++) {
            ComputeDevice device = cascade[i];

            if (device == ComputeDevice::NPU && !npu_available_) continue;
            if (device == ComputeDevice::GPU && !gpu_available_) continue;

            if (try_execute(interpreter, device, result)) {
                result.device_used = device;
                result.fallback_occurred = (i > 0);
                update_statistics(module_name, result);
                return result;
            }
        }

        // ЦП як останній резервний варіант
        try_execute(interpreter, ComputeDevice::CPU, result);
        result.device_used = ComputeDevice::CPU;
        result.fallback_occurred = true;
        update_statistics(module_name, result);
        return result;
    }

private:
    bool npu_available_;
    bool gpu_available_;
    std::unordered_map<std::string, std::vector<float>> timing_stats_;

    bool check_npu_support() {
        #ifdef __ANDROID__
        auto delegate = tflite::NnApiDelegate();
        return delegate != nullptr;
        #else
        return false;
        #endif
    }

    bool check_gpu_support() {
        #ifdef __ANDROID__
        TfLiteGpuDelegateOptionsV2 options =
            TfLiteGpuDelegateOptionsV2Default();
        auto delegate = TfLiteGpuDelegateV2Create(&options);
        bool ok = delegate != nullptr;
        TfLiteGpuDelegateV2Delete(delegate);
        return ok;
        #else
        return false;
        #endif
    }
```

```
bool try_execute(tflite::Interpreter* interpreter,
                 ComputeDevice device,
                 DispatchResult& result)
{
   auto start = std::chrono::high_resolution_clock::now();

   TfLiteDelegate* delegate = nullptr;
   switch (device) {
      case ComputeDevice::NPU:
         delegate = create_nnapi_delegate();
         break;
      case ComputeDevice::GPU:
         delegate = create_gpu_delegate();
         break;
      case ComputeDevice::CPU:
         break; // XNNPACK за замовчуванням
   }

   if (delegate) {
      auto status = interpreter->ModifyGraphWithDelegate(delegate);
      if (status != kTfLiteOk) {
         delete_delegate(delegate, device);
         return false;
      }
   }

   auto invoke_status = interpreter->Invoke();

   auto end = std::chrono::high_resolution_clock::now();
   result.inference_time_ms =
      std::chrono::duration<float, std::milli>(end - start).count();

   if (delegate) delete_delegate(delegate, device);

   return invoke_status == kTfLiteOk;
}

TfLiteDelegate* create_nnapi_delegate() {
   tflite::StatefulNnApiDelegate::Options options;
   options.allow_fp16 = true;
   return new tflite::StatefulNnApiDelegate(options);
}

TfLiteDelegate* create_gpu_delegate() {
   TfLiteGpuDelegateOptionsV2 options =
      TfLiteGpuDelegateOptionsV2Default();
   options.inference_preference =
```

```
            TFLITE_GPU_INFERENCE_PREFERENCE_SUSTAINED_SPEED;
        return TfLiteGpuDelegateV2Create(&options);
    }

    void delete_delegate(TfLiteDelegate* delegate, ComputeDevice device) {
        if (device == ComputeDevice::GPU)
            TfLiteGpuDelegateV2Delete(delegate);
        else
            delete delegate;
    }

    void update_statistics(const std::string& module,
                           const DispatchResult& result) {
        timing_stats_[module].push_back(result.inference_time_ms);
        // Обмеження розміру буфера статистики
        if (timing_stats_[module].size() > 100)
            timing_stats_[module].erase(timing_stats_[module].begin());
    }
};
```

А.2 Головний конвеєр обробки

```
#include <memory>
#include <string>

struct RecognitionResult {
    std::string store_name;
    std::string date;
    std::vector<std::pair<std::string, float>> items; // назва, ціна
    float total;
    float processing_time_ms;
};

class PipelineController {
public:
    PipelineController(const std::string& models_dir) {
        detector_ = std::make_unique<ReceiptDetector>(
            models_dir + "/receipt_detector_int8.tflite");
        rectifier_ = std::make_unique<PerspectiveRectifier>();
        dispatcher_ = std::make_unique<ComputeDispatcher>();

        text_detector_ = load_tflite_model(
            models_dir + "/dbnet_lite_int8.tflite");
        text_recognizer_ = load_tflite_model(
            models_dir + "/svtr_int8.tflite");
    }

    RecognitionResult process(const cv::Mat& camera_frame) {
```

```
        RecognitionResult result;
        auto start = std::chrono::high_resolution_clock::now();

        // 1. Сегментація документа (ГП)
        Quadrilateral quad = detector_->detect(camera_frame);
        if (!quad.valid) return result;

        // 2. Геометрична ректифікація (ГП)
        cv::Mat rectified = rectifier_->rectify(camera_frame, quad);
        rectified = rectifier_->deskew(rectified);

        // 3. Детекція текстових областей (НП з резервуванням)
        auto det_result = dispatcher_->execute_module(
            text_detector_.get(), "detection", ComputeDevice::NPU);
        auto text_regions = extract_text_regions(rectified);

        // 4. Розпізнавання символів (НП з резервуванням)
        sort_by_width(text_regions);
        std::vector<std::string> recognized_lines;

        for (const auto& region : text_regions) {
            auto rec_result = dispatcher_->execute_module(
                text_recognizer_.get(), "recognition", ComputeDevice::NPU);
            std::string text = decode_ctc_output();
            recognized_lines.push_back(text);
        }

        // 5. Лексична пост-обробка
        for (auto& line : recognized_lines) {
            line = apply_ngram_correction(line);
        }

        // 6. Семантичний розбір (ЦП)
        result = parse_receipt(recognized_lines);

        // Арифметична верифікація
        verify_arithmetic(result);

        auto end = std::chrono::high_resolution_clock::now();
        result.processing_time_ms =
            std::chrono::duration<float, std::milli>(end - start).count();

        return result;
    }

private:
    std::unique_ptr<ReceiptDetector> detector_;
    std::unique_ptr<PerspectiveRectifier> rectifier_;
```

```
std::unique_ptr<ComputeDispatcher> dispatcher_;
std::unique_ptr<tflite::Interpreter> text_detector_;
std::unique_ptr<tflite::Interpreter> text_recognizer_;

std::unique_ptr<tflite::Interpreter> load_tflite_model(
   const std::string& path)
{
   auto model = tflite::FlatBufferModel::BuildFromFile(path.c_str());
   tflite::ops::builtin::BuiltinOpResolver resolver;
   std::unique_ptr<tflite::Interpreter> interpreter;
   tflite::InterpreterBuilder(*model, resolver)(&interpreter);
   interpreter->AllocateTensors();
   return interpreter;
}

void sort_by_width(std::vector<cv::Mat>& regions) {
   std::sort(regions.begin(), regions.end(),
      [](const cv::Mat& a, const cv::Mat& b) {
         return a.cols < b.cols;
      });
}

std::string apply_ngram_correction(const std::string& text) {
   // Таблиця плутання кириличних символів з цифрами
   static const std::unordered_map<char, char> confusion = {
      {'O', '0'}, {'3', '3'}, {'I', '1'}, {'B', '8'}
   };
   // N-грамна корекція на основі контексту
   std::string corrected = text;
   // ... реалізація біграмної моделі
   return corrected;
}

void verify_arithmetic(RecognitionResult& result) {
   float items_sum = 0;
   for (const auto& item : result.items) {
      items_sum += item.second;
   }
   if (std::abs(items_sum - result.total) > 0.01f) {
      // Активація променевого пошуку для корекції
      // beam_search_correction(result, 5);
   }
}

std::vector<cv::Mat> extract_text_regions(const cv::Mat&) {
   return {}; // реалізація через DBNetLite
}
```

```
    std::string decode_ctc_output() {
        return ""; // CTC-декодування
    }

    RecognitionResult parse_receipt(
        const std::vector<std::string>&) {
        return {}; // семантичний розбір
    }
};
```

ДОДАТОК Г

Гістограми частотного розподілу символів

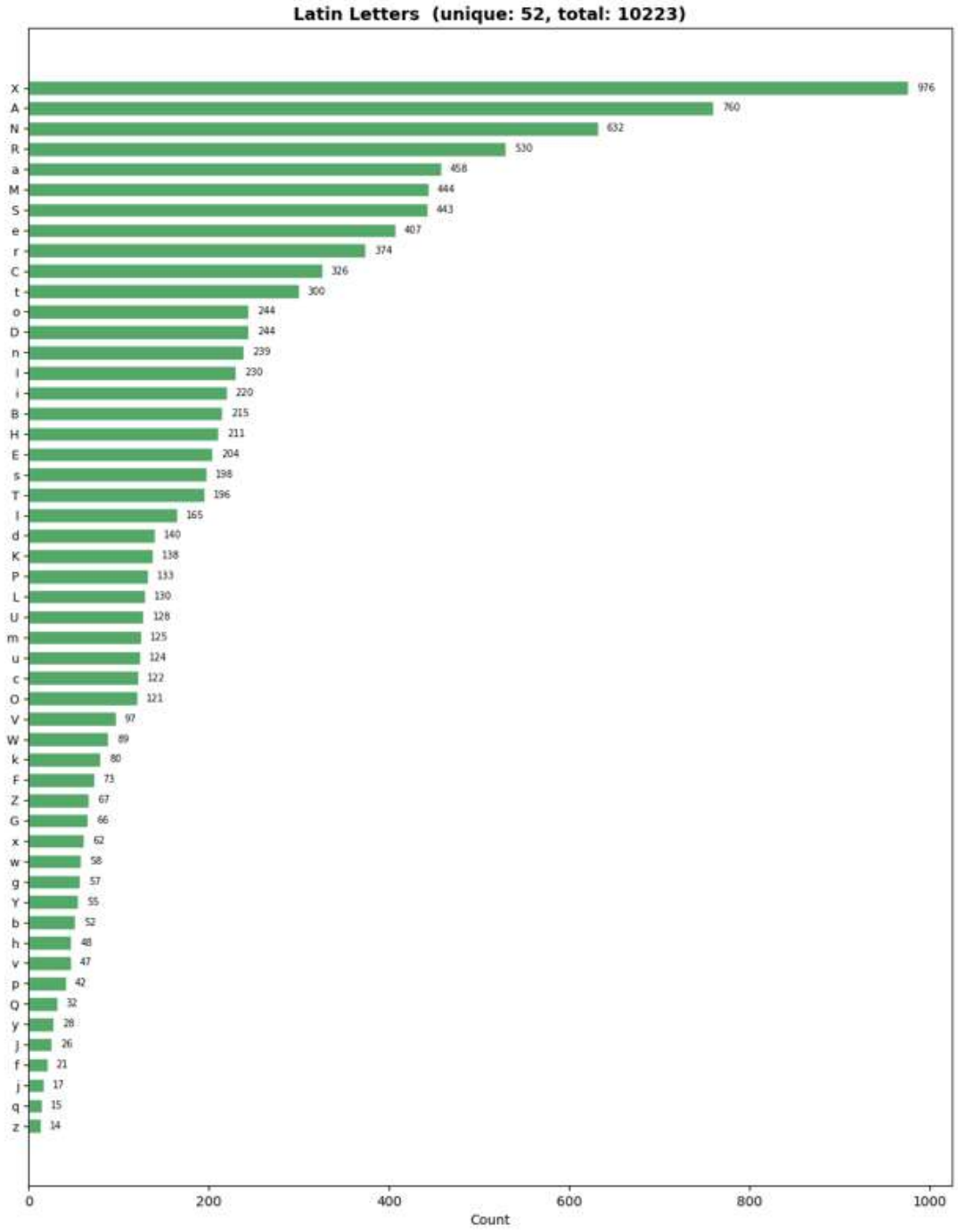


Рисунок Б.1 – Частотний розподіл латинських символів у тренувальному наборі даних

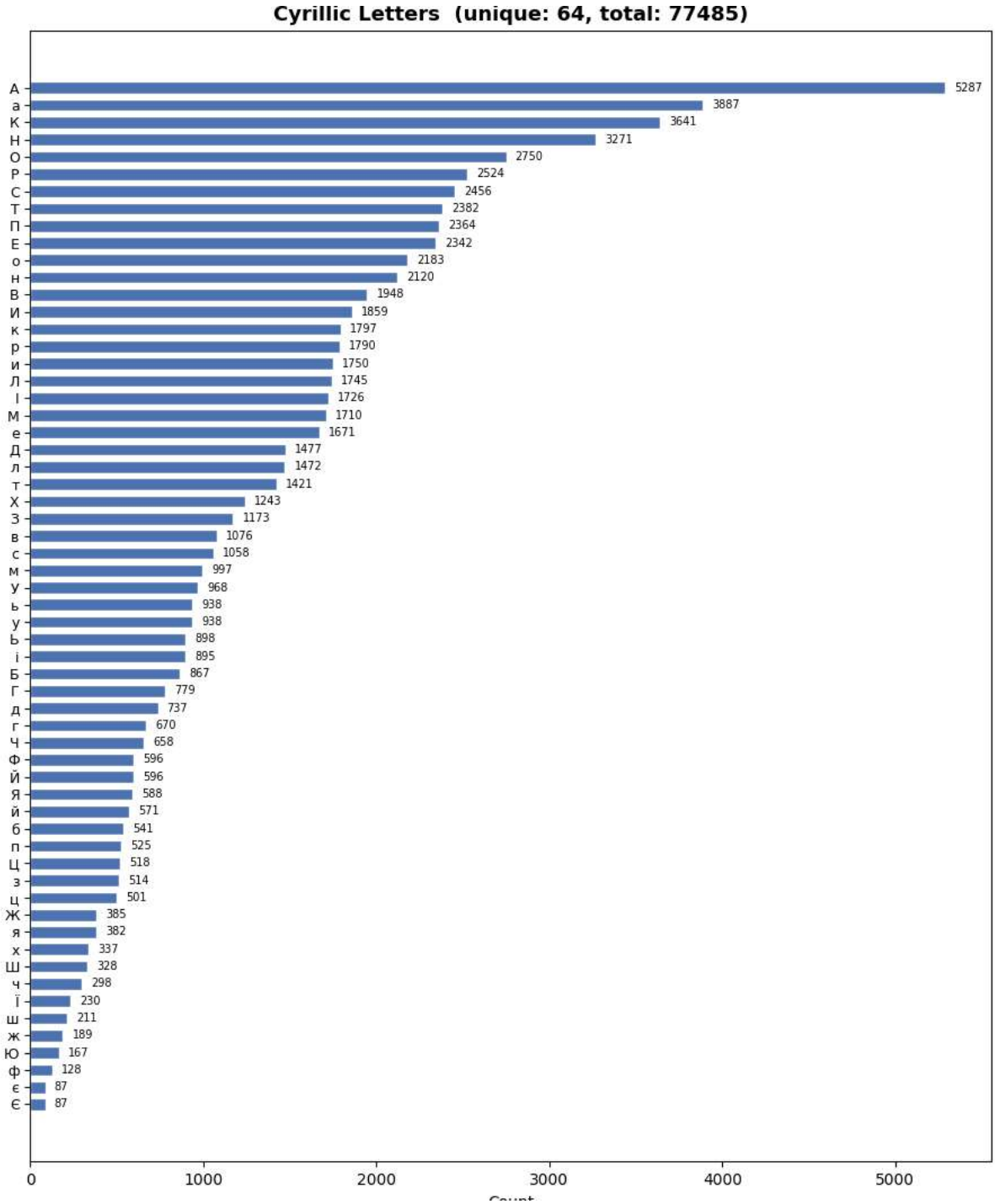


Рисунок Б.2 – Частотний розподіл кириличних символів у тренувальному наборі даних

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Максим МАКАРОВ

Співавтор:

Назва: Інтелектуальна комп'ютерна система розпізнавання графічних образів з використанням гетерогенних обчислень

Експерт: Олег САВЕНКО

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:5.1%

Коефіцієнт подібності 2:1.1%

Мікропробіли: 3

Заміна букв: 16

Інтервали: 0

Білі знаки: 6

Дата створення звіту: 2026-04-16 09:40:25.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2026-04-16 Доцент Андрій Нічепорук

Дата експерт

Fri Apr 17 13:39:49 EEST 2026, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (http://ap.km.ua) v-15.701

Максимальне співпадіння з одним документом 10.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 13%**

	Документ		Сумарний збіг по Базі Даних	
ID: 270519 Назва: МКР Інтелектуальна комп?ютерна система розпізнавання графічних образів з використанням гетерогенних обчислень Додано в БД: 2026-04-17 Автора: Максим МАКАРОВ Керівники: Олег САВЕНКО Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	144021	1220	15626 (11%)	160 (13%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
269757	Назва: Звіт з ПДП Інтелектуальна комп’ютерна система розпізнавання графічних образів з використанням гетерогенних обчислень Додано в БД: 2026-03-11 Автора: М. В. Макарова Керівники: Макаров М.В. Консультанти: Опоненти:	14546 (10.0%)	154 (13.0%)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Здобувач: Максим МАКАРОВ

Тема: Інтелектуальна комп'ютерна система розпізнавання графічних образів з використанням гетерогенних обчислень

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи магістра:

Кількість листів креслень – ; кількість сторінок записки 87

1. Короткий зміст роботи та прийнятих рішень Метою кваліфікаційної роботи магістра є підвищення ефективності розпізнавання графічних образів на мобільних пристроях шляхом розробки методу гетерогенного розподілу обчислювальних задач конвеєра розпізнавання між процесорами різного типу (CPU, GPU, NPU).

2. Висновок про відповідність роботи дипломному завданню Кваліфікаційна робота магістра відповідає виданому завданню

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведено огляд сучасних методів розпізнавання графічних образів та принципів організації гетерогенних обчислень на мобільних пристроях. У другому розділі запропоновано модель інтелектуальної системи, у якій конвеєр розпізнавання формалізовано як спрямований ациклічний граф, та побудовано математичну модель гетерогенного розподілу обчислень. У третьому розділі запропоновано метод розпізнавання графічних образів з каскадним переключенням між NPU, GPU та CPU й адаптацією моделей до особливостей української абетки. У четвертому розділі запропоновано програмну реалізацію системи для платформ Android та iOS і проведено експериментальні дослідження, які підтвердили скорочення часу відгуку на 55–77 % та підвищення точності розпізнавання українського тексту на 31,2 %.

4. Позитивні сторони роботи: Запропонована система розпізнавання графічних образів фіскальних документів дозволяє виконувати обробку безпосередньо на мобільному пристрої без залежності від хмарних сервісів,

забезпечує конфіденційність фінансових даних та роботу в автономному режимі, а також досягає часу обробки 764 мс на документ завдяки гетерогенному розподілу обчислень між CPU, GPU та NPU.

5. Негативні сторони роботи: Відсутні.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Відсутні.

7. Відгук про роботу в цілому: В загальному робота виконана на достатньому рівні, містить наукову новизну та практичну цінність. Отримані результати можуть бути впроваджені в мобільних системах з гетерогенною архітектурою обчислювачів для автономної обробки документів.

8. Інші зауваження: Відсутні.

9. Оцінка кваліфікаційної роботи магістра:

Розглянувши позитивні та негативні сторони представленої кваліфікаційної роботи магістра вважаю, що робота заслуговує оцінки «добре» 85.00 (B)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) д.т.н., професор, Мартинюк В.В., завідувач кафедри автоматизації, комп'ютерно-інтегрованих технологій та робототехніки

"1 травня" 2026р.

Зав. кафедри КІС
д-р. філософії Ользі ПАВЛОВІЙ

Макаров Максим Володимирович
ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-24-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

1 травня 2026 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Інтелектуальна комп'ютерна система розпізнавання графічних образів з використанням гетерогенних обчислень
Автор Максим МАКАРОВ
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти другий (магістерський)
Спеціальність 123 Комп'ютерна інженерія
Науковий керівник: д. техн. наук, професор Олег САВЕНКО

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 5,10% і адресується до 43 першоджерел; та системою Anti-Plagiarism складає 10%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

25.04.2026

Завідувач кафедри _____ Ольга ПАВЛОВА
Підпис Ім'я, ПРІЗВИЩЕ

Гарант освітньої програми _____ Олег САВЕНКО
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Олег САВЕНКО
Підпис Ім'я, ПРІЗВИЩЕ