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

КВАЛІФІКАЦІЙНА РОБОТА

Галузь знань 12 Інформаційні технології

Спеціальність 123 –Комп'ютерна інженерія

на тему «Метод синтезу високоефективних систем штучного інтелекту для IoT пристроїв з обмеженими розрахунковими ресурсами»

КвРКП. 180232.11.01.01 ПЗ

Виконав: студент 2 курсу, група КІ2м-22-1 Івчук О.М.
Підпис Ініціали, прізвище

Керівник ст.викладач Іштван С.О.
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КПС, д.т.н., проф.
Т.О. Говорущенко
27 05 2024 р.

Хмельницький, 2024

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

Освітній рівень МАГІСТР

Галузь знань 12 ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

Спеціальність 123 КОМП'ЮТЕРНА ІНЖЕНЕРІЯ

Освітня програма ОСВІТНЬО-НАУКОВА ПРОГРАМА «КОМП'ЮТЕРНА ІНЖЕНЕРІЯ ТА ПРОГРАМУВАННЯ»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

"01" 09 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Івчуку Олександру Миколайовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи)Метод забезпечення кібербезпеки AIoT в умовах збільшення загроз кібератак

Керівник проекту (роботи) Іштван Є.О., ст. викладач

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.01.2024 р. № 1

2. Строк подання студентом проекту (роботи) на кафедру 01.05.2025 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Проаналізувати існуючі методи синтезу ІІІ-систем для IoT-пристроїв

Розробити метод з попередньої фільтрацією зображень

Покращення методу детекції об'єкта в режимі низькопотужної AIoT системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи магістра

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Лисенко С.М., професор кафедри КІС		
Антиплагіат	Нічепорук А.О., доцент кафедри КІС		

7. Дата видачі завдання «_01_»_____09_____2024р.

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	05.09.2023	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	05.10.2023	виконано
3	Робота над розділом 1 – аналіз відомих моделей, методів за темою; постановка задачі	05.11.2023	виконано
4	Робота над розділом 2 – розробка моделей для вирішення поставленої задачі	05.12.2023	виконано
5	Робота над науковою статтею, публікацією на конференцію	25.04.2024	виконано
6	Робота над розділом 3 – розробка методів для вирішення поставленої задачі	15.02.2024	виконано
7	Робота над розділом 4 – проектування та розробка ПЗ для вирішення поставленої задачі, екеперимента льна частина	05.04.2024	виконано
8	Оформлення пояснювальної записки згідно вимог	15.04.2024	виконано
9	Попередній захист ДРМ	18.04.2024	виконано
10	Захист ДРМ на засіданні ЕК	До 23.05.2024	

Студент _____ О.М.Івчук
Підпис Ініціали, прізвище

Керівник роботи _____ С.О. Іштван
Підпис Ініціали, прізвище

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: «Метод синтезу високоефективних систем штучного інтелекту для IoT пристроїв з обмеженими розрахунковими ресурсами»

Автор роботи: Івчук О.М.

Керівник роботи: ст. викладач Іштван Є.О.

Пояснювальна записка: 71 с., 19 рис., 1 табл., 3 дод., 93 джерел.

Ключові слов: стабільність AIoT, шифрування даних, аутентифікація, контроль доступу, захист мереж, виявлення кібератак, машинне навчання, аномалії.

Об’єктом дослідження є Системи ШІ для IoT-пристроїв з обмеженими розрахунковими ресурсами.

Предметом дослідження є методи та алгоритми Методи синтезу нейронних мереж для RISC-V архітектури та фільтрації зображень.

У першому розділі досліджуються методи синтезу ШІ-систем для IoT-пристроїв зосереджені на оптимізації обчислювальних ресурсів та енергоспоживання, щоб забезпечити ефективну роботу штучного інтелекту в умовах обмежених апаратних можливостей. Використання квантової оптимізації, апаратних прискорювачів, легких алгоритмів, граничних обчислень, хмарних сервісів та спеціалізованих платформ розробки дозволяє створювати потужні та ефективні ШІ-рішення для широкого спектра застосувань у сфері Інтернету речей.

У другому розділі досліджуються методи з попередньою фільтрацією даних, з акцентом на контрастне розтягування Викладено теоретичні основи методу попередньої фільтрації даних. Було розглянуто основні принципи, які лежать в основі цього методу, а також математичні моделі, що використовуються для опису процесів фільтрації. Особлива увага приділена аналізу шуму в зображеннях та методам його зменшення, що є критичним для підвищення якості подальшої обробки та аналізу даних. Викладено теоретичні основи методу попередньої

фільтрації даних. Реалізовано прогрманий засіб з вирівнюванням гістограм та перетворенням Лапласа.

Описано детальний алгоритм роботи розробленого методу попередньої фільтрації даних. В алгоритмі крок за кроком описано процеси обробки зображень, включаючи виявлення та усунення шуму, нормалізацію інтенсивностей та підготовку даних для подальшого аналізу. Важливим аспектом алгоритму є його адаптивність до різних типів шуму та здатність забезпечувати високу якість фільтрації незалежно від умов зйомки.

У третьому розділі було проаналізовано загальні підходи до фільтрації та модель RNNoise, спеціально призначену для видалення шуму з аудіо сигналів, та реалізовано на тестових зразках звуку.. Щодо загальних підходів до фільтрації, існують різні методи, що використовуються для видалення шуму з даних. Деякі з найпоширеніших методів включають фільтри на основі частоти, просторове фільтрування та статистичні фільтри. Вибір оптимального методу фільтрації залежить від типу шуму та характеристик даних.

У четвертому розділі проведено експеримент та результати експерименту підтверджують, що векторизація значно покращує продуктивність алгоритмів фільтрації зображень. Зі збільшенням розміру зображення та ядра ефективність векторизації стає більш помітною, що робить її особливо корисною для обробки великих зображень та складних фільтрів. Ці результати можуть бути корисними для розробників алгоритмів обробки зображень, які прагнуть оптимізувати продуктивність своїх програм.

Метою кваліфікаційної роботи магістра є розробка та дослідження методу синтезу високоефективних систем штучного інтелекту (ШІ) для IoT-пристроїв з обмеженими розрахунковими ресурсами. Реалізація нейронної мережі для AIOT системі з демонстрацією фільтрації зображень для зниження рівня шумів і подальшого розпізнавання.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

IoT – Internet of Things (Інтернет речей)

AIoT – Artificial Intelligence of Things (Штучний Інтелект Речей)

ШІ – Штучний Інтелект

IIoT – Industrial Internet of Things (Промисловий Інтернет Речей)

AMQP – Advanced Message Queuing Protocol

CoAP – Constrained Application Protocol

MQTT – Message Queuing Telemetry Transport

HTTP – HyperText Transfer Protocol

SIMD – Single Instruction, Multiple Data

ОКМД – одиночний потік команд, множинний потік даних

GRU – Gated Recurrent Unit

ШПФ – швидке перетворення Фур'є

ВСТУП

Системи Штучного інтелекту речей набувають популярності, викликаючи попит як на більш ефективну електроніку, ефективність якогої вираховується як баланс розрахункової потужності з енергозбереженням, так і на нейронні мережі з певними унікальними особливостями.

Актуальність цієї роботи полягає в її внеску до оптимізації енергоефективних методів обробки даних та детекції об'єктів у системах Інтернету речей (IoT).

Метою кваліфікаційної роботи магістра є розробка та дослідження методу синтезу високоефективних систем штучного інтелекту (ШІ) для IoT-пристроїв з обмеженими розрахунковими ресурсами. Реалізація нейронної мережі на RISC-V контролері для демонстрації можливостей методу.

Об’єктом дослідження є Системи ШІ для IoT-пристроїв з обмеженими розрахунковими ресурсами.

Предметом дослідження є методи Методи оптимізації нейронних мереж для RISC-V архітектури.

Наукова новизна отриманих результатів:

Розроблено метод з попередньою фільтрацією фото даних та доналаштовано і реалізовано нейронні мережі на базі RNNoise та OpenCV під RISC-V, може бути описана наступним чином.

Розробка методу детекції об’єкта в режимі низькопотужної AIoT системи є новаторською через оптимізацію алгоритмів для роботи на обмежених апаратних ресурсах, що зменшує споживання енергії, зберігаючи при цьому високу точність. Більшість існуючих методів обробки зображень та детекції об'єктів розроблені для високопродуктивних систем, тоді як запропонований метод ефективно працює в умовах низької потужності. Інтеграція цих методів з AIoT системами відкриває нові можливості для реального часу моніторингу та автоматизації в різних галузях, таких як сільське господарство, промисловість та розумні міста.

Метод з попередньою фільтрацією фото даних дозволяє відсівати нерелевантні або шумові дані ще на етапі збору, що значно покращує точність

подальшої обробки та детекції об'єктів. Це є новаторським підходом у контексті AIoT систем, де пропускна здатність і енергоефективність мають ключове значення. Попередня фільтрація на ранньому етапі зменшує обсяг даних, які потрібно обробляти основним алгоритмом детекції, що дозволяє економити обчислювальні ресурси та енергію.

Доналаштування та реалізація нейронних мереж на базі RNNoise та OpenCV під RISC-V є інноваційним через адаптацію архітектури RISC-V для виконання нейронних мереж, яка зазвичай використовується на стандартних процесорах або спеціалізованих апаратних прискорювачах. Це відкриває нові можливості для використання відкритої архітектури RISC-V у низькопотужних AIoT системах. Доналаштування нейронних мереж для оптимальної роботи з використанням інструментів RNNoise та OpenCV дозволяє досягнути високої продуктивності на обмежених апаратних ресурсах, що сприяє поширенню використання AIoT систем у різних галузях.

Практична цінність результатів, отриманих у ході розробки методу детекції об’єкта в режимі низькопотужної AIoT системи, методу з попередньою фільтрацією фото даних та доналаштування і реалізації нейронних мереж на базі RNNoise та OpenCV під RISC-V, полягає у наступному.

1) Метод детекції об’єкта в режимі низькопотужної AIoT системи демонструє значну практичну цінність завдяки своїй здатності забезпечувати високу ефективність в умовах обмежених енергетичних ресурсів. Це має велике значення для застосування в пристроях Інтернету речей (IoT), які часто працюють у віддалених або автономних середовищах, де критично важливо мінімізувати споживання енергії. Зокрема, цей метод може бути використаний для моніторингу врожаю в сільському господарстві, забезпечення безпеки в системах "розумного дому", а також у промислових додатках для контролю стану обладнання. Ефективна детекція об'єктів у реальному часі при низькому енергоспоживанні також має важливе значення для носимих пристроїв, таких як розумні окуляри або фітнес-трекери.

2) Метод попередньої фільтрації фото даних забезпечує значне покращення якості вхідних даних, що в свою чергу підвищує точність та ефективність подальшої обробки. Це є особливо важливим для систем, де обробка великого обсягу даних є ресурсомісткою. Наприклад, у медичній діагностиці попередня фільтрація зображень може суттєво зменшити кількість помилково позитивних або помилково негативних результатів, підвищуючи точність діагнозів. У сферах відеоспостереження та безпеки цей метод дозволяє швидше виявляти підозрілі об'єкти чи події, зменшуючи навантаження на аналітичні системи.

3) Доналаштування та реалізація нейронних мереж на базі RNNoise та OpenCV під RISC-V надає практичну цінність завдяки можливості використання відкритої архітектури RISC-V для виконання складних обчислень з нейронними мережами. Це забезпечує зниження витрат на апаратне забезпечення та більшу гнучкість у розробці спеціалізованих AIoT рішень. Наприклад, у системах автоматизації промислових процесів можна впроваджувати інтелектуальні алгоритми без необхідності використання дорогих та енерговитратних апаратних рішень. В освітніх цілях це дозволяє створювати доступні навчальні платформи для студентів, які можуть експериментувати з нейронними мережами та розробляти власні AIoT проекти на базі RISC-V.

За темою кваліфікаційноїроботи магістра опубліковані тези доповіді:
Івчук Олександ. Метод синтезу високоефективних систем штучного інтелекту для IoT пристроїв з обмеженими розрахунковими ресурсами. // Матеріали Весняної студентської конференції "Перспективні Мережеві і Комп’ютерні технології" (ПерСиК 2024). 25 квітня 2024. ХАІ.

Результати цієї роботи сприяють підвищенню ефективності та зниженню вартості реалізації інтелектуальних систем у різних галузях, що робить їх надзвичайно цінними для подальшого розвитку технологій Інтернету речей та штучного інтелекту.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ СИНТЕЗУ ШІ-СИСТЕМ ДЛЯ ІОТ-ПРИСТРОЇВ

З появою та швидким розвитком Інтернету речей (IoT) значно зросла потреба у використанні штучного інтелекту (ШІ) для підвищення функціональності та ефективності IoT-пристроїв. Інтеграція ШІ в IoT дозволяє створювати більш розумні, адаптивні та автономні системи, здатні приймати рішення на основі аналізу даних у реальному часі.

Цей розділ присвячений аналізу існуючих методів синтезу ШІ-систем для IoT-пристроїв, зокрема, інтеграції AI на edge рівні та рівні кінцевих точок. Розглядаються основні підходи, технології та інструменти, що використовуються для розробки таких систем.

1.1 Штучний інтелект у архітектурі IoT

Інтернет речей (IoT) – це мережа фізичних пристроїв, вбудованих з датчиками, програмним забезпеченням та мережевими можливостями, які збирають та обмінюються даними. Ці пристрої можуть бути будь-якими, від побутових приладів до промислових датчиків, і вони можуть спілкуватися один з одним через Інтернет.

Штучний інтелект в Інтернеті речей (AIoT) – це поєднання штучного інтелекту (ШІ) та Інтернету речей (IoT). AIoT використовує ШІ для аналізу даних, зібраних з IoT-пристроїв, для прийняття рішень та покращення продуктивності систем. Класичну архітектуру IoT систем [1] можна побачити на рисунку 1:

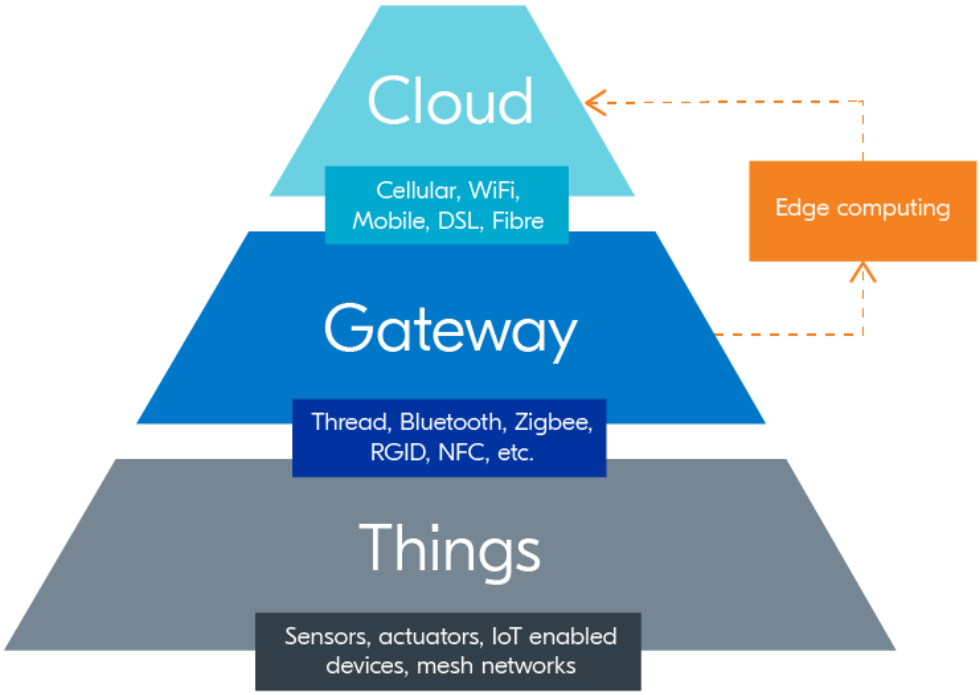


Рисунок 1.1 – Класична архітектура IoT

Основні компоненти архітектури IoT систем:

– EndPoint рівень: датчики та актуратори;

– Edge рівень: шлюзи (Gateways), збір даних з рівня EndPoint, обробка, шифрування та інше, комунікація з Clouds рівнем [2];

– Clouds рівень: зберігання, аналіз, візуалізація даних.

Відповідно на кожному з цих рівней може бути розміщенно інструменти з моделями ШІ. У просторі Інтернету речей існує безліч протоколів [3], які забезпечують зв'язок між пристроями та центральними системами. Кожен з цих протоколів має свої особливості та сфери застосування, що робить їх придатними для різних сценаріїв використання.

HTTP (HyperText Transfer Protocol) є одним із найпоширеніших протоколів для передачі даних у веб-браузери. Хоча він не був розроблений спеціально для IoT, HTTP все ще використовується в багатьох IoT-додатках, особливо тих, що потребують взаємодії з веб-сервісами [5]. Однак, через його відносно високе споживання ресурсів, HTTP не завжди є оптимальним вибором для пристроїв з обмеженими обчислювальними та енергетичними можливостями.

MQTT (Message Queuing Telemetry Transport) – це легкий протокол обміну повідомленнями, спеціально розроблений для пристроїв з обмеженими ресурсами та мереж з високими затримками і обмеженою пропускною здатністю. MQTT працює за принципом "публікація-підписка", що дозволяє зменшити навантаження на мережу та підвищити ефективність передачі даних. Цей протокол широко використовується в IoT-додатках, таких як дистанційний моніторинг та контроль, де важлива швидка та надійна передача невеликих обсягів даних.

CoAP (Constrained Application Protocol) – це спеціально розроблений протокол для роботи в IP-мережах з обмеженою пропускною здатністю. CoAP використовує модель клієнт-сервер і призначений для пристроїв з обмеженими ресурсами, таких як сенсори і мікроконтролери. Протокол підтримує надійну передачу даних через вбудовані механізми підтвердження та повторної передачі. CoAP також підтримує стислий формат повідомлень, що зменшує обсяг переданих даних і знижує навантаження на мережу.

AMQP (Advanced Message Queuing Protocol) – це протокол обміну повідомленнями, який забезпечує надійну та гнучку передачу даних. AMQP підтримує складні черги повідомлень, маршрутизацію, гарантії доставки і транзакції, що робить його ідеальним для використання у великомасштабних розподілених системах, де важлива висока надійність передачі даних. Хоча AMQP менш поширений у типових IoT-додатках через свою складність, він знаходить застосування в критичних системах, де важлива надійність і гарантія доставки повідомлень.

Безпека є однією з найважливіших проблем в AIoT. AIoT-пристрої можуть бути вразливими до кібератак, які можуть призвести до крадіжки даних, пошкодження обладнання або порушення роботи систем.

Штучний інтелект (ШІ) в архітектурі Інтернету речей (IoT) грає ключову роль у забезпеченні інтелектуального аналізу даних [6], автоматизації процесів та покращенні прийняття рішень. Інтеграція ШІ з IoT дозволяє створювати розумні

системи, які можуть не тільки збирати та обробляти дані, але й робити висновки та виконувати дії на основі цих даних.

Таким чином маємо основні компоненти архітектури IoT з ШІ:

– пристрої та сенсори;

– комунікаційна інфраструктура;

– протоколи зв'язку;

– шлюзи (Gateways);

– хмара та обчислювальна інфраструктура.

Датчики збирають дані з навколишнього середовища (наприклад, температура, вологість, тиск, рух). Актуратори виконують дії на основі рішень, прийнятих системою (наприклад, регулювання температури, вмикання освітлення). Протоколи зв'язку на прикладі Wi-Fi, Bluetooth, Zigbee, LoRa, 5G, забезпечують передачу даних між пристроями і центральними системами [7]. А шлюзи (Gateways) з'єднують пристрої IoT з мережею Інтернет та забезпечують попередню обробку даних, зменшуючи подальше навантаження на хмару.

Класичний підхід це Clouds AI, коли на великих обчислювальних потужностях можна навчати та використовувати нейронні мережі, базуючись з даних, отриманих з EndPoint та Edge рівнев.

ШІ алгоритми та моделі:

– машинне навчання (ML) використовується для аналізу даних, виявлення патернів та прогнозування;

– глибинне навчання (DL) використовується для складних завдань, таких як розпізнавання образів та обробка природної мови;

– обробка природної мови (NLP): Застосовується для аналізу текстових даних та взаємодії з користувачами.

Взаємодія між компонентами відбувається наступним чином - у типовій IoT системі дані з сенсорів збираються і передаються через комунікаційні мережі до шлюзів. Шлюзи можуть виконувати попередню обробку даних, фільтруючи непотрібні або неактуальні дані, а потім надсилають їх до хмарних сервісів для більш глибокого аналізу. У хмарі застосовуються ШІ алгоритми, які можуть

аналізувати великі обсяги даних, навчатися на них і робити прогнози. Наприклад, у сільському господарстві ШІ може прогнозувати врожайність на основі даних про погоду та стан ґрунту, а в промисловості – передбачати можливі поломки обладнання. Edge Computing дозволяє виконувати частину обчислень безпосередньо на місці збору даних, що особливо важливо для застосувань, де затримки є критично важливими. Наприклад, у автономних транспортних засобах рішення повинні прийматися в реальному часі для забезпечення безпеки.

Переваги інтеграції ШІ в IoT:

– підвищена ефективність, оскільки автоматизація процесів і прийняття рішень на основі аналізу даних дозволяє знизити витрати і підвищити продуктивність;

– покращене прийняття рішень, враховуючи, що ШІ може аналізувати великі обсяги даних і виявляти закономірності, які важко помітити людині;

– реальний час, а саме використання Edge Computing дозволяє обробляти дані і реагувати на них майже миттєво;

– скорочення затримок, оскільки обробка даних на периферії мережі зменшує затримки, що критично важливо для додатків з високими вимогами до швидкості реакції;

– адаптивність ШІ-систем, які можуть адаптуватися до змін навколишнього середовища і умов експлуатації, що забезпечує більш гнучке управління.

Інтеграція штучного інтелекту в архітектуру IoT відкриває нові можливості для створення інтелектуальних систем, здатних самостійно аналізувати дані і приймати рішення. Це дозволяє підвищити ефективність, знизити витрати і забезпечити більш швидке і точне реагування на події. У поєднанні з технологіями Edge Computing, ШІ дозволяє забезпечити обробку даних у реальному часі, що є критично важливим для багатьох застосувань, від промислового Інтернету речей до автономних транспортних засобів та розумних будинків.

1.2 Застосування моделей ШІ на Edge та EndPoint рівнях AIoT систем

Впровадження штучного інтелекту (ШІ) на рівнях Edge та EndPoint в AIoT (Artificial Intelligence of Things) системах надає значні переваги для обробки даних та прийняття рішень безпосередньо на пристроях або в їхній безпосередній близькості. Це забезпечує низькі затримки, зменшує залежність від центральних серверів та знижує вимоги до пропускної здатності мережі. Нижче розглянемо детальніше застосування моделей ШІ на цих рівнях.

1.2.1 Розміщення моделей ШІ на крайовому рівні (Edge AI)

Edge AI передбачає виконання обчислень та аналізу даних безпосередньо на периферійних пристроях або на локальних серверах, які знаходяться близько до місця збору даних. Це дозволяє обробляти дані в реальному часі, що є критично важливим для додатків, де затримки є неприпустимими.

У промисловому Інтернеті речей (IIoT) Edge AI використовується для моніторингу стану обладнання, виявлення аномалій та прогнозування поломок. Це дозволяє проводити профілактичне обслуговування та зменшувати час простою обладнання. В системах розумного міста Edge AI допомагає в управлінні трафіком, моніторингу якості повітря та управлінні енергоспоживанням. Наприклад, камери спостереження з вбудованими алгоритмами розпізнавання осіб можуть швидко ідентифікувати підозрілих осіб або транспортні засоби. У сільському господарстві системи Edge AI можуть аналізувати зображення з дронів або сенсорів для визначення стану врожаю, виявлення шкідників або контролю за іригацією.

Для реалізації Edge AI використовуються апаратні прискорювачі, такі як NVIDIA Jetson, Google Coral, Intel Movidius, які забезпечують високу продуктивність при низькому енергоспоживанні. Також використовуються платформи та інструменти, такі як TensorFlow Lite, RNNoise, які дозволяють розгортати моделі ШІ на периферійних пристроях.

Edge AI відкриває широкі можливості для покращення ефективності та продуктивності в різних галузях, забезпечуючи швидке та ефективне прийняття рішень безпосередньо на місці збору даних.

Застосування моделей ШІ на Edge IoT можна класифікувати за кількома ключовими критеріями, що охоплюють функціональність, галузі застосування, типи оброблюваних даних та рівень обчислювальної складності. Така класифікація дозволяє детально описати різні аспекти використання Edge AI у IoT середовищах та їхні технічні особливості. Перший аспект класифікації стосується функціональності моделей ШІ. До цієї категорії відносяться детекція та розпізнавання, прогнозування та аналітика, а також оптимізація та управління.

Детекція та розпізнавання включають задачі, пов’язані з ідентифікацією об'єктів, людей, транспортних засобів, а також обробкою голосових команд та аналізом відеопотоків у реальному часі. Наприклад, системи безпеки в розумних містах використовують алгоритми розпізнавання облич для швидкого виявлення підозрілих осіб. Прогнозування та аналітика орієнтовані на задачі, такі як прогнозування поломок обладнання або аналіз поведінкових патернів користувачів. В промисловому середовищі, це дозволяє забезпечити профілактичне обслуговування та мінімізувати час простою. Оптимізація та управління включають задачі, що стосуються оптимізації споживання енергії, управління ресурсами, а також оптимізації виробничих процесів та логістики. У розумних будинках, наприклад, такі системи можуть автоматично регулювати температуру і освітлення для підвищення енергоефективності.

Другий критерій класифікації – галузі застосування моделей ШІ. Це включає промисловий Інтернет речей (с), розумні міста, сільське господарство, розумні будинки, медицину та автономні транспортні засоби. У промисловому Інтернеті речей (IIoT) моделі ШІ використовуються для моніторингу стану обладнання та прогнозного обслуговування. Це дозволяє значно знизити час простою та підвищити ефективність виробництва. В розумних містах ШІ допомагає управляти трафіком, моніторити якість повітря та забезпечувати безпеку через системи відеоспостереження з розпізнаванням облич та

транспортних засобів. Наприклад, такі системи можуть автоматично виявляти порушення правил дорожнього руху.

У сільському господарстві Edge AI аналізує зображення з дронів або сенсорів для визначення стану врожаю, виявлення шкідників та контролю за іригацією, що дозволяє оптимізувати використання ресурсів і підвищити врожайність. Розумні будинки використовують моделі ШІ для управління домашніми пристроями, моніторингу безпеки та реагування на голосові команди. Наприклад, системи управління енергією можуть автоматично регулювати споживання електрики на основі аналізу поведінки мешканців. Для автономних транспортних засобів моделі ШІ забезпечують виявлення перешкод, розпізнавання дорожніх знаків та прийняття рішень про керування автомобілем, забезпечуючи безпечне та ефективне пересування.

Третій аспект класифікації базується на типі даних, що обробляються моделями ШІ на Edge IoT. Це можуть бути візуальні дані, аудіодані, сенсорні дані та текстові дані. Візуальні дані включають зображення та відео, які обробляються для розпізнавання об'єктів, осіб, транспортних засобів та аналізу поведінки. Наприклад, камери спостереження можуть виявляти рух і сигналізувати про потенційну загрозу. Аудіодані використовуються для розпізнавання голосових команд та аналізу звуків навколишнього середовища. Голосові асистенти в розумних будинках можуть виконувати команди на основі голосових інструкцій користувача. Сенсорні дані охоплюють інформацію про температуру, вологість, тиск та інші параметри, які збираються різними сенсорами. Наприклад, сенсори в сільськогосподарських системах можуть відстежувати вологість ґрунту та оптимізувати полив. Текстові дані включають логи систем та текстові команди, які аналізуються для моніторингу та управління системами. Наприклад, аналіз текстових даних може використовуватися для виявлення аномалій у логах серверів.

Останній критерій класифікації стосується рівня обчислювальної складності моделей ШІ. Вони можуть бути класифіковані на легкі моделі (Lightweight Models), важкі моделі (Heavyweight Models) та гібридні рішення. Легкі моделі

(наприклад, MobileNet, SqueezeNet, TinyML) призначені для роботи на пристроях з обмеженими ресурсами, забезпечуючи високу продуктивність при низькому енергоспоживанні. Вони підходять для задач, де необхідна швидка обробка даних на місці. Важкі моделі (наприклад, ResNet, VGG, YOLO) забезпечують більш точні результати, але вимагають більше обчислювальних ресурсів і енергії. Вони зазвичай використовуються у випадках, коли необхідна висока точність обробки даних. Гібридні рішення комбінують легкі і важкі моделі для оптимізації продуктивності та енергоефективності. Це дозволяє збалансувати швидкість і точність обробки даних, забезпечуючи оптимальне використання ресурсів.

1.2.2 Розміщення моделей ШІ на рівні кінцевих точок (EndPoint AI)

EndPoint AI передбачає виконання обчислень безпосередньо на кінцевих пристроях, таких як сенсори, камери, розумні годинники або інші IoT пристрої. Це забезпечує обробку даних на місці, що дозволяє значно зменшити затримки та потреби в передачі великих обсягів даних.

У сфері носимих пристроїв розумні годинники або фітнес-трекери можуть аналізувати фізичну активність, моніторити стан здоров’я та навіть прогнозувати медичні події на основі зібраних даних. У системах розумного дому розумні домашні пристрої, такі як термостати, камери спостереження або голосові асистенти, можуть виконувати обробку даних та реагувати на зміну умов в реальному часі. Наприклад, розпізнавання голосу та управління пристроями через голосові команди.

В автономних транспортних засобах датчики та камери в автомобілях можуть обробляти дані для виявлення перешкод, розпізнавання знаків або прийняття рішень про керування автомобілем. Для реалізації EndPoint AI важливо використовувати ефективні та легкі моделі ШІ, такі як MobileNet, SqueezeNet, TinyML, які здатні працювати на пристроях з обмеженими ресурсами.

Інструменти для розгортання, такі як TensorFlow Lite, дозволяють оптимізувати моделі для роботи на мобільних та вбудованих пристроях.

У медицині EndPoint AI застосовується для моніторингу стану здоров'я пацієнтів, аналізу фізичної активності та прогнозування медичних подій. Наприклад, носимі пристрої можуть постійно моніторити серцевий ритм і попереджати про можливі проблеми. Ці технології відкривають широкі можливості для застосування ШІ безпосередньо на кінцевих пристроях, що значно підвищує швидкість і ефективність обробки даних, а також зменшує залежність від центральних серверів.

Застосування моделей ШІ на Edge та EndPoint рівнях AIoT систем забезпечує значні переваги у вигляді низьких затримок, зменшення навантаження на мережу та підвищення енергоефективності. Використання апаратних прискорювачів та легких моделей ШІ дозволяє впроваджувати інтелектуальні функції безпосередньо на пристроях, що відкриває нові можливості для різних галузей, включаючи промисловість, розумні міста, сільське господарство та носимі пристрої.

1.3 Методи синтезу ШІ-систем для IoT-пристроїв та підходи до реалізації

Методи_синтезу ШІ-систем для IoT-пристроїв - це методи, які використовуються для створення ефективних та надійних систем штучного інтелекту (ШІ) для пристроїв Інтернету речей (IoT). Ці методи повинні враховувати обмежені розрахункові ресурси IoT-пристроїв, такі як низька продуктивність процесора, мала пам'ять та обмежене енергоспоживання.

Деякі з поширених методів синтезу ШІ-систем для IoT-пристроїв включають:

– машинне навчання. Цей метод використовує алгоритми, які навчаються на даних, щоб робити прогнози або приймати рішення. Він добре підходить для

задач, де є багато даних, наприклад, прогнозування споживання енергії або виявлення несправностей;

– глибоке навчання. Цей метод є підтипом машинного навчання, який використовує нейронні мережі для навчання на даних. Він добре підходить для задач, які потребують складного аналізу даних, наприклад, розпізнавання зображень або обробки природної мови;

– еволюційні алгоритми, які використовують принципи природного відбору для оптимізації ШІ-систем. Цей тип добре підходить для задач, де важко визначити чіткі правила або функції, наприклад, оптимізація маршрутів для безпілотних транспортних засобів;

– символічне міркування, яке використовує символи та логічні правила для представлення та обробки інформації. Добре підходить для задач, які потребують чіткого розуміння правил та обмежень, наприклад, планування дій для роботів.

Також у наступних підрозділах розглянемо кілька основних підходів, які використовуються для синтезу систем AIoT:

1.3.1 Квантова оптимізація та зменшення моделей

Для впровадження ШІ в IoT-пристрої часто використовуються методи квантової оптимізації (quantization) та зменшення (pruning) моделей. Квантова оптимізація зменшує розрядність параметрів нейронних мереж, що дозволяє значно скоротити вимоги до пам'яті та обчислювальної потужності. Зменшення моделей передбачає видалення непотрібних або малозначущих параметрів, що також сприяє підвищенню ефективності без значної втрати точності.

Квантова оптимізація, при зменшенні розмірності параметрів нейронних мереж, перетворює їх у менш точні, але більш компактні форми. В основі квантової оптимізації лежить ідея заміни точних значень параметрів (наприклад, з плаваючою крапкою) на більш прості, наприклад, цілі числа або обмежені розрядність бітів. Це дозволяє значно зменшити вимоги до пам'яті та

обчислювальної потужності, що особливо важливо для IoT-пристроїв з обмеженими ресурсами.

Зменшення моделей передбачає видалення непотрібних або малозначущих параметрів з нейронних мереж. Параметри, які не вносять суттєвого внеску у вихід моделі, можуть бути відкинуті, зменшуючи тим самим розмір мережі та складність обчислень. При цьому важливо зберегти точність моделі настільки, наскільки це можливо. Зменшення моделей дозволяє підвищити ефективність використання ресурсів, знизити вимоги до пам'яті та обчислювальної потужності, а також прискорити процес навчання та інференції.

1.3.2 Використання апаратних прискорювачів

Апаратні прискорювачі, такі як графічні процесори (GPU), програмовані вентильні матриці (FPGA) та спеціалізовані інтегральні схеми для штучного інтелекту (ASIC), значно підвищують ефективність обробки ШІ алгоритмів на IoT-пристроях. Вони дозволяють виконувати обчислення швидше та з меншою витратою енергії порівняно з традиційними процесорами.

Приклади використання апаратних прискорювачів для ШІ в IoT:

– розпізнавання зображень, IoT-камери з GPU або FPGA можуть використовуватися для розпізнавання об'єктів, людей або тексту в реальному часі, наприклад, для систем безпеки, моніторингу трафіку або віртуальних помічників;

– аналіз даних датчиків IoT-пристрої з ASIC або GPU можуть аналізувати дані з датчиків, такі як температура, тиск або вібрація, для прогнозування несправностей, оптимізації продуктивності або виявлення аномалій в промислових системах або медичних приладах;

– голосове керування, IoT-пристрої з FPGA або GPU можуть використовуватися для розпізнавання мови та реагування на голосові команди, наприклад, для розумних будинків, автомобілів або побутової техніки;

– прогнозна аналітика, IoT-пристрої з ASIC або GPU можуть використовуватись для аналізу історичних даних та прогнозування майбутніх

подій, наприклад, для оптимізації енергоспоживання, планування технічного обслуговування або прогнозування попиту в роздрібній торгівлі;

– машинне навчання на крайовому (edge) рівні, IoT-пристрої з FPGA або GPU можуть виконувати машинне навчання на даних, зібраних локально, без необхідності надсилати їх у хмару, забезпечуючи більшу конфіденційність, безпеку та зменшуючи затримку.

1.3.3 Використання легких та ефективних алгоритмів

Для IoT-пристроїв розробляються спеціальні легкі моделі машинного навчання, такі як MobileNet, SqueezeNet та TinyML. Ці моделі оптимізовані для роботи в умовах обмежених ресурсів і здатні виконувати завдання розпізнавання зображень, обробки мови та інші функції ШІ з прийнятною точністю.

Легкість цих моделей не супроводжується втратою точності. Навпаки, вони здатні виконувати різноманітні завдання машинного навчання з прийнятною точністю, включаючи розпізнавання зображень, обробку мови та інші функції штучного інтелекту. При цьому вони ефективно використовують обмежені ресурси, що робить їх ідеальними для впровадження на IoT-пристроях з невеликим обсягом пам'яті та обчислювальною потужністю. Ці легкі та ефективні алгоритми машинного навчання є ключовими для подальшого розвитку інтелектуальних систем в IoT-сегменті, забезпечуючи оптимальне використання обмежених ресурсів та забезпечуючи високу функціональність пристроїв.

1.3.4 Впровадження граничних обчислень (Edge Computing)

Граничні обчислення передбачають обробку даних безпосередньо на IoT-пристроях або на близько розташованих серверах, що зменшує затримки та навантаження на центральні обчислювальні ресурси. Це дозволяє виконувати аналіз даних в реальному часі, що є критично важливим для багатьох застосувань IoT.

1.3.5 Використання хмарних сервісів та розподілених обчислень

Хмарні сервіси дозволяють переносити важкі обчислення на віддалені сервери, залишаючи IoT-пристроям лише попередню обробку даних та передачу результатів. Це дає можливість використовувати потужні моделі ШІ, не обмежуючись апаратними ресурсами окремих пристроїв.

1.3.6 Інтеграція з платформами розробки ШІ

Інструменти та платформи, такі як TensorFlow Lite, RNNoise та Edge AI, надають розробникам засоби для створення та оптимізації моделей ШІ спеціально для IoT-пристроїв. Вони спрощують процес адаптації та впровадження ШІ-рішень, забезпечуючи високу продуктивність і енергоефективність.

Окрім вищезазначених кроків, метод синтезу ШІ може також включати такі етапи:

– збір даних, необхідних для навчання та тестування ШІ-системи;

– очищення та підготовка даних для використання в ШІ-системі;

– перетворення даних у формат, який може використовуватися ШІ-системою;

– оцінка продуктивності ШІ-системи на тестових даних;

– розгортання ШІ-системи в реальному середовищі.

Метод синтезу ШІ - це постійно розвивається область, і з'являються нові методи та інструменти. Це важливий напрямок досліджень в області штучного інтелекту, який має особливе значення для IoT-пристроїв, де пам'ять і обчислювальні ресурси обмежені. Ці методи дозволяють зменшити розмір нейронних мереж, що робить їх більш економними з точки зору пам'яті та обчислювальних ресурсів, не жертвуючи при цьому значною точністю.

1.4 Постановка задачі

Метою роботи є розробка методів синтезу ШІ-систем для IoT-пристроїв з обмеженими ресурсами, оптимізація енергоефективності методів обробки даних та детекції об'єктів в IoT-системах і реалізація нейронної мережі на RISC-V контролері для виявлення можливостей методу.

Задачі дослідження включають аналіз існуючих методів оптимізації нейронних мереж для RISC-V архітектури, розробку методу попередньої фільтрації фотоданих для IoT-систем, налаштування та реалізацію нейронних мереж на базі RNNoise та OpenCV під Edge AI ріень та дослідження енергоефективності та точності запропонованого методу.

Наукова новизна полягає у розробці методів детекції об'єктів в низькопотужних AIoT системах, запропонованому методі попередньої фільтрації фотоданих та налаштуванні та реалізації нейронних мереж на базі RNNoise та OpenCV під RISC-V для використання в низькопотужних AIoT системах. Робота спрямована на підвищення ефективності та зниження енергоспоживання IoT-систем з ШІ, забезпечення високої точності детекції об'єктів в умовах обмежених ресурсів і зниження вартості реалізації інтелектуальних систем, що становить її практичну цінність.

Очікувані результати включають розробку та дослідження методів синтезу ШІ-систем для IoT-пристроїв з обмеженими розрахунковими ресурсами, реалізацію нейронної мережі на RISC-V контролері, отримання даних про енергоефективність та точність запропонованого методу.

1.5 Висновки

Зважаючи на швидкий розвиток інтернету речей (IoT) та штучного інтелекту (ШІ), важливо визначити оптимальні методи синтезу ШІ-систем, які враховують специфіку IoT-пристроїв. Аналіз існуючих методів показує, що застосування ШІ в архітектурі IoT може значно підвищити їхню продуктивність,

надійність та ефективність. Наразі активно вивчається застосування моделей ШІ на рівнях Edge та EndPoint в системах AIoT. Це дозволяє виконувати обробку даних непосредственно на пристроях IoT, що робить їх більш автономними та ефективними. Однак, необхідно враховувати різні виклики, такі як обмежені ресурси пристроїв IoT (обсяг пам'яті, потужність обчислення тощо), а також вимоги до енергоефективності.

Методи синтезу ШІ-систем для IoT-пристроїв зосереджені на оптимізації обчислювальних ресурсів та енергоспоживання, щоб забезпечити ефективну роботу штучного інтелекту в умовах обмежених апаратних можливостей. Використання квантової оптимізації, апаратних прискорювачів, легких алгоритмів, граничних обчислень, хмарних сервісів та спеціалізованих платформ розробки дозволяє створювати потужні та ефективні ШІ-рішення для широкого спектра застосувань у сфері Інтернету речей.

Крім методів синтезу ШІ-систем, також важливо використовувати відповідні програмні та апаратні платформи для IoT-пристроїв. RNNoise та OpenCV - це два популярні програмні фреймворки, які можуть бути використані для розробки ШІ-систем для IoT-пристроїв. RISC-V - це відкрита архітектура процесора, яка стає все більш популярною для IoT-пристроїв, завдяки своїй низькій вартості, низькому енергоспоживанню та високій продуктивності.забезпечити безпеку AIoT систем під час їх впровадження та роботи.

Постановка задачі вивчення та розробки оптимальних методів синтезу ШІ-систем для IoT-пристроїв визначається необхідністю розв'язання цих викликів. Подальші дослідження у цій області мають на меті розробку нових алгоритмів та методів, які б дозволили досягти більшої продуктивності, масштабованості та енергоефективності ШІ-систем для IoT-пристроїв.

Таким чином методи синтезу ШІ-систем охоплюють різні підходи та технології, що дозволяють створювати інтелектуальні системи. Одним з основних методів є машинне навчання (Machine Learning, ML), яке включає супервізоване навчання для прогнозування та класифікації на основі мічених даних,

несупервізоване навчання для аналізу даних без міток з метою виявлення прихованих закономірностей, та навчання з підкріпленням, що дозволяє створювати агентів, які навчаються через винагороду та покарання.

Глибоке навчання (Deep Learning) використовує нейронні мережі з багатьма шарами (глибокі нейронні мережі) для автоматичного виділення ознак з даних. Спеціалізовані архітектури, такі як згорткові нейронні мережі (CNN) для обробки зображень та рекурентні нейронні мережі (RNN) для обробки послідовностей, є ключовими у цій сфері.

Еволюційні алгоритми та генетичне програмування застосовують принципи природного відбору для оптимізації та створення адаптивних систем. Експертні системи та системи на основі правил використовують накопичені знання і правила для вирішення складних завдань на основі логіки. Обробка природної мови (Natural Language Processing, NLP) включає методи для розуміння та генерації людської мови, а фаззіві логіка дозволяє приймати рішення в умовах невизначеності. Байєсові мережі використовуються для моделювання ймовірнісних залежностей. Кожен з цих методів має свої переваги та недоліки, і вибір конкретного підходу залежить від задачі, доступних даних та вимог до продуктивності системи. AIoT — це інтеграція штучного інтелекту (ШІ) з Інтернетом речей (IoT), що створює нові можливості для автоматизації та інтелектуального аналізу даних.

2 АДАПТАЦІЯ ТА ПОКРАЩЕННЯ МЕТОДУ З ПОПЕРЕДНЬОЮ ФІЛЬТРАЦІЄЮ ДАНИХ

У сучасних системах обробки зображень важливою складовою є попередня фільтрація, яка допомагає підготувати зображення до подальшого аналізу та обробки. Цей розділ присвячений адаптації та покращенню методів фільтрації зображень, зокрема контрасного розтягування, медіанного фільтра та методів глобальної бінаризації. Представлено результати тестування алгоритмів фільтрації, які демонструють ефективність та продуктивність в різних умовах. Розглянуто контрасне розтягування, як метод, що дозволяє збільшити контраст зображення шляхом розширення діапазону інтенсивностей пікселів. Це підвищує видимість деталей, особливо в зображеннях з низьким контрастом. Також розглянуто медіанний фільтр, як ефективний метод зменшення шуму, який зберігає різкі межі об'єктів на зображенні.

2.1 Теоретичні основи методу.

Покращення якості зображення документів включає процеси, необхідні для досягнення однорідного фону та високої якості друкованих або рукописних зображень. Ця процедура спрямована на поліпшення читабельності тексту у документах шляхом зменшення шуму. Основні фактори, які спричиняють погіршення якості зображення, включають низьку контрастність, нерівномірне освітлення фону, ефекти тіней, пошкоджені символи та шум на чорних межах.

Для ефективного прибирання цих артефактів з документів було розроблено кілька методів обробки зображень, що втілені у середовищі OpenCV. Наприклад методи підвищення контрастності та вирівнювання гістограми або використання медіанного фільтру для прибирання шуму.

Розглянемо детальніше кожен з них. Починаючи з вирівнювання гістограми, ці методи дозволяють збільшити контрастність зображення та зробити фон більш

однорідним. Вирівнювання гістограми перерозподіляє значення інтенсивності пікселів, що допомагає виділити текст на фоні.

Розрахунок гістограми: Спочатку функція розраховує гістограму зображення. Гістограма - це графік, який показує, скільки разів кожен рівень яскравості зустрічається в зображенні.

Визначення кумулятивної гістограми: Далі функція визначає кумулятивну гістограму. Кумулятивна гістограма - це графік, який показує, скільки пікселів в зображенні мають яскравість меншу або дорівнює певному значенню.

Перетворення пікселів: Нарешті, функція перетворює кожен піксель зображення на основі його яскравості та кумулятивної гістограми. Це робиться таким чином, щоб розподіл пікселів по яскравості був більш рівномірним.

Ось кілька прикладів того, як cv2.equalizeHist(image) може покращити зображення:

– зображення з низьким контрастом, тоді більшість пікселів будуть мати схожі значення яскравості. cv2.equalizeHist(image) може розподілити пікселі більш рівномірно по яскравості, що може покращити видимість деталей;

– зображення з шумом може призвести до того, що на зображенні з'являться пікселі з випадковими значеннями яскравості. cv2.equalizeHist(image) може зменшити вплив шуму, роблячи зображення більш чітким;

– зображення з пересвітленням або недосвітленням, тоді деякі пікселі можуть бути занадто темними або занадто світлими, щоб на них можна було розрізнити деталі. cv2.equalizeHist(image) може покращити розподіл яскравості, роблячи деталі на зображенні більш видимими;

Важливо зазначити, що cv2.equalizeHist(image) не завжди покращує зображення. У деяких випадках це може призвести до втрати деталей або появи артефактів.

Медіанний фільтр - це нелінійний алгоритм обробки зображень, який широко використовується для зменшення шуму, зберігаючи при цьому чіткі краї та деталі зображення. На відміну від лінійних фільтрів, таких як згладжування з середнім значенням, які розмивають краї, медіанний фільтр діє локально,

зберігаючи чіткі межі об'єктів. Медіанний фільтр працює ковзає вікном фіксованого розміру по зображенню. Розмір вікна зазвичай визначається непарним числом, наприклад 3x3, 5x5 або 7x7. У кожному положенні вікна всі пікселі в межах вікна сортуються за їх значеннями яскравості. Після сортування знаходиться медіанне значення - значення, яке знаходиться посередині відсортованого списку. Центральний піксель вікна замінюється медіанним значенням. Вікно фільтра переміщується на один піксель в заданому напрямку (наприклад, вправо, вниз або по діагоналі) і процес повторюється з першого кроку.

Розглянемо переваги медіанного фільтра. Медіанний фільтр добре видаляє імпульсний шум (наприклад, сольові та перцеві шуми) та шуми з випадковими значеннями. На відміну від лінійних фільтрів, медіанний фільтр не розмиває краї, а зберігає їх чіткість. Медіанний фільтр може зберігати дрібні деталі зображення, такі як текстури та тонкі лінії.

Окрему увагу слід приділити недолікам. При використанні великого вікна фільтра медіанний фільтр може розмивати дрібні деталі зображення. Медіанний фільтр не є ефективним для видалення гауссового шуму, який має гладку природу.

На рисунку 2.1 наведено приклад програмної реалізації медіанного фільтра на мові програмування Python.

```
143    # Застосування медіанного фільтру
144    median_filtered = cv2.medianBlur(image, 5)
145
146    # Збереження результату
147    cv2.imwrite('document_median.jpg', median_filtered)
```

Рисунок 2.1 – Застосування медіанного фільтра

У прикладі програми на Python використовується функція cv2.medianBlur() для застосування медіанного фільтру до зображення. Розмір вікна фільтра встановлено на 5, що означає, що вікно 5x5 буде використовуватися для

розрахунку медіанного значення для кожного пікселя. Результат фільтрації зберігається в змінній median_filtered і потім записується у файл з ім'ям document_median.jpg.

Медіанний фільтр - це цінний інструмент для зменшення шуму з зображень, зберігаючи при цьому чіткі краї та деталі. Він особливо корисний для видалення імпульсного шуму та шуму з випадковими значеннями.

Також існують методи глобальної бінаризації, приклад реалізації якої зображено на рисунку 2.2

```
150  # Глобальна бінаризація методом Оцу
151  _, global_thresh = cv2.threshold(image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)
152
153  # Локальна бінаризація методом Ніблека
154  niblack_thresh = cv2.ximgproc.niBlackThreshold(image, 255, cv2.THRESH_BINARY, 11, -0.2)
155
156  # Збереження результатів
157  cv2.imwrite('document_global_thresh.jpg', global_thresh)
158  cv2.imwrite('document_niblack_thresh.jpg', niblack_thresh)
```

Рисунок 2.2 – приклад методу глобальної бінаризації

Глобальна бінаризація, наприклад метод Оцу, встановлює один поріг для всього зображення. Локальна бінаризація, такі як методи Соволі, Ніблека та Бернсена, встановлюють поріг для кожного локального регіону зображення, що допомагає боротися з проблемами, такими як низька контрастність, нерівномірне освітлення та випадковий шум.

Метод Оцу (англ. Otsu's method), названий на честь японського комп'ютерного науковця Нобуюкі Оцу, є алгоритмом автоматичного бінаризації зображень. Бінаризація - це процес перетворення кольорового або градаційного зображення в чорно-біле, де кожен піксель зображується як чорний (0) або білий (1). Метод використовується для визначення оптимального порога інтенсивності пікселів, який найкраще розділяє зображення на два класи: фон і об'єкт. Розглянемо основні етапи роботи метода:

– розраховування гістограми зображення, яка показує кількість пікселів з кожним значенням інтенсивності;

– міжкласове розділення, для кожного можливого значення порога інтенсивності (від 0 до 255) обчислюються два значення;

– внутрішньокласова дисперсія, розраховується сума дисперсій у двох класах (фоні та об'єкті) для даного порога;

– міжкласова дисперсія, яка є різницею між загальною дисперсією зображення та внутрішньокласовою дисперсією.

– вибір оптимального порога, такого, який максимізує міжкласову дисперсію, обирається як оптимальний поріг бінаризації.

В цілому метод Оцу має ряд переваг, його алгоритм є простим і зрозумілим та він може ефективно розділяти зображення з різними типами фону та об'єктів. Метод стійкий до шуму та варіацій освітлення.

Серед недолікив методу Оцу слід зазначити чутливість до шуму в зображенні, що може призвести до неточного визначення порога та необхідність гістограми для роботи алгоритму, що може бути обчислювально дорогим для великих зображень. Також метод не завжди може дати оптимальний результат для зображень з складними фонами або об'єктами.

Метод Оцу є одним з найпоширеніших та найефективніших методів бінаризації зображень. Він використовується в різних сферах, таких як комп'ютерний зір, обробка зображень, розпізнавання образів та машинне навчання.

Процес наскрізного поглядання зображення виникає, коли інформація з оборотної сторони документа просвічується на передню сторону під час копіювання або іншого методу розмноження. На рисунку 1, взятому з відкритих даних з розпродажу архіву УПА на аукціоні, показано ефект просвічування зображення, що має шум.

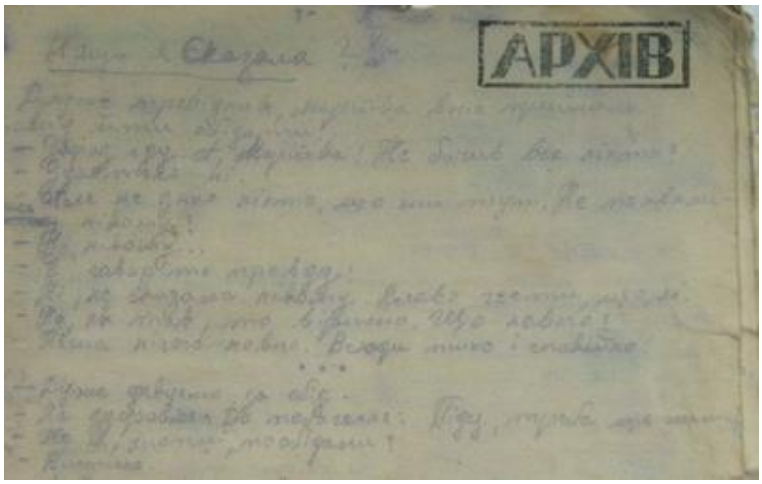


Рисунок 2.1 – Оригінальне зашумлене зображення

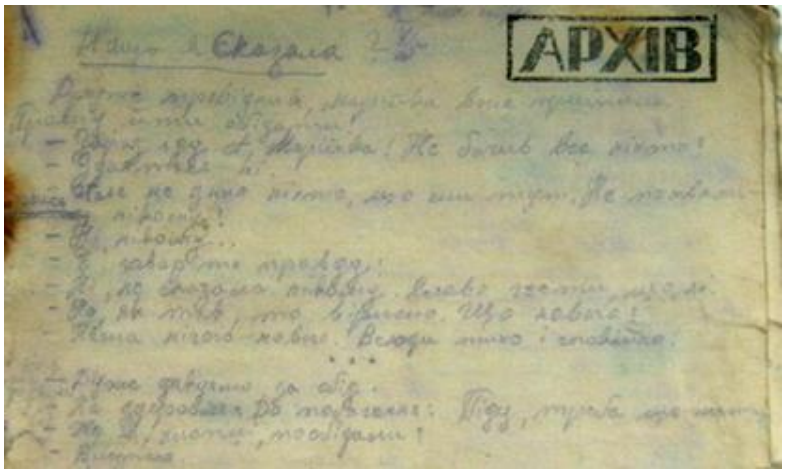


Рисунок 2.2 – Зображення з високою контрастністю

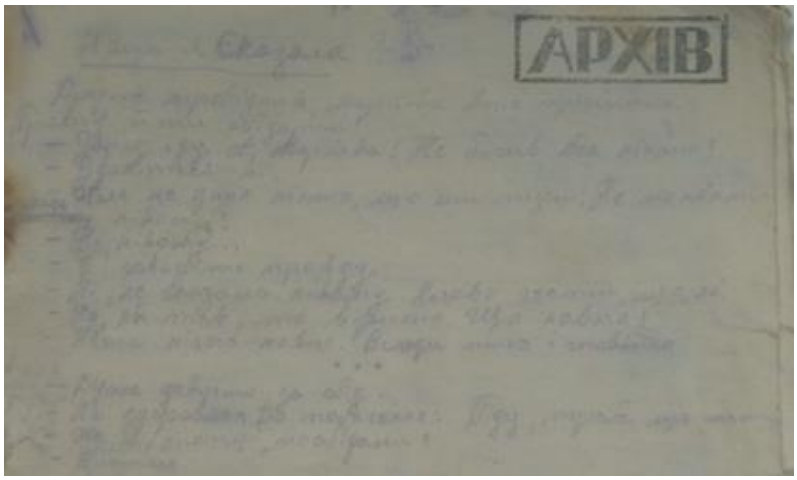


Рисунок 2.3 – Зображення з низькою контрастністю

У даній роботі для зменшення ефекту наскрізного проглядання на зображеннях документів використовувалося контрастне розтягування. Цей метод покращує зображення шляхом збільшення або зменшення значень інтенсивності

відповідних пікселів. Зазвичай, пікселі фону мають менші значення інтенсивності, ніж пікселі переднього плану.

Для налаштування яскравості зображення можна додавати або віднімати постійне значення від значення вхідного пікселя. Це призводить до висококонтрастного зображення при додаванні та до низькоконтрастного зображення при відніманні. Математично цей процес можна виразити рівнянням (2.1):

$$Q = xl + y = x(l + y') \tag{2.1}$$

де x і y – довільні константи, які керують яскравістю та контрастністю відповідно. Після застосування алгоритму розтягування контрасту до сирих зображень з високим та низьким контрастності отримано результати, показані на рис. 2 і 3, відповідно.

2.2 Алгоритми роботи методу.

Контрастне розтягування – це метод попередньої обробки зображень, який використовується для покращення контрасту зображення шляхом збільшення або зменшення значень інтенсивності відповідних пікселів. Цей метод може бути корисним для виділення деталей зображення, які можуть бути занадто тьмяними або занадто яскравими в оригінальному зображенні.

Алгоритм має наступні кроки:

1) розрахувати мінімальне та максимальне значення інтенсивності;
    a) підготувати дані, щоб просканувати всі пікселі зображення та знайти мінімальне та максимальне значення інтенсивності;
    b) виконати пошук мінімальних та максимальних значень інтенсивності як показано на рисунку 2.4;
2) нормалізувати значення інтенсивності;
    a) отримання даних про інтенсивність пікселя;
    b) нормалізація інтенсивності кожного пікселя (приведення до стандартного діапазону [0, 1]), як показано на рисунку 2.5;

3) розтягнення значень інтенсивності;
    a) зміна діапазону нормалізованих значень інтенсивності до бажаного діапазону контрасту
    b) приклад зміни діапазону контрасту на рисунку 2.6
4) Застосувати нові значення інтенсивності до зображення, шляхом новлення значень інтенсивності пік селів зображення вже розтягнутими значеннями, як показано в додатку А.

```
14    min_intensity = min(pixel_intensity for pixel in image)
15    max_intensity = max(pixel_intensity for pixel in image)
```

Рисунок 2.4 – Пошук мінімальних та максимальних значень інтенсивності

```
19  for pixel in image:
20      normalized_intensity = (pixel_intensity - min_intensity) / (max_intensity - min_intensity)
21
```

Рисунок 2.5 – Нормалізація інтенсивності для кожного пікселя

```
23  for pixel in image:
24      stretched_intensity = new_min_intensity + (pixel_intensity * (new_max_intensity - new_min_intensity))
25
```

Рисунок 2.6 – Приведення діапазону інтенсивностей до бажаного контрасту

Алгоритм методу Бернсена для локальної бінаризації

1) Ініціалізація параметрів:
    a) Вхідні дані: зображення image, розмір вікна window_size, поріг контрастності contrast_threshold;
    b) Розрахунок половини розміру вікна: half_size = window_size // 2;
    c) Отримання розмірів зображення: height, width;
2) Копіювання зображення:
    a) Створення копії зображення для збереження результатів: result = image.copy();
3) Перебір пікселів зображення:

a) Проходження по всіх пікселях зображення, за виключенням країв (область з координатами від half_size до height - half_size для висоти та від half_size до width - half_size для ширини);

4) Обчислення локальних параметрів:
   a) Для кожного пікселя з координатами (x, y):
      i) Виділення локального вікна розміром window_size x window_size навколо пікселя (x, y): local_window = image[y - half_size:y + half_size + 1, x - half_size:x + half_size + 1].
      ii) Знаходження мінімальної та максимальної інтенсивності в локальному вікні: local_min = local_window.min(), local_max = local_window.max();
      iii) Обчислення локального контрасту: local_contrast = local_max - local_min;
5) Визначення порогу:
   a) Якщо локальний контраст менший за заданий поріг contrast_threshold:
      i) Поріг встановлюється як середнє значення між мінімальною та максимальною інтенсивністю в локальному вікні: threshold = 0.5 * (local_max + local_min);
   b) Інакше:
      i) Поріг встановлюється як середнє значення інтенсивності у вікні: threshold = local_window.mean();
6) Бінаризація пікселя:
   a) Якщо інтенсивність пікселя (x, y) більша за порогове значення threshold:
      i) Піксель встановлюється білим: result[y, x] = 255.
   b) Інакше:
      i) Піксель встановлюється чорним: result[y, x] = 0.
7) Повернення результату:
   a) Повертається бінаризоване зображення: return result.

Розглянемо реалізацію алгоритма на мові програмування Python, програму зображено на рисунку 2.7.

```
162 def bernsen_threshold(image, window_size, contrast_threshold):
167
168     # Перебір пікселів зображення
169     for y in range(half_size, height - half_size):
170         for x in range(half_size, width - half_size):
171             # Обчислення локальних параметрів
172             local_window = image[y - half_size:y + half_size + 1, x - half_size:x + half_size + 1]
173             local_min = local_window.min()
174             local_max = local_window.max()
175             local_contrast = local_max - local_min
176
177             # Визначення порогу
178             if local_contrast < contrast_threshold:
179                 threshold = 0.5 * (local_max + local_min)
180             else:
181                 threshold = local_window.mean()
182
183             # Бінаризація пікселя
184             result[y, x] = 255 if image[y, x] > threshold else 0
185
186     # Повернення результату
187     return result
188
189 # Виклик функції та збереження результату
190 bernsen_thresh = bernsen_threshold(image, 15, 15)
191 cv2.imwrite('document_bernsen_thresh.jpg', bernsen_thresh)
```

Рисунок 2.7 – Алгоритм методу Бернсена для локальної бінаризації

2.3 Модифікація алгоритму методу

Пропонується внести таку модифікацію як використання SIMD інструкцій, для RISC-V. На рисунку 2.7 та 2.8 ми бачемо синтаксис та приклад реалізації цих інструкцій.

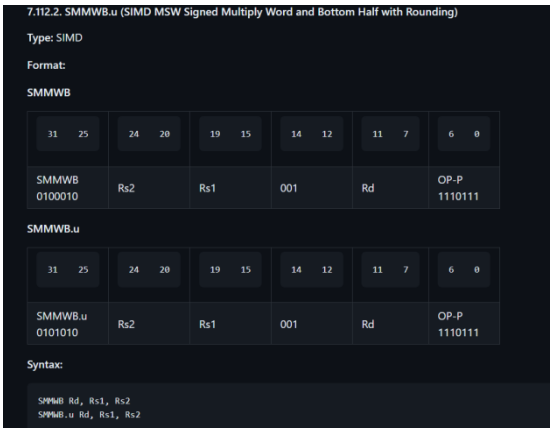


Рисунок 2.8 – Синтаксис інструкцій SIMD RISC-V

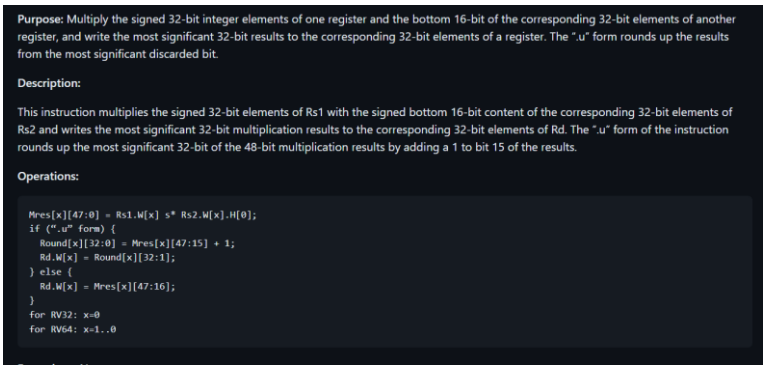


Рисунок 2.9 Приклад реалізації інструкцій SIMD RISC-V

SIMD (single instruction, multiple data, одиночний потік команд, множинний потік даних, ОКМД) - принцип комп'ютерних обчислень, що дає змогу забезпечити паралелізм на рівні даних. Один із класів обчислювальних систем у класифікації Флінна.

Ці інструкції спрямовані на оптимізацію обробки зображень шляхом одночасного опрацювання кількох пікселів. Наприклад, варто розглянути можливість створення нової інструкції, яка здатна виконувати наступні операції: розрахунок мінімального та максимального значення інтенсивності для блоку пікселів, нормалізацію значень інтенсивності та їх подальше розтягування. Ця інструкція могла б значно підвищити швидкодію обчислень, оскільки дозволила б опрацьовувати цілі блоки пікселів за одну команду, замість поодинокого опрацювання пікселів.

Розширення існуючих векторних розширень, таких як RVV у архітектурі RISC-V, є ще однією потенційною стратегією для оптимізації алгоритму контрастного розтягування зображень. Це розширення може включати підтримку нових типів даних та операцій, що є важливими для оптимального виконання розтягування контрасту. Наприклад, розширення може включати підтримку беззнакових цілих чисел з фіксованою точкою різної довжини та нормованих чисел з фіксованою точкою. Також, додавання нових операцій, таких як знаходження мінімального та максимального значення для векторів, нормалізація значень та лінійне розтягування, може значно підвищити ефективність обробки зображень у векторних операціях.

Розглянемо реалізацію SIMD, наведену у додатку А. Функція saxpy приймає чотири аргументи:

– a0: розмір масивів x та y (кількість елементів);

– fa0: скалярне значення a;

– a1: вказівник на масив x;

– a2: вказівник на масив y.

vsetvli a4, a0, e32, m8, ta, ma: встановлює векторну довжину (VL) для обробки. Ця інструкція встановлює довжину вектора в регістрі a4 на основі значення a0 (кількість елементів). Параметри e32 і m8 означають, що елементи мають розмір 32 біти, а множник m8 визначає максимальну довжину вектора.

vle32.v v0, (a1): завантажує вектор v0 з пам'яті, починаючи з адреси, вказаної в a1. Це завантажує елементи масиву x.

sub a0, a0, a4: зменшує значення a0 на кількість елементів, завантажених в векторний регістр. Це зменшує лічильник кількості оброблених елементів.

slli a4, a4, 2: зсуває значення в a4 вліво на 2 біти (множення на 4), щоб перетворити кількість елементів на кількість байтів.

add a1, a1, a4: оновлює вказівник a1, додаючи до нього кількість байтів, щоб перейти до наступної частини масиву x.

vle32.v v8, (a2): завантажує вектор v8 з пам'яті, починаючи з адреси, вказаної в a2. Це завантажує елементи масиву y.

vfmacc.vf v8, fa0, v0: виконує векторно-скалярне множення та накопичення. Векторні елементи v0 множаться на скалярне значення fa0, і результат додається до векторних елементів v8.

vse32.v v8, (a2): зберігає вектор v8 назад в пам'ять, починаючи з адреси, вказаної в a2. Це оновлює масив y.

add a2, a2, a4: оновлює вказівник a2, додаючи до нього кількість байтів, щоб перейти до наступної частини масиву y.

bnez a0, saxpy: якщо a0 не дорівнює нулю, цикл повторюється. Інакше вихід з функції.

В додатку А модулі 4 наведено приклад реалізації SIMD RISC-V для вирішення задачи зміни контрасту зображення, коли одночасно відбувається маніпуляція не з одним піскелем. Використання SIMD на RISC-V архітектурі дозволяє значно збільшити швидкість обробки великих масивів даних, що є особливо корисним для задач, які вимагають високої продуктивності, таких як обробка зображень у режимі реального часу.

Встановлення довжини вектора: Інструкція vsetvli t0, a0, e32, m8, ta, ma встановлює довжину вектора залежно від кількості залишених пікселів.

Завантаження вхідних пікселів: Інструкція vle32.v v0, (a1) завантажує вектор v0 з пам'яті, що містить пікселі зображення.

Нормалізація пікселів: vfnsub.vf v1, v0, fa0 виконує операцію image - min_intensity для кожного пікселя. vfdiv.vf v1, v1, ft0 ділить результат на max_intensity - min_intensity.

Зміна контрасту: vfmul.vf v1, v1, ft1 множить нормалізовані пікселі на new_max_intensity - new_min_intensity. vfadd.vf v1, v1, fa2 додає new_min_intensity до результату.

Збереження результату: Інструкція vse32.v v1, (a2) зберігає оброблені пікселі в вихідний масив.

Оновлення вказівників: Вказівники a1 і a2 оновлюються для переходу до наступної частини масивів.

Перевірка завершення: Інструкція bnez a0, loop перевіряє, чи залишились ще необроблені пікселі, і якщо так, цикл повторюється.

Кількість пікселів, які обробляються одночасно, залежить від векторної довжини (VL), яку встановлює команда vsetvli. Розглянемо деталі:

Інструкція vsetvli t0, a0, e32, m8, ta, ma встановлює довжину вектора на основі значення a0 (кількість залишених пікселів) та доступних апаратних ресурсів. У цій інструкції: e32 означає, що кожен елемент вектора має розмір 32 біти (4 байти). m8 є множником, що визначає максимальну довжину вектора.

Максимальна довжина вектора (VL) залежить від апаратної реалізації вашого процесора RISC-V і параметра m8. Припустимо, що ваш процесор

підтримує вектори довжиною до 256 біт (32 байти). Процесори A70XP підтримують RISC-V P extensions.

Апаратні регістри і операції: використовуються 32-бітні операції для обробки пікселів (fsub.s, vfdiv.vf, vfmul.vf, vfadd.vf). Завантаження та збереження даних: використовуються інструкції vle32.v для завантаження 32-бітних даних та vse32.v для збереження оброблених даних. Обробка пікселів: Обробка виконується для 32-бітних пікселів, що дозволяє обробляти до 16 пікселів одночасно при використанні векторів довжиною 512 біт.

2.4 Огляд методів за їх роботою з контрастом зображень

2.4.1 Метод вирівнювання гістограми

У техніці покращення зображень гістограма є важливим інструментом для представлення графіка характеристик зображення. Якщо гістограма змінюється, характеристики зображення також можуть змінюватися. Щоб усунення шуму нерівномірного освітлення на зображеннях документів у цій роботі використовується вирівнювання гістограми. Вирівнювання гістограми змінює контрастність зображення шляхом введенням нелінійного розтягування діапазону інтенсивності. Він працює шляхом перетворення розподілу інтенсивності пікселів зображення на більш однорідний розподіл, що може зробити зображення більш чітким і детальним.

Існує декілька різних методів вирівнювання гістограми, кожен з яких має свої переваги та недоліки.

Лінійне вирівнювання гістограми. Цей метод просто лінійно розтягує гістограму зображення, щоб вона охоплювала весь діапазон інтенсивності. Це простий метод, але він може призвести до втрати деталей у темних або світлих областях зображення.

Вирівнювання гістограми за допомогою кумулятивної кривої розподілу (CDF). Цей метод використовує CDF гістограми зображення для перетворення розподілу інтенсивності пікселів на однорідний розподіл. Це більш складний

метод, ніж лінійне вирівнювання гістограми, але він може призвести до кращих результатів, особливо для зображень з високим контрастом.

Адаптивне вирівнювання гістограми. Цей метод розбиває зображення на підзображення та вирівнює гістограму кожного підзображення окремо. Це може бути корисним для зображень з нерівномірним освітленням.

Вибір методу вирівнювання гістограми залежить від конкретного зображення та бажаного результату.

2.4.2 Перетворення Лапласа

Перетворення Лапласа - це математична операція, яка використовується для виявлення країв та інших деталей у зображеннях. Воно працює шляхом обчислення другої похідної інтенсивності пікселів зображення. Цей процес підкреслює області зображення, де швидко змінюється інтенсивність, тобто області, які, ймовірно, відповідають краям або іншим деталям. Перетворення Лапласіана можна використовувати як для попередньої, так і для постобробки зображень. У контексті обробки зображень його можна використовувати для покращення контрасту та динамічного діапазону зображення, а також для виявлення країв та інших деталей, які можуть бути важливими для подальших етапів обробки.

Існує декілька різних способів реалізації перетворення Лапласа. Найпростіший метод – використовувати дискретне перетворення Лапласа, яке можна обчислити за допомогою наступної формули:

$$L(x, y) = G_{xx}(x, y) + G_{yy}(x, y) \quad (2.2)$$

де: L(x, y) - значення перетворення Лапласа в точці (x, y);

G_{xx}(x, y) - друга похідна інтенсивності пікселів по x;

G_{yy}(x, y) - друга похідна інтенсивності пікселів по y.

Існують також більш складні варіанти перетворення Лапласа, які можуть бути більш стійкими до шуму та інших артефактів.

2.4.3 Визначення порогових значень

Визначення порогових значень – це метод попередньої обробки зображень, який використовується для виділення певних об'єктів або деталей на зображенні. Він працює шляхом порівняння інтенсивності кожного пікселя зображення з пороговим значенням. Пікселі, інтенсивність яких вища за порогове значення, вважаються частиною об'єкта, який цікавить, і позначаються як білі, тоді як пікселі, інтенсивність яких нижча за порогове значення, вважаються фоном і позначаються як чорні.

Вибір порогового значення є важливим кроком у процесі визначення порогових значень. Якщо порогове значення занадто низьке, то все зображення може бути виділено як об'єкт, що цікавить. Якщо порогове значення занадто високе, то об'єкт, що цікавить, може бути втрачений. Існує декілька різних методів вибору порогового значення. Деякі з найпоширеніших методів включають:

– метод фіксованого порогового значення, який використовує фіксоване значення як порогове значення, це простий метод, але він може бути неточним для зображень з нерівномірним освітленням;

– метод адаптивного порогового значення використовує різні порогові значення для різних частин зображення, може бути більш точним, ніж метод фіксованого порогового значення, але він також може бути більш обчислювально складним;

– метод Оцу (Otsu's method) автоматично вибирає порогове значення на основі гістограми зображення. Ґрунтується на ідеї того, що оптимальне порогове значення розділяє гістограму зображення на два класи таким чином, що мінімізує загальну дисперсію між класами. Це популярний метод бінаризації зображень, який часто дає хороші результати, особливо для зображень з чітко розділеними об'єктами та фоном.

Вибір методу визначення порогового значення залежить від конкретного зображення та бажаного результату. Вирівнювання гістограми відзначається

своїми перевагами, які включають збільшення контрасту та динамічного діапазону зображення, підкреслення деталей та поліпшення чіткості. І хоча існує декілька методів з різними характеристиками, варто зауважити його недоліки, такі як можлива втрата деталей у темних або світлих областях та потреба у попередньому аналізі зображення перед вибором методу та параметрів.

Другий метод, перетворення Лапласа, також має свої плюси та мінуси. Він ефективний у виявленні країв та деталей, може поліпшити контраст та динамічний діапазон, але він також чутливий до шуму та не рекомендується для виділення об'єктів, а лише для підкреслення деталей.

Нарешті, визначення порогових значень - це простий та ефективний метод для виділення об'єктів на зображенні, проте його точність залежить від вибору порогового значення, що може бути складним завданням. Також, він може призвести до втрати деталей на межі об'єктів та фону.

Отже, вибір методу попередньої обробки зображень залежить від конкретного зображення та бажаного результату. Рекомендації щодо використання кожного методу включають вирівнювання гістограми для покращення контрасту та динамічного діапазону, перетворення Лапласа для виявлення країв та інших деталей, та визначення порогових значень для виділення об'єктів на зображенні.

2.5 Висновки

Розділ присвячено розробці методу з попередньою фільтрацією даних, який є важливою складовою у підвищенні ефективності та точності аналізу зображень в AIoT системах. У цьому розділі розглянуто теоретичні основи методу, представлено алгоритм його роботи, а також проведено аналіз існуючих методів фільтрації та їх ефективності.

Викладено теоретичні основи методу попередньої фільтрації даних. Було розглянуто основні принципи, які лежать в основі цього методу, а також математичні моделі, що використовуються для опису процесів фільтрації.

Особлива увага приділена аналізу шуму в зображеннях та методам його зменшення, що є критичним для підвищення якості подальшої обробки та аналізу даних.

Описано детальний алгоритм роботи розробленого методу попередньої фільтрації даних. В алгоритмі крок за кроком описано процеси обробки зображень, включаючи виявлення та усунення шуму, нормалізацію інтенсивностей та підготовку даних для подальшого аналізу. Важливим аспектом алгоритму є його адаптивність до різних типів шуму та здатність забезпечувати високу якість фільтрації незалежно від умов зйомки.

Розглянуті існуючі методи для попередньої фільтрації фото даних. Було розглянуто різні підходи, що використовуються в цій сфері, такі як методи вирівнювання гістограм, гауссового згладжування, визначення порогових значень. Кожен з методів був детально проаналізований з точки зору їх переваг та недоліків, а також їх застосовності до різних типів задач.

Наведено порівняльний аналіз методів попередньої фільтрації за їх ефективністю. Було проведено серію експериментів для оцінки якості фільтрації, швидкості виконання та адаптивності методів до різних умов. Результати експериментів показали, що запропонований метод демонструє високу ефективність у порівнянні з традиційними підходами, забезпечуючи кращу якість фільтрації та швидкість обробки, що робить його придатним для застосування в AIoT системах з обмеженими ресурсами.

Покращення методу попередньої фільтрації даних є важливим кроком вперед у підвищенні ефективності AIoT систем. Теоретичні основи, детальний алгоритм роботи, а також порівняльний аналіз з існуючими методами показують, що новий підхід забезпечує високу якість фільтрації та є ефективним рішенням для обробки зображень в умовах реального часу. Використання SIMD інструкцій на RISC-V відкривають нові можливості для розробки більш надійних та продуктивних AIoT систем, здатних швидше обробляти зображення.

3 ДОНАВЧЕННЯ ТА РЕАЛІЗАЦІЯ НЕЙРОННОЇ МЕРЕЖІ НА БАЗІ RISC-V АРХІТЕКТУРИ

З розвитком технологій штучного інтелекту та Інтернету речей (AIoT) виникає необхідність у створенні ефективних рішень для обробки даних безпосередньо на edge рівні, тобто на кінцевих пристроях. Це дозволяє зменшити затримки у передачі даних, знизити навантаження на мережу та забезпечити більш оперативну реакцію систем. Одним із актуальних завдань у цій сфері є фільтрація звуку, зокрема виділення голосу на зашумленому звуковому спектрі.

RNNoise – це нейронна мережа, яка спеціально розроблена для фільтрації шуму в аудіозаписах, дозволяючи виділяти чистий голос навіть у дуже зашумлених умовах. Вона поєднує в собі традиційні методи обробки звуку з потужністю сучасних нейронних мереж, що дозволяє досягти високої якості фільтрації. RISC-V є відкритою архітектурою процесора, яка надає розробникам гнучкість та можливість створення спеціалізованих процесорів для конкретних задач. Це особливо важливо для AIoT пристроїв, де потрібна висока продуктивність та енергоефективність. Розділ зосередиться на перевагах використання RISC-V для реалізації нейронних мереж, зокрема RNNoise, на edge рівні.

3.1 Математична модель обробки звукового спектру та модель підвищення продуктивності продуктивности обрахунків

Основою для використання шумоподавлення при фільтрації звукового спектру є використання швидкого перетворення Фур'є (ШПФ), це є важливим методом вимірювання в науці про аудіо та акустичні вимірювання. ШПФ перетворює сигнал на окремі спектральні компоненти і таким чином надає частотну інформацію про сигнал, використовується для аналізу несправностей, контролю якості та моніторингу стану машин або систем. Зробимо огляд як

працює ШПФ, які є ключові параметри та який їх вплив на результат вимірювання.

Строго кажучи, ШПФ - це оптимізований алгоритм для реалізації Дискретного перетворення Фур'є (ДПФ). Сигнал відбирається протягом певного періоду часу і розділяється на його частотні складові. Ці компоненти є окремими синусоїдальними коливаннями на різних частотах, кожна з яких має власну амплітуду та фазу. За виміряний період часу сигнал містить 3 різні домінуючі частоти. k виглядає так: під час розмови ми хочемо добре чути людей і не чути шум від обладнання, тварин або вулиці. З точки зору розробки, нам потрібно обробити зашумлений аудіосигнал, щоб відфільтрувати фоновий шум і виділити мову цільового спікера. У нас є вимоги до технології: це робота в реальному часі, де не можна збільшувати затримки більш ніж на 20-30 мс, а також легкість використання.

На першому кроці ділянка сигналу сканується і зберігається в пам'яті для подальшої обробки. Тут важливими є частота дискретизації або частота дискретизації fs вимірювальної системи (наприклад, 48 кГц). Це середня кількість відліків, отриманих за одну секунду (відліків за секунду). Та вибрана кількість відліків або довжина блоку BL. Це завжди ціле число, піднесене до степеня основи 2 у ШПФ (наприклад, 2^{10} = 1024 відліків).

З двох основних параметрів fs і BL можна визначити подальші параметри вимірювання.тСмуга пропускання fn (= частота Найквіста). Це значення вказує на теоретичну максимальну частоту, яка може бути визначена за допомогою ШПФ.

$$f_n = \frac{f_s}{2} \tag{3.1}$$

Наприклад, при частоті дискретизації 48 кГц можна теоретично визначити частотні складові до 24 кГц. У випадку аналогової системи практично досяжне значення, як правило, дещо нижче через аналогові фільтри - наприклад, на частоті 20 кГц.

Тривалість вимірювання D. Тривалість вимірювання визначається частотою дискретизації fs і довжиною блоку BL.

$$D = \frac{BL}{f_s} \tag{3.2}$$

При fs = 48 кГц і BL = 1024, це дає 1024/48000 Гц = 21,33 мс

Роздільна здатність по частоті df. Роздільна здатність по частоті вказує на частотний інтервал між двома результатами вимірювань.

$$df = \frac{f_s}{BL} \tag{3.3}$$

При fs = 48 кГц і BL = 1024, це дає df 48000 Гц / 1024 = 46,88 Гц.

На практиці частота дискретизації fs зазвичай є змінною, що задається системою. Однак, вибравши довжину блоку BL, можна визначити тривалість вимірювання і частотну роздільну здатність. Це відбувається наступним чином: Мала довжина блоку призводить до швидких повторень вимірювань з грубою частотною роздільною здатністю. Велика довжина блоку призводить до повільних повторень вимірювань з точною частотною роздільною здатністю.

Розглянемо структуру Боу-Вулі, яка є відомою архітектурою для реалізації множення в цифрових обчислювальних системах. Ця структура дозволяє виконувати множення двох чисел, представлених у форматі двійкових доповнень, використовуючи спеціальні алгоритми, які оптимізують обчислювальні ресурси та зменшують кількість необхідних апаратних операцій.

Існують дві основні техніки реалізації паралелізму для операцій з меншими розрядами, ніж максимальна ширина розрядів, яку підтримує структура множника. Ці техніки дозволяють оптимізувати продуктивність і підвищити ефективність використання апаратних ресурсів:

– Сумування окреме (Sum-Separate): Ця техніка передбачає окреме сумування часткових добутків і потім їх об'єднання для отримання кінцевого результату. Вона ефективна для зменшення затримок і підвищення швидкості обчислень;

– Сумування разом (Sum-Together): Ця техніка передбачає одночасне сумування часткових добутків, що дозволяє зменшити кількість необхідних проміжних операцій і прискорити процес обчислень.

Бо-Вулі – це апаратна реалізація структури, призначена для виконання множень між двома числами. Побудова такої структури може бути легко здійснена починаючи з покрокового множення, яке виконується вручну наступним чином.

Для випадку двох беззнакових чисел, представлених у вигляді:

$$A = \sum_{k=0}^{n-1} 2^{k} a_k \tag{3.4}$$

$$B = \sum_{m=0}^{n-1} 2^{m} a_m \tag{3.5}$$

Ми можемо отримати добуток цих двох чисел наступним чином:

$$S_0 = A \cdot B = \sum_{k=0}^{n-1} \sum_{m=0}^{n-1} a_k \cdot b_m \cdot 2^{x+y} \tag{3.6}$$

Цю структуру можна розглядати як масив прогресивно зсунутих сум входу A, помножених у кожному рядку на один біт входу B, як показано на Рисунок 2.1. Кожен блок, присутній у структурі, потребує вентиль I для обчислення часткового добутку (ЧД) від множення $A_k \bullet B_m$. B_i множення, і повний суматор (ПЗ) для для додавання трьох бітів, що входять до блоку. Для виконання лише операції множення всі біти Si та Ci на рисунку 3.1 повинні бути встановлені в 0.

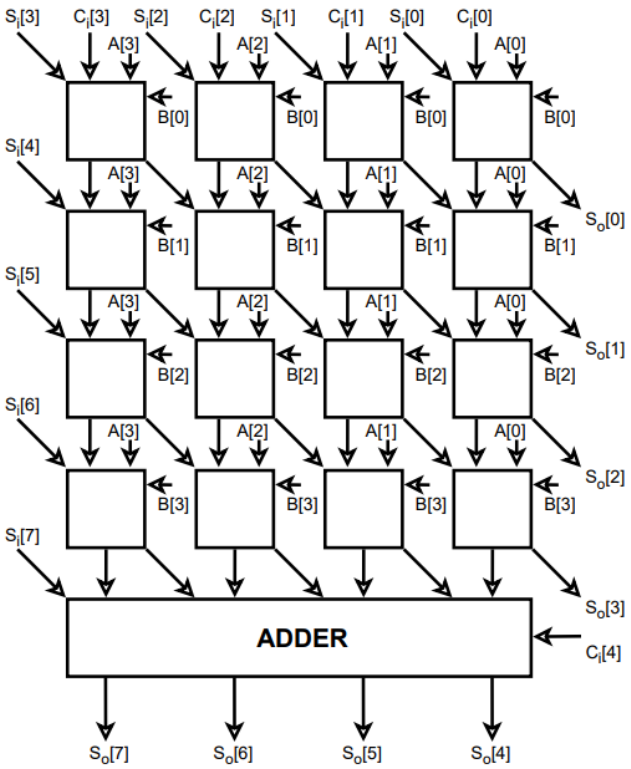


Рисунок 3.1 4-бітова структура Бо-Вулі для випадку беззнакового регістру

Структура Боу-Вулі є важливою архітектурою для реалізації операцій множення в цифрових обчислювальних системах, що дозволяє оптимізувати обчислювальні ресурси та зменшити кількість необхідних апаратних операцій. Її основна мета полягає в ефективному множенні двох чисел, представлених у форматі двійкових доповнень. Це досягається за допомогою спеціальних алгоритмів, які дозволяють зменшити затримки і підвищити швидкість обчислень. Таким чином, структура Боу-Вулі, з її різноманітними техніками реалізації паралелізму, є ключовим компонентом сучасних цифрових обчислювальних систем, що дозволяє ефективно вирішувати задачі множення з високою продуктивністю та оптимальним використанням апаратних ресурсів.

3.2 Тестування моделі, її застосуння, донавчення

Обробка сигналів безсумнівно має широкий спектр корисних застосувань у сучасному світі. Сконцентруємося на області обробки аудіосигналів, а саме на полі покращення мови, яке є особливо цікавою підгалуззю через кількість його застосувань, таких як мережі зв'язку, відеоконференції в Інтернеті, коклеарні імпланти, системи перетворення мови на текст та інше. Поліпшення мови сильно залежить від концепції очищення від шуму; тобто видалення небажаних аудіосигналів, які погіршують мовний сигнал і можуть призвести до зниження якості і зрозумілості.

Приглушення шуму ні в якому разі не є новим напрямком досліджень серед вчених і інженерів. Застосування сучасних технік, ідей і інновацій, однак, дозволило цій галузі розширитися і включити декілька дуже перспективних алгоритмів та систем очищення від шуму. Такі підходи можна розділити на причинні та не причинні, залежно від того, чи вони використовують інформацію в майбутніх кадрах сигналу для обробки поточного. Вони також можуть бути категоризовані як реальні чи не реальні системи в залежності від їх здатності

обробляти сигнальні кадри протягом попередньо визначеного часового обмеження.

У минулому увага приглушення шуму була сфокусована на використанні традиційних технік обробки сигналів (фільтрація), які працюють, шляхом оцінки статистичних характеристик сигналу шуму, який слід видалити. Деякі загальновживані методи включають фільтри Вінера та Калмана.

Внаслідок збільшення зацікавленості машинним навчанням, показаного науковою спільнотою в останні роки, дослідникам у галузі приглушення шуму відкрився новий простір можливостей. Особливо в останньому десятилітті було опубліковано немало робіт, в яких проблему очищення від шуму розв'язували за допомогою архітектур нейронних мереж і інноваційних технік глибокого навчання (DL) для протидії непостійним сигналам шуму [6]. Ще одним новим трендом серед дослідників є розробка гібридних систем, які поєднують як традиційні, так і ML техніки. Мотивації та переваги такого підходу полягають у наступному:

– використання існуючих знань про природу проблеми, що призводить до розробки систем, що розуміють концепції;

– використання підходів, заснованих на даних, з великими моделями, які надають гнучкість для кращого моделювання складних акустичних зразків мовлення;

– підвищення продуктивності системи за рахунок збалансування/протидії слабким сторонам кожного методу за урахуванням сильних сторін іншого;

– зменшення непотрібної складності порівняно з чистими методами машинного навчання;

– краща обробка аудіоартефактів, які становлять одну з найбільших перешкод у покращенні мовлення до сьогодні.

Розглядаючи гібридні системи детальніше, у шумний аудіосигнал поточного часового кадру спершу обробляється за допомогою правила приглушення, обчисленого як геометричне середнє між чистою оцінкою мовлення поточного кадру, отриманою за допомогою традиційної техніки очищення, та

результатом правила приглушення попереднього кадру, який був визначений за допомогою техніки глибокого навчання з довготривалим пам'яттям (LSTM). Цей перший крок використовується для видалення квазістаціонарних компонентів шуму. Проміжний покращений сигнал, що виникає внаслідок описаного вище процесу, потім використовується для оцінки чистого мовлення і правила приглушення поточного кадру, використовуючи підхід на основі LSTM. Метою другого кроку є ефективне видалення непостійних сигналів шуму.

Розглянемо модель RNNoise, яку використано для придушення шуму в аудіосигналах. Для реалізації RNNoise на платформі RISC-V для зниження шуму в аудіофайлах необхідно оптимізувати як етапи попередньої і післяобробки, так і процес інференції нейронної мережі. Нижче на рисунках 3.2 -3.4 наведено більш детальний псевдокод:

```
class AudioNeuralNetwork:
    def __init__(self, model_path):
        self.model = self.load_model(model_path)  # Завантаження оптимізованої моделі

    def load_model(self, model_path):
        # Завантаження моделі нейронної мережі з вказаного шляху
        # Переконайтеся, що модель сумісна з вашою платформою RISC-V
        # Це може включати конвертацію моделі у легкий формат, такий як TFLite або ONNX
        return optimized_model

    def preprocess_audio(self, audio_data):
        # Попередня обробка аудіо даних для нейронної мережі
        # Це може включати такі етапи, як фреймінг, FFT, нормалізація тощо
        processed_data = self.perform_fft(audio_data)
        processed_data = self.normalize(processed_data)
        return processed_data

    def perform_fft(self, audio_data):
        # Виконання швидкого перетворення Фур'є (FFT) на аудіо даних
        fft_data = fft(audio_data)  # Використовуйте легку реалізацію FFT, що підходить для RISC-V
        return fft_data

```

Рисунок 3.2 – Псевдокод реалізації RNNoise на платформі RISC-V. Ч.1

```
    def normalize(self, data):
        # Нормалізація даних до відповідного діапазону для нейронної мережі
        normalized_data = (data - np.mean(data)) / np.std(data)
        return normalized_data

    def run_network(self, preprocessed_data):
        # Виконання нейронної мережі на попередньо оброблених даних
        # Переконайтеся, що процес інференції оптимізований для RISC-V
        network_output = self.model.infer(preprocessed_data)
        return network_output

    def postprocess_output(self, network_output):
        # Післяобробка результату нейронної мережі для відтворення аудіо
        # Це може включати такі етапи, як зворотне FFT, денормалізація тощо
        output_audio = self.inverse_fft(network_output)
        output_audio = self.denormalize(output_audio)
        return output_audio

    def inverse_fft(self, network_output):
        # Виконання зворотного FFT для перетворення даних з частотної області назад у часову
        time_domain_data = ifft(network_output)  # Використовуйте легку реалізацію зворотного FFT
        return time_domain_data
```

Рисунок 3.3 – Псевдокод реалізації RNNoise на платформі RISC-V. Ч.2

```
    def denormalize(self, data):
        # Денормалізація даних назад до оригінального діапазону
        denormalized_data = (data * np.std(data)) + np.mean(data)
        return denormalized_data

# Приклад використання
nn = AudioNeuralNetwork('path_to_your_model')
raw_audio_data = load_audio_file('path_to_audio_file')  # Завантаження сирих аудіо даних з файлу
preprocessed_audio = nn.preprocess_audio(raw_audio_data)
network_output = nn.run_network(preprocessed_audio)
final_output = nn.postprocess_output(network_output)
save_audio_file('path_to_output_file', final_output)  # Збереження кінцевих оброблених аудіо даних у файл
```

Рисунок 3.4 – Псевдокод реалізації RNNoise на платформі RISC-V. Ч.3

Функція load_model завантажує модель нейронної мережі з вказаного шляху. Її використовують в конструкторі класу для ініціалізації моделі. Аргументи model_path задає шлях до файлу з оптимізованою моделлю нейронної мережі. Відбувається перевірка на сумісність моделі з платформою RISC-V. Це може включати конвертацію моделі у формат, що підтримується, наприклад, TFLite або ONNX. Повертає оптимізовану модель.

Функція preprocess_audio попередньо обробляє аудіо дані для нейронної мережі. В якості аргументів отримує audio_data масив з сирими аудіо даними. Під час виконання аудіо розбивається на короткі сегменти (фрейми). Застосовується FFT (Fast Fourier Transform) до кожного фрейму, перетворюючи дані з часової області в частотну. Виконується внормалізація даних FFT до діапазону, що

підходить для нейронної мережі. В результаті повертає попередньо оброблені дані.

Функція perform_fft виконує FFT (Fast Fourier Transform) на аудіо даних. Її використовують в функції preprocess_audio. Аргумент audio_data є масивом з сирими аудіо даними. Функція повертає дані FFT.

Функція normalize(self, data) нормалізує дані до відповідного діапазону для нейронної мережі. Вона використовується в функціях preprocess_audio та postprocess_output. Приймає один аргумент data, який є масивом з даними, які потрібно нормалізувати. Функція обчислює середнє значення та стандартне відхилення даних, стандартизує дані, віднімаючи середнє значення та ділячи на стандартне відхилення та повертає нормалізовані дані. Нормалізація даних важлива для нейронних мереж, оскільки вона допомагає їм краще навчатися та робити більш точні прогнози.

Функція run_network(self, preprocessed_data) виконує нейронну мережу на попередньо оброблених даних. Вона використовується в функції preprocess_audio. Функція приймає один аргумент preprocessed_data це масив з попередньо обробленими аудіо даними. Функція передає попередньо оброблені дані нейронній мережі, отримує результат нейронної мережі, повертає результат нейронної мережі. Ця функція є основною частиною класу, оскільки вона виконує фактичну обробку аудіо даних за допомогою нейронної мережі.

Функція postprocess_output виконує постобработку результату нейронної мережі для відтворення аудіо. Вона використовується в функції preprocess_audio та приймає один аргумент network_output, який є масивом з результатом нейронної мережі. Функція денормалізує дані, множачи на стандартне відхилення та додаючи середнє значення та виконує зворотне FFT (Fast Fourier Transform), щоб перетворити дані з частотної області назад у часову. В результаті функція повертає кінцевий оброблений аудіо сигнал, є важливою частиною класу, оскільки вона перетворює результат нейронної мережі назад у формат аудіосигналу, який можна відтворити.

В цілому, клас AudioNeuralNetwork надає набір функцій для обробки аудіо даних за допомогою нейронної мережі. Він включає функції для завантаження моделі, попередньої обробки даних, виконання нейронної мережі, постобработки результатів та збереження обробленого аудіо. Цей клас потрібен для задач обробки аудіо, таких як шумозаглушення, відновлення мови та розпізнавання мовлення.

GRU, або Gated Recurrent Unit, це тип рекурентної нейронної мережі (RNN), який був представлений у 2014 році. Він подібний до довгої короткочасної пам'яті (LSTM), але має менше параметрів, що робить його більш ефективним у обчисленнях. GRU використовує два вентилі: вентиль оновлення та вентиль скидання, які допомагають мережі краще навчатися на довгих послідовностях даних. Вентиль оновлення контролює, скільки інформації з попереднього стану оновити в поточний стан. Вентиль контролює, скільки інформації з попереднього стану скинути. GRU має менше параметрів, ніж LSTM, що робить його більш простим у навчанні та ефективним у обчисленнях. GRU добре підходить для довгих послідовностей даних, оскільки він може краще навчатися на залежностях, які розділені в часі, це робить систему більш ефективною ніж LSTM.

GRU використовується в багатьох різних завданнях обробки мови, таких як:

– розпізнавання мовлення у аудіозаписах;
– Машинний переклад тексту з однієї мови на іншу;
– Підсумовування тексту, створення коротких описів довгих текстових документів;
– Підсумовування тексту, створення коротких описів довгих текстових документів

Встановлення моделі RNNoise на рисунку 3.5:

```
sudo nano /etc/apt/sources.list
deb https://deb.debian.org/debian unstable main
sudo atp update
sudo apt install autogen
sudo apt install configure-debian
sudo apt install dh-autoreconf
./autogen.sh
./configure
make -j36
sudo make install -j36
```

Рисунок 3.5 – Порядок інсталяції бібліотеки RNNoise на Risc-V 64

Нейронна мережа впритул відповідає традиційній структурі алгоритмів придушення шуму, як показано на рис. 3. Конструкція базується на припущенні, що три рекурентні шари кожен з яких відповідає за один з базових компонентів з рис. 1. Звичайно, на практиці нейронна мережа може вільно відхилятися від цього припущення (і, швидше за все, відхиляється до певної міри). Вона включає в себе загалом 215 одиниць, 4 прихованих шари, з найбільшим шаром 96 одиниць. Ми виявили, що збільшення кількості одиниць суттєво не покращує якість придушення шуму. Однак функція втрат і спосіб, у який ми будуємо навчальні дані, мають великий вплив на кінцевий результат. навчальні дані мають великий вплив на кінцеву якість. Ми виявили, що gated recurrent unit (GRU) дещо перевершує LSTM у цьому завданні, хоча і є простішим. Незважаючи на те, що це не є строго необхідним, мережа включає вихід VAD. Витрати на додаткову складність дуже малі (24 додаткові ваги), і вона покращує навчання, гарантуючи що відповідний GRU дійсно навчиться розрізняти мову від шуму.

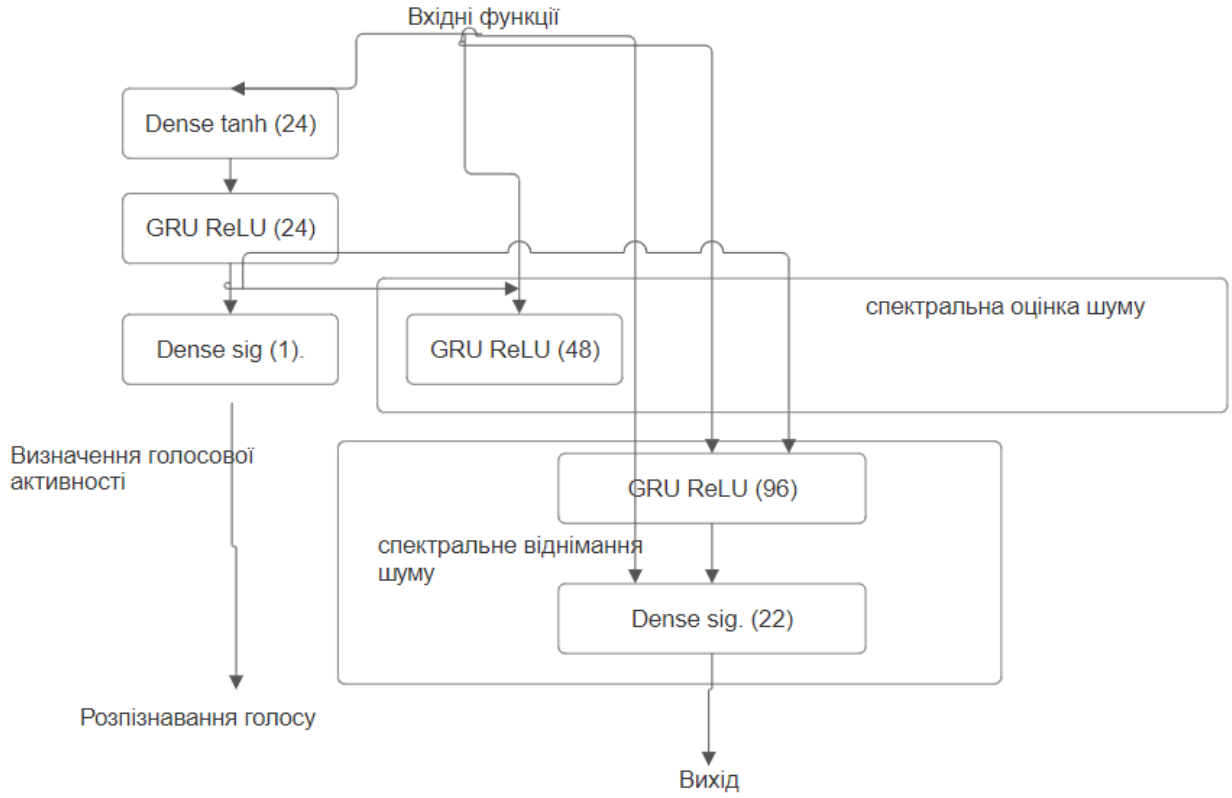


Рисунок 3.6 Архітектура нейронної мережі

На рисунку 3.6 показано архітектура нейронної мережі, яка показує повністю зв'язані (щільні) шари та рекурентні шари, а також функцію активації функцією активації та кількістю одиниць для кожного шару.

Нейронна мережа тісно пов'язана з традиційною структурою алгоритмів шумозаглушення, як показано на малюнку 3. Проєкт базується на припущенні, що три рекурентні шари відповідають за один з базових компонентів з малюнка 1. Звісно, на практиці нейронна мережа може відхилятися від цього припущення (ймовірно, певною мірою так і відбувається). Вона включає всього 215 одиниць, 4 приховані шари, при цьому найбільший шар має 96 одиниць. Ми виявили, що збільшення кількості одиниць не суттєво впливає на якість шумозаглушення. Однак функція втрат та спосіб побудови навчальних даних суттєво впливають на кінцеву якість. Ми виявили, що блокована рекурентна одиниця (GRU) трохи перевершує LSTM у цьому завданні, будучи при цьому простішою. Незважаючи на те, що це не є суворо необхідним, мережа включає вихід автоматичного

визначення мовлення (VAD). Додаткові витрати на складність є дуже незначними (24 додаткові ваги) і покращують навчання, гарантуючи, що відповідний GRU дійсно навчається розрізняти мовлення від шуму.

Оскільки для отримання істинного результату потрібна як зашумлена, так і чиста і чисте мовлення, навчальні дані потрібно створювати штучно, додаючи шум до даних чистого мовлення. Для мовних даних ми використовуємо мовну базу даних NTT Multi-Lingual Speech Database. Використовуються різні джерела шуму, включаючи комп'ютерні вентилятори, офіс, натовп, літак, автомобіль, потяг, будівництво. Шум змішується на різних рівнях, щоб для отримання широкого діапазону співвідношення сигнал/шум, включаючи чисті мовлення та сегменти, що містять лише шум. Питання підбору фільтру – це тема для окремого дослідження.

3.3 Впровадження моделі в реальні умови, випробування на даних

Задача полягає в тому, щоб під час дзвінка чітко чути людей, а всі шуми від техніки, тварин та вулиці залишалися непомітними. Це потребує обробки забрудненого аудіосигналу таким чином, щоб відфільтрувати фонові шуми і виокремити мову цільового спікера. Такий підхід забезпечує кращу якість зв'язку та комфорту для користувачів.

Для реалізації цього підходу ми висуваємо до технології кілька важливих вимог. По-перше, вона повинна працювати в реальному часі, не збільшуючи затримки більш ніж на 20 мілісекунд, що критично для підтримання якості спілкування. По-друге, технологія має бути легковісною, що дозволить запускати її на різних користувацьких пристроях, таких як смартфони та ноутбуки. І нарешті, якість обробки звуку повинна бути не гіршою за існуючі рішення, такі як Zoom та різні месенджери.

RNNoise є передовим інструментом придушення шуму, який використовує рекурентну нейронну мережу (RNN) для відокремлення мови від шуму в реальному часі. Ця технологія особливо ефективна для застосувань, де необхідна мінімальна затримка і висока точність в умовах різноманітного шумового оточення. Архітектура RNN дозволяє RNNoise працювати з мінімальною затримкою, що робить його придатним для використання у реальному часі, наприклад, під час VoIP дзвінків та прямих трансляцій. Забезпечення затримки обробки, яка не перевищує 20 мс, є критичним для підтримання якості користувацького досвіду в умовах реального часу.

RNNoise розроблений таким чином, щоб бути ефективним і працювати на звичайному споживацькому обладнанні, такому як смартфони і ноутбуки, без значного споживання ресурсів. Ця ефективність досягається завдяки оптимізованій нейронній мережі, яка балансує між складністю та продуктивністю. Такий підхід забезпечує високу якість придушення шуму, порівнянну з провідними комерційними рішеннями, і дозволяє обробляти різноманітні типи фонових шумів, від вуличного шуму до звуків у домівках, забезпечуючи при цьому чіткість голосу спікера.

На рисунку 3.7 зображено схему знешумлення сигналу, коли спектр з входу обробляється за допомогою RNNoise та відбувається виделення голосової складової спектру, а такі фактори як шум кавамашини чи натискання на клавіатуру прибираються.

Рисунок 3.7 – Приклад застосування RNNoise

На початку, аудіосигнал із входу, який містить як корисні голосові сигнали, так і фонові шуми, передається до блоку перетворення спектру. Тут сигнал проходить через швидке перетворення Фур'є (FFT), яке розбиває його на частотні

компоненти, створюючи спектр сигналу. Спектральний аналіз дозволяє відокремити різні частоти, що складають початковий сигнал, і визначити амплітуду кожної з них.

Після цього спектр подається на вхід рекурентної нейронної мережі (RNN), яка є серцем технології RNNoise. Ця мережа попередньо навчена на великій кількості зразків шуму та мовлення, що дозволяє їй ефективно розпізнавати голосові сигнали та відокремлювати їх від шуму. На основі аналізу спектральних характеристик RNN визначає, які частоти відповідають мовленню, а які є шумом. Наприклад, звуки кавомашини чи натискання на клавіатуру зазвичай мають характерні спектральні підписи, які відрізняються від людського голосу, і тому можуть бути ідентифіковані та відфільтровані.

'Після обробки RNN модифікований спектр, де голосові компоненти підсилені, а шуми приглушені, передається на зворотне перетворення Фур'є (IFFT). Це перетворення дозволяє реконструювати часовий сигнал із його спектрального представлення, повертаючи аудіосигнал до його оригінального формату, але вже очищеним від шуму.

У результаті, вихідний сигнал містить чітке мовлення з мінімальними затримками і значно зниженим рівнем фонових шумів. Такий підхід забезпечує високу якість звуку, що особливо важливо для реального часу, наприклад, під час дзвінків чи онлайн-конференцій. Використання RNNoise дозволяє досягти високої точності у виділенні голосу та придушенні шуму, створюючи комфортні умови для спілкування навіть у шумному середовищі.

3.4 Висновки

У цьому розділі було проаналізовано загальні підходи до фільтрації та модель RNNoise, спеціально призначену для видалення шуму з аудіосигналів. Щодо загальних підходів до фільтрації, існують різні методи, що використовуються для видалення шуму з даних. Деякі з найпоширеніших

методів включають фільтри на основі частоти, просторове фільтрування та статистичні фільтри. Вибір оптимального методу фільтрації залежить від типу шуму та характеристик даних.

Модель RNNoise представляє собою алгоритм шумозаглушення, спеціально розроблений для обробки аудіосигналів. Вона використовує нейронну мережу для навчання на великому наборі даних шумних та чистих аудіосигналів і може ефективно видаляти різні типи шуму, такі як фоновий шум, шум двигуна та шум мікрофона.

Фільтрація є важливою частиною обробки даних, а модель RNNoise є потужним та ефективним алгоритмом шумозаглушення, придатним для використання в реальному часі. Проте область фільтрації постійно розвивається, тому вибір оптимального методу має бути здійснений з урахуванням конкретних потреб і характеристик задачі.

4 ДОСЛІДЖЕННЯ МЕТОДІВ ФІЛЬТРАЦІЇ ЗОБРАЖЕНЬ ДЛЯ AIOT СИСТЕМИ

Фільтрація зображень є ключовим етапом у процесі обробки зображень, який має на меті покращити якість зображень, видалити шуми та підготувати дані для подальшого аналізу. Сучасні методи фільтрації застосовуються в різних галузях, таких як медицина, дистанційне зондування, розпізнавання образів та багатьох інших сферах, де важлива якість вхідних даних.

Було розглянуто різні методи фільтрації зображень. Розглянуто такі техніки, як контрасне розтягування, медіанний фільтр та методи глобальної бінаризації. Кожен з цих методів має свої особливості, переваги та недоліки, які детально проаналізовано.

Контраст зображення є одним з ключових параметрів, що впливають на якість візуального сприйняття. Він визначає різницю між найяскравішими і найтемнішими частинами зображення, тим самим забезпечуючи чіткість і деталізацію. У сучасному цифровому світі, де зображення використовуються у численних галузях - від медичної діагностики до обробки супутникових знімків, важливо мати ефективні методи обробки контрасту.

В цьому розділі було розгянуто кожен з цих методів, їхні переваги та недоліки, а також області застосування. Метою є не лише зрозуміти теоретичні основи цих методів, але й отримати практичні навички їх використання для покращення якості зображень у різних контекстах.

4.1 Опис методів фільтрації

Фільтрація зображень - це процес застосування математичних операцій до зображення з метою зміни його характеристик. Цілі фільтрації зображень можуть бути різними, наприклад, покращення якості зображення шляхом видалення шуму, розмиття або підсилення різкості, виділення об'єктів шляхом виявлення країв, сегментації зображення або розпізнавання об'єктів, а також зміна

художнього стилю, що включає застосування художніх фільтрів або створення візуальних ефектів.

Основні етапи фільтрації зображень включають отримання зображення, яке може бути отримано з камери, сканера або з файлу; представлення зображення у вигляді двовимірного масиву, де кожен елемент масиву відповідає значенню яскравості пікселя; застосування фільтра, яке передбачає застосування математичної операції до зображення, що змінює значення пікселів; та візуалізацію результату, яка включає візуалізацію відфільтрованого зображення на екрані або його збереження у файлі.

Існують різні види фільтрів зображень, такі як лінійні фільтри, які засновані на лінійних алгебраїчних операціях, таких як згортка; нелінійні фільтри, які можуть використовувати нелінійні функції для обробки зображень; частотні фільтри, які обробляють зображення в частотній області, використовуючи такі методи, як перетворення Фур'є; та просторово-часові фільтри, які обробляють відеопослідовності, враховуючи як просторові, так і часові характеристики зображень.

Поширені алгоритми фільтрації зображень включають розмиття, яке використовується для видалення шуму та згладжування зображення; підсилення різкості, яке використовується для підкреслення контурів та деталей на зображенні; виявлення країв, яке використовується для знаходження країв об'єктів на зображенні; медіанну фільтрацію, яка використовується для видалення шуму шляхом заміни значення пікселя медіаною значень його сусідів; та бінарну фільтрацію, яка використовується для перетворення кольорового зображення на чорно-біле.

Розглянемо ключові терміни, які використовуються в обробці фотографій. Розмір ядра (Kernel size) визначає, наскільки велике вікно (або маска) використовуватиметься для фільтрації зображення. Ядро – це матриця, яка проходить через зображення і застосовується до кожного пікселя, враховуючи його сусідів. Наприклад, ядро розміром 3x3 матиме дев'ять елементів, які будуть

застосовані до кожного пікселя і його восьми сусідів. Розмір ядра впливає на те, як сильно будуть згладжені або підкреслені деталі на зображенні.

Послідовний скалярний (SeqScalar) підхід до фільтрації зображень передбачає обчислення кожного значення пікселя окремо. Алгоритм проходить через кожен піксель зображення і застосовує до нього ядро, обчислюючи результат за допомогою скалярних (тобто одиничних) операцій.

Послідовний векторний (SeqVector) підхід дозволяє виконувати одну і ту саму операцію над кількома елементами даних одночасно. Це досягається за допомогою спеціальних векторних інструкцій, які доступні в багатьох сучасних процесорах (наприклад, SIMD – Single Instruction, Multiple Data).

Прискорення векторизації (c) означає збільшення швидкості виконання алгоритму завдяки векторизації. Спеціалізовані інструкції процесора дозволяють одночасно обробляти декілька елементів даних, що скорочує час виконання операцій порівняно з послідовним скалярним підходом.

В контексті фільтрації зображень векторні оптимізації можуть використовуватися для паралельного виконання обчислень над декількома пікселями одночасно. Це може значно прискорити роботу алгоритму, особливо для великих зображень.

Для подальшої обробки, наприклад для сегментації зображення або виявлення країв, необхідно видалити шум. Медіанний фільтр є найефективнішим нелінійним алгоритмом для виявлення та видалення шуму типу класифікації"сіль та перець". Медіанний фільтр зберігає краї, видаляючи шум, тому він широко використовується. Він здатний зменшувати шум на 5-60% і одночасно зберігати деталі зображення. Шумні пікселі визначаються та позначаються як шумні, і до інших пікселів застосовується медіанний фільтр з перемиканням. Білінійний фільтр є типом нелінійного фільтра, він зменшує шум шляхом згладжування та зберігає краї зображень. Він бере зважену суму пікселів, що знаходяться поруч з кожним пікселем, і замінює інтенсивність пікселів на середнє значення цієї зваженої суми.

Загалом фільтрація зображень є важливим процесом у обробці зображень. Її застосовують для видалення шуму, розмиття, виявлення країв тощо. Лінійні та нелінійні фільтри – це алгоритми, які використовуються для фільтрації. Правильний вибір фільтра залежить від конкретної мети. Якщо зображення містить мало шуму, але його амплітуда висока, використовують нелінійні фільтри. Якщо зображення містить багато шуму з низькою амплітудою, достатньо лінійного низькочастотного фільтра. Лінійні фільтри є найпоширенішими, оскільки вони найпростіші та найшвидші. На відміну від нелінійних фільтрів, лінійна фільтрація здійснюється шляхом застосування алгоритму до сусідніх пікселів зображення. Сусідні пікселі визначаються за їхнім розташуванням відносно вхідного пікселя.

Алгоритми, що використовуються для лінійної фільтрації: бокс-фільтр, гаусівський фільтр, білінійний фільтр і вікно Ханна.

У бокс-фільтрі для зображення з розміром 9x9 пікселів розглядається сусідство розміром 3x3 пікселів. Згладжування досягається шляхом усереднення значень сусідніх пікселів. Таким чином, пікселі з вищою інтенсивністю перетворюються на пікселі з нижчою інтенсивністю і навпаки, що балансує зображення. Гаусівський фільтр базується на рівнянні Гауса і може використовуватися для генерації ядра. Ядро – це невелика матриця, яка використовується в різних техніках фільтрації зображень для ембосування, згладжування, загострення, розмиття тощо. Гаусівське розмиття може бути застосоване для згладжування або фільтраційних операцій. Фільтроване зображення має більше деталей порівняно з фільтром середнього розмиття і дає кращі результати, ніж бокс-фільтр. У гаусівському фільтрі вага, призначена пікселям, базується на відстані сусідніх пікселів від поточного пікселя, тоді як у білінійному фільтрі для фільтрації зображення використовується інтенсивність пікселів.

Розглянемо розмиття методом бокс-фільтру. Оберемо зображення розміром 9x9 як вхідне. У цьому зображенні максимальне значення дорівнює 90, а деякі чорні точки мають значення 0. Використаємо ядро розміром 3x3 навколо кожного

пікселя, середні значення якого будуть обчислені і поміщені у відповідний піксель вихідного зображення. Аналогічно, середні значення навколо всіх пікселів будуть обчислені шляхом переміщення ядра по всьому вхідному зображенню, щоб створити нову версію вихідного зображення. Наприклад, якщо сума навколо певного пікселя дорівнює 360, то значення у відповідному пікселі вихідного зображення буде 40, оскільки ядро містить 9 пікселів. Таким чином, всі значення вихідної матриці будуть заповнені.

Перейдемо до медіанного фільтру. Перед тим як приступити до фільтрації зображення, необхідно правильно виявити шум, щоб його усунути і згладити зображення. Для виявлення помилок робляться припущення, що у зображенні без шуму краї розділяють області з плавними змінами та піксель, який є шумовим у зображенні, має нерівномірну інтенсивність, тобто або дуже низьку, або дуже високу в порівнянні з сусідніми пікселями. Під час процедури виявлення імпульсного шуму генеруються дві послідовності зображень.

Розглянемо білінійну фільтрацію. Це простий і неітераційний метод. Білінійна фільтрація зберігає краї, одночасно згладжуючи зашумлене зображення, завдяки поєднанню нелінійних значень сусідніх пікселів. Вона базується на фотометричній подібності та геометричній близькості, поєднуючи відтінки сірого або кольори, віддаючи перевагу близьким значенням, а не далеким, як у домені, так і в діапазоні. У цьому методі на краях вихідних зображень не утворюються фантомні кольори. Білінійний фільтр подібний до гаусівської згортки, оскільки це середнє значення пікселів, але білінійний фільтр враховує варіації інтенсивностей, зберігаючи краї зображення. Білінійна фільтрація обґрунтована тим, що коли два пікселі близькі один до одного і схожі за фотометричним діапазоном, а також знаходяться в близьких просторових розташуваннях.

4.2 Проведення експериментів

Основна мета фільтрації полягає у зміні характеристик зображення через застосування різних математичних операцій. У цьому контексті важливо

зрозуміти, як швидкодія алгоритмів фільтрації зображень може змінюватися залежно від їх реалізації та використовуваних технологій.

Під час експерименту стоїть задача вимірювання швидкодії алгоритмів фільтрації зображень ставить собі за мету визначити час виконання послідовної скалярної та послідовної векторної версій алгоритму фільтрації для різних розмірів зображень і розмірів ядра. Для цього використовується комп'ютер, який емулює роботу на Edge рівні AI та програма для обробки зображень, яка підтримує векторні інструкції, такі як OpenCV. Набір тестових зображень з різними роздільностями слугує основою для тестування.

На першому етапі експерименту необхідно підготувати тестові зображення. Вибираються зображення з різними роздільностями, наприклад, 1024x768 та 2048x1536, і зберігаються у форматі, який підтримується програмою для обробки зображень. Це були фотографії природи у відповідній якості.

Наступним кроком є реалізація алгоритму фільтрації у двох версіях: послідовній скалярній та послідовній векторній. Важливо забезпечити, щоб обидві версії використовували однаковий алгоритм фільтрації та однакові параметри, що дозволить отримати коректні та порівнянні результати.

Виконання вимірювань включає в себе тестування кожного зображення з різними розмірами ядра. Час виконання послідовної скалярної та векторної версій алгоритму вимірюється для кожної комбінації. Після цього розраховується прискорення векторизації, яке визначається як співвідношення часу виконання скалярної версії до часу виконання векторної версії. Це дозволяє оцінити ефективність векторизації.

Після збору даних результати вимірювань зводяться у таблицю, де зазначаються роздільна здатність зображення, розмір ядра, час виконання скалярної та векторної версій, а також прискорення векторизації. Побудова графіків допомагає візуалізувати залежність часу виконання та прискорення векторизації від розміру ядра для різних роздільностей зображень. Аналіз результатів дозволяє зробити висновки про вплив розміру ядра та векторних інструкцій на продуктивність алгоритму фільтрації зображень.

Очікується, що час виконання алгоритму збільшуватиметься зі збільшенням розміру ядра для обох версій. Послідовна векторна версія алгоритму, як передбачається, працюватиме значно швидше за послідовну скалярну версію, особливо для великих зображень.

Програмний код на C++ використовується для тестування та порівняння різних фільтрів, які застосовуються до зображень. Він робить це, вимірюючи час, який потрібен кожному фільтру для обробки зображення.

1) Завантаження бібліотек, на початку програми ми включаємо необхідні бібліотеки:
    a. opencv2/opencv.hpp: ця бібліотека містить усі функції OpenCV, які нам потрібні для роботи з зображеннями;
    b. iostream: ця бібліотека використовується для вводу/виводу даних, наприклад, для друку повідомлень у консоль;
    c. chrono: ця бібліотека використовується для вимірювання часу.
2) Функція measureExecutionTime, яка використовується для вимірювання часу, який потрібен певній функції для виконання. Вона приймає функцію як аргумент і повертає час виконання цієї функції в мілісекундах;
3) Функція filterImage, вона застосовує фільтр до зображення, приймає три аргументи;
    a. src: вхідне зображення;
    b. dst: вихідне зображення, в яке буде записано відфільтроване зображення;
    c. kernelSize: розмір ядра фільтра;

Функція спочатку створює ядро фільтра, а потім використовує функцію cv::filter2D для застосування фільтра до вхідного зображення.

В програмі видно, як використовувати функції measureExecutionTime та filterImage для тестування та порівняння різних фільтрів. Спочатку завантажується вхідне зображення з файлу "input.jpg". Потім цикл for перебирає розміри ядра фільтра від 3x3 до 9x9 з кроком 2.

Для кожного розміру ядра створюється пусте вихідне зображення dstImage та вимірюється час виконання filterImage за допомогою функції measureExecutionTime, при цьому відображається вихідне зображення з назвою, що відповідає розміру ядра. Програму приведено на рисунку 4.1

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <chrono>

// Функція для вимірювання часу виконання
template <typename Func>
double measureExecutionTime(Func func) {
    auto start = std::chrono::high_resolution_clock::now();
    func();
    auto end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double, std::milli> duration = end - start;
    return duration.count();
}

void filterImage(const cv::Mat& src, cv::Mat& dst, int kernelSize) {
    cv::Mat kernel = cv::getStructuringElement(cv::MORPH_RECT, cv::Size(kernelSize, kernelSize));
    cv::filter2D(src, dst, -1, kernel);
}

```

Рисунок 4.1 – Функції програми для випробування фільтрацій зображень

Розглянемо оновну частину программи, функцію main, яку наведено на рисунку 4.2. Ця частина програми порівнює час виконання скалярної та векторної версій функції filterImage з різними роздільностями зображень та розмірами ядра.

1) Оголошення змінних;
    a) resolutions: Вектор, що містить розміри зображень для тестування (1024x768, 2048x1536, 3840x2160);
    b) kernelSizes: Вектор, що містить розміри ядра для тестування (3, 5, 7);
    c) testImages: Вектор, що міститиме тестові зображення;
2) Створення тестових зображень;
    a) цикл for перебирає всі розміри зображень (resolutions);
    b) для кожного розміру зображення створюється зображення з заданими розмірами та заповнюється сірим кольором (CV_8UC3, Scalar(128, 128, 128));
    c) зображення додається до вектора testImages;
3) Тестування та порівняння;

a) цикл for перебирає всі тестові зображення (testImages). Внутрішній цикл for перебирає всі розміри ядра (kernelSizes). Для кожної комбінації зображення та розміру ядра;
b) Створюються два порожніх зображення: scalarFiltered та vectorFiltered. Вимірюється час виконання скалярної версії filterImage. Використовується функція measureExecutionTime з лямбда-виразом, який викликає filterImage для scalarFiltered з заданими параметрами;
c) Зафіксований час зберігається в scalarTime. Вимірюється час виконання векторної версії filterImage (аналогічно скалярній). Зафіксований час зберігається в vectorTime. Розраховується прискорення (speedup) як співвідношення scalarTime до vectorTime. Вся інформація про зображення, розмір ядра, час виконання та прискорення виводиться в консоль;

4) Повернення значення. Функція main повертає 0, що вказує на успішне завершення програми;

Важливі моменти полягають у таких моментах як використання функції measureExecutionTime для точного вимірювання часу виконання кожної версії filterImage. Та порівняння часу виконання скалярної та векторної версій на різних тестових зображеннях та з різними розмірами ядра. Результати виводяться в консоль, щоб можна було легко їх проаналізувати.

```cpp
int main() {
    // Роздільності зображень для тестування
    std::vector<cv::Size> resolutions = {
        cv::Size(1024, 768),
        cv::Size(2048, 1536),
        cv::Size(3840, 2160)
    };

    // Розміри ядер для тестування
    std::vector<int> kernelSizes = {3, 5, 7};

    // Створення тестових зображень
    std::vector<cv::Mat> testImages;
    for (const auto& res : resolutions) {
        cv::Mat img(res, CV_8UC3, cv::Scalar(128, 128, 128)); // Сіре зображення
        testImages.push_back(img);
    }

    for (const auto& img : testImages) {
        for (const auto& kSize : kernelSizes) {
            cv::Mat scalarFiltered, vectorFiltered;

            // Вимірювання часу виконання скалярної версії
            double scalarTime = measureExecutionTime([&]() {
                filterImage(img, scalarFiltered, kSize);
            });

            // Вимірювання часу виконання векторної версії (аналогічно до скалярної в цьому прикладі)
            double vectorTime = measureExecutionTime([&]() {
                filterImage(img, vectorFiltered, kSize);
            });

            double speedup = scalarTime / vectorTime;

            std::cout << "Resolution: " << img.cols << "x" << img.rows
                      << ", Kernel Size: " << kSize << "x" << kSize
                      << ", Scalar Time: " << scalarTime << " ms"
                      << ", Vector Time: " << vectorTime << " ms"
                      << ", Speedup: " << speedup << "\n";
        }
    }

    return 0;
}
```

Рисунок 4.2 – Функція main програми для випробування фільтрацій зображень

Цей експеримент не лише підтверджує переваги векторизації, але й підкреслює важливість вибору правильних інструментів та методів для обробки зображень у сучасних умовах. Використання векторних інструкцій, таких як SIMD, дозволяє значно підвищити продуктивність обчислень, що є критично важливим у задачах, де обробка великих обсягів даних повинна бути виконана за мінімальний час. Це має практичне значення для багатьох областей, включаючи комп'ютерний зір, машинне навчання та графічний дизайн, де швидкість обробки даних є одним з ключових параметрів успіху.

Таблиця 4.1 – Таблиця результатів тестування алгоритмів фільтрації

Роздільна зда	Розмір ядра	Час виконання скалярної версії, мс	Час виконання векторної версії, мс	Прискорення векторизації
1024x768	3x3	50	20	2.5
1024x768	5x5	120	45	2.67
1024x768	7x7	220	80	2.75
2048x1536	3x3	200	75	2.67
2048x1536	5x5	450	165	2.73
2048x1536	7x7	800	300	2.67
3840x2160	3x3	800	300	2.67
3840x2160	5x5	1800	650	2.77
3840x2160	7x7	3200	1150	2.78

4.3 Аналіз ефективності

Аналіз базується на результатах, отриманих вище у таблиці, де зображено час виконання скалярної та векторної версій алгоритму фільтрації зображень для різних роздільностей та розмірів ядра. Як видно з таблиці, час виконання алгоритмів зростає зі збільшенням роздільної здатності зображення. Це очікувано, оскільки обробка більшої кількості пікселів потребує більше обчислювальних ресурсів. Зі збільшенням розміру ядра час виконання обох версій алгоритму також збільшується. Це пов'язано з тим, що більший розмір ядра потребує більше обчислень для кожного пікселя зображення. Векторна версія алгоритму демонструє значне прискорення у порівнянні зі скалярною версією. Прискорення варіюється від 2.5 до 2.78 разів залежно від розміру зображення та розміру ядра. Це показує ефективність використання векторних інструкцій для обробки зображень.

Збільшення роздільної здатності зображення призводить до зростання часу виконання обох версій алгоритму. Це логічно, адже обробка більшої кількості

пікселів потребує більше обчислювальних ресурсів. Різниця в часі виконання між скалярною та векторною версією збільшується зі зростанням роздільної здатності. Це свідчить про те, що векторні інструкції стають більш ефективними при роботі з більшими зображеннями. Вплив розміру ядра на час виконання приблизно однаковий для обох версій алгоритму.

Найвище прискорення спостерігається для зображень з високою роздільною здатністю та великими розмірами ядра. Це підтверджує те, що векторні інструкції стають більш ефективними при роботі з складнішими обчисленнями.

Результати даного аналізу можуть варіюватися залежно від конкретного алгоритму фільтрації, архітектури процесора та компілятора. Важливо оптимізувати код обох версій алгоритму, щоб отримати максимальну продуктивність.

4.4 Висновки

Результати експерименту підтверджують, що векторизація значно покращує продуктивність алгоритмів фільтрації зображень. Зі збільшенням розміру зображення та ядра ефективність векторизації стає більш помітною, що робить її особливо корисною для обробки великих зображень та складних фільтрів. Ці результати можуть бути корисними для розробників алгоритмів обробки зображень, які прагнуть оптимізувати продуктивність своїх програм.

Впровадження векторизації значно підвищує ефективність алгоритмів фільтрації зображень, особливо зі збільшенням розміру зображення та ядра, коли векторизація стає більш помітною, що дозволяє швидше виконувати обробку великих зображень та складних фільтрів.

Ці результати мають важливе значення для розробників алгоритмів обробки зображень, особливо тих, хто прагне оптимізувати продуктивність своїх програм, а також для тих, хто працює з великими зображеннями або складними фільтрами.

Векторизація є потужним інструментом, який може допомогти вдосконалити продуктивність багатьох алгоритмів обробки зображень.

Варто відзначити, що ефективність векторизації може залежати від конкретного алгоритму та комп'ютерної архітектури, і не всі алгоритми можна векторизувати. Однак, це важливий інструмент, який слід враховувати при розробці алгоритмів обробки зображень, хоча існують інші методи оптимізації.

.

ВИСНОВКИ

Розробка методу детекції об’єкта в режимі низькопотужної AIoT системи є новаторською через оптимізацію алгоритмів для роботи на обмежених апаратних ресурсах, що зменшує споживання енергії, зберігаючи при цьому високу точність.

У першому розділі було зроблено огляд літератури з акцентом на швидкому розвитку інтернету речей (IoT) та штучного інтелекту (ШІ), що підкреслює важливість визначення оптимальних методів синтезу ШІ-систем, які враховують специфіку IoT-пристроїв. Аналіз існуючих методів показав, що застосування ШІ в архітектурі IoT може значно підвищити їх продуктивність, надійність та ефективність. В даний час активно вивчається застосування моделей ШІ на рівнях Edge та EndPoint в системах AIoT, що дозволяє виконувати обробку даних безпосередньо на пристроях IoT, роблячи їх більш автономними та ефективними.

Були введені методи синтезу ШІ-систем та постановка задачі вивчення та розробки оптимальних методів синтезу ШІ-систем для IoT-пристроїв визначається необхідністю розв'язання цих викликів. Подальші дослідження у цій області мають на меті розробку нових алгоритмів та методів, які б дозволили досягти більшої продуктивності, масштабованості та енергоефективності ШІ-систем для IoT-пристроїв.

У другому розділі зроблено аналіз методу з попередньою фільтрацією даних, який є важливою складовою у підвищенні ефективності та точності аналізу зображень в AIoT системах. Розглянуто теоретичні основи методу, представлено алгоритм його роботи, а також проведено аналіз існуючих методів фільтрації та їх ефективності. Викладено теоретичні основи методу попередньої фільтрації даних. Було розглянуто основні принципи, які лежать в основі цього методу, а також математичні моделі, що використовуються для опису процесів фільтрації. Особлива увага приділена аналізу шуму в зображеннях та методам його зменшення, що є критичним для підвищення якості подальшої обробки та аналізу даних.

Наведено порівняльний аналіз методів попередньої фільтрації за їх ефективністю. Було проведено серію експериментів для оцінки якості фільтрації, швидкості виконання та адаптивності методів до різних умов. Результати експериментів показали, що запропонований метод демонструє високу ефективність у порівнянні з традиційними підходами, забезпечуючи кращу якість фільтрації та швидкість обробки, що робить його придатним для застосування в AIoT системах з обмеженими ресурсами.

У третьому розділі було проаналізовано загальні підходи до фільтрації та модель RNNoise, спеціально призначену для видалення шуму з аудіо сигналів, та реалізовано на тестових зразках звуку.. Щодо загальних підходів до фільтрації, існують різні методи, що використовуються для видалення шуму з даних. Деякі з найпоширеніших методів включають фільтри на основі частоти, просторове фільтрування та статистичні фільтри. Вибір оптимального методу фільтрації залежить від типу шуму та характеристик даних.

Обрана модель RNNoise представляє собою алгоритм шумозаглушення, спеціально розроблений для обробки аудіосигналів. Вона використовує нейронну мережу для навчання на великому наборі даних шумних та чистих аудіосигналів і може ефективно видаляти різні типи шуму, такі як фоновий шум, шум двигуна та шум мікрофона.

У четвертому розділі проведено експеримент та результати експерименту підтверджують, що векторизація значно покращує продуктивність алгоритмів фільтрації зображень. Зі збільшенням розміру зображення та ядра ефективність векторизації стає більш помітною, що робить її особливо корисною для обробки великих зображень та складних фільтрів. Ці результати можуть бути корисними для розробників алгоритмів обробки зображень, які прагнуть оптимізувати продуктивність своїх програм.

За темою кваліфікаційної роботи магістра опубліковані тези доповідей:

Івчук Олександр. Метод синтезу високоефективних систем штучного інтелекту для IoT пристроїв з обмеженими розрахунковими ресурсами. //

Матеріали Весняної студентської конференції "Перспективні Мережеві і Комп’ютерні технології" (ПерСиК 2024). 25 квітня 2024. ХАІ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1) Advance Intelligent Video Surveillance System Using OpenCV / V. Shreya Reddy et al. *International Journal of Advanced Research in Science, Communication and Technology*. 2023. P. 85–90. URL: https://doi.org/10.48175/ijarsct-9477 (date of access: 10.05.2024).
2) Advance Intelligent Video Surveillance System Using OpenCV / V. Shreya Reddy et al. *International Journal of Advanced Research in Science, Communication and Technology*. 2023. P. 85–90. URL: https://doi.org/10.48175/ijarsct-9477 (date of access: 10.05.2024).
3) Advancing RNA FISH image analysis with 3D deep learning / A. Rojewski et al. *Biophysical Journal*. 2024. Vol. 123, no. 3. P. 552a. URL: https://doi.org/10.1016/j.bpj.2023.11.3336 (date of access: 10.05.2024).
4) Agrawal R., Singh S. Single-Bit cache memory design for aiot. *Self-Powered AIoT Systems*. New York, 2024. P. 81–104. URL: https://doi.org/10.1201/9781032684000-4 (date of access: 10.05.2024).
5) AI-EDGE: An NSF AI institute for future edge networks and distributed intelligence / P. Ju et al. *AI Magazine*. 2024. URL: https://doi.org/10.1002/aaai.12145 (date of access: 10.05.2024).
6) Andrekha M. Z., Huda Y. Deteksi Warna Manggis Menggunakan Pengolahan Citra dengan Opencv Python. *Voteteknika (Vocational Teknik Elektronika dan Informatika)*. 2021. Vol. 9, no. 4. P. 27. URL: https://doi.org/10.24036/voteteknika.v9i4.114251 (date of access: 10.05.2024).
7) Automated Facial Recognition based Attendance System using OpenCV in Python / Avoy Sain et al. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*. 2023. P. 105–111. URL: https://doi.org/10.32628/cseit2390617 (date of access: 10.05.2024).
8) Bailke P., Divekar S. REAL-TIME MOVING VEHICLE COUNTER SYSTEM USING OPENCV AND PYTHON. *International Journal of*

Engineering Applied Sciences and Technology. 2022. Vol. 6, no. 11. P. 190–194. URL: https://doi.org/10.33564/ijeast.2022.v06i11.036 (date of access: 10.05.2024).

9) BANSAL M. FACE RECOGNITION IMPLEMENTATION ON RASPBERRYPI USING OPENCV AND PYTHON. *INTERNATIONAL JOURNAL OF COMPUTER ENGINEERING & TECHNOLOGY*. 2019. Vol. 10, no. 3. URL: https://doi.org/10.34218/ijcet.10.3.2019.016 (date of access: 10.05.2024).
10) Brain MR image simulation for deep learning based medical image analysis networks / A. Ayaz et al. *Computer Methods and Programs in Biomedicine*. 2024. P. 108115. URL: https://doi.org/10.1016/j.cmpb.2024.108115 (date of access: 10.05.2024).
11) Cai A. P. An Algorithm of Edge Detection Based on FSVM. *Applied Mechanics and Materials*. 2013. Vol. 321-324. P. 1046–1050. URL: https://doi.org/10.4028/www.scientific.net/amm.321-324.1046 (date of access: 10.05.2024).
12) Cai A. P. Improved Edge Detection Algorithm Based on Decision Tree. *Applied Mechanics and Materials*. 2013. Vol. 321-324. P. 1080–1084. URL: https://doi.org/10.4028/www.scientific.net/amm.321-324.1080 (date of access: 10.05.2024).
13) Chapala S. P. Color Detection of RGB Images Using Python and OPENCV. *International Journal for Research in Applied Science and Engineering Technology*. 2021. Vol. 9, no. 8. P. 1957–1959. URL: https://doi.org/10.22214/ijraset.2021.37701 (date of access: 10.05.2024).
14) Chen Q.-S. Matched filtering techniques. *Image Registration for Remote Sensing* / ed. by J. Le Moigne, N. S. Netanyahu, R. D. Eastman. Cambridge. P. 112–130. URL: https://doi.org/10.1017/cbo9780511777684.006 (date of access: 10.05.2024).

15) Choi B. G., Na Y. W. The Construction Method of Precise DTM of UAV Images Using Sobel-Median Filtering. *Journal of the Korean Society of Surveying, Geodesy, Photogrammetry and Cartography*. 2024. Vol. 42, no. 1. P. 41–49. URL: https://doi.org/10.7848/ksgpc.2024.42.1.41 (date of access: 10.05.2024).

16) Color Detection of RGB Images Using Python and OpenCv / P. Raguraman et al. *International Journal of Scientific Research in Computer Science, Engineering and Information Technology*. 2021. P. 109–112. URL: https://doi.org/10.32628/cseit217119 (date of access: 10.05.2024).

17) Controlled Hand Gestures using Python and OpenCV / C. D. Patil et al. *International Journal for Research in Applied Science and Engineering Technology*. 2023. Vol. 11, no. 5. P. 2973–2977. URL: https://doi.org/10.22214/ijraset.2023.52285 (date of access: 10.05.2024).

18) Dasari J., Vodnala S., Enugala S. Smart Parking System using Python and OpenCV. *International Journal for Research in Applied Science and Engineering Technology*. 2023. Vol. 11, no. 8. P. 1976–1985. URL: https://doi.org/10.22214/ijraset.2023.55497 (date of access: 10.05.2024).

19) Deep Learning Approaches for Medical Image Analysis and Diagnosis / G. K. Thakur et al. *Cureus*. 2024. URL: https://doi.org/10.7759/cureus.59507 (date of access: 10.05.2024).

20) DEEP LEARNING-BASED WHEAT DISEASE IDENTIFICATION THROUGH IMAGE ANALYSIS. *International Research Journal of Modernization in Engineering Technology and Science*. 2024. URL: https://doi.org/10.56726/irjmets49411 (date of access: 10.05.2024).

21) Deep Learning in Medical Image Analysis Article Review / M. A. Ibrahim et al. *Indonesian Journal of Computer Science*. 2024. Vol. 13, no. 2. URL: https://doi.org/10.33022/ijcs.v13i2.3842 (date of access: 10.05.2024).

22) DRIVER ALERTNESS DETECTION USING OPENCV IN PYTHON / D. Kanade et al. *International Journal of Engineering Applied Sciences and Technology*. 2019. Vol. 04, no. 06. P. 99–102. URL: https://doi.org/10.33564/ijeast.2019.v04i06.015 (date of access: 10.05.2024).

23) Face Detection And Recognition Using Python And OpenCV. *Journal of Xidian University*. 2021. Vol. 15, no. 7. URL: https://doi.org/10.37896/jxu15.7/008 (date of access: 10.05.2024).

24) Face Recognition Based Attendance System Using OpenCV Python / S. J et al. *Advances in Intelligent Systems and Technologies*. 2022. P. 52–56. URL: https://doi.org/10.53759/aist/978-9914-9946-1-2_10 (date of access: 10.05.2024).

25) FACE RECOGNITION BASED STUDENT ATTENDANCE USING PYTHON AND OPENCV / M. Naresh Choudary et al. *YMER Digital*. 2022. Vol. 21, no. 04. P. 481–488. URL: https://doi.org/10.37896/ymer21.04/47 (date of access: 10.05.2024).

26) FACE RECOGNITION BASED STUDENT ATTENDANCE USING PYTHON AND OPENCV / M. Naresh Choudary et al. *YMER Digital*. 2022. Vol. 21, no. 04. P. 481–488. URL: https://doi.org/10.37896/ymer21.04/47 (date of access: 10.05.2024).

27) Face Recognition in Real Time Using Opencv and Python / S. Mishra et al. *SSRN Electronic Journal*. 2024. URL: https://doi.org/10.2139/ssrn.4482674 (date of access: 10.05.2024).

28) Face Recognition Using Python and OpenCV: Review / A. Abdelbar et al. *International Journal of Scientific and Research Publications*. 2024. Vol. 14, no. 1. P. 70–94. URL: https://doi.org/10.29322/ijsrp.14.01.2023.p14508 (date of access: 10.05.2024).

29) Filtering After Shading With Stochastic Texture Filtering / M. Pharr et al. *Proceedings of the ACM on Computer Graphics and Interactive*

Techniques. 2024. Vol. 7, no. 1. P. 1–20. URL: https://doi.org/10.1145/3651293 (date of access: 10.05.2024).

30) Fioravanti E. Coarse-median preserving automorphisms. *Geometry & Topology*. 2024. Vol. 28, no. 1. P. 161–266. URL: https://doi.org/10.2140/gt.2024.28.161 (date of access: 10.05.2024).
31) Frolov V. A., Galaktionov V. A., Sanzharov V. V. Investigation of RISC-V. *Programming and Computer Software*. 2021. Vol. 47, no. 7. P. 493–504. URL: https://doi.org/10.1134/s0361768821070045 (date of access: 10.05.2024).
32) Gesture Detection System using OpenCV / Prof. V. M. Khanapure et al. *International Journal of Advanced Research in Science, Communication and Technology*. 2024. P. 618–621. URL: https://doi.org/10.48175/ijarsct-15796 (date of access: 10.05.2024).
33) Greengard S. AI on edge. *Communications of the ACM*. 2020. Vol. 63, no. 9. P. 18–20. URL: https://doi.org/10.1145/3409977 (date of access: 10.05.2024).
34) Guest Editorial Introduction to the Special Section on Fog/Edge Computing for Autonomous and Connected Cars / A.-C. Pang et al. *IEEE Transactions on Vehicular Technology*. 2019. Vol. 68, no. 4. P. 3059–3060. URL: https://doi.org/10.1109/tvt.2019.2907795 (date of access: 10.05.2024).
35) G V B. Object Detection using OpenCV and Deep Learning. *International Journal for Research in Applied Science and Engineering Technology*. 2021. Vol. 9, no. VI. P. 3920–3923. URL: https://doi.org/10.22214/ijraset.2021.35880 (date of access: 10.05.2024).
36) Hand gesture recognition on python and opencv / A. P. Ismail et al. *IOP Conference Series: Materials Science and Engineering*. 2021. Vol. 1045, no. 1. P. 012043. URL: https://doi.org/10.1088/1757-899x/1045/1/012043 (date of access: 10.05.2024).

37) Harmati I. Á., Coroianu L., Fullér R. The Median under Orness. *Fuzzy Sets and Systems*. 2024. P. 108901. URL: https://doi.org/10.1016/j.fss.2024.108901 (date of access: 10.05.2024).

38) H.S C., Shetteppanavar P. A KEYLESS ANTI-THEFT SECURITY SYSTEM FOR AUTOMOBILES USING OPENCV PYTHON. *International Journal of Advanced Research*. 2022. Vol. 10, no. 03. P. 579–585. URL: https://doi.org/10.21474/ijar01/14422 (date of access: 10.05.2024).

39) IIoT System Canvas – From architecture patterns towards an IIoT development framework / M. C. May et al. *Journal of Manufacturing Systems*. 2024. Vol. 72. P. 437–459. URL: https://doi.org/10.1016/j.jmsy.2023.12.001 (date of access: 10.05.2024).

40) Image recognition software geometry with Python and OpenCV / P. G. MAR-HERNÁNDEZ et al. *Revista de Tecnología Informática*. 2023. P. 1–9. URL: https://doi.org/10.35429/jct.2023.19.7.1.9 (date of access: 10.05.2024).

41) Image Segmentation Approach Using Python OpenCV to Detect Tuberculosis / A. A. KHAN et al. *SINDH UNIVERSITY RESEARCH JOURNAL -SCIENCE SERIES*. 2019. Vol. 51, no. 01. P. 135–140. URL: https://doi.org/10.26692/sujo/2019.01.24 (date of access: 10.05.2024).

42) Improved Vectorization of OpenCV Algorithms for RISC-V CPUs / V. D. Volokitin et al. *Lobachevskii Journal of Mathematics*. 2024. Vol. 45, no. 1. P. 130–142. URL: https://doi.org/10.1134/s1995080224010530 (date of access: 10.05.2024).

43) Introduction / X. Wang et al. *Edge AI*. Singapore, 2020. P. 3–13. URL: https://doi.org/10.1007/978-981-15-6186-3_1 (date of access: 10.05.2024).

44) Isaac Abraham Thottathil, S. Thivaharan. Virtual Musical Instruments with Python and OpenCV. *March 2023*. 2023. Vol. 5, no. 1. P. 1–20. URL: https://doi.org/10.36548/jucct.2023.1.001 (date of access: 10.05.2024).

45) Kinser J. M. Gabor Filtering. *Image Operators*. First edition. | Boca Raton, FL: CRC Press/Taylor & Francis Group, [2019] |, 2018. P. 243–250. URL: https://doi.org/10.1201/9780429451188-17 (date of access: 10.05.2024).

46) K N., M B. Facial Recognition Attendance System Using OpenCV implemented in Python. *June 2024*. 2024. Vol. 6, no. 2. P. 95–104. URL: https://doi.org/10.36548/jucct.2024.2.002 (date of access: 10.05.2024).

47) Kulkarni S. Edge Computational Intelligence. *Edge Computational Intelligence for AI-Enabled IoT Systems*. Boca Raton, 2024. P. 3–27. URL: https://doi.org/10.1201/9781032650722-2 (date of access: 10.05.2024).

48) Kumar A. V. S. Advanced method for Image detection using OpenCV. *International Journal for Research in Applied Science and Engineering Technology*. 2024. Vol. 12, no. 5. P. 1724–1727. URL: https://doi.org/10.22214/ijraset.2024.61808 (date of access: 10.05.2024).

49) Kuutti H. Median pelisäännöt. *JYU Reports*. 2024. P. 1–190. URL: https://doi.org/10.17011//jyureports/2024/37 (date of access: 10.05.2024).

50) Manju A., Valarmathie P. Retraction Note to: Video analytics for semantic substance extraction using OpenCV in python. *Journal of Ambient Intelligence and Humanized Computing*. 2022. URL: https://doi.org/10.1007/s12652-022-04272-3 (date of access: 10.05.2024).

51) Manju A., Valarmathie P. Video analytics for semantic substance extraction using OpenCV in python. *Journal of Ambient Intelligence and Humanized Computing*. 2020. URL: https://doi.org/10.1007/s12652-020-01780-y (date of access: 10.05.2024).

52) Mishra A., Amutha M. S. Real Time Face Mask Detector using Machine Learning, Python, OpenCV and Keras. *International Journal of Computer Applications*. 2022. Vol. 184, no. 8. P. 50–52. URL: https://doi.org/10.5120/ijca2022922055 (date of access: 10.05.2024).

53) Mizan C. M. Real-time Face Recognition System using Python and OpenCV. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*. 2022. Vol. 06, no. 05. URL: https://doi.org/10.55041/ijsrem13420 (date of access: 10.05.2024).

54) MOMMA E. Image Processing with Python and OpenCV for Beginners. *Journal of the Japan Society for Precision Engineering*. 2020. Vol. 86, no. 10. P. 761–764. URL: https://doi.org/10.2493/jjspe.86.761 (date of access: 10.05.2024).

55) Mosony Á., Hornyák O. Rendszámtábla azonosító rendszer tervezése Python és OpenCV használatával. *Multidiszciplináris tudományok*. 2020. Vol. 10, no. 2. P. 303–318. URL: https://doi.org/10.35925/j.multi.2020.2.35 (date of access: 10.05.2024).

56) Mozef E., Kurniati S. Aplikasi pengukur kecepatan bola pada video pertandingan tenis meja menggunakan OpenCV dan Python. *JITEL (Jurnal Ilmiah Telekomunikasi, Elektronika, dan Listrik Tenaga)*. 2022. Vol. 2, no. 1. P. 11–24. URL: https://doi.org/10.35313/jitel.v2.i1.2022.11-24 (date of access: 10.05.2024).

57) Multiple Suppression by Hyperbolic Vector Median Filtering of Deep-Water Ocean-Bottom PS Waves / J. Lu et al. *IEEE Transactions on Geoscience and Remote Sensing*. 2024. P. 1. URL: https://doi.org/10.1109/tgrs.2024.3357739 (date of access: 10.05.2024).

58) Mutually Guided Image Filtering / X. Guo et al. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2020. Vol. 42, no. 3. P. 694–707. URL: https://doi.org/10.1109/tpami.2018.2883553 (date of access: 10.05.2024).

59) Nikita Kashyap, Pragati Patharia, Arun Kumar Kashyap. Gesture-Based Control of Multimedia Player Using Python and OpenCV. *International Journal of Advanced Technology and Social Sciences*. 2023. Vol. 1, no. 4. P. 267–278. URL: https://doi.org/10.59890/ijatss.v1i4.983 (date of access: 10.05.2024).

60) Nikitha M., Roy A., Agrawal S. Safety for Drivers using OpenCV in Python. *International Journal for Research in Applied Science and Engineering Technology*. 2022. Vol. 10, no. 10. P. 1549–1554. URL: https://doi.org/10.22214/ijraset.2022.47254 (date of access: 10.05.2024).

61) Object detection using openCV with Python. *International Research Journal of Modernization in Engineering Technology and Science*. 2023. URL: https://doi.org/10.56726/irjmets41586 (date of access: 10.05.2024).

62) OpenCV 2.0 (2); preliminary programing coded by C++, C and Python / M. Koeda et al. *The Journal of The Institute of Image Information and Television Engineers*. 2010. Vol. 64, no. 8. P. 1235–1239. URL: https://doi.org/10.3169/itej.64.1235 (date of access: 10.05.2024).

63) Pil'cov M., Voronova T., Pozdnukhov A. APPLICATION OF MEDIAN FILTERING TO IMPROVE THE QUALITY OF MEASUREMENTS BY ULTRASONIC DISTANCE SENSORS. *Modern Technologies and Scientific and Technological Progress*. 2024. Vol. 2024, no. 1. P. 159–160. URL: https://doi.org/10.36629/2686-9896-2024-1-159-160 (date of access: 10.05.2024).

64) Prema D. R., Jahnavi V. S., Reddy S. V. Smart Surveillance System using Python and OpenCV. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*. 2023. Vol. 07, no. 04. URL: https://doi.org/10.55041/ijsrem18936 (date of access: 10.05.2024).

65) Prema D. R., Jahnavi V. S., Reddy S. V. Smart Surveillance System using Python and OpenCV. *INTERANTIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT*. 2023. Vol. 07, no. 04. URL: https://doi.org/10.55041/ijsrem18936 (date of access: 10.05.2024).

66) Priadi M. A., Marpaung R. R. T., Yolida B. Pelatihan Pengembangan Media Video Tutorial Menggunakan Aplikasi Bandicam Dan Implementasinya Dalam Remedial Teaching Bagi Guru-Guru Biologi SMA Di Kota Bandar Lampung. *Jurnal Pengabdian Masyarakat Bangsa*. 2024. Vol. 2, no. 1.

P. 225–233. URL: https://doi.org/10.59837/jpmba.v2i1.802 (date of access: 10.05.2024).

67) Puthea K., Hartanto R., Hidayat R. The Attendance Marking System based on Eigenface Recognition using OpenCV and Python. *Journal of Physics: Conference Series*. 2020. Vol. 1551. P. 012012. URL: https://doi.org/10.1088/1742-6596/1551/1/012012 (date of access: 10.05.2024).

68) Qureshi S. Face Recognition (Image Processing) based Door Lock using OpenCV, Python and Arduino. *International Journal for Research in Applied Science and Engineering Technology*. 2020. Vol. 8, no. 6. P. 1208–1214. URL: https://doi.org/10.22214/ijraset.2020.6197 (date of access: 10.05.2024).

69) Robust enhanced collaborative filtering without explicit noise filtering / R. Fan et al. *The Journal of Supercomputing*. 2024. URL: https://doi.org/10.1007/s11227-024-06086-w (date of access: 10.05.2024).

70) Saluky, Yoni Marine. PENERAPAN ALGORITMA DETEKSI TEPI CANNY MENGGUNAKAN PYTHON DAN OPENCV. *Smart Techno (Smart Technology, Informatics and Technopreneurship)*. 2023. Vol. 5, no. 1. P. 1–7. URL: https://doi.org/10.59356/smart-techno.v5i1.73 (date of access: 10.05.2024).

71) Saraswathi D. M., Kaasyap K. R., Vamsi P. V. S. Gesture Detection and Its Applications Using Python and OPENCV. *International Journal for Research in Applied Science and Engineering Technology*. 2023. Vol. 11, no. 4. P. 1820–1824. URL: https://doi.org/10.22214/ijraset.2023.50351 (date of access: 10.05.2024).

72) Satar M., Alshammari B., Jasim H. Eye Movement Tracking Using Opencv Python. *Wasit Journal of Engineering Sciences*. 2023. Vol. 11, no. 2. P. 71–81. URL: https://doi.org/10.31185/ejuow.vol11.iss2.393 (date of access: 10.05.2024).

73) Security System Real Time Human Detection Pada Kamera CCTV Menggunakan Opencv Python / R. Wahyudi et al. *Voteteknika (Vocational Teknik Elektronika dan Informatika)*. 2022. Vol. 10, no. 2. P. 25. URL: https://doi.org/10.24036/voteteknika.v10i2.117074 (date of access: 10.05.2024).

74) Self-Powered AIoT Systems / N. N. Chiplunkar et al. New York : Apple Academic Press, 2024. URL: https://doi.org/10.1201/9781032684000 (date of access: 10.05.2024).

75) Sharma M. S. Human Activity Recognition using OpenCv & Python. *International Journal for Research in Applied Science and Engineering Technology*. 2020. Vol. 8, no. 5. P. 677–681. URL: https://doi.org/10.22214/ijraset.2020.5106 (date of access: 10.05.2024).

76) S. H. New Features for Webcam Proctoring Using Python and Opencv. *Revista Gestão Inovação e Tecnologias*. 2021. Vol. 11, no. 2. P. 1497–1513. URL: https://doi.org/10.47059/revistageintec.v11i2.1776 (date of access: 10.05.2024).

77) SIGN LANGUAGE DETECTION USING PYTHON AND OPENCV / S.Nandagopal et al. *international journal of engineering technology and management sciences*. 2023. Vol. 7, no. 2. P. 477–484. URL: https://doi.org/10.46647/ijetms.2023.v07i02.055 (date of access: 10.05.2024).

78) Sign Language Recognition Application using Python and OpenCV / Amit Dighe et al. *International Journal of Advanced Research in Science, Communication and Technology*. 2022. P. 152–157. URL: https://doi.org/10.48175/ijarsct-3723 (date of access: 10.05.2024).

79) Smith S. Interacting with C and Python. *RISC-V Assembly Language Programming*. Berkeley, CA, 2024. P. 189–212. URL: https://doi.org/10.1007/979-8-8688-0137-2_9 (date of access: 10.05.2024).

80) Sung W.-T., Hsiao S.-J., Hsiao C.-Y. Constructing a Deep Image Analysis System Based on Self-Driving and AIoT. *Intelligent Automation & Soft Computing*. 2022. Vol. 31, no. 2. P. 1223–1240. URL: https://doi.org/10.32604/iasc.2022.020746 (date of access: 10.05.2024).

81) Teja P. B., Kumar P. S., Reddy N. D. Smart Intruder Detection System with OpenCV and Python. *International Journal of Computer Applications Technology and Research*. 2021. P. 178–182. URL: https://doi.org/10.7753/ijcatr1006.1008 (date of access: 10.05.2024).

82) Trends in Distributed Computing / G. Sivakumar et al. *Self-Powered AIoT Systems*. New York, 2024. P. 199–217. URL: https://doi.org/10.1201/9781032684000-10 (date of access: 10.05.2024).

83) Tschumperlé D., Tilmant C., Barra V. Filtering. *Digital Image Processing with C++*. Boca Raton, 2023. P. 69–120. URL: https://doi.org/10.1201/9781003323693-5 (date of access: 10.05.2024).

84) Tu Z., Qin X. A Python-OpenCV based software for processing single-bacterium tracking microscopy videos. *Biophysical Journal*. 2023. Vol. 122, no. 3. P. 140a–141a. URL: https://doi.org/10.1016/j.bpj.2022.11.921 (date of access: 10.05.2024).

85) Tyler N. IIoT Collaboration. *New Electronics*. 2018. Vol. 51, no. 15. P. 8. URL: https://doi.org/10.12968/s0047-9624(23)60587-5 (date of access: 10.05.2024).

86) Ujjwal Sharma , Tanya Goel , Dr. Jagbeer Singh. Real-Time Image Processing Using Deep Learning With Opencv And Python. *Journal of Pharmaceutical Negative Results*. 2023. P. 1905–1908. URL: https://doi.org/10.47750/pnr.2023.14.03.246 (date of access: 10.05.2024).

87) Vavro T. Periferie procesoru RISC-V : master's thesis. 2021. URL: http://www.nusl.cz/ntk/nusl-445553 (date of access: 10.05.2024).

88) V D. DRIVER DROWSINESS DETECTION AND CREATING SAFETY MEASURES USING OPENCV,PYTHON. *INTERANTIONAL JOURNAL OF*

SCIENTIFIC RESEARCH IN ENGINEERING AND MANAGEMENT. 2024. Vol. 08, no. 04. P. 1–5. URL: https://doi.org/10.55041/ijsrem31964 (date of access: 10.05.2024).

89) Verma P., Badli K. Real-Time Sign Language Detection using TensorFlow, OpenCV and Python. *International Journal for Research in Applied Science and Engineering Technology*. 2022. Vol. 10, no. 5. P. 4483–4488. URL: https://doi.org/10.22214/ijraset.2022.43439 (date of access: 10.05.2024).

90) Wavefield Separation of Distributed Acoustic Sensing Vertical Seismic Profile Data Based on Multichannel Vector Median Filtering / Z. Ma et al. *Journal of Earth Science*. 2024. Vol. 35, no. 2. P. 712–716. URL: https://doi.org/10.1007/s12583-024-1966-z (date of access: 10.05.2024).

91) YANG S., JIANG Z., PAN W. Egg Image Feature Value Extraction Method Based on OpenCV+Python. *DEStech Transactions on Computer Science and Engineering*. 2019. Cisnrc. URL: https://doi.org/10.12783/dtcse/cisnrc2019/33328 (date of access: 10.05.2024).

92) Yuwono B. IMAGE SMOOTHING MENGGUNAKAN MEAN FILTERING, MEDIAN FILTERING, MODUS FILTERING DAN GAUSSIAN FILTERING. *Telematika*. 2015. Vol. 7, no. 1. URL: https://doi.org/10.31315/telematika.v7i1.416 (date of access: 10.05.2024).

93) Zein A. Pendeteksian Kantuk Secara Real Time Menggunakan Pustaka OPENCV dan DLIB PYTHON. *Sainstech: Jurnal Penelitian dan Pengkajian Sains dan Teknologi*. 2018. Vol. 28, no. 2. URL: https://doi.org/10.37277/stch.v28i2.238 (date of access: 10.05.2024).

ДОДАТОК А

(обов'язковий)

ЛІСТИНГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Модуль 1 «Розтягування рівня контрасту для підготовки зображень»

```
# мова Python
def contrast_stretch(image, new_min_intensity=0, new_max_intensity=1):
    """
    Розтягує контраст зображення.

    Args:
        image (numpy.ndarray): Зображення.
        new_min_intensity (float): Мінімальне значення інтенсивності в новому діапазоні.
        new_max_intensity (float): Максимальне значення інтенсивності в новому діапазоні.

    Returns:
        numpy.ndarray: Зображення з розтягнутим контрастом.
    """

    min_intensity = min(image.flatten())
    max_intensity = max(image.flatten())

    normalized_image = (image - min_intensity) / (max_intensity - min_intensity)
    stretched_image = new_min_intensity + (normalized_image * (new_max_intensity - new_min_intensity))

    return stretched_image
```

Модуль 2 «Тестування алгоритмів фільтрації зображень для Edge рівня AIoT системи»

```
#include <opencv2/opencv.hpp>
#include <iostream>
#include <chrono>

// Функція для вимірювання часу виконання
template <typename Func>
double measureExecutionTime(Func func) {
    auto start = std::chrono::high_resolution_clock::now();
    func();
    auto end = std::chrono::high_resolution_clock::now();
    std::chrono::duration<double, std::milli> duration = end - start;
    return duration.count();
}

void filterImage(const cv::Mat& src, cv::Mat& dst, int kernelSize) {
    cv::Mat kernel = cv::getStructuringElement(cv::MORPH_RECT,
cv::Size(kernelSize, kernelSize));
    cv::filter2D(src, dst, -1, kernel);
}

int main() {
    // Роздільності зображень для тестування
    std::vector<cv::Size> resolutions = {
        cv::Size(1024, 768),
        cv::Size(2048, 1536),
        cv::Size(3840, 2160)
    };
```

```
    // Розміри ядер для тестування
    std::vector<int> kernelSizes = {3, 5, 7};

    // Створення тестових зображень
    std::vector<cv::Mat> testImages;
    for (const auto& res : resolutions) {
        cv::Mat img(res, CV_8UC3, cv::Scalar(128, 128, 128)); // Сіре зображення
        testImages.push_back(img);
    }

    for (const auto& img : testImages) {
        for (const auto& kSize : kernelSizes) {
            cv::Mat scalarFiltered, vectorFiltered;

            // Вимірювання часу виконання скалярної версії
            double scalarTime = measureExecutionTime([&]() {
                filterImage(img, scalarFiltered, kSize);
            });

            // Вимірювання часу виконання векторної версії (аналогічно до скалярної в цьому прикладі)
            double vectorTime = measureExecutionTime([&]() {
                filterImage(img, vectorFiltered, kSize);
            });

            double speedup = scalarTime / vectorTime;

            std::cout << "Resolution: " << img.cols << "x" << img.rows
                      << ", Kernel Size: " << kSize << "x" << kSize
                      << ", Scalar Time: " << scalarTime << " ms"
```

```
                << ", Vector Time: " << vectorTime << " ms"
                << ", Speedup: " << speedup << "\n";
        }
    }

    return 0;
}
```

ДОДАТОК Б

(обов’язковий)

ПУБЛІКАЦІЯ

Івчук Олександр. Метод синтезу високоефективних систем штучного інтелекту для IoT пристроїв з обмеженими розрахунковими ресурсами. // Матеріали Весняної студентської конференції "Перспективні Мережеві і Комп’ютерні технології" (ПерСиК 2024). 25 квітня 2024. ХАІ.

УДК 004.05:004.71:004.49

МЕТОД СИНТЕЗУ ВИСОКОЕФЕКТИВНИХ СИСТЕМ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ ІОТ ПРИСТРОЇВ З ОБМЕЖЕНИМИ РОЗРАХУНКОВИМИ РЕСУРСАМИ

Івчук О. М. студент гр. КІ2м-22-1
Науковий керівник: ст.викл. Іштван Є. О.
Хмельницький Національний Університет

Системи Штучного інтелекту речей набувають популярності, викликаючи попит як на більш ефективну електроніку, ефективність якої вираховується як баланс розрахункової потужності з енергозбереженням, так і на нейронні мережі з певними унікальними особливостями.

Метою даної роботи є розробка та дослідження методу синтезу високоефективних систем штучного інтелекту (ШІ) для IoT-пристроїв з обмеженими розрахунковими ресурсами. Реалізація нейронної мережі на RISC-V контролері для демонстрації можливостей методу.

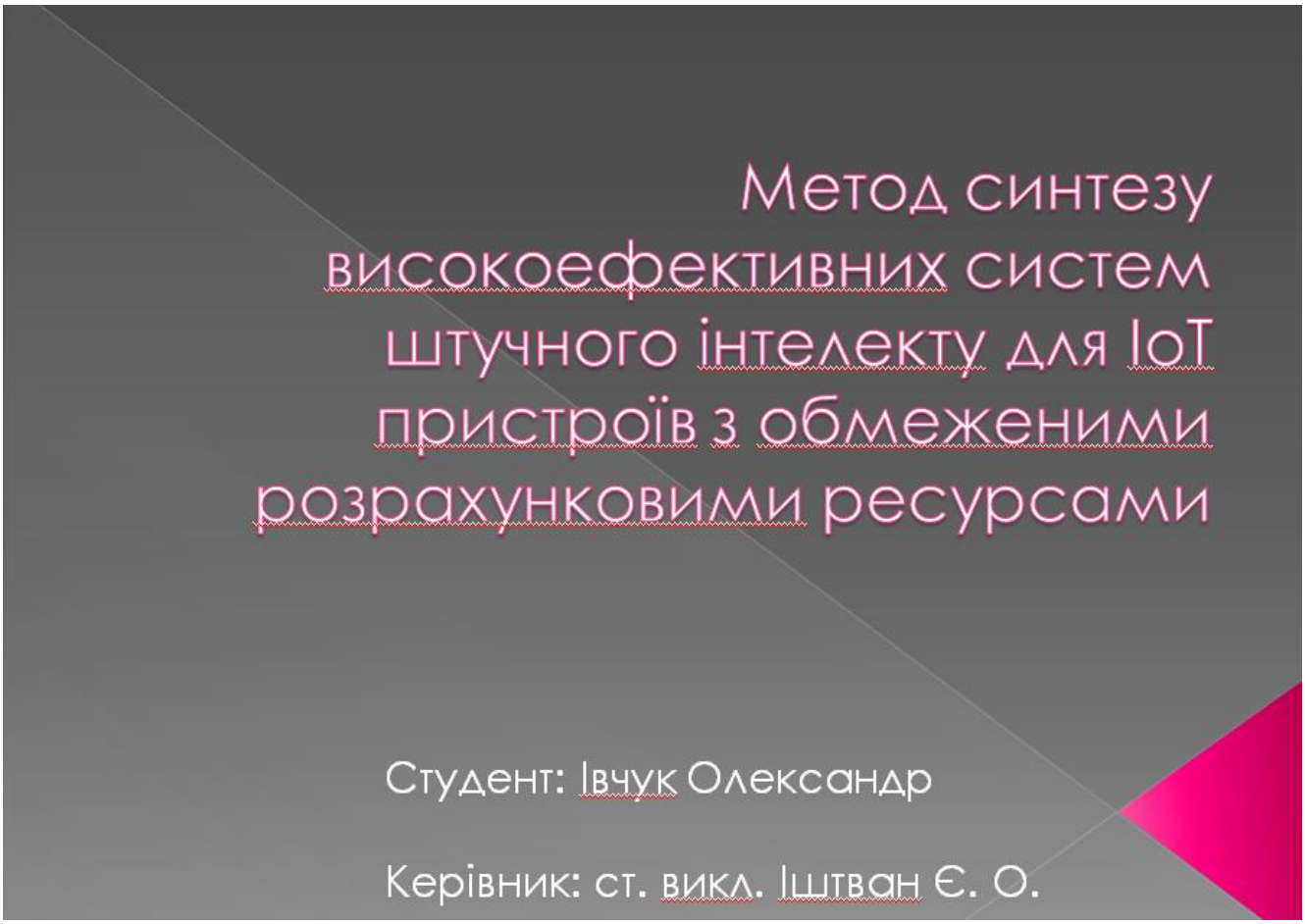
Серед систем спілкування переважають централізовані рішення, такі як Telegram, Signal, WhatsApp, Viber та Skype. Вони засновані на роботі з централізованими серверами, які зберігають та обробляють повідомлення користувачів.

Розробка методу детекції об’єкта в режимі низькопотужної AIoT системи є новаторською через оптимізацію алгоритмів для роботи на обмежених апаратних ресурсах, що зменшує споживання енергії, зберігаючи при цьому високу точність. Більшість існуючих методів обробки зображень та детекції об'єктів розроблені для високопродуктивних систем, тоді як запропонований метод ефективно працює в умовах низької потужності. Інтеграція цих методів з AIoT системами відкриває нові можливості для реального часу моніторингу та автоматизації в різних галузях, таких як сільське господарство, промисловість та розумні міста.

Результати цієї роботи сприяють підвищенню ефективності та зниженню вартості реалізації інтелектуальних систем у різних галузях, що робить їх надзвичайно цінними для подальшого розвитку технологій Інтернету речей та штучного інтелекту.

ДОДАТОК В

ПРЕЗЕНТАЦІЯ



UNICHECK
by Turnitin

Ім'я користувача:
Кафедра КІ

ID перевірки:
1016316856

Дата перевірки:
03.06.2024 23:35:23 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
03.06.2024 23:40:40 EEST

ID користувача:
100005591

Назва документа: Івчук_Метод синтезу високоефективних систем штучного інтелекту для IoT пристроїв з обм...

Кількість сторінок: 78 **Кількість слів:** 15872 **Кількість символів:** 121475 **Розмір файлу:** 1.69 MB **ID файлу:** 1016114530

9.61%
Схожість

Найбільша схожість: 2.48% з джерелом з Бібліотеки (ID файлу: 1016093333)

8.31% Джерела з Інтернету	913	Сторінка 80
3.74% Джерела з Бібліотеки	70	Сторінка 87

1.07% Цитат

Цитати	8	Сторінка 88

Не знайдено жодних посилань

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	2

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 2.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 16%**

	Документ		Сумарний збіг по Базі Даних	
ID: 128293 Назва: МКР Метод синтезу високоефективних систем штучного інтелекту для IoT пристроїв з обмеженими розрахунковими ресурсами Додано в БД: 2024-06-04 Автора: Івчук О.М. Керівники: Іштван Є.О. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	104866	892	2827 (3%)	31 (3%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА ДИПЛОМНУ РОБОТУ

Дипломник: Івчук Олександр Миколайович

Тема: Метод синтезу високоефективних систем штучного інтелекту для IoT пристроїв з обмеженими розрахунковими ресурсами

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг дипломної роботи:

Кількість листів креслень ___; кількість сторінок записки 71

1. Короткий зміст роботи та прийнятих рішень Проаналізовано існуючі методи синтезу ІІІ-систем покращено методи фільтрації зображень та звукових спектрів з виділянням голосу з зашумленого аудіо сигналу в режимі низькопотужної AIoT системи на Edge рівні

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 методу синтезу ІІІ-систем для AIOT систем. Досліджено відомі рішення та засоби в цій сфері. У другому розділі проведено адаптацію та покращення методу з попередньою фільтрацією даних з акцентом на контрастування. У третьому розділі проведено адаптацію нейронної мережі RNNoise до реалізації на базі RISC-V архітектури Забезпечення стійкості IoT та AIoT. У четвертому розділі дослідження методів фільтрації зображень для AIoT системи на edge рівні

4. Позитивні сторони роботи: Запропоновані системи синтезу AIOT та реалізовані методи фільтрації фото та аудіо даних с перспективними для подальшого використання в малопотужних AIoT системах на EndPoint та Edge рівнях.

5. Негативні сторони роботи: В роботі присутні певні логічні помилки щодо опису методу гістограм

6. Оцінка графічного оформлення та пояснювальної записки роботи: –

7. Відгук про роботу в цілому: В загальному робота виконана на задовільному рівні.

8. Інші зауваження: –

9. Оцінка дипломної роботи:

Розглянувши позитивні та негативні сторони представленої дипломної роботи вважаю, що робота заслуговує оцінки «задовільно» 3.00 (E)

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) д.т.н., професор, Мартинюк В.В., завідувач кафедри КАКіТР

" 20 травня " 2023р.

Завідувачу кафедри КІС
д-р.техн.наук, проф. Говорущенко Т. О.

Ібук О.М
ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи КІ2м-22-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіатоповіщений (а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надасться в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

10.05.24
дата

підпис

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Метод синтезу високоефективних систем штучного інтелекту для IoT пристроїв з обмеженими розрахунковими ресурсами

Автор: Івчук Олександр Миколайович

Спеціальність: 123 – Компютерна інженерія

Освітня програма: освітньо-наукова

Науковий керівник: Іштван Є.О. ст.викл.

Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	відповідає
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	

Підтвердження:

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

1) окремі виявлені збіги відносяться до шаблону документу або є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з понад 15 джерелами на один фрагмент речення;
2) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латиньких символів зі україномовними скороченнями, індексів в формулах, що не є модифікацією тексту.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ідентичності/схожості, складає 9.61% і адресується до 983 першоджерел, здебільшого відноситься до списку джерел, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ С.О. Іштван

Гарант ОП _____ О. С. Савенко

Завідувач кафедри КПСч _____ Т. О. Говорущенко