ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«СИСТЕМА З EDGE-ОБЧИСЛЕННЯМИ ДЛЯ АВТОНОМНОГО ПРИЙНЯТТЯ РІШЕНЬ У РЕЖИМІ РЕАЛЬНОГО ЧАСУ»

на здобуття освітнього ступеня магістр
за спеціальності 126 Інформаційні системи та технології
(код,найменування спеціальності)
освітньо-професійної програми Інформаційні системи та технології
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Денис РЕБРОВ
(підпис) *(ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти
група ІСДМ-61 Денис РЕБРОВ
(ім'я, ПРІЗВИЩЕ)

Керівник Ольга ПОЛОНЕВИЧ
к.т.н. *(ім'я, ПРІЗВИЩЕ)*

Рецензент:
(ім'я, ПРІЗВИЩЕ)

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти магістр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ

Завідувач кафедрою ІСТ
Каміла СТОРЧАК_____
“____” _____2025 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Реброву Денису Олександровичу
(прізвище, ім’я, по батькові здобувача)

1.Тема кваліфікаційної роботи: Система з edge-обчисленнями для автономного прийняття рішень у режимі реального часу.
керівник кваліфікаційної роботи: Ольга ПОЛОНЕВИЧ к.т.н., доцент
(ім’я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від “___” жовтня 2025 р. № ____
2. Строк подання кваліфікаційної роботи «26» грудня 2025 р.
3. Вихідні дані кваліфікаційної роботи:
1. Методи обчислень.
2. Edge-обчислення в прогнозному моделюванні.
3. Архітектура системи з edge-обчислень.
4. Наукова література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
1. Дослідження тенденцій розвитку edge-обчислень.
2. Аналіз та дослідження методів інтеграції edge-обчислень в систему для автономного прийняття рішень у режимі реального часу.
3. Аналіз результатів впровадженої технології edge-обчислення в систему для для автономного прийняття рішень у режимі реального часу.

5.Перелік ілюстраційного матеріалу: *презентація*
6. Дата видачі завдання «___» _____ 2025р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Підбір технічної літератури	05.09-01.10	
2.	Дослідження методів обчислень	02.10-14.10	
3.	Дослідження Edge-обчислень в прогнозному моделюванні	14.10-17.10	
4.	Результати впровадженої технології edge-обчислень в систему для для автономного прийняття рішень у режимі реального часу	17.10-24.11	
5.	Висновки по роботі	25.11	
6.	Розробка демонстраційних матеріалів, доповідь.	01.12-06.12	
7.	Оформлення магістерської роботи	06.12-10.12	

Здобувач вищої освіти _____ Денис РЕБРОВ
(підпис) *(ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Ольга ПОЛОНЕВИЧ
(підпис) *(ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття ступеня магістр: 99 стор., 28 рис., 18 табл., 30 джерел.

Мета роботи – дослідження методів інтеграції системи з edge-обчисленнями для автономного прийняття рішень у режимі реального часу.
Об'єкт дослідження – система з edge-обчисленнями для автономного прийняття рішень у режимі реального часу.

Предмет дослідження – предметом дослідження є методи, моделі та архітектурні рішення edge-обчислень, що забезпечують автономне прийняття рішень у режимі реального часу.

Короткий зміст роботи. Перший розділ магістерської роботи детально описує edge-обчислення, його архітектуру та моделі, паралельно порівнюючи його з іншими існуючими методами обчислень. Другий розділ містить інформацію про прогнозне моделювання та його зв'язок з edge-обчисленнями. Третій розділ є одним із, нескладних у реалізації, прикладів системи з edge-обчисленнями для автономного прийняття рішень в режимі реального часу, на базі якої було проведено моделювання, і отримано певні результати ефективності системи. Четвертий розділ представляє концепт більш складної та розумної системи для автоматизації транспортних засобів. На основі проведених симуляцій були отримані результати які зображені у вигляді графіків, таблиць і малюнків, в даному розділі.

КЛЮЧОВІ СЛОВА: EDGE-ОБЧИСЛЕННЯ, ПЕРИФЕРІЯ, ІНТЕРНЕТ РЕЧЕЙ, EDGE-AI, МОДЕЛЬ, ОБЧИСЛЕННЯ, ХМАРНІ ОБЧИСЛЕННЯ, CLOUD-AI, МАШИННЕ НАВЧАННЯ, ТУМАННІ ОБЧИСЛЕННЯ, АВТОНОМНІ ТРАНСПОРТНІ ЗАСОБИ, НЕСПРИЯТЛИВІ ПОГОДНІ УМОВИ, МЕТОД.

ABSTRACT

The text part of the qualifying work for obtaining a bachelor's degree: 99 pp., 28 fig., 18 tables, 30 sources.

The purpose of the work is to study methods for integrating a system with edge computing for autonomous decision-making in real time.

The object of the research is a system with edge computing for autonomous decision-making in real time.

The subject of the research is the methods, models and architectural solutions of edge computing that provide autonomous decision-making in real time.

Summary of the work. The first section of the master's thesis describes in detail edge computing, its architecture and models, while comparing it with other existing computing methods. The second section contains information about predictive modeling and its connection with edge computing. The third section is one of the simple-to-implement examples of a system with edge computing for autonomous decision-making in real time, on the basis of which modeling was conducted and certain results of the system's efficiency were obtained. The fourth section presents the concept of a more complex and intelligent system for vehicle automation. Based on the simulations, the results were obtained, which are depicted in the form of graphs, tables and figures in this section.

KEYWORDS: EDGE-COMPUTING, PERIPHERY, INTERNET OF THINGS, EDGE-AI, MODEL, COMPUTING, CLOUD COMPUTING, CLOUD-AI, MACHINE LEARNING, FOG COMPUTING, AUTONOMOUS VEHICLES, ADVERSE WEATHER CONDITIONS, METHOD.

ЗМІСТ

ВСТУП

Актуальність теми. Протягом останніх шести десятиліть галузь обчислювальних систем зазнала значних змін. Розвиток таких технологій, як Інтернет та комерція, стали основою для широкого поширення обчислювальних технологій. Завдяки постійному технологічному прогресу, обчислювальні системи безперервно розвиваються й адаптуються для задоволення різноманітних потреб суспільства. Це, в свою чергу, стало причиною виникнення нових парадигм, таких як хмарні технології, туманні обчислення, периферійні обчислення та Інтернет речей (IoT), що відкривають нові можливості для економічного і творчого розвитку. Незважаючи на безперервний розвиток технологій, швидкі зміни породжують нові проблеми. Для того, щоб підтримувати рівень економічної продуктивності, що відповідає дедалі жорсткішим вимогам, важливо зрозуміти рушійні сили виникнення нових моделей обчислень, а також розрізняти, чим сучасні виклики відрізняються від попередніх. Зростаюча залежність від обчислювальних ресурсів, призвела до ускладнення та масштабування платформ, що в свою чергу стало поштовхом для розробки нових обчислювальних систем. Вони суттєво покращили функціональні можливості та підвищили вимоги до обчислювального обладнання завдяки швидкому технічному розвитку та орієнтації на користувача. Глобалізація мережевих стандартів дозволяє обмінюватися даними в межах глобальних мереж, що значно покращило ефективність їх взаємодії. В умовах постійного розвитку технологій компанії почали інтегрувати датчики із вбудованими компонентами для підключення до мережі, створюючи нові архітектури та стандарти, які дозволяють передавати завдання віддаленим обчислювальним ресурсам. Це призвело до розширення можливостей технологій за межі традиційних мережевих вузлів через такі інновації, як Інтернет речей (IoT) та периферійні обчислення. Що у свою чергу також підкреслює актуальність тематики цієї роботи.

Мета роботи - дослідження методів інтеграції системи з edge-обчисленнями

для автономного прийняття рішень у режимі реального часу.

Для досягнення мети, у магістерській роботі успішно виконано наступні завдання:

- Дослідження різних видів обчислень.
- Дослідження тенденцій розвитку та поширення застосування edge-обчислень.
- Огляд та розробка методів впровадження edge-обчислень для побудови системи з автономним прийняттям рішень у режимі реального часу.
- Дослідження проблем які можуть виникати під час впровадження в систему edge-обчислень.
- Основні виклики для інтеграції edge-обчислень в систему.
- Побудування моделей на основі edge-обчислень.
- Написання моделі системи з edge-обчисленнями для автономного прийняття рішень у режимі реального часу.

Об'єкт дослідження - система з edge-обчисленнями для автономного прийняття рішень у режимі реального часу.

Предмет дослідження - предметом дослідження є методи, моделі та архітектурні рішення edge-обчислень, що забезпечують автономне прийняття рішень у режимі реального часу.

Методи дослідження. Під час написання магістерської кваліфікаційної роботи були використані методи теоретичного дослідження, імітаційного моделювання та програмування.

Наукова новизна одержаних результатів. У ході мого дослідження різноманітних методів обчислення, мною було виявлено, що використання edge-обчислень є найбільш ефективним та надійним рішенням для роботи системи автономного прийняття рішень у режимі реального часу. Результати наведені в розділах три та чотири підтверджують даний факт.

Практична значущість одержаних результатів. Запропонована система забезпечує автономне прийняття рішень в режимі реального часу.

Апробація результатів магістерської роботи. Основні положення і

результати магістерської роботи доповідались на науково практичних конференціях, що проходили на базі Державного університету інформаційно-комунікаційних технологій.

1. АРХІТЕКТУРА ТА МОДЕЛІ EDGE-COMPUTING

1.1. Історія систем обчислення, їх різновиди та головні відмінності

Технології обчислень зазнали значних змін протягом останніх десятиліть, що зумовлено постійним розвитком нових моделей, здатних задовольнити зростаючі потреби сучасних суспільств і економік. Серед таких моделей особливу увагу привертають високопродуктивні обчислення, мобільні та хмарні технології, автономні системи та інші інноваційні підходи для обробки даних. З кожним роком обчислювальні системи стають дедалі складнішими і гнучкішими, а також здатними інтегрувати різноманітні обчислювальні ресурси для вирішення специфічних задач. Водночас ці зміни відкривають нові можливості для досліджень та технологічного прогресу, створюючи підгрунтя для подальшої еволюції в галузі інформаційних технологій.

Таблиця 1.1.

Різновиди обчислень.

Вид обчислень	Опис
Система високопродуктивних обчислень	Високопродуктивні обчислення є важливим обчислювальним методом для вирішення складних, ресурсоємних задач. У таких системах планувальник керує доступом до різноманітних обчислювальних ресурсів, що дозволяє ефективно розподіляти ресурси для вирішення конкретних задач. Ці системи об'єднують потужні обчислювальні потужності, що дозволяє їм виконувати завдання шляхом одночасного використання різних обчислювальних ресурсів. Зокрема, це включає паралельну обробку даних і оптимізацію доступу до онлайн-ресурсів для максимального ефекту.

Продовження табл. 1.1

Вид обчислень	Опис
Автономні обчислення	Автономні обчислення включають як замкнуті, так і розімкнуті цикли керування, де складні системи можуть мати декілька окремих мереж для управління. В результаті такий підхід дозволяє автоматизувати безліч процесів.
Мобільні обчислення	Мобільні обчислення охоплюють доволі таки широкий спектр IT-компонентів, які забезпечують мобільність та простоту користувачів для доступі до обчислювальних ресурсів, інформації та обладнання. Це, зокрема, можливість отримувати дані та використовувати програми в будь-якому місці, без прив'язки до стаціонарного місця.
Хмарні обчислення	Хмарні обчислення стали важливим етапом в еволюції обчислювальних технологій, дозволяючи зберігати і обробляти дані через віддалені сервери в Інтернеті. Хмарні сервіси, такі як Software as a Service, Platform as a Service та Infrastructure as a Service, надають користувачам доступ до потужних обчислювальних ресурсів без необхідності витрат на інфраструктуру.
Туманні обчислення	Туманні обчислення, є вдосконаленою моделлю, яка широко використовує мобільні пристрої для зберігання та обробки даних, зберігаючи при цьому залежність від Інтернету для з'єднання між вузлами.

Продовження табл. 1.1

Вид обчислень	Опис
Безсерверні обчислення	Модель безсерверних обчислень усуває необхідність управління серверами та іншою інфраструктурою. Користувачам більше не потрібно турбуватися про обслуговування серверів, що спрощує використання обчислювальних ресурсів і знижує витрати на інфраструктуру.
Осмотичні обчислення.	Осмотичні обчислення поєднують в собі концепції Інтернету речей, хмарних і туманних технологій, а також периферійних обчислень.
Квантові обчислення.	Квантові обчислення є радикально новим підходом до аналізу даних, що використовує принципи квантової механіки. Використовуючи явища квантової заплутаності та суперпозиції, квантові комп'ютери дозволяють здійснювати обчислення, недоступні для класичних комп'ютерних систем.
Периферійні обчислення.	Периферійні обчислення (edge computing) означають обробку даних безпосередньо на пристроях, що знаходяться на межі мережі. Це дозволяє значно зменшити затримки, знизити витрати на передачу даних і покращити якість обслуговування. Периферійні обчислення дозволяють ефективно обробляти великі обсяги даних без необхідності їх передачі в хмару, що робить їх ідеальними для таких застосувань, як Інтернет речей (IoT), автономні транспортні системи і промислові додатки.

1.2. Архітектура та характеристики Edge-computing.

Як вже раніше зазначалось, периферійні обчислення - це розподілена обчислювальна архітектура, яка обробляє дані ближче до джерела створення даних, а не в централізованому центрі обробки даних чи хмарі. Останні технологічні досягнення та зростаюча стурбованість користувачів щодо надійності, конфіденційності й автономності, вимагають управління обчислювальними ресурсами, даними та послугами безпосередньо на краю/межі Інтернет мережі, а не на центральних її вузлах (ядро). Централізоване розміщення віддалених обчислювальних ресурсів обмежує їх здатність обробляти величезні обсяги даних, що надходять від географічно розподілених периферійних пристроїв. У зв'язку з цим необхідно переміщати сервери ближче до краю мережі, і цей процес стає неминучим. Кілька факторів роблять периферійні обчислення необхідними, і ці фактори повинні бути враховані як з точки зору кінцевих користувачів, так і операторів.

Філософія периферійних обчислень в першу чергу базується на виконанні обчислень на периферії, яка знаходиться поблизу джерела даних. Ресурси периферійних обчислень можуть бути мережею або обчислювальним ресурсом, що працює між кінцевими користувачами з одного боку, та туманними вузлами та хмарними центрами обробки даних з іншого. Взаємопов'язані датчики/пристрої на рівні Інтернету речей генерують та передають дані один одному за допомогою сучасної інфраструктури комунікаційної мережі. Згенеровані дані обробляються на інших рівнях, що визначаються конкретними вимогами застосування. Архітектуру периферійних обчислень можна пояснити чотирма рівнями:

- рівень Інтернету речей,
- периферійний рівень,
- туманний рівень та
- хмарний рівень.

Рисунок №1.1 ілюструє архітектуру периферійних обчислень. У цій архітектурі рівень Інтернету речей включає мільйони пристроїв/датчиків, які

постійно генерують дані, обмінюються важливою інформацією між собою через сучасну інфраструктуру комунікаційної мережі, а також контролюють критично важливі інфраструктури. Усі ці пристрої є кінцевими користувачами для периферійних обчислень.

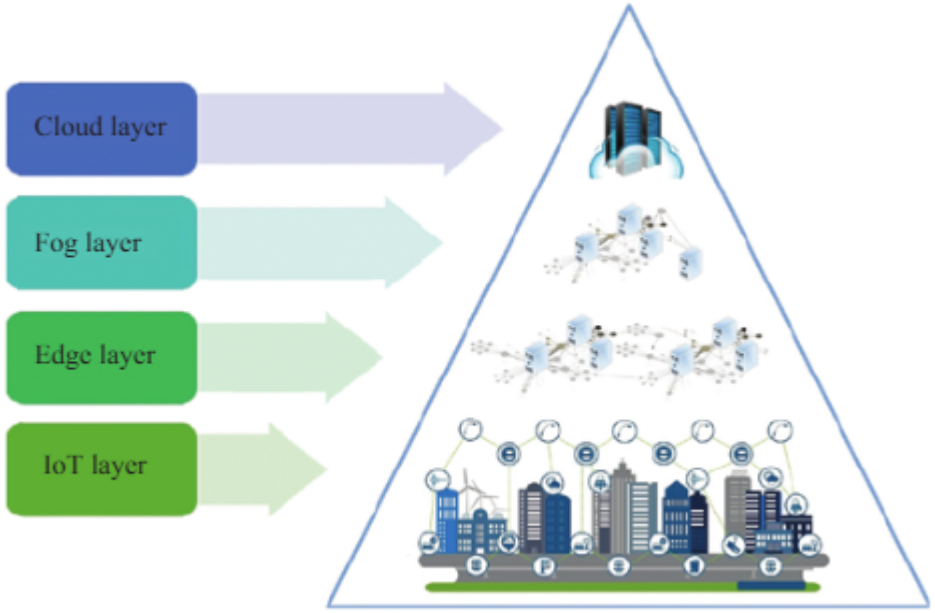


Рис. 1.1 Архітектура периферійних обчислень. [28]

Технології інтернету речей та периферійні обчислення швидко розвиваються незалежно один від одного. Однак платформа периферійних обчислень може допомогти галузі Інтернету речей у вирішенні низки ключових проблем та покращенні продуктивності. Загалом, пристрої Інтернету речей можуть отримати вигоду від високої обчислювальної потужності та великого обсягу пам'яті периферійних, туманних та хмарних обчислювальних систем. Однак периферійні обчислення мають додаткові переваги над туманними та хмарними обчисленнями для Інтернету речей. Інтернет речей вимагає, зокрема, швидкого часу відгуку, а не високої обчислювальної потужності та великого обсягу пам'яті. Периферійні обчислення пропонують прийнятну обчислювальну потужність та достатній простір для зберігання і швидкого часу відгуку що в результаті задовольнятиме вимоги для IoT-пристроїв. Натомість і edge-обчислення можуть отримати вигоду

від IoT, розширивши власну структуру, щоб вона могла обробляти розподілені вузли. Пристрої IoT можна використовувати як периферійні вузли для надання послуг. Оскільки кількість розумних пристроїв продовжує зростати, IoT та периферійні обчислення, скоріше за все можуть стати просто - нероздільними.

Edge-обчислення дозволяють зменшити затримку і заощадити пропускну здатність для програм, чутливих до затримок. Як периферійні, так і туманні обчислення характеризуються розподіленою, ієрархічною та децентралізованою архітектурою, на відміну від централізованої структури хмарних обчислень. Важливою особливістю є також їхнє розташування: периферійні обчислення здійснюються на пристроях на краю мережі, тоді як туманні обчислення відбуваються безпосередньо в мережі.[28]

Таблиця 1.2.

Характеристики Інтернету речей, периферійних обчислень, туманних обчислень та хмарних обчислень.

Характеристика	**IoT**	**Edge**	**Fog (туманні)**	**Cloud (хмарні)**
Розгортання	Розподілене	Розподілене	Розподілене	Централізоване
Компоненти	Фізичні пристрої	Вузли на межі	Вузли “туману”	Віртуальні ресурси
Усвідомлення місцеположення	Усвідомлює	Усвідомлює	Усвідомлює	Не усвідомлює
Обчислювальні ресурси	Обмежені	Обмежені	Обмежені	Необмежені

Продовження табл. 1.1

Характеристика	**IoT**	**Edge**	**Fog (туманні)**	**Cloud (хмарні)**
Зберігання	Дуже обмежені	Обмежене	Обмежене	Необмежене
Дані	Джерело	Обробляється	Обробляється	Обробляється
Відстань до джерела даних	Джерело	Найближчі	Найближчі	Далеко
Час відповіді	Немає часу на відповідь	Найшвидший	Швидкий	Повільний
Кількість вузлів	Найбільша	Дуже велика	Велика	Мала

Порівняно з хмарними обчисленнями, ресурси периферійних та туманних обчислень (наприклад, обчислювальні, комунікаційні та сховищні ресурси) досить обмежені. Іноді моделі периферійних обчислень та туманних обчислень вважаються одним рівнем. Однак між ними все ще існують деякі фундаментальні відмінності, навіть якщо вони мають приблизно однакові цілі. Периферійні пристрої не можуть реалізовувати кілька IoT додатків на рівні периферійних обчислень, оскільки обмежені ресурси спричиняють конкуренцію за ресурси, а також збільшують затримку обробки. Коли як, туманні обчислення можуть успішно подолати ці обмеження та запобігти конкуренції за ресурси на периферії шляхом плавного інтегрування периферійних пристроїв з хмарними ресурсами. Туманні обчислення координують, балансують та покращують використання географічно розподілених мережевих периферійних пристроїв і використовують

хмарні ресурси для цієї мети. Крім того, периферійні обчислення зосереджені на речах, тоді як туманні обчислення більше зосереджені на інфраструктурі. Зверніть увагу на таблицю 1.2 у якій додатково продемонстровано порівняння характеристик усіх рівнів.

1.3. Сфери застосування Edge-computing

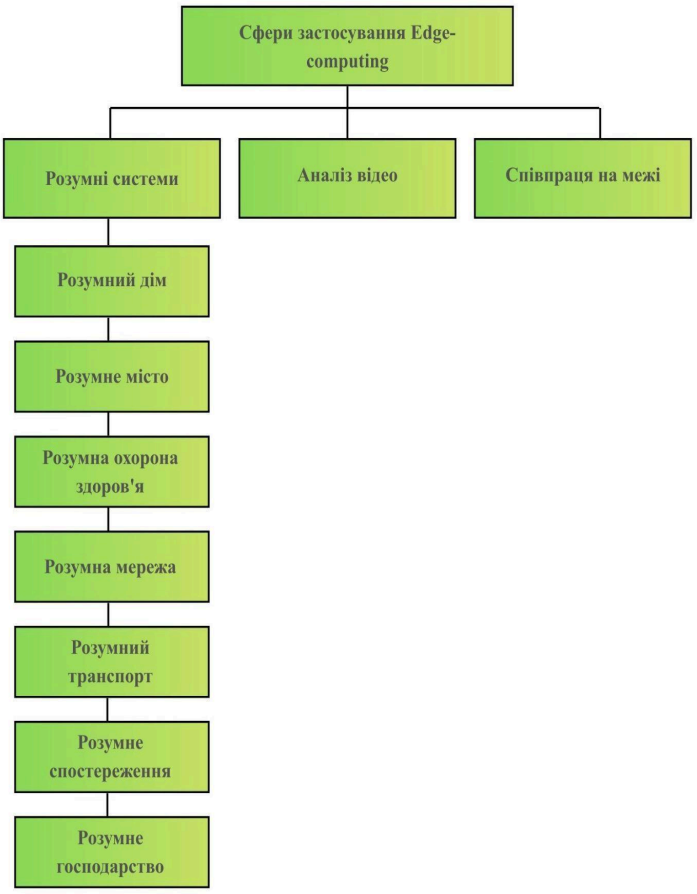


Рис. 1.2. Сфери застосування Edge-computing.

У розумних системах, мережеві технології поєднуються з датчиками та виконавчими механізмами, що передають сигнали до системи для здійснення необхідних дій. Інтеграція датчиків відкриває широкі можливості для збирання даних, керування фізичними процесами, а також для ефективного розподілу й оптимізації ресурсів. Використання в розумних системах периферійних обчислень дає можливість мінімізувати затримку та, за потреби, збільшити обсяг

доступного сховища на пристроях із низькою обчислювальною потужністю. До того ж, аналіз даних на периферії може підвищувати стійкість системи. Розумні системи охоплюють низку важливих сфер, зокрема розумні будинки, розумні енергомережі, розумні міста, розумне сільське господарство, розумний транспорт, розумну медицину та інші напрямки.

Інтернет речей має значний потенціал у домашньому середовищі. На ринку вже доступні такі рішення, як розумне освітлення, смарт телевізори, роботизовані пилососи та інше. Проте створення розумного будинку – це, не просто додати модуль Wi-Fi до звичайного пристрою й підключити його до хмари. У приміщеннях потрібно також розмістити численні бездротові датчики й контролери (див. Рисунок 1.3).

Рис. 1.3. Різновиди датчиків Інтернету речей. [19]

Вони генеруватимуть великі обсяги даних, які бажано обробляти локально, щоб уникнути надмірного навантаження на мережу та мінімізувати ризики для конфіденційності. Саме тому периферійні обчислення є оптимальною моделлю для реалізації розумного будинку. Пристрої можуть легко підключатися та керуватися через периферійний шлюз, що працює під управлінням спеціалізованої операційної системи EdgeOS. В результаті такий підхід дозволяє

обробляти дані на місці, зменшувати використання пропускної здатності Інтернету та забезпечувати якісніше управління й доставку сервісів.

Модель периферійних обчислень може використовуватись як для окремого будинку так і для рівня цілої громади чи навіть усього міста. У такому підході обчислення виконуються максимально близько до джерела даних. Це дає змогу формувати та обробляти запити безпосередньо на периферії, що значно підвищує швидкість прийняття рішень. Розумне місто спирається на велику кількість розподілених пристроїв Інтернету речей, оснащених різними датчиками. Такі пристрої застосовують для покращення транспортного управління, регулювання дорожнього руху, моніторингу якості повітря, освітлення та автоматизованого догляду зелених зон. Периферійні обчислення є ідеальною основою для розумного міста завдяки своїм ключовим властивостям:

- Великий обсяг даних: через перевантаження трафіком неможливо створити централізовані хмарні центри обробки даних, які могли б обробляти величезні обсяги даних (що генеруються системами громадської безпеки, охорони здоров’я, комунальних послуг, транспорту тощо). У цьому випадку edge-обчислення можуть стати ефективним рішенням, оскільки дані оброблятимуться на периферії мережі.
- Низька затримка: периферійні обчислення також добре працюють із програмами, які потребують низької затримки. Наприклад, для тих що використовуються в надзвичайних ситуацій у сфері охорони здоров’я або громадської безпеки, оскільки ця модель може заощадити час передачі даних та спростити структуру мережі. Ефективніше приймати рішення та проводити діагностику на периферії мережі, аніж збирати інформацію та приймати рішення в центральній хмарі.
- Географічне розміщення: Завдяки усвідомленню місцезнаходження, периферійні обчислення працюють краще, ніж хмарні обчислення, коли використовуються для географічно обумовлених застосувань, таких як транспортні та комунальні послуги. У цій моделі дані можуть збиратися та

оброблятися в їхньому географічному розташуванні без перенесення їх до хмари.

Інтернет речей може значно покращити охорону здоров’я, зокрема через бездротові датчики та хмарні технології, що дозволяють зменшити навантаження на пристрої та забезпечити обробку великих обсягів даних. Однак просте використання архітектури датчик-хмара має обмеження через нормативні обмеження зберігання даних пацієнтів та ризики безпеки через мережеві збої. Периферійні обчислення є ідеальним рішенням для таких ситуацій, дозволяючи обробляти дані безпосередньо на місці та знижувати навантаження на мережу. Бездротові та портативні датчики дозволяють здійснювати постійний моніторинг пацієнтів, покращуючи їх мобільність і доступність до медичних послуг. Завдяки аналізу великих даних і машинному навчанню можна покращити ефективність лікування, виявлення хронічних захворювань, а також оптимізувати процеси та створювати персональні плани догляду. Інтернет речей дозволяє використовувати спільну інфраструктуру для зберігання і аналізу даних з різних сенсорів, що значно підвищує ефективність медичних послуг. Цей підхід знижує витрати на охорону здоров’я, забезпечуючи дистанційний моніторинг пацієнтів, що дозволяє зберігати високу якість медичної допомоги та зменшенню навантаження на медперсонал.

При розгортанні пристроїв моніторингу, розумна мережа збирає великі обсяги даних про енергоспоживання з неймовірною швидкістю. Традиційні методи обробки даних покращують ефективність обчислень у часі, але підходять лише для окремих завдань у межах розумної мережі. Крім того, ці рішення не враховують географічний розподіл обчислювальних потужностей в масштабах великої мережі. Розумна мережа вважається наступним етапом розвитку мережевих технологій. Для повноцінного використання її можливостей, таких як безпека, захист і самовідновлення, потрібно інтегрувати численні розумні лічильники, датчики та виконавчі механізми для збору та обміну даними. Периферійні обчислення можуть стати ідеальним рішенням для задоволення вимог розгортання такої мережі.

Для забезпечення безпеки та ефективності автономних транспортних засобів необхідно розгорнути хмарну систему управління транспортом, яка збиратиме дані з датчиків через мережу взаємодії між ними. Така система дозволяє координувати й управляти великою кількістю транспортних засобів.

Інтелектуальні системи спостереження здійснюють контроль, скринінг та відстеження активності для забезпечення безпеки в публічних місцях. Вони об’єднують технології, такі як сканування об’єктів і людей, ведення бази даних потенційних загроз, біометричну ідентифікацію та відеоспостереження, для моніторингу ситуації. Основні функції таких програм включають контроль доступу до зон, ідентифікацію осіб та виявлення подій. Вони також використовуються для моніторингу дорожнього руху, пацієнтів у лікарнях і активності в публічних просторах, таких як торгові центри, промислові об’єкти, банки та державні установи. Будь-яка система спостереження повинна ефективно стискати великі відеофайли для подальшої обробки. Хмарні програми для інтелектуального спостереження стикаються з проблемами, оскільки вони потребують обробки відеопотоків в реальному часі, зібраних з численних розподілених джерел, таких як мережеві камери і мобільні пристрої. Проте передача великих обсягів даних в хмару може перевантажити комунікаційні мережі і створити проблеми з конфіденційністю, оскільки дані стають більш вразливими до атак. Периферійні обчислення забезпечують необхідну для системи спостереження здатність працювати в реальному часі, вирішуючи проблеми з затримками та безпекою.

Для задоволення зростаючих потреб у виробництві продуктів харчування сільське господарство повинно інтегрувати Інтернет речей в усі етапи виробництва, управління та аналізу. Розумні ферми можуть використовувати периферійні обчислення для управління автономними транспортними засобами, такими як трактори, а також для здійснення дистанційного моніторингу та аналітики в реальному часі. IoT пристрої та датчики надають дані про врожайність, рівень опадів, зараження шкідниками та стан грунту. Ця інформація

не тільки допомагає в поточному виробництві, але й дозволяє вдосконалювати сільськогосподарські методи в довгостроковій перспективі.

1.4. Edge-computing в автономних системах керування.

Периферійні обчислення покращують спосіб обробки та аналізу даних в автономних системах, переносячи обчислення з централізованих хмарних серверів на локальні пристрої, розташовані ближче до джерел даних, такі як датчики та виконавчі механізми. Такий підхід дозволяє приймати рішення в режимі реального часу, зменшує затримку та полегшує обмеження пропускної здатності, що робить її особливо цінною для автономних систем, які потребують швидкої реакції. Традиційні моделі хмарних обчислень часто призводять до затримок через передачу великих обсягів даних датчиків на віддалені сервери для обробки. Ці затримки можуть бути негативними в сценаріях, де важливі мілісекунди, таких як запобігання зіткненням в автономних транспортних засобах або навігація в режимі реального часу в дронах. Периферійні обчислення вирішують цю проблему, забезпечуючи локальну обробку даних, що значно скорочує час, необхідний системі для прийняття обгрунтованих рішень. Обробляючи дані датчиків безпосередньо на периферійних пристроях (таких як вбудовані процесори, шлюзи або мобільні пристрої), автономні системи можуть швидше реагувати на зміни в навколишньому середовищі, підвищуючи як безпеку, так і ефективність роботи. Більше того, периферійні обчислення пропонують масштабованість та гнучкість у розгортанні автономних систем, оскільки автономні системи часто працюють у середовищах з ненадійним підключенням, периферійні обчислення гарантують, що критично важливі операції можуть продовжуватися навіть за відсутності стабільного інтернету. Локалізована обробка даних також дозволяє системі адаптуватися до змін, не покладаючись на безперервний хмарний зв'язок. Окрім обробки даних у режимі реального часу, периферійні обчислення відіграють ключову роль у забезпеченні передового машинного навчання та прогнозного моделювання. Завдяки

потужнішим можливостям обробки автономні системи можуть використовувати прогнозні алгоритми для прогнозування майбутніх станів та подій, таких як потенційні перешкоди або оптимальні шляхи, що ще більше підвищує точність прийняття рішень та продуктивність системи. Підсумовуючи, периферійні обчислення є ключовим фактором для наступного покоління автономних систем, забезпечуючи обчислювальну потужність, необхідну для прийняття рішень у режимі реального часу, покращуючи швидкість реагування системи та гарантуючи надійну роботу в динамічних та непередбачуваних ситуаціях.

Обробка даних у реальному часі в автономних системах є важливою для забезпечення швидкого та інтелектуального прийняття рішень у середовищах, що постійно змінюються. Однак досягнення необхідної продуктивності в застосунках реального часу створює низку суттєвих труднощів, особливо в системах, які залежать від великомасштабних даних датчиків, складних алгоритмів та непередбачуваних умов експлуатації. Проблеми обробки даних у реальному часі в автономних системах можна загалом розділити на проблеми, пов'язані з обробкою даних, обчисленнями та комунікацією. У цьому розділі досліджуються ці проблеми та пропонуються рішення шляхом інтеграції периферійних обчислень, об'єднання даних датчиків та передових методів обробки.

Одним із найважливіших завдань обробки даних у реальному часі є мінімізація затримки, тобто затримки між збором даних та прийняттям рішень. В автономних системах навіть невеликі затримки можуть призвести до неоптимальних дій або навіть до системного збою.

Автономні системи покладаються на численні датчики, такі як камери, лідарні сканери, радари та ультразвукові датчики, (див. Рисунки 1.3-1.4) для сприйняття навколишнього середовища. Безперервний потік великих обсягів даних, що генеруються цими датчиками, може перевантажувати канали зв'язку та обмежувати можливість своєчасної передачі цих даних до центральних процесорів. Крім того, обмеження пропускної здатності стають більш очевидними, коли кілька пристроїв або систем передають дані одночасно.

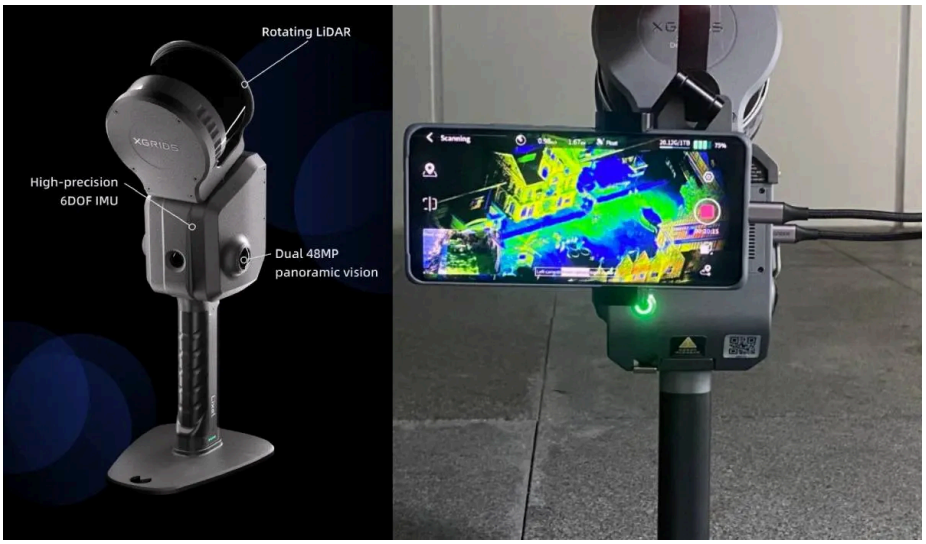


Рис. 1.4. Лідарний датчик.

Автономні системи повинні не лише реагувати на поточне середовище, але й передбачати майбутні події чи зміни, щоб приймати активні рішення. Впровадження прогнозного моделювання в режимі реального часу створює труднощі через необхідність постійного оновлення даних, обробки динамічних середовищ та виконання складних алгоритмів без надмірних обчислювальних витрат. Автономні системи часто працюють у середовищах з ненадійним підключенням, що ускладнює використання виключно хмарних обчислень для безперервної обробки даних та прийняття рішень.

Автономні системи часто вимагають використання складних алгоритмів для таких завдань, як виявлення об'єктів, планування шляху та прийняття рішень. Ці алгоритми можуть бути ресурсоємними, що може бути складно для ефективного виконання з обмеженими обчислювальними ресурсами, доступними на периферійних пристроях.

2. ПРОГНОЗНЕ МОДЕЛЮВАННЯ.

2.1. Роль прогнозного моделювання та типи його моделей.

В автономних системах прогнозне моделювання виконує кілька ключових функцій:

- *Виявлення та уникнення перешкод:* прогнозні моделі можуть прогнозувати рух динамічних об'єктів, таких як пішоходи чи інші транспортні засоби, аналізуючи їхню швидкість та траєкторію. Це дозволяє автономній системі своєчасно вживати заходів уникнення, забезпечуючи запобігання зіткненням та підвищуючи безпеку.
- *Планування та оптимізація шляху:* Прогнозні моделі допомагають автономним системам оптимізувати свої маршрути, прогнозуючи майбутні стани навколишнього середовища, такі як схеми руху, дорожні умови або зміни погоди. Це гарантує, що система може вибрати найефективніший та найбезпечніший маршрут у режимі реального часу.
- *Прогнозування поведінки:* Автономні системи можуть передбачати поведінку інших об'єктів у своєму середовищі. Ця здатність є важливою для спільного прийняття рішень та безперебійної взаємодії між автономними системами та людьми або іншими автономними агентами.
- *Прогнозування відмов та технічне обслуговування:* Прогнозне моделювання також може бути використане для передбачення потенційних системних збоїв або потреб у технічному обслуговуванні, забезпечуючи працездатність автономної системи та мінімізуючи час простою. Аналізуючи дані отримані з датчиків, моделі можуть виявляти ранні ознаки зносу компонентів або погіршення продуктивності.

2.2. Типи прогнозних моделей.

В автономних системах застосовується широкий спектр методів прогнозного моделювання, які дозволяють системі передбачати майбутні стани середовища,

оцінювати поведінку об’єктів та приймати оптимальні рішення в режимі реального часу. Кожен тип моделі вирішує конкретні завдання та має свої переваги залежно від характеристик навколишнього середовища та вимог до точності й швидкодії. Моделювання часових рядів є однією з ключових технологій для передбачення розвитку динамічних процесів. У таких моделях аналізується послідовність значень, що змінюються з часом, після чого будується прогноз на основі виявлених закономірностей та трендів. До найпоширеніших методів належать ARIMA, LSTM та GRU-мережі, експоненціальне згладжування.

ARIMA є узагальненою моделлю яка адаптується до даних часових рядів або для кращого розуміння даних, або для прогнозування. Моделі ARIMA застосовуються у випадках нестаціонарності даних. Модель ARIMA визначається таким чином:

$$X_t - \alpha_1 X_{t-1} - \ldots - \alpha_{p'} X_{t-p'} = \epsilon_t + \theta_1 \epsilon_{t-1} + \ldots + \theta_\varphi \epsilon_{t-\varphi}$$

Також дану модель можна представити в такому вигляді:

$$(1 - \Sigma_{i=1}^{p} \alpha_i L^i) X_t = (1 + \Sigma_{i=1}^{\varphi} \theta_i L^i) \epsilon_t.$$

В обох випадках L - є оператором запізнення (лаг), d_i - є параметром авторегресійної частини моделі θ_i - параметри рухомої середньої частини, ε_t - помилкові члени.[6]

LSTM/ДКЧП (довга короткочасна пам'ять) - це архітектура рекурентних нейронних мереж, запропонована в 1997 році Зеппом Хохрайтером та Юргеном Шмідгубером.

Як і більшість рекурентних нейронних мереж, мережа ДКЧП є універсальною в тому сенсі, що за достатньої кількості вузлів мережі вона може обчислювати будь-що, що може обчислювати звичайний комп'ютер, за умови, що вона має належну матрицю вагових коефіцієнтів, яка може розглядатися як її програма.[7]

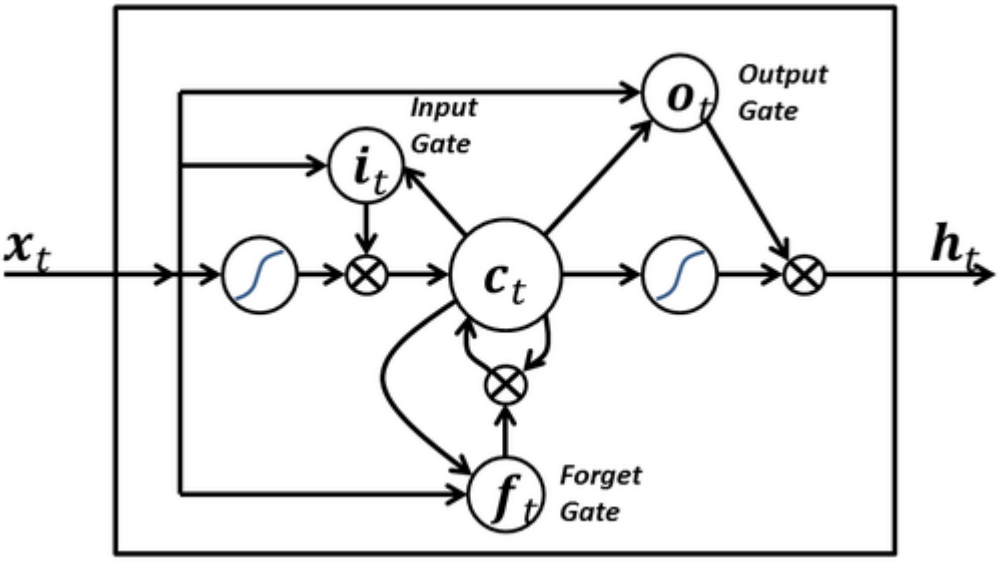


Рис. 2.1. Блок вічкової ДКЧП з входовим, виходовим та забувальним вентилями.[7]

GRU/BPB (вентильні рекурентні вузли) - це, вентильний механізм у рекурентних нейронних мережах, представлений в 2014 році. Вони схожі з довгою короткочасною пам'ятю з вентилем забування, однак на відміну мають меншу кількість параметрів, оскільки не мають вентиля виходу.[8]

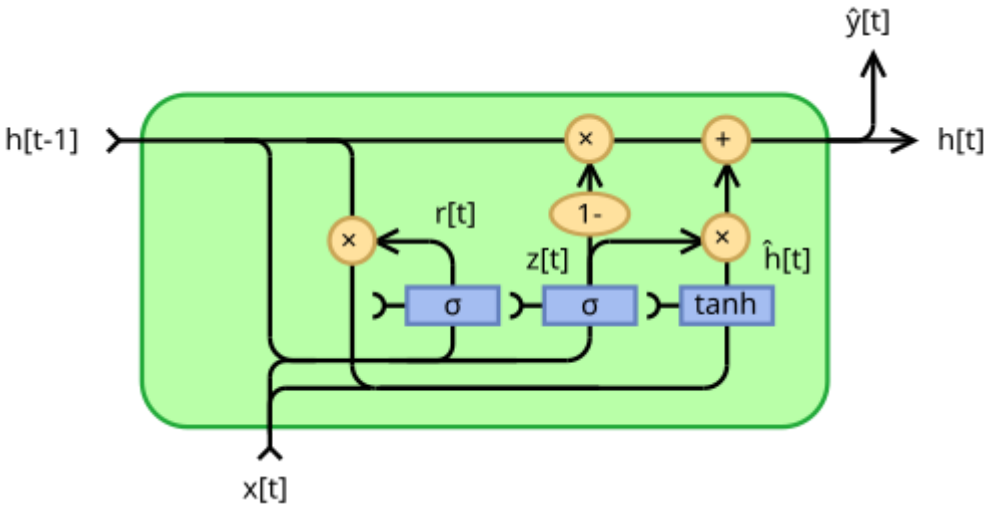


Рис. 2.2. Приклад повного рекурентного вузла.[8]

Експоненційне згладжування - це, метод математичного перетворення, який застосовується при прогнозуванні часових рядів. При кожній наступній ітерації

враховуються всі попередні значення ряду, але ступінь врахування зменшується за експоненційним законом.[9]

$$s_t = \{ c_1 : t = 1$$

$$s_{t-1} + \alpha \cdot (c_t - s_{t-1}) : t > 1 \}$$

Де, s_t - згладжений ряд, c_t - первинний ряд, d - коефіцієнт згладжування.

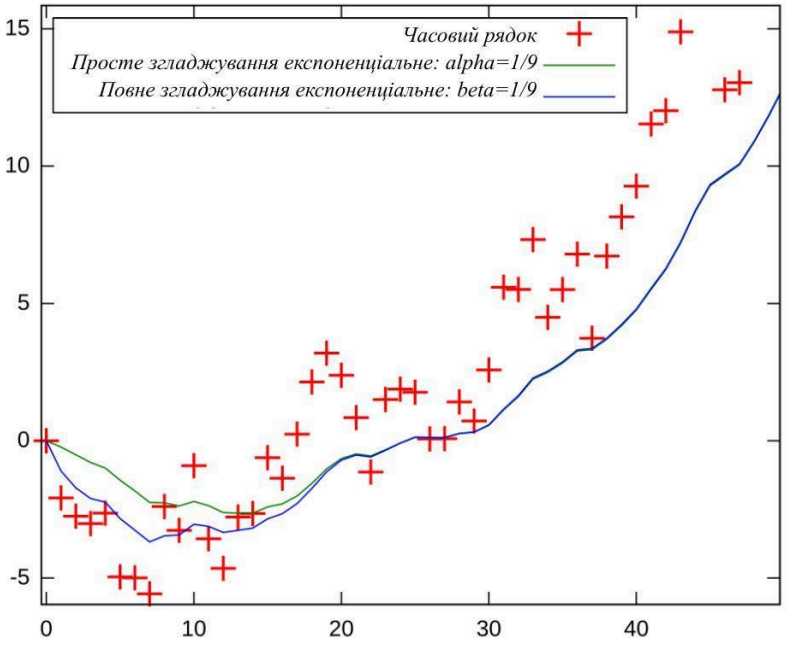


Рис. 2.3. Експоненціальне згладжування.[9]

У контексті автономних систем часові ряди використовуються для прогнозування траєкторії руху транспортних засобів, передбачення зміни відстані до об’єктів, оцінки зміни швидкості або прискорення, аналізу станів сенсорів і систем стабілізації.

Регресійні моделі застосовуються тоді, коли необхідно прогнозувати безперервні числові значення, що залежать від кількох вхідних параметрів, що є одним з базових інструментів прогнозування у технічних системах.

Таблиця 2.1.

Основні типи регресійних підходів.

Лінійна регресія	Проста інтерпретована модель для прогнозування параметрів за наявності лінійної залежності.
Поліноміальна регресія	Розширює можливості лінійної, моделюючи складніші криволінійні залежності.
Регресія на основі “дерев” рішень	Здатна моделювати нелінійні залежності та працювати зі складними структурами даних.
Регресія з регуляризацією	Знижує ризик перенавчання.

У автономних системах регресійні моделі застосовуються для прогнозування швидкості інших транспортних засобів, передбачення прискорення чи гальмівного маневру об’єктів, оцінки параметрів датчиків (температура, вібрації, навантаження), визначення зміни траєкторій у короткостроковій перспективі. Ці моделі мають високу швидкість обчислень та добре підходять для виконання на периферійних пристроях.

Методи ML та DL дозволяють будувати більш складні та високоточні прогнозні моделі, здатні виявляти складні залежності в багатовимірних даних. У порівнянні з класичними статистичними методами, можна сказати, що вони не вимагають ручного визначення правил або особливостей, а саме автоматично навчаються на великих наборах даних.

Таблиця 2.2.

Приклади типових моделей ML.

“Дерева” рішень і випадкові “ліси”	Вони є ефективними для прогнозування параметрів у системах з великою кількістю вхідних ознак.

Продовження табл. 2.2.

Методи опорних векторів	Добре працюють із високорозмірними даними, через що можуть використовуватися для класифікації і регресії.
KNN	Простий та ефективний метод який використовується для локального прогнозування.

Щодо глибокого навчання (DL)... Глибоке навчання (deep learning) - це підмножина методів машинного навчання на основі штучних нейронних мереж із навчанням подань.[10]

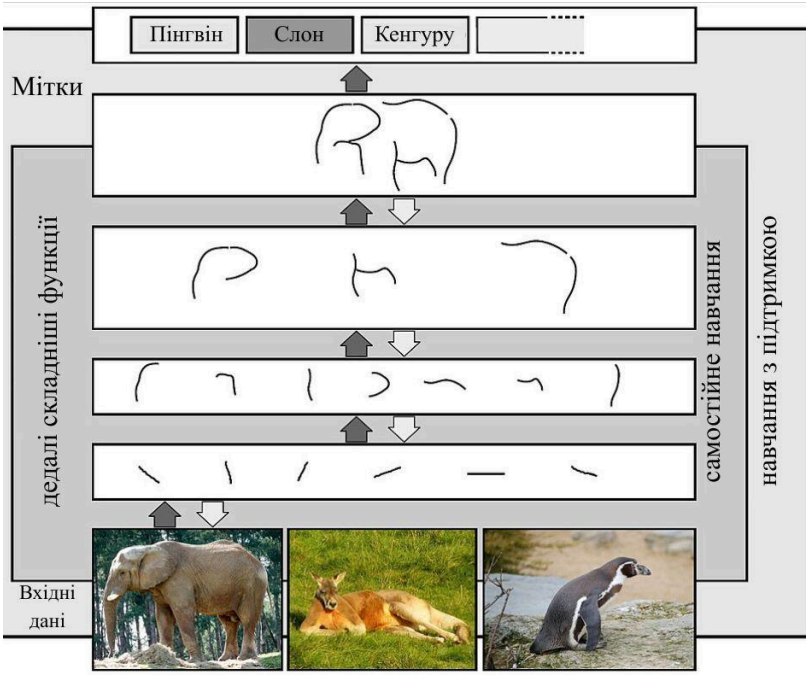


Рис. 2.4. Приклад подання зображень на кількох шарах абстракції в deep learning.[10]

DL-моделі є найпотужнішими у тих задачах де необхідне прогнозування складних процесів. Вони включають конволюційні нейронні мережі (для аналізу зображень та виявлення об’єктів), рекурентні мережі (для роботи з часовими рядами) та трансформери (сучасні архітектури з високою точністю у

прогнозуванні послідовностей). У автономних системах DL використовується для:

1. Розпізнавання об’єктів;
2. Сегментації сцен;
3. Прогнозування майбутніх кадрів;
4. Оцінки ризиків і прийняття рішень;
5. Прогнозування майбутнього стану навколишнього середовища.

Глибоке навчання дозволяє досягати високої точності навіть у нестабільних умовах.

Навчання з підкріпленням є в свою чергу методологією, у якій агент навчається шляхом взаємодії з середовищем і отримує винагороди за свої певні дії. На відміну від моделей, що лише прогнозують значення, RL дозволяє прогнозувати наслідки дій та оптимізувати поведінку системи. А його основні алгоритми включать в себе Q-learning та Deep Q-Network, Policy Gradient та інші моделі. Працює навчання з підкріпленням наступним чином: *Агент отримує стан середовища → обирає дію → отримує винагороду або штраф → оновлює свою політику дій.*

В свою чергу, у автономних системах RL застосовується для планування та оптимізації маршруту, адаптивного керування швидкістю, уникнення перешкод, прогнозування поведінки інших учасників руху, прийняття рішень в умовах невизначеності.

Модель про яку далі піде мова складається з множини станів та дій, ймовірностей переходу між станами, а також функції винагороди. Це MDP математичні моделі, що використовуються для опису прийняття рішень у середовищах, де результат дії має стохастичний характер, але визначається певними ймовірностями (див. Рисунок 2.5).[11] Ця модель у свою чергу дозволятиме автономній системі оцінювати ризики, прогнозувати результат дій, визначати найоптимальнішу стратегію поведінки, а також аналізувати сценарії з невизначеностями. На практиці дану математичну модель можна сміливо застосовувати для прогнозування можливих сценаріїв руху інших об’єктів,

вибору оптимального рішення при обмеженій інформації, підготовки моделі до непередбачених змін середовища та інше. MDP є фундаментом для багатьох сучасних систем автономного планування та часто інтегрується з RL-моделями, які я згадував раніше.

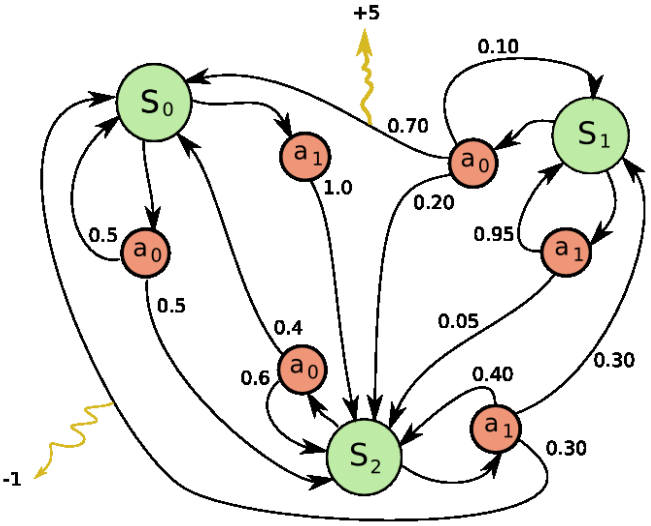


Рис. 2.5. Приклад перехідного автомата процесу прийняття рішень Маркова (MDP) з трьома станами та двома діями.[11]

Незважаючи на свою критичну роль, прогнозне моделювання в автономних системах стикається з кількома проблемами: якість та доступність даних, обчислювальна складність, динамічні та невизначені середовища, узагальнення та перенавчання, виконання в режимі реального часу. Якість та доступність даних - прогнозні моделі залежать від високоякісних і точних даних датчиків. У реальних сценаріях на датчики можуть впливати шум, перешкоди та фактори навколишнього середовища, такі як погодні умови чи освітлення, що в результаті призводить до негативних наслідків. Забезпечення узгодженості та точності даних є життєво важливим фактором для ефективного прогнозного моделювання. Також, є проблема із обчислювальною складністю, адже розширені прогнозні моделі, особливо моделі глибокого навчання, можуть бути ресурсоємними та вимагати значної обчислювальної потужності. У середовищах периферійних

обчислень з обмеженими ресурсами балансування складності моделі з продуктивністю в режимі реального часу є важливим питанням. Третьою проблемою я би відзначив динамічні та невизначені середовища, бо автономні системи саме і працюють у дуже динамічних та непередбачуваних середовищах. Точні прогнози часто спираються на стабільні та передбачувані закономірності, але фактори навколишнього середовища, такі як раптові зміни дорожнього руху, перешкоди чи погода, можуть створювати реальні проблеми, які вже важко враховувати. Розробка моделей, які можуть адаптуватися до цих невизначеностей, є ключовим завданням. Ще одна проблема - це, узагальнення та перенавчання. Прогнозні моделі повинні добре узагальнюватися на нові, невідомі раніше дані. Моделі, які надмірно адаптовані до конкретних навчальних даних, можуть добре працювати в контрольованих сценаріях, але не працювати в реальних умовах, де середовище більш не стабільне. Забезпечення стійкості та адаптивності є важливим для надійних прогнозів. Останнє, але одне із найголовніших питань - робота в режимі реального часу. Прогнозне моделювання в автономних системах має виконуватися в режимі реального часу з низькою затримкою, щоб впливати на рішення та дії. Потреба в швидких та ефективних прогнозах ставить перед системою вимогу балансувати точність та обчислювальну ефективність.

У свою чергу для вирішення проблем, що виникають під час прогнозного моделювання автономних систем, я пропоную кілька стратегій:

Доповнення та попередня обробка даних: однією з ключових умов якісного прогнозного моделювання є наявність “чистих”, узгоджених і репрезентативних даних. У реальних автономних системах дані датчиків часто містять шум, спотворення, тому застосовуються різноманітні методи доповнення та попередньої обробки.

Доповнення даних: дозволяє штучно збільшувати обсяг навчального набору шляхом створення синтетичних прикладів, що імітують реальні варіації в середовищі. Це може включати генерацію нових траєкторій руху, варіювання параметрів датчиків, моделювання шумів чи навіть використання генеративних моделей для формування реалістичних даних.

Попередня обробка: включає фільтрацію шумів, інтерполяцію відсутніх значень, нормалізацію та стандартизацію параметрів.

Крім того, *злиття даних*: об’єднання інформації з декількох датчиків, це дозволяє підвищити точність і надійність прогнозних моделей, оскільки компенсує недоліки окремих сенсорів і створює більш повне уявлення про стан середовища.

Периферійні обчислення відіграють ключову роль у забезпеченні швидкого прогнозного моделювання, необхідного для автономних систем, що працюють у режимі реального часу. Перенесення обчислювальних процесів ближче до джерел даних дає можливість зменшити затримку, уникнути перевантаження мережі та забезпечити оперативне прийняття рішень без постійної залежності від хмарної інфраструктури. Периферійні пристрої можуть виконувати локальні, оптимізовані версії моделей, що спеціально адаптовані для обмежених апаратних ресурсів, наприклад, моделі з пониженим числом параметрів, розрядністю або спрощеною архітектурою. Це також дозволяє обробляти сенсорні дані безпосередньо на транспортних засобах, дронах або роботизованих платформах. Такий підхід забезпечує стабільну роботу навіть у випадках нестабільного зв’язку із хмарою, підвищує надійність системи та дозволяє реалізувати автономні функції незалежно від зовнішніх мережевих умов.

У динамічних середовищах, де умови можуть швидко змінюватися, традиційні статичні моделі часто втрачають свою ефективність. Тому автономні системи усе частіше використовують адаптивні або інкрементальні методи навчання, що дозволяють моделі оновлювати свої параметри під час роботи. Такі моделі здатні навчатися на нових даних у режимі реального часу, реагуючи у свою чергу на зміну поведінки об’єктів, погодних умов чи характеристик середовища. Це є особливо важливим для задач з прогнозування траєкторій, ідентифікації ризиків та оцінки стану системи. Адаптивне навчання дозволяє уникати деградації точності в нових умовах, підтримувати модель актуальною без повного нового перенавчання, забезпечувати довготривалу стабільну роботу прогнозів. Завдяки

цьому автономні системи здатні краще реагувати на непередбачуваність та мінливість реального середовища.

Гібридний підхід передбачає комбінування декількох типів методів прогнозування для досягнення більш високої точності та стійкості. Наведу приклад… Моделі глибокого навчання можуть ефективно обробляти візуальні дані та розпізнавати складні шаблони, тоді як методи навчання з підкріпленням (RL) забезпечують оптимізацію дій у динамічних сценаріях.

Поєднання цих підходів дає змогу системі одночасно розпізнавати об’єкти, оцінювати ризики, прогнозувати події та вибирати найбільш вигідну поведінку. Хочу навести ще ряд інших прикладів гібридних рішень: використання CNN або трансформерів для обробки зображень + RL для планування траєкторій, поєднання регресійних моделей з моделями часових рядів для коротких і довгострокових прогнозів, інтеграція стохастичних моделей (MDP) з нейромережами для прийняття рішень в умовах невизначеності. Такий підхід дозволяє компенсувати недоліки окремих моделей і реалізувати більш складні, точні та стабільні системи прогнозування.

2.3. Методи стиснення.

Для розгортання прогнозних моделей у середовищах з обмеженими ресурсами, до прикладу таких як периферійні модулі або вбудовані системи, необхідно зменшити їхню обчислювальну складність. Як це зробити? За допомогою різних методів стиснення. Наприклад: обрізання моделі, яке передбачає видалення нейронів, шарів, які не дуже впливають на результат. Таким чином модель стає компактнішою та працює швидше, та майже не втрачає точності. Інший метод - квантування. Воно зменшує розрядність ваг та активацій, наприклад, переходячи від 32-бітних чисел до 8-бітних. Це суттєво зменшує обсяг пам’яті та час виконання операцій. Третій метод - дистиляція знань, яка дозволяє передати знання великої моделі до більш компактної моделі-учню. Модель-учень у свою чергу забезпечує ту саму якість прогнозування, але потребує менше ресурсів і швидше працює на периферійних пристроях.

Оптимізовані таким чином моделі можуть ефективно виконуватися у режимі реального часу, забезпечуючи достатню швидкість і точність для автономних систем без залежності від високопродуктивних серверів.

Стрімке поширення смартфонів і мережевих камер сприяло активному розвитку технологій відеоаналітики. Проте традиційна модель хмарних обчислень не підходить для таких застосувань, оскільки потребує передавання великих обсягів відеоданих у хмару, що в результаті створює значні затримки та несе ризики для конфіденційності. Однак передати всі відеодані в хмару не завжди можливо: це або порушує приватність, або є фінансово та технічно затратним. Навіть якщо дані все ж потраплять до хмари, їхнє завантаження та аналіз великого масиву відео можуть займати багато часу. У таких випадках ефективнішими є edge-обчислення. Така модель дозволяє використовувати наявні камери та локальні ресурси для швидкої обробки відео, забезпечуючи значно меншу затримку та швидший пошук необхідної інформації (людини, тощо) порівняно з традиційними хмарними обчисленнями.

Хмарні обчислення стали основною платформою для обробки великих обсягів даних як в академічних установах, так і в промисловості. Для обробки інформації, дані спочатку повинні передаватися в хмару, проте зазвичай, сторони не обмінюються даними через проблеми з конфіденційністю та високі витрати на їх передачу, що обмежує можливість співпраці між ними. В такому контексті периферія може стати частиною логічної концепції. Це невеликий фізичний центр обробки даних, який з’єднує кінцевих користувачів з хмарою. Вона дозволятиме створити спільну межу, яка з’єднує периферійні пристрої багатьох зацікавлених сторін, незалежно від їхнього місцезнаходження чи мережевої структури. Таке миттєве з’єднання дозволяє сторонам обмінюватися даними та ефективно координувати їх. Програми, що обробляють географічно розподілені дані, вимагають від компаній з кількох галузей співпраці та обміну даними. Периферійні обчислення можуть вирішити цю проблему шляхом об'єднання географічно розподілених даних та створення віртуально спільних подань даних. Віртуальні дані потім передаються кінцевим користувачам через попередньо

визначений термінальний інтерфейс сервісу, і програма використовуватиме цей публічний інтерфейс для надання складних послуг вже кінцевим користувачам. Спільні учасники периферійних обчислень надають ці послуги, а обчислення відбуваються лише в центрі обробки даних учасників, щоб забезпечити конфіденційність та цілісність даних.

2.4. Проблеми периферійних обчислень.

Відкриті проблеми периферійних обчислень можна підсумувати в цьому підрозділі. Зрозуміло, що багато програм працюють на вершині периферії, і кожна з них має свою власну структуру та метод обслуговування. Як і в будь-якій іншій комп'ютерній системі, схема іменування в периферійних обчисленнях дуже важлива для програмування, ідентифікації об'єктів та передачі даних. Однак, поки що не існує стандартизованого механізму іменування для моделі периферійних обчислень. Фахівцям з периферійних обчислень часто потрібно вивчати різні комунікаційні та мережеві протоколи, щоб вони могли взаємодіяти з різноманітними речами в системі.

Наприклад, найпоширеніша модель OSI - абстрактна мережева модель для комунікацій і розроблення мережевих протоколів, яка зазвичай представляється у вигляді сімох рівнів мережі.

Рівень OSI	Протоколи
прикладний	HTTP, gopher, Telnet, DNS, DHCP, SMTP, SNMP, CMIP, FTP, TFTP, SSH, IRC, AIM, NFS, NNTP, NTP, SNTP, XMPP, FTAM, APPC, X.400, X.500, AFP, LDAP, SIP, IETF, RTP, RTCP, ITMS, Modbus TCP, BACnet IP, IMAP, POP3, SMB, MFTP, BitTorrent, e2k, PROFIBUS, WebSocket Це всього лише кілька найрозповсюдженіших протоколів прикладного рівня, яких існує неймовірно велика кількість. Всі їх неможливо описати в рамках даної статті.
представлення	ASN.1, XML, TDI, XDR, NCP, AFP, ASCII, Unicode
сеансовий	ASP, ADSP, DLC, Named Pipes, NBT, NetBIOS, NWLink, Printer Access Protocol, Zone Information Protocol, SSL, TLS, SOCKS, PPTP
транспортний	TCP, UDP, NetBEUI, AEP, ATP, IL, NBP, RTMP, SMB, SPX, SCTP, DCCP, STP, TFTP, RTP
мережевий	IPv4, IPv6, ICMP, IGMP, IPX, NWLink, NetBEUI, DDP, IPSec, SKIP
канальний (Ланки даних)	ARCnet, ATM, DTM, SLIP, SMDS, Ethernet, ARP, FDDI, Frame Relay, LocalTalk, Token Ring, PPP, PPPoE, StarLan, WiFi, PPTP , L2F, L2TP, PROFIBUS
фізичний	RS-232, RS-422, RS-423, RS-449, RS-485, ITU-T, RJ-11, T-carrier (T1, E1), модифікації стандарту Ethernet: 10BASE-T, 10BASE2, 10BASE5, 100BASE-TX, 100BASE-FX, 100BASE-T, 1000BASE-T, 1000BASE-TX, 1000BASE-SX

Рис. 2.6. Модель OSI з прикладом найрозповсюджений протоколів різних рівнів мережі.[12]

Кожен рівень обслуговує свою частину процесу взаємодії, завдяки чому спільна робота мережевого обладнання й програмного забезпечення стає набагато прозорішою й зрозумілішою.[12]

Також важливо, що схема іменування для периферійних обчислень повинна враховувати портативну природу речей, високодинамічний географічний розподіл мережі, конфіденційність та безпеку, а також масштабованість величезної кількості ненадійних речей.

Підтримка різних типів пристроїв та задоволення різноманітних вимог до послуг є величезним викликом у середовищі периферійних обчислень. Периферійні обчислення інтегрують комбінацію різних платформ, типологій мереж та серверів, тому гетерогенна природа цієї системи ускладнює програмування та управління даними і ресурсами для багатьох програм, які працюють на різних платформах у різних місцях. З точки зору програмування, всі додатки розгортаються та використовуються на хмарних серверах із застосуванням хмарних обчисленнях. Постачальники хмарних послуг, такі як Google та Amazon, розміщують ці програми у відповідних місцях та на відповідному обладнанні, щоб забезпечити їх безперебійну роботу. Більшість користувачів не знають, як функціонують ці програми або де знаходяться дані та ресурси. Хмарні обчислення пропонують керований, централізований хмарний сервіс як одну зі своїх переваг. Крім того, розробникам потрібно використовувати лише одну мову програмування для розробки програм для певної платформи, оскільки хмарний додаток розгортається лише на одному конкретному постачальникові хмарних послуг. Периферійні обчислення, навпаки, дуже відрізняються від хмарних обчислень, хоча розподілена топологія має багато переваг, периферійні вузли зазвичай є різними платформами. Таким чином, розробники может стикатися з серйозними труднощами під час розробки програм, яка може бути розгорнута та запущена на платформі периферійних обчислень. Певні схеми були розроблені для вирішення проблем програмування в периферійних обчисленнях, але жодна з цих схем не призначена для конкретних цілей Інтернету речей. Першим кроком в Інтернеті речей є пошук периферійних

вузлів. До цього пристрої Інтернету речей не можуть ідентифікувати тип платформ, розгорнутих поблизу. Більше того, багато програм на стороні сервера необхідно розгорнути на периферійних вузлах, і це створює ще одну проблему для постачальників периферійних вузлів, яким потрібно розгортати та керувати цими програмами. Що стосується управління даними, різні сервери зберігання даних працюють під управлінням різних операційних систем, це у свою чегру створює ще одну проблему для іменування файлів, розподілу ресурсів, управління надійністю файлів тощо. Оскільки багато пристроїв Інтернету речей одночасно генерують та завантажують дані, іменування ресурсів даних стає складним завданням.

Трохи повертаючись до хмарних обчислень, можна навести такий приклад: користувачі в моделі хмарних обчислень програмують та розгортають свій код у хмарі. Потім постачальник хмарних послуг вирішує, де виконуються завдання хмарних обчислень. Користувачі зазвичай не знають, як обслуговується додаток. Однак хмарні обчислення відомі своєю прозорою інфраструктурою. Програма зазвичай пишеться певною мовою програмування та компілюється в хмарі для конкретної цільової платформи, оскільки програма працює лише в хмарі. Натомість, обчислення в периферійних обчисленнях виконуються з хмари, а периферійні вузли, ймовірно, є гетерогенними платформами. Таким чином, ці вузли мають різні середовища виконання, і програмісту важко написати додаток, який можна розгорнути в моделі периферійних обчислень.

Гіпотетично кажучи, периферійні обчислення можуть працювати на кількох вузлах, розташованих між периферійним пристроєм та хмарою, включаючи точки доступу, базові станції, шлюзи, точки агрегації трафіку, маршрутизатори, комутатори тощо. Наприклад, базові станції містять цифрові сигнальні процесори, які адаптовані для обробки їхнього робочого навантаження. Однак на практиці базові станції можуть бути не в змозі обробляти аналітичні робочі навантаження, оскільки ці процесори не призначені для виконання обчислювальних завдань загального призначення. Крім того, чи можуть ці вузли виконувати додаткові обчислювальні завдання? Нажаль, це залишається

невідомим. Можливо, було б корисно використовувати такі базові станції в період між піковими навантаженнями, щоб використовувати обчислювальні можливості кількох доступних обчислювальних ядер. Багато постачальників бізнеспродуктів зробили перший крок до програмних рішень, що використовують периферійні обчислення. Наприклад, Nokia розробила програмне рішення MEC для мобільних периферійних обчислень, яке призначене для створення сайтів базових станцій для моделі периферійних обчислень (див. Рисунок 2.7).[13]

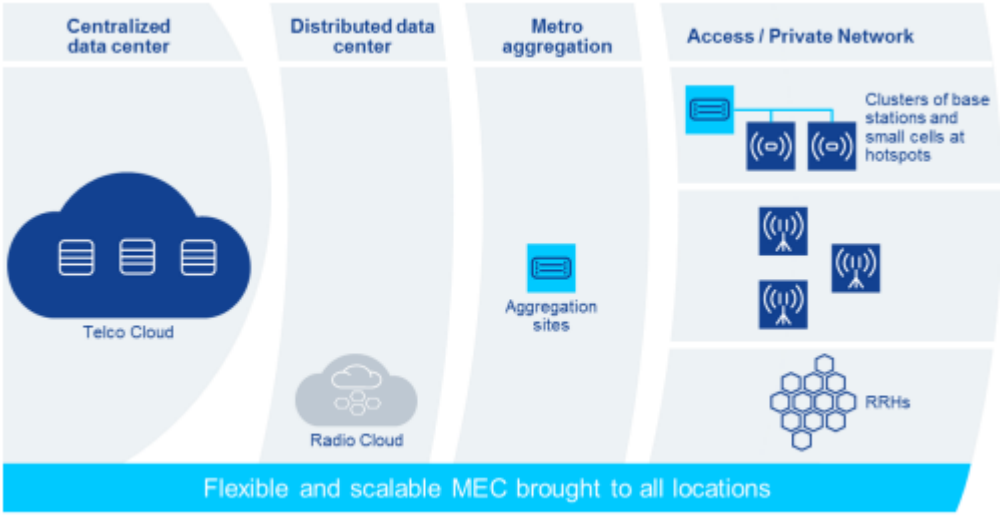


Рис. 2.7. MEC може пропонувати широкий спектр варіантів розгортання.[13]

Аналогічно, Cisco із їх IOx16, яке забезпечує середовище реалізації для своїх вбудованих сервісних маршрутизаторів. Оскільки ці рішення є апаратно-специфічними, вони можуть не працювати в гетерогенному середовищі, адже було б складно розробити портативні програмні рішення, які можуть функціонувати в різних середовищах. Наразі тривають дослідження щодо модернізації ресурсів граничних вузлів, які б підтримували обчислення загального призначення. Центральні процесори загального призначення можуть замінити спеціалізовані цифрові сигнальні процеси як альтернативне рішення, але це потребуватиме значних інвестицій.

Інтеграція інтернету речей та периферійних обчислень вимагає глибокого, всебічного розуміння та посилення управління ресурсами, адже мережевий

трафік та затримка мають значний вплив на пристрої Інтернету речей і часто призводять до дефіциту обчислень та нестачі ресурсів, оскільки це вимагає використання більшої потужності для повторної передачі даних у складних умовах. Периферійні обчислення можуть зменшити затримку в пристроях, а децентралізовані ресурси відіграватимуть важливу роль у спільному використанні цих активів. Цими ресурсами можна керувати різними способами, за умови, що це передбачає недорогі обчислення. Однак, нерівність між постачальниками послуг, пристроями та програмами спричиняє багато складнощів, і таку взаємодію слід також враховувати, зокрема, мотивуючі фактори управління ресурсами Edge/IoT синхронізовані з факторами інтелектуальних систем. Безпосереднє обслуговування системи, яка має кількох постачальників ресурсів та величезну різноманітність програм та потреб користувачів, може бути розподілене або спільне, та оцінене за допомогою якоїсь метрики або будь-якої іншої стратегії.

Для забезпечення надійності системи, управління послугами на периферії мережі вимагає підтримки наступних чотирьох основних функцій: диференціація, розширюваність, ізоляція та надійність. Диференціація - швидкий розвиток впровадження Інтернету речей, що призведе до розгортання багатьох послуг на периферії мережі, таких як наприклад розумні будинки. Іншим прикладом є послуги, пов'язані зі здоров'ям, такі як виявлення серцевої недостатності, які повинні мати вищий пріоритет над іншими послугами, такими як розважальні. Розширюваність - вона створює серйозну проблему на периферії мережі, адже об’єкти в Інтернеті речей є дуже динамічними. Універсальний рівень управління послугами може вирішити деякі поточні проблеми, однак не всі, і це треба розуміти. Коли користувач купує нову річ, йому потрібно легко додати поточну послугу., більше того, коли річ зношується та замінюється, попередня послуга повинна легко адаптуватися до нового вузла. Четверта функція, це ізоляція, та ще одна проблема на периферії мережі. Коли програма виходить з ладу або аварійно завершує роботу в мобільній ОС, вся система зазвичай аварійно завершує роботу або перезавантажується. У розподіленій системі для керування спільним

ресурсом можна використовувати різні механізми синхронізації, такі як блокування або Token Ring. Однак ця проблема є складнішою в інтелектуальній EdgeOS. Багато програм можуть використовувати один і той самий ресурс даних, наприклад керування освітленням. Якщо одна програма виходить з ладу або не відповідає, користувач все ще може керувати своїм освітленням, оскільки це не призведе до збою всієї EdgeOS. Інший приклад, коли користувач видаляє програму, яка керує освітленням із системи, і освітлення все ще працює, не втрачаючи зв'язку з EdgeOS. Впровадження платформи розгортання може вирішити цю проблему. Якщо ОС виявить проблему до розгортання програми, вона попередить користувача, щоб уникнути будь-яких проблем з доступом. Ще одним аспектом цієї проблеми є ізоляція особистих даних користувача від сторонніх програм. Наприклад, ваша програма відстеження активності не повинна мати доступу до даних про використання електроенергії. Щоб вирішити цю проблему, рівень керування послугами EdgeOS повинен бути оснащений надійним механізмом контролю доступу. Надійність також є ще однією серйозною проблемою на периферії мережі, і вона розглядається з різних аспектів обслуговування, системи та даних. З точки зору обслуговування іноді важко точно визначити причину збою служби. Наприклад, система кондиціонування повітря перестає працювати через обрив кабелю, зламаний компресор або навіть розряджену батарею регулятора температури. Вузол датчика міг втратити зв'язок із системою внаслідок відключення батареї, поганого з'єднання, відмови компонента тощо. Таким чином, коли деякі вузли втрачають зв'язок на периферії мережі, недостатньо просто підтримувати поточну службу, було б корисніше забезпечити дії після виходу вузла з ладу. Натомість з системної точки зору, підтримка мережевої топології всієї системи є справді важливою для EdgeOS. Усі компоненти системи можуть надсилати дані про стан та діагностику безпосередньо на периферію. Ця функція дозволяє легко розгортати систему, такі як виявлення несправностей, заміна об’єктів та оцінення якості даних. З точки зору даних, зчитування даних та зв'язок в основному відповідають за проблеми надійності. Речі на периферії мережі виходять з ладу з різних причин, і вони

повідомляють неточні дані за невизначених умов, таких як низький рівень заряду батареї, тощо. Було багато пропозицій щодо нових протоколів зв'язку для збору даних. Такі протоколи можуть підтримувати велику кількість сенсорних вузлів та дуже динамічний стан мережі. Однак з'єднання не таке надійне, як у Bluetooth наприклад або Wi-Fi. Для системи залишається проблемою надання надійного сервісу, використовуючи багато джерел довідкових даних та записів історії даних, навіть коли дані зондування та зв'язок не такі надійні.

У літературі ретельно досліджуються ресурси та послуги, що надаються в розподіленому обчислювальному середовищі. Різні методи використовуються в інструментах моніторингу та брокерських послугах для полегшення такого дослідження як у тісно інтегрованих середовищах, так і в окремих. Для підвищення продуктивності використання такої методики, як бенчмаркінг, застосовується підтримка відображення завдань, налаштовуючи прийняття рішень щодо найбільш підходящих ресурсів. Однак використання периферії мережі вимагає дослідження механізмів, які допомагають визначити відповідні вузли для децентралізованої хмарної конфігурації. Через велику кількість пристроїв на цьому рівні такі механізми не можуть бути ручними. Крім того, механізми повинні добре працювати як з гетерогенними пристроями кількох поколінь, так і з сучасними. Методи бенчмаркінгу повинні дуже швидко визначати доступність та ефективність ресурсів. Ці механізми повинні сприяти плавній інтеграції та усуненню вузлів на різних ієрархічних рівнях обчислювального робочого процесу без збільшення затримки та погіршення взаємодії користувача. Користувачі цінують здатність системи вирішувати проблеми та забезпечувати автоматичне відновлення на вузлі надійним та проактивним чином. У цьому контексті сучасні хмарні рішення практично не виявляють периферійні вузли.

Еволюція обчислювальних середовищ привела до появи багатьох методів, що сприяють розподілу завдань між кількома географічними локаціями. Робочі навантаження розподіляються та виконуються в різних місцях, а розподіл завдань зазвичай здійснюється за допомогою мови програмування або інструменту

управління. Однак, якщо ми використовуємо крайові вузли для розвантаження обчислень, буде важко ефективно розподілити обчислення. Це має робитися автоматично, незалежно від обсягу пам'яті та обчислювальних потужностей крайових вузлів, шляхом розробки планувальників, які розгортають завдання на крайових вузлах.

Прикордонні вузли забезпечують ефективне забезпечення користувачам якості обслуговування та якості досвіду. Одним із принципів периферійних обчислень є уникнення перевантаження вузлів інтенсивними обчислювальними процесами. У цьому випадку важко забезпечити високу продуктивність та надійність вузлів під час виконання запланованих робочих навантажень, а також робочих завдань, отриманих від центру обробки даних або периферійних пристроїв. Користувач такого пристрою або центру обробки даних все ще очікує певного обслуговування, навіть коли периферійний вузол повністю завантажений. Наприклад, перевантаження базової станції може вплинути на її роботу та у подальшому відновити обслуговування, що надається підключеним периферійним пристроям. Це вимагає реалізації пікових годин використання периферійних вузлів, щоб гнучко розподіляти та планувати завдання. Структура управління може бути дуже корисною в цьому сенсі, але вона створює проблеми з точки зору моніторингу, планування та перепланування на рівні інфраструктури та платформи.

Розумні системи, по суті своїй, пов'язують технології мережевого зв'язку з датчиками та виконавчими механізмами для досягнення системної обізнаності та виконавчими механізмами, які призначені для надсилання сигналів системі для вжиття заходів. Розумні системи розглядаються як розширення технологій Інтернету речей. Інтеграція сенсорних пристроїв пропонує безліч можливостей для збору даних, управління системами, а також розподілу та оптимізації ресурсів. Розумні системи в основному використовуються в таких сферах, як мережі, розумні міста, розумний транспорт, розумна охорона здоров'я та інші. Зростаюче використання розумних систем дозволяє периферійним обчисленням знизити затримку до найнижчого можливого рівня та збільшити ємність сховища

в пристроях, які мають обмежені обчислювальні можливості. Так само аналіз периферійних даних також може забезпечити високий ступінь стійкості в скомпрометованих системах.

Периферійні обчислення наразі змінили віддалені обчислення та зберігання даних, оскільки вони усунули бар'єри для пропонування швидких, низькозатримувальних та високо обчислювальних застосувань. Так само, майбутні технології стільникових мереж 5G, включаючи надщільні мережі (UDN), масивні MIMO (множинні входи та виходи) та міліметрові хвилі, вдосконалюються, рухаючись до зменшення затримки, збільшення пропускної здатності та забезпечення підтримки взаємопов'язаних груп у щільних мережах. Це неминуче призведе до розвитку комунікаційних технологій. Як всім відомо 5G вважається наступним поколінням комунікаційних технологій, а його метою є забезпечення користувачів повсюдним мережевим підключенням та доступом до даних. У цьому відношенні 5G, Інтернет речей та периферійні обчислення можуть бути інтегровані для досягнення гнучкого та ефективного зв'язку. Технологія 5G також може допомогти зробити багато застосувань Інтернету речей більш ефективними.

Конфіденційність та безпека - це важливі питання, які потребують ретельного розгляду. Парадигми хмарних та периферійних обчислень мають численні проблеми в кібербезпеці. Периферійні обчислення, зокрема, передбачають об'єднання різних технологій (наприклад однорангових систем, бездротових мереж та віртуалізації), і це вимагає комплексної інтегрованої моделі для захисту та управління кожною технологічною платформою та системою в цілому. Однак розвиток периферійних обчислень може призвести до деяких непередбачуваних проблем. Певні ситуації, які залишаються невивченими, такі як взаємодія різних периферійних вузлів та локальна та глобальна міграція послуг, створюють потенціал для зловмисної поведінки. Крім того, вбудовані функції периферійних обчислень додатково визначають життєздатність заходів конфіденційності та безпеки. Хоча розподілена структура пропонує багато переваг для Інтернету речей, у таких структурах важко підтримувати

конфіденційність та безпеку. З точки зору конфіденційності, периферійні обчислення можуть забезпечити ефективну платформу для майбутнього Інтернету речей. Обробка даних на периферії дозволяє моделі периферійних обчислень використовувати конфіденційні дані, що належать кінцевим користувачам. Дані зчитування з систем Інтернету речей зберігаються на периферійних вузлах, які є більш вразливими, ніж хмарні сервери. Таким чином, периферійні обчислення повинні використовувати ефективні механізми захисту даних конфіденційності. Що стосується питань безпеки, то автентифікація шлюзів на різних рівнях є основною проблемою в периферійних обчисленнях. Наприклад розумні лічильники, що використовуються в житлових будинках, мають власну IP-адресу. Перенесення програми з пристрою кінцевого користувача на периферійний сервер вимагає передачі даних з мобільного пристрою на периферійний сервер. Це вже в свою чергу дозволяє зловмисникам отримувати доступ до даних. Пристрої кінцевих користувачів живляться від батарей, тому вимагають застосування легких механізмів безпеки. Рішення щодо безпеки та конфіденційності також повинні бути гнучкими, щоб сприяти досягненню мети периферійних обчислень, а саме скороченню часу виконання. Однак важко знайти надійне легке рішення для складної проблеми безпеки в різноманітному середовищі. Можна зробити висновок, що безпека є основною проблемою, що перешкоджають розвитку периферійних обчислень.

Апаратні ресурси в центрах обробки даних, суперкомп'ютерних центрах та приватних організаціях можна перетворити на обчислювальні послуги за допомогою віртуалізації. Коли постачальники послуг виявляють пов'язані з ними ризики для користувачів, вони надають обчислювальні послуги на основі оплати за використання. Таким чином, ринок обчислювальних послуг став дуже конкурентним, оскільки пропонує численні варіанти, що відповідають стандартам угод про рівень обслуговування. Однак використання альтернативних пристроїв, таких як комутатори, маршрутизатори або базові станції, як загальнодоступних крайових вузлів створить деякі проблеми. По-перше, необхідно визначити ризик, пов'язаний з пристроями, що належать державним та

приватним організаціям, а також суб'єктами, які їх використовують. По-друге, цільове призначення пристрою не може бути порушене, коли він служить вузлом крайових обчислень. По-третє, багатокористувацька оренда на крайових вузлах вимагає використання технології, орієнтованої на безпеку. Крім того, для розробки відповідних моделей ціноутворення для доступних граничних вузлів необхідно врахувати робоче навантаження, обчислення, розташування даних, передачу даних, а також вартість обслуговування та рахунки за електроенергію.

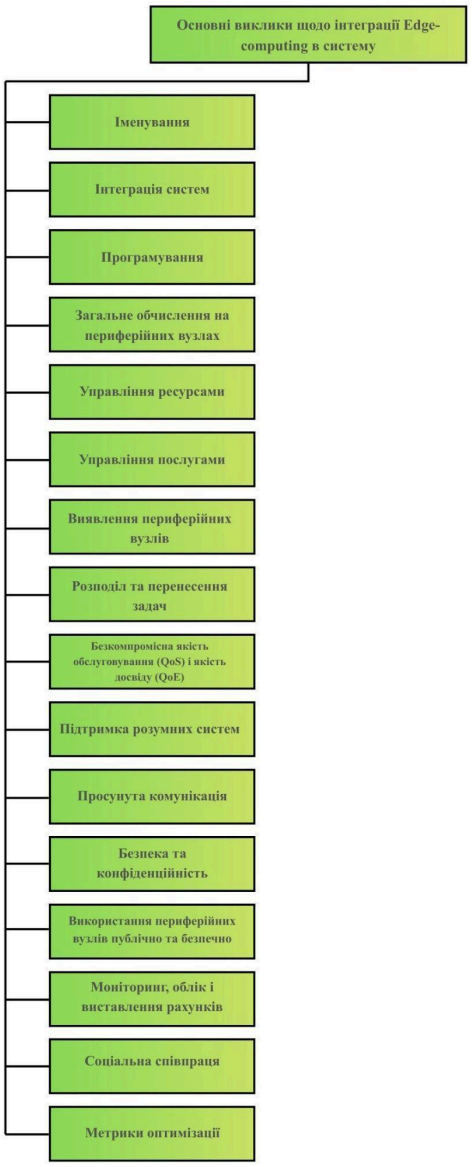


Рис. 2.8. Основні виклики щодо інтеграції Edge-computing в систему.

Важливо керувати ресурсами периферійних обчислень, їх обліком та виставленням рахунків, щоб забезпечити якість обслуговування та встановити

відповідну плату за пропоновані послуги периферійних обчислень, і це вимагає від постачальників послуг застосування стійкої бізнес-моделі. Однак, розробити таку модель складно, враховуючи мобільний характер клієнта та обмежений діапазон послуг. Користувач зазвичай використовує послугу периферійного вузла протягом обмеженого часу, наприклад як студенти, які користуються послугою в університетській їдальні під час обідньої перерви. Таке короткочасне використання ускладнює підтримку бізнес-моделі. Крім того, коли користувач переміщується, а виконання переноситься з однієї периферійної платформи на іншу, стає важко розподілити плату між залученими постачальниками послуг та підтримувати бізнес-модель для моніторингу, обліку та виставлення рахунків. Це вимагає бізнес-моделі, яка включає кілька рівнів деталізованої тарифікації. Різним постачальникам послуг важко співпрацювати та досягати спільної мети. Стандартизація та конкуренція є основними перешкодами, що заважають такому соціальному розвитку.

Різні компанії пропонують різні периферійні послуги. Однак, враховуючи конкурентний ринок, важко досягти соціальної співпраці між ними. Різнорідний характер пристроїв, що надаються компаніями, також створює перешкоду для досягнення такої співпраці. Тому подолання проблеми співпраці може підвищити ефективність аналізу даних.

Периферійні обчислення включають кілька рівнів з різною обчислювальною потужністю. Розподіл робочого навантаження може бути проблематичним, оскільки нам потрібно вибрати рівень, який оброблятиме робоче навантаження, або кількість завдань, які потрібно призначити кожній частині. Для виконання робочого навантаження можна застосувати багато стратегій розподілу, наприклад робоче навантаження може бути рівномірно розподілене на кожному рівні або виконане на кожному рівні якомога більше. У крайніх випадках робоче навантаження буде повністю оброблено в хмарі. Вибираючи оптимальну стратегію розподілу, необхідно враховувати деякі показники, такі як затримка, пропускна здатність, енергоспоживання та вартість. Периферійні обчислення розширюють хмарні обчислення, наближаючи послуги до кінцевого користувача

на периферії мережі. Існує потреба вирішити проблеми впровадження моделі периферійних обчислень, оскільки ця модель стане основним компонентом майбутнього обчислювального ландшафту. Цей розділ служить орієнтиром для відповідних майбутніх напрямків.

3. ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ З EDGE-ОБЧИСЛЕННЯМИ ДЛЯ АВТОНОМНОГО ПРИЙНЯТТЯ РІШЕНЬ У РЕЖИМІ РЕАЛЬНОГО ЧАСУ.

3.1. Модель системи

Для підтвердження коректності архітектури та функціональної структури розробленої системи з edge-обчисленнями, та для наглядного прикладу я створив комплекс наступних UML та DFD діаграм, а також аналіз та обгрунтовано відповідність кожної діаграми реальним процесам, що відбуваються в системі. Усі діаграми базуються на етапах проектування, описаних у попередніх розділах, а також на результатах практичної реалізації.

1. Діаграма компонентів.

Діаграма компонентів зображена на рисунку 3.1.

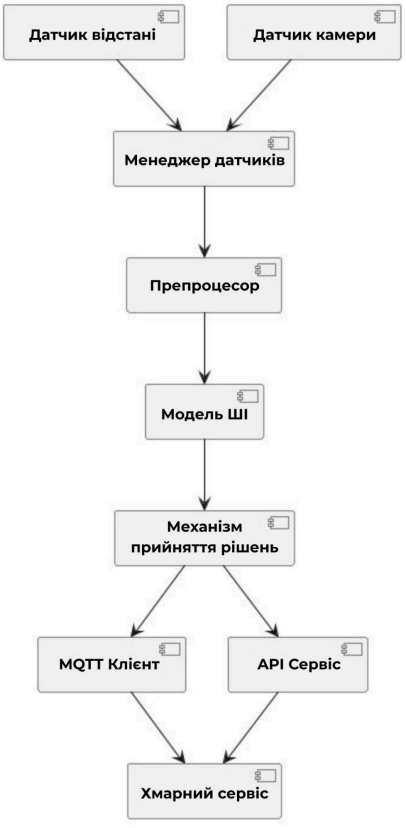


Рис. 3.1. Діаграма компонентів.

Ця діаграма відображає високорівневу архітектуру системи. Оскільки розроблена система належить до класу edge-computing, її ключовою особливістю

є розміщення основних функціональних модулів безпосередньо на периферійному пристрої або edge-вузлі .

Компоненти:

A. Датчик відстані та камера збирають усю можливу інформацію з середовища.

B. Менеджер датчиків приймає на себе усю отриману з датчиків інформацію.

C. Компонент “Препроцесу” - займається попередньою обробкою даних.

D. Так званий “Модуль ШІ” - використовується для локального машинного навчання, на основі попередньо оброблених даних.

E. Далі “в гру” вступає “Механізм прийняття рішень”.

F. Після чого відбуваються процес обміну даними з хмарою через MQTT Клієнт та моніторинг потенційних можливостей для оновлення конфігурації через API Сервіс.

Обраний мною склад компонентів відповідає всім вимогам edge-системи. Автономність, можливість роботи без підключення до хмари та виконання обчислень локально - це все вона включає в себе. Діаграма також містить сенсори як окремі компоненти джерела даних та хмарні сервіси як окремий кластер, що забезпечує зберігання моделей та централізований моніторинг. Такий підхід відповідає сучасним IoT-архітектурним практикам де сенсори, edge-вузол та хмара є логічно розділеними частинами інфраструктури.

2. Діаграма класів.

Діаграма класів описує структурну модель програмної частини системи. Тобто її головна мета зобразити та та пояснити яку структуру має система з edge-обчисленнями у вигляді коду.

В цій діаграми можна чітко побачити логічні залежності між класами. Зверніть увагу, дані проходять шлях від SensorManager → Preprocessor → AIModel → DecisionEngine → до MQTTClient наступним чином (див. Рисунок 3.2).

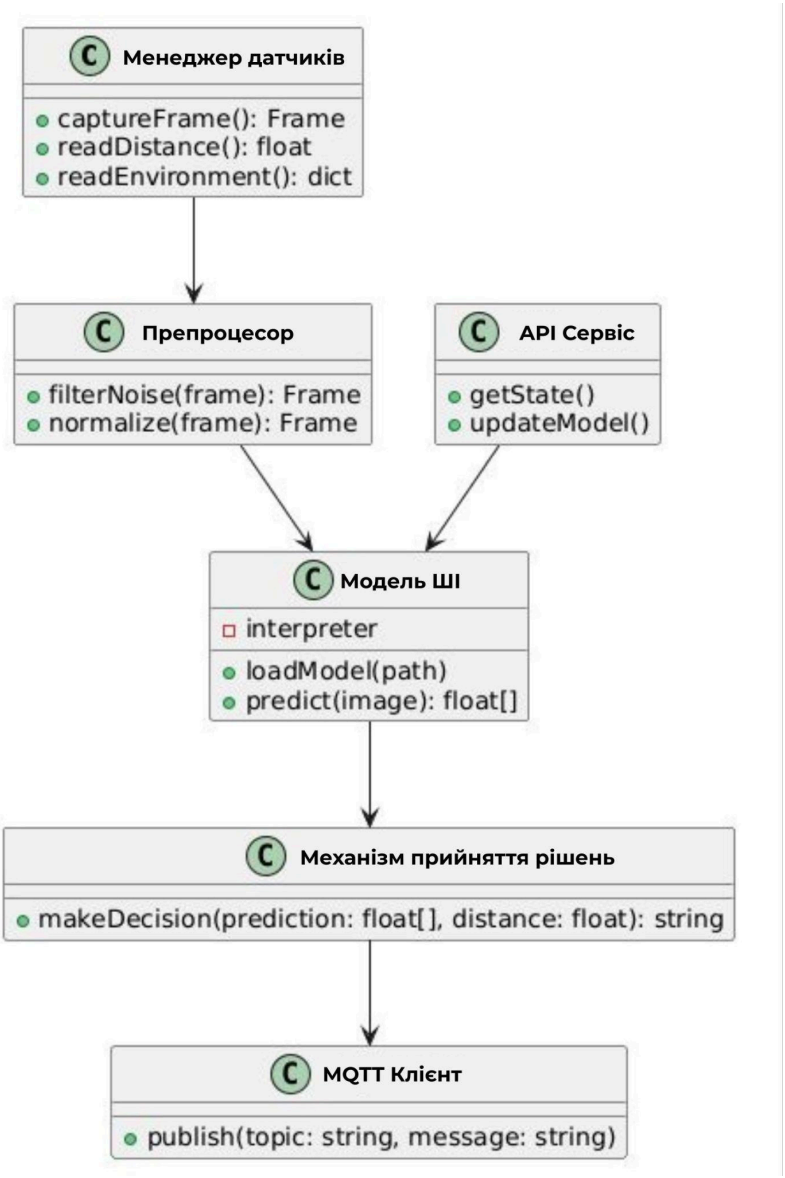


Рис. 3.2. Діаграма класів.

A. Менеджер датчиків відповідає за читання даних із камери, ультразвукового датчика та сенсорів навколишнього середовища.

B. Компонент “Препроцесу” здійснює фільтрацію, нормалізацію та інші операції класичної передобробки.

C. “Модель ШІ” інкапсулює роботу інтерпретатора TensorFlow Lite або ONNX Runtime та забезпечує функцію прогнозування.

D. Механізм прийняття рішень реалізує логіку автономного прийняття рішень.

E. MQTT Клієнт відповідає за передачу повідомлень до MQTT Broker

F. A APIService реалізує простий REST API для отримання стану та оновлення ML-моделі.

3. Діаграма послідовності.

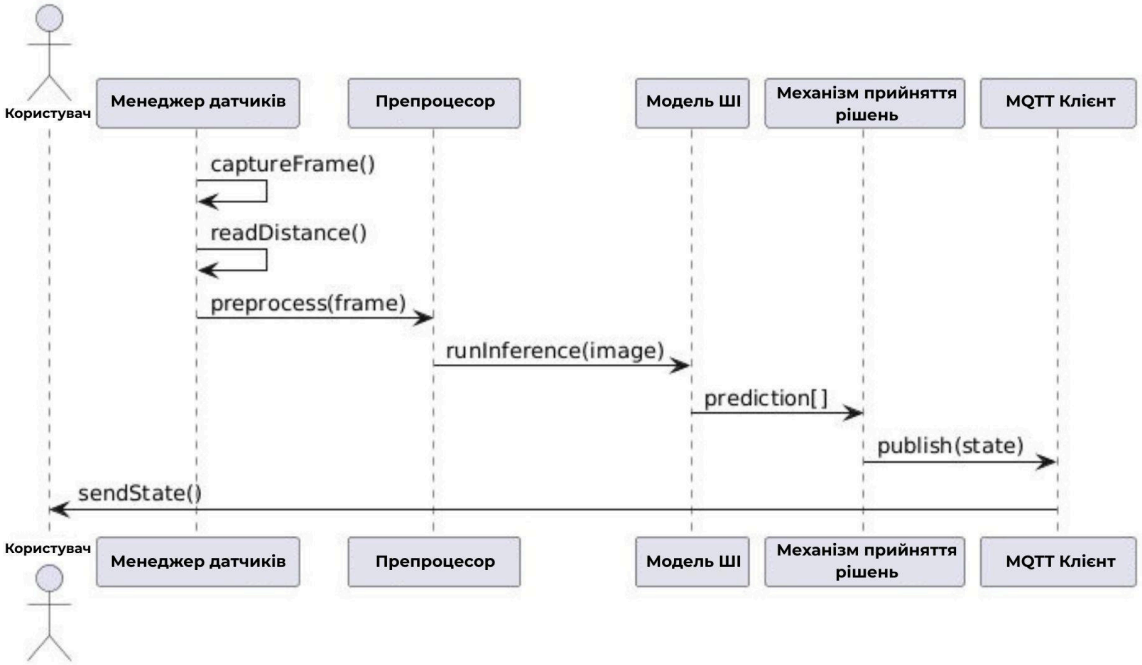


Рис. 3.3. Діаграма послідовності.

В цій діаграмі демонструється часова послідовність викликів, які здійснюються під час одного циклу роботи системи в режимі реального часу. В діаграмі продемонстровано фактичний порядок виклику методів, що був реалізований під час написання програмного забезпечення. Щодо компонентів… Збір даних безпосередньо відбувається за допомогою менеджеру сенсорів. Відповідальність за передачу кадру лежить на компоненті препроцесору. В свою чергу виклик ML-моделей відбувається через модель штучного інтелекту, а формування рішення відбувається вже безпосередньо в механізмі прийняття рішень. Останнім кроком є публікація стану через клієнт MQTT. Ця діаграма відповідає фізичному процесу роботи системи, де цикл обробки становив близько 45–60 мс.

4. Діаграма розгортання.

Ця діаграма (рисунок 3.4) демонструє фізичне розташування програмних компонентів у реальному середовищі. Вона містить три основні типи вузлів. Такі як:

1. Edge-пристрій, на якому розгорнута основна логіка системи.

2. Фізичні сенсори, які підключені через GPIO/CSI-інтерфейси.

3. І хмарний сервер, що відповідає за тривале зберігання даних, розповсюдження моделей та моніторинг.

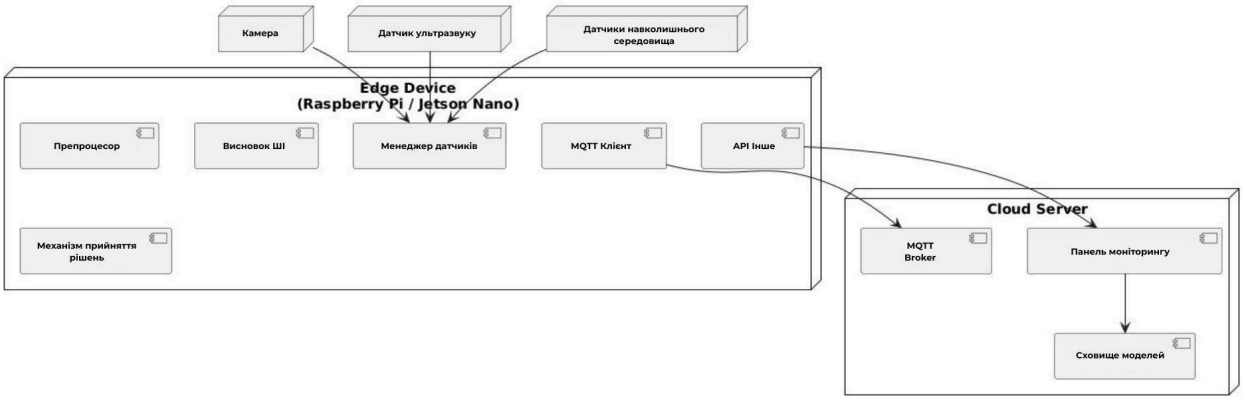


Рис. 3.4. Діаграма розгортання.

Таке розділення відповідає стандартним підходам побудови систем з edge-обчисленнями, де хмара використовується лише для допоміжних сервісів, а основне навантаження виконується на периферії, тобто через edge-обчислення. Діаграма розгортання підтверджує, що система може працювати автономно та не залежить від доступності інтернету для виконання основних функцій.

5. Діаграма потоків даних.

Діаграма потоків даних (рисунок 3.5), або DFD-діаграма подає логічні потоки даних між окремими процесами.

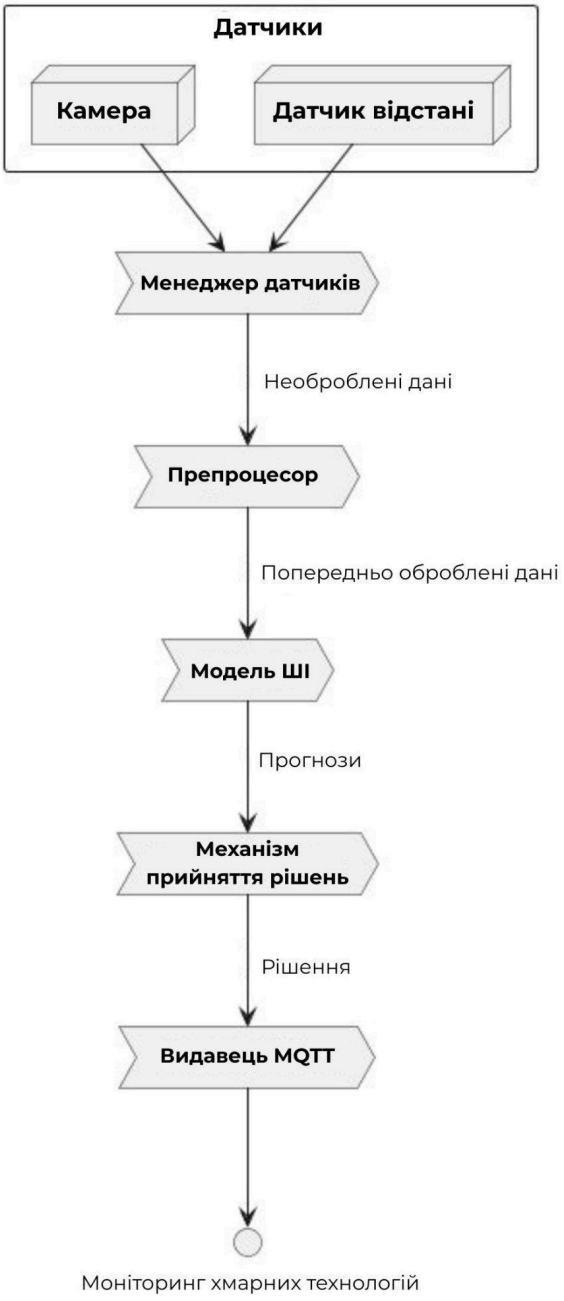


Рис. 3.5. Діаграма потоків даних.

Вона наглядно демонструє шлях даних, від моменту їх отримання, до публікації самого результату. Усі необроблені дані від датчиків (в даному випадку камера та датчик відстані) потрапляють до менеджеру датчиків, який у свою чергу відправляє ці дані до препроцесору (або процесору попередньої обробки даних), який формує пакет з оброблених даних і відправляє їх до моделі ШІ на основі якої вже робляться прогнози в системі. Після чого вся ця інформація поступає на пряму до механізму прийняття рішень, механізму який направляє

своє подальше рішення у вигляді оброблених даних до MQTT Клієнту. Далі дані переходять до хмарного сервісу моніторингу. Мета діаграми показати як саме дані циркулюють у цій системі.

6. Діаграма станів.

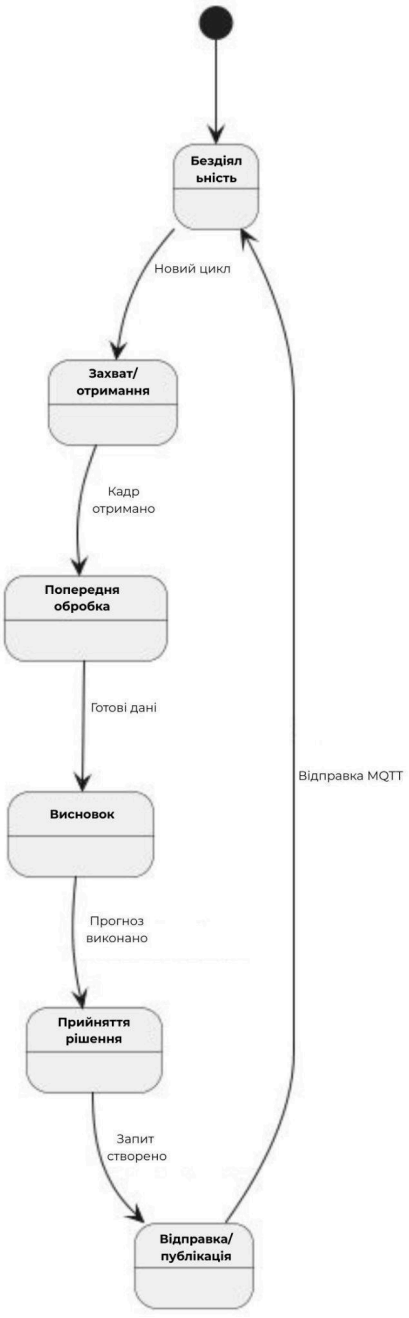


Рис. 3.6. Діаграма станів.

Діаграма станів описує життєвий цикл роботи edge-вузла, від пасивного режиму (Idle) до активних фаз обробки даних та повернення у вихідний стан. Продемонстрований в ній набір станів повністю покриває всі фази роботи

системи та показує, що обробка відбувається циклічно й безперервно. Компоненти діаграми:

A. Бездіяльність/пасивний режим - очікування нового циклу.

B. Захват/отримання - зчитування даних.

C. Препроцес - попередня обробка зображень.

D. Висновок - відбувається ML-прогнозування.

E. Прийняття рішень - формування подальшого рішення в системі.

F. Відправка/публікація/розміщення - передача даних у хмару.

7. Діаграма діяльності.

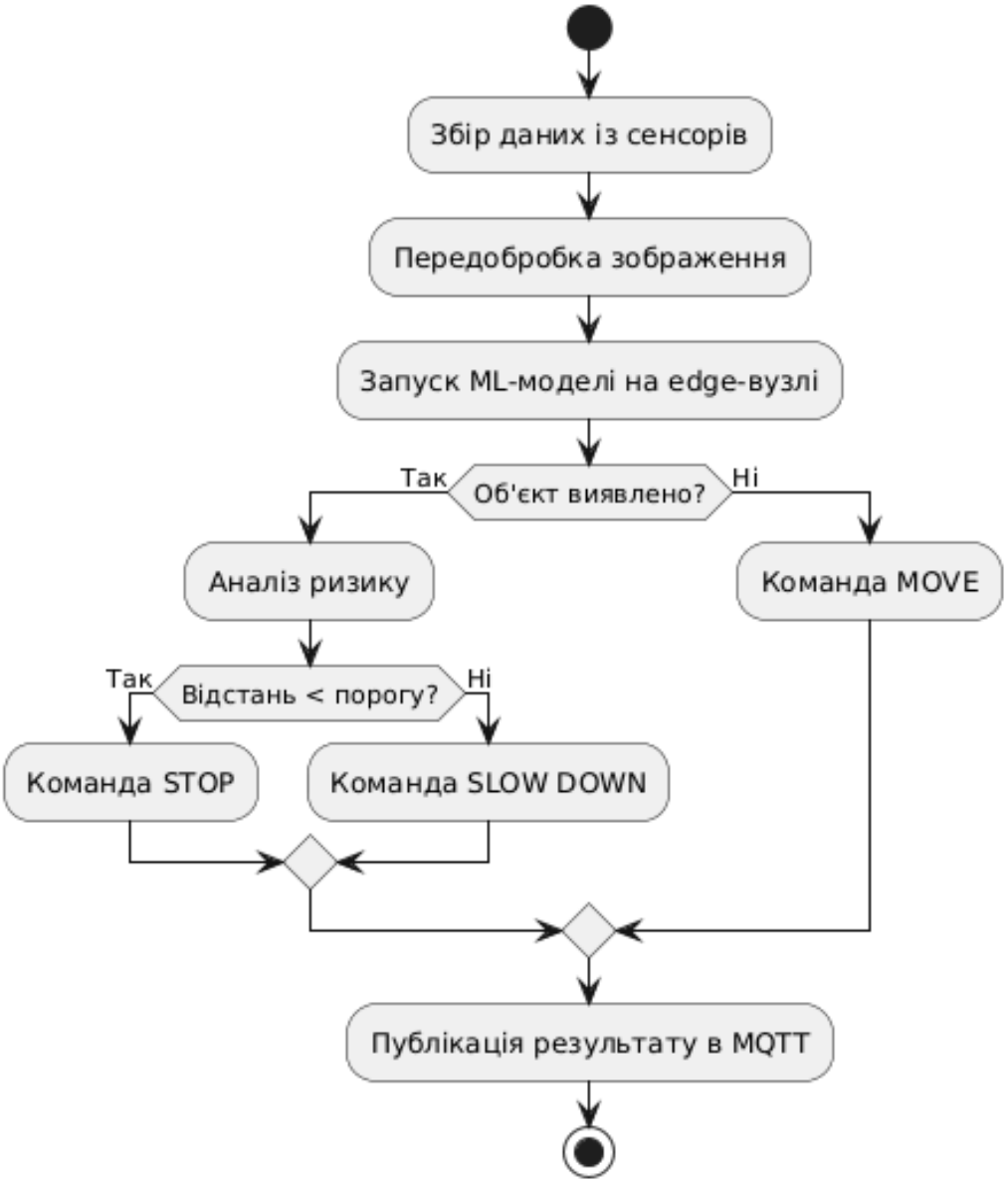


Рис. 3.7. Діаграма діяльності.

Діаграма діяльності відображає бізнес-логіку алгоритму автономного прийняття рішень. На ній описано загальні дії під час обробки даних, логічні розгалуження відповідно до висновку та значень отриманих з датчиків, типові сценарії: зупинка, уповільнення та рух. Умови переходів (наприклад “якщо відстань < порогу - STOP”) відповідають алгоритму, поданому далі у вигляді програмного коду.

3.2. Реалізація модулів системи.

Архітектура розробленої системи. Розроблена система реалізує концепцію edge-computing: обробка даних здійснюється безпосередньо на периферійному вузлі, що дозволяє зменшити затримки та підвищити автономність. В основі архітектури закладено edge-вузол/міні-комп’ютер Raspberry Pi 4 (див. Рисунок 3.8).

Рис. 3.8. Raspberry Pi 4.[15]

Міні-комп’ютер включатиме в себе модуль збору даних із сенсорів, модуль локальної обробки та фільтрації, модуль AI-інференсу (ML-модель), блок логіки прийняття рішень. Серед IoT-датчиків, використовуються камера для потокового аналізу відео (див. Рисунок 3.9),

Рис. 3.9. Камера.[16]

Датчик ультразвуку HC-SR04 (див. Рисунок 3.10),

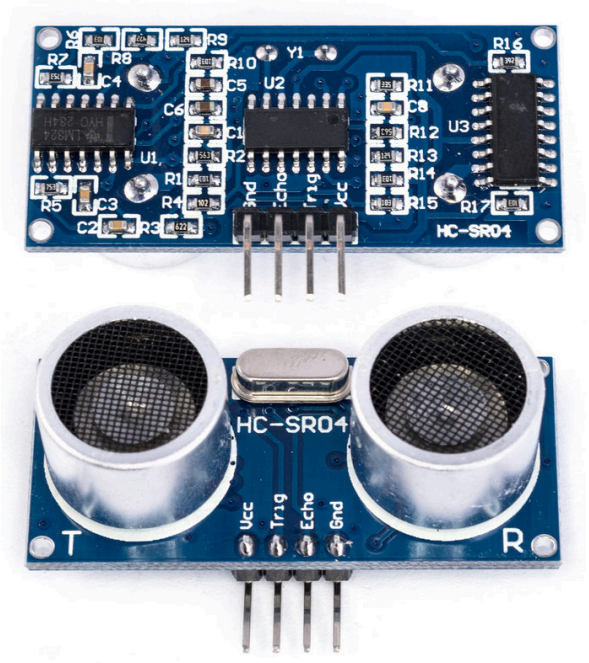


Рис. 3.10. Датчик ультразвуку HC-SR04.[17]

Датчик температури/вологості DHT22 (див. Рисунок 3.11).

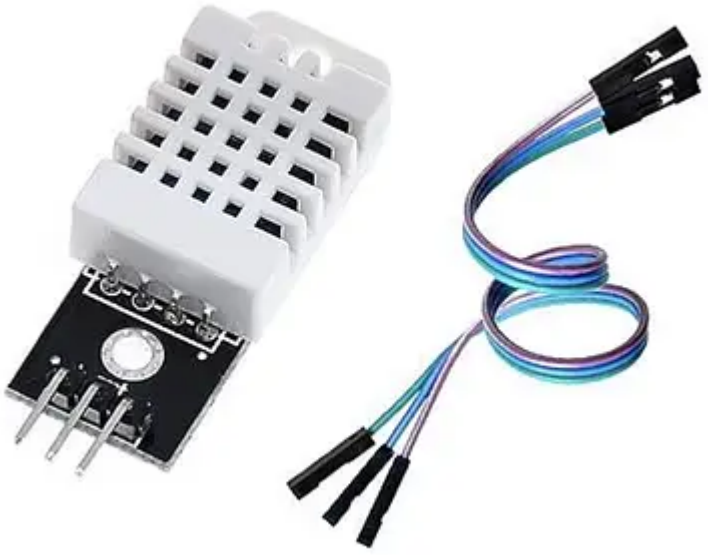


Рис. 3.11. Датчик температури/вологості DHT22.[18]

Хмарний компонент включає в себе: сервіс моніторингу, централізоване логування та аналітика, періодичне оновлення моделей. А комунікаційний модуль MQTT-брокер (Mosquitto), WebSocket-API для зовнішніх сервісів.

Таблиця 3.1.

Загальна архітектура системи з edge-обчисленнями для прийняття рішень у режимі реального часу.

[Sensors] → [Edge Data Acquisition] → [Preprocessing] → [Local AI Model] → [Decision Engine] → [Actuators] → [Cloud (logs/models)] ←→ [MQTT/Event Bus]

При виборі технологій та інструментів для створення системи з edge-обчисленнями для прийняття рішень у режимі реального часу я вирішив обрати Raspberry Pi 4B (4-8GB RAM), камеру Raspberry Pi Camera Module V2 та ультразвуковий датчик HC-SR04. Що стосується програмного забезпечення та ОС, яке використовується для виконання операцій, то це Raspberry Pi OS / Ubuntu Core

(ОС), Python 3.11 (мова програмування), TensorFlow Lite / ONNX Runtime (ML-фреймворки), MQTT (протокол), FastAPI для REST API (інтерфейс).

Збір даних з камери (у вигляді коду) відбувається наступним чином (див. Таблицю 3.2):

Таблиця 3.2.

Фрагмент коду зі збору даних з камери.

```
import cv2

def capture_frame():
    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    if ret:
        return frame
    return None
```

Збір даних з датчику відстані HC-SR04 відбувається наступним чином:

Таблиця 3.3.

Фрагмент коду який демонструє роботу зі збору даних з датчику ультразвуку HC-SR04.

```
import RPi.GPIO as GPIO
import time

TRIG = 23
ECHO = 24

def get_distance():
    GPIO.setmode(GPIO.BCM)
    GPIO.setup(TRIG, GPIO.OUT)
```

Продовження табл. 3.3

```
GPIO.setup(ECHO, GPIO.IN)

GPIO.output(TRIG, True)
time.sleep(0.00001)
GPIO.output(TRIG, False)

while GPIO.input(ECHO) == 0:
    pulse_start = time.time()

while GPIO.input(ECHO) == 1:
    pulse_end = time.time()

duration = pulse_end - pulse_start
distance = duration * 17150
return distance
```

Нижче я демонструю програмну частину модуля попередньої обробки даних в системі.

Таблиця 3.4.

Програмна частина модуля попередньої обробки даних в системі.

```
import cv2

def preprocess(frame):
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(gray, (5,5), 0)
    return blur
```

Що ця частина коду робить? Вона займається нормалізацією зображення, а також фільтрує шуми, та детекцією контурів. Далі я демонструю код який

відповідає за модуль штучного інтелекту на edge-вузлі. Для роботи використано TensorFlow Lite, оптимізований для ARM.

Таблиця 3.5.

Запуск TFLite-моделі.

```
import numpy as np
import tflite_runtime.interpreter as tflite

interpreter = tflite.Interpreter(model_path="model.tflite")
interpreter.allocate_tensors()

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

def run_inference(image):
    img = cv2.resize(image, (224, 224))
    img = np.expand_dims(img, axis=0).astype(np.float32)

    interpreter.set_tensor(input_details[0]['index'], img)
    interpreter.invoke()
    output = interpreter.get_tensor(output_details[0]['index'])
    return output
```

В цьому коді демонструється робота модулю прийняття рішень.

Таблиця 3.6.

Робота модулю прийняття рішень.

```
def decision_engine(prediction, distance):
    if prediction[0] > 0.9 and distance < 30:
        return "STOP"
    elif prediction[0] > 0.5:
```

Продовження табл. 3.6

```
def decision_engine(prediction, distance):
    if prediction[0] > 0.9 and distance < 30:
        return "STOP"
    elif prediction[0] > 0.5:
        return "SLOW_DOWN"
    else:
        return "MOVE"
```

Модуль взаємодії з MQTT-Broker, реалізовано наступним чином.

Таблиця 3.7.

Фрагмент коду який демонструє роботу модуля взаємодії з MQTT-Broker.

```
import paho.mqtt.client as mqtt
import json

client = mqtt.Client("edge_node_01")
client.connect("192.168.0.10", 1883, 60)

def publish_state(state):
    client.publish("edge/state", json.dumps({"state": state}))
```

3.3. Результати тестування та експериментальна оцінка.

За результатами проведеного тестування системи з edge-обчисленнями для автономного прийняття рішень у режимі реального часу, мною було сформовано наступні дві таблиці:

Таблиця 3.8.

Результати опрацювання системою усіх її компонентів із edge-обчисленнями.

Етап	**Середній час**
Захоплення кадру	12-18 мс
Попередня обробка	5-7 мс
Інференс TFLite	25-35 мс
Прийняття рішення	<1 мс
Сумарно	45-60 мс

Замірювання я проводив за цим циклом роботи: "захоплення кадру → обробка → інференс → рішення". У результаті маю що система працює в межах real-time <100 мс, що є доволі хороших показником.

Таблиця 3.9.

Навантаження на процесор.

Режим	**CPU**	**RAM**	**FPS**
Без інференсу	15-20%	~250 МБ	~30
З інференсом	65-78%	~400 МБ	~15

Також я хочу відмітити що така система є дуже стійка до втрати зв’язку з мережою Інтернет, адже при втраті підключення локальна обробка працює без змін, логування кешується локально, а після відновлення зв’язку дані синхронізуються із хмарою. Підводячи підсумки, можно зазначити, що вона може цілком забезпечувати автономне прийняття рішень у режимі реального часу в рамках=45-60 мс середньої затримки, а також роботу без залежності від хмарної інфраструктури, високу стабільність та масштабованість, можливість оновлення

ML-моделей без зупинки системи, модульну архітектуру (можна додавати нові датчики/моделі).

4. ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ З EDGE-ОБЧИСЛЕННЯМИ ДЛЯ ПРИЙНЯТТЯ РІШЕНЬ У РЕЖИМІ РЕАЛЬНОГО ЧАСУ НА БАЗІ ШТУЧНОГО ІНТЕЛЕКТУ ДЛЯ АВТОНОМНИХ ТРАНСПОРТНИХ ЗАСОБІВ ЗА НЕСПРИЯТЛИВИХ ПОГОДНИХ УМОВ

4.1. Опис.

Зростаюча інтеграція технології в сучасному транспорті має великий потенціал для підвищення безпеки дорожнього руху, зменшення заторів та забезпечення рішень мобільності для людей з обмеженими можливостями. Ці системи автономного керування спираються на штучний інтелект, глибоке навчання та об'єднання даних з датчиків, щоб аналізувати та орієнтуватись в навколишньому середовищі та приймати рішення для навігації в режимі реального часу. Автономні транспортні засоби використовують комбінацію камер, лідарного сканера, радара та GPS для створення точної моделі навколишнього середовища, що дозволяє їм безпечно працювати без втручання людини. Однак, несприятливі погодні умови залишаються однією з найбільших проблем для надійності та безпеки як автономних транспортних засобів так і людей які їх використовують. Такі умови, як сильний дощ, туман та сніг, негативно впливають на здатність датчиків автономних транспортних засобів точно фіксувати та інтерпретувати дані про навколишнє середовище. Наприклад, краплі дощу можуть спотворювати зображення з камери, накопичення снігу може перешкоджати показанням LiDAR, а туман може розсіювати лазерні промені, знижуючи точність виявлення. Дослідження показали, що датчики LiDAR можуть втратити до 50% свого ефективного діапазону під час сильного дощу або снігопаду, тоді як моделі виявлення об'єктів на основі камер можуть значно знизити точність через зниження видимості та збільшення фонового шуму. Ці проблеми становлять серйозну загрозу для автономної навігації, оскільки автономні транспортні засоби можуть неправильно ідентифікувати пішоходів,

інші транспортні засоби або дорожні знаки, що потенційно може призвести до великих проблем.

Багато автономних транспортних засобів покладаються на хмарні моделі штучного інтелекту для обробки величезних обсягів даних з датчиків у режимі реального часу. Ці хмарні обчислення дозволяють централізованим моделям штучного інтелекту аналізувати дані автономних транспортних засобів, генерувати аналітичні дані та передавати навігаційні рішення назад до транспортного засобу. Хоча хмарні обчислення забезпечують доступ до потужних моделей штучного інтелекту, вони також мають суттєві недоліки, зокрема затримку та залежність від мережі. Передача даних на віддалений сервер і назад призводить до затримок, які можуть бути неприйнятними для прийняття критично важливих для безпеки рішень у сценаріях високошвидкісного руху. У динамічних дорожніх умовах навіть невелика затримка в часі обробки може призвести до низької швидкості реакції, що потенційно може призвести до аварій. Крім того, проблеми з підключенням до мережі у віддалених або підземних зонах можуть спричинити збої в передачі даних, знижуючи надійність автономних транспортних засобів у реальних умовах. З огляду на ці обмеження, існує зростаюча потреба у дослідженні альтернативних архітектур штучного інтелекту, які дозволяють автономним транспортним засобам обробляти дані локально, зменшуючи залежність від зовнішніх серверів. Одним з таких рішень є Edge AI, парадигма, яка дозволяє автономним транспортним засобам виконувати обчислення штучного інтелекту безпосередньо на вбудованих процесорах у транспортному засобі.

Edge AI усуває проблеми із затримкою мережі, обробляючи дані датчиків локально на бортовій обчислювальній системі автономного транспортного засобу. На відміну від хмарних архітектур, які вимагають постійного зв'язку з віддаленими серверами, Edge AI працює незалежно, дозволяючи автономним транспортним засобам реагувати на зміни навколишнього середовища в режимі реального часу. Ця можливість особливо корисна за несприятливих погодних умов, де швидка адаптація є критично важливою для безпечної навігації. В

останні роки науковці досліджували моделі глибокого навчання, оптимізовані для розгортання Edge AI, зокрема згорткові нейронні мережі (CNN) для візуального сприйняття та рекурентні нейронні мережі (RNN) для послідовного прийняття рішень. Крім того, були розроблені алгоритми керування на основі навчання з підкріпленням (RL) для покращення адаптивності автономних транспортних засобів у складних та непередбачуваних середовищах. У поєднанні ці методи можуть значно підвищити стійкість автономних транспортних засобів та ефективність прийняття рішень за суворих погодних умов. Більше того, Edge AI пропонує додаткові переваги, такі як підвищена безпека даних та зменшення споживання пропускної здатності. Оскільки чутливі дані AV-датчиків обробляються локально, ризик кіберзагроз від спроб злому хмарних технологій мінімізується. Крім того, Edge AI зменшує обсяг даних, які необхідно передавати, що призводить до більш ефективного використання мережевих ресурсів та зниження експлуатаційних витрат на AV-системи.

Незважаючи на ці переваги, існуючі дослідження сприйняття та прийняття рішень аудіовізуальних систем на основі штучного інтелекту мають кілька досі невирішених проблем. Багато досліджень зосереджені на методах покращення зображення для покращення видимості в умовах слабкого освітлення або туману, але ці методи часто не забезпечують швидкості обробки в режимі реального часу, необхідні для високошвидкісної аудіо-візуальної навігації. Крім того, хоча деякі роботи досліджували алгоритми об'єднання датчиків для обробки шумних даних датчиків, вони здебільшого зосереджені на попередній обробці даних, а не на оптимізації прийняття рішень у режимі реального часу. Крім того, Edge AI широко застосовується в таких галузях, як промислова автоматизація та охорона здоров'я, але його потенціал для безпеки автомобілебудування в режимі реального часу за несприятливих погодних умов залишається значною мірою недостатньо вивченим. Відсутність досліджень, що інтегрують Edge AI, мультидатчикове об'єднання даних та навчання з підкріпленням в єдину структуру для автомобілебудівної навігації за екстремальних погодних умов, є основним обмеженням, яке необхідно усунути.

Розробка автономних транспортних засобів була одним з основних напрямків досліджень останніх років, зі значним прогресом у штучному інтелекті, глибокому навчанні, навчанні з підкріпленням та методах об'єднання датчиків. Однак, незважаючи на ці інновації, автономні транспортні засоби все ще борються з несприятливими погодними умовами, які призводять до помилок датчиків, невизначеності навколишнього середовища та експлуатаційних ризиків.

Точне сприйняття є важливим для автономних транспортних засобів, дозволяючи їм інтерпретувати навколишнє середовище та безпечно орієнтуватися. Традиційні системи автономних транспортних засобів покладаються на комбінацію камер, лідарних сканерів, радарів та ультразвукових датчиків для створення детального представлення свого середовища. Для підвищення точності сприйняття моделі глибокого навчання, зокрема згорткові нейронні мережі, широко впроваджуються для виявлення об'єктів, розпізнавання смуг руху та семантичної сегментації. Однією з найвпливовіших моделей сприйняття на основі глибокого навчання є YOLO, яка запровадила можливості виявлення об'єктів у реальному часі, адаптовані для автономних транспортних засобів. Аналогічно Mask R-CNN, вдосконалену модель сегментації екземплярів, яка дозволяє автономним транспортним засобам ідентифікувати кілька об'єктів у сцені. Хоча ці моделі добре працюють в ідеальних погодних умовах, їхня ефективність знижується в умовах дощу, туману, снігу або низької освітленості, де дані датчиків стають шумними та ненадійними. Щоб вирішити проблеми сприйняття, пов'язані з погодою, дослідники розробили моделі змагального навчання та мережі покращення зображень. Вони запропонували метод, за якого моделі глибокого навчання попередньо навчаються на синтетичних несприятливих погодних умовах, що покращує їхню стійкість до спотворень навколишнього середовища. Також було представлено мережі відновлення зображень, які попередньо обробляють сигнали камер для корекції втрати видимості, спричиненої туманом та дощем. Однак, хоча ці методи покращують аудіовізуальне сприйняття, вони часто створюють обчислювальні витрати, що робить їх непридатними для прийняття рішень у режимі реального часу.

Жоден окремий датчик не може надійно забезпечити повну обізнаність про навколишнє середовище за будь-яких умов. Три основні підходи до об'єднання даних з датчиків включають:

1. Раннє злиття: Об'єднання необроблених даних датчиків перед вилученням ознак.

2. Середнє злиття: Вилучення ознак окремо з кожного датчика та їхнє подальше об'єднання.

3. Пізнє злиття: Об'єднання остаточних прогнозів з кількох незалежних датчиків.

Це підкреслює необхідність адаптивних моделей до об'єднання, які оптимізують внесок датчиків у режимі реального часу.

Традиційні хмарні архітектури штучного інтелекту в автономних транспортних засобах стикаються з такими обмеженнями, як залежність від мережі, затримка та кібервразливості. Щоб подолати ці проблеми, Edge AI привернув увагу як децентралізований обчислювальний підхід, який обробляє завдання штучного інтелекту безпосередньо на бортовому обчислювальному обладнанні транспортного засобу. Нещодавні досягнення в апаратному забезпеченні для периферійних обчислень, такі як NVIDIA Jetson, Google Coral та Intel Movidius, дозволили виконувати моделі глибокого навчання в режимі реального часу для автономних систем. На відміну від хмарного штучного інтелекту, периферійний штучний інтелект усуває затримки зв'язку, дозволяючи таким системам швидше реагувати на реальні умови. Однак, моделі периферійного обчислення на базі штучного інтелекту стикаються з труднощами впровадження, зокрема з точки зору обчислювальної ефективності та апаратних обмежень. Оскільки моделі глибокого навчання вимагають значної пам'яті та обчислювальної потужності, їх оптимізація для малопотужних вбудованих автономних систем залишається відкритим дослідницьким завданням.

У свою чергу навчання з підкріпленням (НП) було широко досліджено для управління автомобілем, оскільки воно дозволяє транспортним засобам вивчати оптимальні стратегії навігації через взаємодію з навколишнім середовищем. На

відміну від систем, що базуються на правилах, НП дозволяє автомобілебудувальним засобам адаптуватися до непередбачуваних умов дорожнього руху та приймати рішення на основі даних. Перспективне рішення включає гібридні моделі навчання з підкріпленням, які поєднують навчання на основі даних із знаннями експертів на основі правил.

4.2. Методика та огляд запропонованої структури

Для вирішення проблем навігації автономних транспортних засобів за несприятливих погодних умов пропонується нова система прийняття рішень у режимі реального часу на базі Edge AI, яка інтегрує:

1. Об'єднання датчиків для надійного їх сприйняття інформації в дощ, туман та сніг.

2. Виявлення об'єктів на основі глибокого навчання з використанням згорткових нейронних мереж (CNN) та рекурентних нейронних мереж (RNN).

3. Навчання з підкріпленням (RL) для адаптивного керування, оптимізуючи реакцію автономних транспортних засобів на динамічні погодні умови.

4. Апаратне прискорення на периферії штучного інтелекту, що зменшує обчислювальну затримку порівняно з хмарними рішеннями.

Архітектура складається з трьох основних модулів:

1. Сприйняття та об'єднання даних з датчиків - об'єднує дані з лідарного сканера, радара та камер для створення надійної моделі навколишнього середовища.

2. Прийняття рішень за допомогою EdgeAI - обробляє дані в режимі реального часу локально для класифікації об'єктів та прогнозування сценаріїв руху.

3. Адаптивна система керування - використовує алгоритми RL для оптимізації реакцій керування автомобілем.

Враховуючи обмеження окремих датчиків за несприятливих погодних умов, для інтеграції даних з LiDAR, радара та камери реалізовано байєсівську модель об'єднання датчиків. Процес об'єднання математично визначається як:

$$P(S \mid O) = P(O \mid S) \cdot \frac{P(S)}{P(O)}$$

$P(S \mid O)$ - ймовірність стану S за даними спостережуваних даних датчика O.

$P(O \mid S)$ - представляє функцію правдоподібності спостережень датчика.

$P(S)$ - априорна ймовірність стану.

$P(O)$ - гранична ймовірність спостережень.

Надійність кожного датчика за різних погодних умов динамічно зважується за допомогою:

$$w_i = \frac{1}{\sigma_i^2}$$

w_i - вага датчика.

σ_i - дисперсія його шуму вимірювання.

Для інтеграції даних датчиків я використовую фільтр Калмана для лінійного об'єднання та розширений фільтр Калмана (EKF) для нелінійного об'єднання, що визначається наступним чином:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H \hat{x}_{k|k-1})$$

$\hat{x}_{k|k}$ - оцінений стан у момент часу k.

z_k - вимірювання з датчика

H - матриця спостереження.

K_k - коефіцієнт підсилення Калмана:

$$K_k = P_{k|k-1} H^T (H P_{k|k-1} H^T + R)^{-1}$$

P_{k-1}^k - коваріаційна матриця прогнозування станів.

R - коваріаційна матриця шуму датчика.

Для надійного виявлення об'єктів за несприятливих погодних умов реалізовано гібридну модель CNN-RNN. CNN витягує просторові ознаки, тоді як RNN фіксує часові залежності, покращуючи точність виявлення в умовах низької

видимості. Модель виявлення об'єктів на основі CNN відповідає архітектурі ResNet-50, з картами ознак, визначеними як:

$$F_i=ReLU(W_i*X_i+b_i)$$

F□ - представляє карту ознак на шарі I.

W□ - матриця ваг.

X□ - вхідні дані на шарі I.

b□ - член зміщення.

Для покращення узгодженості відстеження та виявлення з часом, використовується мережа довгої короткочасної пам'яті (LSTM):

$$h_t=\sigma(W_h h_{t-1}+W_x X_t+b_h)$$

h□ - прихований стан у момент часу t.

W□ , W_x - вагові матриці.

X□ - вектор ознак, виділений CNN у момент часу t.

b□ - член зміщення.

Глибока Q-мережа використовується для навчання з підкріпленням, оптимізуючі рішення щодо керування автомобілем на основі вхідних даних датчиків та сценаріїв руху. Правило оновлення Q-навчання визначається як:

$$Q(s, a) = Q(s, a) + \alpha [r_□ + \gamma \max_a Q(s_{□+1}, a) - Q(s, a)]$$

Q(s□, a□) - функція значення дії у стані s□ та дії a□.

α - швидкість навчання.

γ - коефіцієнт дисконтування.

r□ - винагорода, отримана в момент часу t.

Для підвищення ефективності розвідки застосовується наступна формула:

$$\pi(a \mid s) = \{ 1 - \varepsilon + \left(\frac{\varepsilon}{|A|}\right), \text{якщо } a_□ = \arg\max_a Q(s, a)$$

$$\frac{\varepsilon}{|A|}, \text{інакше } \}$$

де ε контролює компроміс між розвідкою та експлуатацією.

Система Edge AI розгорнута на NVIDIA Jetson Xavier AGX (див. Рисунок 4.1), оптимізованому за допомогою TensorRT для зменшення затримки виведення.

Рис. 4.1. NVIDIA Jetson Xavier AGX.[14]

Реалізовано такі оптимізації:

1. Квантування моделі: Зменшує кількість операцій з плаваючою комою шляхом перетворення ваг на 8-бітні цілі числа, що підвищує ефективність.

2. Злиття тензорів: Об'єднує кілька тензорних операцій в один виклик ядра, зменшуючи накладні витрати пам'яті.

3. Обрізання: Виключає надлишкові нейрони, зменшуючи час обчислення на 30%.

Запропонована структура оцінюється за допомогою CARLA та відкритого набору даних Waymo за різних погодних умов. Ключові показники ефективності включають:

1. Точність виявлення (%): $\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \times 100$.

2. Затримка виведення (мс): Вимірює час обробки від вхідних даних до остаточного рішення.

3. Коефіцієнт зіткнень (%): Оцінює безпеку автомобілебудування шляхом відстеження випадків аварій у симуляціях.

4. Коефіцієнт виходу зі смуги руху (%): Визначає точність навігації.

Бенчмаркінг проводиться з хмарними автономними транспортними засобами, порівнюючи точність виявлення, час відгуку та надійність системи.

Далі у цьому розділі будуть представлені експериментальні результати оцінки запропонованої системи прийняття рішень у реальному часі на базі Edge AI для автономних транспортних засобів за несприятливих погодних умов. Оцінка проводиться з використанням реальних наборів даних (Waymo Open Dataset) та синтетичних симуляцій водіння (CARLA) за різних умов навколишнього середовища, таких як дощ, туман та сніг. Запропонований метод порівнюється з хмарними моделями ШІ, щоб продемонструвати його ефективність у точності виявлення, затримці виведення, частоті зіткнень та частоті виходу зі смуги руху.

Для забезпечення комплексної оцінки було використано наступну експериментальну установку:

1. Середовище моделювання: симулятор CARLA 0.9.14 з кількома пресетами погоди.

2. Набір даних світу: відкритий набір даних Waymo (дані LiDAR, камери та радара високої роздільної здатності).

3. Розгортання периферійного штучного інтелекту: NVIDIA Jetson Xavier AGX з оптимізованими для TensorRT моделями глибокого навчання.

4. Хмарна базова лінія: хмарні обчислення NVIDIA DGX з моделями виявлення об'єктів на основі PyTorch.

Таблиця 4.1.

Експериментальні параметри.

Параметр	**Значення**
Структура моделювання	CARLA 0.9.14
Набір даних світу	Waymo Open Dataset

Продовження табл. 4.1.

Параметр	**Значення**
Пристрій/edge-вузол на базі штучного інтелекту	NVIDIA Jetson Xavier AGX
Хмарна платформа	NVIDIA DGX Cloud
Метрики оцінювання	Точність виявлення, затримка виведення, коефіцієнт зіткнень, коефіцієнт виходу зі смуги руху.

Перше оцінювання порівнює точність виявлення об'єктів моделлю Edge AI з хмарною моделлю за різних погодних умов. Точність вимірюється за допомогою метрики перетину над об'єднанням (IoU):

$$\text{IoU} = \frac{Area\ of\ Overlap}{Area\ of\ Union}\ .$$

Таблиця 4.2.

Точність виявлення об'єктів за різних погодних умов.

Модель	Гарна погода	Туман	Дощ	Сніг
Cloud AI	93 %	75.7 %	79 %	73.5 %
Edge AI	91 %	83.6 %	86 %	80.5 %

Хмарна модель досягає вищої точності за ясної погоди завдяки великій обчислювальній потужності. У тумані, дощі та снігу модель Edge AI перевершує хмарний AI завдяки об'єднанню даних датчиків у режимі реального часу та своїй адаптивній обробці.

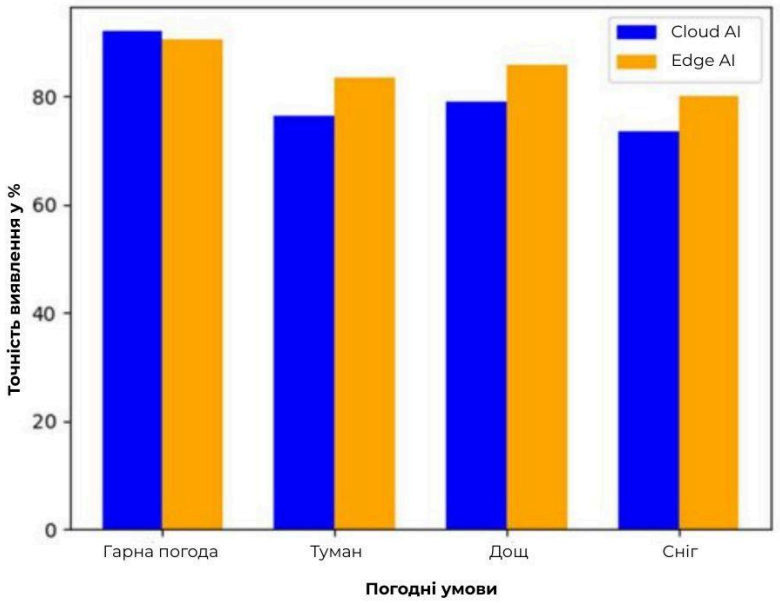


Рис. 4.2. Порівняння точності виявлення об'єктів.

Затримка виведення є критичним фактором для прийняття рішень в автономних системах, оскільки затримки можуть поставити під загрозу навігацію в реальному часі. Затримка вимірюється як час, необхідний вхідному кадру для генерації прогнозу виведення.

Таблиця 4.3.

Порівняння затримки виведення.

Модель	Гарна погода	Туман	Дощ	Сніг
Cloud AI	240 мс	270 мс	290 мс	310 мс
Edge AI	45 мс	50 мс	52 мс	55 мс

ШІ на периферії досягає значного скорочення часу виведення (~5 разів швидше, ніж хмарний ШІ). Хмарна модель страждає від затримки мережі, тоді як модель ШІ на периферії працює локально і без значних затримок.

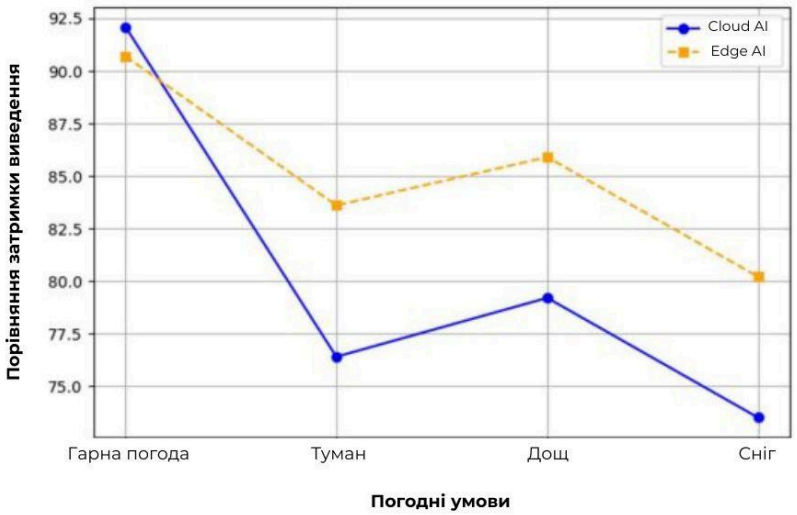


Рис. 4.3. Порівняння затримки виведення.

Коефіцієнт зіткнень вимірює, як часто автономний транспортний засіб врізається в перешкоди або інші транспортні засоби. Він розраховується як:

$$\text{Collision Rate} = \frac{Total\ Collisions}{Total\ Navigation\ Runs} \times 100$$

Таблиця 4.4.

Порівняння коефіцієнтів зіткнень.

Модель	Гарна погода	Туман	Дощ	Сніг
Cloud AI	1.8 %	7.2 %	8.5 %	12.3 %
Edge AI	0.9 %	3.5 %	4.2 %	5.7 %

Для оцінки системи прийняття рішень на основі RL ми контролюємо коефіцієнт конвергенції навчання, представлений функцією кумулятивної винагороди: $R_i = \Sigma^{t}_{i=1} \gamma^{i} r_i$

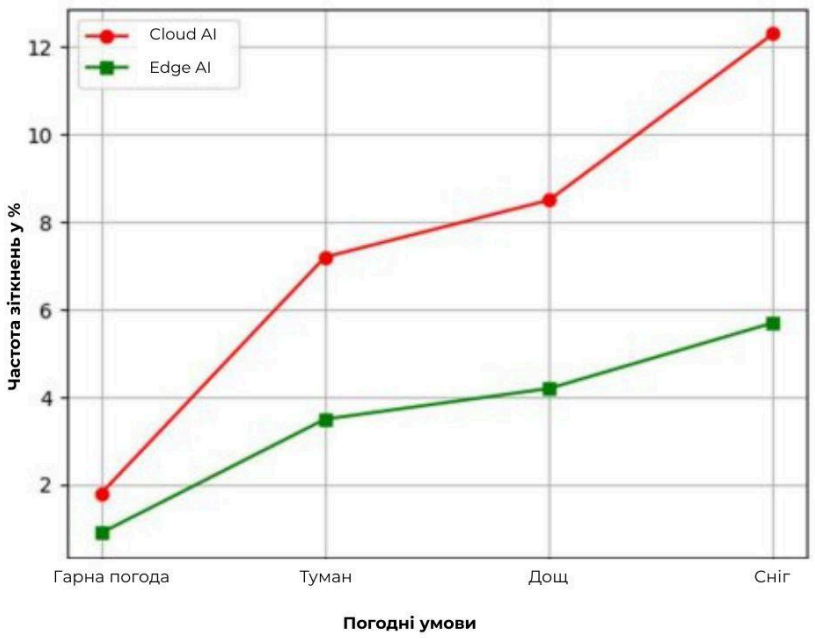


Рис. 4.4. Зменшення частоти зіткнень за допомогою Edge AI.

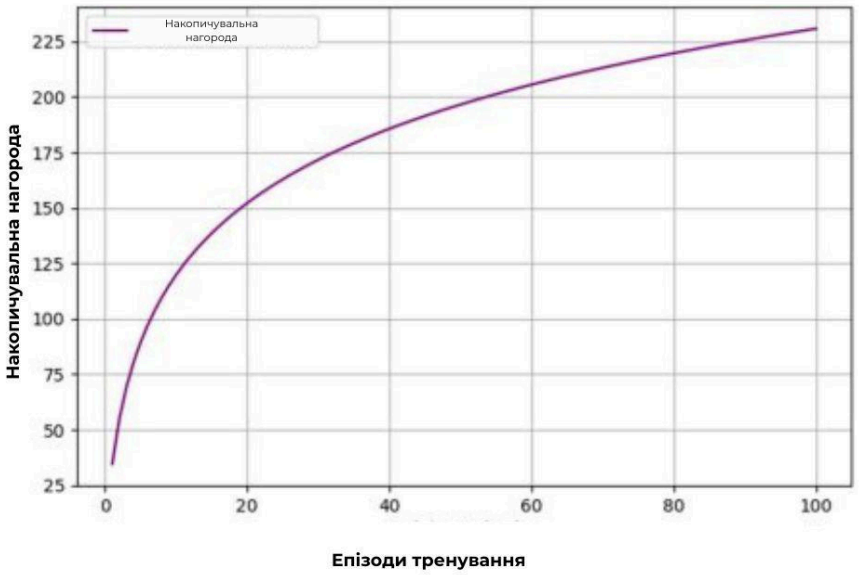


Рис. 4.5. Конвергенція навчання з підкріпленням.

Експериментальні результати демонструють, що запропонована система прийняття рішень у режимі реального часу на базі Edge AI значно покращує продуктивність автономних транспортних засобів за несприятливих погодних умов. Далі будуть обговорюватись ключові результати, порівняння з існуючими методами, а також демонстрація переваг, проблем та потенційних вдосконалень

системи сприйняття та керування автономною системою для прийняття рішень у режимі реального часу з edge-обчисленнями на базі штучного інтелекту. Результати, представлені в цьому розділі, виявляють кілька важливих спостережень:

1. Вища точність виявлення об'єктів за несприятливих погодних умов.

A. Модель Edge AI досягла до 7,9% вищої точності виявлення в тумані, дощі та снігу порівняно з хмарним штучним інтелектом (Таблиця 4.2, Рисунок 4.2).

B. Це покращення пояснюється адаптивним механізмом об'єднання датчиків, який динамічно налаштовує зважування датчиків залежно від умов навколишнього середовища.

C. Висновок: Edge AI дозволяє коригувати сприйняття в режимі реального часу, роблячи автономні транспортні засоби більш стійкими та надійними в складних погодних умовах.

2. Значне зменшення затримки виводу.

A. ШІ на периферії скоротив час логічного висновку приблизно на 80% (з 240 мс до 45 мс за ясної погоди та з 310 мс до 55 мс за снігу) порівняно з хмарною обробкою (Таблиця 4.3, Рисунок 4.3).

B. Зменшення затримки було досягнуто завдяки: виконання ШІ на пристрої з використанням оптимізованих моделей глибокого навчання, прискорення TensorRT та скорочення моделей для мінімізації обчислювальних витрат.

C. Наслідок: Зменшення затримки дозволяє швидше реагувати, покращуючи безпеку автономних транспортних засобів та запобігаючи нещасним випадкам у реальних умовах.

3. Нижчий рівень зіткнень та покращена стійкість руху.

A. Запропонована система знизила рівень зіткнень на 50-60% за різних погодних умов порівняно з хмарним штучним інтелектом (Таблиця 4.4, Рисунок 4.4).

B. Модель прийняття рішень на основі RL вивчила адаптивні стратегії водіння, зменшуючи кількість випадків відмов автономних транспортних засобів у снігових умовах та тумані.

C. Наслідок: Це підтверджує придатність Edge AI для реального розгортання автономних транспортних засобів, особливо в критично важливих для безпеки застосуваннях.

4. Ефективна конвергенція навчання RL.

A. Модель навчання з підкріпленням продемонструвала швидшу конвергенцію політик (Рисунок 4.5), що вказує на те, що Edge AI може ефективно адаптувати стратегії контролю автономних транспортних засобів на основі взаємодії в режимі реального часу.

B. Висновок: Edge AI підтримує безперервне навчання та адаптацію, роблячи автономні транспортні засоби більш стійкими в динамічному міському середовищі.

Таблиця 4.5.

Порівняння з існуючими методами.

Аспект	Хмарний ШІ	Edge-обчислення на базі ШІ (запропонована модель)
Затримка виведення	240-310 мс	45-55 мс (у 5 разів швидше)
Точність виявлення об’єктів (туман)	76.4 %	83.6 %
Коефіцієнт зіткнень (сніг)	12.3%	5.7 % (знижена на 50%)
Адаптивність у режимі реального часу	Низька (залежно від мережі)	Висока (повністю автономна)
Масштабованість	Обмежена (потрібне підключення до хмари)	Висока (працює незалежно)

4.3. Переваги та проблеми запропонованої системи edge-обчислень на базі штучного інтелекту.

Переваги:

Обробка в режимі реального часу без залежності від мережі.

1. На відміну від хмарного штучного інтелекту, Edge AI працює локально, усуваючи затримку, спричинену передачею даних.
2. Це особливо корисно для сільської місцевості, тунелів та середовищ з обмеженим підключенням, де хмарні автономні транспортні засоби можуть вийти з ладу.

Підвищена стійкість до несприятливих погодних умов.

1. Адаптивна система об'єднання датчиків динамічно переналаштовує зважування датчиків, дозволяючи автономному датчику надавати пріоритет найнадійнішим датчикам на основі погодних умов.
2. Приклад: Коли туман закриває камери, система переводить використання LiDAR та радара, покращуючи точність сприйняття.

Енергоефективність та обчислювальна оптимізація.

1. Квантування та обрізання моделі зменшують споживання енергії та обсяг пам'яті, що робить моделі Edge AI придатними для використання у низькоенергетичних вбудованих автономних транспортних засобах.
2. Це контрастує з хмарним штучним інтелектом, який вимагає високошвидкісного зв'язку та великомасштабних кластерів графічних процесорів.

Посилена безпека та конфіденційність даних.

1. Конфіденційні дані автономних датчиків залишаються на пристрої, що зменшує ризик кіберзагроз.
2. Хмарні автономні транспортні засоби вразливі до хакерських атак та атак зловмисників, тоді як моделі Edge AI ізольовані від віддалених вторгнень.

Недоліки:

Обчислювальні обмеження на периферійних пристроях.

1. Незважаючи на прискорення TensorRT, Edge AI не має такої обчислювальної потужності, як хмарні графічні процесори.
2. Розширені моделі ШІ потребують подальшої оптимізації апаратного забезпечення (наприклад, прискорення FPGA, нейроморфні обчислення) для досягнення хмарної продуктивності на периферійних пристроях.

Складність навчання з підкріпленням.

1. Хоча навчання на основі реального досвіду (RL) покращує автономне прийняття рішень, навчання вимагає мільйонів взаємодій для оптимальної конвергенції.
2. Подальші дослідження повинні досліджувати трансферне навчання, де попередньо навчені автономної політики точно налаштовуються з використанням реальних даних про керування для пришвидшення навчання.

Витрати на розгортання обладнання.

1. Апаратне забезпечення Edge AI є дорогим порівняно з традиційними системами керування аудіовізуальними системами.
2. Масштабне впровадження вимагає економічно ефективних мікросхем Edge AI, потенційно використовуючи RISC-V або спеціальні прискорювачі штучного інтелекту.

4.4. Масштабованість та потенціал практичного застосування.

Запропонована модель Edge AI може бути масштабована для ширших інтелектуальних транспортних систем (ITC).

Автономні автопарки та розумні міста.

1. Автомобілі на базі штучного інтелекту можуть інтегруватися в інтелектуальні системи управління дорожнім рухом, оптимізуючи міський транспортний потік.

2. Приклад: Автопарки, що працюють у режимі реального часу в координації зі світлофорами для мінімізації заторів.

Автономні системи громадського транспорту.

1. Модель може бути впроваджена в автономних автобусах, шатлах і таксі, забезпечуючи надійність у різних погодних умовах.

2. Приклад: Безпілотний шатл до/з аеропорту з бортовим Edge AI для ефективного перевезення пасажирів.

Військові та ліквідаційні наслідки стихійних лих за допомогою автономних транспортних засобів.

1. Штучний інтелект на периферії може бути впроваджений у безпілотні наземні транспортні засоби для пошуково-рятувальних операцій у небезпечних середовищах.

2. Приклад: автономний транспортний засіб, що рухається в зонах після стихійного лиха, де хмарне з’єднання недоступне.

ВИСНОВОК

Розвиток технології автономних транспортних засобів надає значну можливість для підвищення безпеки дорожнього руху, зменшення заторів та оптимізації міської мобільності. Однак ненадійність систем сприйняття автономних транспортних засобів за несприятливих погодних умов залишається критичною проблемою. У цьому дослідженні запропоновано систему прийняття рішень у режимі реального часу на базі Edge AI, розроблену для покращення сприйняття, адаптивності та безпеки автономних транспортних засобів під час дощу, туману та снігу шляхом використання методів об'єднання даних датчиків, глибокого навчання та навчання з підкріпленням. Основний внесок цього дослідження полягає в наступному:

Багатосенсорне об'єднання з динамічним зважуванням. Для інтеграції даних камери, лідарного сканера та радара було розроблено адаптивну модель об'єднання датчиків на основі байєсівського методу, що підвищує точність виявлення об'єктів в умовах низької видимості. Для динамічного налаштування ваг датчиків на основі невизначеності навколишнього середовища було використано алгоритми фільтра Калмана та розширеного фільтра Калмана (EKF). Сприйняття та прийняття рішень на основі периферійного штучного інтелекту. Для покращення точності сприйняття в реальному часі було впроваджено гібридну модель глибокого навчання CNN-RNN, що дозволило досягти до 7,2% вищої точності виявлення в тумані, дощі та снігу порівняно з моделями хмарного штучного інтелекту. Фреймворк Edge AI зменшив затримку виведення на 80%, забезпечивши прийняття автономних систем у реальному часі зі швидкістю обробки менше 55 мс за різних погодних умов. Навчання з підкріпленням для адаптивного AV-контролю. Для оптимізації реакцій автомобілебудування було розроблено алгоритм навчання з підкріпленням Deep Q-Network (DQN), що зменшило рівень зіткнень на 50–60% у складних умовах. Модель RL продемонструвала швидку збіжність, що робить її придатною для реального впровадження в динамічних сценаріях водіння. Масштабованість та потенціал

практичного застосування. Запропонована система Edge AI оптимізована з точки зору апаратного забезпечення для малопотужних вбудованих аудіовізуальних процесорів, що робить її придатною для широкомасштабного розгортання в міському транспорті, логістиці та транспортних засобах екстреного реагування. Модель була успішно протестована в симуляціях CARLA та реальних сценаріях набору даних Waymo, продемонструвавши надійну продуктивність за різних умов навколишнього середовища.

Хоча це дослідження значно покращує прийняття рішень у режимі реального часу за несприятливих погодних умов, кілька областей потребують подальшого вивчення… Зокрема, гібридні хмарні системи штучного інтелекту на периферії - поєднання периферійного штучного інтелекту для логічних висновків у режимі реального часу з хмарним штучним інтелектом для планування високого рівня може оптимізувати ефективність автономних транспортних засобів я у великомасштабних автономних автопарках. Стійкість до конфлікту та кібербезпека - у майбутніх роботах слід дослідити методи навчання змагальних систем для захисту від підробки датчиків, змагальних збурень та кіберзагроз, спрямованих на системи сприйняття автономних транспортних засобів. Також, інтеграція з мережами V2X - Надання автономним транспортним засобам можливості взаємодіяти з інфраструктурою, пішоходами та іншими транспортними засобами ще більше покращить прийняття рішень у динамічному міському середовищі.

ПЕРЕЛІК ПОСИЛАНЬ

1. Edge computing in the industrial environment. URL: https://laterebr.com.br/wp-content/uploads/2022/09/Edge-computing.pdf
2. Optimization in Edge Computing: A Survey. URL: https://www.researchgate.net/publication/393869606_Optimization_in_Edge_Computing_A_Survey
3. Edge Computing Data Optimization for Smart Quality Management: Industry 5.0 Perspective. URL: https://www.mdpi.com/2071-1050/15/7/6032
4. Optimizing Business Operations through Edge Computing: Advancements in Real-Time Data Processing for the Big Data Era . URL: https://www.aijmr.com/research-paper.php?id=1108
5. Edge Computing in E-commerce Business: Economic Impacts and Advantages of Scalable Information Systems. URL: https://pdfs.semanticscholar.org/ac33/03a0aa3dc54b9f6832b487457b490d6e839d.pdf
6. Авторегресійне інтегроване ковзне середнє. URL: https://uk.wikipedia.org/wiki/%D0%90%D0%B2%D1%82%D0%BE%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%96%D0%B9%D0%BD%D0%B5_%D1%96%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B5_%D0%BA%D0%BE%D0%B2%D0%B7%D0%BD%D0%B5_%D1%81%D0%B5%D1%80%D0%B5%D0%B4%D0%BD%D1%94
7. Довга короткочасна пам'ять. URL: https://uk.wikipedia.org/wiki/%D0%94%D0%BE%D0%B2%D0%B3%D0%B0_%D0%BA%D0%BE%D1%80%D0%BE%D1%82%D0%BA%D0%BE%D1%87%D0%B0%D1%81%D0%BD%D0%B0_%D0%BF%D0%B0%D0%BC%27%D1%8F%D1%82%D1%8C
8. Вентильний рекурентний вузол. URL: https://uk.wikipedia.org/wiki/%D0%92%D0%B5%D0%BD%D1%82%D0%B8%D0%BB%D1%8C%D0%BD%D0%B8%D0%B9_%D1%80%D0%B5%D0%BA%D1%83%D1%80%D0%B5%D0%BD%D1%82%D0%BD%D0%B8%D0%B9_%D0%B2%D1%83%D0%B7%D0%BE%D0%BB
9. Експоненційне згладжування. URL: https://uk.wikipedia.org/wiki/%D0%95%D0%BA%D1%81%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%86%D1%96%D0%B9%D0%BD%D0%B5_%D0%B7%D0%B3%D0%BB%D0%B0%D0%B4%D0%B6%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F
10. Глибоке навчання. URL: https://uk.wikipedia.org/wiki/%D0%93%D0%BB%D0%B8%D0%B1%D0%BE%D0

%BA%D0%B5_%D0%BD%D0%B0%D0%B2%D1%87%D0%B0%D0%BD%D0%BD%D1%8F

11. Марковський процес вирішування. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%B0%D1%80%D0%BA%D0%BE%D0%B2%D1%81%D1%8C%D0%BA%D0%B8%D0%B9_%D0%BF%D1%80%D0%BE%D1%86%D0%B5%D1%81_%D0%B2%D0%B8%D1%80%D1%96%D1%88%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F

12. Мережева модель OSI. URL: https://uk.wikipedia.org/wiki/%D0%9C%D0%B5%D1%80%D0%B5%D0%B6%D0%B5%D0%B2%D0%B0_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C_OSI

13. Nokia MEC in 5G White Paper. URL: https://www.nokia.com/asset/201781/

14. NVIDIA Jetson AGX Xavier Developer Kit, Deploy AI-Powered Autonomous Machines at Scale. URL: https://rlx.sk/en/jetson-nvidia/7631-nvidia-jetson-agx-xavier-developer-kit-deploy-ai-powered-autonomous-machines-at-scale-ws-17778-614961957305.html

15. Raspberry Pi 4 Model B Starter Kit. URL: https://thepihut.com/products/raspberry-pi-starter-kit?srsltid=AfmBOoq0pjpduqBcZXvLAiuiaw3I54LidhojWyL_p8JS9UCIvuxccIXY

16. Indoor IP camera. URL: https://www.zipato.com/ru/tovar/ip-camera

17. Ультразвуковий датчик відстані HC-SR04. URL: https://www.rcscomponents.kiev.ua/product/ultrazvukovyi-datchyk-vidstani-hc-sr04_103102.html?srsltid=AfmBOoq_oNq9QfR-915K-65lPFlBzoYUi0WYlY14xbyLI1JHpGLNAaZD

18. DHT22 AM2302 Temperature and Humidity Sensor Module. URL: https://www.amazon.in/AM2302-Temperature-Humidity-Sensor-Module/dp/B0CXMJH34D

19. What is Sensor?. URL: https://www.learnelectronicsindia.com/post/what-is-sensor?srsltid=AfmBOorXDtBre1BrGAJZfg6Df_HoSrNUa980fwGR2A0D0D1IvXmOju43

20. Edge AI-Powered Real-Time DecisionMaking for Autonomous Vehicles in Adverse Weather Conditions. URL: https://arxiv.org/pdf/2503.09638

21. Modern computing: Vision and challenges. URL: https://www.researchgate.net/publication/377242890_Modern_computing_Vision_and_challenges

22. Edge Computing: Everything You Need to Know by Bharat Bhooshan Tyagi. URL: https://www.globallogic.com/wp-content/uploads/2024/04/edge-computing.pdf

23. Edge Computing for Real-Time Decision Making in Autonomous Driving: Review of Challenges, Solutions, and Future Trends. URL:

https://thesai.org/Downloads/Volume15No7/Paper_59-Edge_Computing_for_Real_Time_Decision_Making.pdf
24. Edge Computing for Real-Time Decision Making in Autonomous Driving: Review of Challenges, Solutions, and Future Trends. URL: https://www.semanticscholar.org/paper/Edge-Computing-for-Real-Time-Decision-Making-in-of-Xie-Zhou/ddbf516691f2e63d17284eb9dedee89b6b66b3fd
25. Mobile-Edge Computing. URL: https://portal.etsi.org/portals/0/tbpages/mec/docs/mobile-edge_computing_-_introductory_technical_white_paper_v1%2018-09-14.pdf
26. π-Edge: A Low-Power Edge Computing System for Real-Time Autonomous Driving Services. URL: https://arxiv.org/pdf/1901.04978
27. Edge Computing for Real-Time Decision Making in Autonomous Driving: Review of Challenges, Solutions, and Future Trends. URL: https://thesai.org/Downloads/Volume15No7/Paper_59-Edge_Computing_for_Real_Time_Decision_Making.pdf
28. Introduction to edge computing. URL: https://www.researchgate.net/publication/344218125_Introduction_to_edge_computing
29. Integrating Edge Computing for Real-Time Processing and Predictive Modeling in Autonomous Control Systems Using Sensor Fusion. URL: https://www.researchgate.net/publication/388224819_Integrating_Edge_Computing_for_Real-Time_Processing_and_Predictive_Modeling_in_Autonomous_Control_Systems_Using_Sensor_Fusion
30. Modern Computing: Vision and Challenges. URL: https://www.researchgate.net/publication/377242890_Modern_computing_Vision_and_challenges

ДЕМОНСТРАЦІЙНИЙ МАТЕРІАЛ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-
КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

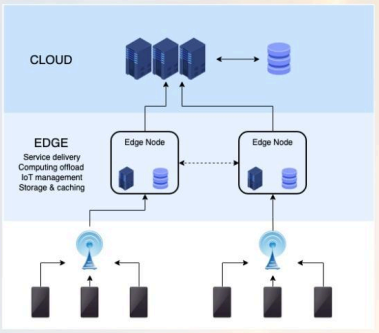
КВАЛІФІКАЦІЙНА РОБОТА

на тему:
«СИСТЕМА З EDGE-ОБЧИСЛЕННЯМИ ДЛЯ АВТОНОМНОГО ПРИЙНЯТТЯ РІШЕНЬ У РЕЖИМІ РЕАЛЬНОГО ЧАСУ»

Виконав: здобувач вищої освіти гр.ІСДМ-61
Денис РЕБРОВ
Керівник: к.т.н., доц.кафедри ІСТ
Ольга ПОЛОНЕВИЧ

Мета роботи

Мета роботи – дослідження методів інтеграції системи з edge-обчисленнями для автономного прийняття рішень у режимі реального часу.
Об'єкт дослідження – система з edge-обчисленнями для автономного прийняття рішень у режимі реального часу.
Предмет дослідження – предметом дослідження є методи, моделі та архітектурні рішення edge-обчислень, що забезпечують автономне прийняття рішень у режимі реального часу.



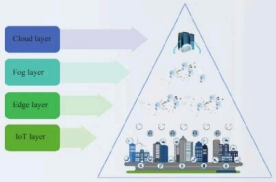
Різновиди обчислень

Хмарні обчислення стали важливим етапом в еволюції обчислювальних технологій, дозволяючи зберігати і обробляти дані через віддалені сервери в Інтернеті.

Автономні обчислення включають як замкнуті, так і розімкнуті цикли керування, де складні системи можуть мати декілька окремих мереж для управління.

Туманні обчислення, є вдосконаленою моделлю, яка широко використовує мобільні пристрої для зберігання та обробки даних, зберігаючи при цьому залежність від Інтернету для з'єднання між вузлами.

Edge-computing



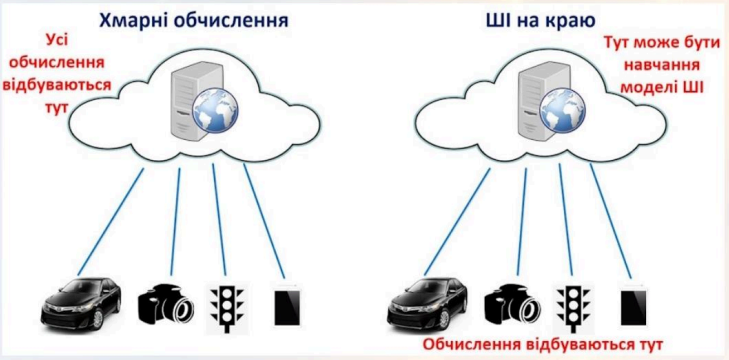
Периферійні обчислення означають обробку даних безпосередньо на пристроях, що знаходяться на межі мережі. Це дозволяє значно зменшити затримки, знизити витрати на передачу даних і покращити якість обслуговування.

Сфери застосування

Розумні системи охоплюють низку важливих сфер, зокрема розумні будинки, розумні енергомережі, розумні міста, розумне сільське господарство, розумний транспорт, розумну медицину та інші напрямки.

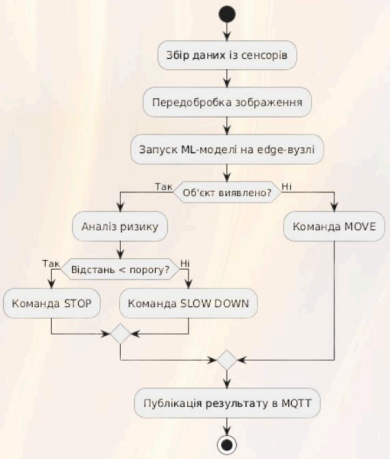
Edge-AI

Edge AI - це підхід, за якого обробка даних і виконання алгоритмів штучного інтелекту відбуваються безпосередньо на пристроях, а не в хмарі.Навчання моделей зазвичай виконується на потужних платформах або в хмарі, після чого готові моделі розгортаються на периферійних пристроях для обробки даних у реальному часі. Edge AI є критично важливим для систем, де навіть мінімальна затримка може призвести до катастрофічних наслідків, зокрема в автономному транспорті, системах пожежної безпеки та моніторингу природних загроз.



Запропонована методика

Система складається з сенсорів, edge-вузла та хмарного рівня. Дані з датчиків надходять на edge-пристрій, де проходять попередню обробку, аналізуються локальною ML-моделлю та використовуються механізмом прийняття рішень. Хмара застосовується лише для моніторингу, логування та оновлення моделей.



Код системи

Система написана мовою програмування Python.

Фрагмент коду зі збору даних з камери.

```
import cv2

def capture_frame():
    cap = cv2.VideoCapture(0)
    ret, frame = cap.read()
    if ret:
        return frame
    return None
```

Робота модулю прийняття рішень.

```
def decision_engine(prediction, distance):
    if prediction[0] > 0.9 and distance < 30:
        return "STOP"
    elif prediction[0] > 0.5:
```

Запуск TFLite-моделі.

```
import numpy as np
import tflite_runtime.interpreter as tflite

interpreter = tflite.Interpreter(model_path="model.tflite")
interpreter.allocate_tensors()

input_details = interpreter.get_input_details()
output_details = interpreter.get_output_details()

def run_inference(image):
    img = cv2.resize(image, (224, 224))
    img = np.expand_dims(img, axis=0).astype(np.float32)

    interpreter.set_tensor(input_details[0]['index'], img)
    interpreter.invoke()
    output = interpreter.get_tensor(output_details[0]['index'])
```

Проблема яку ця система вирішує

У традиційних IoT-системах: дані передаються в хмару; рішення приймаються віддалено; виникають затримки,є залежність від інтернету та ризики втрати зв'язку.
Ця система усуває ці проблеми, оскільки: всі критичні обчислення виконуються локально; рішення формуються без затримок на передачу даних; система продовжує працювати навіть без мережевого з'єднання.

Результати опрацювання системою усіх її компонентів із edge-обчисленнями.

Етап	Середній час
Захоплення кадру	12-18 мс
Попередня обробка	5-7 мс
Інференс TFLite	25-35 мс
Прийняття рішення	<1 мс
Сумарно	45-60 мс

Система дозволяє:

- реагувати на події за 45–60 мс, що відповідає вимогам real-time;
- приймати рішення автоматично, без участі людини; забезпечувати безпечну роботу об'єкта, реагуючи на перешкоди або зміни середовища;
- зменшити навантаження на мережу та хмару;
- підвищити надійність і відмовостійкість всієї системи.

Її головне призначення - забезпечити швидку, стабільну та незалежну від хмари реакцію системи на події в фізичному середовищі.

Подальший розвиток запропонованої структури

Для вирішення проблем навігації автономних транспортних засобів за несприятливих погодних умов пропонується нова система прийняття рішень у режимі реального часу на базі Edge AI.

Експериментальні параметри.

Параметр	Значення
Структура моделювання	CARLA 0.9.14
Набір даних світу	Waymo Open Dataset
Пристрій edge-вузол на базі штучного інтелекту	NVIDIA Jetson Xavier AGX
Хмарна платформа	NVIDIA DGX Cloud
Метрики оцінювання	Точність виявлення, затримка виведення, коефіцієнт зіткнень, коефіцієнт виходу зі смуги руху.

Архітектура складається з трьох основних модулів:

1. Сприйняття та об'єднання даних з датчиків - об'єднує дані з лідарного сканера, радара та камер для створення надійної моделі навколишнього середовища.
2. Прийняття рішень за допомогою EdgeAI - обробляє дані в режимі реального часу локально для класифікації об'єктів та прогнозування сценаріїв руху.
3. Адаптивна система керування - використовує алгоритми RL для оптимізації реакцій керування автомобілем.

Формули за якими проводився розрахунок результатів

Враховуючи обмеження окремих датчиків за несприятливих погодних умов, для інтеграції даних з LiDAR, радара та камери реалізовано байєсівську модель об'єднання датчиків. Процес об'єднання математично визначається як:

$$P(S \mid O) = P(O \mid S) \cdot \frac{P(S)}{P(O)}$$

$P(S \mid O)$ - ймовірність стану S за даними спостережуваних даних датчика O.

$P(O \mid S)$ - представляє функцію правдоподібності спостережень датчика.

$P(S)$ - априорна ймовірність стану.

$P(O)$ - гранична ймовірність спостережень.

Надійність кожного датчика за різних погодних умов динамічно зважується за допомогою:

$$w_i = \frac{1}{\sigma_i^2}$$

w_i - вага датчика.

σ_i - дисперсія його шуму вимірювання.

Для інтеграції даних датчиків я використовую фільтр Калмана для лінійного об'єднання та розширений фільтр Калмана (EKF) для нелінійного об'єднання, що визначається наступним чином:

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k (z_k - H \hat{x}_{k|k-1})$$

$\hat{x}_{kk}$ - оцінений стан у момент часу k.

z_k - вимірювання з датчика

H - матриця спостереження.

K_k - коефіцієнт підсилення Калмана:

$$K_k = P_{k|k-1} H^T (H P_{k|k-1} H^T + R)^{-1}$$

$P_{k-1}{}^k$ - коваріаційна матриця прогнозування станів.

R - коваріаційна матриця шуму датчика.

Точність виявлення об'єктів за різних погодних умов

Вища точність виявлення об'єктів за несприятливих погодних умов.

Модель Edge AI досягла до 7,9% вищої точності виявлення в тумані, дощі та снігу порівняно з хмарним штучним інтелектом.

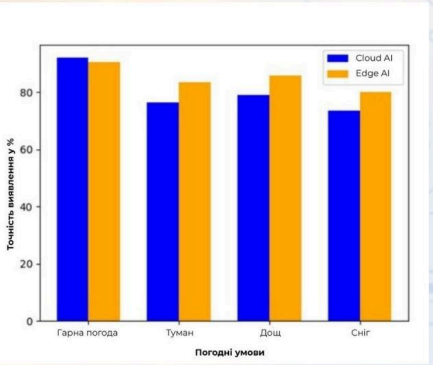
Це покращення пояснюється адаптивним механізмом об'єднання датчиків, який динамічно налаштовує зважування датчиків залежно від умов навколишнього середовища.

Висновок: Edge AI дозволяє коригувати сприйняття в режимі реального часу, роблячи автономні транспортні засоби більш стійкими та надійними в складних погодних умовах.

Формула вимірювання: $\text{IoU} = \frac{Area\ of\ Overlap}{Area\ of\ Union}$

Точність виявлення об'єктів за різних погодних умов.

Модель	Гарна погода	Туман	Дощ	Сніг
Cloud AI	93 %	75.7 %	79 %	73.5 %
Edge AI	91 %	83.6 %	86 %	80.5 %



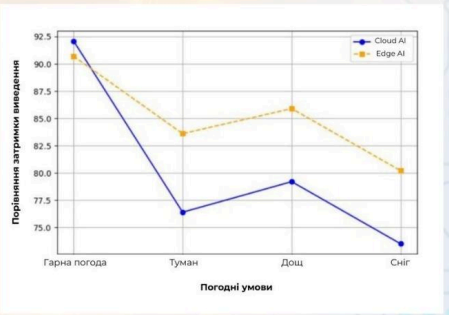
Порівняння затримки виведення

Значне зменшення затримки виводу.
ШІ на периферії скоротив час логічного висновку приблизно на 80% (з 240 мс до 45 мс за ясної погоди та з 310 мс до 55 мс за снігу) порівняно з хмарною обробкою.
Зменшення затримки було досягнуто завдяки: виконання ШІ на пристрої з використанням оптимізованих моделей глибокого навчання, прискорення TensorRT та скорочення моделей для мінімізації обчислювальних витрат.
Наслідок: Зменшення затримки дозволяє швидше реагувати, покращуючи безпеку автономних транспортних засобів та запобігаючи нещасним випадкам у реальних умовах.

Формула вимірювання: $\text{Collision Rate} = \frac{Total\ Collisions}{Total\ Navigation\ Runs} \times 100$

Порівняння затримки виведення.

Модель	Гарна погода	Туман	Дощ	Сніг
Cloud AI	240 мс	270 мс	290 мс	310 мс
Edge AI	45 мс	50 мс	52 мс	55 мс

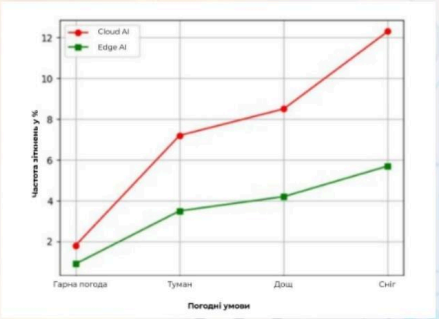


Порівняння коефіцієнтів зіткнень

Нижчий рівень зіткнень та покращена стійкість руху.
Запропонована система знизила рівень зіткнень на 50-60% за різних погодних умов порівняно з хмарним штучним інтелектом.
Модель прийняття рішень на основі RL вивчила адаптивні стратегії водіння, зменшуючи кількість випадків відмов автономних транспортних засобів у снігових умовах та тумані.
Наслідок: Це підтверджує придатність Edge AI для реального розгортання автономних транспортних засобів, особливо в критично важливих для безпеки застосуваннях.

Формула вимірювання: $R_i = \Sigma^t_{i=1}\ \gamma^i r_i$

Модель	Гарна погода	Туман	Дощ	Сніг
Cloud AI	1.8 %	7.2 %	8.5 %	12.3 %
Edge AI	0.9 %	3.5 %	4.2 %	5.7 %

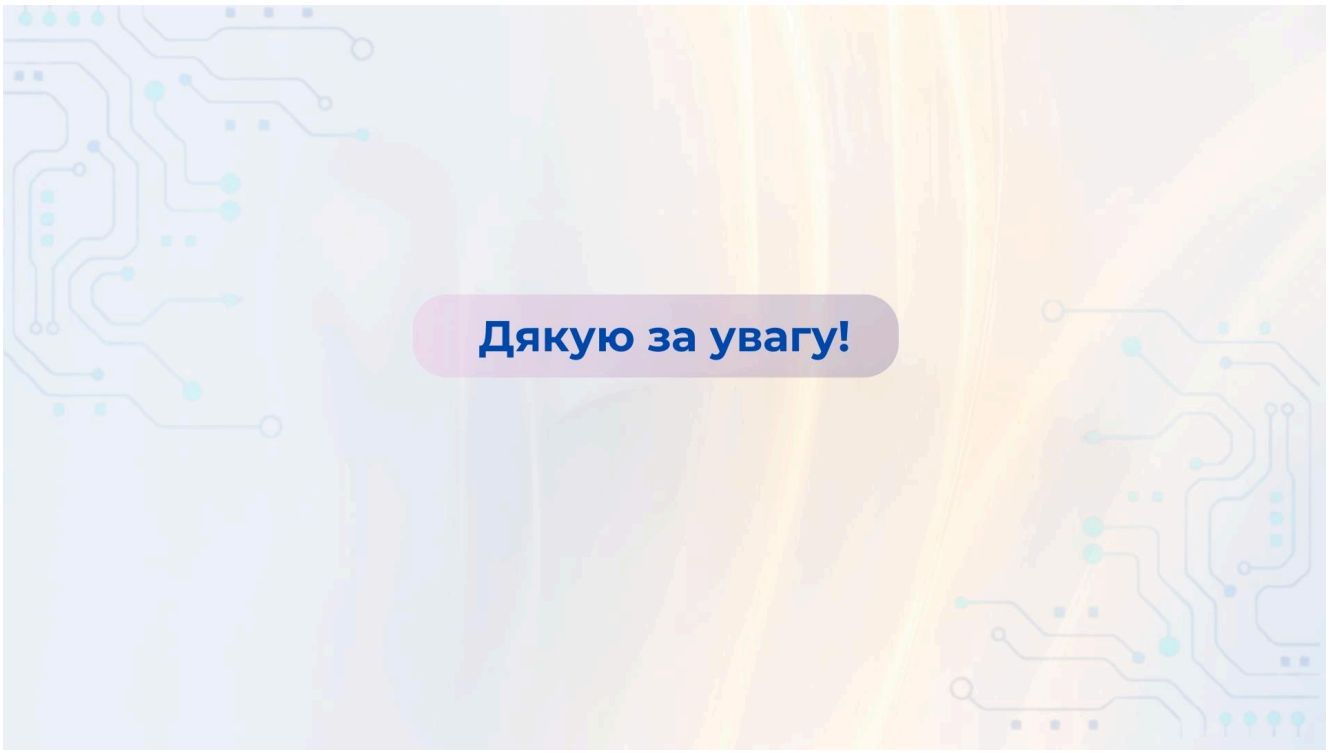


Порівняння з існуючими методами

Аспект	Хмарний ШІ	Edge-обчислення на базі ШІ (запропонована модель)
Затримка виведення	240-310 мс	45-55 мс (у 5 разів швидше)
Точність виявлення об'єктів (туман)	76.4 %	83.6 %
Коефіцієнт зіткнень (сніг)	12.3%	5.7 % (знижена на 50%)
Адаптивність у режимі реального часу	Низька (залежно від мережі)	Висока (повністю автономна)
Масштабованість	Обмежена (потрібне підключення до хмари)	Висока (працює незалежно)

Висновок

1. Запропонований метод, значно покращує та пришвидшує процес прийняття рішень, в системі, у режимі реального часу.
2. Проведені розрахунки показують, що edge-обчислення дозволяють оптимізувати роботу системи та реалізацію процесів, підвищуючи її ефективність на 50%, а також значно знизити показники середньої затримки передачі інформації з 200 мс до 50 мс.
3. Запропонована система за несприятливих погодних умов краще може розпізнавати об'єкти, що значно підвище безпеку використання автономних транспортних засобів.


Дякую за увагу!