Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«КОМП'ЮТЕРНА СИСТЕМА МОНІТОРИНГУ МІСЦЬ ПАРКУВАННЯ АВТОМОБІЛІВ НА ОСНОВІ НЕЙРОННОЇ МЕРЕЖІ»

Виконав: студент 2 курсу, групи 2КІ-24м
спеціальності 123 – Комп'ютерна інженерія
(шифр і назва напряму підготовки, спеціальності)
_____ Лучко Є.М
(прізвище та ініціали)

Керівник: д.т.н., проф. каф. ОТ
_____ Мартинюк Т.Б.
(прізвище та ініціали)
«12» 12 2025 р.

Опонент: к.т.н., доц. каф. ПЗ
_____ Черноволик Г.О.
(прізвище та ініціали)
«12» 12 2025 р.

Допущено до захисту
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«15» 12 2025 р.

_____

ВНТУ 2025

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітньо-кваліфікаційний рівень магістр
Галузь знань – 12 «Інформаційні технології»
Спеціальність – 123 «Комп'ютерна інженерія»
Освітньо-професійна програма «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ
д.т.н., проф. Азаров О.Д.
«25» вересня 2025 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Лучку Євгену Миколайовичу

1 Тема роботи: «Комп'ютерна система моніторингу місць паркування автомобілів на основі нейронної мережі», керівник роботи Мартинюк Т. Б. д.т.н., проф. кафедри ОТ затверджено наказом вищого навчального закладу від «24» вересня 2025 року №313.

2 Строк подання студентом роботи 4.12.2025 р.

3 Вихідні дані до роботи: відеофрагмент роздільною здатністю 1920 на 1080 пікселів, бібліотека Ultralytics, середовище розробки Visual Studio Code.

4 Зміст розрахунково-пояснювальної записки: вступ, аналіз методів моніторингу місць паркування, розробка методу моніторингу місць паркування, розробка засобів моніторингу місць паркування, тестування та оцінка якості роботи програми, розрахунок економічної доцільності створення програми моніторингу місць паркування, висновки, додатки.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): типи сенсорів систем моніторингу місць паркування автомобілів,

типи згорткових нейронних мереж, послідовність роботи системи моніторингу місць паркування автомобілів.

6 Консультанти розділів роботи

Таблиця 1 — Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	д.т.н., проф. каф. ОТ Мартинюк Тетяна Борисівна		
5	к.т.н., доц. кафедри ЕПВМ Адлер Оксана Олександрівна		
Нормоконтроль	асистент кафедри ОТ Швець Сергій Ілліч		

7 Дата видачі завдання 25.09.2025 р.

8 Календарний план наведено в таблиці 2.

Таблиця 2 — Календарний план

№	Назва та зміст етапу	Срок виконання	Примітка
1.	Постановка задачі	26.09.2025	вик.
2.	Аналіз методів та проблематики моніторингу місць паркування	08.10.2025	вик.
3.	Розробка моделі та алгоритму системи моніторингу місць паркування	17.10.2025	вик.
4.	Програмна реалізація системи моніторингу місць паркування	28.10.2025	вик
5.	Тестування та перевірка якості роботи системи моніторингу	01.11.2025	вик.
6.	Розрахунок економічної доцільності розробки	05.11.2025	вик.
7.	Оформлення пояснювальної записки та графічної частини	10.11.2025	вик
8.	Попередній захист	11.11.2025	вик.
9.	Перевірка якості виконання роботи та усунення недоліків	03.12.2025	вик.

Студент _____ Євген ЛУЧКО

Керівник роботи _____ Тетяна МАРТИНЮК

АНОТАЦІЯ

УДК 004.93

Лучко Є.М. Комп’ютерна система моніторингу місць паркування автомобілів на основі нейронної мережі. Магістерська кваліфікаційна робота зі спеціальності 123 — комп’ютерна інженерія, освітня програма комп’ютерна інженерія. Вінниця: ВНТУ, 2025. 119 с.

На укр. мові. Бібліогр.: 32 назв; рис. 19; табл. 8.

В магістерській кваліфікаційній роботі проведено аналіз методів розпізнавання об’єктів із застосуванням нейронної мережі. У магістерській роботі запропоновано здійснювати процес моніторингу місць паркування автомобілів із використанням методу орієнтованих обернених рамок, сформовано алгоритм навчання нейронної мережі та визначено основні критерії оцінки якості роботи системи моніторингу, сформовано склад комп’ютерної системи моніторингу місць паркування автомобілів та розроблено програмний засіб для реалізації запропонованого підходу для моніторингу місць паркування автомобілів. Проведено тестування роботи для визначення надійності та продуктивності створеної системи. Проведено економічні розрахунки із обґрунтування доцільності виконання роботи по розробці системи моніторингу місць паркування, обраховано економічну доцільність та ефективність впровадження.

Ключові слова: розпізнавання, згорткова нейронна мережа, відеопотік, камера відеоспостереження.

ABSTRACT

Luchko Y.M Computer system for monitoring parking spaces based on neural networks. Master’s thesis in the specialty 123 – computer engineering, educational program computer engineering. Vinnytsia: VNTU, 2025. 119 p.

In Ukrainian. Bibliographer: 32 titles; figs. 19; table 8.

The master's thesis analyzes methods of object recognition using neural networks. The thesis proposes monitoring parking spaces using the oriented inverse frame method, an algorithm for training a neural network is formed, and the main criteria for evaluating the quality of the monitoring system are determined. The composition of the computer system for monitoring parking spaces is formed, and a software tool is developed to implement the proposed approach for monitoring parking spaces. The operation was tested to determine the reliability and performance of the created system. Economic calculations were performed to justify the feasibility of developing a parking space monitoring system, and the economic feasibility and effectiveness of its implementation were calculated.

Keywords: recognition, convolutional neural network, video stream.

ЗМІСТ

ВСТУП

Зростання урбанізації та збільшення кількості транспортних засобів у містах створює значні проблеми з пошуком вільних місць паркування, що призводить до транспортних заторів, перевитрати пального та погіршення екологічної ситуації [1]. Традиційні методи моніторингу місць паркування часто є неефективними, вимагають значних людських ресурсів або використання дорогого апаратного забезпечення, наприклад, вбудованих сенсорів у дорожнє покриття, які складно обслуговувати [2].

Сучасні рішення на основі комп’ютерного зору та методів машинного навчання дозволяють використовувати наявну інфраструктуру камер відеоспостереження для оптимізації міського трафіку та підвищення комфорту містян. Така технологія використовує згорткові нейронні мережі для автоматичного виявлення автомобілів та класифікації стану місця паркування (зайнято/вільно) на зображеннях, отриманих з камер [2]. На відміну від систем на основі датчиків, цей підхід забезпечує гнучкість в масштабуванні та простоту інтеграції. Проте, розробка надійних систем комп’ютерного зору вимагає подолання технічних викликів, зокрема, забезпечення стабільної роботи в умовах часткового затемнення або перекриття об'єктів [3].

Особливо актуальною технологія стає для України в контексті значного зростання кількості автомобілів в містах та розбудови сучасної міської інфраструктури. Така система моніторингу місць паркування дозволить муніципалітетам та підприємствам більш оперативно керувати місцями паркування, знижуючи операційні витрати на контроль та збільшуючи ефективність використання місць паркування, і надавати водіям точну інформацію про наявність вільних місць у режимі реального часу. Це, в свою чергу, дозволяє ефективніше використовувати місця паркування, що робить подальше дослідження та розробку таких систем **актуальним**.

Магістерська робота виконана по кафедральній науково-дослідній тематиці студентського наукового гуртка кафедри ОТ «Розробка комп’ютерної системи

пошуку і розпізнавання об’єктів», керівником якого є старший викладач кафедри Очкуров М. А.

Метою дослідження є підвищення точності системи моніторингу місць паркування автомобілів.

Задачі дослідження:

— провести аналіз методів моніторингу місць паркування автомобілів;

— спроектувати систему моніторингу місць паркування автомобілів;

— виконати навчання нейронної мережі;

— розробити програмний засіб системи моніторингу місць паркування автомобілів;

— провести тестування системи моніторингу в різних ситуаціях;

— здійснити обґрунтування доцільності створення нового програмного засобу.

Об'єктом дослідження є процес розпізнавання та оброблення відеопотоку з метою визначення стану зайнятості місць паркування.

Предметом дослідження є методи та алгоритми побудови й навчання згорткових нейронних мереж для визначення наявності транспортних засобів на окремих паркувальних місцях на основі даних відеоспостереження.

У роботі використані такі **методи дослідження**: методи математичного моделювання та формалізації застосовувалися для опису процесу розпізнавання зайнятості місць паркування, методи статистичного аналізу застосовувалися для кількісної оцінки ефективності розробленого методу, методи об’єктно-орієнтованого програмування.

Наукова новизна роботи полягає в вдосконаленні методу розпізнавання об’єктів для задачі моніторингу місць паркування шляхом застосування орієнтованих обмежувальних рамок, що дозволяє підвищити точність визначення стану місць паркування.

Практичне значення одержаних результатів:

— сформовано алгоритм визначення стану місця паркування автомобілів;

— створено систему обробки відео для моніторингу місць паркування автомобілів;

— розроблено програмний засіб системи моніторингу місць паркування автомобілів.

Апробація результатів магістерської роботи: зроблено доповідь на LV Всеукраїнській науково-технічній конференції підрозділів ВНТУ (НТКП ВНТУ, Вінниця, 2025 р.).

За результатами магістерської роботи опубліковано 1 тезу доповіді : Комп’ютерна система моніторингу місць паркування автомобілів на основі нейронної мережі / Є. М. Лучко, Т. Б. Мартинюк, М. А. Очкуров // Матеріали LV Всеукраїнської науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії. Вінниця 2025 р. 2 с. [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2026/paper/view/26651/22002 [4].

1 АНАЛІЗ МЕТОДІВ МОНТІОРИНГУ МІСЦЬ ПАРКУВАННЯ

1.1. Аналіз проблеми пошуку та моніторингу місць паркування

Проблема паркування в містах є багатогранною і не зводиться лише до банального дефіциту площі. Історично, інфраструктура міст, особливо забудова епохи модернізму, не була розрахована на таку кількість автомобілів, яка спостерігається сьогодні. Ця спадщина призвела до постійного дефіциту місць паркування, особливо в житлових кварталах та центральних частинах, де відсутні підземні чи багаторівневі паркінги.

Окрім того, ця проблема часто ускладнюється застарілим міським плануванням. Багато міст дотримуються застарілих положень про будівельні норми, які не враховують значне зростання кількості автомобілів в містах. Це призводить то того, що водії починають займати цінний міський простір на паркування авто. Це створює критичний цикл зворотного зв'язку: застарілі норми призводять до неефективного використання землі в центральних і бізнес центрах та збільшення заторів, що, в свою чергу, підсилює сприйняття необхідності збільшення кількості паркінгів, продовжуючи цикл.

Цей феномен є частиною ширшої урбаністичної проблеми, що демонструється досвідом міст США 1970–1980-х років. У цих містах формувалася інфраструктура, орієнтована на автотранспорт, з розгалуженими мережами фрівеїв, магістралей та величезними комерційними зонами вздовж них. Незважаючи на значні інвестиції у великі паркінги споруди та сотні миль доріг, затори залишилися серйозною проблемою. Залежність від автомобілів призвела до занепаду громадського транспорту та деградації пішохідного руху, що у підсумку не вирішило проблему, а посилило її. Така модель міського розвитку також сприяла низькій щільності забудови, що ще більше збільшувало залежність від приватних автомобілів. Таким чином, ефективні системи моніторингу та управління паркуванням є не просто технологічним рішенням, а критично важливим інструментом для подолання наслідків хибних урбаністичних стратегій та переходу до більш збалансованого та сталого розвитку міст [2].

З точки зору бізнесу та управління, проблеми з паркуванням створюють значне фінансове навантаження. Підприємства, що надають послуги паркування, стикаються з втратою доходів через неефективне використання простору, що часто є наслідком недостатньої прозорості даних. Вони несуть високі операційні витрати, пов'язані з охороною, технічним обслуговуванням та ручною працею, що може призвести до зниження рентабельності. На макрорівні затори на дорогах, спричинені пошуком місць для паркування, є значним гальмом для розвитку економіки в цілому. Основна економічна проблема для операторів паркінгів полягає в складності впровадження динамічної моделі ціноутворення. Не маючи даних про попит у режимі реального часу, оператори змушені покладатися на фіксовані тарифи, які або занадто високі, що змушує клієнтів шукати альтернативні варіанти та паркуватись абиде, або занадто низькі, що призводить до низького обороту та втрачених можливостей для отримання доходу. Ця неефективність ціноутворення та управління призводить до того, що частина коштовного активу постійно залишається недовикористаною.

Довготривалий пошук місця паркування є не просто марною тратою часу. Дослідження показують, що до 25% транспортного потоку в центральних частинах великих міст можуть становити автомобілі, що перебувають у пошуку місця паркування. Це створює додаткове навантаження на вулично-дорожню мережу та знижує її ефективність. В інших аналізах зазначено, що пошук паркування відповідає за 30% усіх заторів у містах. Впровадження "розумних" систем, що спрямовують водіїв безпосередньо до вільного місця, може зменшити викиди CO_2 на 40% , а також знизити загальний рівень шумового забруднення, що суттєво покращує якість життя в місті [3].

Окрім цього, пошук місця для паркування є джерелом значного і вимірюваного стресу. Водії часто повідомляють про глибоке розчарування через оманливі ціни, несподівані штрафи, несправне обладнання та загальну відсутність чіткої інформації. Це розчарування не є лише епізодичним; воно проявляється у вигляді «паркувальної люті» та вимірюваних фізіологічних і поведінкових змінах. Дослідження з використанням автомобілів, оснащених спеціальними датчиками,

показало, що водії, які активно шукають місце для паркування на вулиці, знижують швидкість, їдуть ближче до бордюру та вказують на підвищений рівеннь напргуги. Було доведено, що розумові вимоги що до навігації в складному середовищі та постійний соціальний тиск з боку інших автомобілістів, що їдуть позаду, підвищують частоту серцебиття та рівень стресу, що в деяких дослідженнях наближається до статистичної значущих.

Відволікання уваги та розумове навантаження під час пошуку місця для паркування мають прямий і вимірюваний вплив на безпеку. Процес пошуку місця часто вимагає від водіїв відволікання уваги від дороги, що пов'язано з ризиком ДТП, який у 7,1 рази вищий, ніж під час звичайного водіння. Коли нарешті знаходять вільне місце, терміновість його зайняти може призвести до непередбачуваної поведінки водія, що ще більше підвищує ризик зіткнення. Для комерційних водіїв ця проблема ускладнюється серйозною нестачею вільних місць. Американська асоціація вантажоперевізників (АТА) повідомляє, що час, витрачений на пошук місця паркування, коштує середньому водієві вантажівки майже 7000 доларів на рік у вигляді втраченої продуктивності [4].

Задля пом’якшення зазначених раніше проблем та покращення рівня використання існуючих місця паркування необхідно проводити контроль простору для паркування.

Ручний контроль є найстарішим та найпростішим методом управління паркуванням. Його принцип полягає у безпосередній взаємодії людини з водієм, але цей підхід має значні обмеження у сучасних умовах.

На паркінгах з ручним контролем усі процеси в’їзду та виїзду автомобілів здійснюються під наглядом персоналу, який може бути представлений охоронцями або операторами. Ці співробітники виконують низку завдань: вони перевіряють документи, видають паркувальні талони, приймають оплату та ведуть журнал відвідувань. Окрім цього, персонал відповідає за безпеку та підтримання порядку на території.

Незважаючи на простоту, ручний контроль є найменш ефективним методом для масштабного впровадження та довгострокового функціонування. Ключовий

недолік — це високі операційні витрати, пов'язані з необхідністю постійної присутності персоналу, що працює позмінно, цілодобово. Це призводить до значних витрат на заробітну плату, навіть якщо вона, як правило, значно нижча за середню.

Ще одним критичним обмеженням є людський фактор. Він може призвести до неточностей в обліку в'їздів та виїздів, а також створити можливості для зловживань, що безпосередньо впливає на прибутковість паркінгу. У місцях з високою інтенсивністю руху, таких як великі торговельні центри або бізнес-центри, ручне обслуговування може спричинити значні затримки та утворення черг, що є вкрай незручним для водіїв. Враховуючи тенденції до автоматизації, ця професія має високий ризик бути витісненою технологіями, що підтверджується даними про високий рівень впливу автоматизації на цю сферу. Таким чином, ручний контроль є архаїчним методом, який не забезпечує прозору звітність, точність та ефективність, що є основними вимогами до сучасних систем управління паркуванням.

Як наслідок системи інтелектуального керування паркуванням значно зростають в популярності як ефективний та масштабований засіб керування.

З моменту свого винаходу в 1960-х роках детектор на основі індукційної петлі став стандартом в організації дорожнього руху. Система працює за принципом електромагнітної індукції. Петля з ізольованого дроту закопується в неглибоку борозду в дорожньому покритті. Індуктивна петля – це котушка, що збуджується осцилятором, створюючи в своїй області магнітне поле, яке резонує з постійною частотою (зазвичай від 10 кГц до 200 кГц) і контролюється електронним детектором. Базова частота встановлюється, коли над петлею немає транспортного засобу. Коли транспортний засіб з масивним залізним двигуном проїжджає або зупиняється в петлі, він викликає вихрові струми, що зменшують індуктивність петлі. Електронний блок фіксує цю зміну частоти і сигналізує про наявність транспортного засобу. Технологія проста, економічно ефективна і забезпечує точне виявлення наявності транспортного засобу.

Принцип роботи такого детектора представлено на рисунку 1.1

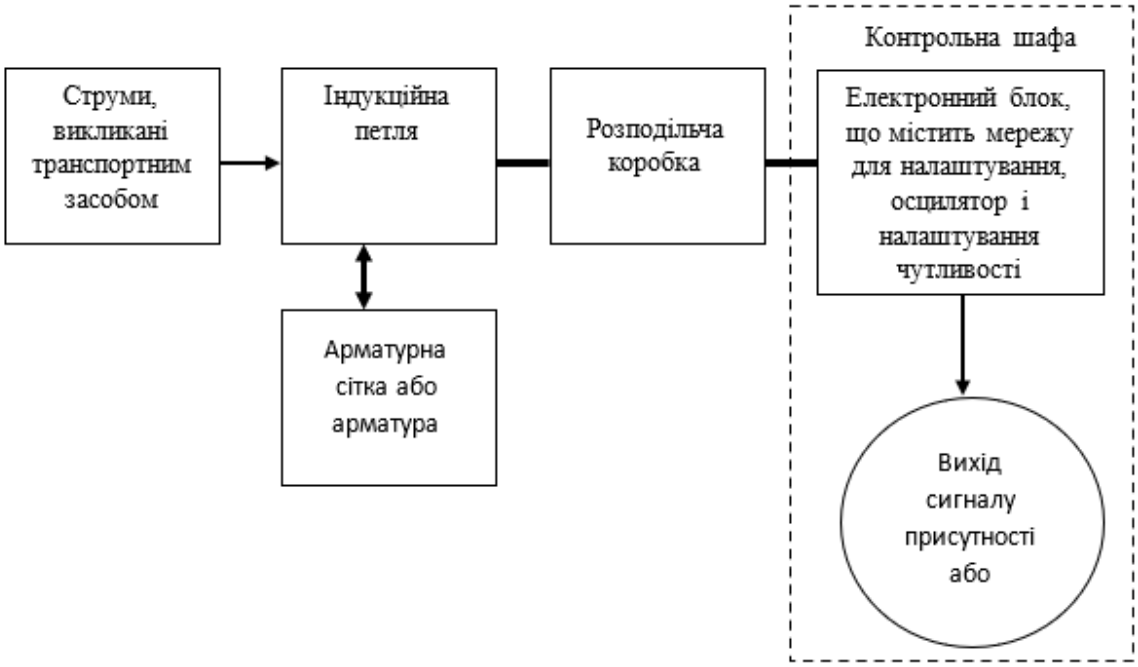


Рисунок 1.1 — Схема роботи індукційного детектора

Незважаючи на свою поширеність, індукційні петлі мають значні недоліки. Їх установка є дорогою, трудомісткою і сильно порушує рух транспорту, оскільки вимагає розрізання дорожнього покриття і закриття смуг руху. До того вони також дуже схильні до поломок через сезонні цикли замерзання/відтавання, проникнення води або пошкодження при оновленні дорожнього покриття. Ця вразливість призводить до короткого терміну експлуатації, зазвичай від 3 до 7 років, і вимагає частого спеціалізованого технічного обслуговування, що може ще більше порушити використання дорожнього простору. Важливо зазначити, що індукційні петлі є датчиками присутності; вони можуть сигналізувати про наявність транспортного засобу, але не можуть класифікувати тип транспортного засобу або виявляти неметалевих учасників дорожнього руху, таких як пішоходи або велосипедисти [5].

Ультразвукові датчики широко використовуються в системах паркування автомобілів як для побутових, так і для комерційних цілей. Ця технологія працює за принципом реєстрації відбиття, коли датчик випромінює високочастотні звукові хвилі і вимірює час, необхідний для повернення відбитого від сусіднього об'єкта сигналу. Потім це вимірювання перетворюється на відстань, що дозволяє системі

визначити, чи зайняте місце. Датчики для комерційного застосування зазвичай працюють на частотах близько 40-52 кГц і можуть мати діапазон виявлення до 550 см. Вони часто інтегровані з багатоколірними світлодіодними індикаторами для візуального орієнтування.

Ультразвукові датчики є недорогим варіантом, який зазвичай легко встановлювати і на який не впливає колір або прозорість транспортного засобу. Однак на їхню роботу можуть впливати різні фактори навколишнього середовища, включаючи бруд, сніг, лід або навіть м'які поверхні об'єктів. Зміна швидкості ультразвуку залежно від температури повітря впливає на інтервал виявлення ультразвукового датчика, а кут між датчиком і об'єктом виявлення впливає на його положення при встановленні. Вони також мають обмежене поле зору в порівнянні з альтернативними технологіями, такими як радар. Фізичні пошкодження від невеликих ударів або каменів можуть призвести до несправності, а потрапляння води може спричинити коротке замикання. Хоча вони підходять для надання навігації в окремому просторі, їхні вроджені вразливості роблять їх менш надійним рішенням для великих систем, критично важливих для виконання завдань, де необхідна безперервна робота в будь-яких погодних умовах [6].

Магнітні датчики (зокрема геомагнітні датчики або магнітометри) використовуються в системах контролю паркування для виявлення наявності або відсутності автомобіля на місці паркування шляхом вимірювання змін магнітного поля Землі. Коли місце паркування порожнє, датчик вимірює магнітне поле Землі в цьому конкретному місці, що становить базове значення. Коли великий залізний предмет, наприклад автомобіль, паркується над датчиком, металева маса спотворює природне магнітне поле Землі. Датчик виявляє значну зміну в магнітному полі і передає сигнал, що вказує на те, що місце зайняте. Коли автомобіль від'їжджає, магнітне поле повертається до базового значення, і датчик сигналізує, що місце вільне.

Такі датчики мають низьке енергоспоживання та часто встановлюються врівень з поверхнею дороги (у вигляді шайби) або під дорожнім покриттям. Це значно спрощує їх установку, робить її простішою та менш витратною, ніж у

випадку з індукційними петлями, оскільки не вимагає прокладання великих траншей або перекриття смуг руху. Оскільки вони працюють від батареї і встановлюються під землею, такі датчики, як правило, не вимагають особливого обслуговування, а батареї служать роками (від 3 до 10 років, залежно від технології). На відміну від ультразвукових датчиків, вони також відносно стійкі до впливу навколишнього середовища, такого як сніг, дощ і бруд.

Недоліками можна визначити не можливість розрізняти типи транспортних засобів як легковий автомобіль або велика вантажівка, що створюють зміну магнітного поля, але датчик повідомляє лише про «зайнятість», вразливість до сильних електромагнітних завад (наприклад, поблизу високовольтних ліній) і можуть мати «мертві зони» з автомобілями з високим шасі або певними типами електромобілів з низьким вмістом металу [7].

Застосування камер відеоспостереження стало значним технологічним проривом у сучасних системах управління паркуванням. Камери забезпечують багато можливостей, що виходять за межі простого виявлення транспортних засобів. На відміну від одноцільових датчиків, таких як ультразвукові або магнітні петльові датчики, камери можуть одночасно фіксувати присутність транспортного засобу, тип транспортного засобу, тривалість перебування та навіть порушення правил паркування. Такий широкий спектр даних дозволяє проводити комплексний аналіз паркування та розробляти стратегії управління. Неінвазивний характер встановлення камер є ще однією значною перевагою. На відміну від індуктивних петель, які вимагають проведення земляних робіт, або ультразвукових датчиків, які потребують фізичного кріплення до конструкцій, камери можна встановлювати на існуючих стовпах, будівлях або інших об'єктах інфраструктури. Це мінімізує ризик пошкоджень під час встановлення та спрощує перестановку в разі зміни місця розташування паркінга. Крім того, одна камера може одночасно контролювати кілька місць паркування, що робить її більш економічно вигідним рішенням у порівнянні з окремими датчиками для кожного місця. Це мінімізує проблеми під час встановлення та дозволяє легше змінювати розташування камер у міру зміни планування зони паркування. Крім того, одна камера може одночасно контролювати

кілька місць паркування, що робить це рішення більш економічно вигідним у порівнянні з окремими датчиками для окремих місць.

Фактори навколишнього середовища суттєво впливають на роботу камери. Несприятливі погодні умови, включаючи дощ, сніг, туман і відблиски від прямих сонячних променів, можуть погіршити якість зображення, знизивши точність виявлення. Аналогічно, зміни умов освітлення між днем і ніччю вимагають використання камер з широким динамічним діапазоном або додаткових систем освітлення, що збільшує загальні вимоги до інфраструктури

Датчики LiDAR, що розшифровується як Light Detection and Ranging (виявлення та вимірювання відстані за допомогою світла), працюють за допомогою лазерного випромінювання, вимірюючи відстані та створюючи високоточні 3D-карти навколишнього середовища.

Щоб створити повну картину навколишнього середовища, система LiDAR зазвичай використовує обертовий або скануючий механізм (у механічному LiDAR) або фіксовані напівпровідникові антени (у сучасному LiDAR) для сканування лазерних імпульсів в широкому діапазоні.

Поєднуючи розраховану відстань кожного імпульсу з кутом, під яким він був відправлений, датчик генерує мільйони точок даних, які разом називаються хмарою точок. Ця хмара точок є точним тривимірним зображенням сцени, що дозволяє комп'ютерам точно визначати форму, розмір і положення всіх навколишніх об'єктів.

LiDAR часто встановлюється над головою на світлових стовпах або будівельній інфраструктурі. Один пристрій LiDAR може контролювати велику територію, одночасно керуючи десятками місць паркування, що зменшує вартість і складність встановлення окремого датчика для кожного місця. Він забезпечує вищу геометричну точність порівняно з традиційними датчиками, створюючи точний 3D-профіль автомобіля для надійного виявлення його присутності та зайнятості [8].

1.2. Огляд сучасних підходів до автоматизованого моніторингу місць паркування на основі комп'ютерного зору

Визначальною характеристикою для класичних систем комп'ютерного зору є ручна розробка засобів вилучення ознак та алгоритмів на основі правил. Цей підхід покладається на експертів у відповідній галузі, які визначають, що таке «ознака» — наприклад, край, кут, візерунок текстури або різниця в інтенсивності між двома сусідніми областями — а потім створюють системи для виявлення цих заздалегідь визначених ознак. У цій парадигмі логіка системи явно закодована в серії правил або фільтрів, що відрізняється від сучасного автоматичного навчання ознак. Ця методологія є ефективною в суворо контрольованих, структурованих середовищах, де варіації мінімальні і можна застосовувати єдиний набір правил. Тим не менш технології комп'ютерного зору здійснили революцію в системах інтелектуального управління паркуванням, автоматизувавши такі важливі функції, як виявлення місць паркування, ідентифікація транспортних засобів та оптимізація транспортних потоків. Ці системи використовують передові алгоритми обробки зображень та машинного навчання, щоб перетворити традиційне управління паркуванням на інтелектуальні, керовані даними операції, які підвищують ефективність, зменшують затори та покращують користувацький досвід. Інтеграція комп'ютерного зору дозволяє здійснювати моніторинг доступності місць паркування у режимі реального часу, автоматизувати системи виставлення рахунків та застосовувати динамічні моделі ціноутворення, що оптимізують використання простору.

Класичні методи обробки зображень є основою систем управління паркуванням на основі комп'ютерного зору. Ці методи, хоча в останні роки вони в значній мірі були замінені підходами глибокого навчання, залишаються актуальними для попередньої обробки та виділення ознак у різних ситуаціях. Алгоритми виявлення контурів, такі як оператори Собеля, Кенні та Робертса, відіграють важливу роль у визначенні меж об'єктів та структурних особливостей у паркувальних середовищах. Ці оператори аналізують градієнти яскравості для

виявлення різких розривів, що відповідають краям об'єктів, хоча їхня ефективність значно варіюється залежно від умов навколишнього середовища.

Алгоритм Кенні, який вважається золотим стандартом серед класичних алгоритмів, використовує багатоступеневий процес, що включає зменшення шуму, обчислення градієнта, обмеження не максимальних значень та гістерезисне порогове значення. Його здатність створювати безперервні контури країв робить його особливо цінним для ідентифікації меж місць паркування та контурів транспортних засобів [9]. Приклад роботи алгоритму Кенні представлено на рисунку 1.2.

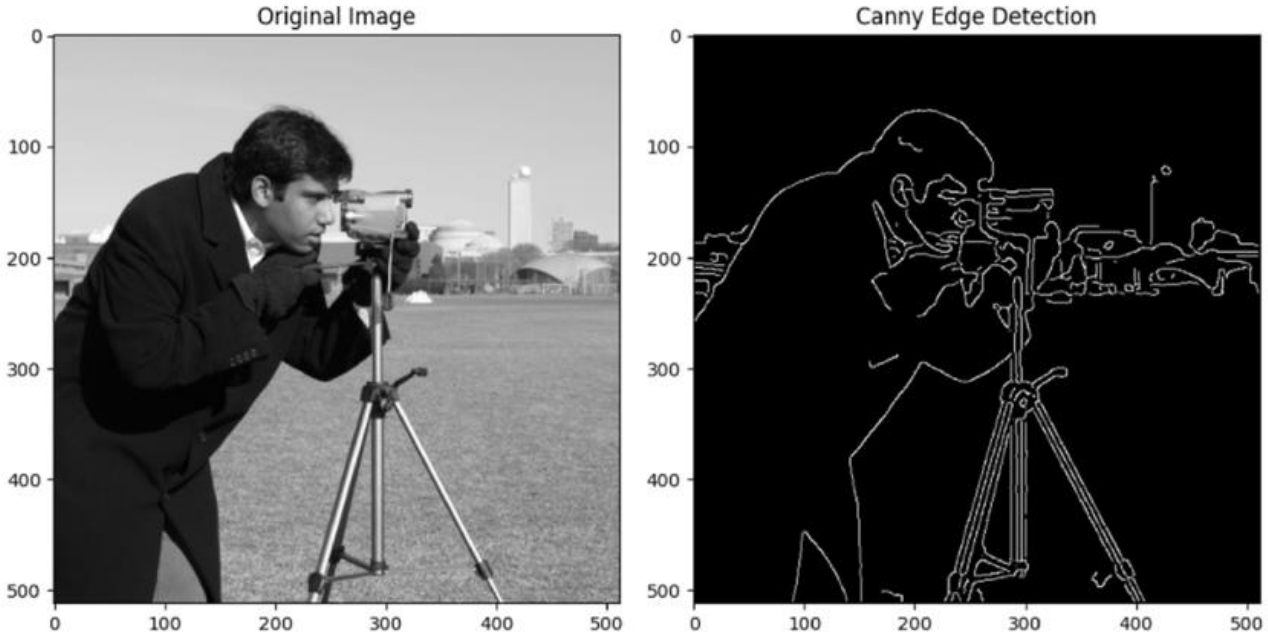


Рисунок 1.2 — Приклад роботи алгоритму Кенні

Морфологічні операції допомагають уточнити сегментовані об'єкти шляхом видалення шуму, заповнення відсутніх даних та відокремлення межуючих об'єктів. Ці операції особливо корисні для попередньої обробки зображень перед застосуванням більш складних алгоритмів або для постобробки результатів моделей глибокого навчання з метою підвищення точності меж. Існують такі морфологічні операції:

— ерозія призводить до звуження об'єктів на зображенні, видаляє невеликі ізольовані шумові пікселі, розриває тонкі перемички та зменшує розмір об'єктів;

— дилація призводить до збільшення об'єктів, згладжує виїмки в контурах і заповнює невеликі прогалини;

— розкриття, що є ерозією, за якою слідує дилація, згладжує контури об'єктів, розриває тонкі перемички та видаляє невеликі виступи на межах об'єктів;

— закриття, що є дилація, за якою слідує ерозія, об'єднує вузькі розриви та довгі тонкі прогалини, а також заповнює невеликі отвори всередині об'єктів.

Порогові та морфологічні операції є одним з фундаментальних компонентів класичних процесів обробки зображень. Методи порогового оброблення перетворюють зображення в сірих тонах у бінарні представлення шляхом класифікації пікселів на основі значень інтенсивності, що дозволяє чітко розрізняти транспортні засоби на передньому плані та дорожнє покриття на задньому плані.

Методи виявлення контурів, зокрема детектор контурів Кенні у поєднанні з фільтром Гауса для зменшення шуму, стали важливими для ідентифікації меж місць паркування та контурів транспортних засобів. Багатоетапний процес алгоритму Кенні — згладжування, обчислення градієнта, обмеження не максимальних значень та порогова обробка з гістерезисом — забезпечив надійне виділення контурів навіть у помірно складних сценах. Після виявлення контурів, перетворення Гафа (Hough Transform) широко використовувалося для виявлення ліній, представляючи розмітку майданчика паркування як параметричні рівняння в полярних координатах. Цей метод виявився особливо ефективним для виявлення прямих ліній, які окреслювали індивідуальні місця паркування.

Методи зіставлення шаблонів (Template matching) порівнювали зображення контурів або специфічні ознаки з попередньо визначеними шаблонами транспортних засобів, хоча вони були обмежені своєю нездатністю обробляти зміни масштабу та обертання. Попри їхню інтерпретованість і прозорість, класичні методи зіткнулися зі значними проблемами обчислювальної складності, вимагаючи послідовного виконання кількох кроків обробки, що перешкоджало продуктивності в реальному часі. Їхня чутливість до умов освітлення — особливо в сценаріях низької освітленості та високого відблиску — ще більше обмежувала їхню ефективність у неконтрольованих середовищах [9]. Залежність від чіткої, добре

визначеної розмітки на місцях паркування являла собою ще одне фундаментальне обмеження, оскільки ці методи мали проблеми, коли розмітка була вицвілою або затемненою. Ці обмеження зрештою каталізували перехід до більш адаптивних підходів машинного навчання, які могли б справлятися зі змінними умовами реального світу паркування.

Алгоритм вододілу є потужною класичною технікою сегментації, яка перетворює зображення на топографічні поверхні, де інтенсивність пікселів відповідає значенням висоти. Цей підхід з'єднує пікселі з подібними просторовими позиціями та значеннями сірого кольору для формування сегментів, створюючи сегментації, які добре відповідають людському сприйняттю областей зображення [10]. У сегментації місць паркування алгоритми вододілу є особливо цінними для відокремлення щільно скупчених або суміжних паркувальних позначок/ліній від бінарної маски сегментації. Однак стандартний алгоритм вододілу страждає від надмірної сегментації, спричиненої шумом, що може бути зменшено за допомогою методів попередньої обробки, таких як морфологічне розділення з використанням ядра 3x3 для 2 ітерацій. Реалізація вододілу на основі маркерів, яка вимагає інтерактивного визначення точок маркерів, що позначають області переднього плану та фону, забезпечує більш контрольовану сегментацію шляхом створення «бар'єрів» між цими маркерами та ефективного об'єднання долин на основі наданих попередніх знань [10].

Гістограма орієнтованих градієнтів (HOG), представлена Далалем та Тріггсом у 2005 році, запропонувала альтернативний підхід, зосереджений на локальному зовнішньому вигляді та формі. HOG фіксувала розподіл градієнта інтенсивності або напрямку контуру, розділяючи зображення на клітинки, групуючи орієнтації градієнта в діапазони кутів та об'єднуючи ці клітинки в блоки для формування векторів ознак фіксованого розміру. Схема, що представляє алгоритм роботи представлена на рисунку 1.3. Такий підхід виявився особливо ефективним для виявлення конкретних об'єктів, як-от транспортні засоби, де контури яких були помітними та одноманітними. HOG забезпечував всебічне покриття областей зображення, хоча й ціною підвищених обчислювальних вимог.

Традиційні конвеєри виявлення об'єктів переважно використовували ковзні вікна (sliding windows) для вибору кандидатних областей.

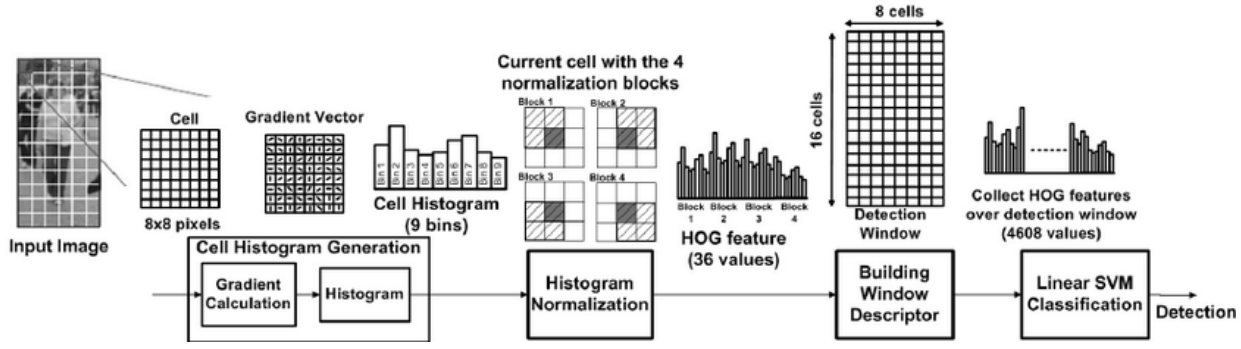


Рисунок 1.3 — Алгоритм роботи гістограми орієнтованих графів

Перехід від класичної обробки зображень до більш досконалих методів виявлення об'єктів став значним кроком вперед у розвитку систем управління паркуванням на основі комп'ютерного зору. У цей період з'явилися два провідні підходи: гістограма орієнтованих градієнтів у поєднанні з методом опорних векторів (SVM) та алгоритм Віоли-Джонса. Обидва підходи стали значним вдосконаленням класичних методів і з'явилися ще до революції у галузі глибокого навчання.

HOG+SVM представляє собою підхід на основі ознак, який поєднує ознаки гістограми орієнтованих градієнтів з лінійним класифікатором. Ця методологія обчислює орієнтації градієнта в локалізованих областях зображення і будує гістограми, що відображають розподіл напрямків градієнта. Потім ці ознаки класифікуються за допомогою SVM, навченого на позитивних і негативних прикладах. Хоча HOG+SVM продемонстрував вражаючу ефективність в контрольованих середовищах, його обмеження стають очевидними в практичних сценаріях паркування. Дослідження показують, що HOG+SVM не працює при оклюзії більше 50%, а точність падає нижче 60%, коли автомобілі частково закриті сусідніми автомобілями або елементами конструкції [11]. Основне обмеження полягає в чутливості HOG до часткової оклюзії, оскільки його характеристики орієнтації градієнта погіршуються, коли відсутні частини об'єктів, а лінійні межі

прийняття рішень SVM не справляються з нелінійними моделями оклюзії, поширеними в умовах паркування [11].

Метод Віоли-Джонса, спочатку розроблений для розпізнавання облич, але придатний для розпізнавання транспортних засобів, використовує інтегральні характеристики зображення та AdaBoost для ефективної класифікації об'єктів. Хоча цей підхід є обчислювально ефективним, він має труднощі з масштабуванням та поворотами об'єктів, що є типовими для задач паркування, особливо коли транспортні засоби з'являються на різній відстані та під різними кутами відносно камери. Як HOG+SVM, так і метод Віоли-Джонса стикаються з серйозними проблемами при зміні освітлення, а їхня ефективність знижується на 40% в умовах слабкого освітлення [11].

Для усунення цих обмежень почали з'являтися гібридні підходи, що поєднують класичне вилучення ознак із більш досконалими методами класифікації. Вилучення ознак HOG, інтегроване з базовими мережами ResNet-50, зменшило кількість помилкових спрацьовувань на 31% в умовах слабкого освітлення, надаючи перевагу структурним ознакам над інтенсивністю пікселів [10].

Класичні методи виявлення об'єктів встановили систематичні основи для ідентифікації транспортних засобів та моніторингу місць паркування місць через ретельно організовані послідовності кроків обробки. Техніка ковзного вікна сформувала основну методологію, скануючи зображення рухомими вікнами різних розмірів для зіставлення з попередньо визначеними шаблонами об'єктів. Цей підхід вимагав ретельного налаштування розміру вікна та параметрів перекриття, щоб збалансувати точність виявлення та обчислювальну складність. Параметр перекриття дозволяє зберігати інформацію між сусідніми вікнами, хоча надмірне перекриття збільшувало обчислювальне навантаження, тоді як недостатнє перекриття ризикувало пропустити об'єкти. Обчислювальне навантаження ковзних вікон було значним, особливо в поєднанні з кроками виділення ознак та класифікації, що знижувало продуктивність в реальному часі без оптимізацій. Методи віднімання фону, включаючи Гаусові сумішеві моделі та адаптивні медіанні фільтри, забезпечували критично важливу попередню обробку, відокремлюючи

рухомі об'єкти переднього плану від статичного фону [8]. Ці методи дозволили подальшій обробці зосередитися окремо на зонах, що містять потенційні транспортні засоби, значно зменшуючи обчислювальне навантаження. У системах виявлення місць паркування віднімання фону дозволило ідентифікацію порожніх місць у реальному часі шляхом динамічного оновлення фонових моделей та виявлення змін, що свідчать про присутність або відсутність транспортного засобу. Інтеграція перетворення перспективи з виявленням контурів та виділенням ліній створила комплексні конвеєри для окреслення місць паркування, де трансформовані зображення спростили застосування фільтрів Кенні та Гауса для виявлення меж.

Сучасні парадигми виявлення на основі машинного навчання зазнали значної оптимізації та призвели до появи гібридних підходів, які поєднали традиційний комп'ютерний зір та методології глибокого навчання. Техніка ковзного вікна еволюціонувала завдяки складним оптимізаціям, які вирішили проблеми обчислювальної складності. Регульований розмір вікна та параметри перекриття стали невід'ємною частиною балансування точності виявлення та вимог до обробки, причому вдосконалені реалізації використовували адаптивний вибір вікна на основі вмісту зображення, а не певних фіксованих значень. Інтеграція класифікаторів машинного навчання з підходами ковзного вікна дозволила більш оптимально обирати регіони розпізнавання, де класифікатори могли надавати пріоритет регіонам, які, найімовірніше, містять об'єкти для розпізнавання.

Ансамблеві методи (Ensemble methods) з'явилися як потужні інструменти для підвищення продуктивності виявлення, поєднуючи прогнози від кількох класифікаторів для покращення стійкості та точності. Дослідження продемонстрували, що ансамблеві класифікатори можуть досягти якості виявлення в діапазоні від 89,3% до 95% на спеціальних наборах даних для завдань виявлення об'єктів, що представляє значні покращення порівняно з однорідними класифікаторами. Підходи, засновані на контекстуальній інформації, дозволили краще розуміти зображення, дозволяючи системам використовувати контекст об’єктів в зображенні для підвищення точності. Ці методи включали просторові

зв'язки між об'єктами, часову узгодженість у відео та семантичне розуміння місць паркування для підвищення надійності виявлення [12].

Гібридні методи, що поєднують класичні засоби розпізнавання з машинним навчанням представляли особливо перспективний напрямок, поєднуючи сильні сторони традиційної обробки зображень з можливостями машинного навчання. Системи, які зберігали класичні кроки попередньої обробки (перетворення перспективи, виявлення контурів), одночасно включаючи класифікатори машинного навчання для остаточного прийняття рішень, продемонстрували підвищену продуктивність у конкретних сценаріях [13].

Поява одноетапних алгоритмів виявлення, як-от YOLO (You Only Look Once) та SSD (Single Shot MultiBox Detector), ознаменувала зміну підходів від традиційних підходів ковзного вікна. Ці алгоритми розглядали виявлення об'єктів як уніфіковану проблему регресії, безпосередньо видаючи результати виявлення без необхідності класифікації кандидатного регіону. Цей підхід усунув необхідність окремих етапів пропозиції регіону та класифікації, поширених у більш ранніх методах, значно підвищуючи обчислювальну ефективність.

1.3. Аналіз застосування нейронних мереж у задачах комп'ютерного зору

Розвиток згорткових нейронних мереж є одним з найважливіших проривів у галузі комп'ютерного зору, коріння якого сягає глибоко в нейробіологію та математичну теорію. Основним джерелом натхнення стала піонерська робота Девіда Г. Губеля та Торстена Н. Візеля, які за допомогою мікроелектродних записів на анестезованих котах склали карту ієрархічної організації зорової кори ссавців [13]. Їхнє важливе відкриття виявило два основні типи нейронів, що обробляють зорову інформацію: прості клітини, які вибірково реагують на лінії певної орієнтації в точних точках сітківки, та складні клітини, які зберігають вибірковість орієнтації, одночасно реагуючи на рухомі лінії в ширших зорових полях.

Така біологічна ієрархія безпосередньо надихнула архітектурні принципи сучасних згорткових нейронних мереж. Згорткові шари в згорткових мережах

аналогічні можливостям простих нейронів щодо виявлення локальних ознак, тоді як операції об'єднання відображають властивості незмінності положення складних нейронів. Ранні реалізації, такі як Neocognitron Фукусіми (1980), явно використовували ручні фільтри Габора для моделювання цих біологічних властивостей рецептивних полів, заклавши основу для сучасних підходів до глибокого навчання [14].

Експерименти Губеля і Візеля з кошенятами продемонстрували критичний період для розвитку зору, показавши, що сенсорний досвід глибоко впливає на формування кортикальних зв'язків. Кошенята, яким протягом перших трьох місяців життя зашивали одне око, зазнали різкого зниження коркового представлення для позбавленого зору ока, причому нейрони отримували в 10 000–100 000 разів менше світла. Цей принцип пластичності підкреслює важливість якості та різноманітності навчальних даних у розробці згорткових мереж, висвітлюючи, як вплив навколишнього середовища формує обчислювальні представлення.

Еволюція архітектур ЗНМ для виявлення об'єктів була обумовлена необхідністю вирішити три критичні показники продуктивності: точність, швидкість і масштабованість. Ранні рішення на основі ЗНМ стикалися з проблемою обчислювальної складності, часто вимагаючи багаторазового проходження вхідних зображень для досягнення прийнятних показників виявлення. Сучасні архітектури подолали ці обмеження завдяки архітектурним інноваціям, що оптимізують як швидкість виведення, так і ємність моделі.

Перехід від традиційних підходів до комп'ютерного зору до методів, заснованих на глибокому навчанні, був особливо драматичним у сфері виявлення об'єктів. Хоча раніше методи в значній мірі покладалися на створені вручну функції та прості класифікатори, ЗНМ продемонстрували здатність навчатися розпізнавати ознаки безпосередньо з даних, що призвело до суттєвого підвищення точності виявлення в різних завданнях і сферах. Ця зміна дозволила розробити системи виявлення в режимі реального часу, здатні обробляти зображення з високою роздільною здатністю з безпрецедентною швидкістю та точністю.

В таблиці 1.1 представлена основна хронологія розвитку підходів до розпізнавання.

Таблиця 1.1 — Розвиток підходів до розпізнавання

Період	Архітектура	Рік	Основні досягнення	Вплив на індустрію
Традиційні підходи	Hand-crafted Feature Extraction	1960s-2000s	SIFT, HOG, SURF, Фільтри Габора, каскдади Гаара	Основа комп’ютерного зору до глибокого навчання
Раннє глибоке навчання	LeNet-5	1998	Перше практичне застосування згорткових мереж	Продемонстровано ефективність використання градієнтів для навчання згорткових мереж
Революція глибокого навчання	AlexNet	2012	Глибші нейронні мережі, функція активації ReLU, навчання на графічних процесорах	Початок революції глибокого навчання в області комп'ютерного зору
	VGGNet	2014	Дуже глибока мережа (16-19 шарів) з малими 3×3 фільтрами	Доведена важливість глибини та простоти мережі
	GoogLeNet (Inception v1)	2014	Inception модуль, ефективне використання параметрів	Представлено ефективну багатомасштабну обробку зображень
	ResNet	2015	Залишкові зв'язки, що дозволили навчати дуже глибокі мережі (152+ шари)	Вирішено проблему зникаючого градієнта, стало стандартною основою для навчання мереж
	YOLO	2015	Потребує лише одного проходження вперед для здійснення передбачень	Швидка та ефективна архітектура для задач розпізнавання в реальному часі
Фокус на ефективності та спеціалізації	DenseNet	2017	Щільні зв'язки між шарами, ефективність параметрів	Альтернатива залишковим з'єднанням з кращою ефективністю параметрів
	MobileNets	2017	Глибинні роздільні згортки для ефективності	Створення ефективних моделей для мобільних пристроїв
	EfficientNet	2019	Комбіноване масштабування глибини, ширини та роздільної здатності	Досягнуто високої ефективності в різних масштабах

Розвиток архітектур характеризується кількома ключовими переходами:

— від виявлення на основі регіонів до виявлення за один раз: перехід від декількох етапів обробки до уніфікованих архітектур, які виконують виявлення за один прохід;

— від фіксованого до адаптивного вилучення ознак: включення механізмів уваги та динамічного вибору ознак для зосередження обчислювальних ресурсів на найбільш інформативних регіонах;

— від виявлення на основі якорів до виявлення без якорів: усунення необхідності у попередньо визначених яркових рамах за допомогою підходів прогнозування на основі точок або кутів;

— від архітектур, що базуються виключно на згорткових нейронних мережах, до гібридних архітектур.

Перехід від архітектур, що базуються виключно на ЗНМ, до моделей з використанням механізму уваги є ще однією важливою тенденцією в дослідженнях у галузі виявлення об'єктів. DETR (DEtection TRansformer), представлений Facebook AI Research, використовує архітектуру Transformer, в якій кодувальник обробляє вхідне зображення для вилучення ознак і розуміння глобального контексту зображення, а декодувальник генерує запити на об'єкти для прогнозування координат обмежувальної рамки та міток класів за допомогою двобічного збігу втрат.

Виходячи з розглянутого в розділі, було вирішено застосувати нейронні мережі, а саме застосовано згорткову нейронну мережу архітектури YOLO для локалізації автомобіля в зображені.

У цьому розділі магістерської роботи був проведений аналіз проблеми пошуку та моніторингу місць паркування, розглянуто сучасні підходи до автоматизованого моніторингу, розглянуто застосування нейронних мереж в задачах комп’ютерного зору.

2 РОЗРОБКА ПОСЛІДОВНОСТІ ТА ЗАСОБІВ РОЗПІЗНАВАННЯ СИСТЕМИ МОНІТОРИНГУ МІСЦЬ ПАРКУВАННЯ

2.1. Розробка засобів розпізнавання автомобілів

Згорткові нейронні мережі кардинально змінили комп'ютерний зір, завдяки використанню ієрархічного вилучення ознак через згорткові шари, що імітують ієрархію обробки інформації в зоровій корі головного мозку людини. Для виявлення об'єктів на основі відео в системах управління паркуванням ЗНМ повинні вилучати як просторові, так і часові ознаки, щоб ефективно відстежувати об'єкти в послідовних кадрах, зберігаючи при цьому здатність розрізняти нерухомі та рухомі транспортні засоби. Основна перевага ЗНМ полягає в їхній здатності автоматично вивчати ознаки, що мають значення, з необроблених даних пікселів без необхідності ручного формування ознак, що робить їх надзвичайно корисними для використання в динамічних умовах майданчиків паркування, де освітлення, погода та кут огляду камер можуть різко змінюватися протягом дня [15].

Сучасні підходи черпають натхнення з біологічних систем зору, де інформація про рух покращує можливості виявлення об'єктів, надаючи додаткові підказки, що виходять за межі статичного зображення. Система зору людини інтегрує сигнали руху з магноцелюлярного шляху з обробкою форми з парвоцелюлярного шляху, створюючи надійну систему сприйняття, яка чудово справляється з виявленням об'єктів у русі. Виявлення об'єктів на відео для сценаріїв паркування може використовувати візуальні характеристики руху, витягнуті з збережених послідовних кадрів, для покращення статичних характеристик зображення в детекторах об'єктів [15]. Цей підхід є особливо цінним для виявлення невеликих або частково перекритих транспортних засобів у переповнених середовищах паркування, де сигнали руху можуть компенсувати обмежену візуальну інформацію.

Математичні операції в зготкових мережах повинні бути оптимізовані спеціально для завдань локалізації, а не для чистої класифікації, оскільки

управління паркуванням вимагає точних координат обмежувальної рамки на додаток до точних прогнозів класу.

Головним теоретичним принципом, що лежить в основі згорткових нейронних мереж, є їх здатність будувати просторову ієрархію ознак за допомогою послідовних згорткових шарів. Ієрархічна обробка відображає принцип роботи біологічної системи зору, де ранні зорові області (V1) виокремлюють прості ознаки, такі як краї та текстури, а вищі області (V2, V4, IT) поєднують їх у все більш складні представлення частин об'єктів та об'єктів у цілому [16]. Ця прогресія математично проявляється як пошарова композиція, де вихід одного згорткового шару стає входом для наступного, що дозволяє мережі будувати складні ознаки з простіших компонентів [17]. На рисунку 2.1 представлено одну з перших архітектур згорткової нейронної мережі LeNet.

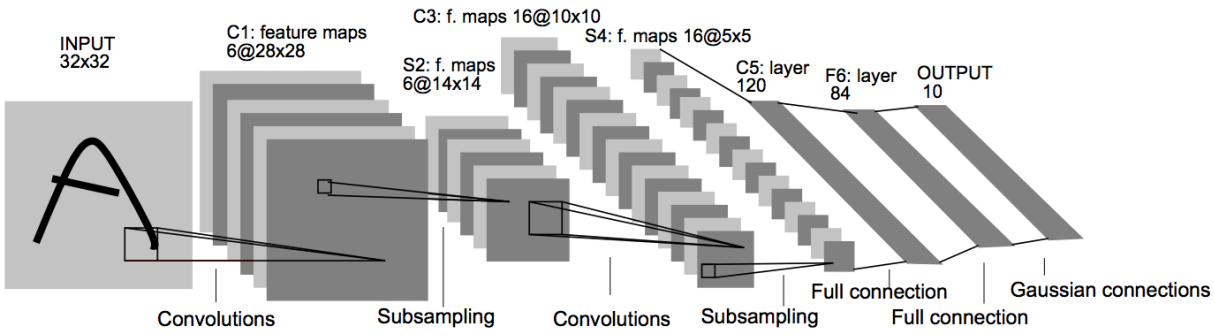


Рисунок 2.1 — Архітектура LeNet-5

Основна операція в згорткових мережах це згорткові фільтри, які сковзають по вхідних зображеннях, витягуючи локальні ознаки за допомогою спільного використання параметрів, що значно зменшує кількість параметрів у порівнянні з повністю з'єднаними мережами.

Ранні шари зазвичай вивчають фільтри типу Габора, які реагують на орієнтовані краї та контрасти кольорів, тоді як середні шари поєднують їх у детектори текстур та розпізнавачі закономірностей, а глибші шари розробляють представлення частин об'єктів та повних об'єктів. Така структура дозволяє

обробляти надзвичайно складні зображення за допомогою розподіленого ієрархічного представлення.

Для вхідного тензора розміром X, операція згортки може бути представлена наступним чином:

$$Y[i,j] = \Sigma_{m=0}^{Fh-1}\Sigma_{n=0}^{Fw-1}X[i+m, j+n] \cdot K[m,n] + b \quad (2.1)$$

де Y — вихідна карта ознак;

K — згортковий фільтр;

F_h, F_w— висота, ширина фільтра;

m,n —розмір кроку фільтра

b — зсув тензора.

Параметр кроку контролює, на скільки пікселів фільтр переміщується між застосуваннями, а параметр відступу визначає, чи відповідають вихідні розміри вхідним розмірам. У системах виявлення роздільні за глибиною згортки значно зменшують обчислювальну складність, зберігаючи при цьому точність виявлення, що робить їх ідеальними для середовищ з обмеженими ресурсами, де необхідно одночасно обробляти багато відеопотоків. Роздільні за глибиною згортки розкладають стандартну згортку на дві операції: згортку за глибиною, яка застосовує один фільтр на кожен вхідний канал, а потім точкову згортку (згортку 1x1), яка об'єднує виходи каналів.

Наступним кроком є функції активації, що вводять нелінійність у нейронні мережі, дозволяючи їм вивчати складні закономірності, які не можуть бути представлені лінійними перетвореннями. ReLU (Rectified Linear Unit) та її варіанти (Leaky ReLU, PReLU) вводять нелінійність, одночасно вирішуючи проблеми зникаючого градієнта, які можуть перешкоджати навчанню в глибоких мережах.

Leaky ReLU вирішує проблему «вмираючого ReLU», коли нейрони можуть стати неактивними і припинити навчання, дозволяючи невеликий, відмінний від нуля градієнт, коли нейрон не активний. Варіант ReLU6 обмежує вихідне значення, запобігаючи безмежному зростанню активацій під час навчання, що покращує

чисельну стабільність, особливо в середовищах розгортання з низькою точністю [18]. Часто застосовувані функції представлено на рисунку 2.2.

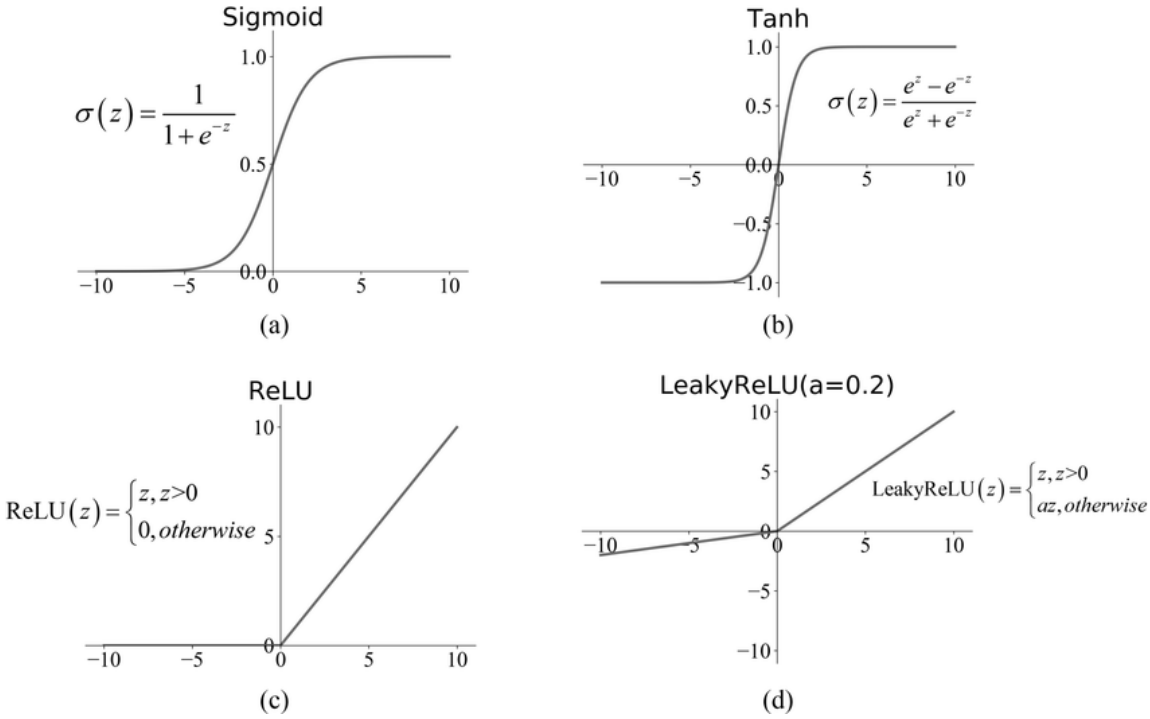


Рисунок 2.2 — Функції активації

Більш сучасні функції активації, такі як Swish і Mish, показали себе перспективними в певних сценаріях виявлення. Однак їх підвищена обчислювальна складність повинна бути зважена з перевагами в продуктивності для паркувальних додатків. Архітектура YOLO завдяки своєму рівню оптимізації дозволяє ефективно застосовувати функцію SiLU(Swish) для більш плавних градієнти і потенційно покращеної динаміки навчання, що дозволяє підвищити точність розпізнавання.

Крок аггрегації шарів є наступним після шару з функцією активації, його задача зменшити розмірність тензора, зберігаючи при цьому важливі ознаки, контролюючи зростання зони чутливості мережі та забезпечує можливість розпізнавання не залежно від положення об’єкта. Функція максимізаційної агрегації, що вибирає максимальне значення з кожної області, залишається популярним завдяки своїй здатності зберігати найсильніші сигнали активації. Функція усередненої агрегації обчислює регіональні середні значення, забезпечуючи більш плавне зменшення роздільної здатності, що може бути

корисним для завдань регресії, таких як прогнозування координат обмежувальної рамки [16]. Для задач розпізнавання місць паркування адаптивні механізми об'єднання дозволяють зберігати дрібні деталі, необхідні для розрізнення подібних моделей автомобілів або виявлення зайнятості в обмеженому просторі, де невеликі візуальні ознаки визначають стан паркування.

Правильна організація агрегації дозволяє збалансувати обчислювальні витрати і точність виявлення, контролюючи швидкість зниження просторової роздільної здатності в мережі. Агресивна агрегація процесу зменшує обчислювальне навантаження, але може призвести до втрати дрібних деталей, необхідних для виявлення невеликих об'єктів, тоді як пізній пулінг зберігає просторову роздільну здатність за рахунок збільшення обчислювальних витрат [19]. Для паркувальних додатків цей компроміс повинен бути оптимізований з урахуванням роздільної здатності камери, типових розмірів об'єктів і доступних обчислювальних ресурсів.

Останні підходи включають операції агрегації, що піддаються навчанню, де стратегія об'єднання оптимізується під час навчання, а не встановлюється заздалегідь. Ці адаптивні механізми об'єднання можуть навчитися зберігати просторово важливі особливості, характерні для сценаріїв паркування, такі як області номерних знаків або характерні особливості транспортних засобів, що допомагають у повторній ідентифікації.

Окрім основних згорткових і агрегаційних шарів, згорткові нейромережі містять кілька спеціалізованих складових, що підвищують їх функціональність та ефективність. Повністю з'єднані шари зазвичай з'являються в кінці мережі, відображаючи високорівневі особливості, витягнуті згортковими шарами, на остаточні дані (наприклад, результати розпізнавання). Такі шари вимагають вирівнювання просторових карт особливостей у 1D-вектори, де кожен вхід з'єднаний з кожним виходом за допомогою ваг, що піддаються навчанню [20]. Хоча цей підхід є ефективним, він може призвести до вибуху параметрів, що спонукає до створення нових рішень.

Згорткова функція 1×1 є особливо важливим архітектурним рішенням, яке виконує перетворення ознак по каналах, застосовуючи лінійну комбінацію вхідних каналів у кожному просторовому положенні. Ця операція дозволяє зменшити розмірність у каналі без зміни просторових розмірностей (H×W). При застосуванні фільтрів f2 до входу з каналами f1 (де f2 < f1) це дозволяє ефективно представляти оригінальні фільтри f1 через вивчені фільтри f2, змушуючи мережу вивчити найбільш ефективне скорочення [21]. Обчислювальна ефективність цього методу є значною: зменшення розміру вхідної карти ознак 28×28×128, що згортається з 32 фільтрами розміром 7×7 (що вимагає ~236 мільйонів операцій), до використання попереднього шару згортки 1×1 (наприклад, зменшення до 32 каналів) знижує кількість операцій до ~21 мільйона.

Спеціально для завдань локалізації деякі архітектури використовують різні функції активації в різних частинах мережі. Класифікаційні функції можуть отримати переваги від активацій, що забезпечують сильні градієнтні сигнали для розрізнення категорій, тоді як регресійні частини для координат обмежувальних рамок можуть працювати краще з активаціями, що підтримують плавне прогнозування координат. Цей спеціалізований підхід може оптимізувати загальну продуктивність виявлення шляхом адаптації математичних властивостей функцій активації до конкретних підзавдань, які виконує кожен компонент мережі.

Архітектурні особливості згорткових мережах дозволяють їм фіксувати ієрархічні представлення, де нижчі шари виявляють прості ознаки, такі як краї та кути, а глибші шари поєднують їх у складні структури, що представляють форми транспортних засобів, номерні знаки та межі місць паркування. Ієрархічна обробка є особливо корисною в сценаріях паркування, де транспортні засоби можуть бути частково перекритими, розташовані під нестандартними кутами або зняті в складних умовах освітлення.

Традиційні двовимірні згорткові мережі обробляють окремі кадри незалежно один від одного, але виявлення об'єктів на відео в системі управління паркуванням вимагає розуміння закономірностей руху та часової стабільності в декількох кадрах. Тривимірні згорткові мережі розширюють цю концепцію, застосовуючи об'ємні

згортки як у просторовому, так і в часовому вимірі, створюючи більш широке розуміння поведінки транспортних засобів. Це дозволяє за допомогою камер розрізняти транспортні засоби, які паркуються, виїжджають з місця паркування або просто проїжджають через майданчик [22].

Часовий вимір у тривимірних згорткових мережах дозволяє мережі вивчати характерні для паркування моделі руху, такі як характерне уповільнення та повороти автомобілів, що шукають місце для паркування, або маневри заднього ходу, пов'язані з паркуванням та виїздом з місця паркування. Часові характеристики можуть значно підвищити точність виявлення, особливо в умовах великого скупчення автомобілів, коли аналіз статичних зображень може ускладнюватися через перекриття або незвичайне розташування автомобілів. Об'ємні згортки в тривимірних згорткових мережах працюють на накладених кадрах, що дозволяє мережі вивчати фільтри, які реагують на конкретні просторово-часові патерни, а не тільки на просторові особливості [22].

Більш складні рішення поєднують тривимірні згорткові функції з рекурентними нейронними мережами або архітектурою трансформер для фіксації довгострокових часових залежностей, що виходять за межі безпосередньої послідовності кадрів, оброблюваних тривирмірними згортковими ядрами. Такі гібридні архітектури можуть зберігати контекст протягом більш тривалих періодів, що дозволяє системі відстежувати транспортні засоби навіть тоді, коли вони тимчасово зникають за перешкодами або переміщуються між зонами покриття камер. Для великих майданчиків паркування з декількома точками огляду камер довгострокове розуміння часових залежностей є необхідним для забезпечення послідовного відстеження транспортних засобів на всій території майданчика.

Архітектури розпізнавання об'єктів поділяються на дві основні категорії: одноступеневі детектори та двоступеневі детектори. Кожен підхід пропонує певні компроміси між швидкістю та точністю, які необхідно оцінювати з урахуванням вимог до управління паркуванням, де критично важливими є як продуктивність у реальному часі, так і надійність розпізнавання.

Одноступеневі детектори виконують локалізацію та класифікацію об'єктів за один прохід мережі, що робить їх за своєю суттю швидшими, ніж двоступеневі підходи, оскільки виключає етапи генерації та уточнення пропозицій. Архітектури YOLO (You Only Look Once) еволюціонували через кілька версій, і YOLOv11 являє собою оптимізований для задач реального часу інструмент, завдяки своїй оптимізованій архітектурі та збалансованому профілю продуктивності [23].

YOLOv11 має кілька ключових рішень, а саме дизайн без якорів, що усуває заздалегідь визначені опорні рамки та зменшує обчислювальні витрати. Це забезпечує динамічне прогнозування розташування об'єктів без налаштування гіперпараметрів, які необхідні для підходів на основі анкерів. Модуль C2f замінює попередній модуль C3, мінімізуючи надлишкові обчислення завдяки оптимізованому вилученню ознак, що зберігає представницьку силу та одночасно зменшує обчислювальне навантаження [23]. Ці вдосконалення роблять YOLOv11 особливо корисним для задач паркування, що вимагають високої частоти кадрів без втрати точності, особливо при одночасній обробці декількох відеопотоків.

Одноступеневий детектор SSD (Single Shot MultiBox Detector), представляє альтернативний підхід, який використовує заздалегідь визначені опорні рамки на декількох масштабах карт ознак. Хоча SSD, як правило, швидше за двоступеневі детектори, він може мати труднощі з виявленням невеликих об'єктів через обмежену роздільну здатність більш глибоких карт ознак. Для систем паркування, де автомобілі можуть виглядати невеликими на оглядових камерах, багатомасштабні можливості виявлення YOLOv11 часто забезпечують чудову продуктивність, незважаючи на складність архітектури [11].

Faster R-CNN використовує двоступеневий алгоритм: спочатку генеруються пропозиції регіонів за допомогою мережі пропозицій регіонів (RPN), а потім на другому етапі ці пропозиції класифікуються та уточнюються. Хоча цей метод загалом є точнішим, ніж одноразові детектори, він вимагає більших обчислювальних витрат через послідовний процес обробки. Інтеграція мереж пропозицій регіонів забезпечує швидке створення пропозицій регіонів, що дозволяє Faster R-CNN досягати помірних швидкостей виявлення на великих наборах даних

завдяки обміну характеристиками між етапами створення пропозицій та виявлення [11,23]. Мережі пропозицій регіонів працюють шляхом переміщення невеликої мережі по згортковій карті ознак, одночасно прогнозуючи межі об'єкта та оцінки об'єктності в кожній позиції. Цей підхід генерує високоякісні пропозиції регіонів з мінімальними обчислювальними витратами в порівнянні з попередніми методами, такими як Selective Search. На другому етапі кожна пропозиція обробляється за допомогою об'єднання регіонів інтересу (Region of Interest) для вилучення ознак фіксованого розміру, які подаються в мережі класифікації та регресії обмежувальних рамок [20].

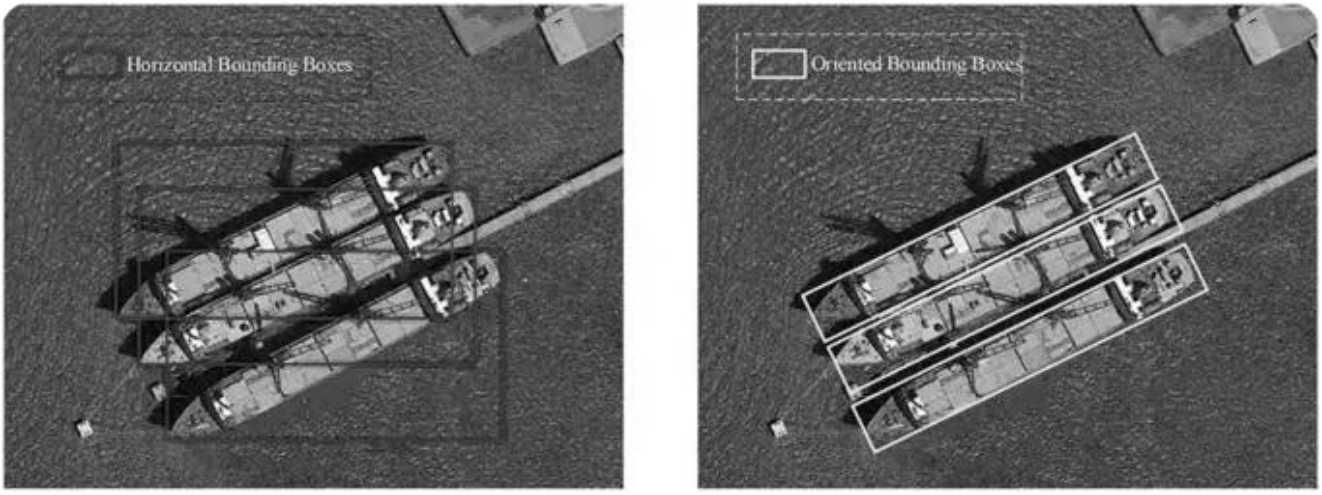
З розглянутого, можна зробити висновок що YOLOv11 є найбільш доцільною для використання нейронною мережею.

Як вже було сказано раніше, гранична рамка є основним інструментом локалізації об’єктів під час розпізнавання, тобто визначення їх положення в просторі. Традиційні обмежувальні рамки, що вирівняні по осям, мають значну похибку визначення при застосуванні її до об'єктів із значним кутовим відхиленням від координатних осей, особливо до подовжених структур, таких як транспортні засоби, кораблі або текст. Це призводить до значного включення пікселів фону в граничну область, що вносить шум у процес вилучення ознак і погіршує ефективність класифікації. Задля підвищення точності необхідно більш точне представлення геометрії об'єкта, а саме включення ступенів свободи обертання, що дозволяє вирівняти граничну область з головними осями. цільового об'єкта, таким чином мінімізуючи перетин між граничною областю та нерелевантними областями фону. Така концепція має назву орієнтованих граничних рамок.

Орієнтовані обмежувальні рамки забезпечують більш компактне та ефективне представлення розмірів об'єкта. Зменшення зайвого простору в межах обмежувальної рамки збільшує точність для отриманих ознак, що в подальшому підвищує точність розпізнавання. Це стає особливо важливим у випадках, коли об'єкти розташовані близько один до одного, оскільки орієнтовані обмежувальні рамки можуть усунути неоднозначність об'єктів, що можуть перекриватися, які в іншому випадку були б об'єднані за допомогою підходів, вирівняних по осі.

Враховуючи, що задачою розроблюваної системи є розпізнавання автомобілів, наявність параметру кута є надзвичайно важливою. Це дозволяє засобу розпізнавання бути не залежним від того, під яким кутом будуть з’являтись об’єкти(в нашому випадку автомобілі) відносно розташування камери, що забезпечує стабільну ефективність виявлення при різних положення об'єктів, усуваючи необхідність у складних стратегіях збільшення обсягу навчальних даних, які в іншому випадку були б необхідні [24].

Незважаючи на ці переваги, впровадження виявлення орієнтованих граничних рамок ускладнює обчислення, особливо на етапі де під час розпізнавання об’єкта моделлю генерується безліч граничних рамок навколо об’єкта розпізнавання. Порівняння роботи класичних обмежувальних рамок та орієнтованих обмежувальних рамок представлено на рисунку 2.3.



а) класичні обмежувальні рамки б) орієнтовані обмежувальні рамки

Рисунок 2.3 — Визначення фінальної обмежувальної рамки

Однак останні досягнення в оптимізації архітектур моделей зробили підходи на основі орієнтованих граничних рамок все більш придатними для застосування в режимі реального часу.

Впровадження орієнтованих граничних рамок у системи виявлення об'єктів вимагає комплексного переосмислення традиційних архітектур виявлення з метою врахування додаткового параметра обертання. Математичне представлення

орієнтованих граничних рамок зазвичай складається з п'яти параметрів: координат центру (x, y), ширини (w), висоти (h) і кута повороту (θ). Ця параметризація розширює традиційне представлення обмежувальних прямокутників з чотирма параметрами та вирівняних по осі, шляхом включення кутової інформації, що фундаментально змінює процес виявлення.

Формування вихідних даних в системах орієнтованих граничних рамок істотно відрізняється від класичних рішень. Існує кілька архітектурних варіантів, серед яких найпопулярнішими є методи прямої регресії та підходи на основі декомпозиції. Методи прямої регресії одночасно прогнозують всі п'ять параметрів (x, y, w, h, θ), часто використовуючи спеціальні функції активації для обмеження діапазонів параметрів. Наприклад, кут повороту зазвичай обмежується [-90°, 90°] або [0°, 180°] за допомогою тригонометричних функцій, щоб уникнути кутових розривів. Підходи на основі декомпозиції передбачають чотири стандартні параметри обмежувальної рамки разом з додатковою класифікацією або регресією кута [24].

Процес навчання моделей розпізнавання з застосуванням орієнтованих граничних рамок зазвичай вимагає спеціальних методів збільшення обсягу даних. Основним методом збільшення обсягу даних для таких систем є випадкове обертання зображення, але його необхідно застосовувати обережно, щоб забезпечити збіжність перетворення як зображень, так і відповідної розмітки. Інші методи збільшення обсягу даних, такі як масштабування, дзеркальне відображення та перетворення перспективи, необхідно адаптувати для правильної обробки розмітки.

2.2. Розробка процесу навчання нейронної мережі

В загальному систему моніторингу місць паркування можна поділити на дві умовні частини:

— навчання згорткової нейронної мережі відповідно до потреб системи моніторингу;

— використання результату роботи нейронної мережі для визначення стану місць паркування.

Підготовка та попередня обробка даних є основним етапом у розробці моделей згорткових нейронних мереж, що значно впливає на здатність моделі навчатися корисним представленням та узагальнюватися на невідомих даних. На цьому етапі використовується низка методів, розроблених для перетворення необроблених даних у формат, придатний для навчання нейронної мережі, та одночасного вирішення потенційних проблем, таких як обмежена доступність даних, незбалансованість класів та специфічні вимоги до конкретної галузі.

Процес навчання нейронної мережі це комплексний процес, що складається з багатьох кроків, та представлено на рисунку 2.4.

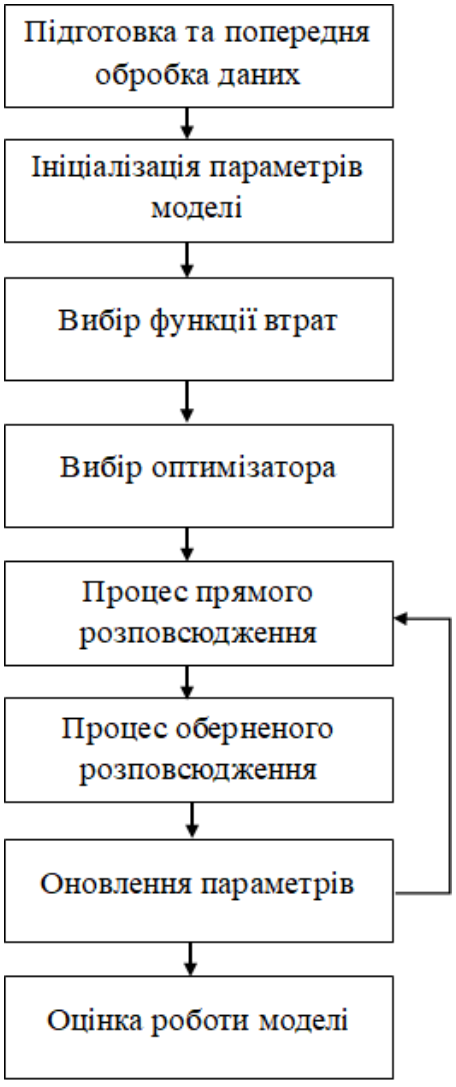


Рисунок 2.4 — Процес навчання моделі

Перший крок у підготовці даних передбачає систематичний збір та організацію зображень відповідно до встановлених класів, що стосуються конкретної задачі комп'ютерного зору. Сюди входить класифікація, де зображення розподіляються за окремими категоріями, сегментація, де необхідна анотація на рівні пікселів, розпізнавання об'єктів із визначенням обмежувальних рамок або задачі регресії, що передбачають прогнозування послідовних значень. Набір даних повинен бути структурований за допомогою відповідних ієрархій каталогів або файлів метаданих, що дозволяють зберігати інформацію про вхідні дані та відповідною анотацією [25].

Після організації набір даних проходить процес поділу на окремі піднабори: навчальний, валідаційний та тестовий. Навчальний піднабір слугує основним джерелом даних для оптимізації моделі за допомогою зворотного поширення, тоді як валідаційний піднабір полегшує налаштування гіперпараметрів та вибір моделі під час розробки. Тестовий піднабір, який повністю відокремлений від процесу навчання, забезпечує об'єктивну оцінку можливостей узагальнення кінцевої моделі. Типові співвідношення поділу включають 70-80% для навчання, 10-15% для валідації та 10-15% для тестування, хоча ці пропорції можуть бути скориговані залежно від розміру набору даних та вимог завдання [25].

Аугментація даних є критично важливою технікою попередньої обробки, яка штучно розширює навчальний набір даних шляхом застосування різних перетворень, що зберігають семантичний зміст зображень, змінюючи їх представлення на рівні пікселів. Поширені стратегії збільшення включають геометричні перетворення, такі як випадкові обертання, перенесення, масштабування та відбиття, які надають різноманіття в просторі вхідних даних. Фотометричні перетворення, включаючи регулювання яскравості, контрасту, насиченості та відтінку, підвищують стійкість моделі до різних умов освітлення. Просунуті методи можуть включати випадкове стирання, вирізання або змішування, що вносить оклюзії або комбінує кілька зразків для подальшої регуляризації моделі та зменшення перенавчання.

Нормалізація є ще одним важливим етапом попередньої обробки, під час якого значення пікселів перераховуються до стандартизованого інтервалу, що сприяє збіжності оптимізації параметрів моделі. Зазвичай це передбачає масштабування інтенсивності пікселів з їхнього вихідного діапазону (наприклад, 0–255 для 8-бітних зображень) до нормалізованого інтервалу, зазвичай [0,1] або [-1,1]. У багатьох випадках застосовується нормалізація за каналами, де кожен колірний канал стандартизується незалежно, шляхом віднімання середнього значення та ділення на стандартне відхилення, обчислене для тренувального набору даних. Цей процес, відомий як нульове центрування та відбілювання, узгоджує розподіл вхідних даних з припущеннями, зробленими багатьма алгоритмами оптимізації та функціями активації.

Кодування категорійних ознак є ще одним важливим аспектом попередньої обробки, особливо в завданнях класифікації. Категорійні ознаки зазвичай перетворюються на вектори з бінарним кодуванням, де n-вимірний вектор містить значення 1 у позиції, що відповідає правильному класу, і 0 в інших позиціях. Для задач класифікації за декількома ознаками використовується бінарне кодування, де кожен клас представлений бінарним значенням у векторі ознак, що дозволяє проводити кілька класифікацій одночасно. У задачах сегментації анотації на рівні пікселів можуть вимагати кодування в канали, специфічні для класу, або використання спеціалізованих схем кодування, таких як карти відстаней, для усунення дисбалансу класів.

Оптимізація формату даних є технічним, але важливим аспектом попередньої обробки, особливо для сценаріїв навчання у великих масштабах. Це може включати перетворення даних зображень у більш ефективні формати, такі як TFRecords, LMDB або HDF5, що забезпечують швидші операції вводу-виводу під час навчання. Зміна розміру зображень до однакових розмірів із збереженням пропорцій за допомогою заповнення або обрізання забезпечує сумісність із фіксованими розмірами вхідних даних згорткової архітектури.

Обмеження пам'яті вимагають застосування ефективних конвеєрів завантаження даних, які часто включають паралельні механізми попередньої

обробки та попереднього завантаження для максимального використання GPU. Спеціальні утиліти для конкретних фреймворків, такі як tf.data API від TensorFlow або клас DataLoader від PyTorch, дозволяють створювати оптимізовані потоки вхідних даних, які можуть виконувати операції доповнення, нормалізації та пакетної обробки одночасно з навчанням моделі, мінімізуючи періоди простою в циклі навчання.

Наступним етапом є ініціалізація параметрів моделі, де правильна ініціалізація вирішує основну проблему різкого порушення симетрії в просторі параметрів мережі, зберігаючи при цьому різноманіття параметрів для належного поширення сигналу через глибокі мережі. Без ретельної ініціалізації згорткові мережі можуть страждати від зникнення або вибуху градієнтів, що значно ускладнює процес оптимізації.

В основі теорії ініціалізації параметрів лежить необхідність збереження дисперсії активацій та градієнтів у всій мережі. Коли вхідні дані поширюються через кілька шарів, невідповідне масштабування може призвести до ослаблення сигналу (зникнення градієнтів) або його надлишкового посилення («вибух» градієнтів). Математичний аналіз цього явища показує, що схеми ініціалізації повинні підтримувати баланс, при якому дисперсія вихідних даних залишається приблизно рівною дисперсії вхідних даних у всіх шарах.

Традиційні підходи до ініціалізації використовували випадкові значення, взяті з рівномірного або нормального розподілу з невеликими величинами, зазвичай в діапазоні [-0,05, 0,05]. Однак ці методи не мали теоретичного обґрунтування щодо збереження дисперсії сигналу і часто вимагали ретельного налаштування на основі конкретних архітектур мережі.

Ініціалізація Ксав'є, також відома як ініціалізація Глоро, є значним прогресом в ініціалізації параметрів. Цей метод враховує кількість вхідних і вихідних з'єднань для кожного нейрона при визначенні відповідної дисперсії початкового розподілу. Такий підхід передбачає лінійні активації та симетричний розподіл ваг, що робить його теоретично оптимальним для шарів із сигмоїдними або тангенціальними функціями активації [25].

Ініціалізація Хе представляє ще один значний крок вперед, який було розроблено спеціально для мереж, що використовують випрямлені лінійні одиниці (ReLU) та їхні варіанти. Цей метод вирішує проблему несиметричності активацій ReLU, які пропускають лише позитивні значення. Ініціалізація Хе використовує дисперсію в межах $2/n_{вх}$ для нормально розподілених ваг, або в межах $\pm\sqrt{(6/ n_{вх})}$ для рівномірно розподілених ваг. Збільшена дисперсія компенсує приблизно вдвічі зменшений сигнал, який проходить через блоки ReLU під час прямого поширення.

Ортогональна ініціалізація пропонує альтернативний підхід, який зберігає норми градієнта під час зворотного поширення. Цей метод ініціалізує матриці ваг як ортогональні матриці, гарантуючи, що всі сингулярні значення матриці ваг дорівнюють одиниці. Хоча ортогональна ініціалізація є більш обчислювально затратною, вона може забезпечити покращену продуктивність у дуже глибоких мережах [26].

Ініціалізація зміщень зазвичай використовує простіші стратегії, часто встановлюючи їх на нуль або невеликі постійні значення. Для шарів з активаціями ReLU невелике позитивне зміщення (зазвичай 0,01) може сприяти початковій активації нейронів, потенційно прискорюючи ранні фази навчання.

Вибір методу ініціалізації значно впливає на критичний період навчання на ранніх етапах тренування. Правильна ініціалізація може прискорити сходимість десятикратно та дозволити тренувати значно глибші архітектури. Однак оптимальна стратегія ініціалізації може залежати від конкретних архітектурних рішень, функцій активації, шарів нормалізації та використовуваних алгоритмів тренування.

Вибір відповідної функції втрат оцінює розбіжність між прогнозованими результатами та реальними мітками, тим самим керуючи процесом оптимізації за допомогою градієнтного спуску. Вибір функції втрат в основному визначається конкретними вимогами завдання, характеристиками даних та бажаною поведінкою моделі.

Для завдань класифікації переважно використовується функція втрат перехресної ентропії.

У сценаріях бінарної класифікації використовується втрата бінарної перехресної ентропії (BCE) [27], яка математично виражається як:

$$L_{BCE} = -\frac{1}{N}\Sigma_{i=1}^{N}[y_i \log(\hat{y}i) + (1 - y_i)\log(1 - \hat{y}i)] \quad (2.2)$$

де N — розмір пакету даних;

y_i — мітка фундаментальної істини;

$\hat{y}i$ — передбачене значення.

Вибір функцій втрат також повинен враховувати динаміку оптимізації, яку вони викликають. Деякі функції втрат можуть створювати незбіжні ландшафти з численними локальними мінімумами, що потенційно перешкоджає збіжності. Крім того, властивості градієнта функції втрат значно впливають на стабільність навчання, причому функції, що демонструють зникаючі або вибухові градієнти, можуть вимагати модифікацій архітектури або альтернативних стратегій оптимізації [27].

Алгоритм оптимізації регулює оновлення параметрів моделі з метою мінімізації функції втрат. Вибір відповідного оптимізатора значно впливає на швидкість збіжності, кінцеву продуктивність та можливості узагальнення навченої моделі.

Основним підходом до оптимізації є стохастичний градієнтний спуск (SGD), який оновлює параметри у напрямку від'ємного градієнта функції втрат. Математично це можна виразити як:

$$\theta t + 1 = \theta t - \eta \nabla J(\theta t) \quad (2.3)$$

де θ — параметри моделі;

η — швидкість навчання;

$J(\theta)$ — цільовою функцією.

Незважаючи на свою простоту, звичайний стохастичний градієнтний спуск має кілька обмежень, включаючи повільну конвергенцію в ущелинах і високу варіативність оцінок градієнта [28].

Для усунення цих обмежень були введені методи, засновані на імпульсі. Імпульс додає термін швидкості, який накопичує градієнти минулих кроків, гальмуючи відхилення та прискорюючи збіжність у необхідних напрямках. Правило оновлення стає $v_{t+1}=\mu v_t+\eta\nabla J(\theta_t)$ та $\theta_{t+1}=\theta_t-v_{t+1}$ де μ є коефіцієнтом імпульсу. Nesterov Accelerated Gradient (NAG) ще більше вдосконалює процес, оцінюючи градієнт у позиції, що випереджає поточну, а не в поточній позиції [28].

Методи адаптивної швидкості навчання є ще одним значним прогресом в оптимізації. AdaGrad адаптує швидкість навчання для кожного параметра на основі історичної суми квадратів його градієнтів, з більшими оновленнями для нечасто використовуваних параметрів. Однак накопичення AdaGrad минулих градієнтів призводить до монотонного зниження швидкості навчання, що може передчасно припинити навчання. RMSprop вирішує це обмеження, використовуючи ковзну середню квадратичних градієнтів замість накопичення, підтримуючи більш ефективну швидкість навчання протягом усього навчання [28].

Оптимізатор Adam поєднує переваги методів імпульсу та адаптивної швидкості навчання, зберігаючи оцінки градієнтів першого та другого моментів. Оптимізатор Adam використовує правило оновлення, яке передбачає корекцію зміщення для цих оцінок моментів, що робить його особливо ефективним для задач з розрідженими градієнтами або шумними цілями.

При виборі оптимізатора для навчання згорткових нейромереж необхідно враховувати кілька факторів. На це рішення впливають характер набору даних, архітектура моделі та обчислювальні ресурси. Для добре обумовлених задач з великими наборами даних стохастичний градієнтний спуск з імпульсом часто досягає кращої генералізації, незважаючи на потенційно повільнішу збіжність. Adam та його варіанти зазвичай збігаються швидше і вимагають менше налаштування гіперпараметрів, що робить їх придатними для складних архітектур або в разі обмежених обчислювальних ресурсів.

Враховуючи наведену інформацію, для донавчання моделі для системи моніторингу місць паркування найбільш доцільним є використання оптимізатора Adam.

Прямий прохід у згортковій нейронній мережі представляє обчислювальний процес, за допомогою якого вхідні дані поширюються через ієрархічну архітектуру мережі для генерації прогнозів. Фактично процес прямого розповсюдження був розглянутий в розділі 2.1, і складається з послідовного застосування згортки, функції активації, агрегації та виконання в кінці функції для класифікації. Головною відмінністю процесу навчання є те, що протягом усього цього процесу мережа підтримує обчислювальний граф, який зберігає проміжні значення та операції, що є необхідним для подальшого зворотного проходження, де градієнти обчислюються за допомогою зворотного поширення. Кінцевий результат прямого проходження разом із мітками істинних значень потім використовується для обчислення функції втрат, яка кількісно оцінює розбіжність між прогнозами та фактичними значеннями і служить метою оптимізації під час навчання.

Зворотний прохід, також відомий як зворотне поширення, є компонентом, що забезпечує оптимізацію на основі градієнта в згорткових нейронних мережах. Цей процес передбачає систематичний обчислення часткових похідних функції втрат щодо кожного параметра в мережі шляхом застосування ланцюгового правила диференціального числення. Коли прямий прохід завершується, створюючи прогноз і відповідне значення втрати, зворотний прохід починається з обчислення градієнта функції втрати щодо вихідних даних останнього шару.

Математично, якщо позначити функцію втрати як L, а вихідні дані останнього шару як z_n, то початкове обчислення градієнта дає $\Delta L/\Delta z_n$, що представляє чутливість функції втрати до змін в кінцевих активаціях. Для згорткових шарів обчислення градієнта передбачає поширення цих сигналів помилки назад через операцію згортки. Градієнт щодо ядер (фільтрів) обчислюється як згортка вхідних активацій з градієнтом вищого рівня, тоді як градієнт щодо вхідних активацій обчислюється як повна згортка ядра з градієнтом вищого рівня, відповідним чином заповнена перед обертанням на 180 градусів. Ця математична формула зберігає

властивості просторової локальності, які роблять ЗНМ особливо ефективними для обробки даних зображень [20].

Приклад процесу показано на рисунку 2.5.

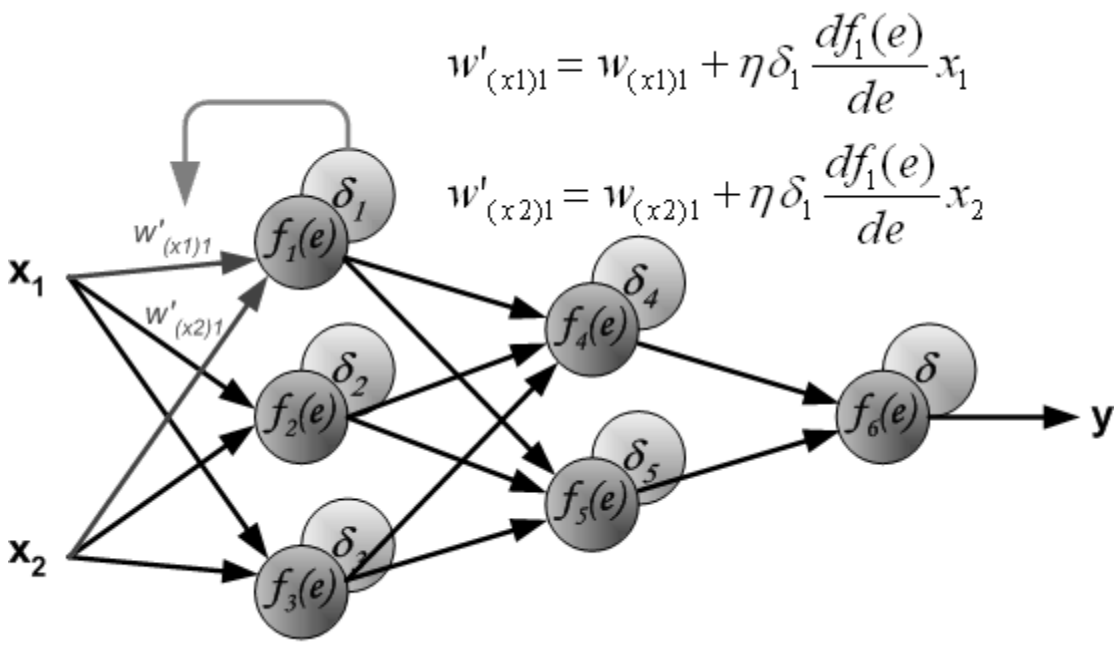


Рисунок 2.5. — Схема процесу зворотнього розповсюдження

При роботі з функціями активації під час зворотного поширення їх похідні відіграють вирішальну роль в обчисленні градієнта. Наприклад, у випадку активацій ReLU (Rectified Linear Unit) похідна є бінарною (0 або 1), що ефективно нівелює градієнт для від'ємних вхідних даних і зберігає його для додатних вхідних даних. Це поелементне перемноження градієнта на попередньому рівні та похідної функції активації гарантує, що потік градієнта враховує нелінійні перетворення, застосовані під час прямого проходження.

Для шарів агрегації, обчислення градієнта відбувається за різними схемами залежно від методу агрегацій. При максимізаційній агрегації градієнт зазвичай направляється тільки до нейрона, який дав максимальне значення під час прямого проходження, тоді як всі інші позиції отримують нульовий градієнт.

У міру проходження зворотного проходження через послідовні шари обчислювальний графік, пройдений під час прямого проходження, простежується у

зворотному порядку. На кожному шарі алгоритм обчислює локальні градієнти втрати щодо параметрів і вихідних даних цього шару, які потім слугують вхідними даними для подальшого обчислення градієнта в попередньому шарі. Це систематичне застосування ланцюгового правила дозволяє ефективно обчислювати градієнти в глибоких мережах без явного обчислення матриць Якобі, що було б надто обчислювально складним [11,20].

Накопичення таких градієнтів у шарах в кінцевому підсумку дає часткові похідні функції втрат щодо кожного параметра мережі, що піддається навчанню. Ці градієнти потім використовуються алгоритмом оптимізації (таким як стохастичний градієнтний спуск або його варіанти) для оновлення параметрів у напрямку, що мінімізує функцію втрат, зазвичай масштабовану гіперпараметром швидкості навчання.

У сучасних фреймворках глибокого навчання цей зворотний прохід зазвичай реалізується за допомогою систем автоматичного диференціювання, які будують обчислювальні графи під час прямого проходу, а потім ефективно проходять ці графіки у зворотному напрямку для обчислення необхідних градієнтів. Ця автоматизація дозволяє дослідникам і практикам зосередитися на проектуванні архітектури мережі, а не на математичних тонкощах обчислення градієнтів, при цьому все одно користуючись математичною строгістю базового алгоритму зворотного поширення.

Налаштування гіперпараметрів є систематичним процесом визначення оптимальних параметрів налаштування, які регулюють процес навчання та архітектурну структуру згорткових нейронних мереж. На відміну від параметрів моделі, які навчаються під час навчання за допомогою градієнтного спуску та зворотного поширення, гіперпараметри встановлюються екзогенно до процесу навчання і суттєво впливають на збіжність моделі, здатність до узагальнення та обчислювальну ефективність. Оптимізація гіперпараметрів становить проблему метанавчання, метою якої є виявлення такої конфігурації, яка забезпечує максимальну продуктивність на валідаційному наборі, зберігаючи при цьому обчислювальну доцільність.

Гіперпараметри включають в себе архітектурні параметри, параметри оптимізації та параметри регуляризації. Архітектурні гіперпараметри включають глибину мережі (кількість згорткових і повнозв'язних шарів), ширину кожного шару (кількість фільтрів або нейронів), розміри ядра, значення кроку, операції об'єднання та моделі зв'язності, такі як залишкові зв'язки в ResNets або модулі Inception в GoogLeNet. Гіперпараметри оптимізації контролюють динаміку навчання і включають швидкість навчання, коефіцієнти імпульсу, розмір пакета, вибір алгоритму оптимізації (SGD, Adam, RMSprop) та стратегії планування швидкості навчання [29].

Методологічні підходи до налаштування гіперпараметрів охоплюють спектр від вичерпних до інтелектуальних стратегій пошуку. Пошук по сітці є найпростішим підходом, який систематично оцінює всі комбінації в межах заздалегідь визначеного дискретного простору гіперпараметрів. Хоча цей метод є всеосяжним, він має експоненціальну обчислювальну складність відносно кількості гіперпараметрів, що робить його непрактичним для високорозмірних просторів. Випадковий пошук часто перевершує пошук по сітці, випадково відбираючи конфігурації гіперпараметрів із заданих розподілів, ефективно розподіляючи ресурси між вимірами різної важливості.

Більш складні підходи використовують замісні моделі для наближення показників ефективності та керування процесом пошуку. Байєсівські методи оптимізації, такі як підходи на основі гауссового процесу, послідовна конфігурація алгоритму на основі моделі (SMAC) та деревоподібні оцінювачі Парзена (TPE), будують імовірнісні моделі цільової функції та використовують функції оцінки для визначення перспективних конфігурацій гіперпараметрів для аналізу. Ці методи забезпечують баланс між пошуком невідомих оптимізацій та використанням відомих перспективних оптимізацій, демонструючи вищу ефективність вибірки порівняно з сітковим та випадковим пошуком [29].

Обчислювальні витрати, пов'язані з налаштуванням гіперпараметрів, мотивують розробку підходів до перенесення навчання, в яких гіперпараметри, оптимізовані для подібних галузей або архітектур, слугують початковими

параметрами для нових завдань. Оптимізація з попереднім запуском за допомогою раніше успішних конфігурацій може значно скоротити час пошуку, особливо при застосуванні попередньо навчених моделей до пов'язаних завдань, як в випадку даної магістерської роботи. Крім того, підходи до метанавчання спрямовані на вивчення попередніх гіперпараметрів у різних завданнях, що дозволяє швидко адаптуватися до нових проблем. Складні взаємозалежності між гіперпараметрами вимагають цілісних стратегій оптимізації, а не послідовного коригування окремих параметрів.

2.3. Обґрунтування критеріїв ефективності роботи системи

Оцінка систем виявлення місця паркування на основі згорткових нейронних мереж вимагає застосування математичних критеріїв, які дозволяють кількісно оцінити як точність класифікації для виявлення зайнятості місця, так і точність локалізації для визначення положення автомобіля. Парадигма тестування охоплює кілька методологічних підходів, призначених для оцінки можливостей моделі за такими параметрами, як точність, надійність та обчислювальна ефективність.

Основою ретельної оцінки моделі є належна підготовка набору даних, під час якої наявні анотовані візуальні дані систематично розподіляються на набори для навчання, валідації та тестування. Як правило, цей розподіл здійснюється за методом стратифікованої випадкової вибірки, щоб зберегти узгодженість розподілу класів у підмножинах. Основні критерії оцінки системи розпізнавання представлені на рисунку 2.6.

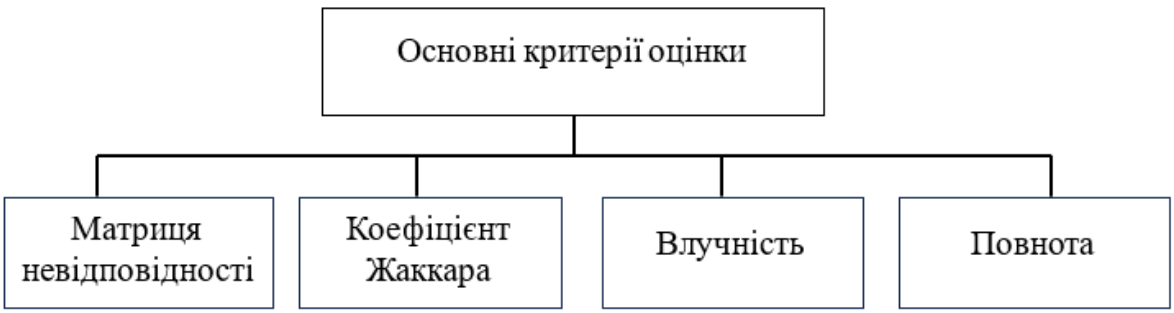


Рисунок 2.6 — Основні критерії оцінки

Навчальний набір сприяє оптимізації параметрів моделі, а валідаційний набір дозволяє налаштувати гіперпараметри та механізми раннього зупинення, щоб запобігти перенавчанню. Тестовий набір, що зарезервований виключно для остаточної оцінки, забезпечує об'єктивну оцінку можливостей узагальнення моделі на раніше небачених даних [30].

Влучність описує те, наскільки точним є передбачення, тобто те який відсоток передбачень є точним та визначається за формулою:

$$P = \frac{TP}{TP+FP} \tag{2.4}$$

де TP — істинно позитивні;

FP — хибно позитивні.

Повнота описує частку успішних передбачень з усіх передбачень та визначається формулою.

$$R = \frac{TP}{TP+FN} \tag{2.5}$$

де TP — істинно позитивні;

FN — хибно негативні.

Коефіцієнт Жаккара вимірює схожість між скінченними непорожніми наборами вибірки і визначається як площа перетину, поділена на площу об'єднання наборів вибірки, що і представлено на рисунку 2.7. Це є основною метрикою під час навчання нейронної мережі.

В розрізі розпізнавання об’єктів під час роботи нейронної мережі він вказує на перетин прогнозованих координат обмежувальної рамки з істинними координатами, більше значення вказує на те, що прогнозовані координати обмежувальної рамки дуже схожі на істинні координати.

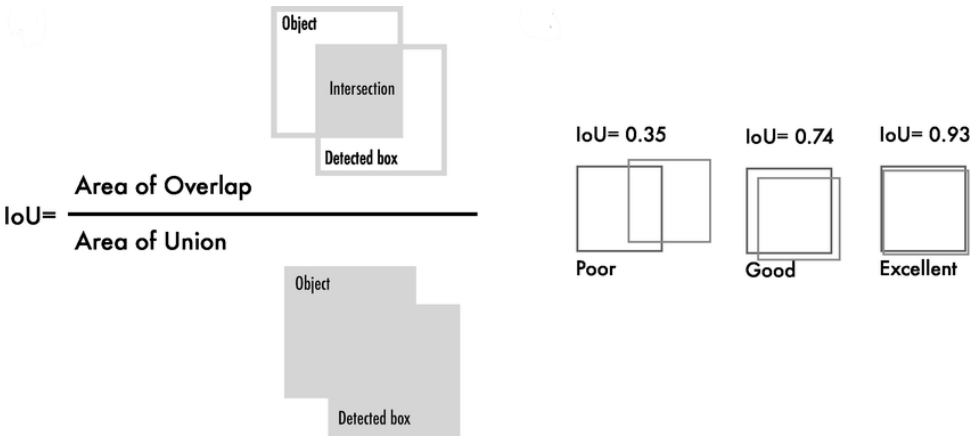


Рисунок 2.7. — Коефіцієнт Жаккара

Матриця невідповідності, також відома як матриця помилок, — це таблиця, яка дозволяє візуалізувати ефективність алгоритму контрольованої класифікації(рисунок 2.8). Це фундаментальний інструмент у галузі машинного навчання та статистики для оцінки моделей.

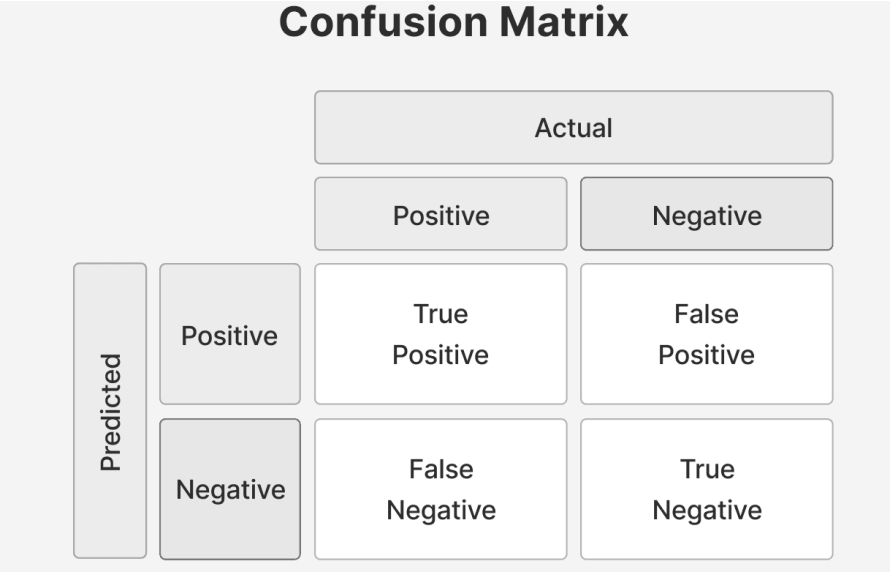


Рисунок 2.8 — Матриця невідповідності

Простіше кажучи, це таблиця співвідношень з двома вимірами: « Дійсне» та «Прогнозоване». Кожен вимір має набір дискретних класів. Порівнюючи фактичні цільові значення з прогнозами, зробленими класифікатором, матриця надає детальний розклад правильних класифікацій та типів допущених помилок [20,30].

Завдяки наявності відкритого програмного забезпечення існує безліч реалізацій систем виявлення місць паркування, що дозволяє проводити порівняльний аналіз різних підходів. Основними наборами даних у цій галузі є PKLot і CNRPark-EXT, які слугують стандартними еталонами для оцінки ефективності моделей у різних умовах.

За результатами розділу було обґрунтовано використання нейронної мережі YOLOv11 та реалізація методу орієнтованих обмежувальних рамок, який є оптимальним для застосування в системах моніторингу місць паркування. В якості оптимізатора було обрано оптимізатор Adam.

В цьому розділі було проведено огляд різних архітектур згорткових нейронних мереж, в результаті було обрано архітектуру YOLO. Після цього було сформовано алгоритм навчання мережі та визначено основні критерії оцінки якості роботи системи моніторингу.

3 РОЗРОБКА ЗАСОБІВ МОНІТОРИНГУ МІСЦЬ ПАРКУВАННЯ АВТОМОБІЛІВ

3.1 Вибір інструментів та засобів розробки системи моніторингу

Вибір мов програмування та пов'язаних з ними фреймворків є критично важливим фактором в екосистемі розробки штучного інтелекту. Для таких задач ефективно застосовувати Python, високорівневу інтерпретовану мову програмування, яка переважає в сферах штучного інтелекту та машинного навчання. Філософія її розробки ставить на перше місце читабельність та виразність коду завдяки чіткій синтаксичній структурі, яка дуже нагадує природну мову, що сприяє швидкому прототипуванню та ітеративній розробці — важливим характеристикам експериментального характеру досліджень у сфері штучного інтелекту. Динамічна система типізації Python, хоча й може спричиняти помилки під час виконання, забезпечує значну гнучкість на етапах дослідження під час розробки моделі. Взаємодія Python з мовами нижчого рівня за допомогою таких механізмів, як CPython API, дозволяє реалізовувати критичні для продуктивності операції в таких мовах, як C++, зберігаючи Python як основний інтерфейс, що створює оптимальний баланс між ефективністю розробки та обчислювальною продуктивністю.

Середовище розробки для систем штучного інтелекту принципово відрізняється від традиційних парадигм програмної інженерії через обчислювальну інтенсивність навчання моделей, складність конвеєрів даних та ітеративний характер експериментів. Інтегровані середовища розробки (IDE), такі як PyCharm Professional, Visual Studio Code зі спеціалізованими розширеннями або JupyterLab, надають очевидні переваги для розробки штучного інтелекту.

Visual Studio Code (VS Code) — це легкий, але потужний редактор вихідного коду, розроблений компанією Microsoft, який набув значної популярності в спільноті розробників штучного інтелекту. На відміну від свого більш комплексного аналога Visual Studio, VS Code використовує розширювану архітектуру, яка дозволяє розробникам налаштовувати середовище за допомогою багатої

екосистеми розширень. Його інтеграція з Python через офіційне розширення Python надає такі функції, як автозавершення коду IntelliSense, налагодження та тестування, спеціально оптимізовані для робочих процесів у галузі науки про дані. Інтеграція Jupyter notebook у VS Code забезпечує зручний перехід між інтерактивними експериментами та розробкою коду, дозволяючи розробникам виконувати кодові комірки, візуалізувати результати та вести документацію в єдиному інтерфейсі. Для розробки TensorFlow VS Code пропонує спеціалізовані розширення, що забезпечують підсвічування синтаксису, фрагменти коду для поширених архітектур моделей та інтегровані інструменти візуалізації тензорів. Інтегрований термінал редактора полегшує роботу з командним рядком без перемикання контексту, а інтегрована підтримка Git оптимізує робочі процеси контролю версій.

Фреймворк TensorFlow, розроблений Google, пропонує комплексний підхід до обчислювальних графіків з надійними можливостями розгортання на різних платформах, від периферійних пристроїв до розподілених хмарних інфраструктур. З іншого боку, PyTorch, що підтримується Meta, забезпечує більш динамічну обчислювальну парадигму завдяки своїй моделі активного виконання, що полегшує інтуїтивне налагодження та орієнтоване на дослідження розроблення.

Вибір правильних обчислювальних засобів має значний вплив на продуктивність моделі, ефективність розробки, масштабованість та експлуатаційні витрати. Особливо ретельно слід підійти до вибору обчислювального обладнання, оскільки воно є основою, на якій працюють алгоритми штучного інтелекту та забезпечують їхню функціональність.

Основним критерієм при виборі обчислювальних потужностей є відповідність можливостей апаратного забезпечення вимогам алгоритмів. Моделі глибокого навчання, особливо ті, що використовують згорткові нейронні мережі, мають значні обчислювальні вимоги, що вимагають спеціалізованого апаратного забезпечення. Графічні процесори (GPU) стали де-факто стандартом для навчання глибоких нейронних мереж завдяки своїм можливостям паралельної обробки, а платформа CUDA від NVIDIA забезпечує домінуючу програмну екосистему.

На основі інформації поданої вище, було обрано Python як мову програмування для роботи нейронної мережі та реалізації алгоритму, фреймворком обрано Pytorch, який є високопродуктивним, гнучким та популярним рішенням. Навчання та запуск нейронної мережі буде виконано на графічних прискорювачу NVIDIA, що завдяки потужній екосистемі платформи CUDA дозволяє ефективно проводити обчислення.

3.2 Формування комп’ютерної системи моніторингу місць паркування

Системи моніторингу паркувальних місць виступають як критично важлива складова інтелектуальних систем спостереження та пошуку, дозволяючи ефективно вирішувати широкий спектр завдань. Основними функціями такої системи є виявлення та класифікація об'єктів (зокрема, автомобілів та вільних місць), відстеження їхнього переміщення, ідентифікація, а також виявлення різних ситуацій, включаючи несанкціоноване паркування чи інші порушення. Ключовою перевагою використання нейронної мережі в цій системі є її здатність до автоматичного розпізнавання та аналізу відеоданих, що надходять з камер, що робить моніторинг високоточним та автономним. За допомогою камер, стратегічно розташованих у зоні спостереження, стає можливим безперервне відстеження заповненості паркінгу та переміщення транспортних засобів.

Типова архітектура такої комп'ютерної системи моніторингу включає кілька основних компонентів. Збір даних здійснюється за допомогою відеокамер (переважно цифрові, зокрема IP-камери), які передають кольорові або чорно-білі зображення високої якості. Обробка та зберігання отриманої інформації відбувається на відеореєстраторі або в даному випадку сервері, який є центральним елементом системи. Саме на сервері розміщується та виконується нейронна мережа, здійснюючи складний аналіз зображень для визначення статусу кожного місця паркування. Для візуального контролю та оперативного керування передбачені засоби спостереження моніторами. Крім того, для забезпечення надійної та безперебійної роботи системи необхідний ряд додаткових пристроїв, таких як джерела безперебійного живлення, маршрутизатори для забезпечення

зв'язку між компонентами, а також кабелі, кронштейни для монтажу та корпуси для захисту обладнання від зовнішніх впливів. Ця конфігурація забезпечує ефективну взаємодію всіх елементів для досягнення поставленої мети – точного та швидкого моніторингу зайнятості паркувальних місць. Для стабільної та швидкої роботи системи, що базується на нейронних мережах, необхідно використовувати комп’ютер з відповідними обчислювальними потужностями.

У комп'ютерній системі моніторингу місць паркування, що використовує нейронну мережу, IP-камера є ключовим елементом. Вона відрізняється наявністю вбудованого відеопроцесора, який дозволяє їй самостійно оцифровувати та попередньо обробляти відеосигнал. Згенерований IP-камерою цифровий сигнал прямує безпосередньо на сервер (відеореєстратор), де відбувається його подальша обробка, включаючи стиснення для зменшення надлишковості та архівацію. Глибина архівування, тобто термін зберігання відеозаписів, може варіюватися від кількох днів до кількох місяців, і залежить від кількості камер, якості зображення та ступеня стиснення інформації.

Системи моніторингу можна реалізувати в двох основних конфігураціях: з використанням реєстратора (сервера) та без використання реєстратора (безсерверна, або хмарна).

Хмарна система моніторингу паркувальних місць є безрекордерною конфігурацією, де основна обробка, зберігання та аналіз даних відбуваються поза межами локальної мережі, в хмарному сервісі. У цій моделі IP-камери отримують відеодані та передають їх через Інтернет безпосередньо до віддаленого хмарного сховища та обчислювальних потужностей. Це означає, що нейронна мережа для розпізнавання зайнятості місць паркування функціонує на потужностях провайдера хмарних послуг. Основні переваги такої системи полягають у мінімальних вимогах до локального обладнання (потрібні лише камери, комутатор та підключення до Інтернету) та високій масштабованості, оскільки ємність сховища та обчислювальна потужність легко адаптуються до кількості камер та обсягу даних. Проте це потребує налаштування програми моніторингу місць паркування

безпосередньо для роботи в хмарі, що може робити вартість функціювання системи дещо непередбачуваною.

Серверна система конфігурація, передбачає використання спеціалізованого відеореєстратора (NVR) або сервера, розташованого безпосередньо на об'єкті моніторингу.У цій системі IP-камери підключаються до локальної мережі, а відеосигнал надходить на локальний сервер для обробки та зберігання.

Нейронна мережа для аналізу зображень паркувальних місць та визначення їхнього статусу виконується на потужностях цього локального сервера або ж обробляється віддалено. Це вимагає від сервера значної обчислювальної потужності, часто з використанням графічних процесорів (GPU), оптимізованих для машинного навчання. Перевагами серверної системи є високий контроль над даними та їхня безпека, оскільки вся інформація зберігається локально. Витрати на таку систему є вищими на етапі початкового придбання обладнання (сервер, ліцензії), але відсутні регулярні платежі за хмарні послуги.

Для задач розроблюваної системи моніторингу місць паркування було обрано серверну конфігурацію.

Класична схема складається з реєстратора і відеокамер. Рекордер може бути багатоканальним, від 4 і більше каналів до 32, 64 і до 192 каналів. NVR реєстратори використовуються в системах IP-відеоспостереження. Для зберігання даних в архіві використовуємо твердотільний накопичувач ємністю від 512 Гб до 8 ТБ. Монітор використовується для локального перегляду, Інтернет — для віддаленого перегляду, IP-камера — мережевий комутатор, блок живлення камери та кабель «вита пара» для підключення. Для виконання задачі спостереження обрано відео камери типу Hikvision DS-2CD2043G2-IU. Для здійснення зв’язку із центральним сервером системи відео спостереження вибираємо медіа конвертер типу TP-LINK MC210CS 1GE. Для підключення відео камер спостереження до медіа конвертера будемо використовувати комутатор Hikvision DS-3E0310P-E/M.

Для роботи нейронної мережі використано графічний прискорювач Nvidia RTX 3070. В якості центрального процесора використано AMD R7 5700. Робота

системи лише на центральному процесорі можлива, але вкрай не рекомендована через значне зниження швидкодії.

Сформований склад комп’ютерної системи будемо використовувати для спостереження за зоною паркування.

3.3 Формування навчального набору даних та навчання нейронної мережі

Щоб усунути недоліки, властиві обмежувальним рамкам, вирівняним по осі, у точному відображенні повернутих об'єктів, було застосовано варіант архітектури YOLO під назвою орієтовані обернені рамки (OBB). Як описано в попередньому розділі, формат OBB доповнює традиційні просторові координати кутовими даними, що дозволяє більш точно відображати орієнтацію об'єкта. Така архітектура моделі вимагає спеціалізованого набору даних, де кожна анотація об'єкта включає не тільки його просторове розташування, але й кут повороту. Як наслідок, існуючий набір даних PKLot, який створений для стандартного виявлення об'єктів, був частково перемаркований. Оригінальні анотації, які визначали об'єкти виключно за допомогою горизонтальних прямокутників. За допомогою універсальної платформи анотацій Label Studio було переанотовано приблизно 120 вихідних зображень.

Для кожного цільового об'єкта на цих зображеннях було намальовано чотириточковий багатокутник, обернений під певним кутом. Цей процес ефективно закодував критичну інформацію про орієнтацію, перетворивши дані з простого формату виявлення на багатий формат з урахуванням орієнтації, придатний для навчання моделі орієнтованих обернених рамок. Отриманий набір даних, хоч і невеликий за розміром, надає високоточні анотації, необхідні для навчання моделі складних взаємозв'язків між зовнішніми ознаками та орієнтацією об'єкта, до того ж наявні засоби аугментації дозволяють дещо збільшити обсяг навчальних даних.

Приклад анотації набору даних в Label Studio представлено на рисунку 3.1.

Рисунок 3.1 — Приклад розмітки зображення

Виконавши розмітку зображення, необхідно експортувати дані розмітки в форматі YOLO OBB, що містить файли зображення та їх відповідної розмітки. Для безпосередньо навчання додатково формується файл формату YAML де визначаються основні поля:

— path — вказує шлях до кореневої папки набору даних;

— train — шлях до зображень та файлів розмітки для навчання;

— val — шлях до зображень та файлів розмітки для валідації;

— test — шлях до зображень та файлів розмітки для остаточного тестуваня;

— names —перелік класів для навчання та їх ідентифікатор.

Після формування набору даних, можна проводити донавчання нейронної мережі на наборові даних.

Першим етапом є ініціалізація моделі, в даному випадку YOLO, що представлено в лістингу 3.1.

Лістинг 3.1 — Фрагмент коду ініціалізації моделі

```
self.model = DetectionModel(cfg=model_cfg, ch=3, nc=num_classes)
self.loss_fn = v8DetectionLoss(self.model)
self.optimizer = torch.optim.Adam(self.model.parameters(), lr=0.001)
```

Цей фрагмент коду створює ієрархічну мережу вилучення ознак, що складається з багатьох згорткових шарів, залишкових з'єднань і механізмів просторового пірамідального об'єднання. Функція DetectionModel приймає вхідні дані RGB з трьома каналами і створює багатомасштабні карти ознак, які є необхідними для виявлення об'єктів різного розміру в зображенні. Одночасно ініціалізується функція v8DetectionLoss, яка реалізує формулювання функції втрат, що одночасно оптимізує три окремі цілі: точність локалізації за допомогою регресії обмежувальних рамок, ступінь впевненості в наявності об'єкта за допомогою бінарної перехресної ентропії та розрізнення між класами за допомогою категоріальної перехресної ентропії.

Конкретна топологія та поведінка цієї архітектури регулюються набором базових гіперпараметрів. Файл model_cfg визначає структурний план мережі, включаючи множники глибини та ширини, які визначають її обчислювальну потужність. Параметр num_classes визначає розмірність кінцевого класифікаційного шару, в даному випадку це клас автомобілів. Розмірність вхідного каналу (ch=3) є фіксованою, щоб відповідати стандартним RGB-зображенням.

Лістинг 3.2 — Фрагмент коду процесу прямого та оберненого розповсюдження

```
preds = self.model(batch['img'])
loss, loss_items = self.loss_fn(preds, batch['bboxes'])
self.optimizer.zero_grad()
loss.backward()
self.optimizer.step()
```

Під час прямого розповсюдження вектори вхідного зображення перетворюються на вектори передбачень. Модель генерує прогнози з різною просторовою роздільною здатністю, що дозволяє виявляти об'єкти в різних масштабах. Потім функція втрат обчислює зважену суму помилок локалізації та класифікації. Вона використовує стратегію розподілу завдань, яка дозволяє зіставляти аннотації реальних даних з прогнозними анкерами на основі їх геометричного вирівнювання та впевненості в класифікації. Розрахунок включає метрики Intersection over Union (IoU) для регресії обмежувальних рамок.

Операція zero_grad обнуляє накопичені градієнти з попередніх ітерацій. Функція backward ініціює автоматичне диференціювання, обчислюючи часткові похідні втрат щодо всіх параметрів, що підлягають навчанню, за допомогою правила ланцюжка.

Оновлення параметрів виконується оптимізатором Adam (Adaptive Moment Estimation), алгоритмом, що поєднує переваги двох інших популярних методів: Momentum і RMSprop. Adam підтримує дві експоненціально спадаючі ковзні середні для кожного параметра в моделі: перша є оцінкою середнього значення градієнтів (аналогічно до momentum), і друга є оцінкою градієнтів (аналогічно до RMSprop). Ці моменти коригуються з урахуванням їх ініціалізації на нулі. Правило оновлення параметрів формулюється як:

$$\theta = \theta - lr * \hat{m}t / (\sqrt{\hat{v}t} + \varepsilon) \quad (3.1)$$

де lr — швидкість навчання;

$\hat{m}_t$ — значення інерції;

$\hat{v}_t$ — значення поправки модуляції;

ε — невелика константа для чисельної стабільності.

Цей адаптивний механізм дозволяє Adam автоматично регулювати ефективну швидкість навчання для кожного параметра, що робить його особливо придатним для глибоких і часто зашумлених градієнтних спусків глибоких нейронних мереж.

Ефективність цього процесу навчання залежить від декількох критичних гіперпараметрів. Швидкість навчання (lr=0,001) контролює величину оновлень параметрів. Оптимізатор Adam додатково налаштовується за допомогою специфічних гіперпараметрів: експоненціальних коефіцієнтів спаду для оцінок моментів, β_1=0,9 для першого моменту і β_2=0,999 для другого моменту, а також числового терміну стабільності $\varepsilon=1e^{-8}$. Регуляризація спаду (weight_decay=$5e^{-4}$) ваги накладає обмеження на величину параметрів, щоб запобігати перенавчанню. У функції v8Loss зважування втрат обмежувальної рамки (box_gain), масштабування втрат об'єктності (obj_gain) та зважування втрат класифікації (cls_gain) визначають відносну важливість кожної цілі під час навчання.

Для підвищення узагальнюваності та надійності моделі використовується алгоритм збільшення обсягу даних. В лістингу 3.3 показано приклад реалізації горизонтального відзеркалення.

Лістинг 3.3 — Фрагмент коду аугментації набору даних

```
def horizontal_flip(self, image, bboxes):
    image = torch.flip(image, dims=[2])
    if len(bboxes) > 0:
        bboxes[:, 1] = 1.0 - bboxes[:, 1]
```

Цей метод застосовує операцію перевертання тензора вздовж ширини, одночасно інвертуючи нормалізовані горизонтальні координати обмежувальних рамок задля збереження просторової відповідності параметрів. Такі перетворення збільшують різноманіття набору даних і підвищують стабільність процесу навчання моделі.

Процес обробки даних контролюється набором гіперпараметрів, включаючи роздільну здатність вхідного зображення, яка впливає на детальність вилучення ознак, та розмір пакета (batch_size=8), який впливає на якість оцінки градієнта. Для навчання оригінальної моделі було застосовано розподільну здатність (img_size=640). Імовірності та значення розширення, такі як імовірність горизонтального перевертання (p=0,5), діапазони коливання кольору (brightness=0,2, contrast=0,2) та межі масштабування (scale_range=[0,8, 1,2]),

безпосередньо визначають ступінь варіативності даних, що вводиться під час навчання. Такий підхід дозволяє генерувати певну кількість додаткових, і що головне, надійних даних для навчання.

Сирий вихід нейронної мережі містить надлишкову кількість обмежувальних рамок. Для отримання чіткого та точного набору прогнозів реалізовується алгоритм не максимального придушення (NMS).

Лістинг 3.4 — Фрагмент реалізації не максимального придушення

```
def _nms(self, boxes, scores):
    sorted_indices = torch.argsort(scores, descending=True)
    while len(sorted_indices) > 0:
        current_idx = sorted_indices[0]
        keep.append(current_idx)
        ious = self._calculate_iou(current_box, remaining_boxes)
        keep_indices = torch.where(ious <= self.iou_thresh)[0]
        sorted_indices = sorted_indices[1:][keep_indices]
```

Цей алгоритм сортує виявлені об'єкти за рівнем достовірності, покроково вибирає об'єкт з найвищим рівнем достовірності і пригнічує всі інші об'єкти, у яких IoU перевищує заздалегідь визначений поріг. Процес проводить оптимізацію між точністю та відтворюваністю, усуваючи зайві обмежувальні рамки та зберігаючи найбільш достовірне представлення кожного об'єкта.

Функціювання алгоритму не максимального придушення регулюється гіперпараметрами постобробки. Поріг достовірності (conf_thresh=0,5) відфільтровує виявлення з низькою ймовірністю перед придушенням, а поріг IoU (iou_thresh=0,5) визначає рівень перекриття, необхідний для того, щоб виявлення вважалося надлишковим.

Загальна стратегія навчання управляється набором гіперпараметрів високого рівня, які визначають тривалість і темп процесу навчання. Загальна кількість епох (epochs=50) визначає тривалість навчання, а частота валідації (val_freq=10) контролює, як часто оцінюється продуктивність на валідаційному наборі даних. Раннє припинення реалізується за допомогою параметра терпіння (patience=10), який припиняє навчання, якщо показники валідації не покращуються протягом

заданої кількості епох. Тоді відбувається повернення до епохи, де показники валідації ще росли.

Графік швидкості навчання динамічно коригується для забезпечення стабільної конвергенції. Він включає фазу розігріву (warmup_epochs=3), яка поступово збільшує швидкість навчання від нуля до цільового значення, а потім графік косинусного відпалу, який періодично зменшує швидкість навчання. Крім того, для запобігання вибуху градієнта використовується поріг обмеження градієнту (max_norm=10.0), який обмежує норму L2 градієнтів параметрів під час зворотного поширення. Ця комплексна конфігурація забезпечує стабільний і ефективний режим навчання. На рисунку 3.2 відображається процес навчання і проміжні результати.

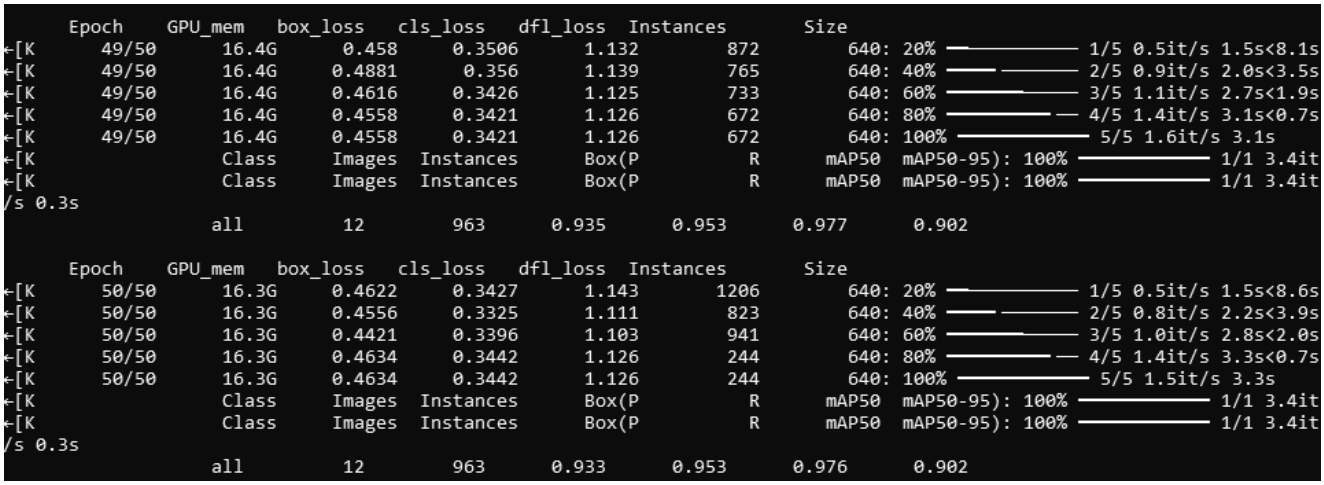


Рисунок 3.2 — Процес навчання мережі

В процесі навчання відображається узагальнена інформація про процес навчання, а саме вказується чисельність пройдених епох, час навчання та зміна функції втрат, що є одним з основних параметрів результативності навчання моделі.

З таблиці 3.1 видно, що прогрес в зниженні функції втрат наявний протягом усіх 50 епох. Основною метрикою є train/box_loss, що являє собою значення функції втрати визначеної нейронною мережею обмежувальної рамки відносно абсолютного значення з навчального набору даних. Саме це значення відповідає за те, чи правильно визначила нейронна мережа автомобіль на зображенні.

Таблиця 3.1 — Результати навчання мережі

Номер епохи	train/box_loss	train/cls_loss	train/dfl_loss	metrics/precision	metrics/recall	metrics/mAP50
1	1,2669	3,3050	1,47781	0,28895	0,54309	0,23868
10	0,64161	0,4990	1.21399	0,90099	0,96382	0,9536
20	0,62191	0,46431	1,17315	0,94005	0,84671	0,89404
30	0,56669	0,445	1,1721	0,941	0,9530	0,96872
40	0,5188	0,3909	1,1453	0,91261	0,96854	0,88127
50	0,46338	0,34418	1,1261	0,9329	0,953	0,9764

Після проведення процесу навчання було отримано нейронну мережу, яку можна використовувати в подальшому для реалізації системи моніторингу місць паркування.

3.4 Розробка програмних засобів системи моніторингу місць паркування

Для забезпечення функціювання системи необхідно реалізувати основні програмні блоки захоплення кадру, розпізнавання авто, аналізу стану місць паркування, візуалізації.

Блок захоплення кадру та розпізнавання відповідає за отримання кадрів зображення з різних джерел (відео, камера відеоспостереження, RTSP тощо). Після цього відбувається попередня обробка кадру з використанням засобів бібліотеки Ultralytics. Коли кадри передаються до моделі, фреймворк YOLO автоматично виконує всі необхідні етапи попередньої обробки, включаючи зміну розміру, нормалізацію та перетворення тензора. Далі відбувається процес розпізнавання та локалізації, що забезпечує роботу орієнтованих обмежувальних рамок з вісьмома координатними точками. Він реалізує обчислення центроїда для аналізу зайнятості.

Блок аналізу стану місця паркування реалізує метод зайнятості місця паркування на основі полігонів, де відбувається співставлення положення центроїда до полігона.

Блок візуалізації надає основні дані моніторингу в режимі реального часу з колірним кодуванням статусу зайнятості, накладенням аналітичних даних з FPS, коефіцієнтами зайнятості та статусом системи, а також напівпрозорими накладеннями зон для чіткої візуалізації. Схема роботи представлена на рисунку 3.3.

Рисунок 3.3 — Схема роботи системи моніторингу

Як вже було сказано вище, попередня обробка кадру виконується засобами бібліотеки Ultralytics, що включає в себе такі параметри:

— автоматичну зміну розміру кадру до очікуваного моделлю, що складає 640x640 пікселів;

— перетворення з формату BGR, що є за замовчуванням OpenCV у формат RGB для YOLO;

— нормалізація значень пікселів з діапазону 0-255 до діапазону 0-1;

— підтримка пакетної обробки для обробки кількох кадрів одночасно.

Визначення автомобіля в кадрі за допомогою моделі YOLO з можливістю налаштування функцій відстеження, що представлено в лістингу 3.5.

Лістинг 3.5 — Розпізнавання та відстеження авто

```
if self.tracking_enabled:
        results = self.model.track(
            frame,
            conf=self.conf_threshold,
            persist=True,
            tracker=self.tracker_config,
            verbose=False
        )
    else:
        results = self.model(
            frame,
            conf=self.conf_threshold,
            verbose=False
        )
    detections = self._extract_detection_data(results[0])
```

Коли відстеження ввімкнено, система зберігає ідентифікаційні дані транспортних засобів у декількох кадрах за допомогою алгоритму BoT-SORT, який передбачає рух транспортних засобів і зіставляє виявлення між кадрами. Це забезпечує послідовне відстеження транспортних засобів навіть тоді, коли об'єкти тимчасово закриті. У стандартному режимі виявлення кожен кадр обробляється незалежно, без відстеження. Цей метод повертає детальну інформацію про виявлення, включаючи обмежувальні рамки транспортних засобів, розраховані центральні точки для перевірки зайнятості, показники надійності. Цей

структурований формат даних дозволяє системі точно відстежувати положення та рух транспортних засобів на всій території паркінгу.

Для подальшого аналізу положення авто відносно місця паркування необхідно обчислити центроїд, код представлено в лістингу 3.6.

Лістинг 3.6 — Обчислення центроїда

```
if obb_points.numel() == 8:
        points = obb_points.reshape(4, 2)
        x_center = int(np.mean(points[:, 0]))
        y_center = int(np.mean(points[:, 1]))
        return x_center, y_center
    else:
        x_center = int((obb_points[0] + obb_points[2]) / 2)
        y_center = int((obb_points[1] + obb_points[3]) / 2)
        return x_center, y_center
```

Цей метод обчислює центральну точку орієнтованих обмежувальних рамок шляхом обробки восьми значень координат, що визначають чотири кути оберненого прямокутника. Алгоритм спочатку перевіряє, чи вхідні дані містять вісім координат (що вказує на орієнтовані обмежувальні рамки), а потім перетворює їх у чотири окремі вершини. Для кожної точки обчислюються середні координати x та y, щоб знайти геометричний центр.

Аналізатор зайнятості місця паркування реалізує виявлення зайнятості на основі полігонів за допомогою функції pointPolygonTest бібліотеки OpenCV, часове фільтрування з налаштованим порогом, встановленим за замовчуванням на три кадри. Ця функція приймає список центроїдів та список зон місць паркування. Реалізація представлена в лістингу 3.7

Лістинг 3.7 — Аналіз зайнятості місця

```
if zones is None:
        zones = self.parking_zones
    occupancy_status = [False] * len(zones)
    for i, zone in enumerate(zones):
        zone_points = zone['points']
        polygon_points = np.array(zone_points, dtype=np.int32).reshape((-1, 1, 2))
        for centroid in vehicle_centroids:
            distance = cv2.pointPolygonTest(polygon_points, centroid, False)
```

```
            if distance >= 0:
                occupancy_status[i] = True
                break
    return occupancy_status
```

Цей метод визначає зайнятість місць паркування, перевіряючи, чи потрапляють центральні точки транспортних засобів у межі визначених зон паркування. Для кожної зони паркування система перетворює координати зони в багатокутник і перевіряє центральну точку кожного транспортного засобу щодо цього багатокутника за допомогою функції геометричного тестування OpenCV. Якщо центральна точка транспортного засобу знаходиться всередині або на краю багатокутника місця паркування, ця зона позначається як зайнята. Алгоритм припиняє перевірку додаткових транспортних засобів для місця після підтвердження зайнятості конкретного, оптимізуючи продуктивність за рахунок зменшення непотрібних обчислень. Такий підхід дозволяє системі працювати з різними формами місць паркування, включаючи стандартні прямокутники, місця під кутом та нестандартні положення камери відеоспостереження відносно паркінга.

Функція оновлення стану місця паркування оновлює стан для кожного місця паркування з використанням методу часової фільтрації для забезпечення точного визначення стану місця паркування. Для кожного місця система відстежує стан зайнятості протягом декількох послідовних кадрів, при цьому за замовчуванням зберігаються останні три кадри. Алгоритм використовує голосування більшістю для визначення остаточного стану зайнятості — якщо більше половини останніх кадрів показують, що місце зайняте, воно вважається стабільно зайнятим.

Лістинг 3.6 — Оновлення стану паркувального місця

```
current_occupancy = self.check_occupancy(vehicle_centroids)
    for zone_id, occupied in enumerate(current_occupancy):
        zone_key = f"zone_{zone_id}"
        self.occupancy_history[zone_key].append(occupied)
        if len(self.occupancy_history[zone_key]) > self.occupancy_threshold:
            self.occupancy_history[zone_key].pop(0)
    stable_occupancy = []
```

```
for zone_id in range(len(self.parking_zones)):
    zone_key = f"zone_{zone_id}"
    history = self.occupancy_history[zone_key]
    if len(history) >= self.occupancy_threshold:
        occupied_count = sum(history)
        stable_occupied = occupied_count > len(history)
    else:
        stable_occupied = current_occupancy[zone_id]
    stable_occupancy.append(stable_occupied)
```

Змінна current_occupancy викликає метод для перевірки, які зони наразі зайняті, на основі положень транспортних засобів (центроїдів).Повертає список, де кожен елемент показує, чи зайнята зона. Для кожної незайнятої зони, якщо доступно достатньо історичних даних які більше порогу occupancy_threshold, використовується голосування (більше половини останніх кадрів показують зайнятість). Якщо немає достатньо даних про стан місця, використовується значення розпізнавання поточного кадру.

Цей механізм фільтрації ефективно усуває тимчасові помилкові виявлення, спричинені тінями, відблисками або короткочасними рухами транспортних засобів. Система також обчислює статистику в режимі реального часу, включаючи загальну кількість зайнятих місць, вільних місць та загальний коефіцієнт зайнятості, надаючи цінні оперативні дані для управління паркувальними майданчиками. Налаштовуваний поріг чутливості дозволяє здійснювати регулювання на основі конкретних умов навколишнього середовища та вимог до продуктивності.

Виведення результатів забезпечує детальний зворотний зв'язок у режимі реального часу за допомогою інтегрованої інформаційної вікна, що відображає зони паркування з кольоровим кодуванням статусу зайнятості, виявлення транспортних засобів з інформацією про відстеження, статистику зайнятості та показники продуктивності системи. Для кожної зони система обчислює геометричну центральну точку для розміщення ідентифікаційних номерів зон та міток про зайнятість. Система колірного кодування використовує зелений колір для зайнятих місць і червоний для вільних, забезпечуючи миттєвий візуальний зворотний зв'язок про наявність вільних місць для паркування та правильність роботи системи в

цілому. Прямокутники на тлі за ідентифікаторами зон забезпечують читабельність тексту на тлі різних кольорів і текстур. Цей комплексний підхід до візуалізації дозволяє операторам швидко оцінювати стан місць паркування та визначати конкретні місця, що їх цікавлять.

Також окремим пунктом є виділення координат місць паркування автомобілів. Для цього був створений графічний інструмент для розмітки використовуючи Tkinter та OpenCV, що підтримують кілька типів зон, включаючи стандартні, розташовані під кутом. Інструмент реалізує масштабування координат між розмірами зображення та оригінального кадру відео для точного визначення зони. Розмітку зон можливо автоматизувати за допомогою нейронної мережі, але це потребує навчання мережі. Також слід враховувати, що часто розмітки може не бути взагалі або вона може бути не повною чи пошкодженою, що додатково може створити більше помилкових спрацьовувань.

У даному розділі магістерської роботи були обрані мова та середовище програмування, підготовано набір даних, проведено навчання нейронної мережі, розроблено програмний засіб для реалізації запропонованого підходу для моніторингу місць паркування автомобілів.

4 ТЕСТУВАННЯ ТА ОЦІНКА ТОЧНОСТІ РОБОТИ СИСТЕМИ

4.1 Тестування програми моніторингу стану місць паркування

Перед запуском програми рекомендовано створити та активувати ізольоване програмне середовище(virtualenv), та встановити основні залежності командою pip install ultralytics opencv-python numpy pillow. Це необхідно щоб усунути потенційні помилки при несумісності версій бібліотек які необхідні програмі для роботи та тих, які потенційно вже можуть бути на користувацькому комп'ютері. Також в залежності від виробника застосовуваного графічного прискорювача необхідно встановити правильну версію фреймворку PyTorch Після цього перейшовши в директорію з основним файлом проекту, ввести в консолі команду «python run_parking_system.py --define-zones»

Після цього буде завантажено програму для розмітки місць паркування, інтерфейс представлено на рисунку 4.1

Рисунок 4.1 — Стартове вікно розмітки місць паркування

Далі необхідно завантажити зображення з паркувальними місцями, які необхідно розмітити. Для цього необхідно натиснути кнопку «Завантажити зображення» та вибрати відповідне зображення з місцями паркування. Обов'язково необхідно впевнитись, що розширення та співвідношення сторін зображення точно відповідає розширенню відеофайлу або відеопотокую. Недотримання цієї вимоги призведе до некоректного визначення координат та помилок у роботі алгоритму.

Процес розмітки здійснюється шляхом побудови рамки кожного паркувального місця. Для цього достатньо послідовно виділити чотири ключові точки. Після успішного визначення чотирьох точок рамка автоматично зафіксується. У разі помилки при розмітці конкретної зони (наприклад, неправильно вибрана точка), можна скористатись функцією «Очистити останню зону, щоб забезпечити максимальну точність розмітки. Після завершення розмітки

всіх необхідних місць, система готова до калібрування та подальшого моніторингу. Приклад вигляду вже розмічених місць представлено на рисунку 4.2.

Рисунок 4.2 — Розмічені місця для паркування

Після виконання розмітки необхідно натиснути кнопку збереження результатів. При успішному збереженні буде виведено відповідне повідомлення на екран.

Зважаючи на те, що в нас наявні відповідні місця, наступним кроком буде безпосередньо запуск системи розпізнавання, що виконується командою «python run_parking_system.py --model models/your_obb_model.pt --zones config/parking_zones.json --source» де:

— model вказує на шлях до нейронної мережі, що буде виконувати розпізнавання;

— zones вказує на файл з розміченими місцями для паркування;

— source вказує на джерело кадрів (відео, веб-камера).

Після цього на екран буде виведено інтерфейс головного вікна з вказанням кількості вільних та зайнятих місць, рівня зайнятості в відсотках, кількості

оброблюваних кадрів на секунду. Для завершення роботи програми необхідно натиснути клавішу q. Вигляд основного інтерфейсу представлено на рисунку 4.3.

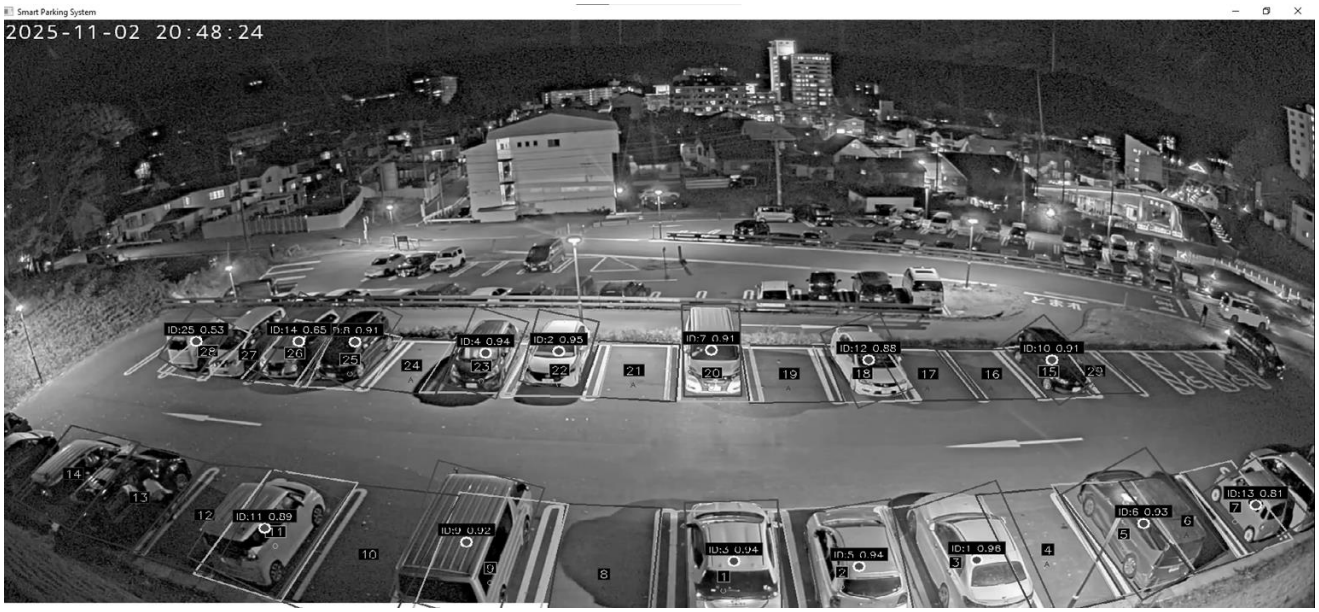


Рисунок 4.3 — Приклад роботи програми розпізнавання

4.2 Перевірка точності моніторингу місць паркування

Для проведення оцінки роботи системи моніторингу місць паркування було використано 20 відеофайлів. В відкритому доступі подібні набори даних відсутні, тому відеофрагменти було отримано з прямих ефірів паркінгів на YouTube. Це дозволяє проводити тестування на даних, які дійсно можуть траплятись в житті. Для перевірки було обрано відеофайли різною порою доби, але без дощу та снігу, такі погодні умови викликають велику кількість помилкових спрацьовувань. Для тестування буде застосовуватись розроблена система з застосуванням орієнтованих обмежувальних рамок та класична мережа з рамками зафіксованими за осями.

Основними показниками оцінювання будуть точність розпізнавання авто та точність визначення стану зайнятості місця

В результаті тестування середня точність розпізнавання авто визначається за формулою 2.4 та становить 86% для мережі з застосуванням орієнтованих обернених рамок та 89% для класичної мережі. Така різниця в першу чергу викликана малою кількістю навчальних даних для мережі орієнтованих обернених

рамок, яка початково була навчена для розпізнавання супутникових знімків, тому розпізнавання автомобілів поблизу викликає певні проблеми.

На рисунку 4.4 представлено графік точності для трьох сценаріїв визначення місця з застосуванням класичної моделі та моделі орієнтованих обернених рамок. Середня точність визначення стану зайнятості на тестових даних становить 88% для мережі з застосуванням орієнтованих обернених рамок та 75% для класичної мережі. Різниця склала 13%, і викликана в першу чергу навчанням з застосуванням набору даних з паркінгів, що завдяки застосуванню орієнтованих обернених рамок дозволило виконувати розпізнавання більш точно на місцях паркування.

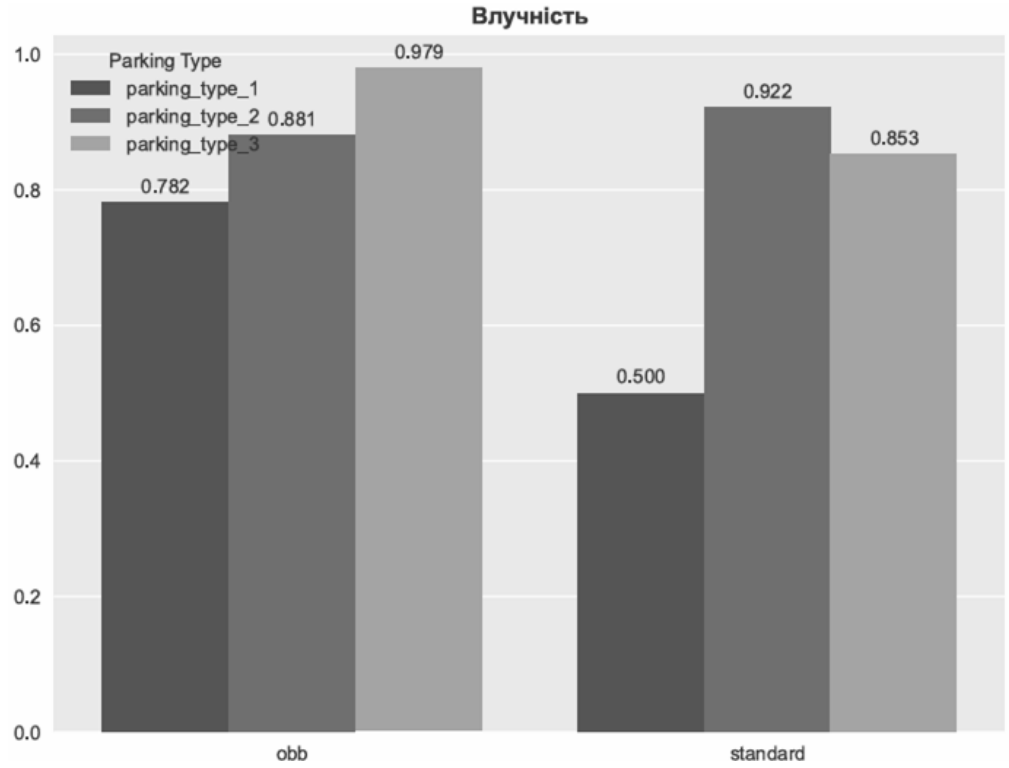


Рисунок 4.4 — Значення влучності для типів розпізнавання

Також було проведено порівняння точності роботи з аналогічною програмою з відкритим кодом Car Parking Space Detection. Середня точність визначення місця склала 71%. Основні помилкові спрацювання при роботі програми-аналога були з високим розташуванням камери, де нейронна мережа не могла провести розпізнавання та при повернутих відносно камери місцях паркування, де іноді

відбувалось помилкове визначення зайнятості місця. Результат порівняння представлено на рисунку 4.5.

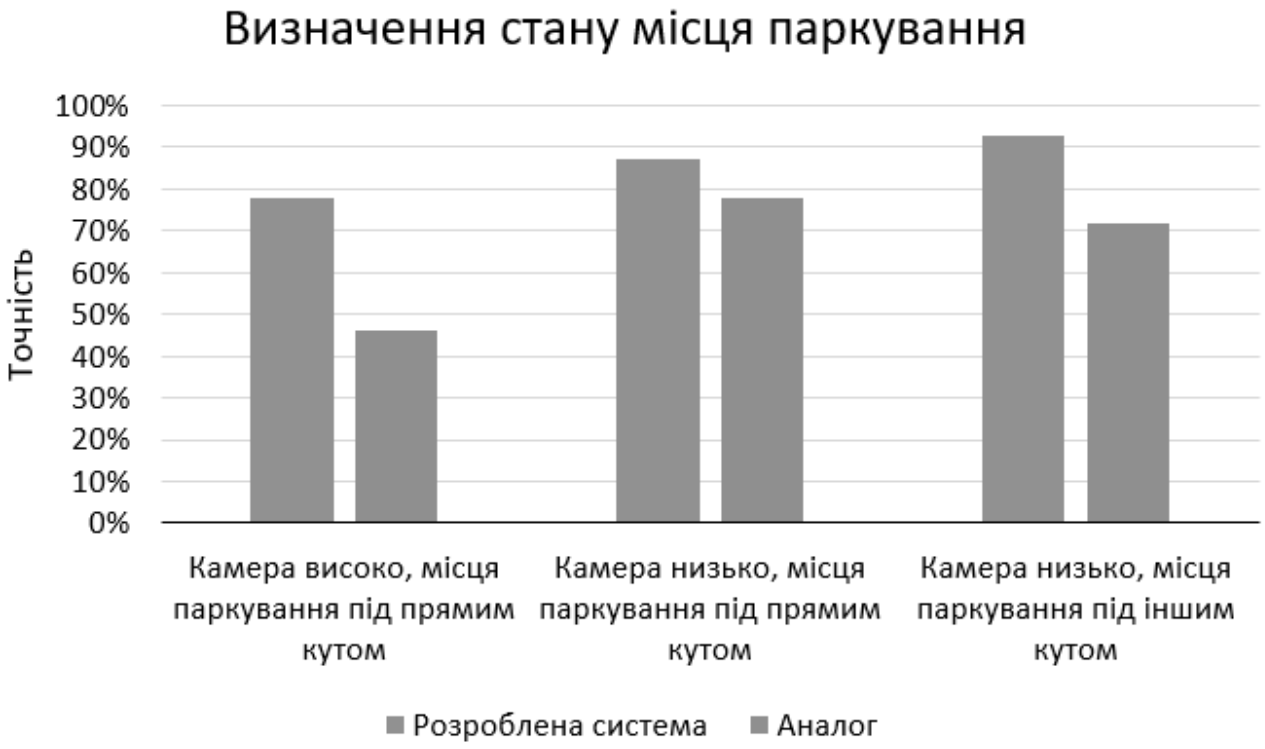


Рисунок 4.5 — Графік порівняння роботи програм моніторингу місць паркування

З тестування можна зробити кілька основних висновків, головним з яких є пряма пропорційність висоти камери відеоспостереження та точності розпізнавання. Високе положення камери дозволяє знизити перекриття автомобілів між собою, що робить їх розпізнавання точніше та робить розпізнавання зайнятого місця також точніше. Таке положення камери додатковим чином вирівнює перспективу, та дає можливість більш точного розпізнавання. Також розпізнавання в нічний час доби відбувалось з нижчою точністю, проте це також частково вирішується подальшим донавчанням нейронної мережі.

Можна зазначити, що подальше донавчання нейронної мережі є дуже перспективним, адже навчальний набір даних склав близько 500 зображень з врахуванням штучного доповнення навчальних даних. Збільшення набору даних та його різноманіття може значним чином підвищити точність роботи системи в різноманітних умовах.

В даному розділі було проведено тестування працездатності та оцінка ефективності роботи системи моніторингу місць паркування автомобілів.

5 РОЗРАХУНОК ЕКОНОМІЧНОЇ ДОЦІЛЬНОСТІ СТВОРЕННЯ ПРОГРАМИ МОНІТОРИНГУ МІСЦЬ ПАРКУВАННЯ

5.1 Комерційний та технологічний аудит науково-технічної розробки

Метою даного розділу є проведення технологічного аудиту розробки програми моніторингу місць паркування автомобілів. Метою роботи є підвищення точності моніторингу місць паркування автомобілів з застосуванням нейронної мережі. Особливістю розробки є застосування методу орієнтованих обмежувальних рамок для підвищення точності визначення об’єкта на зображенні.

Для проведення комерційного та технологічного аудиту було залучено 3-х незалежних експертів. Оцінювання науково-технічного рівня розробки та її комерційного потенціалу здійснено із застосуванням п’ятибальної системи оцінювання за 12-ма критеріями, у відповідності із табл. 4.1.

Таблиця 5.1 — Критерії оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогі	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 5.1

5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово—промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10— ти років	Термін реалізації ідеї від 3— х до 5— ти років. Термін окупності інвестицій більше 5— ти років	Термін реалізації ідеї менше 3— х років. Термін окупності інвестицій від 3— х до 5— ти років	Термін реалізації ідеї менше 3— х років. Термін окупності інвестицій менше 3— х років

Продовження таблиці 5.1

12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Усі дані по кожному параметру занесено в таблиці 5.2

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії оцінювання	ПІБ експертів		
	Експерт 1	Експерт 2	Експерт 3
	Бали		
Технічна здійсненність концепції	4	4	4
Наявність аналогів на ринку	3	3	3
Цінова політика	3	4	3
Технічні та споживчі властивості виробу	3	4	3
Експлуатаційні витрати	4	3	3
Ринок збуту	4	4	4
Конкурентоспроможність	2	3	3
Фахівці з технічної і комерційної реалізації	4	4	4
Фінансування	4	4	3
Матеріально-технічна база	4	4	4
Термін реалізації ідеї	4	4	4
Супровідна документація	4	4	4
Сума	43	45	42
Середньоарифметична сума балів	(43+45+42) / 3 = 43		

За даними таблиці 5.2 можна зробити висновок щодо рівня комерційного потенціалу даної розробки. Для цього доцільно скористатись рекомендаціями, наведеними в таблиці 5.3.

Таблиця 5.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів, розрахована на основі висновків	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Як видно з таблиці, рівень комерційного потенціалу розроблюваного нового програмного продукту є високим, що досягається за рахунок використання вдосконаленого методу визначення розпізнаваного об’єкта на зображені шляхом застосування методу орієнтованих обмежувальних рамок.

5.2 Прогнозування витрат на виконання дослідно-конструкторської роботи з розробки програми моніторингу місць паркування

5.2.1 Основна заробітна плата розробників, які брати участь в розробці програми моніторингу місць паркування

Основна заробітна плата розробників, яка розраховується за формулою:

$$З_о = \frac{M}{T_р} \cdot t \tag{5.1}$$

де M — місячний посадовий оклад конкретного розробника (дослідника), грн.;

$T_р$ — число робочих днів за місяць, 20 днів;

t — число днів роботи розробника (дослідника).

Результати розрахунків зведено до таблиці 5.4.

Таблиця 5.4 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Керівник проекту	45000	2250	36	81000
Програміст	39500	1975	36	71100
Всього				152100

Так як в даному випадку розробляється програмний продукт, то розробник виступає одночасно і основним робітником, і тестувальником розроблюваного програмного продукту.

5.2.2 Додаткова заробітна плата розробників, які брати участь в розробці програми моніторингу місць паркування

Додаткову заробітну плату прийнято розраховувати як 12 % від основної заробітної плати розробників та робітників:

$$Зд = Зо \cdot \frac{Ндод.}{100\%}, \quad (5.2)$$

де Ндод. — норма нарахування додаткової заробітної плати.

$$Зд = (152100 \cdot \frac{12\ \%}{100\%}) = 18252\ (грн.)$$

5.2.3 Нарахування на заробітну плату розробників.

Згідно діючого законодавства нарахування на заробітну плату складають 22 % від суми основної та додаткової заробітної плати.

$$Нз = (Зо + Зд) \cdot \frac{Нзп}{100\%}, \quad (5.3)$$

де Нзп. — норма нарахування на заробітну плату.

$$Нз = (152100 + 18252) \cdot \frac{22\ \%}{100\%} = 37477{,}44\ (грн.)$$

Оскільки для розробки програми моніторингу місць паркування не потрібно витрачати матеріали та комплектуючі, то витрати на матеріали і комплектуючі дорівнюють нулю.

5.2.4 Амортизація обладнання, яке використовувалось для проведення розробки програми.

Амортизація обладнання, що використовувалось для розробки в спрощеному вигляді розраховується за формулою:

$$A = \frac{Ц}{Тв} \cdot \frac{t_{вик}}{12}, \quad (5.4)$$

де Ц — балансова вартість обладнання, грн.;

Т — термін корисного використання обладнання згідно податкового законодавства, років;

$t_{вик}$ — термін використання під час розробки, місяців.

Розрахуємо, для прикладу, амортизаційні витрати на комп'ютер балансова вартість якого становить 59499 грн., термін його корисного використання згідно податкового законодавства – 2 роки, а термін його фактичного використання – 1,66 міс.

$$А_{обл} = \frac{59499}{2} \times \frac{1{,}66}{12} = 4115{,}278\ (грн.)$$

Аналогічно визначаємо амортизаційні витрати на інше обладнання та приміщення. Розрахунки заносимо до таблиці 5.5. Так як вартість ліцензійної ОС та спеціалізованих ліцензійних нематеріальних ресурсів менше 20000 грн, то даний

нематеріальний актив не амортизується, а його вартість включається у вартість розробки повністю, $В_{нем.ак.}$ = 12700 грн.

Таблиця 5.5 — Амортизаційні відрахування на матеріальні та нематеріальні ресурси для розробників

Найменування обладнання	Балансова вартість, грн.	Строк корисного використання, років	Термін використання обладнання, місяців	Амортиза-ційні відрахуван-ня, грн.
Комп'ютер та комп'ютерна периферія	59499	2	1,66	4115,278
Офісне обладнання (меблі)	32300	4	1,66	1114,35
Приміщення	1900000	20	1,66	13100
Всього				18329,62

Тарифи на електроенергію для непобутових споживачів (промислових підприємств) відрізняються від тарифів на електроенергію для населення. При цьому тарифи на розподіл електроенергії у різних постачальників (енергорозподільних компаній), будуть різними. Крім того, розмір тарифу залежить від класу напруги (1-й або 2-й клас). Тарифи на розподіл електроенергії для всіх енергорозподільних компаній встановлює Національна комісія з регулювання енергетики і комунальних послуг (НКРЕКП). Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п, \quad (5.5)$$

де В — вартість 1 кВт-години електроенергії для 1 класу підприємства, В = 12,40 грн./кВт;

П — встановлена потужність обладнання, кВт. П = 0,4 кВт;

Ф — фактична кількість годин роботи обладнання, годин;

$К_п$ — коефіцієнт використання потужності, $К_п$ = 0,9.

$$В_е = 0,9 \cdot 0,4 \cdot 8 \cdot 36 \cdot 12,4 = 1285,632 \text{ (грн.)}$$

5.2.5 Інші витрати та загальновиробничі витрати.

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками. Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників:

$$I_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100\%}, \quad (5.6)$$

де $Н_{ів}$ — норма нарахування за статтею «Інші витрати».

$$І_в = 152100 * 70\% / 100\% = 106470 \text{ (грн.)}$$

До статті «Накладні (загальновиробничі) витрати» належать: витрати, пов’язані з управлінням організацією; витрати на винахідництво та раціоналізацію; витрати на підготовку (перепідготовку) та навчання кадрів; витрати, пов’язані з набором робочої сили; витрати на оплату послуг банків; витрати, пов’язані з освоєнням виробництва продукції; витрати на науково-технічну інформацію та рекламу та ін. Витрати за статтею «Накладні (загальновиробничі) витрати» розраховуються як 100…150% від суми основної заробітної плати дослідників:

$$Н_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100\%}, \quad (5.7)$$

де $Н_{нзв}$ — норма нарахування за статтею «Накладні (загальновиробничі) витрати».

$$Н_{нзв} = 152100 * 120 \% / 100 \% = 182500 \text{ (грн.)}$$

5.2.6 Витрати на проведення науково-технічної роботи з розробки програми моніторингу.

Сума всіх попередніх статей витрат дає загальні витрати на проведення науково-дослідної роботи:

$$В_{заг} = 152100 + 18252 + 182500 + 106470 + 1285,63 + 34918,40 + 12700 + 18329,62 = 529114,69 \text{ (грн.)}$$

5.2.7 Розрахунок загальних витрат на науково-технічну роботу з розробки програми моніторингу та оформлення її результатів.

Загальні витрати на завершення науково-дослідної (науково-технічної) роботи та оформлення її результатів розраховуються за формулою:

$$ЗВ = \frac{В_{заг}}{\eta} \text{ (грн.)}, \quad (5.8)$$

де η — коефіцієнт, який характеризує етап (стадію) виконання науково-технічної роботи.

Так, якщо науково-технічна розробка знаходиться на стадії: науково-дослідних робіт, то η=0,1; технічного проектування, то η=0,2; розробки конструкторської документації, то η=0,3; розробки технологій, то η=0,4; розробки дослідного зразка, то η=0,5; розробки промислового зразка, то η=0,7; впровадження, то η=0,9. Оберемо η = 0,5, так як розробка, на даний момент, знаходиться на стадії дослідного зразка:

$$ЗВ = 529114,69 / 0,5 = 1058229,38 \text{ (грн.)}$$

5.3 Розрахунок економічної ефективності розробки програми моніторингу місць паркування за її можливої комерціалізації потенційним інвестором

В ринкових умовах узагальнювальним позитивним результатом, що його може отримати потенційний інвестор від можливого впровадження результатів тієї чи іншої науково-технічної розробки, є збільшення у потенційного інвестора величини чистого прибутку. Саме зростання чистого прибутку забезпечить потенційному інвестору надходження додаткових коштів, дозволить покращити фінансові результати його діяльності, підвищить конкурентоспроможність та може позитивно вплинути на ухвалення рішення щодо комерціалізації цієї розробки.

Для того, щоб розрахувати можливе зростання чистого прибутку у потенційного інвестора від можливого впровадження науково-технічної розробки необхідно:

— а вказати, з якого часу можуть бути впроваджені результати науково-технічної розробки;

— зазначити, протягом скількох років після впровадження цієї науково-технічної розробки очікуються основні позитивні результати для потенційного інвестора (наприклад, протягом 3-х років після її впровадження);

— кількісно оцінити величину існуючого та майбутнього попиту на цю або аналогічні чи подібні науково-технічні розробки та назвати основних суб’єктів (зацікавлених осіб) цього попиту;

— визначити ціну реалізації на ринку науково-технічних розробок з аналогічними чи подібними функціями.

При розрахунку економічної ефективності потрібно обов’язково враховувати зміну вартості грошей у часі, оскільки від вкладення інвестицій до отримання прибутку минає чимало часу. При оцінюванні ефективності інноваційних проектів передбачається розрахунок таких важливих показників:

— абсолютного економічного ефекту (чистого дисконтованого доходу);

— внутрішньої економічної дохідності (внутрішньої норми дохідності);

— терміну окупності (дисконтованого терміну окупності).

Аналізуючи напрямки проведення науково-технічних розробок, розрахунок економічної ефективності науково-технічної розробки за її можливої комерціалізації потенційним інвестором можна об’єднати, враховуючи визначені ситуації з відповідними умовами.

5.3.1 Розробка програми моніторингу місць паркування для використання масовим споживачем.

В цьому випадку майбутній економічний ефект буде формуватися на основі таких даних:

$$\Delta П_i = (\pm \Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}), \quad (5.9)$$

де $\pm\Delta Ц_о$ — зміна вартості програмного продукту (зростання чи зниження) від впровадження результатів науково-технічної розробки в аналізовані періоди часу;

N — кількість споживачів які використовували аналогічний продукт у році до впровадження результатів нової науково-технічної розробки;

$Ц_о$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки, $Ц_о = Ц_б \pm \Delta Ц_о$;

Цб — вартість програмного продукту у році до впровадження результатів розробки;

ΔN — збільшення кількості споживачів продукту, в аналізовані періоди часу, від покращення його певних характеристик;

λ — коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт λ = 0,8333.

р — коефіцієнт, який враховує рентабельність продукту;

ϑ — ставка податку на прибуток, у 2025 році ϑ =18%.

Припустимо, що при прогнозованій ціні 50000 грн. за одиницю виробу, термін збільшення прибутку складе 3 роки. Кількість одиниць реалізованої продукції збільшиться: протягом першого року – на 250 шт., протягом другого року

– на 300 шт., протягом третього року на 250 шт. До моменту впровадження результатів наукової розробки реалізації продукту не було:

$\Delta П_1 = (0 * 100 + 50000 * 250 * 0{,}8333 * 0{,}32) * (1 - 0{,}18) = 2733224$ грн.

$\Delta П_2 = (0 * 100 + 50000 * (250+300) * 0{,}8333 * 0{,}32) * (1 - 0{,}18) = 6013092{,}8$ грн.

$\Delta П_3 = (0 * 100 + 50000 * (250+300+250) * 0{,}8333 * 0{,}32) * (1 - 0{,}18) = 8746316{,}8$ грн.

Отже, комерційний ефект від реалізації результатів розробки за три роки складе 17492633,6 грн.

5.3.2 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розраховуємо приведену вартість збільшення всіх чистих прибутків ПП, що їх може отримати потенційний інвестор від можливого впровадження та комерціалізації науково-технічної розробки:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \qquad (5.10)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої науково-дослідної (науково-технічної) роботи, грн;

Т — період часу, протягом якою виявляються результати впровадженої науково-дослідної (науково-технічної) роботи, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, $\tau = 0{,}05…0{,}15$;

t — період часу (в роках).

Збільшення прибутку ми отримаємо, починаючи з першого року:

ПП = (2733224 /(1+0,09)[1])+(6013092,8/(1+0,09)[2])+(8746316,8/
/(1+0,09)[3]) = 2507544,5 + 5061099 + 6753764 = 14322407,5 грн.

Далі розраховують величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки. Для цього можна використати формулу:

$$PV = k_{інв} * ЗВ, \quad (5.11)$$

де $k_{інв}$ — коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію, це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай $k_{інв}$=2...5, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 3 * 1058229,38 = 3174688,14 \text{ грн.}$$

Тоді абсолютний економічний ефект Еабс або чистий приведений дохід (NPV, Net Present Value) для потенційного інвестора від можливого впровадження та комерціалізації науково-технічної розробки становитиме:

$$Е_{абс} = ПП – PV, \quad (5.12)$$

$$Е_{абс} = 14322407,5 – 3174688,14 = 11147719,36 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів даної науково-дослідної (науково-технічної) роботи може бути доцільним.

Для остаточного прийняття рішення з цього питання необхідно розра-хувати внутрішню економічну дохідність або показник внутрішньої норми дохідності

(IRR, Internal Rate of Return) вкладених інвестицій та порівняти її з так званою бар'єрною ставкою дисконтування, яка визначає ту мінімальну внутрішню економічну дохідність, нижче якої інвестиції в будь-яку науково-технічну розробку вкладати буде економічно недоцільно.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього використаємо формулу:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \tag{5.13}$$

де $T_ж$ — життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + 11147719{,}36 / 3174688{,}1} - 1 = 0{,}652$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \tag{5.14}$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; в 2025 році в Україні d = (0,09…0,15);

f — показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,5).

$$\tau_{min} = 0{,}15 + 0{,}15 = 0{,}3 .$$

Так як $E_в > \tau_{min}$, то інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}, \tag{5.15}$$

$$Т_{ок} = 1 / 0,652 = 1,53 \text{ р.}$$

Оскільки $T_{ок}$ < 3-х років, а саме термін окупності рівний 1,53 роки, то фінансування даної наукової розробки є доцільним.

Висновки до розділу: економічна частина даної роботи містить розрахунок витрат на розробку програми моніторингу місць паркування, сума яких складає 1058229,38 гривень. Було спрогнозовано орієнтовану величину витрат по кожній з статей витрат. Також розраховано чистий прибуток, який може отримати виробник від реалізації нового технічного рішення, розраховано період окупності витрат для інвестора та економічний ефект при використанні даної розробки. Період окупності складе близько 1,52 роки.

ВИСНОВКИ

Системи моніторингу місць паркування є актуальною сферою досліджень, що матиме значний вплив на життя людей та на розвиток суміжних інформаційних технологій. В процесі виконання магістерської роботи було проведено аналіз та розробку комп'ютерної системи моніторингу місць паркування.

У першому розділі магістерської роботи проведено аналітичний огляд проблематики пошуку місць паркування, проведено огляд підходів до моніторингу місць з застосуванням комп'ютерного зору, проведено аналіз можливостей нейромереж в задачах комп'ютерного зору. В результаті було запропоновано застосування згорткових нейронні мережі як засобу розпізнавання.

У другому розділі магістерської роботи проведено огляд різних архітектур згорткових нейронних мереж, в результаті було обрано архітектуру YOLO, яка є найбільш відповідною до мети магістерської роботи. Проведено розробку методу орієнтованих обернених рамок. Після цього було сформовано алгоритм навчання мережі та визначено основні критерії оцінки якості роботи системи моніторингу.

У третьому розділі магістерської роботи обрано мову та середовище програмування, підготовано набір даних, проведено навчання нейронної мережі, розроблено програмний засіб для реалізації запропонованого підходу для моніторингу місць паркування автомобілів.

У четвертому розділі магістерської роботи проведено тестування працездатності та оцінка ефективності роботи системи моніторингу місць паркування автомобілів.

У п'ятому розділі магістерської роботи проведено економічні розрахунки із обґрунтування доцільності виконання роботи по розробці системи моніторингу місць паркування, обраховано економічну доцільність та ефективність впровадження.

Розроблена система може слугувати базисом для подальшого розширення функціоналу систем моніторингу місць паркування автомобілів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1) Reforming off-street parking is the key to fewer emissions and more livable cities. Institute for Transportation and Development Policy. [Електронний ресурс]. – Режим доступу: https://itdp.org/2023/09/12/reforming-off-street-parking-fewer-emissions-livable-cities/ (дата звернення: 13.10.2025).

2) Fouad W. Smart parking as one of the smart cities mechanisms. The academic research community publication. 2019. Т. 3, № 2. С. 256. [Електронний ресурс]. – Режим доступу: https://doi.org/10.21625/archive.v3i2.515 (дата звернення: 13.10.2025).

3) Ponnambalam C. T., Donmez B. Searching for street parking: effects on driver vehicle control, workload, physiology, and glances. Frontiers in psychology. 2020. Т. 11. [Електронний ресурс]. – Режим доступу: https://doi.org/10.3389/fpsyg.2020.574262 (дата звернення: 13.10.2025).

4) Комп’ютерна система моніторингу місць паркування автомобілів на основі нейронної мережі / Є. М. Лучко, Т. Б. Мартинюк, М. А. Очкуров // Матеріали LV науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії. Вінниця 2025 р. 2 с. [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2026/paper/view/26651/22002

5) Verdes R. P., Klein L. A. Traffic detector handbook: third edition–volume I. [Електронний ресурс]. – Режим доступу: https://www.fhwa.dot.gov/publications/research/operations/its/06108/ (дата звернення: 13.10.2025).

6) An ultrasonic sensor system for vehicle detection application / R. Stiawan та ін. Journal of physics: conference series. 2019. Т. 1204. С. 012017. [Електронний ресурс]. – Режим доступу: https://doi.org/10.1088/1742-6596/1204/1/012017 (дата звернення: 13.10.2025).

7) Marcin B., Miodonska Z., Krecichwost M. Vehicle detection system using magnetic sensors. Transport problems. 2014. [Електронний ресурс]. – Режим доступу:

https://www.researchgate.net/publication/287944531_Vehicle_detection_system_using_magnetic_sensors (дата звернення: 13.10.2025).

8) Robust lidar-based vehicle detection for on-road autonomous driving / X. Jin та ін. Remote sensing. 2023. Т. 15, № 12. С. 3160. [Електронний ресурс]. – Режим доступу: https://doi.org/10.3390/rs15123160 (дата звернення: 13.10.2025).

9) Abdul Rani W. N. H. B., Fadzil L. M. Object detection algorithms for parking detection - survey. International journal of electrical and electronics engineering. 2024. Т. 11, № 4. С. 167–174. [Електронний ресурс]. – Режим доступу: https://doi.org/10.14445/23488379/ijeee-v11i4p118 (дата звернення: 13.10.2025).

10) Watershed segmentation. Science Direct. [Електронний ресурс]. – Режим доступу: https://www.sciencedirect.com/topics/computer-science/watershed-segmentation (дата звернення: 13.10.2025).

11) Muthulakshmi R., Anusha K., Divyatharshni M. A comparative analysis of SSD, mask R-CNN and yolo for object detection. Tijer. 2024. [Електронний ресурс]. – Режим доступу: https://tijer.org/tijer/papers/TIJERC001183.pdf (дата звернення: 13.10.2025).

12) Ji A., Ma X. Vehicle detection and classification for traffic management and autonomous systems using YOLOv10. Alexandria engineering journal. 2025. Т. 127. С. 804–816. [Електронний ресурс]. – Режим доступу: https://doi.org/10.1016/j.aej.2025.06.049 (дата звернення: 13.10.2025).

13) Jadama A. K., Toray M. K. Ensemble Learning: Methods, Techniques, Application. 2024. [Електронний ресурс]. – Режим доступу: https://doi.org/10.13140/RG.2.2.28017.08802.

14) Convolutional neural networks for vision neuroscience: significance, developments, and outstanding issues / A. Celeghin та ін. Frontiers in computational neuroscience. 2023. Т. 17. [Електронний ресурс]. – Режим доступу: https://doi.org/10.3389/fncom.2023.1153572.

15) A review of convolutional neural networks in computer vision / X. Zhao та ін. Artificial Intelligence Review. 2024. Т. 57, № 4. [Електронний ресурс]. – Режим доступу:https://doi.org/10.1007/s10462-024-10721-6.

16) Caceres P. Introduction to neural network models of cognition. Wisconsin, 2024.

17) Yamashita, R., Nishio, M., Do, R.K.G. та ін. Convolutional neural networks: an overview and application in radiology. Insights Imaging 9, 611–629 (2018). https://doi.org/10.1007/s13244-018-0639-9.

18) Yuldashev, Y., Mukhiddinov, M., Abdusalomov, A. B., Nasimov, R., & Cho, J. (2023). Parking Lot Occupancy Detection with Improved MobileNetV3. Sensors, 23(17), 7642. https://doi.org/10.3390/s23177642.

19) Bietti A., Mairal J. Group Invariance, Stability to Deformations, and Complexity of Deep Convolutional Representations. Journal of Machine Learning Research 20 (2019) 1-49. 2017. [Електронний ресурс]. – Режим доступу: https://doi.org/10.48550/arXiv.1706.03078.

20) The Math Behind Convolutional Neural Networks | Towards Data Science. Towards Data Science. [Електронний ресурс]. – Режим доступу: https://towardsdatascience.com/the-math-behind-convolutional-neural-networks-6aed775df076/#92f5.

21) Going Deeper with Convolutions. C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. CoRR, (2014)

22) Lea C., Vidal R., Reiter A. Temporal Convolutional Networks: A Unified Approach to Action Segmentation. CoRR. 2016. Abs/1608.08242.

23) Nugroho W., Afianto A., Arifianto M. J. F. Smart Parking based on Car Detection using Deep Learning YOLOv8. International Journal Of Electrical Engineering And Intelligent Computing. 2024. Т. 2, № 1. С. 1–7. [Електронний ресурс]. – Режим доступу: https://doi.org/10.33387/ijeeic.v2i1.8692 (дата звернення: 06.11.2025).

24) Pu C., Yu J., Su W. Rotated R-CNN: A Two-Stage Object Detection Method Adapted To Oriented Bounding Boxes. Conference: 2024 IEEE International Conference on Image Processing (ICIP). 2024. [Електронний ресурс]. – Режим доступу: https://doi.org/10.1109/ICIP51287.2024.10647377.

25) Nanni, L., Paci, M., Brahnam, S., & Lumini, A. (2021). Comparison of Different Image Data Augmentation Approaches. *Journal of imaging*, *7*(12), 254. https://doi.org/10.3390/jimaging7120254.

26) Xu, C., & Wang, H. (2022). Research on a Convolution Kernel Initialization Method for Speeding Up the Convergence of CNN. Applied Sciences, 12(2), 633. https://doi.org/10.3390/app12020633

27) A comprehensive survey of loss functions and metrics in deep learning / J. Terven et al. Artificial Intelligence Review. 2025. Vol. 58, no. 7.

28) Analysis of the Best Optimizer Used by Convolutional Neural Network Algorithms in Detecting Masked Faces. 2023 Eighth International Conference on Informatics and Computing (ICIC). 2023. URL: https://doi.org/10.1109/ICIC60109.2023.10381927.

29) Nurhopipah A., Larasati N. A. CNN Hyperparameter Optimization using Random Grid Coarse-to-fine Search for Face Classification. Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control. 2021. P. 19–26. URL: https://doi.org/10.22219/kinetik.v6i1.1185 (date of access: 06.11.2025).

30) Lema D. G., Usamentiaga R., García D. F. Quantitative comparison and performance evaluation of deep learning-based object detection models on edge computing devices. Integration. 2024. Vol. 95. P. 102127. URL: https://doi.org/10.1016/j.vlsi.2023.102127 (date of access: 06.11.2025).

31) Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. — Вінниця : ВНТУ, 2021. — 42 с.

32) Тарифи на електроенергію [Електронний ресурс]. Режим доступу: http://index.minfin.com.ua/tarif/electric.php.

ДОДАТОК А

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ
Завідувач кафедри ОТ проф.,
д.т.н.. Азаров О. Д.
«__03__» __жовтня__ 2025 р.

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи
«Комп'ютерна система моніторингу місць паркування автомобілів на основі нейронної мережі»

Науковий керівник: д.т.н., проф. каф. ОТ
_____Мартинюк Т.Б.

Студент групи 2КІ-24м
_____Лучко Є.М.

ВНТУ 2025

1 Підставою для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність дослідження пов’язана з зростаючою урбанізацією та збільшенням кількості транспортних засобів у містах та необхідністю контролю стану зайнятості місць паркування.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — вдосконалення системи моніторингу місць паркування автомобілів.. Вдосконалення досягається шляхом застосування нейронної мережі для виконання задач розпізнавання автомобілів.

2.2 Призначення розробки — створення програмного продукту для реалізації системи моніторингу місць паркування.

3 Вихідні дані для виконання МКР

3.1 Відеофрагмент роздільною здатністю 1920 на1080.

3.2 Застосування бібліотеки Ultralytics.

3.3 Середовище розробки Visual Studio Code.

4 Вимоги до виконання МКР

4.1 Провести аналіз методів моніторингу місць паркування.

4.2 Розробити метод моніторингу місць паркування.

4.3 Виконати розробку програмного засобу моніторингу місць паркування.

4.4 Провести тестування працездатності та ефективності роботи програми моніторингу місць паркування.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 – Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз методів та проблематики моніторингу місць паркування	26.09.2025	07.10.2025	Розділ 1
2	Розробка моделі та алгоритму системи моніторингу місць паркування	08.10.2025	16.10.2025	Розділ 2
3	Програмна реалізація системи моніторингу місць паркування	17.10.2025	27.10.2025	Розділ 3
4	Тестування та перевірка якості роботи системи моніторингу	28.10.2025	30.10.2025	Розділ 4
5	Розрахунок економічної доцільності розробки	01.11.2025	10.11.2025	Розділ 5
5	Оформлення пояснювальної записки, графічного матеріалу і презентації	11.11.2025	24.11.2025	Розроблена програма, графічний матеріал
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	25.11.2025	03.12.2025	Оформлені документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами, довідка про відповідність оформлення МКР діючим вимогам.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої

наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлювання МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— міждержавний ГОСТ 2.104-2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 – «Комп'ютерна інженерія» (освітня програма «Комп'ютерна інженерія»). Кафедра обчислювальної техніки ВНТУ 2023;

— документами на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ-03.02.02-П.001.01:21».

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: _Комп'ютерна система моніторингу місць паркування автомобілів на основі нейронної мережі_

Тип роботи: _____магістерська кваліфікаційна робота_____
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ _____кафедра обчислювальної техніки_____
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism (КП1) ___1___%

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- ✓ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак академічного плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту.
- ☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- ☐ У роботі виявлено ознаки академічного плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень. Робота до захисту не приймається.

Особа, відповідальна за перевірку _____ _Захарченко С. М._
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ _____Мартинюк Т. Б._____
(підпис) (прізвище, ініціали)

Здобувач _____ _____Лучко Є. М._____
(підпис) (прізвище, ініціали)

Завідувач кафедри ОТ _____ _____Азаров О. Д._____
(підпис) (прізвище, ініціали)

Гарант освітньої програми _____ _____Мартинюк Т. Б._____
(підпис) (прізвище, ініціали)

ДОДАТОК В

Типи сенсорів систем моніторингу місць паркування автомобілів

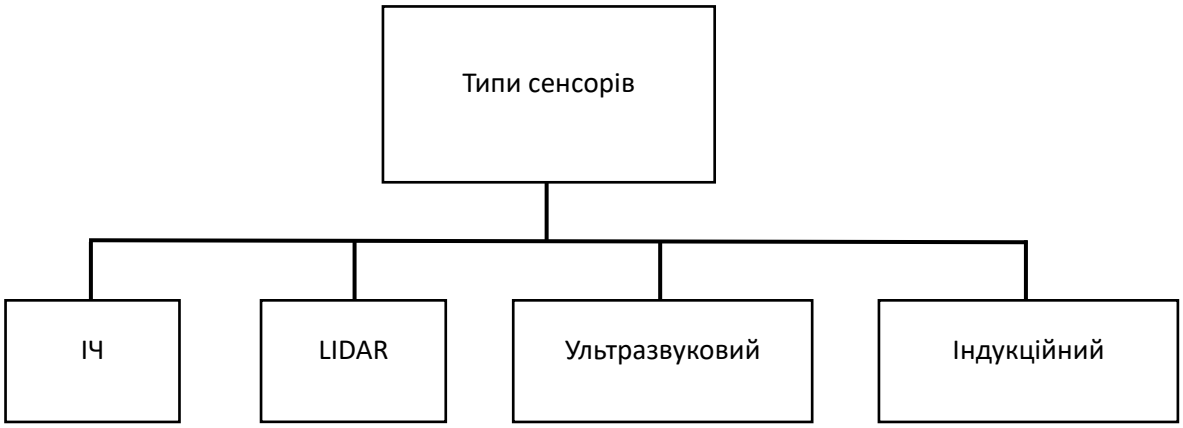


Рисунок В.1 — Типи сенсорів

ДОДАТОК Г

Типи згорткових нейронних мереж

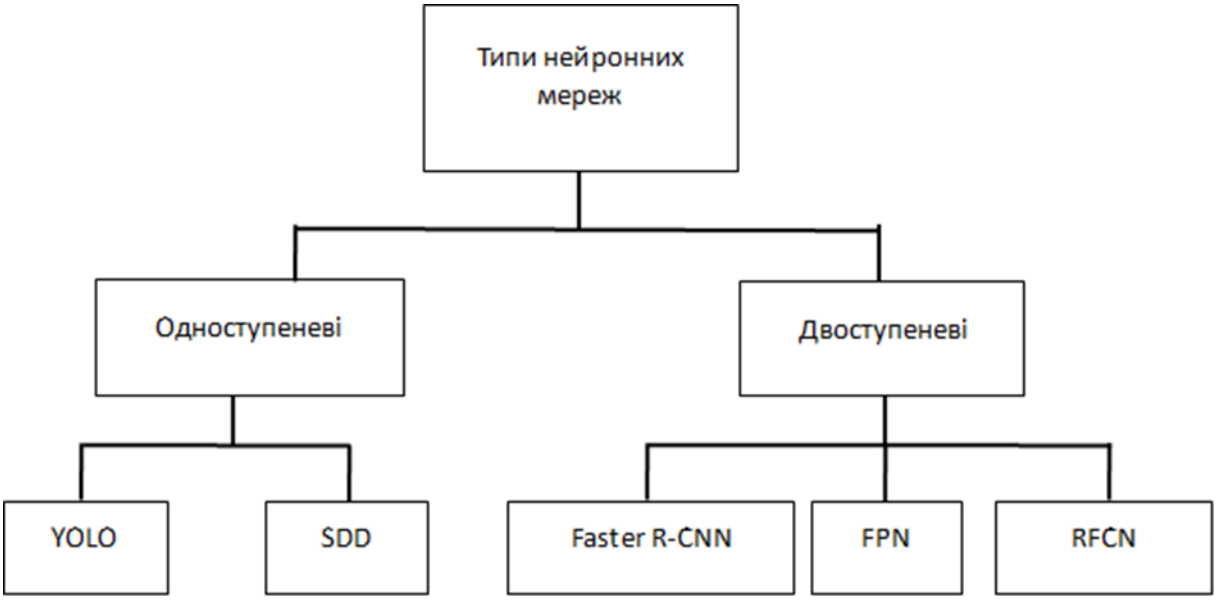


Рисунок Г.1 — Типи нейронних мереж

ДОДАТОК Д

Послідовність роботи системи моніторингу місць паркування автомобілів

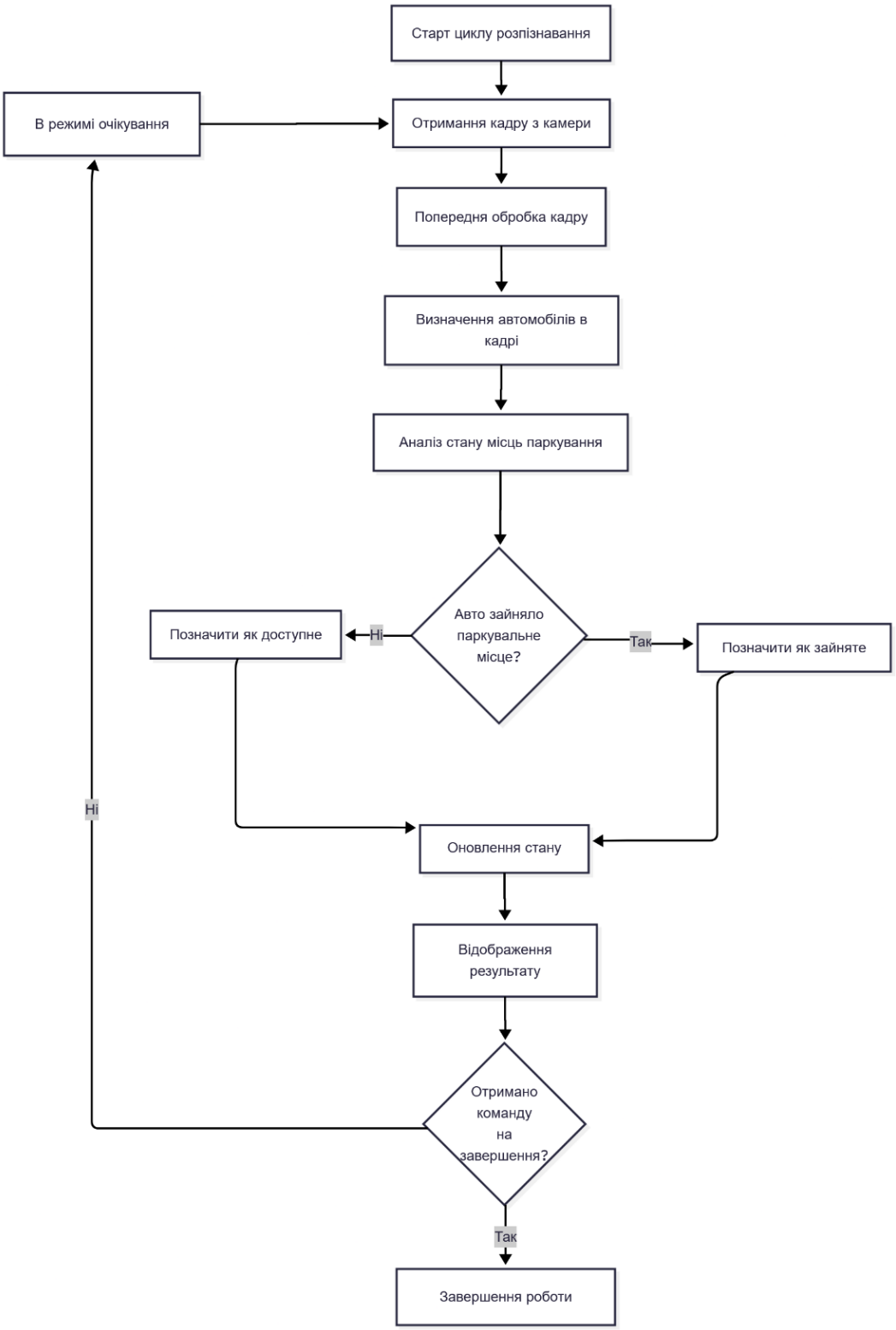


Рисунок Д.1 — Послідовність роботи системи моніторингу місць паркування

ДОДАТОК Е

Лістинг програми донавчання нейронної мережі

```
import torch
import torch.nn as nn
from ultralytics import YOLO
from ultralytics.nn.tasks import DetectionModel
from ultralytics.utils.loss import v8DetectionLoss
from ultralytics.data import build_dataloader
import matplotlib.pyplot as plt
from torch.utils.data import DataLoader
import numpy as np
class YOLOTrainer:
    def __init__(self, model_cfg='yolo11_obb.yaml', num_classes=2):
        self.model = DetectionModel(cfg=model_cfg, ch=3, nc=num_classes)
        self.loss_fn = v8DetectionLoss(self.model)
        self.optimizer = torch.optim.Adam(self.model.parameters(), lr=0.001)
        self.train_losses = []
        self.val_losses = []
    def forward_pass(self, batch):
        preds = self.model(batch['img'])
        loss, loss_items = self.loss_fn(preds, batch['bboxes'])
        return loss, loss_items
    def backward_pass(self, loss):
        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()
    def train_epoch(self, dataloader):
        self.model.train()
        epoch_loss = 0
        for batch_idx, batch in enumerate(dataloader):
            batch['img'] = batch['img'].to('cuda')
            batch['bboxes'] = [bbox.to('cuda') for bbox in batch['bboxes']]
            loss, loss_items = self.forward_pass(batch)
            self.backward_pass(loss)
            epoch_loss += loss.item()
            if batch_idx % 50 == 0:
                print(f'Batch {batch_idx}, Loss: {loss.item():.4f}')
        return epoch_loss / len(dataloader)
class ProcessDataset:
    def __init__(self, images, annotations, img_size=640):
        self.images = images
        self.annotations = annotations
        self.img_size = img_size
```

```
        def __getitem__(self, idx):
        image = self.load_and_preprocess_image(self.images[idx])
        bboxes = self.parse_annotations(self.annotations[idx])
            return {
        'img': image,
        'bboxes': bboxes,
        'cls': torch.tensor([bbox[0] for bbox in bboxes]),  # class IDs
        'batch_idx': idx
        }
    def load_and_preprocess_image(self, image_path):
        image = plt.imread(image_path)
        image = torch.from_numpy(image).permute(2, 0, 1).float() / 255.0
        return F.interpolate(image.unsqueeze(0), size=(self.img_size,
self.img_size)).squeeze(0)
    def parse_annotations(self, annotation):
        bboxes = []
        for ann in annotation:
            x1, y1, x2, y2, cls_id = ann
            x_center = (x1 + x2) / 2
            y_center = (y1 + y2) / 2
            width = x2 - x1
            height = y2 - y1
            bboxes.append([cls_id, x_center, y_center, width, height])
        return torch.tensor(bboxes)
class DetectionVisualizer:
    def __init__(self, model, confidence_threshold=0.5, iou_threshold=0.5):
        self.model = model
        self.conf_thresh = confidence_threshold
        self.iou_thresh = iou_threshold
    def non_max_suppression(self, predictions):
        boxes, scores, classes = [], [], []
        for i in range(predictions.shape[0]):
            pred = predictions[i]
            mask = pred[:, 4] > self.conf_thresh
            pred = pred[mask]
            if len(pred) == 0:
                continue
            box = pred[:, :4]
            score = pred[:, 4]
            cls = pred[:, 5:].argmax(1)
            # Convert to x1, y1, x2, y2
            x1 = box[:, 0] - box[:, 2] / 2
            y1 = box[:, 1] - box[:, 3] / 2
            x2 = box[:, 0] + box[:, 2] / 2
```

```
            y2 = box[:, 1] + box[:, 3] / 2
            box = torch.stack([x1, y1, x2, y2], dim=1)
            keep = self._nms(box, score)
            boxes.append(box[keep])
            scores.append(score[keep])
            classes.append(cls[keep])
          return boxes, scores, classes
    def _nms(self, boxes, scores):
        if len(boxes) == 0:
            return []
        sorted_indices = torch.argsort(scores, descending=True)
        keep = []
        while len(sorted_indices) > 0:
            current_idx = sorted_indices[0]
            keep.append(current_idx)
            if len(sorted_indices) == 1:
                break
            current_box = boxes[current_idx]
            remaining_boxes = boxes[sorted_indices[1:]]
            ious = self._calculate_iou(current_box, remaining_boxes)
            keep_indices = torch.where(ious <= self.iou_thresh)[0]
            sorted_indices = sorted_indices[1:][keep_indices]
        return torch.tensor(keep)
    def _calculate_iou(self, box1, boxes2):
        x1 = torch.max(box1[0], boxes2[:, 0])
        y1 = torch.max(box1[1], boxes2[:, 1])
        x2 = torch.min(box1[2], boxes2[:, 2])
        y2 = torch.min(box1[3], boxes2[:, 3])
        intersection = torch.clamp(x2 - x1, min=0) * torch.clamp(y2 - y1, min=0)
        area1 = (box1[2] - box1[0]) * (box1[3] - box1[1])
        area2 = (boxes2[:, 2] - boxes2[:, 0]) * (boxes2[:, 3] - boxes2[:, 1])
        union = area1 + area2 - intersection
        return intersection / union
class CustomDataAugmentation:
    def __init__(self):
        pass
    def augment_batch(self, images, bboxes):
        augmented_images = []
        augmented_bboxes = []
        for img, bbox in zip(images, bboxes):
            if torch.rand(1) > 0.5:
                img, bbox = self.horizontal_flip(img, bbox)
            img, bbox = self.random_scale(img, bbox)
            augmented_images.append(img)
```

```
            augmented_bboxes.append(bbox)
        return torch.stack(augmented_images), augmented_bboxes
    def horizontal_flip(self, image, bboxes):
        image = torch.flip(image, dims=[2])
        if len(bboxes) > 0:
            bboxes[:, 1] = 1.0 - bboxes[:, 1]  # x_center
        return image, bboxes
        if scale < 1.0:
            pad_h = (image.shape[1] - new_size) // 2
            pad_w = (image.shape[2] - new_size) // 2
            image = F.pad(image, (pad_w, pad_w, pad_h, pad_h))
        else:
            start_h = (new_size - image.shape[1]) // 2
            start_w = (new_size - image.shape[2]) // 2
            image = image[:, start_h:start_h+image.shape[1],
start_w:start_w+image.shape[2]]
        return image, bboxes
```

ДОДАТОК Ж

Лістинг програми моніторингу місць паркування

main_pipeline.py

```
import cv2
import time
import argparse
import json
import sys
import os
import numpy as np
from typing import Dict, Any, Optional
sys.path.append(os.path.dirname(os.path.dirname(os.path.abspath(__file__))))
from src.obb_detector import OBBVehicleDetector, create_obb_detector
from src.occupancy_analyzer import ParkingOccupancyAnalyzer,
create_occupancy_analyzer
from src.visualization import ParkingVisualizer, create_visualizer
class SmartParkingSystem:
    def __init__(self, model_path: str, zones_config: str, source: Any = 0,
                 config: Dict[str, Any] = None):
        self.model_path = model_path
        self.zones_config = zones_config
        self.source = source
        self.config = config or {}
        self.detector = None
        self.analyzer = None
        self.visualizer = None
        self.cap = None
        self.running = False
        self.frame_count = 0
        self.processing_time = 0
        self._initialize_components()
    def _initialize_components(self):
        try:
            detector_config = self.config.get('detector', {})
            self.detector = create_obb_detector(self.model_path, **detector_config)
            if self.config.get('tracking', {}).get('enabled', True):
                tracker_config = self.config.get('tracking', {}).get('tracker_config',
'botsort.yaml')
                self.detector.enable_tracking(tracker_config)
            self.analyzer =
create_occupancy_analyzer(zones_config=self.zones_config)
            visualizer_config = self.config.get('visualization', {})
```

```
            window_name = visualizer_config.get('window_name', 'Smart Parking
System')
            self.visualizer = create_visualizer(window_name)
            self._initialize_video_capture()
        except Exception as e:
            print(f" Error initializing system: {e}")
            raise
    def _initialize_video_capture(self):
        try:
            self.cap = cv2.VideoCapture(self.source)
            if not self.cap.isOpened():
                raise ValueError(f"Could not open video source: {self.source}")
            self.frame_width = int(self.cap.get(cv2.CAP_PROP_FRAME_WIDTH))
            self.frame_height = int(self.cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
            self.fps = self.cap.get(cv2.CAP_PROP_FPS)
        except Exception as e:
            print(f" Error initializing video capture: {e}")
            raise
    def process_frame(self, frame: np.ndarray) -> Dict[str, Any]:
        start_time = time.time()
        try:
            detections = self.detector.detect_vehicles(frame)
            occupancy_result =
self.analyzer.update_occupancy_status(detections['centroids'])
            processing_data = {
                'detections': detections,
                'zones': self.analyzer.parking_zones,
                'occupancy_status': occupancy_result['stable_occupancy'],
                'stats': {
                    **occupancy_result,
                    'detections': detections,
                    'processing_time': self.processing_time
                }
            }
            self.processing_time = time.time() - start_time
            return processing_data
        except Exception as e:
            print(f" Error processing frame: {e}")
def run(self):
        if not self.cap:
            print(" Video capture not initialized")
            return
        print("Starting Smart Parking System...")
        self.running = True
```

```
        paused = False
        frame_skip = 0
        max_frame_skip = 5
        try:
            while self.running:
                if not paused:
                    ret, frame = self.cap.read()
                    if not ret:
                        print(" Could not read frame from video source")
                        break
                    frame_skip += 1
                    if frame_skip > max_frame_skip:
                        frame_skip = 0
                        continue
                    processing_data = self.process_frame(frame)
                    dashboard = self.visualizer.create_dashboard(frame, processing_data)
                    self.visualizer.show_frame(dashboard)
                    self.frame_count += 1
                key = cv2.waitKey(1) & 0xFF
                if key == ord('q'):  # Quit
                    break
obb_detector.py
import cv2
import numpy as np
from ultralytics import YOLO
from typing import List, Tuple, Dict, Any
import time
class OBBVehicleDetector:
    def __init__(self, model_path: str, device: str = 'auto', conf_threshold: float =
0.5):
        self.model_path = model_path
        self.device = device
        self.conf_threshold = conf_threshold
        self.model = None
        self.class_names = {}
        self.tracking_enabled = False
        self.load_model()
    def load_model(self):
        try:
            print(f"Loading OBB model from: {self.model_path}")
            self.model = YOLO(self.model_path)
            self.class_names = self.model.names
            print(f"Model loaded successfully. Classes:
{list(self.class_names.values())}")
```

```
except Exception as e:
        print(f" Error loading model: {e}")
        raise
def enable_tracking(self, tracker_config: str = "botsort.yaml"):
    self.tracking_enabled = True
    self.tracker_config = tracker_config
    print(f"Tracking enabled with config: {tracker_config}")
    def calculate_obb_centroid(self, obb_points) -> Tuple[int, int]:
    if hasattr(obb_points, 'cpu'):
        # PyTorch tensor - convert to numpy
        obb_points = obb_points.cpu().numpy()
    if obb_points.size == 8
        points = obb_points.reshape(4, 2
        x_center = int(np.mean(points[:, 0]))
        y_center = int(np.mean(points[:, 1]))
        return x_center, y_center
def detect_vehicles(self, frame: np.ndarray) -> Dict[str, Any]:
    if self.model is None:
        raise ValueError("Model not loaded. Call load_model() first.")
    try:
        if self.tracking_enabled:
            results = self.model.track(
                frame,
                conf=self.conf_threshold,
                persist=True,
                tracker=self.tracker_config,
                verbose=False
            )
        else:
            results = self.model(
                frame,
                conf=self.conf_threshold,
                verbose=False
            )

        detections = self._extract_detection_data(results[0])
        return detections
    except Exception as e:
        print(f"Error during vehicle detection: {e}")
        return self._empty_detections()
  def _extract_detection_data(self, result) -> Dict[str, Any]:
    detections = {
        'boxes': [],
        'centroids': [],
```

```
            'classes': [],
            'confidences': [],
            'track_ids': [],
            'is_obb': False
        }
        if hasattr(result, 'obb') and result.obb is not None:
            detections['is_obb'] = True
            obb_data = result.obb
            if obb_data and obb_data.is_track:
                boxes = obb_data.xyxyxyxy.cpu().numpy() if obb_data.xyxyxyxy is not
None else []
                classes = obb_data.cls.cpu().tolist() if obb_data.cls is not None else []
                confidences = obb_data.conf.cpu().tolist() if obb_data.conf is not None
else []
                track_ids = obb_data.id.int().cpu().tolist() if obb_data.id is not None else
[]

                for i, box in enumerate(boxes):
                    centroid = self.calculate_obb_centroid(box)
                    detections['boxes'].append(box)
                    detections['centroids'].append(centroid)
                    detections['classes'].append(classes[i] if i < len(classes) else -1)
                    detections['confidences'].append(confidences[i] if i < len(confidences)
else 0.0)
                    detections['track_ids'].append(track_ids[i] if i < len(track_ids) else -1)

        return detections
occupancy_analyzer.py
class ParkingOccupancyAnalyzer:
    def __init__(self, zones_config_path: str = None, zones_data: List[Dict] =
None):
        self.zones_config_path = zones_config_path
        self.parking_zones = []
        self.occupancy_history = defaultdict(list)
        self.occupancy_threshold = 3
        if zones_config_path:
            self.load_parking_zones(zones_config_path)
        elif zones_data:
            self.parking_zones = zones_data
        else:
            print("No parking zones provided.")
    def load_parking_zones(self, config_path: str):
        try:
            with open(config_path, 'r', encoding='utf-8') as f:
```

```
            data = json.load(f)
        if isinstance(data, dict) and 'parking_zones' in data:
            self.parking_zones = data['parking_zones']
        elif isinstance(data, list):
            self.parking_zones = data
        else:
            raise ValueError("Invalid parking zones format")
    except Exception as e:
        print(f"Error loading parking zones: {e}")
        raise
def check_occupancy(self, vehicle_centroids: List[Tuple[int, int]],
                    zones: List[Dict] = None) -> List[bool]:
    if zones is None:
        zones = self.parking_zones
    occupancy_status = [False] * len(zones)
    polygon_points_list = []
    for zone in zones:
        zone_points = zone['points']
        polygon_points = np.array(zone_points, dtype=np.int32).reshape((-1, 1, 2))
        polygon_points_list.append(polygon_points)
    for centroid in vehicle_centroids:
        for i, polygon_points in enumerate(polygon_points_list):
            if not occupancy_status[i]:
                distance = cv2.pointPolygonTest(polygon_points, centroid, False)
                if distance >= 0:
                    occupancy_status[i] = True
    return occupancy_status
def update_occupancy_status(self, vehicle_centroids: List[Tuple[int, int]],
                            frame_timestamp: float = None) -> Dict[str, Any]:
    if not self.parking_zones:
        return self._empty_occupancy_result()
    current_occupancy = self.check_occupancy(vehicle_centroids)
    for zone_id, occupied in enumerate(current_occupancy):
        zone_key = f"zone_{zone_id}"
        self.occupancy_history[zone_key].append(occupied)
        if len(self.occupancy_history[zone_key]) > self.occupancy_threshold:
            self.occupancy_history[zone_key].pop(0)
    stable_occupancy = []
    for zone_id in range(len(self.parking_zones)):
        zone_key = f"zone_{zone_id}"
        history = self.occupancy_history[zone_key]
        if len(history) >= self.occupancy_threshold:
            occupied_count = sum(history)
            stable_occupied = occupied_count > len(history) // 2
```

```
            else:
                stable_occupied = current_occupancy[zone_id]
            stable_occupancy.append(stable_occupied)
        occupied_count = sum(stable_occupancy)
        available_count = len(stable_occupancy) - occupied_count
        return {
            'current_occupancy': current_occupancy,
            'stable_occupancy': stable_occupancy,
            'occupied_count': occupied_count,
            'available_count': available_count,
            'total_zones': len(self.parking_zones),
            'occupancy_rate': occupied_count / len(self.parking_zones) if
self.parking_zones else 0
        }
```