МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХЕРСОНСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ

ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК, ФІЗИКИ ТА МАТЕМАТИКИ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ПРОГРАМНОЇ ІНЖЕНЕРІЇ

Нейромережа для автоматизованого спостереження за транспортним рухом з подальшим виявленням правопорушень

Кваліфікаційна робота (проєкт)
на здобуття ступеня вищої освіти "магістр"

Виконав: здобувач 2 курсу 12-241М групи
Спеціальності: 121 Інженерія програмного забезпечення
Освітньо-професійної програми: «Інженерія програмного забезпечення» другого (магістерського) рівня вищої освіти
Левченко Дмитро Олександрович

Керівник: Завідувач кафедри комп'ютерних наук та програмної інженерії ХДУ, доктор фізико-математичних наук, професор Песчаненко Володимир Сергійович

Рецензент: кандидатка технічних наук, доцентка кафедри програмних засобів і технологій, Захарченко Р. М. Херсонський національно-технічний університет

Івано-Франківськ – 2024

ЗМІСТ

ВСТУП

Технологічний прогрес не стоїть на місці, і з кожним роком ми все більше інтегруємо інформаційні технології в наше повсякденне життя. Різноманітні програми, веб-сайти, соціальні мережі та інші цифрові платформи стали невід'ємними елементами сучасного суспільства, допомагаючи задовольняти повсякденні потреби та вирішувати різноманітні завдання.

Однією з ключових переваг, які пропонує світ інформаційних технологій, є підвищення ефективності процесів і їх абстрагування від користувача. Сучасні додатки не тільки оптимізують процеси, пов'язані з доставкою та вибором товару, але й допомагають користувачам робити усвідомлений вибір на основі аналізу попередніх покупок. Однією з найважливіших технологій, що забезпечують такий рівень автоматизації, є нейронні мережі, які відкривають нові можливості для покращення взаємодії користувачів із цифровими системами.

Нейронні мережі набувають все більшого поширення та використовуються в багатьох галузях, таких як охорона здоров'я, фінанси, транспорт, освіта, виробництво, логістика. Завдяки здатності аналізувати великі обсяги даних і виявляти складні закономірності вони відіграють ключову роль в автоматизації процесів, оптимізації рішень і підвищенні ефективності різних систем.

Таким чином, нейронні мережі є новим кроком в автоматизації, який дозволяє делегувати ще більш складні завдання та оптимізувати процеси, які раніше вважалися непридатними для подальшого вдосконалення. Одним із таких завдань є впровадження ефективних методів контролю за дотриманням правил дорожнього руху. У сучасному світі, в умовах стрімкого зростання інтенсивності руху, це питання постає все гостріше. Традиційні методи моніторингу та виявлення порушень, засновані на ручному управлінні або

стаціонарних камерах спостереження, мають обмежені можливості, особливо у великих містах з інтенсивним трафіком. У цих умовах штучний інтелект, зокрема нейронні мережі, відкриває нові перспективи для автоматизації моніторингу дорожнього руху та виявлення порушень.

Нейромережа для автоматизованого спостереження за транспортним рухом забезпечує можливість швидко та точно обробляти величезні обсяги даних у реальному часі, ідентифікувати транспортні засоби, розпізнавати типи порушень, такі як перевищення швидкості, порушення сигналів світлофора, неправильне паркування тощо. Ця технологія дозволяє не лише підвищити рівень безпеки на дорогах, але й оптимізувати роботу правоохоронних органів, зменшуючи потребу в ручному моніторингу та реагуванні на правопорушення.

Актуальність: Актуальність теми полягає в необхідності автоматизації процесу виявлення правопорушень під час руху транспорту.

Мета: Метою даного дослідження є розробка ефективної системи автоматизованого спостереження за транспортним рухом із застосуванням нейронних мереж, яка забезпечить високу точність виявлення правопорушень. Очікуваний результат полягає в створенні концепції такої системи та реалізації її головної частини. Така система зможе автоматизувати процеси моніторингу дорожнього руху, підвищити ефективність виявлення порушень правил дорожнього руху, зменшити навантаження на правоохоронні органи та мінімізувати вплив людського фактору на процес контролю та фіксації порушень.

Завдання:

1. Аналіз теоретичних основ по темі нейронних мереж.
2. Дослідження сучасних систем контролю та моніторингу транспортного руху. Аналіз існуючих рішень на основі нейронних мереж для виявлення правопорушень

3. Розробка концепції системи автоматизованого спостереження за транспортним рухом.
4. Розробка та навчання нейромереж для відслідковування та сегментації транспортних засобів та подальшого розпізнавання номерних знаків
5. Тестування системи та аналіз результатів.

Об’єкт дослідження: Об’єктом дослідження є системи автоматизованого моніторингу дорожнього руху. Це набір процесів і технологій, які забезпечують моніторинг, збір, аналіз і зберігання даних про транспортні потоки з метою виявлення порушень. Об’єктом дослідження є такі аспекти, як ефективність контролю дорожнього руху, точність виявлення порушень, оптимізація систем спостереження.

Предмет дослідження: Предметом дослідження є технологічні аспекти застосування нейронних мереж для автоматизованого моніторингу трафіку. Це включає дослідження методів використання нейронних мереж для аналізу відеоданих, точного розпізнавання об’єктів і розпізнавання злочинів. Основна увага зосереджена на розробці алгоритмів для підвищення точності та швидкості обробки даних, зменшення впливу людського фактору та оптимізації процесу виявлення правопорушень у режимі реального часу.

РОЗДІЛ 1: ОСНОВИ НЕЙРОННИХ МЕРЕЖ

Поняття нейромережі.

Нейронна мережа - це система штучного інтелекту, яка імітує спосіб обробки даних людським мозком. Вона є варіантом машинного навчання, відомим як глибоке навчання, та використовує міжзв'язані елементи, зазвичай називаються нейронами, розташовані в багатошаровій структурі, що співставляється з архітектурою мозку. Нейромережі розвиваються через адаптивний процес, де комп'ютерні системи навчаються на основі своїх помилок і постійно поліпшують свою роботу[1,3,5].

Нейромережа складається з трьох базових рівнів:

Вхідний рівень: цей рівень отримує початкові дані або функції, які оброблятиме нейронна мережа. Кожен нейрон у вхідному шарі представляє функцію вхідних даних.

Приховані шари: ці шари виконують обчислення вхідних даних. Кожен нейрон прихованого шару отримує вхідні дані від нейронів попереднього шару, застосовує математичну функцію (звану функцією активації) і передає результат нейронам наступного шару.

Вихідний рівень: останній рівень нейронної мережі створює вихідні дані моделі. Кількість нейронів у цьому шарі залежить від типу проблеми, яку вирішує нейронна мережа. Наприклад, у задачі двійкової класифікації (де результатом є «так» або «ні») у вихідному шарі буде один нейрон.[5,6]

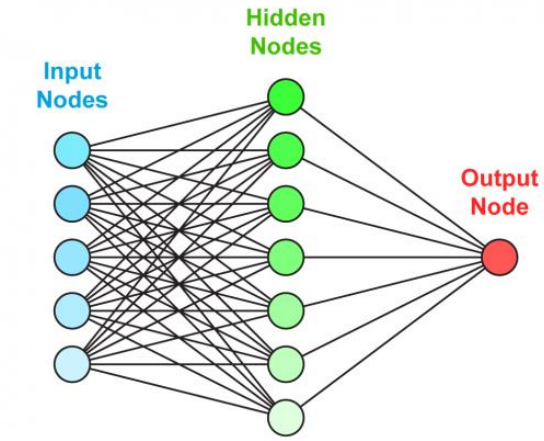

Фото 1 структура нейромережі

Історія розвитку неромереж.

Розвиток нейронних мереж бере свій початок у нейробіологічних дослідженнях і моделюванні людського мозку, які почалися в 1940-х роках. Перший великий прорив у цій галузі стався в 1943 році, коли Уоррен Маккалок і Уолтер Піттс запропонували математичну модель нейрона, яка лягла в основу концепції штучної нейронної мережі. Ця модель нейрона, відома як функція порогової логіки, мала здатність зберігати та передавати інформацію так само, як це роблять нейрони в мозку. Хоча це була дуже спрощена модель, вона стала основою для розвитку подальших концепцій нейронних мереж.[7]

У 1958 році Френк Розенблатт представив персептрон, першу одношарову нейронну мережу, здатну до зворотного навчання. Персептрон складався з вхідного рівня, який передавав сигнали на вихідний нейрон, де приймалося рішення про класифікацію. Такий підхід став основою для досліджень у галузі розпізнавання образів. Проте персептрон мав значні обмеження: через свою одношарову структуру він не міг розв’язувати

складніші нелінійні задачі, які стали відомі як «проблема XOR» (нездатність правильно класифікувати нелінійні дані). Це обмежило його застосування та призвело до зниження інтересу до нейронних мереж у 1960-1970-х роках.

Новий етап у розвитку нейронних мереж розпочався у 1980-х роках завдяки появі нових алгоритмів та архітектур навчання. Дослідники Джеффрі Хінтон, Девід Румельхарт і Рональд Вільямс досягли значного прогресу, розробивши метод зворотного поширення, який дозволяє ефективно навчати багатошарові нейронні мережі. Завдяки такому підходу стало можливим будувати більш складні нейронні архітектури, здатні обробляти нелінійні дані. Цей прорив дозволив розробити багатошарові персептрони (MLP), які могли навчатися та вирішувати складніші завдання класифікації та прогнозування.

У 1990-х роках розвиток нейронних мереж значно прискорився у зв'язку зі зростанням обчислювальних можливостей комп'ютерів і вдосконаленням алгоритмів навчання. Одним із ключових досягнень цього періоду стало широке використання алгоритму зворотного поширення помилок для багатошарових мереж, що дозволило нейронним мережам вирішувати складні задачі з великою кількістю параметрів. У цей час нейронні мережі почали застосовуватися в різних сферах, включаючи економіку, медицину та розпізнавання мови.

Проривною подією стало створення в 1998 році Яном Лекуном і його співавторами архітектури згорткових нейронних мереж (CNN), спеціально розробленої для завдань розпізнавання образів. CNN використовували шари згортки та субдискретизації для виділення характеристик зображення, що робило їх ідеальними для обробки візуальної інформації. Завдяки CNN нейронні мережі використовувалися для розпізнавання обличчя, рукописного тексту, аналізу зображень у медицині та багатьох інших сферах.

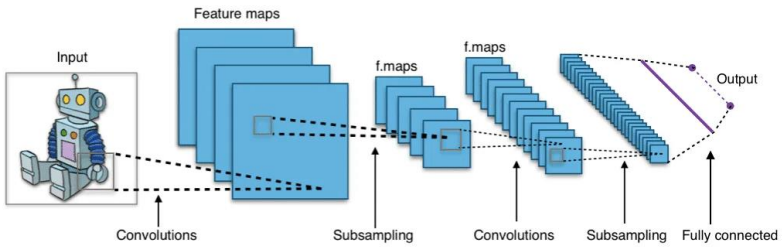

Фото 2 структура CNN згортки

У 2010-х роках розвиток нейронних мереж вийшов на новий рівень завдяки глибокому навчанню (Deep Learning). Зокрема, такі архітектури, як рекурентні нейронні мережі (RNN) і глибокі згорткові нейронні мережі (Deep CNN), дозволили обробляти великі обсяги даних у режимі реального часу. Повторювані мережі, які зберігають контекст послідовностей даних, відкрили нові можливості для обробки тексту, мови та відео.

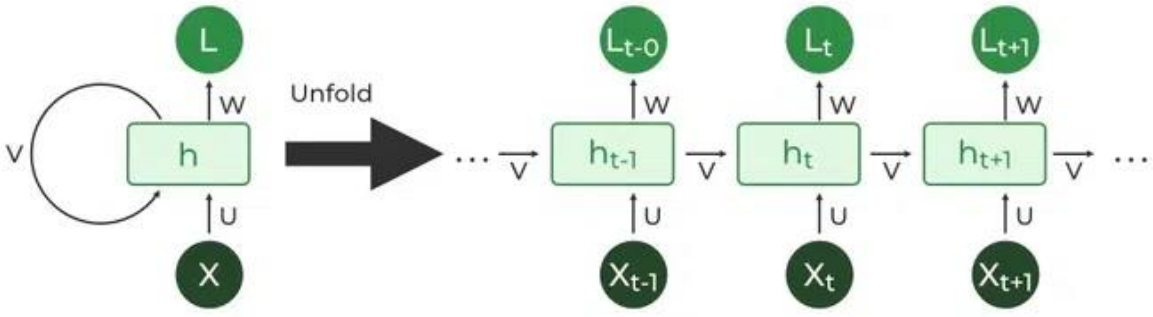

Фото 3 структура RNN нейромережі

В останні роки однією з найбільш революційних архітектур стали трансформери, вперше представлені у 2017 році в статті "Attention is All You

Need". Трансформери використовують механізм уваги для встановлення зв'язків між елементами вхідної послідовності, що дозволяє досягти кращої роботи у задачах

Нижче наведено таблицю перерахування основних моделей нейромереж, які показували кращий результат для свого часу і зробили важливий внесок у розвиток нейроніх мереж.

Назва	**Рік**	**Точність**	**Опис**	**Тип задачі**
LeNet-5	1998	98.2% на MNIST[див. додаток В]	Перша успішна нейромережа для розпізнавання рукописних цифр, побудована Яном Лекуном. Включає кілька згорткових шарів, що дозволяють розпізнавати складні образи.	Розпізнавання рукописних символів
Sparse Coding	1999	~99.1% на MNIST	Одна з перших моделей глибокого навчання, що використовувала нелінійність для ефективного кодування даних. Підходить для стиснення та декодування зображень і сигналів.	Кодування та стиснення даних
Echo State Network	2001	N/A	Перша нейромережа з випадковим зв'язком, яка використовувалась для моделювання динамічних систем, таких як зміна часового ряду. Рекурентна структура дозволяє обробляти послідовні дані.	Моделювання динамічних систем та рядів
Deep Belief Network	2006	99.3% на MNIST	Модель, що використовувала ієрархічну структуру прихованих шарів для передтренування, що покращує стабільність та точність нейронної мережі. Використовується для задач класифікації та кластеризації.	Розпізнавання образів, класифікація

AlexNet	2012	63.3%	Започаткувала популярність глибокого навчання. Використовує обширні згорткові шари та значно збільшує обчислювальні вимоги.	Розпізнавання образів
ZFNet	2013	62.5%	Виграла конкурс ImageNet 2013 та була модифікацією архітектури AlexNet.	Розпізнавання образів
Inception (GoogLeNet)	2014	74.8%	Архітектура від Google, яка використовує "Inception" блоки для ефективного використання обчислювальних ресурсів. Ця модель зменшує кількість параметрів, зберігаючи високу точність.	
DenseNet	2016	77.3%	Використовує щільні зв'язки між шарами, що покращує градієнтний потік і дозволяє глибші архітектури без втрати точності. Застосовується для задач розпізнавання образів.	Розпізнавання образів
GPT-3	2020	N/A	Модель генеративного перетворювача (Generative Pre-trained Transformer) від OpenAI з 175 мільярдами параметрів. Використовується для обробки природної мови та генерації тексту.	Обробка природної мови, генерація тексту
DALL-E 2	2022	N/A	Генеративна модель, здатна створювати зображення за текстовими описами, використовуючи знання про зображення та текст одночасно. Спрямована на задачі генерації образів.	Генерація образів за текстом

Застосування нейромереж у різних галузях

Нейронні мережі можна застосовувати практично в кожній галузі та сфері людської діяльності, включаючи економіку, медицину, зв’язок, системи безпеки та обробку інформації та в багатьох інших.

Основні завдання які можуть виконувати неромережі:

Розпізнавання образів: нейронні мережі успішно застосовуються для розпізнавання образів у зображеннях. Це може включати в себе завдання, такі як розпізнавання осіб, об'єктів або навіть більш складних структур в медичних зображеннях.

Обробка природної мови: в області обробки природної мови (NLP), нейронні мережі використовуються для створення систем автоматичного перекладу, аналізу текстів, генерації мови і навіть відповідей на питання. Вони здатні імітувати розуміння і використання людської мови.

Аналіз даних: нейромережі ефективні під час аналізу великих обсягів даних. Вони можуть виявляти закономірності, тенденції та приховані взаємозв'язки, що робить їх цінними інструментами в галузі аналітики даних.

Прогнозування: здатність нейронних мереж передбачати результати на основі наявних даних робить їх ефективними інструментами для прогнозування, у фінансах, метеорології або інших галузях.[7]

Основні сфери використання нейромереж:

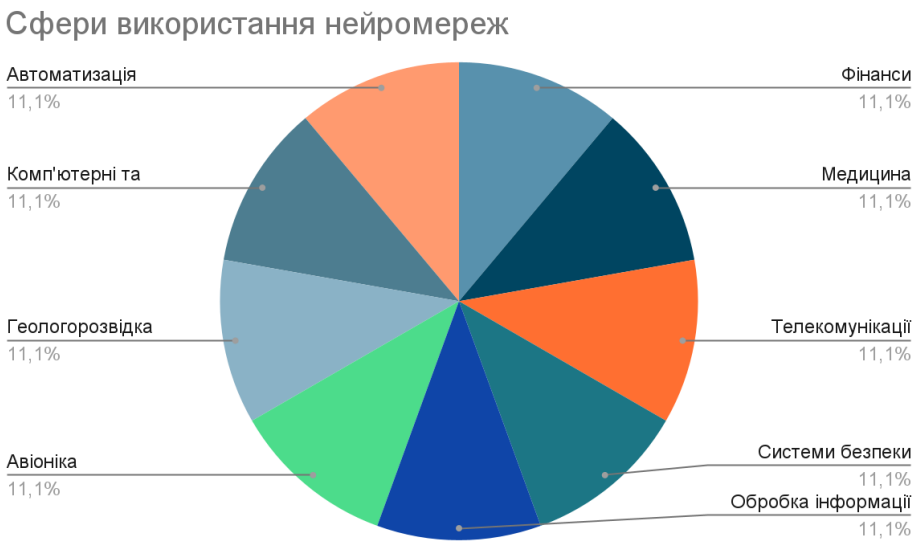

Фото 4 Сфери використання нейромереж

У сфері економіки, зокрема у фінансах, яскравим прикладом застосування нейронних мереж є системи управління кредитними ризиками, які успішно використовуються кількома великими банками США. Для оцінки ймовірності збитків від прострочених або несплачених кредитів банки проводять попередні кредитні розрахунки фінансової надійності позичальника. Ці розрахунки враховують такі фактори, як кредитна історія, тенденції розвитку бізнесу, стабільність ключових фінансових показників та багато інших факторів. Технології нейронної мережі забезпечують ефективний і точний аналіз, допомагаючи точніше ідентифікувати потенційних неплатників.[7]

Інші важливі застосування нейронних мереж в економіці включають прогнозування тенденцій на фондовому ринку, оцінку власності, прогнозування динаміки обмінного курсу, оптимізацію товарних і грошових потоків, а також автоматичне зчитування чеків і форм.

У медицині нейронні мережі в основному використовуються для діагностики захворювань. Наприклад, пакет кардіологічного діагностичного програмного забезпечення, розроблений R Informati, широко використовується в лікарнях Великобританії для раннього виявлення інфаркту міокарда та інших серцево-судинних захворювань, тим самим допомагаючи зменшити їх поширеність.

Нейромережева технологія також використовується в онкологічній діагностиці. Дослідники з Університету Дьюка (США) розробили нейронну систему для виявлення злоякісних тканин, яка успішно використовується в діагностиці раку молочної залози.

У телекомунікаціях нейронні мережі мають практичне застосування при проектуванні та оптимізації комунікаційних мереж. Вони ефективно вирішують критичні проблеми в галузі телекомунікацій, такі як пошук оптимальних маршрутів трафіку між вузлами. Окрім керування маршрутизацією трафіку, нейронні мережі також використовуються в

розробці нових телекомунікаційних мереж і для ефективного кодування даних, декодування та стиснення відео.

У системах безпеки та спостереження нейронні мережі відіграють вирішальну роль у ідентифікації людей, розпізнаванні голосу та обличчя, ідентифікації людей у натовпі, розпізнаванні номерних знаків, аналізі аерофотознімків та супутникових зображень, моніторингу потоків інформації та виявленні підробок.

В обробці інформації нейронні мережі можна використовувати для обробки чеків, розпізнавання підписів, відбитків пальців і голосових шаблонів.

Італійська компанія R Informati розробила пакет нейронної мережі FlexR d, який використовується для розпізнавання та автоматичного введення рукописних платіжних документів і податкових декларацій. У випадку платіжних документів система розпізнає не лише кількість і ціни позицій, але й формат документа. Для податкових декларацій він визначає фіскальні коди та суми податків.

Інші сфери використання нейромереж:

Авіоніка: автопілоти, що навчаються, розпізнавання сигналів радарів, адаптивне пілотування сильно пошкодженого літака, безпілотні літальні апарати (дрони), розпізнавання/детекція об'єктів на фото/відеозйомці з дрона.

Робототехніка: розпізнавання сцени, об'єктів і перешкод перед роботом, прокладання маршруту руху, керування маніпуляторами (наприклад, розв'язання оберненої задачі кінематики), підтримання рівноваги.

Геологорозвідка: аналіз сейсмічних даних, асоціативні методики пошуку корисних копалин, оцінка ресурсів родовищ.

Автоматизація виробництва: оптимізація режимів виробничого процесу, контроль якості продукції, моніторинг і візуалізація багатовимірної диспетчерської інформації, попередження аварійних ситуацій.[6]

Комп'ютерні та настільні ігри: створення нейрогравців(ботів)

Отже, можна зробити висновок, що нейромережі за рахунок їх високої адаптивності, здатності до самонавчання, обробки великих обсягів даних та виявлення складних залежностей широко використовуються в самих різних сферах.

РОЗДІЛ 2 ОГЛЯД ІСНУЮЧИХ СИСТЕМ АВТОМАТИЗОВАНОГО СПОСТЕРЕЖЕННЯ ЗА ТРАНСПОРТНИМ РУХОМ ТА ВИЯВЛЕННЯ ПРАВОПОРУШЕНЬ

Огляд існуючих рішень для автоматизованого спостереження за транспортним рухом.

В наш час нейромережеві технології стрімко розвиваються, знаходячи своє застосування в багатьох галузях, включаючи фінанси, охорону здоров'я, логістику та електронний документообіг. Традиційні системи обробки даних часто стикаються з проблемами обмеженої обчислювальної потужності, недостатньої гнучкості в адаптації до нових завдань і нездатності ефективно обробляти великі обсяги інформації. Нейронні мережі можуть вирішити ці проблеми завдяки своїй здатності до самонавчання, адаптивності та високій точності прогнозування, яка досягається завдяки потужним алгоритмам машинного навчання.

Вже існує кілька нейромережевих рішень, які спеціалізуються на автоматизованому моніторингу трафіку. Такі системи можуть обробляти відео в реальному часі, розпізнавати об'єкти на дорозі, прогнозувати можливі затримки або аварії, оптимізувати маршрути. Завдяки нейронним мережам стає можливим створення інтелектуальних систем моніторингу, які значно підвищують безпеку, знижують витрати на логістику та забезпечують оперативний контроль за транспортними потоками.

NVIDIA Metropolis — це потужна платформа, яка використовує ШІ та глибоке навчання для аналізу відеопотоків у реальному часі. Розроблений для інфраструктури розумного міста, він виявляє порушення правил дорожнього руху, стежить за перехрестями та аналізує схеми руху. Завдяки високопродуктивним графічним процесорам NVIDIA Metropolis забезпечує

точне розпізнавання об’єктів і аналіз поведінки, підвищуючи безпеку дорожнього руху та управління дорожнім рухом. Це рішення ідеально підходить для масштабних міських проектів, але вимагає значної потужності сервера та щомісячного обслуговування.[8]

Hikvision AI Traffic Monitoring — це система відеоспостереження, орієнтована на моніторинг дорожнього руху в режимі реального часу та виявлення порушень, зокрема перевищення швидкості, проїзд на червоне світло та неправильне використання смуги руху. Ця система широко використовується в міських проектах, таких як «Безпечне місто», завдяки готовим до інтеграції камерам Hikvision. Він автоматично сповіщає правоохоронні органи про порушення та потребує мінімальних серверних ресурсів, що робить його економічно ефективним для середніх і великих міст.[9]

OpenALPR — це система розпізнавання номерних знаків, яка інтегрується з камерами для ідентифікації транспортних засобів у режимі реального часу. Широко використовується для керування паркуванням, контролю доступу та моніторингу руху, він ефективно розпізнає номерні знаки транспортних засобів. OpenALPR пропонує гнучкі ціни з базовими місячними та корпоративними ліцензіями. Він підтримує легку інтеграцію з існуючими камерами, забезпечуючи високу точність і знижуючи витрати на налаштування.[10]

BriefCam — це платформа відеоаналітики, яка використовує ШІ для ефективного аналізу великих обсягів відеоданих. Він пропонує виявлення об’єктів, відстеження інцидентів і аналіз поведінки, що робить його популярним для громадської безпеки та моніторингу руху. Завдяки розширеній фільтрації BriefCam підтримує поглиблений аналіз інцидентів і потребує значної потужності сервера. Щомісячні витрати покривають підтримку та оновлення, що робить його придатним для поточних проектів безпеки.[11]

SenseTime - система використовує глибоке навчання для аналізу дорожнього руху та виявлення порушень, таких як перевищення швидкості та проїзд на червоне світло. Розроблений для розгортання у великих містах, він інтегрується з системами управління містом для контролю потоку транспорту та забезпечення безпеки на дорогах. SenseTime вимагає надійної серверної інфраструктури та пропонує хмарну підтримку, що робить його ефективним для моніторингу трафіку в реальному часі та реагування на інциденти.[12]

Huawei TrafficGo — це рішення на основі штучного інтелекту для моніторингу та управління трафіком, розроблене для великих міст. TrafficGo виявляє порушення, такі як перевищення швидкості та проїзд на червоне світло, ефективно керуючи транспортним потоком. Інтегрований із камерами та програмним забезпеченням Huawei, він забезпечує надійну роботу та високу точність. Основні витрати включають обладнання та програмне забезпечення з додатковим щомісячним зберіганням даних і підтримкою плавного моніторингу в режимі реального часу.[13]

Порівняльний аналіз існуючих рішень

Було проведено порівняння існуючих рішень по настпним параметрам:

Основна мета – головне призначення системи (наприклад, моніторинг руху або фіксація порушень).

Виявлення порушень – види порушень, які система може автоматично фіксувати (перевищення швидкості, проїзд на червоне світло тощо).

Аналіз транспортного потоку – можливість системи обробляти та аналізувати інформацію про потоки транспорту в реальному часі.

Розпізнавання осіб – здатність системи розпізнавати обличчя, що дозволяє додатково контролювати безпеку.

Розпізнавання номерних знаків – функція автоматичного розпізнавання номерів для ідентифікації транспортних засобів.

Робота в реальному часі – здатність системи обробляти інформацію без затримок, що є критичним для оперативного реагування.

Підтримка мультимодальності – можливість одночасного аналізу різних типів даних (наприклад, транспорт і пішоходи).

Підтримка аналітики даних – наявність інструментів для аналітичної обробки даних, таких як статистика порушень і моделі прогнозування.

Управління трафіком – функціональність для оптимізації руху транспорту, як-от коригування світлофорів і маршрутизація потоків.

Точність виявлення – рівень точності, з яким система здатна визначати порушення і класифікувати об'єкти.

Масштабованість – можливість збільшення системи для обробки більшого обсягу даних або додавання нових елементів.

Система сповіщень – функція для автоматичних повідомлень або сигналів про зафіксовані порушення або інциденти.

Сумісність з обладнанням – типи камер і іншого обладнання, з якими сумісна система.

Використання ШІ та машинного навчання – наявність алгоритмів штучного інтелекту для підвищення точності і функціональних можливостей.

Варіанти застосування – сфери, де система може бути корисною, наприклад, безпека на дорогах, управління парковками.

Обробка великих обсягів даних – здатність обробляти великі потоки відео й інших даних, що особливо важливо для міст із високою транспортною інтенсивністю.

Захист даних і конфіденційність – заходи для забезпечення безпеки зібраних даних і захисту конфіденційної інформації, як-от шифрування й обмеження доступу.

Параметр	NVIDIA Metropolis	Hikvision AI Traffic Monitoring	OpenALPR	BriefCam	SenseTime Traffic Management System	Huawei TrafficGo
1. Основна мета	Аналіз і моніторинг дорожнього руху	Виявлення та фіксація порушень правил дорожнього руху	Розпізнавання номерних знаків	Комплексний аналіз відеопотоку	Контроль дорожнього руху та фіксація порушень	Забезпечення безпеки та управління трафіком
2. Виявлення порушень	Перевищення швидкості, проїзд на червоне світло	Перевищення швидкості, парковка, проїзд на червоне світло	Відстеження транспортних засобів	Аналіз руху, фіксація інцидентів	Проїзд на червоне світло, перевищення швидкості	Перевищення швидкості, проїзд на червоне світло
3. Аналіз транспортного потоку	Так	Так	Ні	Так	Так	Так
4. Розпізнавання осіб	Так	Ні	Ні	Так	Так	Ні
5. Розпізнавання номерних знаків	Ні	Так	Так	Ні	Так	Так
6. Робота в реальному часі	Так	Так	Так	Так	Так	Так
7. Підтримка мультимодальності	Так	Ні	Ні	Ні	Так	Ні
8. Підтримка аналітики даних	Так	Так	Ні	Ні	Так	Так
9. Управління трафіком	Обмежено	Обмежено	Ні	Обмежено	Так	Так
10. Точність виявлення	Висока	Середня	Середня	Висока	Висока	Висока
11. Масштабованість	Висока	Середня	Висока	Висока	Висока	Висока
12. Система сповіщень	Так	Так	Так	Так	Так	Так
13. Сумісність з обладнанням	Пристрої, сумісні з NVIDIA	Камери Hikvision	Камери з підтримкою LPR	Різні камери	Камери з підтримкою AI	Камери Huawei

14. Використання ШІ та машинного навчання	Так (комп'ютерне бачення, ШІ)	Так (ШІ для виявлення порушень)	Так (машинне навчання)	Так (комп'ютерне бачення, ШІ)	Так (глибоке навчання)	Так (ШІ для трафіку)
15. Варіанти застосування	Безпека на дорогах, управління перехрестями	Автоматична фіксація порушень правил дорожнього руху	Управління парковкою, розпізнавання номерів	Обробка відеоданих, аналіз поведінки	Фіксація порушень і контроль дорожнього руху	Управління дорожнім рухом, контроль безпеки
16. Обробка великих обсягів даних	Так	Так	Обмежено	Так	Так	Так
17. Захист даних і конфіденційність	Так (шифрування, захищене зберігання даних)	Так (шифрування)	Так (шифрування, захист даних)	Так (серверний захист і шифрування даних)	Так (захист даних у хмарі та локально)	Так (шифрування даних та захищене зберігання)

Також була також проаналізована ціна інтеграції та використання цих систем для 1000 камер:

Система	Одноразові витрати (ліцензії, обладнання)	Щомісячні витрати (зберігання, обслуговування)	Примітки
NVIDIA Metropolis	$3,000,000 - $5,000,000	$60,000 - $100,000	Потрібні сервери NVIDIA, вартість залежить від обсягу даних і ліцензій на програмне забезпечення
Hikvision AI Traffic Monitoring	$1,000,000 - $2,500,000	$30,000 - $50,000	Включає камери Hikvision, які часто інтегровані в систему «Безпечне місто».
OpenALPR	$500,000 - $1,500,000	$39 - $100/камера ($39,000 - $100,000)	Вартість залежить від кількості камер і ліцензій; використовується для контролю паркування та моніторингу дороги
BriefCam	$2,000,000 - $4,000,000	$50,000 - $80,000	Базові витрати на ліцензії та потужність сервера для аналізу відеопотоку

SenseTime Traffic Management System	$2,500,000 - $4,500,000	$70,000 - $120,000	Включає камери та програмне забезпечення SenseTime, вимагає потужних серверних ресурсів для поглибленого аналізу
Huawei TrafficGo	$1,500,000 - $3,500,000	$40,000 - $70,000	Залежить від кількості камер і ліцензій, які використовуються для автоматизації та управління трафіком

Порівняння показує, що різні системи автоматизованого відстеження порушень дорожнього руху мають специфічні переваги та функціональність.

NVIDIA Metropolis та SenseTime Traffic Management забезпечують високу точність і масштабованість, що підходить для великих міських проектів із потребою обробки великих даних у реальному часі. Проте такі рішення потребують значних початкових інвестицій.

Hikvision AI Traffic Monitoring і Huawei TrafficGo є більш доступними для середніх міст. Вони ефективно інтегруються в інфраструктуру «Безпечного міста» та виявляють основні порушення, але мають менший набір функцій.

OpenALPR спеціалізується на розпізнаванні номерних знаків, що робить її бюджетним рішенням для локальних проектів, таких як паркування.

Таким чином, з функціональної точки зору найкращим варіантом для інтеграції на даний момент є NVIDIA Metropolis, але значним недоліком цієї системи є її висока вартість.

Використання в Україні

На сьогоднішній день в Україні використовуються деякі з перелічених систем для автоматизованого контролю за транспортом, особливо в межах великих міст і в рамках програми «Безпечне місто». Основними системами,

що використовуються, є Hikvision AI Traffic Monitoring і частково OpenALPR, а також окремі рішення від Huawei у вигляді пілотних проектів. Ось короткий огляд використання цих систем і їхні недоліки:

Hikvision AI Traffic Monitoring

Камери Hikvision встановлені в багатьох містах України, включаючи Київ, Харків і Львів, де вони допомагають фіксувати порушення правил дорожнього руху, такі як перевищення швидкості, проїзд на червоне світло та паркування в заборонених місцях. Основний недолік — залежність від централізованого зберігання даних, що потребує значних обчислювальних ресурсів та безперервної підтримки. Система також має обмеження у виявленні складних порушень і поки не інтегрується з іншими аналітичними платформами, що знижує її ефективність.

OpenALPR

Використовується для автоматизованого розпізнавання номерних знаків, зокрема, у Києві та Одесі. Система встановлена на парковках і в зонах контролю проїзду, де дозволяє ефективно керувати доступом та ідентифікувати порушників. OpenALPR підходить для вирішення локальних завдань, таких як контроль паркування або розпізнавання номерних знаків, але її можливості обмежені в масштабних проектах з моніторингу дорожнього руху. Система не підтримує виявлення складних порушень, таких як неправильне маневрування або порушення розмітки, що обмежує її застосування.

Huawei TrafficGo

В окремих містах України Huawei TrafficGo використовується в рамках експериментальних проектів для аналізу трафіку і базового контролю дорожніх порушень. Система добре інтегрується з іншими рішеннями Huawei, що робить її потенційно корисною для великих міських проектів.

Основний мінус — висока вартість обслуговування і залежність від екосистеми Huawei. Крім того, система має обмежену підтримку для інтеграції з українськими державними системами, що може вплинути на швидкість і ефективність передачі даних.

Як можна зазначити які з існуючих систем вже активно використовуються в Україні. Основні недоліки цих систем пов'язані з високими витратами на обслуговування, обмеженнями в інтеграції з іншими національними системами та залежністю від централізованих ресурсів. Також, у деяких випадках можливості системи обмежені типами порушень, які вона може фіксувати, що знижує її ефективність для повного контролю за дорожнім рухом.

РОЗДІЛ 3 РОЗРОБКА СИСТЕМИ ДЛЯ АВТОМАТИЗОВАНОГО СПОСТЕРЕЖЕННЯ ЗА ТРАНСПОРТНИМ РУХОМ З ПОДАЛЬШИМ ВИЯВЛЕННЯМ ПРАВОПОРУШЕНЬ

Концепція та загальний огляд програми.

Після проведеного аналізу існуючих програм для автоматизованого спостереження за транспортним рухом і виявлення порушень, таких як NVIDIA Metropolis, Hikvision AI Traffic Monitoring і OpenALPR, було визначено їхні сильні й слабкі сторони. Основними перевагами цих рішень є висока точність виявлення порушень, швидкість обробки даних у реальному часі та здатність автоматично фіксувати порушення без участі людини, що значно підвищує ефективність контролю за дорожнім рухом. Однак, одним із найбільших викликів для користувачів є складність інтеграції таких рішень у вже існуючу дорожню інфраструктуру та системи безпеки, зокрема через високі витрати на обладнання та потребу в потужних серверних ресурсах. Це означає, що міста й підприємства прагнуть підвищити безпеку та автоматизувати моніторинг, але стикаються з бар'єрами при впровадженні нових технологій у поточні процеси.

Тож було принйто рішення розробити власну більш доступну версію відслідковування провопорушень та спостереження за транспортним рухом.

Концепція системи полягає в об'єднанні основних та найбільш затребуваних функцій аналогів з доступною ціною та відносно простою інтеграцією

Як це має працювати на практиці:

1. **Збір даних за допомогою камери**

Камери встановлюються на ключових ділянках (перехрестя, дороги з високою швидкістю руху) і підключаються до джерела живлення та локальної мережі.

Камера постійно знімає відео і передає його в цифровому форматі (наприклад, у форматі H.264 або H.265), оптимізованому для зменшення обсягу даних при передачі.

2. **Передача відео по мережі**

Камери можуть бути підключені до локальної мережі через Ethernet або Wi-Fi. Це забезпечує швидку передачу відеопотоку на найближчий вузол обробки або на центральний сервер.

Для передачі відео використовується протокол RTSP (Real-Time Streaming Protocol) або аналогічний, що дозволяє безперервно передавати відео в реальному часі з мінімальною затримкою.

3. **Передача на основний сервер через мережу Інтернет або виділену мережу**

Відео передається на центральний сервер обробки, де встановлено програмне забезпечення для аналізу відеопотоку.

4. **Обробка даних на сервері**

На сервері працює нейромережева модель, яка в режимі реального часу аналізує відео, розпізнає транспортні засоби, виявляє порушення (перевищення швидкості, проїзд на червоне світло тощо). Паралельно із розпізнаванням об'єктів на відео запускається модуль для розпізнавання номерних знаків. Це дозволяє ідентифікувати порушника, якщо порушення було зафіксовано.

5. **Зберігання та обробка метаданих**

У випадку виявлення порушення сервер автоматично створює запис у базі даних, що включає дату, час, тип порушення, номерний знак автомобіля та відеофрагмент або зображення. Відеозапис порушення або зображення зберігаються на сервері або в хмарному сховищі.

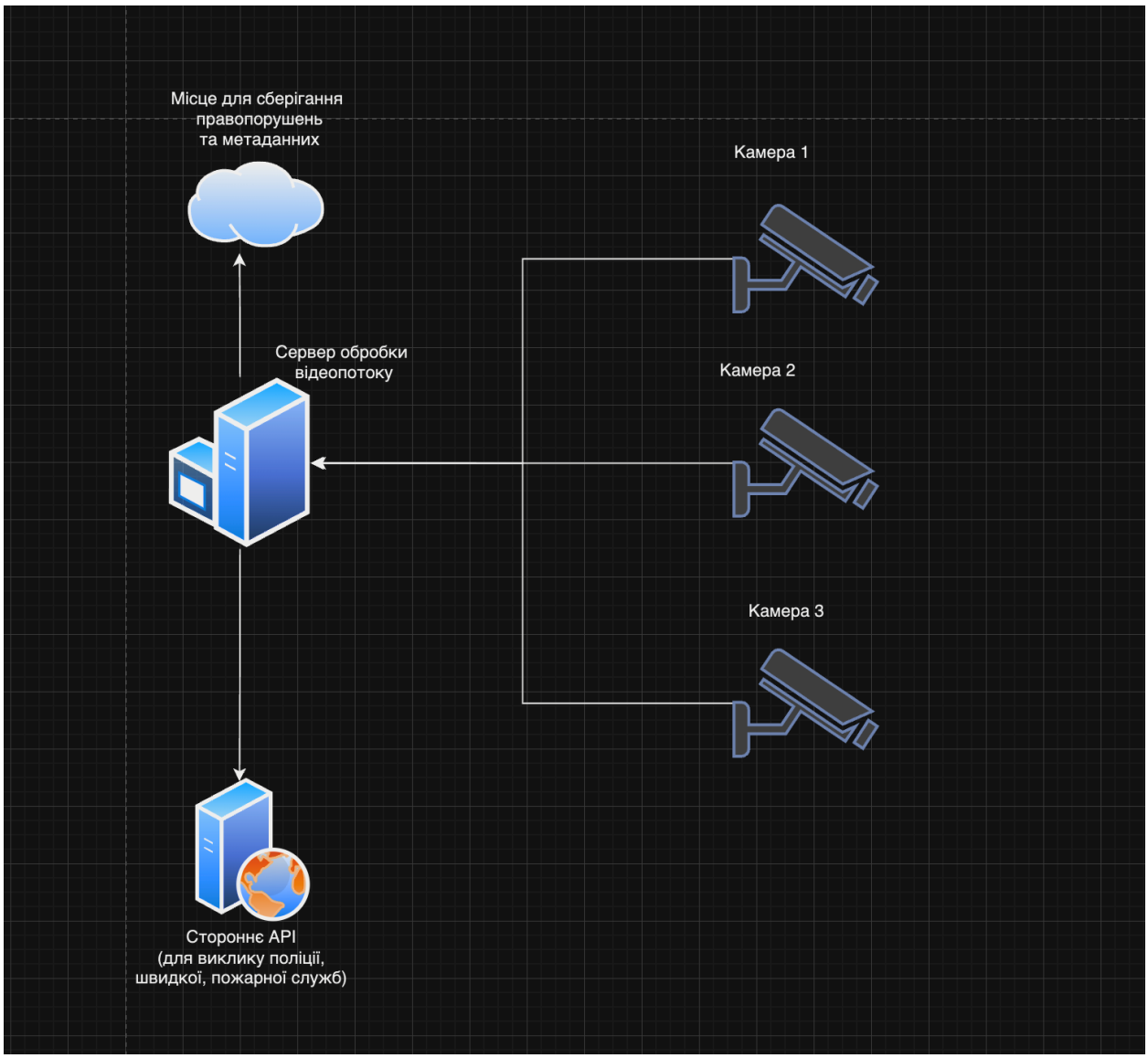

Фото 5 Схематичне відображення роботи системи

Для коректного функціонування система має наступні технічні потреби:

1. **Система фіксації:**

Камери мають підтримувати роздільну здатність мінімум 1080p, щоб забезпечити високу якість зображення, що дозволяє розпізнавати номерні знаки, обличчя та інші деталі. Камери також повинні мати інфрачервоне підсвічування або інші технології нічного бачення для фіксації порушень у темний час доби.

Частота кадрів має бути не меншою за 25-30 кадрів в секунду, щоб забезпечити плавність відео і точність виявлення руху.

2. **Серверні ресурси та обчислювальна потужність**

Для швидкої обробки відеопотоку в реальному часі потрібні потужні сервери з графічними процесорами (GPU), наприклад, NVIDIA з підтримкою CUDA для прискорення нейромереж.

Мережа повинна підтримувати високу пропускну здатність для передачі відео з великої кількості камер у реальному часі, особливо для міст з великою кількістю точок спостереження.

3. **Зберігання данних**

Необхідне зберігання для збереження великої кількості відеозаписів і метаданих. Залежно від тривалості зберігання даних, можуть бути потрібні локальні сервери або хмарне сховище.

Вимоги до системи та ключові технічні аспекти.

Прототипом програмного продукту системи для автоматизованого спостереження за транспортним рухом і виявлення правопорушень стане її основний компонент — нейромережева система для обробки та аналізу даних.

Ключові функції, які має виконувати програма:

Первинна обробка зображення: Здійснення нормалізації зображення, включаючи корекцію кольорів, освітлення та масштабування, щоб забезпечити однакові умови для подальшого аналізу та підвищення точності розпізнавання.

Детекція транспортних засобів: Виявлення транспортних засобів на зображенні для подальшого трекінгу. Це включає ідентифікацію кожного транспортного засобу в кадрі з фіксацією його позиції для подальшого відстеження.

Аналіз положення в просторі: Визначення просторового розташування транспортних засобів для подальшого відстеження змін у їхній позиції, що дозволяє обчислювати швидкість та напрямок руху кожного об’єкта.

Сегментація транспортних засобів та виділення номерного знаку: Розділення зображення для точного виділення транспортних засобів і номерних знаків. Сегментація дозволяє виділити номерний знак для подальшої ідентифікації транспортного засобу за допомогою системи розпізнавання номерів.

РОЗДІЛ 4: ТЕХНІЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

Вибір мови розробки та платформи розробки аналіз існуючих архітектур.

Створення додатку складалось з декількох етапів: постановки задачі, дослідження предметної області, постановки технічного завдання, аналізу інтсрументів та мов програмування, проектування додатку, написання коду, аналізу роботи програми, доопрацювання програми на основі тестування.

Основною задачею додатку є детекція та просторовий аналіз транспортних засобів, а також виявлення та розпізнавання їх номерних знаків для ідентифікації порушників. Для досягнення цієї мети потрібно використовувати три послідовно виконувані нейронні мережі, кожна з яких виконує свою специфічну функцію:

Перша для виявлення транспортних засобів на зображенні (кадрі), що надходить з відеопотоку. Нейромережа повинна визначати позиції транспортних засобів, виділяючи кожен з них прямокутною рамкою (bounding box).

Друга, після того, як транспортний засіб було визначено на зображенні, друга нейромережа сегментує зображення автомобіля, виділяючи його окремі частини, серед яких нас більше всього цікавить номерний знак.

Та третя для розпізнавання символів на виділеному фрагменті номерного знаку, щоб ідентифікувати транспортний засіб.

Вибір мови розробки

Після затвердження технічного завдання треба визначитись з мовою написання додатку. Для написання неромереж можно використовувати настпні мови програмування: Python, JavaScript, C++ та Java [14]
Нижче навдена порівняльння цих мов за наступними параметрами

Параметр	Python	JavaScript	C++	Java
Швидкість виконання	Висока. Оптимізована завдяки використанню бібліотек на C/C++	Середня. Швидкість нижча, ніж у Python при складних обчисленнях	Висока. Відмінна швидкість для низькорівневих обчислень	Висока, але поступається C++ при оптимізації
Доступність фреймворків	Найбільша кількість: TensorFlow, PyTorch, Keras	Основний фреймворк — TensorFlow.js	Обмежена, але підтримує TensorFlow і Caffe	Основний фреймворк — Deeplearning4j
Простота розробки	Дуже висока. Простий синтаксис, інструменти для швидкого прототипування	Середня. Підходить для фронтенд-розробників, але обмежені інструменти	Низька. Складний синтаксис, більше підходить для оптимізації	Середня. Складна, але зручна для великих проєктів
Підтримка GPU	Висока. Добре інтегрується з CUDA, TensorRT для GPU	Обмежена. Підтримка GPU в основному в браузері	Висока. Підтримка CUDA для високої продуктивності	Обмежена, складніше налаштовувати для GPU
Зручність для досліджень	Висока. Найпопулярніший вибір для досліджень і прототипів	Середня. Підходить для невеликих веб-прототипів	Низька. Складна для швидкого прототипування	Низька. Рідко використовується для прототипів
Сумісність з апаратними засобами	Відмінна. Підтримка GPU, TPU, багатоядерних CPU	Обмежена, переважно в браузерах	Відмінна. Гнучкість у налаштуванні для різних систем	Середня, але підходить для великих проєктів
Підтримка спільноти	Дуже велика	Висока	Невелика	Помірна

За результатми проведеного порівняння було прийнято рішення використовувати мову програмувааня Python. Завдяки простоті, доступності потужних фреймворків і високій підтримці GPU.

Вибір бібліотеки для розробки нейромережі

Є три основні бібіліотеки для розроблення нейромереж на Python: TensoFlow, Keras, Pytorch. Для виявлення найращого варінту було проведено порівняльну харастектистику цих бібліотек за такими параметрами: [15,16,17]

Параметр	TensorFlow	Keras	PyTorch
1. Архітектура	Комплексний фреймворк для машинного навчання, працює на низькому рівні	Високоабстрактний інтерфейс для побудови нейронних мереж, що використовує TensorFlow як бекенд.	Динамічний і модульний, що дозволяє будувати моделі в режимі реального часу.
2. Простота використання	Відносно складний через низькорівневий синтаксис.	Дуже простий, інтуїтивний	Помірно простий.
3. Продуктивність і масштабованість	Висока продуктивність і масштабованість для великих проєктів.	Помірна продуктивність..	Висока продуктивність, особливо для дослідницьких і прототипних проєктів.
4. Динамічна / статична графіка	Статичний графік обчислень за замовчуванням (Graph Mode), що ускладнює відладку, але підвищує швидкість.	Використовує статичний графік через TensorFlow або Theano, але зручний завдяки високоабстрактному синтаксису.	Динамічний графік (Eager Execution), що спрощує відладку і адаптивність у реальному часі.
5. Відладка	Складна, потребує роботи з графом.	Легша завдяки простому API, але обмежена можливостями бекенду.	Легка відладка завдяки динамічному графіку, дозволяє використовувати стандартні інструменти Python.
6. Підтримка апаратного прискорення (GPU)	Відмінна підтримка GPU, CPU і TPU	Залежить від бекенду (зазвичай TensorFlow або Theano), тому підтримує GPU, але не оптимізований так сильно, як TensorFlow.	Чудова підтримка GPU, особливо для задач комп'ютерного зору, з нативною інтеграцією з CUDA.

7. Сфера застосування	Підходить для широкого спектру задач: від дослідницьких проєктів до великих комерційних додатків.	В основному використовується для прототипування та швидкого моделювання.	Поширений у дослідницьких проєктах, комп’ютерному зорі та NLP, а також для глибокого навчання.
8. Підтримка спільноти та документація	Велика	Велика	Найбільша
9. Розгортання в продакшн	Відмінний для продакшн-рішень завдяки TensorFlow Serving, TensorFlow Extended (TFX) і підтримці TPU.	Більше підходить для прототипування, менш гнучкий у розгортанні великих продакшн-систем.	Придатний для продакшн, але більше застосовується для дослідницьких проєктів.
10. Наявність попередньо навчених нейромереж для задач комп'ютерного зору	Має численні попередньо навчені моделі, включаючи ResNet, Inception, MobileNet, здебільшого для задач класифікації та сигментації	Не багато	Підтримує велику кількість попередньо навчених моделей, як-от ResNet, Faster R-CNN, YOLO, що робить його зручним для задач комп'ютерного зору.

За результатами проведеного дослідження було вирішено використовувати бібліотеку PyTorch як основний інструмент для розробки нейронної мережі. Це рішення ґрунтується на кількох ключових перевагах PyTorch, таких як швидкість роботи, простота використання та відладки, а також наявність великої кількості попередньо навчених нейромереж для задач комп'ютерного зору, що робить його оптимальним вибором для таких завдань.

Середовище розробки

В якості середовище для розроблення було обрано Jupyter Notebook. Цей інструмент був спеціально розроблений для поліпшення зручності роботи з великими обсягами даних. Завдяки тому що код можна поділити на кластери, кожен з яких після виконання буде мемоізуватися, а під час перезапуску програми результати виконання лежатимуть у кеші пристрою,

що дуже прискорить роботу з обробкою великої кількості даних.[SEP]Також такий підхід значно спрощує процес дебагінгу то логування адже ми можемо почати виконання программи з будь якої ноди.[18]

Опис архітектури і основних компонентів.

Після того як технології розробки програми були обрані, та їх використання затверджене, був розписаний план взаємодії усіх цих програм та компонентів які разом утворювали архітектуру платформи.

Вибір архітектури для неромережі відслідковування транспортних засобів.

Наразі існує 5 найбільш популярних архітектур для відстеження об'єктів (object detection / object tracking): YOLO, Faster R-CNN, SSD, EfficientDet, DeepSORT. Нижче наведена порівняльна таблиця цих архітектур.[19,20]

Архітектура	YOLO	Faster R-CNN	SSD	EfficientDet	DeepSORT
Сумісність з PyTorch	Підтримується	Підтримується	Підтримується	Підтримується	Підтримується
Швидкість (FPS)	Дуже висока (до 45+ FPS)	Середня (до 7–10 FPS)	Висока (до 22–24 FPS)	Висока (18-22 FPS)	Висока для трекінгу об'єктів після виявлення
Точність (mAP)	60–65%	70%	40–45%	70–75%	50–55%
Складність навчання та налаштування	Простий у налаштуванні	Складний	Складний	Складний	Середній
Ресурси обчислення	Помірні для YOLOv5, оптимізований для GPU	Високі (вимагає значних обчислювальних потужностей)	Помірні, працює на CPU з меншою ефективністю	Високі, оптимізований для GPU	Невисокі, може працювати в реальному часі на GPU

За результатами дослідження було обрано архітектуру YOLO. Вона забезпечує відносно високу точність і водночас дозволяє швидко та оптимально обробляти відеопотік.

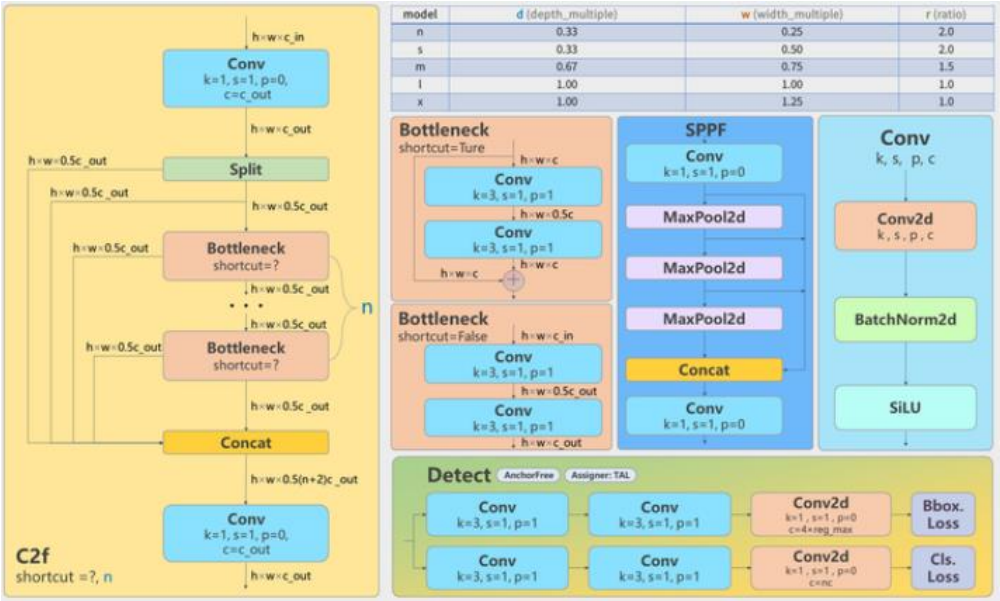

Фото 6 YOLOV8 архітектура

Вибір архітектури для сегментації транспортних засобів.

Наразі існує 4 найбільш розповсюджених архітектур для задач сегментації: U-Net,DeepLabV3+,Mask R-CNN,FCN.Нижче наведена порівняльна таблиця цих архітектур.[21,22,23]

Архітектура	**U-Net**	**DeepLabV3+**	**Mask R-CNN**	**FCN**
Сумісність з PyTorch	Підтримується	Підтримується	Підтримується	Підтримується
Точність сегментації (mIoU)	~75–80% для транспортних об'єктів	~85–90% для сегментації транспорту	~80–85% для складних об'єктів на зображенні	~65–70% для базових задач сегментації
Швидкість обробки	Відносно висока, але менш ефективний на великих зображеннях	Середня, ресурсомісткий для великих обсягів даних	Середня, висока обчислювальна потреба	Висока для стандартних застосувань

Масштабованість	Добра для невеликих наборів даних	Висока для великих наборів даних	Висока, але потребує значних ресурсів	Добра для невеликих та середніх наборів даних
Ресурси обчислення	Помірні, ефективний на GPU	Високі, оптимізований для GPU	Дуже високі, потребує значного часу і ресурсів	Середні, ефективний для CPU та GPU

За результатами дослідження було обрано архітектуру U-Net. Вона забезпечує відносно високу точність і водночас є найменш ресурсомісткою серед інших архітектур.

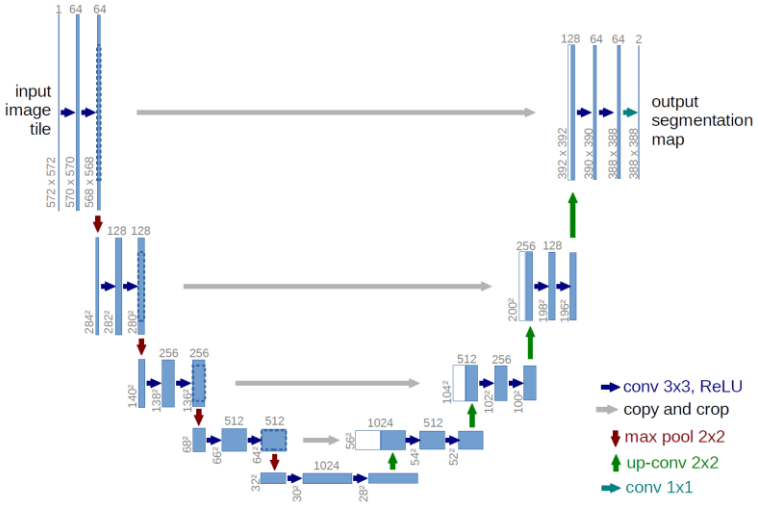

Фото 7 Архітектура U-Net

Вибір архітектури для розпізнавання номерного знаку.

Наразі існує 4 найбільш розповсюджених архітектур для розпізнавання нормерних знаків: CRNN,YOLO-LPR,ANPRNet,DenseNet. Нижче наведена порівняльна таблиця цих архітектур.[24,25]

Архітектура	CRNN	YOLO-LPR	ANPRNet	DenseNet
Сумісність з PyTorch	Підтримується	Підтримується	Не підтримується	Підтримується

Точність розпізнавання	90–95%	85–90%	80–85%	95%
Швидкість обробки	Висока на невеликих наборах даних	Дуже висока, обробка в реальному часі	Висока, оптимізований для мобільних пристроїв	Середня
Застосування	Розпізнавання тексту	Обробка в реальному часі, відеоспостереження	Розпізнавання тексту	Розпізнавання тексту
Масштабованість	Відмінна для великих наборів даних	Висока для складних об'єктів і великих потоків даних	Середня	Відмінна для обробки великих наборів даних
Ресурси обчислення	Помірні, ефективний на GPU та CPU	Помірні для реального часу на GPU	Низькі, адаптований для мобільних процесорів	Високі, потребує значних GPU-ресурсів

За результатами дослідження було обрано архітектуру YOLO-LPR, оскільки вона демонструє високу точність розпізнавання і є однією з найбільш оптимізованих за використанням ресурсів

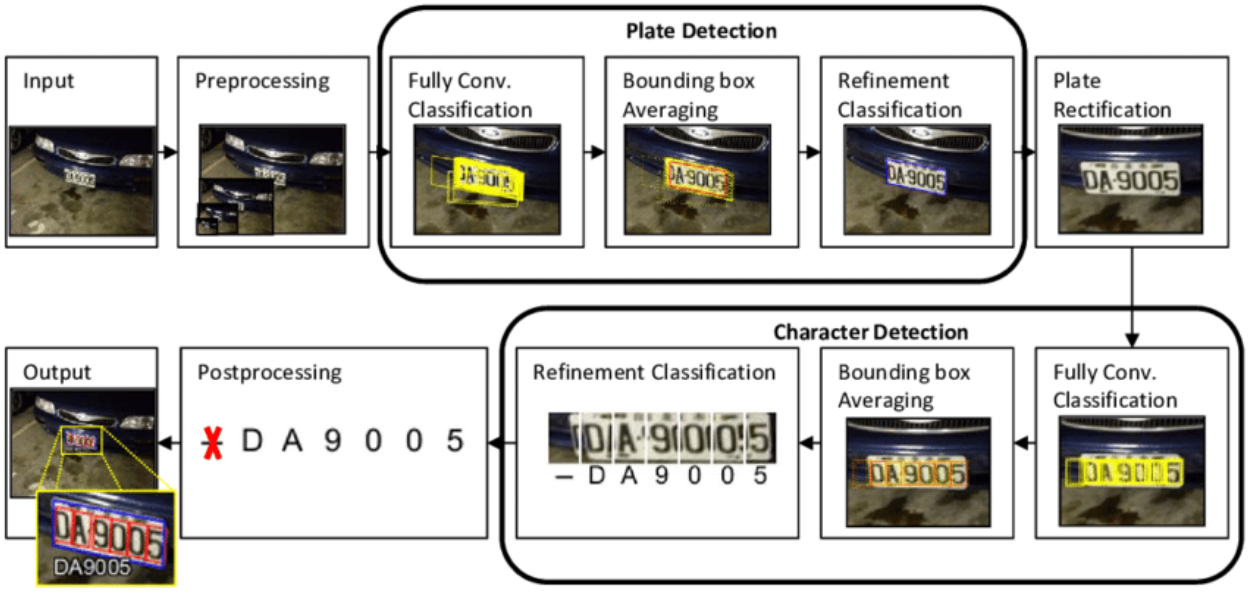

Фото 8 Принцип роботи YOLO-LPR

Навчання нейромереж.

Після вибору архітектур для наших нейронних мереж необхідно інтегрувати їх у систему, додатково навчити та відкалібрувати відповідно до цілей нашого проєкту.

Процесс навчання нейромережі складається з трьох основних етапів:

- Збір, нормалізація та сортування даних для набору даних для навчання
- Написання архітектури нейронної мережі
- Навчання неронної мережі та подальше калібрування конфігурацій навчання

У нашому випадку, оскільки ми працюємо з готовими архітектурами, наше завдання полягає в донавчанні нейромереж на власному датасеті та калібруванні параметрів і гіперпараметрів цих архітектур.

Збір даних для датасету

Для донавчання нейромереж було прийнято рішення взяти найбільш популярні датасети відповідних категорій з платформ PapersWithCode[26] та Kaggle [27]

Таким чином для донавчання YOLO було обрано датасет VEDAI (Vehicle Detection in Aerial Imagery) [28] який включає в себе більш ніж 4 тисячи зображень транпортних засобів та їх обмежувальні рамки

Для архітектури U-Net було обрано датасет “Car segmentation”[29] який включає в себе більше 300 зображень з масками сегментації.

Для архітектури YOLO-LPR було обрано датасет UFPR-ALPR[30] який включаэ себе більш ніж 4500 зображень номерних знаків.

Кожен з датасетів було розділено на дві папки навчання та валідації в пропорції 80 на 20 процентів

Інтеграція та колібрація моделей

Нейромережі були інтегровані та оремо навчені на навчальних наборах данних. Кожна мережа пройшла етап донавчання на обраних датасетах, щоб адаптуватися до специфіки задачі та підвищити точність виявлення й розпізнавання об’єктів.

Нижче приведено фрагент коду - інтеграції моделі YoloV8

```
from ultralytics import YOLO
import cv2
import torch
from torch.utils.data import Dataset, DataLoader
import os
import glob
import matplotlib.pyplot as plt

class ImageDataset(Dataset):
    def __init__(self, folder_path):
        self.image_paths = glob.glob(os.path.join(folder_path, '*.jpg'))

    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        image_path = self.image_paths[idx]
        image = cv2.imread(image_path)
        image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
        return image_rgb, image_path

dataset = ImageDataset('./dataset/yolo/training')
dataloader = DataLoader(dataset, batch_size=1, shuffle=False)

model = YOLO('yolov8n.pt')

for images, paths in dataloader:
    results = model(images[0].numpy())
    plt.imshow(results[0].plot())
    plt.axis('off')
    plt.show()
```

Фрагмент коду 1

Цей код завантажує зображення з папки за допомогою DataLoader, обробляє їх моделлю YOLOv8 для виявлення об'єктів, виводить кожне зображення з передбаченими боксами та друкує деталі розпізнаних об'єктів, включаючи мітки, ймовірності та координати.

Нижче приведено код інтеграції unet

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import torchvision.transforms as transforms
from torchvision.transforms.functional import to_tensor
import cv2
import os
import glob

class UNetModel(nn.Module):
    def __init__(self, in_channels=3, out_channels=1):
        super(UNetModel, self).__init__()
        self.encoder = nn.Sequential(
            nn.Conv2d(in_channels, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Conv2d(64, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.MaxPool2d(2)
        )
        self.decoder = nn.Sequential(
            nn.Conv2d(64, 64, kernel_size=3, stride=1, padding=1),
            nn.ReLU(),
            nn.Conv2d(64, out_channels, kernel_size=3, stride=1, padding=1),
            nn.Sigmoid()
        )

    def forward(self, x):
        x = self.encoder(x)
        x = nn.functional.interpolate(x, scale_factor=2, mode='bilinear',
align_corners=False)
        x = self.decoder(x)
        return x

class SegmentationDataset(Dataset):
    def __init__(self, image_dir, mask_dir, transform=None):
        self.image_paths = glob.glob(os.path.join(image_dir, '*.jpg'))
        self.mask_paths = glob.glob(os.path.join(mask_dir, '*.png'))
        self.transform = transform

    def __len__(self):
        return len(self.image_paths)

    def __getitem__(self, idx):
        image = cv2.imread(self.image_paths[idx])
        image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
```

```
            mask = cv2.imread(self.mask_paths[idx], cv2.IMREAD_GRAYSCALE)
            if self.transform:
                image = self.transform(image)
                mask = self.transform(mask)
            return image, mask

    transform = transforms.Compose([transforms.ToPILImage(),
transforms.Resize((256, 256)), to_tensor])

    train_dataset = SegmentationDataset(image_dir='dataset/unet/training/images',
mask_dir='dataset/unet/training/masks', transform=transform)
    train_loader = DataLoader(train_dataset, batch_size=4, shuffle=True)
```

Фрагмент коду 2

Цей код виконує навчання нейронної мережі U-Net для задачі сегментації зображень. Спершу він завантажує кастомний набір даних, де кожне зображення та його маска обробляються та передаються через DataLoader.

Нижче приведено код інтеграції YOLO-LPR

```
from ultralytics import YOLO
import cv2
import torch
from torch.utils.data import Dataset, DataLoader
import os
import glob
import matplotlib.pyplot as plt

class LicensePlateDataset(Dataset):
   def __init__(self, folder_path):
       self.image_paths = glob.glob(os.path.join(folder_path, '*.jpg'))

   def __len__(self):
       return len(self.image_paths)

   def __getitem__(self, idx):
       image_path = self.image_paths[idx]
       image = cv2.imread(image_path)
       image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)
       return image_rgb, image_path

dataset = LicensePlateDataset('dataset/yolop/training/images')
dataloader = DataLoader(dataset, batch_size=1, shuffle=False)
```

```
model = YOLO('yolopt.pt')

for images, paths in dataloader:
   results = model(images[0].numpy())
   plt.imshow(results[0].plot())
   plt.axis('off')
   plt.show()

   for detection in results[0].boxes.data:
       x1, y1, x2, y2, score, class_id = detection.tolist()
       label = model.names[int(class_id)]
       print(f"File: {paths[0]}, Object: {label}, Confidence: {score:.2f},
Coordinates: ({x1}, {y1}), ({x2}, {y2})")
```

Фрагмент коду 3

Цей код завантажує зображення з папки за допомогою DataLoader, обробляє їх попередньо навченим моделлю YOLO-LPR для розпізнавання номерних знаків, виводить зображення з передбаченими боксами та друкує деталі кожного розпізнаного об'єкта (номерного знака) із зазначенням координат, ймовірності та ідентифікатора об'єкта.

Тестування і перевірка працездатності нейромереж.

По результатам навчання було досягнуто настпної точності на тестових наборах данних для нейромереж:

Нейромережа для відслідковування транспортних засобів на основі архітектури YoloV8 - 86%

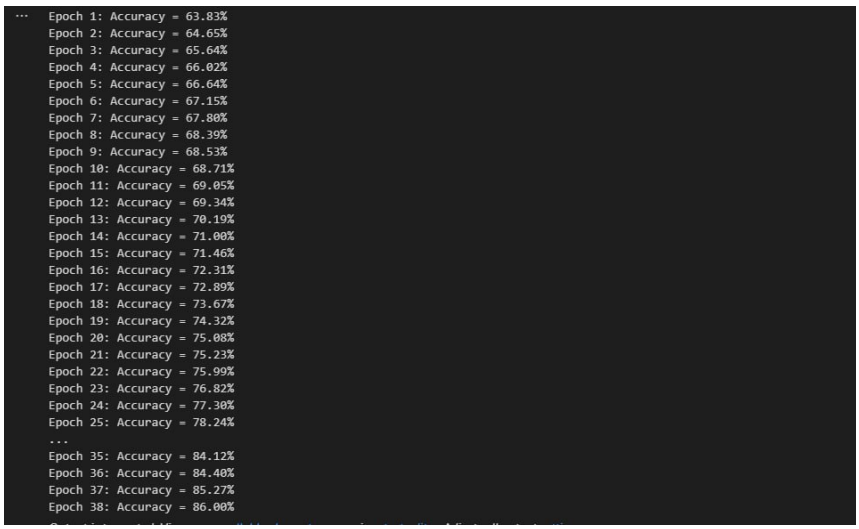

Фото 9 Процесс навчання нейромережі для відслідковування транспортних засобів

Приклади роботи нейромережі

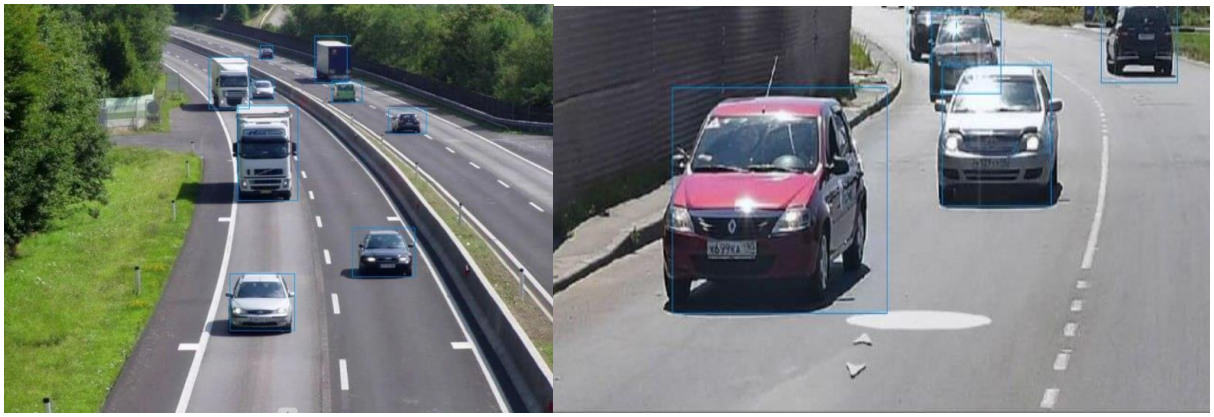

Фото 10 результат роботи нейромережі для відслідковування транспортних засобів

Нейромережа для сегментації транспортних засобів на основі архітектури U-Net - 95.4%

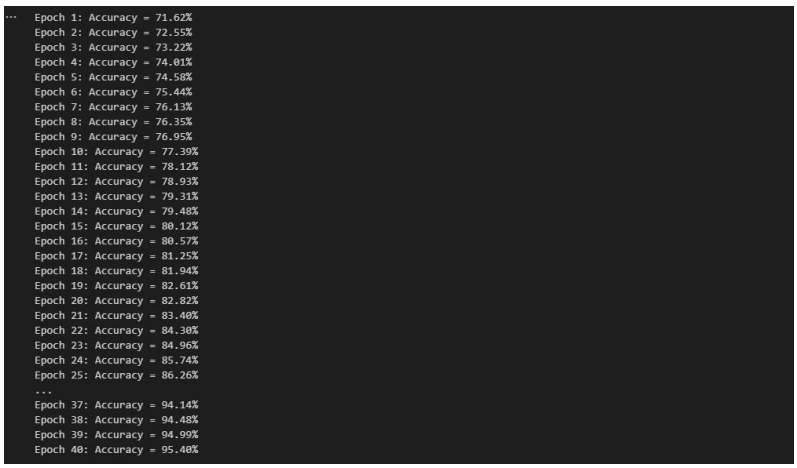

Фото 11 Процесс навчання нейромережі для сегментації транспортних засобів

Приклади роботи нейромережі

Фото 12 Результат роботи нейромережі для сегментації транспортних засобів

Нейромережа для розпізнавання номерних знаків на основі архітектури YOLO-LPR - 79.2%

Можливі удосконалення та майбутні напрями розвитку.

У рамках цієї роботи було розроблено та навчено нейромережі для відстеження транспортних засобів, їх сегментації та розпізнавання номерних знаків. Після калібрування та додаткового навчання на тестових даних нейромережі показали високі результати точності, що дозволяє розглядати її

як ефективну основу для системи автоматизованого моніторингу транспортного руху.

В подальшому систему можна буде вдосконалити за рахунок оптимізації нейронні мережі в режимі реального часу, яка дозволить швидко обробляти велику кількість відеопотоків у високонасиченому міському середовищі.

Іншим важливим кроком стане інтеграція системи з реальними камерами, принаймні в тестовому режимі, для виявлення можливих недоліків і проблем у роботі нейромереж.

Ще одним аспектом, що потребує уваги, є інтеграція системи з базами даних транспортних засобів, що дозволило б автоматично ідентифікувати автомобілі, які перебувають у розшуку або мають заборгованості. Така інтеграція не лише розширить можливості системи, але й підвищить її ефективність як інструмента для правозастосування

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи на тему “Нейромережа для автоматизованого спостереження за транспортним рухом з подальшим виявленням правопорушень” були досягнені усі цілі та розв’язані всі задачі які були поставлені на початкових етапах розробки, а саме:

1. Проаналізовано теоретичні основи по темі нейронних мереж
2. Досліджено сучасні систем контролю та моніторингу транспортного руху. Знайдені їх переваги та недоліки.
3. Розроблена концепція власної системи для автоматизованого спостереження за транпортним рухом.
4. Розроблено та навченно нейромережі для відслідковування, сегментації транспортних засобів та подальшого розпізнавання номерних знаків.

По результатам навчання було досягнуто настпної точності на тестових наборах данних для нейромереж:

Нейромережа для відслідковування транспортних засобів на основі архітектури YoloV8 - 86%

Нейромережа для сегментації транспортних засобів на основі архітектури U-Net - 95.4%

Нейромережа для розпізнавання номерних знаків на основі архітектури YOLO-LPR - 79.2%

За результатами роботи було визначено можливі вдосконалення та майбутні напрями розвитку системи.

Зокрема, оптимізація нейронної мережі для роботи в режимі реального часу дозволить ефективно обробляти великий обсяг відеопотоків у міських умовах.

Наступним етапом стане інтеграція системи з реальними камерами, щоб у тестовому режимі виявити можливі недоліки та проблеми в роботі нейромереж.

Також передбачена інтеграція з базами даних транспортних засобів, що дозволить автоматично ідентифікувати автомобілі, які перебувають у розшуку або мають заборгованості.

Отже, в результаті виконання роботи було розроблено концепцію власної системи для автоматизованого спостереження за транпортним рухом, та розроблено ядро цієї системи, а саме нейромережі для відслідковування, сегментації транспортних засобів та подальшого розпізнавання номерних знаків. Також визначено можливі вдосконалення та майбутні напрями розвитку системи.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Recurrent Neural Networks [Електронний ресурс] – Режим доступу до ресурсу: https://www.ibm.com/cloud/learn/recurrent-neural-networks
2. An Introduction to Recurrent Neural Networks and the Math That Powers Them [Електронний ресурс] – Режим доступу до ресурсу: https://machinelearningmastery.com/an-introduction-to-recurrent-neural-networks-and-the-math-that-powers-them/
3. 9 Types of Neural Networks: Applications, Pros, and Cons [Електронний ресурс] – Режим доступу до ресурсу: https://www.knowledgehut.com/blog/data-science/types-of-neural-networks
4. CNN vs. RNN vs. ANN – Analyzing 3 Types of Neural Networks in Deep Learning [Електронний ресурс] – Режим доступу до ресурсу: https://www.analyticsvidhya.com/blog/2020/02/cnn-vs-rnn-vs-mlp-analyzing-3-types-of-neural-networks-in-deep-learning/
5. Schmidhuber, J. (2015). Deep learning in neural networks: An overview. Neural Networks, 61, 85-117.
6. What is a neural network? [Електронний ресурс] – Режим доступу до ресурсу: https://www.elastic.co/what-is/neural-network
7. Використання нейронних мереж – перспективна сфера науки і суспільства? [Електронний ресурс] – Режим доступу до ресурсу:http://oldconf.neasmo.org.ua/node/139#:~:text=%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D1%96%20%D0%BC%D0%B5%D1%80%D0%B5%D0%B6%D1%96%20%D0%BC%D0%BE%D0%B6%D1%83%D1%82%D1%8C%20%D0%B2%D0%B8%D0%BA%D0%BE%D1%80%D0%B8%D1%81%D1%82%D0%BE%D0%B2%D1%83%D0%B2%D0%B0%D1%82%D0%B8%D1%81%D1%8F%20%D0%BC%D0%B0%D0%B9%D0%B6%D0%B5,%D0%B1%D0%B5%D0%B7%

D0%BF%D0%B5%D1%86%D1%96%20%D0%BE%D1%85%D0%BE%D1%80%D0%BE%D0%BD%D0%BD%D0%B8%D1%85%20%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%2C%20%D0%BE%D0%B1%D1%80%D0%BE%D0%B1%D1%86%D1%96%20%D1%96%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%86%D1%96%D1%9

8. Офіційний сайт NVIDIA Metropolis [Електронний ресурс] – Режим доступу до ресурсу: https://www.nvidia.com/ru-ru/autonomous-machines/intelligent-video-analytics-platform/
9. Офіційний сайт Hikvision AI Traffic Monitoring [Електронний ресурс] – Режим доступу до ресурсу: https://content.hikvision.com/ai-solutions-and-applications/industry/traffic
10. Офіційний сайт OpenALPR [Електронний ресурс] – Режим доступу до ресурсу: https://www.openalpr.com/
11. Офіційний сайт briefcam [Електронний ресурс] – Режим доступу до ресурсу: https://www.briefcam.com/
12. Офіційний сайт sensetime [Електронний ресурс] – Режим доступу до ресурсу: https://www.sensetime.com/en
13. Офіційний сайтHuawei TrafficGo [Електронний ресурс] – Режим доступу до ресурсу: https://e.huawei.com/ua/industries/urban-traffic/urban-traffic-management/intelligent-transportation-system
14. Best Language for Machine Learning [Електронний ресурс] – Режим доступу до ресурсу: https://relevant.software/blog/best-language-for-machine-learning/#:~:text=Why%20is%20Python%20the%20best,sentiment%20analysis%20and%20data%20science.
15. TensorFlow official docks [Електронний ресурс] – Режим доступу до ресурсу: https://www.tensorflow.org/
16. Keras official docks [Електронний ресурс] – Режим доступу до ресурсу: https://keras.io/

17. Pytorch official docks [Електронний ресурс] – Режим доступу до ресурсу: https://pytorch.org/
18. Jupyter official docks [Електронний ресурс] – Режим доступу до ресурсу: https://jupyter.org/
19. Faster R-CNN vs YOLO vs SSD [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/ibm-data-ai/faster-r-cnn-vs-yolo-vs-ssd-object-detection-algorithms-18badb0e02dc
20. EfficientDet docks [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/xuannianz/EfficientDet
21. Semantic segmentation [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/Charmve/Semantic-Segmentation-PyTorch
22. Mask-rcnn[Електронний ресурс] – Режим доступу до ресурсу: https://blog.roboflow.com/mask-rcnn/
23. FCN — Fully Convolutional Network (Semantic Segmentation) [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/review-fcn-semantic-segmentation-eb8c9b50d2d1
24. crnn-tutorial [Електронний ресурс] – Режим доступу до ресурсу: https://github.com/carnotaur/crnn-tutorial/blob/master/CRNN%20Tutorial.ipynb
25. DenseNet : A Complete Guide [Електронний ресурс] – Режим доступу до ресурсу: https://medium.com/@alejandro.itoaramendia/densenet-a-complete-guide-84fedef21dcc
26. Paper with code [Електронний ресурс] – Режим доступу до ресурсу: https://paperswithcode.com/paper/grouped-pointwise-convolutions-reduce?gad_source=1&gclid=Cj0KCQjwj4K5BhDYARIsAD1Ly2r18PGZGK9q6VgSL-E5WblnF91WwS1pVkditeGlIgZ6A9IqyvLLjYEaAsMtEALw_wcB
27. Kaggle [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/

28. Vedai dataset [Електронний ресурс] – Режим доступу до ресурсу: https://paperswithcode.com/dataset/vedai
29. Car segmentation dataset [Електронний ресурс] – Режим доступу до ресурсу: https://www.kaggle.com/datasets/intelecai/car-segmentation
30. ufpr-alpr [Електронний ресурс] – Режим доступу до ресурсу: dataset https://paperswithcode.com/dataset/ufpr-alpr
31. Нейромережа: що це, як працює, найкращі приклади Режим доступу до ресурсу: https://journal.gen.tech/post/sho-take-neiromerezhi
32. History of Artificial Neural Network [Електронний ресурс] – Режим доступу до ресурсу: https://www.javatpoint.com/history-of-artificial-neural-network
33. What is the best programming language for Machine Learning [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7
34. What is the best programming language for Machine Learning [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7
35. Visual Detection of Traffic Incident through Automatic Monitoring of Vehicle Activities https://www.mdpi.com/2032-6653/15/9/382
36. Esteva, A., Kuprel, B., Novoa, R. A., Ko, J., Swetter, S. M., Blau, H. M., & Thrun, S. (2017). Dermatologist-level classification of skin cancer with deep neural networks. Natur
37. Chen, J., Wang, C., & Dong, Y. (2020). Deep learning for financial applications: A survey. Applied Soft Computing, 93, 106384.
38. Karras, T., Aila, T., Laine, S., & Lehtinen, J. (2018). Progressive growing of gans for improved quality, stability, and variation. In Proceedings of the International Conference on Learning Representations (ICLR).

39. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
40. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature