Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

Рівень вищої освіти *другий (магістерський)*

Визначення оптимального шляху транспортних потоків та прогнозування дорожніх заторів за допомогою штучного інтелекту
(тема)

Виконав:
студент 2 курсу, групи *ІМІм-20-2*
Гнилицький Я.В.

Спеціальності *172 Телекомунікації та радіотехніка*
(код і повна назва спеціальності)
Тип програми *Освітньо-наукова*
(освітньо-професійна або освітньо-наукова)
Освітня програма *Інформаційно-мережна інженерія*
(повна назва освітньої програми)
Керівник *проф. Безрук В.М.*
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ *Безрук В.М.*
(підпис) (прізвище, ініціали)

2022 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____ *Гнилицький Я.В.*
(підпис) (прізвище та ініціали)

Керівник _____ *Безрук В.М.*
(підпис) (прізвище та ініціали)

Харківський національний університет радіоелектроніки

Факультет *Інфокомунікацій*
(повна назва)

Кафедра *Інформаційно-мережної інженерії*
(повна назва)

Рівень вищої освіти *другий (магістерський)*

Спеціальність *172 Телекомунікації та радіотехніка*
(код і повна назва)

Тип програми *Освітньо-наукова*
(освітньо-професійна або освітньо-наукова)

Освітня програма *Інформаційно-мережна інженерія*
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри ІМІ_____В.М. Безрук
(підпис)

“____” _____2022 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Студентові *Гнилицькому Яну Владиславовичу*
(прізвище, ім’я, по батькові)

1. Тема роботи *Визначення оптимального шляху транспортних потоків та прогнозування дорожніх заторів за допомогою штучного інтелекту*
затверджені наказом університету від 14 березня 2022 року № 379 *Ст*
2. Термін подання студентом роботи до екзаменаційної комісії *13 травня 2022 р.*
3. Вихідні дані до роботи
1 .Відкрита програмна бібліотека для машинного навчання Tensor Flow
2. Модель визначення об'єктів Efficient Det та камери
3. Графічний процесор MSI GeForce GTX 1050
4. Набір даних для тренування моделі
5. Мова програмування Python

4. Перелік питань, що потрібно опрацювати в роботі
Вступ
1. Штучний інтелект
2. Огляд використаних технологій
3. Тренування моделі розпізнавання об'єктів
4. Моделювання алгоритму роботи світлофору
5. Пошук найкоротшого шляху з алгоритмом Дейкстри
Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) *Слайди у форматі Power Point (назва, мета і задачі роботи, типи штучного інтелекту, підрозділи штучного інтелекту, використання Python для штучного інтелекту, фреймворки для машинного навчання, тренування моделі розпізнавання об'єктів, удосконалення алгоритму роботи світлофорів, пошук найкоротшого шляху, висновки)*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів атестаційної роботи	Строк виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ*	*14.03.22*	*виконано*
2	*Підбір літератури за темою роботи*	*15.03-18.03.22*	*виконано*
3	*Виконання розділу 1*	*19.03-29.03.22*	*виконано*
4	*Виконання розділу 2*	*30.03-09.04.22*	*виконано*
5	*Виконання розділу 3*	*10.04-20.04.22*	*виконано*
6	*Виконання розділу 4*	*21.04-01.05.22*	*виконано*
7	*Виконання розділу 5*	*02.05-08.05.22*	*виконано*
8	*Оформлення пояснювальної записки*	*09.05-11.05.22*	*виконано*
9	*Оформлення презентаційного матеріалу,*	*12.05-13.05.22*	*виконано*
	підготовка до захисту у ЕК		

Дата видачі завдання *14.03.2022 р.*

Студент _____ *Гнилицький Я.В.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Безрук В.М.*
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка: 62 с., 24 рис.,1 табл.,17 джерел, 2 додатки.

Об'єкт дослідження – система визначення оптимального шляху транспортних потоків.

Мета роботи – розробити систему, що дозволяє відстежувати транспортний рух, та на основі цієї інформації контролювати роботу світлофорів і перенаправляти транспортний потік на більш оптимальні маршрути.

В роботі були розглянуті: питання, щодо штучного інтелекту та сфери його застосування, процес тренування моделі для розпізнавання об'єктів, використовуючи претреновану модель EfficientDetз використанням публічного набору даних автомобілів, важливість використання розробленої системи у роботі світлофорів та застосування алгоритму Дейкстри для перенаправлення транспортних потоків.

ЗАТОРИ НА ДОРОГАХ, ШТУЧНИЙ ІНТЕЛЕКТ, PYTHON, РОЗПІЗНАВАННЯ ОБ'ЄКТІВ, АЛГОРИТМ ДЕЙКСТРИ

THE ABSTRACT

Explanatory note: 62 p., 24 fig., 1 tabl., 17 sources, 2 app.

The object of study is the system for determining the optimal path of traffic flows.

The purpose of the work is to develop a system that allows you to track traffic, and based on this information to control the operation of traffic lights and redirect traffic to more optimal routes.

The work examined: issues related to artificial intelligence and its scope, the process of training a model for object recognition using a pre-trained EfficientDet model using a public car dataset, the importance of using the developed system in traffic lights and using the Dijkstra algorithm to redirect traffic flows.

TRAFFIC CONGESTION, ARTIFICIAL INTELLIGENCE, PYTHON, OBJECT RECOGNITION, DIJKSTRA'S ALGORITHM

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 9
ВСТУП 10
1 ШТУЧНИЙ ІНТЕЛЕКТ 11
1.1 Історія штучного інтелекту 12
1.2 Типи штучного інтелекту 13
1.2.1 Реактивна машина 13
1.2.2 Штучний інтелект з обмеженою пам'яттю 13
1.2.3 Теорія розуму штучного інтелекту 14
1.2.4 Самоусвідомлювальні машини штучного інтелекту 14
1.3 Підрозділи штучного інтелекту 14
1.3.1 Машинне навчання 15
1.3.2 Нейронні мережі 17
1.3.3 Глибоке навчання 18
1.4 Штучний інтелект у різних галузях 19
1.4.1 Охорона здоров'я 19
1.4.2 Фінансові послуги та банківська справа 20
1.4.3 Роздрібна торгівля та електронна комерція 21
2 ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ 23
2.1 Використання Python на проектах зі штучним інтелектом 23
2.1.1 Простий та послідовний 23
2.1.2 Великий вибір бібліотек та фреймворків 24
2.1.3 Незалежність від платформи 25
2.1.4 Відмінна спільнота та популярність 25
2.2 Бібліотека TensorFlow 27
2.2.1 Як працює TensorFlow 28
2.2.2 Переваги TensorFlow 29
2.2.3 TensorFlow у порівнянні з конкуруючими бібліотеками 29
3 ТРЕНУВАННЯ МОДЕЛІ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ 31
3.1 EfficientDet для масштабованого та ефективного виявлення об'єктів 31
3.2 Встановлення та налаштування 32
3.2.1 Створення нового віртуального середовища 32
3.2.2 Скачування та вилучення TensorFlow Model Garden 33

3.2.3 Встановлення COCO-API 33
3.2.4 Встановлення Object Detection API 34
2.3 Підготовка данних 34
3.4 Перетворення даних 35
3.5 Створення картки міток 35
3.6 Вибір моделі 36
3.7 Процес налаштування 36
3.8 Навчання моделі 37
4 МОДЕЛЮВАННЯ АЛГОРИТМУ РОБОТИ СВІТЛОФОРУ 38
4.1 Алгоритм управління пристроями регулювання дорожнього руху 40
4.1.1 Занадто багато машин 41
4.1.2 Максимально можлива кількість машин 41
4.1.3 Не дуже багато машин 42
5 ПОШУК НАЙКОРОТШОГО ШЛЯХУ З АЛГОРИТМОМ ДЕЙКСТРИ 44
5.1 Ефективність 45
5.2 Переваги алгоритму Дейкстри 45
5.3 Недоліки алгоритму Дейкстри 45
5.4 Застосування алгоритму Дейкстри 46
5.5 Алгоритм Дейкстри в Python 47
5.5.1 Клас Графу 47
5.5.2 Алгоритм Дейкстри 48
5.5.3 Допоміжна функція 51
5.6 Перевірка роботи алгоритму 51
ВИСНОВКИ 53
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ 54
ДОДАТОК А СЛАЙДИ ПРЕЗЕНТАЦІЇ 56
ДОДАТОК Б ПУБЛІКАЦІЯ ЗА ТЕМАТИКОЮ РОБОТИ 60

ПЕРЕЛІК СКОРОЧЕНЬ

AI – (Artificial Intelligence) штучний інтелект;

NN – (Neural Network) нейронна мережа;

DL – (Deep Learning) глибоке навчання;

ML – (Machine Learning) машинне навчання;

API – (Application Programming Interface) інтерфейс прикладного програмування;

TPU – (Tensor Processing Unit) тензорний блок обробки;

COCO – (Common Objects in Context) загальні об'єкти у контексті.

ВСТУП

Внаслідок технічного прогресу у минулому столітті, багато країн світу пережили масову міграцію із сільських районів у міста. В результаті населення цих міст зростало, що призводило до численних цивільних проблем, у тому числі до заторів на дорогах. Затори є розповсюдженою проблемою у будь-якій дорожній мережі. Якщо на дорозі кількість автомобілів перевищить максимально допустиму, утворюватимуться затори різної щільності. Відстеження та моніторинг трафіку в реальному часі, а також довгострокова оцінка є бажаними для широкої громадськості. Зростання населення у великих містах призвело до постійно зростаючого попиту на громадський транспорт, протягом багатьох років це було одним із головних факторів, що спричиняють затори на дорогах. Пасажири приміських потягів проводять багато часу в поїздках і мають проблеми з плануванням подорожі. Транспортні затори слід враховувати при містобудуванні. Країни, що розвиваються, хоча їхні великі міста мають добре розвинену дорожню інфраструктуру, страждають від заторів переважно через високу щільність населення. Крім того, на збільшення заторів сприяли нижчі ціни на автомобілі та зростаючий попит з боку багатьох споживачів [1]. Фактично через відсутність належної інфраструктури затори більш поширені в країнах третього світу. Аналіз транспортних заторів і прогнозування відіграють важливу роль у розвитку інтелектуальних транспортних систем.

Метою роботи являється аналіз та проектування системи знаходження оптимального маршруту для транспортних потоків та усунення дорожніх заторів, використовуючи такі затребувані новітні технології, як штучний інтелект. Саме тому тема кваліфікаційної роботи є актуальною.

1 ШТУЧНИЙ ІНТЕЛЕКТ

Про штучний інтелект (ШІ) говорять ще із 1950-х років. Останнім часом йому приділяється більше уваги в результаті численних дебатів про великі дані, аналіз даних та суперкомп'ютери. Це робить його актуальним і широко використовуваним у різних цілях. Завдяки нещодавнім досягненням у галузі технологій штучного інтелекту, таких як глибоке навчання, розпізнавання зображень, машинне навчання та обробка мови, стає ясно, що штучний інтелект вплине на повсякденне життя так само, як і цифровізація. Так само штучний інтелект робить більший вплив на ділову активність, ніж соціальні мережі [1].

Дослідники все ще працюють над штучним інтелектом, та їхні зусилля можуть допомогти йому удосконалюватися емоційно, розумово та соціально. В результаті очікується, що найближчими роками ШІ стане невід'ємною частиною всіх дисциплін [1]. Вчені вивчали, як штучний інтелект впроваджувався у різні галузі, і цей прогрес було задокументовано у літературі.

З іншого боку, визначення штучного інтелекту трохи розпливчасте. Однією з причин є те, що існує безліч визначень, запропонованих експертами. Кожен експерт формує його по-своєму, виходячи зі своєї думки та досвіду. Так само будь-яка спроба дати визначення ШІ ще більше ускладнює завдання. Наприклад, деякі вчені намагалися класифікувати його як завдання, тоді як інші намагалися включити до нього різні аспекти. Зосередивши увагу на «розвідці», деякі вчені зробили свій внесок у суспільне замішання. В результаті використання інтелекту для опису ШІ викликає суперечки. Вчені стверджують, що цю концепцію необхідно визначити з погляду того, який тип інтелекту, глибина інтелекту та широта інтелекту потрібні для класифікації штучного інтелекту як такого. Всі ці умови призводять до фундаментального нерозуміння визначення штучного інтелекту [2]. Дилема ще більше ускладнюється тим, що штучний інтелект постійно розвивається в міру появи нових технологій. Через технічний прогрес те, що раніше вважалося штучним інтелектом, більше не може кваліфікуватися як таке.

1.1 Історія штучного інтелекту

Штучний інтелект – не новий термін, він має довгу і славну історію, що сягає корінням у глибоку давнину. Люди віддавна хотіли створити машину, здатну на людську діяльність [2].

– Карел Чапек, чеський драматург, вигадав термін «штучний інтелект» у своїй науково-фантастичній п'єсі у 1921 році. У цій п'єсі був персонаж на ім'я Робот. У нього була здатність думати та діяти. Роботів не цікавили фільми, п'єси та література до 1950 року.

– "Таким чином, машина може думати" – вперше написав відомий письменник Едмунд Берклі у 1949 році. Він написав це у своїй книзі «Гігантські мізки» або машини, які думають.

– Алан Т'юрінг опублікував книгу «Обчислювальні машини та інтелект» у 1950 році, яка стала відома як тест Тюрінга. Цей тест використовується, як метод визначення інтелекту машини.

– SNARC, перший комп'ютер з нейронною мережею, був винайдений Марвін Мінськ і Дейном Едмондом у 1950 році. У тому ж році Клод Шеннон опублікував статтю про програмування комп'ютера для гри в шахи.

– У ході літнього дослідницького проекту в Дартмуті Джон Маккарті та його команда запровадили термін «штучний інтелект». Була організована конференція, на якій були висвітлені деякі особливості та цілі штучного інтелекту, що було його започаткуванням [1].

– Загальне Вирішення Проблем – це програма, створена Аланом Ньюеллом, Гербертом Саймоном і Дж. К. Шоу в 1959 для імітації здібностей людини вирішувати проблеми. Працюючи в IBM того ж року, Артур Семюел ввів термін «машинне навчання». Проект штучного інтелекту Массачусетського технологічного інституту було запущено у тому році Джоном Маккарті та Марвіном Мінськ.

– Консультативний комітет з автоматичної обробки мови (ALPAC) відзначив відсутність прогресу уряду США у дослідженнях машинного перекладу у 1966 році. Усі звіти, що фінансувалися державою, було скасовано в результаті досліджень ALPAC.

– Френ Розенблатт у 1967 році побудувала перший комп'ютер, персептрон Марка, який використовував парадигму нейронної мережі та міг

навчатися методом спроб та помилок. Наступного року Марвін Мінськ та Сеймур Пейперт опублікували «Сприйняття», що відкрило нову еру досліджень нейронних мереж.

– У 1997 році Гаррі Каспаров, знаменитий чемпіон з шахів, зазнає поразки від Deep Blue, комп'ютеру IBM, що грає в шахи.

– Google розробила технологію розпізнавання мови та додала нові можливості у 2008 році [2].

– У 2012 році Google створила нейронну мережу, яка використовувала 10 мільйонів відео на YouTube, як навчальний набір даних, використовуючи алгоритм глибокого навчання. Не будучи повідомленим у тому, що це кіт, нейрона мережа ідентифікувала його. Це був величезний крок уперед у галузі нейронних мереж та досліджень глибокого навчання [1].

– У 2015 році комп'ютер Minwa від Baidu зміг розпізнавати та класифікувати фотографії краще, ніж люди. Для цього було використано спеціальний алгоритм глибокої нейронної мережі.

1.2 Типи штучного інтелекту

Нині існують різні типи штучного інтелекту.

1.2.1 Реактивна машина

Це найстаріший тип ШІ, оскільки в неї не вистачає можливостей, заснованих на пам'яті. Реактивна машина – це машина, яка виконує прості завдання та не потребує навчання. Реактивні машини зазвичай обмежуються однією сферою застосування [3]. Наприклад, статичні моделі навчання – це реактивні машини. Deep Blue, система штучного інтелекту IBM, є чудовим прикладом реактивного ШІ.

1.2.2 Штучний інтелект з обмеженою пам'яттю

ШІ з обмеженою пам'яттю збирає знання із попередніх даних та зберігає їхв пам'яті. AIвиносить судження з урахуванням зібраної інформації. Машинне навчання ускладнюється, оскільки цей тип ШІ має обмежену пам'ять. Програми

розпізнавання зображень, наприклад, є хорошим прикладом обмежень пам'яті [3]. Вони запам'ятовують на згадку тисячі фотографій і роблять чудові міркування на основі цих посилань.

1.2.3 Теорія розуму штучного інтелекту

В майбутньому буде доступна машина штучного інтелекту теорії. Ці типи машин розумітимуть поведінку та емоції людей, а також зможуть вступати у соціальні відносини та поважати почуття людей [2]. Сирі та Алекса, наприклад, сприймають прохання людей і виконують їх, але не здатні інтерпретувати емоції.

1.2.4 Самоусвідомлювальні машини штучного інтелекту

Самоусвідомлюючі машини – це майбутнє штучного інтелекту та машинного навчання. Вони стануть розумнішими та самодостатніми. На цей час їх немає в реальному світі та можна знайти лише у вигадці [3].

1.3 Підрозділи штучного інтелекту

Штучний інтелект розділений на підрозділи, кожен з яких має унікальний набір функцій. Нижче наведені найважливіші області ШІ, вони постійно розвиваються. На рисунку(1.1) зображено безліч підрозділів ШІ.

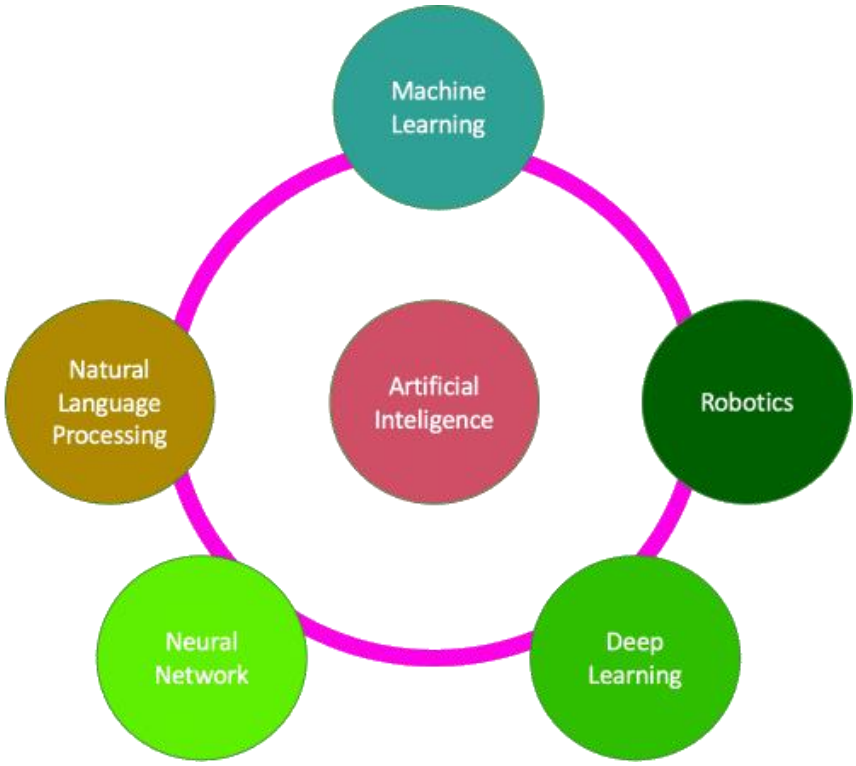


Рисунок 1.1 – Приклади підрозділів штучного інтелекту

1.3.1 Машинне навчання

Незважаючи на те, що машинне навчання та машинне навчання з ШІ взаємозамінні, машинне навчання є підмножиною штучного інтелекту. Це керований даними алгоритм, який навчається та вдосконалюється. Машинне навчання – це тип штучного інтелекту, який може навчатися на попередніх даних та оновлювати їх з часом [2].

Програми, що базуються на машинному навчанні, працюють так само, як і люди. Ці програми навчаються на даних, і коли нові дані стають доступними, вони автоматично оновлюються та розвиваються.

Існують різні види моделей машинного навчання, що класифікуються на основі характеристик даних. В даний час поширені три типи моделей машинного навчання.

а) Контрольоване машинне навчання

Контрольоване машинне навчання передбачає нагляд чи керівництво завданням, щоб переконатися, що воно виконано належним чином. Наприклад, як діти у школі, що навчаються під керівництвом вчителів. У моделі

контрольованого навчання процесс навчання виконується на даних міток, що виступають як вхідні дані і вказує, як мають виглядати вихідні дані. Інакше кажучи, контрольована машина вивчає дані з попередніх прикладів і використовує їх, щоб робити прогнози на майбутнє. І вхідні, і вихідні дані допомагають зробити чіткий прогноз. На рисунку (1.2) зображено весь контрольований процес навчання. Сегментація зображень, медична діагностика та виявлення шахрайства — це приклади алгоритмів навчання з учителем.

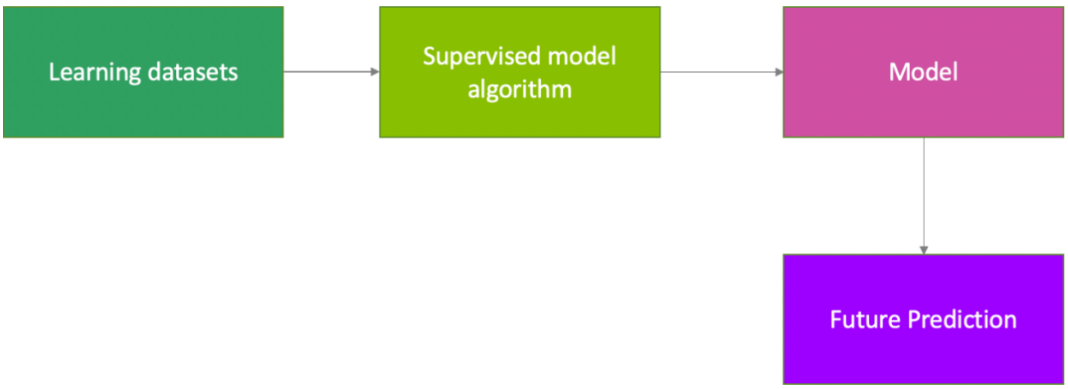


Рисунок 1.2 – Процес контрольованого машинного навчання

b) Неконтрольоване машинне навчання

Використання неконтрольованого машинного навчання має на увазі те, що немає жодних порад чи вказівок у роботі штучного інтелекту. Машина зчитує вхідні дані і починає знаходити правильний шаблон інформації, яка не класифікується і не маркується і легко доступна з різних джерел. Алгоритм у цій моделі шукає порівняні закономірності, рівняння та кореляції. Неконтрольоване навчання виявляє приховані закономірності та подібності, які можуть виявитися корисними у майбутньому [4]. Методи з неконтрольованим навчанням використовуються у різних додатках, включаючи мережевий аналіз та пошукові системи.

c) Машинне навчання з підкріпленням

Машинне навчання з підкріпленням відрізняється від двох інших тим, що воно навчається методом проб та помилок, а також отримує зворотний зв'язок про свою діяльність. Хоча контрольована і посилена моделі містять зіставлення вхідних і вихідних даних, контрольована модель забезпечує належний набір

дій. З іншого боку, вона визначає правильний курс дій на основі своїх помилок та генерує вихідні дані. У вигляді прикладу моделі машинного навчання з підкріпленням можна навести безпілотний автомобіль [3]. У безпілотному автомобілі необхідно враховувати багато факторів, включаючи зони, обмеження швидкості та інші високошвидкісні транспортні засоби. Інші галузі застосування машинного навчання з підкріпленням включають динамічне ціноутворення та промислову автоматизацію.

1.3.2 Нейронні мережі

Поняття нейронних мереж (NN) ґрунтується на нейронній мережі людського мозку, та її метою є цифрова реконструкція мозку. Штучні нейрони необхідні для створення штучного мозку. Нейрони мозку спілкуються між собою через синаптичні сполуки, а людський мозок містить сто мільярдів нейронів. Вони утворюють велику нейронну мережу, посилаючи імпульси від одного нейрона до іншого. Штучна NN працює так само як нейронні мережі мозку. Вони отримують вхідні сигнали з інших нейронів і передають цю інформацію іншому нейрону як виведення.

Штучна нейронна мережа, що містить вхідний шар, прихований шар і вихідний шар, показана на рисунку 1.3. Кожен зв'язок важливий, і деякі зв'язки більш ефективні, ніж інші. У реальному сценарії робота навчається з різними вхідними вагами, а потім вхідні дані змінюються для отримання необхідного результату [4]. При навчанні з вхідними даними та вагами ця нейронна мережа може справлятися з реальними проблемами, у яких можна використовувати ШІ. Нейронна мережа на цьому зображенні є простим прикладом. Чим складніше завдання, тим більше вхідних даних та процес буде складнішим.

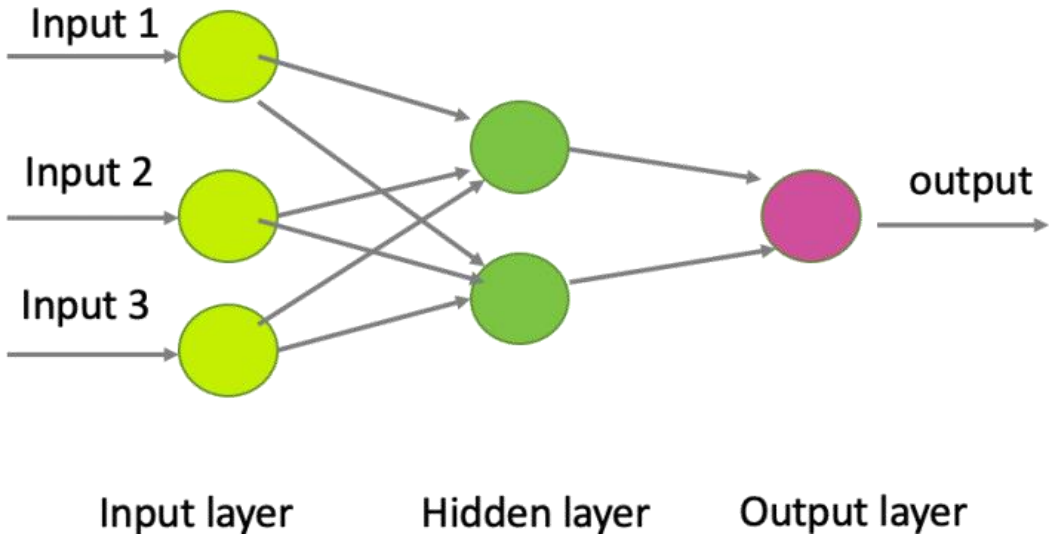


Рисунок 1.3 – Проста нейронна мережа

1.3.3 Глибоке навчання

Штучний інтелект розробляється з використанням глибокого навчання, яке є підмножиною машинного навчання та алгоритмів машинного навчання. Глибоке навчання являє собою цікаву техніку машинного навчання [3].

Глибоке навчання (DL) - це процес аналізу вхідних даних та їх зіставлення з використанням різних підходів доти, доки не буде знайдено правильний результат. Це процес самонавчання машини. Алгоритми DL намагаються виконувати логічні завдання так само, як це роблять люди. Він завжди прагне аналізувати дані з певною логічною структурою. Це досягається за рахунок використання нейронних мереж. Нейронні мережі є набором алгоритмів, і глибоке навчання ділить їх на шари, щоб сформувати нейронну мережу.

Нейронна мережа з глибоким навчанням аналізує дані та знаходить особливості та шаблони, використовуючи численні шари та складний алгоритм. Вхідний шар, прихований шар та вихідний шар – це три шари, що використовуються. Вхідний і вихідний шари видно, а приховані шари — ні, й у залежності від вхідних даних вони можуть бути досить великими. Дані приймаються вхідними шарами, які потім передають їх до прихованого шару. У нейронній мережі вихідним шаром є шар категоризованих прогнозуючих даних. Вхідні дані прихованого шару використовуються для всіх математичних обчислень, а дані попереднього шару обробляються більш складним

алгоритмом [4]. Декілька прихованих шарів у нейронних мережах глибокого навчання виконують різні обчислення на основі вхідних даних, тому це називається глибоким навчанням. DL широко використовується у сфері фінансових послуг, обслуговування клієнтів та охорони здоров'я. У фінансових послугах він використовується для прогнозного аналізу та чат-ботів для обслуговування клієнтів.

1.4 Штучний інтелект у різних галузях

У цьому розділі буде розглянуто, як ШІ використовується у різних галузях і яку користь AIприносить у цих сферах.

1.4.1 Охорона здоров'я

Штучний інтелект дуже впливає на галузь охорони здоров'я. За даними Всесвітньої організації охорони здоров'я (ВООЗ), до 2030 року глобальна нестача фахівців у галузі охорони здоров'я становитиме 9,9 мільйонів людей. Крім того, ясно, що того ж року буде створено 40 мільйонів нових робочих місць у галузі охорони здоров'я [5]. AIможе допомогти у вирішенні цієї проблеми різними способами. Нижче наведено декілька галузей охорони здоров'я, в яких використовується ШІ.

a) Відкриття ліків

Штучний інтелект в даний час використовується низкою фармацевтичних компаній для допомоги у відкритті нових ліків. Коли справа доходить до відкриття нових ліків, процедура зазвичай досить тривала [4]. ШІ може забезпечити короткі терміни та добре збалансовані методи пошуку та виведення ліків на ринок.

b) Клінічні випробування

Клінічні дослідження для людей із тяжкими захворюваннями можуть бути важкими. У автономному режимі більшість цих клінічних випробувань

неадекватно управляються. Немає складних методів відстеження прогресу в режимі реального часу, збору даних або результатів випробувань ліків. Процедура тепер більш керована, ніж будь-коли раніше, завдяки технологіям штучного інтелекту та блокчейну.

ШІ допомагає у розробці віртуальних помічників “медсестер” для автоматизованої візуалізації, роботизованих хірургічних операцій, зменшення помилок дозування та досліджень нових ліків, а також в інших медичних послугах [4]. Існують різні рішення на основі ШІ, які використовуються, наприклад, у критичних схемах лікування. Його можна використовувати для вивчення медичних карт пацієнта та призначення лікування. Він діє аналогічно до людського лікаря.

1.4.2 Фінансові послуги та банківська справа

Є безліч прикладів того, як штучний інтелект широко використовується в банківській та фінансовій сфері, нижче приведено декілька з них.

a) Досвід роботи з клієнтами

Довгі черги є найбільшою проблемою банківського сектора. Задоволення потреб клієнтів має життєво важливе значення у банківській та фінансовій галузях загалом [5]. Банківські чат-боти та віртуальні помічники дуже поширені в наші дні, і вони можуть допомогти клієнтам із повсякденними проблемами, такими як перевірка балансу, грошові перекази та оплата рахунків.

b) Персоналізований банкінг

Людські агенти замінюються роботами у різних випадках. Ці роботи можуть оформити кредит, перевірити кредитну історію клієнта та порекомендувати відповідні інвестиції на основі попередньої роботи та кредитної історії клієнта. Роботу різних кампаній в соціальних мережах та електронній пошті, можна автоматизувати за допомогою роботів.

c) Страховий сектор

У страховій галузі чат-боти використовуються для покращення якості обслуговування клієнтів та розробки більш ефективних планів страхування шляхом аналізу даних про споживачів [6]. Вони можуть прискорити обробку претензій та надати істотну допомогу клієнтам та компаніям.

d) Економія витрат

ШІ може допомогти скоротити витрати та збільшити прибуток у різних сферах діяльності. За даними Juniper Research, до 2022 року чат-боти заощадять банківській галузі 8 мільярдів доларів у банківському секторі. Чат-боти стають розумнішими з кожним днем. Незабаром вони зможуть виконувати складніші справи та економити набагато більше грошей.

e) Плавний потік інформації

Чат-боти і віртуальні агенти можуть надавати постійний потік інформації з різних тем. Вони можуть відповідати на запитання клієнтів без допомоги співробітника. Чат-боти та віртуальні агенти можуть за необхідності переводити дзвінки на реальних агентів [6].

Щоб виявляти фішингові електронні листи, штучний інтелект активно використовується для виявлення шахрайства та фільтрації спаму.

1.4.3 Роздрібна торгівля та електронна комерція

Штучний інтелект найчастіше використовується у роздрібній торгівлі та електронній комерції, і попит на нього зростає з кожним днем. На рисунку(1.4) показано 600-відсоткове збільшення використання ШІ всього за два роки.

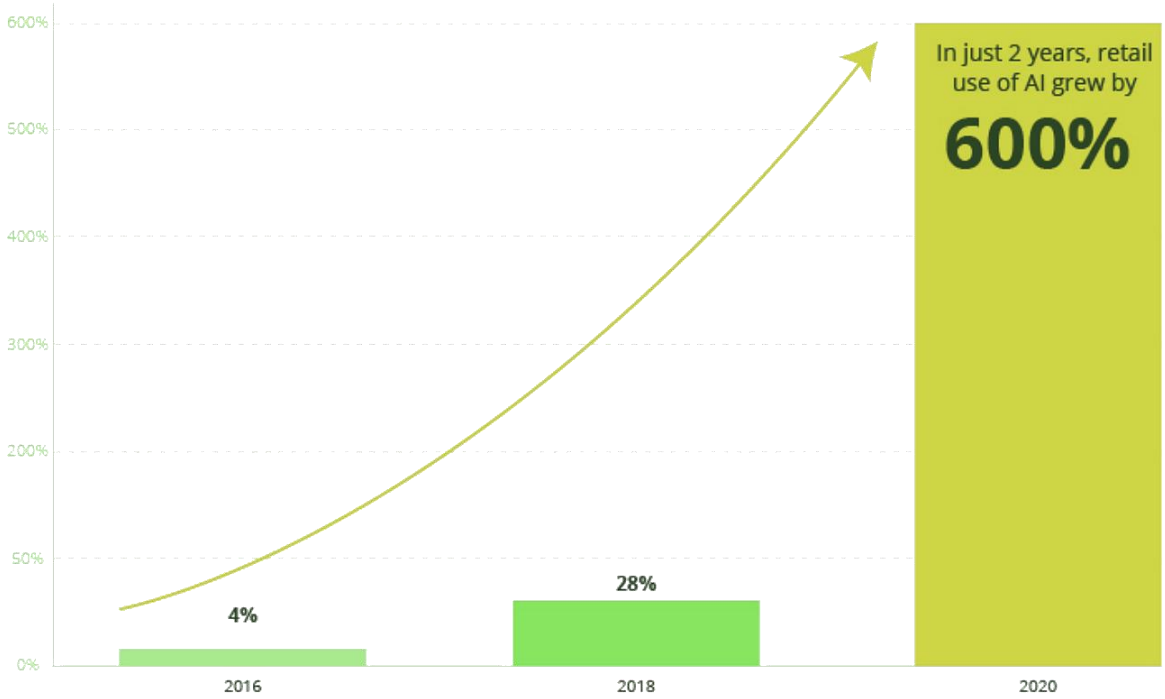


Рисунок 1.4 – Зростання штучного інтелекту в рітейлі

Штучний інтелект може допомогти у виявленні та прогнозуванні продуктів що будуть мати попит, сегментацію клієнтів та покращене обслуговування клієнтів, а також аналізувати поведінку клієнтів та проводити прогнозний аналіз у сфері роздрібної торгівлі та електронної комерції.

2 ОГЛЯД ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

2.1 Використання Python на проектах зі штучним інтелектом

Проекти в галузі штучного інтелекту відрізняються від розробки стандартного програмного забезпечення. Відмінності полягають у технологічному стеку, талантах, необхідних для проекту на основі ШІ, та вимогах до поглибленого вивчення. Щоб реалізувати проекти на ШІ, потрібно використовувати мову програмування, яка є стабільною, гнучкою та має доступні інструменти. Python має всі ці функції, тому зараз існує так багато проектів штучного інтелекту на основі Python [7].

Python дозволяє розробникам працювати продуктивно та впевнено протягом усього процесу створення програмного забезпечення, від проектування до розгортання та обслуговування. Python – ідеальний вибір для машинного навчання та програм на основі AIзавдяки його простоті та узгодженості, доступності відмінних бібліотек та фреймворків для AIта машинного навчання (ML), адаптивності, свободі платформи та широкій спільноті. Це сприяє загальній привабливості мови.

2.1.1 Простий та послідовний

Python дає лаконічний код, що легко читається. За машинним навчанням та штучним інтелектом стоять складні алгоритми та різноманітні процедури, але простота використання Python дозволяє розробникам створювати надійні рішення. Замість того, щоб концентруватися на технічних тонкощах мови, розробники можуть присвятити всю увагу вирішенню проблем ML.

Крім того, простота вивчення Python робить його бажаним для багатьох розробників. Зрозумілість програмування Python для людини полегшує розробку моделей машинного навчання.

На думку ряду програмістів, Python інтуїтивніше зрозумілий, ніж інші мови програмування. Інші виділяють безліч фреймворків, бібліотек та розширень, що полегшують розробку різних функцій. Широко визнано, що Python підходить для спільної реалізації за участю багатьох розробників.

Python, будучи мовою програмування загального призначення, здатний виконувати безліч складних завдань машинного навчання та дозволяє швидко створювати прототипи для тестування продуктів машинного навчання.

2.1.2 Великий вибір бібліотек та фреймворків

Реалізація алгоритмів ШІ та ML може бути складною та трудомісткою. Дуже важливо мати добре структуроване та добре протестоване середовище, щоб розробники могли надавати найефективніші рішення для кодування.

Численні фреймворки та бібліотеки Python використовуються програмістами для мінімізації часу розробки [7]. Бібліотека програмного забезпечення – це набір попередньо написаного коду, який використовується програмістами для вирішення спільних завдань програмування. Python пропонує велику бібліотеку для штучного інтелекту та машинного навчання завдяки своєму потужному технологічному стеку. Ось кілька прикладів:

– TensorFlow, Keras, та Scikit-learn для машинного навчання;

– NumPy для високопродуктивних аналізу даних та наукових обчислень;

– SciPy для просунутих обчислень;

– Pandas для загального аналізу даних;

– Seaborn для візуалізації даних.

Scikit-learn включає ряд методів класифікації, регресії та кластеризації, у тому числі методи опорних векторів, випадкові ліси, підвищення градієнта, k-середніх та DBSCAN, та сумісний з числовими та науковими бібліотеками Python NumPy та SciPy.

За допомогою цих опцій можна прискорити розробку продукту. Замість відтворення колеса, команда розробників може використовувати існуючу бібліотеку для забезпечення бажаних функцій.

Для чого використання Python краще усього підходить наведено у таблиці (2.1).

Таблиця 2.1 – Популярні варіанти використання AIта найбільш підходящі для них технології

Аналіз і візуалізація даних	Seaborn, SciPy, NumPy, Pandas
Машинне навчання	Scikit-learn, TensorFlow, Keras

Комп'ютерний зір	OpenCV
Обробка природної мови	NLTK, spaCy

2.1.3 Незалежність від платформи

Незалежність від платформи відноситься до мови програмування або середовища, яке дозволяє розробникам створювати речі в одній системі, а потім використовувати їх на іншій машині з мінімальними змінами, або зовсім без них. Привабливість Python частково пояснюється тим, що вона не залежить від платформи. Python підтримується кількома операційними системами, включаючи Linux, Windows та MacOS [7]. Код Python може використовуватися для створення автономних програм, що виконуються для більшості популярних операційних систем, що дозволяє легко поширювати програмне забезпечення Python і використовувати його на цих платформах без інтерпретатора Python.

Крім того, розробники часто використовують Google або Amazon, щоб задовольнити свої обчислювальні потреби. Однак багато компаній та фахівці за даними навчають свої моделі машинного навчання на власних робочих станціях, оснащених потужними графічними процесорами (GPU). Оскільки Python не залежить від платформи, таке навчання набагато дешевше і простіше.

2.1.4 Відмінна спільнота та популярність

Python увійшов до п'ятірки найпопулярніших мов програмування в опитуванні розробників 2020 року, проведеному StackOverflow, що передбачає, що дуже легко знайти та найняти компанію з розробки з необхідним набором навичок для створення проекту на основі AI.

Згідно з опитуванням Python Developers Survey 2020, Python широко використовується для веб-розробки. Веб-розробка становить майже 26 відсотків варіантів використання, показаних на рисунку (2.1). Однак, коли наука про дані та машинне навчання об'єднуються, на їхню частку припадає приголомшливі 27 відсотків.

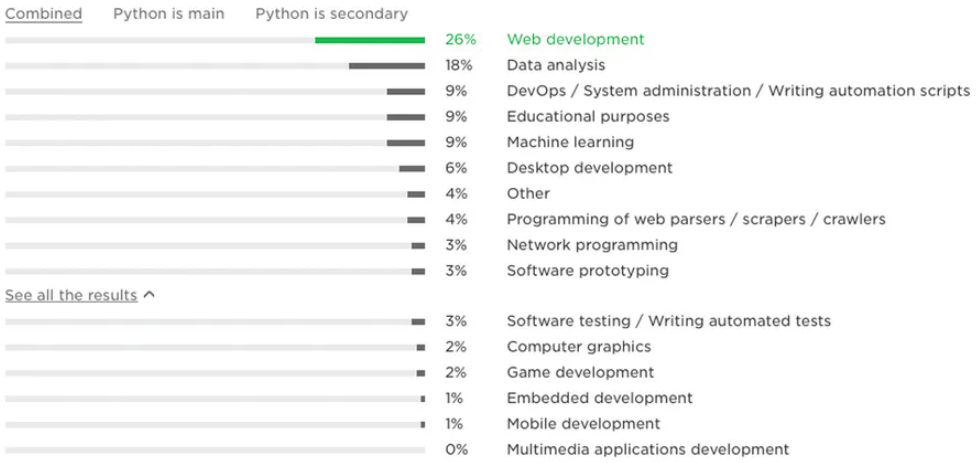


Рисунок 2.1 – Результати опитування на StackOverflow щодо того, для чого зазвичай використовується Python

В онлайн-репозиторіях є близько 140 000 спеціально створених програмних пакетів Python. У програмах на основі Python можуть бути встановлені наукові пакети Python, такі як Numpy, Scipy та Matplotlib. Ці пакети підтримують машинне навчання і дозволяють розробникам виявляти закономірності у великих наборах даних. Python настільки надійний, що Google використовує його для перегляду веб-сторінок, Pixar – для створення фільмів, а Spotify – для пошуку мелодій.

Відомо, що спільнота Python AI розширилася по всьому світу. Існують форуми Python та активний обмін знаннями про рішення для машинного навчання [7]. Для будь-якого завдання, яке може виникнути, досить високою є ймовірність того, що хтось вже мав справу з тією ж проблемою. Розробники можуть дати пораду та направлення. Якщо звернутися до спільноти Python, велика ймовірність, що оптимальна відповідь щодо унікальних вимог, буде обов’язково знайдена.

Роблячи підсумок, Екосистема Python добре підходить для проектів на основі AI. Простота Python, велика спільнота та доступність багатьох інструментів дозволяють розробникам створювати практично ідеальні інфраструктури, зберігаючи при цьому акцент на бізнес-завданнях (рис. 2.2). Python має доступ до надійного бібліотечного середовища, розробники можуть виконувати складні завдання без значного обсягу коду. Завдяки її адаптованості тестування робляться дуже просто. Тести можуть виконуватись на будь-якій платформі, включаючи Windows, MacOS, Linux та Unix [7].

Python стає однією з найпопулярніших мов програмування у світі, якою користуються як новачки, так і експерти. Залучати розробників із необхідними навичками для участі у проектах з машинного навчання буде простіше зі зростанням його популярності.

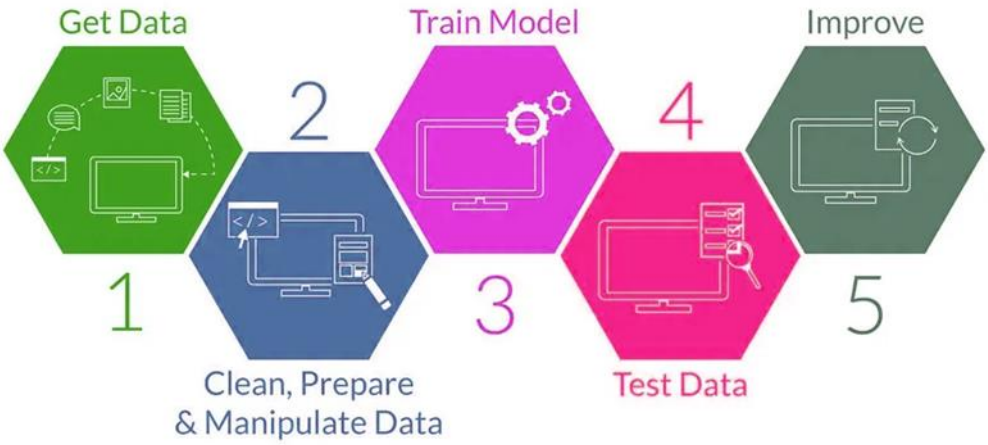


Рисунок 2.2 – Екосистема Python для штучного інтелекту

2.2 Бібліотека Tensor Flow

Машинне навчання – складна сфера дослідження. Фреймворки машинного навчання, такі як TensorFlow від Google, спрощують процес збору даних, навчання моделей, надання прогнозів та покращення майбутніх результатів, роблячи впровадження моделей машинного навчання набагато менш складним, ніж це було раніше.

TensorFlow – це бібліотека з відкритим вихідним кодом, розроблена командою Google Brain для чисельних обчислень та великомасштабного машинного навчання [8]. TensorFlow поєднує різні моделі та алгоритми машинного навчання і глибокого навчання (також відомого як нейронні мережі) і робить їх придатними для використання за допомогою загальної парадигми. Він використовує Python для надання зручного інтерфейсного API для створення програм за допомогою платформи при виконанні цих програм на високопродуктивному C++.

TensorFlow може навчати та виконувати глибокі нейронні мережі для класифікації рукописних цифр, розпізнавання зображень, вбудовування слів, рекурентних нейронних мереж, моделей послідовностей для машинного

перекладу, обробки природної мови та моделювання на основі PDE. Найкраще те, що TensorFlow пропонує великомасштабне прогнозування виробництва з використанням тих самих моделей навчання.

2.2.1 Як працює TensorFlow

TensorFlow дозволяє розробникам створювати графи потоків даних, що є структурами та визначають, як дані проходять через граф або набір вузлів обробки. Кожен вузол у графі символізує математичний процес, а кожне ребро між вузлами – це тензор, що є багатовимірним масивом даних [8].

TensorFlow надає програмістам ці можливості через мову програмування Python. Python простий у вивченні та використанні та пропонує прості методи для представлення зв'язку високорівневих абстракцій. Вузли та тензори в TensorFlow – це об'єкти Python, а програми TensorFlow самі по собі є програмами Python.

Проте справжні математичні розрахунки у Python не виконуються. Бібліотеки перетворення доступні через TensorFlow створюються, як ефективні двійкові файли C++. Python просто розподіляє трафік між компонентами та пропонує програмні абстракції на високому рівні для їхнього з'єднання.

Програми TensorFlow можуть виконуватися практично на будь-якому зручному пристрої, включаючи локальний ПК, хмарний кластер, пристрої iOS та Android, центральні та графічні процесори. Використовуючи хмару Google, можна запустити TensorFlow на унікальному обладнанні Google Tensor Processing Unit (TPU) для додаткового прискорення. Однак моделі, згенеровані TensorFlow, можна встановити практично на будь-який пристрій, де вони використовуватимуться для прогнозування [8].

TensorFlow 2.0, випущений у жовтні 2019 року, багато в чому змінив структуру на основі відгуків користувачів, щоб спростити роботу (наприклад, за рахунок використання щодо простого API-інтерфейсу Keras для навчання моделей) та зробили її продуктивнішою. Новий API спрощує виконання розподіленого навчання, а підтримка TensorFlow Lite дозволяє розгортати моделі у широкому діапазоні систем. Проте, щоб повною мірою використати нові можливості TensorFlow 2.0, код, розроблений для більш старих версій TensorFlow, необхідно переписати.

2.2.2 Переваги TensorFlow

Абстракція – найбільша перевага TensorFlow для розробки машинного навчання. Замість того, щоб зосереджуватись на особливостях реалізації алгоритмів або визначенні того, як пов'язати вихід однієї функції з входом іншої, розробник може зосередитись на загальній логіці програми. TensorFlow обробляє тонкощі у фоновому режимі.

TensorFlow надає додаткові зручності для розробників, яким необхідно налагоджувати програми TensorFlow та розбиратися в них. Замість того, щоб генерувати весь граф як один непрозорий об'єкт та оцінювати його відразу, режим активного виконання дозволяє оцінювати та змінювати кожну дію графа незалежно та відкрито. Використовуючи інтерактивний веб-інтерфейс, пакет візуалізації TensorBoard дозволяє досліджувати та аналізувати виконання графіків [8].

TensorFlow також отримує вигоду від спонсорства Google як комерційного підприємства. Google не тільки сприяла швидкому розвитку проекту, але й створила безліч важливих пропозицій на основі TensorFlow, які спрощують розгортання та використання:

– графічний процесор TPU для підвищення продуктивності у хмарі Google;

– онлайн центр для обміну моделями, створеними за допомогою фреймворку;

– браузерна та мобільна версії фреймворку.

2.2.3 TensorFlow у порівнянні з конкуруючими бібліотеками

TensorFlow конкурує з багатьма альтернативними фреймворками для машинного навчання. PyTorch, CNTK і MXNet – це три важливі фреймворки, що відповідають кільком схожим вимогам [8]. Їх переваги та недоліки щодо TensorFlow:

– PyTorch, крім того, що він написаний на Python, має багато спільного з TensorFlow, включаючи компоненти з апаратним прискоренням, високоінтерактивну парадигму розробки, яка дозволяє працювати в процесі проектування, а також включення кількох корисних компонентів. PyTorch –

чудовий варіант для швидкої розробки проектів, тоді як TensorFlow краще підходить для більших проектів і складніших процесів.

– CNTK, Microsoft Cognitive Toolkit, використовує структуру графа для представлення потоку даних, аналогічну TensorFlow, але переважно орієнтований на розробку нейронних мереж глибокого навчання. CNTK швидше виконує різні завдання нейронної мережі та пропонує ширший набір API (Python, C++, C#, Java). Однак CNTK не такий простий для розуміння або розгортання, як TensorFlow.

– Apache MXNet, прийняте Amazon як провідне середовище глибокого навчання на AWS, може майже лінійно масштабуватися на безліч графічних процесорів та безліч комп'ютерів. Він також підтримує широкий спектр мовних API, включаючи Python, C++, Scala, R, JavaScript, Julia, Perl та Go, хоча його власні API не такі зручні для користувача, як у TensorFlow.

3 ТРЕНУВАННЯ МОДЕЛІ РОЗПІЗНАВАННЯ ОБ'ЄКТІВ

Ідентифікація об'єктів – це завдання комп'ютерного зору, на яку останнім часом вплинули досягнення у галузі машинного навчання.

У минулому створення детектора предметів здавалося трудомісткою та складною операцією. Тепер за допомогою таких інструментів, як TensorFlow Object Detection API, можна легко та швидко створювати надійні моделі.

У створеному проекті використовується друга ітерація TensorFlow Object Identification API, яка:

- Підтримує TensorFlow 2;
- Дозволяє використовувати передові архітектури моделей для виявлення об'єктів;
- Забезпечує простий метод налаштування моделей.

Була використана модель на основі EfficientDet. Для підтримки навчання на основі графічного процесора, також необхідно встановити декілька бібліотек на комп'ютер з графічним процесором NVIDIA з підтримкою CUDA. В операційній системі повинні бути встановлені як остання версія CUDA Toolkit, так і cuDNN [9].

3.1 EfficientDet для масштабованого та ефективного виявлення об'єктів

Натхненням до створення EfficientDet послужило підвищення ефективності обчислень шляхом систематичного аналізу більш ранніх сучасних методів виявлення (рис. 3.1). Як правило, детектори об'єктів складаються з трьох основних компонентів: основи, яка витягує ознаки з даного зображення, мережі ознак, яка приймає кілька рівнів ознак з основи як вхідні дані та виводить список об'єднаних ознак, що представляють характерні характеристики зображення, та мережу класів/блоків, яка використовує об'єднані функції для прогнозування класу та розташування кожного об'єкта.

EfficientDet було оцінено на широко використовуючому наборі COCO, який зазвичай використовується для ідентифікації об'єктів. EfficientDet-D7 досягає середньої точності (mAP) 52,2, використовуючи в 4 рази менше параметрів та в 9,4 рази менше обчислень, ніж попередня сучасна модель [9].

Крім того, розмір параметра та затримка центрального/графічного процесора оцінювалась між EfficientDet та більш ранніми моделями. У порівнянні з іншими детекторами моделі EfficientDet працюють у 2-4 рази швидше на графічному процесорі та в 5-11 разів швидше на центральному процесорі при тих же обмеженнях точності.

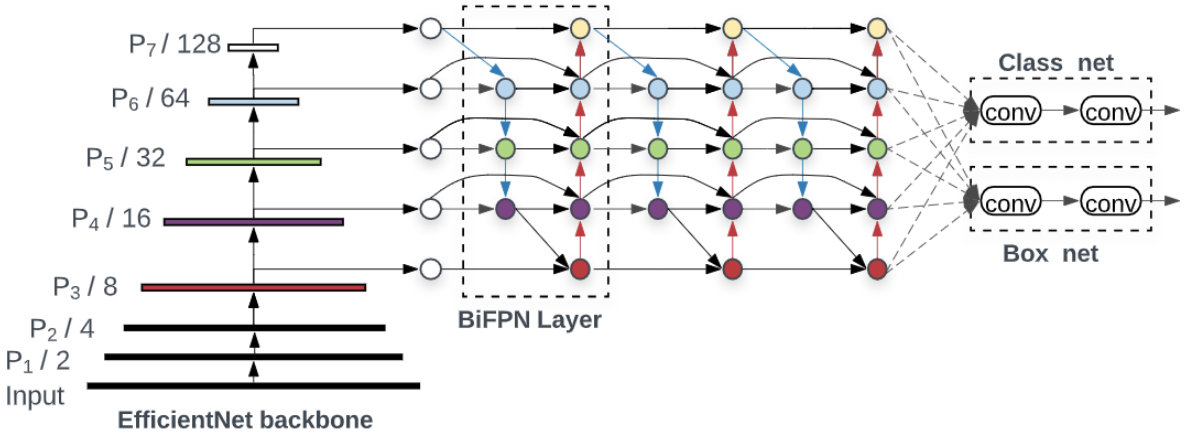


Рисунок 3.1 – Структура EfficientDet, яка використовує EfficientNet як основу мережі та нещодавно розроблену функціональну мережу BiFPN

3.2 Встановлення та налаштування

Необхідно створити папку з ім'ям Tensorflow у будь-якому місці.

3.2.1 Створення нового віртуального середовища

Відкривши вікно терміналу використовуючи команду cd необхідно перейти до папки Tensorflow, створеної раніше (лістинг3.1). Використовуючи бібліотеку venv, необхідно створити нове віртуальне середовище (рис. 3.2).

```
pip3 install virtualenv
python -m venv tensorflow2_env
tensorflow2_env\Scripts\Activate.ps1
pip install tensorflow==2.*
```

Лістинг 3.1 – Встановлення бібліотеки venv та tensorflow другої версії, створення нового середовища та його активація

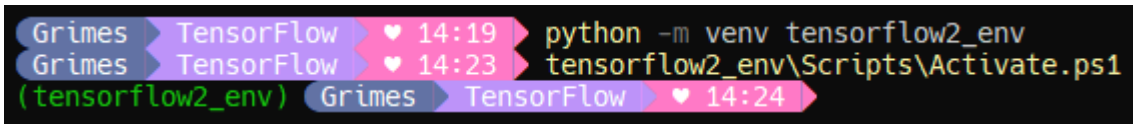


Рисунок 3.2 – Активація віртуального середовища

3.2.2 Скачування та вилучення TensorFlow Model Garden

Model Garden – це офіційний репозиторій TensorFlow на github. На цьому етапі необхідно клонувати цей репозиторій на локальну машину. Далі необхідна бібліотека Protobuf, оскільки TensorFlow Object Detection API використовує її за замовчуванням для вказівки моделі та параметрів навчання [10]. Всередині TensorFlow необхідно створити директорію protoc та розпакувати туди файли завантажені з релізів на github. Після цього необхідно скомпілювати прото файли (лістинг 3.2).

```
git clone https://github.com/tensorflow/models.git
protoc/bin/protoc models/research/object_detection/protos/*.proto
--python_out=.
```

Лістинг 3.2 – Клонування моделей Model Garden та компілювання всіх proto файлів

3.2.3 Встановлення COCO-API

COCO API – це масивний набір даних для ідентифікації об'єктів, сегментації та підписів. Є залежністю, яка безпосередньо не пов'язана з API виявлення об'єктів, томуйого необхідно встановлювати окремо (лістинг 3.3). Ручне встановлення COCO API додає безліч додаткових можливостей, таких як доступність набору загальних показників виявлення або сегментації для оцінки моделі.

```
pip install cython
pip install git+https://github.com/philferriere/cocoapi.git
```

Лістинг 3.3 – Встановлювання Cython та клонування COCO API

3.2.4 Встановлення Object Detection API

На цьому етапі завершується встановлення всіх необхідних компонентів для тренування власної моделі розпізнавання автомобілів. Необхідно встановити API виявлення об'єктів (рис. 3.3), для цього потрібно перейти в директорію /models/research та виконати низку команд (лістинг 3.4).

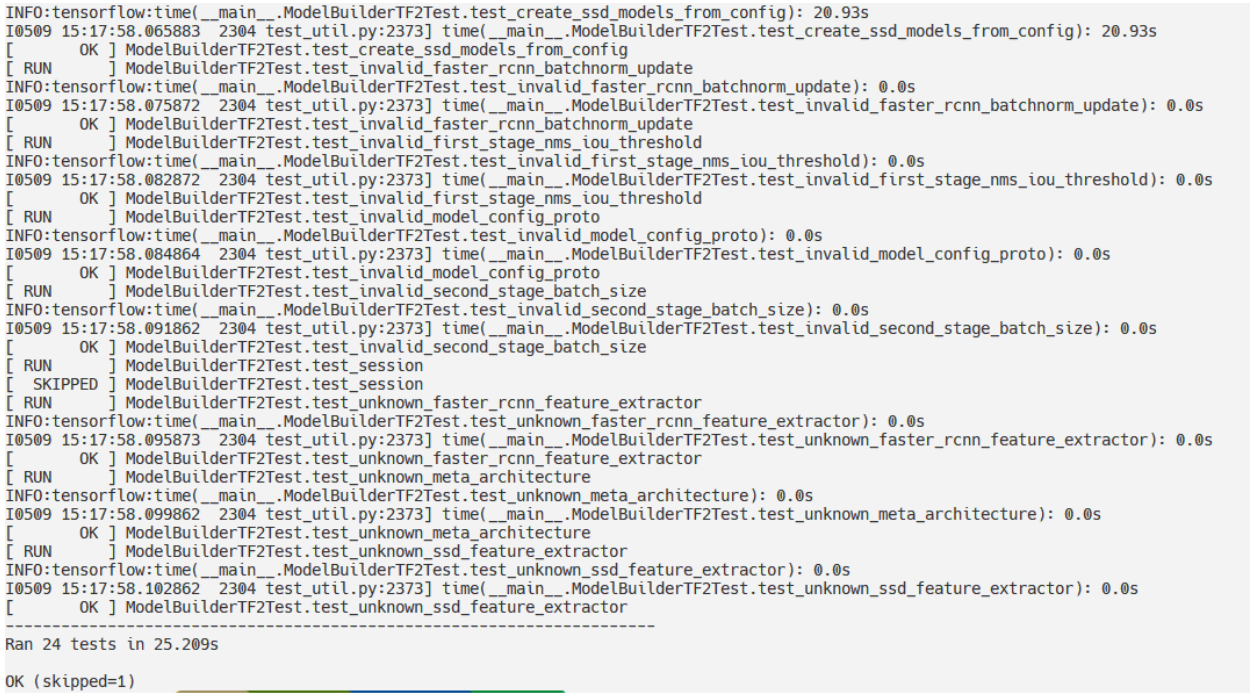


Рисунок 3.3 – Виведення до терміналу інформації про успішне проходження всіх тестів

```
cp object_detection/packages/tf2/setup.py .
python -m pip install .
python object_detection/builders/model_builder_tf2_test.py
```

Лістинг 3.4 – Встановлення API виявлення об'єктів та перевірка працездатності моделі

2.3 Підготовка данних

Усі вхідні дані для моделей на основі API виявлення об'єктів TensorFlow мають бути у форматі TFRecord [10]. У файлі TFRecord дані зберігаються у вигляді серії двійкових рядків. Це означає, що потрібно визначити структуру даних, що будуть використовуватись, перш ніж їх записувати в файл. Для цієї

мети Tensorflow пропонує два компоненти: tf.train.Example та tf.train.SequenceExample. Потрібно зберегти кожен зразок даних в одній із цих структур, серіалізувати його, а потім відправити на диск за допомогою tf.python_io.TFRecordWriter.

Для початку роботи необхідний набір даних, а саме:

– Дані (зображення) для навчання, перевірки та тестування моделі;

– Зображення повинні бути забезпечені анотаціями для виявлення об'єктів, що означає, що області для всіх цікавих об'єктів, які можуть бути присутніми в наборах даних, визначаються вручну як рамки, що обмежують, і кожній рамці призначаються мітки істинності.

У вигляді набору даних для навчання був використаний набір даних з nguyentruonglau/cars-dataset репозиторію на github.

3.4 Перетворення даних

Щоб перетворити анотації з формату xml в TFRecord, необхідно перетворити ці файли у таблицю csv, після чого конвертувати в TFRecord. Бувають різні формати xml анотацій, тому необхідно власноруч знайти відповідний скрипт для перетворення використаного формату даних в TFRecord (рис. 2.4).

```
Grimes  workspace  ♥ 17:20  python generate_tfrecord.py --csv_input=data/train_labels.csv
  --output_path=train.record --image_dir=D:\TensorFlow\cars-dataset\Cars\JPEGImages
Successfully created the TFRecords: D:\TensorFlow\workspace\train.record
```

Рисунок 3.4 – Конвертація таблиці даних до типу TFRecord

3.5 Створення картки міток

Карта міток є простим текстовим файлом у форматі pbtxt. Він пов'язує певні мітки з цілими значеннями (рис. 3.5). TensorFlow API виявлення об'єктів використовує цей файл з метою навчання та виявлення.

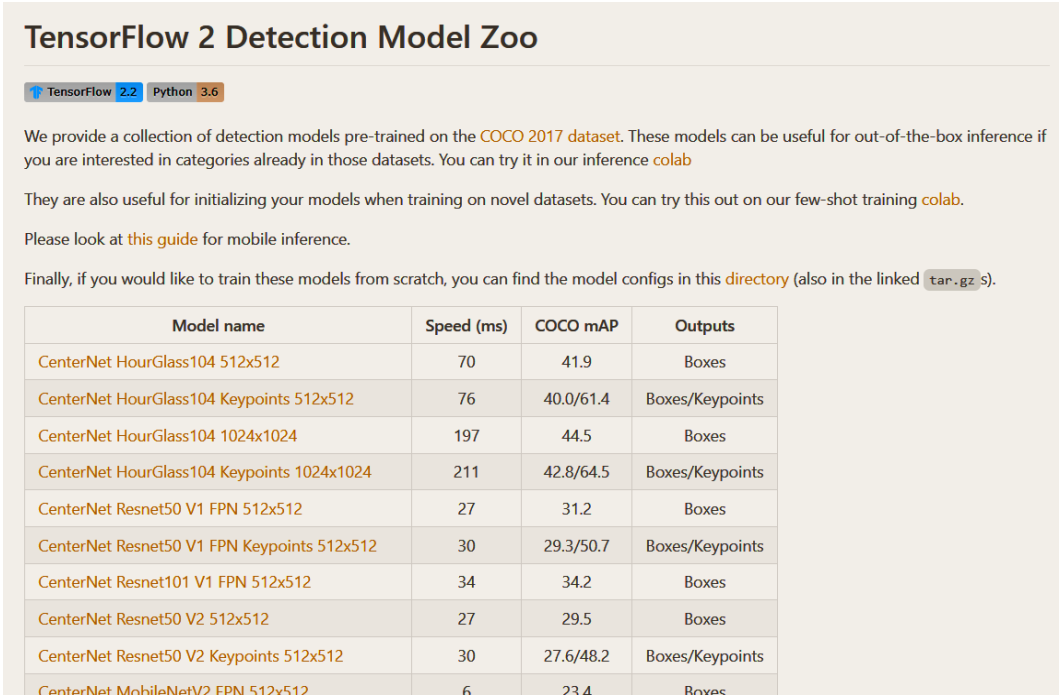
```
1 item {
2     id: 1
3     name: "vehicle"
4 }
```

Рисунок 3.5 – Файл карти міток label_map.pbtxt

3.6 Вибір моделі

Одним із найприємніших аспектів API виявлення об'єктів TensorFlow є можливість взаємодії із сучасними моделями, які вже були навчені на наборі даних COCO [10]. Ми можемо адаптувати ці моделі до наших потреб та отримати відмінні результати.

Перейшовши за посиланням до файлу розташованому на github models/research/object_detection/g3doc/tf2_detection_zoo.md, можна побачити велику кількість вже натренованих моделей, у цій роботі була використана модель EfficientDet D0 512x512 (рис. 3.6). Його потрібно завантажити та розпакувати в TensorFlow\workspace\pre_trained_models\efficientdet_d0. Після цього усередині потрібно створити папку v1 і скопіювати до неї файл pipeline.config.



TensorFlow 2 Detection Model Zoo

TensorFlow 2.2 Python 3.6

We provide a collection of detection models pre-trained on the COCO 2017 dataset. These models can be useful for out-of-the-box inference if you are interested in categories already in those datasets. You can try it in our inference colab

They are also useful for initializing your models when training on novel datasets. You can try this out on our few-shot training colab.

Please look at this guide for mobile inference.

Finally, if you would like to train these models from scratch, you can find the model configs in this directory (also in the linked `tar.gz`s).

Model name	Speed (ms)	COCO mAP	Outputs
CenterNet HourGlass104 512x512	70	41.9	Boxes
CenterNet HourGlass104 Keypoints 512x512	76	40.0/61.4	Boxes/Keypoints
CenterNet HourGlass104 1024x1024	197	44.5	Boxes
CenterNet HourGlass104 Keypoints 1024x1024	211	42.8/64.5	Boxes/Keypoints
CenterNet Resnet50 V1 FPN 512x512	27	31.2	Boxes
CenterNet Resnet50 V1 FPN Keypoints 512x512	30	29.3/50.7	Boxes/Keypoints
CenterNet Resnet101 V1 FPN 512x512	34	34.2	Boxes
CenterNet Resnet50 V2 512x512	27	29.5	Boxes
CenterNet Resnet50 V2 Keypoints 512x512	30	27.6/48.2	Boxes/Keypoints
CenterNet MobileNetV2 FPN 512x512	6	23.4	Boxes

Рисунок 3.6 – Сторінка скачування попередньо натренованих моделей

3.7 Процес налаштування

Необхідно відкрити pipeline.config текстовим редактором і незалежно від того, з якою моделлю виконується робота, до базової конфігурації повинні бути включені такі параметри моделі:

– num_classes (int). Потрібно вказати точну кількість класів, які буде виявляти модель. Значення за замовчуванням – 90, оскільки попередньо вивчена модель призначена для використання 90 елементів набору даних COCO;

– batch_size (int, повинно бути кратним 2). Це число має бути встановлене на основі обсягу доступної оперативної пам'яті. Чим більший розмір пакета, тим більше оперативної пам'яті потрібно машині/графічному процесору. У цій роботі було використано значення batch_size=8;

– fine_tune_checkpoint (str). Тут вказується шлях до контрольної точки попередньо навченої моделі;

– fine_tune_checkpoint_type (str). Оскільки ми хочемо навчити модель виявлення, у цьому полі має бути встановлено значення detection;

– use_bfloat16 (boolean). Для цієї змінної слід встановити false, якщо модель не буде навчатися на тензорному процесорі від Google;

– label_map_path (str).Тут слід вказати місце розташування раніше створеного файлу мітки map.pbtxt;

– input_path (str). Тут вказується розташування файлів train.record та validation.record, які були створені раніше. Маршрут до validation.record повинен бути встановлений у eval_input_reader, тоді як шлях train.record повинен бути встановлений у train_input_reader.

3.8 Навчання моделі

Із папки API виявлення об'єктів TensorFlow необхідно скопіювати файл model_main_tf2.py до Tensorflow/workspace, а потім запустити процес навчання використовуючи раніше налаштований файл конфігурації та претреновану модель (лістинг 3.5).

```
      python model_main_tf2.py --
pipeline_config_path="pre_trained_models\efficientdet_d0\v1\pipeline.config"
      --model_dir="pre_trained_models\efficientdet_d0\v1"
      --num_workers=8
      --alsologtostderr
```

Лістинг 3.5 – Запуск процесу навчання власної моделі розпізнавання дорожнього транспорту

4 МОДЕЛЮВАННЯ АЛГОРИТМУ РОБОТИ СВІТЛОФОРУ

Сигнали керування дорожнім рухом є корисними інструментами для керування автомобільним та пішохідним рухом при правильному використанні. Вони визначають право проїзду для різних транспортних потоків і в результаті суттєво впливають на транспортний потік [11].

Правильно побудовані, розташовані, експлуатовані та обслуговувані світлофори забезпечують одну або кілька з наступних переваг:

1) Вони полегшують упорядкований потік трафіку.

2) Вони збільшують пропускну здатність перехрестя, якщо:

– використовуються належні фізичні схеми та заходи контролю;

– робочі параметри сигналу переглядаються та оновлюються (при необхідності) на регулярній основі, щоб максимізувати здатність сигналу керування дорожнім рухом відповідати поточним потребам руху.

3) Вони зменшують частоту і тяжкість деяких видів нещасних випадків, особливо тих, що трапляються під прямим кутом.

4) Вони координуються для забезпечення безперервного або майже безперервного транспортного потоку за певним маршрутом у заданому темпі за сприятливих обставин.

5) Вони використовуються для зупинки інтенсивного руху через певні проміжки часу, щоб дозволити пішоходам або транспортним засобам перетнути дорогу. На перехрестях світлофори часто розглядаються як засіб від усіх проблем із дорожнім рухом. Ця помилка призвела до встановлення сигналів керування дорожнім рухом у кількох місцях, де вони не потрібні, що вплинуло на безпеку та ефективність руху транспортних засобів, велосипедів та пішоходів.

Навіть якщо це виправдано дорожнім рухом та дорожніми умовами, сигнали керування дорожнім рухом можуть бути погано спроектовані, невдало розташовані, неправильно експлуатуватись та погано обслуговуватися. Неправильні або необґрунтовані сигнали керування дорожнім рухом можуть мати один або кілька таких негативних наслідків:

1) Надмірна затримка;

2) Надмірна непокора сигнальним показанням;

3) Більш широке використання менш відповідних маршрутів, оскільки учасники дорожнього руху намагаються уникнути сигналів керування рухом;

4) Значне збільшення частоти зіткнень (особливо ударів ззаду).

Фундаментальна мета контролювання сигналу даними зі штучного інтелекту полягає в тому, щоб наладити ефективний проїзд транспортних засобів на перехрестях. Для досягнення цієї мети потрібен алгоритм, який розподіляє право проїзду між різними користувачами. Цей алгоритм враховує кількість машин на світлофорі, використовуючи штучний інтелект. Машини будуть перенаправлені на інший маршрут, якщо кількість машин перевищить допустиме число і розвантаження світлофором в даний момент часу не допоможе [11].

Тривалість циклу (час, необхідний для обслуговування всіх фаз сигналу), час зеленого руху та інтервали очищення є одними з численних параметрів синхронізації сигналів світлофора, які впливають на ефективність перетину. Збільшення зеленого періоду для транспортного потоку може зменшити його затримку і кількість автомобілів, що зупиняються. Однак збільшення тривалості зеленого кольору для одного руху часто призводить до більшої затримки та зупинки для іншого руху [12]. Інтелектуальна схема синхронізації сигналів розподіляє час пропорційно до потреби перехрестя і підтримує мінімальну тривалість циклу.

Зв'язок між синхронізацією сигналу та безпекою також вирішується за допомогою конструкції перехрестя та точних критеріїв синхронізації. Наприклад, ціль жовтого інтервалу зміни полягає в тому, щоб полегшити безпечну передачу переважного права руху від одного руху до іншого. Перевага цього періоду з точки зору безпеки, швидше за все, буде досягнута, коли його довжина відповідає потребам автомобілів, що наближаються до перехрестя на початку жовтого сигналу. Ця вимога стосується здатності водія розпізнавати жовтий сигнал і визначати, чи можуть вони зупинитися перед обмежувальною лінією або безпечніше проїхати перехрестя [12]. Їх вибір зупинитися чи продовжити залежить від низки змінних, включаючи швидкість. Жовті інтервали, заплановані належним чином, можуть запобігти плутанини водіїв. Синхронізації сигналів, які обмежують кількість пауз та усувають затримки, можуть забезпечити додаткові переваги у плані безпеки. Синхронізація сигналів світлофора повинна враховувати пішоходів, дорожню

ситуацію, інтервали заміни та очищення. На ці параметри синхронізації сигналу можуть впливати сусідні перехрестя, але вони застосовуються до кожного перехрестя, що розглядається як незалежний об'єкт.

Переваги із добре наладженою системою контролю дорожнього руху на всій території можуть перевищувати витрати у співвідношенні 40:1 [13]. Система із штучним інтелектом може скорочувати час у дорозі, покращить якість повітря, знизить кількість зіткнень певних видів та серйозності, а також знизить незадоволеність автомобілістів. Таким чином, використання цієї системи може допомогти досягнути:

– Скорочення затримок на дорогах на 15–40%, економія часу на шляху до 25% та скорочення зупинок на 10–40%. Наприклад, якщо водій витрачатиме дві години щодня на виконання доручень та дорогу на роботу і назад, добре налаштована система заощадить йому 50 годин на рік.

– Скорочення витрат пального до 10 відсотків. Це спричинить щорічну економію близько 170 мільярдів галонів моторного палива в національному масштабі.

– Скорочення шкідливих викидів (окис вуглецю, оксиди азоту та леткі органічні сполуки) до 22 відсотків. Згідно з офіційними даними у галузі наземного транспорту, основною причиною забруднення повітря у містах є автотранспортні засоби. Більше того, за даними EPA, автомобілі щороку виробляють 3 мільярди фунтів забруднювачів повітря.

4.1 Алгоритм управління пристроями регулювання дорожнього руху

Завантаженість доріг – це відношення кількості автомобілів, які проїжджають дорогою за певний період часу, до максимальної кількості автомобілів, які можуть проїхати за той же період часу [12]. Автомобілі прибувають послідовно з усіх напрямків на перехресті, а це означає, що затори на дорогах у кожному напрямі постійні. Затор на дорозі_1 (в обох напрямках) дорівнює v_1, а на дорозі_2 – v_2, тому обмеження будуть такі ($0 \leq v_1, v_2 \leq 1$). Частка часу горіння зеленого світла у напрямі дороги_1 позначається за t_1, а частка часу горіння зеленого світла у напрямі дороги_2 – за $t_2 = 1 - t_1$.

4.1.1 Занадто багато машин

Якщо $v_1 + v_2 > 1$, то на перехресті утворюватиметься затор незалежно від налаштування світлофора.

4.1.2 Максимально можлива кількість машин

Якщо $v_1 + v_2 = 1$, то ділянки часу, протягом яких світиться зелене світло на дорогах дорога_1 і дорога_2, повинні також бути v_1 і v_2, щоб запобігти заторам на дорогах. Таким чином, $t_1 = v_1$ та $t_2 = v_2$. T секунд – час, за який сигнал світлофора на перехресті здійснює один цикл (горить червоний, горить зелений). На дорозі дорога_2 червоне світло горить протягом v_1T секунд, а потім зелене світло протягом v_2T секунд. Відстань між двома автомобілями, що рухаються у цьому напрямку, дорівнює m_1 секунд. Час очікування для транспортних засобів, що прибувають на перехрестя на червоне світло, дорівнює v_1T, $v_1T - m_1$, $v_1T - m_2$. Час очікування для транспортних засобів, що прибувають на перехрестя на зелене світло, дорівнює нулю. Тому весь час очікування кожного циклу наведено у виразі (4.1).

$$\sum_{i=0}^{\left\lfloor \frac{Tv_1}{m_1} \right\rfloor} \left(Tv_1 - im_1\right). \tag{4.1}$$

Таким чином, вираз перетворюється у наступний вигляд (вираз 4.2):

$$\sum_{i=0}^{\left\lfloor \frac{Tv_1}{m_1} \right\rfloor} \left(Tv_1 - im_1\right) = \left(\frac{Tv_1}{m_1} + 1\right) Tv_1 - m_1 \frac{\frac{Tv_1}{m_1}\left(\frac{Tv_1}{m_1} + 1\right)}{2} = \frac{Tv_1\left(\frac{Tv_1}{m_1} + 1\right)}{2}. \tag{4.2}$$

Кількість автомобілів, що прибувають на перехрестя за T секунд, однаково, тому середній час очікування наведено у виразі (4.3).

$$\frac{T\nu_1\left(\frac{T\nu_1}{m_1}+1\right)}{2\frac{T}{m_1}}=\frac{\nu_1(T\nu_1+m_1)}{2}. \quad (4.3)$$

Оскільки функція лінійна *T*, середній час очікування буде зменшуватися в міру зменшення *T*. Для дороги дорога_1ν_1 замінюється на ν_2, а m_1 замінюється на m_2.

Після виміру інтенсивності руху машин з використанням аналітичної системи із штучним інтелектом, натренованою раніше, необхідно налаштувати світлофори за цією формулою.

4.1.3 Не дуже багато машин

У ситуації, коли ν_1 + ν_2< 1, можна варіювати вибір частки зеленого і червоного для циклу світлофора. Якщо позначити за t_1 частку зеленого для дороги дорога_1, обмеження, що вийде для t_1 наведено о виразі (4.4).

$$\nu_1 \le t_1 \le 1-\nu_2. \quad (4.4)$$

Для оптимізації виберемо досягнення мінімального значення максимуму середнього часу очікування для машин кожному напрямку.

Обчислення середнього очікуваного значення для обох доріг будуть ідентичні наведеним у попередньому абзаці, за винятком того, що ν_i не можна замінити на t_i (вираз 4.5).

$$E_{EW}=\frac{t_1(Tt_1+m_1)}{2},$$
$$E_{NS=}\frac{t_2(Tt_2+m_2)}{2}=\frac{(1-t_1)\left(T(1-t_1)+m_2\right)}{2}. \quad (4.5)$$

Обидві функції є квадратичними параболами для постійної *T*. Протягом періоду [ν_1, 1 - ν_2] функція очікування на дорозі_2 зростає, а функція очікування

на дорозі_1 спадає. Точка, в якій зменшується найбільша з значень цих двох функцій, є точкою, в якій значення функцій збігаються (вираз 4.6).

$$t_1 = \frac{T + m_2}{m_1 + 2T + m_2},$$
$$t_2 = \frac{T + m_1}{m_1 + 2T + m_2}. \qquad (4.6)$$

Вимірявши частоту проїзду автомобіля через перехрестя, буде можливим визначити значення T і t_1, що дасть повну інформацію щодо налаштування світлофорів.

5 ПОШУК НАЙКОРОТШОГО ШЛЯХУ З АЛГОРИТМОМ ДЕЙКСТРИ

Едсгер Дейкстра, голландський вчений-комп'ютерник, вигадав алгоритм Дейкстри в 1959 році. Алгоритм Дейкстри – це алгоритм пошуку графа, який створює дерево найкоротших маршрутів шляхом вирішення проблеми найкоротшого шляху з одного джерела для графа з невід'ємною вартістю ребра [14]. Цей метод часто використовують у протоколах маршрутизації та інших мережевих протоколах, але він також гарно застосовується і в маршрутизації транспортних засобів.

Для кожної вихідної вершини (вузла) у графі метод визначає маршрут із найменшою вартістю (тобто найкоротший шлях) між цією вершиною та будь-якою іншою вершиною. Його також можна використовувати для знаходження вартості найкоротших шляхів з однієї вершини до однієї кінцевої вершини, зупинивши алгоритм після того, як найкоротший шлях до кінцевої вершини буде знайдено [15]. Наприклад, якщо вершини мережі представляють міста, а вартість крайового шляху відображає відстань між містами, з'єднаними прямою дорогою, метод Дейкстри можна використовувати для найкоротшого маршруту між одним містом та іншими містами.

Метод Дейкстри працює шляхом вирішення підзавдання k, яке обчислює найкоротший шлях від джерела до вершин серед k вершин, найближчих до джерела. Щоб метод Дейкстри працював, граф має бути спрямованим, зваженим та мати невід'ємні ребра [15]. Якщо ребра негативні, неможливо визначити найкоротший маршрут. На k-му раунді буде набір з k вершин, званий Кордон, що складається з вершин, найближчих до джерела, а вершини, що лежать поза Кордоном, будуть обчислені та поміщені у Новий Кордон. Розрахована найкоротша відстань зберігається в sDist[w]. Він містить розрахункову відстань із півдня на захід. Підхід Дейкстри визначає наступну найближчу вершину, зберігаючи чергу з мінімальним пріоритетом для вершин Нового Кордону [16].

Метод працює, зберігаючи в масиві sDist найкоротшу відстань між вершиною v та джерелом. Нуль - це найменша відстань між джерелом і собою. sDist встановлюється рівним нескінченності для решти вершин, щоб вказати, що вони ще не оброблені. Після завершення обробки алгоритмом вершин sDist

матиме найкоротшу відстань між вершинами w та s. Кордон та Новий Кордон зберігаються, що допомагає у обробці алгоритму. Кордон має k найближчих до джерела вершин і обчислив найкоротші відстані до цих вершин для шляхів, обмежених k вершинами. Вершини поза Кордоном поміщуються в колекцію з ім'ям Новий Кордон.

5.1 Ефективність

Для представлення складності/ефективності можна використовувати позначення Big-O. Big-O забезпечує альтернативний підхід для опису того, як введення впливає на годину виконання алгоритму. Він забезпечує верхню межу тривалості [16].

Ефективність алгоритму Дейкстрі змінюється на основі |V|=n DeleteMins та |E| оновлення для пріоритетних очеред.

Якщо використовується купа Фібоначчі, оптимальною межею складності є O(|E|+|V|log|V|). DeleteMins вимагає O(log|V|).

Якщо використовується стандартна двійкова купа, складність O(|E|log |E|), де | E | журнал | E | термін результатів стандартних оновлень купи.

Складність дорівнює O(|E|+|V|), якщо використаний набір є пріоритетною чергою [17]. Термін O(V|2) виглядатиме як |V| сканування неупорядкованого набору Нового Кордону для визначення розташування вершини з найменшим значенням sDist.

5.2 Переваги алгоритму Дейкстри

– Після його виконання може бути визначений маршрут із найменшою вагою до всіх постійно помічених вузлів [17].

– Для кожного проходу не потрібна нова схема.

– Складність алгоритму Дейкстри дорівнює n^2, що робить його придатним для великих завдань [17].

5.3 Недоліки алгоритму Дейкстри

– Найбільшим недоліком алгоритму є те, що він виконує пошук наосліп, тому витрачається багато часу та ресурсів [17].

– Крім того, він не здатний обробляти негативні краї. Це призводить до ациклічних графів і часто перешкоджає визначенню оптимального найкоротшого маршруту.

5.4 Застосування алгоритму Дейкстри

Алгоритм Дейкстри працює із графами, тому він може вирішити проблему лише в тому випадку, якщо її можна подати у вигляді графоподібної структури. За приклад буде взято знаходження найшвидшого маршруту від гуртожитку №7 до Харківського національного університету радіоелектроніки.

Варто зазначити, що:

– Точки відправлення, приїзду, регульовані/нерегульовані переходи та перехрестя позначені як вузли.

– Кожна дорога представлена у вигляді ребра.

– Кожна дорога має пов'язане значення. Значенням виступає відстань між точками призначення та навантаження на трафік біля світлофора. Як правило, віддається перевага ребрам з нижчими значеннями. У конкретному випадку пов'язане значення визначається відстанню між точками відправлення та приїзду.

Тепер потрібно знайти маршрут із найменшим загальним значенням, використовуючи алгоритм Дейкстри.

По-перше, алгоритм ініціалізується таким чином:

1) Призначаємо гуртожиток №7 першим вузлом.

2) Ми встановлюємо відстані між гуртожитком №7 і всіма іншими вузлами рівними нескінченності, за винятком відстані між гуртожитком №7 і самим собою, яке ми встановлюємо рівним 0.

Потім багаторазово виконуємо наступні кроки:

1) Ми вибираємо вузол з найменшим значенням як «поточний вузл» і відвідуємо всі його сусідні вузли. Коли ми відвідуємо кожного сусіда, ми оновлюємо їхню орієнтовну відстань від початкового вузла.

2) Як тільки ми відвідуємо всіх сусідів поточного вузла та оновлюємо їх відстані, ми помічаємо поточний вузол як відвідуваний. Позначка вузла як «відвіданого» означає, що ми досягли його остаточної вартості.

3) Ми повертаємось до кроку один. Алгоритм зациклюється, доки не відвідає всі вузли графа.

Оскільки значення гуртожитку №7 дорівнює 0, у нашому випадку він є поточним вузлом. Потім продовжуємо відвідування двох найближчих до нього вузлів: перехрестка на проспекті Перемоги та перехрестка на Лозовенському проспекті. На початку процедури їх значення встановлені на нескінченність, але коли ми відвідуємо вузли, ми змінюємо їх значення враховуючи відстань та кількість машин у пробці.

Потім ми відзначаємо гуртожиток №7 як «відвіданий». Ми знаємо, що його остаточна вартість дорівнює нулю і нам не потрібно відвідувати його знову. Ми продовжуємо з наступним вузлом із найменшим значенням, яким може бути Лозовенський проспект.

Ми відвідуємо всі сусідні вузли Лозовенського проспекту, які не відзначені як «відвідані». Поряд з Лозовенським проспектом є гуртожиток №7 та вулиця Клочківська, але ми ігноруємо гуртожиток №7, бо вже відвідали його. Натомість ми оновлюємо значення вулиці Клочківської, додаючи значення ребра, що з'єднує Лозовенський проспект та вулицю Клочківську, до значення Лозовенського проспекту.

5.5 Алгоритм Дейкстри в Python

5.5.1 Клас Графу

Насамперед треба створити клас Graph. Цей клас не включає жодної логіки алгоритму Дейкстри, але спростить реалізацію алгоритму.

Граф буде реалізовано як словник Python (лістинг5.1). Ключі словника відповідатимуть регульованими/нерегульованими переходам та перехрестям, а його значення відповідатимуть словникам, що записують відстані до інших переходів та перехресть на графі.

```
import sys

class Graph(object):
    def __init__(self, nodes: List, init_graph: Dict):
        self.nodes = nodes
        self.graph = self.graph_constructor(nodes, init_graph)

    def graph_constructor(self, nodes: List, init_graph: Dict):
        '''
        Цей метод забезпечує симетричність графа. Іншими словами, якщо є
        шлях від вузла A до вузла B зі значенням V, то має бути шлях від
        вузла B до вузла A зі значенням V.
        '''
        graph = {node: {} for node in nodes}
        graph.update(init_graph)

        for node, edge in graph.items():
            for adjacent_node, value in e.items():
                if nodenot in graph[adjacent_node]:
                    graph[adjacent_node][node] = value
        return graph

    def get_nodes(self):
        "Повертає вузли графа."
        return self.nodes

    def get_exterior_edges(self, node):
        "Повертає сусідів вузла."
return [out_node for out_node in self.nodes if out_node in
self.graph[node]]

    def value(self, node1: List, node2: List):
        "Повертає значення ребра між двома вузлами."
        return self.graph[node1][node2]
```

Лістинг 5.1 – Клас графа та функції для взаємодії з ним

5.5.2 Алгоритм Дейкстри

Далі реалізується алгоритм Дейкстри. Спочатку визначаться головна функція (лістинг5.2).

```
def dijkstra_algorithm(graph, start_node):
```

Лістинг 5.2 – Основна функціяалгоритм Дейкстри

Функція приймає два аргументи: graph та start_node.graph – це екземпляр класу Graph, який був створений на попередньому кроці, тоді як start_node – це вузол, з якого починається обчислення. Далі визивається метод get_nodes() для ініціалізації списку невідвіданих вузлів (лістинг 5.3).

```
unvisited_nodes = list(graph.get_nodes())
```

Лістинг 5.3 – Ініціалізації списку невідвіданих вузлів

Далі створюється два словники, shortest_path і previous_nodes (лістинг 5.4):

– shortest_path – буде зберігати найвідомішу вартість відвідування кожного переходу та перехрестя у графі, починаючи з start_node. На початку вартість починається з нескінченності, але вона будет оновлювати значення по міру руху за графом.

– previous_nodes – зберігатиме траєкторію поточного найбільш відомого шляху для кожного вузла. Наприклад, якщо ми знаємо, що найкращий спосіб дістатися вулиці Клочківської – через проспект Перемоги, previous_nodes["вулицяКлочківська"] поверне "проспектПеремоги", а previous_nodes["проспектПеремоги"] поверне "Гуртожиток7". Цей словник буде використовуватися для пошуку найкоротшого шляху.

```
shortest_path = {}
previous_nodes = {}

# max_value буде використовуватися для ініціалізації "нескінченного"
значення невідвіданих вузлів.
max_value = sys.maxsize

for node in unvisited_nodes:
    shortest_path[node] = max_value

# Однак значення початкового вузла ініціалізується із значенням 0
shortest_path[start_node] = 0
```

Лістинг 5.4 – Ініціалізація початкових значень

Тепер можливо запустити алгоритм. Алгоритм Дейкстри буде виконуватися до тих пір, поки не відвідає всі вузли графа, тому це буде представлено, як умова виходу із циклу while (лістинг 5.5).

```
while unvisited_nodes:
```

Лістинг 5.5 – Цикл відвідування усіх вузлів

Тепер алгоритм може почати відвідувати вузли. Лістинг (5.6) вказує алгоритму спочатку знайти вузол із найменшим значенням.

```
cur_min_node = None
for node in unvisited_nodes: # Ітерація по вузлам
    if cur_min_node == None:
        cur_min_node = node
    elif shortest_path[node] < shortest_path[cur_min_node]:
        cur_min_node = node
```

Лістинг 5.6 – Ітерація по вузлам

Як тільки це буде зроблено, алгоритм відвідує всіх сусідів вузла, які ще не відвідувалися (лістинг 5.7). Якщо новий шлях до сусіда краще, ніж поточний шлях, алгоритм вносить корективи в словники shortest_path і previous_nodes.

```
    # Після відвідування всіх його сусідів треба помітити поточний вузол як
«відвіданий»:
    neighbors = graph.get_exterior_edges(cur_min_node)
    for neighbor in neighbors:
        tentative_value = shortest_path[cur_min_node] +
                          graph.get_value(cur_min_node, neighbor)
        if tentative_value < shortest_path[neighbor]:
            shortest_path[neighbor] = tentative_value
            # Також оновлюється найкращий шлях до поточного вузла
            prev_nodes[neighbor] = cur_min_node
```

Лістинг 5.7 – Відвідує всіх сусідів вузла, які ще не відвідувалися

Після відвідування всіх його сусідів слід помітити поточний вузол як «відвіданий» (лістинг 5.8).

```
unvisited_nodes.remove(cur_min_node)
```

Лістинг 5.8 – Видалення вузла із списку невідвіданих вузлів

Насамкінець, повернуться два словники (лістинг 5.9):

```
return prev_nodes, shortest_path
```

Лістинг 5.9 – Повернення списку з попередніми вузлами та вузлом найкоротшого шляху

5.5.3 Допоміжна функція

Також можна створити функцію, яка виводитиме результати (лістинг 5.10). Ця функція приймає два словники, повернуті функцією dijskstra_algorithm, а також імена початкового та цільового вузлів. Він буде використовувати два словники, щоб знайти найкращий шлях і обчислити оцінку шляху.

```
def show_result(prev_nodes, shortest_path, start_node, target_node):
    path = []
    node = target_node
    while node != start_node:
        path.append(node)
        node = prev_nodes[node]
    # Додати початковий вузол вручну
    path.append(start_node)
    print(f"Було знайдено наступний кращий шлях зі значенням
{shortest_path[target_node]}.")
    print(" -> ".join(reversed(path)))
```

Лістинг 5.10 – Функція для виводу результатів

5.6 Перевірка роботи алгоритму

Тепер можливо перевірити працездатність алгоритму. Були ініціалізовані вузли та значення їх ребер (лістинг 5.11).

```
    nodes = ["LyudvihaSvobodyAve", "PeremohyAve", "LozovenkivskyyProspekt",
"KlochkivskaSt", "AkhasarovaSt", "NaukyAve", "NaukyAve2", "NovhorotskaSt",
"NovhorotskaSt2", "Bakulina", "NURE"]

    init_graph = {node: {} for node in nodes}

    init_graph["LyudvihaSvobodyAve"]["PeremohyAve"] = 425
    init_graph["PeremohyAve"]["AkhasarovaSt"] = 700
    init_graph["AkhasarovaSt"]["NaukyAve"] = 2500
    init_graph["NaukyAve"]["Bakulina"] = 3000
    init_graph["PeremohyAve"]["KlochkivskaSt"] = 1200
    init_graph["KlochkivskaSt"]["NovhorotskaSt"] = 3700
    init_graph["NovhorotskaSt"]["NaukyAve"] = 800
    init_graph["NaukyAve"]["Bakulina"] = 600
    init_graph["LyudvihaSvobodyAve"]["LozovenkivskyyProspekt"] = 450
    init_graph["LozovenkivskyyProspekt"]["KlochkivskaSt"] = 1360
    init_graph["KlochkivskaSt"]["NovhorotskaSt2"] = 4500
    init_graph["NovhorotskaSt2"]["NaukyAve2"] = 800
    init_graph["NaukyAve2"]["Bakulina"] = 600
    init_graph["Bakulina"]["NURE"] = 35
```

Лістинг 5.11 – Ініціалізація змінних для перевірки працездатності алгоритму

Ці значення були використані для створення класу об'єкта Graph (лістинг 5.12).

```
graph = Graph(nodes, init_graph)
```

Лістинг 5.12 – Створення графу

Коли граф повністю збудований, слід передати його у функцію dijkstra_algorithm() (лістинг 5.13).

```
prev_nodes, shortest_path = dijkstra_algo(graph=graph,
start_node="LyudvihaSvobodyAve")
```

Лістинг 5.13 – Одержання вирішеного алгоритмом найкоротшого шляху

А також відобразити результати алгоритму (лістинг 5.14).Допоміжна функція вивела результати алгоритму у консолі (рис. 5.1).

```
show_result(prev_nodes, shortest_path,start_node="LyudvihaSvobodyAve",
target_node="NURE")
```

Лістинг 5.14 – Виведення результату обробленого алгоритмом

```
init_graph["LyudvihaSvobodyAve"]["PeremohyAve"] = 425
init_graph["PeremohyAve"]["AkhasarovaSt"] = 700
init_graph["AkhasarovaSt"]["NaukyAve"] = 2500
init_graph["NaukyAve"]["Bakulina"] = 3000
init_graph["PeremohyAve"]["KlochkivskaSt"] = 1200
init_graph["KlochkivskaSt"]["NovhorotskaSt"] = 3700
init_graph["NovhorotskaSt"]["NaukyAve"] = 800
init_graph["NaukyAve"]["Bakulina"] = 600
init_graph["LyudvihaSvobodyAve"]["LozovenkivskyyProspekt"] = 450
init_graph["LozovenkivskyyProspekt"]["KlochkivskaSt"] = 1360
init_graph["KlochkivskaSt"]["NovhorotskaSt2"] = 4500
init_graph["NovhorotskaSt2"]["NaukyAve2"] = 800
init_graph["NaukyAve2"]["Bakulina"] = 600
init_graph["Bakulina"]["NURE"] = 35
 Grimes  Desktop  ♥ 20:57  python .\testyy.py
Було знайдено наступний кращий шлях зі значенням 4260.
LyudvihaSvobodyAve -> PeremohyAve -> AkhasarovaSt -> NaukyAve -> Bakulina -> NURE
```

Рисунок 5.1 – Демонстрація роботи алгоритму знаходження найкоротшого шляху від гуртожитку №7 до Харківського національного університету радіоелектроніки.

ВИСНОВКИ

На даний момент наявність пробок на дорогах є актуальною проблемою інфраструктури великих міст у всьому світі, у зв'язку зі збільшенням попиту на транспорт, витратами на паливо та рівнем викидання шкідливих речовин в атмосферу.

В роботі було розглянуто ряд питань, що стосуються реалізації системи управління дорожнім потоком, заснований на штучному інтелекті, а також її застосування.

В першому розділі кваліфікаційної роботи був проаналізованштучний інтелект, його типи та підрозділи, а також сфери в яких він активно застосовується.

У другому розділі була навчена модель штучного інтелекту з розпізнавання різних об'єктів, а саме автомобілів, для подальшого використання отриманих даних у роботі.

Третій розділ присвячений питанням значимості правильного використання та налаштування світлофорів, в рамках розділу був розроблений алгоритм визначення наявності заторів для подальшого їх усунення.

В четвертому розділі описані переваги використання алгоритму Дейкстри для перенаправлення потоку автомобілів при виникненні заторів на перехрестях, для зменшення навантаженості світлофору.

В роботі підкреслено важливістьрозробки системи, яка сприятиме вільному потоку трафіку та допоможе в управлінні світлофором та використання штучного інтелекту у повсякденному житті в цілому.

Результати роботи було апробовано на двацять шостому міжнародному молодіжному форумі "Радіоелектроніка та молодь у XXI столітті", та подані на публікацію тези доповіді за тематикою кваліфікаційної роботи, але у зв'язку з військовим становищем у країні, вони будуть опубліковані пізніше.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Artificial Intelligence Enabled IoT: Traffic Congestion Reduction in Smart Cities / C. Сумро та ін. Smart Cities Symposium 2018, м. Bahrain, Bahrain. 2018.

2. Акхтар М., МорідпурС. A Review of Traffic Congestion Prediction Using Artificial Intelligence. Journal of Advanced Transportation. 2021. Т. 2021. С. 1–18.

3. Зафар Н., Уль Хак И. Traffic congestion prediction based on Estimated Time of Arrival. PLOS ONE. 2020. Т. 15, № 12. С. e0238200.

4. КумарН., Раубаль М. Applications of deep learning in congestion detection, prediction and alleviation: A survey. Transportation Research Part C: Emerging Technologies. 2021. Т. 133. С. 103432.

5. МарсландС. Python. Machine Learning. 2011. Т. 1. С. 381–398.

6. Сінгх П., МанюрА. Introduction to TensorFlow 2.0. Learn TensorFlow 2.0. Berkeley, CA, 2019. С. 1–24.

7. Traffic Flow Prediction for Smart Traffic Lights Using Machine Learning Algorithms / A. Наварро-Еспіноза та ін. Technologies. 2022. Т. 10, № 1. С. 5.

8. A Research of Traffic Prediction using Deep Learning Techniques / Б. Картика та ін. International Journal of Innovative Technology and Exploring Engineering. 2019. Т. 8, 9S2. С. 725–728.

9. ВіджаярагхаванВ., Лааванья М. AI Vehicle Classification and Detection using Deep Learning. International Journal of Engineering and Advanced Technology. 2019. Т. 9, 1S5. С. 24–28.

10. Artificial Intelligence (AI) Enabled Vehicle Detection and counting Using Deep Learning / Мохана та ін. 2022 International Conference on Computer Communication and Informatics (ICCCI), м. Coimbatore, India, 25–27 січ. 2022 р. 2022.

11. Хайбабає А., Бенекогаль Р. Traffic Signal Timing Optimization. Transportation Research Record: Journal of the Transportation Research Board. 2013. Т. 2355, № 1. С. 10–19.

12. Гедеон Г. Signal Timing Optimization. 2005. С. 55.

13. Гордон Р., Брауд К. Traffic Signal Operations and MaintenanceStaffing Guidelines. 2009. С. 84.

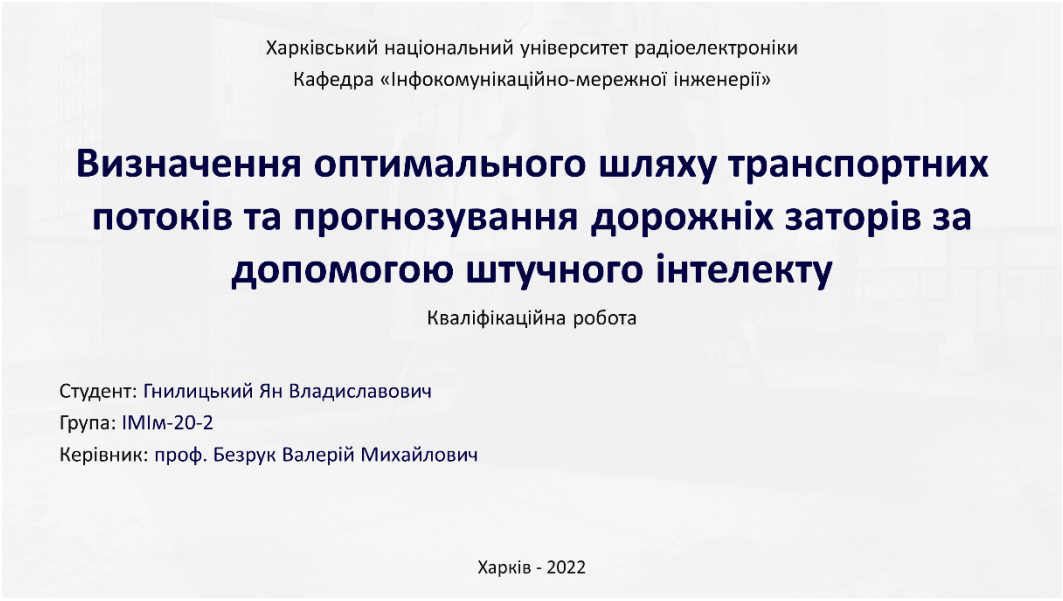
14. A fast and optimal pathfinder using airborne LiDAR data / М. Йермо та ін. ISPRS Journal of Photogrammetry and Remote Sensing. 2022. Т. 183. С. 482–495.

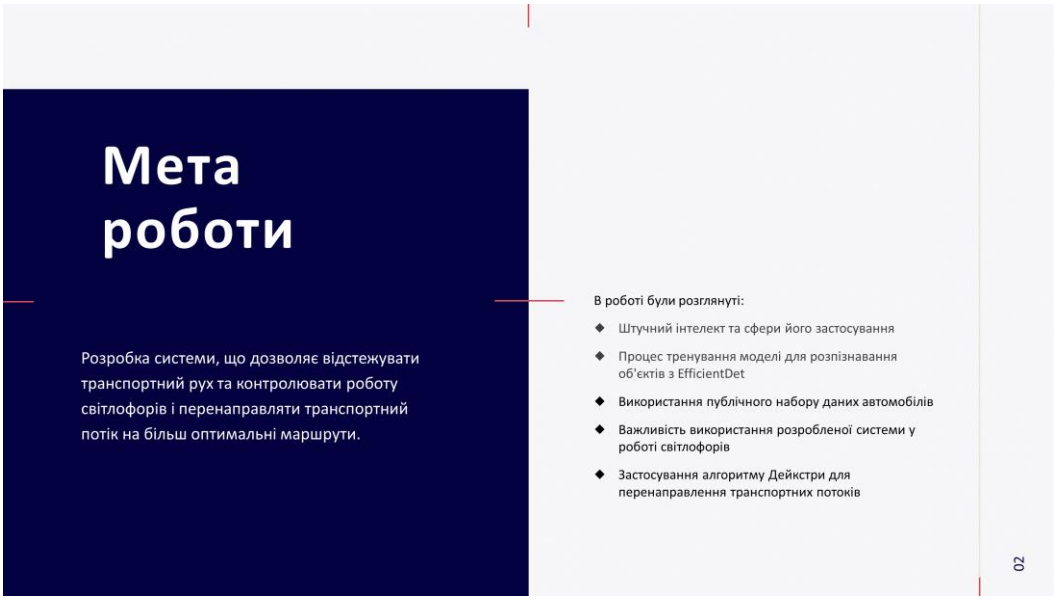
15. Бхатія Д. С. Survey of shortest Path Algorithms. International Journal of Computer Science and Engineering. 2019. Т. 6, № 11. С. 33–39.

16. Distributed shortest path query processing on dynamic road networks / Д. Жанг та ін. The VLDB Journal. 2017. Т. 26, № 3. С. 399–419.

17. Dijkstra’s Algorithm. Encyclopedia of Operations Research and Management Science. Boston, MA, 2013. С. 428.

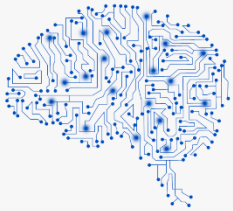
ДОДАТОК А
СЛАЙДИ ПРЕЗЕНТАЦІЇ





Типи штучного інтелекту

Нині існують різні типи штучного інтелекту



Реактивна машина

Виконує прості завдання та не потребує навчання.

З обмеженою пам'яттю

Виносить судження з урахуванням зібраної інформації, яку зберігає в пам'яті.

Теорія розуму

Розуміють поведінку та емоції людей, а також вступають у соціальні відносини та поважають почуття людей

Самоусвідомлювальні машини

Майбутнє штучного інтелекту та машинного навчання.

Підрозділи штучного інтелекту

Машинне навчання

Вчиться на попередніх даних та оновлює їх з часом.

Нейронні мережі

Побудована за принципом організації та функціонування біологічних нейронних мереж.

Глибоке навчання

Процес аналізу вхідних даних та їх зіставлення з використанням різних підходів для знаходження правильного результату.

Використання Python для штучного інтелекту

python™

- Простий та послідовний.
- Великий вибір бібліотек та фреймворків, особисто для машинного навчання.
- Незалежність від платформи (Windows, Linux, macOS, Unix).
- Відмінна спільнота та популярність серед розробників.

Фреймворки для машинного навчання

TensorFlow

- Неперевершена продуктивність
- Відкритий вихідний код інтерфейсу
- Економічно вигідний

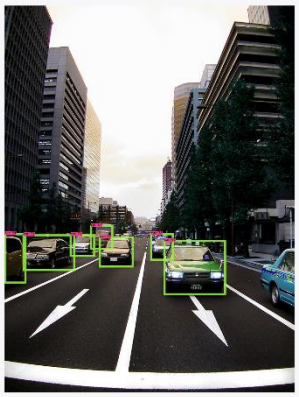
PyTorch

- Здійснення SQL-подібних запитів без чіткої схеми
- Автоматично масштабується при нових користувачах
- Висока масштабованість

CNTK

- Документи і колекції з потужними запитами
- iOS, Android та веб набори із засобів розробки з автономним доступом до даних
- Синхронізація даних у реальному часі

Тренування моделі розпізнавання об'єктів



EfficientDet

Тип моделі виявлення об'єктів, що використовує кілька оптимізаційних налаштувань, а також метод складного масштабування.

TensorFlow 2 Detection Model Zoo

Колекція моделей виявлення об'єктів, попередньо навчена на наборі даних COCO 2017.

COCO API

Великий набір даних, випущений Microsoft, який може використовуватися для виявлення об'єктів, сегментації екземплярів, семантичної сегментації, виявлення ключових точок та опису сцени.

Використання публічних наборів даних

Набір структурованих даних та інформації, призначений для навчання моделей нейронних мереж. Ідентифікація необроблених даних та додавання до них позначок, щоб показати моделі машинного навчання цільові атрибути.

Удосконалення алгоритму роботи світлофорів

Переваги налаштованого світлофора

- Скорочення затримок на дорогах
- Скорочення витрат пального
- Скорочення шкідливих викидів

Негативні наслідки необґрунтованих сигналів світлофора

- Надмірна затримка
- Надмірна непокора сигнальним показанням
- Використання менш відповідних маршрутів
- Значне збільшення частоти зіткнень

Пошук найкоротшого шляху

Застосування алгоритму Дейкстри

- Щоб знайти найкоротший шлях
- У програмах соціальних мереж
- В телефонній мережі
- Щоб знайти розташування на карті

Переваги алгоритму Дейкстри

- Маршрут із найменшою вагою до всіх постійно помічених вузлів
- Для кожного проходу не потрібна нова схема
- Складність алгоритму Дейкстри дорівнює n^2

Недоліки алгоритму Дейкстри

- Виконує пошук наосліп
- Не здатний обробляти негативні краї

Висновки

В роботі було розглянуто ряд питань, що стосуються реалізації системи управління дорожнім потоком, заснований на штучному інтелекті, а також її застосування.

В роботі підкреслено важливість розробки системи, яка сприятиме вільному потоку трафіку та допоможе в управлінні світлофором та використання штучного інтелекту у повсякденному житті в цілому.

Дякую за увагу!

yan.hnylytskyi@nure.ua

ДОДАТОК Б
ПУБЛІКАЦІЯ ЗА ТЕМАТИКОЮ РОБОТИ

ВИЗНАЧЕННЯ ОПТИМАЛЬНОГО ШЛЯХУ ТРАНСПОРТНИХ ПОТОКІВ ТА ПРОГНОЗУВАННЯ ДОРОЖНІХ ЗАТОРІВ ЗА ДОПОМОГОЮ ШТУЧНОГО ІНТЕЛЕКТУ

Гнилицький Я. В.
Науковий керівник – ст. викл. каф. ІМІ Малінін О.П.
Харківський національний університет радіоелектроніки,
61166, Харків, пр. Науки, 14, каф. ІМІ.

Traffic congestion reduces the efficiency of road networks. The decline in efficiency leads to direct and indirect costs to society such as reduced working hours or environmental pollution.

The purpose of the report is to develop a system that allows you to prevent traffic jams through artificial intelligence. The possibility of using it on city roads by using cameras installed next to traffic lights.

Сучасний світ вступив у нову еру обчислювальної техніки, яка привнесла багато визначних технологій, включаючи штучний інтелект (AI). Штучний інтелект дозволяє машинам або пристроям сприймати навколишнє середовище, а потім приймати розважливі рішення з подальшим виконанням ефективних дій, щоб максимізувати шанси на успішне виконання бажаного завдання або мети. Тим не менш, з швидким розвитком суспільства чисельність населення також різко зростає у всьому світі, особливо у міських районах порівняно із сільськими районами. Різке зростання населення призводить до збільшення попиту на транспорт і, отже, кількість транспортних засобів невпинно зростає. В результаті управління дорожнім рухом стає однією з основних проблем інфраструктури великих міст у всьому світі.

Затори на дорогах знижують рівень працездатності дорожніх мереж. Зниження рівня призводить до прямих і непрямих витрат суспільства. Безпосереднім наслідком заторів на дорогах є втрачені робочі години. Згубні наслідки заторів різко зростають, коли цінність часу як товару різко зростає під час надзвичайних ситуацій. Знаходження в пробці впливає на поведінку людей.

Високий рівень заторів може призвести до агресивної поведінки водіїв. Ця агресія може виявлятися в агресивному керуванні, що збільшує ймовірність нещасних випадків. Високий рівень заторів також призводить до збільшення викидів парникових газів, що згубно впливає на навколишнє середовище.

Метою доповіді є розробка системи, яка дозволяє запобігати заторам на дорогах за рахунок штучного інтелекту. Інфраструктура, необхідна для збору даних про трафік, удосконалювалася протягом десятиліть. Це поліпшення у поєднанні з підвищеною доступністю обчислювальних ресурсів дозволило використати можливості прогнозування глибоких нейронних мереж. Система може забезпечити зменшення заторів та може сприяти вільному потоку трафіку, а також допомогти в керуванні світлофором, але справжня перевага технології у тому, що вона звільнює людей від стомлюючої та трудомісткої повсякденної роботи та надає можливість виконувати більш важливу працю. До уваги беруться п'ять параметрів, включаючи обсяг трафіку, щільність трафіку, зайнятість, індекс завантаженості трафіку та час у дорозі під час моніторингу та прогнозування заторів на дорогах.

Прогнозуючі моделі трафіку є ключовою частиною визначення маршруту руху. Якщо прогнозується, що трафік в одному напрямку може стати інтенсивним – автоматично знаходиться альтернатива з меншим трафіком. Також враховується низка інших факторів, таких як якість доріг. Дорога асфальтована чи не асфальтована, покрита гравієм чи брудом? Такі елементи можуть утруднити рух дорогою, у зв'язку з цим ця дорога зарекомендована як частина маршруту користувача з меншою ймовірністю. Також відіграють ключову роль розмір і прямолінійність дороги - їхати шосе часто більш ефективно, ніж меншою дорогою з декількома зупинками. А звіти про інциденти дозволять швидко показати, чи закрита дорога чи провулок, чи є поблизу будівництво, несправний автомобіль чи об'єкт на дорозі і чи є несподівані зміни через зсуви, снігові бурі чи інші стихійні лиха.

Залежно від характеру даних, що збираються, застосовуються різні підходи AI для оцінки параметрів перевантаження. Прогнозування заторів на дорогах складається із збору даних та розробки моделі прогнозування. Кожен крок методології важливий і може вплинути на результати, якщо не буде виконаний правильно. Після збору даних обробка даних грає важливу роль для підготовки наборів даних для навчання та тестування. Область дослідження

відрізняється для різних досліджень. Після розробки моделі вона перевіряється з іншими базовими моделями та підтверджує справжні результати.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Moranduzzo T., Melgani F. Automatic Car Counting Method for Unmanned Aerial Vehicle Images. IEEE, 2013. 1635 c. URL: https://doi.org/10.1109/TGRS.2013.2253108.

2. Zhuang P., Shang Y., Hua B. Statistical methods to estimate vehicle count using traffic cameras. Multidimensional Systems and Signal Processing. 2008. Т. 20, № 2. С. 121–133. URL: https://doi.org/10.1007/s11045-008-0068-x.

3. Janson B. N. Dynamic traffic assignment for urban road networks. Transportation Research Part B: Methodological. 1991. Т. 25, № 2-3. С. 143–161. URL: https://doi.org/10.1016/0191-2615(91)90020-j.