Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Інформаційна система розпізнавання дорожніх знаків із використанням нейромережевих інструментів TensorFlow.

Виконав: студент IV курсу, групи СНс-41
спеціальності 122 Комп’ютерні науки
(шифр і назва спеціальності)

_____ Савчишин Я.С.
(підпис) (прізвище та ініціали)

Керівник _____ Фриз М. Є.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Литвиненко Я.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Шенгера. Н.Я.
(підпис) (прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Боднарчук І.О.
(підпис) (прізвище та ініціали)
«___» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Савчишину Ярославу Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна система розпізнавання дорожніх знаків із використанням нейромережевих інструментів TensorFlow.

Керівник роботи Фриз Михайло Євгенович, кандидат технічних наук, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « 7 » лютого 2023 року № 4/7-133

2. Термін подання студентом завершеної роботи 2023р.

3. Вихідні дані до роботи Літературні та інтернет джерела інформації про технології розробки інформаційної системи для тестування з використанням нейромережних інструментів Tensorflow

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та формування вимог до системи. Аналіз предметної області. Постановка завдання. Формування вимог до системи. Вибір мови програмування. Обґрунтування використовуваних технологій. Висновок до першого розділу. 2. Проєктування задачі та формування вимог до системи для проведення тестувань. Проєктування структури системи. Проєктування користувацького інтерфейсу. Алгоритм роботи системи. Середовище розробки PyCharm. Основні функції опрацювання даних, реалізовані в системі. Тестування програмного забезпечення. Висновки до другого розділу. 3. Безпека життєдіяльності, основи охорони праці. Вплив технологій на природу людини. Безпека життєдіяльності при водінні авто. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульний слайд. 2. Мета та актуальність. 3. Аналоги. 4. Вимоги до системи. 5. Проєктування системи. 6. Опис технологій та засобів реалізації. 7. Тестування системи. 8. Покращення системи. 9. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці			

7. Дата видачі завдання 23 січня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	25.03.2023	*Виконано*
2.	Підбір джерел про технології розробки інформаційної	27.03.2023-05.04.2023	*Виконано*
	системи для проведення тестування з використанням		
	нейромережних інструментів Tensorflow		
3.	Опрацювання джерел по темі кваліфікаційної роботи	07.04.2023-15.04.2023	*Виконано*
4.	Виконання дослідження щодо розробки інформаційної	16.04.2023-28.04.2023	*Виконано*
	системи проведення тестувань		
5.	Оформлення розділу «Аналіз предметної області та	01.05.2023-07.05.2023	*Виконано*
	формування вимог до системи»		
6.	Оформлення розділу «Проєктування задачі та	08.05.2023-15.05.2023	*Виконано*
	формування вимог до системи для проведення		
	тестувань»		
7.	Виконання завдання до розділу «Безпека	16.05.2023-24.05.2023	*Виконано*
	життєдіяльності, основи охорони праці»		
8.	Оформлення кваліфікаційної роботи	08.06.2023-12.06.2023	*Виконано*
9.	Нормоконтроль	12.06.2023-17.06.2023	*Виконано*
10.	Перевірка на плагіат		*Виконано*
11.	Попередній захист кваліфікаційної роботи		*Виконано*
12.	Захист кваліфікаційної роботи		

Студент _____ Савчишин Я.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Фриз М. Є.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Інформаційна система розпізнавання дорожніх знаків із використанням нейромережевих інструментів TensorFlow // Кваліфікаційна робота освітнього рівня «Бакалавр» // Савчишин Ярослав Сергійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра комп'ютерних наук, група СНс-41 // Тернопіль, 2023 // С. 52, рис. – 27 , табл. – 0, кресл. – 0, додат. – 3, бібліогр. – 27.

Ключові слова: система, розпізнавання дорожніх знаків, python, pycharm, tensorflow, машинне навчання, програмне забезпечення, автопілот, взаємодія з користувачем, er-діаграма.

Кваліфікаційна робота присв'ячена дослідженню нейронних мереж. Метою роботи є розробка системи для розпізнавання дорожніх знаків на основі завантажених зображень.

В першому розділі кваліфікаційної роботи проведено докладний аналіз обраної предметної області, висвітлено існуючі рішення, а також сформовано перелік вимог до розроблюваного програмного забезпечення, описано обґрунтування доцільності системи.

В другому розділі кваліфікаційної роботи було описано проєктування основних алгоритмів та структури системи, спроєктовано ключові елементи системи, протестовано систему.

В третьому розділі кваліфікаційної роботи описано безпеку життєдіяльності під час водіння авто, описано основні правила та принципи безпеки, шляхи підвищення охорони праці.

ANNOTATION

Road Signs Recognition Information System Using Tensorflow Neural Network Tools // Qualification work of the educational level "Bachelor" // Savchyshyn Yaroslav // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SNs-41 // Ternopil, 2023 // P. 52, fig. - 27, , tabl. – 0, chair. – 0, annexes. – 3, references - 27.

Keywords: road sign recognition system, python, pycharm, tensorflow, machine learning software, autopilot, interaction with correspondence, er diagram.

In the first chapter of the qualification work, a detailed analysis of the chosen subject area was conducted, existing solutions were highlighted, and a list of requirements for the developed software was formulated. The justification for the system's feasibility was also described.

In the second chapter of the qualification work, the design of the main algorithms and system structure was described, key elements of the system were designed, and the system was tested.

In the third chapter of the qualification work, occupational safety during driving was described, including the basic rules and principles of safety, as well as ways to improve occupational safety.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ACC – Adaptive cruise control.

CAGR – Compound annual growth rate.

GPS – Global Positioning System.

GTSRB – German Traffic Sign Recognition Benchmark.

GUI – Graphical User Interface.

LDW – Lane departure warning system.

LKA – Lane keeping assistance.

LCA – Life cycle assessment.

ML – Machine learning.

RFID – Radio-frequency identification.

RSA – Road Sign Assist.

TSR – Traffic sign recognition.

TSS – Toyota Safety Sense .

ДТП – Дорожньо-транспортна пригода.

ЗМІСТ

ВСТУП

Розпізнавання дорожніх знаків (TSR) - це технологія, яка дозволяє автомобілям розпізнавати дорожні знаки, такі як "стоп", "проїзд заборонений" або "обмеження швидкості", розташовані на дорозі. Ця технологія розробляється різними виробниками автомобілів і використовує методи обробки зображень для виявлення дорожніх знаків. Багато сучасних легкових, вантажних та транспортних автомобілів оснащені камерами, спрямованими вперед, які допомагають обробляти дорожні знаки.

Нейронні мережі є комп'ютерними системами, що складаються з взаємопов'язаних вузлів, що функціонують на засадах, подібних до нейронів у людському мозку. Вони використовують алгоритми для виявлення прихованих закономірностей та кореляцій в необроблених даних, здатні кластеризувати і класифікувати ці дані, а з часом постійно вдосконалюються і навчаються.

Одним з основних застосувань цієї системи є розпізнавання обмежень швидкості. Більшість систем GPS надають інформацію про ліміти швидкості, але системи розпізнавання дорожніх знаків можуть додатково відобразити цю інформацію на приладовій панелі автомобіля, щоб попередити водія про дорожні знаки. Багато автомобілів високого класу мають функцію допомоги водію, що поширена в транспорті європейських країн. Ця система може повідомляти водія про перевищення швидкості за допомогою звукових сигналів, що допомагає знизити ризик автомобільних аварій.

Метою даної кваліфікаційної роботи освітнього рівня "Бакалавр" є дослідження та розробка ефективної системи розпізнавання дорожніх знаків з використанням нейромереж. Розпізнавання дорожніх знаків з використанням нейромереж відкриває широкі перспективи для поліпшення систем безпеки на дорогах та забезпечення більш ефективного та економічного управління транспортом.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

1.1 Аналіз предметної області

Кожного року у світі понад півтора мільйона людей втрачають життя в результаті автокатастроф та від 20 до 50 мільйонів отримують травми. Існує багато причин ДТП, але основними серед них є перевищення безпечної швидкості руху, порушення правил маневрування, проїзд перехресть та пішохідних переходів, а також недотримання безпечної дистанції. Найбільша частка випадків припадає на порушення правил дорожнього руху водіями та невиконання безпечної швидкості руху.

Завдяки технологіям, життя людей стало простішим і безпечнішим. У сфері медицини, лікарі використовують високотехнологічні пристрої, які контролюють стан пацієнтів та допомагають під час операцій, на підприємствах, машини виконують важку та небезпечну роботу. У наші дні, технологія "машинного навчання" розвивається зі швидкістю блискавки. За його допомогою, комп'ютери можуть самостійно розпізнавати об'єкти, виявляти закономірності та робити логічні висновки на основі великих обсягів вхідних даних [1].

Одним з перспективних напрямків застосування технологій «машинного зору» є впровадження систем контролю руху транспортних засобів. Ці системи мають допоміжний інтерфейс, який повідомляє водія про дорожні знаки, сигнали світлофорів та інші дорожні обмеження. У деяких випадках, ці системи також надають можливість автоматичного керування автомобілем. На жаль, на сьогоднішній день такі системи доступні лише у елітних моделях автомобілів, що робить їх недоступними для більшості водіїв через їх високу ціну. Тому створення власної автоматизованої системи розпізнавання дорожніх знаків та попередження водіїв про них є вкрай актуальною задачею.

Глибоке навчання здобуло широкого огляду завдяки своїм перевагам у досягненні точних результатів. У порівнянні з традиційними алгоритмами машинного навчання, які, незважаючи на свою складність, мають просту основу, алгоритми глибокого навчання виявляються більш потужними. Навчання таких алгоритмів вимагає значних ресурсів і людського втручання при виявленні помилок, і вони працюють лише над завданням яким вони були треновані. З іншого боку, алгоритми глибокого навчання використовують нейронні мережі, які втілюють ієрархію концепцій для розуміння поставленого завдання. Складні концепції визначаються через комбінацію простіших концепцій. Все це алгоритми можуть здійснити автоматично. У контексті комп'ютерного зору, це означає, що спочатку вони визначають світлі та темні ділянки, потім класифікують лінії, потім фігури, і тільки після цього переходять до повного розпізнавання зображення. Алгоритми глибокого навчання також є більш ефективними, коли мають доступ до більшого обсягу даних, що відрізняє їх від звичайних алгоритмів машинного навчання.

Глибоке навчання не лише дало можливість використовувати велику кількість фотографій та відеозаписів для тренування алгоритмів, але також спростило процес анотування та маркування даних.

Перед початком розробки нової системи, необхідно пройти огляд існуючих технологій та здійснити оцінку їх переваг і недоліків.

Системи розпізнавання дорожніх знаків (TSR, англ. Traffic Sign Recognition), ця система зображена на рисунку 1.1.

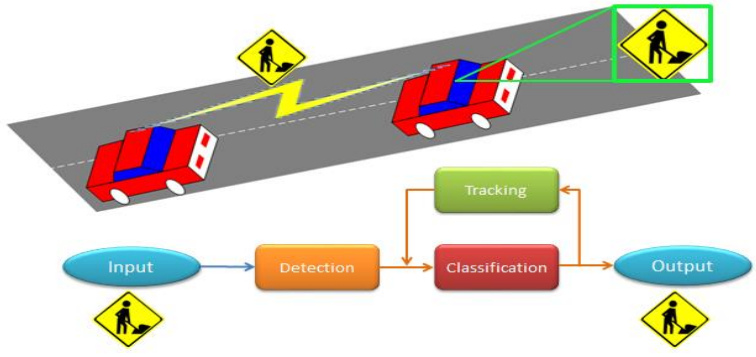

Рисунок 1.1 – Система розпізнавання дорожніх знаків (TSR)

В системах розпізнавання дорожніх знаків (TSR) використовуються камери, встановлені на транспортних засобах, що дозволяють виявляти дорожні знаки під час руху автомобіля. Зазвичай такі системи розпізнають знаки, наприклад, ті що обмежують швидкість, знаки зупинки та попереджувальні знаки, пішохідні переходи, залізничні переїзди і т.д. Основною функцією цих систем є інформування водія про недавні дорожні знаки, які можуть бути пропущені через відволікання або неуважність. Вбудована камера сканує обочину, шукаючи наявність знаків, а програмне забезпечення для обробки зображень у режимі реального часу визначає, інтерпретує та відображає їх на панелі приладів автомобіля.

Системи TSR виконують такі основні функції:

- виявлення цікавих регіонів на зображенні, де, ймовірно, можна знайти дорожній знак;
- інформування водія, відображаючи символ, що представляє розпізнаний знак;
- класифікація дорожнього знаку на основі внутрішніх наборів даних та алгоритмів нейронної мережі;
- відстеження виявленої області інтересу;

Системи TSR, засновані на машинному баченні, мають деякі проблеми розпізнавання, зокрема:

- погодні умови (дощ, туман тощо);
- варіація умов освітлення;
- спотворення зображення внаслідок руху та вібрації автомобіля;
- закриті знаки (деревами або іншими знаками в міській місцевості).

Завдяки цим факторам, інші технології, наприклад, системи на основі зв'язку RFID, вивчаються для вдосконалення або доповнення систем TSR на основі камер. Знаки, які взаємодіють з транспортними засобами за допомогою технологій V2V (Vehicle-to-Vehicle) та V2I (Vehicle-to-Infrastructure), а також

відіграють важливу роль у системах розпізнавання дорожніх знаків у найближчому майбутньому [2].

Зрештою, системи розпізнавання дорожніх знаків будуть надавати інформацію іншим електронним системам автомобіля, таким як адаптивний круїз-контроль (ACC) та системи безпеки, для розширення їх можливостей. Наприклад, у поєднанні з ACC, система TSR може розпізнавати обмеження швидкості, попереджати водія, якщо він/вона рухається занадто швидко, та автоматично пристосовувати швидкість транспортного засобу до встановленого обмеження без втручання водія.

На рисунку 1.2 зображено роботу TSS (Toyota Safety Sense).

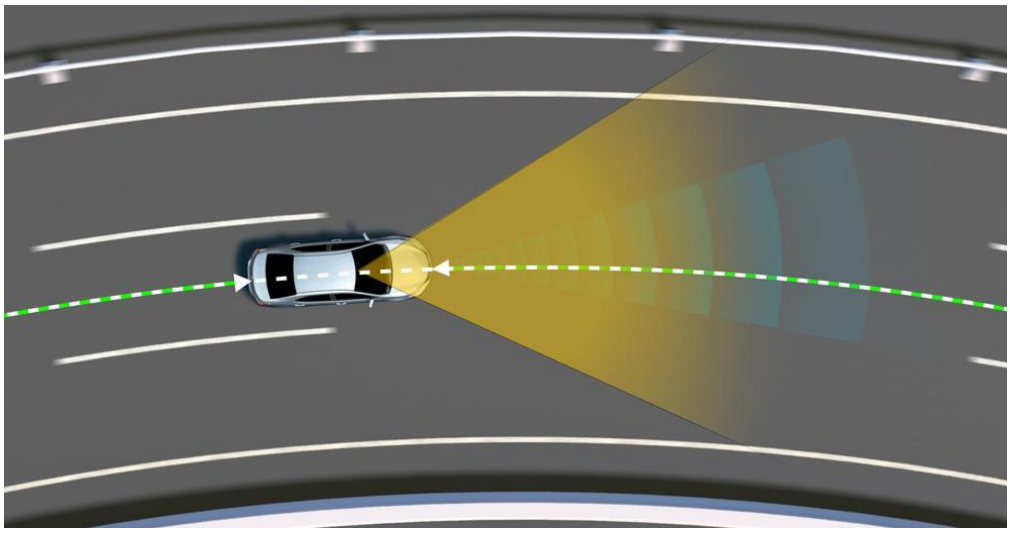

Рисунок 1.2 – Toyota Safety Sense

Система автоматичного гальмування перед іншими автомобілями на порівняно простих моделях (з TSS C) діяла в діапазоні 10-80 км/год. Однак, у новому поколінні цього комплексу, який буде встановлюватися на всі нові легкові автомобілі, стандартним діапазоном буде 10-180 км/год [3]. Щодо системи контролю рядності з автоматичним підрулюванням (LDA), в минулому вона могла розпізнавати лише білі і жовті лінії на дорозі, а тепер вона також здатна виявляти краї дороги, що зменшує ризик виїзду за межі траси. Збережено автоматичне перемикання дальнього світла.

Японські дослідники повідомляють, що впровадження першого комплексу TSS, який вже встановлено на п'ять мільйонів автомобілів, та статистика аварій свідчать про зниження на 50% ризику наїзду на інше авто. Крім того, в поєднанні з системою Intelligent Clearance Sonar, яка включає "віртуальний бампер" з автоматичним гальмуванням для уникнення зіткнень при маневруванні на низьких швидкостях, ризик знижується на 90% [4]. Друге покоління TSS очікується здатним знизити ризик ДТП ще більше. У складі TSS також впроваджено систему розпізнавання знаків RSA, яка не тільки розпізнає знаки обмеження швидкості, але і такі знаки, як «цеглина», «стоп», «обгін заборонений» та інші. Вона відображає знак на панелі приладів і надає водієві попередження, якщо порушуються правила дорожнього руху.

Преміальний відділ Toyota, Lexus, також представив наступне покоління свого комплексу безпеки - Lexus Safety System +, яке, по суті, є аналогом нового TSS. Обидва набори будуть поступово впроваджуватися в серійну продукцію компанії в Японії, Північній Америці і Європі, починаючи з нових моделей, що виводяться на ринок з 2018 року [5].

Рисунок 1.3 демонструє роботу системи LKA (Lane Keep Assist), яка допомагає утримувати автомобіль в межах смуги руху.

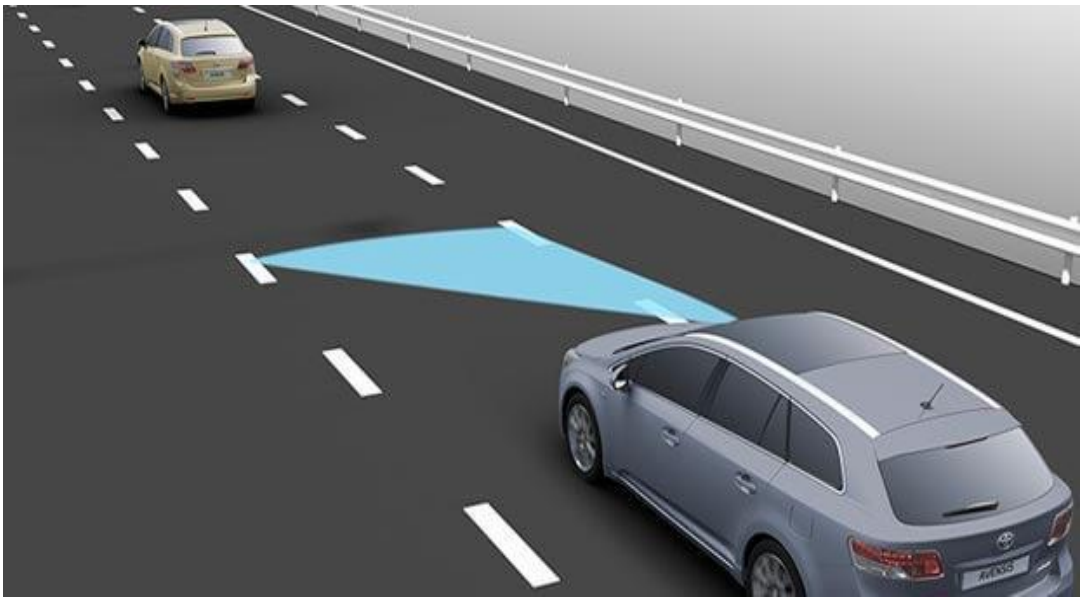

Рисунок 1.3 – Lane Keep Assist

Lane Keep Assist (LKA) і Lane Departure Warning (LDW) - це дві електронні системи, які мають невеликі варіації в назвах залежно від виробника, що намагається просунути їх продукцію. Ці системи спроєктовані для того, щоб попереджати водія, коли його автомобіль виїжджає за межі смуги руху, і водій не активує індикатори повороту. Спочатку вони були розроблені як простий спосіб виявлення того, що водій, можливо, задрімав за кермом або не звернув уваги.

Однак, з часом ці системи стали більш розширеними і вдосконаленими. Вони використовують камери та датчики для виявлення меж смуги руху і сповіщають водія про наближення до них за допомогою звукових сигналів, вібрації керма або легкого керування автомобілем, щоб повернути його в межі смуги руху.

Ці системи допомагають зменшити ризик виїзду зі смуги руху, особливо в ситуаціях, коли водій втрачає концентрацію або втомлюється. Вони сприяють підвищенню безпеки на дорозі та зменшенню аварійних ситуацій, пов'язаних з виїздом за межі смуги руху.

Системи варіюються. Деякі використовують інфрачервоні датчики, але все більш популярними стають відеокамери, які встановлюються у верхній частині лобового скла в блоці дзеркала заднього виду. Обидва типи систем відстежують дорожню розмітку, і коли автомобіль наближається до білих ліній та виходить за їх межі, водій отримує попередження. Деякі системи надають звукове попередження, інші вібрують водієве сидіння, а є такі, що легко підтягують рульове колесо й відправляють автомобіль назад до центру смуги руху, не заважаючи водієві.

Ці системи не є дешевими і, зазвичай, вони входять до складу комплексу інших технологій, таких як система попередження про "сліпі" зони, яка сповіщає водія про наявність автомобіля, який може бути непомітним через його плече. Наприклад, Nissan вимагає 400 фунтів стерлінгів за встановлення цієї системи на своєму компактному позашляховику Juke. Seat також стягує

таку саму суму за пакет безпеки, який включається до їх невеликого сімейного автомобіля Leon. У розкішному седані Mercedes E-Class ця система пропонується в рамках більш широкого пакету Lane Tracking, який коштує 735 фунтів стерлінгів.

Існують кілька загальних варіацій систем утримання смуги руху:

Попередження про виїзд з смуги руху (LDW): Ця система надає водію звукові або візуальні попередження, коли автомобіль наближається до розмітки смуги або перетинає її, якщо сигнал повороту не активований.

Асистент утримання смуги руху (LKA): Ця система автоматично керує рулем та гальмує для того, щоб утримувати автомобіль на своїй дорожній смузі.

Допомога при виїзді з дороги. Ця система забезпечує автоматичне керування рулем та гальмування, щоб уникнути виїзду автомобіля з проїжджої частини дороги.

На рисунку 1.4 зображено автопілот автомобіля Tesla.

Рисунок 1.4 – Автопілот Tesla

Автопілот - це вдосконалена система допомоги водію, яка підвищує безпеку та зручність за кермом. Вона працює завдяки використанню 8

зовнішніх камер, радару, 12 ультразвукових датчиків та потужного бортового комп'ютера, що забезпечує додатковий рівень безпеки під час поїздок.

Існують два пакети автопілота, які доступні для придбання: автопілот та повна можливість самокерування. Обидва пакети призначені для використання з уважним водієм, який готовий взяти на себе керування в будь-який момент, а система автопілота лише доповнює його дії за кермом. Важливо зауважити, що на даний момент ці функції не роблять автомобіль повністю автономним, хоча вони розроблені з метою поступового розширення самостійності в майбутньому.

1.2 Постановка завдання

Метою цієї роботи є створення програми, яка здатна розпізнавати дорожні знаки, включаючи пішохідні переходи, знаки "стоп" та інші важливі дорожні знаки, та надавати попередження про них.

Створювана система повинна задовольняти наступні основні вимоги:

- навчати модель нейронної мережі;
- розпізнавати виявлені дорожні знаки;
- повідомляти назву виявленого знаку.

Після оцінки наявних рішень та способів їх використання, поставлено завдання створити просту портативну систему розпізнавання дорожніх знаків з достатнім функціоналом та зручним інтерфейсом.

На основі аналізу переваг та недоліків існуючих рішень були сформульовані основні вимоги до цієї програми:

- простота використання;
- швидкий та простий інтерфейс;
- просте оновлення моделі нейронної мережі;
- ефективне розпізнавання дорожніх знаків.

1.3 Формування вимог до системи

Для безпечної та ефективної роботи даного програмного забезпечення, необхідно встановити конкретні вимоги до системи. Невиконання цих вимог може призвести до порушення безпеки та досягнення низької продуктивності даного програмного забезпечення.

Оскільки ця кваліфікаційна робота призначена для надання допомоги водіям під час дорожнього руху, необхідно ретельно продумати систему. Вона повинна мати зручний інтерфейс та високу швидкодію.

Високоякісне програмне забезпечення має наступні ключові властивості:

- Ефективність. Програмний продукт повинен мати здатність забезпечувати необхідну працездатність при заданих умовах щодо виділених для цього ресурсів.
- Практичність. Здатність програмного продукту бути комфортним у навчанні та використанні, та бути привабливим для користувачів.
- Функціональність. Вимога до програмного продукту ставить акцент на його здатність вирішувати задачі, які є важливими та необхідними для користувачів.
- Надійність. Ця вимога до програмного продукту вказує на його здатність до безперервної роботи та коректного виконання задач протягом тривалого періоду використання.
- Доступність. Вибіркове використання окремих компонентів програмного продукту.

1.4 Вибір мови програмування

Для розробки програми з розпізнавання дорожніх знаків в даній роботі буде використана мова програмування Python.

Python - це високорівнева об'єктно-орієнтована мова програмування з динамічною семантикою та строгим динамічним типізацією. Вона приваблює розробників своїми вбудованими структурами даних високого рівня, гнучкістю текстового обробки та динамічним зв'язуванням.

Python має простий і легкий у вивченні синтаксис, що сприяє швидкій розробці програм та полегшує читабельність коду. Інтерпретатор Python та його стандартна бібліотека доступні на всіх основних платформах у вихідній або двійковій формі.

Python був створений Гвідо ван Россумом і вперше був випущений 20 лютого 1991 року. Версія Python 2.0 з'явилася у 2000 році з новими функціями, такими як розуміння списків та система збору сміття (garbage collection) - автоматичне керування пам'яттю. Python 3.0 був випущений у 2008 році і приніс багато змін, що призвели до несумісності з великою частиною коду, написаного для Python 2 [6].

Гвідо ван Россум ставив перед собою завдання створити інтуїтивну та просту мову програмування, яка має наступні характеристики:

– бути настільки потужною, як і інші конкуруючі мови;

– мати синтаксис, який буде так само простим, як англійська мова;

– чистий синтаксис, який сприяє зрозумілості та читабельності коду;

– бути зручною для повсякденних завдань, що дозволить скоротити час розробки.

Основні переваги цієї мови програмування включають:

– велику кількість готових модулів, що полегшують розробку програм;

– переносимість програм, що дозволяє їх запускати на різних платформах без змін;

– наявність потужного та простого середовища розробки під назвою IDLE, що є частиною стандартного дистрибутиву Python;

– можливість виконання Python у діалоговому режимі, що дозволяє швидко перевіряти та експериментувати з кодом.

Сьогодні Python є однією з найпопулярніших мов програмування, завдяки своїй простоті, зрозумілому синтаксису та широкому спектру застосувань. Правила синтаксису в Python полегшують читання коду та підтримку додатків, що робить його привабливим для розробників.

1.5 Обґрунтування використовуваних технологій

Для реалізації задачі з розпізнавання дорожніх знаків на Python використовувалися такі бібліотеки: Tensorflow, NumPy, Tkinter, Pillow.

Ці бібліотеки допомагають спростити розробку програми з розпізнавання дорожніх знаків, надаючи потужні інструменти для роботи з машинним навчання.

Розвиток нейронних мереж та штучного інтелекту є актуальною темою, а популярність таких бібліотек, як TensorFlow, свідчить про широкий інтерес до цих технологій.

TensorFlow є потужною та гнучкою платформою для машинного навчання, яка надає великий функціонал для роботи з нейронними мережами, обробки зображень та іншими завданнями машинного навчання. Вона підтримується популярними мовами програмування, такими як Python та JavaScript, що спрощує її використання для розробників [7].

Комп'ютерне бачення, яке використовується для розпізнавання дорожніх знаків, є однією з областей застосування TensorFlow. Завдяки своїй потужності та екосистемі інструментів, TensorFlow дозволяє створювати та тренувати моделі глибокого навчання для розпізнавання зображень, включаючи дорожні знаки. Використовуючи методи комп'ютерного бачення, модель може визначати унікальні елементи на зображеннях та присвоювати їм відповідні класи, що відповідають різним дорожнім знакам.

TensorFlow є лише однією з багатьох бібліотек, доступних для роботи з нейронними мережами та обробкою зображень. Інші популярні бібліотеки,

такі як PyTorch, Keras та OpenCV, також використовуються в цій області і можуть мати свої переваги залежно від конкретних потреб проєкту [8].

За допомогою Tensorflow можна виконати низку завдань, таких як:

– розпізнавання голосу;
– розпізнавання дорожніх знаків;
– розпізнавання символів;
– розпізнавання одягу;
– аналіз медичних знімків;
– розпізнавання хвороби рослин.

TensorFlow використовується такими компаніями як Apple, Facebook і Google. Apple використовує Tensorflow для розробки свого голосового помічника Siri, Facebook використовує його для розпізнавання об'єктів у DeepFace, а Google використовує Tensorflow для розробки багатьох своїх додатків та послуг [9].

Tensorflow має наступний функціонал:

– збереження моделей;
– машинне навчання та кластеризація;
– навчання моделі нейронної мережі;
– виявлення об’єктів;
– обробка введеного зображення.

Навчання моделі є першим кроком у роботі з нейронною мережею. Для цього потрібні датасети, які містять об'єкти, ретельно впорядковані за їх класами. Кожен клас представляє тип об'єкта, який відрізняє його від інших (наприклад, фрукти від столових приборів або яблуко від кавуна). Наступним кроком є виявлення об'єктів на зображенні. Нейронна мережа здійснює пошук необхідного об'єкта та передає його для подальшої обробки зображення. Обробка зображення включає виділення особливих елементів для кожного об'єкта та порівняння його з вже навченою моделлю з метою призначення об'єкту певного класу.

Модуль Tkinter є потрібним для створення простого користувацького інтерфейсу (graphical user interface). Цей модуль заснований на бібліотеці Tk і використовує мову програмування Tcl для реалізації компонентів GUI. Ці компоненти включають функціональні кнопки, текстові поля, вікна, зображення та інші. Tkinter дозволяє легко керувати програмою шляхом використання готових елементів інтерфейсу без необхідності прямого виклику функцій або написання команд.

Для розробки GUI необхідно виконати наступні дії:

– імпортувати бібліотеку;

– створити головне меню;

– додати елементи користувацького інтерфейсу;

– встановлення їх функцій та властивостей;

– визначити події та їх обробники;

– розташувати елементи на головному вікні;

– відобразити головне вікно.

Модуль Tkinter дозволяє використовувати різні елементи управління, такі як текстові поля, списки та кнопки, для створення графічного інтерфейсу користувача. Ці елементи, відомі як віджети, включають:

– Label (надпис): віджет, який відображає текст або зображення.

– Text (текст): віджет для редагування багаторядкового тексту.

– Listbox (список): віджет для відображення списку елементів, з яких можна вибрати один або більше.

– Checkbutton (прапорець): віджет для відображення параметрів, які можна увімкнути або вимкнути.

– Entry (поле введення): віджет для введення однорядкових значень.

– Frame (рамка): віджет-контейнер для організації інших компонентів.

– Button (кнопка): віджет для створення кнопок.

– Message (повідомлення): віджет для відображення багаторядкових незмінних текстових полів.

– Radiobutton (перемикач): віджет, який дозволяє вибрати один варіант з декількох [10].

Бібліотека NumPy, з іншого боку, дозволяє здійснювати операції над великими масивами даних у Python. Це особливо корисно, оскільки робота з великими масивами в Python може бути повільною порівняно з компільованими мовами, такими як C++ або Java. NumPy покращує швидкість обробки масивів у Python і розширює функціональність для проведення різних операцій над ними.

Масиви NumPy мають фіксований розмір при створенні, відмінно від динамічного зростання списків Python. Зміна розміру масиву NumPy створює новий масив та видаляє оригінал.

Усі елементи масиву NumPy повинні мати однаковий тип даних, що дозволяє їм мати однаковий розмір у пам'яті. За винятком масивів об'єктів Python, змішаних з NumPy, які дозволяють мати елементи різного розміру.

Масиви NumPy сприяють вдосконаленим математичним та іншим операціям над великими об'ємами даних. Ці операції зазвичай виконуються ефективніше та з меншим об’ємом кодом, порівняно зі вбудованими функціями Python.

Багато науково-математичних пакетів, заснованих на Python, використовують масиви NumPy. Хоча більшість з них підтримують введення Python-послідовностей, вони перетворюють їх на масиви NumPy для обробки та часто виводять результати у формі масивів NumPy. Таким чином, для ефективного використання багатьох науково-математичних програм на основі Python необхідно не тільки розуміти вбудовані типи послідовностей Python, але й знати, як працювати з масивами NumPy.

Бібліотека зображень Pillow розширює можливості обробки зображень у Python. Вона підтримує різноманітні формати файлів, має ефективне

внутрішнє представлення та потужні функції обробки зображень. Ця бібліотека зображень створена для швидкого доступу до даних, збережених у різних піксельних форматах. Це надає міцну основу для загального інструменту обробки зображень.

1.6 Висновок до першого розділу

У першому розділі кваліфікаційної роботи проведено аналіз предметної області системи і визначено ряд вимог, які повинні бути задоволені системою. Цей аналіз включав детальне дослідження потреб користувачів і визначення їх вимог щодо функціональності, зручності використання та естетичного оформлення користувацького інтерфейсу.

У процесі аналізу вимог до користувацького інтерфейсу, було враховано побажання та потреби користувачів щодо інтуїтивно зрозумілого та зручного інтерфейсу. Задоволення цих вимог допоможе забезпечити зручне та ефективне взаємодію з системою.

Крім того, вимоги до структури системи були встановлені з урахуванням потреб користувачів та вимог до функціональності системи. Було визначено, які функції та можливості має мати система для задоволення потреб користувачів та досягнення поставлених цілей.

Для забезпечення повноти аналізу, був проведений огляд схожих систем, які вже існують на ринку або використовуються в подібних сферах. Цей огляд дозволив оцінити переваги та недоліки існуючих систем, а також зрозуміти, які підходи та функціональні можливості можуть бути використані в розробці нової системи.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ЗАДАЧІ ТА ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ ДЛЯ ПРОВЕДЕННЯ ТЕСТУВАНЬ

2.1 Проєктування структури системи

Метою системи є виконування наступних функцій: розпізнавання дорожніх знаків, повідомлення водію про дорожні знаки, збереження відповідних даних.

Для правильної роботи системи необхідні наступні модулі:

- Модуль для порівняння зображень з вже розпізнаними дорожніми знаками.
- Модуль розпізнавання дорожніх знаків.
- Модуль навчання моделі нейронної мережі.
- Модуль користувацького інтерфейсу.

Модуль навчання моделі нейронної мережі відповідає за навчання системи за допомогою німецького датасету дорожніх знаків.

Модуль розпізнавання дорожніх знаків є головним модулем, який коректно розпізнає знаки та передає їх у цифровому форматі.

Модуль для порівняння зображень визначає дорожній знак на зображенні шляхом порівняння з попередньо збереженими даними.

Модуль користувацького інтерфейсу забезпечує взаємодію користувача з системою розпізнавання дорожніх знаків та управління нею.

На рисунку 2.1 зображено IDEF3 діаграму роботи розпізнавання дорожнього знаку.

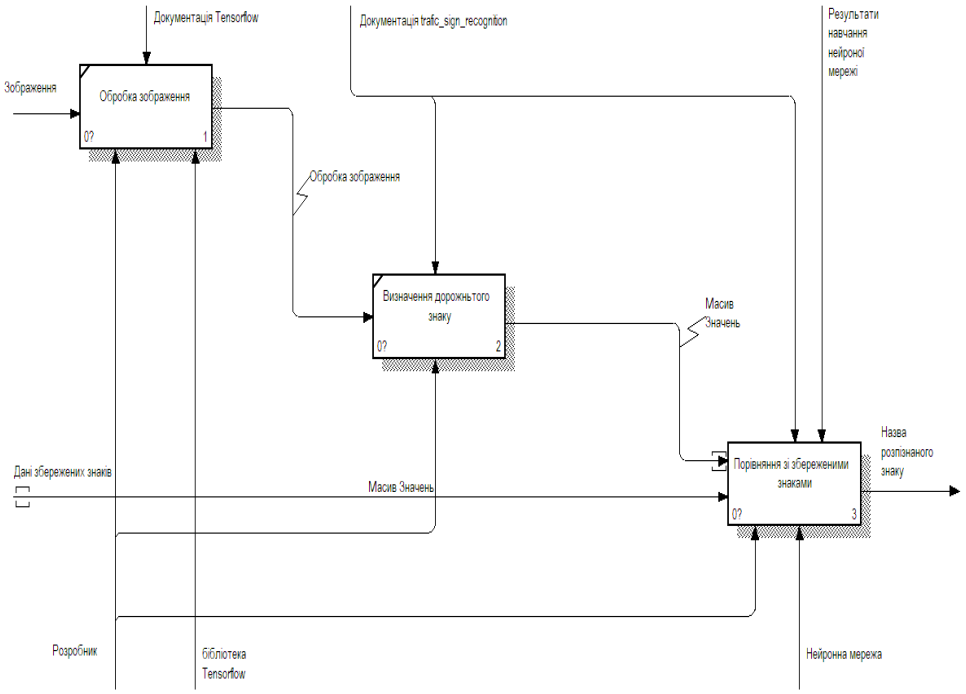

Рисунок 2.1 – IDEF3-діаграма обробки зображення

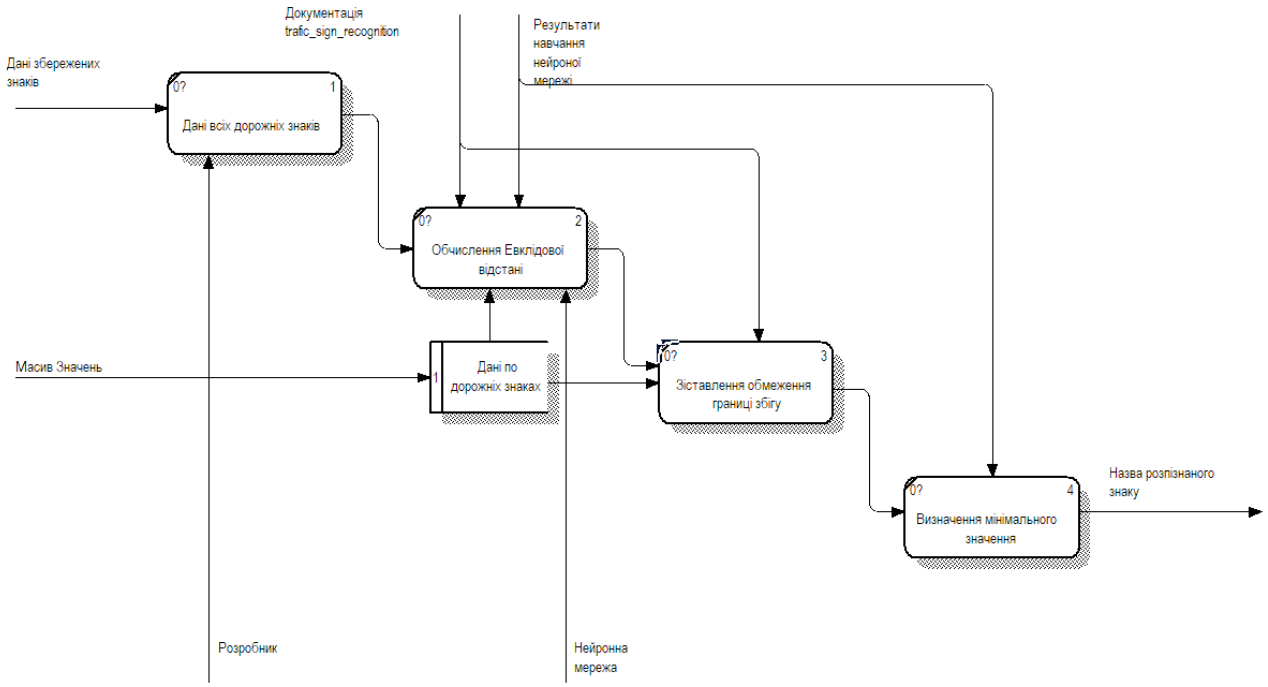

Рисунок 2.2 – IDEF3-діаграма порівняння зображень

На рисунку 2.2 зображено IDEF3 діаграму, яка демонструє порівняння просканованих дорожніх знаків зі збереженими даними.

2.2 Проєктування користувацького інтерфейсу

Для користувачів, які не володіють необхідними знаннями або досвідом роботи з програмними консолями, важливим є наявність користувацького інтерфейсу. Користувацький інтерфейс полегшує взаємодію з програмою, застосунком або веб-сайтом, дозволяючи виконувати дії за допомогою графічних елементів і кнопок. Розробка користувацького інтерфейсу є складним завданням, яке вимагає урахування уподобань користувачів та створення зручного та ефективного інтерфейсу.

Мета проєктування користувацького інтерфейсу полягає у створенні такого інтерфейсу, який спрощує та полегшує управління програмою, забезпечуючи досягнення потрібних результатів. Ключові вимоги до інтерфейсу включають високу швидкодію, простоту використання та наявність необхідного функціоналу. Важливо передбачити можливі помилки та недоліки та забезпечити безперервну роботу програми.

Елементи інтерфейсу повинні включати кнопки для завантаження дорожніх знаків, вікно для перегляду завантажених фотографій або зображень, кнопки оновлення моделі та виходу з системи. Метою є створення зручного та функціонального інтерфейсу, який задовольняє потреби користувачів.

Макет користувацького інтерфейсу зображено на рисунку 2.3.

Рисунок 2.3 – Макет користувацького інтерфейсу

З метою запобігання випадковим натисканням, функціональні кнопки можуть бути розміщені у різних частинах екрану. Це сприятиме уникненню недбалого натискання на кнопки та забезпечить більш точне та свідоме взаємодію з користувацем.

2.3 Алгоритм роботи системи

Алгоритм - це послідовність інструкцій, які чітко визначені і реалізовані на комп'ютері. Зазвичай вони використовуються для розв'язання конкретного класу задач або для виконання обчислень.

Створення комп'ютерних програм для вирішення практичних задач включає кілька етапів, таких як:

- опрацювання алгоритму вирішення задачі;
- результат вихідного завдання після виконання програми;
- формалізація та запис технічного завдання на вихідну задачу;
- написання, тестування, налагодження та документування програми.

Початковим кроком є визначення вимог до кінцевого результату програми, що вирішує практичні задачі. Алгоритм функціонування системи буде залежати від кінцевої цілі користувача. Першим етапом є ініціалізація користувацького інтерфейсу, який забезпечує взаємодію з програмою. Для цього використовується модуль створення інтерактивного користувацького інтерфейсу, який дозволяє створити вікно з текстовими елементами, функціональними кнопками та вікно для відображення завантаженого користувачем зображення, яке буде піддане розпізнаванню програмою.

Першим етапом роботи програми є навчання та збереження моделі нейронної мережі. Для навчання використовується GTSRB Dataset (German Traffic Sign Recognition Benchmark Dataset), який є великим набором даних для класифікації дорожніх знаків. Цей набір даних був створений дослідницькою групою Real-Time Computer Vision і використовувався на змаганнях IJCNN

2011 у Сан-Хосе, Каліфорнія, США. GTSRB включає 43 класи дорожніх знаків, 39209 навчальних зображень та 12630 тестових зображень. Зображення мають різну освітленість та різноманітний фон. Після успішного навчання модель зберігається у файлі програми для подальшого використання. Користувач також має можливість оновити модель нейронної мережі в разі потреби.

Користувачу наступною дією необхідно завантажити фотографію або зображення дорожнього знаку, який потрібно розпізнати. Після завантаження, модуль обробки та розпізнавання зображення приймає це зображення і проводить необхідну обробку. Використовуючи збережену модель, він порівнює завантажене зображення з навченими даними та виконує процес розпізнавання. Результат розпізнавання виводиться користувачу. Більш детальний опис обробки зображення подано на рисунку 2.4.

Рисунок 2.4 - Алгоритм роботи моделі обробки зображення

Після завершення роботи з програмою, користувач може вибрати опцію «Вихід», щоб закрити програму.

2.4 Середовище розробки PyCharm

Для того щоб розпочати розробку будь-якого програмного забезпечення, необхідно спочатку встановити середовище розробки. У даному випадку, оскільки була обрана мова програмування Python, середовищем розробки, що було використано, є JetBrains PyCharm. Для того, щоб встановити це середовище, потрібно перейти на веб-сайт JetBrains та до розділу «Download PyCharm». Далі, знайдіть кнопку «Download» для версії «Community» [11]. Фрагмент цієї сторінки зображено на рисунку 2.5.

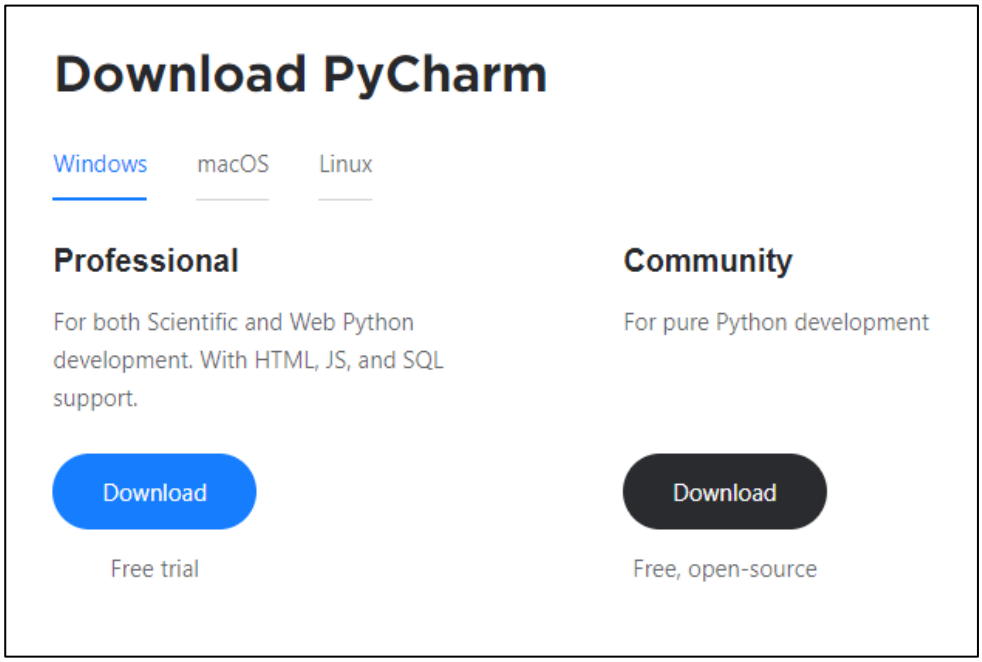

Рисунок 2.5 – Сторінка завантаження PyCharm

Після завершення завантаження, ви можете запустити цей файл .exe для початку процесу інсталяції середовища на вашому ПК. Під час інсталяції вам буде запропоновано вибрати шлях для встановлення програмних файлів PyCharm Community Edition, який зображений на рисунку 2.6.

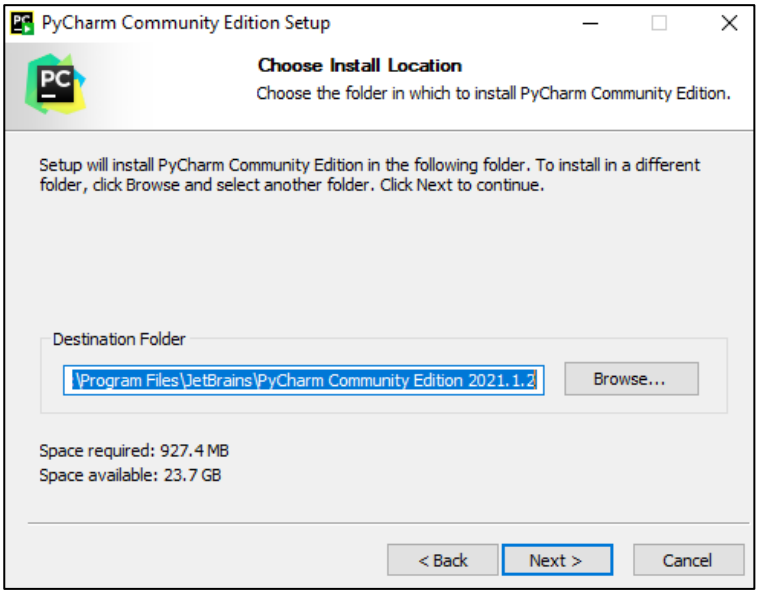

Рисунок 2.6 – Вказування шляху інсталяції

Наступним кроком є модифікація встановлення, а саме прапорці:

– створення асоціацій;

– оновлення контекстного меню;

– оновлення шляху змінних;

– створення ярлику на робочому столі.

Ці налаштування зображено на рисунку 2.7.

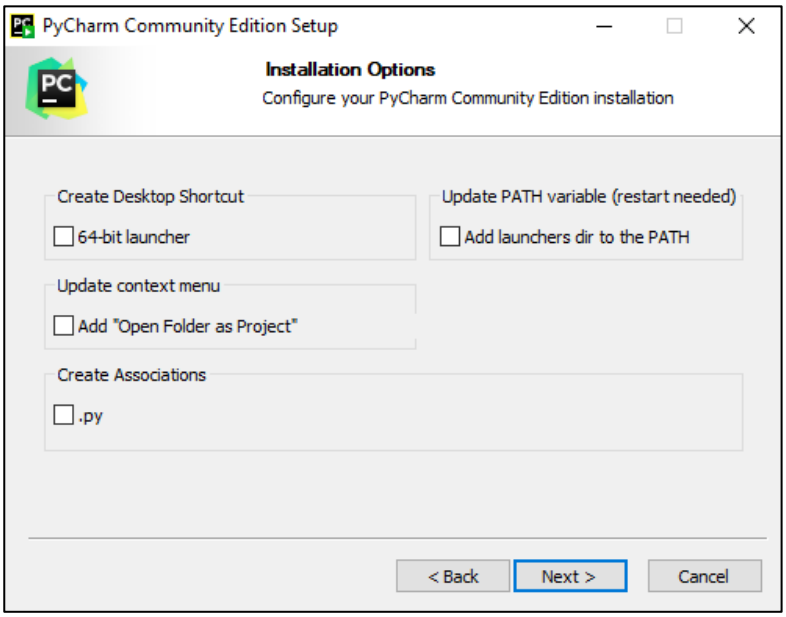

Рисунок 2.7 – Додаткові налаштування

У наступному вікні вам буде запропоновано створити папку з назвою «JetBrains» у меню «Пуск», яке зображене на рисунку 2.8. Після натискання кнопки «Install» розпочнеться процес інсталяції. Процес інсталяції буде показано на рисунку 2.9.

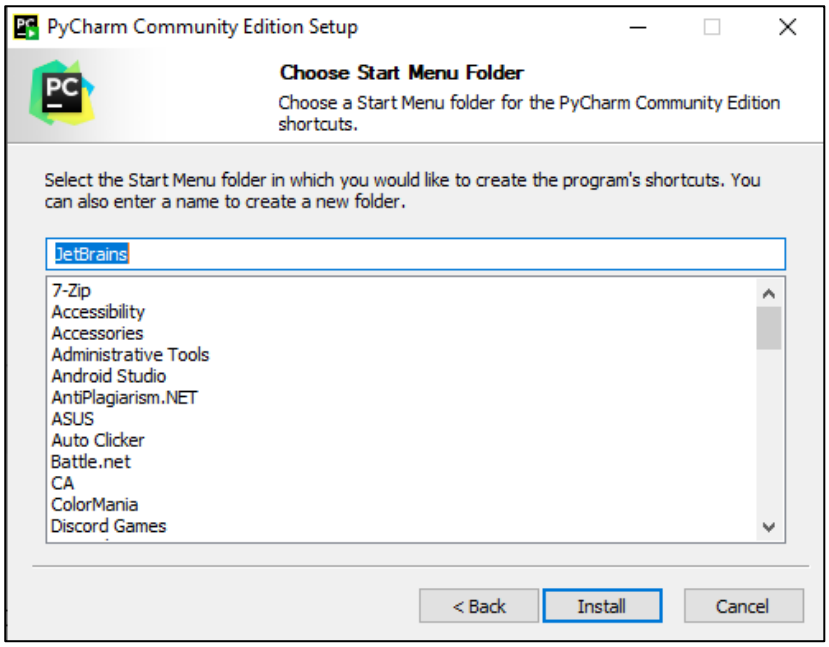

Рисунок 2.8 – Створення папки в меню «Пуск»

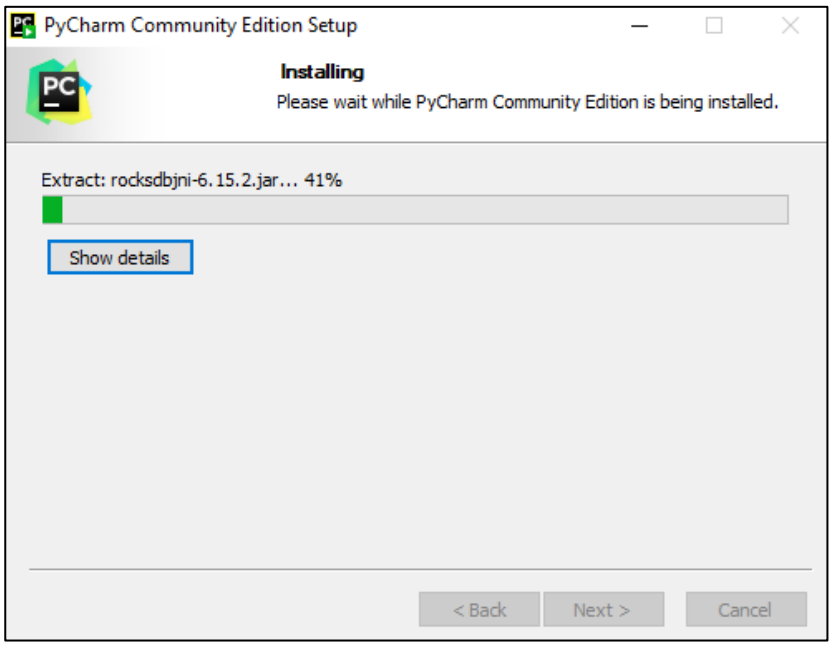

Рисунок 2.9 – Процес інсталяції

Після завершення процесу інсталяції ви зможете запустити програму. Для цього вам потрібно встановити прапорець біля «Run PyCharm Community Edition » та натиснути кнопку «Finish», як зобржено на рисунку 2.10.

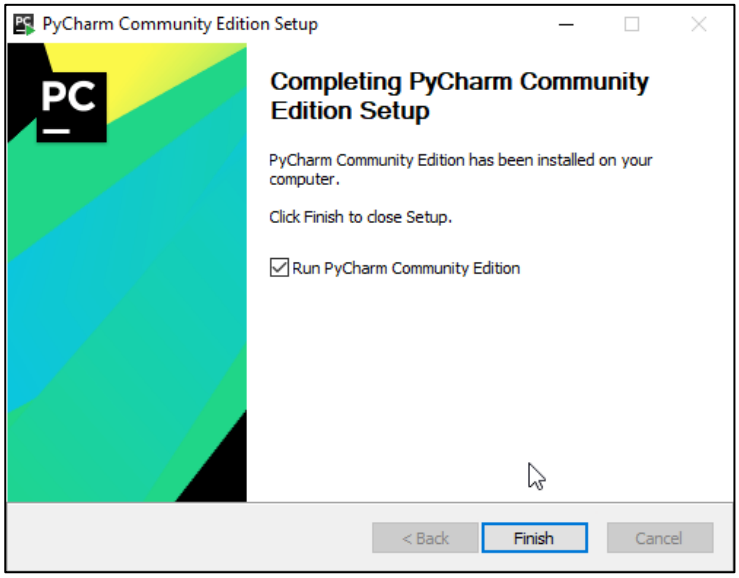

Рисунок 2.10 – Завершення інсталяції та запуск програми

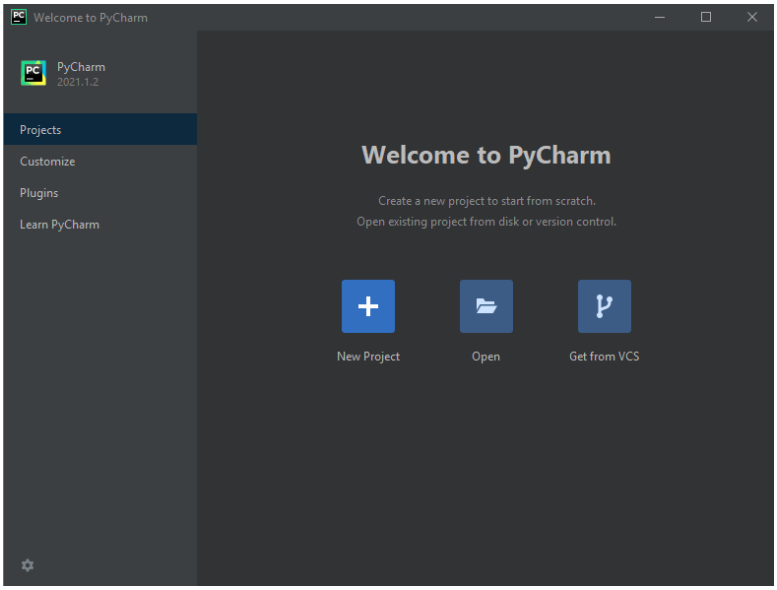

Рисунок 2.11 – Завершення інсталяції та запуск програми

Після запуску програми, відкриється початкове вікно, де ви після натискання кнопки «New Project», можна перейти до етапу реалізації роботи.

2.5 Основні функції опрацювання даних, реалізовані в системі

Першим кроком у реалізації програмного забезпечення є тренування нейронної мережі. Для цього використовується файл з назвою "RecognizeItem.py", який надано у лістингу А2 у додатку А.

Лістинг 2.1 – Ініціалізація навчання моделі починається з коду:

```
model = get_model()
     model.summary()
     tensorboard = TensorBoard(log_dir='logs/{}'.format(Name))
     history = model.fit(x_train, y_train, epochs=EPOCHS,
     callbacks=[tensorboard])
```

Лістинг 2.2 – Завантаження класів моделей з зображенням:

```
images, labels = load_data(os.path.dirname(sys.argv[0]))
```

Лістинг 2.3 – Виклик функції load_data():

```
def load_data():
     data = []
     labels = []
     for i in range(NUM_CATEGORIES):
          path =
     os.path.join(r"D:\pycharmproj\TensorflowTrafficSignRecogniti
     on\gtsrb\Train", str(i))
          images = os.listdir(path)
          for j in images:
               try:
                    image = cv2.imread(os.path.join(path, j))
                    image_from_array = Image.fromarray(image,
     'RGB')
                    resized_image =
     image_from_array.resize((IMG_HEIGHT, IMG_WIDTH))
                    data.append(np.array(resized_image))
                    labels.append(i)
               except AttributeError:
                    print("Помилка завантаження зображення!")
     images_data = (data, labels)
     return images_data
```

Ця функція отримує посилання на директорію датасету і виконує наступні дії:

– форматує зображення, змінюючи їх розмір до 30x30 пікселів;

– змінює розширення зображень для подальшого використання;

– повертає класи відповідних зображень, які будуть використовуватись для подальшого поділу на категорії.

Лістинг 2.4 – Архітектура методу обробки зображень описана у функції get_model().

```
def get_model():
     # ініціалізація моделі
     model = Sequential()
     ch_dimension = -1
     model.add(Conv2D(8, (5, 5), padding="same",
     input_shape=inputShape))
     model.add(Activation("relu"))
     model.add(BatchNormalization(axis=ch_dimension))
     model.add(MaxPooling2D(pool_size=(2, 2)))
     model.add(Flatten())
     model.add(Dense(512))
     model.add(Activation("relu"))
     model.add(BatchNormalization())
     model.add(Dropout(0.7))
     model.add(Dense(NUM_CATEGORIES))
     model.add(Activation("softmax"))
     model.compile(loss="categorical_crossentropy",
     optimizer='adam', metrics=["accuracy"])

     return model
```

Для зменшення обсягу отримуваних даних та підкреслення унікальностей кожного дорожнього знаку, було прийнято рішення використовувати один конволюційний шар. У цьому шарі було використано 512 нейронів для обробки зображень, оскільки це вважається оптимальним вибором.

Лістинг 2.5 – При успішному навчанні виконується наступний код:

```
if len(sys.argv) == 1:
          folder_name = os.path.dirname(sys.argv[0])
          print(os.path.join(folder_name, "model1.h5"))
          model.save(os.path.join(folder_name, "model1.h5"))
          print(f"Model saved to {folder_name}.")
```

```
     plt.figure(0)
     plt.plot(history.history['loss'], label='training
loss')
     plt.title("Втрати")
     plt.xlabel("епохи")
     plt.ylabel("втрати")
     plt.legend()
     plt.show()

     plt.figure(1)
     plt.plot(history.history['accuracy'], label='training
accuracy ')
     plt.title("Точність")
     plt.xlabel("епохи")
     plt.ylabel("точність")
     plt.legend()
     plt.show()
```

Цей програмний код здійснює навчання моделі та виводить результати, такі як відсоток точності та відсоток втрат. Файл "LearnModel.py", наведений у лістингу А3 у додатку А, виконує наступні дії: запускає навчання моделі, підставляє змінні імпортовані з файлу RecognizeItem.py, зберігає успішні етапи навчання і модель та підводить підсумки роботи навчання.

Наступний програмний код виводить графіки, на яких показана точність і втрати моделі відповідно до епох навчання. Зображення графіків можна побачити на рисунках 2.12 та 2.13.

Лістинг 2.6 – Вивід графіків точностей та втрат.

```
pyplot.figure(0)
     pyplot.plot(history.history['loss'], label='training loss')
     pyplot.plot(history.history['val_loss'], label='validation
     loss')
     pyplot.title("Втрати")
     pyplot.xlabel("епохи")
     pyplot.ylabel("втрати")
     pyplot.legend()
     pyplot.show()
     pyplot.figure(1)
     pyplot.plot(history.history['accuracy'], label='training
     accuracy ')
     pyplot.plot(history.history['val_accuracy'],
     label='validation accuracy')
     pyplot.title("Точність")
```

```
pyplot.xlabel("епохи")
pyplot.ylabel("точність")
pyplot.legend()
pyplot.show()
```

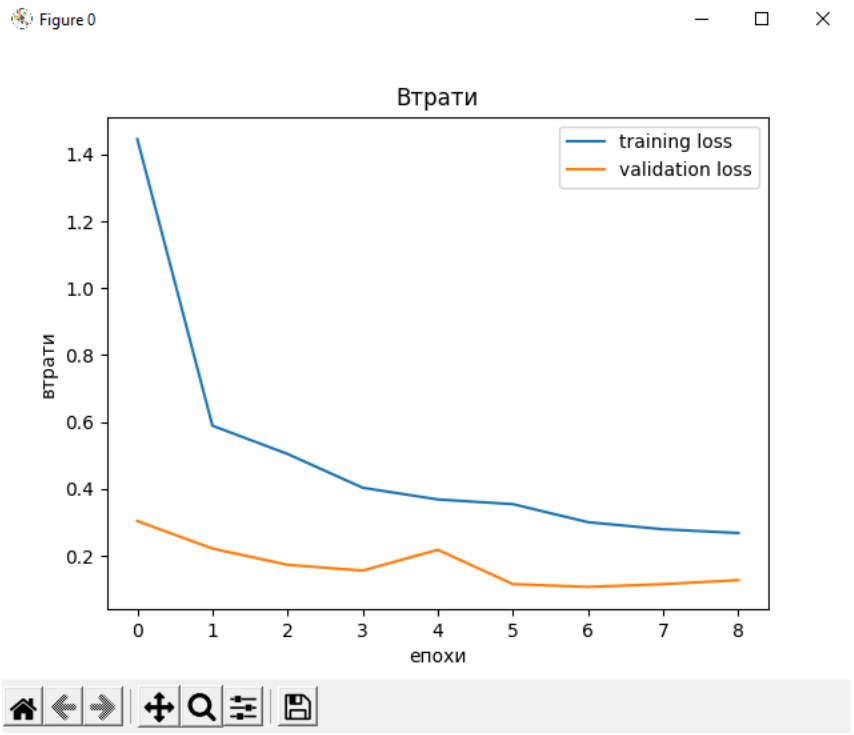

Рисунок 2.12 – Втрати під час навчання моделі

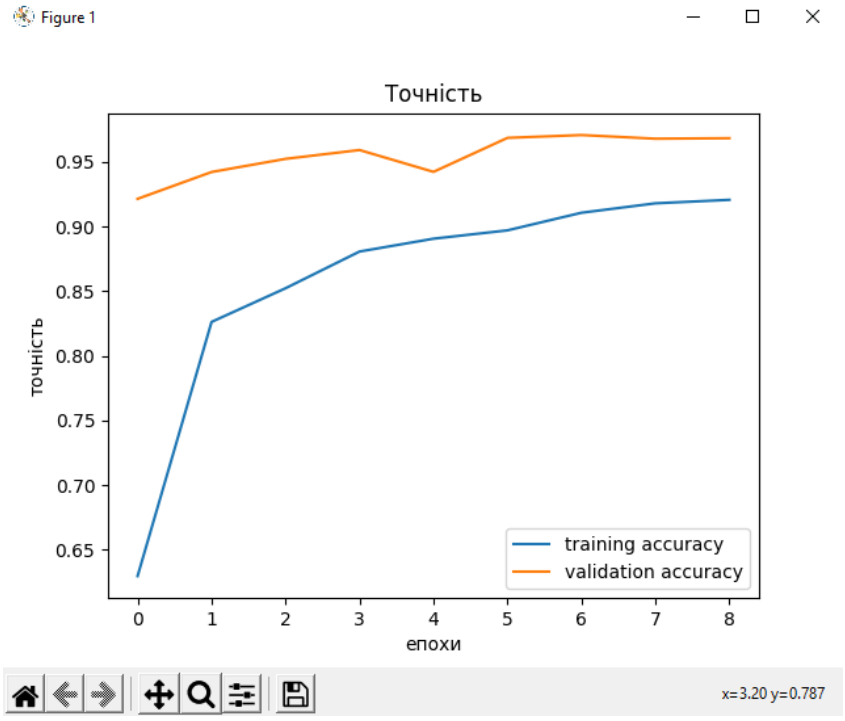

Рисунок 2.13 – Точність під час навчання моделі

Щоб уникнути збоїв навчання або зниження точності моделі, навчання припиняється та зберігається, якщо досягнуті високі показники навчання, а відсоток валідації не змінився від попередньої епохи.

Файл "RecognizeSigns.py", який наведений у лістингу А1 у додатку А, відповідає за завантаження збереженої моделі та ініціалізацію користувацького інтерфейсу.

Програмний код, відповідальний за ініціалізацію користувацького інтерфейсу, наведено у лістингу А4 у додатку А.

У файлі "RecognizeSigns.py" міститься словник, який містить назви всіх дорожніх знаків, що програма може розпізнати. Значення в цьому словнику встановлюються відповідно до класів, до яких відносяться розпізнані знаки. За це відповідає функція def_classify:

Лістинг 2.7 – Співвідношення зображення з назвою знаку

```
def classify(file_path):
        global label_packed
        image = Image.open(file_path)
        image = image.resize((30, 30))
        image = np.expand_dims(image, axis=0)
        image = np.array(image)
        print(image.shape)
        pred = model.predict_classes([image])[0]
        sign = classes[pred + 1]
        print(sign)
        label.configure(foreground='#364196', text=sign)
```

Функція upload_image виконує завантаження зображення, обраного користувачем, та форматує його до глибини кольору 24.

Лістинг 2.8 – Завантаження зображення

```
def upload_image():
        try:
            file_path = filedialog.askopenfilename()
            uploaded = Image.open(file_path).convert('RGB')

    uploaded.save(r'D:\pycharmproj\TensorflowTrafficSignRecogni
    tion\temp\temp_converted_image.png')
```

```
            uploaded.thumbnail(((top.winfo_width() / 2.25),
(top.winfo_height() / 2.25)))
            im = ImageTk.PhotoImage(uploaded)
            sign_image.configure(image=im)
            sign_image.image = im
            label.configure(text='')

classify(r'D:\pycharmproj\TensorflowTrafficSignRecognition\
temp\temp_converted_image.png')
        except:
            pass
```

Це робиться, оскільки бібліотека Tensorflow працює з певними властивостями зображень, включаючи глибину кольору.

2.6 Тестування програмного забезпечення

Тестування є невід'ємною складовою процесу розробки програмного забезпечення, оскільки воно дозволяє оцінити коректність роботи програми та виявити недоліки.

Першим кроком є додавання або оновлення нейронної моделі. Для цього необхідно натиснути кнопку «Оновити модель» яка зображена на рисунку 2.14.

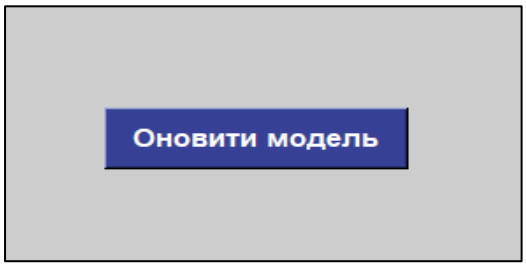

Рисунок 2.14 – Кнопка оновлення моделі

Під час навчання моделі програма буде тимчасово недоступною і не буде функціонувати. Процес навчання може зайняти кілька хвилин. Однак, після завершення навчання модель буде збережена у системних файлах і

може бути використана без необхідності повторного навчання. Цей процес зображений на рисунках 2.15-2.17.

```
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 8)         608
_________________________________________________________________
activation (Activation)      (None, 30, 30, 8)         0
_________________________________________________________________
batch_normalization (BatchNo (None, 30, 30, 8)         32
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 8)         0
_________________________________________________________________
flatten (Flatten)            (None, 1800)              0
_________________________________________________________________
dense (Dense)                (None, 512)               922112
_________________________________________________________________
activation_1 (Activation)    (None, 512)               0
_________________________________________________________________
batch_normalization_1 (Batch (None, 512)               2048
_________________________________________________________________
dropout (Dropout)            (None, 512)               0
_________________________________________________________________
dense_1 (Dense)              (None, 43)                22059
_________________________________________________________________
activation_2 (Activation)    (None, 43)                0
=================================================================
Total params: 946,859
Trainable params: 945,819
Non-trainable params: 1,040
_________________________________________________________________
```

Рисунок 2.15- Загальний об’єм параметрів для навчання

```
728/728 [==============================] - 12s 15ms/step - loss: 1.5024 - accuracy: 0.6214
Epoch 2/15
728/728 [==============================] - 11s 15ms/step - loss: 0.4908 - accuracy: 0.8578
Epoch 3/15
728/728 [==============================] - 11s 15ms/step - loss: 0.3373 - accuracy: 0.9020
Epoch 4/15
728/728 [==============================] - 11s 15ms/step - loss: 0.2945 - accuracy: 0.9140
Epoch 5/15
728/728 [==============================] - 13s 18ms/step - loss: 0.2506 - accuracy: 0.9259
Epoch 6/15
144/728 [====>.........................] - ETA: 9s - loss: 0.2015 - accuracy: 0.9345
```

Рисунок 2.16 - Процес навчання нейронної мережі

```
Epoch 00010: val_loss did not improve from 0.12224
Epoch 00010: early stopping
Model saved to D:/pycharmproj/TensorflowTrafficSignRecognition.
Train accuracy : 0.967, Test accuracy: 0.958
```

Рисунок 2.17 - Результат навчання нейронної мережі

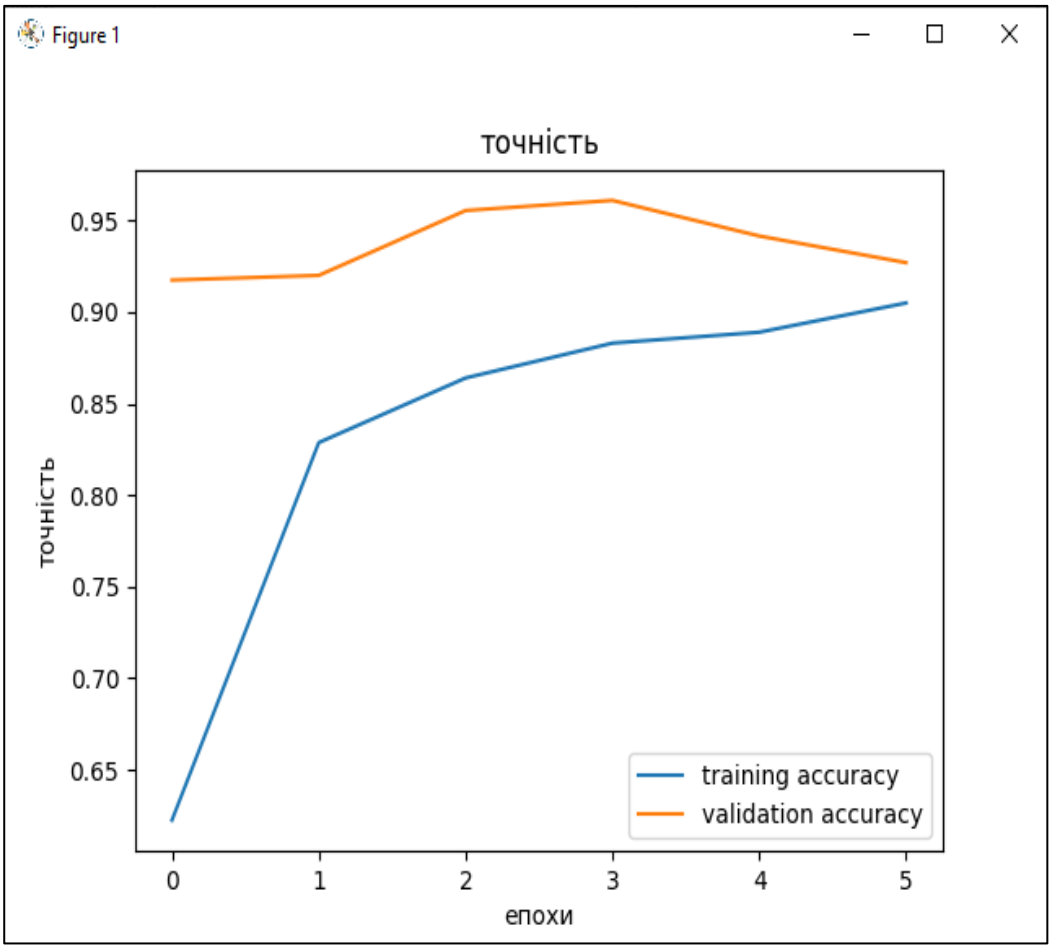

Рисунок 2.18 – Графік точності навчання моделі відносно до епох

Після завершення навчання ви можете продовжити роботу з програмою. Для завантаження зображення, яке потрібно розпізнати, вам потрібно натиснути кнопку «Завантажити знак» і вибрати дорожній знак з вашого пристрою. Цю дію зображено на рисунку 2.19.

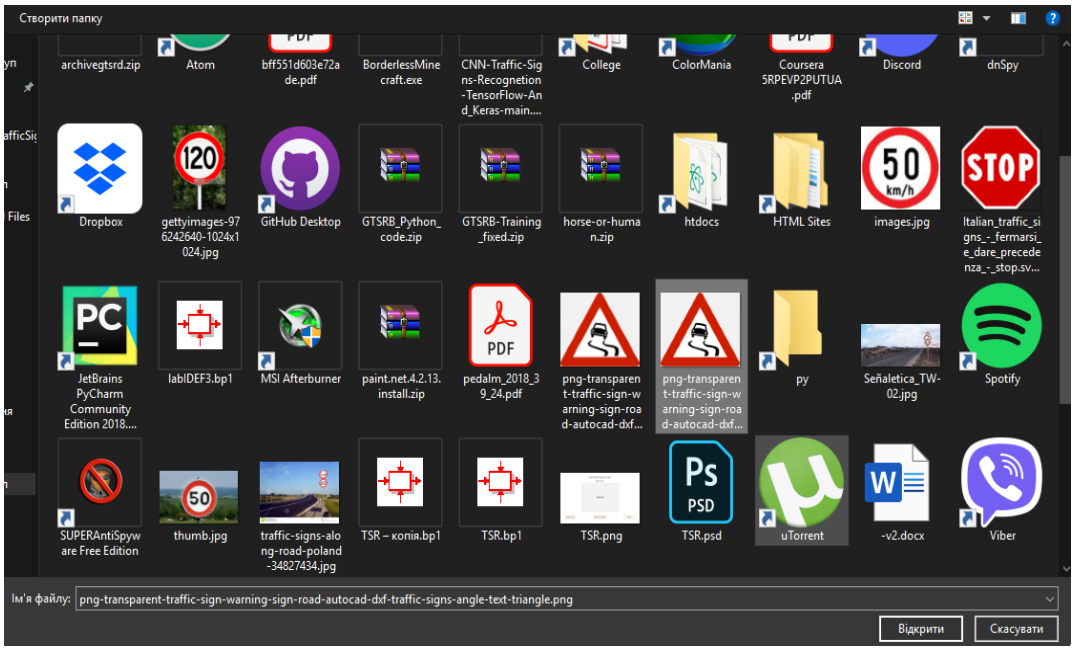

Рисунок 2.19 – Вибір зображення за допомогою файлового провідника

Як тільки знак буде завантажено, програма надасть результат його розпізнавання. Цей результат буде на екрані, зображено на рисунку 2.20.

Рисунок 2.20 – Результат роботи програми

Для перевірки точності нейронної мережі ми проведемо додаткові тести з розпізнавання знаків. Вони відображають, які знаки були правильно

розпізнані, а також вказують на ті, що були помилково ідентифіковані. Це дозволить нам оцінити точність роботи нейронної мережі при розпізнаванні дорожніх знаків.

Результати цього тестування можна побачити на рисунках 2.21-2.22.

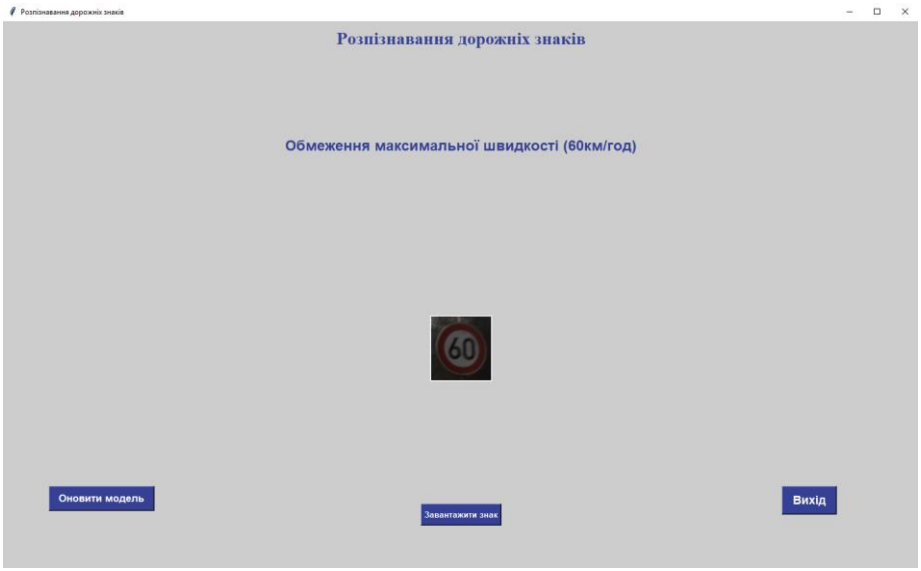

Рисунок 2.21– Додаткове тестування роботи програми

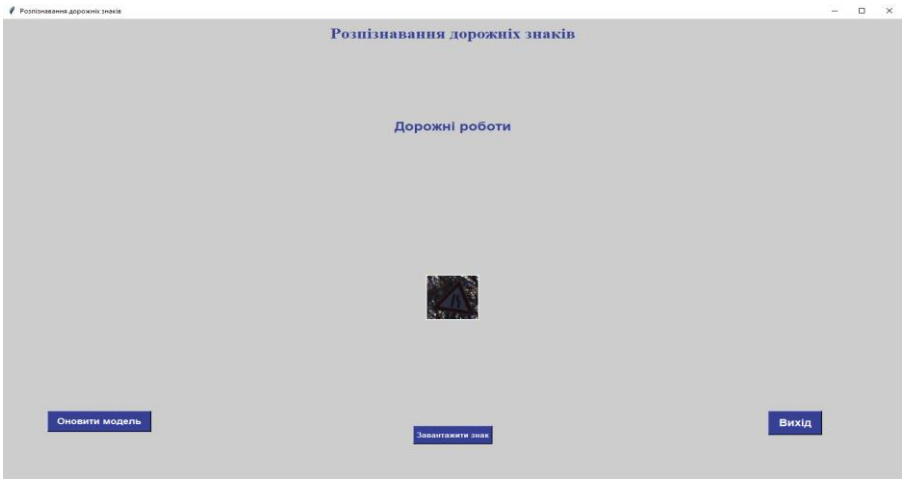

Рисунок 2.22 – Помилкове розпізнавання дорожнього знаку

Точність нейронних мереж залежить від багатьох факторів, включаючи якість навчального набору даних, розмір датасету, розподіл зображень у наборі даних, архітектуру моделі, гіперпараметри та інші фактори.

Для підвищення точності розпізнавання знаків, як ви вказали, важливо мати розношерстий датасет з великим обсягом зображень, які включають різні умови освітлення, ракурси та вигляд знаків. Також, застосування конволюційних перетворень для обробки зображень може допомогти впоратися з різними вхідними даними та виділити унікальні особливості на зображеннях, що сприяє поліпшенню точності.

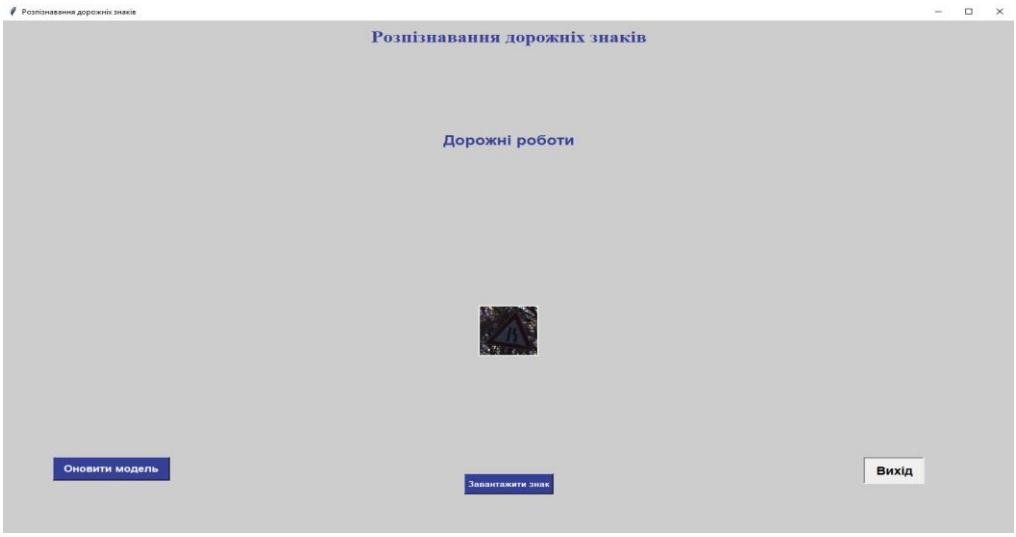

Рисунок 2.23 – Натискання на кнопку «Вихід» та закриття програми

Для завершення роботи програми, необхідно натиснути кнопку «Вихід», що зображено а рисунку 2.23.

2.7 Висновки до другого розділу

У другому розділі кваліфікаційної роботи було спроєктовано структуру системи, користувацький інтерфейс та алгоритм роботи системи. Також було реалізовано користувацький інтерфейс та проведено перевірку на точність розпізнавання зображень. Проведено тестування програмного забезпечення, описано процес роботи та взаємоємодії з програмою. Розібрано точність та можливість хибних результатів.

РОЗДІЛ 3. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

3.1 Безпека життєдіяльності при водінні авто

Безпека життєдіяльності є однією з найважливіших аспектів під час водіння автомобіля та використання технологій. Особиста безпека, безпека пасажирів та інших учасників дорожнього руху залежать від уважного ставлення до правил та відповідальної поведінки за кермом, а також від свідомого використання технологій.

Під час водіння автомобіля основні правила безпеки полягають у дотриманні швидкісного режиму, правил перестроєння, використанні поворотників, а також дотриманні правил дистанції та безпеки на перехрестях. Водій повинен бути уважним, уникати відволікань, таких як використання мобільного телефону або розмова з пасажирами, що можуть вплинути на концентрацію та реакцію. Регулярний огляд технічного стану автомобіля, зокрема гальм, світлової сигналізації та гуми, також є важливим аспектом безпеки на дорозі.

У сучасному світі технології можуть бути корисними, але також можуть становити загрозу безпеці, якщо використовуються некоректно. Використання мобільних пристроїв під час водіння є небезпечним і може призвести до відволікання від дороги. Тому водії повинні уникати використання телефону або використовувати спеціальні пристрої, які дозволяють їм залишатися підключеними, не відводячи уваги від керування автомобілем.

Також варто зазначити про розумне використання автомобільних систем навігації та розваг. Вони можуть бути корисними і допомагати водію, але їх використання повинно здійснюватися безпечно. Водії повинні програмувати маршрути перед поїздкою або зупинятися в безпечному місці, щоб не відволікатися від керування автомобілем.

Загальні принципи безпеки життєдіяльності при водінні автомобіля і використанні технологій полягають у свідомому ставленні до власної безпеки та безпеки оточуючих, у дотриманні правил дорожнього руху та уникає відволікань. Безпечне водіння та розумне використання технологій сприяють зменшенню ризиків та забезпечують безпеку на дорозі [12].

Крім правил безпеки під час водіння автомобіля і використання технологій, також важливо пам'ятати про основні принципи безпеки:

1. Навчання і підготовка: Перед тим як сісти за кермо, важливо пройти належне навчання та отримати відповідні права. Знання правил дорожнього руху, навички керування автомобілем та поведінка в небезпечних ситуаціях допоможуть зменшити ризик аварій.

2. Використання ременів безпеки: Ремені безпеки є одним з найважливіших засобів захисту під час водіння. Вони допомагають утримати водія та пасажирів на місці під час зіткнення або гострого гальмування.

3. Регулярне обслуговування автомобіля: Важливо регулярно перевіряти стан автомобіля та проводити заплановане технічне обслуговування. Це включає перевірку гальм, світлової сигналізації, шин та інших систем, що впливають на безпеку.

4. Використання технологій з розумінням: Технології, такі як GPS-навігатори, допоміжні системи керування, розпізнавання знаків дорожнього руху та інші, можуть покращити безпеку на дорозі. Однак, важливо користуватися ними з розумінням, не дозволяючи їм стати джерелом відволікання або залежності.

5. Відпочинок і відсутність втоми: Водій повинен бути відпочившим та бути у здатності до керування автомобілем. Довгі періоди безперервного водіння можуть призвести до втоми, що погіршує реакцію та здатність приймати рішення. Регулярні перерви під час довгих поїздок допоможуть запобігти втомі [12].

Загальна мета безпеки життєдіяльності при водінні автомобіля і використанні технологій полягає у зниженні ризиків та створенні безпечного середовища для всіх учасників дорожнього руху. Свідоме та відповідальне ставлення до безпеки є ключем до успішного та безпечного використання автомобілів та технологій на дорозі.

3.2 Шляхи підвищення ефективності охорони праці

У Законі України «Про охорону праці» задекларовані основні принципи державної політики в галузі охорони праці, використання економічних методів управління охороною праці, проведення політики пільгового оподаткування, комплексне розв’язання завдань охорони праці на основі національних програм з цих питань, досягнень у галузі науки і техніки та охорони навколишнього середовища тощо. Перехід суспільства до широкого використання ринкових відносин, виникнення різноманітних форм власності потребують розроблення нових підходів до побудови сучасної моделі управління охороною й безпекою праці на національному, регіональному й виробничому рівнях.

У суспільстві із соціально орієнтованою економікою охорона праці має бути одним з найважливіших завдань соціально-економічної політики як держави, так і кожного підприємства та організації. Охорона праці – проблема складна і багатогранна.

Проте нинішній рівень науково-технічного прогресу та соціально-економічні орієнтири розвитку сучасного суспільства не спроможні створити сприятливі умови для забезпечення добробуту людини, збереження її здоров’я. Особливо гостро ця проблема постає на промислових підприємствах, зокрема машинобудівних, гірничо-видобувних, ливарних виробництвах, де зберігається переважно застаріла матеріально-технічна база виробництва при незадовільних обсягах фінансування заходів з охорони праці. Усе це

призводить до високого рівня травматизму та професійної захворюваності, особливо в гірничо-видобувній галузі і, як наслідок, до збільшення видатків підприємства держави та Фонду соціального страхування на виплати й компенсації потерпілим. Тому вкрай необхідним є вдосконалення системи охорони праці як важливого фактора підвищення ефективності виробництва на підставі детального дослідження економічних і соціальних її аспектів[13].

Охорона праці є важливим аспектом безпеки під час водіння автомобіля. Існує багато шляхів підвищення ефективності охорони праці, та зокрема при водінні авто.

Дотримання правил дорожнього руху: ретельне дотримання правил дорожнього руху є основою безпечного водіння. Це включає дотримання швидкісного режиму, зупинку на світлофорах, повороти за сигналами і правильний використання смуг руху.

Уникання відволікань: використання мобільних телефонів, їжа, макіяж, розмови з пасажирами та інші види відволікань можуть суттєво знизити увагу водія. Важливо уникати таких дій під час керування автомобілем, щоб забезпечити максимальну концентрацію на дорозі.

Підтримка нормального стану здоров'я: недостатній сон і погане фізичне самопочуття можуть призвести до зниження реакційної швидкості та уваги під час водіння. Водії повинні регулярно відпочивати перед довгими поїздками і уникати керування автомобілем під впливом алкоголю або наркотиків.

Використання систем безпеки: сучасні автомобілі оснащені різними системами безпеки, такими як системи контролю стійкості, системи автоматичного екстреного гальмування і системи попередження про зіткнення. Використання цих систем може значно знизити ризик нещасних випадків.

Стеження за дорожніми умовами: перед поїздкою варто ознайомитися з прогнозом погоди і дорожніми умовами. У разі поганої погоди або складних

дорожніх умов слід бути особливо уважним і приймати відповідні заходи для збереження безпеки.

Важливо пам'ятати, що безпека на дорозі залежить від водія і його усвідомлення відповідальності за безпеку себе та інших учасників руху. Заходи, перераховані вище, можуть допомогти покращити ефективність охорони праці під час водіння автомобіля.

3.3 Висновок до третього розділу

В результаті виконання даного розділу було проаналізовано наступні питання з безпеки життєдіяльності та основ охорони праці: вплив технологій на природу людини та безпека життєдіяльності під час водіння авто. Розібрано основні принципи безпеки та важливість їх дотримання.

Технології можуть бути корисними, але також можуть становити загрозу безпеці. Використання мобільних пристроїв під час водіння є небезпечним і може призвести до відволікання. Тому важливо уникати використання телефону під час керування автомобілем або використовувати спеціальні пристрої, які дозволяють залишатися підключеним без відволікання.

Крім того, розумне використання автомобільних систем навігації та розваг є важливим. Вони можуть бути корисними, але їх використання повинно здійснюватися безпечно, з урахуванням безпеки на дорозі.

ВИСНОВКИ

Метою даної кваліфікаційної роботи було дослідження комп'ютерного зору з метою подальшої розробки програми для розпізнавання дорожніх знаків. В процесі виконання кваліфікаційної роботи були проведені аналіз подібних систем, дослідження технології розпізнавання дорожніх знаків та вивчення основних аспектів, необхідних для реалізації кваліфікаційної роботи. Також було розглянуто питання вибору засобів реалізації, включаючи мову програмування Python, функціональні можливості модуля Tensorflow та конволюційні нейронні мережі.

Незважаючи на високу точність розпізнавання дорожніх знаків, все ще існують помилкові результати, що свідчить про необхідність вдосконалення процесу навчання нейронної мережі.

Також важливо зазначити, що систему можна покращити шляхом поліпшення графічного користувацького інтерфейсу, що зробить його більш зручним та інтуїтивно зрозумілим для користувачів. Крім того, можна реалізувати можливість розпізнавання дорожніх знаків прямо з відеопотоку під час руху автомобіля. Це дозволить системі надавати розпізнавання в реальному часі, що є особливо корисним у ситуаціях, коли водій активно взаємодіє з дорожнім середовищем. Ці покращення сприятимуть зручності та ефективності системи розпізнавання дорожніх знаків.

ПЕРЕЛІК ДЖЕРЕЛ

1. У світі щороку в ДТП гинуть 1,25 млн людей – ООН. Громадське телебачення - *Останні новини дня, всі надзвичайні новини в Україні.* URL: https://hromadske.ua/posts/u-sviti-shoroku-v-dtp-ginut-125-mln-lyudej-oon (дата звернення 25.02.2023).

2. Autonomous Vehicle Technology and Autonomous Car Technology | V2I/V2V *Communication. Autotalks* URL: https://www.auto-talks.com/architected-for-autonomous/ (дата звернення: 26.02.2023).

3. Toyota Safety Sense TSS. *WestBury Toyota:* URL: https://westburytoyota.com/safety-sense (дата звернення: 26.02.2023).

4. Toyota Safety Sense: Preventive Safety Package-equipped Vehicles Top 10 Million Units Globally. *Toyota Newsroom:* URL: https://pressroom.toyota.com/toyota-safety-sense-preventive-safety-package-equipped-vehicles-top-10-million-units-globally/ (дата звернення: 03.03.2023).

5. One step closer to a world without accidents. Lexus safety. *Lexus:* URL: https://www.lexus.com/safety (дата звернення: 05.04.2023).

6. Python (programming language). *Lexus:* URL: https://en.wikipedia.org/wiki/Python_(programming_language) (дата звернення: 12.05.2023).

7. TensorFlow. *Wikipedia:* URL: https://en.wikipedia.org/wiki/TensorFlow (дата звернення: 13.05.2023).

8. Комп'ютерний зір. *Wikipedia:* URL: https://uk.wikipedia.org/wiki/Комп%27ютерний_зір (дата звернення: 13.05.2023).

9. Real-Time Object Detection Using TensorFlow. *Great Learning:* URL: https://www.mygreatlearning.com/blog/object-detection-using-tensorflow/ (дата звернення: 13.05.2023).

10. Python - GUI Programming (Tkinter). *Tutorialspoint:* URL: https://www.tutorialspoint.com/python/python_gui_programming.htm (дата звернення: 15.05.2023).

11. Download PyCharm. *JetBrains:* URL: https://www.jetbrains.com /pycharm/download/ (дата звернення 17.05.2023).

12. Global Autonomous Cars Market (2020 to 2030) - Opportunities and Strategies with COVID-19 Growth and Change. *Intrado:* URL: https://www.globenewswire.com/news-release/2020/12/28/2150806/0/en/Global-Autonomous-Cars-Market-2020-to-2030-Opportunities-and-Strategies-with-COVID-19-Growth-and-Change.html (дата звернення: 03.05.2023).

13. Перспективи розвитку автомобілів. Novaton - інтернет магазин запчастин. URL: https://novaton.ua/news/374 (дата звернення: 03.05.2023).

14. Навчально-науковий центр перепідготовки та заочного навчання ННЦПЗН НУ "Чернігівська політехніка". URL: https://cpo.stu.cn.ua/ Oksana/posibnik/120.html (дата звернення: 03.05.2023).

15. Гнідий Д. О. Метод прискорення статистичного моделювання мов програмування на основі використання фреймворку Tensorflow: дис. ступінь магістра. Київ, 2018, 85с.

16. Mark Summerfield Programming in Python 3: A Complete Introduction to the Python Language (Developer's Library) 2nd Edition. Boston, 2009, 599p.

17. Mark Lutz Programming Python: Powerful Object-Oriented Programming Fourth Edition. Sebastopol, Canada, 2010, 1557p

18. Introduction to TensorFlow. *Tensorflow:* URL: https:// www.tensorflow.org/learn (дата звернення: 11.05.2023)

19. Tensorflow Documentation. *Documentation:* URL: https://docs.wandb.ai/guides/integrations/tensorflow (дата звернення: 11.05.2023)

20. Python 3.9.5 documentation. *Python Documentation*: URL: https://docs.python.org/3/ (дата звернення: 11.05.2023)

21. GitHub - tensorflow/docs: *TensorFlow documentation. GitHub.* URL: https://github.com/tensorflow/docs (дата звернення: 15.04.2023).

22. Géron A. Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems. O'Reilly Media, 2017. 574p.

23. Müller A. C., Guido S. Introduction to Machine Learning with Python: A Guide for Data Scientists. O'Reilly Media, Incorporated, 2016. 400 p.

24. Idris I. NumPy Beginner's Guide (Second Edition). Packt Publishing, Limited, 2013. 348p.

25. Computer Science Computer Science Academy. Python for Data Analysis: A Complete Guide for Beginners, Including Python Statistics and Big Data Analysis. Independently Published, 2020. 132 p.

26. Islam Q. N. Mastering Pycharm. Packt Publishing, Limited, 2015. 232 p.

27. Савчишин Я.С. Сучасні технології комп’ютерного розпізнавання об’єктів. Збірник наукових тез: за матеріалами студентських наукових читань. Тернопіль: Навчально-практична майстерня редакційно-видавничих технологій Галицького коледжу імені В’ячеслава Чорновола, 2021р., С. 192-197.

ДОДАТКИ

Додаток А

Програмний код RecognizeSigns

```
import os
import sys
import tkinter as tk
from tkinter import filedialog
from tkinter import *
from PIL import ImageTk, Image
import pickle

from tensorflow.keras.models import load_model
from sklearn.metrics import accuracy_score

import numpy as np

with open('test.pkl', 'rb') as f:
    x_test, y_test = pickle.load(f)
y_test = np.argmax(y_test, axis=1)

model = load_model("model2.h5")
prediction = np.argmax(model.predict(x_test), axis=1)
accuracy = float(accuracy_score(y_test, prediction.round()))
print('Точність моделі при тестувані становить
{:.2f}'.format(accuracy * 100), "%")

# словник назв
classes = {1: 'Обмеження максимальної швидкості(20км/год)',
           2: 'Обмеження максимальної швидкості (30км/год)',
           3: 'Обмеження максимальної швидкості (50км/год)',
           4: 'Обмеження максимальної швидкості (60км/год)',
           5: 'Обмеження максимальної швидкості (70км/год)',
           6: 'Обмеження максимальної швидкості (80км/год)',
           7: 'Кінець обмеження максимальної швидкості
(80км/год)',
           8: 'Обмеження максимальної швидкості (100км/год)',
           9: 'Обмеження максимальної швидкості (120км/год)',
           10: 'Обгін заборонено',
           11: 'Обгін вантажним автомобілям заборонено',
           12: 'Перехрещення з другорядною дорогою',
           13: 'Головна дорога',
           14: 'Дати дорогу',
           15: 'Проїзд без зупинки заборонено',
           16: 'Рух заборонено',
           17: 'Рух вантажних автомобілів заборонено',
           18: 'В''їзд заборонено',
           19: 'Інша небезпека (аварійно-небезпечна ділянка)',
           20: 'Небезпечний поворот ліворуч',
           21: 'Небезпечний поворот праворуч',
```

```
            22: 'Декілька поворотів',
            23: 'Нерівна дорога',
            24: 'Слизька дорога',
            25: 'Звуження дороги',
            26: 'Дорожні роботи',
            27: 'Світлофорне регулювання',
            28: 'Пішоходний перехід',
            29: 'Діти',
            30: 'Виїзд велосипедистів',
            31: 'Сніг',
            32: 'Дикі тварини',
            33: 'Кінець усіх заборон і обмежень',
            34: 'Рух праворуч',
            35: 'Рух ліворуч',
            36: 'Рух прямо',
            37: 'Рух прямо або праворуч',
            38: 'Рух прямо або ліворуч',
            39: 'Рух праворуч',
            40: 'Об’їзд перешкоди з лівого боку',
            41: 'Круговий рух',
            42: 'Кінець заборони обгону',
            43: 'Кінець заборони обгону вантажним автомобілям'}

# GUI
top = tk.Tk()
top.geometry('1600x900')
top.title('Розпізнавання дорожніх знаків')
top.configure(background='#CDCDCD')

label = Label(top, background='#CDCDCD', font=('arial', 18,
'bold'))
sign_image = Label(top)

def classify(file_path):
    global label_packed
    image = Image.open(file_path)
    image = image.resize((30, 30))
    image = np.expand_dims(image, axis=0)
    image = np.array(image)
    print(image.shape)
    pred = model.predict_classes([image])[0]
    sign = classes[pred + 1]
    print(sign)
    label.configure(foreground='#364196', text=sign)

def upload_image():
    try:
        file_path = filedialog.askopenfilename()
        uploaded = Image.open(file_path).convert('RGB')

uploaded.save(r'D:\pycharmproj\TensorflowTrafficSignRecognition\
temp\temp_converted_image.png')
```

```
        uploaded.thumbnail(((top.winfo_width() / 2.25),
(top.winfo_height() / 2.25)))
        im = ImageTk.PhotoImage(uploaded)
        sign_image.configure(image=im)
        sign_image.image = im
        label.configure(text='')

classify(r'D:\pycharmproj\TensorflowTrafficSignRecognition\temp\
temp_converted_image.png')
    except:
        pass
def initialize_gui():
    learn_b = Button(top, text="Оновити модель", command=lambda:
os.system('python LearnModel.py'), padx=10, pady=5)
    learn_b.configure(background='#364196', foreground='white',
font=('arial', 13, 'bold'))
    learn_b.place(relx=0.05, rely=0.85)

    exit_button = Button(top, text="Вихід", command=top.destroy,
font=('arial', 14, 'bold'), padx=10, pady=5)
    exit_button.configure(background='#364196',
foreground='white')
    exit_button.place(relx=0.85, rely=0.85)
    upload = Button(top, text="Завантажити знак",
command=upload_image, pady=5)
    upload.configure(background='#364196', foreground='white',
font=('arial', 10, 'bold'))
    upload.pack(side=BOTTOM, pady=70)
    sign_image.pack(side=BOTTOM, expand=True)
    label.pack(side=BOTTOM, expand=True)
    heading = Label(top, text="Розпізнавання дорожніх знаків",
pady=10, font=('Times New Roman', 22, 'bold'))
    heading.configure(background='#CDCDCD',
foreground='#364196')
    heading.pack()
    top.mainloop()
initialize_gui()
```

Додаток Б

Програмний код RecognizeItem

```
import time
import cv2
import os
import sys
from PIL import Image
import numpy as np
from matplotlib import pyplot as plt
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.layers import Activation
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dropout
from tensorflow.keras.layers import Dense
import tensorflow.keras as tfk
from tensorflow.python.keras.callbacks import TensorBoard
EPOCHS = 15
IMG_WIDTH = 30
IMG_HEIGHT = 30
NUM_CATEGORIES = 43
TEST_SIZE = 0.4
batch_size = 16
pool_size = (2, 2)
inputShape = (IMG_WIDTH, IMG_HEIGHT, 3)
Name = "trafficsSignsModel-{}".format(int(time.time()))


def main():
# Перевірка аргументів командної строки
if len(sys.argv) not in [1, 3]:
     sys.exit("Usage: python RecognizeItem.py
data_directory [model.h5]")

images, labels = load_data(os.path.dirname(sys.argv[0]))

labels = tfk.utils.to_categorical(labels)
x_train, x_test, y_train, y_test = train_test_split(
     np.array(images), np.array(labels),
test_size=TEST_SIZE)

model = get_model()
model.summary()
tensorboard = TensorBoard(log_dir='logs/{}'.format(Name))

history = model.fit(x_train, y_train, epochs=EPOCHS,
callbacks=[tensorboard])

# Оцінка роботи нейронної мережі
_, acc_train = model.evaluate(x_train, y_train, verbose=2)
_, acc_test = model.evaluate(x_test, y_test, verbose=2)
print('Точність навчання : %.3f, Точність тестування: %.3f'
% (acc_train, acc_test))

# Зберігання моделі
if len(sys.argv) == 1:
     folder_name = os.path.dirname(sys.argv[0])
     print(os.path.join(folder_name, "model1.h5"))
     model.save(os.path.join(folder_name, "model1.h5"))
     print(f"Model saved to {folder_name}.")
```

```
     plt.figure(0)
     plt.plot(history.history['loss'], label='training
loss')
     plt.title("Loss")
     plt.xlabel("Epochs")
     plt.ylabel("loss")
     plt.legend()
     plt.show()

     plt.figure(1)
     plt.plot(history.history['accuracy'], label='training
accuracy ')
     plt.title("accuracy")
     plt.xlabel("Epochs")
     plt.ylabel("accu")
     plt.legend()
     plt.show()


def load_data(data_dir):
data = []
labels = []
for i in range(NUM_CATEGORIES):
     path =
os.path.join(r"D:\pycharmproj\TensorflowTrafficSignRecogniti
on\gtsrb\Train", str(i))
     images = os.listdir(path)
     for j in images:
          try:
               image = cv2.imread(os.path.join(path, j))
               image_from_array = Image.fromarray(image,
'RGB')
               resized_image =
image_from_array.resize((IMG_HEIGHT, IMG_WIDTH))
               data.append(np.array(resized_image))
               labels.append(i)
          except AttributeError:
               print("Помилка завантаження зображення!")

images_data = (data, labels)
return images_data


def get_model():
# ініціалізація моделі
model = Sequential()
ch_dimension = -1
model.add(Conv2D(8, (5, 5), padding="same",
input_shape=inputShape))
model.add(Activation("relu"))
model.add(BatchNormalization(axis=ch_dimension))
model.add(MaxPooling2D(pool_size=(2, 2)))
```

```
model.add(Flatten())
model.add(Dense(512))
model.add(Activation("relu"))
model.add(BatchNormalization())
model.add(Dropout(0.7))
model.add(Dense(NUM_CATEGORIES))
model.add(Activation("softmax"))

# компілювання моделі
model.compile(loss="categorical_crossentropy",
optimizer='adam', metrics=["accuracy"])

return model


if __name__ == "__main__":
main()
```

Додаток В

Програмний код LearnModel

```
import pickle
from matplotlib import pyplot
from sklearn.model_selection import train_test_split
from tensorflow import keras as tfk
import numpy as np
from RecognizeItem import load_data
from RecognizeItem import get_model
from RecognizeItem import EPOCHS,batch_size
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras.callbacks import ModelCheckpoint
from tensorflow.keras.callbacks import TensorBoard
import pandas as pd
import time, os, sys
from imblearn.over_sampling import SMOTE

Name = "trafficsSignsModel-{}".format(int(time.time()))
checkPointModel = "Model-{}.h5".format(int(time.time()))
modelName = "model{}.h5".format(int(time.time()))
model = get_model()
model.summary()
images, labels = load_data(os.path.dirname(sys.argv[0]))
labels = tfk.utils.to_categorical(labels)
x_train, x_test, y_train, y_test = train_test_split(
   np.array(images), np.array(labels),
test_size=0.1,random_state=1)
with open('test.pkl', 'wb') as f:
   pickle.dump([x_test, y_test], f)


filepath
="Saved_models_checkpoints/{}".format(checkPointModel)
print(y_train.shape)
nsamples, npx, npy, rgb = x_train.shape

d2_train_x = x_train.reshape((nsamples, npx*npy*rgb))

smote = SMOTE()
X_train_smote, y_train_smote =
smote.fit_resample(d2_train_x, y_train)

df = pd.DataFrame({"label": np.argmax(y_train_smote,
axis=1)})
print(df['label'].value_counts())

es = EarlyStopping(monitor='val_loss', mode='min',
verbose=1, patience=2)
```

```
M_checkP = ModelCheckpoint(filepath=filepath,
monitor='val_loss', verbose=1, save_best_only=True,
mode='min')
tensorboard = TensorBoard(log_dir='logs/{}'.format(Name))

# fit model
history = model.fit(x_train, y_train_smote,
validation_split=0.22222, epochs=EPOCHS, verbose=1,
callbacks=[es, M_checkP, tensorboard],
batch_size=batch_size)
# evaluate the model
_, train_acc = model.evaluate(x_train, y_train, verbose=0)
_, test_acc = model.evaluate(x_test, y_test, verbose=0)
if len(sys.argv) == 1:
   folder_name = os.path.dirname(sys.argv[0])
   saved_model = model.save(os.path.join(folder_name,
"model2.h5"))
   print(f"Model saved to {folder_name}.")
print('Train accuracy : %.3f, Test accuracy: %.3f' %
(train_acc, test_acc))

pyplot.figure(0)
pyplot.plot(history.history['loss'], label='training loss')
pyplot.plot(history.history['val_loss'], label='validation
loss')
pyplot.title("втрати")
pyplot.xlabel("епохи")
pyplot.ylabel("втрати")
pyplot.legend()
pyplot.show()

pyplot.figure(1)
pyplot.plot(history.history['accuracy'], label='training
accuracy ')
pyplot.plot(history.history['val_accuracy'],
label='validation accuracy')
pyplot.title("точність")
pyplot.xlabel("епохи")
pyplot.ylabel("точність")
pyplot.legend()
pyplot.show()
def initialize_gui():
    learn_b = Button(top, text="Оновити модель",
command=lambda: os.system('python LearnModel.py'), padx=10,
pady=5)
    learn_b.configure(background='#364196',
foreground='white', font=('arial', 13, 'bold'))
    learn_b.place(relx=0.05, rely=0.85)

    exit_button = Button(top, text="Вихід",
command=top.destroy, font=('arial', 14, 'bold'), padx=10,
pady=5)
```

```
    exit_button.configure(background='#364196',
foreground='white')
    exit_button.place(relx=0.85, rely=0.85)

    upload = Button(top, text="Завантажити знак",
command=upload_image, pady=5)
    upload.configure(background='#364196',
foreground='white', font=('arial', 10, 'bold'))

    upload.pack(side=BOTTOM, pady=70)
    sign_image.pack(side=BOTTOM, expand=True)
    label.pack(side=BOTTOM, expand=True)
    heading = Label(top, text="Розпізнавання дорожніх
знаків", pady=10, font=('Times New Roman', 22, 'bold'))
    heading.configure(background='#CDCDCD',
foreground='#364196')
    heading.pack()
    top.mainloop()
```