Галицький коледж імені В’ячеслава Чорновола
відділення комп’ютерних та видавничих технологій
циклова комісія інформатики та комп’ютерних дисциплін

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач відділенням
комп’ютерних та видавничих
технологій
Чубей О.О. /_____/
підпис
«___» _____ 2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проєкту
освітньо-кваліфікаційного рівня «молодший спеціаліст»
зі спеціальності 122 «Комп’ютерні науки»

на тему: «Розпізнавання дорожніх знаків для керування автомобілем»

Студент групи К-47 Савчишин Я.С. _____
(підпис)

Керівник проєкту Павлюс В.П. _____
(підпис)

Консультанти:

з техніко-економічного
обґрунтування Меленчук Л.І. _____
(підпис)

нормоконтролер Гавришків Н.Г. _____
(підпис)

Тернопіль – 2021

Галицький коледж імені В’ячеслава Чорновола
відділення комп’ютерних та видавничих технологій
циклова комісія інформатики та комп’ютерних дисциплін

ЗАТВЕРДЖУЮ
Завідувач відділенням
комп’ютерних та видавничих
технологій
Чубей О.О. /_____/
підпис
«___» _____ 2020 р.

ЗАВДАННЯ

на дипломне проєктування
на здобуття освітньо-кваліфікаційного рівня «молодший спеціаліст»
студенту Савчишину Ярославу Сергійовичу

(прізвище, ім’я та по-батькові студента)

1. Тема проєкту _____
_____
_____
затверджено наказом по коледжу від “___” _____ 202_ р., №___
2. Термін здачі студентом завершеного проєкту “___” _____ 202_ р.
3. Вихідні дані до проєкту _____
_____
_____
4. Перелік питань, які повинні бути розроблені в проєкті:
а) основна частина _____
_____
б) техніко-економічне обґрунтування _____
_____
_____
_____
5. Перелік графічного матеріалу _____
_____

6. Консультанти проєкту: _____

Розділ	Консультанти	Підпис, дата	
		Завдання видано	Завдання прийнято
з техніко-економічного обґрунтування	_____ (вчена ступінь, звання П.І.Б. _____ консультанта)		

КАЛЕНДАРНИЙ ПЛАН
дипломного проєктування

№ п/п	Найменування етапу	Терміни	
		початку	завершення
1.	Вибір теми, ознайомлення з вимогами до дипломного проєктування	16.11.20	30.11.20
2.	Огляд типових рішень та написання відповідного розділу ПЗ	01.12.20	26.01.21
3.	Дослідження технологій реалізації та написання відповідного розділу ПЗ	27.01.21	15.02.21
4.	Розробка функціональних вимог до проєкту та робота над структурою програмного продукту. Написання відповідного розділу ПЗ	15.02.21	02.03.21
5.	Встановлення та налаштування середовища реалізації та написання відповідного розділу ПЗ	02.03.21	16.03.21
6.	Проєктування програмного засобу (функціоналу, інтерфейсу, бази даних продукту) та написання відповідного розділу ПЗ	16.03.21	16.04.21
7.	Реалізація та налаштування програмного засобу та написання відповідного розділу ПЗ	17.04.21	03.05.21
8.	Доопрацювання модулів	03.05.21	17.05.21
9.	Опрацювання економічного розділу дипломного проєкту та оформлення спеціального розділу	17.05.21	18.06.21
10.	Тестування та налагодження програмного продукту та написання відповідного розділу ПЗ	18.05.21	04.06.21
11.	Робота над оформленням пояснювальної записки	04.06.21	11.06.21
12.	Попередній захист дипломного проєкту, доопрацювання	11.06.21	
13.	Підготовка до захисту дипломного проєкту	18.06.21	23.06.21
14.	Захист дипломного проєкту	24.06.21	

7. Дата видачі “___” _____ 2020р. Керівник _____/

Завдання прийняв до виконання _____/

Реферат

Розпізнавання дорожніх знаків для керування автомобілем. Дипломний проєкт. Савчишин Ярослав. Галицький коледж імені В’ячеслава Чорновола, відділення комп’ютерних та видавничих технологій. Спеціальність 122 «Комп’ютерні науки». ГК, 2021. Сторінок – 64, рисунків – 27 додатків – 2.

Об’єкт дослідження – нейронні мережі, розпізнавання дорожніх знаків.

Метою проєкту є система для розпізнавання дорожніх знаків, яка при завантаженні зображення дорожнього знаку буде його розпізнавати. Це буде реалізовано за допомогою мови програмування Python та середовища розробки PyCharm.

Система повинна бути реалізована у вигляді програми, бути ретельно протестована для уникнення помилок та розпізнавати дорожні знаки з високою точністю.

Для реалізації даного програмного забезпечення було використано велику кількість різноманітних інструментів, системних бібліотек, доступних в середовищі програмування та мову Python.

Результатом виконання роботи є готова до використання програма.

СИСТЕМА, РОЗПІЗНАВАННЯ ДОРОЖНІХ ЗНАКІВ, PYTHON, PYCHARM, TENSORFLOW, МАШИННЕ НАВЧАННЯ, ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ, АВТОПІЛОТ, ВЗАЄМОДІЯ З КОРИСТУВАЧЕМ, ER-ДІАГРАМА.

Abstract

Traffic Sign Recognition for car driving. Diploma project. Savchyshyn Yaroslav. Galytskyy College named after Viacheslav Chornovil, Department of Computer and Publishing Technologies. Specialty 122 "Computer Science". GC, 2021. Pages – 64, figures – 27, appendixes – 2.

The object of research - neural networks, road sign recognition.

The aim of the project is a system for recognizing road signs, which will recognize the road sign when downloading it. This will be implemented using the Python programming language and the PyCharm development environment.

The system must be implemented in the form of a program, be thoroughly tested to avoid errors and recognize road signs with high accuracy.

A large number of different tools, system libraries available in the programming environment and the Python language were used to implement this software.

The result of the work is a ready-to-use program.

ROAD SIGN RECOGNITION SYSTEM, PYTHON, PYCHARM, TENSORFLOW, MACHINE LEARNING SOFTWARE, AUTOPILOT, INTERACTION WITH CORRESPONDENCE, ER DIAGRAM

ЗМІСТ

					ДП. КН 21.448.18.000 ПЗ		
Зм.	*Арк.*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Савчишин Я.С.*			Розпізнавання дорожніх знаків для керування автомобілем	*Літ.*	*Арк.* 5 / *Аркушів* 65
Перев.		*Павлюс В.П.*					
Рецензет.		*Кузик В.М.*				ГК. КВТ. К-47	
Н. Контр.		*Гавришків Н.Г.*					
Зав. від.		*Чубей О.О.*					

СКОРОЧЕННЯ І УМОВНІ ПОЗНАКИ

TSR – Traffic sign recognition

GPS – Global Positioning System

ДТП – Дорожньо-транспортна пригода

ACC – Adaptive cruise control

LDW – Lane departure warning system

LKA – Lane keeping assistance

LCA – Life cycle assessment

RFID – Radio-frequency identification

TSS – Toyota Safety Sense

RSA – Road Sign Assist

ML – Machine learning

GTSRB – German Traffic Sign Recognition Benchmark

GUI – Graphical User Interface

CAGR – Compound annual growth rate

ВСТУП

Нейронні мережі - це обчислювальні системи з взаємопов'язаними вузлами, які працюють подібно до нейронів людського мозку. Використовуючи алгоритми, вони можуть розпізнавати приховані закономірності та кореляції в необроблених даних, кластеризувати та класифікувати їх, і - з часом - постійно вивчати та вдосконалювати.

Розпізнавання дорожніх знаків (TSR) – це технологія, за допомогою якої транспортний засіб може розпізнавати дорожні знаки, розставлені на дорозі, наприклад "стоп", "проїзд заборонений" або "обмеження швидкості". Дана технологія розробляється різноманітними постачальниками автомобілів. Для виявлення дорожніх знаків ця система використовує методи обробки зображень. Дорожні знаки можна опрацьовуються за допомогою камер, спрямованих уперед, які є в наявності багатьох сучасних легкових, вантажних та транспортних автомобілях. Одним з основних використань такої системи є розпізнавання дорожніх знаків, а саме обмеження швидкості. Більшість GPS надають інформацію про ліміт швидкостей, однак за допомогою систем розпізнавання можна отримати додаткову інформацію та відобразити її на приладовій панелі автомобіля, щоб попередити водія про дорожній знак. Більшість автомобілів високого класу мають вдосконалену функцію допомоги водію, яка здебільшого наявна в транспорті європейських країн. Така система може повідомляти, наприклад, звуковим сигналом, водія про перевищення швидкості і з цим знизити ризик автомобільної аварії. На основі вимог автономних та самокерованих автомобілів сучасні системи розпізнавання дорожніх знаків розробляються із використанням конволюційних нейронних мереж. Даний метод буде використаний при розробці проєкту для даного дипломного проєкту.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Обґрунтування доцільності створення системи

Щороку у світі в автокатастрофах гине близько півтора мільйони людей, від 20 до 50 мільйонів травмуються. Причин ДТП існує багато і основними є: перевищення безпечної швидкості руху, порушення правил маневрування, проїзду перехресть та проїзду пішохідних переходів, недотримання дистанції. Найбільший відсоток припадає на порушення правил дорожнього руху водіїв та недотримання безпечної швидкості руху.

Завдяки технологіям життя людини стало простішим та безпечнішим: лікарі використовують високотехнологічні прибори, що контролюють стан пацієнта та допомагають під час операцій, машини виконують важку та небезпечну роботу на підприємствах. Стрімкими темпами в наші дні розвиваються технології «машинного навчання», коли комп'ютер можна навчити самостійно розпізнавати об'єкти, знаходити закономірності, робити логічні висновки на основі великих об'ємів вхідних даних [1].

Одним з перспективних напрямків застосування технологій «машинного зору» стали системи контролю руху транспортних засобів, в яких допоміжний інтерфейс повідомляє водія про той чи інший дорожній знак, сигнал світлофора тощо, або іноді допомагає керувати автомобілем в автоматичному режимі. На жаль, такі системи поки що доступні лише в елітних моделях автомобілів, які через надто високу ціну недоступні більшості водіїв. Саме з цієї причини створення власної автоматизованої системи розпізнавання дорожніх знаків та попередження водія про них є дуже актуальною задачею.

Глибоке машинне навчання набуло своєї популярності завдяки своїй перевазі у забезпеченні точних результатів. Традиційні алгоритми машинного навчання, якими б складними вони не були, все ще досить прості в своїй основі. Їх навчання вимагає великих затрат, людського втручання при виникненні помилок, вони працюють тільки над завданням яким вони були треновані. Алгоритми глибокого навчання, навпаки, дізнаються про

поставлене завдання через мережу нейронів, які відображають їх як ієрархію понять. Кожне складне поняття визначається низкою більш простих понять. І все це алгоритми можуть зробити самі. У контексті комп'ютерного зору це означає спочатку ідентифікувати світлі та темні ділянки, потім класифікувати лінії, потім фігури, перш ніж рухатись до повного розпізнавання зображення. Алгоритми глибокого навчання також є більш ефективними, коли отримують більше даних, що не характерно для алгоритмів звичайного машинного навчання.

Глибоке навчання не тільки дозволило використовувати багато інших фотографій та відеозаписів для навчання алгоритмів навчання, але й полегшило роботу пов’язану з анотацією та маркуванням даних.

1.2 Огляд існуючих рішень

Перед початком розробки нової системи необхідно здійснити огляд існуючих технологій та оцінити їхні переваги та недоліки. Порівняльний аналіз проведемо на прикладі самокерованих автомобілів.

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

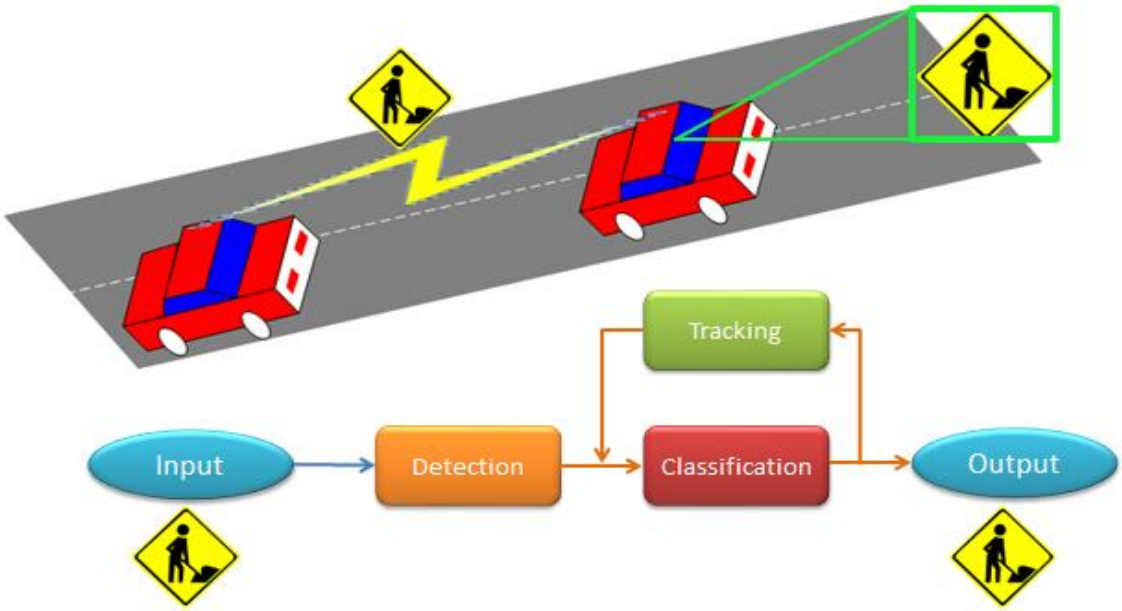

Рисунок 1.1 – Система розпізнавання дорожніх знаків (TSR)

У системах розпізнавання дорожніх знаків (TSR) використовуються встановлені на транспорті камери, які ідентифікують дорожні знаки під час руху автомобіля. Як правило, ці системи розпізнають знаки обмеження швидкості, знаки зупинки та попереджувальні знаки, такі як пішохідний перехід, залізничний переїзд тощо. Їх основною функцією є інформування водія про недавні дорожні знаки, які можуть бути пропущені через відволікання уваги чи неуважність. Вбудована камера сканує узбіччя на 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;
- відстеження виявленої області інтересу;
- класифікація дорожнього знаку на основі внутрішніх наборів даних та алгоритмів нейронної мережі;
- інформування водія, відображаючи символ, що представляє розпізнаний знак.

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

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

З цих причин інші технології, такі як системи на основі зв’язку RFID, вивчаються з метою вдосконалення або доповнення систем TSR на основі камер. Знаки, що спілкуються з транспортними засобами на основі технологій V2V та V2I, також, швидше за все, відіграватимуть важливу роль у системах розпізнавання дорожніх знаків у недалекому майбутньому [2].

Зрештою, системи розпізнавання дорожніх знаків надаватимуть інформацію іншим автомобільним електронним системам, таким як адаптивний круїз-контроль (ACC) та системи безпеки, щоб розширити свої можливості. Наприклад, при використанні спільно з ACC, система TSR може розпізнавати обмеження швидкості, попереджати водія, якщо він / вона їде занадто швидко і автоматично переводити транспортний засіб до встановленого обмеження швидкості без допомоги водія.

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

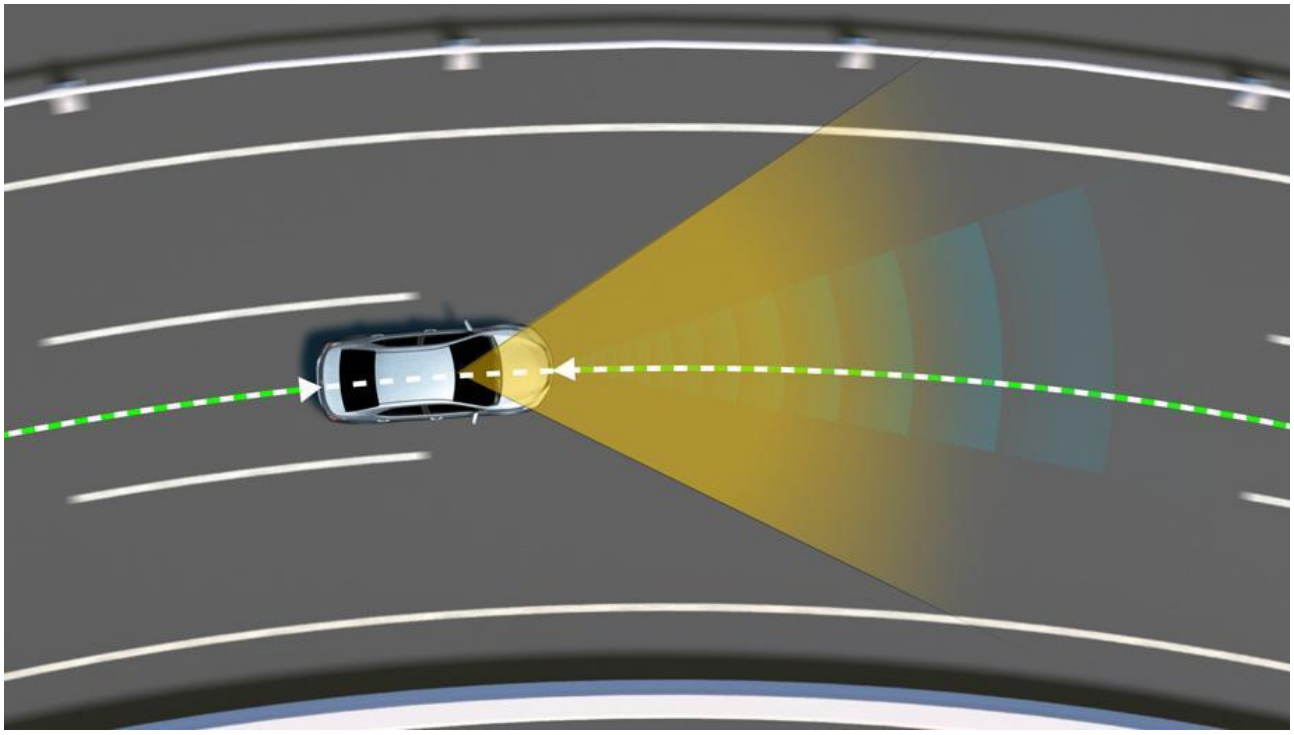

Рисунок 1.2 – Toyota Safety Sense

Система автоматичного гальмування перед іншими автомобілями на порівняно простих моделях (з TSS C) діяла в діапазоні 10-80 км/год. У новому поколінні комплексу для всіх нових легковиків стандарт буде вище – від 10 до 180 км/год [3]. Система контролю рядності з автоматичним підруленням (LDA) в минулому бачила тільки білі і жовті лінії на асфальті, а тепер ще вміє розпізнавати краї дороги, знижуючи ризик виїзду за межі траси. Збережено автоматичне перемикання дальнього світла.

Японці кажуть, що досвід розгортання першого комплексу TSS (а він вже встановлений на п’ять мільйонів машин) і статистика аварій свідчить про зниження на 50% ризику наїзду на машину і на 90% - в поєднанні з системою Intelligent Clearance Sonar (це «віртуальний бампер» з автоматичним гальмуванням для запобігання зіткнень з перешкодами при маневруванні на малих швидкостях) [4]. Друге покоління TSS має знижувати ризик ДТП ще більше. До можливостей TSS додана система розпізнавання знаків RSA, яка розрізняє не тільки знаки обмеження швидкості, але і знаки «цеглина», «стоп», «обгін заборонений». Вона дублює знак на панелі приладів, а також видає водієві попередження при порушенні правил.

Одночасно преміальне відділення Тойоти представило наступне покоління свого комплексу Lexus Safety System +, яке по суті є калькою з нового TSS. І той і інший набори будуть поступово впроваджені на серійної продукції фірми в Японії, Північній Америці і Європі, починаючи з нових моделей, що запускаються на ринок з 2018 року [5]. На рисунку 1.3 зображено роботу LKA (Lane Keep Assist).

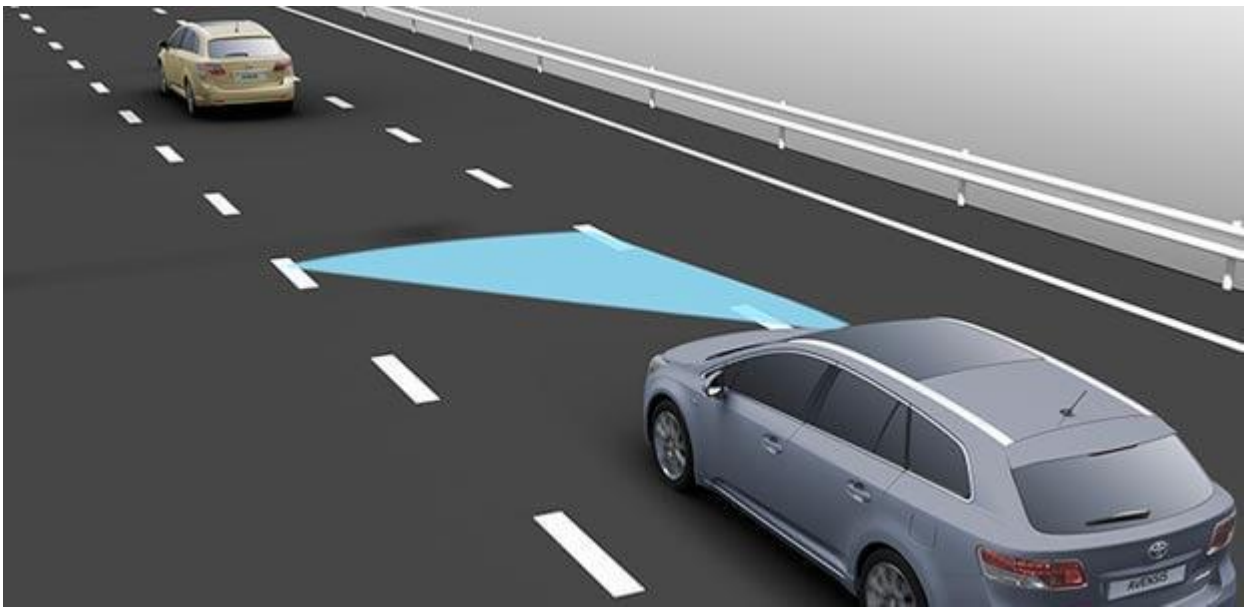

Рисунок 1.3 – Lane Keep Assist

Lane Keep Assist, Lane Departure Warning – технологія має кілька назв, залежно від виробника, який намагається її продати.

З незначними варіаціями обидві електронні системи, які попереджають водія, коли його транспортний засіб виїжджає зі своєї смуги руху, а водій не активує індикатори повороту. Спочатку він був розроблений як простий спосіб виявлення автомобілем того, що водій, можливо, задрімав за кермом або не звернув уваги.

Це, зазвичай, називається попередженням про виїзд із смуги руху. Lane Keep Assist – це трохи вдосконалена система, яка попереджає водія та допомагає утримувати автомобіль на своїй смузі руху без жодного керування водієм.

Системи різняться. Деякі використовують інфрачервоні датчики, але машини все частіше використовують відеокамери, встановлені у верхній частині лобового скла в блоці дзеркала заднього виду. Обидва читають дорожню розмітку, і коли автомобіль дрейфує до білих ліній і виходить за їхні межі, водій отримує попередження.

Деякі дають звукове попередження. Інші вібрують водійське сидіння, а деякі легко підтягують рульове колесо і переміщують машину назад до центру смуги руху, не турбуючи при цьому водія.

Вони недешеві і, як правило, вони є частиною колекції інших технологій, таких як попередження про “сліпі" зони, яка попереджає водія про наявність машини, яка може бути схована в такій зоні через їх плече. Nissan стягує 400 фунтів стерлінгів на своєму маленькому позашляховику Juke. Seat стягує стільки ж за пакет безпеки, який включає його в невеликий сімейний автомобіль Leon. На шикарному салоні Mercedes E-Class він поставляється як частина більш широкого пакету Lane Tracking, який коштує 735 фунтів стерлінгів.

Системи утримання смуги руху та виїзду за неї використовують камери, спрямовані вперед, для моніторингу ліній смуги навколо вашого автомобіля, а також забезпечують візуальні, звукові та / або тактильні попередження для попередження водія, коли автомобіль наближається або перетинає смугову розмітку. Ці системи не активуються, коли використовується сигнал повороту.

Якщо у транспортному засобі є допоміжна система утримання смуги руху (LKA), автоматичне рульове управління або гальмування, то система спробує виправити транспортний засіб, якщо він почне виходити з смуги.

Існує кілька загальних варіацій лінійних систем:

Попередження про виїзд з смуги руху (LDW): водії отримують звукові та / або візуальні попередження про те, що їх транспортний засіб наближається або перетинає розмітку смуги, коли сигнал повороту не активований.

Асистент утримання смуги руху (LKA): забезпечує автоматичне рульове управління та / або гальмування для утримання транспортного засобу на його дорожній смузі.

Допомога при виїзді з дороги: забезпечує автоматичне рульове управління та / або гальмування, щоб спробувати утримати транспортний засіб від виїзду з проїжджої частини.

Допомога в центруванні смуги руху (LCA): забезпечує автоматичне керування та / або гальмування для постійного центрування автомобіля на його смузі.

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. При правильному використанні автопілот зменшує загальне навантаження як водія. 8 зовнішніх камер, радар, 12 ультразвукових датчиків та потужний бортовий комп'ютер забезпечують додатковий рівень безпеки, який допоможе у подорожі.

Для придбання доступні два пакети автопілота: автопілот та повна можливість самокерування, які призначені для використання з повністю уважним водієм, який доповнює кермо і готовий взяти на себе керування в будь-який момент. Незважаючи на те, що ці функції розроблені для того, щоб з часом стати більш самостійними, наявні в даний час функції не роблять автомобіль повністю автономним.

1.3 Постановка завдання

Ціллю даної роботи є реалізація програми розпізнавання дорожніх знаків та попередження про пішохідні переходи, знаки "стоп" та інші важливі дорожні знаки.

Створювана система повинна задовольняти наступні основні вимоги:

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

Проаналізувавши та оцінивши готові рішення та способи їх застосування, було поставлено задачу створити просту портативну систему розпізнавання дорожніх знаків з достатнім функціоналом та зручним інтерфейсом.

Проаналізувавши переваги та недоліки розглянутих існуючих рішень, сформовано основні вимоги до даної програми:

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

2 ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Формалізація вимог до системи

Для безпечної та ефективної роботи даного програмного забезпечення потрібно встановити певні вимоги до даної системи. Недотримання даних вимог може призвести до порушення безпеки та низької продуктивності цього програмного забезпечення.

Оскільки даний дипломний проєкт створений для допомоги водіям під час дорожнього руху, необхідно коректно продумати дану систему. Вона повинна мати зручний інтерфейс та високу швидкодію.

Відмінне програмне забезпечення володіє такими проміжними властивостями:

- Функціональність. здатність програмного продукту вирішувати задачі, потрібні користувачам.
- Ефективність. Здатність програмного продукту забезпечувати необхідну працездатність при заданих умовах стосовно виділених для цього ресурсів.
- Практичність. Здатність програмного продукту бути комфортним у навчанні та використанні, та бути привабливим для користувачів.
- Доступність. Вибіркове використання окремих компонентів програмного продукту
- Надійність. Здатність програмного продукту до безперервної роботи, та коректного виконання задач при тривалому використанні та відповідність стандартам.

2.2 Проєктування структури системи

Дана система повинна виконувати такі функції як: розпізнавання дорожніх знаків, збереження даних про дорожні знаки та оповіщення водія про дорожні знаки.

Для роботи даної системи необхідні наступні модулі:

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

Модуль навчання моделі нейронної мережі відповідає за навчання даного проєкту за допомогою датасету німецьких дорожніх знаків.

Модуль розпізнавання дорожніх знаків є головним модулем, що відповідає за коректне розпізнавання та передачу в цифровий код зображень.

Модуль для порівняння зображень відповідає за визначення дорожнього знаку на зображенні з заздалегідь збереженими даними.

На рисунку 2.1 зображено IDEF3 діаграму роботи розпізнавання дорожнього знаку.

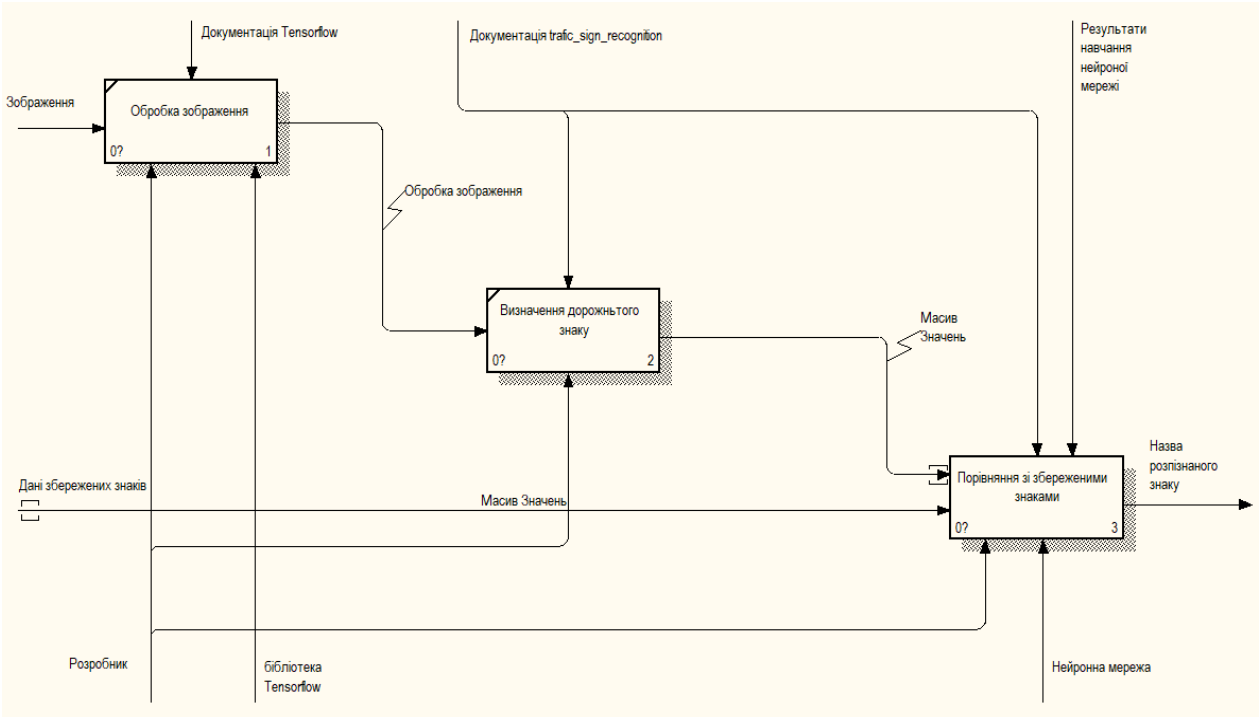

Рисунок 2.1 – IDEF3-діаграма обробки зображення

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

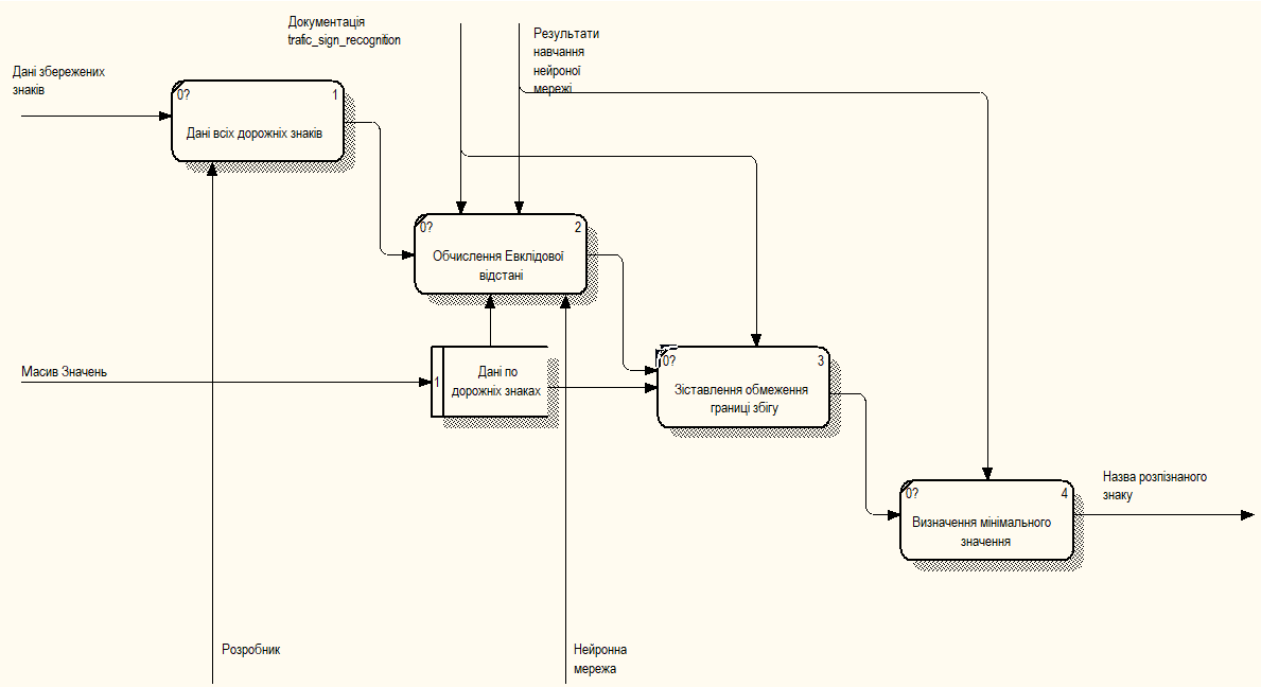

Рисунок 2.2 – IDEF3-діаграма порівняння зображень

Модуль користувацького інтерфейсу відповідає за керування системою користувачем для розпізнавання дорожнього знаку.

2.3 Проєктування користувацького інтерфейсу

Для користувачів, які не мають необхідних знань чи досвіду роботи з програмними консолями необхідний такий елемент як користувацький інтерфейс. Користувацький інтерфейс полегшує взаємодію між користувачем та програмою, застосунком чи веб-сайтом. Також його можна використати під час розробки, щоб підвищити ефективність та зменшити витрати часу на введення тої чи іншої команди.

Створення користувацького інтерфейсу досить складне завдання: необхідно передбачити уподобання користувачів та створити такий інтерфейс, який виконує поставлену задачу.

Мета проектування користувальницького інтерфейсу полягає у створенні такого інтерфейсу, який робить легким, ефективним та приємним управління програмою таким чином, що б досягався необхідний результат.

Користувацький інтерфейс для даного проєкту повинен мати високу швидкодію, простий, але достатній функціонал та бути приємним для

використання. Необхідно забезпечити безперервну роботу програми та передбачати усі можливі недоліки та помилки при активному функціонуванні програми. Необхідними елементами інтерфейсу повинні бути кнопки завантаження дорожнього знаку, вікно завантаженої фотографії чи картинки, кнопка оновлення моделі та кнопка виходу з системи.

Макет користувацького інтерфейсу зображено на рисунку 2.3.

Рисунок 2.3 – Макет користувацького інтерфейсу

Для запобігання випадкових натискань функціональні кнопки будуть розміщені у різних частинах екрану.

2.4 Проєктування алгоритму роботи системи

Алгоритм – це скінченна послідовність чітко визначених, реалізованих на комп'ютері інструкцій, як правило, для вирішення класу конкретних задач або для виконання обчислень.

Процес вирішення практичних задач зі створення комп'ютерних програм поділяється на декілька етапів:

- формалізація та конспектування технічного завдання на вихідну задачу;
- опрацювання алгоритму вирішення задачі;

- написання, тестування, налагодження та документування програми;
- результат вихідного завдання шляхом виконання комп'ютерної програми.

Для початку необхідно зрозуміти вимоги кінцевого результату роботи даної програми. Алгоритм функціонування системи залежить від кінцевої цілі користувача. Першим кроком є ініціалізація користувацького інтерфейсу. Для цього використовується модуль створення інтерактивного користувацького інтерфейсу. За допомогою даного модуля буде створено вікно з текстовими елементами, функціональними кнопками та вікно для зображення, яке завантажив користувач, та яке буде розпізнане програмою.

Для початку роботи програми необхідне навчання моделі нейронної мережі та її подальше збереження. Навчання буде відбуватись на базі GTSRB Dataset (German Traffic Sign Recognition Benchmark Dataset, з англ. Німецький набір даних розпізнавання дорожніх знаків). GTSRB – це великий багатокатегорійний набір даних для класифікування дорожніх знаків. Даний набір даних був створений дослідницькою групою Real-Time Computer Vision. Його було використано під час змагань на IJCNN (International Joint Conference on Neural Networks) 2011 у Сан-Хосе, Каліфорнія, США. GTSRB містить 43 класи дорожніх знаків, 39209 навчальних зображень і 12630 тестових зображень. Зображення мають різну освітленість та різноманітний фон. При успішному навчанні моделі вона зберігається у файли програми для подальшого використання. За необхідності користувач може оновити модель нейронної мережі.

Подальші дії користувача передбачають завантаження фотографії чи картинки дорожнього знаку який необхідно розпізнати.

Наступним кроком є обробка та розпізнавання зображення. Даний модуль обробляє завантажене зображення та звіряє його з уже збереженою моделлю та виводить результат для користувача. Дана обробка зображення подана на рисунку 2.4.

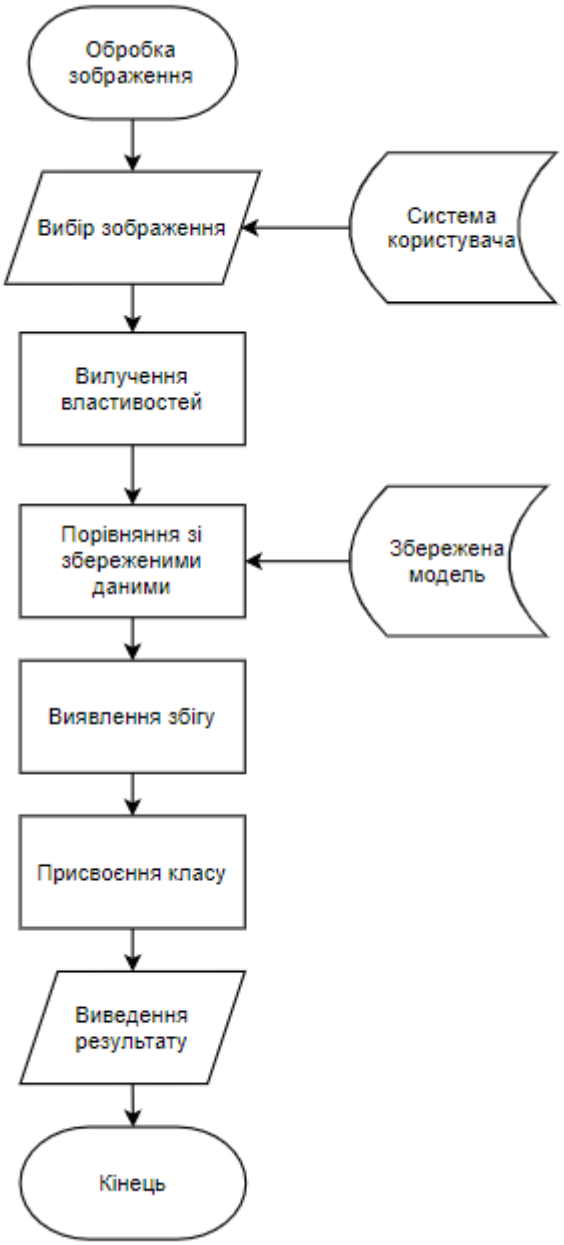

Рисунок 2.4 - Алгоритм роботи моделі обробки зображення

Користувач може продовжити роботу з програмою, або завершити її, натиснувши на кнопку «Вихід».

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Опис технологій та засобів реалізації

Для розробки програми з розпізнавання дорожніх знаків, можна використати різноманітні високорівневі мови програмування. Для даного проекту буде використано мову Python. Python – це інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою та строгою динамічною типізацією. Вбудовані структури даних високого рівня в поєднанні з динамічним набором тексту та динамічним прив'язуванням роблять її дуже привабливою для швидкої розробки застосунків, а також для використання в якості мови сценаріїв для з'єднання існуючих компонентів разом. Простий та легкий у вивченні синтаксис Python зменшує витрати на обслуговування програми підкреслює читабельність. Python інтерпретатор та його стандартна бібліотека є у доступі на всіх основних платформах у вихідній або двійковій формі. Python був створений Гідо ван Россумом, та вперше випущений 20 лютого 1991 року. Python 2.0 був випущений в 2000, з новими функціями такими як розуміння списків та система збору сміття(garbage collection – форма автоматичного керування пам'яттю). Python 3.0 випущений в 2008 році приніс багато змін, і тому велика частина коду більше не є сумісною з Python 2 [6].

Гвідо ван Россум поставив для себе завдання створити інтуїтивну та просту мову програмування, яка:

– є настільки ж потужною як і інші конкуренти;

– знаходиться у відкритому доступі, щоб кожен міг взяти участь у її розвитку;

– має синтаксис, який є настільки ж простим, як англійська мова;

– є зручною для повсякденних завдань, дозволяючи скоротити час розробки.

Переваги цієї мови:

- велика кількість готових модулів;
- переносимість програм;
- чистий синтаксис;
- стандартний дистрибутив має потужне та просте середовище розробки, написане мовою Python, яке називається IDLE;
- можливість виконання Python в діалоговому режимі.

На сьогоднішній день Python є однією з найпопулярніших мов програмування. Зрозумілі правила синтаксису мови програмування також полегшують читання коду та підтримку додатків.

Для реалізації завдання на Python було використано дані бібліотеки: Tensorflow, NumPy, Tkinter, Pillow.

На сьогоднішній час розвиток нейронних мереж та штучного інтелекту стрімко зростає, а також зростає і попит на дану тему. Існують різноманітні бібліотеки, що надають великий функціонал роботи у сфері нейронних мереж, обробки зображень. Одна з таких бібліотек - Tensorflow.

TensorFlow - це платформа для машинного навчання з відкритим кодом. Вона підтримується такими мовами програмування як Python та JavaScript та працює на більшості сучасних операційних системах. Ця платформа має всеосяжну, гнучку екосистему інструментів, бібліотек та ресурсів спільноти, що дозволяє дослідникам просувати найсучасніші технології ML (machine learning, з англ. машинне навчання), а розробникам легко створювати та розгортати додатки, що працюють на ML[7]. Комп’ютерне бачення – це процес, за допомогою якого комп’ютер може розрізняти унікальні елементи на зображеннях, і базуючись на таких елементах присвоювати їм заздалегідь визначені класи[8].

За допомогою Tensorflow можна виконати низку завдань, таких як:

- розпізнавання символів;
- розпізнавання одягу;

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

Компанія Apple використовує Tensorflow для розробки та роботи Siri, а DeepFace від Facebook виконує розпізнавання об’єктів за допомогою Tensorflow [9]. Та насамперед його використовує Google для розробки більшості своїх додатків та послуг.

Tensorflow має наступний функціонал:
– навчання моделі нейронної мережі;
– збереження моделей;
– машинне навчання та кластеризація;
– обробка введеного зображення;
– виявлення об’єктів.

Навчання моделі – це перший крок над роботою з нейромережею, для цього необхідні датасети – каталоги зі строгим сортуванням об’єктів за їх класами, а клас - це тип об’єкта, що відрізняє його від решти (наприклад, фрукти від столових приборів, або яблуко від кавуна).

Наступним кроком є виявлення об’єктів на зображенні – тут все просто: нейронна мережа знаходить необхідний об’єкт та продовжує роботу програми з подальшої обробки зображення.

Обробка зображення передбачає підкреслення особливих елементів для того чи іншого об’єкта та порівняння його до вже навченої моделі з метою присвоєння класу об'єкту.

Модуль Tkinter необхідний для створення простого користувацького інтерфейсу. Даний пакет створений для роботи з бібліотекою Tk. Для реалізації компонентів користувацького інтерфейсу (GUI) з бібліотекою TK було використано мову програмування Tcl. Такими елементами GUI є функціональні кнопки, текстові поля, вікна, зображення і тд. За допомогою

цього можливе просте керування програмою без написання команд та виклику потрібних функцій.

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 такі елементи управління як текстові поля, списки, кнопки для відображення на графічному інтерфейсі. Такими елементами керування є віджети.

Основними елементами GUI є:

– Label (надпис) – віджет що реалізує вікно відображення, куди можна помістити текст або зображення.

– Text (текст) – віджет що надає розширені можливості для редагування багаторядкового тексту та його формату.

– Listbox (список) – віджет що використовується для відображення списку елементів, з яких користувач може вибрати один або більше елементів.

– Checkbutton (прапорець) – віджет для зображення ряду параметрів для користувача.

– Entry (поле введення) – віджет однорядкового елементу для вводу значень користувачем.

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

– Button (кнопка) – віджет для створення кнопок.

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

– Radiobutton (перемикач) – віджет що реалізує кнопку вибору з декількома варіантами, обрати можливо тільки один [10].

NumPy – це бібліотека Python призначена для здійснення різноманітних операцій над великими масивами даних. Оскільки Python є інтерпретованою мовою програмування, робота з великими масивами є помітно довшою ніж у інших мовах, які є компільованими, такими як C++ та Java. NumPy підвищує швидкість роботи над традиційними масивами Python та дозволяє проводити більше операцій над ними.

Масиви NumPy мають фіксований розмір при створенні, на відміну від списків python (які можуть динамічно зростати). Зміна розміру ndarray створить новий масив і видалить оригінал.

Всі елементи масиву NumPy повинні мати однаковий тип даних, а, отже, вони матимуть однаковий розмір у пам'яті. Є виняток: можна мати масиви об’єктів Python разом з NumPy, що дозволяє мати масиви елементів різного розміру.

Масиви NumPy сприяють вдосконаленим математичним та іншим типам операцій над великою кількістю даних. Як правило, такі операції виконуються ефективніше та з меншим кодом, не відміну від вбудованих функцій Python.

Безліч науково-математичних пакетів на основі Python використовують масиви NumPy, хоча вони, як правило, підтримують введення послідовності Python, вони перетворюють таке введення у масиви NumPy до обробки, і часто виводять масиви NumPy. Іншими словами, для ефективного використання більшої частини сучасного науково-математичного програмного забезпечення на основі Python, простого знання того, як використовувати вбудовані типи послідовностей Python, недостатньо - необхідно також знати, як використовувати масиви NumPy.

Бібліотека зображень Pillow додає можливості обробки зображень до інтерпретатора Python. Ця бібліотека забезпечує широку підтримку формату

файлів, ефективне внутрішнє представлення та досить потужні можливості обробки зображень. Дана бібліотека зображень розроблена для швидкого доступу до даних, що зберігаються у декількох основних піксельних форматах. Це повинно забезпечити міцну основу для загального інструменту обробки зображень.

Застосування цієї бібліотеки:

– архіви зображень – бібліотека зображень Python Pillow ідеально підходить для архівування зображень та пакетної обробки програм. За допомогою бібліотеки можна створювати мініатюри, конвертувати між форматами файлів, друкувати зображення тощо. Поточна версія визначає та читає велику кількість форматів;

– відображення зображення – включає інтерфейси Tk PhotoImage та BitmapImage, а також інтерфейс DIB Windows, який можна використовувати з PythonWin та іншими наборами інструментів на базі Windows. Багато інших наборів графічного інтерфейсу постачаються з підтримкою PIL. Для налагодження програми існує метод show(), який зберігає зображення на диск і викликає утиліту зовнішнього дисплея;

– обробка зображень – бібліотека містить основні функціональні можливості обробки зображень, включаючи точкові операції, фільтрацію з набором вбудованих ядер згортки та перетворення кольорового простору. Бібліотека також підтримує зміну розміру зображення, обертання та довільні перетворення. Також існує метод гістограми, який дозволяє витягувати деякі статистичні дані із зображення. Це може бути використано для автоматичного посилення контрасту та для глобального статистичного аналізу.

3.2 Реалізація користувацького інтерфейсу та основних функцій системи

Для початку розробки будь-якого програмного забезпечення необхідно встановити середовище розробки. Оскільки було обрано мову програмування Python, то в якості середовища розробки було використано JetBrains PyCharm.

Для встановлення цього середовища потрібно перейти на сторінку сайту JetBrains, Download PyCharm та натиснути кнопку «Download» версії «Community» [11]. Фрагмент даної сторінки зображено на рисунку 3.1.

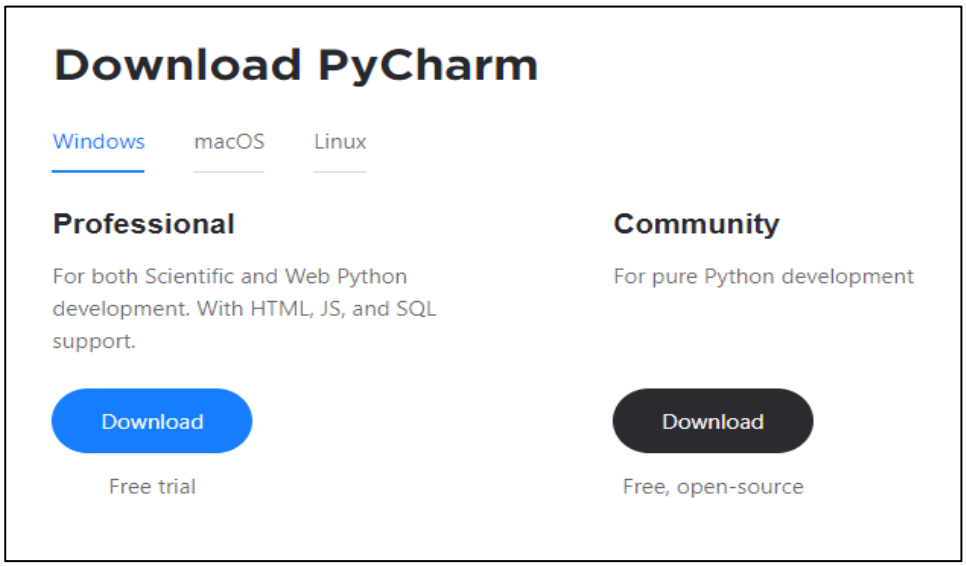

Рисунок 3.1 – Сторінка завантаження PyCharm

Після кінця завантаження, запускаєм цей exe-файл для інсталювання середовища для системи ПК. Обираєм шлях для інсталяції програмних файлів PyCharm Community Edition, що зображено на рисунку 3.2.

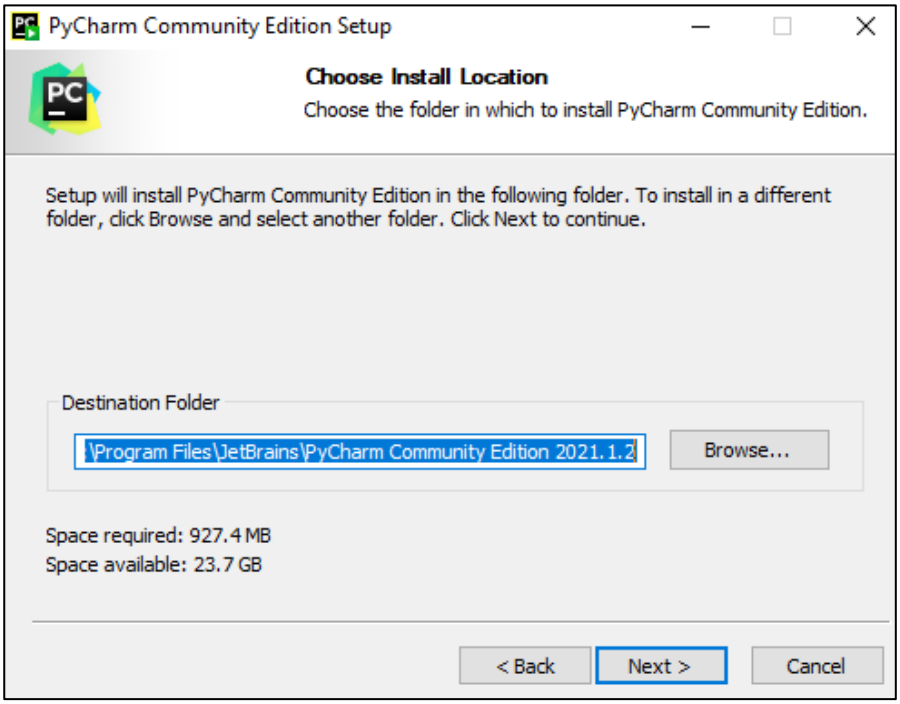

Рисунок 3.2 – Вказування шляху інсталяції

Наступним кроком є модифікація встановлення, а саме прапорці:

– створення ярлику на робочому столі;
– оновлення контекстного меню;
– створення асоціацій;
– оновлення шляху змінних.

Ці налаштування зображено на рисунку 3.3.

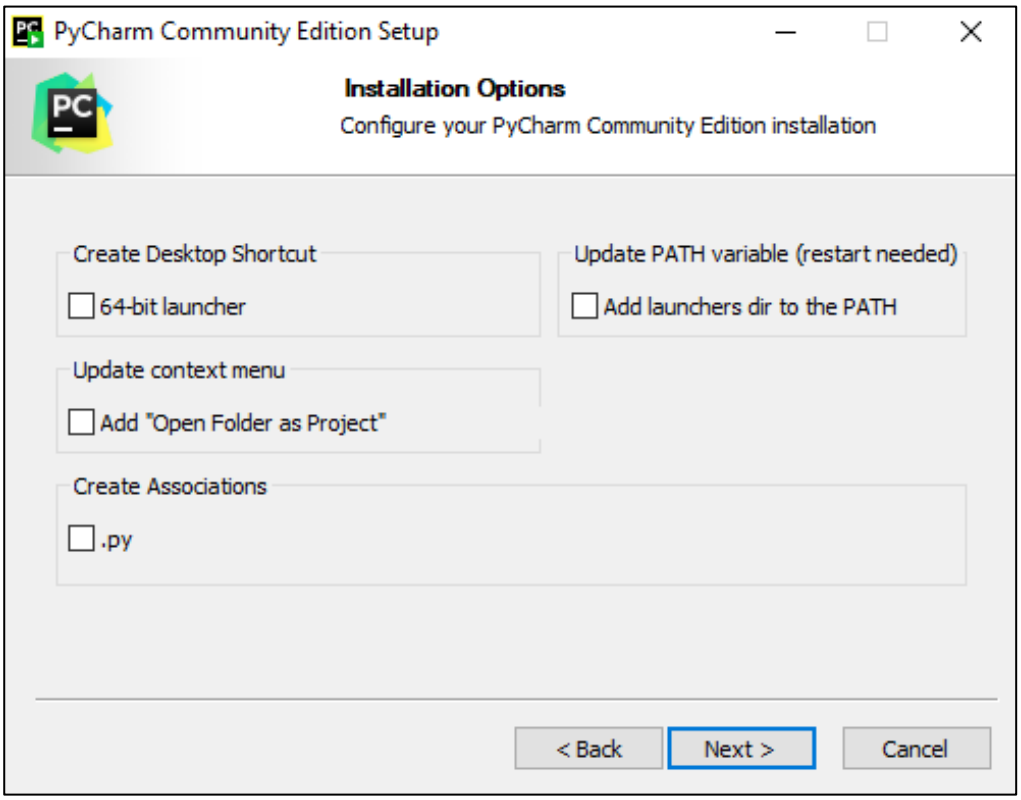

Рисунок 3.3 – Додаткові налаштування

У наступному вікні пропонується створити папку з іменем «JetBrains» в меню «Пуск» що зображено на рисунку 3.4. Після натискання кнопки «Install» розпочнеться інсталяція. Процес інсталяції зображено на рисунках 3.4 та 3.5.

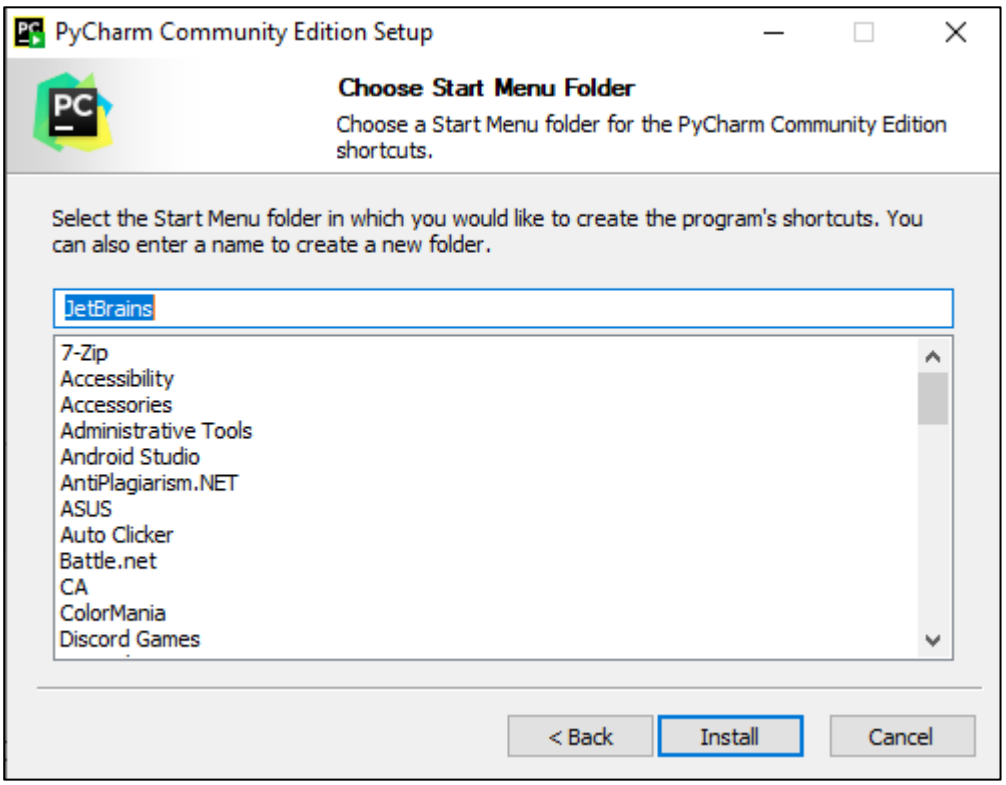

Рисунок 3.4 – Створення папки в меню «Пуск»

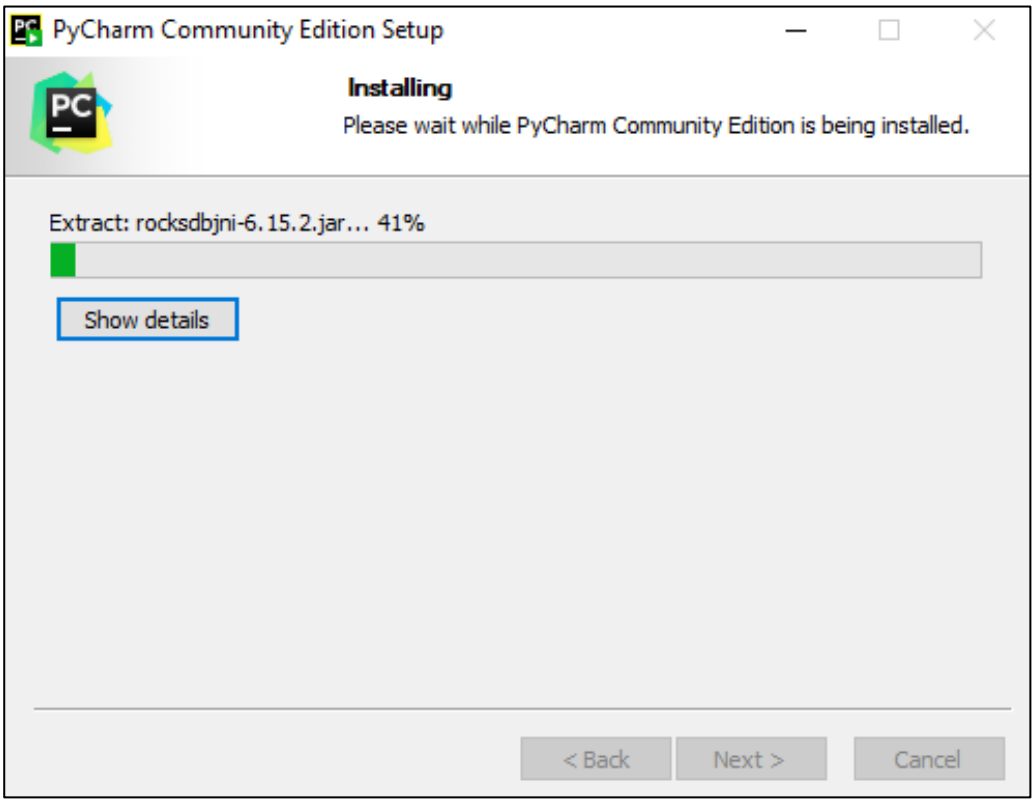

Рисунок 3.5 – Процес інсталяції

Після завершення інсталяції можна запустити дану програму. Для цього потрібно поставити галочку біля «Run PyCharm Community Edition» та натиснути кнопку «Finish», як зображено на рисунку 3.6.

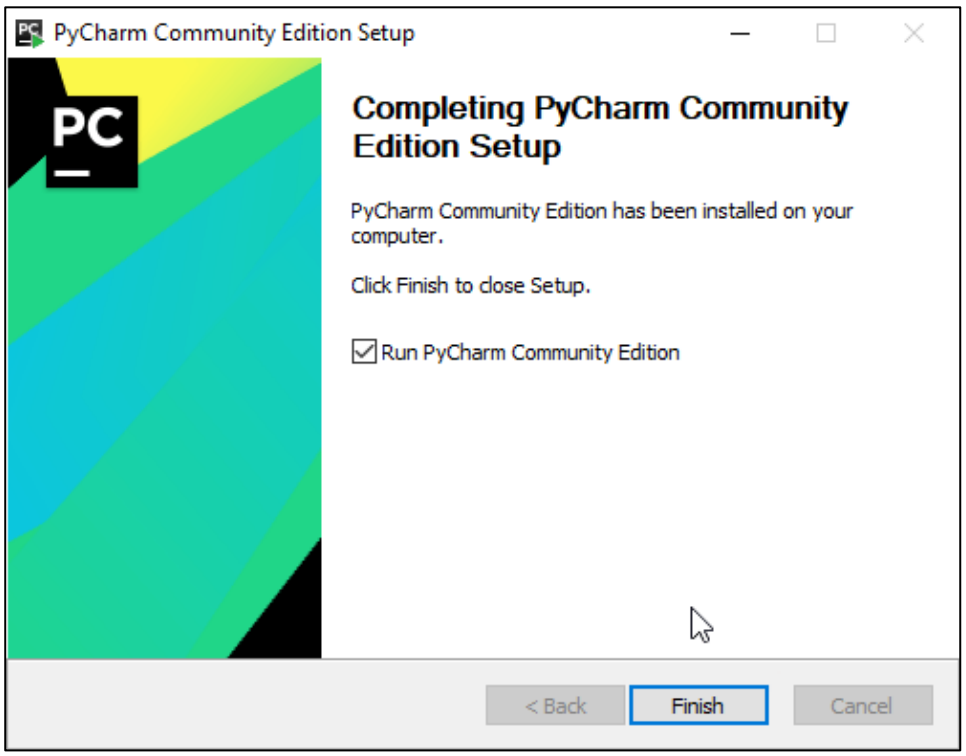

Рисунок 3.6 – Завершення інсталяції та запуск програми

Після запуску, відривається початкове вікно, де натиснувши на кнопку «New Project», почнеться етап реалізації проєкту. Це вікно зображено на рисунку 3.7.

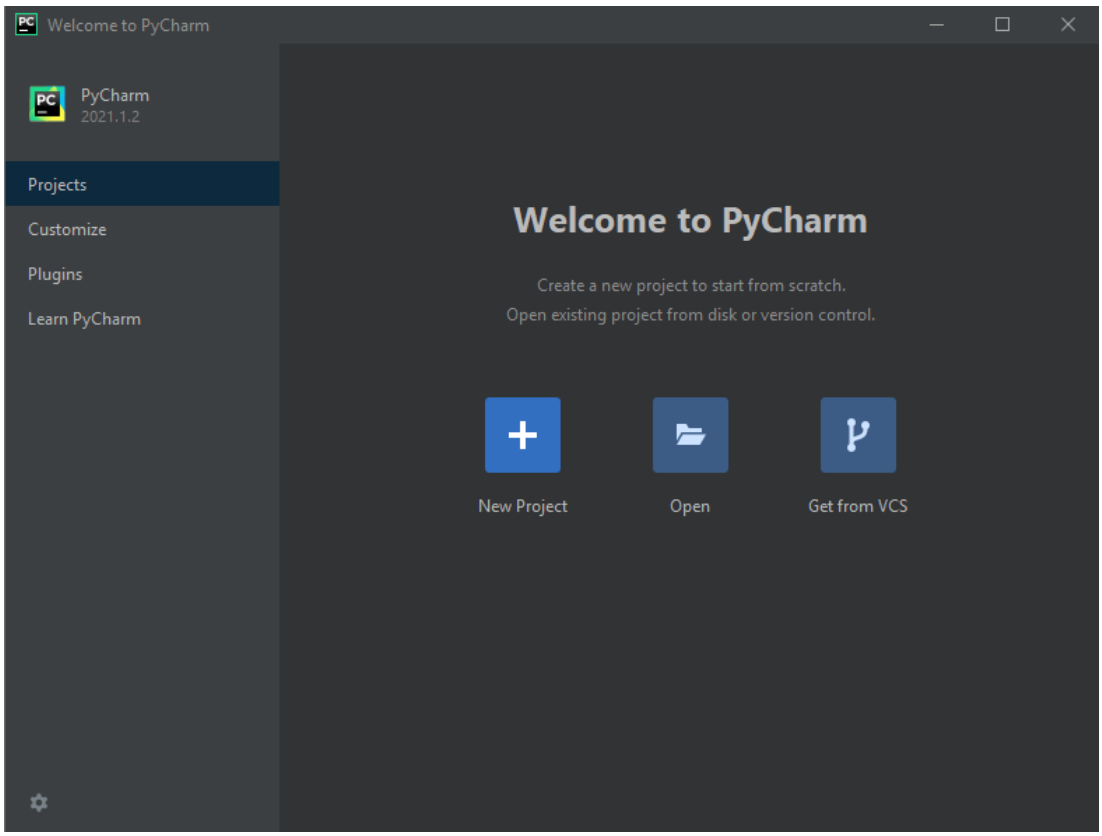

Рисунок 3.7 – Завершення інсталяції та запуск програми

Першим етапом реалізації програмного забезпечення є тренування нейронної мережі. Даний процес відбувається у файлі RecognizeItem.py, як поданий у лістингу А2, додаток А.

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])
```

Завантаження класів моделей з зображеннями починається з:

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

та з викликом 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 приймає посилання на директорію датасету, форматує, змінює розширення зображень до 30х30 та повертає класи відповідних зображень для їх ділення на категорії.

Архітектура методу обробки зображень описана в 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
```

Було вирішено використати один конволюційний шар для зменшення об’єму отримуваних данних та підкреслення унікальностей кожного дорожнього знаку. Було використано 512 нейронів для обробки зображень оскільки це є най оптимальнішим вибором.

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 зберігає модель та показує результати навчань, а саме відсоток точності та відсоток втрат.

Файл LearnModel.py, поданий у лістингу лістингу А3, додаток А, запускає навчання моделі, підставляє змінні імпортовані з файлу RecognizeItem.py, зберігає успішні етапи навчання і модель та підводить підсумки роботи навчання.

З наступним програмним кодом виводяться графіки(рисунки 3.8 та 3.9) на яких зображено точність та втрати моделі відповідно до епох навчання:

```
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()
```

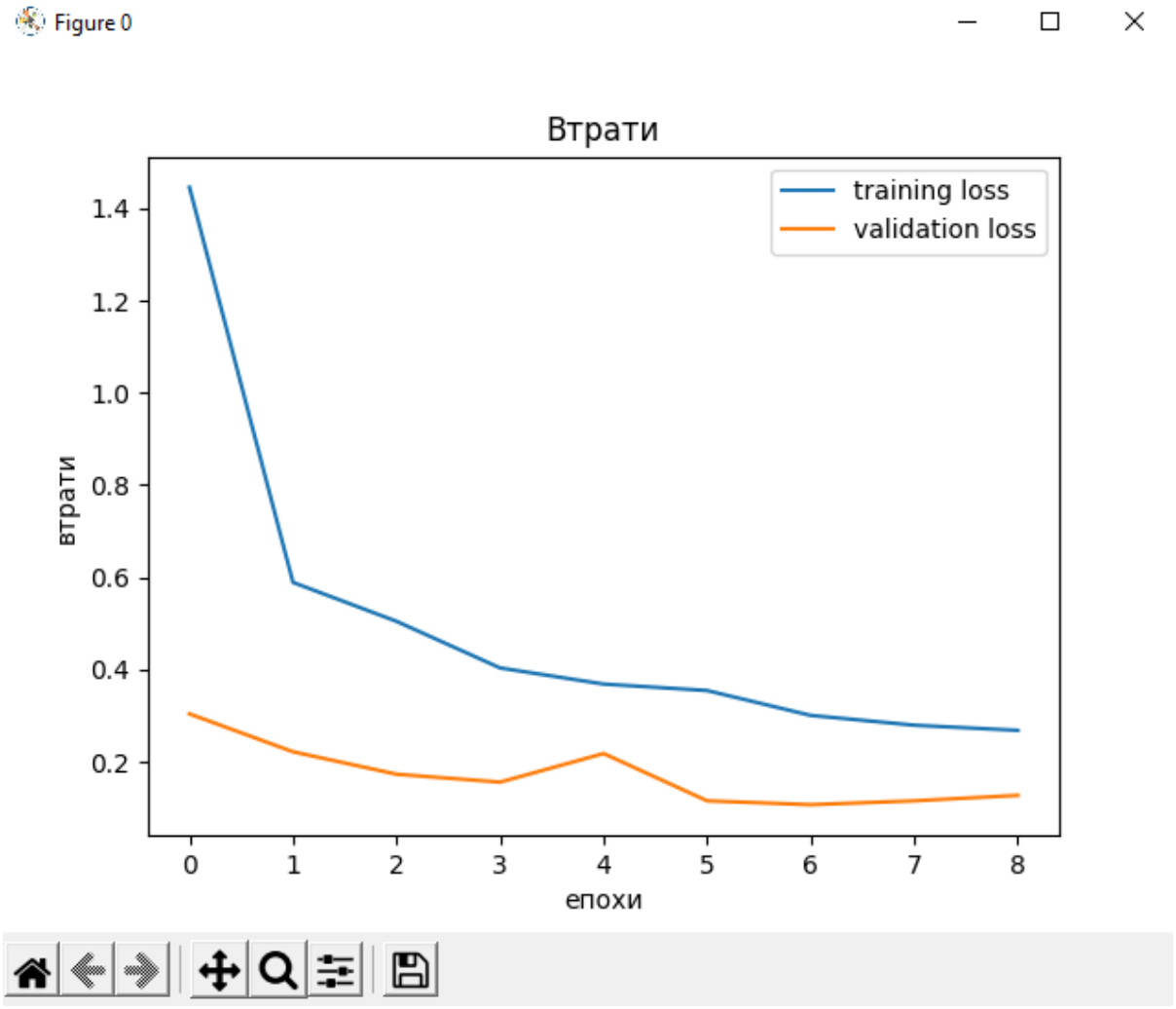

Рисунок 3.8 – Втрати під час навчання моделі

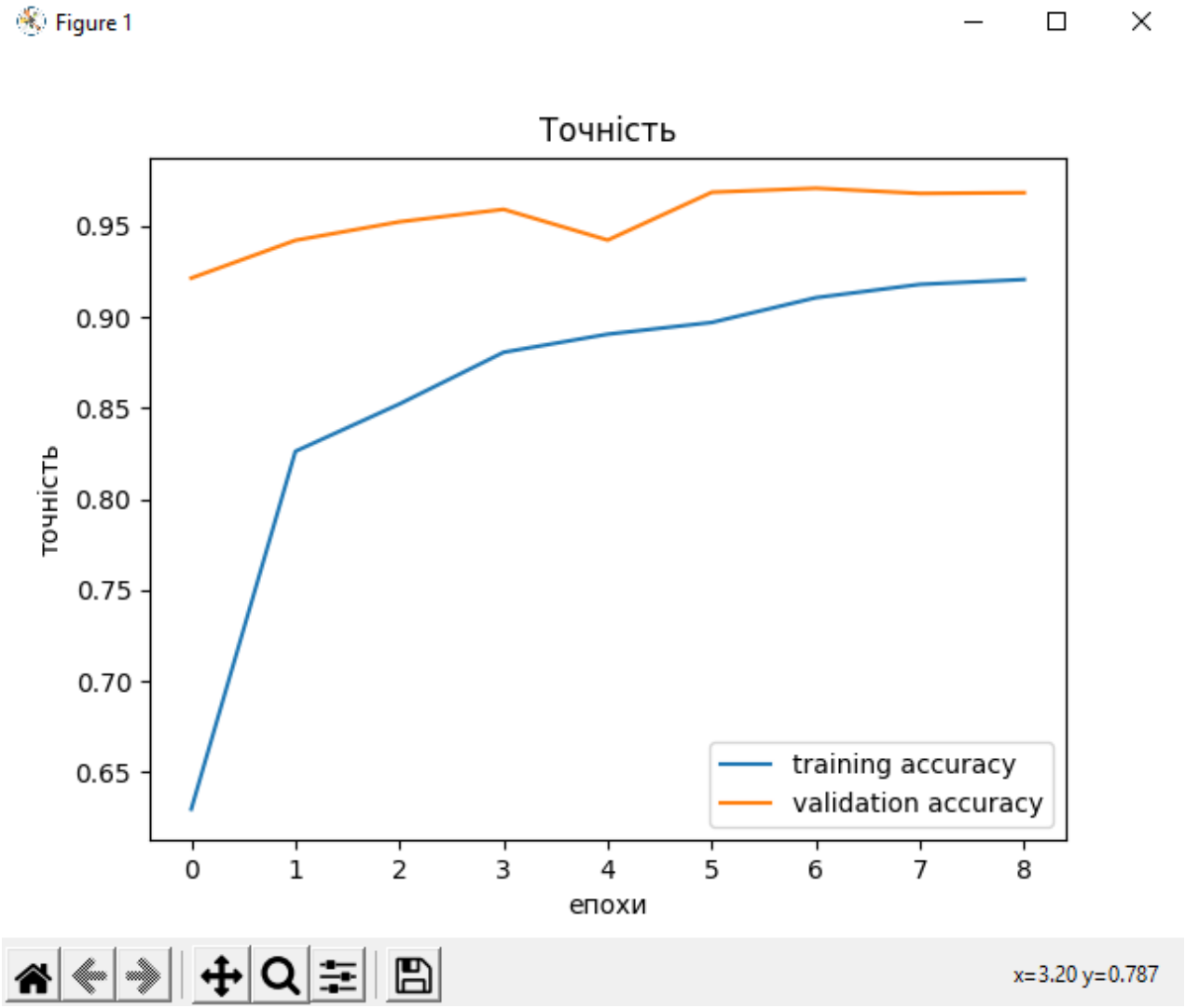

Рисунок 3.9 – Точність під час навчання моделі

При високих показниках навчання та, якщо, відсоток валідації не змінився від попередньої епохи, навчання припиняється та зберігається.

Це необхідно робити щоб не виникало збоїв навчання, або зменшення відсотку точності моделі.

У файлі RecognizeSigns.py, що подано в лістингу А1 додатку А, відбувається завантаження збереженої моделі та ініціалізація користувацького інтерфейсу.

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

Також у даному файлі находиться словник з усіма назвами дорожніх знаків, які програма здатна розпізнати. Значення даного словника

присвоюються відповідно до класів розпізнаних знаків. За це відповідає функція def_classify:

```
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 завантажує зображення обране користувачем та форматує його до глибини кольору 24, оскільки Tensorflow працює з певними властивостями зображень.

```
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
```

3.3 Тестування програмного забезпечення

Тестування є невід’ємним елементом про розробці програмного забезпечення. Необхідно оцінити коректність та виявити недоліки під час роботи програми.

Спочатку необхідно добавити або оновити нейронну модель. Для цього необхідно натиснути на кнопку «Оновити модель», яку зображено на рисунку 3.10.

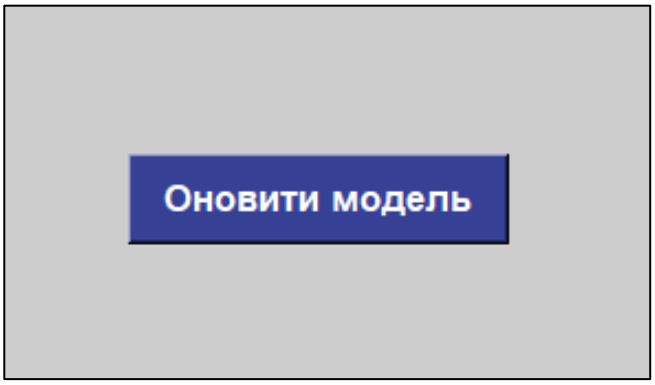

Рисунок 3.10 – Кнопка оновлення моделі

Навчання моделі триватиме кілька хвилин тому програма буде не в робочому стані на протязі цього часу, але по закінченню ця модель буде збережена у файлах системи та її можна буде використовувати без повторного навчання. Даний процес зображено на рисунках 3.11-3.13.

```
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 3.11 - 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 3.12 - 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 3.13 - Результат навчання нейронної мережі

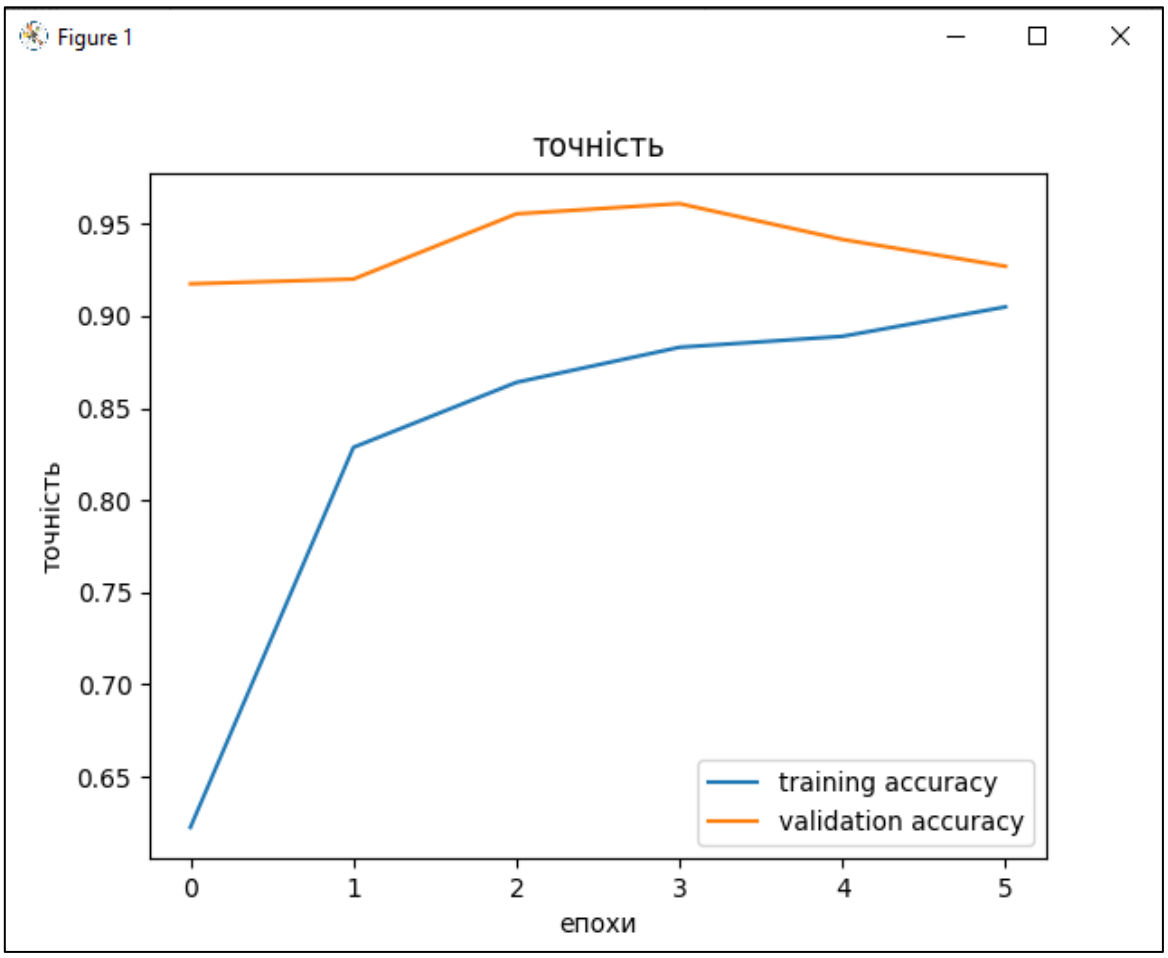

Рисунок 3.14 – Графік точності навчання моделі відносно до епох

Після закінчення навчання можна продовжити роботу з програмою.

Для завантаження зображення яке необхідно розпізнати потрібно натиснути кнопку «Завантажити знак» та обрати дорожній знак (рисунок 3.15).

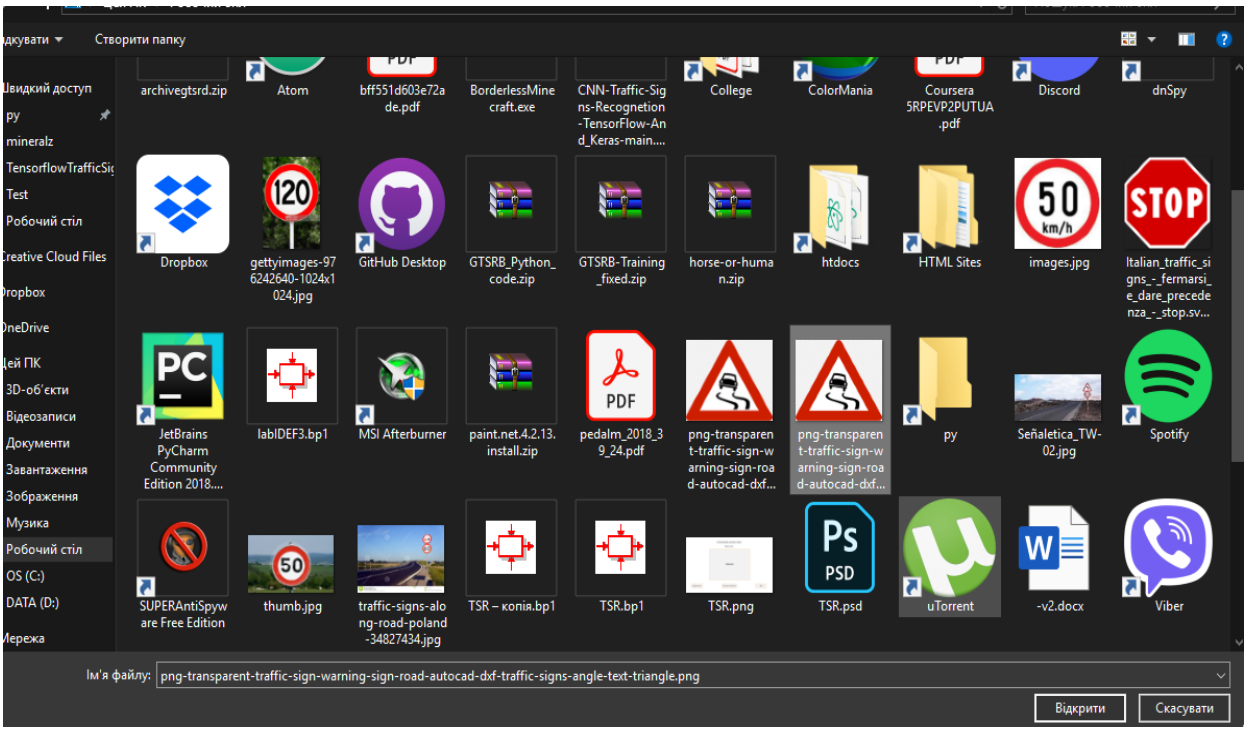

Рисунок 3.15 – Вибір зображення за допомогою файлового провідника

Як тільки знак буде завантажено, програма надасть результат розпізнавання (рисунок 3.16).

Рисунок 3.16 – Результат роботи програми

Для перевірки точності нейронної мережі проведемо додаткові тести по розпізнаванню знаків. Дане тестування зображено на рисунках 3.17-3.8.

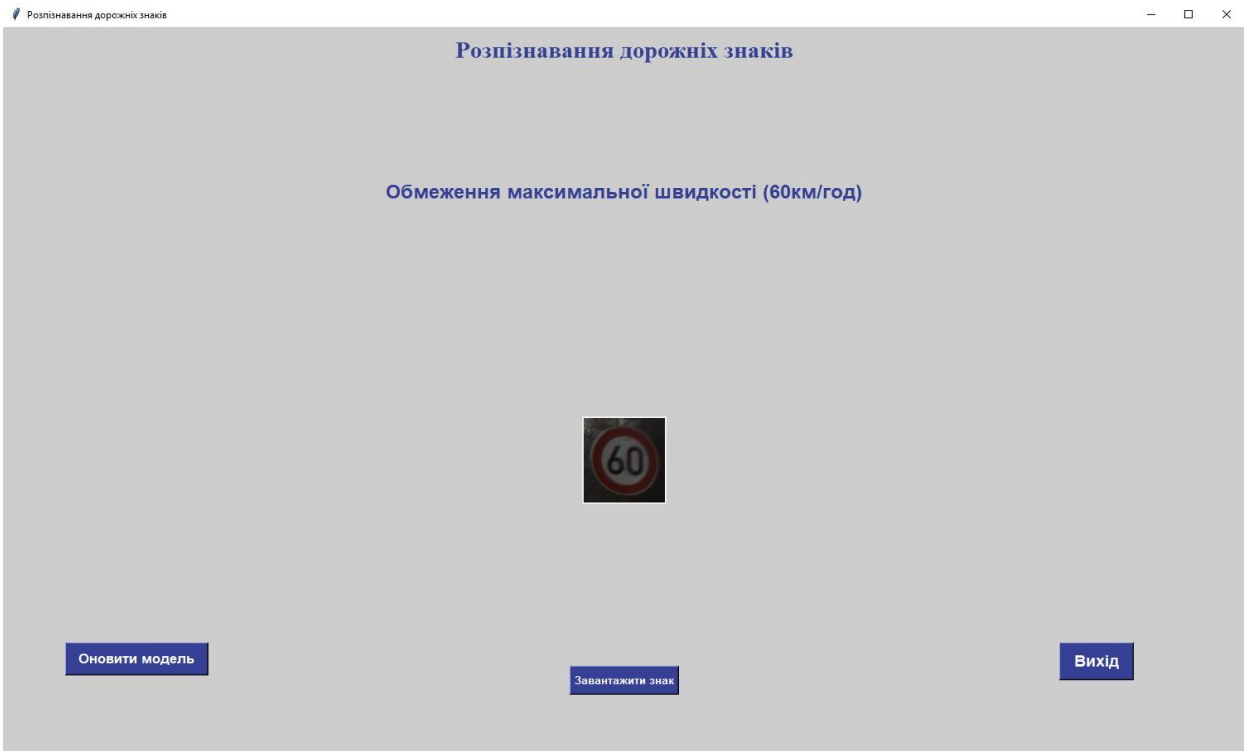

Рисунок 3.17 – Додаткове тестування роботи програми

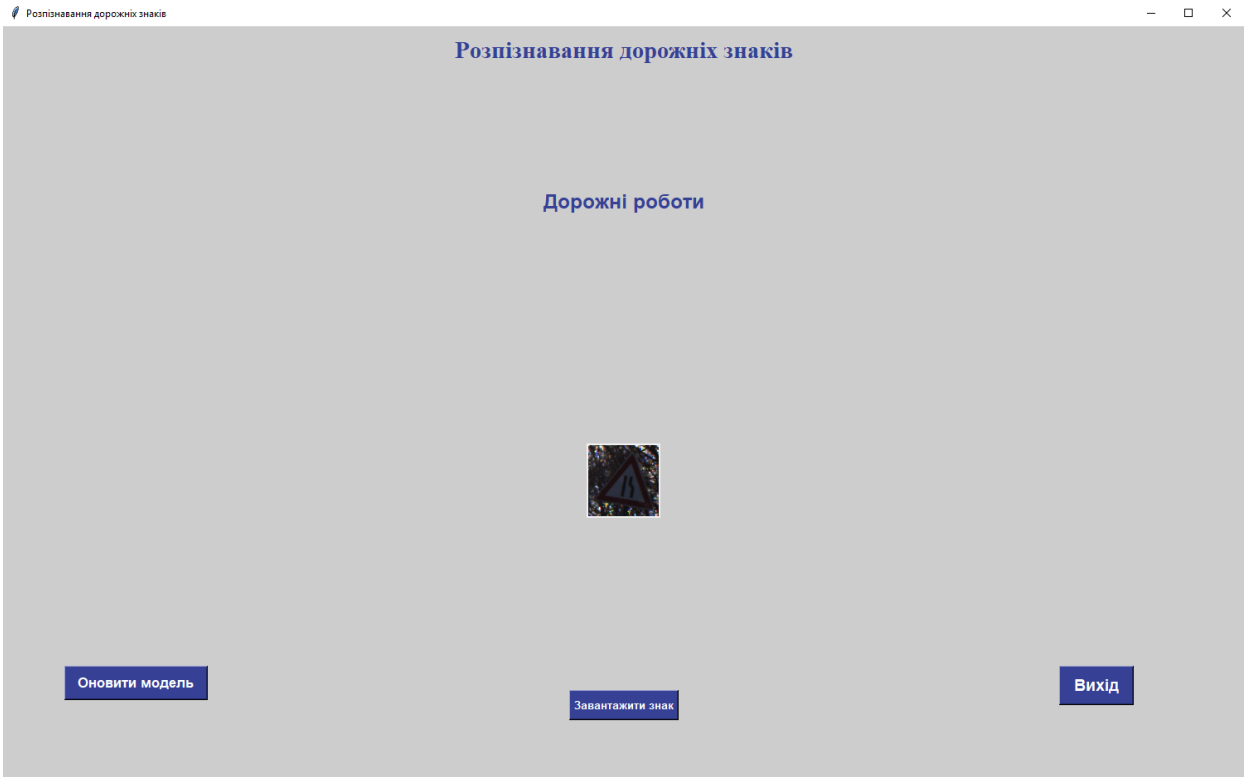

Рисунок 3.18 – Помилкове розпізнавання дорожнього знаку

Точність нейронних мереж не може бути 100%, однак можливо її наблизити до такої мітки.

На точність розпізнавання об’єктів впливають багато факторів, такі, як освітлення, різкість, вигляд знаку та, насамперед, навчання.

Для початкового підвищення точності необхідно використати датасет з великим обсягом зображень, а частина цих зображень повинна бути з різним освітленням та з різними ракурсами. Також необхідно використати конволюційні перетворення зображень для зменшення вхідних даних, що пришвидшує навчання моделі та підкреслює унікальні елементи на зображеннях, що значно збільшує точність.

Для закінчення роботи необхідно натиснути на кнопку «Вихід» (рисунок 3.19).

Рисунок 3.19 – Натискання на кнопку «Вихід» та закриття програми

4 ТЕХНІКО-ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ

4.1 Аналіз ринку

У 2020 році ринок автономних автомобілів оцінюється у 19,46 млрд. доларів. Очікується, що ринок прогнозуватиме показник CAGR близько 18,06% протягом прогнозованого періоду, 2020-2030 [12].

Спалах COVID-19 та подальші закриття підприємств вплинули на ринок автономних автомобілів у ряді країн. Це видно з мінімальної загальної відстані, яку проходять випробувальні машини в будь-якій країні, оскільки основні гравці були змушені припинити тестування під час блокування.

Автономні машини використовують технології, такі як RADAR, LIDAR, GPS та комп'ютерний зір, щоб відчути своє оточення. Вдосконалені системи управління, вбудовані в автомобіль, можуть інтерпретувати сенсорні входи для виявлення вивісок або уникнення зіткнень.

Основні компанії-виробники автомобілів, технологічні гіганти та спеціалізовані стартапи протягом останніх п’яти років інвестували понад 50 млрд. доларів у розробку технологій автономних автомобілів(АВ), причому 70% грошей надходило за межі автомобільної промисловості. Оскільки державні органи розуміють, що АВ мають величезний потенціал та пропонують економічні та соціальні вигоди, вони, ймовірно, продовжуватимуть підтримувати відповідний ринок.

Одними з провідних компаній на ринку автомобілей, що використовують розпізнавання дорожніх знаків є: Ford, Honda, Audi, BMW, Lexus, Mercedes, Nissan, Opel, Citroën, Infiniti, Jaguar, Porsche, Renault, Toyota, Volkswagen, Volvo Land Rover та Peugeot.

Така компанія як Tesla просунула цю ідею на наступний рівень, та стала першою в масовому виробництві автомобілей з автопілотом. Ціни таких автомобілей коливаються від 38тис. до 150тис. доларів США. На даний момент можна придбати такі моделі:

– Tesla Model 3 - 40,190 доларів США.
– Tesla Model S - 70,620 доларів США.
– Tesla Model Y - 52,190 доларів США.
– Tesla Model X - 81,190 доларів США.
– Tesla Roadster - MT Estimate: 200,000 доларів США.
– Tesla Cybertruck -MT Estimate: 39,900 доларів США.
– Tesla Semi - MT Estimate: 150,000 доларів США.

На сьогоднішній день компактні пристрої з функцією розпізнавання дорожніх знаків малопоширені, компанії-виробники проектують та конструюють новітні автомобілі з вже встановленими датчиками, які слідкують за всім що відбувається на дорозі, однак не кожен може придбати такі автомобілі, на відміну від портативного пристрою з меншим, однак з корисним функціоналом.

4.2 Розрахунок витрат на проектування

Продуктивність та мотивація виконання розробки проекту насамперед залежить від оплати праці. Від об’єму праці, її складності та умов, господарської діяльності підприємства, кваліфікації особи та результатів праці залежить розмір зарплати працівника. На даний момент 2021 року, Законом України встановлено такі норми: прожитковий мінімум для працездатної особи на місяць становить 2189 грн., мінімальна заробітна плата 6000 гривень, а в погодинному розмірі – 36.11 гривень.

Заробітна плата не може бути меншою за мінімальний розмір оплати праці, який встановлений законодавством, а також повинна регулярно виплачуватись працівникові.

Табл. 4.1 – Кошторис витрат на проектування

Затрати	Сума, грн	Обґрунтування
1. Зарплата проектувальників.	25360	
2. Відрахування на соціальні потреби.	7650	

продовження таблиці 4.1

3. Контрагентські роботи і послуги.	-	не здійснювались
4. Витрати на відрядження.	4975	Відрядження керівника ДП
5. Інші прямі витрати.	16500	20МР камери, 2шт.
6. Усього прямих витрат.	7600	
7. Накладні витрати.	3040	
8. Планові накопичення.	4256	
9.Усього, кошторисна вартість проекту.	14896	
10. Податок на додану вартість.	2979,2	
11. Загальна договірна ціна розробки	22751,4	

Над розробкою даної системи працювало троє спеціалістів: . Оплата роботи працівників залежить від їх кваліфікації, їх кількості, та терміну для розробки даного проекту.

Створенням даної системи займалось троє спеціалістів: дизайнер, python-розробник, тестувальник. Оплата роботи працівників залежить від їх кваліфікації, кількості, та терміну для розробки даного проекту.

Табл. 4.2 – Розрахунок заробітної плати працівників

N	Посада	Оклад	Оподаткування	К-сть		Сума
п/п	виконавця	грн/міс	грн/міс	чол.	місяців	з/п, грн.
1	керівник ДП	6570	989	1	3	22150
2	python-розробник	12000	2200	1	3	28980

продовження таблиці 4.2

3	тестувальник	9000	1650	1	1	6440
4	дизайнер	18020	3510	1	1	14490
		Усього зарплати:				72060

Розрахунок посадової ставки проводиться шляхом обчислення добутку ставки першого розряду (в розмірі 2102 грн.) і тарифного коефіцієнту. Дробова частина результату не враховується.

Таблиці обрахунків подано у додатку Б, таблиці Б1-Б5.

4.3 Обґрунтування необхідності розробки

Виявлення та розпізнавання дорожніх знаків відіграє важливу роль у експертних системах, таких як системи керування дорожнім рухом та автоматичні системи водіння. Це моментально допомагає водіям або системам автоматичного водіння ефективно розпізнавати та розпізнавати дорожні знаки.

Завдяки даній системі можливо попереджати водіїв про перевищення швидкості, що може зберегти їх здоров’я та життя, а також інших людей.

На сьогоднішній день більшість таких технологій вбудовані в автомобілі високих класів, однак через надто високу ціну недоступні більшості водіїв.

Саме з цієї причини було обрано мету створення власної системи розпізнавання дорожніх знаків.

ВИСНОВКИ

Метою даного дипломного проєкту було дослідження питання комп'ютерного бачення з метою подальшої розробки програми розпізнавання дорожніх знаків.

В процесі виконання завдань проєкту було проаналізовано аналогічні системи, здійснено дослідження технології розпізнавання дорожніх знаків, вивчено основні аспекти, необхідні для реалізації даного проєкту. Також було розглянуто питання вибору засобів реалізації. Зокрема, було обрано мову програмування Python, оглянуто основні функції модуля Tensorflow та конволюційні нейронні мережі.

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

Додатково варто відмітити, що систему можна удосконалити шляхом покращення графічного користувацького інтерфейсу, а також реалізувати можливість розпізнавання дорожніх знаків напряму з відеопотоку під час руху автомобіля.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. У світі щороку в ДТП гинуть 1,25 млн людей — ООН. *Hromadske:* вебсайт. URL: https://hromadske.ua/posts/u-sviti-shoroku-v-dtp-ginut-125-mln-lyudej-oon (дата звернення 25.04.2020)

2. V2X Communication for Autonomous Driving. *AutoTalks:* вебсайт. URL: https://www.auto-talks.com/architected-for-autonomous/ (дата звернення: 15.05.2021).

3. Toyota Safety Sense TSS. *WestBury Toyota:* вебсайт. URL: https://westburytoyota.com/safety-sense (дата звернення: 04.05.2021).

4. Toyota Safety Sense: Preventive Safety Package-equipped Vehicles Top 10 Million Units Globally. *Toyota Newsroom:* вебсайт. URL: https://pressroom.toyota.com/toyota-safety-sense-preventive-safety-package-equipped-vehicles-top-10-million-units-globally/ (дата звернення: 04.05.2021).

5. One step closer to a world without accidents. Lexus safety. *Lexus:* вебсайт. URL: https://www.lexus.com/safety (дата звернення: 04.05.2021).

6. Python (programming language). *Lexus:* вебсайт. URL: https://en.wikipedia.org/wiki/Python_(programming_language) (дата звернення: 12.05.2021).

7. TensorFlow. *Wikipedia:* вебсайт. URL: https://en.wikipedia.org/wiki/TensorFlow (дата звернення: 13.05.2021).

8. Комп'ютерний зір. *Wikipedia:* вебсайт. URL: https://uk.wikipedia.org/wiki/ Комп%27ютерний_зір (дата звернення: 13.05.2021).

9. Real-Time Object Detection Using TensorFlow. *Great Learning:* вебсайт. URL: https://www.mygreatlearning.com/blog/object-detection-using-tensorflow/ (дата звернення: 13.05.2021).

10. Python - GUI Programming (Tkinter). *Tutorialspoint:* вебсайт. URL: https://www.tutorialspoint.com/python/python_gui_programming.htm (дата звернення: 15.05.2021).

11. Download PyCharm. *JetBrains:* вебсайт URL: https://www.jetbrains.com/pycharm/download/ (дата звернення 17.05.2021).

12. Global Autonomous Cars Market (2020 to 2030) - Opportunities and Strategies with COVID-19 Growth and Change. *Intrado:* вебсайт. 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: 03.05.2021).

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

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

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

16. Introduction to TensorFlow. *Tensorflow:* вебсайт. URL: https://www.tensorflow.org/learn (дата звернення: 11.05.2021)

17. Tensorflow Documentation. *Documentation:* вебсайт. URL: https://docs.wandb.ai/guides/integrations/tensorflow (дата звернення: 11.05.2021)

18. Python 3.9.5 documentation. *Python Documentation*: вебсайт. URL: https://docs.python.org/3/ (дата звернення: 11.05.2021)

19. 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

Реалізація програмних систем на Python

Лістинг А1 – 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)',
```

Продовження лістингу А1

```
            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',
```

Продовження лістингу А1

```
            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')
```

Продовження лістингу А1

```
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'))
```

Продовження лістингу А1

```
    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()
```

Лістинг А2 – 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
```

Продовження лістингу А2

```
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)
```

Продовження лістингу А2

```
_, 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):
```

Продовження лістингу А2

```
     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))
```

Продовження лістингу А2

```
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()
```

Лістинг А3 – 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()
```

Продовження лістингу А3

```
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)
```

Продовження лістингу А3

```
# 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()
```

Лістинг А4 – Функція initialize_gui

```
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()
```

Додаток Б

Техніко-економічні обчислення

Таблиця Б1 – Податок на доходи фізичних осіб

Посада	Обчислення згідно чинного законодавства в сфері податкування та середньої ринкової заробітної плати
Керівник ДП	7850×18%=1413 грн
Python-розробник	18000×18%=3240грн
Дизайнер	12000×18%=2160 грн
Тестувальлник	8000×18%=1440 грн

Таблиця Б2 – Військовий збір

Посада	Обчислення згідно чинного законодавства в сфері податкування та середньої ринкової заробітної плати
Керівник ДП	7850×1.5%=117.75 грн
Python-розробник	18000×1.5%=270грн
Дизайнер	12000×1.5%=180 грн
Тестувальлник	8000×1.5%=120 грн

Таблиця Б3 – Єдиний внесок

Посада	Обчислення згідно чинного законодавства в сфері податкування та середньої ринкової заробітної плати
Керівник ДП	7850×22%=1727 грн
Python-розробник	18000×22%=3960грн
Дизайнер	12000×22%=2640 грн
Тестувальлник	8000×22%=1760 грн

Таблиця Б4 – Утримання

Посада	Обчислення згідно чинного законодавства в сфері податкування та середньої ринкової заробітної плати
Керівник ДП	1530.75 грн (1413 грн+117.75 грн)
Python-розробник	3510 грн (3240 грн+270 грн)
Дизайнер	2340 грн (2160 грн+180 грн)
Тестувальник	1560 грн (1440 грн+120 грн)

Таблиця Б5 – Виплати працівникам

Посада	Обчислення згідно чинного законодавства в сфері податкування та середньої ринкової заробітної плати
Керівник ДП	6319.25 грн (7850 грн-1530.75 грн)
Python-розробник	14490 грн (18000 грн-3510 грн)
Дизайнер	9660 грн (12000 грн-2340 грн)
Тестувальник	6440 грн (8000 грн-1560 грн)