Державний торговельно-економічний університет

Кафедра комп’ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка автоматизованої системи для розпізнавання номерних знаків автомобілів»

Студента 2 курсу, 4м групи
спеціальності
122 «Комп’ютерні науки» _____ *підпис студента* Силенко Олександра Олегович

Науковий керівник
кандидат фізико-математичних наук _____ *підпис керівника* Філімонова Тетяна Олегівна

Гарант освітньої програми
доктор фізико-математичних наук, професор _____ *підпис керівника* Пурський Олег Іванович

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій

Кафедра комп'ютерних наук та інформаційних систем

Спеціальність 122 «Комп'ютерні науки»

Освітня програма «Комп'ютерні науки»

Затверджую

Зав. кафедри _____Пурський О.І.

«9» грудня 2022р.

Завдання

на випускну кваліфікаційну роботу студенту

Силенко Олександру Олеговичу

(прізвище, ім'я, по батькові)

1.Тема випускного кваліфікаційного проекту

«Розробка автоматизованої системи для розпізнавання номерних знаків автомобілів»

Затверджена наказом ректора від *«06» грудня 2022 р. № 3284*

2. Строк здачі студентом закінченого проекту *26 листопада 2023 року*

3. Цільова установка та вихідні дані до проекту

Мета роботи: розробка додатку для розпізнавання автомобільних знаків, а також навчання згорткової нейронної мережі для пришвидшення та поліпшення кількості корректних результатів.

Об'єкт дослідження: Проєктування додатку для розпізнавання автомобільних знаків.

Предмет дослідження: *Методи та моделі проєктування додатку та згорткової мережі для розпізнавання автомобільних знаків*.

4. Перелік графічного матеріалу _____
_____
_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.
2	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.
3	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП..... **Ошибка! Закладка не определена.**
РОЗДІЛ 1. РОЗПІЗНАВАННЯ АВТОМОБІЛЬНИХ ЗНАКІВ ЯК ОСНОВА БЕЗПЕКИ ДОРОЖНЬОГО РУХУ..... **Ошибка! Закладка не определена.**
1.1. Основні поняття та визначення розпізнавання автомобільних знаків . **Ошибка! Закладка не определена.**
1.2. Огляд існуючих систем..... **Ошибка! Закладка не определена.**
1.3 Методи розпізнаваня символів ..... **Ошибка! Закладка не определена.**
1.4. Висновки до розділу 1 ..... **Ошибка! Закладка не определена.**
РОЗДІЛ 2. ЗАСОБИ РОЗРОБКИ ..... **Ошибка! Закладка не определена.**
2.1. Огляд середовища розробки ..... **Ошибка! Закладка не определена.**
2.2. Мови програмування, фреймворки, розмітки та стилі **Ошибка! Закладка не определена.**

2.2.1. Мова програмування Python ..... **Ошибка! Закладка не определена.**
2.2.2. Бібліотека Sort ..... **Ошибка! Закладка не определена.**
2.2.3. Yolov8 ..... **Ошибка! Закладка не определена.**
2.2.4. Flask ..... **Ошибка! Закладка не определена.**
2.3. Опис програмної реалізації ..... **Ошибка! Закладка не определена.**
2.4. Модель згорткової мережі ..... **Ошибка! Закладка не определена.**
2.5. Висновки до розділу 2 ..... **Ошибка! Закладка не определена.**
РОЗДІЛ 3. РОЗРОБКА ІНТЕРФЕЙСУ ТА ФУНКЦІОНАЛУ ВЕБ ДОДАТКУ ..... **Ошибка! Закладка не определена.**
3.1. Робота з веб додатком ..... **Ошибка! Закладка не определена.**
3.2. Процесс опрацювання відео згортковою мережею **Ошибка! Закладка не определена.**
3.3. Висновки до розділу 3 ..... **Ошибка! Закладка не определена.**
ВИСНОВКИ ТА ПРОПОЗИЦІЇ ..... **Ошибка! Закладка не определена.**
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ..... **Ошибка! Закладка не определена.**
ДОДАТКИ ..... **Ошибка! Закладка не определена.**

7. Календарний план виконання роботи.

№ По р.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*01.11.2022*	*01.11.2023*
2	*Розробка та затвердження завдання для випускної кваліфікаційної роботи*	*06.12.2022*	*05.12.2023*
3	*Вступ*	*01.06.2023*	*01.06.2023*
4	*РОЗДІЛ 1. Розпізнавання*	*25.06.2023*	*25.06.2023*

	автомобільних знаків як основа безпеки дорожнього руху.		
5	*РОЗДІЛ 2. Засоби розробки.*	*02.09.2023*	*02.09.2023*
6	*Підготовка статті у збірник наукових статей магістрів*	*09.09.2023*	*09.09.2023*
7	*РОЗДІЛ 3. Розробка інтерфейсу та функціоналу веб додатку.*	*21.10.2023*	*21.10.2023*
8	*Висновки*	*02.11.2023*	*02.11.2023*
9	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*05.11.2023*	*05.11.2023*
10	*Попередній захист випускної кваліфікаційної роботи*	*20.11.2023*	*20.11.2023*
11	*Виправлення зауважень, зовнішнє рецензування випускного кваліфікаційної роботи*	*22.11.2023*	*22.11.2023*
12	*Представлення готової зшитої випускної кваліфікаційної роботи*	*26.11.2023*	*26.11.2023*
13	*Публічний захист випускної кваліфікаційної роботи*	*Згідно роботи ЕК*	

8. Дата видачі завдання «15» грудня 2023 р.

9. Керівник випускної кваліфікаційної роботи. Філімонова Т.О._____

(прізвище, ініціали, підпис)

10. Гарант освітньої програми. Пурський О.І._____

(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник.

Силенко О.О._____

(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи

У випускній кваліфікаційній роботі розроблено згорткову нейронну мережу для розпізнавання номерних знаків. Побудовані графіки функції втрат і точності.

Розроблено web-сервіс для розпізнавання номерів авто.

Робота оформлена згідно з вимогами. Всі поставлені завдання виконані. Робота
може бути допущена до захисту._____
_____
_____
_____
_____
_____
_____
_____
_____
_____
_____

Керівник випускної кваліфікаційної роботи

_____

(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу

Випускний кваліфікаційний проект студента Силенко О.О._____
(прізвище, ініціали)
може бути допущений до захисту в екзаменаційній комісії.
Гарант освітньої програми_____Пурський О.І.___
_____

(підпис, прізвище, ініціали)

Завідувач кафедри_____Пурський О.І._____
(підпис, прізвище, ініціали)
«_____»_____2023 р.

ЗМІСТ

ВСТУП 10
РОЗДІЛ 1. РОЗПІЗНАВАННЯ АВТОМОБІЛЬНИХ ЗНАКІВ ЯК ОСНОВА БЕЗПЕКИ ДОРОЖНЬОГО РУХУ 12
1.1. Основні поняття та визначення розпізнавання автомобільних знаків 12
1.2. Огляд існуючих систем 15
1.3 Методи розпізнаваня символів 16
1.4. Висновки до розділу 1 19
РОЗДІЛ 2. ЗАСОБИ РОЗРОБКИ 21
2.1. Огляд середовища розробки 21
2.2. Мови програмування, фреймворки, розмітки та стилі 22
2.2.1. Мова програмування Python 22
2.2.2. Бібліотека Sort ё23
2.2.3. Yolov8 23
2.2.4. Flask 24
2.3. Опис програмної реалізації 25
2.4. Модель згорткової мережі 28
2.5. Висновки до розділу 2 30
РОЗДІЛ 3. РОЗРОБКА ІНТЕРФЕЙСУ ТА ФУНКЦІОНАЛУ ВЕБ ДОДАТКУ 31
3.1. Робота з веб додатком 31
3.2. Процес опрацювання відео згортковою мережею 34
3.3. Висновки до розділу 3 39
ВИСНОВКИ ТА ПРОПОЗИЦІЇ 40
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 41
ДОДАТКИ 42

Анотація

У випускній кваліфікаційній роботі здійснено комплексну розробку моделей нейронної згорткової мережі, а також розробка веб додатку для неї. Теоретично обґрунтовано основні положення формування і проведення розробки саме цією мовою програмування, а також показано про усі технології та бібліотеки що були використані при розробці. Розроблено веб додаток для знайомства з роботою нейронної згорткової мережі яка була навчена на цій роботі, а також показані її можливості та статистика.

Ключові слова: Згорткова нейронна мережа, веб додаток, розпізнавання автомобільних знаків, Yolov8.

Abstract

In the final qualification work, the complex development of neural convolutional network models was carried out, as well as the development of a web application for it. The basic provisions of forming and carrying out development in this programming language are theoretically substantiated, and all the technologies and libraries that were used in the development are also shown. A web application has been developed to familiarize yourself with the work of the neural convolutional network that was trained in this work, as well as its capabilities and statistics are shown.

Keywords: Convolutional neural network, web application, car sign recognition, Yolov8.

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

YOLOv8 (You Only Look Once) version 8.

SOTA в області інформатики та досліджень зв'язаних із штучним інтелектом відомий як «станом мистецтва» (англ. State of the Art). Це вираз використовується для позначення поточного рівня досягнень, прогресу чи технічного розвитку в конкретній галузі чи конкретному дослідженні.

CNN або згорткова нейронна мережа (Convolutional Neural Network).

ВСТУП

На даний час безпека дорожнього руху та усіх його учасників є однією з найважливіших у суспільстві. Ми не можемо уявити наше життя без автомобіля, кожен день ми зіштовхуємося з ними у різних сферах життя. І як кожна людина усі бажають щоб усе було планомірно та у межах правил за якими ми живемо. Тому правила дорожнього руху повинні виконуватися усіма учасниками, аякщо ні то порушники повинні нести відповідальність за свої дії, бо кожен хоче відчувати себе у безпеці.

Тому для полегшення роботи служб було вирішено розробити застосунок що буде по відео фіксувати усі автомобілі які являются учасниками дорожнього руху, а також записувати іх актуальні дані.

Також даний застосунок можно використовувати для ведення звітності по автомобільному транспорту що в'їзджає на тереторію підприємства, банку, ресторану, парковки, тощо.

Актуальність. Розробка програм для розпізнавання автомобільних номерів за допомогою згорткових мереж залишається актуальною. Це сприяє покращенню безпеки дорожнього руху, виявленню злочинців, автоматизації паркування та управління транспортом, а також безконтактній оплаті. Технологія CNN дозволяє автоматично розпізнавати номери на зображеннях, а розробники постійно прагнуть покращити їх точність та швидкість.

Мета дослідження: розробка додатку для розпізнавання автомобільних знаків, а також навчання згорткової нейронної мережі для пришвидшення та поліпшення кількості корректних результатів.

Об'єкт дослідження: Проєктування додатку для розпізнавання автомобільних знаків.

Предмет дослідження: Методи та моделі проєктування додатку та згорткової мережі для розпізнавання автомобільних знаків.

У відповідності з метою дослідження поставлено *завдання*:

дослідження предметної галузі;

порівняльний аналіз існуючих програмних рішень;

вибір інструментальних засобів та середовища розробки;

розробка схеми даних додатку;

розробка алгоритму роботи додатку;

розробка користувацького інтерфейсу додатку;

програмна реалізація додатку;

Методи дослідження*:* Аналіз методів розпізнавання, дослідження датасетів для навчання нейромережі.

Наукова новизна дослідження розробка веб-додатку що дозволить користувачам швидко розпізнавати номерні знаки автомобілів.

Практичним значенням дослідження є розробка веб додатку що спростить людям отримання даних з камер відео спостереження.

Публікації. Результати дослідження опубліковано у збірнику наукових статей студентів, які здобувають освітній ступінь магістра за спеціальністю «Комп'ютерні науки» ДТЕУ. Розробка автоматизованої системи для розпізнавання номерних знаків автомобілів // Прикладні комп'ютерні технології: зб. наук. ст. студ. / відп. ред.— Київ : Держ. торг.-екон. ун-т, 2023. – С. 112-117.

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із 9 найменувань, додатків і містить 41 сторінки основного тексту та 27 рисунків.

РОЗДІЛ 1.
РОЗПІЗНАВАННЯ АВТОМОБІЛЬНИХ ЗНАКІВ ЯК ОСНОВА БЕЗПЕКИ ДОРОЖНЬОГО РУХУ

1.1. Основні поняття та визначення розпізнавання автомобільних знаків

У сучасному світі неможливо уявити контроль за автомобільним потоком, що постійно збільшується, без технології розпізнавання реєстраційних знаків автомобілів. Ідентифікація автомобілів за їх номерним знаком є важливим аспектом забезпечення безпеки дорожнього руху.

Одним із підходів до розпізнавання зображень символів автомобільного номера є використання згорткових нейронних мереж. Методика розпізнавання номера складається з п'яти етапів – пошук номерної пластини на фотографії, виділення на знімку символів номера, розбиття номера на символи, розпізнавання кожного з цих символів, та оцінка надійності розпізнавання кожного символу. Алгоритми розпізнавання номерних знаків автомобілів мають бути стійкими до спотворень зображень номерних пластин, пов'язаних зі швидкістю руху автомобілів та положенням камери щодо номерного знака.

До цього часу багато компаній для навчання нейронної мережі, що розглядається, використовувалися фотографії, отримані за допомогою реальних зйомок. Ці фотографії відбиралися та оброблялися вручну, зокрема, на кожному знімку мало бути лише одне зображення номерного знака. Крім того, ім'я кожного файлу має містити номер, зображений на фотографії. Такий процес приводив до великих витрат часу створення навчальної вибірки.

Однією з проблем розпізнавання автомобільних номерів є різноманітність їх видів, які відрізняються зображенням символів і розмірами номерних пластин (Рис. 1.1). Для навчання нейронної мережі потрібна якісна та велика вибірка зображень автомобільних номерів. Але іноді важко отримати достатню кількість прикладів зображень номерних знаків певного виду за допомогою реальних зйомок. У зв'язку з цим у роботі було вирішено протестувати підхід з машинним

навчанням на зображеннях номерних знаків, які згенеровані програмно.

Рис 1.1 Зразки українських номерних знаків.

Один із важливих аспектів розпізнавання полягає в тому, що алгоритми розпізнавання державних реєстраційних знаків автотранспорту мають бути стійкими до спотворень зображень номерних пластин, пов'язаних зі швидкістю руху автомобілів, положенням камери щодо номерного знака, а також його забруднення.

Також ще однією з проблем розпізнавання автомобільних номерів є різноманітність їх типів, що відрізняються зображенням символів, фоном або розміром номерних пластин. Дані машинного навчання повинні включати велику кількість зображень номерів різних типів. Що завдяки проєкту Nomeroff.net зараз не э проблемою. Була використана їх база з кількома тисячами реальних та згенерованих автомобільних знаків для навчання даної згорткової мережі.

Згорткова нейронна мережа, CHC, CNN – основний інструмент для класифікації та розпізнавання об'єктів, обличь на фотографіях, розпізнавання мови. Є безліч варіантів застосування CNN, такі як Deep Convolutional Neural Network (DCNN), RegionCNN (RCNN), Fully_Convolutional_Neural_Networks (FCN N), Mask R-CNN та інші.

Згорткова нейронна мережа за рахунок застосування спеціальної операції – власне згортки – дозволяє водночас зменшити кількість інформації, що зберігається в пам'яті інформації, за рахунок чого краще справляється з картинками більш високої роздільної здатності, і виділити опорні ознаки зображення, такі як ребра, контури або грані. На наступному рівні обробки з цих ребр і граней можна розпізнати повторювані фрагменти текстур, які далі можуть скластися в фрагменти зображення.

За допомогою систем технічного зору та нейромережі, такої як YOLOv8, можна швидко та точно виявляти предмети або автомобілі. YOLOv8 — це один із найпопулярніших алгоритмів виявлення об'єктів у реальному часі, який використовує глибокі нейронні мережі для класифікації та локалізації об'єктів. YOLOv8 може виявляти об'єкти на зображеннях та відео з високою швидкістю та точністю. Ця модель може працювати на різних апаратних платформах, включаючи мобільні пристрої та комп'ютери. Коли системи технічного зору оснащені нейромережами YOLOv8, вони можуть оперативно реагувати на автомобілі що рухаються дорогою. Алгоритм YOLOV8 заснований на архітектурі згорткових нейронних мереж і використовує методи навчання з учителем. Ця модель приймає зображення як вхідні дані та видає оцінки ймовірності того, що на зображенні присутній певний об'єкт у режимі реального часу. Для цього YOLOV8 використовує методи визначення областей інтересу (ROI), що дозволяють визначити області зображення, на яких можуть бути розташовані об'єкти. Загальними математичними методами, які використовуються в YOLOV8, є методи згорткової нейронної мережі, навчання з учителем та оптимізації функції втрат. Вона ґрунтуються на алгоритмах глибокого навчання, таких як стохастичний градієнтний спуск та зворотне поширення помилки.

1.2. Огляд існуючих систем

Одним з найкращих open-source рішень для українських, європейських та снг країн є Nomeroff.Net.

Nomeroff.Net є проєктом заснованим програмістами такого відомого сайту як авторіа. Вони надихнулися тим що МВС України відкрило доступ до державних реєстрів номерів, що дозволяло отримувати більш детальну інформацію про автомобільний транспорт що розміщаєтеся на сайті за його номером. Тому за кілька років навчаючись на величезній кількості даних, проект Nomeroff.Net досяг 97% вірності у розпізнавані українських номерів. Серед Плюсів насамперед можна виділити те що це безкоштовний проєкт, а також те що є версія для приладів з слабкою відеокартою.

Додаток SecurOS™ Auto License Plate Recognition (LPR/ANPR) надає користувачам численні безпрецедентні переваги, зокрема здатність точно фіксувати інформацію про номерні знаки на швидкості 155 миль/год (250 км/год) і за будь-яких погодних умов, включаючи туман, дощ, і сніг.

SecurOS™ Auto використовує вдосконалене глибоке навчання та алгоритми на основі шаблонів, щоб забезпечити високу точність, яка відрізняє літери від цифр, щоб, наприклад, «8» не було помилково прийнято за «B», що є важливою перевагою, особливо в ситуаціях, коли один має лише мілісекунди, щоб зробити це правильно.

SecurOS™ Auto підтримує номерні знаки з більшості країн світу, має варіанти як низької, так і високої швидкості, підтримує кілька смуг руху з 1 камерою та багато інших функцій. SecurOS™ Auto легко інтегрується зі сторонніми системами керування паркуванням або розумними дорожніми системами, а також зі старим обладнанням безпеки та зовнішніми базами даних. Відчуйте потужність розпізнавання номерних знаків (LPR/ANPR) із SecurOS™ Auto, вашим надійним вибором для технології розпізнавання номерних знаків.

Можливості системи:

• Розпізнавання номерних знаки понад 60 країн світу;

• Розпізнавання номерів по окремих кадрів без використання відео;

• Швидко налаштувати оновлення для розпізнавання номерів нових стандартів;

• Формувати базу даних розпізнаних номерів зі збереженням супутньої інформації про дату, час і місце виявлення автомобіля, швидкості і напрямку його руху, а також посилання на відео фрагмент;

• Організувати пошук розпізнаних раніше номерів по заданих параметрам;

• Налаштувати необхідні реакції системи на розпізнавання номера, на результати пошуку;

• Оперативно інформувати оператора (підтримується, в тому числі, і голосове сповіщення) і / або відправити повідомлення (e-mail, SMS та ін.) зовнішнім службам про результати розпізнавання номера і / або про результати зіставлення даних про автомобіль зі списками номерів;

• Автоматично генерувати звіти різних видів на основі результатів розпізнавання, пошуку по базах даних.

Можливості інтеграції та спільне використання:

• Стаціонарне і мобільне обладнання спостереження й прилади контролю а обліку: бар'єри та шлагбауми, автоматичні ворота, радари, телекомунікаційні мережі, термінали оплати та ін.;

• Сторонні програмні комплекси (автоматизовані системи розрахунку оплати, СУБД та ін.);

• Інтеграція з «SecurOS Traffic Scanner» (системою автоматичного визначення фактів порушення правил дорожнього руху) – можливість побудувати потужний комплексне рішення для контролю і управління ситуацією на дорогах.

1.3 Методи розпізнаваня символів

1. Tesseract OCR.

Це програмне забезпечення з відкритим кодом, яке реалізує автоматичне розпізнавання умовних позначень (символів) і повного тексту. Краса Tesseract полягає в тому, що це програма доступна для всіх операційних систем, проста у вивченні та прогнозована. Але мінусом являться робота алгоритму з розірваним, брудного та спотвореним текстом. Приблизна ймовірність розпізнавання автомобільних номерів лежить у 20-30%.

Рис. 1.2 Tesseract OCR

2. k-nearest.

Цей метод розпізнавання не зважаючи на простоту використання часто дає хороші результати.

Як це працює:

Певна кількість зображень була записана заздалегідь, та правильно розділена на класи.

Введено вимірювання відстані між символами (якщо зображення двійкове). типу, операція XOR є оптимальною.)

При спробі розпізнати літери відстані обчислюються одна за одною. Між розпізнаним символом і всіма базовими символами. Серед найближчих сусідів

вони можуть бути представники різних класів. Символу присвоюється той самий клас, що й у більшості із найближчих символів.

Недоліком цього методу є необхідність у швидкості. Обчислити відстань між зображеннями та перетворити їх у двійкові та використання XOR. Бінаризація змінює символи абсолютно непередбачуваними способами. Брудні або зношені номери можуть спричинити проблеми. Однак, якщо існує велика база даних прикладів символів, записаних за різних умов, теоретично k-nearest – це те, що потрібно.

3. Нейронна мережа.

Штучна нейронна мережа (ШНМ, нейронна мережа) — це група нейронів, Поєднаних між собою. Робота нейронної мережі полягає в перетворенні вхідних даних перетворення векторів вхідних даних на вихідні вектори.

Майже будь-яке завдання можна звести до завдання, яке вирішує нейронна мережа. Щоб нейронна мережа почала працювати, її потрібно навчити. Для цього в нейроні записується певний стандарт, а потім подається вхідний сигнал, порівняння з базовою лінією та різниця між базовою лінією та вхідними даними додається до стандарту. Чим більше вхідних даних, тим краще нейронна мережа навчається.

Переваги:

• За умови належного встановлення та навчання працює краще інших методів;

• При великій кількісті тренувальних даних з’являється певна стійкість до спотворення символів.

Недоліки:

• Складність навчання;

• Неможливо діагностувати аномальну поведінку в багаторівневих мережах.

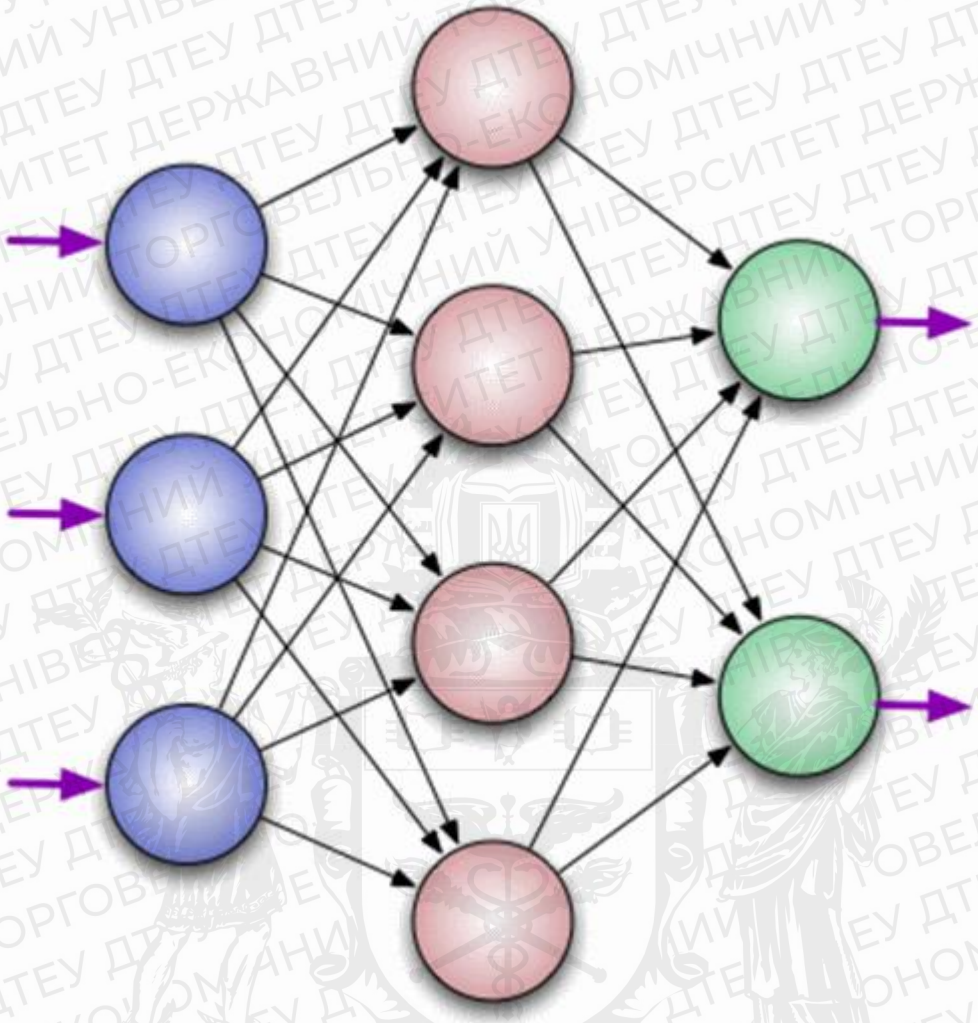

Рис. 1.3. Приблизна схема роботи нейронної мережі.

1.4. Висновки до розділу 1

В Україні та у світі в цілому створено багато платних та безкоштовних проєктів для розпізнавання номерів автомобільного транспорту. Такі систему впроваджуються як фірмами так і на державному рівні.

В даний час це дуже перспективна галузь яка все ще розвивається тому саме час розвиватися у сфері нейронних систем та розпізнавання. Розпізнавання по камерам допоможе не лише у вирішенні проблем на дорогах, а й у побутових рішеннях, наприклад в'їзд та виїзд на тереторію банку.

На українському ринку багато різних додатків та рішень на дану тему, це створює конкуренцію та бажання розвинутися краще за конкурента що покращує якість для користувача та клієнта.

Аналіз алгоритмів і програмного забезпечення на ринку.

Вартість на 1500-3000 доларів та вище. Твердження щодо точності ідентифікації часто перебільшені, у тестах надається надійно розпізнана лише демо-версія Чіткі цифри та висока контрастність. В результаті заявлена точність 90-98%, фактична ж - 80-87%. Опубліковано, лише деякі компанії назвали їх види. для визнання. Зазвичай використовуються нейронно-подібні алгоритми та шаблонні алгоритми. Важливим обмежуючим фактором у використанні системи є максимальна швидкість автомобілів на яких програма може знаходити та ідентифікувати номерні знаки транспортних засобів, що рухаються. По-перше, на це впливає спосіб кріплення камери - висота і нахил; і по-друге, швидкість обробки зображень автомобіля.

Тому розробка системи являється доцільною на сьогодні.

РОЗДІЛ 2.
ЗАСОБИ РОЗРОБКИ

2.1. Огляд середовища розробки

Програмування в PyCharm має свої переваги, і ось кілька реальних прикладів, чому варто використовувати цей інструмент, особливо для студентів адже для них компанія JetBrains надає безкоштовну ліцензію з якою програмувати саме задоволення.

Спростити проектну роботу:

Приклад: ви можете легко створювати та керувати проектами в PyCharm. Завдяки зручному інтерфейсу ви можете швидко додавати файли, бачити структуру проекту та легко перемикатися між різними частинами коду.

Автоматичне усунення несправностей і налагодження:

Приклад: PyCharm надає функцію автоматичного виявлення помилок під час написання коду. Наприклад, якщо ви використовуєте неправильний синтаксис або визначите змінну, яка не існує, PyCharm виділить це та дасть вам підказки, як це виправити.

Автозаповнення коду:

Приклад: під час написання коду PyCharm забезпечує функцію автозавершення, яка значно покращує продуктивність. Наприклад, ви можете швидко вибрати доступні методи або властивості об’єкта, ввівши лише першу літеру.

Інтеграція з системами контролю версій:

Приклад: інтеграція Git у PyCharm дозволяє легко відстежувати зміни коду, фіксувати, розгалужувати та вирішувати конфлікти через простий у використанні інтерфейс.

Підтримка віртуальних середовищ:

Приклад. Ви можете легко налаштувати віртуальні середовища Python і керувати ними безпосередньо з PyCharm. Це дозволяє вам ефективно керувати залежностями вашого проекту.

Аналіз та реконструкція програмного коду:

Приклад: PyCharm допомагає проаналізувати код і визначити потенційні покращення. Наприклад, ви можете використовувати функції рефакторингу для автоматичного перейменування змінних, методів або класів.

Підтримка інструментів веб-розробки:

Приклад. Якщо ви розробляєте веб-проект, PyCharm може надати вам інструменти для роботи з HTML, CSS і JavaScript. Крім того, ви можете легко налаштувати інші інструменти веб-розробки та фреймворки.

Ці приклади показують, як PyCharm полегшує роботу розробників і надає широкий спектр функцій для підвищення продуктивності.

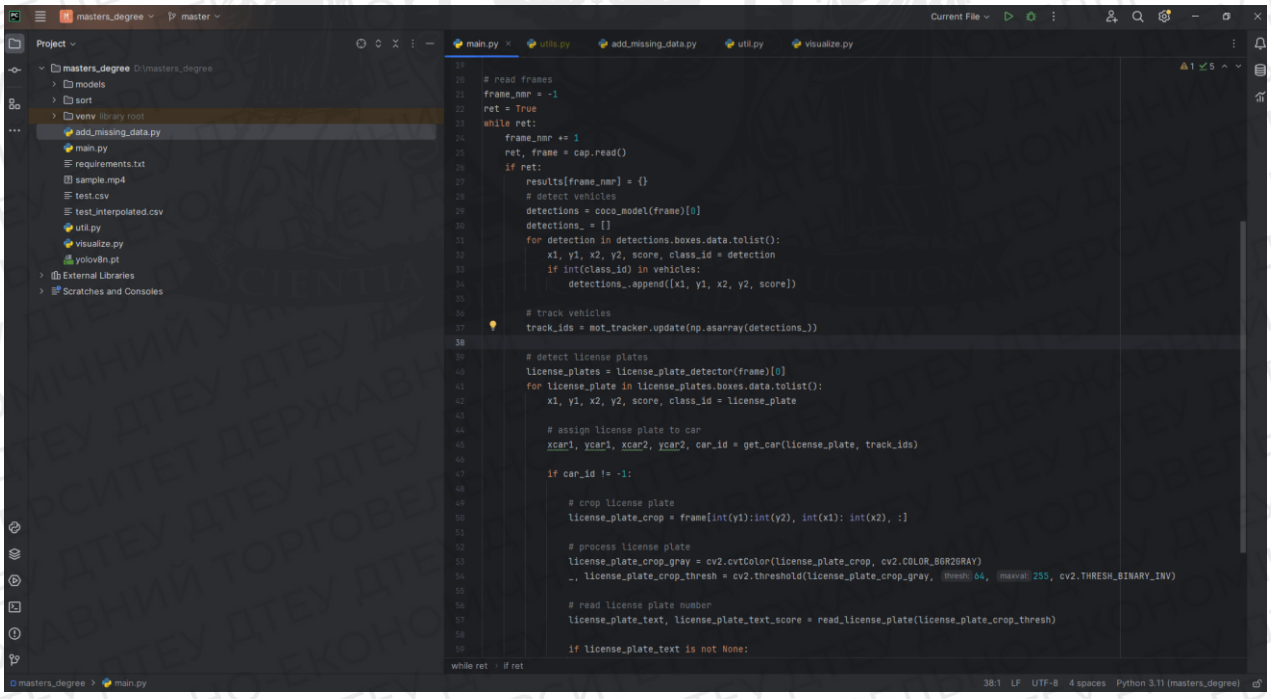

Рис 2.1. IDE PyCharm

2.2. Мови програмування, фреймворки, розмітки та стилі

2.2.1. Мова програмування Python

Python — потужна мова програмування, яку легко освоїти. Це дозволяє ефективно використовувати високорівневі структури даних і забезпечує простий, але ефективний підхід до об'єктно-орієнтованого програмування. Поєднання елегантного синтаксису та динамічної типізації інтерпретованої мови робить Python ідеальною мовою для прискореного створення сценаріїв і розробки додатків у різноманітних галузях і платформах.

Інтерпретатор Python і стандартна бібліотека доступні у вільному доступі як вихідний код і двійкові файли для всіх основних платформ на офіційному веб-сайті Python і можуть вільно поширюватися. Сайт також містить дистрибутиви та посилання на різні сторонні модулі для Python, різні програми та інструменти, а також додаткову документацію. 28 Інтерпретатор Python можна легко розширити новими функціями та типами даних, написаними на C/C++ (або іншими мовами, доступними на C). Python також можна використовувати як мову розширення для спеціальних програм.

2.2.2. Бібліотека Sort

SORT — це базова реалізація візуальної системи відстеження кількох об’єктів на основі елементарних методів асоціації даних і оцінки стану. Він розроблений для додатків онлайн-відстеження, де доступні лише минулі та поточні кадри, а метод виробляє ідентифікатори об’єктів на льоту. Хоча цей мінімалістичний трекер не обробляє оклюзію або повторне введення об’єктів, його мета — служити базовою лінією та випробувальним майданчиком для розробки майбутніх трекерів.

Дуже цікава бібліотека яка є одним з ключових стовпів даної програми, дозволяє відстежувати кілька об’єктів одночасно наприклад автомобіль плюс номерний знак, що запобігає спотворенню даних внесенням автомобільного знаку без автомобіля.

2.2.3. Yolov8

YOLO (You Only Look Once), популярна модель виявлення об’єктів і сегментації зображення, розроблена Джозефом Редмоном і Алі Фархаді з Вашингтонського університету. Випущений у 2015 році, YOLO швидко завоював популярність завдяки своїй високій швидкості та точності.

YOLOv8 — остання версія YOLO від Ultralytics. Будучи найсучаснішою моделлю (SOTA), YOLOv8 спирається на успіх попередніх версій, представляючи нові функції та вдосконалення для підвищення продуктивності, гнучкості та ефективності. YOLOv8 підтримує повний спектр завдань штучного інтелекту зору, включаючи виявлення, сегментацію, оцінку пози, відстеження та класифікацію. Ця універсальність дозволяє користувачам використовувати можливості YOLOv8 у різноманітних програмах і доменах.

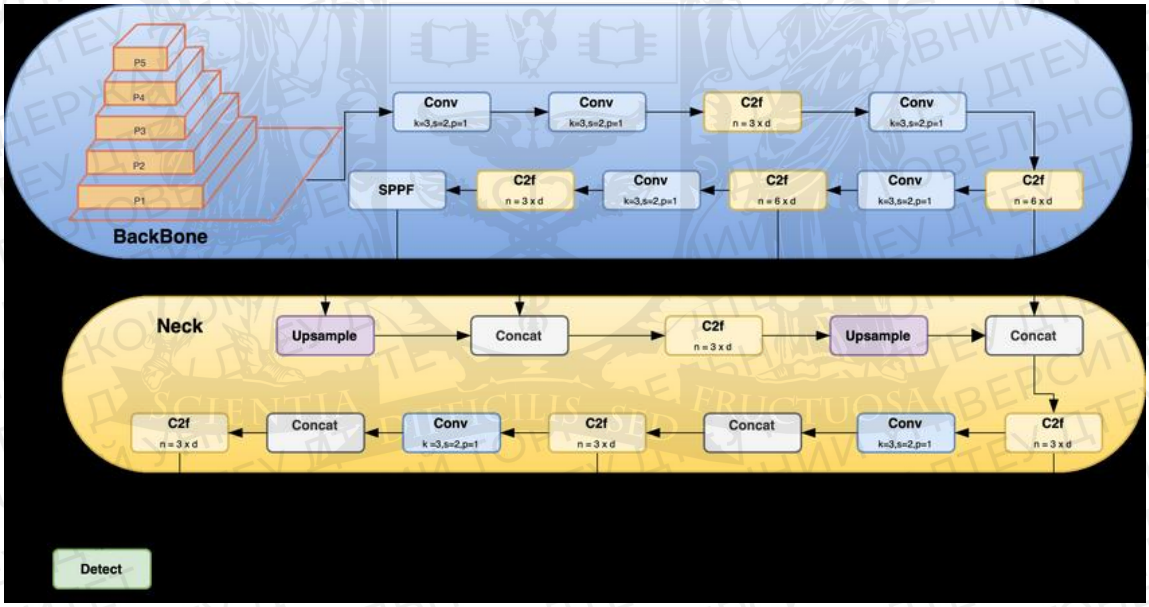

Рис. 2.2. Архітектура Yolov8

2.2.4. Flask

Ніхто не буде сперечатися що фреймворк джанго є набагато кращим та зручнішим для розробки, але фласк також не втрачає своїх позицій за займає свою позицію не зважаючи на зовнішні подразники. Обидва фреймворки мають свої переваги, і правильний вибір залежить від конкретної ситуації. Ось кілька причин, чому Flask краще підходить для невеликих проектів:

Простота використання: Flask — це мікро фреймворк, який надає мінімальний набір інструментів для створення веб-додатків. Якщо ваш проект невеликий і не потребує всіх функцій, які пропонує Django, Flask може бути легшим у вивченні та використанні.

Гнучкість: Flask дає вам велику свободу у виборі інструментів і бібліотек для вашого проекту. Замість того, щоб дотримуватися жорсткої структури, як Django, ви можете вибрати компоненти, які найкраще відповідають вашим потребам.

Прозорість: Flask забезпечує більш прозору структуру, що корисно для людей, які люблять контролювати структуру свого коду та програм.

Швидкість розробки: у деяких випадках Flask може забезпечити більш високу швидкість розробки завдяки своїй простоті та мінімалістичному підходу. Він не пропонує стільки функцій, але може бути перевагою для невеликих проектів.

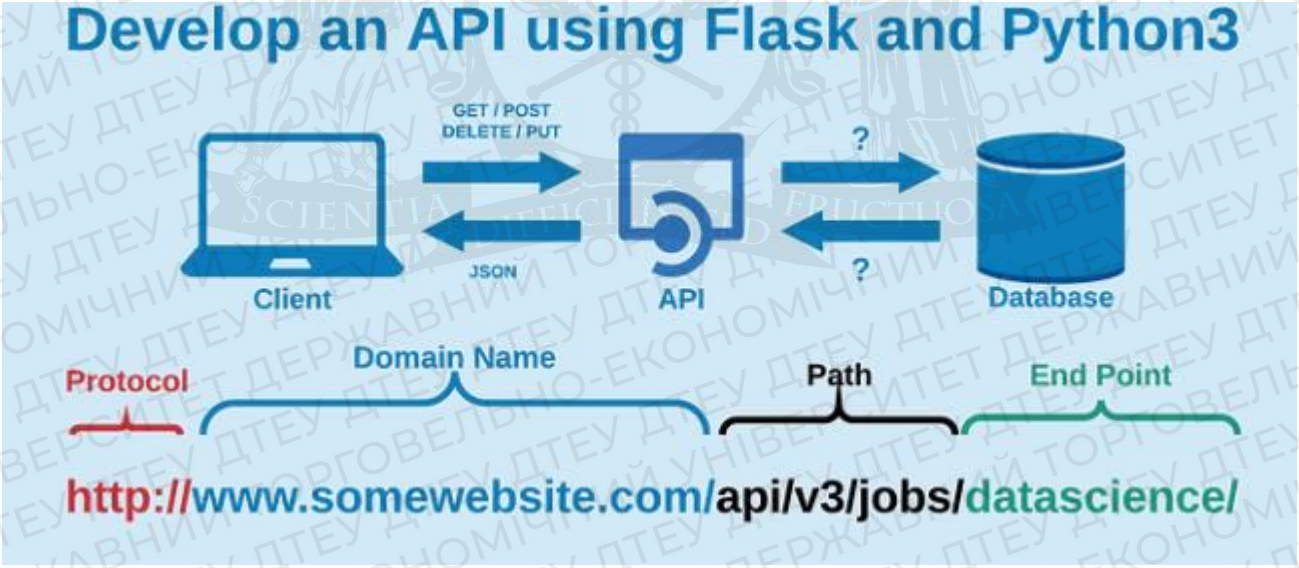

Рис. 3.1 Flask

2.3. Опис програмної реалізації

Опис алгоритму.

1. Обробка вхідних відео та отримання набору кадрів;

2. Сегментація кадрів, виділення областей з номерним знаком та автомобілем;

3. Розпізнавання номерного знаку з кожної виділеної області;

4. Опрацювання та аналіз даних з кожного відео потоку;

5. Інтерполяція значень для покращення результатів виводу даних;

6. Відображення номерного знаку на поточному кадрі;

7. Створення результуючого відео з опрацьованих кадрів;

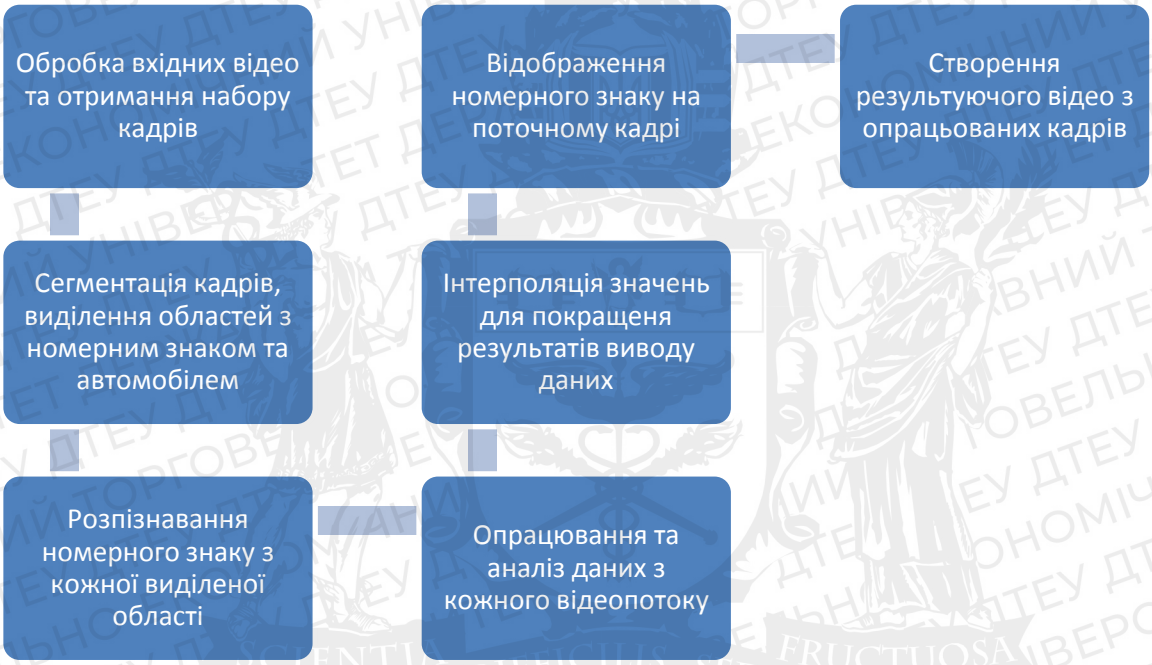

Рис.2.1. Опис програмної реалізації

YOLO — це алгоритм виявлення в реальному часі. Це один із найефективніших алгоритмів виявлення об’єктів, який охоплює багато завдань. Інноваційні ідеї дослідницької спільноти з інформатики. Виявлення об'єктів є однією з класичних проблем комп'ютерного зору. Це алгоритм, який визначає, які об’єкти є на зображенні а також де вони знаходяться. Проблема виявлення об'єктів більш складна аніж класифікація, яка також може розпізнавати об’єкти, але не вказує, де вони знаходяться знаходиться на фото. Класифікація також не працює для зображень що містить кілька об'єктів.

YOLO популярний, тому що він може реалізувати та працювати з високою точністю. Алгоритм «знаходить тільки один раз» у тому значенні, що вам

потрібен лише один прохід через нейронну мережу, щоб зробити прогноз. Тоді розпізнані об’єкти відображаються разом із їхніми обмежувальними полями.

За допомогою YOLO CNN прогнозує кілька границь одночасно та кожен клас має особливості. Ця модель має ряд переваг перед іншими засобами ідентифікації об'єктів:

• Працює дуже швидко.

• Все зображення видиме та неявно закодоване під час навчання та тестування

• Отримайте загальні уявлення щодо предметів та досліджує їх. А також тестуючись на підроблених даних, алгоритм перевершує інші методи виявлення. YOLO розбиває вхідне зображення на сітку SxS. Коли центр предмета падає на комірку сітки то об’єкт автоматично прив’язується до неї.

YOLO виконує класифікацію та локалізацію для кожної комірки сітки SxS одночасно для всіх. Оскільки нейронні мережі можуть судити лише за класифікацією та локалізацією, тобто кожна комірка сітки може містити лише один об’єкт. Через це сітки YOLO стикається з кількома проблемами:

1. Через використання сітки SxS і визначати кожну сітку окремо максимальна кількість об’єктів, які модель може виявити в сітці, становить одну.

2. Шаблон не може виконати це завдання, якщо клітинка сітки містить кілька об’єктів. Ідентифікація їх усіх є проблемою ідентифікації об’єктів, яка страждає від YOLO.

3. Об’єкт може знаходитися у кількох сітках одночасно і шаблонів визначить один і той самий об’єкт кілька разів (у кількох сітках).

YOLO вважається найкращим, оскільки всі комірки сітки SxS визначаються одночасно. Кожна клітинка сітки SxS передбачає межу, для кожної межі модель дає оцінку точності вимірювання. Оцінка вказує наскільки модель впевнена, що об’єкт знаходиться в полі. Цей прапорець можна використовувати, щоб запобігти виявленню фону моделлю. Якщо в комірці немає об'єктів, то оцінка має дорівнювати 0.

Окрім цього, модель генерує чотири числа ((x,1 y1), x2, y2) для відображення розташування та розмір запропонованої границі поля.

Координати (x1, y1) представляють центр вікна відносно краю клітинки сітки. Ширина та висота 0 <(x,1 y1, x2, y2) <1, оскільки вони проектуються відносно всього зображення.

2.4. Модель згорткової мережі

На графіках нижче наведені результати навчання моделі за 200 епох. За результатами можна побачити що для тестової вибірки точність 0.91, функція втрат 0.32 для тренувальної вибірки точність 0.9, функція втрат 0.035.

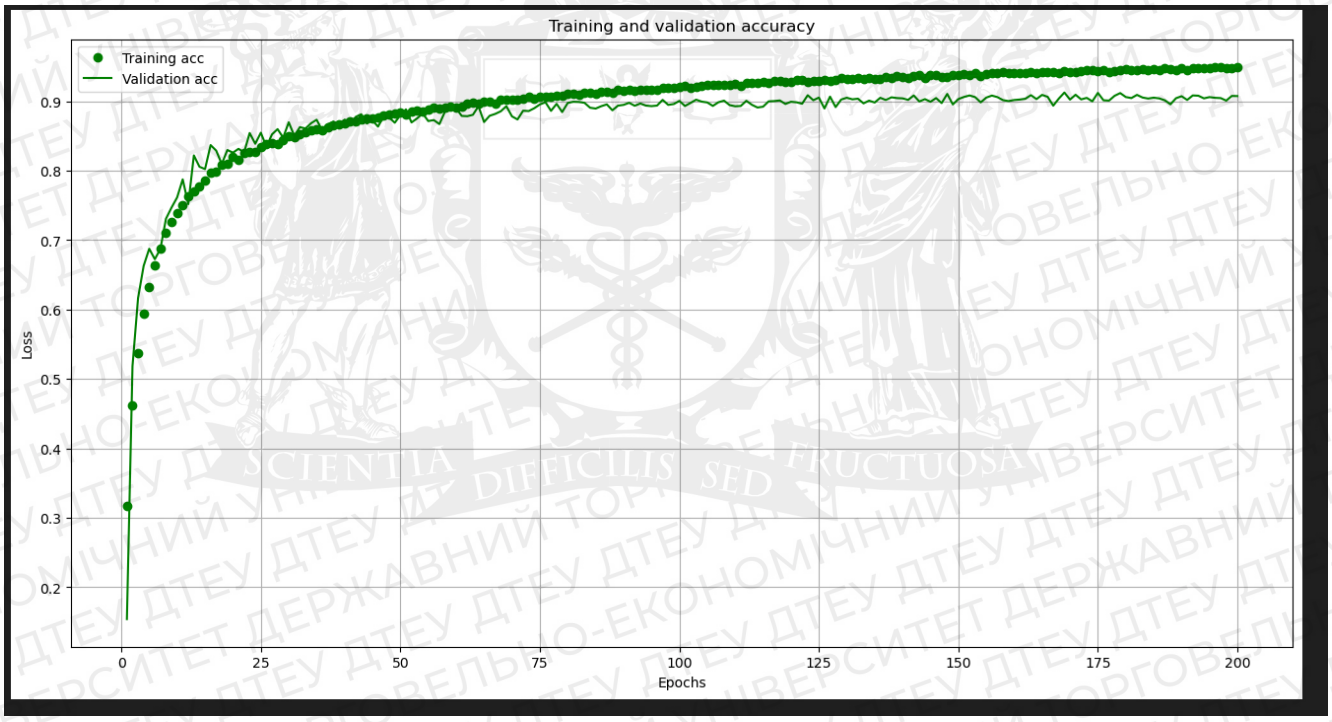

Рис.2.2. Графік точності

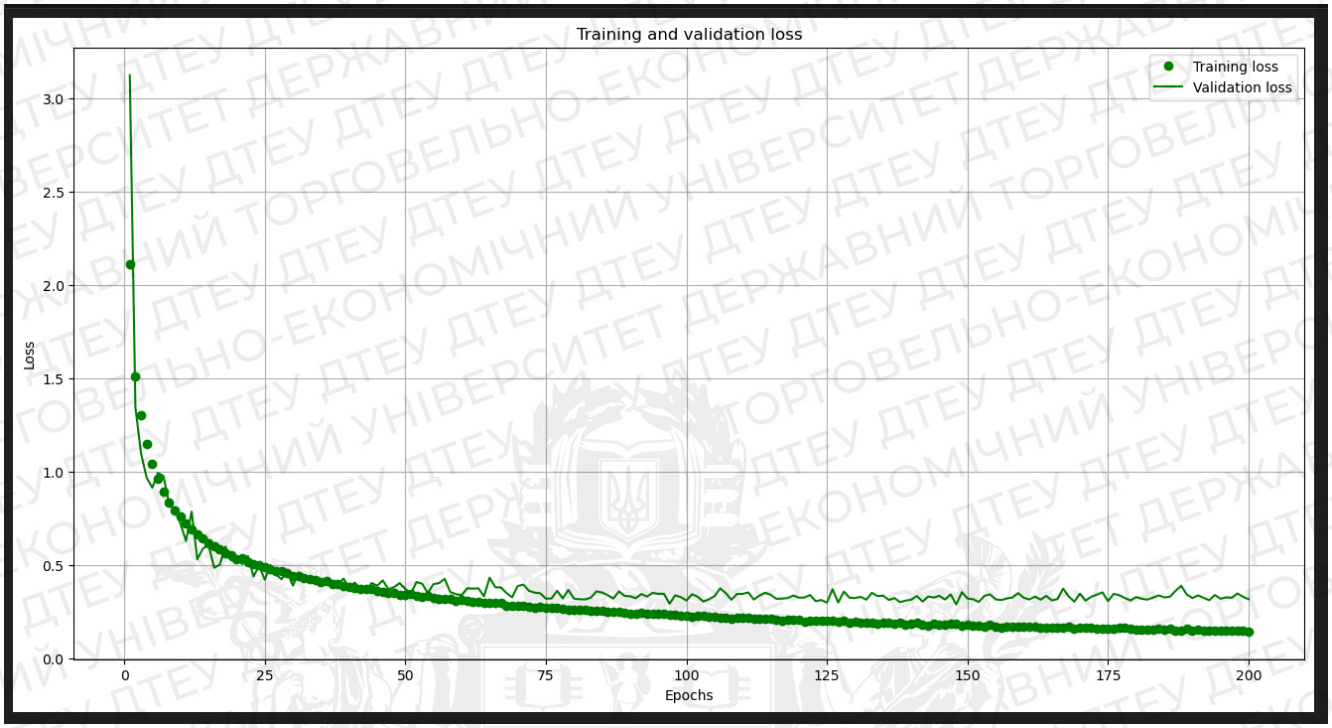

Рис.2.3. Графік втрат

На рисунку 2.3. зображена архітектура моделі та показані шари які використовувалися при роботі цієї моделі такі як:

- Згорткові шари (Conv2D)
- Шари Max_pooling;
- Dropout;
- Flatten:
- Повнозв’язний шар Dense.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 32, 32, 32)	896
batch_normalization (BatchNo	(None, 32, 32, 32)	128
conv2d_1 (Conv2D)	(None, 32, 32, 32)	9248
batch_normalization_1 (Batch	(None, 32, 32, 32)	128
max_pooling2d (MaxPooling2D)	(None, 16, 16, 32)	0
dropout (Dropout)	(None, 16, 16, 32)	0
conv2d_2 (Conv2D)	(None, 16, 16, 64)	18496
batch_normalization_2 (Batch	(None, 16, 16, 64)	256
conv2d_3 (Conv2D)	(None, 16, 16, 64)	36928
batch_normalization_3 (Batch	(None, 16, 16, 64)	256
max_pooling2d_1 (MaxPooling2	(None, 8, 8, 64)	0
...		

Рис. 2.3. Архітектура моделі

2.5. Висновки до розділу 2

Цей розділ був присвячений обґрунтуванню методів та засобів розробки. В розділі було описано технічне та програмне забезпечення потрібне для написання веб додатку, а також проаналізовано мови програмування та самі програми які використовувались в ході написання.

Також було показано модель згорткової мережі її графіки та архітектура, також було доведено що модель не страждає від перенавчання та здатна виковувати вимоги згідно з статистикою.

РОЗДІЛ 3.
РОЗРОБКА ІНТЕРФЕЙСУ ТА ФУНКЦІОНАЛУ ВЕБ ДОДАТКУ

3.1. Робота з веб додатком

Для початку роботи з додатком у ньому потрібно зареєструватися або увійти до нього. Рис. 3.1.

Рис. 3.1. Вхід до додатку

Після натискання кнопки реєстрація відкриється меню реєстрації Рис 3.2., якщо ж користувач вже зареєстрований то він натискає кнопку логін та після відкриття діалогового вікна Рис. 3.3. вводить логін та пароль та заходить до додатку.

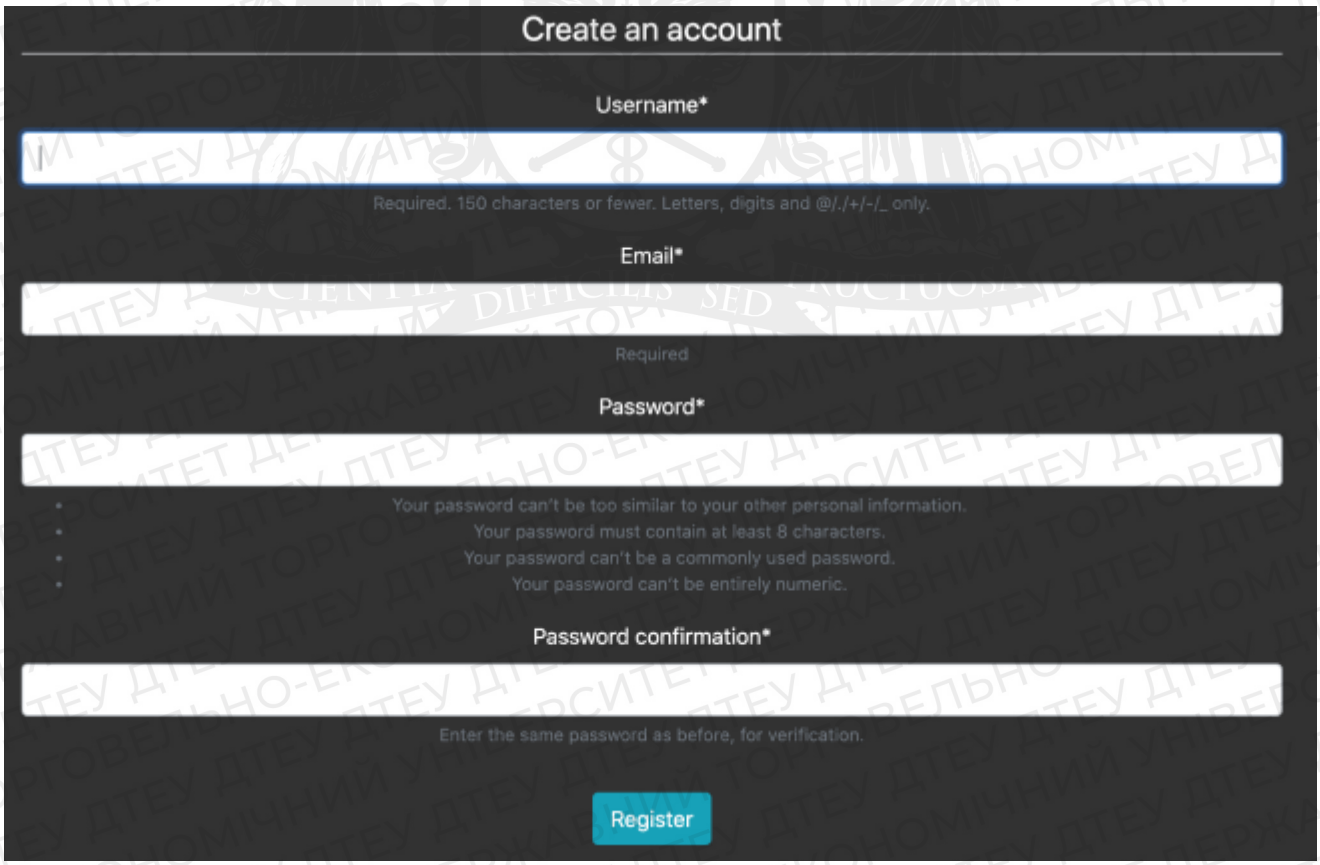

Рис. 3.2. Форма реєстрації

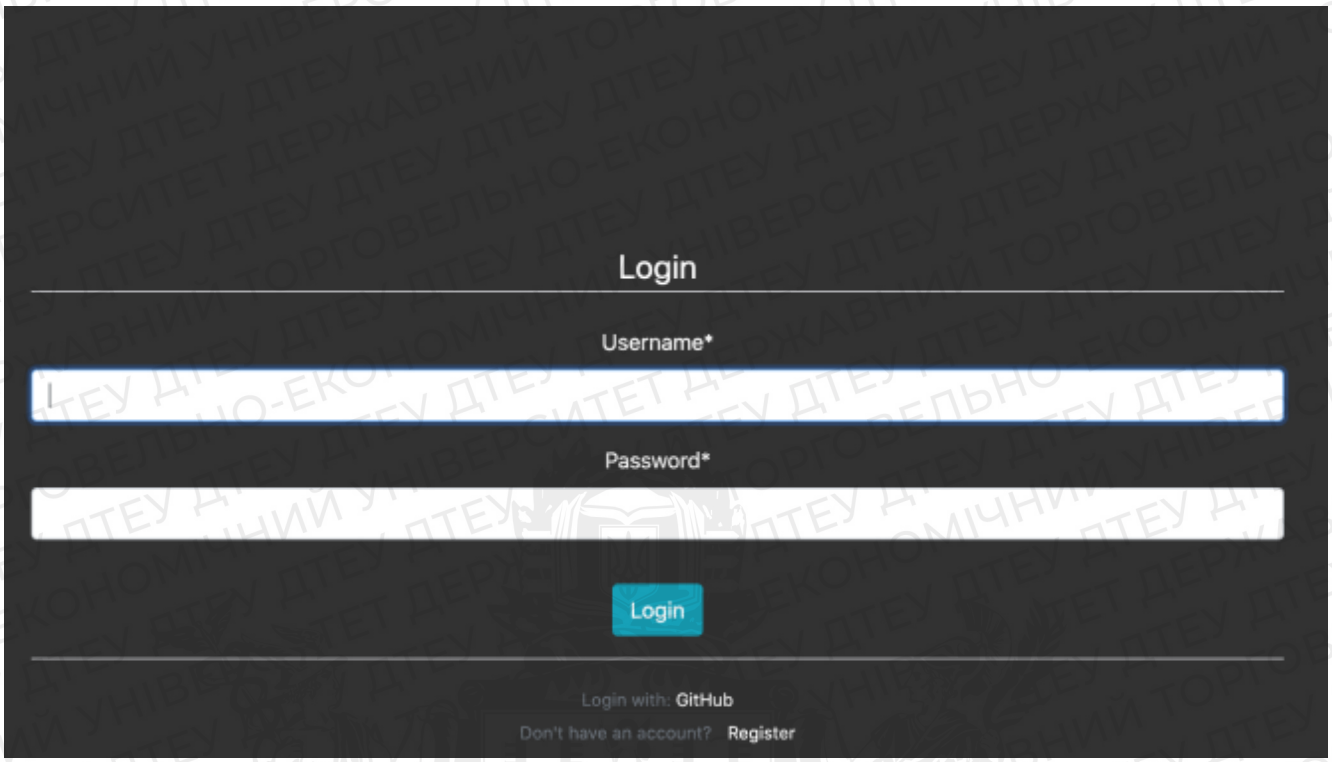

Рис. 3.3. Форма авторизації

Користувачі також можуть зареєструватися та увійти через соціальну мережу GitHub. Для цього користувачі повинні перейти на сторінку входу або реєстрації та вибрати бажану соціальну мережу GitHub (Рис 3.4.)

Рис. 3.4. Меню вибору реєстрації за допомогою GitHub

Після цього користувач буде перенаправлений на вибрану соціальну мережу. На рисунку 3.5. показано реєстрацію користувачів через платформу GitHub.

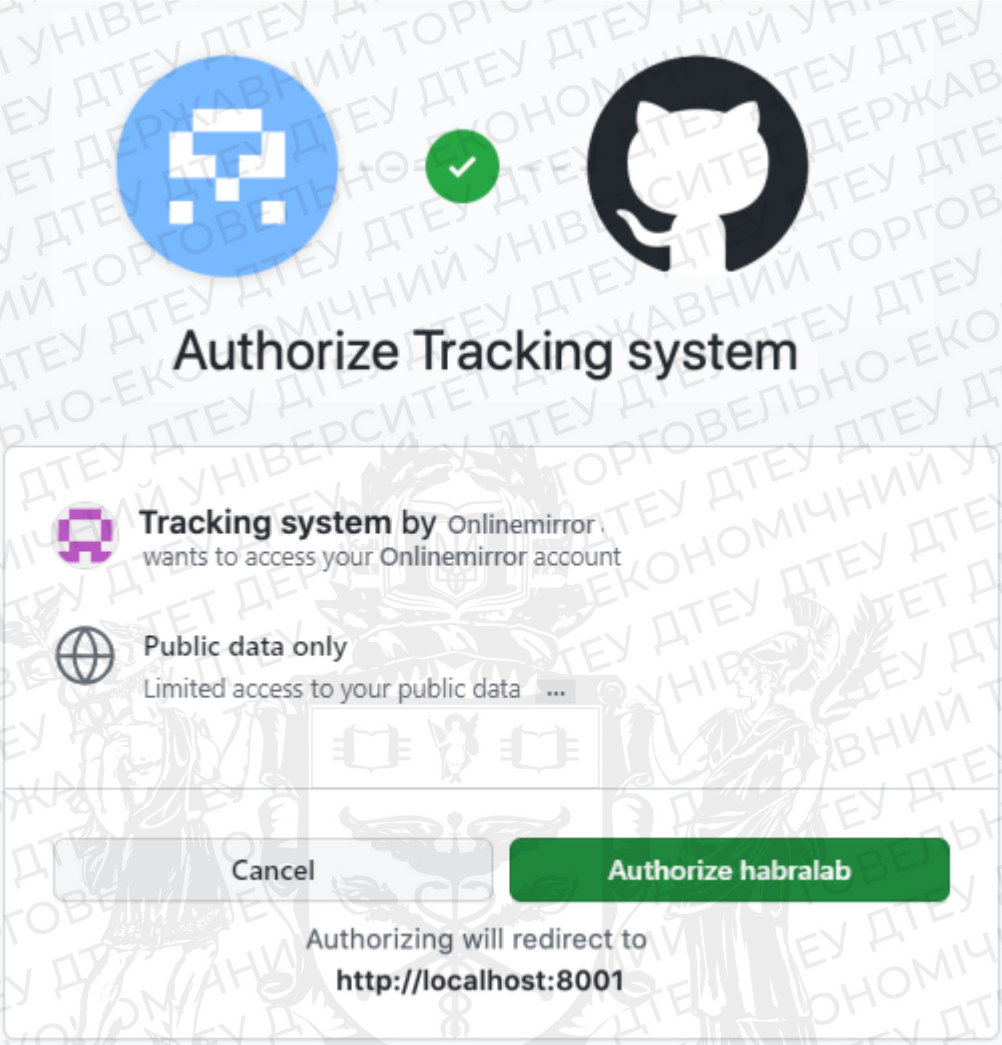

Рис. 3.5. — Реєстрація за допомогою платформи GitHub

Щоб запустити систему, користувачі можуть завантажити один або кілька відеофайлів у форматі «.MOV» або «.MP4» (рис. 3.7). Після завантаження відеофайлу користувачі можуть перевірити назву та тип (Рис. 5.8.), а якщо є помилки, вони можуть видалити всі завантажені відео (Рис. 5.9.).

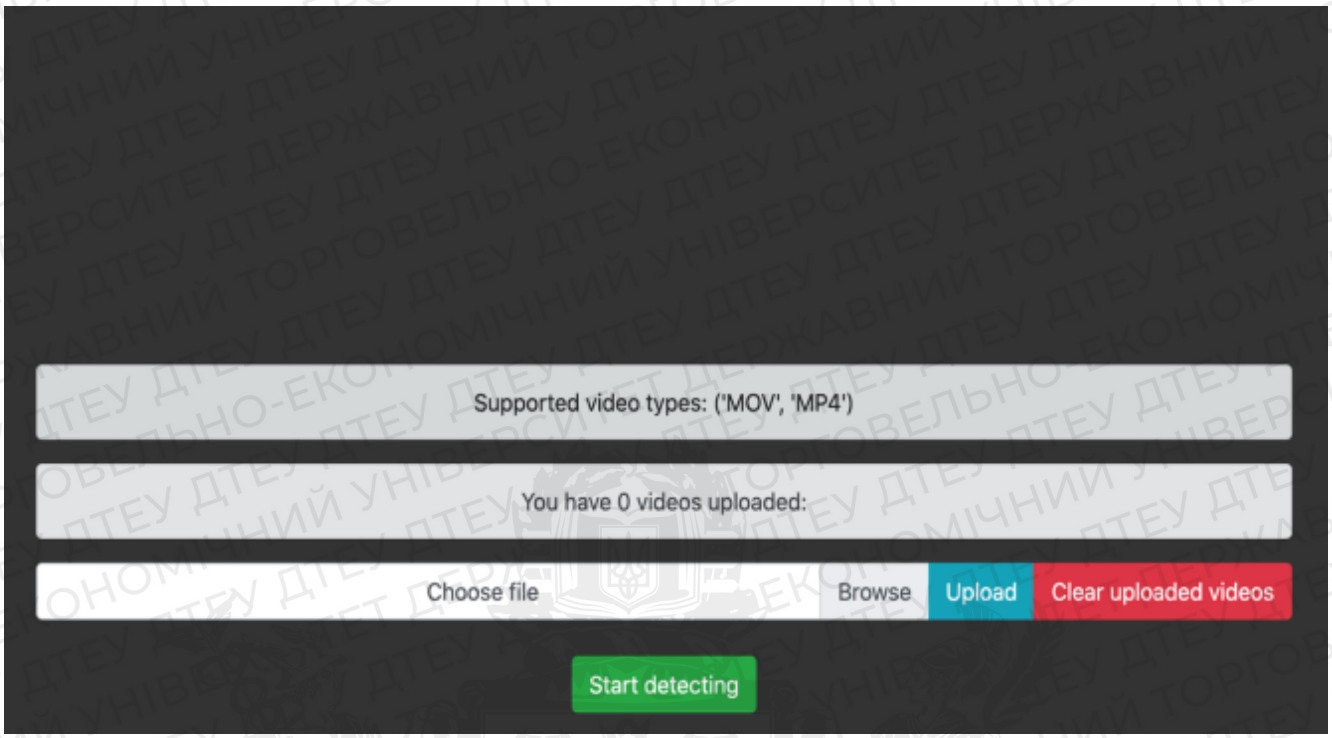

Рис. 3.6. Вікно завантаження файлів

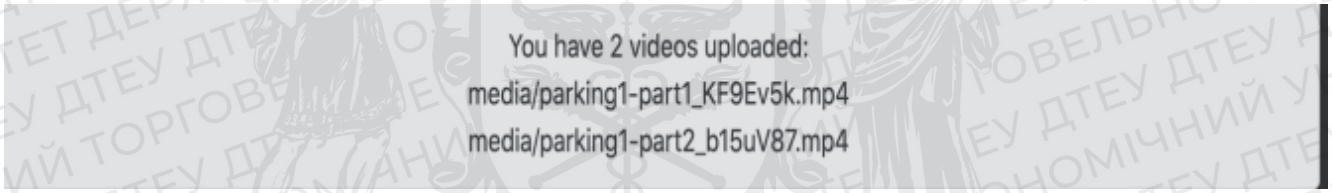

Рис. 5.7. Назва та тип завантажених файлів

Рисунок 5.8. — Кнопка видалення всіх завантажених відео файлів

Рисунок 5.9. — Кнопка початку опрацювання всіх завантажених відео файлів

3.2. Процес опрацювання відео згортковою мережею

Спочатку згорткова нейронна мережа опрацьовує файл розбиваючи його покадрово для розпізнавання на кожному з них автомобіля та номерного знака для запису кожного з них під унікальний айді у файл .csv.(Рис. 3.10., Рис. 3.11.).

	frame_nr	car_id	car_bbox	license_plate_bbox
2	0	3	[752.872314453125 1369.210693359375 1430.7803955078125 1982.91064453125]	[973.7655639648438 1753.3367919921875 1199.7279052734375 1851.619750
3	1	3	[752.7639976912175 1369.258419879689 1430.4360597313307 1982.8931884247208]	[973.7571411132812 1753.3359375 1199.6212158203125 1851.615234375]
4	1	5	[2197.3992408544573 1154.283572734545 2785.5587527173193 1736.5472180728339]	[2407.91455078125 1551.371337890625 2576.935791015625 1638.369873046
5	2	5	[2201.1786508376003 1163.4532689748805 2783.907397862044 1740.85193632114]	[2411.8359375 1549.122802734375 2583.56640625 1641.93603515625]
6	4	3	[749.0850912619109 1380.2562055258386 1430.1695306789852 2000.7945676572444]	[963.6485595703125 1783.3861083984375 1184.638916015625 1873.4183349
7	4	5	[2196.6996496906427 1171.5290002843537 2784.9977650817477 1751.7257927071043]	[2410.735107421875 1562.9150390625 2585.170654296875 1653.1937255859
8	5	5	[2194.0965823800507 1174.1965260770342 2789.763941468501 1755.6259285096055]	[2411.2822265625 1562.439453125 2586.25634765625 1652.39111328125]
9	8	5	[2189.132862151616 1176.7713453667247 2789.446758306771 1762.0337995041377]	[2416.160400390625 1575.296630859375 2587.7529296875 1653.5280761718
10	9	3	[737.1985271232963 1396.0504239310667 1425.6083192894187 2026.1469226896393]	[939.8256225585938 1790.5682373046875 1188.2000732421875 1892.258422
11	9	5	[2190.648555793981 1175.7442635446482 2792.2374736959214 1764.4548119766368]	[2414.810546875 1579.6453857421875 2589.49755859375 1661.13830566406
12	12	3	[727.5475398805889 1404.1161260091458 1422.2767128198707 2041.4148846270082]	[943.5031127929688 1802.426513671875 1187.0267333984375 1903.0832519
13	13	3	[726.2982943576301 1404.2620743184652 1420.75621644627 2043.2459775349575]	[943.4718017578125 1802.43310546875 1187.15625 1903.107421875]
14	13	5	[2192.415269199877 1169.956511853126 2809.747869927357 1772.347551773621]	[2417.90625 1588.09912109375 2590.99658203125 1673.6485595703125]
15	14	3	[725.9480844713642 1407.0738087051027 1423.1740196275157 2052.190660203413]	[938.2769775390625 1806.962158203125 1184.478759765625 1911.06677246
16	14	5	[2193.6217447581125 1176.172134676592 2814.818369847498 1777.3520259828672]	[2415.3896484375 1588.543212890625 2587.59912109375 1672.47802734375
17	15	3	[723.079203235948 1408.5604556348042 1418.8182278524212 2053.3830910647425]	[936.7955932617188 1821.8829345703125 1156.3184814453125 1914.359985
18	16	3	[722.4587091407063 1412.9163300380678 1419.6811684470551 2056.8653027446203]	[959.2699584960938 1826.1456298828125 1167.9412841796875 1918.695190
19	17	3	[716.9063291975472 1417.3287470347584 1415.1856753079305 2062.0351064039583]	[957.8289184570312 1835.7568359375 1171.025390625 1923.843017578125]
20	18	5	[2195.4810519936304 1194.762439772446 2822.631631150261 1793.3513914315129]	[2421.121826171875 1596.1175537109375 2594.187255859375 1690.1033935
21	19	5	[2195.8235322983173 1195.7431914982008 2825.2582532133592 1795.1075967636807]	[2421.060302734375 1596.0872802734375 2594.220703125 1690.1639404296
22	20	5	[2198.09216463595 1196.8977833815034 2825.7974104021578 1798.1970723749037]	[2424.370361328125 1600.628173828125 2596.72705078125 1695.707885742
23	21	5	[2196.5309959026654 1177.5447711586796 2821.5997851996344 1792.5383422483562]	[2421.150390625 1607.1339111328125 2597.283935546875 1701.6910400390
24	24	5	[2194.2257855451326 1198.023031608395 2813.672617158896 1810.2381703651502]	[2424.47265625 1619.255859375 2596.346435546875 1701.387451171875]
25	26	5	[2194.861533317529 1204.262373064968 2813.155114569165 1814.7577549078583]	[2425.511474609375 1626.451904296875 2598.78149414062 5 1708.24877929
26	27	3	[691.3609929938143 1445.8004353132624 1404.5451020551022 2099.9763759855978]	[918.0625 1869.5701904296875 1154.2078857421875 1959.16259765625]
27	27	5	[2196.8174343572014 1208.2147786659125 2813.733746386901 1816.5295504730134]	[2421.14599609375 1626.3387451171875 2599.56689453125 1709.456909179
28	29	5	[2200.225085276938 1213.6011788563578 2816.7855608163677 1822.1260229212226]	[2426.165283203125 1636.0209609375 2599.717041015625 1723.998779296
29	30	5	[2201.943013942075 1219.355851001601 2819.8456201864133 1828.6652555230366]	[2426.889404296875 1637.1014404296875 2597.40576171875 1721.10620117
30	35	3	[674.2249904416399 1475.4229690325176 1395.861026897332 2129.46660781837]	[912.9992065429688 1892.782958984375 1157.5447998046875 1994.5222167

Рис. 3.10. Файл з результатами розпізнавання

	license_plate_bbox	license_plate_bbox_sco	license_numb	license_number_sco	S
2	[973.7655639648438 1753.3367919921875 1199.7279052734375 1851.6197509765625]	0.565711915493012	OA13NRU	0.311114546985471	
3	[973.7571411132812 1753.3359375 1199.6212158203125 1851.615234375]	0.56566309928894	OA13NRU	0.186278822049473	
4	[2407.91455078125 1551.371337890625 2576.935791015625 1638.369873046875]	0.595081508159637	MU51KSU	0.124257471920163	
5	[2411.8359375 1549.122802734375 2583.56640625 1641.93603515625]	0.586198568344116	HV51VSU	0.204080291702356	
6	[963.6485595703125 1783.3861083984375 1184.638916015625 1873.4183349609375]	0.499943137168884	NA13NRU	0.466064216490127	
7	[2410.735107421875 1562.9150390625 2585.170654296875 1653.1937255859375]	0.42640033364296	HU51KSU	0.228259984834154	
8	[2411.2822265625 1562.439453125 2586.25634765625 1652.39111328125]	0.497613728046417	HU51VSU	0.410648627634629	
9	[2416.160400390625 1575.296630859375 2587.7529296875 1653.528076171875]	0.565631091594696	MO51YSU	0.0818216271345439	
10	[939.8256225585938 1790.5682373046875 1188.2000732421875 1892.2584228515625]	0.386960834264755	NA13NRU	0.391677719708131	
11	[2414.810546875 1579.6453857421875 2589.49755859375 1661.1383056640625]	0.582193732261658	KT53YSU	0.0581356737745266	
12	[943.5031127929688 1802.426513671875 1187.0267333984375 1903.083251953125]	0.523254752159119	NA13NRU	0.502694615282663	
13	[943.4718017578125 1802.43310546875 1187.15625 1903.107421875]	0.52321320772171	NA13NRU	0.502694615282663	
14	[2417.90625 1588.09912109375 2590.99658203125 1673.6485595703125]	0.626428842544556	NO51YSU	0.0531676897384063	
15	[938.2769775390625 1806.962158203125 1184.478759765625 1911.0667724609375]	0.56396222114563	NA13NRU	0.516853025493898	
16	[2415.3896484375 1588.543212890625 2587.59912109375 1672.47802734375]	0.609500885009766	MV51VSU	0.176935753478185	
17	[936.7955932617188 1821.8829345703125 1156.3184814453125 1914.3599853515625]	0.585464477539063	NA13MRU	0.211440746021335	
18	[959.2699584960938 1826.1456298828125 1167.9412841796875 1918.6951904296875]	0.546614587306976	NA13NRU	0.184412391256291	
19	[957.8289184570312 1835.7568359375 1171.025390625 1923.843017578125]	0.581541657447815	NA13NRU	0.35992706962599	
20	[2421.121826171875 1596.1175537109375 2594.187255859375 1690.1033935546875]	0.630939185619354	ML51NSU	0.245839673529339	
21	[2421.060302734375 1596.0872802734375 2594.220703125 1690.1639404296875]	0.630464732646942	ML51NSU	0.245839673529339	
22	[2424.370361328125 1600.628173828125 2596.72705078125 1695.7078857421875]	0.630373477935791	MV51VSU	0.45188132898378	
23	[2421.150390625 1607.1339111328125 2597.283935546875 1701.6910400390625]	0.615781664848328	ML51YSV	0.0594327576555846	
24	[2424.47265625 1619.255859375 2596.346435546875 1701.387451171875]	0.583320021629334	MI51VSU	0.144442820023826	
25	[2425.511474609375 1626.451904296875 2598.781494140625 1708.248779296875]	0.613147735595703	MI51VSU	0.267633074482529	
26	[918.0625 1869.5701904296875 1154.2078857421875 1959.16259765625]	0.504813373088837	NA13MRU	0.151810115461 64	
27	[2421.14599609375 1626.3387451171875 2599.56689453125 1709.4569091796875]	0.609328210353851	MI51VSU	0.0592854367880044	
28	[2426.165283203125 1636.02099609375 2599.717041015625 1723.998779296875]	0.668651103973389	IV51VSU	0.0487461554367906	
29	[2426.889404296875 1637.1014404296875 2597.40576171875 1721.106201171875]	0.646485269069672	MI51WSU	0.0712310323573176	
30	[912.9992065429688 1892.782958984375 1157.5447998046875 1994.522216796875]	0.577380299568176	NA13NRU	0.435135720213308	

Рис 3.11. Файл з результатами розпізнавання

Далі якщо відсортувати автомобілі за айді, видно що через один з мінусів роботи нейромережі є прогалини у фреймах через що автомобіль не розпізнається постійно. (Рис. 3.12., Рис.3.13.). Це загалом не впливає на якість введення результату, але спотворює візуалізацію роботи нейромережі.

1	frame_nr	car_id	car_bbox	license_plate_bbox
2	0	3	[752.872314453125 1369.210693359375 1430.7803955078125 1982.91064453125]	[973.7655639648438 1753.3367919921875 1199.7279052734375 1851.619750
3	1	3	[752.7639976912175 1369.258419879689 1430.4360597313307 1982.8931884247208]	[973.7571411132812 1753.3359375 1199.6212158203125 1851.615234375]
6	4	3	[749.0850912619109 1380.2562055258386 1430.1695306789852 2000.7945676572444]	[963.6485595703125 1783.3861083984375 1184.638916015625 1873.418334
10	9	3	[737.1985271232963 1396.0504239310667 1425.6083192894187 2026.1469226896393]	[939.8256225585938 1790.5682373046875 1188.2000732421875 1892.25842
12	12	3	[727.5475398805889 1404.1161260091458 1422.2767128198707 2041.4148846270082]	[943.5031127929688 1802.426513671875 1187.0267333984375 1903.083251
13	13	3	[726.2982943576301 1404.2620743184652 1420.75621644627 2043.2459775349575]	[943.4718017578125 1802.43310546875 1187.15625 1903.107421875]
15	14	3	[725.9480844713642 1407.0738087051027 1423.1740196275157 2052.190660203413]	[938.2769775390625 1806.962158203125 1184.478759765625 1911.0667724
17	15	3	[723.079203235948 1408.5604556348042 1418.8182278524212 2053.3830910647425]	[936.7955932617188 1821.8829345703125 1156.3184814453125 1914.359985
18	16	3	[722.4587091407063 1412.9163300380678 1419.6811684470551 2056.8653027446203]	[959.2699584960938 1826.1456298828125 1167.9412841796875 1918.69519
19	17	3	[716.9063291975472 1417.3287470347584 1415.1856753079305 2062.0351064039583]	[957.8289184570312 1835.7568359375 1171.025390625 1923.843017578125]
26	27	3	[691.3609929938143 1445.8004353132624 1404.5451020551022 2099.9763759855978]	[918.0625 1869.5701904296875 1154.2078857421875 1959.16259765625]
30	35	3	[674.2249904416399 1475.4229690325176 1395.861026897332 2129.46660781837]	[912.9992065429688 1892.782958984375 1157.5447998046875 1994.522216
33	37	3	[670.8262681522235 1483.0549949860729 1391.7801707930332 2135.1642048317376]	[903.099609375 1902.1248779296875 1156.3721923828125 2005.616088867
35	38	3	[669.2003318278507 1488.6218128371079 1392.4475297137446 2139.0662083194597]	[924.1004028320312 1911.2764892578125 1148.2674560546875 2009.43054
36	40	3	[668.4866748788718 1496.8438601603393 1393.350227823054 2146.5423220941593]	[900.3173217773438 1923.8775634765625 1133.605712890625 2019.758544
38	42	3	[660.3863664382179 1508.2792437788694 1387.4493830975075 2151.209148316935]	[882.327392578125 1927.5765380859375 1152.17138671875 2039.91467285
39	43	3	[658.1813792938541 1511.1973585548285 1386.0698277145798 2151.282270068789]	[882.4586791992188 1927.748291015625 1152.0616455078125 2039.771484
40	44	3	[656.5110008374583 1511.9779191204466 1384.863786157965 2151.1813158863993]	[877.4298706054688 1923.5225830078125 1135.0654296875 2037.60681152
42	45	3	[654.8167429802346 1515.5609418447832 1384.724738570339 2153.09725355673]	[895.3402099609375 1934.03515625 1150.44580078125 2043.954467773437
43	46	3	[652.6581512620118 1517.900489291081 1382.983650313562 2152.606199597907]	[904.351318359375 1940.736328125 1140.9927978515625 2045.4448242187
44	47	3	[650.7496019023786 1518.821565377566 1382.4609077592263 2151.172037494821]	[882.0919189453125 1945.1195068359375 1147.5347900390625 2060.54321
46	51	3	[648.460439203172 1532.195257557516 1380.5399962566025 2152.74701827904]	[871.4727172851562 1970.402099609375 1129.554931640625 2073.3806152
47	52	3	[645.7620197470794 1536.4298321530791 1379.9629347434102 2153.2775457008875]	[885.2010498046875 1974.8790283203125 1138.8665771484375 2084.52978
48	53	3	[645.9769819726953 1539.6491162747627 1381.3639564678344 2152.517257500118]	[870.7135620117188 1973.669677734375 1120.692626953125 2083.4245605
49	54	3	[639.6811705649826 1544.0458843136228 1379.8754493094434 2152.5053517828037]	[860.2695922851562 1977.109130859375 1127.8582763671875 2091.520751
50	55	3	[637.6314606795745 1546.441537132678 1380.5108586891595 2151.7382097782233]	[860.3241577148438 1977.1964111328125 1127.708984375 2091.402832031
52	58	3	[629.9808805122499 1558.2746756626075 1372.9729196643084 2152.4954749412877]	[862.688232421875 1996.0975341796875 1119.650390625 2105.6472167968
53	60	3	[632.8344029312893 1565.5166501083672 1372.9208970427173 2152.3565780055123]	[885.548583984375 2017.80029296875 1094.9659423828125 2118.11450195
54	61	3	[633.5502288383686 1566.9399770944308 1372.1749989033567 2151.5053429399622]	[885.2470703125 2017.694580078125 1095.1405029296875 2118.353515625]

Рис. 3.12. Файл з відсортованими результатами розпізнавання

1	license_plate_bbox	license_plate_bbox_sco	license_numb	license_number_sco	S
2	[973.7655639648438 1753.3367919921875 1199.7279052734375 1851.6197509765625]	0.565711915493012	OA13NRU	0.311114546985471	
3	[973.7571411132812 1753.3359375 1199.6212158203125 1851.615234375]	0.56566309928894	OA13NRU	0.186278822049473	
6	[963.6485595703125 1783.3861083984375 1184.638916015625 1873.4183349609375]	0.499943137168884	NA13NRU	0.466064216490127	
10	[939.8256225585938 1790.5682373046875 1188.2000732421875 1892.2584228515625]	0.386960834264755	NA13NRU	0.391677719708131	
12	[943.5031127929688 1802.426513671875 1187.0267333984375 1903.083251953125]	0.523254752159119	NA13NRU	0.502694615282663	
13	[943.4718017578125 1802.43310546875 1187.15625 1903.107421875]	0.52321320772171	NA13NRU	0.502694615282663	
15	[938.2769775390625 1806.962158203125 1184.478759765625 1911.0667724609375]	0.56396222114563	NA13NRU	0.516853025493898	
17	[936.7955932617188 1821.8829345703125 1156.3184814453125 1914.3599853515625]	0.585464477539063	NA13MRU	0.211440746021335	
18	[959.2699584960938 1826.1456298828125 1167.9412841796875 1918.6951904296875]	0.546614587306976	NA13NRU	0.184412391256291	
19	[957.8289184570312 1835.7568359375 1171.025390625 1923.843017578125]	0.581541657447815	NA13NRU	0.35992706962599	
26	[918.0625 1869.5701904296875 1154.2078857421875 1959.16259765625]	0.504813373088837	NA13MRU	0.15181011546164	
30	[912.9992065429688 1892.782958984375 1157.5447998046875 1994.522216796875]	0.577380299568176	NA13NRU	0.435135720213308	
33	[903.099609375 1902.1248779296875 1156.3721923828125 2005.6160888671875]	0.560267567634583	NA13NRU	0.186641180284947	
35	[924.1004028320312 1911.2764892578125 1148.2674560546875 2009.4305419921875]	0.639698028564453	NA13NRU	0.181152727807862	
36	[900.3173217773438 1923.8775634765625 1133.605712890625 2019.758544921875]	0.619913339614868	NA13NRU	0.31105867026596	
38	[882.327392578125 1927.5765380859375 1152.17138671875 2039.9146728515625]	0.609655022621155	NA13NRU	0.459747550238893	
39	[882.4586791992188 1927.748291015625 1152.0616455078125 2039.771484375]	0.609971523284912	NA13NRU	0.459747550238893	
40	[877.4298706054688 1923.5225830078125 1135.0654296875 2037.6068115234375]	0.637619435787201	NA13MRU	0.51448898909834	
42	[895.3402099609375 1934.03515625 1150.44580078125 2043.9544677734375]	0.619776248931885	NA13NRU	0.197217769848014	
43	[904.351318359375 1940.736328125 1140.9927978515625 2045.44482421875]	0.653043746948242	NA13MRU	0.377651258147259	
44	[882.0919189453125 1945.1195068359375 1147.5347900390625 2060.543212890625]	0.628783464431763	NA13MRU	0.33702545063279	
46	[871.4727172851562 1970.402099609375 1129.554931640625 2073.380615234375]	0.643819332122803	NA13NPU	0.205479604469372	
47	[885.2010498046875 1974.8790283203125 1138.8665771484375 2084.52978515625]	0.599873185157776	NA13NRU	0.748869662888482	
48	[870.7135620117188 1973.669677734375 1120.692626953125 2083.424560546875]	0.648566246032715	NA13NPU	0.285549692757662	
49	[860.2695922851562 1977.109130859375 1127.8582763671875 2091.520751953125]	0.670482456684113	NA13MRU	0.226444855676471	
50	[860.3241577148438 1977.1964111328125 1127.708984375 2091.40283203125]	0.669727981090546	NA13MRU	0.226444855676471	
52	[862.688232421875 1996.0975341796875 1119.650390625 2105.647216796875]	0.656189560890198	NA13NRU	0.620971165423034	
53	[885.548583984375 2017.80029296875 1094.9659423828125 2118.114501953125]	0.671475291252136	NA13NRU	0.401401971507914	
54	[885.2470703125 2017.694580078125 1095.1405029296875 2118.353515625]	0.671384751796722	NA13NRU	0.370819606154495	

Рис. 3.13. Файл з відсортованими результатами розпізнавання

Через це застосовуємо інтерполяцію даних щоб усереднити данні щоб покращити вихідне зображення. (Рис. 3.14. Рис. 3.15.).

1	frame_nr	car_	car_bbox	license_plate_bbox
230	0	3	752.872314453125 1369.210693359375 1430.7803955078125 1982.91064453125	973.7655639648438 1753.3367919921875 1199.7279052734375 1851.61975097
231	1	3	752.7639976912175 1369.258419879689 1430.4360597313307 1982.8931884247208	973.7571411132812 1753.3359375 1199.6212158203125 1851.615234375
232	2	3	751.5376955481153 1372.9243484284054 1430.3472167138823 1988.8603148355621	970.3876139322916 1763.3526611328125 1194.6271158854167 1858.88293457
233	3	3	750.3113934050131 1376.5902769771221 1430.2583736964336 1994.8274412464032	967.0180867513021 1773.369384765625 1189.6330159505208 1866.150634765
234	4	3	749.0850912619109 1380.2562055258386 1430.1695306789852 2000.7945676572444	963.6485595703125 1783.3861083984375 1184.638916015625 1873.418334960
235	5	3	746.707778434188 1383.4150492068843 1429.257288401072 2005.8650386637235	958.8839721679688 1784.8225341796874 1185.3511474609375 1877.18635253
236	6	3	744.330465606465 1386.5738928879298 1428.3450461231587 2010.9355096702025	954.119384765625 1786.2589599609375 1186.06337890625 1880.95437011718
237	7	3	741.9531527787422 1389.7327365689755 1427.4328038452452 2016.0059806766812	949.3547973632812 1787.6953857421875 1186.7756103515626 1884.72238769
238	8	3	739.5758399510192 1392.891580250021 1426.520561567332 2021.0764516831603	944.5902099609375 1789.1318115234376 1187.487841796875 1888.490405273
239	9	3	737.1985271232963 1396.0504239310667 1425.6083192894187 2026.1469226896393	939.8256225585938 1790.5682373046875 1188.2000732421875 1892.25842285
240	10	3	733.9815313757272 1398.7389912904264 1424.4977837995693 2031.236243335429	941.0514526367188 1794.52099609375 1187.8089599609375 1895.8666992187
241	11	3	730.7645356281581 1401.427558649786 1423.38724830972 2036.3255639812185	942.2772827148438 1798.4737548828125 1187.4178466796875 1899.47497558
242	12	3	727.5475398805889 1404.1161260091458 1422.2767128198707 2041.4148846270082	943.5031127929688 1802.426513671875 1187.0267333984375 1903.083251953
243	13	3	726.2982943576301 1404.2620743184652 1420.75621644627 2043.2459775349575	943.4718017578125 1802.43310546875 1187.15625 1903.107421875
244	14	3	725.9480844713642 1407.0738087051027 1423.1740196275157 2052.190660203413	938.2769775390625 1806.962158203125 1184.478759765625 1911.0667724609
245	15	3	723.079203235948 1408.5604556348042 1418.8182278524212 2053.3830910647425	936.7955932617188 1821.8829345703125 1156.3184814453125 1914.35998535
246	16	3	722.4587091407063 1412.9163300380678 1419.6811684470551 2056.8653027446203	959.2699584960938 1826.1456298828125 1167.9412841796875 1918.69519042
247	17	3	716.9063291975472 1417.3287470347584 1415.1856753079305 2062.0351064039583	957.8289184570312 1835.7568359375 1171.025390625 1923.843017578125
248	18	3	714.3517955771739 1420.1759158626087 1414.1216179826476 2065.829233362122	953.8522766113281 1839.1381713867188 1169.3436401367187 1927.37497558
249	19	3	711.7972619568006 1423.0230846904592 1413.0575606573648 2069.6233603202863	949.875634765625 1842.5195068359376 1167.6618896484374 1930.906933593
250	20	3	709.2427283364273 1425.8702535183097 1411.9935033320821 2073.41748727845	945.8989929199219 1845.9008422851562 1165.9801391601563 1934.43889160
251	21	3	706.688194716054 1428.71742234616 1410.9294460067993 2077.211614236614	941.9223510742188 1849.282177734375 1164.298388671875 1937.9708496093
252	22	3	704.1336610956807 1431.5645911740103 1409.8653886815164 2081.0057411947782	937.9457092285156 1852.6635131835938 1162.6166381835938 1941.50280761
253	23	3	701.5791274753075 1434.4117600018608 1408.8013313562335 2084.799868152942	933.9690673828125 1856.0448486328125 1160.9348876953125 1945.03476562
254	24	3	699.0245938549342 1437.2589288297113 1407.7372740309506 2088.593995111106	929.9924255371094 1859.4261840820313 1159.2531372070312 1948.56672363
255	25	3	696.4700602345608 1440.1060976575616 1406.673216705668 2092.3881220692697	926.0157836914062 1862.80751953125 1157.57138671875 1952.098681640625
256	26	3	693.9155266141876 1442.9532664854119 1405.609159380385 2096.182249027434	922.0391418457032 1866.1888549804687 1155.8896362304688 1955.63063964
257	27	3	691.3609929938143 1445.8004353132624 1404.5451020551022 2099.9763759855978	918.0625 1869.5701904296875 1154.2078857421875 1959.16259765625
258	28	3	689.2189926747925 1449.5032520281693 1403.459592660381 2103.6626549646944	917.4295883178711 1872.4717864990234 1154.625 1963.5825500488281

Рис. 3.14. Файл з результатами розпізнавання після інтерполяції.

1	license_plate_bbox	license_plate_bbox_sco	license_numb	license_number_sco
230	973.7655639648438 1753.3367919921875 1199.7279052734375 1851.6197509765625	0.565711915493012	OA13NRU	0.311114546985471
231	973.7571411132812 1753.3359375 1199.6212158203125 1851.615234375	0.56566309928894	OA13NRU	0.186278822049473
232	970.3876139322916 1763.3526611328125 1194.6271158854167 1858.8829345703125	0	0	0
233	967.0180867513021 1773.369384765625 1189.6330159505208 1866.150634765625	0	0	0
234	963.6485595703125 1783.3861083984375 1184.638916015625 1873.4183349609375	0.499943137168884	NA13NRU	0.466064216490127
235	958.8839721679688 1784.8225341796874 1185.3511474609375 1877.1863525390625	0	0	0
236	954.119384765625 1786.2589599609375 1186.06337890625 1880.9543701171874	0	0	0
237	949.3547973632812 1787.6953857421875 1186.7756103515626 1884.7223876953126	0	0	0
238	944.5902099609375 1789.1318115234376 1187.487841796875 1888.4904052734375	0	0	0
239	939.8256225585938 1790.5682373046875 1188.2000732421875 1892.2584228515625	0.386960834264755	NA13NRU	0.391677719708131
240	941.0514526367188 1794.52099609375 1187.8089599609375 1895.86669921875	0	0	0
241	942.2772827148438 1798.4737548828125 1187.4178466796875 1899.4749755859375	0	0	0
242	943.5031127929688 1802.426513671875 1187.0267333984375 1903.083251953125	0.523254752159119	NA13NRU	0.502694615282663
243	943.4718017578125 1802.43310546875 1187.15625 1903.107421875	0.52321320772171	NA13NRU	0.502694615282663
244	938.2769775390625 1806.962158203125 1184.478759765625 1911.0667724609375	0.56396222114563	NA13NRU	0.516853025493898
245	936.7955932617188 1821.8829345703125 1156.3184814453125 1914.3599853515625	0.585464477539063	NA13MRU	0.211440746021335
246	959.2699584960938 1826.1456298828125 1167.9412841796875 1918.6951904296875	0.546614587306976	NA13NRU	0.184412391256291
247	957.8289184570312 1835.7568359375 1171.025390625 1923.843017578125	0.581541657447815	NA13NRU	0.35992706962599
248	953.8522766113281 1839.1381713867188 1169.3436401367187 1927.3749755859376	0	0	0
249	949.875634765625 1842.5195068359376 1167.6618896484374 1930.90693359375	0	0	0
250	945.8989929199219 1845.9008422851562 1165.9801391601563 1934.4388916015625	0	0	0
251	941.9223510742188 1849.282177734375 1164.298388671875 1937.970849609375	0	0	0
252	937.9457092285156 1852.6635131835938 1162.6166381835938 1941.5028076171875	0	0	0
253	933.9690673828125 1856.0448486328125 1160.9348876953125 1945.034765625	0	0	0
254	929.9924255371094 1859.4261840820313 1159.2531372070312 1948.5667236328125	0	0	0
255	926.0157836914062 1862.80751953125 1157.57138671875 1952.098681640625	0	0	0
256	922.0391418457032 1866.1888549804687 1155.8896362304688 1955.6306396484374	0	0	0
257	918.0625 1869.5701904296875 1154.2078857421875 1959.16259765625	0.504813373088837	NA13MRU	0.15181011546164
258	917.4295883178711 1872.4717864990234 1154.625 1963.5825500488281	0	0	0

Рис. 3.15. Файл з результатами розпізнавання після інтерполяції

На виході користувач отримує відео з візуалізацією розпізнавання усіх автомобільних знаків автомобілів на відео тривалістю рівною завантаженому а також файли до та після інтерполяції, з вірогідностями правильності розпізнавання даних та впевненості нейромережі в правильному розпізнаванні номерного знаку. (Рис. 3.16., Рис. 3.17., Рис. 3.18.)

Рис. 3.16. Результати візуалізації.

Рис. 3.17. Результати візуалізації.

Рис. 3.18. Результати візуалізації.

3.3. Висновки до розділу 3

У цьому розділі було показано процес роботи в веб-додатком. Покроково пояснено для користувача як реєструватися та входити до додатку, а також можливість користування зовнішніми платформами для реєстрації.

Також стало зрозуміло які саме файли та відеоматеріали отримує користувач після запиту, пояснено необхідність інтерполяції даних та показані результати візуалізації.

ВИСНОВКИ ТА ПРОПОЗИЦІЇ

В ході написання випускної кваліфікаційної роботи «*Розробка автоматизованої системи для розпізнавання номерних знаків автомобілів*» досягли поставленої мети в розробці, а саме:

— розробка додатку для розпізнавання автомобільних знаків.

— навчання нейронної згорткової мережі.

— пришвидшення та поліпшення кількості коректних результатів нейронної мережі.

Було розглянуто технічне та програмне забезпечення потрібне для коректної роботи веб-додатку. Також для написання сайту використане таке програмне забезпечення як:

— Pycharm.

— Як мова програмування python3.

— Бібліотеки Python що допомогли у розробці: Yolov8, Flask, easyocr та sort.

Також були зображені результати навчання нейронної згорткової мережі та показана її працездатність.

У майбутньому система потребуватиме значної зміни у своєму підході та буде розширена для роботи з потоковим відео що буде значно ліпше для клієнтів та користувачів. Також потрібно вивести веб додаток на онлайн хостинг для демонстрації тестової версії майбутнім покупцям. Та буде доданий модуль реєстрації швидкості транспортних засобів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Герасименко, В., Яковлєв, А., & Бурдаков, О. (2019). "Дослідження та порівняння архітектур алгоритмів детекції об'єктів."
2. Герасименко, В., & Яковлєв, А. (2020). "Розробка системи детекції об'єктів на базі алгоритму YOLO."
3. Гриневич, А., & Сердюк, А. (2018). "Використання мови програмування Python у наукових дослідженнях."
4. Бурков, А., Мельник, А., & Литвиненко, Д. (2021). "Розробка системи розпізнавання тексту з використанням бібліотеки EasyOCR.
5. Марченко, Д., & Паламарчук, М. (2021). "Використання технології OCR для розпізнавання тексту в зображеннях."
6. Документація Yolov8 [Електронний ресурс] — режим доступу: https://docs.ultralytics.com/
7. Shakhnarovish, Darrell, and Indyk — Nearest-Neighbour Methods in Learning and Vision / Shakhnarovish, Darrell, and Indyk // MIT Press. — 2005.
8. Арлазаров В.Л., Троянкер В.В., Котович Н.В. — Адаптивное распознавание Символов [Електронний ресурс] / В.Л. Арлазаров, В.В. Троянкер, Н.В. Котович. — 2000. — Режим доступу до ресурсу: http://ocrai.narod.ru/adaptive.html.
9. Котович Н.В., Славин О.А. — Распознавание скелетных образов [Електронний ресурс] / Н.В.Котович, О.А.Славин. — 2003. — Режим доступу до ресурсу: http://ocrai.narod.ru/skeletrecognize.html.

ДОДАТКИ

Додаток А

Main.py

```
import cv2
from ultralytics import YOLO

from sort.sort import *
from util import get_car, read_license_plate, write_csv

results = {}

mot_tracker = Sort()

# load models
coco_model = YOLO('yolov8n.pt')
license_plate_detector = YOLO('./models/license_plate_detector.pt')

# load video
cap = cv2.VideoCapture('./sample.mp4')

vehicles = [2, 3, 5, 7]

# read frames
frame_nmr = -1
ret = True
while ret:
    frame_nmr += 1
    ret, frame = cap.read()
    if ret:
        results[frame_nmr] = {}
        # detect vehicles
        detections = coco_model(frame)[0]
        detections_ = []
        for detection in detections.boxes.data.tolist():
            x1, y1, x2, y2, score, class_id = detection
            if int(class_id) in vehicles:
                detections_.append([x1, y1, x2, y2, score])

        # track vehicles
        track_ids = mot_tracker.update(np.asarray(detections_))

        # detect license plates
        license_plates = license_plate_detector(frame)[0]
        for license_plate in license_plates.boxes.data.tolist():
            x1, y1, x2, y2, score, class_id = license_plate

            # assign license plate to car
            xcar1, ycar1, xcar2, ycar2, car_id = get_car(license_plate, track_ids)

            if car_id != -1:

                # crop license plate
                license_plate_crop = frame[int(y1):int(y2), int(x1): int(x2), :]

                # process license plate
                license_plate_crop_gray = cv2.cvtColor(license_plate_crop,
cv2.COLOR_BGR2GRAY)
                _, license_plate_crop_thresh =
cv2.threshold(license_plate_crop_gray, 64, 255, cv2.THRESH_BINARY_INV)
```

```
                # read license plate number
                license_plate_text, license_plate_text_score =
read_license_plate(license_plate_crop_thresh)

                if license_plate_text is not None:
                    results[frame_nmr][car_id] = {'car': {'bbox': [xcar1, ycar1,
xcar2, ycar2]},
                                                  'license_plate': {'bbox': [x1,
y1, x2, y2],
                                                                    'text':
license_plate_text,
                                                                    'bbox_score':
score,
                                                                    'text_score':
license_plate_text_score}}

# write results
write_csv(results, './test.csv')
```

Додаток Б

util.py

```
import string
import easyocr

# Initialize the OCR reader
reader = easyocr.Reader(['en'], gpu=False)

# Mapping dictionaries for character conversion
dict_char_to_int = {'O': '0',
                    'I': '1',
                    'J': '3',
                    'A': '4',
                    'G': '6',
                    'S': '5'}

dict_int_to_char = {'0': 'O',
                    '1': 'I',
                    '3': 'J',
                    '4': 'A',
                    '6': 'G',
                    '5': 'S'}


def write_csv(results, output_path):
    """
    Write the results to a CSV file.

    Args:
        results (dict): Dictionary containing the results.
        output_path (str): Path to the output CSV file.
    """
    with open(output_path, 'w') as f:
        f.write('{},{},{},{},{},{},{}\n'.format('frame_nmr', 'car_id', 'car_bbox',
                                                'license_plate_bbox',
'license_plate_bbox_score', 'license_number',
                                                'license_number_score'))

        for frame_nmr in results.keys():
            for car_id in results[frame_nmr].keys():
                print(results[frame_nmr][car_id])
                if 'car' in results[frame_nmr][car_id].keys() and \
```

```
                    'license_plate' in results[frame_nmr][car_id].keys() and \
                    'text' in results[frame_nmr][car_id]['license_plate'].keys():
                     f.write('{},{},{},{},{},{},{}\n'.format(frame_nmr,
                                                             car_id,
                                                             '[{} {} {}
{}]'.format(

results[frame_nmr][car_id]['car']['bbox'][0],

results[frame_nmr][car_id]['car']['bbox'][1],

results[frame_nmr][car_id]['car']['bbox'][2],

results[frame_nmr][car_id]['car']['bbox'][3]),
                                                             '[{} {} {}
{}]'.format(

results[frame_nmr][car_id]['license_plate']['bbox'][0],

results[frame_nmr][car_id]['license_plate']['bbox'][1],

results[frame_nmr][car_id]['license_plate']['bbox'][2],

results[frame_nmr][car_id]['license_plate']['bbox'][3]),

results[frame_nmr][car_id]['license_plate']['bbox_score'],

results[frame_nmr][car_id]['license_plate']['text'],

results[frame_nmr][car_id]['license_plate']['text_score'])
                                 )
        f.close()


def license_complies_format(text):
    """
    Check if the license plate text complies with the required format.

    Args:
        text (str): License plate text.

    Returns:
        bool: True if the license plate complies with the format, False otherwise.
    """
    if len(text) != 7:
        return False

    if (text[0] in string.ascii_uppercase or text[0] in dict_int_to_char.keys())
and \
       (text[1] in string.ascii_uppercase or text[1] in dict_int_to_char.keys())
and \
       (text[2] in ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'] or text[2]
in dict_char_to_int.keys()) and \
       (text[3] in ['0', '1', '2', '3', '4', '5', '6', '7', '8', '9'] or text[3]
in dict_char_to_int.keys()) and \
       (text[4] in string.ascii_uppercase or text[4] in dict_int_to_char.keys())
and \
       (text[5] in string.ascii_uppercase or text[5] in dict_int_to_char.keys())
and \
       (text[6] in string.ascii_uppercase or text[6] in dict_int_to_char.keys()):
        return True
    else:
        return False
```

```
def format_license(text):
    """
    Format the license plate text by converting characters using the mapping
dictionaries.

    Args:
        text (str): License plate text.

    Returns:
        str: Formatted license plate text.
    """
    license_plate_ = ''
    mapping = {0: dict_int_to_char, 1: dict_int_to_char, 4: dict_int_to_char, 5:
dict_int_to_char, 6: dict_int_to_char,
               2: dict_char_to_int, 3: dict_char_to_int}
    for j in [0, 1, 2, 3, 4, 5, 6]:
        if text[j] in mapping[j].keys():
            license_plate_ += mapping[j][text[j]]
        else:
            license_plate_ += text[j]

    return license_plate_


def read_license_plate(license_plate_crop):
    """
    Read the license plate text from the given cropped image.

    Args:
        license_plate_crop (PIL.Image.Image): Cropped image containing the license
plate.

    Returns:
        tuple: Tuple containing the formatted license plate text and its
confidence score.
    """

    detections = reader.readtext(license_plate_crop)

    for detection in detections:
        bbox, text, score = detection

        text = text.upper().replace(' ', '')

        if license_complies_format(text):
            return format_license(text), score

    return None, None


def get_car(license_plate, vehicle_track_ids):
    """
    Retrieve the vehicle coordinates and ID based on the license plate
coordinates.

    Args:
        license_plate (tuple): Tuple containing the coordinates of the license
plate (x1, y1, x2, y2, score, class id).
        vehicle_track_ids (list): List of vehicle track IDs and their
corresponding coordinates.
```

```
    Returns:
        tuple: Tuple containing the vehicle coordinates (x1, y1, x2, y2) and ID.
    """
    x1, y1, x2, y2, score, class_id = license_plate

    foundIt = False
    for j in range(len(vehicle_track_ids)):
        xcar1, ycar1, xcar2, ycar2, car_id = vehicle_track_ids[j]

        if x1 > xcar1 and y1 > ycar1 and x2 < xcar2 and y2 < ycar2:
            car_indx = j
            foundIt = True
            break

    if foundIt:
        return vehicle_track_ids[car_indx]

    return -1, -1, -1, -1, -1
```

Додаток В

add_missind_data.py

```
import csv
import numpy as np
from scipy.interpolate import interp1d


def interpolate_bounding_boxes(data):
    # Extract necessary data columns from input data
    frame_numbers = np.array([int(row['frame_nmr']) for row in data])
    car_ids = np.array([int(float(row['car_id'])) for row in data])
    car_bboxes = np.array([list(map(float, row['car_bbox'][1:-1].split())) for row
in data])
    license_plate_bboxes = np.array([list(map(float, row['license_plate_bbox'][1:-
1].split())) for row in data])

    interpolated_data = []
    unique_car_ids = np.unique(car_ids)
    for car_id in unique_car_ids:

        frame_numbers_ = [p['frame_nmr'] for p in data if int(float(p['car_id']))
== int(float(car_id))]
        print(frame_numbers_, car_id)

        # Filter data for a specific car ID
        car_mask = car_ids == car_id
        car_frame_numbers = frame_numbers[car_mask]
        car_bboxes_interpolated = []
        license_plate_bboxes_interpolated = []

        first_frame_number = car_frame_numbers[0]
        last_frame_number = car_frame_numbers[-1]

        for i in range(len(car_bboxes[car_mask])):
            frame_number = car_frame_numbers[i]
            car_bbox = car_bboxes[car_mask][i]
            license_plate_bbox = license_plate_bboxes[car_mask][i]

            if i > 0:
                prev_frame_number = car_frame_numbers[i-1]
```

```
                prev_car_bbox = car_bboxes_interpolated[-1]
                prev_license_plate_bbox = license_plate_bboxes_interpolated[-1]

                if frame_number - prev_frame_number > 1:
                    # Interpolate missing frames' bounding boxes
                    frames_gap = frame_number - prev_frame_number
                    x = np.array([prev_frame_number, frame_number])
                    x_new = np.linspace(prev_frame_number, frame_number,
num=frames_gap, endpoint=False)
                    interp_func = interp1d(x, np.vstack((prev_car_bbox,
car_bbox)), axis=0, kind='linear')
                    interpolated_car_bboxes = interp_func(x_new)
                    interp_func = interp1d(x, np.vstack((prev_license_plate_bbox,
license_plate_bbox)), axis=0, kind='linear')
                    interpolated_license_plate_bboxes = interp_func(x_new)

                    car_bboxes_interpolated.extend(interpolated_car_bboxes[1:])

license_plate_bboxes_interpolated.extend(interpolated_license_plate_bboxes[1:])

            car_bboxes_interpolated.append(car_bbox)
            license_plate_bboxes_interpolated.append(license_plate_bbox)

        for i in range(len(car_bboxes_interpolated)):
            frame_number = first_frame_number + i
            row = {}
            row['frame_nmr'] = str(frame_number)
            row['car_id'] = str(car_id)
            row['car_bbox'] = ' '.join(map(str, car_bboxes_interpolated[i]))
            row['license_plate_bbox'] = ' '.join(map(str,
license_plate_bboxes_interpolated[i]))

            if str(frame_number) not in frame_numbers_:
                # Imputed row, set the following fields to '0'
                row['license_plate_bbox_score'] = '0'
                row['license_number'] = '0'
                row['license_number_score'] = '0'
            else:
                # Original row, retrieve values from the input data if available
                original_row = [p for p in data if int(p['frame_nmr']) ==
frame_number and int(float(p['car_id'])) == int(float(car_id))][0]
                row['license_plate_bbox_score'] =
original_row['license_plate_bbox_score'] if 'license_plate_bbox_score' in
original_row else '0'
                row['license_number'] = original_row['license_number'] if
'license_number' in original_row else '0'
                row['license_number_score'] = original_row['license_number_score']
if 'license_number_score' in original_row else '0'

            interpolated_data.append(row)

    return interpolated_data


# Load the CSV file
with open('test.csv', 'r') as file:
    reader = csv.DictReader(file)
    data = list(reader)

# Interpolate missing data
interpolated_data = interpolate_bounding_boxes(data)

# Write updated data to a new CSV file
```

```
header = ['frame_nmr', 'car_id', 'car_bbox', 'license_plate_bbox',
'license_plate_bbox_score', 'license_number', 'license_number_score']
with open('test_interpolated.csv', 'w', newline='') as file:
    writer = csv.DictWriter(file, fieldnames=header)
    writer.writeheader()
    writer.writerows(interpolated_data)
```

Додаток Г

visualize.py

```
import ast

import cv2
import numpy as np
import pandas as pd


def draw_border(img, top_left, bottom_right, color=(0, 255, 0), thickness=10,
line_length_x=200, line_length_y=200):
    x1, y1 = top_left
    x2, y2 = bottom_right

    cv2.line(img, (x1, y1), (x1, y1 + line_length_y), color, thickness)  #-- top-
left
    cv2.line(img, (x1, y1), (x1 + line_length_x, y1), color, thickness)

    cv2.line(img, (x1, y2), (x1, y2 - line_length_y), color, thickness)  #--
bottom-left
    cv2.line(img, (x1, y2), (x1 + line_length_x, y2), color, thickness)

    cv2.line(img, (x2, y1), (x2 - line_length_x, y1), color, thickness)  #-- top-
right
    cv2.line(img, (x2, y1), (x2, y1 + line_length_y), color, thickness)

    cv2.line(img, (x2, y2), (x2, y2 - line_length_y), color, thickness)  #--
bottom-right
    cv2.line(img, (x2, y2), (x2 - line_length_x, y2), color, thickness)

    return img


results = pd.read_csv('./test_interpolated.csv')

# load video
video_path = 'sample.mp4'
cap = cv2.VideoCapture(video_path)

fourcc = cv2.VideoWriter_fourcc(*'mp4v')  # Specify the codec
fps = cap.get(cv2.CAP_PROP_FPS)
width = int(cap.get(cv2.CAP_PROP_FRAME_WIDTH))
height = int(cap.get(cv2.CAP_PROP_FRAME_HEIGHT))
out = cv2.VideoWriter('./out.mp4', fourcc, fps, (width, height))

license_plate = {}
for car_id in np.unique(results['car_id']):
    max_ = np.amax(results[results['car_id'] == car_id]['license_number_score'])
    license_plate[car_id] = {'license_crop': None,
                             'license_plate_number': results[(results['car_id'] ==
car_id) &

(results['license_number_score'] == max_)]['license_number'].iloc[0]}
    cap.set(cv2.CAP_PROP_POS_FRAMES, results[(results['car_id'] == car_id) &
                                             (results['license_number_score'] ==
```

```
max_)]['frame_nmr'].iloc[0])
    ret, frame = cap.read()

    x1, y1, x2, y2 = ast.literal_eval(results[(results['car_id'] == car_id) &
                                              (results['license_number_score'] ==
max_)]['license_plate_bbox'].iloc[0].replace('[ ', '[').replace('   ', '
').replace('  ', ' ').replace(' ', ','))

    license_crop = frame[int(y1):int(y2), int(x1):int(x2), :]
    license_crop = cv2.resize(license_crop, (int((x2 - x1) * 400 / (y2 - y1)),
400))

    license_plate[car_id]['license_crop'] = license_crop


frame_nmr = -1

cap.set(cv2.CAP_PROP_POS_FRAMES, 0)

# read frames
ret = True
while ret:
    ret, frame = cap.read()
    frame_nmr += 1
    if ret:
        df_ = results[results['frame_nmr'] == frame_nmr]
        for row_indx in range(len(df_)):
            # draw car
            car_x1, car_y1, car_x2, car_y2 =
ast.literal_eval(df_.iloc[row_indx]['car_bbox'].replace('[ ', '[').replace('   ',
' ').replace('  ', ' ').replace(' ', ','))
            draw_border(frame, (int(car_x1), int(car_y1)), (int(car_x2),
int(car_y2)), (0, 255, 0), 25,
                        line_length_x=200, line_length_y=200)

            # draw license plate
            x1, y1, x2, y2 =
ast.literal_eval(df_.iloc[row_indx]['license_plate_bbox'].replace('[ ',
'[').replace('   ', ' ').replace('  ', ' ').replace(' ', ','))
            cv2.rectangle(frame, (int(x1), int(y1)), (int(x2), int(y2)), (0, 0,
255), 12)

            # crop license plate
            license_crop =
license_plate[df_.iloc[row_indx]['car_id']]['license_crop']

            H, W, _ = license_crop.shape

            try:
                frame[int(car_y1) - H - 100:int(car_y1) - 100,
                      int((car_x2 + car_x1 - W) / 2):int((car_x2 + car_x1 + W) /
2), :] = license_crop

                frame[int(car_y1) - H - 400:int(car_y1) - H - 100,
                      int((car_x2 + car_x1 - W) / 2):int((car_x2 + car_x1 + W) /
2), :] = (255, 255, 255)

                (text_width, text_height), _ = cv2.getTextSize(

license_plate[df_.iloc[row_indx]['car_id']]['license_plate_number'],
                    cv2.FONT_HERSHEY_SIMPLEX,
                    4.3,
                    17)
```

```
                cv2.putText(frame,

license_plate[df_.iloc[row_indx]['car_id']]['license_plate_number'],
                            (int((car_x2 + car_x1 - text_width) / 2), int(car_y1 -
H - 250 + (text_height / 2))),
                            cv2.FONT_HERSHEY_SIMPLEX,
                            4.3,
                            (0, 0, 0),
                            17)

            except:
                pass

        out.write(frame)
        frame = cv2.resize(frame, (1280, 720))

        # cv2.imshow('frame', frame)
        # cv2.waitKey(0)

out.release()
cap.release()
```