НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

До захисту допущено

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)

"___" червня 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Комп'ютерні системи та компоненти»

спеціальності **123 «Комп'ютерна інженерія»**

на тему: Програмна система розпізнавання типу транспортного засобу

Виконав:

студент IV курсу, групи КВ-61
(шифр групи)

Бідяк Михайло Андрійович _____
(прізвище, ім'я, по батькові) (підпис)

Керівник доц.каф.СПСКС, к.т.н., доцент, Петрашенко А.В.
_____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М.

(назва розділу) (посада, вчене звання, науковий ступінь, прізвище, ініціали) (підпис)

Рецензент_____
_____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2020 року

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма «Комп'ютерні системи та компоненти»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» червня 2020 р.

ЗАВДАННЯ
на дипломний проєкт студента
Бідяка Михайла Андрійовича
(прізвище, ім'я, по батькові)

1. Тема проєкту «Програмна система розпізнавання типу транспортного засобу»,
керівник проєкту Петрашенко Андрій Васильович, к.т.н., доц.каф.СПСКС,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «__» _____ 20__ р. №_____

2. Термін подання студентом проєкту: дивись технічне завдання.

3. Вихідні дані до проєкту: Назва. Програмна система розпізнавання типу транспортного засобу.

4. Зміст пояснювальної записки: перелік скорочень, умовних позначень та термінів; вступ; аналіз предметної області та існуючих рішень; аналіз мов програмування та обґрунтування вибору засобів реалізації; структурно-алгоритмічна організація; аналіз та тестування розроблених програмних засобів; висновки; список використаних літературних джерел.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо) презентація; структурні схеми: структурна схема модулів системи, архітектура нейронної мережі; схеми алгоритмів: алгоритм тренування моделі, алгоритм розпізнавання об'єктів.

6. Консультанти розділів проєкту*

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормконтроль	Клятченко Я.М., доц.каф.СПСКС, к.т.н.	17.05.2020	20.05.2020

7. Дата видачі завдання 12.11.2019.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проекту	10.12.2019	
2.	Розроблення та узгодження технічного завдання	03.03.2020	
3.	Аналіз існуючих рішень	04.03.2020	
4.	Підготовка матеріалів першого розділу дипломного проекту	09.03.2020	
5.	Підготовка матеріалів другого розділу дипломного проекту	15.03.2020	
6.	Підготовка графічної частини дипломного проекту	17.05.2020	
7.	Оформлення документації дипломного проекту	18.05.2020	
8.	Попередній огляд матеріалів диплому на кафедрі	20.05.2020	

Студент _____ Михайло БІДЯК
(підпис) (Ім'я та ПРІЗВИЩЕ)

Керівник проєкту _____ Андрій ПЕТРАШЕНКО
(підпис) (Ім'я та ПРІЗВИЩЕ)

* Консультантом не може бути зазначено керівника дипломного проєкту.

АНОТАЦІЯ

Даний дипломний проект присвячений створенню програмної системи для розпізнавання типу транспортних засобів за допомогою штучного інтелекту, а саме розділу глибокого машинного навчання. Модель машинного навчання буде навчатись на наборах даних, представлених у вигляді набору зображень.

Тема роботи обумовлена необхідністю пришвидшення та автоматизації розпізнавання транспортних засобів в режимі реального часу або на фото та відео матеріалах, для запобігання можливих збитків при некоректному або несвоєчасному розпізнаванню транспорту, людиною.

Розроблені програмні засоби являють собою кросплатформенний десктоп-додаток з користувацьким інтерфейсом, що надає можливість роботи із пристроями введення, а саме веб-камерами та камерами, які підключені до пристрою, а також зображеннями та відеофайлами. Функціональність додатку забезпечує можливість завантаження медіафайлів або потоку даних, обробку їх системою та вивантаження результату, який містить модель, марку, рік випуску та тип кузову транспортного засобу.

Результатом роботи над дипломним проєктом є: розроблена архітектура системи, користувацький інтерфейс та графічні елементи десктоп-додатку, створена глибока нейронна мережа, яка навчалась на великих наборах даних.

Ключові слова:
ПРОГРАМНА СИСТЕМА РОЗПІЗНАВАННЯ ТИПУ ТРАНСПОРТНИХ ЗАСОБІВ, ШТУЧНИЙ ІНТЕЛЕКТ, ГЛИБОКЕ МАШИННЕ НАВЧАННЯ, TENSORFLOW, SSD.

ABSTRACT

This diploma project is devoted to the creation of a software system for recognizing the type of vehicles using artificial intelligence, namely the section of deep machine learning. The machine learning model will learn from data sets presented as a set of images.

The theme of the work is due to the need to accelerate and automate the recognition of vehicles in real-time or in photos and videos, to prevent possible damage in case of incorrect or untimely recognition of transport by a person.

The developed software is a cross-platform desktop application with a user interface that allows you to work with input devices, namely webcams and cameras connected to the device, as well as images and video files. The functionality of the application provides the ability to download media files or data streams, process them by the system and upload the result, which contains the model, make, year of manufacture, and body type of the vehicle.

The result of work on the thesis project is: developed system architecture, user interface, and graphical elements of the desktop application, created a deep neural network, which was studied on a large data set.

Keywords:

SOFTWARE SYSTEM RECOGNITION TYPE OF VEHICLES, ARTIFICIAL INTELLIGENCE, DEEP MACHINE LEARNING, TENSORFLOW, SSD.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
			Документація загальна			
			Новорозроблена			
	А4	ІАЛЦ.045480.002 ТЗ	Програмна система	4		
			розпізнавання типу			
			транспортного засобу			
			Технічне завдання			
	А4	ІАЛЦ.045480.003 ТП	Програмна система	1		
			розпізнавання типу			
			транспортного засобу			
			Відомість технічного			
			проєкту			
	А4	ІАЛЦ.045480.004 ПЗ	Програмна система	71		
			розпізнавання типу			
			транспортного засобу			
			Пояснювальна записка			
	А4	ІАЛЦ.045480.005 Д1	Програмна система	1		
			розпізнавання типу			
			транспортного засобу			
			Структура модулів			
			системи			
			Схема структурна			

Зм	Лист	№ докум.	Підп	Дата	ІАЛЦ.045480.001 ОА		
Розроб.		Бідяк М.А.			Програмна система розпізнавання типу транспортного засобу. *Опис альбому*	Літ.	Лист 1 / Листів 2
Перев.		Петрашенко А.В.				КПІ ім. Ігоря Сікорського, ФПМ, КВ-61	
Н. контр.		Клятченко Я.М.					
Затв.		Романкевич В.О.					

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	А4	ІАЛЦ.045480.006 Д2	Програмна система	1		
			розпізнавання типу			
			транспортного засобу			
			Алгоритм тренування			
			моделі			
			Схема алгоритму			
	А4	ІАЛЦ.045480.007 Д3	Програмна система	1		
			розпізнавання типу			
			транспортного засобу			
			Архітектура нейронної			
			мережі			
			Схема структурна			
	А4	ІАЛЦ.045480.008 Д4	Програмна система	1		
			розпізнавання типу			
			транспортного засобу			
			Алгоритм розпізнавання			
			об’єктів			
			Схема алгоритму			
		Диск CD-ROM	Текст ПЗ. Тексти програм.	1		
			Графічний матеріал.			

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Програмна система розпізнавання типу транспортного засобу»

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення програмної системи розпізнавання типу транспортного засобу за допомогою відео чи фото файлів або в режимі реального часу.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації в періодичних виданнях та електронні статті у мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється

- розпізнавання по фото чи відео файлах;
- розпізнавання в реальному часі;
- класифікація за маркою, моделлю, роком випуску;
- збереження результату класифікації;

- наявність графічного інтерфейсу;
- можливість працювати без доступу до мережі;

5.2 Вимоги до апаратного забезпечення

- оперативна пам’ять: 8 Гб;
- відеокарта: Nvidia;
- пам’ять відеокарти: 2 Гб;
- процессор: 2,4-ядерний процессор;

5.3 Вимоги до програмного та апаратного забезпечення користувача

- операційна система Windows, Linux;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів
1.	Видача завдання на дипломне проектування	12.11.2019
2.	Вивчення літератури за тематикою роботи	10.12.2019
3.	Розроблення та узгодження технічного завдання	03.03.2020
4.	Розроблення структури додатку	05.03.2020
5.	Розроблення дизайну та графічних елементів	07.03.2020
6.	Програмна реалізація додатку	25.03.2020
7.	Тестування додатку	16.04.2020
8.	Підготовка матеріалів текстової частини проекту	19.04.2020
9.	Підготовка матеріалів графічної частини проекту	17.05.2020
10.	Оформлення технічної документації проекту	18.05.2020

ВІДОМІСТЬ ДИПЛОМНОГО ПРОЄКТУ

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
1	А4		Завдання на дипломний проєкт	2	
2	А4	ІАЛЦ.045480.004 ПЗ	Пояснювальна записка	71	
3	А4	ІАЛЦ.045480.005 Д1	Структура модулів системи	1	
4	А4	ІАЛЦ.045480.006 Д2	Алгоритм тренування моделі	1	
5	А4	ІАЛЦ.045480.007 Д3	Архітектура нейронної мережі	1	
6	А4	ІАЛЦ.045480.008 Д4	Алгоритм розпізнавання об’єктів	1	

				ІАЛЦ.045480.003		
	ПІБ	Підп.	Дата			
Розробн.	Бідяк М.А			Відомість дипломного проєкту	Лист	Листів
Керівн.	Петрашенко А.В.				1	1
Консульт.					КПІ ім. Ігоря Сікорського Каф. СПіСКС Гр. КВ-61	
Н/контр.	Клятченко Я.М.					
Зав.каф.	Романкевич В.О.					

Пояснювальна записка
до дипломного проєкту

на тему: Програмна система розпізнавання типу транспортного засобу

Київ – 2020 року

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

ГП (англ. *Graphics Processing Unit, GPU,* укр. *Графічний процесор*) — окремий пристрій персонального комп'ютера, виконує графічний рендеринг. Сучасні графічні процесори дуже ефективно обробляють і зображують комп'ютерну графіку, завдяки спеціалізованій конвеєрній архітектурі вони набагато ефективніші в обробці графічної інформації, ніж типовий центральний процесор.

Краудсорсинг (англ. *Crowd source*, укр. *Натовп ресурс*) — це модель пошуку джерел інформації та залучення до вирішення якоїсь проблеми чи задачі інноваційної діяльності великого кола людей, що призведе до спрощення і пришвидшення процесу.

Об'єктно-орієнтоване програмування (англ. *Object-oriented programming*, укр. *ООП*) — одна з парадигм програмування, яка розглядає програму як множину «об'єктів», що взаємодіють між собою. Основу ООП складають чотири основні концепції: інкапсуляція, успадкування, поліморфізм та абстракція.

ПЗ (англ. *Software,* укр. *Програмне Забезпечення*) — одна із складових інформаційної системи, сукупність програм обробки інформації і програмних документів, необхідних для експлуатації цих програм.

ЦП (англ. *Central processing unit, CPU*, укр. *Центральний процесор*) — функціональна частина комп'ютера, що призначена для інтерпретації команд.

АІ (англ. *Artificial intelligence,* укр. *шту́чний інтеле́кт*) — один з напрямків комп'ютерних наук, який вивчає методи розв'язання задач, для яких не існує способів вирішення. Формалізує проблеми та завдання, які подібні до дій, що виконує людина. Системи штучного інтелекту можуть оперувати даними та самонавчатися.

API (англ. *Application Programming Interface,* укр. *інтерфейс прикладного програмування*) — це сукупність визначень підпрограм, протоколів взаємодії та засобів, таких як, класів, функцій, методів, процедур для створення програмного забезпечення або взаємодії між програмним чи апаратним забезпеченням.

AR (англ. *augmented reality*, укр. *доповнена реальність*), — термін, що позначає всі проекти, спрямовані на доповнення реальності будь-якими віртуальними елементами за допомогою цифрових даних, яке забезпечується комп'ютерними пристроями (смартфонами, планшетами та окулярами) в режимі реального часу.

CNN (англ. *convolutional neural network*, укр. *Згорткові нейронні мережі*) — це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовувався до аналізу візуальних зображень.

CSV (англ. *comma-separated values,* укр. *значення, розділені комою*) — файловий формат, котрий є обмежувальним форматом для представлення табличних даних, у якому поля відокремлюються символом коми та переходу на новий рядок.

CUDA(англ. *Compute Unified Device Architecture*) — програмно-апаратна архітектура паралельних обчислень, яка дозволяє істотно збільшити обчислювальну продуктивність завдяки використанню графічних процесорів фірми Nvidia.

CV (англ. *Computer vision*, укр. *Комп'ютерний зір*) — теорія та технологія створення машин, які можуть проводити виявлення, стеження та визначення об'єктів. Як наукова дисципліна, комп'ютерний зір належить до теорії та технології створення штучних систем, які отримують інформацію у вигляді зображень.

Dataset(англ. *Data set*, укр. *Набір даних*) — колекція однотипних даних, що застосовується в задачах машинної обробки даних та навчанні. Найчастіше,

в машинному навчанні, набори даних представлені набором зображень, назв та розташуванням предмету.

DL (англ. *deep learning, deep machine learning,* укр. *глибинне машинне навчання*) — це функція штучного інтелекту, яка імітує роботу мозку людини при обробці даних та створенні зразків для використання при прийнятті рішень. Глибоке навчання - підмножина машинного навчання штучному інтелекту (AI), яка має мережі, здатні навчатися без нагляду за неструктурованими або не маркованими даними. Також поняття зустрічається як глибоке нейронне навчання або глибока нейронна мережа.

GIL (англ. *Global Interpreter Lock*, укр. *Глобальне блокування інтерпретатора*) — це механізм інтерпретатора, що гарантує те, що в кожен момент часу виконується код лише одного процесу, для уникнення конкурентного доступу до спільних структур даних.

GUI (англ. *Graphical user interface,* укр. *Графічний інтерфейс користувача*) — тип інтерфейсу, який дає змогу користувачам взаємодіяти з програмним продуктом через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації.

IoT (англ. *Internet of Things, укр. Інтернéт речéй*) — концепція мережі, що складається із взаємозв'язаних фізичних пристроїв, які мають вбудовані давачі, а також програмне забезпечення, що дозволяє здійснювати передачу і обмін даними між фізичним світом і комп'ютерними системами, за допомогою використання стандартних протоколів зв'язку. Окрім давачів, мережа може мати виконавчі пристрої, вбудовані у фізичні об'єкти і пов'язані між собою через дротові чи бездротові мережі. Ці взаємопов'язані пристрої мають можливість зчитування та приведення в дію, функцію програмування та ідентифікації, а також дозволяють виключити необхідність участі людини, за рахунок використання інтелектуальних інтерфейсів.

IoU (англ. *Intersection Over the Union*, укр. *Перетин над об'єднанням*) — це відношення між областю, що перетинається й об'єднаною ділянкою для двох регіонів.

mAP (англ. *Mean average precision*, укр. *Усереднене середня точність*) — це середнє значення середніх балів точності для кожного запиту.

ML (англ. *machine learning*, укр. *машинне навчання*) — це підгалузь штучного інтелекту в галузі інформатики, яка часто застосовує статистичні прийоми для надання комп'ютерам здатності «навчатися», без того, щоби бути програмованими явно. Досліджує методи, які дозволяють комп'ютерам покращувати свої характеристики на основі отриманого досвіду.

NLP (англ. *Natural-language processing*, укр. *Обробка природної мови*) — загальний напрям інформатики, штучного інтелекту та математичної лінгвістики. Він вивчає проблеми комп'ютерного аналізу та синтезу природної мови. Стосовно штучного інтелекту аналіз означає розуміння мови, а синтез — генерацію розумного тексту.

Open Source (англ. *open-source software*, укр. *Відкрите програмне забезпечення*) — програмне забезпечення з відкритим сирцевим кодом.

SSD (англ. *Single Shot Multibox Detector*, укр. *детектор з одним пострілом*) — це модель сімейства мереж, які передбачають обмежувальні границі об'єктів у заданому зображенні. Це проста, цільна до кінця, одна мережа, яка робить багато кроків, одним проходом. Інші мережі, які намагаються досягти того самого завдання, під час публікації роблять більше проходів.

Tk (англ. *Toolkit*, укр. *набір інструментів*) — крос-платформна бібліотека базових елементів графічного інтерфейсу, поширювана з відкритими вихідними текстами. Tk був розроблений як розширення для інтерпретованої мови програмування Tcl.

ВСТУП

У повсякденному житті дуже часто люди стикаються з задачею розпізнавання образів, у тому числі об'єктів, структур, текстів і зображень. Взагалі, дуже часто, задача ідентифікації та класифікації використовується у повсякденному житті. Люди розпізнають об'єкт із низки інших, класифікують, роблять висновки про предмет, спираючись на класифікацію.

У загальному, задача розпізнавання образів — це задача віднесення вихідних даних до певного класу за допомогою виділення істотних ознак, що характеризують ці дані, із загальної маси несуттєвих даних. Саме дані грають головну роль в розпізнаванні та класифікації об'єкта. Якщо даних недостатньо або їх не має, то класифікація стає неможлива, ми не можемо розпізнати предмет чи образ.

Задача розпізнавання образів набула особливого значення в умовах інформаційних перевантажень, коли людина не справляється з лінійно-послідовним розумінням надходжень до неї повідомлень, в результаті чого його мозок перемикається на режим одночасності сприйняття і мислення, якому таке розпізнавання властиво. Не випадково, задача розпізнавання образів з'явилася в полі міждисциплінарних досліджень - у тому числі, зв'язаних з роботою зі створення штучного інтелекту (AI).

Таким чином, актуальною є задача розпізнавання об'єктів людиною, метою даного дипломного проєкту є створення програмної системи розпізнавання типу транспортного засобу. За допомогою технології штучного інтелекту, а саме напряму машинного навчання (ML) та комп'ютерного зору (CV), буде розроблений десктопний додаток для розпізнавання та класифікації типу транспортних засобів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1 Загальні положення та аналіз предметної області

Розвиток автомобілебудівництва в світі досяг величезних масштабів і в майбутньому буде тільки збільшуватись. Кожного року, або й частіше, з'являються нові марки, моделі авто, нові дизайни для вже існуючих транспортних засобів. Об’єми продажу транспортних засобів є завжди великими, про що говорить статистика. За 2017 рік в світі було продано 95.5 млн новітніх авто. Об'єм продажу світового авторинку за підсумками 2018 року склав 94 мільйони 791 тисячу легкових і легких комерційних авто. У 2019 році продаж транспортних засобів становить 88 мільйонів 720 тисяч[1]. Потрібно враховувати, що кожного року дуже багато авто обмінюються або продаються на вторинному ринку. Також, значна частина транспортних засобів перетинає кордон транзитом або для інших цілей, таких як доставка і тому подібне. Виходячи з цього, відбувається великий потік авто, різних марок, дизайнів, років випуску.

Таким чином, виникає задача ідентифікування транспортних засобів, особливо коли це потрібно зробити точно та швидко, без великих затрат, можливо навіть без доступу до інтернет з'єднання. Думка, яка приходить першою, про те, що марку авто транспорту можна визначити за емблемою виробника, хибна. Не кожен, навіть автолюбитель, визначить марку машини тільки по значку. Зображення знака має глибоку історію. Процес становлення будь-якого з них відбувався дуже довго, адже не кожне автомобільне підприємство починало відразу виробляти саме транспортні засоби. Тому значки, як і машини, постійно вдосконалювалися. При цьому коріння обох знаходяться глибоко в минулому столітті. Слід зазначити, що емблем в світі стільки, скільки і марок авто, а можливо, навіть більше. Всі марки автомобілів світу не можна перерахувати і порахувати. У жодному джерелі немає точної інформації з цього приводу. Безліч марок випускається в межах однієї країни,

але не всі люди знають про їхнє існування. На сьогоднішній день ніхто не відповість на питання, скільки саме зареєстровано автомобільних марок. При цьому найвідоміших налічується більше 60 штук.

У рамках даного дипломного проекту було проведено опитування серед студентів групи КВ і всі вони відповіли, що знають менше ніж 60 марок автомобілів та не можуть точно і швидко визначити модель чи марку авто. Визначення року випуску автомобіля ще складніша задача, яка потребує ще більшої уваги до деталей.

Проаналізувавши дані опитування, я прийшов до висновку, що потрібна програмна система розпізнавання транспортних засобів, що дозволяє аналізувати дані в реальному часі, фото або відео файли, яка не потребує постійного доступу інтернет з'єднання, не використовує великі ресурси платформи. Це значно спростить та пришвидшить, для більшості людей, ідентифікацію типу транспортних засобів, марки, моделі та року випуску.

Розроблене програмне забезпечення у вигляді десктопного додатку спрямоване на вирішення вище описаного завдання та задовольняє вимоги до системи.

1.2 Аналіз існуючих програмних рішень

У рамках дипломного проєкту досліджено уже існуючі системи та програмні засоби, які дозволять розпізнавати об'єкти в тому числі і транспортні засоби. Метою даного дослідження було виявлення програмних продуктів, які б вирішували дану задачу або допомогли у її аналізі та відповідали вимогам до системи.

1.2.1 Axiomtek Vehicle Detection System

Система виявлення транспортних засобів помічає транспортні засоби із зайвою вагою, які рухаються до надземних перешкод, таких як мости, тунелі та інші споруди, і це попереджає водіїв індивідуально. Попередження відбувається за допомогою спеціальних світлових пристроїв. Ця система може розпізнавати дуже багато критеріїв:

- Визначення швидкості;
- Визначення кількості транспортних засобів;
- Виявлення зупинки транспортного засобу;
- Попередження неправильного руху;
- Заповненість шляху;
- Розрив між транспортними засобами;
- Виявлення черги.

Система складається з підсистем:

- Відеоспостереження за трафіком,
- Управління потоком руху,
- Контроль воріт платних станцій,
- Дистанційний моніторинг споруди та центральне управління.

Дана система призначена для контролю шляхів та шляхопроводів. Використовує не тільки програмне забезпечення, яке розпізнає транспортні засоби, а й багато інших, супутніх приладів, таких як датчик руху, відеомоніторинг, моніторинг навколишнього середовища, системи контролю воріт та інші. Тобто, це комплексна система, яка використовує досить велику кількість пристроїв та ресурсів[2].

1.2.2 Object detection 2.0

Безкоштовне програмне забезпечення для виявлення об'єктів в режимі реального часу та розпізнавання номерів транспортних засобів. Поточна версія підтримує Windows 7 - 10 (64-розрядні). Дана програма, дозволяє виявляти об'єкти в режимі реального часу на стандартному комп'ютері. Звичайно, якщо на пристрої є відеокарта і підтримує CUDA. Базується на сучасних технологіях на основі нейронних мереж, що навчаються на великих наборах даних. Має такий функціонал:

- У разі конкретної події програмне забезпечення для виявлення об'єктів автоматично завантажує відео на сервер.
- Дає можливість переглянути всі трансляції та події за допомогою YouTube каналу.
- Захоплювати зображення з декількох веб-камер або IP-камер, екрана монітора, файлів та інших пристроїв.
- Виявлення руху дозволяє відстежувати та зберігати відео, як тільки рух виявлено.

Для використання цього програмного продукту, потрібна спеціальна платформа, велика кількість вільного місця на дисковому просторі, постійне інтернет з'єднання. Програмне забезпечення не розпізнає тип транспортного засобу, марку, модель чи рік випуску[3].

1.2.3 Blippar’s Vehicle Recognition App

Використовуючи AI та автомобільного розпізнавання, "Blippar" розробив додаток для розпізнавання автомобілів, який може обробляти інформацію

швидше, ніж це може зробити більшість людей. Цей додаток для мобільних телефонів. Призначений для розпізнавання автомобілів Американського автопрому у яких рік випуску більший ніж 2000-ий. Технологія візуального пошуку порівнює фотографії, зроблені в Інтернеті, та в базах США, знаходячи те, що найкраще підходить до заданої картинки[4]. Після знайдення співпадіння, описує авто та видає результат користувачу. Також, програмний продукт може розпізнавати марку авто та модель по відео. Після розпізнавання транспортного засобу, користувачу, за допомогою AR-технології, надається можливість розглянути авто на 360 градусів, побачити салон, та середній рейтинг від користувачів. Даний програмне рішення, об'єднує глибоке навчання, комп'ютерний зір та доповнену реальність. Доступний додаток тільки для Американських користувачів смартфонів на операційній системі Android або IOS. При цьому потрібне постійне інтернет з'єднання.

1.2.4 Vehicle Recognition App Assists Car Buyers

Завдяки технології розпізнавання об'єктів CarCapture розроблений для того, щоб дозволити споживачеві зробити знімок задньої частини автомобіля та миттєво отримати детальну інформацію. Vehicle Recognition App [5] використовує CV для розпізнавання транспортного засобу по зображенню, яке завантажується в додаток. Фото саме задньої частини автомобіля використовується тому, що воно має більш виразний стиль, і це вигляд, який ви, швидше за все, бачите на парковці. Програмний продукт здатний визначити: марку, модель, тип кузова, рік випуску. Також, виводиться деяка інформація про авто, відгуки споживачів, зображення з інших ракурсів, потенціальні дилерське розташування. Додаток доступний тільки для смартфонів з операційними системами Android та iOS. State Farm створив базу фотографій

понад 1000-чі автомобілів (рахуючи кожен модельний рік як різний транспортний засіб), зроблених з 2000-го року. Застосування працює найбільш коректно для користувачів Америки та Канади, тому що зображення для бази даних сформувались саме там. Тим не менш, программа дуже часто помиляється, навіть при чіткому фото. Потребує постійного підключення до мережі.

1.3 Висновок та аналіз вимог до функціональності системи

У результаті аналізу та дослідження існуючих рішень, огляду сфери розпізнавання образів та об'єктів, можна стверджувати, що завдання розпізнавання і класифікації є актуальною. На даний момент існують деякі рішення, які використовуються і зараз. Але, все таки, ці рішення не ідеальні, деякі з них вирізняються жорсткою логікою, прив'язкою до локації, не точністю у класифікації, обов'язковим доступом до мережі або залученням великих апаратних ресурсів.

У процесі аналізу існуючих рішень було сформовано наступні функціональні вимоги до системи розпізнавання типу транспортних засобів:

- Застосунок має надавати можливість швидкої та якісної ідентифікацію та класифікацію транспортного засобу. Тобто, виводити всю можливу інформацію, марку, модель, рік випуску, тип кузову та робити це точно та за мінімальний проміжок часу.
- Система має забезпечувати можливість розпізнавання транспорту по фотом та відео файлах, а також у режимі реального часу.
- Програмне рішення має коректно працювати з багатьма об'єктами, великою кількістю транспортних засобів на фото, відео.

- Система має працювати з файлами різного формату, якості, величини, чіткості.
- Додаток повинен розпізнавати об'єкти з будь-якого ракурсу та коректно працювати з будь-яким фоном на фото або відео зображенні.
- Система не має супроводжуватись залученням зайвих апаратних ресурсів.
- Програмне рішення має давати змогу збереження результату класифікації.

Розроблене ПЗ має відповідати таким нефункціональним вимогам:

- Можливість використання у будь-якій точці світу.
- Розпізнавання та класифікація має ефективно виконуватись без доступу до Інтернет з'єднання.
- Застосунок має мати зручний для користувача інтерфейс, який дає можливість вибору потоку даних над якими буде проводитись класифікація, камер з якими буде працювати додаток.

Після проведення порівняльного аналізу існуючих рішень, можна прийти до висновку, що при виконанні всіх вимог до ПЗ, проект набуде унікальності, адже станом на сьогодні дана функціональність не реалізована жодним аналоговим рішенням.

2 АНАЛІЗ МОВ ПРОГРАМУВАННЯ ТА ОБҐРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РЕАЛІЗАЦІЇ

2.1 Вибір цільової платформи для реалізації проекту

Рорзроблювана програмна система розпізнавання типу транспортного засобу, до якої поставлені вимоги швидкої та якісної ідентифікації, буде використовувати складні та громіздкі алгоритми. Взагалі, задача розпізнавання використовує засоби машинного навчання, зокрема галузь глибокого навчання (DL)[6], що ґрунтується на наборі алгоритмів, які намагаються моделювати високорівневі абстракції в даних, застосовуючи глибинний граф із декількома обробними шарами, що побудовано з кількох лінійних або нелінійних перетворень. Існують багато алгоритмів, але ті, які здатні осилити завданням комп'ютерного зору, мають декілька мільйонів параметрів та є досить складними і важкими з точки зору обчислень та пам'яті, що буде використовувати програма. Для того, щоб уникнути необхідності стискання моделі, зменшення кількості параметрів, класів, тестових та перевірочних вибірок, що призведе до зменшення коректності роботи програми, потрібно вибрати цільову платформу, яка має досить великі ресурси пам'яті, обчислювальні потужності, як наслідок швидку взаємодію з системою. Також, потрібно враховувати, що додаток має зберігати дані обчислень, користувач може додавати дані для класифікації.

Потрібно вибрати оптимальною платформу для застосунку системи розпізнавання типу транспортних засобів, яка здатна впоратись з усіма поставленими задачами, та задовольнити всі вимоги. Дана архітектура має відкривати доступ до вагомих ресурсів обчислення. Саме використання центрального та графічних процесорів дасть змогу розкрити весь потенціал системи.

У випадку використання відеокарти, це значно прискорить усю систему, адже обчислення саме на відеокартах найбільш ефективні для комп'ютерного зору та глибокого навчання. Центральні процесори здатні виконувати операції обчислення швидко одна за одною, навіть якщо у ньому є багато ядер та потоків. Більшість користувацьких задач для ЦП не потребує паралельних обчислень і працює на одному ядрі. Сучасні ГП дуже ефективно обробляють і зображують комп'ютерну графіку, завдяки спеціалізованій конвеєрній архітектурі вони набагато ефективніші в обробці графічної інформації, ніж типовий центральний процесор. Графічний процесор набагато швидше обробить пікселі зображення для розпізнавання, тому що у нього є багато обчислювальних ядер, які працюють паралельно, саме задля обробки такого типу.

У випадку з відеокартами, потрібно підкреслити, що більшість відеокарт мають у своєму архітектурному складі TENSOR ядра[7]. Такі графічні процессори використовують потужність тензорних ядер, революційної технології, що забезпечує неперевершену продуктивність штучного інтелекту. Тензорні ядра дозволяють прискорювати великі матричні операції, які лежать в основі штучного інтелекту. З їх допомогою можна здійснювати множення матриць зі змішаною точністю і зводити обчислення в одну операцію. Завдяки сотням тензорних ядер, що працюють паралельно в одному GPU NVIDIA, продуктивність і енергоефективність значно підвищуються. Також, потрібно зазначити, що за допомогою цих ядер виконується CUDA[8] - програмно-апаратна архітектура паралельних обчислень, яка робить більш ефективні транзакції між пам'яттю центрального процесора і відео пам'яттю, апаратно підтримує цілочисельні і побітові операції. Архітектура CUDA дає розробнику можливість на свій розсуд організовувати доступ до набору інструкцій графічного прискорювача й керувати його пам'яттю.

Проаналізуємо переваги та недоліки існуючих платформ, які могли б підійти для розробки даної системи.

Говорячи про недоліки мобільних застосунків, можна сказати, що вони не мають можливість використовувати потужні обчислювальні ресурси, також мають досить малий обсяг пам'яті, як графічної, оперативної та і вбудованої. При розробці для смартфонів є необхідність розробки для декількох операційних систем. Також, великий недолік те, що неможливо буде застосовувати будь-яку іншу камеру, окрім камери самого смартфону. Перевагами програм для мобільних пристроїв: простота у використанні, мають хорошу продуктивність, а ще, за статистикою, в них проводять найбільше часу.

Недолік веб-додатку це те, що додаток для розпізнавання буде на архітектурі клієнт-сервер. Це призводить до того, що потрібен постійний доступ до інтернет мережі. Така архітектура працює набагато повільніше мобільних чи десктоп додатків. Перевагами таких програмних продуктів: такі додатки встановлюються один раз та оновлюється на сервері; не обмежені фізичним розташуванням; не обмежені в доступі з різних пристроїв.

Порівнюючи усі платформи, я прийшов до висновку, що цільовою платформою для реалізації проекту є десктоп додаток. Вона задовольняє всі функціональні і нефункціональні вимоги до ПЗ. Ця платформа дозволяє використовувати потужні ресурси обчислення для комп'ютерного зору та глибокого навчання. Забезпечує більшу продуктивність завдяки взаємодії з ОС та архітектурою ЦП і ГП. Це дозволить швидко та найбільш коректно класифікувати об'єкти, дасть можливість розпізнавати транспортні засоби на файлах різної якості, величини, чіткості. Дана платформа дозволяє підключати довільну кількість відеокамер, що можливо буде потрібно для системи розпізнавання в реальному часі. Також, не буде залучати зайві апаратні ресурсів та може працювати без доступу до мережі Інтернет.

2.2 Вибір мови програмування для розроблення системи

У сфері штучного інтелекту та комп'ютерного зору великого поширення набули такі мови програмування, як C++ та Python. Саме за допомогою цих мов відбувається програмування десктоп-застосунків. Таким чином, дані мови програмування найбільш вдало підходять для розробки проєкту. Для вибору оптимальної мови програмування було проведено аналіз мов C++ та Python.

2.2.1 C++

Найстаріші серед найбільш поширених мов програмування. Володіючи можливостями одночасно як низькорівневої, так і високорівневої мови програмування, в контексті machine learning, C ++ забезпечує більш високий рівень контролю та ефективності, ніж інші мови програмування.

C++ – поширена мова в сферах розробки апаратного забезпечення, оскільки код, написаний на C++, займає суттєво менше місця ніж низькорівневі мови такі як Assembler, та більш підходить для реалізації алгоритмів у сфері IoT. Також, гнучкість мови добре підходить для ресурсномістких додатків, і підмножина програм штучного інтелекту тут - не виняток. Враховуючи, що мова C++ жорстко типізована та компільована вона може виконувати задачі з відносно високою швидкістю. Для мови існує багато фреймворків з широкою підтримкою, що дещо спрощує написання програм. До переваг мови можна віднести:

- Компільованість та типізованість.
- Масштабованість.

- Можливість працювати з пам'яттю, адресами, портами на низькому рівні.
- Швидкість роботи.
- Підтримуються різні стилі та типи програмування, у тому числі ООП, функціональне програмування, метапрограмування.

C++ досить стара мова програмування, тому вона має ряд своїх недоліків. Основна з них це те, що для створення нового додатку потрібно написати великий об'єму складного коду, що займає багато часу і може викликати деякі труднощі в обслуговуванні. Дуже часто, початківці програмісти, в розробці нового проекту, допускають невимушені помилки. Тому, до недоліків можна віднести:

- Складність синтаксису.
- Погана підтримка модульності додатку.
- Потреба повністю керувати пам'яттю, можливе протікання.
- Потребує високого рівня підготовки розробника.

Звичайно, мову C++ часто використовують у продуктах, пов'язаних з компютерним зором, коли виникає необхідність швидкої обробки багатьох зображень. Одним з варіантів використання мови може бути те, що її можна використати у основному додатку, написаному на іншій мові, наприклад Python. Таким чином, можна описувати деякі частини алгоритму, модулі обробки зображення мовою C++, що дозволить пришвидшити роботу програми.

2.2.2 Python

Python - це високорівнева мова програмування, яка має безліч різних способів застосувань, включаючи науку про дані, штучний інтелект і

внутрішню веб-розробку. Мова була створена Python Foundation на початку 1990-х і є потужним інструментом для аналізу даних, широко використовується в технології великих даних. Вважається, найбільш популярною мовою серед усіх, що використовуються для штучного інтелекту, завдяки його синтаксичній простоті. Внаслідок цього, мова дуже легко піддається вивченню, що дозволяє швидко та ефективно реалізовувати алгоритми. Час розробки цією мовою, в порівнянні з іншими, досить низький, що є великою перевагою в наукових сферах, де потрібно часто експериментувати та отримувати швидкий результат. AI досить швидко розвивається, Python теж, завдяки активній спільноті, яка постійно його покращує. З'являється безліч готових бібліотек для машинного навчання. Це досить відкрита мова, вона побудована на базі технології Open Source, тому розробники можуть доповнювати мову, отримувати доступ до будь-якого стеку мови.

Тож, до переваг мови програмування Python можна віднести:

- Простота та лаконічність.
- Безліч фреймворків для машинного навчання.
- Малий час розробки.
- Відкритий код мови.
- Динамічна типізація.
- Підтримується розбиття програм на модулі.
- Підтримка різних стилів та типів, парадигм програмування, таких як ООП, функціональні та процедурно-орієнтовані стилі програмування.
- Автоматичний збір сміття, не потрібно слідкувати за витоком пам'яті.
- Кросплатформність, мову можна адаптувати майже під будь-яку операційну систему.
- Інтеграція з C/C++, якщо можливостей мови не достатньо.

- Механізм обробки виключень.

Python дещо неоднозначний у своїх перевагах. Наприклад, мова мінімізує участь розробника у керуванні пам'яті, у цьому полягає простота мови, але динамічна типізація, накладає певні обмеження на розробника, змушує більш ретельно тестувати роботу програми, також, через це, робота з машинним навчанням може викликати проблеми. Одна з них, є складність відслідковування помилок в коді, що пов'язано зі зростанням кодової бази програми і, відповідно, з її складністю. У деяких випадках аудит коду може вилитись у значні фінансові витрати або забирати занадто багато часу, що впливає на продуктивність проекту. Також, динамічна типізація, суттєво впливає на швидкість виконання програми. Розробник може не витрачати час на явне задання типу даних, але цю роботу доведеться виконати інтерпретатору, що уповільнює програму.

Отже, недоліки мови Python:

- Повільна швидкість роботи коду.
- Неможливість керувати пам'яттю.
- Багатопотоковість в Python може мати непередбачувані результати.
- Динамічна типізація.

Оскільки інтерпретатор Python використовує GIL, виконання лише одного потоку в межах одного процесу за один квант часу, то використання паралельних обчислень стає складним завданням для розробника, якому необхідно реалізовувати механізм міжпроцесорної взаємодії, що теж вплине на час виконання.

Python дуже поширений і сильний у десктоп додатках та серверних платформах, підтримує C/C++ інтеграцію. Мова здатна впоратись з багатьма завданнями, у тому числі з розробкою алгоритмів машинного навчання, для цього розроблено багато бібліотек і фреймворків. З огляду на це, можна зробити висновок, що хорошим вибором для розробки системи розпізнавання

транспортних засобів є саме мова програмування Python. Вона добре підходить для реалізації десктоп додатку, для обробки зображень та для реалізації нейронної мережі.

2.3 Вибір бібліотеки для машинного навчання

Одна з найбільших переваг Python - широкий спектр бібліотек. Бібліотеки є наборами підпрограм та функцій, написаних цією мовою. Хороший комплект бібліотек може полегшити здійснення складних завдань без необхідності написання великого об'єму коду програми. Машинне навчання багато в чому ґрунтується на математиці. Зокрема, на математичній оптимізації, статистиці та теорії ймовірності. Бібліотеки Python допомагають навіть не маючи змістовних знань, працювати з машинним навчанням. З огляду на те, що нам потрібна бібліотека, яка буде підтримувати паралельне обчислення для графічних процесорів, я вибрав декілька відповідних фреймворків. Розглянемо та проаналізуємо існуючі бібліотеки для створення моделей машинного навчання.

2.3.1 MXNet

MXNet являє собою фреймворк глибокого машинного навчання з відкритим кодом, що розробляється компанією Amazon. Фреймворк початково підтримує велику кількість мов програмування, таких як C++, Python, R, Go, Scala. Використовується такими компаніями-гігантами, як Microsoft, Intel. Компанія Apple використовує його після придбання компаній Dato, Turi. Бібліотека для гнучких та швидких дослідницьких прототипів. Основний

акцент зроблений на тому, що фреймворк дуже ефективно паралелится на безлічі графічних процесорів, що дозволяє швидко навчатись моделям. Зокрема, його хорошу роботу продемонстровано на Amazon Web Services. Фреймворк відомий своєю вискою масштабованістю, тому він використовується великими компаніями в основному для розпізнавання мови і почерку, NLP і прогнозування.

Фреймворк MXNet[9] має багато переваг:

- Досить швидкий, гнучкий і ефективний в питаннях роботи з алгоритмами глибокого навчання.
- Забезпечує просунуту підтримку ГП.
- Може запускатись на будь-якому пристрої.
- Має високопродуктивне імперативне API.
- Вкрай масштабований.
- Забезпечує хорошу підтримку безлічі мов.

Недоліки фреймворку:

- Не велика спільнота.
- Не популярний в наукових сферах.
- У більшості підтримується тільки для великих компаній.

Виходячи з вищеперелічених аргументів, MXNet - це хороший фреймворк для великих промислових проектів, але він ще досить молодий, слід пам'ятати про те, що можна не отримати технічну підтримку у вирішенні свого питання так швидко, як хотілося б.

2.3.2 TensorFlow

Відкрита програмна бібліотека для машинного навчання від Google, який призначений для проектування, створення і вивчення моделей глибокого навчання. Ви можете використовувати TensorFlow[10], щоб виробляти чисельні обчислення. Само по собі це не здається специфічним, проте ці обчислення проводяться за допомогою data-flow графів. У цих графах вершини представляють собою математичні операції, в той час як ребра є дані, які зазвичай подаються у вигляді багатовимірних масивів або тензорів, які об'єднуються між цими ребрами.

Google активно використовує власний фреймворк глибокого навчання для таких великомасштабних сервісів як Gmail і Google Translate. TensorFlow вже застосували до своїх сервісів і такі значні бренди як Uber, Airbnb, Dropbox, DeepMind.

Найбільш зручною клієнтською мовою роботи з TensorFlow є Python, але доступні і експериментальні інтерфейси на JavaScript, C++, Java і Go. Спільнота open source розробила також рішення для C# і Julia.

TensorFlow хороша для складних проектів, таких як створення багатошарових нейронних мереж. Вона використовується для розпізнавання голосу або картинок і додатків для роботи з текстом. Як і більшість фреймворків з глибоким навчанням, TensorFlow написаний з API Python через двигун C/C++, завдяки чому він працює швидше. Активно може працювати з CUDA ядрами ГП, ефективно проводить паралельні обчислення.

Аналізуючи дану бібліотеку, можна виділити такі переваги:

- Ефективна в питаннях роботи з алгоритмами машинного навчання.
- Забезпечує просунуту підтримку ГП.
- Пітримує безліч мов програмування.

- Активна спільнота, написано багато розширень, посібників та документації. Підтримується великою спільнотою розробників і технічними компаніями.
- Пропонує потужні засоби моніторингу процесу навчання моделей і візуалізації (Tensorboard).
- Можливість виконувати частину графа, що дає можливість ефективного відлагодження.
- Підтримує розподілене навчання.
- Статичний обчислювальний граф абстракції.
- Паралелізм та використання різних апаратних частин.
- Кросплатформність.
-

Недоліки даної бібліотеки:

- Обчислювальний граф чистий Python, тому повільний.
- Не багато наперед тренованих моделей.
- Використовує Python для завантаження кожної нової навчальної партії.
- Динамічна типізація.
- Низькорівнева бібліотека.

Це низькорівневий інструмент, тому потрібно ретельно продумувати архітектуру нейромережі, правильно оцінювати розмірність і обсяги вхідних і вихідних даних. Таким чином, робота з TensorFlow вимагає написання значної кількості програмного коду. Також, він оперує статичним обчислювальним графом. Тобто спочатку ми визначаємо граф, далі запускаємо обчислення і, якщо необхідно внести зміни в архітектуру, заново навчаємо модель. Такий підхід обраний заради ефективності, але потрібно враховувати це, коли змінюємо архітектуру.

2.3.3 Порівняння бібліотек

Проаналізувавши бібліотеки машинного навчання, можна зрозуміти, що вони можуть обидві підійти для розробки системи розпізнавання типу транспортних засобів. Таким чином, можна порівняти бібліотеки MXNet та TensorFlow по критеріям, які задовольняють вимоги та важливі для реалізації системи розпізнавання. У процесі було створено таблицю 2.1 для порівняння.

Таблиця 2.1- Порівняння фреймворків для машинного навчання

Програмне забезпечення	TensorFlow	MXNET
Автор	Google	Distributed (Deep) Machine Learning Community, Amazon
Відкритий програмний код	Так	Так
Платформа	Linux, Mac OS X, Windows, Android, iOS.	Linux, Mac OS X, Windows, AWS, Android, iOS.
Підтримка мов програмування	C++, Python, JavaScript, Java, Go, C#, Julia	C++, Python, Julia, Matlab, R, Scala
Згорткові мережі	Так	Так

Продовження таблиці 2.1

Засоби моніторингу і візуалізації навчання	TensorBoard	Немає
Підтримка CUDA	Так	Так
Програмне забезпечення	TensorFlow	MXNET
Активна підтримка спільнотою	Так	Ні
Можливість ефективного відлагодження	Так	Ні
Низькорівневі оператори,	Так	Ні
Підтримка паралельних і розподілених обчислень	Multi-GPU	Розподілені обчислення

Отже, після аналізу та порівняння існуючих фреймворків, проаналізувавши таблицю 2.1, для використання в даному проекті було обрано фреймворк TensorFlow, оскільки в ньому підтримується паралельні та

розподілені обчислення, згорткові мережі, які використовуються у розпізнаванні образів, підтримує обрану платформу та мову програмування, має підтримку CUDA, особливо добре працює із NVIDIA GPU. Крім цього, має можливість ефективного відлагодження за допомогою виконання частини графа. У складі бібліотеки є спеціальний засіб моніторингу процесу навчання моделей і візуалізації. Також, фреймворк використовує активну підтримку спільноти та дуже швидко розвивається, а авторитету додають компанії-гіганти, які використовують саме цю бібліотеку.

2.4 Огляд існуючих наборів даних транспортних засобів

Перш ніж розробляти додаток із машинним навчанням, потрібно чітко визначити, на яких даних буде натренована модель, вартість цих даних, скільки часу займає збір. Оскільки можливим може бути те, що затрачений час та ресурси не принесуть бажаного результату та очікуваного прибутку.

Головним чином, вибраний dataset впливає на ефективність моделі розпізнавання. Вважалось, що коректні, ефективні, успішні моделі побудовані на великій кількості тренувальних даних. Насправді ж, набагато важливішою є якість тренувальних даних, ніж їх кількість. На просторах мережі інтернет можна знайти велику кількість наборів даних різних форматів. Я постарався вибрати найкращий із представлених, який задовольняє вимоги майбутньої систем, для побудови правильної моделі розпізнавання транспортних засобів.

2.4.1 KITTI Object Detection with Bounding Boxes

Цей набір даних, взятий із “Karlsruhe Institute of Technology”, складається із зображень із розділу виявлення об'єктів. Цей набір даних включає в себе понад 14000 фото, з них 7518 тестових зображень та 7481 навчальних зображень, також в наборі присутній файл із обмежувальними полями. Усі зображення кольорові та зберігаються у форматі PNG. Набір містить 8 різних класів, але оцінюються тільки два класи, “автомобіль” і “пішохід”. Датасет створений для навчання моделі, яка буде використовуватись штучним інтелектом для автономного керування автомобілем. Даний набір даних нам не дуже підходить, адже ми створюємо систему розпізнавання типу транспортних засобів, яка буде визначати модель та марку авто.

2.4.2 GTI Vehicle Image Database

Ці дані були зібрані та створені, за допомогою відеореєстратора, який був встановлений в машині. Фрагменти відео були розбиті на окремі зображення та додано визначення рамки об'єкта. База даних включає в себе 3425 зображень задніх частин транспортних засобів, знятих з різних точок зору, та 3900 зображень, доріг, що не містять транспортних засобів. Зображення підбираються для максимальної репрезентативності класу транспортного засобу, що передбачає природно високу мінливість. Зображення витягнуті таким чином, що вони не ідеально підходять до контуру транспортного засобу, щоб зробити класифікатор більш стійким до зрушень на етапі формування гіпотези. Зображення мають розміри 64x64 та обрізані із послідовностей 360x256 пікселів, записаних на шосе Мадрида, Брюсселя та Туріна.

Даний датасет, створений для класифікації транспортних засобів, який може бути використаний у відеореєстраторах або у системах автономного керування авто, але дані містять тільки задні частини транспортних засобів, не містять іншої інформації про автомобілі.

2.4.3 Stanford Cars Dataset

Датасет прямо із Stanford AI Laboratory. Набір даних “Автомобілі” складається із 16185 зображень транспортних засобів, що містить 196 класів автомобілів. Дані розділені на 8144 навчальних зображень та 8041 тестових зображень, де кожен клас розподілений по ним, приблизно 50 на 50. Класи, як правило, містять в собі опис марки, моделі та року випуску транспортного засобу. Набір даних побудований так, щоб модель могла ефективно розпізнавати об'єкти на різних зображення, тобто у складі датасету є чіткі та нечіткі фото, зображення різного розширення, малі, середні, великі. У наборі зображень часто змінюється фон та ракурс.

Провівши аналіз та порівняння датасетів, можна зробити висновок, що даний набір цілком задовольняє вимоги, які поставлені до дипломного проекту та підходить для побудови системи розпізнавання типу транспортних засобів.

2.5 Вибір моделі для машинного навчання

Після того, як було обрано фреймворк за допомогою якого буде відбуватись машинне навчання, потрібно обрати модель. На протязі декількох десятків років вчені створюють різні моделі для розпізнавання об'єктів на

зображенні. Так, як ми обрали фреймворком TensorFlow, то спільнота на даний момент створила десяток моделей, які можна використовувати для розпізнавання об'єктів. TensorFlow пропонує колекцію моделей[11] для розпізнавання, попередньо нетренованих на наборах даних COCO, Open Images, Kitti. Ці моделі досить корисні, якщо використовувати їх для вхідних даних, які є вже в даних датасетах. Також, вони корисні для ініціалізації нових моделей під час навчання новими наборами даних. Отже, після вибору фреймворку машинного навчання та датасета транспортних засобів, потрібно вибрати модель, яка буде за допомогою бібліотеки навчатись на вибраних даних.

На даний момент, моя система працює на мобільному графічному процесорі, тому найкращим вибором для даної системи є “ssd_mobilenet” моделі, які показують хорошу швидкість для мобільних ГП.

Таблиця 2.2 - COCO треновані моделі для мобільних ГП

Назва моделі	Швидкість(ms)	COCO mAP[^1]
ssd_mobilenet_v1_coco	30	21
ssd_mobilenet_v2_coco	31	22
ssd_mobilenet_v2_quantized_coco	29	22

Розглянувши таблицю 2.2, можна побачити ім'я моделі, яке буде відповідати кофігураційному файлу, який використовується для підготовки моделі. Швидкість моделі, відображається час в мілісекундах, який витрачається на обробку зображення з розширенням 600x600. Останнім параметром відображається продуктивність детектора для підмножини набору даних COCO, знайдене за допомогою вимірювання mAP (тут чим більше, тим краще), параметр заокруглений до найближчого цілого числа.

mAP - усереднене середня точність для нобуру запитів, визначається за формулою:

$$mAP = \frac{\sum_{q=1}^{Q} AveP(q)}{Q}, \quad (1)$$

де, Q - кількість запитів, AveP (q) - середня точність.

Отже, порівнявши і проаналізувавши дані таблиці 2.2, я прийшов до висновку, що найкращим вибором для даної системи є модель “ssd_mobilenet_v2_quantized_coco”, яка має найкращі показники швидкості та продуктивності серед представлених моделей, які підходять для розробки системи та задовольняють функціональні вимоги.

2.6 Вибір інструментів для створення графічного інтерфейсу

Будь-яке застосування, яке було створено для масового користувача має мати в своєму складі графічний інтерфейс користувача(GUI). Графічний інтерфейс дасть можливість користувачу системи здійснювати маніпуляції над програмним продуктом, безпосередньо за допомогою використання графічних елементів. Це набагато покращує користувацький досвід, адже, досить часто, користувачу інтуїтивно зрозуміло як комунікувати з програмою. Даний дипломний проєкт розроблений для розпізнавання типу транспортних засобів та дозволяє проводити аналіз фото та відео файлів, а також камер, які підключені до системи. Важливо, щоб користувач даного застосунку мав можливість вибрати функцію, яка йому в даний момент потрібна.

Таким чином, було прийнято рішення створення графічного інтерфейсу користувача, який надасть можливість клієнту вибрати який файл або камеру потрібно аналізувати та покаже результат розпізнавання та класифікації.

Для мови Python є безліч інструментів для створення графічного інтерфейсу. Серед них можна виділити tkinter [12], pyQt [13]. Дані інструменти розробки широко використовуються та надають можливість створення мінімалістичного інтерфейсу.

2.6.1 pyQt

Даний інструмент - оболонка на мові програмування Python для бібліотеки Qt. Бібліотека реалізована в Python-модулях, та охоплює біля 1,000 класів. Класи використовують механізм сигналів для зв'язку між об'єктами, що спрощує створення програмних компонентів багаторазового використання. PyQt розробляється англійською компанією Riverbank Computing. Підтримуються операційні системи Microsoft Windows, Linux, OS X, iOS та Android. Даний інструмент є вільним програмним забезпеченням і розповсюджується для некомерційного використання. Разом із бібліотекою можна скачати Qt Designer - інтегроване середовище розробки, призначене для створення крос-платформних застосунків з використанням бібліотеки Qt. Дане середовище не потребує знань у програмуванні, за допомоною нього можна створити графічний інтерфейс та перекласти його у код написаний на Python. PyQt та Qt Designer доступний для Python2 і Python3. Ці інструменти об'єднують Qt C++ міжплатформний фреймворк і мову інтерпретації Python.

2.6.2 Tkinter

Tkinter є найбільш часто використовуваним інструментарієм програмування Python GUI. Tkinter — багатоплатформна графічна бібліотека інтерфейсів на основі засобів Tk, поширюється з відкритим програмним кодом. Tkinter - це набір обгорток, які реалізують віджети Tk як класи Python. Крім того, внутрішній модуль _tkinter забезпечує безпечний механізму, який дозволяє Python та Tcl взаємодіяти. Головні переваги бібліотеки - швидкість і простота, а ще те, що вона постачається в комплекті з Python. Спільнота активно створює та доповнює документацію бібліотеки, довідники, навчальні посібники, книги. Не зважаючи на те, що це досить давня бібліотека, вона постійно розвивається.

Проаналізувавши та порівнявши дані інструменти розробки, я прийшов до висновку, що обидві бібліотеки задовольняють вимоги до проекту та можуть бути використані у розробці. Але, зважаючи на те, що Tkinter постачається у комплекті з Python, не потребує додаткових інсталяцій, я вибрав його у якості бібліотеки для розробки графічного інтерфейсу користувача, що дозволить швидко розробити рішення.

3 СТРУКТУРНО-АЛГОРИТМІЧНА ОРГАНІЗАЦІЯ

3.1 Структура програмних засобів

У даному підрозділі, розглянемо детальний опис структури програмних засобів, а саме загальну структуру системи.

Програмне забезпечення представляє собою десктоп-застосування з архітектурою, яка зображена на рисунку 3.1.

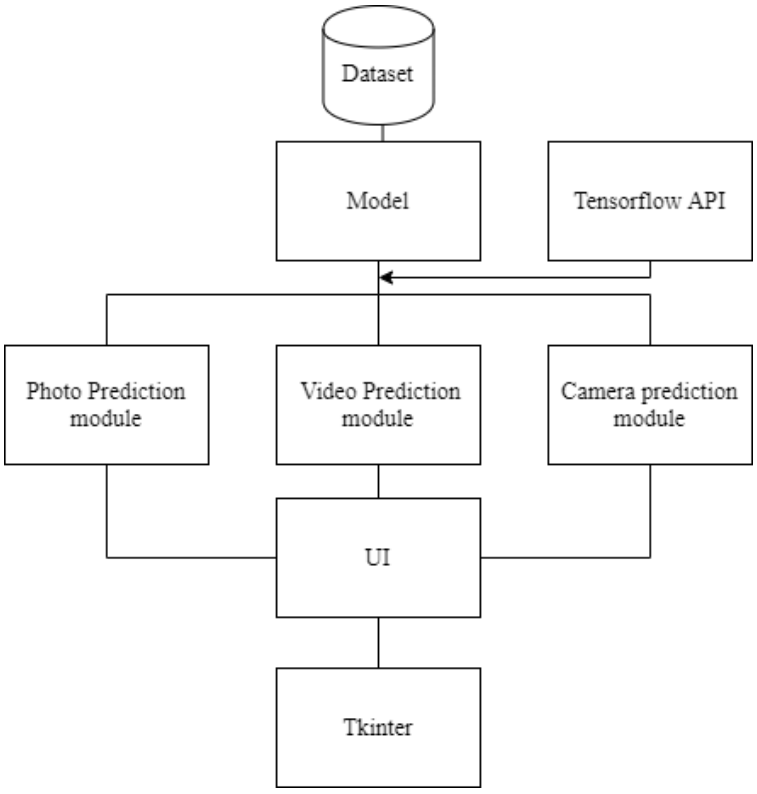

Рисунок 3.1 - Структура модулів системи

Система надає можливість користувачу взаємодіяти із трьома модулями розпізнавання транспортних засобів, за допомогою інтерфейса користувача. GUI функціонує за допомогою модуля Tkinter, стандартному інструменту створення інтерфейсу для Python.

За допомогою датасету та Tensorflow API, було створено глибоку нейронну мережу, яка використовується у модулях Prediction модулях для виявлення та класифікації типу транспортних засобів.

Model – модуль моделі глибокого навчання, яка була вибрана для побудови системи. Вона створена за допомогою бібліотеки Tensorflow, та спеціальних натренованих моделей, які можна знайти в зоопарку моделей Tensorflow. Дана модель тренувалась на наборі даних, представлених статичними зображеннями та розділеними на тренувальну та навчальну вибірки.

Dataset – набір даних, які будуть використовуватись для навчання та тестування вибраної моделі глибинного навчання. Набір даних містить 16 185 зображень із 196 класів автомобілів. Дані розділені на 8144 навчальних зображень та 8,041 тестових зображень, де кожен клас був розділений приблизно на 50-50. Класи представлені на рівні марки, моделі, року випуску.

Tensorlow API – бібліотека Tensorlow, створена командою Google, яка має відкритий програмний код, та включає в себе машинне навчання та глибоке машинне навчання. Бібліотека представлена набором функцій, підпрограм, протоколів взаємодії та засобів, таких як, класи, методи, процедури та алгоритми навчання, розпізнавання, тестування. Використовується дане API для мови програмування, яка була обрана раніше – Python. Дана бібліотке дає можливість навчати вибрану модель, а також дає можливість модулям розпізнавання виконувати їхнє призначення, адже вони використовують функції та алгоритми саме цієї бібліотеки.

Photo prediction module – модуль розпізнавання транспортних засобів, який визначає параметри автотранспорту. Модуль використовує вхідні дані представлені у вигляді статичного зображення, фото файлу, з розширенням файлу “jpg” або “png”. За допомогою бібліотеки Tensorflow взаємодіє з моделлю глибокого навчання, яка аналізує вхідні дані. Результатом роботи даного модуля є передбачення марки, моделі та року випуску транспортного

засобу. Результат передається у модуль графічного інтерфейса, для того, щоб зобразити користувачу інформацію користувачу. У випадку вдалого розпізнавання, коли на фото файлі, модель впевнена більш ніж на 50 відсотків у правильності прогнозу, виводиться також зображення та граничні рамки розпізнаного об’єкту.

Video prediction module – модуль розпізнавання транспортних засобів, який визначає параметри автотранспорту. Модуль використовує вхідні дані представлені у вигляді набору зображень, відео файлу, з розширенням файлу “mp4”, “flv” або “avi”. За допомогою бібліотеки Tensorflow взаємодіє з моделлю глибокого навчання, яка аналізує вхідні дані. Результатом роботи даного модуля є передбачення марки, моделі та року випуску транспортного засобу. У ході роботи модуля, створюється вікно з потоком даних із вхідного файлу, який буде оброблятись. У випадку вдалого розпізнавання, коли на відео файлі, модель впевнена більш ніж на 50 відсотків у правильності прогнозу, результат обробки буде виводиться безпосередньо у потік даних.

Camera prediction module – модуль розпізнавання транспортних засобів, який визначає параметри автотранспорту. Модуль використовує вхідні дані представлені у вигляді набору зображень, які надходять із пристрою введення, а саме веб-камери або іншої камери підключеної до пристрою з даною системою. За допомогою бібліотеки Tensorflow взаємодіє з моделлю глибокого навчання, яка аналізує вхідні дані. Результатом роботи даного модуля є передбачення марки, моделі та року випуску транспортного засобу. У ході роботи модуля, створюється вікно з потоком даних із вхідного набору даних, який буде оброблятись. У випадку вдалого розпізнавання, коли модель впевнена більш ніж на 50 відсотків у правильності прогнозу, результат обробки буде виводиться безпосередньо у потік даних.

UI – модуль користувацького інтерфейсу, який взаємодіє з модулями розпізнавання. Дає можливість користувачу взаємодіяти з модулями розпізнавання транспортних засобів. Інформує користувача про можливість

вибору файлу або функції для аналізу, також повідомляє про процес обробки даних та виводить результат роботи модулів розпізнавання.

Tkinter – модуль бібліотеки, за допомогою якого створюються всі елементи графічного інтерфейсу, які використовує UI модуль. За допомогою бібліотеки, створюються кнопки для взаємодії, інформаційні поля, вікно меню, прокрутки для великих фото файлів. Усі ці функції використовуються і виводяться у модулі графічного інтерфейсу користувача.

Таким чином, у сукупності, дані модулі складають повноцінну систему розпізнавання транспортних засобів, яка використовує у якості вхідних даних статичні зображення або фото файли, відео файли або потік даних утворених пристроєм вводу.

3.2 Алгоритм навчання моделі розпізнавання та класифікації

Перш, ніж створювати будь-яку систему розпізнавання об'єктів, потрібно визначити з яким набором даних вона буде працювати. Навчання мережі відбувається на наборі даних, підготовлених спеціалістами. Вкрай важливо, на яких даних буде навчатись модель, адже від цього залежить правильність та ефективність розпізнавання. Датасет є основою класифікатора, за яким проводиться виявлення об'єктів.

У ході розробки, було підготовлено алгоритм за яким можна тренувати модель, який зображений на рисунку 3.2.

Рисунок 3.2 - Алгоритм тренування моделі

Для підготовки надійного класифікатора потрібно створити велику кількість фото, які значною мірою відрізняються один від одного. Чим більший такий набор даних, тим краща точність класифікатора. Тому, потрібно створити значну базу зображень, які будуть відрізнятись величиною, роздільною здатністю, чіткістю, кожен об'єкт має мати різний фон, освітлення, добре, якщо образ зображений з різних ракурсів. Також, важливо, щоб на деяких фото, крім об'єкта, був зображений випадковий об'єкт, який не має відношення до системи,

яка розробляється. Потім ці фото розділяються на фото, які будуть використовуватись для тренування та фото, які будуть використовуватись для тестування.

Створення міток для зображень. Для того, щоб процес навчання нейронної мережі відбувався коректно, потрібно визначити границі в яких знаходиться об'єкт. Потім, ці границі будуть використовуватись моделями для навчання. Часто, для визначення міток зображень використовують різні інструменти, програмні забезпечення. Один з таких інструментів - LabelImg [14]. Під час використання таких програмних засобів, можна визначати прямокутні поля (границі), де присутній об'єкт та присвоїти цьому образу відповідну назву, яку потім буде використовувати мережа. Після кожного визначення, інформація зберігається у форматі xml, який буде містити координати границь та назву об'єкта.

Генерування TFRecords для навчання. Записи TFRecords, використовують фреймворком TensorFlow, як вхідні дані для навчання детектора об'єктів. Генерація записів відбувається у два етапи:

- Конвертація кожних *.xml файлів в єдині *.csv файли для тренування та тестування.
- Конвертація кожних csv файлів в *.record файли для кожного набору.

Для цього можна скористатись двома скриптами, які скачати з гіт репозиторія [15]. Перш ніж конвертувати csv файли, саме в generate_tfrecords.py, потрібно замінити назви класів на власні, які будуть використовуватись у роботі. Таким чином, буде створено два файла записи, які будуть використані для тренування нейронної мережі.

Створення карти міток. Використовуючи текстовий редактор, створюється новий файл, з розширенням ".pbtxt". Цей файл буде використовуватись як карта у навчанні моделі. Карта міток повідомляє тренеру, що таке кожен об'єкт, відображає відношення назви класу до цілого номеру. Вміст у файлі має виглядати таким чином, як на зображенні 3.3.

```
item {
  id: 1
  name: 'AM General Hummer SUV 2000'
}

item {
  id: 2
  name: 'Acura RL Sedan 2012'
}

item {
  id: 3
  name: 'Acura TL Sedan 2012'
}

item {
  id: 4
  name: 'Acura TL Type-S 2008'
}
```

Рисунок 3.3 - Вміст файлу карти міток

Ідентифікаційний номер кожного елемента повинен відповідати ідентифікатору, визначеному у файлі, TFRecords.

Налаштування навчальної моделі. Як було вирішено у розділі 2, у даному проектів буде використовуватись “ssd_mobilenet_v2_quantized_coco” модель. У папці з цією моделлю, можна знайти файл “ssd_mobilenet_v2_quantized_coco.config”, який можна відкрити за допомогою текстового редактора. У даному файлі потрібно змінити кількість класів на ту кількість, яка буде використовуватись у проекті, а також шляхи до TFRecords файлів та до карти міток. Даний файл відповідає за процес навчання моделі та взаємодії з файлами системи.

Тренування моделі. Навчальний процес детектора можна почати, використовуючи сценарій Tensorflow Object Detection. Дане API, включає в себе функції, які потрібні на різних етапах тренування системи, контроль моделі, розпізнавання об'єктів. Для запуску тренування використовують команду, яка зображена на рисунку 3.4.

```
python model_main.py \
       --pipeline_config_path=models/ssd_mobilenet_v2_coco_2018_03_29/pipeline.config \
                                          --model_dir=output --num_train_steps=100000 --num_eval_steps=100
```

Рисунок 3.4 – Команада запуску навчання

model_main.py - це файл Tensorflow API, за допомогою якого буде відбуватись навчання, pipeline_config_path - шлях до налаштування навчання моделі, model_dir - шлях до моделі, яка буде навчатись, num_train_steps - кількість кроків навчання, num_eval_steps - кількість кроків еволюції моделі.

Після цього, розпочнеться процес тренування моделі. Це займе деякий час, який залежить від багатьох факторів, таких як:

- Обчислювальна потужність апаратури, яка використовується для навчання. Очевидно, чим потужніший персональний комп'ютер, тим швидше буде відбуватись навчальний процес.
- Використовується для навчання ГП чи ЦП. Яким би потужним не був центральний процесор, графічний процесор здатний в декілька разів пришвидшити процес навчання.
- Наскільки великий набір даних. Чим більша кількість зображень у наборі, тим більше потрібно часу, щоб модель досягла задовільних результатів виявлення, але через це буде краща ефективність.
- Складність виявлення. Чим складніший об'єкт, який може мати різну форму, розмір, колір, має дуже динамічний фон, тим складніше стане навчання.

Процес навчання можна контролювати за допомогою інструменту, який надає бібліотека Tensorflow. Tensorboard - інструмент, який дозволяє постійно контролювати та візуалізувати ряд показників тренування й оцінювання, під час навчання нейронної мережі. Запустивши даний інструмент, ініціалізується сервер, на якому кожні 5 хвилин оновлюється інформація про проказники навчання.

Як видно з Рисунка 3.5, за допомогою інструменту, можна контролювати значення “TotalLoss”, що показує загальну оцінку помилки системи, також цей параметр ще називають “загальним збитком”.

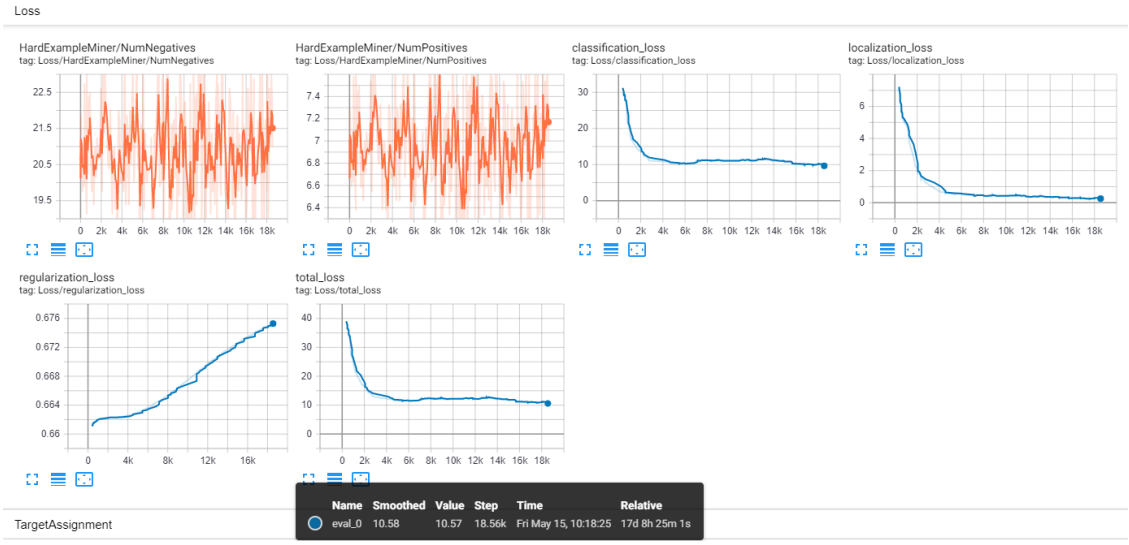

Рисунок 3.5 - Контроль навчання за допомогою Tensorboard

Для кожного датасету, моделі, оптимальне значення загальної помилки різне. Зазвичай, для отримання “справедливих” результатів, потрібно, щоб дане значення було в межах від 1 до 2. Очевидно, що чим нижче значення приймає TotalLoss, тим менша помилка, як наслідок, тим краще працює визначення об'єктів. Але слід бути обережними з цим показником, модель може перевиконати навчання і в кінцевому підсумку, коректно працювати тільки з даними, які визначені в наборі. Тому, потрібно часто слідкувати за показниками навчання, які коректно та зручно відображаються за допомогою Tensorboard.

Експорт обчислювального графу. Якщо параметр TotalLoss залишався в межах норми та довгий час не змінювався, то можна стверджувати, що навчання моделі пройшло успішно. Таким чином, потрібно сформувати обчислювальний граф, який можна використовувати для запуску моделі. Конвертація результатів навчання моделі в обчислювальний граф відбувається за допомогою Tensorflow

API. Перетворення відбувається за допомогою команди, яка зображена на рисунку 3.6.

```
python export_inference_graph.py
    --input_type=image_tensor --pipeline_config_path=./models/ssd_mobilenet_v2_coco_2018_03_29/pipeline.config
    --trained_checkpoint_prefix=output/model.ckpt-XXXX --output_directory=./inference_graph/
```

Рисунок 3.6 – Команда еспорту графа

export_inference_graph.py - це файл Tensorflow API, за допомогою якого буде відбуватись експорт графа, input_type - тип вузла введення, pipeline_config_path - шлях до налаштування моделі, trained_checkpoint_prefix - шлях до результату тренування, “XXXX” потрібно замінити на найбільше число контрольної точки, яка створена у папці навчання моделі, output_directory - шлях, де буде створено обчислювальний граф.

За допомогою цих кроків, буде створено обчислювальний граф, який описує модель та дозволяє взаємодіяти програмі з моделлю розпізнавання об'єктів.

3.3 Алгоритм виявлення та класифікації об'єктів

У 21 столітті, сфера виявлення об'єктів та класифікації, почала стрімко розвиватись. Із створенням моделей комп'ютерного зору, заснованих на глибокому машинному навчанні, програми у цій сфері стали набагато ефективнішими. Крім значного покращення продуктивності, ці методи почали використовувати датасети з меншою кількістю зображень, але з більш якісними тестовими виборками, що зменшило потреби у великих наборах даних.

З урахуванням того, що зорова кора головного мозку становить найбільш потужну і гнучку зорову систему з існуючих на даний момент, що допомагають виявленню об'єктів, поява моделей, які емулюють її поведінку, є зрозумілим кроком. Однією з найуспішніших моделей в області розпізнавання зображень є згорткова нейронна мережа [16]. Згорткові мережі являють собою варіацію архітектури багатошарового перцептрону, і включають в себе згорткові шари, шари підвибірки (субдискретизація) і повнозв'язні шари.

Архітектура згорткових мереж використовує переваги двовимірної структури вхідних даних - зображень за допомогою метода локальної зв'язності, обмежуючи кількість зв’язків між нейронами прихованого згорткового шару і вхідними даними. Конкретніше, кожен нейрон згорткового шару зв'язаний тільки з обмеженою локальною ділянкою зображення.

У роботі використовується SSD модель, що показує хороші показники швидкості та точності в завданнях на виявлення об'єктів. Вона розмежовує вихідний простір обмежувальних границь у набір полів за замовчуванням з різним співвідношенням сторін та масштабами за місцем розташування карти функції. На час прогнозування мережа генерує бали за наявність кожної категорії об'єктів у кожному полі за замовчуванням та виробляє коригування поля, щоб краще відповідати формі об'єкта. Крім того, мережа поєднує прогнози з декількох функціональних карт з різною роздільною здатністю для природного огинання об'єктів різного розміру.

3.3.1 Архітектура SSD моделі

Архітектура SSD – представляє собою нейронну мережу із шарами згортки, яка вчиться прогнозувати обмежувальні границь на зображенні та класифікувати їх за один прохід. До цього, моделі мали здійснювати багато

проходів, щоб зробити прогнозування. Дана архітектура покращила швидкість розпізнавань образів та дозволила працювати з відеофрагментами, які мають приблизно 60 кадрів в секунду. Швидкість цієї мережі дозволяє обробляти 58 кадрів за секунду, що є хорошим показником. Розглянемо структурну архітектуру моделі.

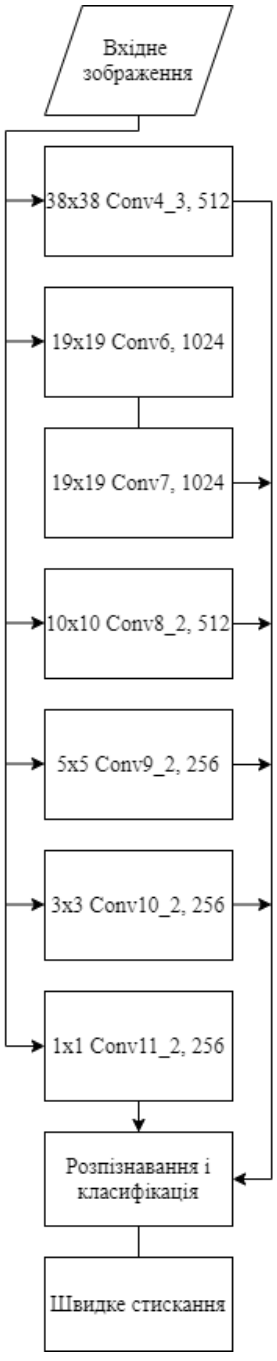

Рисунок 3.7 - Архітектура моделі нейронної мережі

Модель, зображена на Рисунку 3.7, складається з багатьох частин.

Вхідне зображення, яке подається на всі згорткові шари (предикатори). Зображення, як правило, береться із набору даних та приводиться до розширення 300x300 або 500x500.

Фільтри згортки, які обробляють вхідне зображення. Детектори Conv*, згорткові шари, слугують для розбиття зображення на деякий набір комірок, які будуть аналізуватись. Ці шари мають свої просторові розмірності. Наприклад, Conv11_2, має просторову розмірність 1x1, що добре підходить для розпізнавання об’єктів з великим масштабом. Для даного згорткового шару робиться 4 предбачення об’єкту для кожної комірки. Загалом, у трьох детекторах, Conv7, Conv8_2, Conv9_2 робиться по 6 прогнозів на один набір комірок, а всі інші – по 4. У результаті, SSD робить 8732 передбачення, використовуючи 6 шарів.

Результати фільтрів згортки подаються на спеціальний модуль детекції та класифікації, який буде аналізувати кожен прогноз для комірки та залишати тільки один.

Модуль швидкого стискання приймає набір передбачень від усіх згорткових шарів та залишає тільки один прогноз, який буде вважатись результатом.

3.3.2 Алгоритм розпізнавання образів за допомогою моделі SSD

Найперше, модель вибирає карту ознак та приймає зображення на вхід. Зображення буде використовуватись предикаторами (або згортальними шарами). Цей вхідний потік даних в майбутньому буде розбитий на області, які будуть покривати все зображення, кожна область буде аналізуватись та в залежності від карти ознак класифікуватись.

Наступний крок - застосування згортальних предикторів. Після отримання зображення, воно розбивається на області в залежності від згорткового шару. Коли просторова розмірність велика у згортковому шарі, то є вірогідні розпізнати малі об'єкти. Зі зменшенням просторової розмірності, виростає шанс розпізнати великі об'єкти. Оскільки, згорткова нейронна мережа зменшує просторову розмірність поступово, роздільна здатність карт ознак також зменшується. SSD використовує шари нижчої роздільної здатності для виявлення об'єктів більшого масштабу. Дані згортальні предикатори мають розмірність від 38 до 1, що дає можливість визначати як малі так і великі об'єкти.

Після подачі на предикатори, утворюється набір границь, величина яких буде залежати від того, який предикатор утворив цей набір. Дана сітка границь подається на модуль розпізнавання. Для кожної утвореної комірки (місцезнаходження), робиться деяка кількість предбачень, що залежить від конволюційного шару та не залежить від глибини карт зображень. Кожне таке передбачення складається з граничного поля, корекції розташування (прогнозу локалізації) та конфіденсу (бали) класифікації.

Визначення граничного поля. Як і у глибокому навчанні, ми можемо почати з випадкових прогнозів і використовувати градієнтний спуск для оптимізації моделі. Однак, під час початкового тренування, може вестись конкуренція між класами, щоб визначити, які саме форми потрібно оптимізувати під передбачення. Емпіричні результати свідчать про те, що початкове навчання може бути не стабільним. Якщо прогнози охоплюють більше фігур, модель може виявити більше типів об'єктів. Цей вид тренування набагато легший та стабільніший.

Вибір правильного прогнозу локалізації. Прогнози SD класифікуються як позитивні або негативні. SSD використовує лише позитивні збіги при розрахунку локалізації. Якщо відповідне граничне поле за замовчуванням має

IoU, що перевищує 0,5 - збіг є позитивним. Така стратегія заохочує кожен прогноз бути ближчим до реального граничного поля фігури.

Обчислення балів класифікації. Для кожного утвореного граничного поля утворюється вектор оцінок. Утворений вектор подається softmax функції, яка повертає бали для кожного класу. Клас з найбільшим балом присвоюється граничному полю.

Після кожного такого передбачення отримується набір таких даних, таких як граничне поле, яке складається із координати центральної точки границі та габарити цієї границі (висота та ширина), а також корекція розташування та оцінка ймовірності класу.

Таким чином, після модуля розпізнавання буде отримано велика кількість передбачень, що будуть мати набори даних. Ці передбачення фільтруються за допомогою модуля швидкого стискання. Чим більша оцінка та менша корекція розташування, тим більший шанс, що алгоритм вибере саме цю обмежувальну границю як результат.

Загальна структура алгоритму зображена на рисунку 3.8.

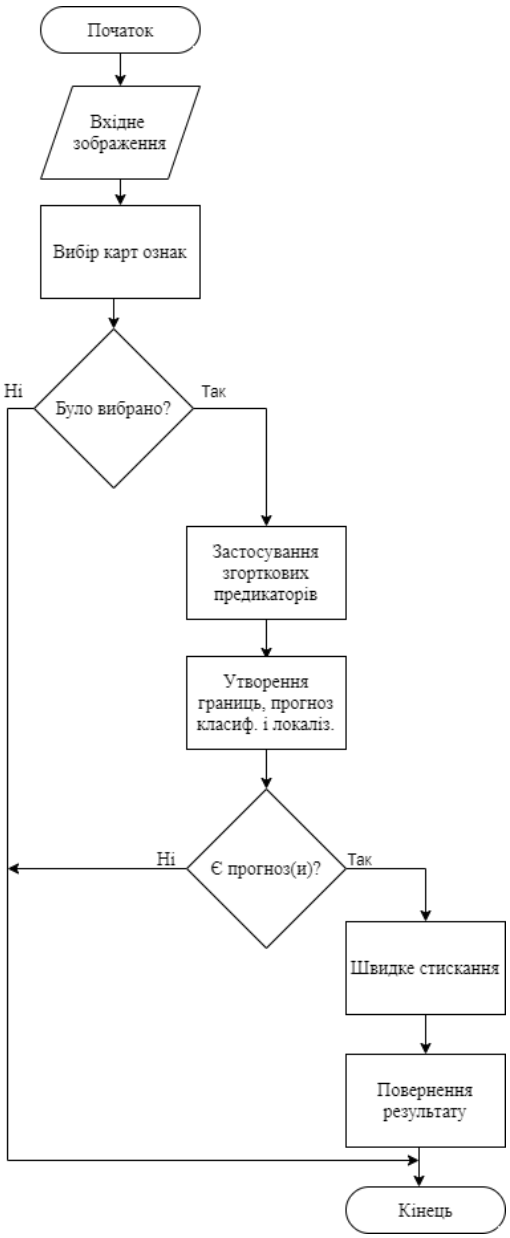

Рисунок 3.8 - Алгоритм розпізнавання об'єктів

Чим більша роздільна здатність зображення тим легше виявити дрібні об'єкти. Перший шар для виявлення об'єкта conv4_3 має просторовий розмір 38 × 38, досить велике зменшення від вхідного зображення. Отже, SSD зазвичай погано спрацьовує для невеликих об'єктів порівняно з іншими методами виявлення. Цю проблему можна вирішити, якщо використовувати зображення з більшою роздільною здатністю. Дана проблема відноситься до недоліків згорткових моделей. Також, таким архітектурам складно справлятися з

спотворенням, такими як розмивання або шумом. При цьому, згорткові мережі порівняно легко справляються з проблемами вискоточного розпізнавання, які викликають труднощі у людей. Наприклад, розпізнавання окремих моделей машин, що вимагає виділення специфічних ознак і що є ціллю даного проєкту.

3.4 Розробка програмної системи

Розробка програмної системи розпізнавання типу транспортного засобу представляє собою написання системи, яка складається із набору модулів, які взаємодіють один з одним.

У даному проєкті, було визначено набір інструментів, які потрібні для проектування програмної системи. Власне, у розділі 2, було проаналізовано мови програмування, бібліотеки, набори даних, які дозволять задовольнити вимоги до програмного забезпечення та дадуть можливість спроектувати систему.

Таким чином, для розробки програмної системи розпізнавання, було обрано мову програмування Python, бібліотеку глибинного машинного навчання Tensorflow, архітектуру моделі та визначено набір даних по яких модель буде навчатись розпізнавати об’єкти та їх класифікувати.

Загалом, архітектура програмної системи, яка описана в підрозділі 3.1, складається з набору модулів, 3 з яких – модулі для розпізнавання об’єктів та класифікації. Дані модулі займають ключову роль у системі. Неправильне написання модулів призведе до некоректної роботи усієї системи та хибних результатів розпізнавання та клисифікації.

Бібліотека Tensorflow, призначена полегшити написання програмних модулів, які виконують функції штучного інтелекту, за допомогою API в якому описані функції, алгоритми та методи, які дають можливість навчати модель та

взаємодіяти з нею. Для роботи з бібліотекою потрібно встановити її у середовище розробки та підключати її як модуль до програмних засобів, які будуть використовувати дане API. Встановлення відбувається за допомогою подачі команди пакетному менеджеру.

Першим кроком в розробці програмної системи є навчання моделі на вибраному наборі даних. Датасетом для системи обрано “Stanford cars dataset”, який доступний в мережі інтернет. Сам набір даних представлений набором зображень, розподіленим 50 на 50 у навчальну та тренувальну вибірки. У підрозділі 3.2, детально викладено розроблений алгоритм навчання моделі (рисунок 3.2) на задовільних наборах даних. Виходячи з того, що вибраний набір даних уже готовий до використання, тобто заздалегідь створені зображень та мітоки розташування, тому перші два пункти алгоритму потрібно пропустити. Але, в загальному, алгоритм навчання не змінний і у цьому проєкті використовувався саме він. Модель навчалась до загального кроку 20 тисяч, що зайняло великий проміжок часу (у підрозділі 3.2 описано від чого він залежить). Варто зазначити, що навчена модель не є ідеальною й у майбутньому навчання краще продовжити до загального кроку, який має складати більш ніж 80 тисяч, що покращить точність визначення розташування та класифікації.

Таким чином, було отримано три модуля, які будуть напряму брати участь у процесі розпізнавання. Модуль набору даних, модуль Tensorflow API, модель. Дані модулі будуть використовуватись програмними модулями для виконання основної функції системи – розпізнавання типу транспортного засобу.

Після створення та навчання моделі, можна можна почати розробку програмних модулів, які будуть використовувати модель для розпізнавання образів та об’єктів. Розробка даних модулів – це головна мета та ціль проєкту. При розробці даних модулів будуть використовуватись фреймворк Tensorflow та Tensorflow API, а також, бібліотеку numpy та в деяких випадках OpenCV. Буде розроблено три модулі (згідно архітектури системи, яка описана у

підрозділі 3.1) для розпізнавання типу транспортного засобу на фото, відоеофрагментах та у реальному часі.

Кожен програмний модуль має у своєму складі частини коду, які мають шаблонний тип. Ця частина коду використовується у кожному модулі та слугує початковою ініціалізацією всіх потрібних компонентів для реалізації розпізнавання. На рисунку 3.9, зображено початкову частину коду, яка характерна для усіх трьох програмних модулів.

```
import cv2
import numpy as np
import tensorflow as tf

from distutils.version import StrictVersion
from object_detection.utils import label_map_util
from object_detection.utils import visualization_utils as vis_util

NUM_CLASSES = 196

# Checking the tensorflow version
if StrictVersion(tf.__version__) < StrictVersion('1.9.0'):
    raise ImportError('Please upgrade your tensorflow installation to v1.9.* or later!')


def load_image_into_numpy_array(image):
    (im_width, im_height) = image.size
    return np.array(image.getdata()).reshape(
        (im_height, im_width, 3)).astype(np.uint8)


path_to_graph = "E:/NTUU/Py/Vehicle_detection/cars_inference_graph1/frozen_inference_graph.pb"
path_to_label = "E:/NTUU/Py/Vehicle_detection/stanford_cars_label_map.pbtxt"

detection_graph = tf.Graph()
with detection_graph.as_default():
    od_graph_def = tf.GraphDef()
    with tf.gfile.GFile(path_to_graph, 'rb') as fid:
        serialized_graph = fid.read()
        od_graph_def.ParseFromString(serialized_graph)
        tf.import_graph_def(od_graph_def, name='')

label_map = label_map_util.load_labelmap(path_to_label)
categories = label_map_util.convert_label_map_to_categories(label_map, max_num_classes=NUM_CLASSES,
                                                            use_display_name=True)
category_index = label_map_util.create_category_index(categories)
```

Рисунок 3.9 – Початкова частина коду програмних модулів

У даній частині коду, імпортуються всі необхідні бібліотеки та методи із API. Створюється спеціальна функція, яка допоможе завантажити зображення у numpy масив, що зручно для подання даних у матрицю, над якою у подальшому буде проводитись деякі перетворення. Також ініціалізуються глобальні змінні, задаються шляхи до карти ознак та експортованого графу, який утворюється після останнього кроку алгоритму навчання (рисунок 3.2) та

перевіряється версія встановленого фреймворку Tensorflow, адже якщо він застарілий, то деякі функції можуть не коректно працювати або їх може не бути взагалі.

Під час розробки програмного модуля для розпізнавання об'єктів на статичному зображенні, було прийнято рішення, розробити допоміжну функцію, яка буде виконувати функцію аналізу одного зображення, та передавати у основну функцію результат розпізнавання. Дана допоміжна функція звертається до фреймворка Tensorflow та отримує вхідні та вихідні тензори, проводить ініціалізацію сесії розпізнавання, ініціалізує необхідні змінні для роботи та трансформує координати маски поля в координати зображення, щоб відповідний кадр відповідав розміру зображення. Розпочинає процес розпізнаванн, звертаючись до Tensorflow API, що з самого початку виконує пошук графічного процесору, який може виконати аналіз зображення, після знаходження обробляє зображення та повертає результат у спеціальні змінні, які були ініціалізовані до цього допоміжною функцією, дані змінні присвоюються у основній функції.

При розробці даних модулів, враховувалось, що вони можуть підключатись та використовуватись у будь-якому додатку. Мова програмування Python, дозволяє це зробити, було прйнято рішення використати це. Тому, для комфортного виклику функцій програмної системи у інших модулях, була розроблена основна функція, яка буде викликати потрібні функції аналізу вхідного зображення і повертати результат туди, де була викликана.

Основна функція модуля для розпізнавання типу транспортного засобу, отримує шлях до зображення, відкриває його та передає для аналізу в допоміжну функцію, після того, як був отриманий результат, основна функція проводить візуалізацію отриманих даних, зберігає зображення з візуалізацією в дисковий простір та повертає рядок із назвою розпізнаного класу та впевненість мережі у правильності прогонзу (Бали. Як вони отримуються та алгоритм

розпізнавання описано у підрозділі 3.3). У майбутньому основна функція буде викликатись у графічному інтерфейсі, де буде виводитись інформація, яка повертає дана функція.

При розробці програмного модуля для розпізнавання типу транспортного засобу по відеофрагментам, було використано бібліотеку OpenCV, що використовується для систем спрямованих на технологію комп'ютерного зору у реальному часі. Дана бібліотека буде використовуватись для захоплення вхідного відео, для чого у спеціальну функцію фреймворку передається шлях до потрібного файлу. Дане відео розбивається на набір статичних зображень (кадри), які будуть аналізуватись, як це робилось для простого зображення. Основна функція викликає даний метод, ініціалізує усі потрібні змінні для аналізу, та у нескінченному циклі аналізує кадри та візуалізує результат безпосередньо у потік даних. У даному циклі, виводиться вже оброблений набір даних у нове вікно, яке створене за допомогою методу "imshow()", бібліотеки OpenCV. Процес продовжується допоки не буде оброблений останній кадр або користувач не натисне кнопку "q" для виходу.

Розробка програмного модуля для розпізнавання типу транспортного засобу у потоці даних, отриманих за допомогою пристрою введення, а саме – веб-камері, зіштовхується із задачею обробки наперед не визначеного зображення, що ускладнює проектування модуля. Для вирішення даної задачі, у основній функції, було використано бібліотеку OpenCV, а саме функцію для захоплення відео, але в якості шляху відео файлу, було передано пристрій введення та спеціальний флаг захоплення відеофрагментів. Дана функція розбиває вхідні дані пристрою введення на статичні зображення (кадри). У нескінченному циклі, отримується кадр, ініціалізуються змінні для розпізнавання та відбувається обробка кадру, як звичайного статичного зображення та візуалізується результат безпосередньо у потік даних. За допомогою методу бібліотеки комп'ютерного зору, результат візуалізації

відображається у новому вікні. Процес продовжується допоки користувач не натисне кнопку “q”.

Таким чином, було створено три програмні модулі, кожен з яких працює для розпізнавання типу транспортного засобу. Вони представляють собою окремі модулі, які можуть бути використані у будь-якому програмному продукті, як зовнішня бібліотека, яка надає доступ до основної функції, що приймає шлях до файлу (або потік даних у випадку модуля з розпізнаванням в реальному часі) та повертає результати розпізнавання. Для побудови модулів, було використано бібліотеки Tensorflow та деякі функції OpneCV, які допомогли вирішити проблеми, які з’являлись у ході розробки.

Також, у даному проєкті, було створено графічний інтерфейс користувача, як окремий модуль, який під’єднує до себе розроблені програмні модулі. Графічний інтерфейс дає можливість вибору між розпізнаванням зображення на фото чи відеофрагментах або обробка зображення із пристрою введення. GUI викликає основну функцію вибраного користувачем модуля та виводить результат розпізнавання у спеціальне інфо поле.

Після розробки усіх програмних модулів, можна впевнитись, що архітектура програмної системи, яка зображена на рисунку 3.1, було розроблена корректно, адже має в своєму складі усі модулі, які взаємодіють між собою. Розроблені модулі виконують потрібні функції та можуть бути підключені в якості зовнішніх бібліотек.

4 АНАЛІЗ ТА ТЕСТУВАННЯ РОЗРОБЛЕНОЇ ПРОГРАМНОЇ СИСТЕМИ

4.1 Опис інтерфейсу та тестування системи

У процесі розробки даного проекту, було створено систему розпізнавання транспортних засобів. Дана система містить деяку кількість модулів для розпізнавання, а саме модуль для розпізнавання на статичному зображенні, модуль для розпізнавання відеофрагментів, модуль розпізнавання потоку даних в реальному часі. Кожен модуль буде протестовано окремо. Результати тест-кейсів буде подано графічно та пояснено результат роботи модуля.

4.1.1 Опис інтерфейсу

Користувацький інтерфейс даної системи був створений за допомогою стандартної python бібліотеки - tkinter. Цей інструмент допоміг швидко та легко розробити мінімалістичний інтерфейс, який зображено на рисунку 4.1.

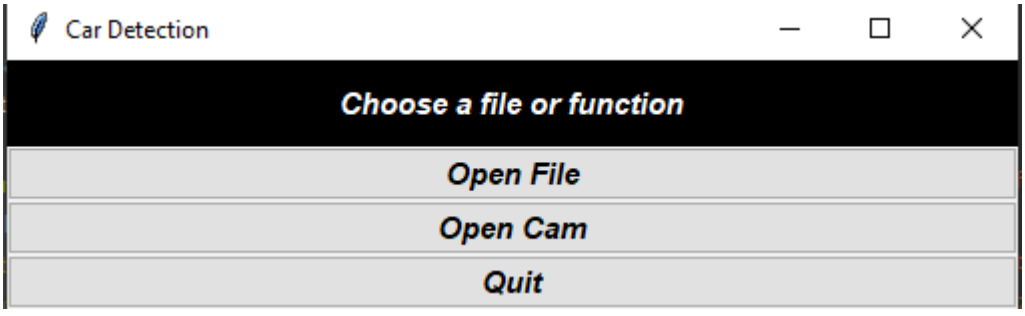

Рисунок 4.1 - Інтерфейс системи розпізнавання

При запуску розробленої системи, відкривається вікно, яке містить чотири графічні елементи.

Перший елемент (зверху) - поле міток. Вміст поля змінюється в залежності від стану програми. Слугує для інформування користувача про можливості взаємодії з програмою, про те, що виконується обробка зображення. Також, саме у цьому полі може виводитись результат розпізнавання.

Другий елемент - кнопка вибору файлу. Програмне забезпечення надає можливість вибору файлу, який потрібно проаналізувати та обробити. За допомогою цієї кнопки, можна вибрати фото файли, такі як “.jpg”, “.png” або відео файли, такі як “.avi”, “.mp4”, “.flv”. В залежності від файлу, який був вибраний користувачем, буде викликано програмний модуль, який буде обробляти даний файл та виконувати розпізнавання об'єктів.

Якщо користувач вибрав “.jpg” або “.png”, то результат виконання модуля, а як наслідок, розпізнавання об'єктів, відобразиться у полі міток, додасться ще один елемент над полем міток. Цей елемент - це оброблене фото, на якому система виділила розпізнані об'єкти.

У випадку, якщо користувач вибрав відео файл, то система створить нове вікно, у якому буде відтворюватись відео фрагмент та проходити процес розпізнавання об'єктів.

Третій елемент - кнопка вибору камери. Система надає можливість вибору пристроя введення графічної інформації для розпізнавання об'єктів в реальному часі. При виборі користувачем даного функціоналу, програмне забезпечення створить нове вікно, у якому буде обролятись потік відеоданих за допомогою модуля, який аналізує відеофрагменти в реальному часі та виводить результат розпізнавання прямо у потік даних.

Останній елемент - кнопка виходу з системи. Дана кнопка ініціює функцію, яка завершує програму.

4.1.2 Тестування модуля фото розпізнавання

Розроблений модуль розпізнавання об'єктів на фото зображеннях має різну поведінку на різних вхідних зображеннях. Опис результатів наведені далі.

Тест-кейс 1. Опис зображення: робота чітко видима, добре освітлена, немає шумів, розмір фото - середній, на фото передня частина транспортного засобу, назва якого “AM General Hummer”. На рисунку 4.2, зображено результат тест-кейсу 1.

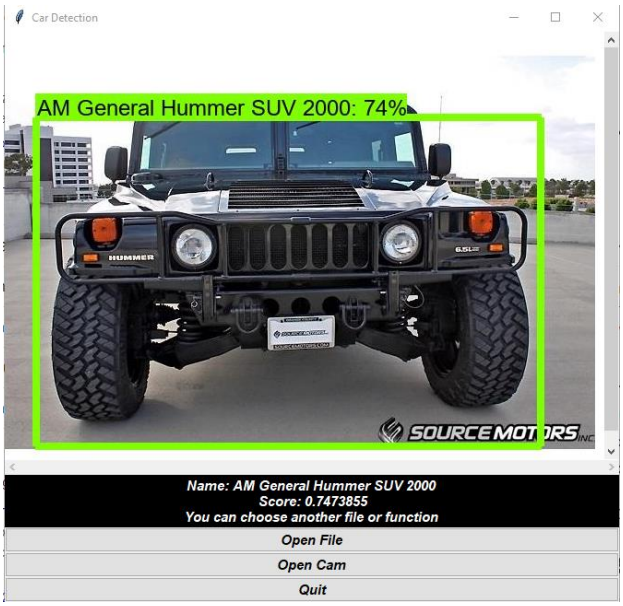

Рисунок 4.2 - Результат текст-кейсу 1

Можна побачити, що по результатам даного тест-кейсу, система коректно розпізнала даний транспортний засіб, вивела впевненість системи у прогнозі, марку, модель, тип кузова та рік випуску, також на фото позначено границі розпізнаного об’єкту.

4.1.3 Тестування модуля відео розпізнавання

Розроблений модуль розпізнавання об'єктів на відео фрагментах має різну поведінку, яка залежить від вмісту вхідного зображення. Опис результатів наведені далі.

Тест-кейс 2. Опис відеофайлу: На відео фрагменті всі об'єкти добре освітлені, немає шумів, розширення відеофайлу - 720p. У відео можна побачити транспортний засіб під назвою “Bugatti Veyron”. На рисунку 4.3, зображено результат тест-кейсу 2.

Рисунок 4.3 - Результат тест-кейсу 2

Можна побачити, що по результатам даного тест-кейсу, система коректно розпізнала даний транспортний засіб, вивела впевненість системи у прогнозі, марку, модель, тип кузова та рік випуску, також на відеофрагменті позначено границі розпізнаного об’єкту.

4.1.4 Тестування модуля розпізнавання об'єктів в реальному часі

Розроблений модуль розпізнавання об'єктів у потоку даних має різну поведінку, яка залежить від вмісту вхідного зображення. Опис результатів наведені далі.

Тест-кейс 3. Опис потоку даних: використовується пристрій введення, а саме - веб-камера ноутбука для створення потоку даних. Відеофрагменти вводяться у розширенні 720p. У потоці даних можна побачити транспортний засіб під назвою “AM General Hummer”. На рисунку 4.4, зображено результат тест-кейсу 3.

Рисунок 4.4 - Результат тест-кейсу 3

Можна зробити висновок, що по результатам даного тест-кейсу, система коректно розпізнала даний транспортний засіб, вивела впевненість системи у прогнозі, марку, модель, тип кузова та рік випуску, також безпосередньо у потоці даних позначено границі розпізнаного об’єкту.

4.2 Рекомендації щодо використання програмної системи

Під час використання декстоп додатку користувачу рекомендується дотримуватись деяких правил.

По-перше, завантажені фото або відео зображення, повинні бути добре освітлені, зняті на хорошу камеру, оскільки система розпізнавання об'єктів може видати непередбачувані результати. Бажано, щоб фото, відео, містили на фоні нейтральні кольори, які не зливаються з обєктом.

По-друге, рекомендується використовувати файли для розпізнавання, які мають велику роздільну здатність, для того, щоб модель краще могла виконувати згортковий алгоритм.

По-третє, рекомендується, щоб на одному фото або відео файлі, були не більше двох об'єктів для розпізнавання, тому що результити можуть бути непередбачуваними. Рекомендується завантажувати файли, які підтримується програмною системою, вони вказані в самій системі.

Виконувати даний програмний продукт рекомендується на комп'ютерній системі, яка в своїй архітектурі графічний процессор NVIDIA, тому що, якраз для даних ГП оптимізована робота програмної системи. Память графічного процесору має бути більше 2 Гб, оперативна память 8 Гб, що забезпечує швидку та коректну роботу системи. Операційна система рекомендується Windows або будь-які дистрибутиви Linux.

4.3 Аналіз розробленої системи та рекомендації щодо вдосконалення

У ході розроблення даного програмного продукту, було натреновано нейронну мережу, яка працює на Stanford Cars Dataset. Вона була натренована

за допомогою бібліотеки Tensorflow. Основою моделі, яка використовується в даному проєкті, стала модель “ssd_mobilenet_v2_quantized_coco”. Тренування відбувалось із застосуванням засобів візуалізації та моніторингу навчання. Дані засоби дають можливість візуалізувати параметри моделі, що дає змогу проаналізувати модель.

Таким чином, ми можемо проаналізувати модель за допомогою параметрів, які можна отримати за допомогою інструменту TensorBoard. Даний засіб моніторингу відображає mAP, IoU, total loss, classification loss, localization loss.

Тут і далі, досліджувані параметри будуть подаватись у відношенні до загального кроку навчання. На даному етапі навчання і розробки цей крок становить 22 110, часу на таке навчання було витрачено приблизно 23 дні.

У рамках даної роботи, було розглянуто параметр mAP. Даний параметр показує середнє значення середніх балів точності для кожного запиту. Визначити цей параметр можна згідно з формулою (1). На даному етапі тренування моделі, парметри mAP можна побачити на рисунку 4.5.

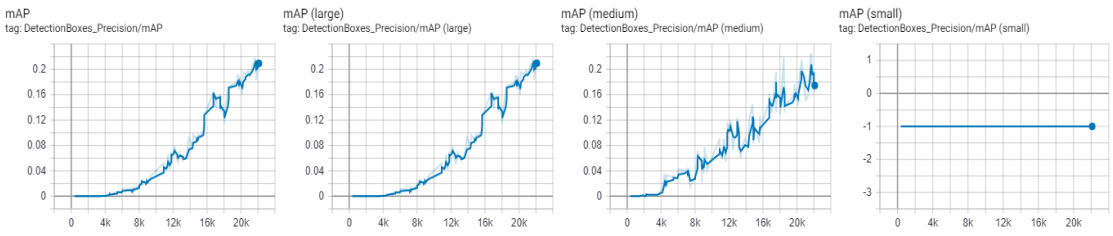

Рисунок 4.5 – Параметр mAP для розробленої моделі

Із рисунка 4.5 видно що середня точність даної моделі зростає і не знаходиться у постійному значенні, з цього можна зробити висновок, що дана модель може показувати більш точні результати, ніж у даний момент. Також, видно, що для малих за розміром зображень (площа яких менша 32 ^ 2 пікселя), середнє значення не змінювалось. Це наслідок того, що у наборі даних немає

дуже малих за розміром зображень, що й буде впливати на кінцевий результат моделі, яка не може коректно визначати малі об’єкти на вхідних даних.

Ще одним з параметрів, які були розглянуті у даній роботі, є відношення між областю, що перетинається й об'єднаною ділянкою для двох регіонів, коротко IoU. Даний параметр відіграє важливу роль у визначенні правильного прогнозу розташування об’єкту. Якщо дане відношення буде більше 0.5, це рахується хорошим результатом та коректним визначенням розташування та розміру об’єкту. На рисунку 4.6 зображено графіки даного параметру.

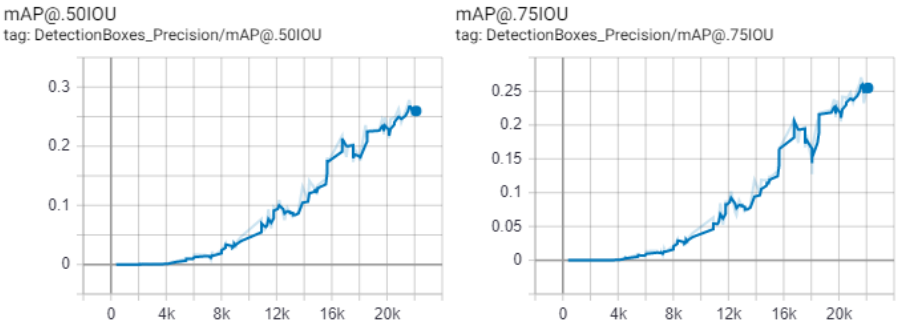

Рисунок 4.6 – Параметр IoU

Розглянувши рисунок 4.6, можна побачити, графіки для параметру IoU, які визначені у випадку, коли середня точність приблизно 50% або 75%. У обох випадках значення даного параметру становить не більше 0.3, що є досить хорошим значенням, але не рахується хорошим. Тому, можна вважати, що по наявними параметрами, модель ще не досить натренована і вона ще може показувати більш точні результати.

Також, варто звернути увагу на параметри втрати. Даний параметр показує наскільки відхиляються прогнозовані значення від фактичних значень у навчальних даних. При надходженні вхідного зображення, надходять також ваги, які використовуються системою для розпізнавання і класифікації. Змінюючи модель ваг, можна вплинути на кінцевий результат розпізнавання.

Тому модель старається змінювати вагу, щоб мінімізувати втрати, саме так вона тренується на наборі даних. На рисунку 4.7 зображено графіки даних параметрів.

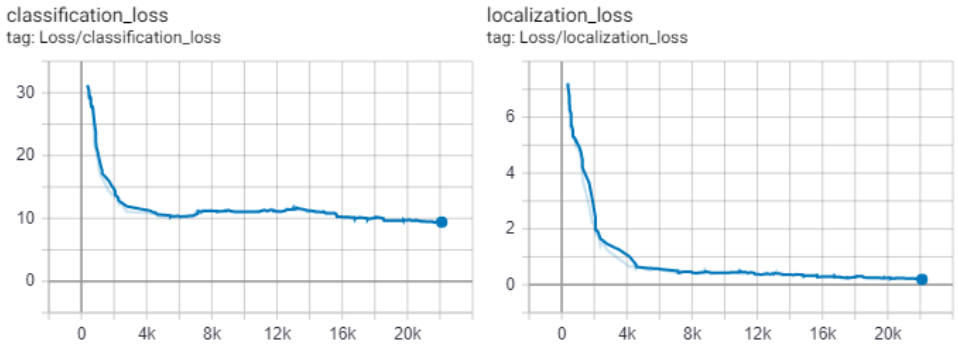

Рисунок 4.7 – Параметри втрати

Розглянувши рисунок 4.7, можна побачити, параметри classifiacation loss localization loss.

Перший графік зображує витрати класифікації. Цей параметр показує наскільки часто модель неправильно визначає клас об’єкту. Значення даного параметру не сильно змінюється по відношенні до загальних кроків, але все ж зменшується, хоч і повільно. Точного значення, яке вкаже, що дана модель коректно виконує класифікацію - немає. Враховуючи, що у наборі даних 196 класів і значення не досить швидко змінюється, можна вважати, що модель може коректно класифікувати об’єкти, але все ж при подальшому тренуванні може покращити свої результати.

Другий графік - витрати локалізації. Даний параметр показує які витрати генерує модель при визначення правильного розміру та розташування об’єкту на зображенні. На графіку чітко видно, що дана величина зменшується, що є хорошим показником. Але даний параметр ще не знаходить у стабільному значенні, тому можна сказати, що витрати на локалізації можуть бути менші.

Таким чином, можна можна прийти до висновку, що дана модель навчається коректно, що можна визначити з показників витрат, які

зменшуються та показників середньої точності, які збільшуються. Але, також видно, що тренування до кінця не завершено, тому що деякі параметри ще можуть бути покращені й вони не мають стабільного значення. Тому, для коректної роботи системи, рекомендується продовження тренування до стабільних показників моделі. Це вдосконалить систему та покращить результати роботи модулів розпізнавання об’єктів.

Також, можна надати ряд рекомендацій, щодо вдосконалення даної системи. В Україні транспортні засоби дещо відрізняються від тих, що представлені в датасеті, у результаті чого, існують випадки, коли алгоритм неправильно визначає марку і модель автотранспорту. Цього можна уникнути створенням власного набору даних та зробити його анотацію. Цей процес досить трудомісткий, займає великий проміжок часу, тому цей спосіб не був використаний. Один із способів збору набору даних є краудсорсинг, який можна було б використати у даній ситуації. Таким чином, можна навіть не відмовлятись вже від нетренованої моделі, а використати техніку transfer learning [17], що дозволяє тренувати мережу, яка вже навчалась на деяких даних іншого набору. Ця техніка використовується у проектах з компютерним зором, що дозволяє використати даний прийом і в дипломному проєкті.

Крім цього, дана система побудована на моделі машинного навчання, що є не дуже глибокою, яка використовує не багато шарів згортки, що напряму впливає на досягнення хороших результатів розпізнавання об'єктів. Цього можна уникнути використовуючи іншу, складнішу модель машинного навчання, але потрібно бути готовими до того, що процес навчання буде ресурсозатратним. Часто, для таких моделей, потрібно хороше апаратне забезпечення, яке дозволить тренування та використання моделі. Слід зазначити, що існують хмарні сервіси, як Google Cloud [18], AWS, Google Colab, Kaggle, Microsoft Azure, Oracle, які надають можливість використовувати відеокарти, які розташовані у них на сервері. У будь-якому випадку, оренда

графічних пристроїв у хмарних сервісів або купівля власного апаратного забезпечення, призведе до значних економічних витрат.

Також, важливо розуміти, що архітектура згорткових мереж, складно справлятися з спотворенням зображення. Часто це може бути причиною не коректної роботи системи. Щоб уникнути неправильних результатів, можна використовувати видалення шуму. Перед розпізнаванням, можна застосувати розмиття по Гауссу до зображення. Також, виконувати вирівнювання зображення та проводити тональний аудит, тобто отримання одного і того ж зображення у різних умовах освітлення.

ВИСНОВКИ

Метою даного дипломного проєкту було створення десктоп-системи розпізнавання типу транспортних засобів. Перед початком розробки було проведено аналіз предметної області, оцінено актуальність проблеми розпізнавання об’єктів, розглянуто головні задачі, які ставлять потенційні користувачі, проаналізував існуючі програмні рішення та було складено вимоги до програмного забезпечення. У результаті аналізу існуючих програмних рішень було встановлено, що жодна із запропонованих систем не відповідає вимогам та потребам користувачів, що підтверджує те, що розробка такої системи є необхідною і актуальною. Наведено змістовне обґрунтування вибору технологій розроблення, мов програмування, бібліотек машинного навчання, готових наборів даних, основи моделі та бібліотк для графічного інтерфейсу. Крім того, пояснювальна записка містить опис структури програмних засобів, алгоритму навчання нейронної мережі, алгоритм розпізнавання об'єкту цією мережею, а також приклад роботи системи, аналіз натренованої моделі, рекомендації щодо використання програмного забезпечення та подальшого розроблення.

Розроблена система здатна розпізнати тип кузову, марку, модель та рік випуску авто по фото, відео файлах або по за допомогою пристроїв введення, таких як веб-камери, з розпізнаванням в реальному часі. Програма показує границі розпізнаного об'єкту та виводить інформацію про нього. Для зручності було створено мінімалістичний інтерфейс користувача, який забезпечує доступ до основних функцій системи.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Статистика автопродаж у світі. *Статистика автопродаж* : URL: https://proautomoto.com/category/69-mirovie-prodazhi-avto (дата звернення: 04.03.2020).
2. Система виявлення автомобіля Axiomtek. *Axiomtek Vehicle Detection System* : URL: https://axiomtek.com/Default.aspx?MenuId=Solutions&FunctionId=SolutionView&ItemId=36&Title=Vehicle+Detection+System (дата звернення: 04.03.2020).
3. Програма виявлення об'єктів 2.0. *Object Detection 2.0* : URL: https://www.object-detection.com/ (дата звернення: 04.03.2020).
4. Програма розпізнавання автотранспорту. *TURNING US STREETS INTO CAR SHOWROOMS WITH AUTO RECOGNITION* : URL: https://www.blippar.com/blog/2017/05/25/introducing-automotive-recognition-us-streets-become-car-showroom (дата звернення: 04.03.2020).
5. Додаток розпізнавання автомобіля для покупців автомобілів. *Vehicle Recognition App Assists Car Buyers* : URL: http://www.multivu.com/players/English/7292856-state-farm-carcapture-vehicle-recognition-app-assists-car-buyers/ (дата звернення: 06.03.2020).
6. Глибинне навчання. Deep Learning : URL: https://www.investopedia.com/terms/d/deep-learning.asp (дата звернення: 06.03.2020).
7. Як ядра Tensor прискорюють моделі. *How Tensor Cores Accelerate Your Mixed Precision Models* : URL:

https://developer.nvidia.com/tensor-cores (дата звернення: 06.03.2020).

8. Паралельні обчислення за допомогою CUDA. *Параллельные вычисления CUDA* : URL: https://www.nvidia.com.ua/object/cuda-parallel-computing-ru.html (дата звернення: 15.03.2020).
9. MXNet гнучка та ефективна бібліотека для глибокого навчання. *Apache MXNet a flexible and efficient library for deep learning* : URL: https://mxnet.apache.org/ (дата звернення: 16.03.2020).
10. Tensorflow платформа машинного навчання з відкритим кодом. *Tensorflow end-to-end open source machine learning platform* : URL: https://www.tensorflow.org/ (дата звернення: 16.03.2020).
11. Tensorflow зоопарк попередньо навчених моделей. *Tensorflow detection model zoo* : URL: https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md (дата звернення: 18.03.2020).
12. Tkinter - інтерфейс Python для Tcl / Tk. *tkinter — Python interface to Tcl/Tk* : URL: https://docs.python.org/3/library/tkinter.html#module-tkinter (дата звернення: 07.03.2020).
13. Документація бібліотеки Qt для Python. *Qt for Python* : URL: https://doc.qt.io/qtforpython/ (дата звернення: 07.03.2020).
14. Інструмент для анотування графічного зображення LabelImg. *Graphical image annotation tool LabelImg* : URL: https://tzutalin.github.io/labelImg/ (дата звернення: 25.03.2020).
15. Репозиторія з скріптами для конвертації файлів. *Raccoon Detector Dataset* : URL: https://github.com/datitran/raccoon_dataset (дата звернення: 25.03.2020).
16. Згортальні нейронні мережі. *Convolutional Neural Networks* : URL: https://towardsdatascience.com/a-comprehensive-guide-to-

convolutional-neural-networks-the-eli5-way-3bd2b1164a53 (дата звернення: 26.03.2020).

17. Transfer Learning / Qiang Yang, Yu Zhang, Wenyuan Dai, Sinno Jialin Pan. Publisher: “Cambridge University Press”, 2020.
18. Вирішення задач за допомогою Google Cloud. *Solve more with Google Cloud* : URL: https://cloud.google.com/ (дата звернення: 18.05.2019).