ДВНЗ «ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ»
Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики

«До захисту допущено»
В.о. завідувача кафедри
_____Наталія МАСЛОВА
(підпис) (ім'я та прізвище)
«_____»_____2022
р.

Випускна кваліфікаційна робота
магістра
(освітній ступінь)

на тему «Дослідження та розробка методів та алгоритмів машинного навчання для автопілоту машини»/ «Research and development of machine learning methods and algorithms for machine autopilot»

Виконала: студента 2м курсу, групи КНм-21
(шифр групи)
спеціальності (напрямку підготовки) 122 Комп'ютені науки
(шифр і назва напряму підготовки, спеціальності)

Дмитро СЕРДЮКОВ
(прізвище та ініціали) (підпис)

Керівник доц. кафедри ПМІ, к.т.н., проф. Ірина НАЗАРОВА _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

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

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

Нормоконтроль:

к.т.н., доц. Ірина НАЗАРОВА

_____
(підпис)

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.

Студент _____
(підпис)

Луцьк – 2022р.

ДВНЗ «Донецький національний технічний університет»
Факультет комп'ютерно-інформаційних технологій та автоматизації
Кафедра прикладної математики та інформатики
Освітній ступінь магістр
Напрям підготовки (спеціальність) 122 Комп'ютерні науки

ЗАТВЕРДЖУЮ:
В.о. завідувача кафедри ПМІ
Наталія Маслова
/_____/
"____" _____2022 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ

Сердюкова Дмитра Юрійовича
(прізвище, ім'я, по батькові)

1. Тема роботи «Дослідження та розробка методів та алгоритмів машинного навчання для автопілоту машини»/ «Research and development of machine learning methods and algorithms for machine autopilot»

Керівник роботи Назарова І. А., к.т.н., доц. каф. ПМІ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом від " 28 " вересня 2022 року № 446

2. Строк подання студентом роботи "05" грудня 2022 року

3. Вихідні дані до роботи результати науково-дослідної роботи, матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
1) аналіз предметної області;
2) Аналіз існуючих алгоритмів, моделей та методів;
3) Пояснення до обраних алгоритмів, моделей та методів, що будуть використані для подальшої розробки;
4) Опис розробки;
5) перевірка роботи готових результатів роботи;

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) робота містить 43 рисунки та інший матеріал в достатній кількості для демонстрації результатів роботи магістра

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачізавдання 30 вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломної роботи	Строк виконання етапів роботи	Прим.
1	Аналіз та дослідження обраної предметної області, ознаймовлення з інструментами розробки	5.09.22-24.09.22	
2	Проектування моделей за вимогами	24.09.22-16.10.22	
3	Написання функціоналу моделей системи за створеними вимогами	16.10.22-29.10.22	
4	Процес тестування, опис функціонування моделей системи	29.10.22-17.11.22	
5	Оформлення супутніх матеріалів (розробка графічного матеріалу, тощо) та рецензування роботи	17.11.22-20.11.22	
6	Написання доповіді	21.11.22-23.11.22	
7	Захист роботи	6.12.22	

Студент _____ Дмитро СЕРДЮКОВ
(підпис) (прізвище та ініціали)

Керівник роботи _____ Ірина НАЗАРОВА
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сердюков Д. Ю. Дослідження та розробка методів та алгоритмів машинного навчання для автопілоту машини / Випускна кваліфікаційна робота на здобуття освітнього ступеня «магістр» за спеціальністю 122 «Комп’ютерні науки». – ДВНЗ ДонНТУ, Луцьк, 2022.

Для проведення досліджень існуючих алгоритмів, методів та моделей навчання, для аналізу даних та оцінки якості навчання, було обрано сферу, що стосується реалізації автопілоту для машини.

Об’єктом дослідження – дослідження та розробка методів та алгоритмів машинного навчання для автопілоту машини.

Предметом дослідження – алгоритми та моделі навчання нейромережі, аналіз якості їх роботи.

Мета дослідження провести аналіз існуючих алгоритмів, моделей навчання нейронних мереж для автопілоту машини та створити власні алгоритми, які можуть виступати альтернативою в реалізації автопілоту, провести дослідження швидкості навчання, якості роботи.

Практичне значення полягає в покращенні роботи автопілоту за рахунок конфігурації алгоритмів та нейромережі.

Ключові слова: автопілот, алгоритми розпізнавання об’єктів, існуючи автопілоти, OpenCV, процес обробки зображення.

Список публікацій здобувача:

Сердюков Д.Ю. Реалізація автопілоту автомобіля на основі технології carla / Д. Ю. Сердюков, І. А. Назарова // Всеукраїнський науково-практичний форум "ТАК", 2022.

ABSTRACT

Serdyukov D.Yu. Research and development of machine learning methods and algorithms for the autopilot of a car / Graduation qualification work for obtaining an educational degree "Master" in the specialty 122 "Computer Science". - DVNZ DonNTU, Lutsk, 2022.

To carry out research on existing algorithms, methods and models of learning, to analyze data and evaluate the quality of learning, the field related to the implementation of an autopilot for a car was chosen.

The object of the research is the research and development of machine learning methods and algorithms for the autopilot of the machine.

The subject of research is neural network learning algorithms and models, analysis of the quality of their work.

The purpose of the study is to analyze the existing algorithms, learning models of neural networks for the autopilot of the machine and create our own algorithms that can act as an alternative in the implementation of the autopilot, to conduct a study of the speed of learning, the quality of work.

The practical value is in improving the operation of the autopilot due to the configuration of algorithms and neural networks.

Keywords: autopilot, object recognition algorithms, existing autopilots, OpenCV, image processing.

List of publications of the acquirer:

Serdyukov D.Yu. Implementation of car autopilot based on carla technology / D. Yu. Serdyukov, I. A. Nazarova // All-Ukrainian Scientific and Practical Forum "TAK", 2022.

ЗМІСТ

ВСТУП

Завдяки створенню нейронних мереж, люди можуть спрощувати різноманітний набор завдань, таких як водіння автомобілю.

Нейронні мережі мають великий потенціал до вирішення всіх можливих видів задач, що спричиняє постійний зріст їх популярності. Вони можуть знаходити закономірності в наборі інформації, що надають можливість мережі вчитися розпізнавати об’єкти, які потрібні для її коректного функціонування. Основною перевагою нейронної мережі – можливість адаптовуватися до змін, тобто, нейромережа може працювати з динамічною середою, і, якщо буде потрібно, виконає повторне навчання, щоб уникнути помилок. Через дану властивість нейромережа є потрібною і викликає інтерес у різних за величиною компаній [1].

Якщо ми – компанія, що виготовлює власні автомобілі і в нас однією з функцій є автопілот, то ми виграємо в наступних показниках:

- підвищується безпека на дорозі;
- не потрібно витрачати додаткові кошти (аварії, витрати на паливо);
- ми проводимо меньше часу в дорозі.

Використання нейромережі вносить велику кількість покращень, але якщо система погано навчилась виконувати свої задачі та функції – ми не отримаємо нічого.

Чіткого методу для навчання не має, є велика кількість мов програмування, фреймворків, що здатні надати всі можливості для створення алгоритму, моделі навчання [2]. Але якби добре нейромережа не виконала навчання, нам потрібно не забувати про зміни. Автопілот автомобілю може добре виконувати свої функції в місті, але що може трапитись, коли машина буде їхати по бездоріжжю, або в тих умовах, з якими нейромережа не мала справи раніше. Щоб уникнути можливих проблем через змінливі умови, не потрібно забувати про якість набору даних, на яких нейронна мережа буде виконувати навчання.[3]

РОЗДІЛ 1
СУЧАСНІ АВТОМОБІЛЬНІ АВТОПІЛОТИ. ОГЛЯД ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Розвиток автопілотів

Автопілот в машині вже не є чимось незвичним для сучасної людини. На протязі часу вносяться нові функції, змінюються алгоритми та моделі навчання, система автопілотування все більш забирає контроль у водія для зменшення витрат його сил. За часи свого існування автопілот пройшов через наступні етапи:

1) автопілот «без ніг»;
2) автопілот «бер рук»;
3) автопілот «без очей»;
4) автопілот «без уваги»;
5) автопілот «без водія».

Автопілот «без ніг» мав круїз-контроль, систему попередження виїзду з полоси та система, що допомагає під час паркування. Система вже мала вплив на управління машини та надавала допомогу. Круїз-контроль відповідав за безпечну відстань між автомобілями та учасниками дорожнього руху. Парктронік брав під контроль рух машини заднім ходом в складних умовах. Якщо водій втомився – система попередження виїзду з полоси спрацьовує і допомагає людині.[4]

Системи мають вплив на машину і можуть частково забирати відповідальність, але водій все одно відіграє головну роль.

Наступним з’явився автопілот «без рук». Система вже може керувати автомобілем без участі водія. Автопілоту під силу керувати швидкість машини та рульовим управлінням. Система може працювати сама, але водій

все ще повинен слідкувати за роботою. Автоматична система не завжди реагує на ситуацію на дорозі (рис. 1.1).

Рисунок 1.1 – Автопілот «без рук» (https://vc.ru/transport/48947-bespilotnye-avtomobili-obyasnenie-6-urovney-avtonomnosti)

Автопілот «без уваги» – повністю самостійна система, яка бере під контроль управління автомобілем. Водій вже може не слідкувати за роботою системи. Автопілот керує машиною в певних умовах, в більш важких випадках управління переходить до водія (рис. 1.2).

Рисунок 1.2 – Автопілот «без уваги» https://vc.ru/transport/48947-bespilotnye-avtomobili-obyasnenie-6-urovney-avtonomnosti

Автопілот «без очей» може брати повну відповідальність на себе, працюючи без допомоги водія. В даній системі вже використовуються радари, датчики, камери та штучний інтелект, але контроль зі сторони водія все ще має бути.

Автопілот «без водія» – не потребує ніякого контролю зі сторони водія.

1.2 Огляд існуючих автопілотів. Переваги і недоліки

Компанія Tesla була першою компанією яка виготовила автопілот для своїх машин та запустила у масове виробництво.

Автопілот компанії Tesla має наступний функціонал:

- круїз-контроль;
- авто-керування;
- навігація;
- автоматичне включення вказівника повороту та самостійний з'їзд з траси;
- автопаркінг;
- дистанційне переміщення в обмеженому просторі за допомогою мобільного додатку;
- розпізнавання світлофорів та дорожніх вказівників.

Автопілот Тесли працює за допомогою систем ультразвукових датчиків, набору радарів та камер з круговим обзором. Ультразвукові датчики сканують в радіусі до 10 метрів навкруги автомобіля для знаходження перешкод під час паркування або іншого транспортного засобу. Радар надсилає довгі хвилі, знаходить перешкоди на великій швидкості та дає сигнал системі адаптивного круїз-контролю [5]. Камера заднього виду має широкий кут для можливості рухатися заднім ходом. Бічні камери заднього виду для стеження за сліпими зонами. Передні бічні камери – працюють при розвороті автомобілю або коли інший транспортний засіб планує перейти на

іншу полосу. Фронтальні камери працюють для знаходження розмітки, знаків дорожнього руху та світлофорів. Інформація з камер надходить до блоку рульового управління, управління рушієм, системи гальмування.

Автопілот Тесли навчається за допомогою датчиків, дані з яких, зберігаються в комп'ютері. У комп'ютера в наявності нейронна мережа, що оброблює зображення, сонарів та радарів. За допомогою цієї системи полегшується керування для водія, бо він не може услідкувати за всіма подіями на дорозі [6].

Також у системи є і недоліки:

1) руки повинні бути на рулі;
2) потрібно триматися середніх поліс під час руху на автомагістралі;
3) підтримувати швидкісний режим в місті;
4) не адаптований під всі види поганих умов (темрява, погана розмітка дороги і т.д.)
5) через погані погодні умови не правильно розпізнає перешкоди та знаки дорожнього руху.

Поступово починали виходити альтернативи Tesla. Однією з альтернатив був автопілот від компанії Yandex.

Автомобіль має «зір» та «органи». Він може сканувати простір навколо себе і розпізнає інші машини, дорожні знаки та розмітку, перешкоди, людей, тварин та інші елементи навколишнього середовища [7]. Якщо станеться так, що автопілот не може об'їхати перешкоду, автомобіль зупиняється і продовжує рух після того як перешкода буде усунена.

Датчики збирають інформацію про навколишню середу, а спеціальний софт її оброблює. За допомогою Лідару (лазерний радар), що створює тривимірну карту простору, скануючи його з повним обзором в 360 градусів. Тривимірна карта дозволяє визначити відстань до об'єктів [8]. Дані приймаються не лише з лідару, а й з радарів котрі можуть «бачити» набагато далі, ніж лідари.

Машину з автопілотом від компанії Yandex наведено на рисунку 1.3.

Рисунок 1.3 – Машина з автопілотом від компанії Yandex
(https://www.cnews.ru/news/top/2021-07-05_bespilotnoe_taksi_yandeksa)

Також в блоці на даху автомобіля встановлені 3 лідари, 5 камер, антени GPS та мобільного зв'язку GSM, антена супутникової системи навігації GNSS. У передній частині машини є 4 радара, також можуть бути встановлені додаткові радари. Саме такий набір даних, що отримані з різних пристроїв формує найбільш точне позиціонування та безпечну траєкторію для руху машини з автопілотом (рис. 1.4).

Рисунок 1.4 – Робота систем машини з автопілотом
від компанії Yandex
(https://www.cnews.ru/news/top/2021-07-05_bespilotnoe_taksi_yandeksa)

Також автопілот використовує комп'ютерний зір та спеціальні сенсори контролю безпеки руху машини. Точне місце розташування визначається за

HD-картою місцевості [9]. На основі всіх отриманих даних спеціальний алгоритм приймає рішення як буде рухатися автомобіль.

Під час руху автопілот збирає та аналізує дані, всі дані поступають на комп’ютер, що знаходиться в задній частині машини [10]. Потужність комп’ютерів дозволяє оброблювати сотні гігабайт інформації за час кожної поїздки. Дані з кожної машини аналізуються в єдиному центрі після поїздки.

Під час поїздок по реальним дорогам оцінюються всі ситуації, які можуть трапитися:

- освічення;
- тіні;
- сніг;
- туман;
- проблеми з зв’язком;
- транспортні засоби, що порушують ПДР;
- пішохід, що вибіг на трасу.

Автопілот швидко реагує на неочікувані перешкоди.

У 2021 році компанія Toyota представила свій власний автопілот – Advanced Drive.

Даний автопілот має автономність другого рівня по класифікації SAE. Система вже доступна на двох моделях компанії, що продаються в Японії.

Автопілот функціонує на основі даних, що отримуються від цілого комплексу різноманітних датчиків. В набір входять радар, що працює в міліметровому діапазоні, а також, оптичний далекомір та лідар [11]. Окрім того, використовуються високоточні карти для визначення кількості рядів на дорозі, обмеження швидкості та крутизни поворотів.

Робота системи можливо тільки на загородніх шосе, в городі система автокерування не активується.

Автопілот має широкий спектр можливостей: підтримування швидкості та дистанції; утримання автомобіля в смузі; обгін. Автомобілі з автопілотом Advanced Drive представлено на рисунку 1.5.

Рисунок 1.5 – Автомобілі з автопілотом Advanced Drive (https://www.cnews.ru/news/top/2021-07-05_bespilotnoe_taksi_yandeksa)

1.3 Проблеми автопілотів

Автопілоти хоч і можуть брати на себе відповідальність керуванням автомобілем але вони не можуть бути готові до всіх можливих подій на дорозі. На даний момент машини з автопілотами потребують водія, який зможе перехватити керування у незвичних або важких ситуаціях, де система автопілотування не може в повну силу виконувати свої функції [12]. Автопілот автомобіля може лише бути пристосований для перевезень у межі міста, за межами – можуть початися проблеми (відсутність розмітки, погані дороги і т. д.).

Основними проблемами автопілотів є:

- датчики;
- не універсальність;
- ринкова проблема.

Машина погано розуміє навколишнє середовище.

Щоб автопілоти добре виконували свої функції потрібно реалізувати велику кількість речей що будуть надавати можливість виконувати свої обов'язки, незважаючи на:

- погодні умови;
- якість дороги;
- дорожню розмітку;
- дорожні знаки.

Основними речами з якими взаємодія система автоматичного керування автомобіля – це дорога і дорожня розмітка [13]. Для більш кращого розуміння навколишньої ситуації було би великою допомогою оснастити системами, що інформують учасників дорожнього руху більшість вуличної та дорожньої інфраструктури.

Автопілот міг би орієнтуватися за допомогою цих систем, що значно покращило якість його роботи та запобігло виникненню помилок. Також автопілоту було б не зайвим мати системи, що надають інформацію про стан автомобілю, ситуації на дорозі та дії, які вони планують виконати у найближчий час (зупинка, поворот наліво і т.д.) з допомогою таких систем, як машини з автопілотом [14].

На даний момент поширення таких автомобілів неможливе. Щоб це трапилось – спеціалісти з ПЗ та розробники автомобілів з урядами країн повинні працювати разом велику кількість часу для повноцінного втілення ідеї в реальність.

Також є проблемою навчання штучного інтелекту, який буде приймати рішення, керуючи автомобіль на дорозі в реальному часі в постійно мінливому навколишньому середовищі.

Стосовно жертв, бувають моменти коли в створеній ситуації неможливо уникнути сутички і автомобіль для того, щоб врятувати комусь одному життя, може зробити це керуючись власними алгоритмами. Як машина обирає, ким вона має пожертвувати – незрозуміло.

Ситуація з вибором пріоритету представлена на рисунку 1.6.

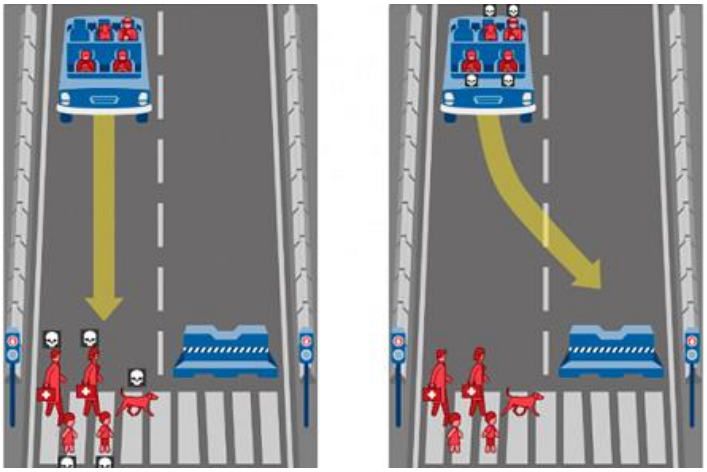

Рисунок 1.6 – Вибір пріоритету

Для того щоб реалізувати поведінку керування автомобіля можна використовувати теорії, такі як деонтологія та утилітаризм. До деонтології відносяться закони робототехніки Азімова, за якими безпілотний технічний засіб дотримується наступних правил:

− автопілот не повинен завдавати шкоди людському здоров'ю, впливати на життя або своєю бездіяльністю допускати такі наслідки;

− автопілоту ставиться в обов'язковому порядку виконувати накази, що віддаються людиною, винятковий виняток – наказ, що віддається, не повинен проводити обстеження положення;

− автопілоту ставиться в провину думка про власну безпеку в тій мірі і в тому обсязі, в якому його дію не спростовують два попередні пункти.

Згідно філософії утилітаризму – повинно робитися все, що добре та зменшує страждання. Якщо кількість задоволення збільшується і зменшується кількість болі – це вважається позитивним результатом. Тобто, щоб виконувати цю умову, система автоматичного керування автомобілем приймаючи кожне рішення повинна робити це у користь зменшення жертв [18].

Нажаль, вже трапилися випадки аварій з машинами, що мало автоматичну систему керування (рис. 1.7).

Рисунок 1.7 – Аварія з машинами, що мали автоматичну систему керування (https://www.ixbt.com/news/2022/01/24/tesla-model-3.html)

1.4 Технології та методи, що використовуються для реалізації автопілоту машини

Для реалізації тестування автоматичної системи керування було обрано симулятор CARLA – симулятор, який використовується для того щоб навчити алгоритми систем автоматичного керування автомобілем. За його допомогою можливо реалізувати імітацію різноманітних наборів сенсорів. Симулятор містить наступний функціонал:

- середовище «Місто»;
- пішоходи;
- різні учасники дорожнього руху;
- динамічна погода.

Для реалізації автопілоту потрібно встановити вимоги, задачі які діляться на наступні етапи:

- задача базових навичок водіння автомобілю;
- задачу реалізації планування руху автомобіля;
- реалізація системи, що приймає рішення під час руху на дорозі.

Задача базових навичок володіння автомобілю – автопілот вчиться визначати навколишнє середовище навколо автомобіля під час руху. Він повинен відстежувати рух машини для відстеження рухомих об’єктів та роботи прогноз їх можливих наступних дій.

Задача реалізації планування руху автомобіля – за допомогою планування руху машини дає можливість успішно дібратися до потрібного місця. Залежно від цілі автопілот обирає найкращий шлях. Він починає обирати як краще проїхати, коли потрібно змінити смугу чи переїхати перехрестя, як він буде виконувати маневри.

Реалізація системи, що приймає рішення під час руху на дорозі – система, що буде приймати рішення при керуванні автомобілем: гальмування, прискорення, утримання в смузі, виконання поворотів. Вона також відповідає за контроль швидкості автомобіля на дорозі. Виконання цих трьох задач під час руху автомобіля обов’язкове для забезпечення належного керування.

Для проєктування автопілотів використовуються спеціальні стандарти. Дані стандарти визначають вимоги та задачі, яких потрібно дотримуватися для повної якісної реалізації.

Одним з таких стандартів є стандарт SAE J3016, в якому написано про рівні оптимізації. В даному стандартів описано наступні рівні:

- нульовий рівень;
- перший рівень;
- другий рівень;
- третій рівень;
- четвертий рівень;
- п’ятий рівень;

В нульовому рівні описується що всю роботу робить водій, будь-який вид автоматизації роботи відсутній.

В першому рівні описується що є допоміжна система, яка допомагає виконувати якусь роботу, що не потребує великих витрат. Як варіант можна

описати система підтримування скорості автомобіля. Система може брати невелику відповідальність, але головний – водій.

Другий рівень – система може виконувати додаткові функції. Окрім підтримування швидкості автомобіля, система може виконувати утримування машини в смузі та зміна полоси руху.

Третій рівень – система може орієнтуватися в навколишньому середовищі та зчитувати об’єкти. Може брати на себе відповідальність в певних умовах і подіях, але водій все ще потрібен. Дана система може лише виконувати свою роботи в звичайних ситуаціях на дорозі і водія може не звертати увагу, але в певних ситуаціях система має повідомити водія, що він має взяти контроль.

Четвертий рівень – система даного рівня вже може уникати ситуацій, в яких може трапитися дорожньо-транспортна пригода. Вона може працювати самостійно але в деяких ситуацій система попрохає водія взяти контроль над машиною (рис. 1.8).

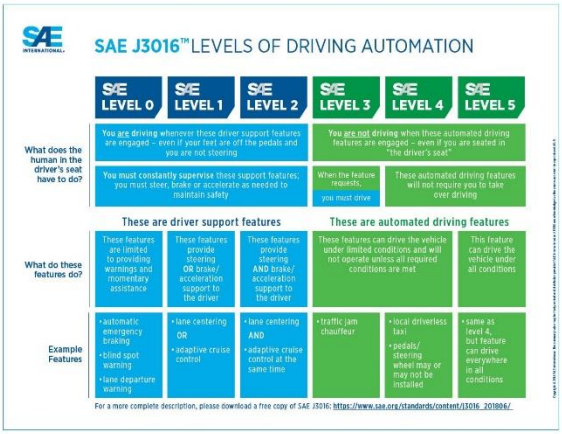

Рисунок 1.8 – Стандарт SAE J3016

Що стосується п’ятого рівня – система п’ятого рівня автоматизації повинна зчитувати навколишнє середовище та приймати рішення в керуванні під кожну ситуації без втручання водія. На даний момент таких систем немає, але ведуться розробки які можливо реалізують цей рівень.

Дана система побудована на закономірностях, може сприймати ситуацію в реальному часі в постійно мінливому середовищі і відповідно до цих ситуацій приймати рішення, які найкраще вирішують поставлені питання.

Система повинна:

- зчитувати навколишнє середовище, щоб розуміти ситуацію на дорозі;
- визначати об’єкти, що рухаються на дорозі;
- визначати рух об’єктів;
- чи є закономірності в русі певних об’єктів;

Система має вміти виконувати прогнозування.

1.5 Процес обробки розпізнавання об’єктів у реальному часі за допомогою нейромереж

В наш час існує багато напрямів, я яких використовується штучний інтелект. Однією з цих сфер створення автопілоту для автомобілю. При русі дорогою автомобіль повинен чітку розуміти ситуацією. Одним з завдань під час руху на дорозі є орієнтування за знаками дорожнього руху.

Системи розпізнавання дорожніх знаків використовують для попередження водія про різноманітні перешкоди на дорозі, а також подають сигнали системі керування безпілотним автомобілем. Дані системи використовуються на автомобільних марках:

- Audi;
- BMW;
- Mercedes;
- Tesla;
- Volvo;

Система що розпізнає дорожні знаки працює наступним чином (рис. 1.9):

1) на вхід поступає відеопотік;

2) передобробка зображення для виключення непотрібної інформації для аналізу;

3) розпізнавання форми дорожнього знаку для подальшого вилучення знака з початкового зображення.

4) надалі виконується розпізнавання форми знаку для подальшої класифікації.

Рисунок 1.9 – Процес роботи з даними

На стадії обробки зображення виконуються наступні руху:

1) приведення зображення до чорно-білого варіанту;

2) згладжування за допомогою фільтра Гауса для зменшення шуму на зображенні;

3) бінаризація – для більш простого пошуку контурів дорожнього знаку.

Нейронна мережа, що займається розпізнавання дорожнього знаку має наступну структуру:

- вхідний шар;
- скритий шар;
- вихідний шар;

Нейронна мережа може мати різні моделі, глибину закритого шару, функції активації. Немає одного варіанту навчання оскільки кожен варіант має плюси та мінуси [28]. Якщо робити багато шарів, то система може довго вчитися, але вона буде більше якісно виконувати свою роботу, якщо шарів буде мало – мережа може погано розпізнавати дорожні знаки, що означає, що навчання не виконує свою роль (рис. 1.10).

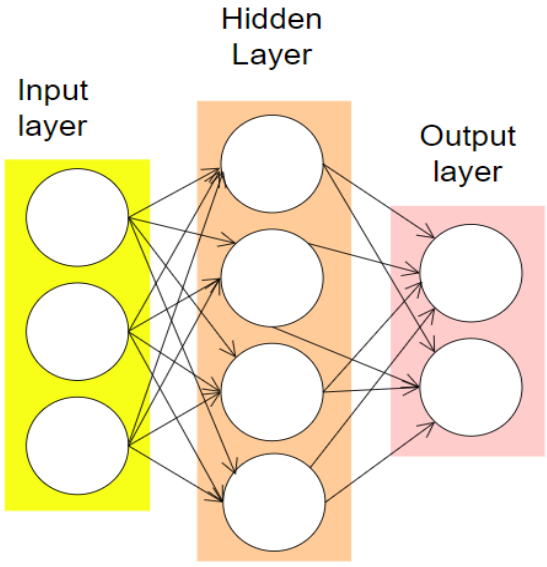

Рисунок 1.10 – Структура нейромережі

Всі сучасні системи розпізнавання образів наділені чітким алгоритмом розпізнавання знаків, що не може підлаштовуватись під умови навколишньої середи, що постійно змінюються [17]. До того ж такі системи виготовлюються тільки машин бізнес класу, в більшості. Системи в таких автомобілях працюють як допоміжні, тобто вони лише інформують водія про обмеження на дорозі, тому що дані системи на даний час не можуть відповідати всім вимогам.

1.6 Представлення базових одиниць нейронної мережі в математичному вигляді.

Базовою одиницею в нейромережі є нейрон. В нейромережі нейрон – це основний елемент, який використовується в усіх процесах, що задіяні в нейромережі (рис. 1.11).

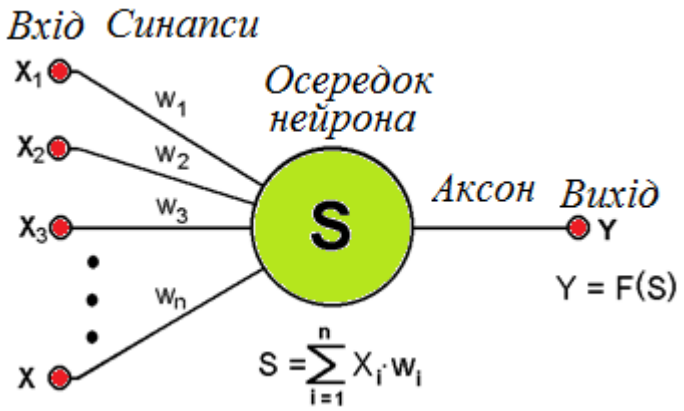

Рисунок 1.11 – Представлення нейрона (https://www.wikiwand.com/uk/Методи_представлення_знань#Media/Файл:Штучний_нейрон.png)

Нейрон собою представляє наступну структуру:

- набір входів;
- набір ваг для входів;
- осередок нейрона;
- вихід нейрона.

Набір ваг також має назву синапси. За допомогою даних синапсів можна зв’язати входи з нейронами.

Осередок нейрона – це суматор. Даний суматор виконує функцію сумування множення входу на синапс.

Також у нейрона є функція активації. Функція активації займається завданням обчислення виходу нейрона, який подається на вхід до інших нейронів.

Для роботи нейромережі можуть використовуватися наступні функції активації:

- сигмоїдальна;
- гіперболічний тангенс;
- ReLu;
- ступінчата;
- лінійна.

Сигмоїдальна функція також має назву логічна функція. До ознак сигмоїдальної функції належать наступні:

- гладка;
- монотонна;
- зростаюча;
- нелінійна.

Виходячи з того, що дана функція є нелінійною, то сигмоїдальна функція (рис. 1.12) може бути використана в нейронній мережі в великій кількості шарів. Значення даної функції лежать в межах [0;1].

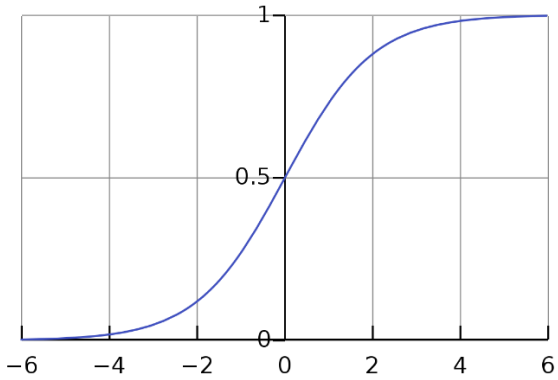

Рисунок 1.12 – Сигмоїдальна функція (https://en.wikipedia.org/wiki/Sigmoid_function#/media/File:Logistic-curve.svg)

Гіперболічний тангенс – дана функція є також сигмоїдальною функцією, але вона скорегована

До ознак функції гіперболічного тангенса належать наступні:

- гладка;
- монотонна;
- зростаюча;
- нелінійна.

Гіперболічний тангенс, в більшості випадків, використовується, коли не потрібно нормалізувати дані, в інших випадках використовується сигмоїдальна функція. Через специфічність функції, а саме те, що вона відцентрована відносно 0, що дає можливість обчислювати градієнт для того, щоб рухатися в якомусь певному напрямі [42].

Функція гіперболічного тангенса дає велику амплітуду для градієнтів, через що отримується більш швидке сходження (рис. 1.13).

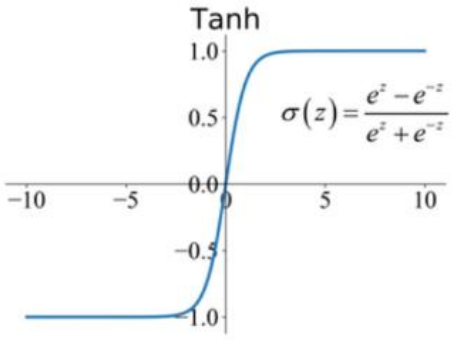

Рисунок 1.13 – Гіперболічний тангенс (https://production-media.paperswithcode.com/methods/Screen_Shot_2020-05-27_at_4.23.22_PM_dcuMBJl.png)

Функція ReLu – функція яка використовується як функція активації для глибокого навчання найчастіше.

На відміну від попередніх функцій, гіперболічного тангенса та синусоїда, функція ReLu має наступні переваги:

- функція дуже швидко обчислює похідну;
- не потребує багато ресурсів;
- діапазон значень [0;1];
- функція має розріджену активацію

В функції є одна проблема. Дана функція під час роботи може «помирати». Коли функція після виконання обчислень дорівнює 0, то градієнт для даної функції буде таким самим [19]. Дана проблема є причиною того, що через ваги, що не змінюються під час градієнтного спуску – нейронна мережа не просувається в навчанні і стоїть на місці (рис. 1.14).

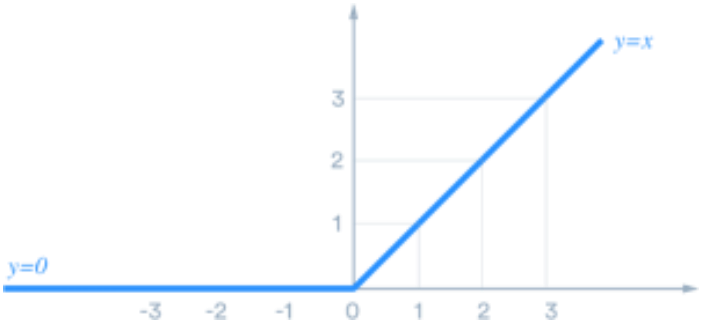

Рисунок 1.14 – Функція ReLu

(https://neerc.ifmo.ru/wiki/images/6/60/LReLuFunction.jpg)

Стосовно ступінчатої функції, дана функція є пороговою. Вона працює наступним чином: якщо значення x більше чи менше певного значення – нейрон буде активований [29].

Дана функція дуже добре підходить для роботи процесу бінарної класифікації. Але дана функція не виконує свою роботу, коли є велика кількість класів (рис. 1.15).

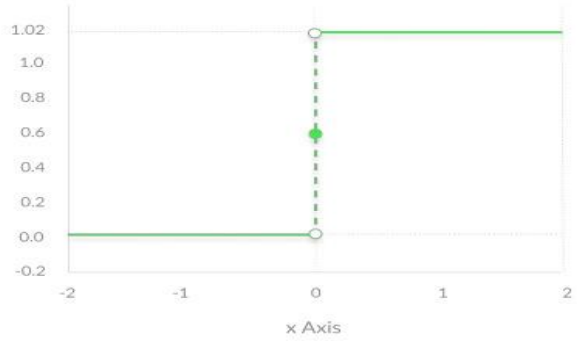

Рисунок 1.15 – Ступінчата функція
(https://neerc.ifmo.ru/wiki/images/f/f2/BinaryStepFunction.jpg)

Говорячи про лінійну функцію, то дана функція також може виступати, як функція активації [13].

Це пряма лінія, результат даної функції залежить від переданого до неї значення. В даної функції є відмінність від інших. В попередніх функціях можливо було отримати результат обчислень 0 або 1. Дана функція, окрім значень 0 та 1, може повертати вихід, що дуже допомагає під час виконання процесу класифікації об’єкта і де є велика кількість класів.

Лінійна функція також має свої недоліки:

− використовуючи дану функцію активації, неможливо використати метод зворотного просунення похибки;

− при використанні функції в великій кількості шарів – результат завжди буде залежати від початкового значення;

В основі методу зворотнього просунення похибки лежить градієнтний спуск. Для того, щоб виконати пошук даного градієнту, потрібно взяти похідну, що буде слугувати для даної функції активації як константа, та не буде залежати від вхідних значень

Під час оновлення ваг важко зробити висновки, чи збільшується емпіричний ризик, чи ні.

Через те що, для кожного шару значення лінійне – трапляється комбінація з лінійностей, що є результатом роботи лінійної функції (рис. 1.16).

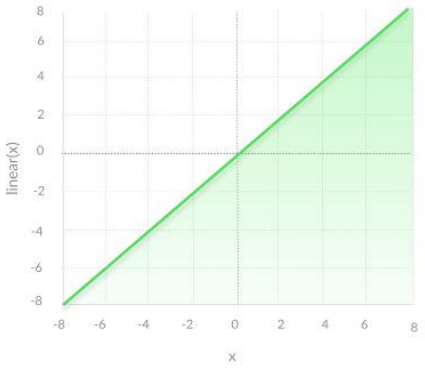

Рисунок 1.16 – Лінійна функція
(https://neerc.ifmo.ru/wiki/images/0/08/LinearFunction.jpg)

1.7 Алгоритми що використовуються при навчанні нейронних мереж

Існує такий алгоритм як метод Віоли-Джонса. За допомогою даного алгоритму можна розпізнавати різні об’єкти в реальному часі.

В роботі даного алгоритму застосовані наступні принципи:

1) зображення представляється в інтегральному представленні. це матриця в якої однаковий розмір з вихідним зображенням;

2) в алгоритмі використовуються ознаки Хаара, в роботі використовуються суміжні прямокутні частини, вони знаходяться на зображенні, де піксельні інтенсивності мутуються для подальшого вирахування різності сум;

3) бустінг – використовується набір алгоритмів, кожний з яких покриває недоліки інших;

4) признаки збираються у класифікаторі, який видає результат true чи false. до входу подається певна кількість об’єктів, які відповідають певним класам;

5) якщо потрібний об’єкт не знайдено в певній частині – вона відкидається.

В своїй роботі даний метод використовує метод «скануючого вікна» [33].

Даний алгоритм працює наступним чином: є вихідне зображення, встановлено вікно для скасування, встановлені ознаки; вікно проходить по зображенню повільно; при скануванні змінюється масштаб в вікні; якщо було знайдено признаки – вони потрапляють у класифікатор, який виносить результат роботи.

Існують спеціальні інструменти які надають функціонал та полегшують знаходження та розпізнавання об’єктів:

1) K-Nearest Neighbour;
2) OpenCV.

K-Nearest Neighbour – використовує дані для класифікації базуючись на мірі, за якою вони подібні.

1.8 Висновки за розділом 1

В даному розділі було досліджено існуючі автопілоти, що використовуються в житті, їх переваги та недоліки. Було визначено проблеми автопілотів, які є важливими та на які потрібно звертати увагу і намагатися знайти спосіб їх вирішення.

Було розглянуто технології та методи, що використовують у розробці моделей нейромереж, що виконують завдання керування транспортного засобу під час руху на дорозі.

Було розглянуто процесс обробки об’єктів у реальному часі за допомогою нейронних мереж. Визначено базові одиниці нейронної мережі в математичному вигляді.

Досліджено роботу алгоритмів, що застосовуються під час навчання нейронних мереж.

РОЗДІЛ 2
АНАЛІЗ ІСНУЮЧИХ АЛГОРИТМІВ КЕРУВАННЯ АВТОМОБІЛЕМ

2.1 Огляд алгоритмів, що використовуються в реалізації автопілоту для машини.

В реалізації роботи автопілоту для машини може використовуватися велика кількість алгоритмів.

Регулятором називається пристрій, котре слідкує за роботою об’єкту управління та робить спеціальні сигнали, за допомогою який, даний пристрій керується.

Серед існуючих алгоритмів можна виділити наступні:

- П-регулятори;
- ПІ-регулятори;
- ПІД-регулятори;
- лінійні регулятори;
- нелійнійні регулятори;
- управління за помилкою;
- алгоритм Калмана та варіанти його реалізації.

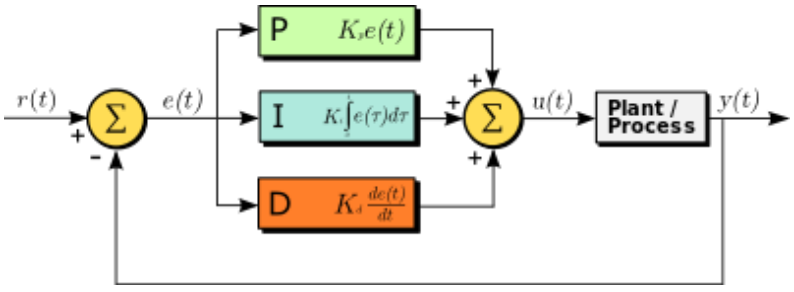

Рисунок 2.1 – Пропорційно-інтегрально-диференційований регулятор (https://product-help.schneiderelectric.com/Machine%20Expert/V1.1/en/Util/topics/pid.htm)

Пропорційний-інтегрально-диференційний регулятор – даний пристрій використовується для системах управління, що формують керуючий сигнал для отримання потрібної точності та якості (рис. 2.1).

У ПІД-регуляторі є недоліки, які, якщо не враховувати, то в системі регулювання можуть виникнути сторонні ефекти під час реалізації каналу сигналу похибки. Вони з'являються через посилання сигналу каналу, через що й зростає частота.

ПІД регулятори використовуються зазвичай в к круїз контролі транспортного засобу.

Під регулятор може використовуватися наступним чином: кількість палива, що подається до двигуна, відображається на виході. Спідометр займається порівнюванням з швидкістю, що є бажаною, вихідний сигнал, в свою чергу налаштовується. Таким чином автомобіль може регулювати газ, під час руху по місцевості [11].

Фільтр Калмана – даний фільтр працює оптимізатором помилок і основною вимогою є те, що система повинна бути лінійною.

Один із підходів фільтру Калмана – лінеаризація системи навкруги стану, що є в даний момент, та змушення фільтра, щоб він використовував цю версію як модель. Такий варіант має назву EKF.

Проблема даної моделі в її нестабільності, коли вона багато разів доходить до рішення, що вважається правильним, це робиться надто повільно:

$$x = \hat{x} + \delta x.$$

В принципі роботи використовуються наступні елементи:

x – справжнє значення;

$\hat{x}$ – номінальне значення;

δx – значення помилки.

Кінематики стану помилки:

$$\underbrace{x_k - x_{k-1}(\hat{x}_{k-1}, u_{k-1}, 0)}_{\delta x_k} = F_{k-1}\underbrace{(x_{k-1} - \hat{x}_{k-1})}_{\delta x_{k-1}} + L_{k-1}W_{k-1}.$$

Для виразу лінеаризованої моделі вимірювання через стан похибки використовують рівняння:

$$y_k = h_k(\check{x}_k, 0) + H_k(x_k - \check{x}_k) + M_k v_k.$$

2.2 Категорії нейронних мереж

В побудові алгоритму навчання використовуються різні види мереж. Кожна з них має свої недоліки та переваги.

Існують наступні види нейромереж:

- модель, що орієнтована на увагу;
- генеративна модель;
- згорткова модель;
- пірамідальна модель;
- мережа, що повністю згортується.

Модель на основі уваги – модель що орієнтується на частину інформації. Вона відкидає велику кількість зайвої інформації, що не потребує уваги. Дана модель дуже добре працює з нейромережами як CNN.

Генеративна модель – це алгоритми які потрібні для навчання без вчителя, побудовані на основі двох нейронних мереж, що змагаються поміж собою.

Даний вид нейромережі може бути корисний для створення фотореалістичних зображень, реконструкцій тривимірних об’єктів зображень.

Згорткова нейронна мережа відноситься до класу нейромереж прямого поширення. В даній мережі застосовуються різні види багатошарових перцептронів.

Операція на якій побудована дана нейромережа – згортка. За рахунок згортки нейромережа зменшує об’єм інформації, яких займає місце в пам’яті. Таким чином нейромережа краще знаходить ознаки зображення, на які вона може опиратися.

Дана модель має шари входу, виходу та приховані шари. В прихованих шарах можуть використовуватися наступні:

- згорткові шари;
- арегувальні шари;
- повноз’єднані шари;
- шари нормалізації.

Під час роботи з зображенням, нейромережа поділяє його на сектори, в яких може міститися об’єкт. Виконується ряд операцій та виконується поступове зменшення сектору (рис. 2.2).

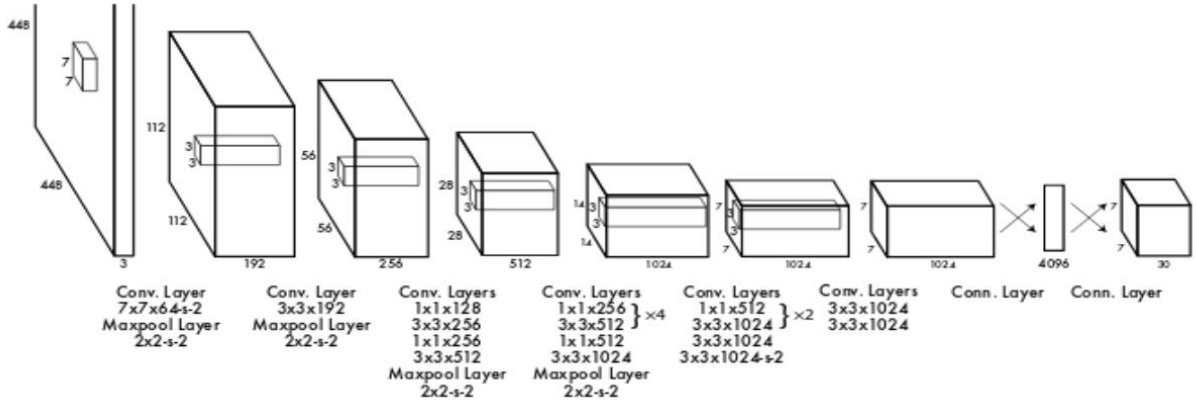

Рисунок 2.2 – Структура згорткової нейромережі (https://www.researchgate.net/figure/Architecture-of-the-convolutional-neural-network_fig2_321875630)

Робота згорткової нейромережі була запропонована інформаційна технологія. Розпізнавання нетипових ситуацій на дорозі складається з таких етапів:

1) відеопотік розбивається на ряд послідовних фреймів;

2) фрейми подаються на вхід нейронної мережі;

3) нейромережа робить розпізнавання об’єктів на зображенні, які беруть участь у дорожньому рухові;

4) кожний об’єкт окреслюється прямокутних відповідного розміру;

5) для кожного об’єкта встановлюються розміри та геометричний центр;

6) проводиться відстеження об’єктів, що були знайдені;

7) значення об’єктів, що були знайдені, записуються у пам’ять та контролюються;

8) якщо пороги були перевищені – висока імовірність виникнення нетипової ситуації, про яку повідомляється.

Для навчання нейромереж, як варіант може використовуватися метод «Transfer Learning». Сенс даного методу в використанні великого набору даних з якими нейромережа раніше не працювала. Через те що нейромережа потребує великі об’єми даних, оскільки може трапитися ситуація, коли нейромережі недостатньо даних, щоб виконати класифікацію.

Даний метод використовується наступним чином. До раніше навченої нейромережі додається певна кількість шарів, що приховані, які в свою чергу, дозволяють в вирішенні більш конкретної задачі використовувати знання, які були вже отримані.

Алгоритм навчання містить наступні кроки:

1) будування пайплайнів даних;

2) будуються графи мережі;

3) звеличення значень «вузьких міст»;

4) створення зображених, що спотворені;

5) налаштування функцій втрат та обчислення якості передбачень.

Тренування може проходити з наступними спотвореннями зображення:

- обрізка;
- масштаб;
- переворот.

Для проведення тренування з даними спотвореннями потрібно виконувати розрахунки для кожного зображеннями.

Застосування спотворень допоможе покращити навчання, якщо пропустити такі етапи як масштабування, обрізання, переворот зображення.

Операція обрізання зображення робиться через розміщення рамки, які заходиться в якомусь місті на зображенні (рис. 2.3).

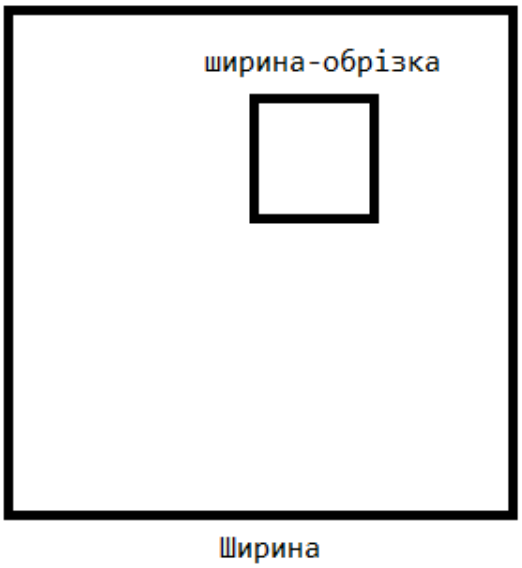

Рисунок 2.3 – Обрізання зображення

Для того, щоб розрахувати критерій похибки в задачі класифікації використовується наступна формула:

$$E = -\sum_{i=1}^{n}\sum_{j=1}^{m} t_j^{(i)} \log\left(o_j^{(i)}\right).$$

В даній формулі використовуються наступні елементи:

- сумма навчаючої вибірки;
- сумма класів;
- передбачене моделлю;

- справжній клас.

Для розпізнавання зображень застосовується R-CNN. В роботі використовується вибірковий пошук, який виділяє множину регіонів з зображення та називає їх пропозиціями. Робота з обмеженою кількістю частин зображення, що покращує швидкість роботи (рис. 2.4).

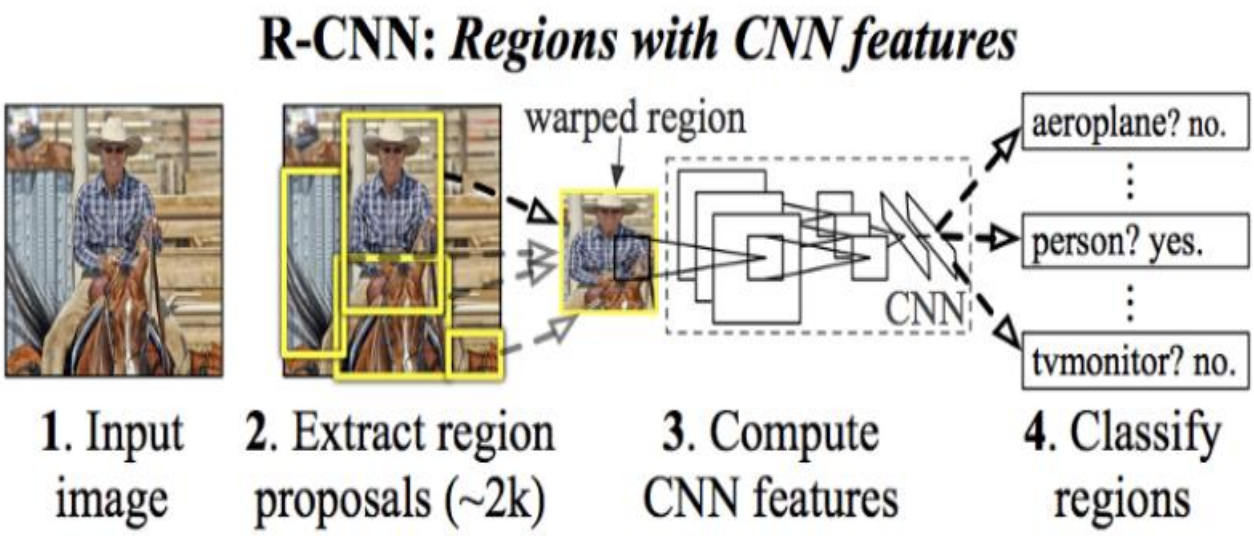

Рисунок 2.4 – R-CNN
(https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e)

Проблеми R-CNN:

- займає велику кількість часу для тренування;
- не можливо використовувати у реальному часі через затримку під час класифікації;
- фіксований алгоритм (не проходить ніякого навчання, може призвести до поганої вибірки кандидатів).

Також ще існує такий вид як Fast R-CNN. Даний алгоритм працює швидше за попередній.

Алгоритм працює наступним чином:

1) на вхід подається зображення;
2) створюється згорткова карта;

3) визначається область пропозицій та за допомогою об’єднання ROI встановлюється фіксований розмір;

4) з вектора ознак використовується шар softmax, для того, щоб передбачити клас області, що запропонована.

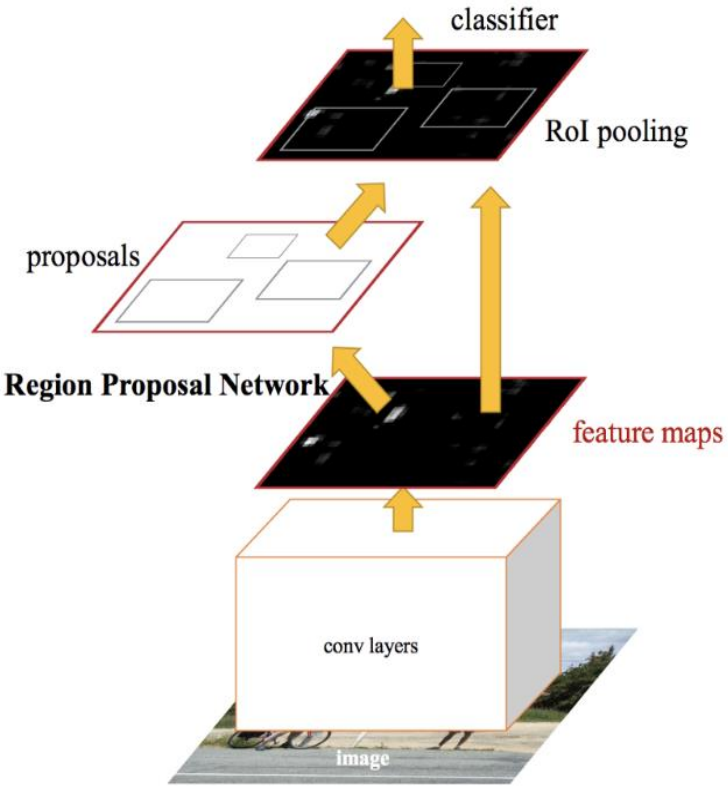

Рисунок 2.5 – Fast R-CNN (https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e)

Yolo – модель згорткової нейронної мережі. Дана модель може виконати розпізнавання великої множини об’єктів та вказати їх властивості:

- розмір елемента;
- клас, до якого відноситься елемент;
- місце розташування елемента на кадрі.

В основу даної моделі закладено фреймворк DarkNet. Фреймворк надає моделі структуру розрахунків, що потрібні для процесу навчання та роботи моделі (рис. 2.6).

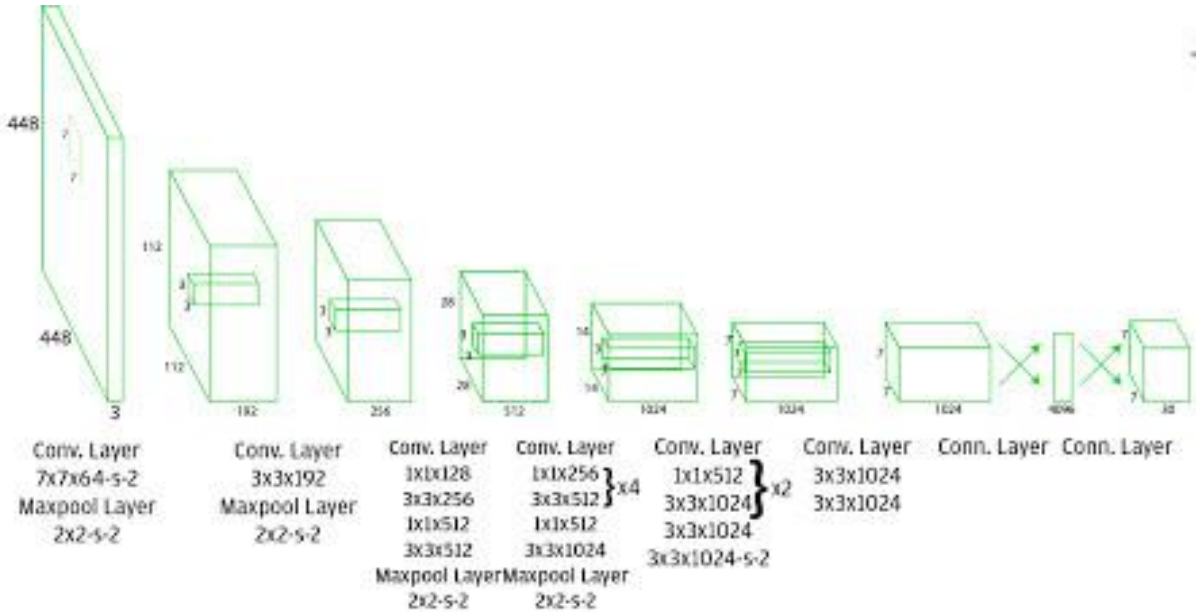

Рисунок 2.6 – Yolo Structure
(https://odsc.medium.com/overview-of-the-yolo-object-detection-algorithm-7b52a745d3e0)

Дана модель складається з великою кількості різноманітних шарів. Одними з таких шарів є:

- згортковий шар;
- залишковий шар.

Основним шаром в нейромережі є згортковий. Даний шар виконує операцію виділення ознак, що необхідні для роботи (межі, лініїї і тд.), їх можливі варіанти розташування, що допомагає робити розпізнавання об’єктів, що важко знайти на зображенні.

В свою чергу, залишковий шар займається розрахунками та приймає вхідну інформацію.

Залишковий та згортковий шари складають основу розрахунків. У кожного шару присутній свій шар активації, що виконує роль збільшення нелінійності. Даний шар використовує функцію активації ReLu (рис. 2.7).

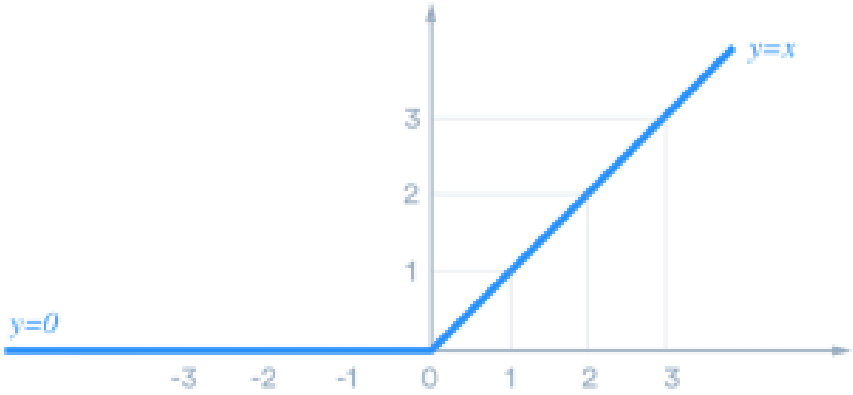

Рисунок 2.7 – Функція активації ReLu

((https://neerc.ifmo.ru/wiki/images/6/60/LReLuFunction.jpg))

Під час процесу розпізнавання елементів на зображенні, нейромережа приводить зображення до квадратного та поділяє його на блоки, що є меншими квадратами. Після операції отримуємо матрицю (рис. 2.8).

$$B = \begin{pmatrix} x_1 y_1 h_1 w_1 c_1 \\ \vdots \; \vdots \; \vdots \; \vdots \; \vdots \\ x_q y_q h_q w_q c_q \end{pmatrix},$$

Рисунок 2.8 – Матриця

В даній матриці присутні наступні елементи:

- x_i, y_i – координати об’єкту;
- h_i, w_i – ширина та висота класу об’єкту;
- a_i – номер, що відповідає класу об’єкта.

Кожен такий блок займається прогнозування класу за признаками, що були виділені під час операції згортування. Відповідно до кількості класів, що були реалізовані в даній моделі нейронної мережі – кількість вихідних показників буде змінюватися.

Навчання данної нейромережі виконується за декілька етапів. За час навчання моделі кожний етап виконує такі операції як оптимізація ваги,

використовуючи градієнтний спуск, та зворотне поширення похибки (рис. 2.9-2.10).

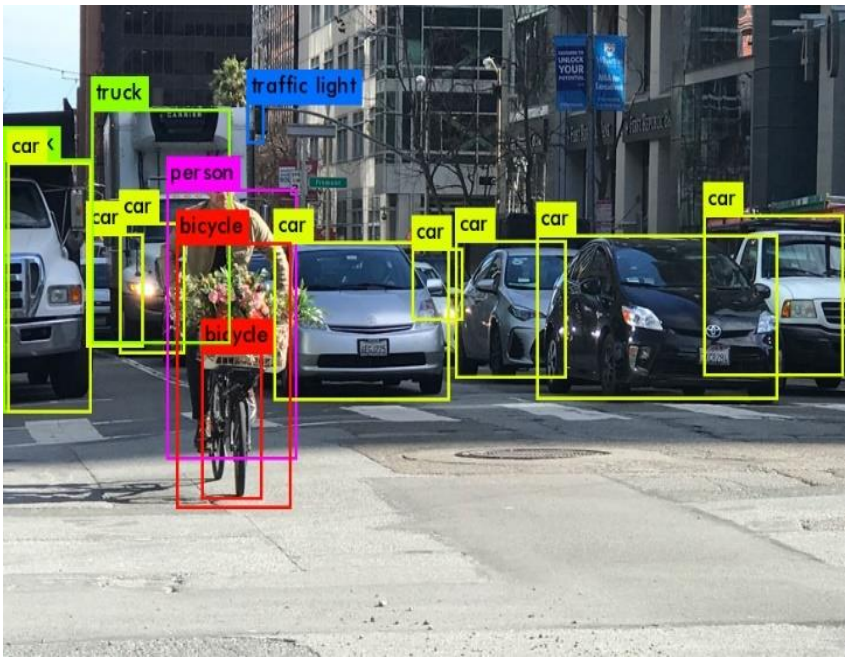

Рисунок 2.9 – Результати навчання Yolo
(https://medium.com/analytics-vidhya/yolo-explained-5b6f4564f31)

Дана технологія може бути використана в реальному житті в реальному часі в якості зору транспортного засобу[44]. Незалежно що буле відбуватися навкруги (рух автомобілів, пішоходів), алгоритм Yolo здатний швидко ідентифікувати. Також за допомогою інших сенсорів, транспортний засіб зможе завчасно прийняти рішення про подальше планування руху без затримок в реальному часі.

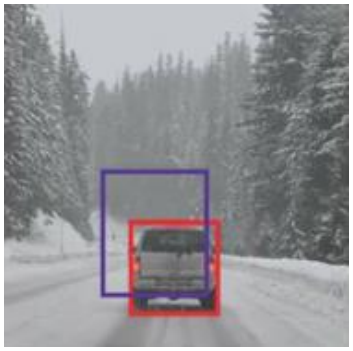

Рисунок 2.10 – Розпізнавання автомобіля на дорозі

Для нейронних мереж що використовуються для розпізнавання зображень використовується такі підходи як «fine-tuning» та «тонке налаштування».

Використовуючи даний підхід, береться вже навчена мережа. Передається структура початкової мережі, а її ваги використовуються для ініціалізації. Після цього нейромережа виконує навчання на обраному наборі даних.

Задля запобігання перенавчення нейромережі швидкість навчання встановлюється менше, ніж під час значення швидкості під час звичайного навчання нейронної мережі (рис. 2.11).

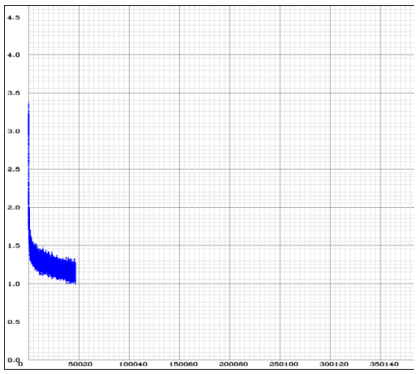

Рисунок 2.11 – Графік помилки нейромережі під час навчання

Для розрахування відстані між об’єктами використовується метрика Евкліда для двовимірного простору:

$$dist_j = \min\left(d\left(v_j, v'_k\right)\right) = \min\left\{d\left(v_j, v'_1\right), d\left(v_j, v'_2\right), \dots, d\left(v_j, v'_m\right)\right\}:$$

$$= \min\left\{\sqrt{\left(\left(x_j - x_1\right)^2 + \left(y_j - y_1\right)^2\right)}, \dots, \sqrt{\left(\left(x_j - x_m\right)^2 + \left(y_j - y_m\right)^2\right)}\right\},$$

де $dist_j$ – мінімальна відстань від v_j;

v_j – j-й об’єкт із V;

v_k – k-й об’єкт із V.

Далі всі об’єкти проходять сортування за відповідними їм значенням d_j. Кожен з цих об’єктів, починаючи з об’єктів з найменшим в d_j, ставитися в відповідність найближчий до нього об’єкт V.

Але дана модель також має свої недоліки і один з таких недоліків – модель не може розпізнавати декілька об’єктів, групу, які знаходяться близько або далеко.

Region Proposal Network – даний модуль використовується для того, щоб заміни алгоритм Selective Search повністю.

Для роботи необхідні загальні ваги з мережею, для того щоб максимально покращити швидкість роботи, що може діставати ознаки елементів, що необхідні для роботи. Для нейромережі RPN, вхід – це карта, що складається з ознак, що були отримані після цієї нейромережі.

В рецептивного поля, що знаходиться в даній мережі, наявні наступні характеристики, що застосовуються у роботі:

- ефективне згортування – 16;
- розмір рецептивного поля -16.

Дані характеристики значать, що на відміну від вихідного зображення, а саме його розміру, карта ознак в результаті роботи буде в 16 разів менша. Пікселі вихідного зображення мають вплив на кожне значення ознаки, в її комірках. Дані пікселі знаходяться в області, що має розміри 196x196 (рис. 2.12).

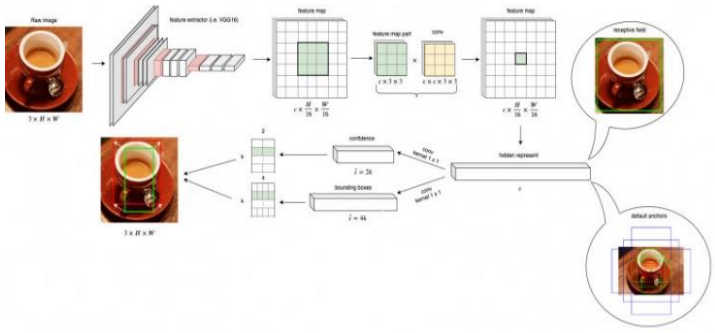

Рисунок 2.12 – Алгоритм RPN

(https://hsto.org/webt/l_/ih/9h/l_ih9hxpiggn2ryaey7r2akxlxk.jpeg)

Алгоритм роботи RPN складається з наступних етапів:

1) отримуємо карту ознак;
2) застосовується шар, що згортується, розміром 3x3.

Під час навчання використовується функція, що робить розрахунки втрат. Для визначення втрат використовуються наступні класи:

- позитивні, які мають перетин більш, ніж 0.7 або ті класи які мають найбільший перетин серед інших;
- негативні це всі ті, що мають перетин менше, ніж 0.3;
- всі інші, які не беруть участь в навчанні.

Є правило яке використовується, для присудження класу:

$$p_i^* = \begin{cases} 1 & if\ IoU > 0.7 \\ 0 & if\ IoU < 0.3. \\ nothing & otherwise \end{cases}$$

З такими значення наступна функція мінімізується:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \gamma \frac{1}{N_{loc}} \sum_i p_i^* L_{reg}(t_i, t_i^*)$$

В даній функції використовуються:

- i –номер якор;
- p_i – ймовірність знаходження об’єкта в i якорі;
- p_i^* – номер класу, що є правильним;
- t_i – передбачені поправки до координат;
- t_i^* – поправки, що є очікуваними до координат;
- $L_{cls}(p_i, p_i^*)$ – log-loss, що бінарним;
- $L_{reg}(t_i, t_i^*)$ – SmoothL1 втрати (даний loss тільки активується якщо $p_i^* = 1$, або гіпотеза містить хоч який-небудь елемент);
- $\{p_i\},\{t_i\}$ – виходи, класифікованої та регресійної моделі відповідно;

− γ – коефіцієнт, що використовується для налаштування балансу між класифікацією та регресією.

Під час процесу розпізнавання зображення, використання нейронних мереж робиться наступним чином:

1) на вхід до нейромережі поступає зображення, що генерує карту ознак;

2) кожна комірка кожної ознаки проходить обробку через RPN, після виконання обробки, результатом є поправки до положення якорів та ймовірність розташування об’єкта відповідного класу;

3) передбачені межі надходять на подальшу обробку Fast R-CNN частини;

4) на виході отримуємо готовий клас елементів та їх розташування на зображенні.

Від попередніх мереж дана нейромережа відрізняється наступним:

− за допомогою спеціального модуля, що окремо диференціюється, проходить операція генерації гіпотез;

− через появу RPN модуля з’явилися зміні під час виконання операції обробки зображення;

− дана нейромережа має найкращу швидкість роботи, з усіх представлених;

− дана нейромережа виділяється тим, що вона дуже точно виконує свою роботу.

2.3 Створення навколишнього середовища та вибір датасету для навчання автопілоту

Для навчання автопілоту було обрано датасет KITTI Vision. Даний датасет надає великий вибір набору даних. В наявності як статичні так і динамічні елементи (рис. 2.13).

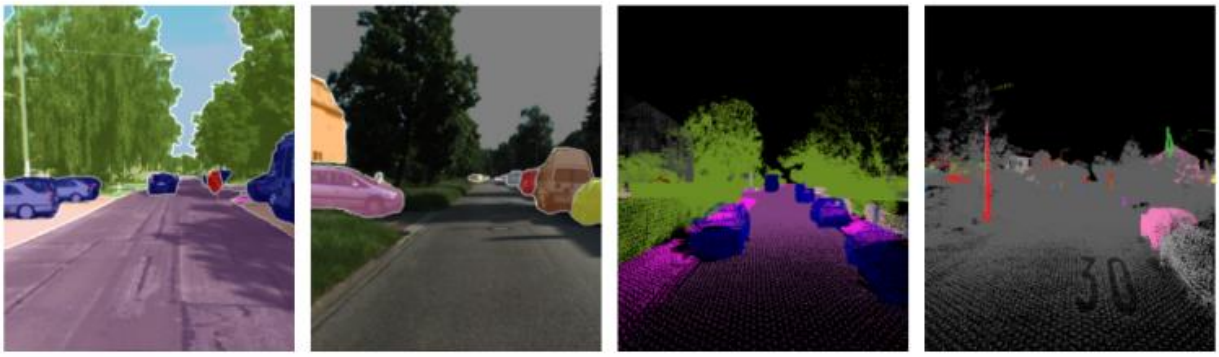

Рисунок 2.13 – Елементи з набору датасету KITTI Vision

За допомогою набору даних, які містять інформацію про різноманітні кількості об’єктів, що можуть задіяні в навчанні системи автоматичного керування транспортного засобу, система навчається робити задачу класифікації.

Рисунок 2.14 – Вид спереду

Процес, під час якого система намагається встановити клас об’єктів, згідно їх конкретних ознак, що відрізняють їх від інших елементів чи класів, називається класифікація.

Певний елемент набору, що знаходиться в певному наборі датасету, що використовується для навчання автопілоту відноситься до певного класу.

Класом називають множину об’єктів, що знаходяться в одному наборі, які схожі між собою за певними ознаками, які є для всіх єдиними, що ідентифікує їх як особливий вид серед інших.

До ознак може відноситися деяка властивість, яку можна охарактеризувати, описати та яка має пряме відношення до елемента. За даними ознаками проходить класифікація елементу. За допомогою даних класифікатор відносить елемент до певного класу.

В ситуації навчання системи автокерування транспортними засобом також присутня задача класифікації, оскільки під час руху на дорозі зустрічається велика множина об’єктів, за якою система повинна орієнтуватися для подальшого планування своїх рухів: утримання авто в смузі під час руху, регулювання швидкості під час руху в трафіку, виконання поворотів ліворуч або праворуч, виконання розворотів.

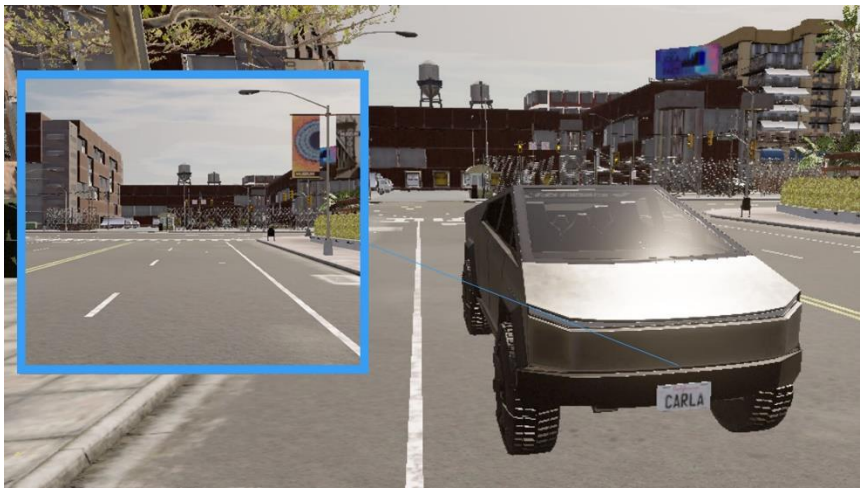

Рисунок 2.15 – Знаходження сенсору з переду транспортного засобу

Процес розпізнавання образів – процес, під час якого вихідні дані відносяться до певних класів, з якими в них є певні однакові риси, ознаки, за якими їх класифікувати до класу.

Зазвичай, вихідні дані надходять з камери автопілота, які під час руху на знімає ситуацію на дорозі, паралельно розпізнаючи образи, елементи та класифікує їх до певних класів за ознаками які вона знаходить.

До елементів, які можуть бути розпізнані та класифіковані відносяться наступні:

- інші транспортні засоби;
- люди;
- дорожня розмітка;
- тварини;
- природні елементи
- світлофори.

Всі ці елементи відносяться до дорожнього руху, і камера повинна вчасно їх класифікувати задля запобігання дорожньо-транспортних пригод, в яких можуть не трапитися жертви.

Під час будування датасетів для системи автоматичного керування можна зіткнутися з двома проблемами:

- збирання даних;
- маркування даних.

Перша проблема збирання набору даних, що будуть використані в датасеті для навчання нейромережі автопілоту. Набори даних можуть збиратися наступним чином:

1) напів-автономні автомобілі та системи підтримки водіння;
2) штучне симульована реальність на основі механізмів комп’ютерних ігор;

Дуже значущою проблемою доставка автомобілів на великі відстані. Автомобіль з автопілотом можуть виникнути труднощі з керуванням потрапивши в незнайоме середовище.

Коли збирається датасет для навчання нейронної мережі, яка зможе керувати автомобілем по всьому світу – дуже тяжко створити датасет, що буде містити в собі зображення з усіх частин світу.

Саме головне та базове завдання для комп’ютерних алгоритмів – розпізнавання об’єкту на зображенні.

Під час розпізнавання зображення алгоритмами виникають такі проблеми:

− потрібно, щоб розпізнавання об’єкту відбувалось в умовах реального часу;

− велика кількість елементів в середовищі, що може збити автономну систему.

Перша проблема може бути вирішена за допомогою тренування моделі на даних, що були надіслані сенсором як результатом, переключивши модель на аналіз сигналу, а не на розпізнавання об’єктів на зображенні.

Друга проблема є прикладом звичайної проблеми для штучного інтелекту, що нездатний узагальнити на має ніяких попередніх знань щодо об’єкту. Рішення проблеми – збагачення розпізнавання зображений за допомогою часткових доказів – метод, котрий дозволяє нейронній мережі використовувати частину додаткової інформації, для виключення найменш відповідного результату.

Відстеження об’єктів виконується для надання системі автономного контролю транспортного засобу інформацію про поточний рух та передбачення руху об’єкта

В межах виконання даної задачі існують наступні проблеми:

− оцінка риску;

− мінливий фон.

− незрозумілі об’єкти.

Стосовно оцінки риску, нейромережа повинна виконувати передбачення руху елементу, а й його поведінку.

Мінливий фон – під час відстеження об’єкта система автономного контролю потрібно розбиратися з змінами на фоні. Інші транспортні засоби наближаються, змінюється колір дороги, види навколишньої середи (від будинків до лісів та степів). Реальний водій може легко впоратися з цим

завданням, але у системи автономного контролю транспортним засобом можуть виникнути певні труднощі у виконанні даного завдання.

Об’єкти на дорозі можуть повторюватися кожний день. На дорозі можна зустріти велику кількість різних марок транспортного засобу одного кольору. Може статися що система не зможе класифікувати одне авто через до контролера надійде неточні дані.

2.4 Розробка автомобілю автопілота

Автопілот виконує багато функцій та має багато модулів, кожен з яких виконує певні обов’язки. В автопілоті можна виділити 5 основних елементів:

- сприйняття автопілотов середовища, що оточує його;
- керування автомобілем;
- планування руху;
- контролер.

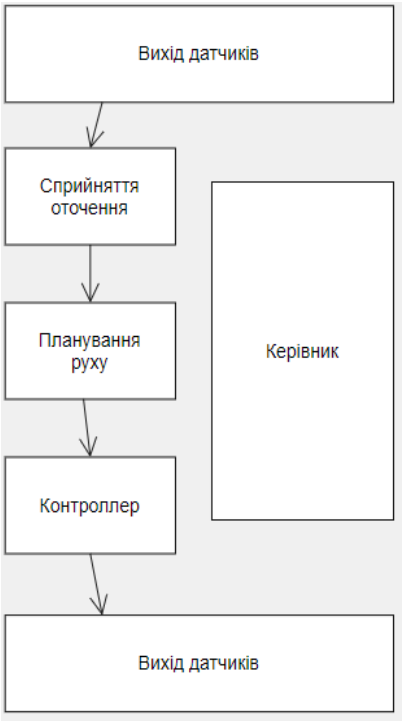

Рисунок 2.16 – Структура системи

Дана модель показує які функції є необхідними для роботи автопілоту автомобіля.

Модулі, які відповідають за роботу з навколишнім середовищем виконують наступні обов’язки:

1) ідентифікація розташування транспорту;
2) розпізнавання елементів навколо під час керування автомобілем.

До елементів навколишнього середовища відносяться такі елементи як:

- автомобілі;
- велосипеди;
- пішоходи;
- дорога;
- дорожня розмітка;
- дорожні знаки.

За допомогою вище перерахованих елементів модуль планування руху може обирати рішення та дії що він буде робити під час руху.

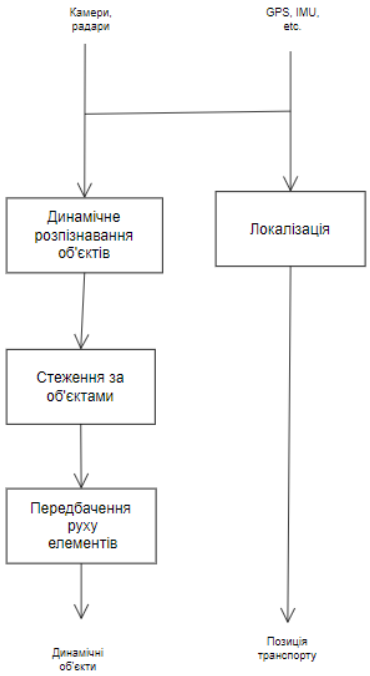

Рисунок 2.17 – Структура модуля, що сприймає навколишнє середовище

Для автопілоту автомобіля планування подальшого руху є одним з складних і важких завдань, що потребує великої кількості обчислень. Обчислення мають бути точними і тому потребують множину процесів. Сучасні автопілоти поділяють весь великий процес на маленькі частини, з котрими легко працювати. Після декомпозиції утворюється множина шарів, кожен з яких виконує свою частину роботи:

1) планувальник поведінки;
2) локальний планувальник.

Планувальник поведінки відповідає за проблеми планування, що займають короткі строки. До його відповідальності входить вибір дій, що не несуть небезпеки для руху автомобіля та які є базовими під час руху на дорозі.

Під час встановлення безпечних для руху автомобіля на дорозі також встановлюється дії, та обмеження, що можуть призвести до подій, що можуть вплинути на ситуацію на дорозі, утворити дорожньо-транспорту пригоду чи просто ускладнити рух на дорозі для інших транспортних засобів.

Частина роботи, що відноситься до відповідальності Локального планувальника поведінки – це вибір дій та рухів автомобіля, які будуть виконані одразу. До цих дій відноситься вибір швидкості, шлях руху автомобіля.

Важливим критерієм є плавність руху, який є безпечним та ефективним, що водночас не перетинає обмеження, що були встановлені планувальником поведінки.

Для забезпечення таких умов уся дата, що надходить до планувальника, комбінується. Після усіх операцій, планувальник видає кінцевий результат – заплановану траєкторію руху, що складається з пріоритетного шляху, швидкості.

Наступний модуль – це Керівник. Даний модуль відповідає за керування транспортного засобу. Даний модуль поділяється на два модулі:

1) модуль, що виконує апаратний нагляд;

2) модуль, що наглядає за апаратним забезпеченням.

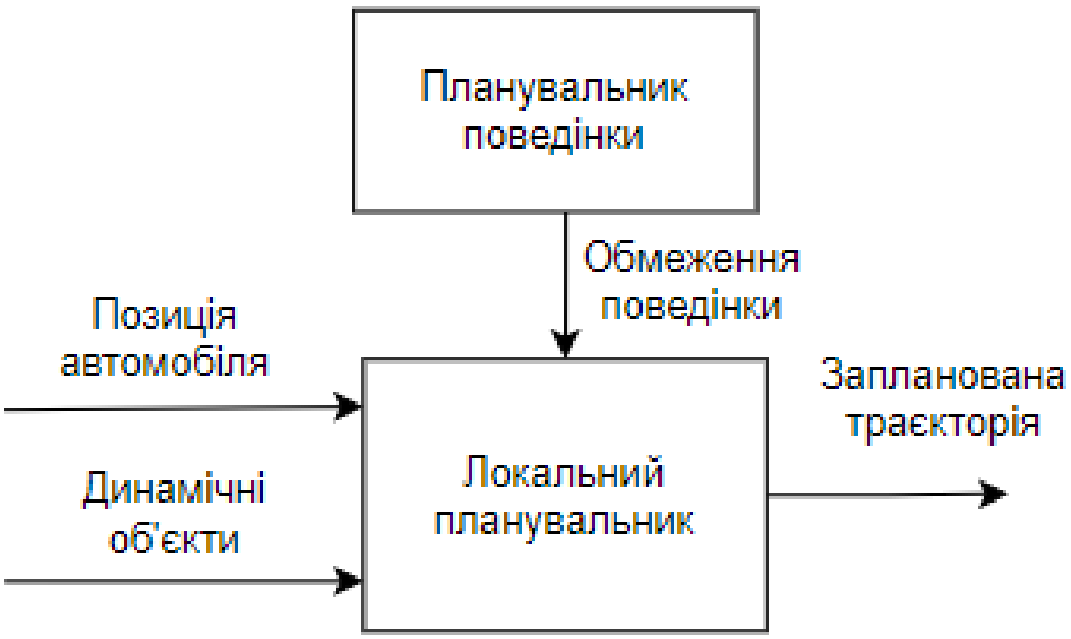

Рисунок 2.17 – Структура модуля, шо планує рух

Керівник, що керує апаратним забезпеченням слідкує за справністю всіх компонентів. До несправностей апаратного забезпечення можуть бути віднесені такі несправності:

- несправний датчик;
- некоректні або відсутні вимірювання;
- інформація поганої якості;

Якщо в якомусь модулі з’явились програмні помилки, що заважають йому виконувати свою роботу, або на нього вплинули погодні умови, якщо це датчик, то керівник програмних забезпеченням повинен зробити аналіз програмного забезпечення і виявити несправність. Керівник програмним забезпеченням несе відповідальність за несправності в програмному забезпеченні.

2.5 Математична модель вузлів автоматичної системи керування транспортним засобом

Лінійна та кутова швидкості є вхідними даними для моделі руху транспортного засобу. Дані показання видає положення транспортного засобу під час руху на дорозі.

$$x_k = x_{k-1} + T\begin{bmatrix} cos\theta_{k-1} & 0 \\ sin\theta_{k-1} & 0 \\ 0 & 1 \end{bmatrix}\left(\begin{bmatrix} v_k \\ \omega_k \end{bmatrix} + w_k\right), w_k = N(0, Q)$$

При оцінці положення транспортного засобу використовується:

- Xk – положення автомобіля в даний час;
- Vk – лінійна швидкість;
- ωk – кутова швидкість.

Для прогнозу руху використовується наступна модель:

$$\check{x}_k = f(\hat{x}_{k-1}, u_{k-1}, 0)$$

$$\check{P}_k = F_{k-1}\hat{P}_{k-1}F_{k-1}^T + L_{k-1}Q_{k-1}L_{k-1}^T$$

$$F_{k-1} = \frac{\partial f}{\partial x_{k-1}}\Big|_{\hat{x}_{k-1}, u_k, 0}, L_{k-1} = \frac{\partial f}{\partial w_k}\Big|_{\hat{x}_{k-1}, u_k, 0}$$

Для того, щоб оцінити керований простій, розрахування координат x, y, z пікселів виконується наступним чином:

$$z = depth$$

$$x = \frac{(u - c_u) * z}{f}$$

$$y = \frac{(v - c_v) * z}{f}$$

Для розрахунки потрібні наступні елементи:

- cu – внутрішні параметри калібрування;
- cv – внутрішні параметри калібрування;
- f – внутрішні параметри калібрування.

Внутрішні параметри калібрування в калібрування камери K

$$K = \begin{pmatrix} f & 0 & u_c \\ 0 & f & u_v \\ 0 & 0 & 1 \end{pmatrix}$$

Рисунок 2.19 – Матриця калібрування камери

2.6 Підбір алгоритму для підбору курсу під час роботи автоматичної системи керування транспортним засобом.

Проблема транспортних засобів, в яких присутні системи автоматичного керування – недостатня стійкість та якість роботи. Під час руху заданою траєкторією, транспортний засіб може трохи відхилятися від курсу і через це автопілот, що вирівнятися назад почне підрулювати. Така поведінка не є бажаною, та може призвести до проблем, можливо і до дорожньо-транспортної пригоди:

$$W_o(p) = \frac{1}{p(0.01p + 1)(0.04p + 1)}.$$

Є список вимог, яким повинен відповідати автопілот транспортного засобу:

- похибка при одиничному впливові має бути нульова;
- коефіцієнт похибки за швидкістю має бути таким як на рисунку ().
- менше перевлаштувань та більше швидкості дії.

$$K_y = \lim_{p \to 0} W_{\text{роз}}(p) = 100.$$

Також для проведення процесу оцінювання глибини, потрібно щоб було виконано наступні дії:

− використовуючи два зображення повинна бути визначення диспропорція;

− за допомогою вхідних даних, треба виконати процес визначення глибини;

− в результаті роботи повинно бути матриця K камери та зовнішня інформація після того, як буде розкладена матриця проекції;

Маючи пару зображень, щоб виконати процес визначення диспропорції між цими зображенням можна використати вже готові рішення для даної задачі, що були реалізована за допомогою мови програмування Python.

Під диспропорцією між зображенням розуміється бінокулярна диспропорція.

Бінокулярна диспропорція відноситься – це різниця розташування об’єкта на зображенні, у правому чи лівому оці людини, коли вона дивиться на даний об’єкт.

Бінокулярна диспропорція використовується для отримання інформації, що стосується глибини, яка присутня в зображенні сітківки.

Бінокулярна диспропорція використовується в комп’ютерному зорі та використовується у двох зображеннях при пошуку різниці координат в подібних ознаках.

Очі людини знаходяться на відстані 5-7 см, у кожної людини по різному. Через це очі можуть дивится на зовнішній світ з невеликими відмінностями. Якщо закрити одне око та подивитись уверх, а потім закрити це око, та відкрити інше і подивитись так само, можна побачити цю різницю між очима.

Це може бути корисно під час далекоміру використовуючи далекомір для збігу щоб визначити відстань чи відстань до потрібної цілі.

Даний термін використовується у геометричних вимірюваннях. Величина розбіжності залежить від сітківки ока, точніше від його ознак.

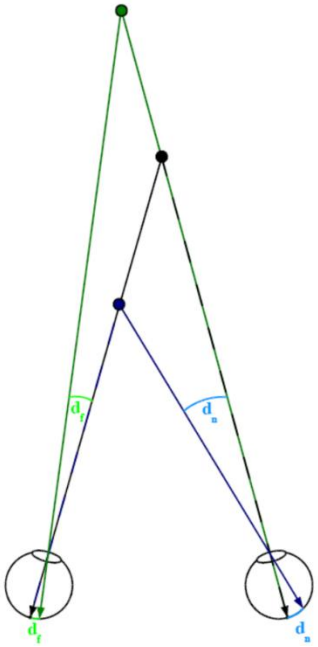

Рисунок 2.20 – Визначення бінокулярної диспропорції

Ліній зору, що мають очі людини можуть пересікатися та утворювати точку зору у просторі. Дана точка знаходиться в одному місці в очах на їх сітківці зору.

Бінокулярна диспропорція визначається у вигляді різниці, що є між такими точками, як точка проекцій двох очей людини і представлена у вигляді кута зору в мірі градусу.

2.7 Висновки за розділом 2

Було розглянуто алгоритми, що використовуються в реалізації автопілоту, категорії нейронних мереж, їх особливості, плюси та мінуси. Було підібрано набір даних, що можуть бути використані для навчання

нейромережі в спеціальному створеному середовищі для проведення навчання нейронної мережі.

Було підібрано алгоритми, що можуть бути корисними для виконання процесу навчання нейронної мережі.

Представлено математичну модель вузлів системи керування транспортним засобом.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ АВТОКЕРУВАННЯ ТРАНСПОРТНИМ ЗАСОБОМ

3.1 Вибір інструментів для реалізації автопілоту

Для реалізації автопілоту для транспортного засобу було обрано мову програмування Python. Дана мова програмування широкі можливості: широкий вибір бібліотек для роботи з нейромережами та немала кількість вбудованих інструментів гарної якості.

Середовищем розробки була обрано IDE PyCharm, що має в наявності всі інструменти для якісної розробки коду додатку.

3.2 Постановка задачі

Було розроблено наступний план:

- розробити автомобільну модель;
- реалізувати та інтегрувати контролер в симулятор CARLA;
- налаштувати роботу контролера;
- виконати оцінку середовища;
- перевірити роботу автопілоту з різними моделями.

При роботі, автомобільна модель приймає дані на основі яких за допомогою рівнянь виконує рух. Коли модель буде реалізована – зробити аналіз роботи даної моделі.

Контролер бере на себе відповідальність за таку річ, як керування транспортного засобу. Він відповідає за рух автомобіля, рішенні зменшити швидкість, підняти, утримувати чи взагалі зупинити автомобіль. Також за допомогою контролера машина може виконувати різні маневри: повороти та розвороти.

В Carla присутні наступні атрибути:

- дросель газу;
- кермо (повороти ліворуч, праворуч)
- швидкість транспортного засобу.

На основі вихідних даних, що отримує контроллер під час роботи, виконати налаштування його роботи з такими атрибутами як дросель, кермо та швидкість.

Налаштувати роботи з вхідними даними та провести аналіз запустивши контролер.

Зробити аналіз середовища, що оточує транспортний засіб та виконати його оцінювання. Також потрібно виконати оцінку стану автомобіля під час руху на дорозі.

3.3 Виконання задачі

Потрібно реалізувати модель транспортного засобу з моделлю нейромережі що згортується. Дана нейромережа, під час своєї роботи, для полегшення процесу обробки даних, що надходять ззовні, зменшує набір інформації, який не відноситься до задачі автопілоту. Якщо системі керування транспортним засобом потрібно виконувати задачу утримування автомобіля вона зменшує кількість інформації, що проходить на дорозі, доки не залишиться набір даних, що необхідний для виконання функціональних обов’язків системи автокерування транспортним засобом.

Структуру нейромережі було реалізовано наступним чином:

- вхідний шар;
- приховані шари;
- вихідний шар;

Вхідними шарами конволюційної нейронної мережі виступають наступні шари:

- Convolutional;

- Subsampling.

Convolutional шар складається з множини карт (карти ознак), кожна карта має своє власне синаптичне ядро (ядро або фільтр). Було вирішено, що оптимальною кількістю карт для нейронної мережі буде 6.

Ядро виконує роботу вікна, що проходиться по всі області попередньої карти та шукає ознаки об’єктів. Було обрано розмір ядра в межах 7x7. Якщо б розмір ядра був маленький – пошук ознак об’єкта буде важче, оскільки в малому вікні буде важко знаходити ознаки, що відносяться до нього. Також може бути так, що це лише шум, який не має ніякого відношення до об’єкту, що може мати значення. Якщо зробити вікно більшим, то можуть виникнути проблеми з занадто великою кількістю зв’язків між нейронами нейронної мережі. Потрібно приділяти належну увагу до розміру вікна, тому що розмір карт, що знаходяться в шарі, що згортається, повинен бути парним. Це дозволяє зменшити кількість інформації, що може бути втрачена під час проходження під час зменшення розміру в підвибірному шарі.

Шар, що має назву Subsampling, займається процесом, в межах якого лежить завдання розмірності мап, що належать попередньому шару.

Перевага використання даного шару в тому, що попередній шар, що займається тим, що вже знайшов певні ознаки, що належать до об’єкта, що потрібно знайти, передається до даного шару, якому для подальшої обробки даних вже не потрібно все велике та детальне зображення. Також використання даного шару дозволяє не перенавчати нейронну мережу, тим самим позбавляє її непотрібних кроків під навчання.

Під час сканування фільтром карти, що належить попередньому фільтру, фільтр не перетинається, на відміну від згорткового шару. В даному випадку використовується мапа, що має ядро в межах 2x2, що надає можливість зменьшити мапи попереднього шару у два рази.

Серед функцій активації був широкий вибір, але після детального розгляду, для даної мережі, було вирішено зробити вибір між такими функціями, як ReLu та гіперболічним тангенсом.

Функція гіперболічний тангенс виступає функцією активацію та використовується шарами, що прихованими та вихідними.

До переваг гіперболічного тангенсу відноситься:

- розрахунок похідної, що не потребує багато ресурсів;
- значення функції лежать в межах в -1 до 1.

Дана функція має наступні недоліки:

- градієнти можуть затухати або збільшуватися
- якщо порівнювати з функцією Relu, то дана функція потребує багато ресурсів.

В свою чергу, функція ReLu використовуються як функція активації в шарах, що згортуються.

До переваг даної функції активації відносяться наступні:

- функція під час розрахунків не потребує великої кількості ресурсів;
- дана функція не використовує не потрібні дані;
- швидко навчається.

Функція також має недоліки. До недоліків даної функції відносяться:

- під час навчання функція іноді показує результатів;
- результати розрахунків даної функції активації залежать значень ваг під час ініціалізації.

Останніми шарами мережи, що згортується – шари звичайного перцептрона, що має багато шарів. Дані шари виконують процес, в якого поставлено завдання класифікації.

Було обрано рішення реалізувати рекурентну нейромережу для реалізації роботи системи автокерування транспортного засобу як другу мережу.

Рекурентна система дещо відрізняється від звичного представлення. Після виконання всіх обчислень, вихід нейронної мережі, що водночас є результатом – становиться входом для наступного кроку навчання нейронної мережі. Таким чином, можна сказати що дана мережа може «пам’ятати».

Ще однією нейронною системою для тестування була обрана нейронна система, що має специфічну структуру LSTM.

Даний різновид нейромережі відрізняється від інших видів нейронних мереж тим, що він, на відміну від інших, здатний до такого процесу, як навчання довготривалих залежностей. Даний вид архітектури для нейронної мережі дуже якісно вирішує велику низку задач та широко використовується.

Дана архітектура нейромережі була сконструйована для уникнення проблеми довготривалих залежностей.

Для архітектури типу LSTM – це стан комірки. Це схоже на стрічку конвеєра. Все проходить по ланцюгу з деякими незначними взаємодіями. Працюючи таким чином, інформації може дуже легко проходити та залишатися незмінною під час роботи. В даному виді архітектури не передбачено можливості видалити чи додати певну інформації до стану клітини, що обережно регулюється через спеціальні структури, що мають назви «ворота». За допомогою даних воріт можна обирати яку інформацію буде пропущено далі. Вони складаються з сигмоїдального нейронного шару та операції початкового множення.

Вихідні значення сигмоїдального шару знаходяться у межах [0;1] та надають інформацію, наскільки багато повинно бути пропущено далі. Якщо значення дорівнює 0 – це означає, що нічого не повинно бути пропущено, якщо значення 1 – то можно пропускати все.

В даному виді нейронної мережі, функцією активації виступає гіперболічний тангенс та три сигмоїди.

3.4 Результати роботи автопілоту. Аналіз досліджень

Було перевірено роботу автопілоту для транспортного засобу під час виконання маневру «поворот праворуч» на перехесті на моделі нейромережі, що згортуєтсья.

Результати роботи троселя зображені на рисунку (Тросель).

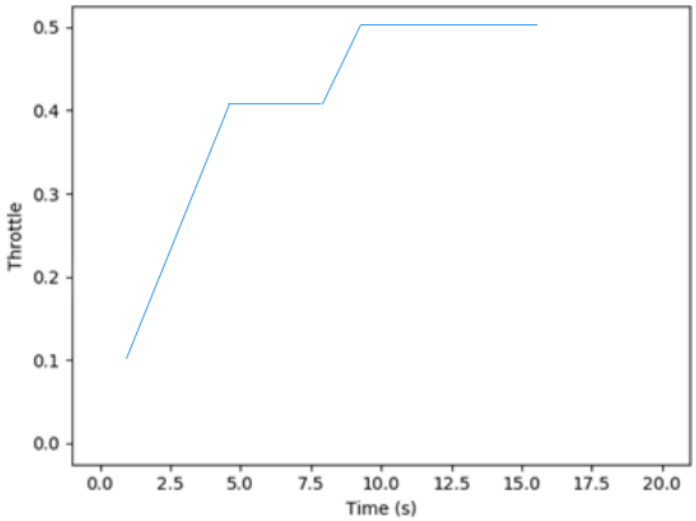

Рисунок 3.1 – Тросель

Автомобіль розпочинає свій рух з дроселем на 10% і дросель з часом збільшується і становиться 50%. Процес повороту транспортного засобу праворуч проходить протягом 12 секунд. Транспортний засіб, після здійснення маневру «поворот праворуч», продовжує свій рух в смузі прямо.

Рисунок 3.2 – Автомобіль перед початком руху

Після початку руху, нейронна мережа, що керує транспортним заособом починає виконувати маневр «поворот праворуч». Під час виконання маневру.

Рисунок 3.3 – Автомобіль під час виконання повороту праворуч

Рисунок 3.4 – Продовження руху автомобіля після повороту праворуч

За допомогою спеціальних камер що передають інформацію в контрастних відтінках автопілот може легко орієнтуватися на дорозі, легко розпізнаючи розмітку на дорозі під час руху. Крім утримання транспортного засобу в смузі, автокеруюча система також виконує контроль швидкості, що є дуже важливим під час виконання руху на дорозі в потоці інших транспортних засобів.

Рисунок 3.5 – Вид з сенсору з переду транспортного засобу

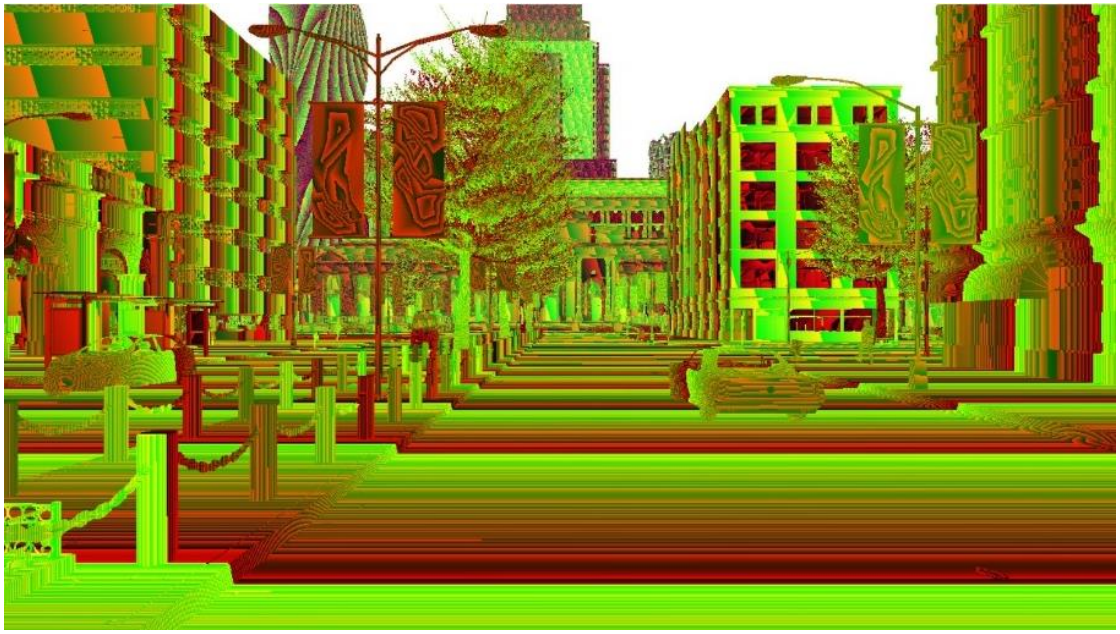

Рисунок 3.6 – Відображення роботи сенсору

Було протестовано роботу рекурентної нейромережі.

Автомобіль розпочинає свій рух з дроселем на 10% і дросель з часом збільшується і становиться 50%. Процес повороту транспортного засобу праворуч проходить протягом 10 секунд. Транспортний засіб, після здійснення маневру «поворот праворуч», продовжує свій рух в смузі прямо.

На рисунку 3.8 зображено транспортний засіб, що починає рух для виконання поставленого завдання «поворот праворуч».

Рисунок 3.8 – Початок маневру «поворот праворуч» RNN

Рисунок 3.9 – Продовження виконання маневру «поворот праворуч»

Після здійснення маневру «повороту праворуч» автомобіль повинен продовжувати рух в смузі, з чим виникають проблеми, оскільки під час завершення повороту автомобіль не зміг вирівнятися одразу по смузі та йому потрібен час, для вирівнювання. Системі потрібно більше часу для виправлення помилок, тому що, якщо дана ситуація трапиться у реальному

житті, через погане навчання системи може виникнути дорожньо-транспортна пригода, що може призвести до жертв.

Рисунок 3.10 – Завершення маневру «поворот праворуч» та продовження руху у смузі

Було досліджено роботу нейромережі з архітектурою LTSM.

Автомобіль розпочинає свій рух з дроселем на 10% і дросель з часом збільшується і становиться 40%. Процес повороту транспортного засобу праворуч проходить протягом 14 секунд. Транспортний засіб, після здійснення маневру «поворот праворуч», продовжує свій рух в смузі прямо.

На рисунку 3.11 зображено роботу дроселя на протязі всього руху транспортного засобу під час виконання маневру.

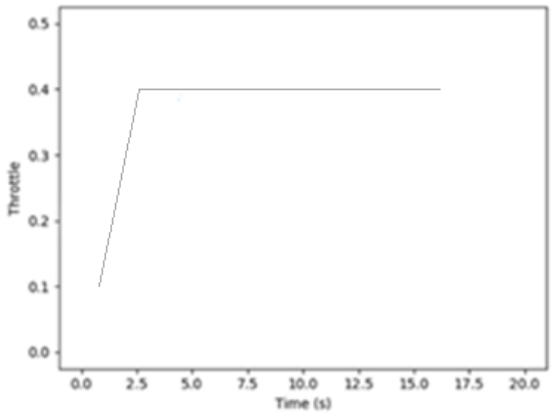

Рисунок 3.11 – Дросель

На рисунку 3.12 зображено початок виконання руху транспортного для виконання маневру «поворот праворуч», що є його завданням.

Рисунок 3.12 – Початок виконання маневру «поворот праворуч» нейромережею LTSM

Рисунок 3.13 – Продовження маневру «поворот праворуч» нейромережею LTSM

Після завершення нейромережею «поворот праворуч» нейромережа повинна далі виконувати утримування транспортного засобу в смузі. Автопілот з даною моделю нейромережі виконує повороти добре, але після виконання маневру може трапитися проблема з вирівнюванням у смузі.

Рисунок 3.14 – Продовження руху по смузі після виконання маневру «поворот праворуч» нейромережею LTSM

На рисунку 3.15 зображення роботу сенсору, що працює в динаміці. За допомогою даного сенсору нейронна мережа, отримує дані, які допомогають їй орієнтуватися під час руху дорогою. Даний сенсор дуже корисний, якщо на дорозі велика кількість об’єктів що рухається.

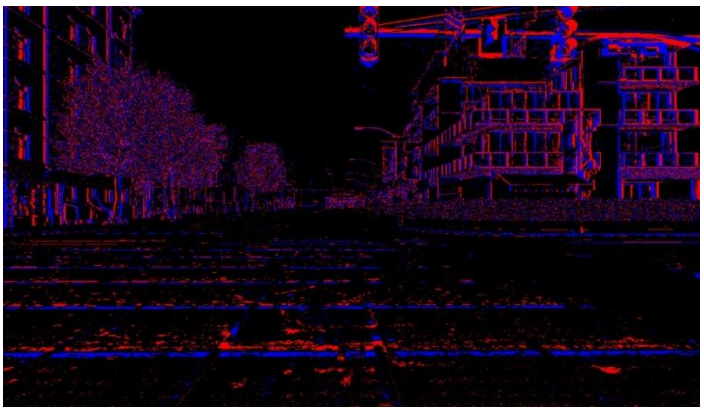

Рисунок 3.15 – Дані з динамічного сенсора

На рисунку 3.16 зображено результати роботи датчику, що зчитує простір навколо автомобіля. У певному діапазоні датчик збирає дані на обробки до системи, що допомгає зчитувати навколишнє середовище.

Рисунок 3.16 – Дані

На рисунку можна побачити, як впродовж руху, нейронна мережа оцінює відстань до об’єктів які можуть бути іншими транспортними засобами, перешкодами і т.д.

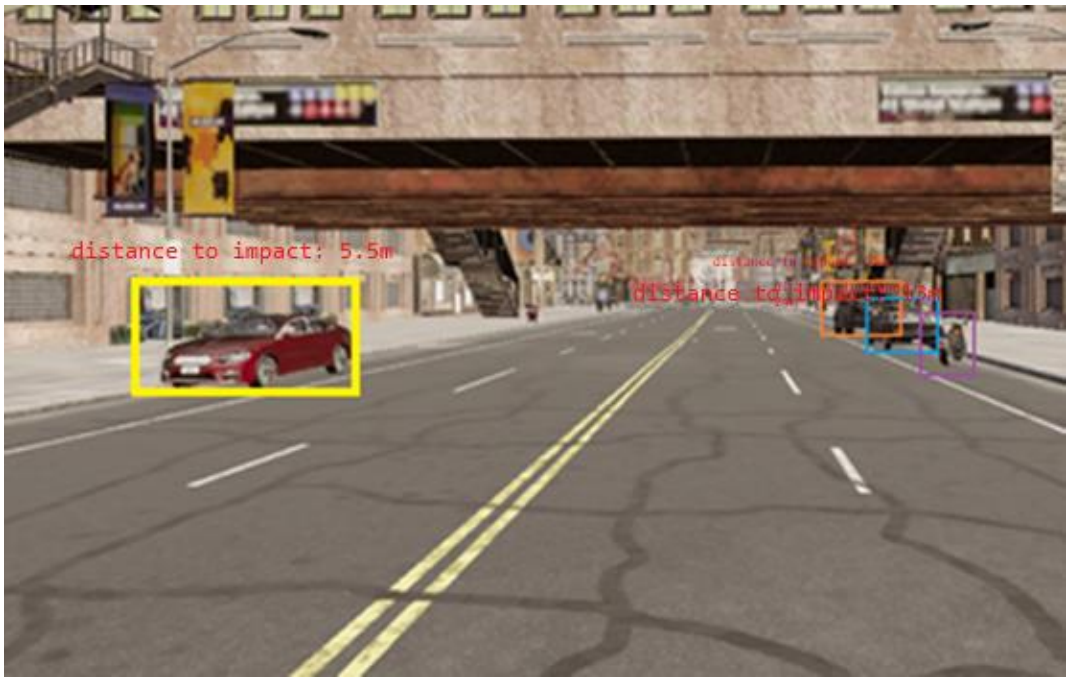

Рисунок 3.17 – Розпізнавання відстані до автомобілів

За допомогою визначення відстані до об’єкта, нейронна мережа може заздалегідь планувати свої дії відносно свого напряму руху.

Було розглянуто види нейронних мереж:

- конволюційна мережа;
- RNN;
- LTSM.

Найкращі результати показала конволюційна мережа. За допомогою зменшення інформації що подається на вхід нейронній мережі легше працювати та шукати ознаки. Під час виконання рухів транспортний засіб засіб вів себе адекватно і цілком зправлявся з поставленими до нього задачами.

Що стосується рекурентної нейронної мережі, то результати виявилися трохи гіршими. Під час руху, під керування даної архітектури нейронної мережі, автомобіль не зовсім добре впорався з поставленими до нього завданнями. При виконанні маневру «поворот праворуч», транспортний засіб перетинав дорожню розмітку, що є краєм дороги. Також автомобіль погано вирівнювася в смузі після завершення маневру. Система потребувала час для вирівнювання у смузі, унаслідок чого утворювалися рвані рухи, що є поганим результатом. Іноді автомобіль перетинав іншу смугу в тому ж напрямі, для вирівнювання.

Можливими шляхами вирішення є:

- змінення структури шарів нейронної мережі;
- зміна функції активації;
- використання двох функцій активації для різних мереж;
- використання іншого алгоритму для роботи нейронної мережі.

Що стосується нейронної мережі, то дана мережа також мала певні проблеми під час керування транспортного засобу на дорозі.

Можливі шляхи покращення роботи нейронної мережі:

- зміна функції, що виконує роль функції активації;
- додавання/видалення шарів в нейронній мережі;
- реалізування іншого алгоритму чи його покращення для даної нейронної мережі.

3.5 Висновки за розділом 3

Було обрано набір інструментів для реалізації автопілоту. Поставлено задачу, яку необхідно виконати.

Описано процесс розробки різних моделей мереж, що були обрані для реалізації завдання автопілоту для транспортного засобу.

Було досліджено роботу різних моделей нейронних мереж, що виконують завдання керування транспортного засобу під час руху на дорозі. Було визначено їх особливості, сильні та слабкі сторони.

Було представлено можливі шляхи покращення поточних реалізованих видів архітектури нейронних мереж.

ВИСНОВКИ

Під час виконання дипломної роботи було виконано наступне:

- Розглянуто види автопілотів, що використовуються на сучасних машинах, їх плюси та мінуси;
- Розглянуто існуючи автомобілі в яких встановлені автопілоти;
- досліджені алгоритми, що використовуються при навчанні нейронних мереж;
- досліджено процес навчання нейромережі;
- обрані інструменти, що використовуються при проєктуванні нейромережі.

Було досліджено методи та технології, які використовуються для розробки моделей нейромережі. Було досліджено як нейронна мережа оброблює об’єкти в реальному часі. Було виконано математичне представлення базових одиниць, що використовуються в нейронних мережах.

Було реалізовано різні види нейронних мереж, що виконують завдання керування транспортного засобу. Досліджено їх сильні та слабкі сторони, розглянуто можливі шляхи покращення ефективності їх роботи.

Результатом виконання дипломної роботи – створені різні види нейронних мереж, що реалізують роботу керування транспортного засобу, в середовищі симулятора Carla. Всі задачі, що були поставлені в дипломній, роботі були виконані.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Automated and Autonomous Driving, Regulation under Uncertainty, Corporate Partnership Board Repor [Електронний ресурс]. – Режим доступу: www.internationaltransportforum.org;
2. Elaine L. Chao. National Highway Traffic Safety Administration [Електронний ресурс]. – Режим доступу: https://www.nhtsa.gov/;
3. Bartels A., Eberle U., Knapp A. Automated Driving Applications and Technologies for Intelligent vehicles [Електронний ресурс]. – Режим доступу: https://www.adaptive-ip.eu/;
4. Bailey N. R., Scerbo M. W. Automation-induced complacency for monitoring highly reliable systems: the role of task complexity, system experience, and operator trust, 2007. 321–348;
5. Beggiato M., Krems J. F. The evolution of mental model, trust and acceptance of adaptive cruise control in relation to initial information. Transportation Research Part F: Traffic Psychology and Behaviour, 2013. 47– 57;
6. Система распознавания дорожных знаков [Електронний ресурс]. – Режим доступу: http://systemsauto.ru/active/traffic_sign_recognition.html;
7. Image Filtering [Електронний ресурс]. – Режим доступу: http://docs.opencv.org/3.0- beta/modules/imgproc/doc/filtering.html;
8. Стаття “Beginner’s Guide to Object Detection Algorithms” [Електронний ресурс]. – Режим доступу: https://medium.com/analytics-vidhya/beginners-guide-to-object-detectionalgorithms-6620fb31c375;
9. Книга “Изучаем OpenCV 3” (Адріан Келер, Гєри Бредскі, 2017 р.);
10. Kyriakidis M., Happee R. J. Public Opinion on Automated Driving: Results of an International Questionnaire Among 5,000 Respondents, 2014. 127-140;

11. Застосування згорткових нейронних мереж для безпеки розпізнавання об'єктів у відеопотоці [Електронний ресурс]. – Режим доступу: https://csecurity.kubg.edu.ua/index.php/journal/article/view/160/154;
12. Використання нейронної мережі для розроблення системи уникнення перешкод на дорозі [Електронний ресурс]. – Режим доступу: https://science.lpnu.ua/sites/default/files/journalpaper/2020/dec/22977/avtomatyka-2020doi-24-33.pdf;
13. Розпізнавання образів та їх класифікація [Електронний ресурс]. – Режим доступу: https://studopedia.com.ua/1_42779_zagalnaharakteristika-zadach-rozpiznavannya-obraziv-ta-matematichna-modelzadachi.html;
14. European Roadmap Smart Systems for Automated Driving. EPoSS. 2015;
15. G. Huang, Z. Liu, and K. Q. Weinberger, Densely connected convolutional networks [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1608.06993v1;
16. Minaee S., Boykov Y., Porikli F., Plaza A., Kehtarnavaz N., Terzopoulos D. Image Segmentation Using Deep Learning. 2020;
17. Нильсен М.А. Нейронні мережі та глибоке навчання. Determination Press, 2017;
18. Pettersson I., Karlsson I. C. M. Setting the stage for autonomous cars: a pilot study of future autonomous driving experiences, 2015. 694–701;
19. Офіційна сторінка Open CV [Електронний ресурс]. – Режим доступу: https://opencv.org/about/;
20. Зображення роботи алгоритму пошуку найближчого сусіда [Електронний ресурс]. – Режим доступу: https://www.pinterest.com/pin/817755244810706876/;
21. Payre W., Cestac J., Delhomme P. Intention to use a fully automated car: Attitudes and a priori acceptability, 2014. 252–263;
22. Lassa, Todd. The Beginning of the End of Driving. Motor Trend. 2014;

23. Geometric Image Transformations [Електронний ресурс]. – Режим доступу:http://docs.opencv.org/3.0/beta/modules/imgproc/doc/geometric_transformations
24. Image Filtering [Електронний ресурс]. – Режим доступу: http://docs.opencv.org/3.0- beta/modules/imgproc/doc/filtering.html;
25. Miscellaneous Image Transformations [Електронний ресурс]. – Режим доступу:http://docs.opencv.org/3.0/beta/modules/imgproc/doc/miscellaneous_transformat;
26. How the backpropagation algorithm works [Електронний ресурс]. – Режим доступу: http://neuralnetworksanddeeplearning.com/chap;
27. F. Chollet, “Keras,” 2015 [Електронний ресурс]. – Режим доступу: https://github.com/fchollet/keras;
28. N. Srivastava, G. E. Hinton et al., Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research [Електронний ресурс]. – Режим доступу: https://www.cs.toronto.edu/~hinton/absps/JMLRdropout.pdf;
29. Understanding Loss Functions in Machine Learning. [Електронний ресурс]. – https://www.section.io/engineering-education/understanding-loss-functions-in;
30. Згорткові нейронні мережі [Електронний ресурс]. – Режим доступу: http://ru.datasides.com/code/cnn-convolutional-neural-networks/;
31. CNN Achitectures [Електронний ресурс]. – Режим доступу: https://medium.com/@RaghavPrabhu/cnn-architectures-lenet-alexnet-vgggooglenet-and-resnet-7c81c017b848;
32. Що таке Python? [Электронный ресурс]. – Режим доступу: http://www.plug.org.ua/documentation/about-python;
33. TensorFlow [Електронний ресурс]. – Режим доступу: https://www.tensorflow.org;
34. Keras [Електронний ресурс]. – Режим доступу: https://keras.io;

35. Zeiler M. Visualizing and Understanding Convolutional Networks / M. Zeiler, R. Fergus. // ICCV. – 2015.
36. Krizhevsky A. Imagenet classification with deep convolutional neural networks / A. Krizhevsky, I. Sutskever, G. E. Hinton. // Advances in neural information processing systems. – 2012.
37. Szegedy C. Deep neural networks for object detection / C. Szegedy, A. Toshev, D. Erhan. // NIPS. – 2013.
38. Zeiler M. Visualizing and Understanding Convolutional Networks / M. Zeiler, R. Fergus. // ICCV. – 2015.
39. Szegedy C. Going deeper with convolutions / C. Szegedy, W. Liu, Y. Jia. // CVPR. – 2015.
40. Новотарський М.А., Нестеренко Б.Б. Штучні нейронні мережі: обчислення [Електронний ресурс]. – Режим доступу: http: //www.immsp .kiev.ua/postgraduate/Bibliotekatrudy/ShtuchnNejronMeregNester2004.pdf
41. Automated and Autonomous Driving, Regulation under Uncertainty, Corporate Partnership Board Report [Електронний ресурс]. – Режим доступу : www.internationaltransportforum.org
42. Згорткова нейронна мережа [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа.
43. Ruder S. An overview of gradient descent optimisation algorithms [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1609.04747
44. Policy Gradient. [Электронный ресурс]. – Режим доступа: http://www.scholarpedia.org/article/Policy_gradient_methods;
45. Online deep learning: learning deep neural networks on the fly [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1711.03705;
46. Maguolo G., Nanni L., Ghidoni S. Ensemble of convolutional neural networks trained with different activation functions [Електронний ресурс]. – Режим доступу: https://arxiv.org/abs/1905.02473;

47. RNN with Keras [Електронний ресурс]. – Режим доступу: https://habr.com/ru/post/487808/;
48. Рекурентні нейронні мережі. Типи, навчання приклади. [Електронний ресурс]. – Режим доступу: https://neurohive.io/ru/osnovy-data-science/rekurrentnye-nejronnye-seti/
49. LTSM – мережі короткострокової пам’яті. [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/wunderfund/blog/331310/;
50. Адаптація нейронної мережі LTSM для рішення комплексної задачі розпізнавання образів [Електронний ресурс]. – Режим доступу: https://cyberleninka.ru/article/n/adaptatsiya-modeli-neyronnoy-seti-lstm-dlya-resheniya-kompleksnoy-zadachi-raspoznavaniya-obrazov/viewer$
51. Carla autosimulator [Електронний ресурс]. – Режим доступу: https://carla.org;

Додаток А
Зауваження нормоконтролера

Таблиця А.1 – Зауваження нормоконтролера

Сторінка	Позначення	Зміст зауваження

Додаток Б

Програмний код

Код файлу CameraClass.py

```
def main():
    """Start function"""
    argparser = argparse.ArgumentParser(
        description='CARLA Sensor sync and projection tutorial')
    argparser.add_argument(
        '--host',
        metavar='H',
        default='127.0.0.1',
        help='IP of the host server (default: 127.0.0.1)')
    argparser.add_argument(
        '-p', '--port',
        metavar='P',
        default=2000,
        type=int,
        help='TCP port to listen to (default: 2000)')
    argparser.add_argument(
        '--res',
        metavar='WIDTHxHEIGHT',
        default='680x420',
        help='window resolution (default: 1280x720)')
    argparser.add_argument(
        '-f', '--frames',
        metavar='N',
        default=500,
        type=int,
        help='number of frames to record (default: 500)')
    argparser.add_argument(
        '-d', '--dot-extent',
        metavar='SIZE',
        default=2,
        type=int,
        help='visualization dot extent in pixels (Recomended [1-4]) (default: 2)')
    argparser.add_argument(
        '--no-noise',
        action='store_true',
        help='remove the drop off and noise from the normal (non-semantic) lidar')
    argparser.add_argument(
        '--upper-fov',
```

```
        metavar='F',
        default=30.0,
        type=float,
        help='lidar\'s upper field of view in degrees (default: 15.0)')
    argparser.add_argument(
        '--lower-fov',
        metavar='F',
        default=-25.0,
        type=float,
        help='lidar\'s lower field of view in degrees (default: -25.0)')
    argparser.add_argument(
        '-c', '--channels',
        metavar='C',
        default=64.0,
        type=float,
        help='lidar\'s channel count (default: 64)')
    argparser.add_argument(
        '-r', '--range',
        metavar='R',
        default=100.0,
        type=float,
        help='lidar\'s maximum range in meters (default: 100.0)')
    argparser.add_argument(
        '--points-per-second',
        metavar='N',
        default='100000',
        type=int,
        help='lidar points per second (default: 100000)')
    args = argparser.parse_args()
    args.width, args.height = [int(x) for x in args.res.split('x')]
    args.dot_extent -= 1

    try:
        tutorial(args)

    except KeyboardInterrupt:
        print('\nCancelled by user. Bye!')


if __name__ == '__main__':

    main()
```

Код файлу Environment:

```
class World(object):
    """ Class representing the surrounding environment """

    def __init__(self, carla_world, hud, args):
        """Constructor method"""
        self._args = args
        self.world = carla_world
        try:
            self.map = self.world.get_map()
        except RuntimeError as error:
            print('RuntimeError: {}'.format(error))
            print('  The server could not send the OpenDRIVE (.xodr) file:')
            print('  Make sure it exists, has the same name of your town, and is correct.')
            sys.exit(1)
        self.hud = hud
        self.player = None
        self.collision_sensor = None
        self.lane_invasion_sensor = None
        self.gnss_sensor = None
        self.camera_manager = None
        self._weather_presets = find_weather_presets()
        self._weather_index = 0
        self._actor_filter = args.filter
        self.restart(args)
        self.world.on_tick(hud.on_world_tick)
        self.recording_enabled = False
        self.recording_start = 0

    def restart(self, args):
        """Restart the world"""
        # Keep same camera config if the camera manager exists.
        cam_index = self.camera_manager.index if self.camera_manager is not None else 0
        cam_pos_id = self.camera_manager.transform_index if self.camera_manager is not None
else 0

        # Get a random blueprint.
        blueprint                                                                    =
random.choice(self.world.get_blueprint_library().filter(self._actor_filter))
        blueprint.set_attribute('role_name', 'hero')
        if blueprint.has_attribute('color'):
            color = random.choice(blueprint.get_attribute('color').recommended_values)
            blueprint.set_attribute('color', color)

        # Spawn the player.
```

```
        if self.player is not None:
            spawn_point = self.player.get_transform()
            spawn_point.location.z += 2.0
            spawn_point.rotation.roll = 0.0
            spawn_point.rotation.pitch = 0.0
            self.destroy()
            self.player = self.world.try_spawn_actor(blueprint, spawn_point)
            self.modify_vehicle_physics(self.player)
        while self.player is None:
            if not self.map.get_spawn_points():
                print('There are no spawn points available in your map/town.')
                print('Please add some Vehicle Spawn Point to your UE4 scene.')
                sys.exit(1)
            spawn_points = self.map.get_spawn_points()
            spawn_point = random.choice(spawn_points) if spawn_points else carla.Transform()
            self.player = self.world.try_spawn_actor(blueprint, spawn_point)
            self.modify_vehicle_physics(self.player)

        if self._args.sync:
            self.world.tick()
        else:
            self.world.wait_for_tick()

        # Set up the sensors.
        self.collision_sensor = CollisionSensor(self.player, self.hud)
        self.lane_invasion_sensor = LaneInvasionSensor(self.player, self.hud)
        self.gnss_sensor = GnssSensor(self.player)
        self.camera_manager = CameraManager(self.player, self.hud)
        self.camera_manager.transform_index = cam_pos_id
        self.camera_manager.set_sensor(cam_index, notify=False)
        actor_type = get_actor_display_name(self.player)
        self.hud.notification(actor_type)

    def tick(self, clock):
        """Method for every tick"""
        self.hud.tick(self, clock)

    def render(self, display):
        """Render world"""
        self.camera_manager.render(display)
        self.hud.render(display)

    def destroy_sensors(self):
        """Destroy sensors"""
        self.camera_manager.sensor.destroy()
```

```
        self.camera_manager.sensor = None
        self.camera_manager.index = None

    def destroy(self):
        """Destroys all actors"""
        actors = [
            self.camera_manager.sensor,
            self.collision_sensor.sensor,
            self.lane_invasion_sensor.sensor,
            self.gnss_sensor.sensor,
            self.player]
        for actor in actors:
            if actor is not None:
                actor.destroy()
```

Код файлу Movement_Control:

```
class KeyboardControl(object):
    def __init__(self, world):
        world.hud.notification("Press 'H' or '?' for help.", seconds=4.0)

    def parse_events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                return True
            if event.type == pygame.KEYUP:
                if self._is_quit_shortcut(event.key):
                    return True

    @staticmethod
    def _is_quit_shortcut(key):
        """Shortcut for quitting"""
        return (key == K_ESCAPE) or (key == K_q and pygame.key.get_mods() & KMOD_CTRL)

class HUD(object):
    """Class for HUD text"""

    def __init__(self, width, height):
        """Constructor method"""
        self.dim = (width, height)
        font = pygame.font.Font(pygame.font.get_default_font(), 20)
        font_name = 'courier' if os.name == 'nt' else 'mono'
        fonts = [x for x in pygame.font.get_fonts() if font_name in x]
        default_font = 'ubuntumono'
        mono = default_font if default_font in fonts else fonts[0]
        mono = pygame.font.match_font(mono)
```

```
        self._font_mono = pygame.font.Font(mono, 12 if os.name == 'nt' else 14)
        self._notifications = FadingText(font, (width, 40), (0, height - 40))
        self.help = HelpText(pygame.font.Font(mono, 24), width, height)
        self.server_fps = 0
        self.frame = 0
        self.simulation_time = 0
        self._show_info = True
        self._info_text = []
        self._server_clock = pygame.time.Clock()

    def on_world_tick(self, timestamp):
        """Gets informations from the world at every tick"""
        self._server_clock.tick()
        self.server_fps = self._server_clock.get_fps()
        self.frame = timestamp.frame_count
        self.simulation_time = timestamp.elapsed_seconds
```

Код файлу ltsm_nn:

```
        import numpy as np
        import tensorflow as tf
        from tensorflow import keras
        from tensorflow.keras import layers
LSTM_model = keras.Sequential()
        LSTM_model.add(layers.Embedding(input_dim=1000, output_dim=64))
        LSTM_model.add(layers.LSTM(128))
        LSTM_model.add(layers.Dense(10))
        # reshape to [1, n_input]
            x = tf.reshape(x, [-1, n_input])
        vocab_size = len(dictionary)
        n_input = 3
        # number of units in RNN cell
        n_hidden = 512
        # RNN output node weights and biases
        weights = {
            'out': tf.Variable(tf.random_normal([n_hidden, vocab_size]))
        }
        biases = {
            'out': tf.Variable(tf.random_normal([vocab_size]))
        }


            # generate prediction
            outputs, states = rnn.static_rnn(rnn_cell, x, dtype=tf.float32)
            # Generate a n_input-element sequence of inputs
            # (eg. [had] [a] [general] -> [20] [6] [33])
            x = tf.split(x,n_input,1)

            # 1-layer LSTM with n_hidden units.
```

```
    rnn_cell = rnn.BasicLSTMCell(n_hidden)

    # there are n_input outputs but
    # we only want the last output
    return tf.matmul(outputs[-1], weights['out']) + biases['out']
```

Код файлу rnn_nn:

```
class Rnn(object):
def __init__(self, embedding_mat, non_static, hidden_unit, sequence_length, max_pool_size,num_classes, embedding_size, filter_sizes, num_filters, l2_reg_lambda=0.0):
        self.input_x = tf.placeholder(tf.int32, [None, sequence_length], name='input_x')
        self.input_y = tf.placeholder(tf.float32, [None, num_classes], name='input_y')
        self.dropout_keep_prob = tf.placeholder(tf.float32, name='dropout_keep_prob')
        self.batch_size = tf.placeholder(tf.int32, [])
        self.pad = tf.placeholder(tf.float32, [None, 1, embedding_size, 1], name='pad')
        self.real_len = tf.placeholder(tf.int32, [None], name='real_len')

l2_loss = tf.constant(0.0)

with tf.device('/cpu:0'), tf.name_scope('embedding'):
        if not non_static:
                W = tf.constant(embedding_mat, name='W')
        else:
                W = tf.Variable(embedding_mat, name='W')
        self.embedded_chars = tf.nn.embedding_lookup(W, self.input_x)
        emb = tf.expand_dims(self.embedded_chars, -1)

pooled_concat = []
reduced = np.int32(np.ceil((sequence_length) * 1.0 / max_pool_size))

for i, filter_size in enumerate(filter_sizes):
        with tf.name_scope('conv-maxpool-%s' % filter_size):
                sequence_length x emb_size x channel
                num_prio = (filter_size-1) // 2
                num_post = (filter_size-1) - num_prio
                pad_prio = tf.concat([self.pad] * num_prio,1)
                pad_post = tf.concat([self.pad] * num_post,1)
                emb_pad = tf.concat([pad_prio, emb, pad_post],1)

                filter_shape = [filter_size, embedding_size, 1, num_filters]
                W = tf.Variable(tf.truncated_normal(filter_shape, stddev=0.1), name='W')
                b = tf.Variable(tf.constant(0.1, shape=[num_filters]), name='b')
                conv = tf.nn.conv2d(emb_pad, W, strides=[1, 1, 1, 1], padding='VALID', name='conv')
```

```
                    h = tf.nn.relu(tf.nn.bias_add(conv, b), name='relu')
                    pooled = tf.nn.max_pool(h, ksize=[1, max_pool_size, 1, 1], strides=[1,
max_pool_size, 1, 1], padding='SAME', name='pool')
                    pooled = tf.reshape(pooled, [-1, reduced, num_filters])
      pooled_concat.append(pooled)
             pooled_concat = tf.concat(pooled_concat,2)
             pooled_concat = tf.nn.dropout(pooled_concat, self.dropout_keep_prob)
             lstm_cell = tf.contrib.rnn.GRUCell(num_units=hidden_unit)
             #lstm_cell=tf.nn.rnn_cell.DropoutWrapper(lstm_cell,
output_keep_prob=self.dropout_keep_prob)
             lstm_cell=tf.contrib.rnn.DropoutWrapper(lstm_cell,
output_keep_prob=self.dropout_keep_prob)
             self._initial_state = lstm_cell.zero_state(self.batch_size, tf.float32)
             inputs=[tf.squeeze(input_,[1])for                    input_                    in
tf.split(pooled_concat,num_or_size_splits=int(reduced),axis=1)]
```

Код файлу conv_nn:

```
      model = models.Sequential()
      model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32,
3)))
      model.add(layers.MaxPooling2D((2, 2)))
      model.add(layers.Conv2D(64, (3, 3), activation='relu'))
      model.add(layers.MaxPooling2D((2, 2)))
      model.add(layers.Conv2D(64, (3, 3), activation='relu'))
      model = models.Sequential()
      model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32,
3)))
      model.add(layers.MaxPooling2D((2, 2)))
      model.add(layers.Conv2D(64, (3, 3), activation='relu'))
      model.add(layers.MaxPooling2D((2, 2)))
      model.add(layers.Conv2D(64, (3, 3), activation='relu'))
      model.compile(optimizer='adam',
        loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                    metrics=['accuracy'])
      for i, filter_size in enumerate(filter_sizes):
             with tf.name_scope('conv-maxpool-%s' % filter_size):
                    sequence_length x emb_size x channel
                    num_prio = (filter_size-1) // 2
                    num_post = (filter_size-1) - num_prio
                    pad_prio = tf.concat([self.pad] * num_prio,1)
                    pad_post = tf.concat([self.pad] * num_post,1)
                    emb_pad = tf.concat([pad_prio, emb, pad_post],1)
```

```
                    filter_shape = [filter_size, embedding_size, 1, num_filters]
                    W = tf.Variable(tf.truncated_normal(filter_shape, stddev=0.1), name='W')
                    b = tf.Variable(tf.constant(0.1, shape=[num_filters]), name='b')
                    conv = tf.nn.conv2d(emb_pad, W, strides=[1, 1, 1, 1], padding='VALID',
name='conv')

                    h = tf.nn.relu(tf.nn.bias_add(conv, b), name='relu')
                    pooled = tf.nn.max_pool(h, ksize=[1, max_pool_size, 1, 1], strides=[1,
max_pool_size, 1, 1], padding='SAME', name='pool')
                    pooled = tf.reshape(pooled, [-1, reduced, num_filters])
```