Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Системотехніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Розроблення та дослідження нейронної мережі для розпізнавання дорожніх знаків

(тема)

Виконав:
студент 2 курсу, групи ІТПм-22-1
Слатін М.Ю.
(прізвище, ініціали)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Освітня програма Інформаційні технології проектування
(повна назва освітньої програми)
Керівник проф. Калита Н.І.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри СТ _____ Гребеннік І.В
(підпис) (прізвище, ініціали)

2024 р.

Я як студент ХНУРЕ розумію і підтримую політику закладу із академічної доброчесності. Я не надавав і не одержував недозволену допомогу під час підготовки кваліфікаційної роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

13.01.2023 *Слатін М.Ю.* 

Кваліфікаційна робота не містить відомостей заборонених до відкритого опублікування

Керівник кваліфікаційної роботи 

Кваліфікаційна робота виконана у відповідності до стандартів, що діють в Україні

Керівник кваліфікаційної роботи 

Попередній захист проведено *13.01.2024*

Керівник кваліфікаційної роботи 

Харківський національний університет радіоелектроніки

Факультет_____*Комп'ютерних наук*_____

Кафедра_____ _____*Системотехніки*_____

Рівень вищої освіти_____другий (магістерський)_____

Спеціальність_____122 *Комп'ютерні науки*_____
(код і повна назва)

Тип програми _____освітньо-професійна_____

Освітня програма____*Інформаційні технології проектування* _____
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«___» _____20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові _____*Слатіну Микиті Юрійовичу*_____
(прізвище, ім'я, по батькові)

1. Тема роботи: *Розроблення та дослідження нейронної мережі для розпізнавання дорожніх знаків*
затверджена наказом по університету від ___*20.11*___ 2023 р. №___*1373Ст*___
2. Термін подання студентом роботи до екзаменаційної комісії: 15.01.2024 р
3.Вихідні дані до роботи: *Література із комп'ютерного зору, розпізнавання образів, машинного навчання. Статті про навчання нейронних мереж, існуючи рішення засобів розпізнавання дорожніх знаків, існуючі нейронні мережі. Існуючи підходи для розроблення веб додатків. Статті про вплив перешкод на роботу систем розпізнавання дорожніх знаків. Нейронні мережі VGG-19 та LeNet-5*

4. Перелік питань, що потрібно опрацювати в роботі: *Аналіз предметної області та постановка задачі. Огляд існуючих систем розпізнавання дорожніх знаків. Аналіз досліджуваних технологій в існуючих системах. Обґрунтування вибору технологій проектування, середовища та існуючих нейронних мереж для порівняння. Створення авторської нейронної мережі. Тестування нейронних мереж у порівнянні між собою. Створення вебзастосунку.*

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій: *Архітектура нейронної мережі 13, Модель нейронної мережі VGG-19, Модель нейронної мережі LeNet-5, Кількість класів та зображень у наборі GTRSB,*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Отримання завдання*	*16.10.2023*	
2	*Постановка задачі та узгодження з керівником*	*18-19.10.2023*	
3	*Опис та аналіз предметної області*	*22-25.10.2023*	
4	*Аналіз існуючих систем розпізнавання дорожніх знаків*	*25-31.10.2023*	
5	*Постановка задачі*	*03-05.11.2023*	
6	*Огляд літератури*	*05-07.11.2023*	
7	*Огляд критеріїв для створення авторської нейронної мережі розпізнавання на основі ШІ*	*07-10.11.2023*	
8	*Математичний опис задач*	*10-15.11.2023*	
9	*Проектування нейронних мереж*	*15-30.11.2023*	
10	*Розробка вебзастосунку*	*01-20.12.2023*	
11	*Оформлення пояснювальної записки*	*20-30.12.2023*	
12	*Попередній захист*	*13.01.2024*	
13	*Подання роботи в ЕК*	*15.01.2024*	

Дата видачі завдання 16.10.2023 р.

Студент _____
(підпис)

Керівник роботи _____ *проф. Калита Н.І.*
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Кваліфікаційна робота: 85 с., 1 табл., 42 рис., 2 додатки, 26 джерел інформації

ВЕБЗАСТОСУНОК, РОЗПІЗНАВАННЯ ДОРОЖНІХ ЗНАКІВ, РОЗПІЗНАВАННЯ ОБРАЗІВ, КОМП`ЮТЕРНЕ БАЧЕННЯ, МАШИННЕ НАВЧАННЯ, НЕЙРОННА МЕРЕЖА, NUMPY, MATPLOTLIB, SCIKIT-LEARN, PANDAS, TENSORFLOW, KERAS, OPENCV.

Об’єктом дослідження кваліфікаційної роботи є розробка авторської нейронної мережі з інтеграцією у вебзастосунок, подальше порівняння продуктивності з мережами VGG-19 та LeNet-5.

Предметом дослідження є архітектурні особливості існуючих нейронних мереж та їх здатність розпізнавання образів, порівняння їх продуктивності в з авторською мережею. Можливість інтеграції у вебзастосунок.

Метою кваліфікаційної роботи є розробка та дослідження ефективності нейронних мереж, таких як, VGG-19, LeNet-5 та 13, які використовуються у веб застосунку для розпізнавання дорожніх знаків водіями під час руху.

Методи дослідження – системний аналіз, теорія розпізнавання образів, машинне навчання, мова програмування Python, бібліотеки Keras, NumPy, TensorFlow, Matplotlib, Scikit-learn, Pandas, OpenCV. Методи машинного навчання, розпізнавання образів та сегментації зображення.

Робота включає теоретичний аналіз досліджуваних технологій в існуючих системах допомоги водієві на дорозі. Розробку власної нейронної мережі для розпізнавання дорожніх знаків, створення штучних перешкод у тестовій вибірці та включення у досліди двох популярних нейронних мереж для порівняння ефективності системи. Також передбачено створення вебзастосунку для візуалізації процесів розпізнавання образів.

Використання даного вебзастосунку має обмеження лише у ступені навченості моделей нейронних мереж, та обсягом набора даних, на яких вони навчались.

Сфера застосування – дослідницька робота з виявлення впливу певних перешкод на результат роботи розпізнавання нейронних мереж, практичне застосування для розробників ПЗ та навчальних цілей.

ABSTRACT

Master’s Thesis: 85 p., 1 tab., 42 fig., 2 appendices, 26 title.

COMPUTER VISION, DATASET, DEEP LEARNING, KERAS, MACHINE LEARNING, MATPLOTLIB, NEURAL NETWORK, NOISE, NUMPY, OPENCV, PANDAS, PATTERN RECOGNITION THEORY, ROAD SIGN RECOGNITION, SCIKIT-LEARN, TENSORFLOW, WEB APPLICATION.

The object of research of the qualification work is the development of an author's neural network with integration into a web application

The subject of the research is the architectural features of existing neural networks and their ability to recognize patterns, comparing their performance with the author's network. The possibility of integration into a web application.

The purpose of the qualification work is to develop and study the effectiveness of neural networks, such as VGG-19, LeNet-5 and 13, which are used in a web application for recognizing road signs by drivers while driving.

Research methods - system analysis, pattern recognition theory, machine learning, Python programming language, Keras, NumPy, TensorFlow, Matplotlib, Scikit-learn, Pandas, OpenCV libraries. Machine learning, pattern recognition, and image segmentation methods.

The work includes a theoretical analysis of the studied technologies in existing driver assistance systems on the road. Development of a proprietary neural network for road sign recognition, creation of artificial obstacles in the test sample, and inclusion of two popular neural networks in the experiments to compare the system's efficiency. It is also envisaged to create a web application for visualizing the process of pattern recognition.

The use of this web application has limitations only in the degree of training of neural network models and the size of the data set on which they were trained.

The scope of the application is research work to identify the impact of certain interferences on the result of neural network recognition, practical application for software developers and educational purposes.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МН – Машинне навчання;

ПЗ – Програмне забезпечення;

РО – Розпізнавання образів;

ШІ – Штучний інтелект;

ШНМ – Штучна нейронна мережа;

ADAS – Advanced driving assistance systems;

CV – Computer Vision;

CNN – Convolutional Neural Network;

DNN – Deep Neural Network. ;

MVC – Model View Controller.

NLP – Neuro-Linguistic Programming;

NN – Neural Network;

RNN – Recurrent Neural Network;

RSI – Road Sign Information;

TSR – traffic-sign recognition;

ВСТУП

Сучасний стан комп’ютерних технологій дозволяє нам мінімізувати вплив зовнішніх факторів на роботу системи з штучним інтелектом. Комп'ютерні технології, зокрема нейронні мережі, особливо в галузі комп'ютерного зору, надзвичайно важливі для сучасного суспільства. Вони трансформують наше життя, надаючи невід'ємність у різних сферах. На сьогоднішній день існує значна кількість високопотужних обчислювальних систем, і з часом цей тренд буде лише зростати. Розвиток їх потужностей визначатиме напрямок подальших технологічних досягнень, що сприятиме прогресу в усіх сферах життя.

Комп'ютерний зір стає ключовим напрямком в дослідженнях та розвитку комп'ютерних наук. Від автономних автомобілів, які розпізнають дорожні знаки, до медичних систем, що аналізують зображення для точної діагностики, ці технології революціонізують різні галузі.

Ці технології роблять наше життя зручнішим, безпечнішим і допомагають розв'язувати завдання, які раніше вважалися непередбачуваними. Комп'ютерний зір, завдяки нейронним мережам, відкриває нові перспективи для подальшого розвитку та прогресу в усіх сферах нашого життя. Програмне забезпечення на основі комп'ютерного зору вже успішно інтегроване в багато аспектів нашого повсякденного життя. Воно допомагає вирішувати різноманітні завдання і виконувати складні завдання, які за допомогою альтернативних технологій були б складними або дорогими. Одним з таких важливих завдань є розпізнавання дорожніх знаків. Системи, які вирішують цю задачу, призначені для автоматизації та підвищення комфорту водія і безпеки дорожнього руху, а також можуть використовуватися для навчання водіїв.

Наявність візуальних перешкод, таких як артефакти, шум і розмиті зображення, може суттєво впливати на розпізнавання зображень нейронними мережами. Ці фактори призводять до зниження точності та надійності результатів. Артефакти та шум призводять до втрати важливих деталей, а розмиті

зображення ускладнюють ідентифікацію ознак об'єктів. Нейронні мережі, оптимізовані для таких шумів, можуть потребувати додаткового навчання на релевантних даних або використання методів доповнення даних для підвищення надійності в реальних умовах.

Об’єктом дослідження кваліфікаційної роботи є розробка авторської нейронної мережі з інтеграцією у вебзастосунок, подальше порівняння продуктивності з мережами VGG-19 та LeNet-5.

Предметом дослідження є архітектурні особливості існуючих нейронних мереж та їх здатність розпізнавання образів, порівняння їх продуктивності в з авторською мережею. Можливість інтеграції у вебзастосунок.

Метою кваліфікаційної роботи є розробка та дослідження ефективності нейронних мереж, таких як, VGG-19, LeNet-5 та 13, які використовуються у веб застосунку для розпізнавання дорожніх знаків водіями під час руху.

.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ ДОСЛІДЖЕННЯ

1.1 Аналіз предметної області

1.1.1 Актуальність завдання

Комп'ютерний зір - це галузь штучного інтелекту (ШІ), яка дозволяє комп'ютерам і системам отримувати значущу інформацію з цифрових зображень, відео та інших візуальних даних - і виконувати дії або давати рекомендації на основі цієї інформації. Якщо ШІ дозволяє комп'ютерам мислити, то комп'ютерний зір дозволяє їм бачити, спостерігати і розуміти [1].

Тема комп'ютерного зору з використанням нейронних мереж є надзвичайно актуальною та важливою в сучасному світі. Стрімкий розвиток цієї галузі має значний вплив на багато аспектів нашого життя і показує великий потенціал для технічного прогресу.

У медицині комп'ютерний зір на базі нейронних мереж широко використовується для аналізу медичних зображень. Наприклад, в діагностиці раку він допомагає розпізнавати аномалії на зображеннях мамографії. Комп'ютерний зір може ефективно підтримувати будь-яке медичне завдання, яке потребує тренованого ока, щоб розпізнати і класифікувати проблему зі здоров'ям [2]. Ця технологія на основі штучного інтелекту використовує передові алгоритми для обробки зображень і зчитує їх у режимі реального часу, визначаючи конкретні ознаки хвороби. Правильне використання комп'ютерного зору в медицині допоможе скоротити час, витрачений на непотрібні діагностичні процедури, і наддасть медичним працівникам засоби для постановки більш точних діагнозів і призначення більш ефективного лікування. На рисунку 1.1 та рисунку 1.2 ШІ проводить аналіз зображення легень та мозку пацієнта з метою виявлення наявності захворювань [2].

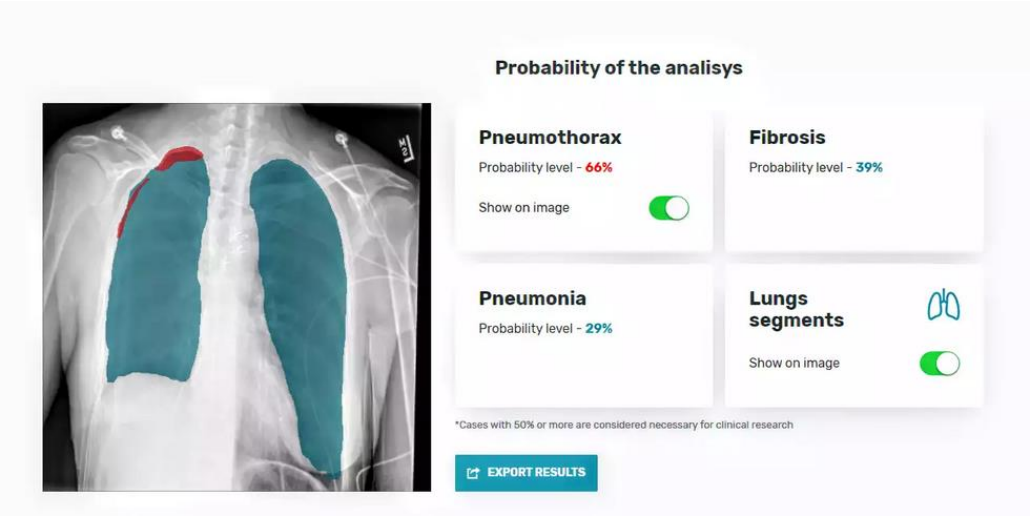

Рисунок 1.1 – Інтерфейс інструменту діагностики на основі ШІ.

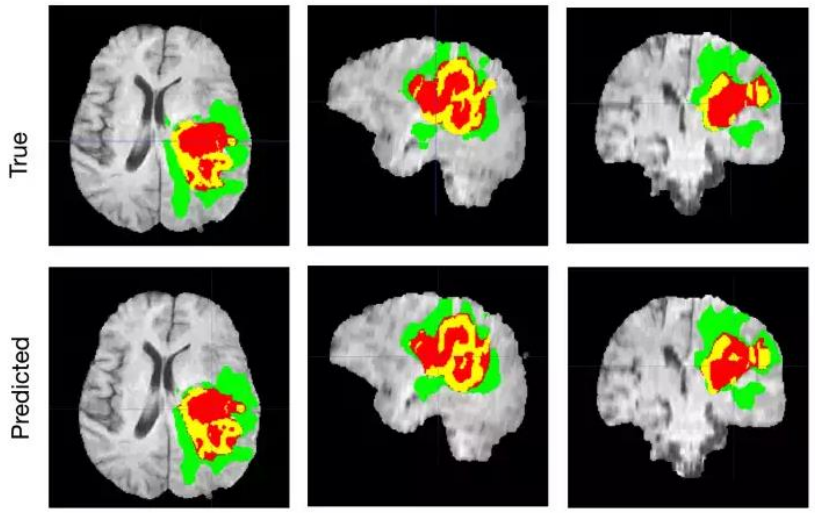

Рисунок 1.2 – Сегментація пухлини мозку за допомогою ШІ.

На сьогоднішній день комп'ютерний зір розвинувся настільки, що може охоплювати широкий спектр завдань і допомагати людям у виконанні багатьох видів медичної діяльності. У свою чергу, ці розробки принесли цілий ряд переваг, якими може користуватися все більша кількість фахівців [2].

У сфері транспорту системи розпізнавання дорожніх знаків та перешкод

стають основою для автономних транспортних засобів, допомагаючи розпізнавати дорожні знаки і забезпечувати безпечне водіння. Виробники транспортних засобів активно використовують комп'ютерний зір для покращення функціональності та безпеки автомобілів. Приклади включають Tesla Autopilot для автономного водіння, BMW Driving Assistance для допомоги на дорозі, Ford Active Park Assist для автоматичного паркування, Mercedes-Benz Driver Assist Package для розпізнавання дорожніх знаків і Volvo City Safety для уникнення зіткнень [3]. Ці програми комп'ютерного зору розвивають технологію автономного водіння і підвищують безпеку та зручність для водіїв (рисунок 1.3).

Рисунок 1.3 – Що бачить Tesla Autopilot через сенсори

У сфері безпеки системи відеоспостереження на основі нейронних мереж дають змогу ефективніше аналізувати та виявляти небезпечні ситуації. У виробництві вони допомагають контролювати якість продукції та оптимізувати виробничі процеси.

Відеоаналіз ШІ використовується для виявлення аномалій у дорожньому русі, метро, поїздах, суднах, будівлях і громадських місцях. Приклади

моніторингу відеоспостереження для виявлення аномалій у візуальному ШІ включають виявлення зупинених транспортних засобів (рисунок 1.4), виявлення паніки, виявлення вторгнення або розпізнавання аномальної активності пішоходів [3].

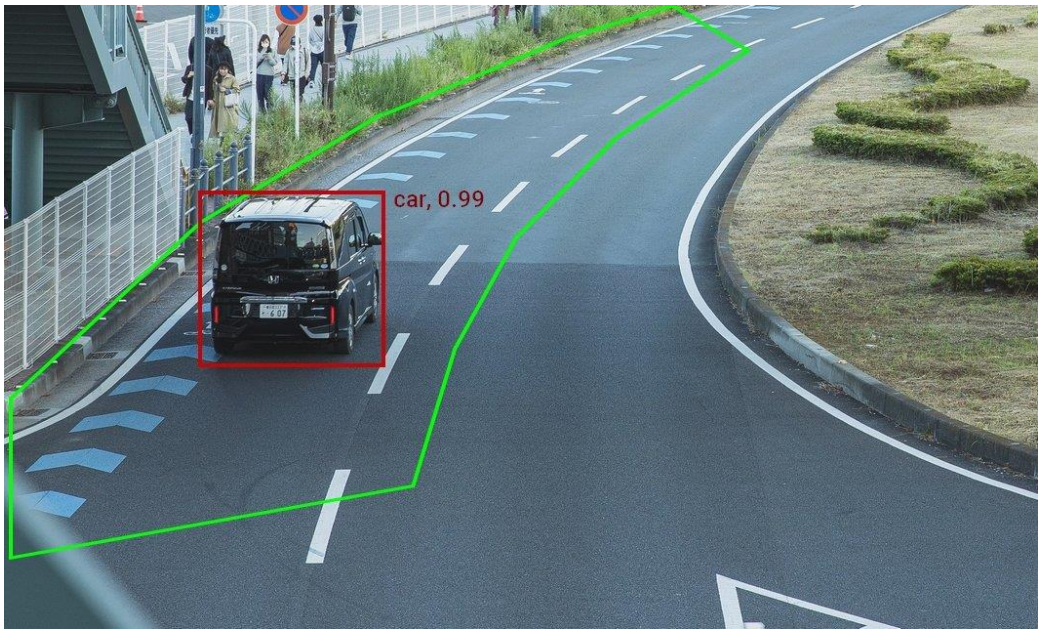

Рисунок 1.4 – Виявлення аномалій за допомогою глибокого навчання для виявлення зупинених транспортних засобів

У програмах виявлення та розпізнавання облич комп'ютерний зір відіграє важливу роль. Розпізнавання облич - це технологія, яка дозволяє комп'ютерам і машинам зіставляти зображення, що містять обличчя людей, з їхніми особистими даними. У цьому розділі алгоритми комп'ютерного зору виявляють риси обличчя на зображеннях. Після цього вони порівнюють їх у різних базах даних [4].

Ця технологія дозволяє виокремлювати ключові ознаки, такі як положення очей, носа та рота, і створювати унікальний біометричний підпис для ідентифікації. Нейронні мережі навчаються на великих масивах даних, щоб оптимізувати ідентифікацію та розпізнавання шаблонів обличчя.

Однією з переваг цього підходу є можливість працювати в режимі

реального часу, що дозволяє ефективно використовувати розпізнавання облич у різних галузях, включаючи безпеку, автоматизацію та індустрію розваг.

Технологія також є дуже гнучкою, оскільки може адаптуватися до різних умов освітлення, різних ракурсів і локацій. Використання розпізнавання облич на основі нейронних мереж відкриває можливості у сферах безпеки, автентифікації та персоналізації, що робить цю технологію цікавою та перспективною.

На рисунку 1.5 зображено один з методів розпізнавання обличчя. метод дає змогу одночасно виявити обличчя, локалізувати орієнтири, оцінити позу і розпізнати стать. Сині квадрати позначають виявлені чоловічі обличчя, а рожеві - жіночі. Зелені точки вказують місце розташування орієнтирів. Оцінки пози для кожної особи показані поверх осередків у порядку крену, тангажу і рискання [4].

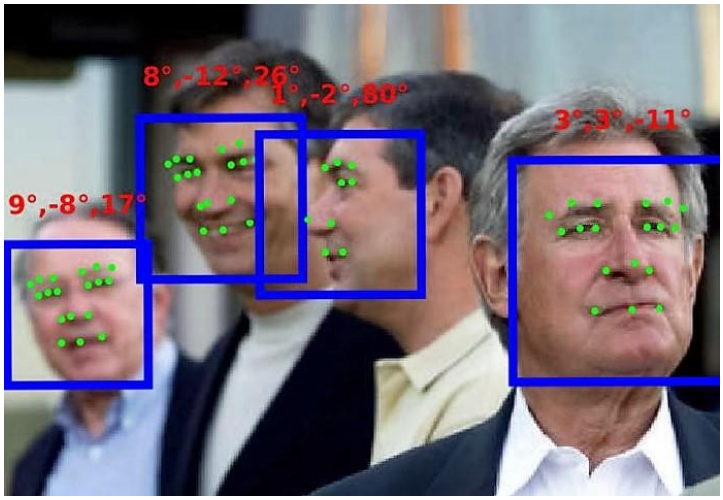

Рисунок 1.5 – Приклад розпізнавання обличчя з допомогою комп’ютерного зору

Використання нейронних мереж у комп'ютерному зорі є ключовим елементом у розвитку інтелектуальних систем та штучного інтелекту. Активний інтерес до цієї теми в сучасному світі відображається в безперервному зростанні досліджень і розробок, що свідчить про важливість цієї сфери для подальшого технологічного прогресу і поліпшення якості життя.

Комп'ютерний зір потребує багато даних. Він аналізує дані знову і знову, поки не побачить відмінності і, зрештою, не розпізнає зображення. Наприклад, щоб навчити комп'ютер розпізнавати автомобільні шини, йому потрібно надати величезну кількість зображень шин і предметів, пов'язаних з шинами, щоб він зміг розпізнати відмінності і розпізнати шину, особливо ту, яка не має дефектів [1].

1.1.2 Розпізнавання образів та машинне навчання

Образ – це об’єкт, процес або явище реального та абстрактного світу, який розпізнається за даними (ознаками), що збираються та оброблюються індивідуально і у сукупності. Розпізнавання образу не є безцільною грою і завжди супроводжується дією. «Адекватна дія» на розпізнаний конкретний об’єкт є основою життєдіяльності будь-якого живого організму. В штучних інтелектуальних системах дія має вигляд порівняння з еталоном, увімкнення в роботу виконавчого пристрою, вилучення з розгляду образу, запису або видачі інформації тощо [5].

Образи для розпізнавання обличчя включають змінні умови освітлення, різні позиції та кути, різноманітні вирази обличчя, зміну волосся та аксесуарів, різні фонові зображення, зміну макіяжу, врахування зовнішніх змін та віку, різні етнічні групи та динамічні обличчя у відеоформаті. Ці різноманітність у вхідних даних допомагає моделям нейронних мереж адаптуватися до різних умов та сценаріїв розпізнавання обличчя.

Образи для розпізнавання включають в себе широкий спектр реальних і абстрактних об'єктів, а також характеристики людей, виробництво товарів, економічні явища, ситуації та процеси, соціальні взаємодії, небезпеки в роботі компаній, комунікації (листи), автомобілі та інше. Кожна з цих сфер діяльності має свої унікальні образи, які можуть бути визначені та розпізнані за допомогою різних технологій, таких як розпізнавання образів та нейронні мережі. Оскільки

обсяг знань та інформаційних технологій постійно зростає, дисципліна розпізнавання образів постійно розвивається, ставлячи перед собою завдання ефективно впоратися з різноманіттям областей та об'ємом інформації.

Розпізнавання образів у системах штучного інтелекту та машинного навчання включає в себе ряд важливих аспектів, які визначають ефективність і точність процесу.

Початковим етапом розпізнавання образів є визначення ключових ознак, які репрезентують основні характеристики об'єктів чи явищ у зображеннях. Цей процес включає в себе екстракцію важливих піксельних, текстурних, геометричних та структурних параметрів, що формують унікальну сигнатуру для кожного об'єкта. Після екстракції ознак використовуються алгоритми класифікації для призначення об'єкта до відповідної категорії. Нейронні мережі, методи опорних векторів та різноманітні статистичні класифікатори використовуються для навчання моделей розпізнавання образів. Точність цього етапу напряму залежить від якості екстракції ознак.

В процесі розпізнавання може бути важливим виділення областей зображення, які мають велику значущість або концентрацію інформації. Методи сегментації та відокремлення областей інтересу допомагають зосередити увагу моделі на ключових деталях образу. Після класифікації та виділення образів, необхідно створити адекватний опис для отриманих результатів. Це може включати в себе створення текстових описів, графічних анотацій чи інших форм репрезентації інформації, що допомагає зрозуміти сутність класифікованого об'єкта.

Загалом, характеристики об'єкта можуть приймати різні форми, охоплюючи різні аспекти та властивості об'єкта. Наприклад, опис людини може включати виміряні числові параметри (зріст, вага), якісні аспекти (розумові дані, мудрість) та описові характеристики (словесні описи зовнішності та особистості).

Для товарів характеристики можна комбінувати, враховуючи числові

показники (вага, ціна), якісні аспекти (якість виготовлення, конкурентоспроможність) та описові дані (виробник, умови зберігання, термін придатності).

Сучасні системи розпізнавання образів є необхідною складовою будь-яких штучних інтелектуальних систем. Вони застосовуються у експертних системах, базах даних, базах знань та інших інформаційних системах, включаючи прогнозні системи. Як приклад, можуть слугувати комплексні інтелектуальні системи «Здобуття знань» для прийняття рішень, які використовують ряд методів РО (рисунок 1.6)

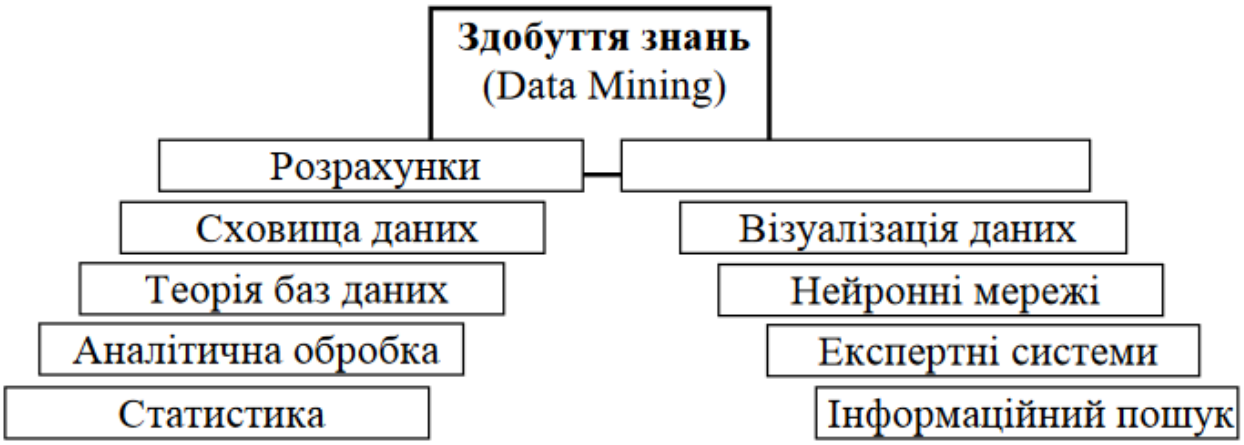

Рисунок 1.6 – Система «Здобуття знань»

Мета розпізнавання образів полягає в класифікації об'єкта за визначеними ознаками, щоб система розпізнавання образів могла виконати оптимальні дії щодо розпізнаного об'єкта.

Машинне навчання – це один із способів застосування штучного інтелекту в комп'ютерних технологіях під час роботи з різними даними. Завдяки машинному навчанню програмні програми можуть точніше прогнозувати результати та аналізувати дані. Основна мета та ідея машинного навчання – дозволити комп'ютерам навчатися самим, автоматично та без втручання людини

[5]

За прогнозами експертів, машинне навчання визначає майбутнє. З поширенням залежності людей від машин та гаджетів відбувається глобальна технологічна революція, що призводить до появи нових професій і зникнення старих.

Навчання з вчителем – це напрямок машинного навчання, який об'єднує алгоритми та методи конструювання моделей на основі множини прикладів з парами "відомий вхід – відомий вихід".

Щоб алгоритм був віднесений до навчання з вчителем, він повинен працювати з прикладами, що включають не лише вектор незалежних змінних (атрибутів, ознак), але й значення, яке модель має виводити після навчання (це значення відоме як цільовий). Різниця між цільовим і фактичними виходами моделі називається помилки навчання (різниці, залишки), яка мінімізується в процесі навчання та виступає як "вчитель". Значення цієї помилки використовується для коригування параметрів моделі на кожній ітерації навчання. (рисунок 1.7)

В алгоритмах навчання без вчителя вихідна помилка моделі на навчальній сукупності не розраховується. Замість цього використовується інформація про поточний стан параметрів моделі та приклади навчального набору. Наприклад, це може бути евклідова відстань між вектором прикладу та вектором ваги нейрона, який визначає корекцію параметрів моделі під час навчання.

Основне використання навчання без вчителя полягає в створенні моделей для кластеризації. Оскільки кластерна структура даних заздалегідь невідома і визначається процесом навчання моделі, цільові значення використовувати не можна.

Типовими прикладами моделей навчання без вчителя є мережі та карти Кохонена, які широко застосовуються в аналізі даних. В їхньому основі лежить конкурентне навчання, де коригування масштабів нейронів здійснюється на основі відстані між їхніми масштабними векторами та векторами ознак

навчальних прикладів.

Альтернативною технологією є навчання з одним вчителем, яке передбачає встановлення цільового значення для кожного навчального прикладу та розрахунок вихідної похибки, на основі якої визначаються корекції розміру та знаку параметрів моделі.

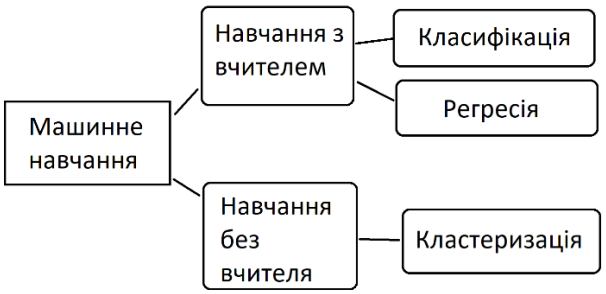

Рисунок 1.7 – Поділ машинного навчання на категорії

Алгоритми машинного навчання представляють собою фрагменти коду, які допомагають людям досліджувати, аналізувати та знаходити значення в складних наборах даних. Кожен алгоритм є кінцевим набором чітких послідовних інструкцій, за якими машина може виконувати завдання з досягнення певної мети. У моделі машинного навчання мета полягає в тому, щоб встановити або виявити шаблони, які люди можуть використовувати для прогнозування або категоризації інформації.

Алгоритми машинного навчання використовують параметри, які базуються на навчальних даних — підмножині даних, яка представляє більший набір. З розширенням навчальних даних для реалістичнішого відображення світу, алгоритм обчислює більш точні результати.

Різні алгоритми аналізують дані за різними методами. Їх часто групують за методами машинного навчання, такими як контрольоване навчання, неконтрольоване навчання та навчання з підкріпленням. Алгоритми, які найчастіше використовуються, використовують регресію та класифікацію для

прогнозування цільових категорій, виявлення незвичайних точок даних, прогнозування значень і виявлення подібностей.

Перцептрон - це найпростіша архітектура нейронної мережі. Це тип нейронної мережі, яка отримує ряд входів, застосовує певні математичні операції над цими входами і виробляє вихід. Вона приймає на вхід вектор реальних значень, виконує лінійну комбінацію кожного атрибуту з відповідною вагою, присвоєною кожному з них. Зважені входи підсумовуються в єдине значення і проходять через функцію активації. Ці перцептрони об'єднуються для формування більшої архітектури штучної нейронної мережі.

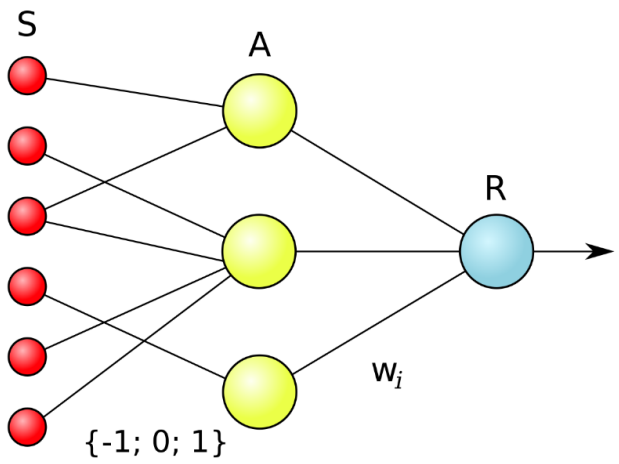

Рисунок 1.8 – Логічна схема елементарного перцептрону

Мережі прямого поширення. Перцептрон представляє роботу окремого нейрона. Це багатошарова нейронна мережа, і, як випливає з назви, інформація передається в прямому напрямку - зліва направо. При прямому проходженні інформація надходить всередину моделі через вхідний шар, проходить через серію прихованих шарів і, нарешті, потрапляє до вихідного шару. Ця архітектура нейронних мереж є прямою за своєю природою - інформація не зациклюється на двох прихованих шарах. Наступні шари не мають зворотного зв'язку з попередніми шарами. Основний процес навчання мереж прямого поширення залишається таким же, як і у персептрона.

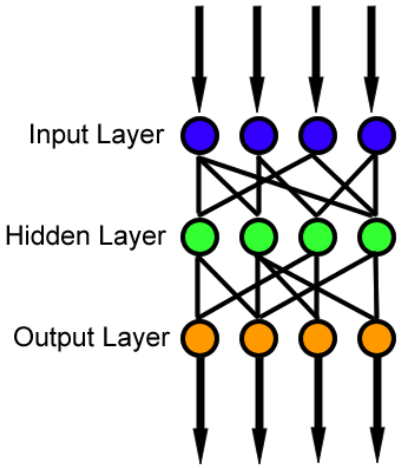

Рисунок 1.9 – Логічна схема мережі прямого поширення

Залишкові мережі (ResNet) Чим більше прихованих шарів, тим краще процес навчання. Більше шарів збагачують рівні можливостей. Дуже глибокі нейронні мережі надзвичайно складно навчати через проблеми зникаючих і вибухаючих градієнтів. ResNets забезпечують альтернативний шлях для потоку даних, щоб зробити процес навчання набагато швидшим і простішим. Це відрізняється від прямого підходу попередніх архітектур нейронних мереж. Основна ідея ResNet полягає в тому, що глибша мережа може бути створена з дрібної мережі шляхом копіювання ваги з дрібних аналогів за допомогою ідентифікаційного мапування. Дані з попередніх шарів швидко перемотуються вперед і копіюються набагато далі в нейронних мережах. Це те, що ми називаємо "пропускними зв'язками", вперше введеними в залишкових мережах для вирішення проблеми зникаючих градієнтів.

Згорткові нейронні мережі (CNN) - це тип нейронних мереж прямого поширення, які використовуються в таких задачах, як аналіз зображень, обробка природної мови та інших складних задачах класифікації зображень. ШНМ має приховані шари згорткових шарів, які формують основу ConvNets. Особливості стосуються найдрібніших деталей у даних зображення, таких як краї, межі, форми, текстури, об'єкти, кола тощо. На вищому рівні шари згортки виявляють

ці особливості в даних зображення за допомогою фільтрів. Деталі вищого рівня опрацьовуються першими кількома згортковими шарами. Чим глибше мережа занурюється, тим складнішим стає пошук закономірностей. Наприклад, у наступних шарах замість країв і простих форм фільтри можуть виявляти специфічні об'єкти, такі як очі або вуха, і, зрештою, кота, собаку і не тільки [6].

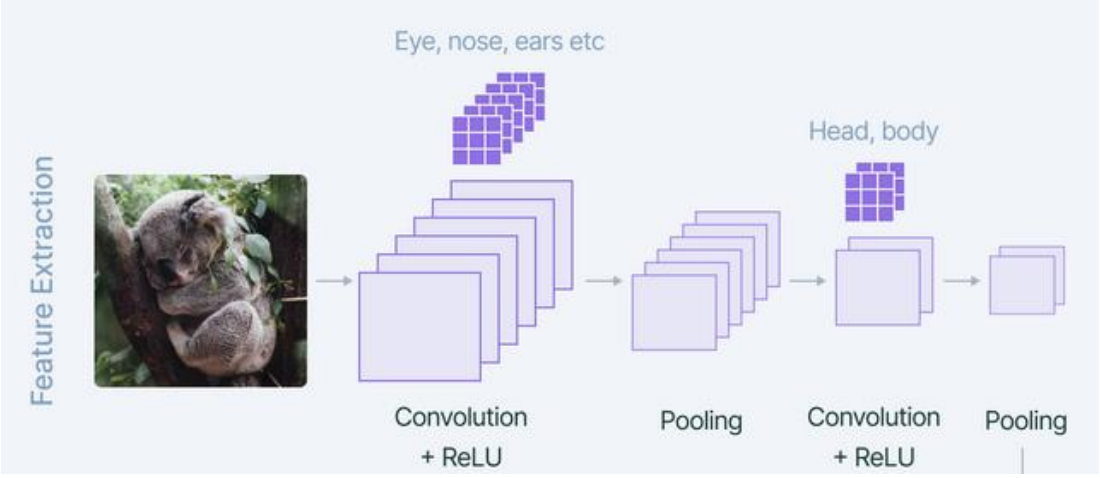

Рисунок 1.10 - Виділення та класифікація ознак у згорткових нейронних мережах

При додаванні згорткового шару до мережі нам потрібно вказати кількість фільтрів. Фільтр можна розглядати як відносно невелику матрицю, для якої ми визначаємо кількість рядків і стовпців. Значення цієї матриці ознак ініціалізується випадковими числами. Коли цей згортковий шар отримує значення пікселів вхідних даних, фільтр буде згортатися над кожною ділянкою вхідної матриці. Вихід згорткового шару зазвичай пропускається через функцію активації ReLU, щоб додати моделі нелінійності. Вона бере карту ознак і замінює всі від'ємні значення на нуль. Об'єднання є дуже важливим кроком у ConvNets, оскільки воно зменшує обчислення і робить модель толерантною до спотворень і варіацій. Повністю зв'язані щільні нейронні мережі використовують сплющену матрицю ознак і прогнозують відповідно до сценарію використання.

Deconvolutional Neural Networks (DNN) - це ШНМ, які працюють у

зворотному напрямку. Розмір зображення зменшується за допомогою згорткових шарів і максимального об'єднання. Для відновлення початкового розміру використовується передискретизація та транспоновані згорточні шари. Повторна вибірка не має параметра навчання і просто повторює рядки і стовпчики даних зображення у відповідному розмірі.

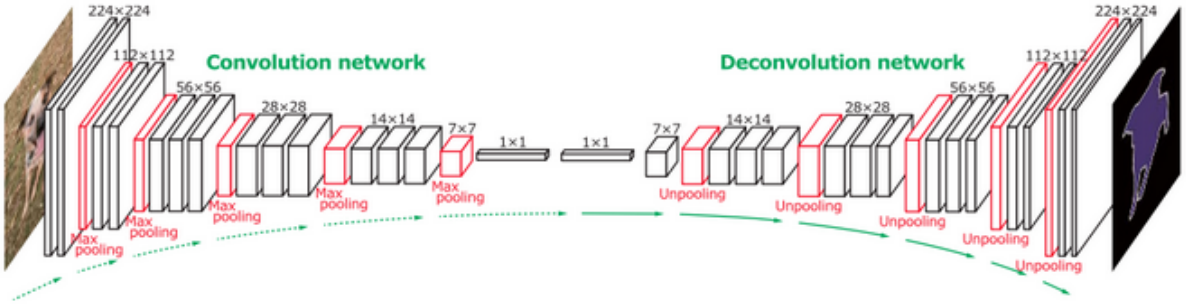

Рисунок 1.11 – Ілюстрація роботи DNN

Транспонований згорточний шар означає одночасне застосування операцій згортки та апсемплінгу, що описується параметром Conv2DTranspose (кількість фільтрів, розмір фільтра, крок). stride=1, апсемплінг не виконується і отримується той самий розмір вхідних даних.

Overfeat - ця архітектура нейронних мереж досліджує три добре відомі задачі технічного зору - класифікацію, локалізацію та виявлення, використовуючи єдиний фреймворк. Вона тренує моделі на всіх трьох завданнях одночасно, щоб підвищити точність. Це модифікація AlexNet. Вона прогнозує обмежувальні рамки для кожного просторового положення і масштабу. Для локалізації класифікаційну голову замінено на регресійну мережу.

Після аналітичного огляду архітектур нейронних мереж можна вибрати ту, яка найкраще виконує функцію розпізнавання дорожніх знаків. Порівнявши сфери застосування та характеристики кожної з них, можна зробити висновок, що згорткові нейронні мережі найкраще підходять для цього завдання

Згортковий шар є ключовим елементом згорткової нейронної мережі. У

цьому шарі для кожного каналу використовується фільтр яке є ядром згортки. Кожне ядро згортки обробляє попередній шар фрагментами, підсумовуючи результати поелементного добутку для кожного фрагмента. Вагові коефіцієнти ядра згортки, представлені невеликою матрицею, є невідомими і встановлюються під час навчання процесу [5].

Однією з особливостей згорткового шару є його відносно невелика кількість параметрів, яка визначається під час навчання. Наприклад, якщо вихідне зображення має розмірність 100 × 100 пікселів для трьох каналів (загалом 30000 вхідних нейронів), а згортковий шар використовує фільтри з ядром розміром 3x3 пікселя та генерує 6 каналів на вихід, то в процесі навчання потрібно визначити лише 9 ваг ядра. Однак, при урахуванні всіх комбінацій каналів, це означає лише 9 × 3 × 6 = 162 параметри. У порівнянні з кількістю параметрів у повнозв'язаній нейронній мережі, це значно менше, що робить згортковий шар більш ефективним [6].

Шар активації призначений для обробки скалярного результату кожного згорткового фільтру за допомогою нелінійної функції активації. Зазвичай вважається, що функцію активації можна вбудувати безпосередньо в згортковий шар. Щодо шару субдискретизації (пулінгу), він виконує нелінійне стиснення карти ознак, об'єднуючи групи пікселів (зазвичай 2 × 2) в один піксель через нелінійне перетворення. Функція максимуму часто використовується для вибору пікселя з максимальним значенням у кожній групі. Операція пулінгу дозволяє суттєво зменшити просторовий обсяг зображення, зберігаючи важливі особливості.

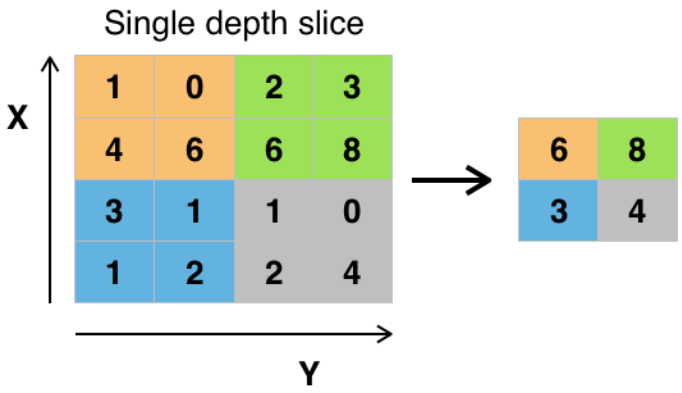

Рисунок 1.12 - Максимальне об'єднання з фільтром 2x2 і кроком = 2

Пулінг можна розглядати як стратегію оптимізації: якщо попередній згортковий шар вже виявив певні ознаки, то для подальшої обробки можна ущільнити зображення, зменшивши його докладність. Це обумовлено тим, що докладніше зображення може стати зайвим для подальшого виявлення ознак. Крім того, під час пулінгу фільтруються непотрібні деталі, що сприяє уникненню перенавчання. Зазвичай шар пулінгу вставляється після згорткового шару та перед наступним згортковим шаром.

1.2 Аналіз досліджуваних технологій в існуючих системах

1.2.1 Принцип роботи TSR системи

Оскільки автовиробники знаходять нові способи убезпечити пасажирів транспортних засобів, список передових систем допомоги водієві (ADAS) продовжує розширюватися. Одна з останніх інновацій у сфері ADAS називається розпізнавання дорожніх знаків (TSR) [7]. Розпізнавання дорожніх знаків, по суті, слугує другою парою очей для водія, підвищуючи обізнаність про певні дорожні знаки, що допомагає приймати кращі та безпечніші рішення знаходячись на дорозі.

Система розпізнавання дорожніх знаків - це технологічна система безпеки,

яка розпізнає дорожні знаки і передає інформацію з них водієві через приладову панель, інформаційно-розважальний дисплей або дисплей на лобовому склі. Більшість систем TSR можуть розпізнавати знаки обмеження швидкості, зупинки та заборони в'їзду. Більш досконалі системи можуть розпізнавати інші типи знаків; основна мета TSR - підвищити пильність водія. Якщо водій пропускає знак, TSR нагадує йому про нього, щоб він міг відреагувати належним чином. Ідея проста: TSR ідентифікує дорожні знаки, які водій міг пропустити, і попереджає його про їхню наявність. Технологія використовує передову камеру, спрямовану вперед, розташовану у верхній частині лобового скла, зазвичай поруч з корпусом дзеркала заднього виду. Щоб "бачити" дорожні знаки, камера сканує край дороги по відношенню до автомобіля. Коли камера виявляє знак, програмне забезпечення системи обробляє зображення, щоб визначити його класифікацію та значення. Потім система майже миттєво передає цю інформацію водієві у вигляді піктограм або графічного зображення знаку. Однак здатність TSR точно ідентифікувати знаки залежить від швидкості автомобіля та відстані до знаку; деякі системи TSR працюють у поєднанні з сучасним круїз-контролем, який підлаштовується так, щоб підтримувати швидкість вище або нижче розпізнаного знаку.

Наприклад, якщо система TSR виявляє обмеження швидкості 40 миль/год, круїз-контроль оновить швидкість до 40 миль/год, якщо водій не встановить параметр вище або нижче виявленого обмеження швидкості. На додаток до функцій, пов'язаних з круїз-контролем, TSR може використовувати ті ж камери ADAS, які стежать за розміткою смуги руху, для інформування систем попередження про виїзд зі смуги руху і попередження про відволікання водія. Тому ці функції часто пропонуються в тому ж пакеті, що і ADAS з TSR.

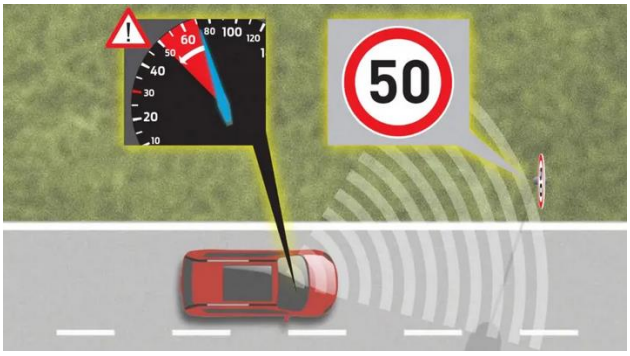

Рисунок 1.13 – Приклад системи розпізнавання дорожніх знаків

Системи розпізнавання дорожніх знаків в автомобільному секторі використовують ряд технологій для ефективного функціонування. Основний алгоритм роботи подібних систем можна узагальнити наступним чином. Збір даних – система використовує різні датчики та камери, встановлені на автомобілі, для збору зображень з дороги та навколишнього середовища.

Обробка зображень – отримані зображення піддаються обробці за допомогою алгоритмів комп'ютерного зору та обробки зображень. Це може включати в себе фільтрацію шумів, виокремлення контурів та інші методи покращення якості зображення.

Визначення областей інтересу (ROI) – система визначає області зображення, які є потенційно важливими для розпізнавання дорожніх знаків. Це може бути обмеження шляху, де ймовірно знаходиться дорожній знак.

Виявлення знаків – алгоритми розпізнавання використовуються для виявлення дорожніх знаків в обраних областях. Це може включати в себе класифікацію форм та кольорів, оскільки багато дорожніх знаків мають характерні риси.

Класифікація знаків – після виявлення система класифікує тип дорожнього знаку. Це може бути обмеження швидкості, знаки заборони, інформаційні знаки тощо.

Виведення інформації – класифікована інформація виводиться для водія. Це може бути відображено на бортовому комп'ютері, водійському дисплеї або

іншому інтерфейсі.

Оновлення та навчання – система може мати механізми для постійного оновлення та навчання, оскільки дорожні знаки можуть змінюватися, а систему треба адаптувати до нових умов.

Ці кроки демонструють загальний підхід до функціонування систем розпізнавання дорожніх знаків. Важливо враховувати, що кожен виробник може використовувати власні техніки та технології для досягнення мети розпізнавання дорожніх знаків в їхніх автомобільних системах.

1.2.2 Система Traffic Sign Assist у Mercedes-Benz

Traffic Sign Assist від Mercedes-Benz — це система, яка розпізнає дорожні знаки, надаючи водію важливу інформацію на бортовому екрані або панелі приладів автомобіля [8]. Система включає в себе розпізнавання знаків швидкості та інших дорожніх знаків, таких як знаки заборони чи пріоритету. Інформація про обмеження швидкості відображається, щоб підвищити увагу водія та забезпечити безпечний рух. Система може також надавати аудіосигнали або вібрацію для додаткового попередження. Важливим елементом є адаптація до місцевих стандартів, дозволяючи ефективно працювати в різних регіонах світу. Однією з основних функцій є попередження водія, якщо він перевищує встановлене обмеження швидкості, підкреслюючи важливість безпеки на дорозі.

1.2.3 Система Speed Limit Info у BMW

Speed Limit Info – це система допомоги водієві BMW, на рисунку 1.14 [9]. Вона допомагає водієві дотримуватися поточного обмеження максимальної швидкості, визначаючи обмеження швидкості за допомогою цифрових дорожніх карт і дорожніх знаків. Вона використовує вбудовану камеру, яка розпізнає дорожні знаки та інформацію про швидкість. Після розпізнавання система

відображає відповідні дані на бортовому комп'ютері або на водійському дисплеї. Якщо водій перевищує встановлене обмеження швидкості, система може надати візуальне або звукове сповіщення для попередження. Speed Limit Info сприяє безпечному та відповідальному водінню, допомагаючи водієві дотримуватися правил дорожнього руху.

Рисунок 1.14 – Необхідні компоненти для роботи BMW - Speed Limit Info

1.2.4 Система Tesla Autopilot

Tesla Autopilot - це вдосконалена система автоматизованого водіння, розроблена компанією Tesla для електромобілів . Автопілот використовує декілька датчиків, камер, радарів і вбудованих обчислювальних ресурсів для забезпечення функцій автоматизованого водіння [10].

Камери автомобіля Tesla оснащені декількома камерами, розташованими навколо автомобіля, щоб забезпечити огляд навколишнього середовища.

Вбудований радар дозволяє автомобілю виявляти об'єкти навіть за поганих погодних умов.

Дані, зібрані з камер і радарів, обробляються вбудованим обчислювальним

блоком.

Використовуючи глибоке навчання і штучний інтелект, Tesla впровадила нейронну мережу для аналізу зображень і прийняття рішень в режимі реального часу.

Автопілот Tesla може автоматично керувати автомобілем на шосе, підтримувати смугу руху, регулювати швидкість і виконувати маневри обгону.

Система управління враховує дані та інформацію з датчиків для визначення команд, які система управління надсилає автомобілю, забезпечуючи безпечне та ефективне водіння.

Tesla пропонує можливість дистанційного оновлення програмного забезпечення, що дозволяє оснащувати автомобіль новими функціями та вдосконалювати його з часом.

Автопілот є проміжним кроком на шляху до повністю автономного водіння і є частиною ширшої стратегії Tesla в цьому напрямку.

1.3 Постановка задачі дослідження

Існуючі рішення систем розпізнавання дорожніх знаків від автовиробників мають закритий вихідний код, що ускладнює процес розробки власної системи, або модифікації вже існуючої системи. Такі системи мають сувору прив’язку до оновлень що випускає компанія, виявити чи порівняти їх точність неможливо. Види попереджень що можуть виводити на дисплей теж обмежені системою, це може бути попередження про ліміт швидкості, знаки пріоритету на дорозі, та виявлення об’єктів на шляху автомобіля. Створення власної системи з штучним інтелектом дозволить розробникам ПЗ та дослідникам модифікувати та навчати її під будь які потреби та з легкістю додавати власні набори даних.

Потрібно розробити, дослідити та порівняти кілька нейронних мереж для розпізнавання дорожніх знаків і вебзастосунок у який буде інтегровано ці мережі.

Дослідити вплив певних перешкод на роботу навчених нейронних мереж,

на прикладі VGG-19, LeNet-5 та 13 у сфері розпізнавання дорожніх знаків. Навчити ці нейронні мережи, а потім розробити програмний продукт на мові програмування Python, до функціоналу якого буде входити, завантаження зображення, його розпізнавання за допомогою обраної нейронної мережі, а також відображення результату операції.

Окрім цього, розроблена і навчена нейронна мережа має бути інтегрована у вебзастосунок, який повинен підтримувати інтеграцію нейронних мереж за допомогою сучасних бібліотек машинного навчання та надавати користувачам такі можливості:

– перегляд даних про архітектуру кожної навченої мережі;

– виведення статистики про продуктивність та точність кожної мережі;

– можливість завантажити зображення для подальших операцій;

– використання обраної нейронної мережі для розпізнавання знаків на завантажених зображеннях;

– отримання детальної інформації про результати операції розпізнавання.

Щоб виконати ці задачі, слід виконати такі етапи:

– знайти відповідний датасет для навчання нейронних мереж у сфері розпізнавання дорожніх знаків;

– обрати підходящу архітектуру для розпізнавання дорожніх знаків;

– визначити гіперпараметри для нових нейронних мереж;

– провести процес навчання для кількох нейронних мереж;

– оцінити якість навчених мереж, використовуючи метрики ефективності;

– завантажити навчені моделі нейронних мереж;

– обрати модель вебзастосунка для подальшої інтеграції з нейронними мережами;

– обрати архітектуру для вебзастосунка, що відповідає потребам інтеграції;

– розробити вебзастосунок для виконання вищезазначених завдань;

– інтегрувати навчені нейронні мережі у вебзастосунок, виконати тестування розробленого вебзастосунка для перевірки його функціональності та ефективності.

2 МАТЕМАТИЧНИЙ ОПИС ЗАДАЧ

2.1 Архітектура нейронної мережі

Архітектура нейронної мережі - це дизайн і структура штучної нейронної мережі (ШНМ), моделі машинного навчання, натхненної людським мозком. Вона описує, як розташовані шари, вузли та зв'язки мережі для обробки та аналізу даних. Нейронна мережа складається з шарів взаємопов'язаних вузлів, які називаються нейронами, що обробляють вхідні дані і передають результати на наступний шар. Кожен нейрон застосовує математичну функцію до входу, щоб отримати вихід, який потім передається на наступний рівень. Архітектура нейронної мережі визначає складність, потужність і функціональність моделі [11].

Штучна нейронна мережа зазвичай складається з численних процесорів, які працюють паралельно та організовані в шари. Перший шар, схожий на зорові нерви в людському зоровому аналізаторі, отримує необроблену вхідну інформацію. Кожен наступний шар отримує вихідні дані від попереднього, а не необроблену вхідну інформацію, аналогічно тому, як віддалені нейрони отримують сигнали від ближчих. Останній шар генерує вихідні дані системи. Кожен обчислювальний вузол має свою власну область знань, включаючи інформацію, яку він обробив, та правила, які він може був початково запрограмований або навчитися. Шари тісно пов'язані, тобто вузол рівня N пов'язаний з вузлами рівня N-1 (його входами) та рівня N+1, які подають вхідні дані для цих вузлів. Вихідний шар може містити один чи більше вузлів, які містять вихідні дані системи [12].

На Рисунку 2.1 можна побачити нейронну мережу, що складається з взаємопов'язаних нейронів. Кожен нейрон характеризується своїми вагами, зсувами та функцією активації.

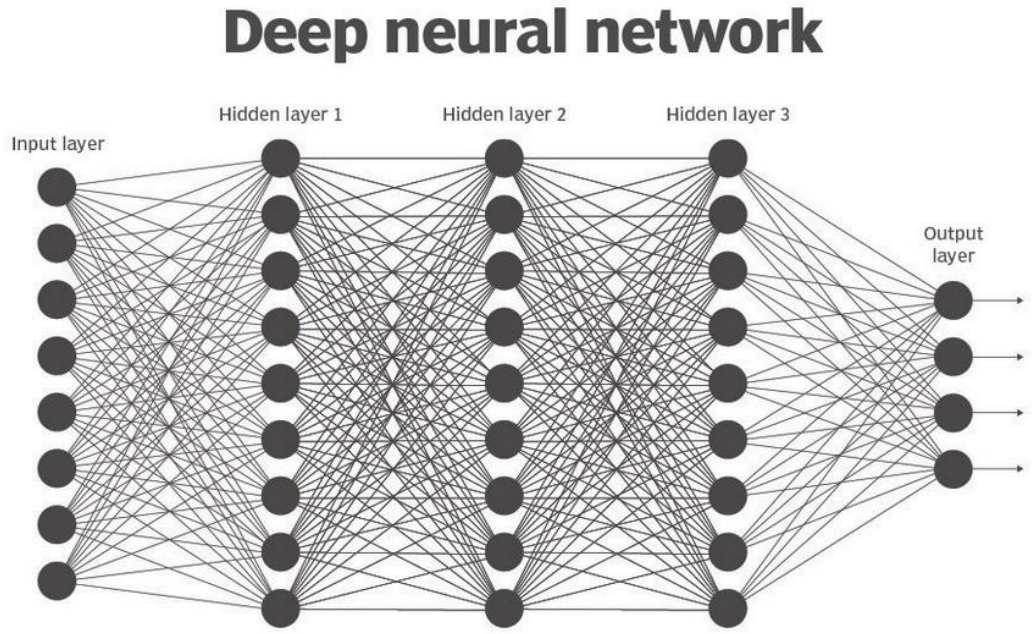

Рисунок 2.1 - Кожен шар нейронної мережі складається з маленьких окремих нейронів.

Нижче наведено інші елементи цієї мережі:

- вхідний шар отримує необроблені дані з поля. На цьому рівні не відбувається жодних обчислень. Вузли тут лише передають інформацію (ознаки) до прихованого шару;
- прихований шар - як випливає з назви, вузли в цьому шарі не є відкритими. Це абстракція нейронної мережі;
- прихований шар виконує всі види обчислень над ознаками, введеними через вхідний шар, і передає результати вихідному шару;
- вихідний шар - це останній шар мережі, який обробляє інформацію, отриману через прихований шар, і в результаті видає остаточне значення.

Функція активації визначає вихідне значення нейрона залежно від результату зваженої суми входів і порогового значення.

Функція активації визначає, чи активований нейрон, чи ні. Іншими словами, вона використовує прості математичні операції, щоб визначити, чи є

вхідні дані нейрона важливими для процесу прогнозування. Роль функції активації полягає в тому, щоб отримати вихід з набору вхідних значень, наданих вузлу (або шару). Якщо порівняти нейронну мережу з нашим мозком, то вузол - це копія нейрона, яка отримує набір вхідних сигналів, які є зовнішніми стимулами. Залежно від характеру та інтенсивності цих вхідних сигналів мозок обробляє їх і вирішує, чи активувати ("запускати") нейрони. Це роль функції активації в глибокому навчанні, і тому її часто називають передавальною функцією в нейронних мережах. Основна роль функції активації полягає в перетворенні суми зважених входів з вузлів у вихідне значення, яке надсилається до наступного прихованого шару або виходу. [13].

Мета функції активації - додати нейронній мережі нелінійну властивість. Функції активації створюють додатковий крок у кожному шарі під час зворотного поширення, але вони варті того, щоб їх обчислювати. Причини цього наступні. Припустимо, що існує нейронна мережа, яка не використовує функції активації. У цьому випадку кожен нейрон виконує лише лінійне перетворення вхідних даних за допомогою ваг і зсувів. Це відбувається незалежно від того, скільки прихованих шарів додано до нейронної мережі, оскільки композиція двох лінійних функцій сама по собі є лінійною функцією, тому всі шари поводяться однаково. Хоча нейронні мережі стають дедалі простішими, їх не можна навчити виконувати складні завдання, і наша модель - це просто модель лінійної регресії.

Для прикладу слід привести декілька видів функції активації. Бінарна ступінчаста функція залежить від порогового значення (рисунок 2.2), яке визначає, активується нейрон чи ні. Вхід функції активації порівнюється з пороговим значенням. Якщо вхід більше порогового значення, нейрон активується, інакше він деактивується, тобто його вихід не передається на наступний прихований шар [13].

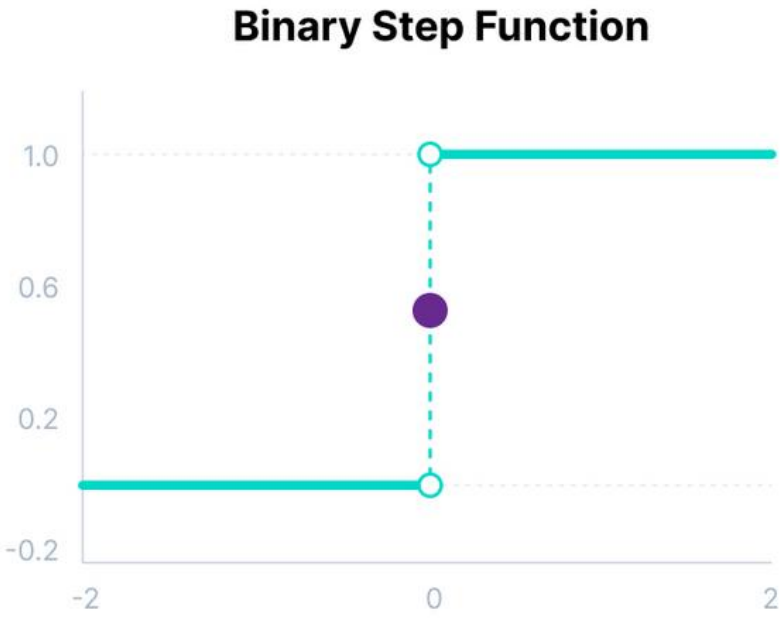

Рисунок 2.2 – Бінарна ступінчата функція

Обмеження бінарної крокової функції такі:

- багатозначні результати недоступні - наприклад, вона не може бути використана для задач багатокласової класифікації;
- градієнт ступеневої функції дорівнює нулю, що спричиняє шум у процесі зворотного розповсюдження.

Лінійна функція активації (рисунок 2.3) також відома як "без активації" або "тотожна функція" (x1.0 разів) - це функція, активація якої пропорційна її входам. Ця функція нічого не робить зі зваженою сумою входів і просто виводить задане значення [13].

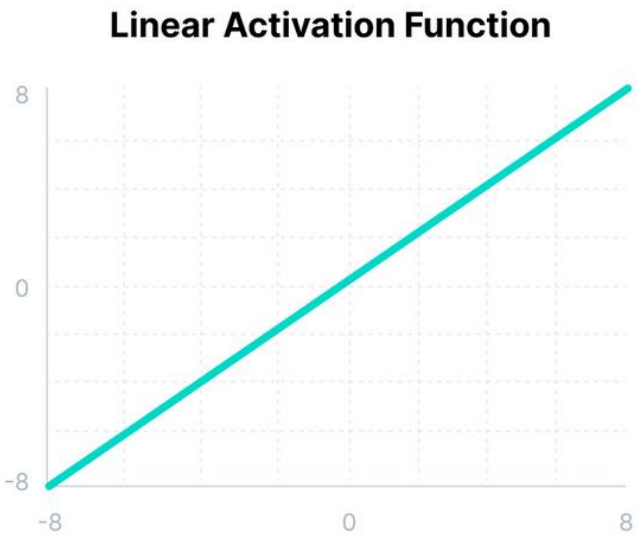

Рисунок 2.3 – Лінійна функція активації

Однак, є дві основні проблеми з лінійними функціями активації:

- зворотне поширення не може бути використане, оскільки похідні функції є постійними і не залежать від вхідних даних x;
- при використанні лінійної функції активації всі шари нейронної мережі зводяться до одного шару. Незалежно від кількості шарів у нейронній мережі, останній шар є лінійною функцією першого шару. Отже, по суті, лінійна функція активації зводить нейронну мережу лише до одного шару.

Сигмоїдальна/логістична функція активації, рисунок 2.4. Ця функція приймає на вхід будь-яке дійсне значення і видає на виході значення від 0 до 1.

Чим більше вхідне значення (чим ближче до додатного), тим ближче вихідне значення до 1.0; чим менше вхідне значення (чим ближче до від'ємного), тим ближче вихідне значення до 0.0 [13].

Рисунок 2.4 – Сигмоїдальна функція активації

Ось чому сигмоїдальна/логістична функція активації є однією з найпоширеніших функцій:

- вона часто використовується в моделях, де ймовірності мають бути оцінені як вихідні дані. Оскільки ймовірність чогось знаходиться лише між 0 та 1, сигмоїдальна функція є правильним вибором через свій діапазон;
- дана функція є диференційованою і уникає стрибків у вихідних значеннях, малюючи плавний градієнт. Це представлено S-подібною формою сигмоїдної функції активації.

Функція гіперболічного тангенсу, рисунок 2.5, дуже схожа на сигмоїдальну/логістичну функцію активації, за винятком того, що діапазон вихідних значень становить від -1 до 1. У функції Тана, чим більшим є вхідний сигнал (чим він додатніший), тим ближче вихідне значення до 1.0, і чим меншим є вхідний сигнал (чим він від'ємніший), тим ближче вихідне значення до -1.0.

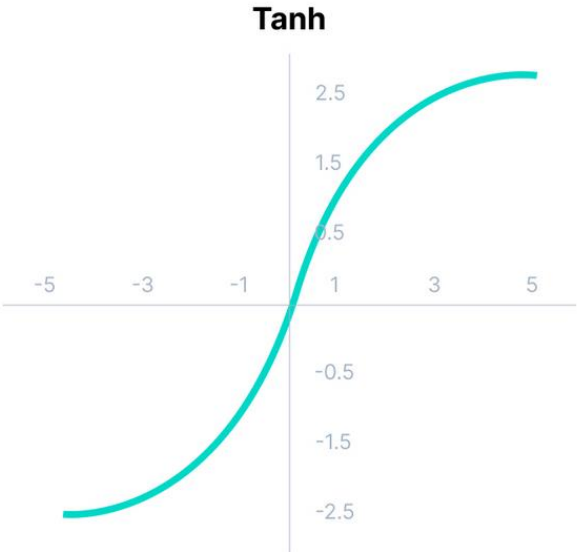

Рисунок 2.5 – Функція гіперболічного тангенсу

Перевагами використання цієї функції активації є те, що оскільки вихід тангенціальної функції активації зосереджений на нулі, вихідні значення можуть відображатися як сильно від'ємні, нейтральні або сильно додатні. Таким чином, середнє значення прихованого шару дорівнює або дуже близьке до нуля. Тому середнє значення прихованого шару буде дорівнювати нулю або дуже близьким до нього. Це гарантує, що дані відцентровані і робить наступний шар набагато простішим для вивчення.

Далі йде функція ReLU, рисунок 2.6. ReLU розшифровується як випрямлена лінійна одиниця. Хоча вона справляє враження лінійної функції, ReLU має похідну функцію і дозволяє зворотне поширення, одночасно роблячи її обчислювально ефективною. Основна проблема полягає в тому, що функція ReLU не активує всі нейрони одночасно. Нейрони будуть деактивовані лише тоді, коли результат лінійного перетворення буде меншим за 0 [13].

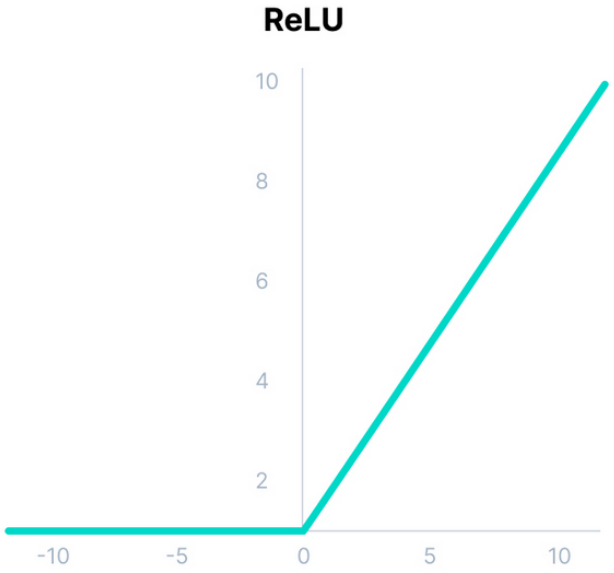

Рисунок 2.6 – Функція активації ReLU

Переваги використання ReLU як функції активації полягають у наступному:

– Оскільки активується лише певна кількість нейронів, функція ReLU є набагато ефективнішою в обчислювальному плані порівняно з сигмоїдною та тангенціальною функціями.

– ReLU прискорює збіжність градієнтного спуску до глобального мінімуму функції втрат завдяки своїй лінійній ненасичуваній властивості.

2.2 Вибір архітектури нейронної мережі

Навчання, або тренування є процесом оптимізації ваг в нейронній мережі з метою мінімізації помилки передбачення та досягнення необхідного рівня точності. Один із найбільш використовуваних методів для визначення внеску кожного нейрона у помилку - це зворотне поширення помилки, яке використовується для обчислення градієнта. Це є модифікацією методу градієнтного спуску. Використання додаткових прихованих шарів дозволяє

зробити систему більш гнучкою і потужною. Нейронні мережі з багатьма прихованими шарами отримали назву глибоких нейронних мереж (DNN), і вони формують складні нелінійні зв'язки [14].

Серед усіх зазначених у розділі 1 архітектур, слід відмітити переваги згорткової нейронної мережі, яку було обрано для подальшої розробки:

- здатність виявлення просторових особливостей - ЗНМ взаємодіють з даними, визначаючи просторові особливості, такі як краї, текстури та форми. Це дозволяє їм ефективно впоратися з завданнями обробки зображень;
- зменшення кількості параметрів - Згорткові шари в ЗНМ використовують фільтри, які діють локально, що дозволяє значно зменшити кількість параметрів, порівняно з повнозв'язаними шарами, зберігаючи при цьому важливу інформацію;
- спільне використання параметрів (підсумовування) - згорткові шари в ЗНМ можуть використовувати однакові фільтри на різних частинах вхідних даних, що дозволяє спільне використання параметрів і виявлення подібних ознак у різних частинах вхідного зображення;
- інваріантність до трансляції - ЗНМ можуть бути інваріантними до трансляційних зміщень, тобто вони можуть розпізнавати об'єкти, незалежно від їхнього місця на зображенні;
- локальна структура - ЗНМ добре пристосовані для роботи з даними, де локальні взаємозв'язки та структури грають важливу роль, таким чином, вони ефективно використовуються для обробки зображень, текстів і сигналів;
- використання шарів пулінгу - введення шарів пулінгу дозволяє зменшити просторовий обсяг зображення, зберігаючи ключові ознаки та сприяючи уникненню перенавчання;
- здатність до ієрархічного вивчення ознак - ЗНМ можуть автоматично вивчати ієрархічні рівні ознак, від простих до більш складних, що робить їх ефективними для розпізнавання об'єктів на різних рівнях абстракції.

2.3 Процес навчання нейронних мереж

Найважливішою характеристикою нейронних мереж є їх здатність до навчання на основі оточуючого середовища і поступового підвищення продуктивності в процесі навчання. Цей підвищений рівень продуктивності набувається з часом відповідно до певних правил. Процес навчання нейронної мережі здійснюється через взаємодію, під час якої коригуються синаптичні ваги і пороги. В ідеалі, на кожній ітерації навчального процесу нейронна мережа здобуває нові знання про своє оточення.

Існує кілька основних способів навчання нейронних мереж, кожен з яких відповідає різним задачам та вимагає відповідних методів підготовки та оптимізації. Основні способи навчання включають:

Навчання з вчителем (Supervised Learning) - Це найпоширеніший тип навчання, де модель навчається на основі пар вхід-вихід (зразків та відповідей). Модель спробує зробити прогноз для нових вхідних даних, і втримувана відповідь порівнюється з очікуваною відповіддю для корекції ваг.

Навчання без вчителя (Unsupervised Learning) - У цьому випадку модель навчається без надання пар вхід-вихід. Замість того, щоб вчити модель передбачати конкретні відповіді, вона намагається виявити внутрішню структуру та закономірності в наборі даних.

Навчання з підкріпленням (Reinforcement Learning) - В цьому підході модель навчається приймати рішення в середовищі, де вона знаходиться. Вона отримує винагороду або покарання в залежності від дій, які вона здійснює, і намагається максимізувати загальну винагороду.

Навчання з підганянням (Semi-Supervised Learning) - Це комбінований підхід, в якому модель навчається на деякій кількості пар вхід-вихід, а наступна намагається вивчити структуру навколо цих зразків для підготовки до роботи з новими даними.

Навчання з передачею (Transfer Learning) - Використовується, коли модель,

навчена для виконання одного завдання, використовується для розв'язання іншого подібного завдання. Це може зекономити час та ресурси, особливо, якщо вихідна модель має високу ефективність.

Навчання з вчителем (Supervised Learning) є одним з основних методів машинного навчання. У цьому підході модель отримує тренувальний набір даних, де кожен приклад має вхідні дані та відповідний очікуваний вихід. Основна мета полягає в тому, щоб модель вивчила функцію відображення між вхідними та вихідними даними.

Процес навчання включає в себе роботу моделі з тренувальним набором, де її параметри підлаштовуються так, щоб максимально точно відображати залежності між вхідом і вихідом. У цьому контексті використовується функція втрат, що вимірює різницю між прогнозованими та фактичними вихідними даними.

Застосування навчання з вчителем різноманітне і включає класифікацію, де модель визначає клас об'єкта, регресію, де модель прогнозує числове значення, та розпізнавання образів, де визначаються та розпізнаються об'єкти або шаблони на зображеннях.

Для оцінки продуктивності застосовуються різні метрики, такі як точність чи середньоквадратична помилка. Однак, існують виклики, такі як схильність до перенавчання при обмеженому обсязі даних та необхідність наявності достатньої кількості маркованих даних.

Деякі приклади алгоритмів навчання з вчителем включають лінійну регресію для задач регресії, логістичну регресію для класифікації та метод опорних векторів (SVM) для класифікації та регресії. Цей підхід залишається одним із найпоширеніших і використовується у різних сферах, де можливе маркування даних для тренування.

Вибірка представляє собою підмножину даних, що взята з загального набору даних, і використовується для того щоб зробити висновки про параметри та властивості популяції.

Навчальна вибірка, або тренувальна вибірка, є підмножиною даних, що використовується для тренування моделі в машинному навчанні. Модель вивчає залежності між вхідними та вихідними даними, виправляючи свої параметри.

Перевірочна вибірка, або тестова вибірка, - це інша підмножина даних, яка не використовується під час тренування моделі. Вона використовується для оцінки продуктивності та узагальнюючої здатності моделі на нових, раніше не бачених даних. Після тренування модель оцінюється на перевірочній вибірці для визначення її точності та здатності до узагальнення.

Тестова (контрольна) вибірка в машинному навчанні представляє собою підмножину даних, яка використовується для оцінки ефективності моделі після завершення її тренування на навчальній вибірці. Основна мета - визначити, наскільки добре модель може узагальнити свої навички на нові, раніше не бачені дані.

Основні аспекти тестової (контрольної) вибірки включають незалежність від тренувальної вибірки, визначення ефективності моделі, запобігання перенавчанню та визначення потреб у налаштуванні. Ця вибірка допомагає виявити, чи може модель вирішувати завдання на нових даних та чи вона піддається перенавчанню. Визначення ефективності на тестовій вибірці є ключовим етапом в оцінці загальної продуктивності моделі та визначенні її придатності для реальних застосувань.

3 РОЗРОБКА ТА ОПИС ЗАСТОСУНКУ

3.1 Вибір мови програмування.

Python – це інтерпретована об'єктно-орієнтована мова програмування високого рівня з динамічною семантикою. Вбудовані високорівневі структури даних у поєднанні з динамічною типізацією та динамічним зв'язуванням роблять її дуже привабливою не лише для швидкої розробки додатків, але й як мову сценаріїв та зв'язування для зв'язування існуючих компонентів Простий і легкий для вивчення синтаксис Python підкреслює читабельність і зменшує витрати на підтримку додатків; Python підтримує модулі та пакети для полегшення модуляризації та повторного використання коду; інтерпретатор Python та велика стандартна бібліотека вільно доступні у вигляді вихідних кодів або двійкових файлів на всіх основних платформах та вільно доступні і вільно розповсюджуються у вигляді вихідних кодів або двійкових файлів на всіх основних платформах [15].

Python є однією з найбільш популярних та ефективних мов програмування для роботи з нейронними мережами. Ця популярність обумовлена кількома ключовими факторами.

По-перше, в Python існує велика кількість бібліотек для роботи з машинним навчанням та глибоким навчанням, які роблять використання нейронних мереж зручним та ефективним. Такі бібліотеки, як TensorFlow, PyTorch, та Keras, надають широкий спектр інструментів для розробки, навчання та валідації нейронних мереж.

По-друге, Python має простий та читабельний синтаксис, що робить його доступним для широкого кола розробників та дослідників. Це важливо у галузі штучного інтелекту, де важливо мати зрозумілу та ефективну мову програмування для розробки та експериментів.

По-третє, Python має велику та активну спільноту розробників, яка вносить

значний вклад у розвиток та вдосконалення інструментів для нейронних мереж. Це забезпечує постійну підтримку, швидкі виправлення помилок та розвиток нових можливостей у сфері машинного навчання.

Крім того, в Python існує безліч додаткових бібліотек для обробки даних, візуалізації результатів та інших завдань, пов'язаних з роботою в галузі нейронних мереж. Це робить його повноцінним інструментом для енд-ту-енд розробки та впровадження проектів у галузі глибокого навчання.

Найпопулярніша бібліотека - NumPy для наукових розрахунків. Існує SciPy для більш складних обчислень та scikit для розвідки та аналізу даних.

Ці бібліотеки працюють разом з потужними фреймворками, такими як TensorFlow, CNTK та Apache Spark. Ці бібліотеки та фреймворки необхідні для проектів машинного навчання та глибокого навчання.

Простота: Код на Python корисний для проектів машинного та глибокого навчання, тому що він лаконічний і легкий для читання навіть для нових розробників. Завдяки простому синтаксису розробка додатків на Python відбувається швидше, ніж на багатьох інших мовах програмування. Крім того, розробники можуть тестувати алгоритми без необхідності їх реалізації.

Оскільки Python є знайомою платформою, легко знайти Python розробників, які приєднаються до команди. Тому нові розробники можуть швидко ознайомитися з концепціями Python і одразу ж почати працювати над проектами. Більшість вчених прийняли Python для проектів машинного навчання та глибокого навчання, а це означає, що більшість найяскравіших розумів світу можна знайти в спільнотах Python [15].

Продуктивність: деякі розробники стверджують, що Python відносно повільний порівняно з іншими мовами програмування. Хоча швидкість не є однією з сильних сторін Python, він пропонує рішення під назвою Cython. Це наднабір мови Python, розроблений для досягнення продуктивності коду так само, як і мова C.

Розробники можуть використовувати Cython для кодування розширень C

так само, як вони кодують у Python, оскільки синтаксис майже той самий. Cython значно підвищує продуктивність мови.

Інструменти візуалізації: Python поставляється з різноманітними бібліотеками. Деякі з цих фреймворків забезпечують хороші інструменти візуалізації. У ШІ, машинному навчанні та глибокому навчанні важливо подавати дані у форматі, зрозумілому людині. Тому Python є ідеальним вибором для реалізації цієї функції.

Деякі бібліотеки, такі як matplotlib, дозволяють науковцям з даних створювати діаграми, гістограми та графіки для кращого побудови та візуалізації даних. Крім того, різні API, підтримувані Python, покращують процес візуалізації.

3.2 Вибір бібліотек

3.2.1 Бібліотека Keras

Коли справа доходить до проектів машинного навчання та глибокого навчання, написаних на Python, є тисячі бібліотек на вибір. Але не всі вони мають однакову якість, різноманітність та розмір коду.

Python славиться своїм розмаїттям бібліотек. Іноді здається, що програмування на Python, яким ми його знаємо, вже ніколи не буде таким, як раніше.

Спільнота Python має топ-лист улюблених бібліотек для машинного навчання:

– numPy;

– sciPy;

– scikit-learn;

– theano;

– tensorFlow;

– keras;

– pyTorch;

– pandas;

– matplotlib;

– beautiful soup;

– scrapy;

– seaborn;

– pyCaret.

Навіть у зоні популярності глибоких нейронних мереж, складність основних фреймворків стала бар'єром для розробників, які тільки входять у сферу машинного навчання. Щоб вирішити цю проблему, було представлено кілька пропозицій щодо вдосконалених та спрощених API високого рівня для конструювання нейронних мереж. Ці пропозиції, як правило, демонструють схожість на поверхні, але виявляють відмінності при уважному розгляді. Один із таких інструментів – Keras, який виступає бібліотекою для мови програмування Python, спрямованою на глибоке машинне навчання. Keras дозволяє швидше створювати та налаштовувати моделі, але водночас не виконує складних математичних обчислень, функціонуючи як надбудова над іншими бібліотеками [16].

Крім того, Keras включає в себе розгорнуту документацію та посібники для розробників, і активно використовується серед топ-5 команд-переможців на Kaggle. Його використання спрощує проведення нових експериментів, надаючи можливість випробувати більше ідей, ніж у конкурентів, і робити це швидше. Keras становить ключову частину злагодженої екосистеми TensorFlow 2, яка охоплює всі аспекти робочого процесу машинного навчання, починаючи від керування даними і закінчуючи тренуванням гіперпараметрів та прийняттям рішень для розгортання [17].

3.2.2 Бібліотека NumPy

NumPy є важливою бібліотекою для математичних обчислень у мові програмування Python, і його повна назва — "Числові розширення Python". Однією з ключових переваг є відкритий вихідний код, розміщений на GitHub. Ця бібліотека, написана на мовах програмування C і Fortran, є компільованою, що призводить до високої швидкодії обчислень.

NumPy знаходить широке використання в наукових розрахунках, де вчені вирішують складні задачі з математики, фізики, біоінформатики та інших галузей. Вона служить основою для створення нових бібліотек, таких як Dask, CuPy та XND. У сфері науки про дані NumPy використовується на всіх етапах обробки даних, включаючи вилучення, трансформацію, аналіз, моделювання та візуалізацію.

Бібліотека також є основою для екосистеми машинного навчання, такої як Scikit-Learn і SciPy. Її потужність обчислень використовується в наукових, технічних та великих обчислювальних завданнях, забезпечуючи високу ефективність та можливості візуалізації даних порівняно з самою мовою програмування Python.

3.2.3 Бібліотека TensorFlow

TensorFlow, відкрита платформа для машинного навчання, стала визнаною лідером у світі глибокого навчання. Розроблена Google, вона використовує символічну математичну бібліотеку та потік даних для вирішення завдань, зосереджених на глибокому навчанні та висновках. TensorFlow дозволяє розробникам створювати програми машинного навчання, використовуючи різноманітні інструменти та ресурси.

Механізм роботи TensorFlow ґрунтується на створенні графів потоків даних для визначення, як дані переміщаються в графі. Вхідні дані, представлені як

тензори (багатовимірні масиви), проходять через граф, де виконуються математичні операції.

Архітектура TensorFlow включає обробку даних, побудову моделі, а також тренування та оцінку цієї моделі. Бібліотека отримала назву "TensorFlow" через використання тензорів для опису шляхів графа. Вона може працювати як на звичайних процесорах, так і на графічних процесорах, а також підтримує спеціалізований процесор Tensor TPU.

TensorFlow здобув популярність завдяки високому рівню абстракції, гнучкості, кросплатформеності та великій спільноті. Він є найпопулярнішим фреймворком глибокого навчання на GitHub.

Хоча TensorFlow має численні переваги, такі як інтерактивні можливості та високий рівень абстракції, він також має свої недоліки, зокрема, власні стандарти, високе споживання пам'яті та складність у вивченні. Встановлення TensorFlow вимагає використання pip та Anaconda для коректної роботи.

PyTorch і Apache MXNet конкурують з такими фреймворками, як TensorFlow, для класифікації рукописного тексту, розпізнавання зображень, вбудовування слів, рекурентних нейронних мереж, моделей послідовностей для машинного перекладу, обробки природної мови і моделювання PDE (диференціальних рівнянь в частинних похідних). Найкраще TensorFlow підтримує великомасштабні прогнози у виробничих умовах, використовуючи ті самі моделі, що й для навчання [18].

3.2.4 Бібліотека Matplotlib

Matplotlib - це бібліотека для мови програмування Python, яка використовується для створення візуалізацій та графіків. Вона є однією з найпопулярніших бібліотек для візуалізації даних через свою простоту використання та гнучкість. Матеріали Matplotlib можна використовувати як для вивчення данних, так і для створення якісних графіків для публікації.

Основні особливості Matplotlib:

- простота використання: Matplotlib надає простий та зрозумілий інтерфейс для створення графіків. З використанням кількох рядків коду можна побудувати різноманітні типи графіків;
- різноманітність типів графіків: Бібліотека підтримує широкий спектр типів графіків, таких як лінійні графіки, стовпчасті діаграми, гістограми, кругові діаграми, розсіювання та багато інших;
- контроль над деталями графіків: Matplotlib надає користувачеві повний контроль над деталями графіків. Від заголовків та міток до стилю ліній та кольорів – всі ці параметри можна легко налаштувати;
- підтримка LaTeX: Матеріали Matplotlib підтримують використання LaTeX для введення математичних символів та формул безпосередньо на графіках;
- широкі можливості конфігурації: Можливості конфігурації включають в себе налаштування розмірів графіків, шкал осей, кольорів, шрифтів та багато іншого.

3.2.5 Бібліотека Scikit-learn

Бібліотека машинного навчання (ML) для мови програмування Python, Scikit-Learn, пропонує різноманітні алгоритми, які можуть бути легко впроваджені програмістами та науковцями з обробки даних для створення моделей машинного навчання [19]. Scikit-Learn славиться своєю надійністю та популярністю, надаючи широкий спектр алгоритмів і інструментів для візуалізації машинного навчання, попередньої обробки, вибору моделі та оцінки її результатів.

Побудована на основі NumPy, SciPy та Matplotlib, Scikit-Learn включає ефективні алгоритми для класифікації, регресії та кластеризації. Деякі визначені алгоритми, які підтримує Scikit-Learn, включають машини опорних векторів,

дерева рішень, підвищення градієнту, k-середні та DBSCAN.

Бібліотека славиться своєю відносною простотою розробки завдяки послідовним API та детально розробленій документації для більшості алгоритмів, а також численним онлайн-підручникам. Scikit-Learn сумісна з популярними платформами, такими як Linux, MacOS і Windows [19].

Scikit-Learn стала де-факто стандартом для реалізації машинного навчання завдяки своєму зручному API, продуманому дизайну та активній спільноті. Бібліотека надає модулі для створення, підгонки та оцінки моделей машинного навчання, охоплюючи аспекти, такі як попередня обробка, класифікація, регресія, кластеризація, зменшення розмірності, вибір моделі та створення конвеєрів.

Ключові модулі в Scikit-Learn включають:

- попередня обробка – інструменти для вилучення ознак та нормалізації даних під час аналізу;
- класифікація – інструменти для ідентифікації категорій, пов'язаних з даними в моделі машинного навчання;
- регресія – створення моделі, яка намагається зрозуміти зв'язок між вхідними та вихідними даними;
- інструменти кластеризації – автоматично групують дані з схожими характеристиками в набори;
- зменшення розмірності – зменшення кількості випадкових величин для полегшення аналізу;
- вибір моделі – алгоритми та їхні здатності надавати інструменти для порівняння та вибору оптимальних параметрів для проектів машинного навчання;
- конвеєр – утиліти для конструювання робочого процесу моделі.

3.2.6 Бібліотека Pandas

Pandas – це бібліотека Python з відкритим вихідним кодом, яку широко

використовують для роботи з даними у науці про дані, аналізі даних та машинному навчанні. Вона ґрунтується на Numpy і надає підтримку багатовимірних масивів. Завдяки своїй популярності, Pandas інтегрується з іншими модулями науки про дані в екосистемі Python і зазвичай входить в різноманітні дистрибутиви Python, включаючи ті, що входять до складу операційних систем або комерційних постачальників, таких як ActivePython від ActiveState [20].

DataFrames в Pandas використовуються для виконання різноманітних завдань з обробки даних, таких як очищення, заповнення, нормалізація, об'єднання та об'єднання, візуалізація, статистичний аналіз, перевірка, завантаження та збереження. Завдяки Pandas можна легко виконувати складні завдання з аналізу та маніпулювання даними, що робить її високо оціненою серед науковців з даних.

3.2.7 Бібліотека OpenCV

OpenCV – це бібліотека програмного забезпечення з відкритим вихідним кодом для комп'ютерного зору та машинного навчання. Повна форма OpenCV – Бібліотека комп'ютерного зору з відкритим вихідним кодом. Вона була створена, щоб забезпечити спільну інфраструктуру для додатків комп'ютерного зору і прискорити використання машинного сприйняття в споживчих продуктах. OpenCV, як програмне забезпечення з ліцензією BSD, спрощує для компаній використання та зміну коду. Існує декілька попередньо визначених пакетів та бібліотек, які спрощують наше життя, і OpenCV є однією з них. Гері Бредскі винайшов OpenCV у 1999 році, і невдовзі вийшов перший реліз у 2000 році. Ця бібліотека базується на оптимізованому C/C++ і підтримує Java та Python разом з C++ через інтерфейс.

Бібліотека містить понад 2500 оптимізованих алгоритмів, включаючи велику колекцію алгоритмів комп'ютерного зору та машинного навчання, як

класичних, так і найсучасніших. Використовуючи OpenCV, стає легко виконувати складні завдання, такі як ідентифікація та розпізнавання облич, ідентифікація об'єктів, класифікація дій людини на відео, відстеження руху камери, відстеження рухомих об'єктів, вилучення 3D-моделей об'єктів, генерування 3D-хмар точок зі стереокамер, зшивання зображень для створення цілої сцени із зображенням високої роздільної здатності і багато іншого.

Python – це зручна мова, з якою легко працювати, але за цю перевагу доводиться платити швидкістю, оскільки Python працює повільніше, ніж такі мови, як C або C++. Тому ми розширюємо Python за допомогою C/C++, що дозволяє нам писати обчислювально інтенсивний код на C/C++ та створювати обгортки Python, які можна використовувати як модулі Python. При цьому код є швидким, оскільки він написаний в оригінальному C/C++ коді (оскільки це власне C++ код, що працює у фоновому режимі), а також, Python є більш легким для написання, ніж C/C++. OpenCV-Python – це Python-обгортка для оригінальної C++ реалізації OpenCV [21].

3.3 Вибір набору даних

Для вирішення завдання з розпізнавання дорожніх знаків необхідно вирішити проблему класифікації. Дорожні знаки дуже різноманітні, мають різні розміри, кольори, текст, числа, форми, тощо. Підготована і упорядкована інформація у формі таблиці є необхідним набором даних для навчання нейронних мереж.

В якості вхідних даних я взяв GTRSB, з платформи Kaggle [22]. Цей набір даних містить усю необхідну інформацію для дослідів, та чудово підходить для навчання нейронних мереж та інших алгоритмів класифікації. В ньому присутні:

- 43 різних класи дорожніх знаків, які необхідно розпізнати (рисунок 3.1);
- загальна кількість зображень в цьому наборі даних перевищує 50 000,

що робить його достатньо ємним;

- ця база даних містить реальні зображення дорожніх знаків, що робить її корисною для навчання та тестування алгоритмів машинного навчання в галузі розпізнавання дорожніх знаків.

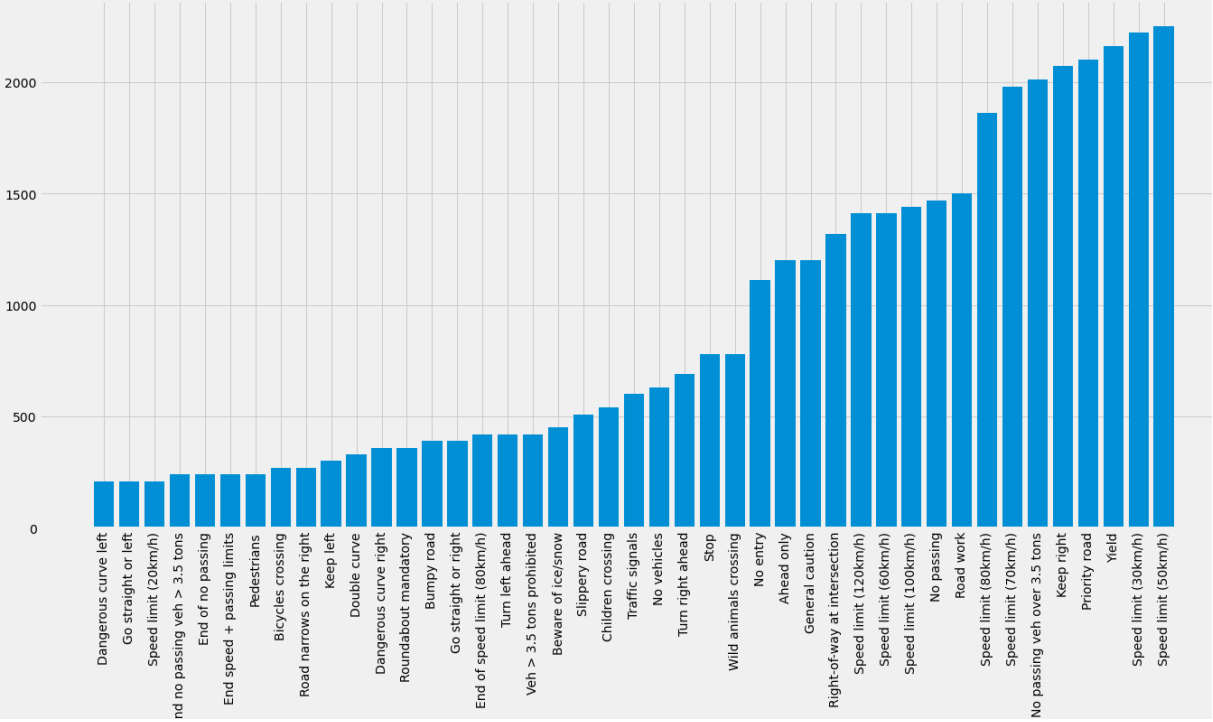

Рисунок 3.1 – Кількість класів та зображень у наборі GTRSB

Аналізуючи завдання розпізнавання дорожніх знаків, важливо розуміти, що це складна задача класифікації із незбалансованими частотами класів. Різноманітність дорожніх знаків може призводити до значного розмаїття між класами, оскільки вони можуть відрізнятися за кольором, формою та наявністю піктограм або тексту. Додатково, може виникнути потреба виділити підмножини класів з характерними загальними рисами, такими як знаки обмеження швидкості. Опрацьована та структурована інформація у табличному вигляді є необхідною для машинного навчання. Стовпці в такій таблиці представляють ознаки, а рядки – об'єкти. Виділяють два види ознак: залежні змінні (цільові ознаки) та незалежні змінні (предиктори).

Цей опис характерний для задач класифікації, де вибірка представляє собою кінцевий набір об'єктів, і відомо, до яких класів вони належать. Модель, побудована під час процесу машинного навчання, здатна класифікувати об'єкти з початкової множини.

4 ОРГАНІЗАЦІЯ ТА ПРОВЕДЕННЯ ЕКСПЕРИМЕНТІВ

4.1 Середовище розробки

Для подальшої розробки було обрано безкоштовну платформу Google Collab.

Google Colab (Colaboratory) - це безкоштовна платформа для виконання коду Python у хмарному середовищі, яка надає вам можливість писати та виконувати код, особливо у сфері машинного навчання. Основні переваги використання Google Colab для розробки нейронних мереж включають:

- безкоштовний доступ до обчислювальних ресурсів - Google Colab надає безкоштовний доступ до GPU (графічних процесорів) та TPU (тензорних процесорів) в хмарному середовищі, що дозволяє прискорити тренування нейронних мереж;
- легкий доступ до бібліотек машинного навчання - Встановлено багато популярних бібліотек, таких як TensorFlow, PyTorch, Keras, OpenCV, і багато інших, що спрощує розробку та експерименти з нейронними мережами;
- спільна робота та обмін даними - Легко ділитися ноутбуками, що має велике значення для спільної роботи та отримання зворотного зв'язку від колег чи співробітників;
- інтеграція з Google Drive - Можливість інтеграції з Google Drive дозволяє зберігати та обмінюватися ноутбуками та даними, забезпечуючи стійкість до втрати інформації;
- використання Jupyter ноутбуків - Інтерфейс Google Colab базується на Jupyter ноутбуках, що дозволяє вам легко писати та виконувати код частинами, а також вставляти текстові коментарі та зображення
- широкі можливості налаштувань - Наявність обчислювальних ресурсів різної потужності, можливість встановлення додаткових бібліотек та пакетів, що розширює функціональність;

Зручний доступ до даних - Завдяки інтеграції з Google Drive, ви легко можете отримати доступ до даних, необхідних для тренування нейронних мереж.

4.2 Навчання нейронних мереж

Гіперпараметри нейронної мережі - це параметри, які визначаються перед початком навчання. Серед них входять:

- кількість шарів у нейронній мережі;
- тривалість навчання (кількість епох);
- вибір функцій активації для кожного шару;
- кількість нейронів у кожному шарі;
- вибір оптимізатора для навчання нейронної мережі**.**

Оскільки кожен гіперпараметр може впливати на результат навчання по-різному, визначення їх оптимальних значень є важливим завданням. Розробник повинен експериментувати з гіперпараметрами, вносячи зміни на розсуд, і аналізувати, як це впливає на результати навчання нейронної мережі.

На рисунку 4.1 зображено результат експериментів з гіперпараметрами. З цього можна дізнатися, що:

- у сумарному, 13 шарів;
- оптимальна кількість епох обрана у кількості 30-ти;
- функція активації була обрана ReLu, для усіх згорткових шарів та для одного повнозв’язкового шару класифікації, лише вихідний повнозв’язковий шар працює на функціїї активації softmax;
- оптимізатора Adam.

Оптимізація Адама - це метод стохастичного градієнтного спуску, який базується на адаптивному оцінюванні моментів першого та другого порядку.

Згідно з Kingma, 2014, метод є "обчислювально ефективним, має невеликі вимоги до пам'яті, інваріантний до діагонального масштабування градієнтів і добре підходить для задач з великими обсягами даних/параметрів" [23].

```
# Створюємо послідовну модель
model = Sequential()
# Перший згортковий шар
model.add(Conv2D(filters=16, kernel_size=(3,3), activation='relu',
                 input_shape=(image_height,image_width,channels)))
# Другий згортковий шар
model.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu'))
# Перший шар підвиборки
model.add(MaxPool2D(pool_size=(2, 2)))
# Шар нормалізації
model.add(BatchNormalization(axis=-1))
# Третій згортковий шар
model.add(Conv2D(filters=64, kernel_size=(3,3), activation='relu'))
# Четвертий згортковий шар
model.add(Conv2D(filters=128, kernel_size=(3,3), activation='relu'))
# Другий шар підвиборки
model.add(MaxPool2D(pool_size=(2, 2)))
# Шар нормалізації
model.add(BatchNormalization(axis=-1))
# Шар перетворення даних з 2D подання в плоске
model.add(Flatten())
# Повнозв'язковий шар для класифікації
model.add(Dense(512, activation='relu'))
# Шар нормалізації
model.add(BatchNormalization())
# Шар регуляризації Dropout
model.add(Dropout(0.5))
# Вихідний повнозв'язковий шар
model.add(Dense(classes_amount, activation='softmax'))
```

Рисунок 4.1 – Гіперпараметри мережі 13

Різноманітні дослідження довели ефективність згорткових нейронних мереж у багатьох сферах комп'ютерного зору, включаючи розпізнавання зображень. Найкращі результати в задачах обробки та розпізнавання зображень показали згорткові нейронні мережі, тож розглянемо окремі з них, які обрані для проведення досліджень [24].

Для вирішення завдань з розпізнавання дорожніх знаків створено авторську нейронну мережу з умовною назвою 13, яка складається з тринадцяти шарів. Нейронна мережа приймає зображення розміром 32x32 пікселів, що мають три кольорових каналів. На вході має два шари для згортки, один шар пулінгу, один шар нормалізації, ще два шари згортки, один шар пулінгу, ще шар нормалізації. Після цього йде шар конвертації з 2D подання, до плоского виду, за ним повнозв’язний шар та шар нормалізації з регуляризацією. На виході повнозв’язний шар з 43-ма нейронами, що відповідає кількості класів в обраному

наборі даних. На рисунку 4.2 наведена архітектура розробленої нейромережі [24].

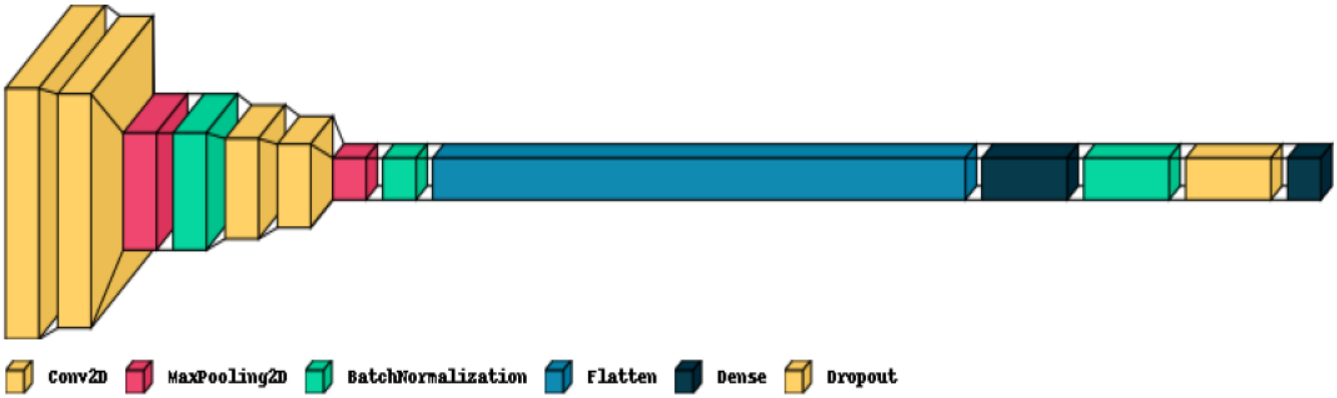

Рисунок 4.2 – Модель нейронної мережі 13

На рисунку 4.3 наведена архітектура цієї мережі. Output Shape означає форму виходу, вона визначає розмір та структуру вихідних даних після проходження вхідних даних через нейронну мережу. Кожен шар мережі може мати свою власну форму виходу, і це впливає на структуру та форму вхідних даних для наступних шарів. Шар Input має три параметри, окрім None. None у розмірності партії (batch dimension), це зазвичай означає, що розмір партії не є фіксованим і може змінюватися. Під час навчання можна використовувати різні розміри партій, і None дозволяє моделі обробляти змінні розміри партій. Три інші параметри це висота зображення у пікселях, його ширина та кількість каналів кольорів. Шар Convolutional має 16 фільтрів з розміром ядра (3,3) і використовує функцію активації ReLu. Результуючу фігуру після застосування цієї операції згортки можна обчислити наступним чином:

$$output\ height = \frac{\text{input height} - \text{kernel height} + 2 \times \text{padding}}{\text{strides}} + 1 \qquad (4.1)$$

$$output\ width = \frac{\text{input width} - \text{kernel width} + 2 \times \text{padding}}{\text{strides}} + 1 \quad , \qquad (4.2)$$

де висота входу (input_height) =32;

ширина входу (input_width) = 32;

висота ядра (kernel_height) = 3;

ширина ядра (kernel_width) = 3;

пробіл (padding) = 0, якщо його не вказано;

Strides зазвичай = 1, якщо не вказано.

Обчисливши за формулами (4.1) та (4.2) отримаємо відповідно значення 30 і 30. Таким чином, результуюча форма після виконання наданого шару Convulational має вигляд (None, 30, 30, 16).

Форма виходу останнього шару часто визначається кількістю класів у задачі класифікації. Ще один стовпець під назвою Param – це параметри. У контексті нейронних мереж "параметри" зазвичай вказують на ваги та зсуви (biases), які модель навчається оптимізувати під час процесу навчання. Ваги визначають силу зв'язків між нейронами у різних шарах мережі, тоді як зсуви додають константне значення до взваженої суми входів. Кількість параметрів у моделі прямо впливає на її складність та здатність адаптуватися до різних вхідних даних. Зазвичай, більша кількість параметрів дозволяє моделі вивчати більш складні залежності, але може також призвести до перенавчання. Деякі шари у Param, не мають параметрів. Сам шар Max Pooling не має параметрів, що навчаються. На відміну від згорткових шарів або повнозв'язних шарів, які мають ваги та зміщення, які потрібно вивчати під час навчання, операція Max Pooling є фіксованою і визначається розміром об'єднання та кроками. Шар Flatten і шар Dropout, як і шар Max Pooling, не мають параметрів, що навчаються, оскільки вони не передбачають навчання вагових коефіцієнтів у процесі навчання.

Layer		Output Shape	Param
Input	Image	(None, 32, 32, 3)	0
1	Convolutional	(None, 30, 30, 16)	448
2	Convolutional	(None, 28, 28, 32)	4640
3	Max Pooling	(None, 14, 14, 32)	0
4	Batch Normalization	(None, 14, 14, 32)	128
5	Convolutional	(None, 12, 12, 64)	18 496
6	Convolutional	(None, 10, 10, 128)	73 856
7	Max Pooling	(None, 5, 5, 128)	0
8	Batch Normalization	(None, 5, 5, 128)	512
9	Flatten	(None, 3200)	0
10	Dense	(None, 512)	1 638 912
11	Batch Normalization	(None, 512)	2048
12	Dropout	(None, 512)	0
Output	Dense	(None, 43)	22059
Сума параметрів:			1 761 099

Рисунок 4.3 – Архітектура нейронної мережі 13

Нейронна мережа VGG-19 (Visual Geometry Group-19) - це глибока згорткова нейронна мережа, розроблена для завдань визначення об'єктів та класифікації зображень. Вона є частиною сімейства моделей VGG, яке було представлено командою зорової геометрії при Університеті Оксфорда. Назва "19" вказує на загальну кількість шарів (включаючи згорткові та пов'язані) у цій мережі.

VGG-19 складається з 19 шарів, включаючи 16 згорткових шарів і 3 пов'язані шари.

Згорткові шари використовують фільтри розміром 3x3 пікселі, завдяки чому отримується менше параметрів і забезпечується більш ефективна навчання.

Пов'язані шари включають два повністю з'єднані шари з 4096 нейронами та один остаточний повністю з'єднаний шар з кількістю вихідних класів (зазвичай 1000, як у зображеннях ImageNet).

Щоб зменшити просторові розміри зображення, VGG-19 використовує згорткові шари зі стрибками (stride) 2.

Використання згорткових шарів зі стрибками допомагає послідовно

зменшувати розмір зображення, зберігаючи важливу інформацію.

В якості функцій активації використовуються ReLU (Rectified Linear Unit) на всіх шарах, що допомагає уникнути проблеми з виципленням градієнтів.

Мережа VGG-19 відома своєю глибокою структурою, що робить її обчислювально витратною, особливо при використанні на великих зображеннях або на ресурсозберігаючих пристроях.

Зазвичай VGG-19 використовується для передбачення об'єктів на зображеннях, класифікації та визначення областей із зображеннями.

Навчені ваги VGG-19 часто використовуються як основа для подальших завдань передбачення, передавання навчання та витягання ознак.

Загальною ідеєю VGG-19 є створення глибокої мережі з невеликою кількістю типових шарів, що дозволяє їй взяти участь у конкуренції на облаштування найкращих моделей у класифікації зображень.

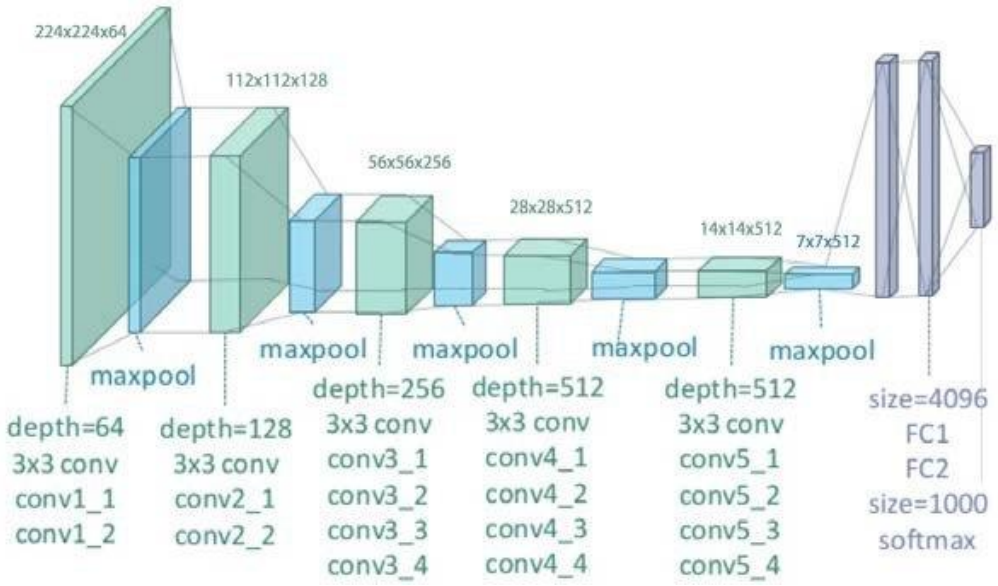

Рисунок 4.4 – Модель нейронної мережі VGG-19

LeNet-5 — це згорткова нейронна мережа, розроблена Яном ЛеКуном та його командою в 1998 році. Ця архітектура зазвичай визначається як одна з перших успішних моделей для розпізнавання образів та знаків на чеках банків.

Нижче подано деталі щодо LeNet-5 [25].

LeNet-5 складається з трьох основних блоків. Перший блок включає два шари згорткових нейронів для виявлення локальних особливостей в зображеннях. Використовуються фільтри згортки для витягання різних аспектів зображення. Слідом за блоком згорток іде блок підсумовування, який зменшує розмірність зображення, зберігаючи важливі інформаційні ознаки. Завершальний блок є повністю з'єднаним, де використовуються повністю з'єднані шари для класифікації отриманих ознак [25].

LeNet-5 використовує гіперболічний тангенс і сигмоїдні функції активації на різних етапах мережі. Це допомагає уникнути проблем, таких як зниклий або вибухаючий градієнт, і забезпечити стабільність навчання.

LeNet-5 в першу чергу розроблялася для розпізнавання рукописних знаків та символів на чеках. Вона відіграла важливу роль у визначенні та створенні підґрунтя для подальшого розвитку згорткових нейронних мереж у сфері комп'ютерного зору [25].

LeNet-5, як одна з перших успішних згорткових нейронних мереж, визначила основи для подальших досліджень та застосувань у галузі машинного навчання.

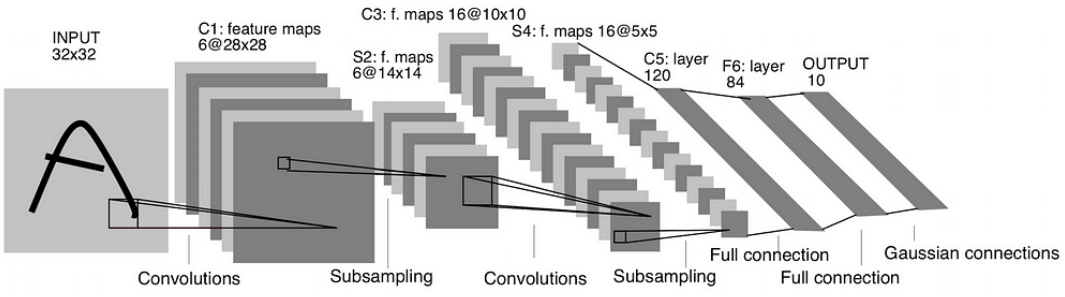

Рисунок 4.5 – Модель нейронної мережі LeNet-5

Всі нейронні мережі були навчені в однакових умовах для забезпечення можливості їх порівняння. Процес навчання включав 30 епох (рисунок 3.6), при

цьому вибірки були сформовані в співвідношенні 70% для навчання та 30% для перевірки. Розмір вибірки склав 858 зображень. Навчання нейронної мережі виконувалося з використанням графічного процесора Tesla K80. Після завершення процесу навчання нейронні мережі були збережені для подальшого використання.

```
[ ] historyVGG19 = modelVGG19.fit(aug.flow(X_train, y_train, batch_size=batch_size), epochs=epochs, validation_data=(X_val, y_val))
Epoch 2/30
858/858 [==============================] - 34s 40ms/step - loss: 2.3807 - accuracy: 0.2202 - val_loss: 3.0942 - val_accuracy: 0.1000
Epoch 3/30
858/858 [==============================] - 34s 40ms/step - loss: 1.7403 - accuracy: 0.4071 - val_loss: 1.1058 - val_accuracy: 0.6173
Epoch 4/30
858/858 [==============================] - 34s 40ms/step - loss: 1.1272 - accuracy: 0.6026 - val_loss: 0.7604 - val_accuracy: 0.7449
Epoch 5/30
858/858 [==============================] - 35s 41ms/step - loss: 0.8236 - accuracy: 0.7094 - val_loss: 0.5015 - val_accuracy: 0.8217
Epoch 6/30
858/858 [==============================] - 34s 40ms/step - loss: 0.6386 - accuracy: 0.7759 - val_loss: 0.5103 - val_accuracy: 0.8247
Epoch 7/30
858/858 [==============================] - 34s 40ms/step - loss: 0.4908 - accuracy: 0.8307 - val_loss: 0.2643 - val_accuracy: 0.9060
Epoch 8/30
858/858 [==============================] - 34s 40ms/step - loss: 0.3773 - accuracy: 0.8713 - val_loss: 0.1334 - val_accuracy: 0.9603
Epoch 9/30
858/858 [==============================] - 34s 40ms/step - loss: 0.3216 - accuracy: 0.8898 - val_loss: 0.2951 - val_accuracy: 0.9254
Epoch 10/30
858/858 [==============================] - 34s 40ms/step - loss: 0.2772 - accuracy: 0.9079 - val_loss: 0.1488 - val_accuracy: 0.9540
Epoch 11/30
858/858 [==============================] - 34s 40ms/step - loss: 0.2315 - accuracy: 0.9247 - val_loss: 0.0706 - val_accuracy: 0.9780
Epoch 12/30
858/858 [==============================] - 34s 40ms/step - loss: 0.2021 - accuracy: 0.9323 - val_loss: 0.1125 - val_accuracy: 0.9637
Epoch 13/30
858/858 [==============================] - 34s 40ms/step - loss: 0.1932 - accuracy: 0.9375 - val_loss: 0.0530 - val_accuracy: 0.9842
Epoch 14/30
858/858 [==============================] - 36s 41ms/step - loss: 0.1433 - accuracy: 0.9556 - val_loss: 0.0600 - val_accuracy: 0.9833
Epoch 15/30
858/858 [==============================] - 34s 40ms/step - loss: 0.1505 - accuracy: 0.9529 - val_loss: 0.1082 - val_accuracy: 0.9622
Epoch 16/30
858/858 [==============================] - 34s 40ms/step - loss: 0.1215 - accuracy: 0.9628 - val_loss: 0.0680 - val_accuracy: 0.9826
Epoch 17/30
858/858 [==============================] - 36s 42ms/step - loss: 0.1164 - accuracy: 0.9651 - val_loss: 0.0404 - val_accuracy: 0.9874
Epoch 18/30
858/858 [==============================] - 34s 40ms/step - loss: 0.1119 - accuracy: 0.9678 - val_loss: 0.0834 - val_accuracy: 0.9747
Epoch 19/30
858/858 [==============================] - 34s 40ms/step - loss: 0.0924 - accuracy: 0.9727 - val_loss: 0.0851 - val_accuracy: 0.9759
Epoch 20/30
858/858 [==============================] - 35s 40ms/step - loss: 0.0890 - accuracy: 0.9742 - val_loss: 0.0384 - val_accuracy: 0.9928
Epoch 21/30
858/858 [==============================] - 34s 40ms/step - loss: 0.0730 - accuracy: 0.9787 - val_loss: 0.0363 - val_accuracy: 0.9906
Epoch 22/30
858/858 [==============================] - 34s 40ms/step - loss: 0.0803 - accuracy: 0.9765 - val_loss: 0.0267 - val_accuracy: 0.9930
Epoch 23/30
858/858 [==============================] - 37s 43ms/step - loss: 0.0722 - accuracy: 0.9804 - val_loss: 0.0168 - val_accuracy: 0.9963
Epoch 24/30
858/858 [==============================] - 34s 40ms/step - loss: 0.0572 - accuracy: 0.9835 - val_loss: 0.0337 - val_accuracy: 0.9934
Epoch 25/30
858/858 [==============================] - 34s 40ms/step - loss: 0.0779 - accuracy: 0.9798 - val_loss: 0.0497 - val_accuracy: 0.9861
Epoch 26/30
858/858 [==============================] - 36s 42ms/step - loss: 0.0439 - accuracy: 0.9879 - val_loss: 0.0161 - val_accuracy: 0.9978
Epoch 27/30
858/858 [==============================] - 34s 40ms/step - loss: 0.0446 - accuracy: 0.9882 - val_loss: 0.0693 - val_accuracy: 0.9793
Epoch 28/30
858/858 [==============================] - 34s 40ms/step - loss: 0.0504 - accuracy: 0.9850 - val_loss: 0.0247 - val_accuracy: 0.9935
Epoch 29/30
858/858 [==============================] - 35s 40ms/step - loss: 0.0437 - accuracy: 0.9876 - val_loss: 0.0488 - val_accuracy: 0.9870
Epoch 30/30
858/858 [==============================] - 34s 40ms/step - loss: 0.0441 - accuracy: 0.9881 - val_loss: 0.0131 - val_accuracy: 0.9974
```

Рисунок 4.6 – Процес навчання мережі VGG-19 на платформі Google Collab

На наступному Рисунку 3.5 буде показано чому саме 30 епох було обрано для кожної нейронної мережі, на прикладі LeNet-5. Я зобразив графік історії

точності прогнозу кожної епохи навчання для цієї нейронної мережі.

Нейронна мережа із LeNet-5:

```
[ ]  pd.DataFrame(historyLN5.history).plot(figsize=(10, 5))
     plt.grid(True)
     plt.gca().set_ylim(0, 1)
     plt.show()
```

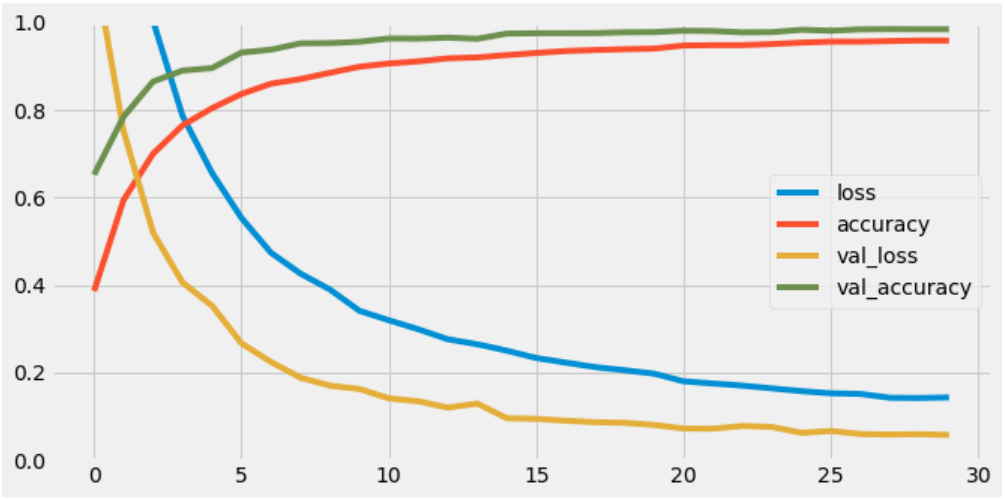

Рисунок 4.7 – Точність прогнозу кожної епохи навчання LeNet-5

Бібліотека TensorFlow зберігає навчену нейронну мережу у каталозі з вкладеними файлами. Інформація про архітектуру моделі та конфігурацію навчання (включаючи оптимізатор, функцію втрат та метрики) зберігається у файлі saved_model.pb. Цей файл містить саму програму або модель і набір іменованих сигнатур, кожна з яких ідентифікує функцію, яка обробляє тензорні вхідні дані і генерує тензорні вихідні дані. Ваги моделі зберігаються у каталозі variables/.

4.3 Оцінка впливу перешкод на якість розпізнавання

Далі потрібно провести тестування навчених нейронних мереж, використовуючи тестову вибірку та метрику точності (accuracy score) з бібліотеки

scikit-learn. Ця метрика приймає фактичні та прогнозовані мітки та повертає точність прогнозу.

Для оцінки роботи моделей розділімо наші вибірки на чотири сегменти:

- Істинно позитивний (TP) - коли фактична мітка = 1 і прогнозована мітка = 1
- Помилковий позитивний (FP) - коли фактична мітка = 0 і прогнозована мітка = 1
- Помилковий негативний (FN) - коли фактична мітка = 1 і прогнозована мітка = 0
- Істинно негативний (TN) - коли фактична мітка = 0 і прогнозована мітка = 0

Система може працювати некоректно у випадках, коли:

- слабкість освітлення фар, яке може бути викликана забрудненням чи відхиленням оптичної осі.
- лобове скло може бути забрудненим чи запітнілим;
- яскраве світло, спрямоване на передню частину автомобіля, наприклад, підсвічування або дальнє світло фар зустрічних автомобілів, також може впливати на видимість;
- різкий поворот автомобіля може створити додаткові труднощі;
- сильне світло, яке відбивається від дороги, може призводити до погіршення видимості;
- погодні умови, такі як дощ, туман або сніг, або забруднення можуть зробити зображення знаку нечітким;
- забруднення або сніг можуть приховати дорожні знаки;
- дерева чи інші транспортні засоби можуть перешкоджати відображенню світла від фар;
- дорожній знак може бути частково затінений або деформований;
- яскравість, розмір чи стан дорожніх знаків також можуть впливати на сприйняття інформації на дорозі;

- інші об'єкти, схожі на дорожні знаки, також можуть створювати певний ризик непорозуміння чи неправильної інтерпретації на дорозі.

Враховуючі ці фактори, буде доречним розглянути вплив різних шумів та перешкод на роботу навчених нейронних мереж. Було підготовлено декілька тестових вибірок з перешкодами.

```
data =[]
data_blur =[]
data_gaussian_blur =[]
data_gaussian_noise =[]
data_uniform_noise =[]
data_impulse_noise =[]

for img in imgs:
    try:
        image = cv2.imread(data_dir + '/' +img)[...,::-1]

        data.append(resizeAndTransformImageToNumpyArray(image))
        data_blur.append(createBlurImage(image))
        data_gaussian_blur.append(createGaussianBlurImage(image))
        data_gaussian_noise.append(createGaussianNoiseImage(image))
        data_uniform_noise.append(createUniformNoiseImage(image))
        data_impulse_noise.append(createImpulseNoiseImage(image))

    except:
        print("Error in " + img)

X_test = np.array(data)
X_blur_test = np.array(data_blur)
X_gaussian_blur_test = np.array(data_gaussian_blur)
X_gaussian_noise_test = np.array(data_gaussian_noise)
X_uniform_noise_test = np.array(data_uniform_noise)
X_impulse_noise_test = np.array(data_impulse_noise)

# Нормалізація тестових даних
X_test = X_test / 255
X_blur_test = X_blur_test / 255
X_gaussian_blur_test = X_gaussian_blur_test / 255
X_gaussian_noise_test = X_gaussian_noise_test / 255
```

Рисунок 4.8 – Тестові вибірки на платформі Google Collab

Потім ми проведемо оцінку ефективності наших нейронних мереж за допомогою метрики accuracy score, провівши їх тестування на шести різних тестових вибірках. Кожна з цих вибірок включає 12630 зображень різних дорожніх знаків, які не були включені до навчальної вибірки.

Перевіряємо якість роботи нейронної мережі із тринадцятьма шарами:

```
[ ]  # Стандартні зображення:
     pred = getTestDataPrediction(model, X_test, "Default")
     # Зображення із розмиванням:
     pred_blur = getTestDataPrediction(model, X_blur_test, "Blur")
     # Зображення із гаусовим розмиванням:
     pred_gaussian_blur = getTestDataPrediction(model, X_gaussian_blur_test, "Gaussian blur")
     # Зображення із гаусовим шумом:
     pred_gaussian_noise = getTestDataPrediction(model, X_gaussian_noise_test, "Gaussian noise")
     # Зображення із рівномірним шумом:
     pred_uniform_noise = getTestDataPrediction(model, X_uniform_noise_test, "Uniform noise")
     # Зображення із імпульсним шумом:
     pred_impulse_noise = getTestDataPrediction(model, X_impulse_noise_test, "Impulse noise")
```

```
395/395 [==============================] - 1s 2ms/step
[Default] Частка вірних відповідей на тестових даних, у відсотках: 98.33729216152018
395/395 [==============================] - 1s 2ms/step
[Blur] Частка вірних відповідей на тестових даних, у відсотках: 83.72129849564529
395/395 [==============================] - 1s 2ms/step
[Gaussian blur] Частка вірних відповідей на тестових даних, у відсотках: 96.270783847981
395/395 [==============================] - 1s 2ms/step
[Gaussian noise] Частка вірних відповідей на тестових даних, у відсотках: 67.11797307996832
395/395 [==============================] - 1s 2ms/step
[Uniform noise] Частка вірних відповідей на тестових даних, у відсотках: 62.62074425969912
395/395 [==============================] - 1s 2ms/step
[Impulse noise] Частка вірних відповідей на тестових даних, у відсотках: 91.0134600158353
```

Рисунок 4.9 – Оцінка якості нейронної мережі на прикладі мережі 13

Перевіряємо якість роботи нейронної мережі із архітектурою LeNet-5:

```
[ ]  # Стандартні зображення:
     predLN5 = getTestDataPrediction(modelLN5, X_test, "Default")
     # Зображення із розмиванням:
     predLN5_blur = getTestDataPrediction(modelLN5, X_blur_test, "Blur")
     # Зображення із гаусовим розмиванням:
     predLN5_gaussian_blur = getTestDataPrediction(modelLN5, X_gaussian_blur_test, "Gaussian blur")
     # Зображення із гаусовим шумом:
     predLN5_gaussian_noise = getTestDataPrediction(modelLN5, X_gaussian_noise_test, "Gaussian noise")
     # Зображення із рівномірним шумом:
     predLN5_uniform_noise = getTestDataPrediction(modelLN5, X_uniform_noise_test, "Uniform noise")
     # Зображення із імпульсним шумом:
     predLN5_impulse_noise = getTestDataPrediction(modelLN5, X_impulse_noise_test, "Impulse noise")
```

```
395/395 [==============================] - 1s 2ms/step
[Default] Частка вірних відповідей на тестових даних, у відсотках: 89.04988123515439
395/395 [==============================] - 1s 2ms/step
[Blur] Частка вірних відповідей на тестових даних, у відсотках: 82.55740300870941
395/395 [==============================] - 1s 2ms/step
[Gaussian blur] Частка вірних відповідей на тестових даних, у відсотках: 86.87252573238321
395/395 [==============================] - 1s 2ms/step
[Gaussian noise] Частка вірних відповідей на тестових даних, у відсотках: 34.069675376088675
395/395 [==============================] - 1s 2ms/step
[Uniform noise] Частка вірних відповідей на тестових даних, у відсотках: 34.84560570071259
395/395 [==============================] - 1s 2ms/step
[Impulse noise] Частка вірних відповідей на тестових даних, у відсотках: 77.31591448931117
```

Рисунок 4.10 – Оцінка якості нейронної мережі на прикладі мережі LeNet-5

Під час обчислювальних експериментів та тестування моделей отримано перші результати ефективності нейронних мереж. Тестова вибірка без змін зображення показала такі відсотки по точності: Мережа 13 – 98,33%, LeNet-5 – 89,04%, VGG-19 – 96,87%. Ці результати демонструють перевагу авторської нейронної мережі. Окрім цього, серед переваг можна зазначити як вагу файлу моделі – 310КБ, так й розмір файлу ваг, який складає для мережі 13 – 20 646КБ. VGG-19 при цьому має більші значення, а саме: файл моделі – 564КБ та розмір файлу ваг – 241 382КБ. LeNet-5 має обсяги 152КБ та 770КБ відповідно, але ціною цього є точність розпізнавання, яка буде ще зменшуватись з кожною модифікацією досліджуваних об’єктів [24].

4.4 Архітектура вебзастосунку

У цьому розділі розглядається архітектура майбутнього вебзастосунку. Для розробки використовуватиметься мова програмування Python, яка має потрібні бібліотеки для завантаження та використання навчених моделей нейронних мереж. Під час проектування вебзастосунку важливо врахувати сучасні архітектури, що пропонують веб-фреймворки. Популярні фреймворки для веб-розробки часто використовують архітектуру MVC. З урахуванням цього вибору, архітектура майбутнього вебзастосунку буде базуватися на MVC.

У вебзастосунку буде кілька структурних компонент з різними завданнями для відповіді на вимоги проекту. Така архітектура передбачає наявність трьох різних компонентів, які взаємодіють між собою для забезпечення ефективної роботи системи.

Модель-Вид-Контролер (MVC) - це архітектурна модель, яка використовується при проектуванні та розробці програмного забезпечення.

Ця модель розділяє систему на три частини: модель даних, представлення даних і управління. Вона використовується для відокремлення даних (моделі) від

користувацького інтерфейсу (представлення) таким чином, щоб зміни в користувацькому інтерфейсі мали мінімальний вплив на дані, а зміни в моделі даних можна було вносити без зміни користувацького інтерфейсу.

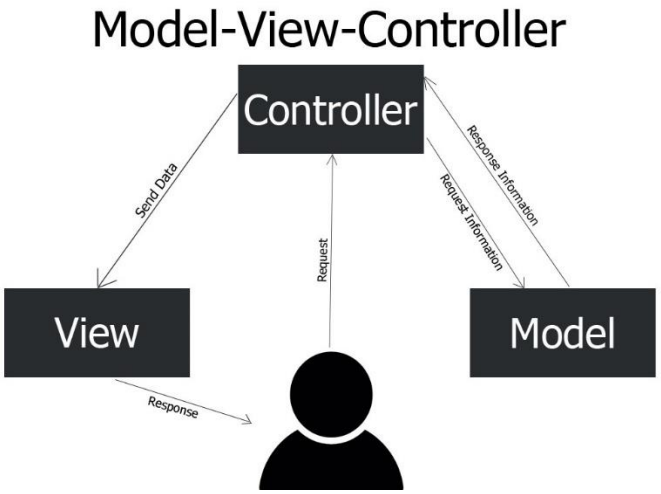

Рисунок 4.11 – Архітектура вебзастосунку

Мета цього шаблону - забезпечити гнучкий дизайн програмного забезпечення, полегшити подальші модифікації та розширення програми, а також уможливити повторне використання окремих компонентів програми. Крім того, використання цього шаблону у великих системах створює певний порядок у їхній структурі та зменшує їхню складність, роблячи їх легшими для розуміння.

Проаналізуємо компоненти додатка, розглядаючи їхні функції та взаємодію на прикладі.

Контролер (Controller) - Цей компонент відповідає за обробку запитів користувача, які надходять у вигляді HTTP-запитів (GET або POST). Основне призначення контролера - викликати та координувати дії різних ресурсів і об'єктів, необхідних для виконання користувацьких дій. Зазвичай контролер взаємодіє з моделлю для вирішення завдань та обирає відповідний вигляд для подальшого відображення результатів.

Модель (Model) - Модель включає в себе дані та правила для їх обробки. У

будь-якому додатку структура моделі моделює дані, які підлягають обробці за певними правилами. Модель представляє собою концепцію управління додатком, де дані обробляються згідно з встановленими правилами. Це включає в себе перевірку даних на відповідність правилам (наприклад, перевірка дати на необхідність вказівки минулого часу, формат електронної пошти, обмеження на довжину імені тощо). Модель надає контролеру представлення даних, які запитав користувач, і виконує важливу логіку додатку, вирішуючи конкретні завдання.

Вигляд (View) - Відповідальний за різні способи відображення даних, отриманих від моделі. Це може бути шаблон, який заповнюється даними, та інші різновиди представлення. В контексті вебзастосунку вид обирається контролером в залежності від поточної ситуації. Зазвичай вебзастосунок складається з набору контролерів, моделей та видів, і контролер може викликати інші контролери для виконання певних дій відповідно до обраної стратегії.

Така архітектура дозволяє добре організувати та взаємодіяти з компонентами додатку, розділяючи їх функціональні обов'язки та полегшуючи розширення та підтримку коду.

Використання концепції MVC надає нам кілька значущих переваг. Найочевидніша з них - це чіткий розділ між логікою представлення (інтерфейс користувача) та логікою додатка. Це стає особливо важливим у світі, де різні типи користувачів використовують різні пристрої. Забезпечення підтримки для різних пристроїв вимагає адаптації інтерфейсу, але при цьому модель залишається незмінною. Контролер вибирає різні види для відображення даних в залежності від типу пристрою, забезпечуючи адаптованість інтерфейсу [25].

Крім того, концепція MVC робить великі застосунки менш складними. Структурований код полегшує підтримку, тестування та повторне використання рішень. Ізоляція логіки представлення від логіки додатка сприяє покращенню організації коду та зменшенню його складності. В результаті використання концепції MVC видається досить обґрунтованим для досягнення більшої ефективності та зручності у розробці та підтримці програмного продукту [25].

4.5 Інтерфейс вебзастосунку

Необхідно розробити інтерфейс, який буде зручний для користувача. Для цього повторно проаналізував функціональні можливості розробленого додатку. Зробивши огляд всього функціоналу, який був розроблений до цього моменту, визначив, що для додатку необхідна форма для завантаження зображення та його подальшої обробки нейронною мережею. З урахуванням цього функціоналу був розроблений інтерфейс вебзастосунку, який має наступний вигляд.

Рисунок 4.12– Головна сторінка вебзастосунку

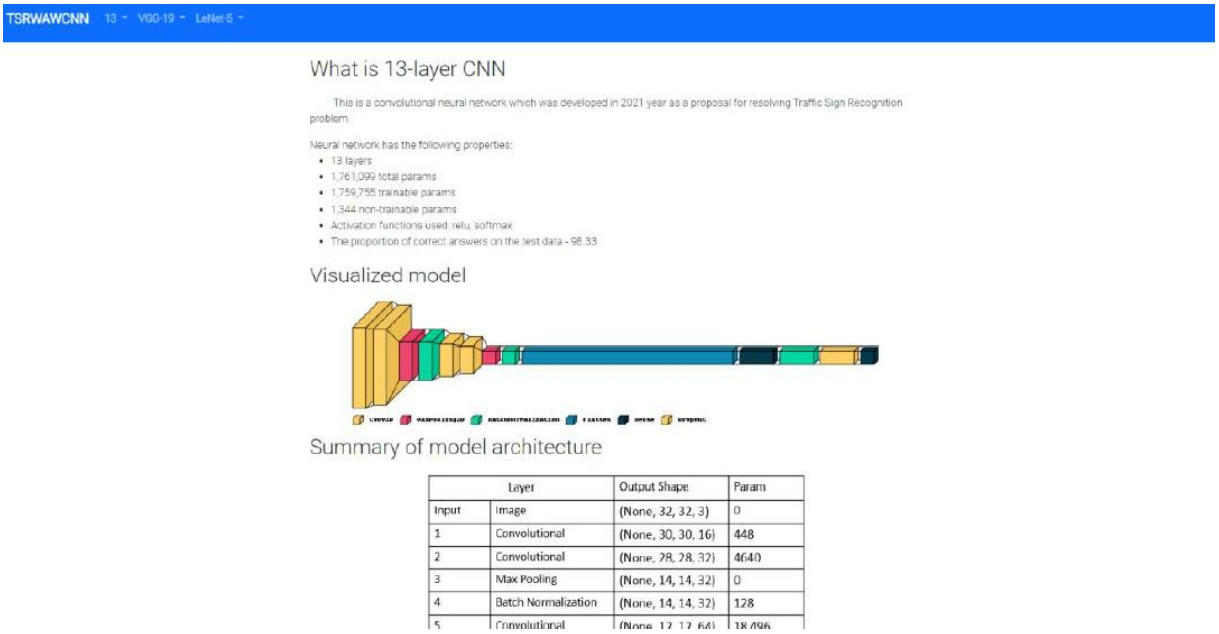

Рисунок 4.13 – Довідка про нейрону мережу 13

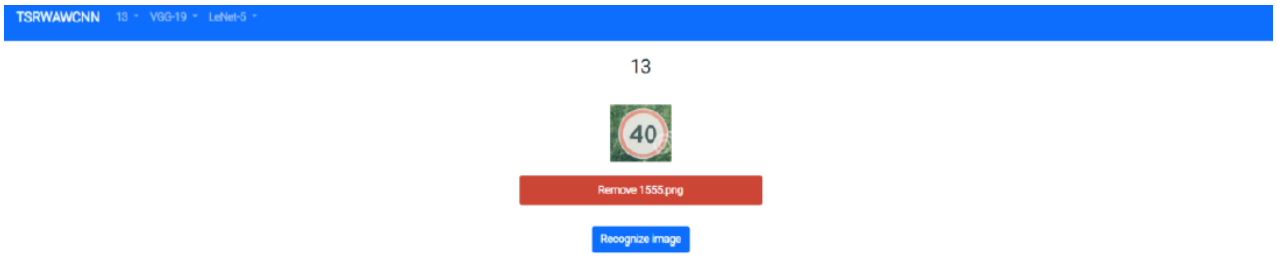

Рисунок 4.14 – Розділ завантаження зображень для нейронної мережі 13

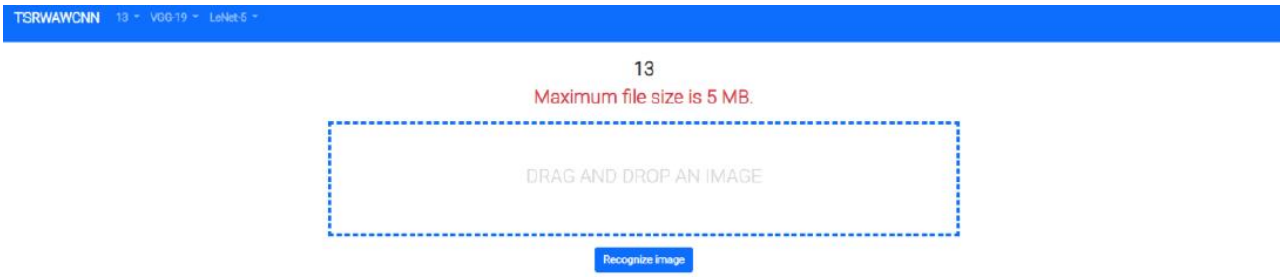

Рисунок 4.15 – Завантаження зображення вагою більше ніж 5МБ

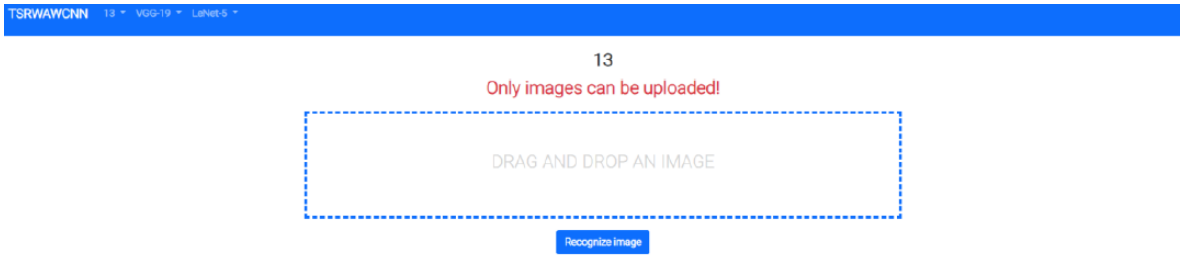

Рисунок 4.16 – Завантаження файлу, що не є зображенням

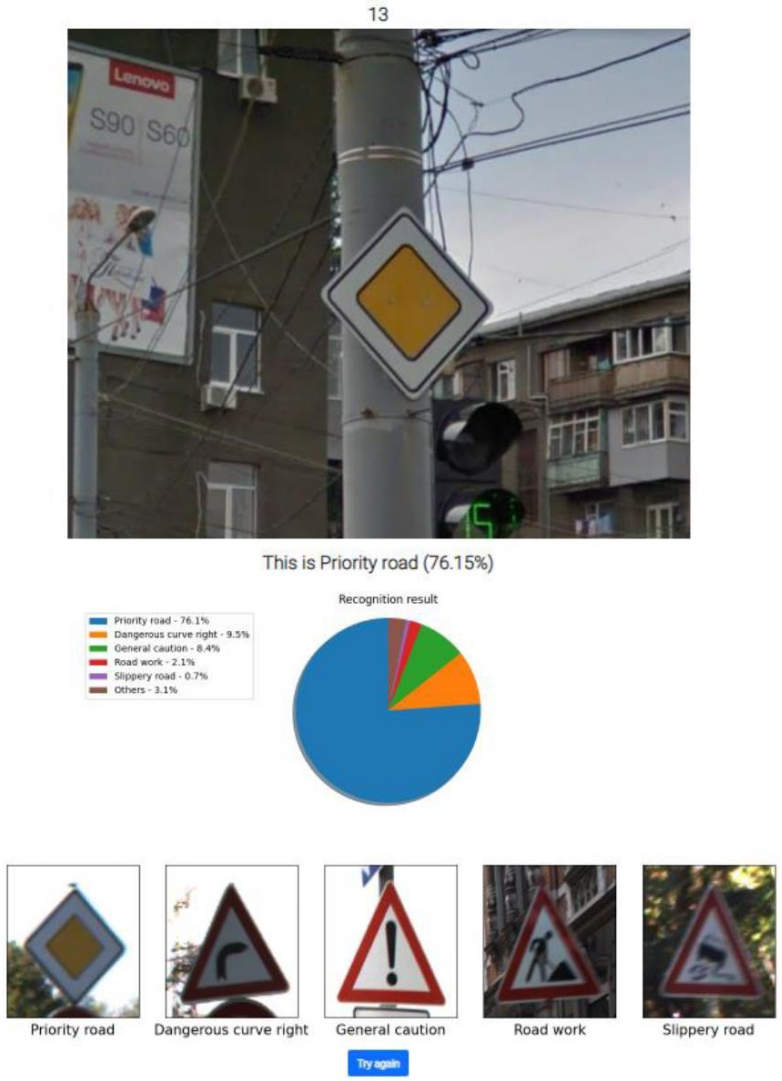

Рисунок 4.17 – Нейронна мережа 13 має декілька варіантів розпізнавання з відсотками ймовірності по кожному

Таблиця 4.1 – Результати оцінок нейронних мереж на модифікованих тестових вибірок при наявності певних перешкод, або їх відсутності

Тестова вибірка	13	VGG-19	LeNet-5
Без змін	98.33	96.87	89.04
Розмивання	83.72	80.50	82.55
Гаусове розмивання	96.27	93.68	86.87
Гаусовий шум	67.11	67.18	34.06
Рівномірний шум	62.62	63.04	34.84
Імпульсний шум	91.01	89.51	77.31

Коли модель демонструє ефективність на навчальній вибірці, але виявляється неефективною на тестовій вибірці (яка не брала участь у навчанні), виникає проблема перенавчання. Для подолання цього явища використовувались батч-нормалізація та регуляризація dropout.

Батч-нормалізація вирішує проблему ефективного навчання нейронних мереж, нормалізуючи вхідні дані так, щоб отримати нульове математичне очікування і одиничну дисперсію. Це допомагає уникнути серйозних розбіжностей у градієнтах між різними рівнями мережі.

Регуляризатор dropout допомагає уникнути ситуацій перенавчання, випадково виключаючи нейрони з мережі під час навчання. Це змушує мережу обробляти помилки та не покладатися на конкретні нейрони, сприяючи боротьбі із занадто сильним адаптивним вивченням шуму в даних.

Якщо нейронна мережа зробила правильний прогноз, то підпис під зображенням буде зеленим; якщо вона допустила помилку, то підпис буде червоним.

ВИСНОВКИ

У ході виконання даної кваліфікаційної роботи було проведено дослідження предметної області, розібрані види систем розпізнавання дорожніх знаків, визначені сфери у яких застосовується комп’ютерний зір ще. Проаналізовано вже існуючі системи у виробників авто, визначені їх принципи роботи. На основі цього, була сформульована постановка задачі, з метою створення вебзастосунку, який буде мати у собі навчені моделі нейронних мереж, здатних визначати зображення дорожніх знаків при наявності певних перешкод.

Для вирішення цієї задачі була обрана мова програмування Python та набір бібліотек для роботи з неї у сфері глибокого навчання нейронних мереж по розпізнаванню образів. В результаті аналізу архітектур нейронних мереж було визначено, що найбільш підходящим варіантом для завдання розпізнавання дорожніх знаків є використання згорткових нейронних мереж. Для вебзастосунку була обрана архітектура MVC. Після аналізу можливих проблем у роботі системи, навчені нейронні мережі протестовані на шести різних наборах для тестування. Результати нейронної мережі 13 авторської розробки показала у середньому більше відсоток відповідності, ніж дві інші, що були обрані у якості порівняння.

Оцінювання виявило критичну залежність навчених нейронних мереж від цілісності, коректності та чистоти зображення.

Для подальшої модернізації можна розглядати навчання нейронної мережі для завдання сегментації, додавання нових нейронних мереж та інтеграцію їх у вебзастосунок, введення інструкцій користувача, а також можливість локалізації вебзастосунка. Додатково можна розглядати інтеграцію навченої нейронної мережі в апаратне рішення..

Як висновок з даного дослідження можна сказати впевнено що використання методів машинного навчання має великі перспективи не тільки у

сфері транспортних засобів, а й у сфері медицини, охорони та безпеки та інше. Теорія розпізнавання образів має дуже великий спектр використання.

Результати досліджень доповідались на 12-й Міжнародній наук.-техн. конф. «Інформаційні системи та технології» (Харків, 28 листопада – 01 грудня 2023 р.).

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. IBM. What is computer vision? URL: https://www.ibm.com/topics/computer-vision (дата звернення: 21.10.2023).
2. Journal of Hematology & Oncology: Novel research and future prospects of artificial intelligence in cancer diagnosis and treatment URL: https://jhoonline.biomedcentral.com/articles/10.1186/s13045-023-01514-5 (дата звернення 21.10.2023)
3. Mubi: City Safety – system automatycznego hamowania od Volvo URL: https://mubi.pl/poradniki/city-safety/ (дата звернення 21.10.2023)
4. Ar5iv labs: HyperFace: A Deep Multi-task Learning Framework for Face Detection, Landmark Localization, Pose Estimation, and Gender Recognition URL: https://ar5iv.labs.arxiv.org/html/1603.01249 (дата звернення 04.11.2023)
5. В. Я. Кутковецький Розпізнавання образів: навч.посіб. / В. Я. Кутковецький. – Миколаїв : Вид-во ЧНУ ім. Петра Могили. – 2017. – 420 с.
6. Introduction to Random Forest in Machine Learning URL: https://www.section.io/engineering–education/introduction–to–random–forest–in–machine–learning/ (дата звернення: 14.11.2023).
7. J.D.Power. What is Traffic-Sign Recognition? URL: https://www.jdpower.com/cars/shopping-guides/what-is-traffic-sign-recognition (дата звернення: 14.11.2023).
8. Xenons4u. Mercedes Benz Traffic Sign Assist Inoperative Explained URL: https://xenons4u.co.uk/blog/mercedes-benz-traffic-sign-assist-inoperative-explained/ (дата звернення: 29.11.2023).
9. BMW. What is the Speed Limit Info in my BMW? URL: https://faq.bmw.com.au/s/article/Driver-assistance-systems-General-Safety-

Regulation-GSR-Speed-Limit-Info-Operating-principle-bSxxl?language=en_AU (дата звернення: 29.11.2023).

10. Tesla: Autopilot and Full Self-Driving Capability URL: https://www.tesla.com/support/autopilot (дата звернення 29.11.2023)
11. Dremio: Neural Network Architecture URL: https://www.dremio.com/wiki/neural-network-architecture/ (дата звернення 29.11.2023)
11. TechTarget: How do artificial neural networks work? URL: https://www.techtarget.com/searchenterpriseai/definition/neural-network (дата звернення 29.11.2023)
12. V7Labs: Neural Networks URL: https://www.v7labs.com/blog/neural-networks-activation-functions (дата звернення 29.11.2023)
13. Neurohive: Data Science URL: https://neurohive.io/ru/osnovy-data-science/7-arhitektur-nejronnyh-setej-nlp/ (Дата звернення 09.12.2023)
14. V7Labs: Neural Networks Architectures Guide URL: https://www.v7labs.com/blog/neural-network-architectures-guide (дата звернення 09.12.2023)
15. Why Python is Good for Machine Learning URL: https://www.section.io/engineering–education/why–python–is–good–for–machine–learning/ (дата звернення: 15.11.2023).
16. What is Keras? The deep neural network API explained URL: https://www.infoworld.com/article/3336192/what–is–keras–the–deep–neural–network–api–explained.html (дата звернення: 15.11.2023).
17. Keras URL: https://keras.io/ (дата звернення: 15.11.2023).
18. InfoWorld. What is TensorFlow? The machine learning library explained URL: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning-library-explained.html/ (дата звернення: 19.11.2023).
19. What Is Scikit-Learn? URL: https://www.nvidia.com/en–us/glossary/data–science/Scikit-Learn/ (дата звернення: 19.11.2023).

20. What Is Pandas in Python? Everything You Need to Know URL: https://www.activestate.com/resources/quick–reads/what–is–pandas–in–python–everything–you–need–to–know/ (дата звернення: 19.11.2023).
21. Great Learning. OpenCV Tutorial: A Guide to Learn OpenCV in PythonURL: https://www.mygreatlearning.com/blog/opencv-tutorial-in-python/ (дата звернення: 19.11.2023).
22. Kaggle: GTSRB - German Traffic Sign Recognition Benchmark URL: https://www.kaggle.com/datasets/meowmeowmeowmeowmeow/gtsrb-german-traffic-sign (дата звернення: 22.11.2023).
23. Keras: Adam URL: https://keras.io/api/optimizers/adam/ (дата звернення 09.12.2023)
24. Слатін М. Ю., Калита Н. І. Розроблення та дослідження нейронної мережі для розпізнавання дорожніх знаків.: тези доповідей 12-ї Міжнародної наук.-техн. конф. «Інформаційні системи та технології» (Харків, 28 листопада – 01 грудня 2023 р.). Харків: ХНУРЕ, 2023. с. 66
25. Medium: LeNet-5 своими руками URL: https://congyuzhou.medium.com/lenet-5-своими-руками-b60ae3727cd3 (дата звернення 09.12.2023)
26. SchoolboyProgrammer: Алгоритмізація та програмування URL: https://schoolboyprog10.blogspot.com/p/mvc-mvc-model-view-controller.html (дата звернення 12.12.2023)