Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Українсько-німецький навчально-науковий інститут
Кафедра кібербезпеки та програмного забезпечення

Далеченко Іван Васильович,
студент групи НРЗ-181

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Класифікація автомобільних марок за їхніми зображеннями

Спеціальність:
125 Кібербезпека

Спеціалізація, освітня програма:
Кібербезпека

Керівник:
Пенко Валерій Георгійович,
к.т.н., доцент

Одеса – 2022

Міністерство освіти і науки України
Національний університет «Одеська політехніка»
Українсько-німецький навчально-науковий інститут
Кафедра кібербезпеки та програмного забезпечення
Рівень вищої освіти перший (бакалаврський)
Спеціальність 125 – Кібербезпека
Освітня програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ
_____
д.т.н., проф. А.А.Кобозєва
____ _____2022р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Далеченку Івану Васильовичу

1.Тема роботи: *Класифікація автомобільних марок за їхніми зображеннями*
керівник роботи: *Пенко В.Г.. к.т.н., доцент*.
затверджені наказом ректора від „17” 05. 2022р. № 168-в.

2.Зміст роботи: *Аналіз предметної галузі, проектування розробляємої системи, тестування системи, охорона праці*

3.Перелік графічного матеріалу*: рисунки інтерфейсу програмного продукту, рисунки вибірок зображень*.

4. Консультанти розділів роботи

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Охорона праці	*доц. Ярова І.А.*		

Дата видачі завдання “_____”_____2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз літератури за темою кваліфікаційної роботи*	*02-03-2022*	*виконано*
2	*Аналіз існуючих алгоритмів вирішення завдання класифікації*	*22-03-2022*	*виконано*
3	*Аналіз та проектування перспективної конфігурації*	*06-04-2022*	*виконано*
4	*Реалізація алгоритмів*	*28-04-2022*	*виконано*
5	*Тестування алгоритмів*	*10-05-2022*	*виконано*
6	*Підготовка пояснювальної записки.*	*16-05-2022*	*виконано*
7	*Підготовка презентації та доповіді*	*23-05-2022*	*виконано*
8	*Попередній захист*	*02-06-2022*	*виконано*
9	*Нормоконтроль, рецензування*	*17.06.2022*	*виконано*

Здобувач вищої освіти _____ *Далеченко І.В.*

Керівник роботи _____ *Пенко В.Г.*

ЗАВДАННЯ

на розробку розділу «Охорона праці» у кваліфікаційній роботі бакалавра

студенту *Далеченку Івану Васильовичу*

Інститут інформаційної безпеки радіоелектроніки та телекомунікацій

Кафедра кібербезпеки та програмного забезпечення

Дата отримання завдання 13.04.2022

Консультації 16.05.2022, 23.05.2022

Дата закінчення розділу 16.06.2022

Тема роботи: *Класифікація автомобільних марок за їхніми зображеннями*

Зміст розділу

1. Аналіз умов праці і вибір основних заходів виробничої безпеки
2. Аналіз пожежної безпеки і вибір заходів і засобів пожежної безпеки

Керівник роботи _____ (В.Г. Пенко) «____» _____ 2022 р.

Консультант з охорони праці _____ (Ярова І.А.) «____» _____2022 р

АНОТАЦІЯ

Кваліфікаційна робота на тему “Класифікація автомобільних марок за їхніми зображеннями” на здобуття першого (бакалаврського) рівня вищої освіти за спеціальністю 125 - Кібербезпека, спеціалізація, освітня програма: Кібербезпека, містить 19 рисунків, 1 таблиця, 1 додаток, 10 літературних джерел за переліком посилань. Робота виконана на 49 сторінках загального тексту і 39 сторінках основного тексту.

Метою роботи є розробка автоматизованої програмної системи, що здійснює класифікацію марок автомобілів за їхніми зображеннями, за умови того, що зображення може бути пошкоджені або спотворені.

У роботі був виконаний огляд методів класифікації зображень та аналіз їх властивостей. Для реалізації виконання поставленої задачі кваліфікаційної роботи, було підхід на основі згорткових нейромереж.

У результати проведеної роботи було розроблено декілька варіантів згорткових нейронних мереж, з різною оцінкою якості їх роботи. Найкраща конфігурація досягла рівня 81% точності класифікації.

НЕЙРОННІ МЕРЕЖІ, ЗНМ, КЛАСИФІКАЦІЯ, МЕТОДИ ВИРІШЕННЯ ЗАДАЧІ КЛАСИФІКАЦІЇ.

ABSTRACT

Qualification work on "Classification of car brands by their images" for the first (bachelor's) level of higher education in specialty 125 - Cybersecurity, specialization, educational program: Cybersecurity, contains 19 figures, 1 table, 1 appendix, 10 references according to the list of references. The work is performed on 49 pages of general text and 39 pages of main text.

The goal of the work is to develop an automated software system that classifies car brands according to their images, provided that the images may be damaged or distorted.

The paper reviews the methods of image classification and analysis of their properties. To implement the task of qualification work, there was an approach based on convolutional neural networks.

As a result of this work, several variants of convolutional neural networks were developed, with different assessments of the quality of their work. The best configuration reached the level of 81% classification accuracy.

NEURAL NETWORKS, CNN, CLASSIFICATION, METHODS OF SOLVING THE CLASSIFICATION PROBLEM.

ЗМІСТ

ВСТУП 8

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ 10

1.1 Аналіз різних систем вирішення задачі класифікації 10

1.2 Обгрунтування застосування згорткової нейронної мережи 15

1.3 Згорткові нейронні мережі 15

2 ПРОЄКТУВАННЯ СИСТЕМИ 18

2.1 Збір даних для навчання 19

2.2 Підготовка даних 21

2.3 Вибір топології мережі 22

2.4 Експериментальний вибір характеристик мережі. 23

2.5. Навчання 25

3 ТЕСТУВАННЯ СИСТЕМИ 26

3.1 Оцінка якості роботи мережі 26

3.2 Поліпшення результатів шляхом зміни параметрів навчання 27

3.3 Поліпшення результатів шляхом зміни структури нейронної мережі. 28

3.4 Демонстрація роботи розпізнавання зображень. 31

4 ОХОРОНА ПРАЦІ 33

ВИСНОВКИ 38

ПЕРЕЛІК ПОСИЛАНЬ 39

Додаток А. Лістинг програми 40

ВСТУП

Стрімке зростання обчислювальних потужностей в останні роки значно розширило спектр завдань, які комп'ютер може вирішувати в автоматичному режимі, практично повністю позбавивши користувача необхідності займатися рутинною роботою, виконання якої може відбуватися без участі людини.

Одним з таких завдань стало спостереження за транспортним потоком, автоматичне виявлення порушень правил дорожнього руху та відправлення штрафу порушнику. Це допомогло частково позбавити співробітників поліції, необхідності контролювати проблемні ділянки. Так само важливим є збір статистики, яка в свою чергу допомагає в проектуванні дорожніх розв'язок з метою підвищення безпеки та зменшення кількості та довжини пробок.

Для реалізації даної можливості використовують камери, які цілодобово ведуть спостереження за ділянками, де найчастіше відбуваються перевищення швидкості, проїзд на забороняючий знак світлофора, або в місцях з обмеженим або забороненим паркуванням.

Ідентифікацію автомобіля порушника комп'ютер проводи шляхом аналізу потокового зображення з камери спостереження. При виявленні маневрів, які заборонені чинними правилами, фіксує реєстраційний номер транспортного засобу, який, залежно від правил дорожнього руху держави та розташування самої камери, розташований на передньому або задньому бампері.

Проте нерідкі ситуації коли номер частково пошкоджений, неправильно встановлений або з ним був проведений ряд певних дій з метою зниження його читання, таких як: умисне нанесення товстого шару бруду і пилу, усунення частини фарби або навпаки нанесення спеціальних світловідбивних наліпок.

Проблемою, яку вирішує мій дипломний проект, є ситуації, коли при фіксації частково пошкодженного номерного знака, система не може чітко визначити особистість і автомобіль порушника. Досить часто трапляються випадки коли звернення системи до бази даних з неповною інформацією про

номер авто, повертає їй вибірку з кількох транспортних засобів. Однак якби вона додатково мала марку автомобіля, то шанс на однозначну ідентифікацію порушника міг би бути збільшений багаторазово.

Метою даної дипломної роботи є проектування та розробка засобу для ідентифікації марки автомобіля по зображенням їх емблем, враховуючи можливість спотворення зображення або деформації самої емблеми.

Для досягнення цієї мети слід вирішити кілька завдань:

- Розглянути існуючі методи класифікації зображень;
- Провести аналіз їх властивостей стосовно мети роботи;
- Спроектувати та реалізувати перспективні моделі, що здійснюють високоякісну класифікацію емблем автомобілів.

АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

У сучасному світі дедалі виникає все більше сфер людської діяльності де можливо позбутися рутинної та монотонної роботи. Однією з таких областей є розпізнавання автомобілів. Це завдання може зустрічатися у різних ситуаціях, як приклад:

- контроль швидкісних лімітів, автомобільні магістралі та перехресть, пошук порушників, що покинуле місце дорожньо-транспортної пригоди, або просто контроль зони з обмеженнями по паркуванню.

- різноманітні контрольно-пропускні пункти або шлагбауми, починаючи від в'їзду на паркувку торговельного центру, закінчуючи межами сусідніх держав.

- дошки оголошень, присвячені продажу автомобілів. У цій сфері автоматизація може бути використана як для прискорення роботи самих користувачів, так і для перевірки коректності введених ними даних.

1.1 Аналіз різних систем вирішення задачі класифікації

Як приклад реалізації задач автоматизації, пов'язаний з автомобільною сферою, можна розглянути рішення задачі визначення автомобільного номера. У цьому прикладі, завдання реалізоване за допомогою методу зіставлення шаблонів для розпізнавання символів.

Даний метод розпізнавання передбачає наявність шаблонів для всіх зображень символів, що зустрічаються в процесі його роботи. Винесення рішення про належність поточного зображення, до певного класу, здійснюються за критерієм мінімуму чи максимуму певної метрики подібності зображення символу та його шаблона.

Однією з можливих для даного методу, може бути реалізація, в якій кожному пікселю вхідного зображення, ставиться у відповідність число від 0 до 5. Пікселі шаблону, що відповідають пікселю символу, позначаються нульовими

значення, а пікселі, що не належать символу, позначаються значеннями від 1 залежно від відстані пікселя до найближчого пікселя символу.

Якщо значення кореляції з шаблоном певного класу максимально, це означає, що зображення символу номера відноситься до цього класу. Тобто метрикою цього методу є абсолютне значення коефіцієнта кореляції.

Цей спосіб був реалізований досить давно і має свій список переваг та недоліків. До його переваг можна віднести: відносно низькі вимоги до обчислювальних потужностей комп'ютера, на якому відбувається класифікація та порівняно простий спосіб початкового налаштування системи.

До недоліків даного методу можна віднести його низьку точність роботи, в порівнянні з сучаснішими способами вирішення подібних завдань. Збільшити якість виконання завдання класифікації, можна використовувати нейронні мережі.

Нейронна мережа - це математична модель, програмне втілення якої, побудоване за аналогією до організації та функціонування біологічних нейронних мереж - мереж нервових клітин живого організму. Дане визначення виникло в процесі вивчення мозку, та моделювання процесів, які в ньому відбуваються [1].

Після розробки алгоритмів використовуваних у навчанні, одержувані моделі стали використовувати в практичних цілях: завдання, розпізнавання образів, прогнозування, завдання управління різними механізмами та ін.

У процесі створення нейронної мережі для вирішення того чи іншого завдання, участь людини зводиться до підготовки навчальної вибірки та визначення вдалої структури, яка зможе задовольнити поставлені спочатку вимоги. Людина формує дані на яких мережа буде тренуватися, далі відбувається обчислення вагових коефіцієнтів шляхом використання алгоритму зворотного розповсюдження помилки, в процесі чого, мережа не потребує уваги людини до закінчення свого навчання.

Нейронні мережі застосовуються для вирішення досить широкого спектра завдань, такі як: прогнозування, кластеризація, переклад тексту, класифікація та інших.

Розглянемо вдалі проекти, які виконують саме задачу класифікації, тобто заздалегідь відомо, для яких конкретно класів даних використовується нейронна мережа.

Гарним прикладом вирішення цього завдання є розпізнавання номерного знаку автомобіля. Розглянемо роботу Петрова С. П., у якій він вирішив задачу завдяки згорковій нейронній мережі [2].

На вхід вона приймає зображення символу номера розміром 11x17 пікселей у градаціях сірого, а на виході видає один з 21 класів, якому це зображення належить.

Її структура складається з:

- Вхідного шару складеного з 187 нейронів.
- Згорткового шару складеного з 20 карт ознак розміром 4x7 пікселів.
- Повнозв'язного шару складеного з 100 нейронів.
- Вихідного шару складеного з 21 нейрона.

Генерація тренувальної вибірки здійснювалася шляхом застосування перетворень, що спотворюють, зображення оригінальних символів. Зображення символів шрифту попередньо наводилися до розміру 10*16 пікселів.

Зображення оригінальних символів шрифту зображені на рисунку 1.1.

Рисунок 1.1 - зображення оригінальних символів шрифту

Зображення спотворених символів шрифту зображені на рисунку 1.2.

Рисунок 1.2 - зображення спотворених символів шрифту

Навчання нейронної мережі виконувалося на тренувальній вибірці, що складається з 184233 зображень символів. Зображення тестової вибірки символів

що складається з 6378 зображень не брали участь у навчанні. Як алгоритм навчання використовувався алгоритм зворотного поширення помилки.

Навчання проводилося протягом 60-ти епох. Тестування нейронної мережі проводилося на зображеннях тестової вибірки: помилка розпізнавання склала 11,8%, що є на багото кращим результатом, ніж метод зіставлення шаблонів, використовуючи який, максимальним результатом було 23,1%.

Також як приклад успішно реалізованої роботи, можна розглянути нейронні мережі, що використовують базу даних рукописних цифр MNIST, яку розробив Національний інститут стандартів і технологій США з метою калібрації та порівняння структур нейронних мереж [3].

Дані, які містить база MNIST, є модифікованими зображеннями з використовуваної раніше NIST. Відмінності між ними укладені у збільшенні розміру зображення з 20×20 до 28×28 пікселів, у нормалізації та збільшенні загального обсягу даних та перетворенні з чорно-білого формату в градації сірого.

Декілька елементів з бази MNIST зображені на рисунку 1.3.

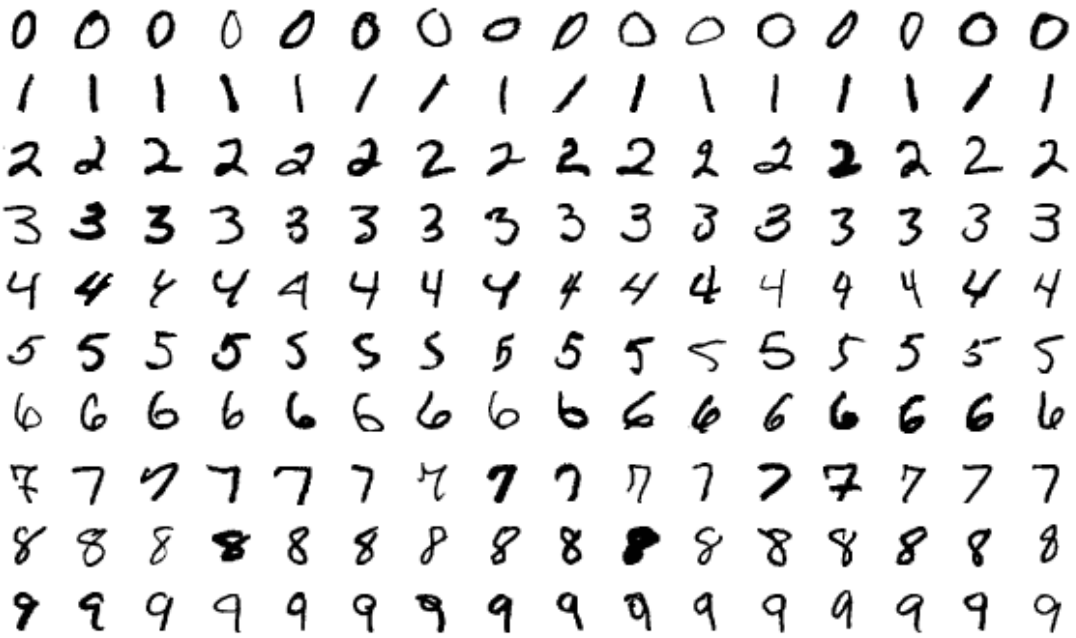

Рисунок 1.3 – декілька зображень з бази MNIST

На вхід нейронні мережі, що використовують при навчанні MNIST, отримують масив з 784 числових значень від 0 до 1, які відповідають значенням яскравості в кожній точці зображення розміру 28x28. Вони видають масив з 10 числових значень від 0 до 1, які у сумі дають 1. Кожен нейрон вихідного шару відповідає числу від 0 до 9, а величина його значення відображає ймовірність відповідності зображення на вході з певним числом на виході.

До складу аналізованої бази даних входили 60 тисяч зображень у вибірці, що використовується для навчання, і 10 тисяч зображень, що використовуються для оцінки якості роботи проектованої мережі. У створенні набору брало участь 500 чоловік.

У таблиці 1.1 наведені результати навчання нейронних мереж з різними типами архітектур, в навчанні та перевірки якості роботи яких була застосована одна база зображень.

Таблиця 1.1 – результати навчання різноманітних нейронних мереж

Тип	Структура	Помилка(%)
Лінійний класифікатор	Однорівневий перцептрон	12.0
Лінійний класифікатор	Попарний лінійний класифікатор	7.6 [4]
Метод найближчих k-сусідів	К-ПН з нелінійною деформацією (P2DHMDM)	0.52
Метод градієнту підвищення	Обробка залишків на базі ознак Хаара	0.87
Нелінійний класифікатор	40 PCA + квадратичний класифікатор	3.3
Метод опорних векторів	Віртуальна система опорних векторів, град-9 поли, 2-пікселя jittered	0.56
Нейронна мережа	2-рівнева мережа 784-800-10	1.6
Нейронна мережа	2-рівнева мережа 784-800-10	0.7
Глибока нейронна мережа	6-рівнева мережа 784-2500-2000-1500-1000-500-10	0.35
Згорткова нейронна мережа	6-рівнева мережа 784-40-80-500-1000-2000-10	0.31
Згорткова нейронна мережа	6-рівнева мережа 784-50-100-500-1000-10-10	0.27
Згорткова нейронна мережа	5 CNN-мереж, 6-784-50-100-500-1000-10-10	0.21 [5]

1.2 Обгрунтування застосування згорткової нейронної мережи

Повноз'єднані нейронні мережі у яких реалізован метод прямого поширення можуть бути застосовані як для навчання ознак об'єктів, так і для класифікування даних, але застосування цього типу архітектури для роботи з класифікування зображень не є практичним навіть у архітектурі з одними прихованим шаром, через велику кількість вхідних нейронів. Наприклад, шар для маленького за сучасними мірками зображення розміром 200 × 200 пікселів має 40 000 ваг. Операція згортки дає змогу сильно спростити цю проблему, оскільки вона зменшує кількість вільних параметрів, дозволяючи мережі бути глибшою за меншої кількості параметрів. Наприклад, незалежно від розміру зображення, області замощування розміру 5 × 5, кожна з одними й тими ж спільними вагами, вимагають лише 25 вільних параметрів. Таким чином, це розв'язує проблему зникання або вибуху градієнтів у тренуванні традиційних багатошарових нейронних мереж з багатьма шарами за допомогою зворотного поширення.

В результаті дослідження існуючих реалізацій розв'язання задачі побудови нейронної мережи здатної з великою точністю класифікувати зображення, маючи однакові бази для навчання, також було виявлено що мережі побудовані з використанням згортки, показують більшу ефективність.

1.3 Згорткові нейронні мережі

Згорткові нейронні мережі в машинному навчанні - це клас глибинних штучних нейронних мереж прямого поширення, який успішно застосовується до аналізу візуальних зображень [6]. В основі роботи згорткові мережі полягає біологічний процес, а саме схему з'єднання нейронів зорової кори тварин. Окремі нейрони кори реагують на стимули лише в обмеженій області зорового поля, відомій як рецептивне поле. Рецептивні поля різних нейронів частково перекриваються таким чином, що вони покривають усе зорове поле [7].

Структура цього типу мереж складається з шарів входу та виходу, а також із декількох прихованих шарів. Зазвичай приховані шари складаються із: згорткових шари, агрегувальних шарів та повноз'єднаних шарів.

Демонстрація роботи шарів згортки зображена на рисунку 1.4 та на рисунку 1.5.

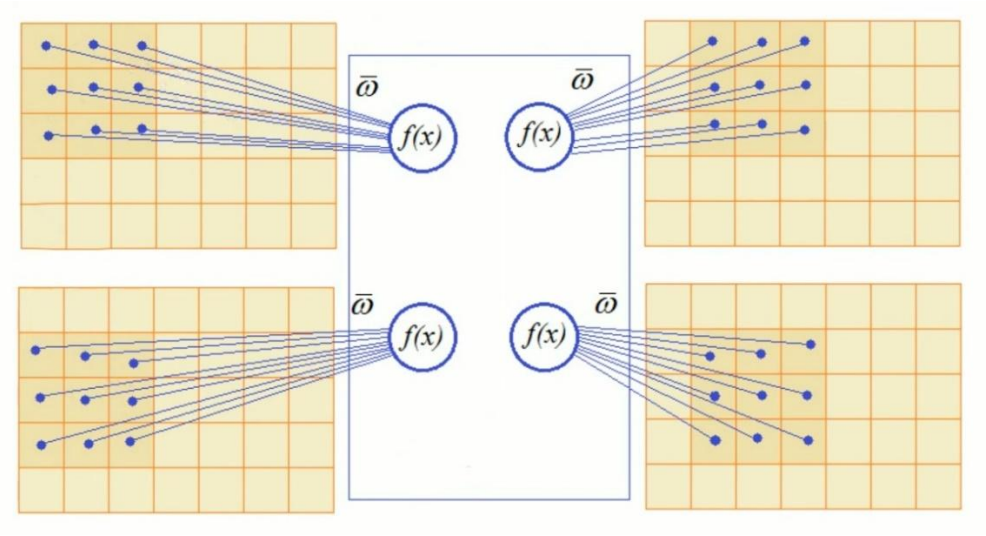

Рисунок 1.4 – демонстрація роботи шарів згортки

Згорткові шари застосовують до входу операцію згортки, передаючи результат до наступного шару. Згортка імітує реакцію окремого нейрону на зоровий стимул. Кожен згортковий нейрон обробляє дані лише для свого рецептивного поля.

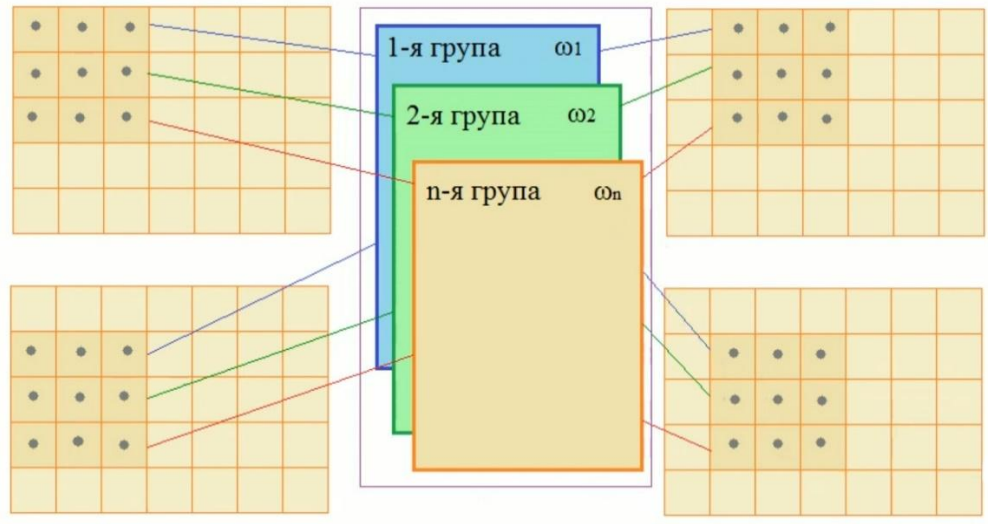

Рисунок 1.5 – демонстрація роботи шарів згортки

Згорткові мережі можуть включати шари локального або глобального агрегування, які об'єднують виходи кластерів нейронів одного шару до одного нейрону наступного шару. Наприклад, максимізаційне агрегування (англ. max pooling) використовує максимальне значення з кожного з кластерів нейронів попереднього шару.

Демонстрація роботи шарів агрегування зображена на рисунку 1.6.

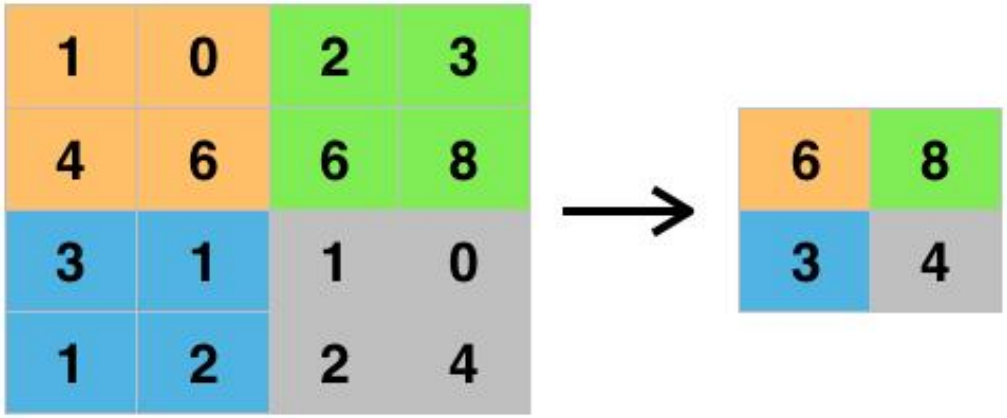

Рисунок 1.6 демонстрація роботи шару агрегування

ПРОЕКТУВАННЯ СИСТЕМИ

При навчанні нейронної мережі для вирішення задачі класифікації, їй на етапі навчання пропонуються різні зразки з маркуванням того, якому класу вони належать. Зразок зазвичай представляється як вектор значень ознак. У ньому сукупність всіх ознак має однозначно визначати клас, до якого належить зразок. У разі якщо у зразку присутня недостатня кількість ознак, мережа не зможе однозначно визначати до якого певного класу він належить, що не задовольняє первісному завданню. Після закінчення навчання мережі, їй можна подати на вхід невідомі раніше образи і на виході отримати належність до певному класу.

Структура такої мережі охарактеризована тим, що кількість нейронів розташованих у вихідному шарі така сама, як і кількість визначених класів. При цьому встановлюється однозначна відповідність між виходом нейронної мережі та класом, який він представляє. Коли мережі надається на розпізнавання якийсь образ, на одному з її вихідних нейронів має з'явитися ознака того, що образ належить цьому класу. У той самий час, на інших виходах нейронної мережі, повинна бути ознака того, що образ певному класу не належить. При виникненні ситуації, коли на декількох вихідних нейронах значення приблизно однакові, прийнято вважати, що нейронна мережа, що використовується, не може однозначно класифікувати образ, який був поданий їй на вхід.

Існує величезна кількість мов програмування, а також програмних бібліотек для роботи з штучними нейронними мережами, реалізованих на різних мовах програмування, таких як C++, C#, Python і ін. У якості інструменту для вирішення задачі поставленної у рамках дипломного проекту була обрана мова програмування Python [8].

Python - високорівнева мова програмування загального призначення, орієнтований на підвищення продуктивності розробника і читання коду. Синтаксис ядра Python мінімалістичний. У той же час стандартна бібліотека включає великий обсяг корисних функцій.

Для реалізації нейронної мережі було обрано середовище розробки «PyCharm».

Для реалізації розпізнавання обрана бібліотека Keras - відкрита нейромережева бібліотека, написана на мові Python. Вона націлена на оперативну роботу з мережами глибинного навчання [9].

У процесі створення нейронної мережі потрібно пройти певний перелік етапів:

- Збір даних для навчання;
- Підготовка данних;
- Вибір топології мережі;
- Експериментальний підбір параметрів мережі;
- Експериментальний підбір параметрів навчання;
- Навчання.

2.1 Збір даних для навчання

Створення вибірки, яка буде використовуватися для навчання нейронної мережі, і її попередня обробка, як правило, є найбільш трудомістким етапом вирішення задачі. Набір даних, що використовується в навчанні, повинен задовольняти певним критеріям:

- навчальні дані повинні, наскільки це можливо, відповідати даним, із якими нейромережа буде зіштовхуватися у процесі своєї роботи. Як приклад, не варто навчати нейромережу на зображеннях іграшкових автомобілів, якщо від неї потрібно розрізняти справжні авто;

- виходячи з ресурсів, наданих на створення навчальної вибірки, потрібно забезпечити максимально можливе розмаїття даних;

- дані мають бути чітко структуровані і в процесі навчання не повинно виникати протиріч, оскільки це призведе до поганого результату.

Вихідні дані перетворюються на вид, у якому вони можуть бути подані на вхідні нейрони мережі. Кожен запис у файлі даних називається навчальною

парою. Навчальна пара містить по одному значенню на кожен вхід та вихід мережі.

Для отримання задовільних результатів, необхідно виконати нормування вхідних значень. Наприклад, на перший нейрон мережі подаються величини зі значеннями від нуля до одиниці, але в другий — від нуля до ста. За відсутності нормування значення, вихід другого нейрону завжди буде істотно більший, ніж вихід у першому нейроні. При нормуванні розмірності всіх вхідних даних зводяться докупи.

Створені, у рамках цієї дипломної роботи, значення були переведені з формату яскравості пискелю (від 0 до 255), до формату більш прийнятного для нейронної мережі (від 0 до 1).

У зв'язку з відсутністю готових навчальних вибірок було прийнято рішення про її самостійне створення. Вибірка на половину складається із зображень, що знаходяться у відкритому доступі в інтернеті, наполовину сфотографована самостійно.

На рисунку 2.1 зображені декілька зображень зі створеної вибірки.

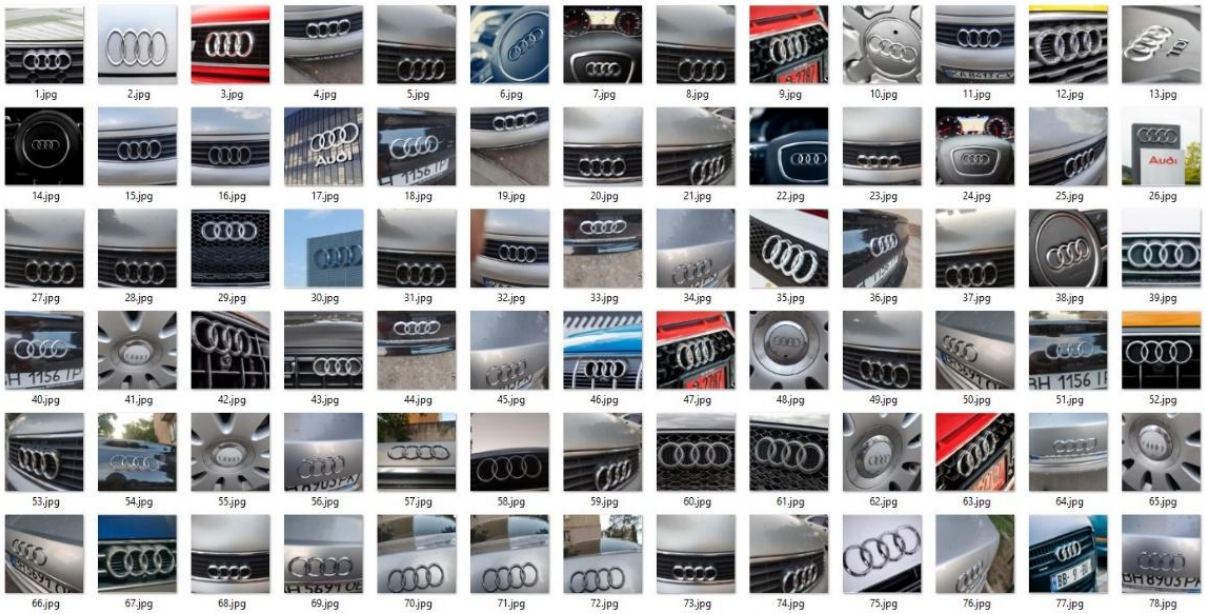

Рисунок 2.1 – декілька зображень зі створеної вибірки

2.2 Підготовка даних

Після збору даних їх необхідно привести до одного формату. У зв'язку з відсутністю достатньо значної інформації у кольорових каналах, було здійснено перетворення всіх зображень із повнокольорових у градації сірого. Також співвідношення сторін всіх зображень були приведені у формат 1 до 1, та розміру 260x260 пікселів. Важливим фактом є присвоєння кожному елементу навчальної вибірки імені у форматі:

```
назва_марки+'.'+номер_зображення_серед_одного_класу
```

Також всі зображення були позбавлені шумів та зайвих елементів на які нейромережа може 'відволікатися' в процесі навчання.

Модифікована вибірка зображена на рисунку 2.2.

Рисунок 2.2 – декілька зображень з модифіковоної вибірки

У якості першого шару нейромережі, був доданий шар попередньої обробки, який створює випадкові деформації кожного зображення, що міститься в навчальній вибірці. У випадковому порядку, у них змінюється яскравість, контрастність, відбувається зсув центру у випадкову сторону, випадкове

збільшення або зменшення загального розміру, що буквально розширює вибірку, що містить по 200 різноманітних зображень кожної моделі у кілька разів [10].

Для його створення, був написаний наступний програмний код:

```
data_augmentation = keras.Sequential([
    layers.RandomRotation(0.5),
    layers.RandomZoom(height_factor=0.3, width_factor=0.3,
    fill_mode='reflect'),
    layers.RandomFlip(mode='horizontal_and_vertical'),
    layers.RandomContrast([0.5, 0]),
    layers.RandomTranslation(0.15, 0.15)])
```

На рисунку 2.3 - зображений результат роботи шару попередньої обробки

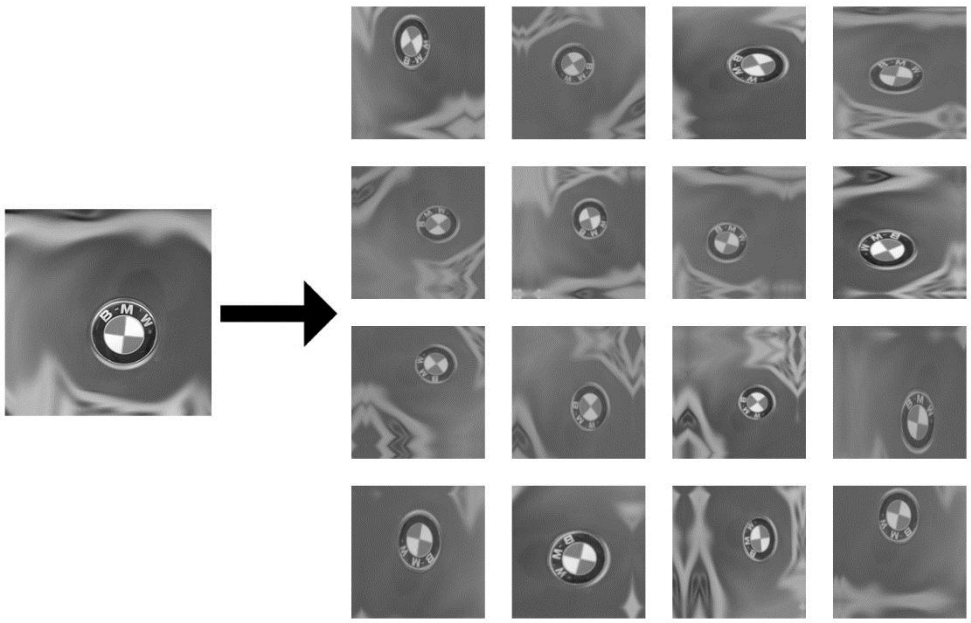

Рисунок 2.3 – результат роботи шару попередньої обробки

2.3 Вибір топології мережі

Для реалізації поставленого завдання було прийнято рішення використовувати згорткову нейронну мережу, тому що при роботі з зображеннями кількість вхідних нейронів є параметром прямо пропорційним розміру зображення. І при використанні простого повнозв'язкового перцептрону,

кількість вагових коефіцієнтів, які будуть постійно змінюватися в результаті навчання, буде надмірно великою.

В якості функції активації була обрана функція ReLU, її математичний вид зображений на рисунку 2.4.

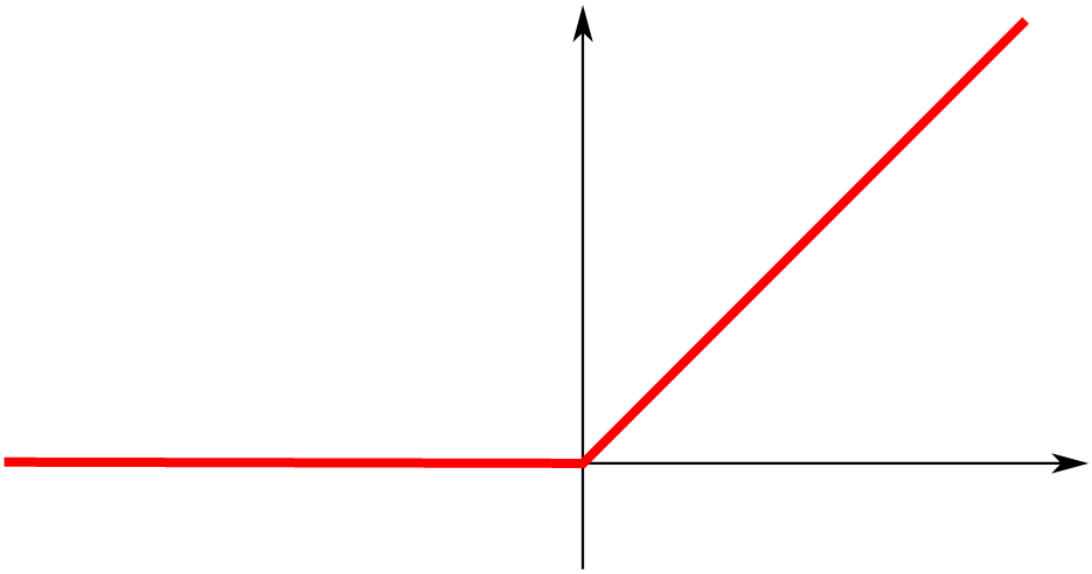

Рисунок 2.4 – графік функції ReLU

Її головним плюсом є те, що вона вимагає малу кількість обчислювальних потужностей, що витрачаються на її обчислення, через це знижується фактичний час навчання нейронної мережі. Це відмінне рішення для реалізації поставленого завдання. Також, використовуючи ReLU, ми можемо вдаватися до використання методу вимикання деяких нейронів, це технологія, що при її правильному застосуванні дозволяє збільшити узагальнюючу здатність нейронної мережі та прискорити скоротити час, що витрачається на навчання.

2.4 Експериментальний вибір характеристик мережі.

Спираючись на досвід авторів статей, розглянутих у першому розділі дипломної роботи, було прийнято рішення використати таку структуру нейронної мережі:

- Шар попередньої обробки;

- Вхідний шар, що складається з 67600 нейронів;
- Згортковий шар, що містить 16 карт ознак;
- Пулінговий шар із маскою 2x2;
- Згортковий шар, що містить 32 карт ознак;
- Пулінговий шар із маскою 2x2;
- Згортковий шар, що містить 64 карт ознак;
- Пулінговий шар із маскою 2x2;
- Згортковий шар, що містить 128 карт ознак;
- Пулінговий шар із маскою 2x2;
- Повнозв'язний шар, що складається з 256 нейронів;
- Вихідний шар що складається з 5 нейронів;

Для створення цієї структури, був написаний наступний програмний код:

```
model = Sequential([
data_augmentation,
layers.Rescaling(1./255,input_shape=(260, 260, 1)),
layers.Conv2D(16,(3,3),padding='same',
activation='relu'),
layers.MaxPooling2D((2,2), strides=2),
layers.Conv2D(32,        (3,3),        padding='same',
activation='relu'),
layers.MaxPooling2D((2,2), strides=2),
layers.Conv2D(64,        (3,3),        padding='same',
activation='relu'),
layers.MaxPooling2D((2,2), strides=2),
layers.Conv2D(128,        (3,3),        padding='same',
activation='relu'),
layers.MaxPooling2D((2,2), strides=2),
layers.Flatten(),
layers.Dense(256, activation='relu'),
layers.Dense(num_classes)])
```

Схематична структура нейронної мережі зображена на рисунку 2.5.

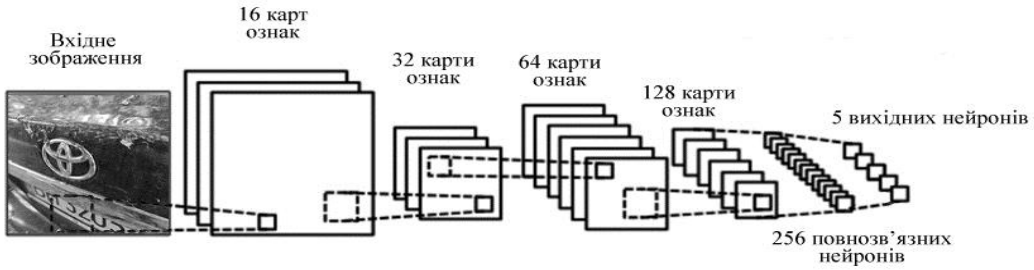

Рисунок 2.5 - схематична структура нейронної мережі.

2.5. Навчання

Для прискорення навчання нейронної мережі було прийнято використовувати міні-вибірки розміром 16 зображень.

При навчанні використовувалася метрика точності, вона показує кількість чітко визначених зображень у процесі навчання. Максимальне її значення було досягнуто під час навчання протягом 250 епох.

У якості оптимізатору алгоритму обратного поширення помілки, був застосований оптимізатор Adam, зі стандартним значенням кроку збіжності.

Графік точності роботи нейронної мережі на навчальній вибірці та вибірки валідації представлений на рисунку 2.6.

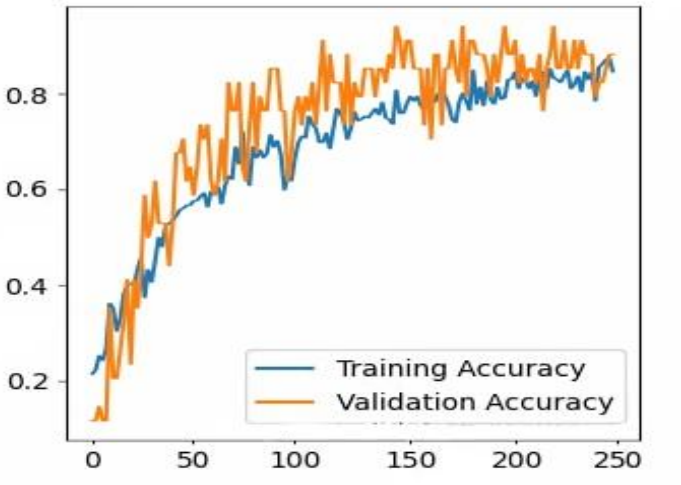

Рисунок 2.6 – графік точності роботи нейронної мережі під час навчання

На прикінці навчання, кількість правильних класифікацій на навчальній вибірці, та на вибірці валідації, становить 84.2%.

3 ТЕСТУВАННЯ СИСТЕМИ

3.1 Оцінка якості роботи мережі

Для оцінки якості навчання нейронної мережі необхідно подати їй на вхід дані, які не брали участь у процесі навчання. Така вибірка, як і навчальна, також була також створена самостійно. На рисунку 3.1 представлена частина тестової вибірки.

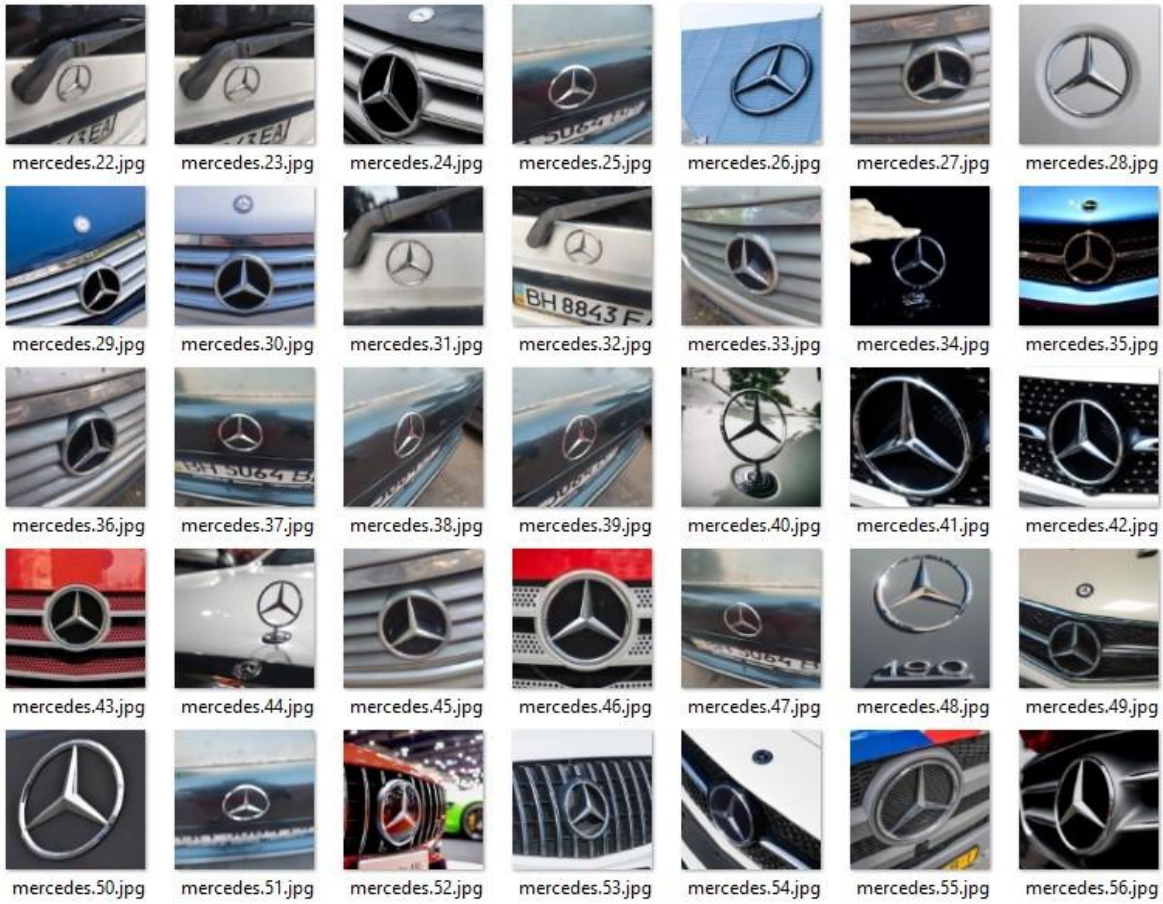

Рисунок 3.1 – декілька зображень з тестової вибірки

Інтерфейс користувача, для реалізації завдання перевірки якості навчань, був реалізований за допомогою бібліотеки TKinter. Його демонстрація виконана на рисунку 3.2.

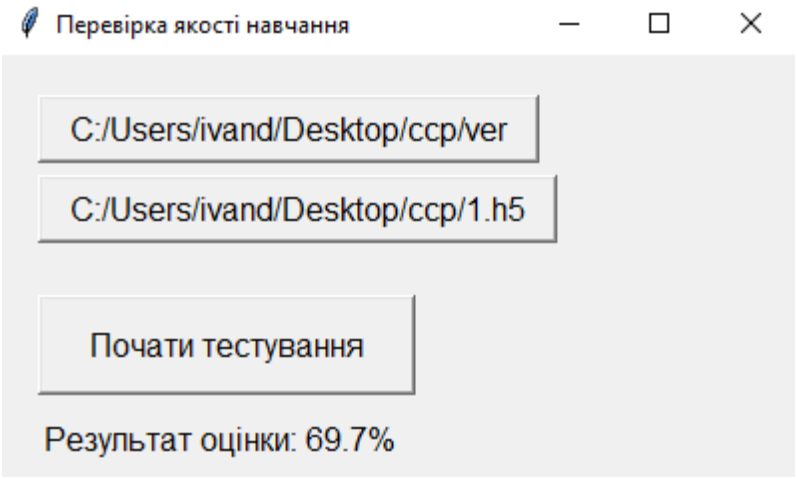

Рисунок 3.2 – користувацький інтерфейс для перевірки якості

В результаті перевірки проектованої нейронної мережі було отримано значення точності рівне 69,7%, що не є задовільним результатом.

3.2 Поліпшення результатів шляхом зміни параметрів навчання

Спочатку, для поліпшення якості навчання нейронної мережі, було прийнято продовжити навчання протягом додаткових 50 епох, що збільшить загальне їх число до 300. Інтерфейс дозволяє провести донавчання мережі, зображений на рисунку 3.3. Результат тестування мережі після додаткового навчання зображений на рисунку 3.4.

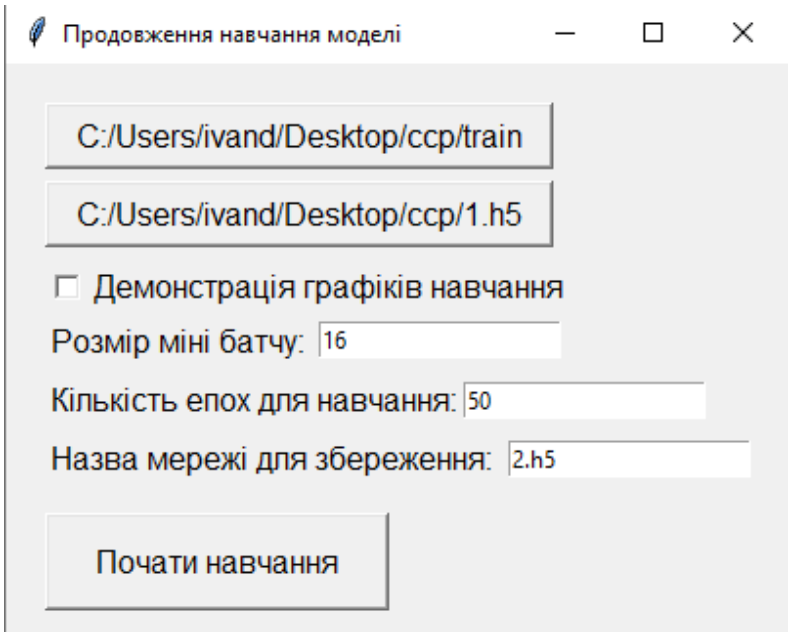

Рисунок 3.3 – користувацький інтерфейс для продовження навчання

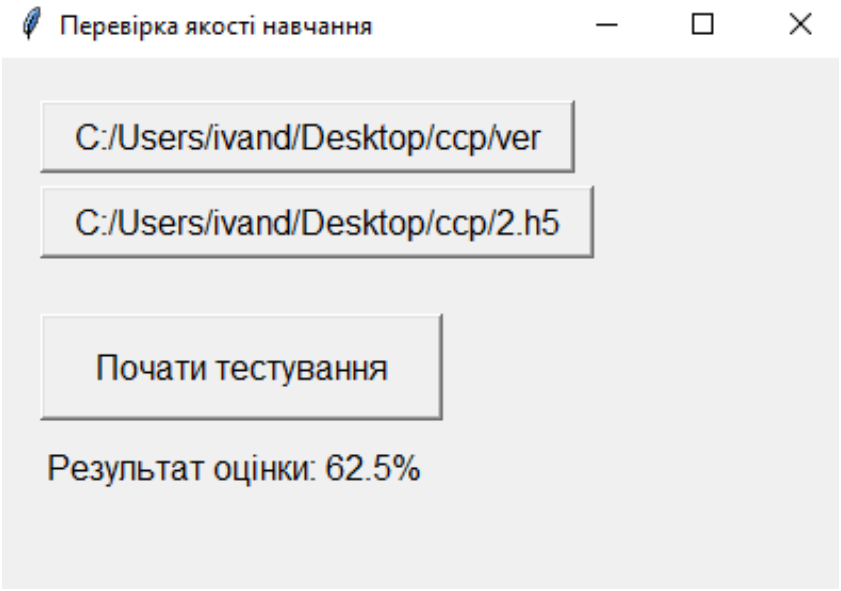

Рисунок 3.4 – результат перевірки якості навчання

Додаткове навчання погіршило результат на 7,2%, що за підсумком спричинило фінальний результат у 62,5%. Уважно проаналізувавши отримані дані, було прийнято рішення, що максимальний результат, який може видати структура, що розглядається, для даного набору навчання, не перевищує 73-75%.

3.3 Поліпшення результатів шляхом зміни структури нейронної мережі.

У результаті аналізу результатів якості спроектованих систем, було вирішено, що первисна структура нейронної мережі, має досить багато зв'язків, з чого випливає, її досить мала узагальнююча здатність. Такі ситуації виникають, через те, що велика кількість нейронних зв'язків сприяє 'запам'ятовуванню' нейронної мережі тестових даних, через що на навчальній вибірці та на вибірці валідації, нейромережа показує хороші результати, а на тестовій вибірці, показники знижуються.

Це ще раз показує важливість тестування проектованої мережі на даних, які не брали участь у процесі навчання.

Як нова структура нейронної мережі, було прийнято використовувати мережу з наступною структурою:

- Шар попередньої обробки;

- Вхідний шар, що складається з 67 600 нейронів;

- Вертковий шар, що містить 16 карт ознак;
- Пулінговий шар із маскою 2x2;
- Згортковий шар, що містить 32 карт ознак;
- Пулінговий шар із маскою 2x2;
- Згортковий шар, що містить 64 карт ознак;
- Пулінговий шар із маскою 2x2;
- Повнозв'язний шар, що складається із 128 нейронів;
- Вихідний шар що складається з 5 нейронів.

Для створення цієї структури, був написаний наступний програмний код:

```
    model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255,input_shape=(260, 260, 1)),
    layers.Conv2D(16,(3,3),padding='same',
activation='relu'),
    layers.MaxPooling2D((2,2), strides=2),
    layers.Conv2D(32,        (3,3),        padding='same',
activation='relu'),
    layers.MaxPooling2D((2,2), strides=2),
    layers.Conv2D(64,        (3,3),        padding='same',
activation='relu'),
    layers.MaxPooling2D((2,2), strides=2),
    layers.Flatten(),
    layers.Dense(256, activation='relu'),
        layers.Dense(num_classes)])
    model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(
from_logits=True),
metrics =['accuracy'])
```

Для навчання нейронної мережі було реалізовано інтерфейс, зображений рисунку 3.5.

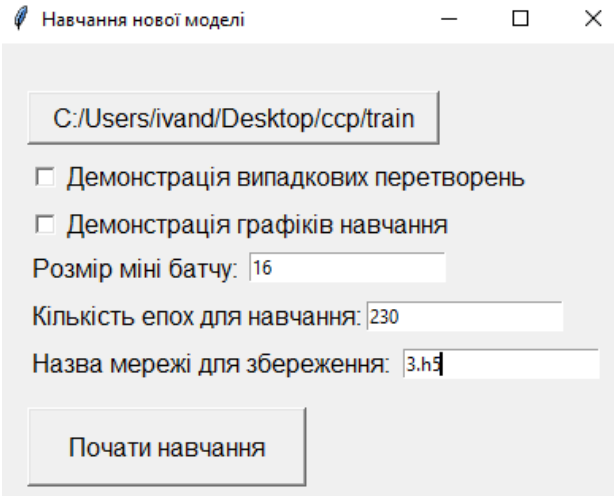

Рисунок 3.5 – користувацький інтерфейс для навчання мережі

Максимальний результат, якого вдалося досягти використовуючи описану структуру, становить 61,3%, що так само не є задовільним. Він був досягнутий під час навчання протягом 230 епох.

Виходячи з досвіду отриманого при проектуванні двох попередніх варіантів, було прийнято рішення використати наступну структуру:

- Шар попередньої обробки;
- Вхідний шар, що складається з 67 600 нейронів;
- Згортковий шар, що містить 16 карт ознак;
- Згортковий шар, що містить 32 карт ознак;
- Пулінговий шар з маскою 2x2 з коефіцієнтом вимикання нейронів 0.2;
- Згортковий шар, що містить 64 карт ознак;
- Згортковий шар, що містить 128 карт ознак;
- Пулінговий шар з маскою 2x2 з коефіцієнтом вимикання нейронів 0.3;
- Повнозв'язний шар, що складається з 256 нейронів;
- Вихідний шар що складається з 5 нейронів;

Функція активації ReLU яка використовується як функція активації, дозволяє використовувати технологію Dropout. Сенс її роботи полягає у відключенні, опреляемого коефіцієнтом, кількості нейронів, що дозволяє досягти вищої узагальнюючої здібності від нейронної мережі, що у своє чергу дозволяє отримати вищі результати точності на тестової вибірці.

Для створення цієї структури було написано наступний програмний код:

```
model = Sequential([
    data_augmentation,
    layers.Rescaling(1./255, input_shape=(260, 260, 1)),
    layers.Conv2D(16, (3,3), padding='same', activation='relu'),
    layers.Conv2D(32, (3,3), padding='same', activation='relu'),
    layers.MaxPooling2D((2,2), strides=2),
    layers.Dropout(0.2),
    layers.Conv2D(64, (3,3), padding='same', activation='relu'),
    layers.Conv2D(128, (3,3), padding='same', activation='relu'),
    layers.MaxPooling2D((2,2), strides=2),
    layers.Flatten(),
    layers.Dropout(0.3),
    layers.Dense(256, activation='relu'),
    layers.Dense(num_classes)])
model.compile(optimizer='adam',
loss=tf.keras.losses.SparseCategoricalCrossentropy(
from_logits=True),
metrics =['accuracy'])
```

Така зміна структури дозволила досягти результату в 81.2 відсотка точності на тестовій вибірці, що є задовільним результатом.

3.4 Демонстрація роботи розпізнавання зображень.

Для класифікації одиночних зображень, був реалізований інтерфейс користувача зображений на рисунку 3.6

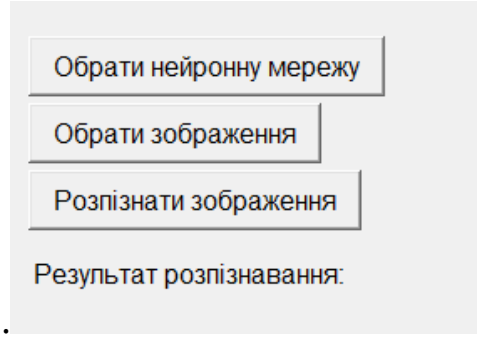

Рисунок 3.6 – користувацький для тестування

Експлуатація даного програмного компонента складається з кількох етапів:

- Вибір навченої нейронної мережі;

- Вибір зображення для розпізнавання;

- Розпізнавання.

При натисканні на кнопку 'розпізнати' у нижній частині вікна виводиться результат. Він представлений у вигляді відсортованого за кількістю відсотків списку, у форматі:

```
значення_в_процентах+' - '+марка_авто
```

Виведення програми зображено на рисунку 3.7.

Рисунок 3.7 – користувацький інтерфейс для навчання мережі

4 ОХОРОНА ПРАЦІ

У сучасному світі середньостатистична людина витрачає на роботу значну частину свого життя. У процесі виконання трудової діяльності працівник обов'язково зустрічається з небезпечними шкідливими факторами, які супроводжують робочий процес. З цього виходить що турбота про безпечні умови праці для працівників є однією із головних задач сучасного суспільства. Метою створення системи охорони праці, було забезпечення належної безпеки праці на підприємствах та турботу про здоров'я працюючого.

Сьогодні дуже багато спеціалістів які працюють на робочих місцях, де основним їхнім інструментом є комп’ютер, тому слід визначити, які негативні фактори можуть впливати на людину під час роботи за комп’ютером. Програмісти у більшості випадків працюють в офісних приміщеннях. Ці приміщення обладнані комп’ютерами та іншими електронними пристроями які можуть створювати фактори які негативно впливають на людину.

В розділі охорони праці виявлені основні небезпечні і шкідливі виробничі фактори при роботі для заданого робочого місця. Згідно зі стандартом небезпечними та шкідливими факторами виробництва для роботи з персональним комп’ютером є:

- недостатня або надлишкова освітленість робочої зони;
- підвищений рівень шуму на робочому місці;
- відсутність або нестача природного світла;
- підвищена чи знижена вологість повітря;
- підвищена чи знижена температура повітря робочої зони;
- тривале статичне напруження;
- підвищене значення напруги в електричному ланцюзі, замикання якого може статися через тіло людини.

На робочому місці, особливу при наявності великої кількості електронних пристроїв, треба подбати про справність електронної мережі та про наявність задовільного заземлення чи занулення.

Одним із найважливіших факторів на робочому місці є освітлення. У ситуаціях коли працівники підприємства не дотримуються норм, вони ризикують погіршити свій зор. У зв’язку з тим, що робота проходить у приміщенні, освітлення має бути приближеним до сонячного світла завдяки штучним джерелам світла.

Частіше за все на робочу місці освітлення представляє собою комбінацію природного та штучного освітлення. У вечірній період часу природне освітлення у приміщенні складає 300лк. Такий показник не є нормою що до документу ДБН В.2.5-28-2006. Нормою освітлення при роботі з комп’ютером або документами складає не менше ніж 400лк. Тому в вечірній період робоче місце має бути обладнане штучними джерелами освітлення.

Для організації тривалої роботи працівників за комп'ютерами, треба встановити розсіяне освітлення. Кількість та вид джерел світла визначається на підставі категорії роботи, яку виконує персонал. Якщо використовуються світильники місцевого освітлення, слід подбати про те, щоб вони не створювали дратуючих зір відблисків від монітору або поверхні стола. Також слід потурбуватися про наявність джерела природного освітлення, оскільки ніякий високотехнологічний світильник не зрівняється за показниками зі світлом сонця. Без природного освітлення людина швидше втомлюється, і швидкість її реакції падає.

Для того, щоб освітлення в офісі було оптимальним, потрібно провести перевірку та виправити недоліки, якщо вони знайдені:

- Колірна температура всіх джерел освітлення має бути приблизно однаковою.

- Робітники мають сидіти поодаль від вікна, якщо вікно виходить на сонячну сторону і на ньому немає фіранок, штор або жалюзі;

- Освітлювальні лампи не повинні бути розташовані на рівні очей.

Крім освітлення, на очі людини впливає положення екрану відносно очей людини, ці норми описані у пунктах 4.14-4.15 ДСанПіН 3.3.2.007-98. Згідно документа, нормою у відстані очей людини до екрану має складати не менше 600-

700мм. Окрім відстані важливим пунктом є забезпечення зручного зорового спостереження у вертикальній площині під кутом 30 градусів.

У сучасних офісах, де найчастіше знаходяться люди, робота яких пов’язана с компьютером, обладнані пристроями які контролюють вологість повітря та кондиціонери, їх робота призводить до великої кількості шуму.

Шум - це звуки які видають пристрої під час своєї роботи, це може бути прилади друку та копіювання, комп’ютери, пристрої які контролюють вологість повітря та кондиціонери. Ці звуки мають різну інтенсивність та гучність, їх одночасова робота може призвести до пошкодження слухового апарату людини.

Згідно документи ДСН 43.3.6 037-99 рівень шуму який допустимий на робочому місці людини яка тривалий час працює за комп’ютером становить не більше 50 дБА. Якщо на робочому місці відсутні прилади другу, кондиціонери та інші прилади які видають звук, при таких умовах фактичне зашумлення робочого місця має дорівнювати 45дБА.

Для того щоб дотримуватися норм потрібно:

1. Розташувати пристрої для дотримання мікроклімату не менш ніж в 4 метрах від робочих місць.

2. Зменшити кількість приладів друку та копіювання до мінімума, та розташувати поодаль від працівників

Також важливим фактором є положення тіла під час роботи за комп’ютером. Під час роботи за комп’ютером робоче місце має бути обладнане кріслом яке буде тримати у правильному напрямку вашу спину, колінні суглоби та інші частини тіла які фізично напружуються під час тривалої роботи.

Якщо не дотримуватися правил які вказують на правильне положення тіла під час сидіння за робочим місцем, то можуть трапитися ситуації через які здоров’я працівника постраждає.

Рівень напруженості електричного поля, під впливом якого може потрапити людина яка працю за комп’ютером становить 15кВ/м протягом восьмигодинного робочого дня. У документі ДСанПІН 3.3.2.007-98, норма напруження електричного поля має дорівнювати 15кВ/м.

Параметри мікроклімату теж мають відповідати нормам які указані у документі ДСанПіН 3.3.2.007-98. Температура повітря на робочому місці нормативне значення має дорівнювати 22-24°С тим часом як фактичне значення має дорівнювати 22°С. Не менш важливим чинником є вологість повітря. Нормою вологості повітря у приміщені де працюють з комп'ютером становить 40-60%. Швидкість руху повітря в приміщені має становити як указано у документі - 0,1 м/с.

Пожежна небезпека - можливість виникнення та розвитку пожежі в будь-якій речовині, процесі, стані. Безпечних не буває, тому якщо вони і не створюють прямої загрози життю та здоров'ю людини, то завдають чи призводять до значних матеріальних втрат. Пожежа є типовим ризиком для будь-якого виробничого процесу або іншої діяльності людини, пов'язаної з роботою в приміщенні. Тому пожежна безпека є особливо важливою в галузі охорони праці.

Важливим завданням для роботодавця є організація місця для паління. Краще, щоб залежні від паління люди не робили цього на робочому місці, та не викидували підпалене сміття для паперових відходів, що також є надзвичайно пожежонебезпечним. Слід донести до співробітників, що курити в офісі суворо заборонено.

В першу чергу слід зазначити наявність вогнегасників їх кількість та тип на тому чи іншому виробничому процесі. Тип вогнегасників залежить від наявності речовин, матеріалів та пристроїв які використовуються у робочих приміщеннях, а їх кількість залежить від площі приміщення та його типу.

Основними параметрами від яких залежить пожежна небезпека речовин та матеріалів, складається з горючості, температури спалаху та концентраційна межа поширення полум'я. При наявності первинних речовин у приміщені поділяються на класи, що є об'єктами пожежної небезпеки визначає клас будівлі. У документі НАПБ Б.03.002-2007 зазначені категорії приміщень та будівель.

Категорія В (Пожежонебезпечна). Горючі гази, легкозаймисті, горючі і важко горючі рідини, а також речовини та матеріали, які здатні при взаємодії з водою, киснем повітря або один з одним вибухати і горіти або тільки горіти;

горючий пил і волокна, тверді горючі та важко горючі речовини і матеріали, за умови, що приміщення, в яких вони знаходяться (обертаються), не відносяться до категорій А, Б

В результаті дослідження документа НАПБА 01.001-2014 був прийнятий висновок, що для виконання поставленного завдання підходить категорія В. Приміщення с цією категорією потребує наявність таких засобів пожежної безпеки, як евакуаційні виходи, плани евакуації та первинні засоби пожежогасіння.

Щоб якомога раніше дізнатися про виникнення вогнища, слід також забезпечити встановлення в офісі необхідної кількості детекторів диму, і переконатися, що вони справні і мають джерело живлення.

Спираючись на розмір приміщення, в якому розташоване робоче місце, який становить до 50 кв.м включно і того факту, що в приміщенні є персональний комп’ютер (ПК), можна зробити висновок, що таке приміщення має бути обладнане двома вогнегасниками вуглекислотного типу, кожен з яких повинен містити не менш ніж 6 кг вогнегасної речовини. Цей тип вогнегасника був обраний тому що ми прямцюємо в приміщені у якому знаходиться багато електричних приладів.

Також слід зазначити що відстань між вогнегасником та робочим місцем не повинна перевищувати 20 метрів.

Вогнегасники вуглекислотні належать до категорії газових вогнегасників, оскільки вогнегасним засобом у них є рідкий вуглекислий газ в зарядному балоні. Перебуваючи під власним надлишковим тиском від 5,7 до 15 МПа, при застосувані він випускає незайманий газ назовні ш тим самим гасить осередки займання.

Вогнегасники цього типу є найпоширенішими в сучасному світі, віни прості у використанні та не вимагає спеціальних навичок.

ВИСНОВКИ

В результати виконання кваліфікаційної роботи було виконанні наступні задачі:

- Аналіз існуючих систем класифікації зображень;
- Вибір оптимальної технології для реалізації поставленного завдання;
- Реалізація програмного компонента, та його вдосконалення.

Проаналізувано сучасні методи вирішення завдання класифікаціі цифрових зображень..

Попередній аналіз дозволив визначити що згорткові нейронні мережі є перспективною моделлю для вирішення завдань даної роботи.

Розроблено декілька варіантів конфігурацій згорткових нейронних мереж, оцінка якості роботи яких показала максимальну точність класифікації у 81.2%

Для покращення результату потрібно розширити тренувальну вибірку, та проводити подальші експерименти зі структурою та іншими параметрами нейронних мереж.

ПЕРЕЛІК ПОСИЛАНЬ

1. Штучна нейронна мережа. URL: https://uk.wikipedia.org/wiki/Штучна_нейронна_мережа

2. Петоров С.П. Свёрточная нейронная сеть для распознавания символов номерного знака автомобиля. URL: https://old.nordavind.ru/node/550

3. MNIST database URL: https://en.wikipedia.org/wiki/MNIST_database

4. LeCun Y. - Gradient-Based Learning Applied to Document Recognition. URL: http://yann.lecun.com/exdb/publis/pdf/lecun-98.pdf

5. Romanuke V. Parallel Computing Center (Khmelnytskyi, Ukraine) represents an ensemble of 5 convolutional neural networks which performs on MNIST at 0.21 percent error rate URL: http://bulletin.kpi.ua/article/view/84115

6. Згорткова нейронна мережа URL: https://uk.wikipedia.org/wiki/Згорткова_нейронна_мережа

7. Kaneda Y. Subject independent facial expression recognition with robust face detection using a convolutional neural network URL: https://web.archive.org/web/20131213022740/http://www.iro.umontreal.ca/~pift6080/H09/documents/papers/sparse/matsugo_etal_face_expression_conv_nnet.pdf

8. Офіційний сайт Python. URL: https://www.python.org/

9. Офіційний сайт Keras. URL: https://keras.io/

10. Документація випадкових перетворень. URL: https://www.tensorflow.org/api_docs/python/tf/keras/layers

Додаток А. Лістинг програми

```
from tkinter import *
from PIL import Image, ImageTk, ImageFont, ImageDraw
from tkinter import filedialog
import os
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow import keras
from keras import layers
from keras.models import Sequential, load_model, Model

img_height = 260
img_width = 260

os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

def createNewNeuro():
    baseWindow = Toplevel(app)
    baseWindow.title("Навчання новоі моделі")
    baseWindow.geometry("400x300+550+100")

    def open_dir():
        global pathdir
        pathdir = filedialog.askdirectory()
        meow.config(text=pathdir)

    def edu():
        trainpath = pathdir
        demo_graph = int(var2.get())
        demo_training = int(var1.get())
        epochs = int(cep_neuro.get())
        batch_size = int(batch_entry_size.get())
        train_ds =
keras.utils.image_dataset_from_directory(
            trainpath,
            validation_split=0.1,
            subset="training",
            seed=123,
            image_size=(img_height, img_width),
            batch_size=batch_size)
        val_ds = keras.utils.image_dataset_from_directory(
            trainpath,
            validation_split=0.1,
```

```
            subset="validation",
            seed=123,
            image_size=(img_height, img_width),
            batch_size=batch_size)
        class_names = train_ds.class_names
        print(class_names)
        AUTOTUNE = tf.data.AUTOTUNE
        train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUN
E)
        val_ds =
val_ds.cache().prefetch(buffer_size=AUTOTUNE)
        normalization_layer = layers.Rescaling(1. / 255)
        normalized_ds = train_ds.map(lambda x, y:
(normalization_layer(x), y))
        image_batch, labels_batch =
next(iter(normalized_ds))
        first_image = image_batch[0]
        data_augmentation = keras.Sequential(
            [
                layers.RandomRotation(0.5),
                layers.RandomZoom(height_factor=0.3,
width_factor=0.3, fill_mode='reflect'),

layers.RandomFlip(mode='horizontal_and_vertical'),
                layers.RandomContrast([0.5, 0]),
                layers.RandomTranslation(0.15, 0.15)
            ]
        )
        if demo_training:
            plt.figure(figsize=(10, 10))
            for images, _ in train_ds.take(1):
                for i in range(16):
                    augmented_images =
data_augmentation(images)
                    ax = plt.subplot(4, 4, i + 1)

plt.imshow(augmented_images[0].numpy().astype("uint8"))
                    plt.axis("off")
            plt.show()
        num_classes = len(class_names)
        model = Sequential([
            data_augmentation,
            layers.Rescaling(1. / 255,
input_shape=(img_height, img_width, 1)),
```

```
            layers.Conv2D(16, (3, 3), padding='same',
activation='relu'),
            layers.Conv2D(32, (3, 3), padding='same',
activation='relu'),
            layers.MaxPooling2D((2, 2), strides=2),
            layers.Conv2D(64, (3, 3), padding='same',
activation='relu'),
            layers.Conv2D(128, (3, 3), padding='same',
activation='relu'),
            layers.MaxPooling2D((2, 2), strides=2),
            layers.Flatten(),
            layers.Dropout(0.3),
            layers.Dense(256, activation='relu'),
            layers.Dense(num_classes)
        ])
        model.compile(optimizer='adam',

loss=tf.keras.losses.SparseCategoricalCrossentropy(from_log
its=True),
                      metrics=['accuracy'])
        history = model.fit(
            train_ds,
            validation_data=val_ds,
            epochs=epochs
        )
        if demo_graph:
            acc = history.history['accuracy']
            val_acc = history.history['val_accuracy']
            loss = history.history['loss']
            val_loss = history.history['val_loss']
            epochs_range = range(epochs)
            plt.figure(figsize=(8, 8))
            plt.subplot(1, 2, 1)
            plt.plot(epochs_range, acc, label='Training
Accuracy')
            plt.plot(epochs_range, val_acc,
label='Validation Accuracy')
            plt.legend(loc='lower right')
            plt.title('Training and Validation Accuracy')
            plt.subplot(1, 2, 2)
            plt.plot(epochs_range, loss, label='Training
Loss')
            plt.plot(epochs_range, val_loss,
label='Validation Loss')
            plt.legend(loc='upper right')
```

```
            plt.title('Training and Validation Loss')
            plt.show()
        model.save(str(name_neuro.get()))
        print(f'Нейросеть {name_neuro.get()} успешно
сохранена! ')

    meow = Button(baseWindow, text="Обрати папку з
тренировочною вибіркою", font="14", padx="10", pady="2",
                  command=open_dir)
    meow.place(x=20, y=30)
    var1 = IntVar()
    c1 = Checkbutton(baseWindow, text='Демонстрація
випадкових перетворень', font="16", variable=var1,
onvalue=1,
                     offvalue=0)
    c1.place(x=20, y=70)
    var2 = IntVar()
    c2 = Checkbutton(baseWindow, text='Демонстрація
графіків навчання', font="16", variable=var2, onvalue=1,
offvalue=0)
    c2.place(x=20, y=100)
    batch_entry_size = StringVar()
    batch_entry = Entry(baseWindow,
textvariable=batch_entry_size)
    Label(baseWindow, text="Розмір міні батчу: ",
font="14").place(x=20, y=130)
    batch_entry.place(x=160, y=132)
    Label(baseWindow, text="Кількість епох для навчання: ",
font="14").place(x=20, y=160)
    cep_neuro = StringVar()
    cep_neuro_entry = Entry(baseWindow,
textvariable=cep_neuro)
    cep_neuro_entry.place(x=234, y=163)
    Label(baseWindow, text="Назва мережі для збереження: ",
font="14").place(x=20, y=190)
    name_neuro = StringVar()
    name_neuro_entry = Entry(baseWindow,
textvariable=name_neuro)
    name_neuro_entry.place(x=257, y=193)
    meow2 = Button(baseWindow, text="Почати навчання",
font="26", padx="20", pady="10", command=edu)
    meow2.place(x=20, y=230)

def createNewNeuro2():
    baseWindow = Toplevel(app)
```

```
    baseWindow.title("Продовження навчання моделі")
    baseWindow.geometry("400x300+100+450")
    def edu():
        trainpath = pathdir
        neuroname = pathneuro
        demo_graph = int(var2.get())
        epochs = int(cep_neuro.get())
        batch_size = int(batch_entry_size.get())
        train_ds =
keras.utils.image_dataset_from_directory(
            trainpath,
            validation_split=0.1,
            subset="training",
            seed=123,
            image_size=(img_height, img_width),
            batch_size=batch_size)
        val_ds = keras.utils.image_dataset_from_directory(
            trainpath,
            validation_split=0.1,
            subset="validation",
            seed=123,
            image_size=(img_height, img_width),
            batch_size=batch_size)
        class_names = train_ds.class_names
        print(class_names)
        AUTOTUNE = tf.data.AUTOTUNE
        train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=AUTOTUN
E)
        val_ds =
val_ds.cache().prefetch(buffer_size=AUTOTUNE)
        normalization_layer = layers.Rescaling(1. / 255)
        normalized_ds = train_ds.map(lambda x, y:
(normalization_layer(x), y))
        image_batch, labels_batch =
next(iter(normalized_ds))
        first_image = image_batch[0]
        model = load_model(str(neuroname))
        history = model.fit(
            train_ds,
            validation_data=val_ds,
            epochs=epochs
        )
        if demo_graph:
            acc = history.history['accuracy']
```

```
            val_acc = history.history['val_accuracy']
            loss = history.history['loss']
            val_loss = history.history['val_loss']
            epochs_range = range(epochs)
            plt.figure(figsize=(8, 8))
            plt.subplot(1, 2, 1)
            plt.plot(epochs_range, acc, label='Training
Accuracy')
            plt.plot(epochs_range, val_acc,
label='Validation Accuracy')
            plt.legend(loc='lower right')
            plt.title('Training and Validation Accuracy')
            plt.subplot(1, 2, 2)
            plt.plot(epochs_range, loss, label='Training
Loss')
            plt.plot(epochs_range, val_loss,
label='Validation Loss')
            plt.legend(loc='upper right')
            plt.title('Training and Validation Loss')
            plt.show()
        model.save(neuroname)
        print(f'Нейросеть {name_neuro.get()} успешно
сохранена! ')

    def open_dir():
        global pathdir
        pathdir = filedialog.askdirectory()
        meow.config(text=pathdir)

    def open_neuro():
        global pathneuro
        pathneuro = filedialog.askopenfile().name
        meow2.configure(text=pathneuro)

    meow = Button(baseWindow, text="Обрати папку з
тренировочною вибіркою", font="14", padx="10", pady="2",
                  command=open_dir)
    meow.place(x=20, y=20)
    meow2 = Button(baseWindow, text="Обрати нейронну
мережу", font="14", padx="10", pady="2",
                  command=open_neuro)
    meow2.place(x=20, y=60)
    var2 = IntVar()
```

```
    c2 = Checkbutton(baseWindow, text='Демонстрація
графіків навчання', font="16", variable=var2, onvalue=1,
offvalue=0)
    c2.place(x=20, y=100)
    batch_entry_size = StringVar()
    batch_entry = Entry(baseWindow,
textvariable=batch_entry_size)
    Label(baseWindow, text="Розмір міні батчу: ",
font="14").place(x=20, y=130)
    batch_entry.place(x=160, y=132)
    Label(baseWindow, text="Кількість епох для навчання: ",
font="14").place(x=20, y=160)
    cep_neuro = StringVar()
    cep_neuro_entry = Entry(baseWindow,
textvariable=cep_neuro)
    cep_neuro_entry.place(x=234, y=163)
    Label(baseWindow, text="Назва мережі для збереження: ",
font="14").place(x=20, y=190)
    name_neuro = StringVar()
    name_neuro_entry = Entry(baseWindow,
textvariable=name_neuro)
    name_neuro_entry.place(x=257, y=193)
    meow3 = Button(baseWindow, text="Почати навчання",
font="26", padx="20", pady="10", command=edu)
    meow3.place(x=20, y=230)

def createNewNeuro3():
    baseWindow = Toplevel(app)
    baseWindow.title("Перевірка якості навчання")
    baseWindow.geometry("400x250+550+450")
    def open_dir():
        global pathdir
        pathdir = filedialog.askdirectory()
        meow.config(text=pathdir)
    def open_neuro():
        global pathneuro
        pathneuro = filedialog.askopenfile().name
        meow2.configure(text=pathneuro)
    def edu():
        verpath = pathdir
        modelname = pathneuro
        ver_ds =
keras.utils.image_dataset_from_directory(verpath, seed=123,
image_size=(img_height, img_width),
```

```
batch_size=1)
        model = load_model(modelname)
        scores = model.evaluate(ver_ds, verbose=0)
        pri = 'Результат оцінки:
'+str(round(scores[1]*100,1))+'%'
        l.configure(text = pri)
    meow = Button(baseWindow, text="Обрати папку з тестовою
вибіркою", font="14", padx="10", pady="2",
                  command=open_dir)
    meow.place(x=20, y=20)
    meow2 = Button(baseWindow, text="Обрати нейронну
мережу", font="14", padx="10", pady="2",
                   command=open_neuro)
    meow2.place(x=20, y=60)
    meow3 = Button(baseWindow, text="Почати тестування",
font="26", padx="20", pady="10", command=edu)
    meow3.place(x=20, y=120)
    l = Label(baseWindow, font="20", text='Результат
оцінки:')
    l.place(x=20, y=180)

def createNewNeuro4():
    baseWindow = Toplevel(app)
    baseWindow.title("Розпізнавання зображення")
    baseWindow.geometry("300x560+1000+100")
    def open_image():
        global pathimg
        pathimg = filedialog.askopenfilename()
        file = pathimg
        img = Image.open(file)
        img = img.resize((260, 260))
        img = img.convert('L')
        img = ImageTk.PhotoImage(img)
        label = Label(baseWindow, image=img)
        label.image = img
        label.place(x=20, y=20)

    def open_neuro():
        global pathneuro
        pathneuro= filedialog.askopenfile().name
        meow.configure(text=pathneuro)

    def raspozn():
        neuroname = pathneuro
```

```
        model = load_model(str(neuroname))
        path = pathimg
        img = tf.keras.utils.load_img(path,
target_size=(img_height, img_width))
        img_array = tf.keras.utils.img_to_array(img)
        img_array = tf.expand_dims(img_array, 0)
        predictions = model.predict(img_array)
        score = tf.nn.softmax(predictions[0])
        score = score.numpy()
        class_names = ['audi', 'bmw', 'chevrolet',
'mercedes', 'toyota']
        result = []
        for i in range(len(score)):
            result.append([round(score[i], 4),
class_names[i]])
        result.sort(reverse=True)
        outtext = 'Результат розпізнавання:\n'
        for i in range(len(score)):
            outtext += str(round(result[i][0]*100,2))+'% -
'+result[i][1]+'\n'
        l.configure(text=outtext)



    meow = Button(baseWindow, text="Обрати нейронну
мережу", font="16", padx="9", pady="3", command=open_neuro)
    meow.place(x=20, y=290)
    Button(baseWindow, text="Обрати зображення", font="16",
padx="9", pady="3", command=open_image).place(x=20, y=330)
    Button(baseWindow, text="Розпізнати зображення",
font="16", padx="9", pady="3", command=raspozn).place(x=20,
y=370)
    l = Label(baseWindow, font="20", text = 'Результат
розпізнавання:')
    l.place(x=20, y=420)

app = Tk()
app.title("Дипломний проект Далеченко")
app.geometry("400x300+100+100")
buttonExample = Button(app, text="Створити нову нейронну
мережу", font="25", padx="9", pady="8",
command=createNewNeuro)
buttonExample2 = Button(app, text="Продовжити навчання
нейронної мережі", font="25", padx="9", pady="8",
command=createNewNeuro2)
```

```
buttonExample3 = Button(app, text="Оцінити якість
навчання", font="25", padx="9", pady="8",
command=createNewNeuro3)
buttonExample4 = Button(app, text="Розпізнати зображення",
font="25", padx="9", pady="8", command=createNewNeuro4)
buttonExample.pack()
buttonExample2.pack()
buttonExample3.pack()
buttonExample4.pack()
app.mainloop()
```