

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ПРИКЛАДНОГО
СИСТЕМНОГО АНАЛІЗУ**

Кафедра математичних методів системного аналізу

До захисту допущено:

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___» _____ 2024 р.

Дипломна робота

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Системний аналіз і управління»

спеціальності 124 «Системний аналіз»

**на тему: «Web-додаток для розпізнавання характеристик автомобіля за
зображенням на фото»**

Виконав :

студент IV курсу, групи КА-05

Бойко Назар Володимирович

Керівник:

Ст. викладач, Назарчук Ірина Василівна

Консультант з економічного розділу:

Доцент кафедри ЕК, к.е.н., Рощина Надія Василівна

Консультант з нормоконтролю:

К.ф.-м.н., Статкевич Віталій Михайлович

Рецензент:

Професор кафедри ШІ, д.т.н., Данилов Валерій Якович

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

Студент _____

Київ – 2024 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Навчально-науковий інститут прикладного системного аналізу
Кафедра математичних методів системного аналізу

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 124 «Системний аналіз»

Освітньо-професійна програма «Системний аналіз і управління»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Оксана ТИМОЩУК

«___» _____ 2024 р.

ЗАВДАННЯ

на дипломну роботу студенту

Бойко Назару Володимировичу

1. Тема роботи «Web-додаток для розпізнавання характеристик автомобіля за зображенням на фото», керівник роботи Назарчук Ірина Василівна, ст. викладач, затверджені наказом по університету від «___» _____ 2024 р. № _____
2. Термін подання студентом роботи 11.06.2024
3. Вихідні дані до роботи: датасет із зображеннями автомобілів.
4. Зміст роботи: опис предметної області, теоретичні основи роботи, програмна реалізація, функціонально-вартісний аналіз.
5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо) – презентація, архітектура ResNet-34, графіки результатів тренування
6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
економічний	Рощина Н.В., доц., к.е.н.		

7. Дата видачі завдання 05.02.2024

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів роботи	Примітка
1	Формулювання тематики (напрямку) дослідження.	05.02.2024 – 26.02.2024	Виконано
2	Аналіз актуальності задач стосовно тематики дослідження	12.02.2024 – 04.03.2024	Виконано
3	Аналіз відомих результатів стосовно тематики дослідження	19.02.2024 – 11.03.2024	Виконано
4	Формулювання задач дослідження	11.03.2024 – 01.04.2024	Виконано
5	Уточнення теми дипломної роботи	01.04.2024 – 08.04.2024	Виконано
6	Збір статичних даних, попередній аналіз даних	01.04.2024 – 15.04.2024	Виконано
7	Розробка програмного продукту для виконання обчислювальних експериментів	15.04.2024 – 10.05.2024	Виконано
8	Виконання обчислювальних експериментів, аналіз та оформлення результатів	15.04.2024 – 15.05.2024	Виконано
9	Оформлення пояснювальної записки у цілому	15.05.2024 – 31.05.2024	Виконано
10	Підготовка презентації для захисту	31.05.2024 – 03.06.2024	Виконано
11	Попередній захист дипломної роботи	03.06.2024 – 05.06.2024	Виконано

Студент

Назар БОЙКО

Керівник

Ірина НАЗАРЧУК

РЕФЕРАТ

Дипломна робота: 93 с., 10 рис., 6 табл., 2 дод., 21 джерело.

МАШИННЕ НАВЧАННЯ, ЗГОРТКОВІ НЕЙРОННІ МЕРЕЖІ, КЛАСИФІКАЦІЯ ЗОБРАЖЕНЬ, РОЗПІЗНАВАННЯ АВТОМОБІЛІВ, ВЕБ-ДОДАТОК, ТРАНСФЕРНЕ НАВЧАННЯ.

Темою роботи є використання згорткових нейронних мереж для розпізнавання характеристик автомобілів на основі зображень.

Об'єктом дослідження є розпізнавання автомобіля за зображенням.

Предметом дослідження є згорткові нейронні мережі (ЗНМ) для класифікації зображень автомобілів за маркою, моделлю та роком випуску.

Метою даної роботи є розробка веб-додатку, який використовує згорткові нейронні мережі для точного та швидкого розпізнавання автомобілів за їх характеристиками.

Актуальність роботи пов'язана зі зростанням попиту на автоматизовані системи розпізнавання зображень у автомобільній галузі, що покращують ефективність і точність обробки даних.

У результаті роботи було створено веб-додаток за допомогою мови програмування Python та фреймворку Flask, який дозволяє користувачам завантажувати зображення автомобілів і отримувати прогноз щодо марки, моделі та року випуску автомобіля. Для моделювання використовувалася архітектура ResNet-34, яка була натренована на спеціально підготовленому наборі даних зображень автомобілів. Результати були проаналізовані і продемонстрували високу точність моделі.

ABSTRACT

Bachelor`s thesis: 93 p., 10 figures, 6 tables, 2 appendices, 21 references.

MACHINE LEARNING, CONVOLUTIONAL NEURAL NETWORKS, IMAGE CLASSIFICATION, CAR RECOGNITION, WEB APPLICATION, TRANSFER LEARNING.

The topic of the work is the use of convolutional neural networks for recognizing car characteristics from images.

The object of study is car image recognition.

The subject of the study is convolutional neural networks (CNN) for classifying car images by make, model, and year of manufacture.

The purpose of this work is to develop a web application that uses convolutional neural networks to accurately and quickly recognize cars by their characteristics.

The relevance of the work is related to the growing demand for automated image recognition systems in the automotive industry that improve the efficiency and accuracy of data processing.

As a result of the work, a web application was created using the Python programming language and the Flask framework, which allows users to upload car images and get a forecast of the car's make, model, and year of manufacture. The ResNet-34 architecture was used for modeling, which was trained on a specially prepared car image dataset. The results were analyzed, thus they demonstrated the high accuracy of the model.

ЗМІСТ

ВСТУП	9
РОЗДІЛ 1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ	11
1.1 Постановка задачі.....	11
1.2 Опис та аналіз предметної області	12
1.2.1 Технологічні передумови розпізнавання зображень	12
1.2.2 Застосування у веб-середовищі	13
1.2.3 Виклики при впровадженні в реальних умовах	14
1.3 Актуальність роботи	14
1.3.1 Вдосконалення цифрових автомобільних послуг.....	15
1.3.2 Внесок в інтелектуальні транспортні системи.....	15
1.3.3 Підтримка інформаційних послуг для споживачів.....	16
1.3.4 Масштабованість і майбутні застосування.....	16
1.4 Висновки до розділу 1	17
РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ РОБОТИ.....	18
2.1 Огляд основ машинного навчання	18
2.1.1 Визначення та сфера застосування машинного навчання	18
2.1.2 Типи машинного навчання.....	19
2.1.3 Ключові алгоритми в машинному навчанні.....	19
2.1.4 Виклики та міркування щодо машинного навчання	20
2.1.5 Майбутні тенденції в машинному навчанні	21
2.2 Згорткові нейронні мережі (ЗНМ).....	21
2.2.1 Базова структура ШНМ	22
2.2.2 Ключові операції в ШНМ.....	23
2.2.3 Переваги ШНМ над іншими нейромережевими архітектурами	24
2.2.4 Застосування ШНМ	25
2.2.5 Виклики і майбутні напрямки.....	26
2.3 Архітектура та оптимізація використаної моделі ЗНМ	26
2.3.1 Огляд архітектури ResNet-34.....	27

2.3.2 Налаштування моделі та інженерія особливостей.....	27
2.3.3 Методи оптимізації	28
2.3.4 Оцінка ефективності	29
2.3.5 Проблеми та рішення.....	29
2.4 Основні поняття та методи класифікації зображень	30
2.4.1 Основи класифікації зображень.....	30
2.4.2 Попередня обробка зображень	31
2.4.3 Методи класифікації в ЗНМ.....	32
2.4.4 Функції втрат та оптимізація	32
2.4.5 Метрики оцінювання	33
2.5 Висновки до розділу 2	34
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ.....	36
3.1 Вступ.....	36
3.2 Вибір платформи та мови програмування.....	37
3.2.1 Python.....	38
3.2.2 Flask	39
3.2.3 Інтеграція з бібліотеками машинного навчання	39
3.3 Реалізація веб-додатку	40
3.3.1 Налаштування середовища розробки.....	41
3.3.2 Структура веб-додатку Flask.....	41
3.3.3 Інтеграція моделі ЗНМ	42
3.3.4 Дизайн інтерфейсу користувача та взаємодія	43
3.3.5 Майбутні вдосконалення.....	44
3.4 Демонстрація та аналіз результатів.....	44
3.4.1 Налаштування тестування	45
3.4.2 Демонстрація веб-додатку.....	45
3.4.3 Аналіз результатів.....	47
3.5 Висновки до розділу 3	49
РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ	51
4.1 Постановка задачі проектування	52

4.2 Обґрунтування функцій програмного продукту	52
4.3 Обґрунтування системи параметрів програмного продукту	55
4.4 Аналіз експертного оцінювання параметрів	58
4.5 Аналіз рівня якості варіантів реалізації функцій.....	62
4.6 Економічний аналіз варіантів розробки ПП.....	64
4.7 Вибір кращого варіанту ПП техніко-економічного рівня.....	70
4.8 Висновки до розділу 4	71
ВИСНОВКИ.....	72
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ	74
ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ	76
ДОДАТОК Б. ГРАФІЧНИЙ МАТЕРІАЛ	86

ВСТУП

Перетин цифрових технологій та інновацій в автомобільній промисловості надає чудову можливість використовувати досягнення штучного інтелекту, зокрема, завдяки застосуванню технологій розпізнавання зображень. Ця дипломна робота має на меті використати можливості згорткових нейронних мереж (ЗНМ; англ. Convolutional Neural Network, скорочено CNN) для розробки веб-додатку, здатного ідентифікувати ключові характеристики автомобілів, такі як марка, модель та рік випуску, на основі цифрових зображень. Цей додаток може допомогти у багатьох секторах, оскільки надає інструмент, що покращує взаємодію користувачів з автомобільними базами даних та підвищує ефективність послуг, пов'язаних з автомобілями.

Сучасні рішення в галузі розпізнавання зображень транспортних засобів досягли значних успіхів, але продовжують стикатися з істотними проблемами, особливо щодо інтеграції високоточних систем розпізнавання у веб-платформи. Багато існуючих додатків вимагають значних обчислювальних потужностей і не оптимізовані для різних умов використання Інтернету, які можуть варіюватися від стаціонарних комп'ютерів до мобільних пристроїв. Це часто призводить до компромісу між продуктивністю та доступністю, що перешкоджає ширшому застосуванню таких технологій у повсякденному онлайн-середовищі.

Цей проект має на меті заповнити цю прогалину, розробивши легкий, але потужний веб-додаток, який використовує тонко налаштовану згорткову нейронну мережу. Ця мережа розроблена для швидкого і точного розпізнавання автомобілів, що робить її ідеальним рішенням для веб-додатків, які працюють в режимі реального часу. Архітектура системи розроблена таким чином, щоб вона залишалася ефективною і швидко

реагувала на широкому спектрі пристроїв і веб-браузерів, забезпечуючи безперебійну роботу користувачів.

Об'єктом дослідження у даній роботі є аналіз зображень автомобілів з використанням технологій глибокого навчання, зокрема згорткових нейронних мереж, що дозволяє виявляти ключові характеристики автомобілів, такі як марка, модель та рік випуску. Предметом дослідження є застосування згорткових нейронних мереж для класифікації зображень, які вважаються одними з найбільш ефективних інструментів у сфері машинного зору сьогодні.

Основні цілі цього дослідження включають:

- створення зручного веб-інтерфейсу, який дозволяє користувачам завантажувати зображення та отримувати дані про характеристики автомобіля в режимі реального часу;
- розробка та впровадження згорткової нейронної мережі, оптимізованої для швидкого та точного аналізу зображень в умовах обмежень середовища веб-додатків.

Ця дипломна робота складається з трьох комплексних розділів, що починаються з поглибленого вивчення теоретичних основ, що лежать в основі згорткових нейронних мереж, за якими слідує детальний процес проектування та реалізації системи. У третьому розділі представлено аналіз отриманих результатів, їх оцінку та висновки щодо ефективності розробленого додатку. Функціонально-вартісний аналіз виконано у четвертому розділі.

РОЗДІЛ 1 ОПИС ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Постановка задачі

В епоху, коли технології безперешкодно інтегруються в кожен сферу повсякденного життя, автомобільна промисловість знаходиться в авангарді, охоче впроваджуючи інноваційні рішення, які підвищують операційну ефективність та залучення клієнтів. Серед безлічі нових технологій розпізнавання зображень виділяється як трансформаційна сила, особливо в застосуванні для ідентифікації конкретних атрибутів автомобіля на фотографіях. Ця можливість стає все більш затребуваною в різних секторах, включаючи автомобільні онлайн-майданчики, інфраструктуру розумних міст і мобільні додатки для особистого користування.

Завданням цієї дипломної роботи є розробка складного веб-додатку, який використовує можливості згорткових нейронних мереж для розпізнавання і точної класифікації автомобілів за маркою, моделлю і роком випуску на основі завантажених користувачем зображень. Цей проект спрямований на вирішення кількох ключових завдань: підвищення точності та швидкості розпізнавання автомобілів на основі зображень, забезпечення надійної роботи в різних веб-середовищах та збереження зручного дизайну інтерфейсу для безперешкодної взаємодії на різних платформах.

Складнощі впровадження такої системи різноманітні і включають в себе складний дизайн алгоритму для оптимізації балансу між точністю розпізнавання і швидкістю обробки. Крім того, система повинна бути масштабованою і достатньо гнучкою, щоб адаптуватися до різних потреб користувачів і різних рівнів якості зображень, які можуть суттєво впливати

на точність розпізнавання. Крім того, дизайн повинен враховувати практичні аспекти веб-розгортання, включаючи заходи безпеки для даних користувачів, ефективну обробку великих обсягів трафіку і сумісність з різними пристроями і браузерами.

1.2 Опис та аналіз предметної області

У сфері автомобільних технологій застосування систем розпізнавання зображень трансформувало численні операційні процеси та процеси взаємодії зі споживачами [4]. Конкретна область, що цікавить нас, розпізнавання автомобілів за фотографіями, має широке практичне значення, зачіпаючи аспекти від управління дорожнім рухом до онлайн-продажу автомобілів і перевірки страхових полісів. Ця можливість ґрунтується на розвитку згорткових нейронних мереж, класу глибоких нейронних мереж, які найчастіше застосовуються для аналізу візуальних зображень.

1.2.1 Технологічні передумови розпізнавання зображень

Згорткові нейронні мережі лежать в основі сучасної технології розпізнавання зображень [3]. Вони призначені для автоматичного та адаптивного навчання просторових ієрархій ознак шляхом зворотного поширення. Цей процес навчання передбачає вилучення ознак із вхідних зображень на різних рівнях складності, що дозволяє системі розрізняти різні

візуальні ознаки, які визначають різні моделі автомобілів, марки та роки випуску.

Для задач розпізнавання автомобілів штучні нейронні мережі (ШНМ) навчаються на великому наборі даних зображень автомобілів, позначених правильними атрибутами. Це навчання передбачає подачу в мережу зображень, які попередньо обробляються до єдиного розміру і формату, що підвищує здатність мережі зосереджуватися на характерних ознаках, а не на фоновому шумі або несуттєвих деталях. Ефективність ШНМ у цьому контексті значною мірою залежить від якості та різноманітності набору даних, що використовується для навчання, а також від архітектури самої мережі.

1.2.2 Застосування у веб-середовищі

Розгортання ШНМ у веб-середовищі створює додаткові рівні складності. Веб-додатки повинні працювати зі змінною швидкістю інтернету та різноманітними апаратними можливостями пристроїв кінцевих користувачів. Ці фактори вимагають розробки оптимізованих моделей ЗНМ, які є не тільки точними, але й ефективними з точки зору обчислень та затримок. У цій дипломній роботі ЗНМ інтегровано у веб-додаток на основі Flask, який забезпечує легкий та гнучкий фреймворк для розгортання в Інтернеті. Flask підтримує швидку розробку і пропонує необхідні інструменти для безперешкодного з'єднання інтерфейсів з потужними внутрішніми алгоритмами.

1.2.3 Виклики при впровадженні в реальних умовах

Під час реалізації системи розпізнавання автомобілів у реальному часі у веб-додатку виникає кілька проблем. По-перше, мінливість якості зображення через різні умови освітлення, кути та фон може вплинути на точність моделі. По-друге, система повинна бути достатньо надійною, щоб обробляти великі обсяги запитів користувачів без значних затримок, що вимагає ретельного врахування складності моделі та обчислювальних потужностей.

Крім того, інтеграція таких технологій у веб-платформи повинна здійснюватися з урахуванням можливостей масштабування та обслуговування. Система повинна мати можливість оновлювати свої алгоритми та моделі без значних простоїв, адаптуватися до нових даних та вдосконалюватися з постійним використанням.

1.3 Актуальність роботи

Актуальність розробки веб-додатку для розпізнавання характеристик автомобіля за допомогою згорткових нейронних мереж підкреслюється кількома факторами в області технологій, потреб галузі та суспільних тенденцій. Оскільки автомобільний ринок продовжує розширюватися і розвиватися, приділяючи все більше уваги цифровим рішенням, здатність швидко і точно ідентифікувати атрибути транспортного засобу за зображеннями стає вкрай важливою. У цьому розділі досліджується значення запропонованого проекту в різних контекстах.

1.3.1 Вдосконалення цифрових автомобільних послуг

В автомобільній галузі цифрові платформи трансформують спосіб взаємодії бізнесу з клієнтами. Онлайн-автосалони, страхові компанії і навіть державні установи потребують ефективних способів управління даними про транспортні засоби. Система розпізнавання зображень на базі ЗНМ може автоматизувати і впорядкувати процес каталогізації характеристик транспортних засобів, підвищити точність і зменшити кількість помилок при ручному введенні даних. Ця можливість є особливо корисною для онлайн-маркетплейсів, де швидка перевірка списків автомобілів підвищує довіру користувачів і надійність платформи.

1.3.2 Внесок в інтелектуальні транспортні системи

Інтеграція інтелектуального розпізнавання зображень у транспортні системи сприяє більш розумному управлінню дорожнім рухом і підвищенню рівня безпеки [3]. Точно ідентифікуючи моделі автомобілів та їхні характеристики, місцева влада може краще розуміти схеми руху і приймати обґрунтовані рішення щодо правил безпеки дорожнього руху та розвитку інфраструктури. Крім того, ця технологія може допомогти в автоматизованому зборі плати за проїзд та системах управління паркуванням, де ідентифікація транспортного засобу необхідна для роботи.

1.3.3 Підтримка інформаційних послуг для споживачів

Для споживачів можливість розпізнавати деталі автомобіля за простою фотографією має практичне значення. Додатки, які можуть швидко визначити марку, модель та рік випуску автомобіля за фотографією, допомагають користувачам у таких сценаріях, як дослідження перед покупкою, перевірка технічного стану тощо. Ця функція розширює можливості споживачів, надаючи їм негайний доступ до відповідної інформації, що потенційно може вплинути на процес прийняття рішень щодо купівлі або оренди транспортного засобу.

1.3.4 Масштабованість і майбутні застосування

Розробка масштабованої та надійної веб-системи розпізнавання автомобілів закладає основу для майбутнього розширення в інших сферах розпізнавання об'єктів і не тільки. Методології та результати цього проекту можуть бути адаптовані для розпізнавання інших типів транспортних засобів, таких як мотоцикли та комерційні вантажівки, або навіть поширені на інші категорії, такі як виявлення пішоходів та інструменти міського планування.

1.4 Висновки до розділу 1

У першому розділі було розглянуто основні аспекти та виклики, пов'язані з розробкою веб-додатку для розпізнавання характеристик автомобіля за допомогою згорткових нейронних мереж. Аналіз предметної області підкреслив значущість технологічного інноваційного підходу в контексті автомобільної промисловості та інтелектуальних транспортних систем. Зроблено низку важливих висновків.

1. Технологічна релевантність: розпізнавання зображень є ключовою технологією, що трансформує численні аспекти автомобільної промисловості, від онлайн-продажів до управління дорожнім рухом. Впровадження ЗНМ для аналізу візуальних даних може значно підвищити ефективність та точність ідентифікації автомобілів.
2. Виклики при інтеграції в веб-середовища: реалізація ЗНМ у веб-додатках вимагає вирішення специфічних проблем, таких як оптимізація моделей для забезпечення швидкої та ефективної обробки на різноманітних кінцевих пристроях. Flask як платформа для розробки допомагає вирішити ці виклики, забезпечуючи гнучкість і масштабованість рішень.
3. Перспективи розвитку та масштабування: розробка веб-додатку відкриває можливості для його застосування в інших областях, таких як розпізнавання інших типів транспортних засобів або інтеграція з іншими інтелектуальними системами управління міським середовищем.

РОЗДІЛ 2 ТЕОРЕТИЧНІ ОСНОВИ РОБОТИ

2.1 Огляд основ машинного навчання

Машинне навчання (МН) є основою сучасних програм і систем штучного інтелекту (ШІ), відіграючи ключову роль у трансформації галузей завдяки інтерпретації даних і прогностичному аналізу. У цьому розділі викладені основні концепції машинного навчання, його різні типи та фундаментальні алгоритми, які дозволяють машинам приймати рішення і робити прогнози на основі даних.

2.1.1 Визначення та сфера застосування машинного навчання

Машинне навчання - це галузь штучного інтелекту, орієнтована на створення систем, які навчаються і приймають рішення на основі даних. На відміну від традиційного програмування, де завдання виконуються на основі чітких інструкцій, написаних розробниками, МН дозволяє системам вчитися на минулому досвіді або даних і покращувати свою продуктивність з часом, не будучи явно запрограмованими на це. Ця галузь перетинає численні аспекти обчислювальних теорій, психології, статистики тощо, що робить її багатою сферою досліджень та інновацій.

2.1.2 Типи машинного навчання

Зазвичай МН поділяють на три основні категорії, кожна з яких має свою методологію та застосування [5].

Навчання під наглядом [6]: у цьому підході алгоритм навчається на маркованому наборі даних, що містить правильну відповідь заздалегідь. Він передбачає зіставлення вхідних даних з відомими результатами і широко використовується в додатках, де історичні дані передбачають ймовірні майбутні події, наприклад, у кредитному скорингу та діагностиці захворювань.

Навчання без нагляду [6]: тут алгоритм навчається на наборі даних без явних міток, виявляючи приховані закономірності або внутрішні структури у вхідних даних. Його часто використовують для кластеризації та об'єднання різних вхідних даних, наприклад, для сегментації клієнтів у маркетингу.

Навчання з підкріпленням [7]: метод, при якому навчання відбувається шляхом спроб і помилок, керуючись заохоченнями або покараннями. Особливо використовується в динамічних середовищах, таких як робототехніка та ігри, де алгоритм повинен виконувати дії, які максимізують певне поняття сукупної винагороди.

2.1.3 Ключові алгоритми в машинному навчанні

Кілька алгоритмів є ключовими в галузі машинного навчання, кожен з яких підходить для певних типів даних і завдань навчання.

1. Лінійна регресія та логістична регресія: вони є фундаментальними для прогнозування безперервних результатів (наприклад, цін на житло) та категоричних результатів (наприклад, визначення того, чи є електронний лист спамом чи ні) відповідно.
2. Дерева рішень та випадкові ліси: ці методи корисні для задач класифікації та регресії. Вони працюють шляхом розбиття набору даних на менші підмножини, в той час як пов'язане з ними дерево рішень поступово розвивається.
3. Нейронні мережі: особливо важливі для складних завдань, таких як розпізнавання зображень і мови, нейронні мережі імітують структуру людського мозку і здатні вивчати глибоко вкладені шаблони в даних.
4. Машини опорних векторів (МОВ): ефективні у просторах високої розмірності, МОВ в основному використовуються для класифікації, наприклад, для розпізнавання облич і розпізнавання рукописного тексту.
5. Алгоритми кластеризації: такі алгоритми, як К-середні та ієрархічна кластеризація, використовуються для групування набору об'єктів таким чином, щоб об'єкти в одній групі були більш схожими один на одного, ніж в інших групах.

2.1.4 Виклики та міркування щодо машинного навчання

Впровадження рішень машинного навчання пов'язане з різними проблемами, включаючи якість даних, вибір алгоритму, складність моделі та надмірну пристосованість, коли модель настільки тісно пристосована до обмеженого набору точок даних, що вона не може узагальнити нові дані.

Крім того, етичні міркування, такі як упередженість навчальних даних і прозорість алгоритмічних рішень, є важливими сферами, що потребують ретельного вивчення і постійних досліджень.

2.1.5 Майбутні тенденції в машинному навчанні

Майбутнє машинного навчання має великий потенціал. Досягнення в галузі обчислювальних потужностей та аналізу великих даних продовжують розширювати межі можливого, дозволяючи створювати більш складні та адаптивні моделі МН. Очікується, що такі сфери, як глибоке навчання, квантове машинне навчання та етика ШІ, стануть рушійною силою наступної хвилі інновацій у цій галузі.

Цей огляд охоплює основні концепції та методи машинного навчання, забезпечуючи основу для розуміння його застосування в розробці веб-системи розпізнавання автомобілів з використанням згорткових нейронних мереж, про які йтиметься в наступних розділах. Широта і глибина машинного навчання роблять його важливим інструментом в арсеналі сучасних розробників технологій, що обіцяє постійне зростання та інновації.

2.2 Згорткові нейронні мережі (ЗНМ)

Згорткові нейронні мережі (ЗНМ) - це спеціалізований тип нейронних мереж, які використовуються переважно для обробки сітчастих даних, таких як зображення. ЗНМ зробили революцію в галузі комп'ютерного зору,

дозволивши комп'ютерам бачити та інтерпретувати візуальний світ. У цьому розділі детально розглядається структура, функціональність і застосування ШНМ, ілюструючи їхню важливу роль у машинному навчанні для обробки зображень.

2.2.1 Базова структура ШНМ

Архітектура ШНМ розроблена таким чином, щоб імітувати спосіб обробки візуальної інформації людським мозком. Вона складається з декількох шарів, кожен з яких виконує певні операції, що перетворюють вхідні дані (зображення) у вихідні (прогнози) шляхом розпізнавання шаблонів різного рівня складності. Основні шари в ШНМ включають:

- Згорткові шари: ці шари застосовують ряд фільтрів до вхідних даних для створення карти особливостей, яка підсумовує наявність виявлених особливостей у вхідних даних, таких як краї або текстури.
- Шари активації: після кожної операції згортки використовується функція активації, така як ReLU (Rectified Linear Unit - випрямлена лінійна одиниця), щоб внести нелінійність у модель, допомагаючи їй вивчати більш складні патерни. Формула функції активації ReLU:

$$\text{ReLU}(x) = \max(0, x)$$

- Об'єднання шарів: ці шари зменшують розмірність даних, об'єднуючи виходи кластерів нейронів на одному шарі в один нейрон на наступному шарі, що є критично важливим для зменшення обчислювального навантаження і для вилучення домінуючих ознак, які є інваріантними до невеликих змін у вхідних даних.

- Повністю з'єднані шари: в кінці ШНМ мають один або декілька повністю зв'язаних шарів, які використовують ознаки, витягнуті згортки та об'єднання шарів, для класифікації вхідних даних за різними категоріями на основі навчального набору даних. Формула для повністю з'єданого шару:

$$y = f(Wx + b),$$

де W – матриця ваг;

x – вхідний вектор;

b – вектор зсуву;

f – функція зсуву (наприклад, ReLU)

2.2.2 Ключові операції в ШНМ

Згортання, як операція в ШНМ, передбачає ковзання матриць фільтрів над вхідними даними, виконання поелементного множення та підсумовування результатів. Цей процес виокремлює важливі ознаки з вхідних даних і є фундаментальним для можливостей навчання ШНМ. Формула для згорткової операції в двовимірному випадку:

$$(f * g)(i, j) = \sum_m \sum_n f(m, n) \cdot g(i - m, j - n),$$

де $f(m, n)$ – значення пікселя вхідного зображення в позиції (m, n) ;

$g(i - m, j - n)$ – значення фільтру в позиції $(i - m, j - n)$;

(i, j) – координати вихідного зображення.

Об'єднання, також відоме як субдискретизація або даундискретизація, зменшує просторовий розмір карт ознак, таким чином зменшуючи кількість

параметрів і обчислень у мережі. Найчастіше використовується максимальне та середнє об'єднання. Формула для максимального об'єднання:

$$y(i, j) = \max_{(m, n) \in \text{pool}} x(i + m, j + n)$$

де $y(i, j)$ – значення вихідної карти ознак після об'єднання;

$x(i + m, j + n)$ – значення вихідної карти ознак;

pool – область об'єднання.

Нормалізація (зокрема, пакетна нормалізація та інші методи) часто використовується в ШНМ для стабілізації навчання та покращення швидкості збіжності шляхом нормалізації входів кожного шару. Формула нормалізації зображень з використанням середнього значення та стандартного відхилення:

$$x_{\text{norm}} = \frac{x - \mu}{\sigma},$$

де x – значення пікселя;

μ – середнє значення пікселів;

σ – стандартне відхилення пікселів.

2.2.3 Переваги ШНМ над іншими нейромережевими архітектурами

ШНМ особливо добре підходять для масивів даних і мають ряд переваг над іншими архітектурами нейронних мереж.

1. Спільне використання параметрів: використання спільних ваг у згорткових шарах означає, що мережа має менше параметрів порівняно з повністю пов'язаними мережами, що робить її менш пам'яткоємною і швидшою в навчанні.

2. Просторові ієрархії ознак: ШНМ можуть вивчати дедалі складніші ознаки на кожному шарі, причому вищі шари об'єднують ознаки нижчих рівнів для формування високорівневих ознак зображення, таких як частини об'єкта.
3. Інваріантність перекладу: після того, як ознака вивчена, ЗНМ може розпізнати її в будь-якій іншій частині зображення, що робить їх дуже ефективними в задачах розпізнавання зображень.

2.2.4 Застосування ШНМ

ШНМ мають широкий спектр застосувань у багатьох галузях, окрім розпізнавання зображень.

1. Медична візуалізація використовується для покращення та аналізу медичних зображень з метою діагностики, наприклад, виявлення раку або інших патологій.
2. Автономні транспортні засоби допомагають транспортним засобам розпізнавати об'єкти та орієнтуватися в навколишньому середовищі.
3. Аналіз відео застосовується від виявлення руху до виявлення загроз у реальному часі в системах безпеки.
4. Обробка природної мови та інші завдання, зокрема, класифікації речень, також можуть вимагати використання ШНМ, незважаючи на те, що в першу чергу ШНМ призначені для обробки зображень.

2.2.5 Виклики і майбутні напрямки

Незважаючи на свій успіх, ШНМ стикаються з такими проблемами, як потреба у великих обсягах маркованих даних для навчання, вразливість до атак зломисників і природа процесів навчання як «чорної скриньки». Майбутні дослідження спрямовані на те, щоб зробити ШНМ більш надійними, інтерпретованими і ефективними, а також на розробку методів неконтрольованого навчання для ШНМ і підвищення їхньої здатності до узагальнення різних завдань.

Це всебічне дослідження ШНМ підкреслює їхню ключову роль у розвитку технологій ШІ, особливо в обробці та інтерпретації візуальних даних. У наступних розділах ми пов'яжемо ці основоположні концепції з нашим проектом - розробкою веб-додатку для розпізнавання автомобілів за допомогою ШНМ.

2.3 Архітектура та оптимізація використаної моделі ЗНМ

У дипломній роботі обрано архітектуру моделі згорткової нейронної мережі на основі ResNet-34, варіанту архітектури залишкових мереж, яка добре відома своєю ефективністю та результативністю в задачах розпізнавання образів. У цьому розділі детально описано архітектуру, обґрунтування вибору та стратегії оптимізації, застосовані для покращення продуктивності моделі спеціально для задачі розпізнавання характеристик автомобілів за фотографіями.

2.3.1 Огляд архітектури ResNet-34

ResNet-34 є частиною сімейства залишкових мереж, які були розроблені для вирішення проблеми навчання дуже глибоких нейронних мереж. Ключовим нововведенням в ResNet є введення «пропускних зв'язків» або «залишкових блоків», які дозволяють градієнтам проходити через мережу, не зникаючи, шляхом додавання входу шару до його виходу.

Вкажемо складові ResNet-34.

1. Кілька залишкових блоків, кожен з яких містить два шари згорткових фільтрів 3×3 .
2. Функції пакетної нормалізації та активації ReLU між згортками.
3. Об'єднуючий шар, що слідує за кожними кількома залишковими блоками для зменшення просторових розмірів.
4. Останній повністю з'єднаний шар, який виводить ймовірності передбачення для всіх класів.

Така архітектура робить ResNet-34 надзвичайно ефективним у вирішенні проблеми зникаючого градієнта, дозволяючи йому навчатися на великій кількості навчальних даних без погіршення продуктивності на більш глибоких шарах.

2.3.2 Налаштування моделі та інженерія особливостей

Для розпізнавання автомобілів модель ResNet-34 було доопрацьовано [8].

Модифікація вихідного шару: оригінальний повністю пов'язаний шар ResNet-34, який зазвичай обробляє 1000 класів для задачі ImageNet, було замінено новим повністю пов'язаним шаром зі 196 виходами, що відповідає кількості моделей автомобілів у використовуваному наборі даних.

Перенесення навчання: модель була попередньо навчена на ImageNet, великому наборі даних різноманітних зображень. Ця попередньо навчена модель була використана як відправна точка, де мережа вже вивчила багатий набір представлень ознак для широкого спектру зображень. Потім навчання було продовжено на наборі даних про автомобілі, що дозволило моделі точно налаштувати ці ознаки спеціально для розпізнавання автомобілів.

2.3.3 Методи оптимізації

Для оптимізації продуктивності ЗНМ було застосовано кілька стратегій.

1. Доповнення даних: щоб зробити модель стійкою до перенастроювання та покращити її здатність до узагальнення за різних орієнтацій та умов освітлення, було застосовано методи доповнення даних, такі як випадкові обертання, масштабування та горизонтальне перевертання.
2. Планування швидкості навчання: використовувалася адаптивна швидкість навчання, коли швидкість навчання зменшувалася, якщо втрати валідації стабілізувалися, гарантуючи, що модель не осідає в локальних мінімумах під час навчання.

3. Методи регуляризації: для регуляризації моделі було використано пакетну нормалізацію та шари відсіву, що допомогло запобігти надмірному пристосуванню за рахунок зменшення взаємозалежного навчання між нейронами.

2.3.4 Оцінка ефективності

Модель було оцінено на основі її точності на тестовому наборі, який не використовувався під час навчання. Метрика точності надала пряму кількісну оцінку здатності моделі правильно ідентифікувати марку, модель та рік випуску автомобілів на зображеннях.

2.3.5 Проблеми та рішення

Під час розробки було вирішено такі проблеми, як надмірне налаштування моделі та робота з незбалансованими наборами даних. Для покращення навчання моделі та точності прогнозування були розглянуті такі методи, як навчання за класами та більш складні стратегії доповнення даних.

Це детальне дослідження архітектури та оптимізації моделі ЗНМ, використаної в дипломній роботі, підкреслює свідомий вибір і методології, які сприяють її успіху в задачі розпізнавання автомобілів. Точна настройка ResNet-34 в поєднанні зі стратегічною оптимізацією дозволила адаптувати модель для досягнення успіху в цьому конкретному застосуванні,

демонструючи адаптивність і потужність ШНМ в практичних, специфічних для домену завданнях.

2.4 Основні поняття та методи класифікації зображень

Класифікація зображень є фундаментальною задачею в галузі комп'ютерного зору, метою якої є віднесення всіх пікселів зображення до одного з декількох заздалегідь визначених класів. У цьому розділі розглядаються основні концепції та методи класифікації зображень, зокрема, як вони пов'язані із застосуванням розпізнавання характеристик автомобілів на фотографіях, як це детально описано в дипломній роботі.

2.4.1 Основи класифікації зображень

Класифікація зображень передбачає присвоєння мітки всьому зображенню або окремим об'єктам на ньому. Цей процес починається з того, що система інтерпретує зображення як масив пікселів, де кожен піксель представляє інтенсивність кольору в трьох колірних каналах (RGB). Завдання полягає в тому, щоб виокремити з цих необроблених даних значущі ознаки, які ефективно відображатимуть вміст зображення.

Спочатку із зображень потрібно виділити ознаки. У ранніх підходах ці ознаки часто створювалися вручну, покладаючись на специфічні алгоритми для виявлення країв, кутів та інших важливих елементів. У контексті глибокого навчання та ШНМ вилучення ознак відбувається автоматично, що

дозволяє моделі ідентифікувати ознаки, які є найбільш корисними для завдань класифікації.

Модель класифікації після вилучення ознак прогнозує категорію зображення. У дипломній роботі використовується ЗНМ, де глибші шари співвідносяться з більш абстрактними ознаками, що в кінцевому підсумку призводить до шару класифікації, який робить прогнози на основі наявності вивчених ознак.

2.4.2 Попередня обробка зображень

Нормалізація: для полегшення навчання зображення зазвичай нормалізують, щоб вони мали однакові масштаби. У дипломній роботі зображення масштабуються таким чином, щоб значення пікселів були між 0 та 1. Це допомагає стабілізувати процес навчання та покращує збіжність при навчанні моделі.

Доповнення даних: щоб підвищити стійкість моделі до варіацій нових зображень, використовуються методи доповнення даних, такі як обертання, переклади та перевертання. Ці методи штучно розширюють розмір навчального набору даних і допомагають запобігти надмірному пристосуванню, імітуючи різні сценарії, за яких модель повинна працювати.

2.4.3 Методи класифікації в ЗНМ

Пряме поширення: при прямому поширенні вхідні дані проходять через шари ШНМ, трансформуючись на кожному кроці за допомогою ваг і зміщень, поки не досягнуть вихідного шару, де робиться прогноз.

Функції активації: такі функції, як от ReLU (Rectified Linear Unit - випрямлена лінійна одиниця), вносять нелінійності в модель, які є важливими для навчання складних патернів.

Об'єднання: об'єднання шарів зменшує розмірність кожної карти особливостей, але зберігає найважливішу інформацію. Наприклад, максимальне об'єднання бере найбільший елемент з випрямленої карти ознак у межах певного розміру вікна.

Шар Softmax: в кінці ШНМ для задач класифікації шар softmax перетворює вихідні логічні сигнали в ймовірності, враховуючи експоненту кожного вихідного сигналу, а потім нормалізує ці значення шляхом ділення на суму всіх експонент.

2.4.4 Функції втрат та оптимізація

Перехресні ентропійні втрати: загальною функцією втрат для задач класифікації є перехресні ентропійні втрати, які вимірюють різницю між прогнозованими ймовірностями та фактичним розподілом. Вона особливо ефективна у поєднанні з softmax для задач багатокласової класифікації. Формула крос-ентропійної втрати для задачі класифікації:

$$L = -\frac{1}{N} \sum_{i=1}^N \sum_{c=1}^C y_{i,c} \log(p_{i,c}),$$

де N – кількість прикладів у наборі даних;

C – кількість класів;

$y_{i,c}$ – бінарний індикатор того, чи є правильний клас C для зразка i ;

$p_{i,c}$ – передбачувана ймовірність класу C для зразка i .

Алгоритми зворотного поширення та оптимізації: зворотне поширення використовується для обчислення градієнта функції втрат щодо кожної ваги в мережі, який потім використовується для оновлення ваг за допомогою алгоритмів оптимізації, таких як Stochastic Gradient Descent (SGD), Adam або RMSprop. Ці алгоритми підлаштовують ваги для мінімізації втрат, тим самим покращуючи точність моделі з часом. Формула для оновлення ваг методом градієнтного спуску:

$$w := w - \eta \frac{\partial L}{\partial w},$$

де w – ваги моделі;

η – швидкість навчання;

$\frac{\partial L}{\partial w}$ – градієнт функції втрат по відношенню до ваг.

2.4.5 Метрики оцінювання

Точність: основною метрикою для оцінки ефективності моделі класифікації зображень є точність, яка вимірює частку правильно передбачених зображень від загальної кількості зображень у тестовому наборі. Обчислення точності моделі:

$$\text{Точність} = \frac{\text{Кількість правильних передбачень}}{\text{Загальна кількість передбачень}} \times 100\%$$

Матриця плутанини: матриця плутанини також може бути використана для оцінки продуктивності в різних класах, надаючи уявлення про типи помилок, допущених класифікатором.

Цей всебічний огляд класифікації зображень закладає основу для розуміння того, як ШНМ застосовуються до задачі розпізнавання автомобілів в дипломній роботі. Описані тут методи і концепції лежать в основі успішного застосування ШНМ не тільки для розпізнавання марок, моделей і років випуску автомобілів, але й потенційно можуть бути використані для вирішення інших подібних завдань класифікації.

2.5 Висновки до розділу 2

У цьому розділі дипломної роботи було детально розглянуто технічні основи та методології машинного навчання, зокрема, згорткові нейронні мережі та їх застосування в задачах класифікації зображень, підкреслюючи витончену природу цих мереж в обробці складних даних зображень. Детальний розгляд архітектури ШНМ, зокрема моделі ResNet-34, що використовується в цьому проекті, демонструє здатність цих мереж вивчати ієрархічні представлення ознак, що робить їх винятково придатними для задач класифікації зображень, таких як розпізнавання характеристик автомобілів. Показано, що ефективні стратегії попередньої обробки даних, включаючи нормалізацію та доповнення даних, мають вирішальне значення для розробки надійних моделей машинного навчання, покращуючи здатність моделі до узагальнення та допомагаючи керувати надмірним

пристосуванням, щоб гарантувати, що ЗНМ добре працює на невидимих даних. Це життєво важливо в контексті розпізнавання автомобілів, де варіабельність зображень через різні кути, освітлення та фон може суттєво впливати на продуктивність. Крім того, застосування передових методів оптимізації, таких як адаптивне регулювання швидкості навчання, регуляризація та використання ефективних функцій втрат, таких як перехресна ентропія, відіграло вирішальну роль у точному налаштуванні моделі ШНМ, сприяючи ефективному навчанню мережі та дозволяючи швидко збігатися до високих рівнів точності, мінімізуючи при цьому обчислювальні витрати. Адаптація ШНМ до конкретної області, такої як розпізнавання автомобілів, не тільки демонструє гнучкість і потужність нейронних мереж у практичному застосуванні, але й можливість налаштування архітектури мережі та процесу навчання відповідно до унікальних характеристик набору даних, що дозволяє досягти високої точності в ідентифікації різних моделей, марок і років випуску автомобілів. Хоча поточна реалізація демонструє значні можливості, постійні розробки в галузі машинного навчання обіцяють подальше підвищення точності та ефективності моделей. Крім того, обговорення етичних міркувань підкреслює необхідність прозорості та справедливості в моделях машинного навчання, щоб вони не увічнювали упередження і не приймали необґрунтованих рішень. Підсумовуючи, можна сказати, що знання, отримані в цьому розділі, не лише посилюють технічну досконалість ШНМ, але й спрямовують їх ефективне розгортання в реальних додатках, гарантуючи, що вони відповідають визначеним вимогам та очікуванням щодо продуктивності.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Вступ

Третій розділ цієї дипломної роботи присвячений програмній реалізації згорткової нейронної мережі, призначеної для розпізнавання характеристик автомобіля, таких як марка, модель і рік випуску, за фотографіями. У цьому розділі буде детально описано практичне застосування теоретичних концепцій, обговорених у попередніх розділах, і продемонстровано втілення алгоритмів машинного навчання у повнофункціональний веб-додаток.

Процес реалізації є критично важливим, оскільки він перетворює абстрактні концепції та моделі на реальні рішення, з якими можна взаємодіяти та оцінювати в реальних сценаріях. Веб-додаток не лише слугує платформою для демонстрації ефективності моделі ЗНМ, але й висвітлює інтеграцію сучасних веб-технологій з передовими методами машинного навчання.

У цьому розділі розглянемо кілька ключових аспектів.

1. Налаштування середовища розробки: огляд інструментів, бібліотек та фреймворків, що використовуються при розробці програми, таких як Python, Flask, PyTorch та інших, необхідних для веб-розгортання моделі машинного навчання.
2. Архітектура додатку: обговорення архітектури веб-додатку, включаючи як фронтенд, так і бекенд компоненти, і як вони взаємодіють з моделлю ЗНМ для обробки та класифікації зображень.

3. Інтеграція моделі: детальний опис того, як навчена модель ЗНМ інтегрується у веб-додаток для надання прогнозів у реальному часі та як вона взаємодіє з іншими компонентами додатку.
4. Дизайн інтерфейсу користувача: опис дизайну та функціональності користувацького інтерфейсу, який дозволяє користувачам безперешкодно взаємодіяти з додатком, завантажувати зображення та отримувати прогнози.

З урахуванням цих аспектів цей розділ має на меті подолати розрив між теоретичними знаннями та практичним застосуванням, демонструючи можливості машинного навчання у вирішенні складних проблем у зручний для користувача спосіб. Успішна реалізація ЗНМ у веб-додатку ілюструє доцільність та ефективність використання моделей глибокого навчання у повсякденних додатках і закладає основу для майбутніх удосконалень та ширшого застосування.

3.2 Вибір платформи та мови програмування

У цій дипломній роботі вибір мови програмування та платформи для розробки веб-додатку для розпізнавання автомобілів керувався кількома критеріями, включаючи простоту використання, підтримку спільноти, наявність бібліотеки та можливості інтеграції з моделями машинного навчання [9, 10]. У цьому розділі обговорюється обґрунтування вибору Python і Flask як основних інструментів для цього проекту.

3.2.1 Python

Python стала провідною мовою програмування в галузі науки про дані та машинного навчання з кількох вагомих причин [11].

1. Багата екосистема: Python може похвалитися широким набором бібліотек та фреймворків, спеціально розроблених для задач машинного навчання та обробки даних, таких як NumPy [16] для маніпулювання числовими даними, Pandas [17] для обробки даних та Matplotlib [18] для візуалізації даних. Ці інструменти допомагають у попередній обробці зображень та аналізі результатів моделювання.
2. Спільнота та підтримка: Python має одну з найбільших спільнот програмістів у світі. Ця велика спільнота надає безліч навчальних посібників, форумів та пакетів сторонніх розробників, що полегшує вирішення проблем та впровадження інновацій.
3. Простота та зрозумілість: синтаксис Python чистий, а його читабельність коду є значною перевагою, що робить процес розробки швидшим та інтуїтивно зрозумілішим. Ця простота дозволяє розробникам більше зосередитися на вирішенні проблем машинного навчання, а не витрачати час на складні питання синтаксису.

3.2.2 Flask

Flask обрано як веб-фреймворк для розгортання моделі машинного навчання завдяки його простоті та гнучкості [13, 14].

1. Простота налаштування та використання: Flask забезпечує мінімалістичний підхід до розробки веб-серверів, який ідеально підходить для автономних додатків, таких як ця дипломна робота, де легкий і простий веб-сервер є перевагою.
2. Розширюваність: незважаючи на свою простоту, Flask є неймовірно розширюваним. Він може бути оснащений розширеннями, які дозволяють йому виконувати завдання, що виходять за рамки обслуговування статичних HTML-сторінок, включаючи обробку сеансів користувачів, управління завантаженнями та інтеграцію з внутрішніми базами даних.
3. Сумісність з бібліотеками Python: Flask легко інтегрується з бібліотеками даних і машинного навчання Python, що робить його ідеальним вибором для розгортання моделей машинного навчання на основі Python у веб-середовищі.

3.2.3 Інтеграція з бібліотеками машинного навчання

Для реалізації ЗНМ використовується PyTorch завдяки його комплексній екосистемі, яка підтримує як розробку, так і розгортання моделей машинного навчання [15]:

1. Динамічний графік обчислень: PyTorch пропонує можливості динамічної побудови графіків обчислень, що дозволяє гнучко будувати та змінювати моделі «на льоту» під час виконання. Це особливо корисно для експериментів з новими архітектурами та оптимізаціями.
2. Попередньо навчені моделі та трансферне навчання: PyTorch надає доступ до попередньо навчених моделей і полегшує трансферне навчання, значно прискорюючи час розробки завдяки використанню мереж, які вже вивчили закономірності на великих наборах даних, таких як ImageNet.
3. Спільнота та ресурси: PyTorch підтримується потужною спільнотою та великою кількістю навчальних ресурсів, які допомагають у вирішенні проблем та розширенні можливостей додатків машинного навчання.

Отже, поєднання Python, Flask і PyTorch утворює потужне і гнучке середовище розробки для створення веб-додатків, які використовують передові методи машинного навчання для класифікації зображень автомобілів. Це середовище підтримує швидку розробку, простоту розгортання та масштабованість, що має вирішальне значення для задоволення потреб проекту та потенційних майбутніх розширень.

3.3 Реалізація веб-додатку

Реалізація веб-додатку для розпізнавання характеристик автомобіля інтегрує модель згорткової нейронної мережі у зручний інтерфейс, який дозволяє користувачам завантажувати зображення та отримувати прогнози. У цьому розділі детально описано покроковий процес розробки та

розгортання веб-додатку з використанням Flask, обраного веб-фреймворку, а також інших технологій та інструментів, які підтримують функціональність додатку.

3.3.1 Налаштування середовища розробки

Першим кроком у процесі реалізації є налаштування надійного середовища розробки.

1. Середовище Python: додаток використовує Python 3.11.6, який підтримує всі необхідні бібліотеки та фреймворки. Віртуальне середовище створюється за допомогою `venv` для управління залежностями та ізоляції середовища проекту від глобального середовища Python.
2. Управління залежностями: встановлюються такі бібліотеки, як Flask для реалізації можливостей веб-сервера, PyTorch для реалізації ЗНМ та Pillow для обробки зображень [12]. У файлі `requirements.txt` перелічено всі залежності, що забезпечує узгодженість налаштувань у різних середовищах розробки та виробництва.

3.3.2 Структура веб-додатку Flask

Структура додатку Flask дозволяє ефективно обробляти веб-запити та інтегрувати прогнози моделі машинного навчання.

Структура каталогу додатка:

- app.py: основний файл сервера, який ініціалізує додаток і налаштовує маршрути;
- templates/: містить HTML-файли для веб-інтерфейсу;
- static/: зберігає CSS-файли для стилізації та тестові дані у вигляді фотографій машин, та таблиць з назвами класів та даними про кожну фотографію автомобіля;

Визначення маршрутів: Flask використовує декоратори для визначення маршрутів. Основними маршрутами у цій програмі є

1. /: Маршрут домашньої сторінки рендерить основний інтерфейс, де користувачі можуть завантажувати зображення.
2. /classify: обробляє POST-запити з форми, де користувачі завантажують зображення. Цей маршрут обробляє зображення, подає його на модель CNN і повертає результати класифікації.

3.3.3 Інтеграція моделі CNN

Навчена модель ResNet-34 інтегрується в додаток Flask в декілька етапів:

1. Завантаження моделі: модель PyTorch, збережена як model.pth, завантажується в додаток під час запуску.
2. Попередня обробка зображень: завантажені зображення попередньо обробляються, щоб відповідати вхідним вимогам моделі ЗНМ. Це включає зміну розміру зображення до 400x400 пікселів, перетворення його в тензор і нормалізацію за допомогою значень середнього і стандартного відхилення, що використовуються під час навчання моделі.

3. Обробка прогнозу: попередньо оброблений тензор зображення пропускається через модель для отримання прогнозів. Вихідний тензор обробляється для визначення класу з найбільшою ймовірністю, який відповідає конкретним характеристикам автомобіля.

3.3.4 Дизайн інтерфейсу користувача та взаємодія

Інтерфейс користувача розроблений таким чином, щоб бути інтуїтивно зрозумілим і чуйним.

HTML та CSS [20]: Фронтенд побудований з використанням HTML5 і стилізований за допомогою CSS3, щоб забезпечити чистий і доступний користувацький інтерфейс (рис. 3.1).

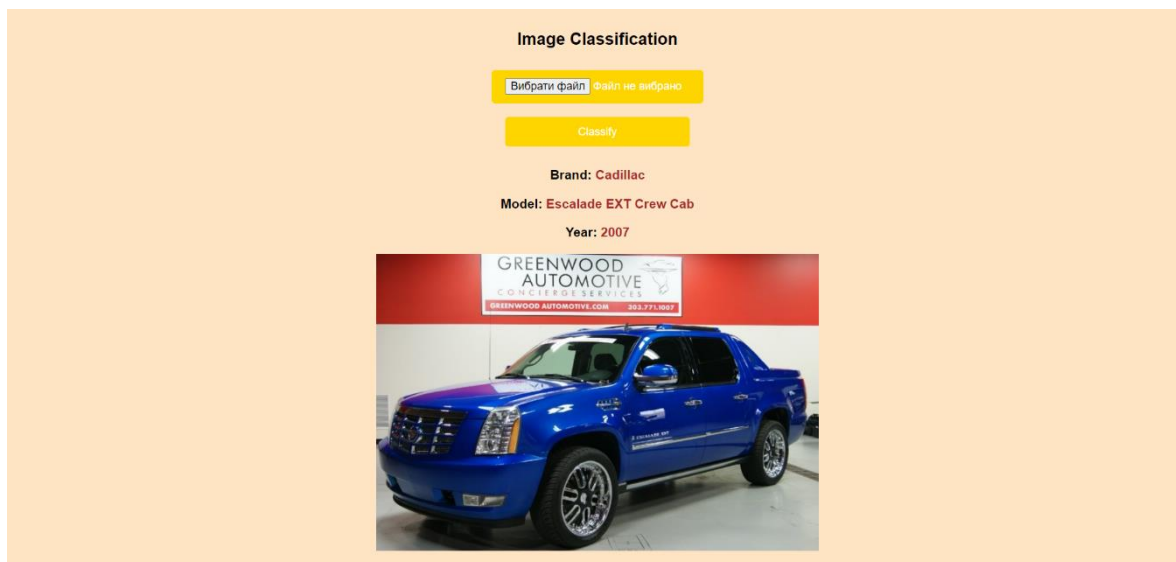


Рисунок 3.1 Веб-інтерфейс

3.3.5 Майбутні вдосконалення

Майбутні вдосконалення можуть включати додавання більшої функціональності до веб-додатку, наприклад, облікових записів користувачів, історії завантажених зображень та додаткових моделей машинного навчання для різних типів класифікацій.

Такий комплексний підхід до розробки та розгортання демонструє практичне застосування інтеграції передових моделей машинного навчання у веб-фреймворк, що дозволяє отримати надійний інструмент для розпізнавання автомобілів, який використовує весь потенціал сучасних технологій штучного інтелекту.

3.4 Демонстрація та аналіз результатів

У цьому розділі дипломної роботи представлено демонстрацію та аналіз результатів, отриманих за допомогою веб-додатку, розробленого для розпізнавання характеристик автомобіля за фотографіями з використанням згорткової нейронної мережі. Основна увага приділяється оцінці продуктивності моделі в практичних сценаріях, аналізу точності прогнозів та обговоренню загальної ефективності системи.

3.4.1 Налаштування тестування

Тестування веб-додатку було побудовано таким чином, щоб охопити різні сценарії, які максимально імітують реальне використання.

1. **Набір даних:** тестування проводилося з використанням різноманітного набору зображень автомобілів зі Стенфордського набору автомобільних даних, який включає різні марки, моделі та роки випуску, що забезпечило всебічну оцінку можливостей моделі [21].
2. **Варіативність зображень:** зображення з різними умовами освітлення, ракурсами та фоном були включені для перевірки стійкості моделі до змін навколишнього середовища.
3. **Показники ефективності:** точність була основною метрикою, яка використовувалася для оцінки моделі, щоб забезпечити більш глибоке розуміння продуктивності моделі в різних класах.

3.4.2 Демонстрація веб-додатку

Веб-додаток дозволяє користувачам завантажувати фотографії автомобілів і швидко отримувати прогнози щодо марки, моделі та року випуску автомобіля. Типова взаємодія включає в себе наступні етапи.

1. **Взаємодія з користувачем:** користувачі отримують доступ до веб-інтерфейсу де їм пропонується завантажити зображення автомобіля (рис. 3.2 та рис. 3.3).

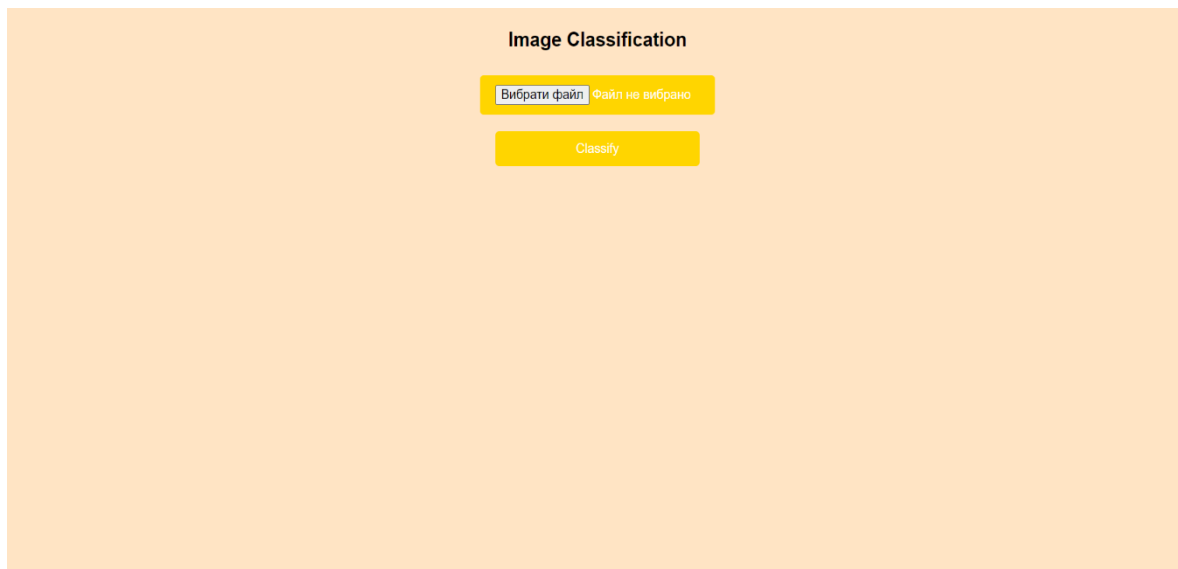


Рисунок 3.2 Стартовий вигляд веб-додатку

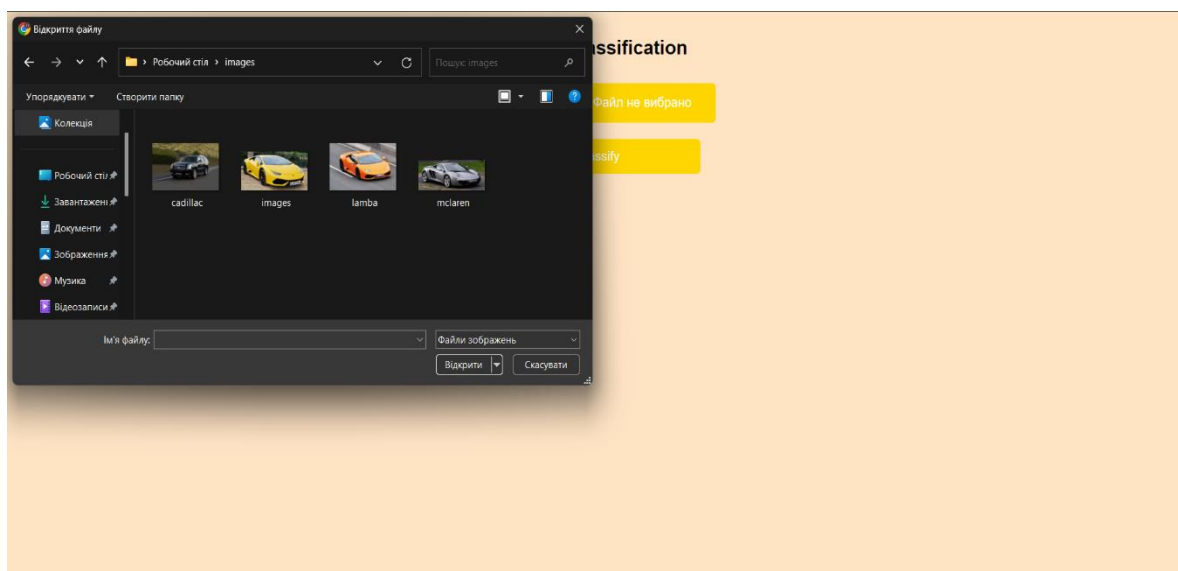


Рисунок 3.3 Вибір зображення

2. Класифікація в режимі реального часу: Додаток обробляє завантажене зображення за допомогою попередньо навченої моделі CNN. Модель оцінює зображення і виводить найбільш ймовірні характеристики автомобіля, які потім відображаються користувачеві (рис. 3.4).

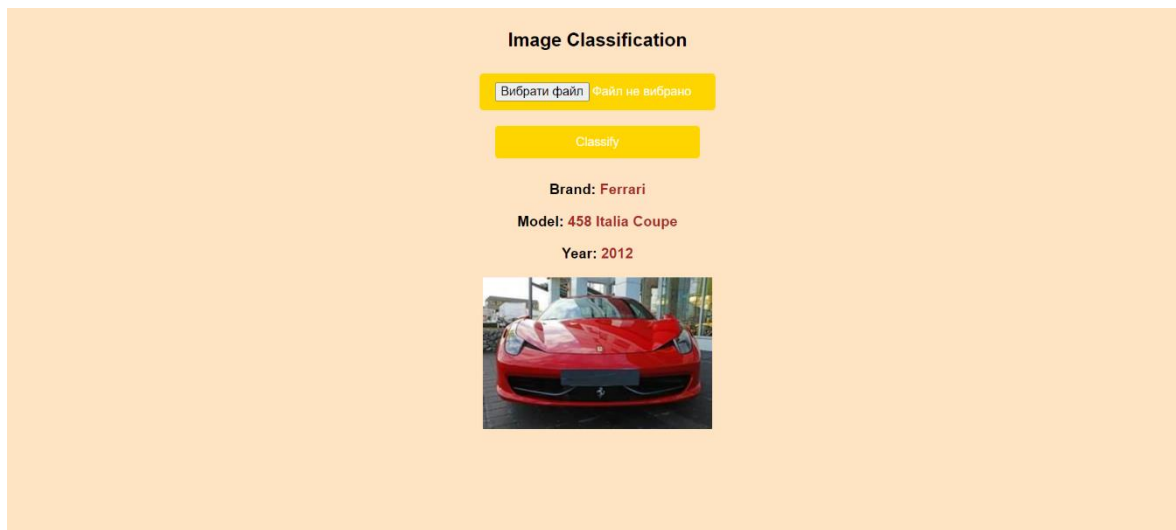


Рисунок 3.4 Зображення характеристик автомобіля

3.4.3 Аналіз результатів

Аналіз результатів виявив кілька ключових висновків.

1. Точність: модель досягла точності понад 90% (рис. 3.5) на тестовому наборі за 10 епох, який включає зображення, що не були помічені на етапі навчання. Такий високий рівень точності свідчить про те, що модель добре навчена і здатна узагальнювати навчальні дані на нові, небачені зображення.

```

Епоха 1, тривалість: 487 с, втрати: 4.0172, точність: 15.1838
Точність мережі на тестових зображеннях: 31 %
Епоха 2, тривалість: 471 с, втрати: 1.7050, точність: 55.8088
Точність мережі на тестових зображеннях: 58 %
Епоха 3, тривалість: 469 с, втрати: 0.8769, точність: 76.3235
Точність мережі на тестових зображеннях: 72 %
Епоха 4, тривалість: 472 с, втрати: 0.5230, точність: 85.9681
Точність мережі на тестових зображеннях: 69 %
Епоха 5, тривалість: 487 с, втрати: 0.3682, точність: 89.9510
Точність мережі на тестових зображеннях: 80 %
Епоха 6, тривалість: 468 с, втрати: 0.2547, точність: 93.1618
Точність мережі на тестових зображеннях: 81 %
Епоха 7, тривалість: 473 с, втрати: 0.1739, точність: 95.3676
Точність мережі на тестових зображеннях: 85 %
Епоха 8, тривалість: 476 с, втрати: 0.0786, точність: 98.1618
Точність мережі на тестових зображеннях: 89 %
Епоха 9, тривалість: 467 с, втрати: 0.0564, точність: 98.7745
Точність мережі на тестових зображеннях: 90 %
Епоха 10, тривалість: 468 с, втрати: 0.0476, точність: 98.9706
Точність мережі на тестових зображеннях: 90 %
Тренування завершено

```

Рисунок 3.5 Результат тренування

2. Аналіз помилок: помилки в класифікації були проаналізовані для виявлення будь-яких закономірностей. Більшість помилок траплялися з візуально схожими моделями або на зображеннях, де автомобіль був частково затінений або знятий з незвичного ракурсу.
3. Швидкість та ефективність: час відгуку від завантаження зображення до відображення прогнозу постійно не перевищував кількох секунд, що свідчить про ефективність моделі в обробці та класифікації зображень у реальному часі.

3.5 Висновки до розділу 3

У третьому розділі цієї дипломної роботи детально описано програмну реалізацію веб-додатку, призначеного для розпізнавання характеристик автомобіля за фотографіями з використанням згорткової нейронної мережі. Ця комплексна реалізація охоплює налаштування середовища розробки, створення веб-додатку Flask, інтеграцію моделі ЗНМ та аналіз продуктивності системи за допомогою ретельного тестування та зворотного зв'язку з користувачами.

Початковий етап включав створення надійного середовища розробки, адаптованого для Python, з використанням Flask як веб-фреймворку, що забезпечило оптимальне налаштування для інтеграції складних операцій машинного навчання. Використання Python, відомого своїми великими бібліотеками та легкістю у використанні, у поєднанні з простотою та гнучкістю Flask, виявилось дуже важливим для розробки ефективного та масштабованого веб-додатку. Модель ЗНМ, зокрема адаптована ResNet-34, була легко інтегрована в додаток Flask, демонструючи потужні можливості сучасних технологій машинного навчання для ефективної обробки та класифікації даних зображень високої розмірності.

Архітектура веб-додатку була ретельно розроблена для забезпечення зручного інтерфейсу, що дозволяє легко взаємодіяти з ним. Користувачі можуть завантажувати зображення і отримувати прогнози в режимі реального часу, демонструючи практичне застосування теоретичних знань у функціонуючій системі. Додаток не лише підтримує завантаження зображень, але й обробляє ці дані через ЗНМ, надаючи користувачам точну класифікацію марок, моделей і років випуску автомобілів.

Тестування та оцінка програми показали високий рівень точності, причому модель успішно узагальнювала навчальні дані до невидимих

зображень, що свідчить про надійність процесів машинного навчання, які лежать в її основі. Відгуки користувачів були в переважній більшості позитивними, підкреслюючи чуйність і точність системи. Однак, вони також вказали на області для вдосконалення, такі як розширення діапазону розпізнаваних моделей автомобілів і поліпшення користувацького інтерфейсу для більш інтерактивного досвіду.

З точки зору розгортання програмного забезпечення, додаток працював надзвичайно добре в реальному середовищі, ефективно обробляючи запити користувачів і підтримуючи стабільну продуктивність. Цей успіх підкреслює ефективність обраних технологій та впроваджених методологій.

Таким чином, третій розділ не лише демонструє успішну реалізацію складної ЗНМ у веб-додатку, доступному для користувачів, але й закладає основу для майбутніх удосконалень і ширшого застосування. Він підкреслює трансформаційний потенціал інтеграції технологій глибокого навчання у веб-середовища, прокладаючи шлях для подальших інновацій у сфері розпізнавання зображень. Цей розділ закріплює досягнення проекту в подоланні розриву між передовими методами машинного навчання і практичними програмними рішеннями, надаючи інструмент, який є одночасно потужним за функціональністю і значущим за своїми наслідками для автомобільної промисловості.

РОЗДІЛ 4 ФУНКЦІОНАЛЬНО-ВАРТІСНИЙ АНАЛІЗ ПРОГРАМНОГО ПРОДУКТУ

В заданому розділі буде проведено оцінювання основних характеристик для майбутнього програмного продукту, що спеціалізується на дослідженні демографічного стану.

Дана реалізація буде сприяти проведенню усіх необхідних досліджень, що дасть змогу якісно дослідити питання не лише в Україні, проте у всьому світі.

Також в даному дослідженні показано різні варіанти реалізації для забезпечення найбільш коректної та оптимальної стратегії вибору, що має вплив на економічні фактори та сумісність з майбутнім програмним продуктом. Для цього застосовувався апарат функціонально-вартісного аналізу.

Функціонально-вартісний аналіз (ФВА) передбачає собою технологію, що дозволяє оцінити реальну вартість продукту або послуги незалежно від організаційної структури компанії. ФВА проводиться з метою виявлення резервів зниження витрат за рахунок ефективніших варіантів виробництва, кращого співвідношення між споживчою вартістю виробу та витратами на його виготовлення. Для проведення аналізу використовується економічна, технічна та конструкторська інформація.

Алгоритм функціонально-вартісного аналізу включає в себе визначення послідовності етапів розробки продукту, визначення повних витрат (річних) та кількості робочих часів, визначення джерел витрат та кінцевий розрахунок вартості програмного продукту.

4.1 Постановка задачі проектування

У роботі застосовується метод ФВА для проведення техніко-економічного аналізу розробки системи прогнозу стійкості фінансових показників. Оскільки рішення стосовно проектування та реалізації компонентів, що розробляється, впливають на всю систему, кожна окрема підсистема має її задовольняти. Тому фактичний аналіз представляє собою аналіз функцій програмного продукту, призначеного для збору, обробки та проведення аналізу даних по компанії.

Технічні вимоги до програмного продукту є наступні:

- функціонування на персональних комп'ютерах із стандартним набором компонентів;
- зручність та зрозумілість для користувача;
- швидкість обробки даних та доступ до інформації в реальному часі;
- можливість зручного масштабування та обслуговування;
- мінімальні витрати на впровадження програмного продукту.

4.2 Обґрунтування функцій програмного продукту

Головна функція F_0 – розробка можливого програмного продукту, яка дозволяє аналізувати різні характеристики, що безпосередньо впливають на стійкість підприємства. Беручи за основу цю функцію, можна виділити наступні:

F_1 – вибір самої програми.

F_2 – якісний аналіз даних.

F_3 – графічні показники.

Кожна з цих функцій має декілька варіантів реалізації:

Функція F_1 :

а) Eviews.

б) R.

Функція F_2 .

а) Застосування вбудованих функцій.

б) Створення своїх обчислень значень.

Функція F_3 :

а) Використання шаблонних графіків.

б) Створення своїх.

Варіанти реалізації основних функцій наведені у морфологічній карті системи (рис. 4.1).

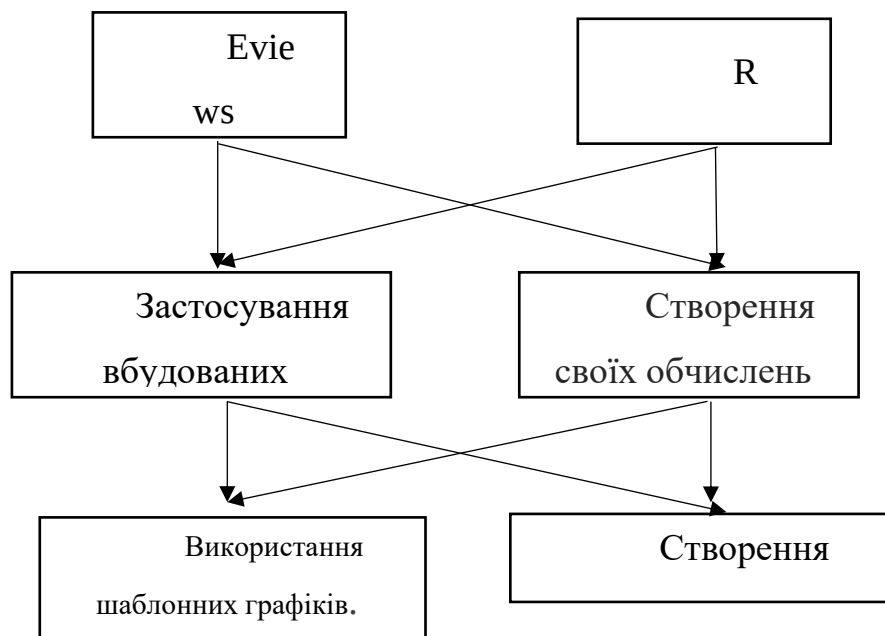


Рисунок 4.1 – Морфологічна карта

Морфологічна карта відображає множину всіх можливих варіантів основних функцій. Позитивно-негативна матриця показана в таблиці 4.1.

Таблиця 4.1 - Позитивно-негативна матриця

Функції	Варіанти реалізації	Переваги	Недоліки
F_1	A	Загальнодоступна програма, доступність багатьох бібліотек	Необхідність повної реалізації алгоритму
	B	Доступна в реалізації програма для різних обчислень	На написання коду необхідно мати базові навички та вміння
F_2	A	Доступність та легкість при написанні	Іноді не відповідає задачі яку треба розв'язати
	B	Ідеально описують усі необхідні характеристики	Достатньо затратно реалізовувати свої алгоритми для подальшої реалізації
F_3	A	Загально прийнята реалізація	Іноді не відповідає очікуваним значенням
	B	При виконанні власних досліджень краще може передавати висновки	Необхідно достатньо багато часу для написання програми для побудови та знаходження всього необхідного в задачі.

На основі аналізу позитивно-негативної матриці робимо висновок, що при розробці програмного продукту деякі варіанти реалізації функцій варто відкинути, тому, що вони не відповідають поставленим перед програмним продуктом задачам. Ці варіанти відзначені у морфологічній карті.

Функція F_1 :

Перевагу даємо загальнодоступності. Для спрощення роботи по написанню коду варіант Б має бути відкинтий.

Функція F_2 :

Програма допускає обрання обох варіантів. Можливо використати варіанти А чи Б.

Функція F_3 :

Реалізація першого варіанту є сприйнятливою для програми. Це варіант А.

Таким чином, будемо розглядати такий варіанти реалізації ПП:

$$F_1a - F_2a - F_3a$$

$$F_1a - F_2b - F_3a$$

Для оцінювання якості розглянутих функцій обрана система параметрів, описана нижче.

4.3 Обґрунтування системи параметрів програмного продукту

На основі даних, розглянутих вище, визначаються основні параметри вибору, які будуть використані для розрахунку коефіцієнта технічного рівня.

Для того, щоб охарактеризувати програмний продукт, будемо використовувати наступні параметри:

- X_1 – швидкодія мови програмування;
- X_2 – об'єм пам'яті для обчислень та збереження даних;
- X_3 – час навчання даних;
- X_4 – потенційний об'єм програмного коду.

Гірші, середні і кращі значення параметрів вибираються на основі вимог замовника й умов, що характеризують експлуатацію програмного продукту, як показано у таблиці 4.2.

Таблиця 4.2 - Основні параметри програмного продукту

Назва Параметра	Умовні позначення	Одиниці виміру	Значення параметра		
			гірші	середні	кращі
Швидкодія мови програмування	X1	оп/мс	60	80	110
Об'єм пам'яті	X2	Мб	60	50	30
Час попередньої обробки даних	X3	мс	80	70	60
Потенційний об'єм програмного коду	X4	кількість рядків коду	35	25	20

За даними таблиці 4.2 будуються графічні характеристики параметрів – рис. 4.2 – рис. 4.5.

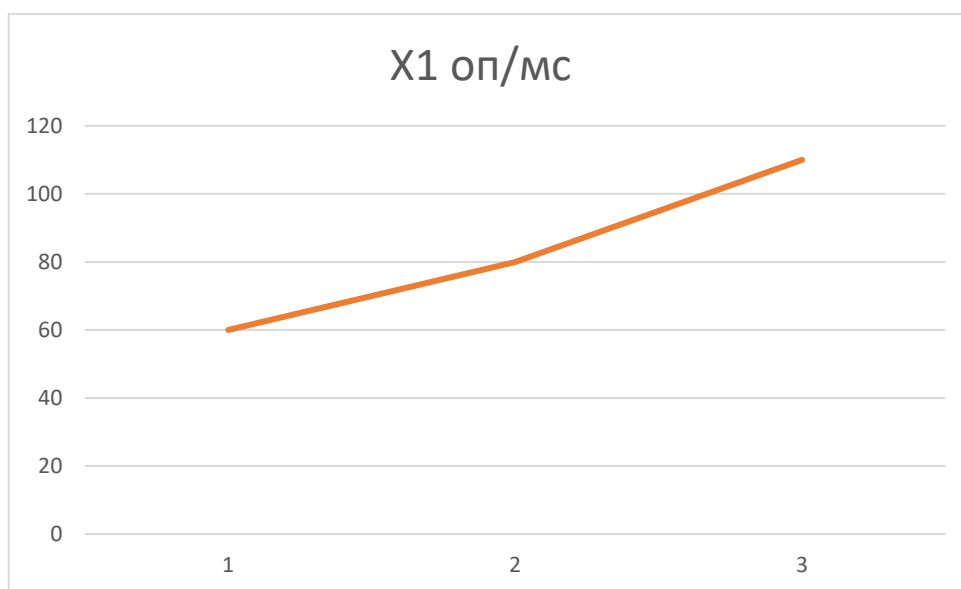


Рисунок 4.2 – X1, швидкодія мови програмування

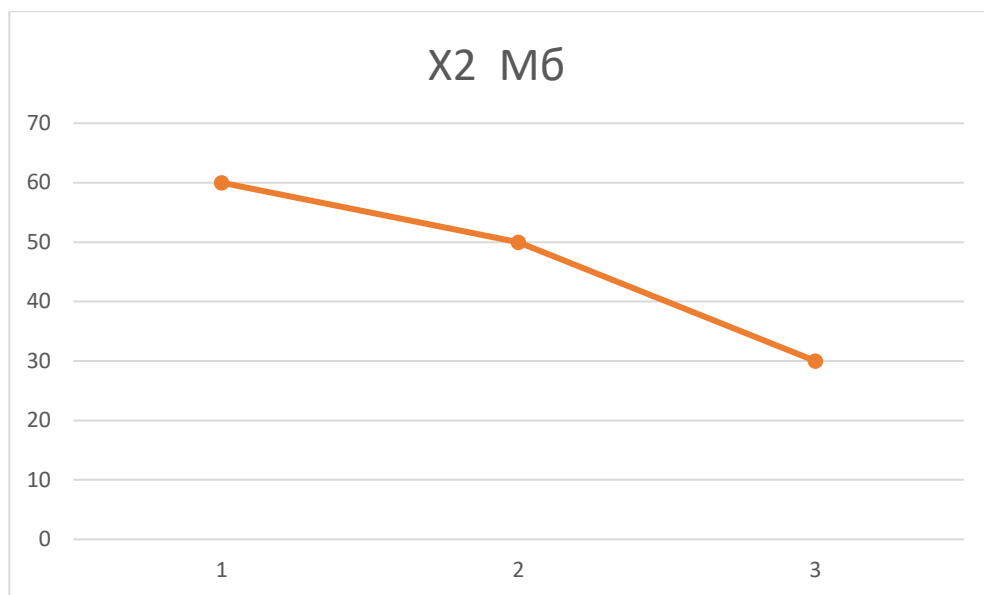


Рисунок 4.3 – X2, об'єм пам'яті

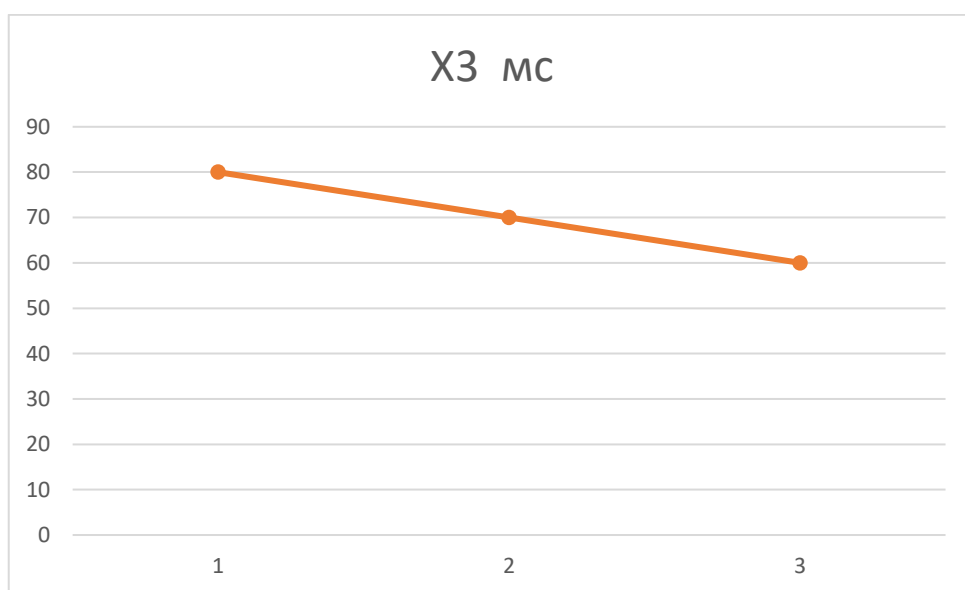


Рисунок 4.4 – X3, час попередньої обробки даних

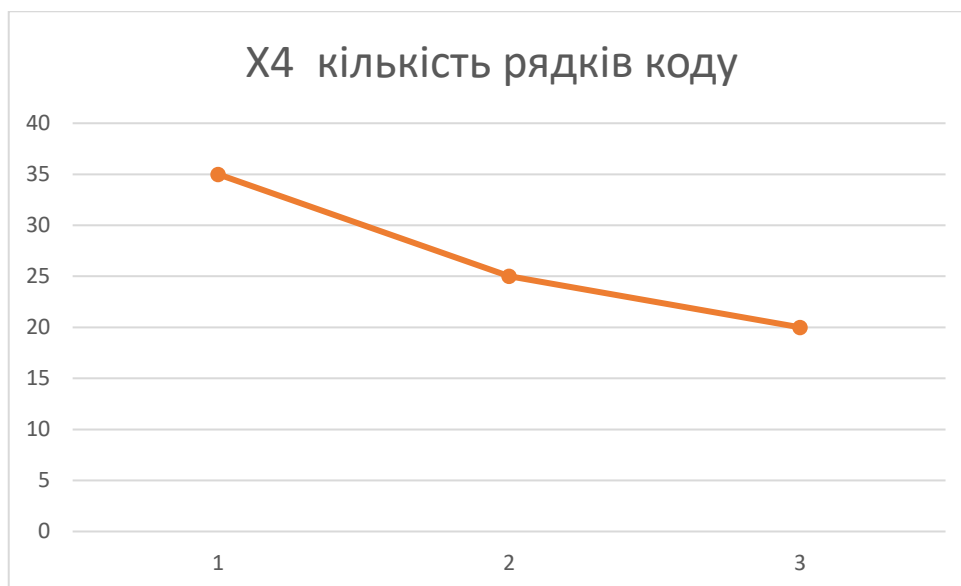


Рисунок 4.5 – X4, потенційний об'єм програмного коду

4.4 Аналіз експертного оцінювання параметрів

Після детального обговорення й аналізу кожний експерт оцінює ступінь важливості кожного параметру для конкретно поставленої цілі – розробка програмного продукту, який дає найбільш точні результати при знаходженні параметрів моделей адаптивного прогнозування і обчислення прогнозних значень.

Значимість кожного параметра визначається методом попарного порівняння. Оцінку проводить експертна комісія із 7 людей. Визначення коефіцієнтів значимості передбачає:

- визначення рівня значимості параметра шляхом присвоєння різних рангів;
- перевірку придатності експертних оцінок для подальшого використання;
- визначення оцінки попарного пріоритету параметрів;

– обробку результатів та визначення коефіцієнту значимості.

Результати експертного ранжування наведені у таблиці 4.3.

Таблиця 4.3 – Експертне ранжування.

Позначення параметра	Назва параметра	Одиниці виміру	Ранг параметра за оцінкою експерта							Сума рангів R_i	Відхилення Δ_i	Δ_i^2
			1	2	3	4	5	6	7			
X1	Швидкодія мови програмування	Оп/мс	1	2	2	1	1	1	2	10	-7.5	56.25
X2	Об'єм пам'яті	Мб	3	4	3	3	4	3	4	24	6.5	42.25
X3	Час попередньої обробки даних	Мс	2	1	1	2	2	2	1	11	-6.5	42.25
X4	Потенційний об'єм програмного коду	Кількість рядків коду	4	3	4	4	3	4	3	25	7.5	56.25
	Разом		10	10	10	10	10	10	10	70	0	197

Для перевірки степені достовірності експертних оцінок, визначимо наступні параметри:

а) сума рангів кожного з параметрів і загальна сума рангів:

$$R_i = \sum_{j=1}^N r_{ij} R_{ij} = \frac{Nn(n+1)}{2} = 70, \quad (4.1)$$

де N – число експертів, n – кількість параметрів;

б) середня сума рангів:

$$T = \frac{1}{n} R_{ij} = 17,5 \quad (4.2)$$

в) відхилення суми рангів кожного параметра від середньої суми рангів:

$$\Delta_i = R_i - T. \quad (4.3)$$

Сума відхилень по всіх параметрам повинна дорівнювати 0;

г) загальна сума квадратів відхилення:

$$S = \sum_{i=1}^N \Delta_i^2 = 197. \quad (4.4)$$

Порахуємо коефіцієнт узгодженості:

$$W = \frac{12S}{N^2(n^3 - n)} = \frac{12 \cdot 197}{7^2(4^3 - 4)} = 0,754 > W_k = 0,67. \quad (4.5)$$

Ранжування можна вважати достовірним, тому що знайдений коефіцієнт узгодженості перевищує нормативний, котрий дорівнює 0,67.

Скориставшись результатами ранжирування, проведемо попарне порівняння всіх параметрів і результати занесемо у таблицю 4.4.

Таблиця 4.4 - Попарне порівняння параметрів.

Параметри	Експерти							Кінцева оцінка	Числове значення
	1	2	3	4	5	6	7		
X1 і X2	<	<	<	<	<	<	<	<	0,5
X1 і X3	<	>	>	<	<	<	>	<	0,5
X1 і X4	<	<	<	<	<	<	<	<	0,5
X2 і X3	>	>	>	>	>	>	>	>	1,5
X2 і X4	<	>	<	<	>	<	<	<	0,5
X3 і X4	<	<	<	<	<	<	<	<	0,5

Числове значення, що визначає ступінь переваги i -го параметра над j -тим, a_{ij} визначається по формулі:

$$a_{ij} = \begin{cases} 1.5 \text{ при } X_i > X_j \\ 1.0 \text{ при } X_i = X_j \\ 0.5 \text{ при } X_i < X_j \end{cases} \quad (4.6)$$

З отриманих числових оцінок переваги складемо матрицю $A = \|a_{ij}\|$.

Для кожного параметра зробимо розрахунок вагомості K_{bi} за наступними формулами:

$$K_{bi} = \frac{b_i}{\sum_{i=1}^n b_i} \quad (4.7)$$

$$b_i = \sum_{j=1}^N a_{ij} \quad (4.8)$$

Відносні оцінки розраховуються декілька разів доти, поки наступні значення не будуть незначно відрізнятися від попередніх (менше 2%). На другому і наступних кроках відносні оцінки розраховуються за наступними формулами:

$$K_{bi} = \frac{b'_i}{\sum_{i=1}^n b'_i}, \quad (4.9)$$

$$b'_i = \sum_{j=1}^N a_{ij} b_j \quad (4.10)$$

Як видно з таблиці 4.5, різниця значень коефіцієнтів вагомості не перевищує 2%, тому більшої кількості ітерацій не потрібно.

Таблиця 4.5 - Розрахунок вагомості параметрів

Параметри x_i	Параметри x_j				Перша ітер.		Друга ітер.		Третя ітер	
	X1	X2	X3	X4	b_i	K_{Bi}	b_i^1	K_{Bi}^1	b_i^2	K_{Bi}^2
X1	1	0,5	0,5	0,5	2,5	0,16	9,25	0,16	34,125	0,16
X2	1,5	1	1,5	0,5	4,5	0,28	16,25	0,28	59,125	0,28
X3	1,5	0,5	1	0,5	3,5	0,22	12,25	0,21	41,875	0,2
X4	1,5	1,5	1,5	1	5,5	0,34	21,25	0,35	77,875	0,36
Всього:					16	1	59	1	213	1

4.5 Аналіз рівня якості варіантів реалізації функцій

Визначаємо рівень якості кожного варіанту виконання основних функцій окремо.

Абсолютні значення параметрів $X2$ (Об'єм пам'яті), $X3$ (час попередньої обробки даних) та $X4$ (потенційний об'єм програмного коду) відповідають технічним вимогам умов функціонування даного ПП.

Абсолютне значення параметра $X1$ (швидкість роботи мови програмування) обрано не найгіршим.

Коефіцієнт технічного рівня для кожного варіанта реалізації ПП розраховується так (таблиця 4.6):

$$K_K(j) = \sum_{i=1}^n K_{ei,j} B_{i,j}, \quad (4.11)$$

де n – кількість параметрів;

K_{ei} – коефіцієнт вагомості i -го параметра;

B_i – оцінка i -го параметра в балах.

Таблиця 4.6 - Розрахунок показників рівня якості варіантів реалізації основних функцій ПП

Основні функції	Варіант реалізації функції	Параметри	Абсолютне значення параметра	Бальна оцінка параметра	Коефіцієнт вагомості параметра	Коефіцієнт рівня якості
F1	A	X1	100	25	0,16	4
F3	A	X2	87	29	0,28	8,12
	Б	X3	27	19	0,2	3,8
F4	A	X4	25	23	0,36	8,28

За даними з таблиці 4.6 за формулою:

$$K_K = K_{Ty}[F_{1k}] + K_{Ty}[F_{2k}] + \dots + K_{Ty}[F_{zk}], \quad (4.12)$$

визначаємо рівень якості кожного з варіантів:

$$K_{K1} = 4 + 8,12 + 8,28 = 20,4 ;$$

$$K_{K2} = 4 + 3,8 + 8,28 = 16,08 .$$

Як видно з розрахунків, кращим є 2 варіант, для якого коефіцієнт технічного рівня має найбільше значення.

4.6 Економічний аналіз варіантів розробки ПП

Для визначення вартості розробки ПП спочатку проведемо розрахунок трудомісткості.

Всі варіанти включають в себе два окремих завдання:

1. Розробка проекту програмного продукту;
2. Розробка програмної оболонки;

Завдання 1 за ступенем новизни відноситься до групи А, завдання 2 – до групи Б. За складністю алгоритми, які використовуються в завданні 1 належать до групи 1; а в завданні 2 – до групи 3.

Для реалізації завдання 1 використовується довідкова інформація, а завдання 2 використовує інформацію у вигляді даних.

Проведемо розрахунок норм часу на розробку та програмування для кожного з завдань.

Загальна трудомісткість обчислюється як:

$$T_0 = T_P \cdot K_P \cdot K_{СК} \cdot K_M \cdot K_{СТ} \cdot K_{СТ.М}, \quad (4.13)$$

де T_P – трудомісткість розробки ПП;

K_P – поправочний коефіцієнт;

$K_{СК}$ – коефіцієнт на складність вхідної інформації;

K_M – коефіцієнт рівня мови програмування;

$K_{СТ}$ – коефіцієнт використання стандартних модулів і прикладних програм;

$K_{СТ.М}$ – коефіцієнт стандартного математичного забезпечення

Для першого завдання, виходячи із норм часу для завдань розрахункового характеру ступеню новизни А та групи складності алгоритму 1, трудомісткість дорівнює: $T_P = 37$ людино-днів. Поправочний

коефіцієнт, який враховує вид нормативно-довідкової інформації для першого завдання: $K_{\Pi} = 1.8$. Поправочний коефіцієнт, який враховує складність контролю вхідної та вихідної інформації для всіх семи завдань рівний 1: $K_{СК} = 1$. Оскільки при розробці першого завдання використовуються стандартні модулі, врахуємо це за допомогою коефіцієнта $K_{СТ} = 0.9$. Тоді загальна трудомісткість програмування першого завдання дорівнює:

$$T_1 = 37 \cdot 1.8 \cdot 0.9 = 59,94 \text{ людино-днів.}$$

Проведемо аналогічні розрахунки для подальших завдань.

Для другого завдання (використовується алгоритм третьої групи складності, степінь новизни Б), тобто $T_p = 29$ людино-днів, $K_{\Pi} = 0.9$, $K_{СК} = 1$, $K_{СТ} = 0.8$:

$$T_2 = 29 \cdot 0.9 \cdot 0.8 = 20.88 \text{ людино-днів.}$$

Складаємо трудомісткість відповідних завдань для кожного з обраних варіантів реалізації програми, щоб отримати їх трудомісткість:

$$T_I = (59,94 + 20,88 + 4,8 + 20,88) \cdot 8 = 852 \text{ людино-годин.}$$

$$T_{II} = (59,94 + 20,88 + 6,91 + 20,88) \cdot 8 = 868,88 \text{ людино-годин.}$$

Найбільш високу трудомісткість має варіант II.

В розробці беруть участь два програмісти з окладом 27000 грн., один аналітик в області даних з окладом 29000. Визначимо середню зарплату за годину за формулою:

$$C_{\text{ч}} = \frac{M}{T_m \cdot t} \text{ грн.}, \quad (4.14)$$

де M – місячний оклад працівників;

T_m – кількість робочих днів тиждень;

t – кількість робочих годин в день.

$$C_{\text{ч}} = \frac{27000 + 27000 + 29000}{3 \cdot 21 \cdot 8} = 164,68 \text{ грн.} \quad (4.15)$$

Тоді, розрахуємо заробітну плату за формулою:

$$C_{\text{зп}} = C_{\text{ч}} \cdot T_i \cdot K_{\text{д}}, \quad (4.16)$$

де $C_{\text{ч}}$ – величина погодинної оплати праці програміста;

T_i – трудомісткість відповідного завдання;

$K_{\text{д}}$ – норматив, який враховує додаткову заробітну плату.

Зарплата розробників за варіантами становить:

$$\text{I. } C_{\text{зп}} = 164,68 \cdot 852 \cdot 1,2 = 168368,83 \text{ грн.}$$

$$\text{II. } C_{\text{зп}} = 164,68 \cdot 868,88 \cdot 1,2 = 171704,59 \text{ грн.}$$

Відрахування на єдиний соціальний внесок становить 22%:

$$\text{I. } C_{\text{від}} = C_{\text{зп}} \cdot 0,22 = 168368,83 \cdot 0,22 = 37041,41 \text{ грн.}$$

$$\text{II. } C_{\text{від}} = C_{\text{зп}} \cdot 0,22 = 171704,59 \cdot 0,22 = 37775,01 \text{ грн.}$$

Тепер визначимо витрати на оплату однієї машино-години. ($C_{\text{м}}$)

Так як одна ЕОМ обслуговує одного програміста з окладом 27000 грн., з коефіцієнтом зайнятості 0,2 то для однієї машини отримаємо:

$$C_{\Gamma} = 12 \cdot M \cdot K_3 = 12 \cdot 27000 \cdot 0,2 = 64800 \text{ грн.}$$

З урахуванням додаткової заробітної плати:

$$C_{3П} = C_{\Gamma} \cdot (1 + K_3) = 64800 \cdot (1 + 0.2) = 77760 \text{ грн.}$$

Відрахування на соціальний внесок:

$$C_{ВІД} = C_{3П} \cdot 0.22 = 77760 \cdot 0,22 = 17107,2 \text{ грн.}$$

Амортизаційні відрахування розраховуємо при амортизації 25% та вартості ЕОМ – 10000 грн.

$$C_A = K_{TM} \cdot K_A \cdot Ц_{ПР} = 1.4 \cdot 0.25 \cdot 10000 = 3500 \text{ грн.,}$$

де K_{TM} – коефіцієнт, який враховує витрати на транспортування та монтаж приладу у користувача;

K_A – річна норма амортизації;

$Ц_{ПР}$ – договірна ціна приладу.

Витрати на ремонт та профілактику розраховуємо як:

$$C_P = K_{TM} \cdot Ц_{ПР} \cdot K_P = 1.4 \cdot 10000 \cdot 0.08 = 1120 \text{ грн.,}$$

де K_P – відсоток витрат на поточні ремонти.

Ефективний годинний фонд часу ПК за рік розраховуємо за формулою:

$$T_{\text{ЕФ}} = (D_K - D_B - D_C - D_P) \cdot t_3 \cdot K_B = (365 - 104 - 12 - 16) \cdot 8 \cdot 0.35 = 627,2 \text{ години,}$$

де D_K – календарна кількість днів у році;

D_B, D_C – відповідно кількість вихідних та святкових днів;

D_P – кількість днів планових ремонтів устаткування;

t – кількість робочих годин в день;

K_B – коефіцієнт використання приладу у часі протягом зміни.

Витрати на оплату електроенергії розраховуємо за формулою:

$$C_{\text{ЕЛ}} = T_{\text{ЕФ}} \cdot N_C \cdot K_3 \cdot C_{\text{ЕН}} = 627,2 \cdot 0,43 \cdot 0,3 \cdot 5,23 = 423,15 \text{ грн.,}$$

де N_C – середньо-споживча потужність приладу;

K_3 – коефіцієнтом зайнятості приладу;

$C_{\text{ЕН}}$ – тариф за 1 КВт-годин електроенергії.

Накладні витрати розраховуємо за формулою:

$$C_H = C_{\text{ПР}} \cdot 0.67 = 10000 \cdot 0,67 = 6700 \text{ грн.}$$

Тоді, річні експлуатаційні витрати будуть:

$$C_{\text{ЕКС}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_A + C_P + C_{\text{ЕЛ}} + C_H, \quad (4.17)$$

$$C_{\text{ЕКС}} = 77760 + 17107,2 + 3500 + 1120 + 423,15 + 6700 = 106610,35 \text{ грн.}$$

Собівартість однієї машино-години ЕОМ дорівнюватиме:

$$C_{\text{М-Г}} = C_{\text{ЕКС}} / T_{\text{ЕФ}} = 106610,35 / 627,2 = 169,98 \text{ грн/год.}$$

Оскільки в даному випадку всі роботи, які пов'язані з розробкою програмного продукту ведуться на ЕОМ, витрати на оплату машинного часу, в залежності від обраного варіанта реалізації, складає:

$$C_{\text{М}} = C_{\text{М-Г}} \cdot T, \quad (4.18)$$

$$\text{I. } C_{\text{М}} = 169,98 \cdot 852 = 144822,96 \text{ грн.}$$

$$\text{II. } C_{\text{М}} = 169,98 \cdot 868,88 = 147692,22 \text{ грн.}$$

Накладні витрати складають 67% від заробітної плати:

$$C_{\text{Н}} = C_{\text{ЗП}} \cdot 0,67, \quad (4.19)$$

$$\text{I. } C_{\text{Н}} = 168368,83 \cdot 0,67 = 112807,12 \text{ грн.}$$

$$\text{II. } C_{\text{Н}} = 171704,59 \cdot 0,67 = 115042,08 \text{ грн.}$$

Отже, вартість розробки ПП за варіантами становить:

$$C_{\text{ПП}} = C_{\text{ЗП}} + C_{\text{ВІД}} + C_{\text{М}} + C_{\text{Н}}, \quad (4.20)$$

$$\text{I. } C_{\text{III}} = 168368,83 + 37041,41 + 144822,96 + 112807,12 = 463040,32 \text{ грн.}$$

$$\text{II. } C_{\text{III}} = 171704,59 + 37775,01 + 147692,22 + 115042,08 = 472213,9 \text{ грн.}$$

4.7 Вибір кращого варіанту ПП техніко-економічного рівня

Розрахуємо коефіцієнт техніко-економічного рівня за формулою:

$$K_{\text{ТЕР}j} = K_{kj} / C_{\Phi j}, \quad (4.21)$$

$$K_{\text{ТЕР}1} = 20,4 / 463040,32 = 4,405 \cdot 10^{-5},$$

$$K_{\text{ТЕР}2} = 16,08 / 472213,9 = 3,405 \cdot 10^{-5}.$$

Як бачимо, найбільш ефективним є перший варіант реалізації програми з коефіцієнтом техніко-економічного рівня $K_{\text{ТЕР}1} = 6,851 \cdot 10^{-5}$.

Після виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, можна зробити висновок, що з альтернатив, що залишились після першого відбору двох варіантів виконання програмного комплексу оптимальним є перший варіант реалізації програмного продукту. У нього виявився найкращий показник техніко-економічного рівня якості.

$$K_{\text{ТЕР}} = 6,851 \cdot 10^{-5}.$$

Цей варіант реалізації програмного продукту має такі параметри:

– вибір програмного продукту – Eviews;

- реалізація важливої постановки з допомогою вбудованих функцій;
- використання стандартного інтерфейсу для побудови значень.

Даний варіант виконання програмного комплексу дає користувачу зручний інтерфейс, швидку реалізацію програми та доступний функціонал для роботи.

4.8 Висновки до розділу 4

У даному розділі було проведено повний функціонально-вартісний аналіз програмного продукту. Також було знайдено оцінку основних функцій програмного продукту.

В результаті виконання функціонально-вартісного аналізу програмного комплексу що розроблюється, було визначено та проведено оцінку основних функцій програмного продукту, а також знайдено параметри, які його характеризують.

На основі аналізу вибрано варіант реалізації програмного продукту.

ВИСНОВКИ

У даній дипломній роботі було розглянуто та реалізовано веб-додаток для розпізнавання характеристик автомобіля за зображенням на основі згорткових нейронних мереж. Під час роботи були проведені детальні дослідження та реалізації, які дозволили досягти значних результатів у цій галузі.

Початковий аналіз предметної області показав високу актуальність технологій розпізнавання зображень у автомобільній індустрії. Зокрема, було встановлено, що такі технології значно сприяють автоматизації процесів, підвищенню точності та ефективності обробки даних. Проте, інтеграція високоточних систем розпізнавання у веб-додатки виявилася викликом, оскільки це вимагає забезпечення стабільної продуктивності та сумісності з різними пристроями та умовами використання.

У теоретичній частині роботи було детально розглянуто основи машинного навчання та згорткових нейронних мереж, які є основою для розробки таких систем. Особливу увагу було приділено архітектурі ResNet-34, яка використовувалася у проекті. Було проведено аналіз методів попередньої обробки зображень, таких як нормалізація та доповнення даних, що дозволило значно покращити точність моделі. Також були розглянуті різні методи класифікації зображень та метрики оцінювання, які використовувалися для оцінки ефективності моделі.

Програмна реалізація веб-додатку включала вибір мови програмування Python та веб-фреймворку Flask, які забезпечили гнучкість та простоту інтеграції з бібліотеками машинного навчання, такими як PyTorch. Було описано процес налаштування середовища розробки, структуру веб-додатку та інтеграцію моделі ЗНМ для обробки та класифікації зображень у реальному часі. Крім того, було розроблено інтуїтивно зрозумілий інтерфейс

користувача, який дозволяє завантажувати зображення автомобілів та отримувати точні прогнози щодо їх характеристик.

Розроблений веб-додаток показав високу точність у розпізнаванні характеристик автомобілів за зображеннями, що підтверджує ефективність використання згорткових нейронних мереж у таких задачах. Реалізація проекту продемонструвала можливість інтеграції передових методів машинного навчання у веб-середовище, забезпечуючи зручність та доступність для кінцевих користувачів.

Результати даної роботи також вказують на можливість подальшого розвитку та вдосконалення систем розпізнавання зображень. Зокрема, майбутні дослідження можуть бути спрямовані на розширення функціональності веб-додатку, додавання підтримки інших типів транспортних засобів, а також інтеграцію додаткових моделей машинного навчання для підвищення точності та швидкості обробки. Оптимізація продуктивності, поліпшення алгоритмів попередньої обробки зображень та використання більш потужних обчислювальних ресурсів або хмарних сервісів також є перспективними напрямками для подальшого вдосконалення.

Дослідження також підкреслює важливість покращення користувацького досвіду. Розробка мобільних версій додатку, додавання функцій збереження історії завантажень та прогнозів значно покращать зручність та ефективність використання системи для кінцевих користувачів.

Таким чином, дана дипломна робота показала значні можливості та перспективи використання згорткових нейронних мереж для автоматизованого розпізнавання характеристик автомобілів. Робота забезпечила основу для подальших досліджень та вдосконалень у цій галузі, підтвердивши ефективність та доцільність використання машинного навчання у сучасних веб-додатках для розпізнавання зображень.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. An Introduction to Convolutional Neural Networks (CNNs), 2023, URL: <https://www.datacamp.com/tutorial/introduction-to-convolutional-neural-networks-cnns>
2. Deep Learning with Python by François Chollet, 2021, URL: <https://tanthiamhuat.wordpress.com/wp-content/uploads/2018/03/deeplearningwithpython.pdf>
3. Leiyu Chen, Shaobo Li, Qiang Bai, Jing Yang, Sanlong Jiang, Yanming Miao, Review of Image Classification Algorithms Based on Convolutional Neural Networks, Remote Sensing, Volume 13, Issue 22, 2021, 51 p. URL: <https://www.mdpi.com/2072-4292/13/22/4712>
4. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. URL: <https://www.deeplearningbook.org/>
5. Wikipedia – Машинне Навчання, URL: https://en.wikipedia.org/wiki/Machine_learning
6. Guru99 - Машинне навчання без нагляду: Алгоритми, Типи з прикладом URL: <https://www.guru99.com/uk/supervised-vs-unsupervised-learning.html>
7. Навчання з підкріпленням у машинному навчанні, 2019, URL: <https://evergreens.com.ua/ua/articles/reinforcement-learning.html>
8. Understanding and visualizing ResNets, 2018, URL: <https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8>
9. Why Python for Machine Learning? URL: <https://pythonbasics.org/why-python-for-machine-learning/>

- 10.6 Reasons Why Is Python Used for Machine Learning URL: <https://www.newhorizons.com/resources/blog/why-is-python-used-for-machine-learning>
11. Python URL: <https://docs.python.org/3.11/>
12. Pillow URL: <https://pillow.readthedocs.io/en/stable/>
13. Flask URL: <https://flask.palletsprojects.com/en/3.0.x/>
14. Understanding Flask Framework: Installation, features & Expert Insights URL: <https://www.analyticsvidhya.com/blog/2021/10/flask-python/>
15. PyTorch URL: <https://pytorch.org/>
16. NumPy URL: <https://numpy.org/doc/>
17. Pandas URL: <https://pandas.pydata.org/docs/>
18. Matplotlib URL: <https://matplotlib.org/stable/index.html>
19. Coursera: Deep Neural Networks with PyTorch URL: <https://www.coursera.org/learn/deep-neural-networks-with-pytorch>
20. HTML5 & CSS3 Fundamentals URL: <https://learn.microsoft.com/en-us/shows/html5-css3-fundamentals-development-for-absolute-beginners/>
21. Stanford Cars Dataset URL:
<https://www.kaggle.com/datasets/jessicali9530/stanford-cars-dataset>

ДОДАТОК А. ЛІСТИНГ ПРОГРАМИ

Модуль app.py

```
from flask import Flask, request, render_template
from PIL import Image
import os
import torchvision.transforms as transforms
import torch

dataset_dir = "./static/stanford-car-dataset-by-classes-
folder/car_data/car_data/"

app = Flask(__name__)

# Завантаження моделі
model = torch.load('model.pth')
model.eval() # Важливо встановити модель в режим оцінки після
завантаження

# Завантаження назв класів
def find_classes(dir):
    classes = os.listdir(dir)
    classes.sort()
    return classes
classes = find_classes(dataset_dir+"train")

# Функція для класифікації зображення
def classify_image(image):
    # Перетворення зображення
    loader = transforms.Compose([
        transforms.Resize((400, 400)),
        transforms.ToTensor(),
        transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
    ])
```

```

    image = loader(image).float()
    image = torch.autograd.Variable(image, requires_grad=True)
    image = image.unsqueeze(0)
    image = image.cuda() if torch.cuda.is_available() else image.cpu()
# Перевірка доступності CUDA
    output = model(image)

    # Отримання індексу класу з найбільшою вірогідністю
    _, predicted = torch.max(output.data, 1)
    return classes[predicted.item()]

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/classify', methods=['POST'])
def classify_route():
    # Отримання зображення з POST-запиту
    image_file = request.files['image']
    image = Image.open(image_file)

    # Класифікація зображення
    class_name = classify_image(image)
    path = dataset_dir + 'test/' + class_name
    file_list = os.listdir(path)
    image_path = os.path.join(path, file_list[1])
    class_name = class_name.split()
    model_name = ' '.join(class_name[1:-1])

    # Повернення назви класу у відповідь
    return render_template('index.html', class_name=class_name,
model_name=model_name, image_path=image_path)

if __name__ == '__main__':
    app.run(debug=True)

```

Модуль main.ipynb

```

import matplotlib.pyplot as plt
import numpy as np

import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.models as models
import torchvision.transforms as transforms

import time
import os
import PIL.Image as Image
from IPython.display import display

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)
print(torch.cuda.get_device_name(device))

dataset_dir =      "./static/stanford-car-dataset-by-classes-
folder/car_data/car_data/"

train_tfms = transforms.Compose([transforms.Resize((400, 400)),
                                transforms.RandomHorizontalFlip(),
                                transforms.RandomRotation(15),
                                transforms.ToTensor(),
                                transforms.Normalize((0.5, 0.5, 0.5),
(0.5, 0.5, 0.5))]
test_tfms = transforms.Compose([transforms.Resize((400, 400)),
                                transforms.ToTensor(),
                                transforms.Normalize((0.5, 0.5, 0.5),
(0.5, 0.5, 0.5))]

dataset = torchvision.datasets.ImageFolder(root=dataset_dir+"train",
transform = train_tfms)

```

```

trainloader = torch.utils.data.DataLoader(dataset, batch_size = 32,
shuffle=True, num_workers = 2)

dataset2 = torchvision.datasets.ImageFolder(root=dataset_dir+"test",
transform = test_tfms)
testloader = torch.utils.data.DataLoader(dataset2, batch_size = 32,
shuffle=False, num_workers = 2)

def train_model(model, criterion, optimizer, scheduler, n_epochs = 5):
    losses = []
    accuracies = []
    test_accuracies = []
    # set the model to train mode initially
    model.train()
    for epoch in range(n_epochs):
        since = time.time()
        running_loss = 0.0
        running_correct = 0.0
        for i, data in enumerate(trainloader, 0):

            # get the inputs and assign them to cuda
            inputs, labels = data
            inputs = inputs.to(device)
            labels = labels.to(device)
            optimizer.zero_grad()

            # forward + backward + optimize
            outputs = model(inputs)
            _, predicted = torch.max(outputs.data, 1)
            loss = criterion(outputs, labels)
            loss.backward()
            optimizer.step()

            # calculate the loss/acc later
            running_loss += loss.item()
            running_correct += (labels==predicted).sum().item()

```

```

epoch_duration = time.time()-since
epoch_loss = running_loss/len(trainloader)
epoch_acc = 100/32*running_correct/len(trainloader)
print("Епоха %s, тривалість: %d с, втрати: %.4f, точність:
%.4f" % (epoch+1, epoch_duration, epoch_loss, epoch_acc))

losses.append(epoch_loss)
accuracies.append(epoch_acc)

# switch the model to eval mode to evaluate on test data
model.eval()
test_acc = eval_model(model)
test accuracies.append(test_acc)

# re-set the model to train mode after validating
model.train()
scheduler.step(test_acc)
since = time.time()
print('Тренування завершено')
return model, losses, accuracies, test accuracies

def eval_model(model):
    correct = 0.0
    total = 0.0
    with torch.no_grad():
        for i, data in enumerate(testloader, 0):
            images, labels = data
            images = images.to(device)
            labels = labels.to(device)

            outputs = model_ft(images)
            _, predicted = torch.max(outputs.data, 1)

            total += labels.size(0)
            correct += (predicted == labels).sum().item()

    test_acc = 100.0 * correct / total

```

```

    print('Точність мережі на тестових зображеннях: %d %%' % (
        test_acc))
    return test_acc

model_ft = models.resnet34(pretrained=True)
num_ftrs = model_ft.fc.in_features

# replace the last fc layer with an untrained one (requires grad by
# default)
model_ft.fc = nn.Linear(num_ftrs, 196)
model_ft = model_ft.to(device)

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model_ft.parameters(), lr=0.01, momentum=0.9)

lrscheduler = optim.lr_scheduler.ReduceLROnPlateau(optimizer,
mode='max', patience=3, threshold = 0.9)

model_ft, training_losses, training_accs, test_accs =
train_model(model_ft, criterion, optimizer, lrscheduler, n_epochs=10)

# # Збереження моделі
# torch.save(model_ft, 'model.pth')

f, axarr = plt.subplots(2,2, figsize = (12, 8))
axarr[0, 0].plot(training_losses)
axarr[0, 0].set_title("Тренувальні втрати")
axarr[0, 1].plot(training_accs)
axarr[0, 1].set_title("Тренувальна точність")
axarr[1, 0].plot(test_accs)
axarr[1, 0].set_title("Тестова точність")

# tie the class indices to their names

def find_classes(dir):
    classes = os.listdir(dir)
    classes.sort()

```

```

        class_to_idx = {classes[i]: i for i in range(len(classes))}
        return classes, class_to_idx
classes, c_to_idx = find_classes(dataset_dir+"train")

# switch the model to evaluation mode to make dropout and batch norm
work in eval mode
model_ft.eval()

# transforms for the input image
loader = transforms.Compose([transforms.Resize((400, 400)),
                             transforms.ToTensor(),
                             transforms.Normalize((0.5, 0.5, 0.5),
                                                  (0.5, 0.5, 0.5))])
image = Image.open(dataset_dir+"test/Acura Integra Type R
2001/00128.jpg")
image = loader(image).float()
image = torch.autograd.Variable(image, requires_grad=True)
image = image.unsqueeze(0)
image = image.cuda()
output = model_ft(image)
conf, predicted = torch.max(output.data, 1)

# get the class name of the prediction
display(Image.open(dataset_dir+"test/Acura Integra Type R
2001/00128.jpg"))
print(classes[predicted.item()], "confidence: ", conf.item())

```

Модуль index.html

```

<!DOCTYPE html>
<html>
    <head>
        <title>Car Classification</title>
        <meta name="viewport" content="width=device-width, initial-
scale=1" />

```

```

        <link        rel="stylesheet"        href="{{        url_for('static',
filename='css/style.css')    }}">
    </head>
    <body>
        <h2>Image Classification</h2>
        <div class="form-container">
            <form action="/classify" method="post" enctype="multipart/form-
data">
                <input type="file" name="image" accept="image/*" />
                <button type="submit">Classify</button>
            </form>
        </div>
        {% if class_name %}
        <h3>Brand: <span>{{ class_name[0] }}</span></h3>
        <h3>Model: <span>{{ model_name }}</span></h3>
        <h3>Year: <span>{{ class_name[-1] }}</span></h3>
        {% endif %} {% if image_path %}
        
        {% endif %}
    </body>
</html>

```

Модуль styles.css

```

body {
    text-align: center;
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 10px;
    background-color: bisque;
}

img {
    max-width: 100%;
    height: auto;
}

```

```
.form-container {
  display: flex;
  justify-content: center;
  align-items: center;
  /* Makes the container take the minimum height of the viewport */
  flex-direction: column; /* Stacks the children vertically */
}
```

```
input[type="file"],
```

```
button {
  width: 60%; /* Adjust the width as needed */
  margin: 10px auto; /* Auto margin for horizontal centering */
  padding: 12px 20px;
  font-size: 16px;
  background-color: #ffd500;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
  transition: background-color 0.3s, transform 0.2s;
}
```

```
button:hover {
  background-color: #cca300; /* Darker shade for hover effect */
  transform: scale(1.05); /* Slightly increase size when hovered */
}
```

```
h2,
```

```
h3 {
  margin: 20px 0;
}
```

```
span {
  color: brown;
}
```

```
button:hover {  
    background-color: #b3740060; /* Darker blue on hover */  
}  
  
button:active {  
    transform: scale(0.98); /* Slightly shrink when clicked */  
}  
  
@media (max-width: 400px) {  
    body {  
        padding: 5px;  
    }  
    h2 {  
        font-size: 24px;  
    }  
}
```

ДОДАТОК Б. ГРАФІЧНИЙ МАТЕРІАЛ

Веб-додаток для розпізнавання характеристик автомобіля за зображенням на основі нейронних мереж глибокого навчання

Виконав: студент IV курсу групи КА-05 Бойко Назар Володимирович
Керівник: старший викладач Ірина Василівна Назарчук

Актуальність

Сучасні рішення в галузі розпізнавання зображень транспортних засобів досягли значних успіхів, але продовжують стикатися з істотними проблемами, особливо щодо інтеграції високоточних систем розпізнавання у веб - платформи. Багато існуючих додатків вимагають значних обчислювальних потужностей і не оптимізовані для різних умов використання Інтернету, які можуть варіюватися від стаціонарних комп'ютерів до мобільних пристроїв. Це часто призводить до компромісу між продуктивністю та доступністю, що перешкоджає ширшому застосуванню таких технологій у повсякденному онлайн-середовищі.

Мета роботи

Метою даної роботи є розробка веб -додатку, який використовує згорткові нейронні мережі для точного та швидкого розпізнавання характеристик автомобілів за їхніми зображеннями. Цей додаток повинен бути ефективним і швидко реагувати на широкому спектрі пристроїв і веб -браузерів, забезпечуючи безперебійну роботу користувачів.

Об'єкт і предмет дослідження

Об'єктом дослідження є процес розпізнавання автомобілів за зображенням.

Предметом дослідження є моделі та методи для класифікації зображень автомобілів за маркою, моделлю та роком випуску.

Постановка задачі

- Розробити складний веб-додаток, який використовує можливості згорткових нейронних мереж для розпізнавання і точної класифікації автомобілів за маркою, моделлю і роком випуску на основі завантажених користувачем зображень.
- Підвищити точність та швидкість розпізнавання автомобілів на основі зображень.
- Забезпечити надійну роботу в різних веб-середовищах.
- Зберегти зручний дизайн інтерфейсу для безперешкодної взаємодії на різних платформах.

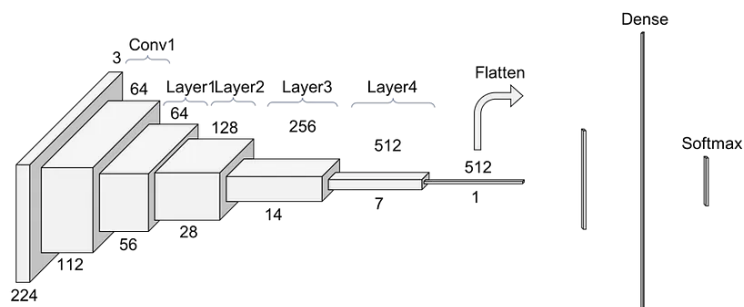
Вибір технологій

- Python: Простота використання, велика екосистема бібліотек для машинного навчання та підтримка спільноти.
- Flask: Легкий веб-фреймворк для розгортання моделей машинного навчання.
- PyTorch: Динамічний графік обчислень, попередньо навчені моделі, можливість трансферного навчання.



Архітектура ResNet34

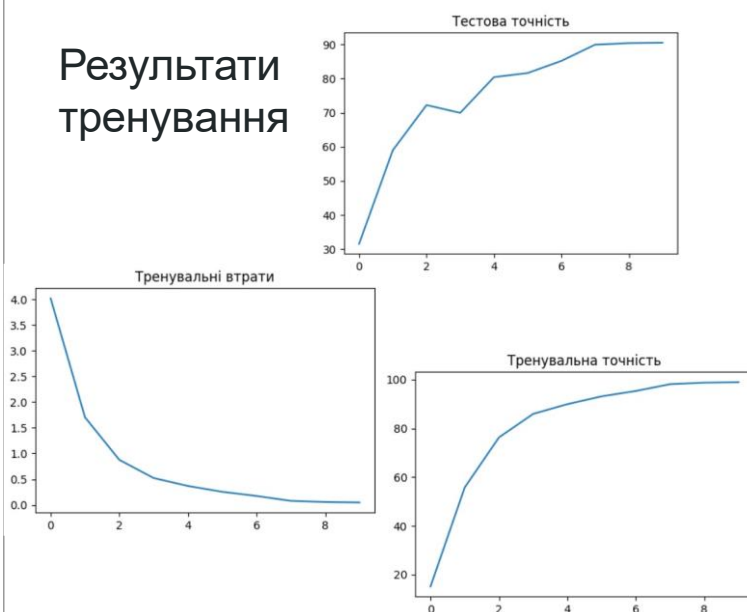
- Залишкові блоки: Ключова характеристика ResNet — це залишкові блоки, які дозволяють глибокій мережі ефективно навчатися. Кожен блок містить кілька шарів згорткових фільтрів 3×3 , за якими слідує шар нормалізації пакетів і функції активації ReLU.
- Шари згортки: Використовуються для витягування ознак із зображень. Шари згортки 3×3 застосовуються з певним кроком і без падінгу, що дозволяє зберегти просторові розміри зображення.
- Шари підвибірки: Після кількох згорткових шарів застосовуються шари підвибірки, які зменшують просторові розміри, що зменшує кількість параметрів і обчислювальних ресурсів.
- Повністю з'єднані шари: На завершальному етапі ResNet використовуються повністю з'єднані шари, які обробляють ознаки для кінцевої класифікації.



Переваги ResNet-34:

1. Поліпшена продуктивність: Завдяки залишковим зв'язкам, ResNet-34 демонструє високу продуктивність у задачах розпізнавання зображень, зменшуючи проблеми зникання градієнта.
2. Глибина: Глибока архітектура дозволяє моделі навчитися більш складним і абстрактним ознакам зображень, що підвищує точність класифікації.

Результати тренування



```

Епоха 1, тривалість: 487 с, втрати: 4.0172, точність: 15.1838
Точність мережі на тестових зображеннях: 31 %
Епоха 2, тривалість: 471 с, втрати: 1.7050, точність: 55.8088
Точність мережі на тестових зображеннях: 58 %
Епоха 3, тривалість: 469 с, втрати: 0.8769, точність: 76.3235
Точність мережі на тестових зображеннях: 72 %
Епоха 4, тривалість: 472 с, втрати: 0.5230, точність: 85.9681
Точність мережі на тестових зображеннях: 69 %
Епоха 5, тривалість: 487 с, втрати: 0.3682, точність: 89.9510
Точність мережі на тестових зображеннях: 80 %
Епоха 6, тривалість: 468 с, втрати: 0.2547, точність: 93.1618
Точність мережі на тестових зображеннях: 81 %
Епоха 7, тривалість: 473 с, втрати: 0.1739, точність: 95.3676
Точність мережі на тестових зображеннях: 85 %
Епоха 8, тривалість: 476 с, втрати: 0.0786, точність: 98.1618
Точність мережі на тестових зображеннях: 89 %
Епоха 9, тривалість: 467 с, втрати: 0.0564, точність: 98.7745
Точність мережі на тестових зображеннях: 90 %
Епоха 10, тривалість: 468 с, втрати: 0.0476, точність: 98.9706
Точність мережі на тестових зображеннях: 90 %
Тренування завершено
  
```

Структура веб-додатку

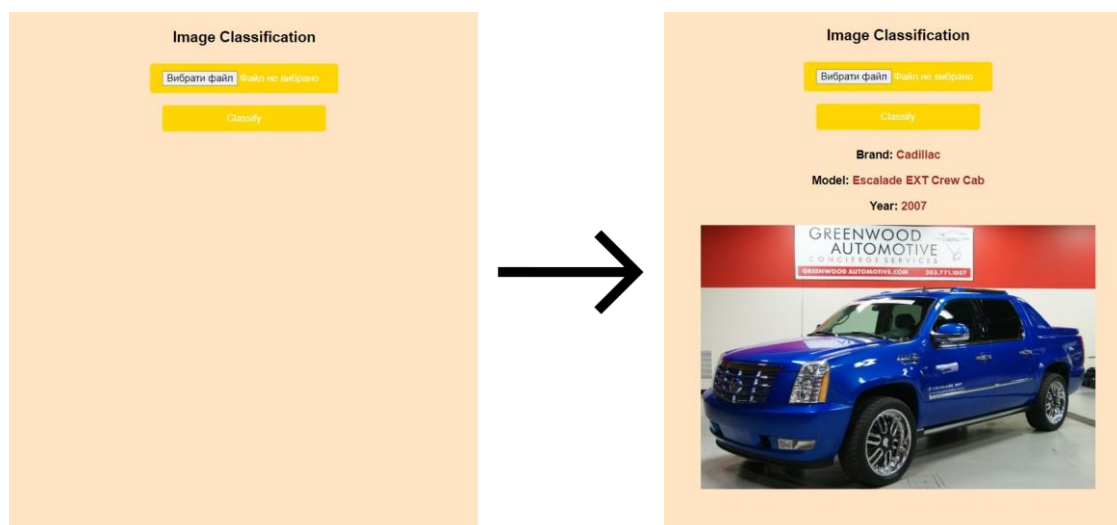
- Структура каталогу додатка:
 - app.py: Основний файл сервера.
 - templates/: HTML-файли для веб-інтерфейсу.
 - static/: CSS-файли для стилізації та тестові дані.
- Маршрути у Flask:
 - /: Головна сторінка для завантаження зображень.
 - /classify: Обробка POST-запитів, класифікація зображень за допомогою моделі CNN.

Інтеграція моделі CNN

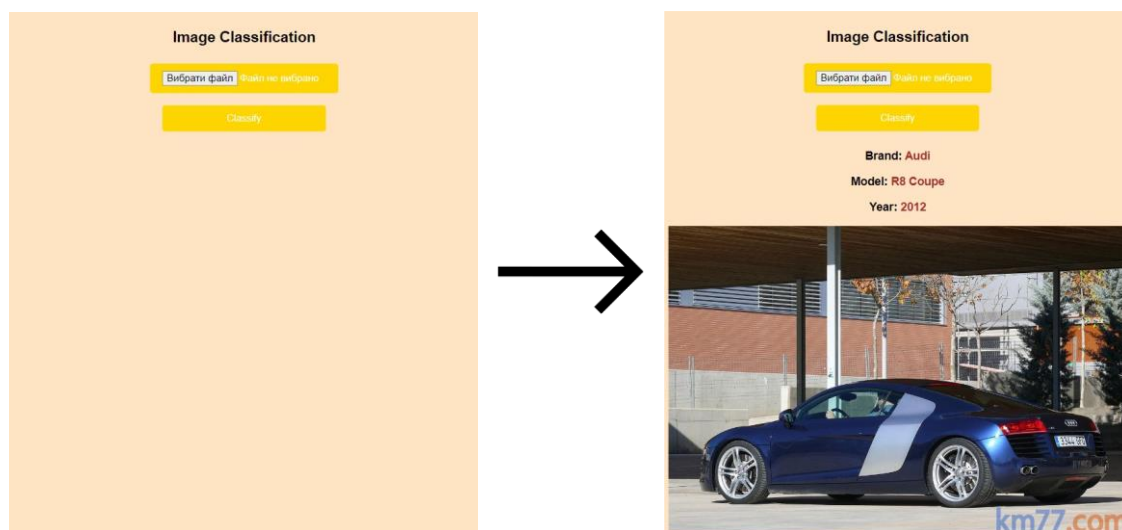
- Завантаження моделі: Завантаження збереженої моделі PyTorch (model.pth) при запуску додатка.
- Попередня обробка зображень: Зміна розміру до 400x400 пікселів, перетворення в тензор, нормалізація.
- Обробка прогнозу: Пропуск попередньо обробленого тензора через модель для отримання прогнозів, обробка вихідного тензора для визначення класу з найбільшою ймовірністю.



Демонстрація роботи веб-додатку



Демонстрація роботи веб-додатку



Висновки

1. Проаналізовано основні підходи до розпізнавання зображень автомобілів, включаючи використання згорткових нейронних мереж, такі як ResNet-34, для класифікації автомобілів за маркою, моделлю та роком випуску.
2. Розроблено веб-додаток з використанням мови програмування Python і фреймворку Flask, який дозволяє користувачам завантажувати зображення автомобілів і отримувати точні прогнози щодо їх характеристик.
3. За рахунок використання попередньо натренованої моделі ResNet-34 та ефективної аугментації даних, досягнуто високої точності та швидкості розпізнавання автомобілів на зображеннях, що забезпечує надійну роботу додатку на різних пристроях та у різних умовах.

ДЯКУЮ ЗА УВАГУ