Державний торговельно-економічний університет

Кафедра комп'ютерних наук та інформаційних систем

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«Розробка системи розпізнавання кольорових зображень»

Студента 4 курсу, 9 групи,

спеціальності
122 «Комп'ютерні науки»

_____
підпис студента

Доновський
Владислав
Олександрович

Науковий керівник
кандидат фізико-математичних наук

_____
підпис керівника

Філімонова
Тетяна
Олегівна

Гарант освітньої програми
кандидат технічних наук, доцент

_____
підпис керівника

Демідов Павло
Георгійович

Київ 2023

Державний торговельно-економічний університет

Факультет інформаційних технологій
Кафедра комп'ютерних наук та інформаційних систем Спеціальність 122 «Комп'ютерні науки»

Затверджую
Зав. кафедри _____Пурський О.І.
«12» грудня 2022р.

Завдання
на випускну кваліфікаційну роботу студенту

Доновський Владислав Олександрович
(прізвище, ім'я, по батькові)

1. Тема випускної кваліфікаційної роботи
«Розробка системи розпізнавання кольорових зображень»
Затверджена наказом ректора від *«09» грудня 2022 р. № 3332*
2. Строк здачі студентом закінченої роботи *30 травня2023 року*
3. Цільова установка та вихідні дані до роботи
Мета роботи: обґрунтування та розробка системи розпізнавання кольорових зображень.
Об'єкт дослідження: процес розробки системи розпізнавання кольорових зображень.
Предмет дослідження: засоби створення системи розпізнавання зображень.
4. Перелік графічного матеріалу_____

5. Консультанти по роботі із зазначенням розділів, за якими здійснюється консультування:

Розділ	Консультант (прізвище, ініціали)	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.
2	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.
3	Філімонова Т.О.	15.12.2022 р.	15.12.2022 р.

6. Зміст випускної кваліфікаційної роботи (перелік питань за кожним розділом)

ВСТУП

РОЗДІЛ 1 Аналітичне дослідження проблеми розпізнавання образів

1.1 Сучасні технології розпізнавання образів

1.2 Актуальність проблеми розпізнавання образів

1.3 Класифікація методів розпізнавання образів

РОЗДІЛ 2 Організація розробки системи розпізнавання образів

2.1 Обґрунтування вибору програмних засобів для розробки системи розпізнавання образів

2.2 Розробка моделі системи розпізнавання образів

2.3 Алгоритм створення системи розпізнавання образів

РОЗДІЛ 3 Реалізація розробки системи розпізнавання кольорових зображень

3.1 Програмна реалізація системи розпізнавання кольорових образів

3.2 Апробація результатів дослідження

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

7. Календарний план виконання роботи

№ Пор.	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	фактично
1	2	3	4
1	*Вибір теми випускної кваліфікаційної роботи*	*04.10.2022*	*04.10.2022*
2	*Розробка та затвердження завдання на випускну кваліфікаційну роботу*	*15.12.2022*	*15.12.2022*
3	*Вступ*	*03.02.2023*	*03.02.2023*
4	*РОЗДІЛ 1 Аналітичне дослідження проблеми розпізнавання образів*	*28.02.2023*	*28.02.2023*
5	*РОЗДІЛ 2 Організація розробки системи розпізнавання образів*	*06.04.2023*	*06.04.2023*
6	*РОЗДІЛ 3 Реалізація розробки системи розпізнавання кольорових зображень*	*12.05.2023*	*12.05.2023*
7	*Висновки*	*15.05.2023*	*15.05.2023*
8	*Здача випускної кваліфікаційної роботи на кафедру науковому керівнику*	*30.05.2023*	*30.05.2023*
9	*Попередній захист випускної кваліфікаційної роботи*	*31.05.2023 - 01.06.2023*	*31.05.2023 - 01.06.2023*
11	*Виправлення зауважень, зовнішнє рецензування випускної кваліфікаційної роботи*	*02.06.2023*	*02.06.2023*
12	*Представлення готової зшитої випускної кваліфікаційної роботи на кафедру*	*05.06.2023*	*05.06.2023*
13	*Публічний захист випускної кваліфікаційної роботи*	*За розкладом роботи ЕК*	

8. Дата видачі завдання «15» грудня 2022 р.

9. Керівник випускної кваліфікаційної роботи
Філімонова Т.О.
(прізвище, ініціали, підпис)

10. Гарант освітньої програми Демідов П.Г.
(прізвище, ініціали, підпис)

11. Завдання прийняв до виконання студент-дипломник
Доновський В.О.
(прізвище, ініціали, підпис)

12. Відгук керівника випускної кваліфікаційної роботи

У випускній кваліфікаційній роботі розроблено згорткову нейронну мережу з використанням бібліотеки keras для розпізнавання типів одягу з використанням датасету fashion mnist. Модель складається з трьох шарів, на виході десять нейронів, функція активації softmax. Тренування моделі відбувалось за десять епох, отримана точність майже 90%. Результати роботи нейронної мережі візуалізовано, зроблено висновки.

Випускна кваліфікаційна робота відповідає всім вимогам до випускних кваліфікаційних робіт. Всі поставленні завдання виконані. Випускна кваліфікаційна робота може бути допущена до захисту.

Керівник випускної кваліфікаційної роботи

_____ 30.05.2023 р.

(підпис, дата)

13. Висновок про випускну кваліфікаційну роботу

Випускна кваліфікаційна робота студента Доновський В.О.

(прізвище, ініціали)

може бути допущена до захисту в екзаменаційній комісії.

Гарант освітньої програми _____ Демідов П.Г.

(підпис, прізвище, ініціали)

Завідувач кафедри _____ Пурський О.І.

(підпис, прізвище, ініціали)

«_____»_____2023 р.

Анотація

У випускній кваліфікаційній роботі здійснено комплексну розробку моделі згорткової нейронної мережі для розпізнавання предметів одягу на мові програмування Python та із застосуванням популярних бібліотек і фреймворку з метою дослідження можливостей сучасних тенденцій та покращення особистих навичок у сфері машинного навчання. Теоретично обгрунтовано вибір поточних програмних рішень, підхід та вибір характеристик для написання згорткової нейронної мережі. Розроблено нейронну мережу у сфері комп'ютерного зору та запропоновано методи оцінки метрик точності та втрат моделі завдяки матриці помилок. Створено функцію для наглядної оцінки нейронної мережі шляхом демонстрації випадкових зображень з підписом точності передбачення.

Ключові слова: нейронна мережа, штучний інтелект, машинне навчання, модель, нейрон, шари, метрики, точність, втрати.

Anotation

The graduation qualification work includes a comprehensive development of a convolutional neural network model for recognizing clothing items using the Python programming language and utilizing popular libraries and frameworks, aiming to explore the capabilities of modern trends and enhance personal skills in the field of machine learning. The selection of current software solutions, approach, and feature choices for designing the convolutional neural network are theoretically justified. A neural network in the computer vision domain is developed, and methods for evaluating accuracy metrics and model losses using a confusion matrix are proposed. A function is created to visually assess the neural network by demonstrating random images with prediction accuracy labels.

Keywords: neural network, artificial intelligence, machine learning, model, neuron, layers, metrics, accuracy, losses.

ЗМІСТ

ВСТУП

В сучасних реаліях створення системи розпізнавання зображень є дуже актуальним завданням. Впровадження цифрової техніки, а також інтеграція комп’ютерних систем всебічно підсилює цей процес. Важливим є те, що методи цифрового розпізнавання зображень становлять значну частину загального трафіку мультисервісних мереж. [1]

Методи обробки зображень застосовуються у багатьох сферах, наприклад, медичній - розпізнавання кольорових зображень може бути використано для аналізу зображень рентгенівських знімків, сканування мозку, судин.

Також, може використовуватися штучним інтелектом для розв'язання задач комп'ютерного зору, таких як розпізнавання обличчя, визначення об'єктів на зображеннях та аналіз зображень для визначення контексту.

Вирішення наукових та інженерних завдань під час роботи з візуальними даними вимагає особливих зусиль, спираючись на знання специфічних методів. [2] Доцільним та актуальним науково-практичним завданням є діяльність, що пов’язана з удосконаленням сучасних та розробкою нових методів цифрової обробки зображень.

Всі цифрові зображення складаються з кінцевої кількості елементів, кожен з яких має певне розташування та значення. Ці елементи називаються елементами зображення, або іншими словами - пікселями. Піксель – це термін, який найчастіше використовується для позначення елементів цифрового зображення. [3] Зір є найрозвиненішим із наших почуттів, тому не дивно, що зображення відіграють найважливішу роль у людському сприйнятті. Проте слід зазначити, що людина обмежена візуальним діапазоном електромагнітного (ЕМ) спектру, а машини для обробки зображень охоплюють майже весь спектр ЕМ, починаючи від гамма до радіохвиль.

Актуальність роботи. Розробка системи розпізнавання зображень є актуальним завданням з причин описаних вище, а також продовжить набирати актуальність у майбутньому.

Мета і завдання дослідження. розробка і практична реалізація системи розпізнавання кольорових зображень.

Для досягнення поставленої мети необхідно виконати наступні **завдання**.

1. Аналіз сучасних підходів до розпізнавання зображень.
2. Порівняння існуючих методів.
3. Розробка системи розпізнавання зображень.
4. Тестування готового продукту та висновки.

Об’єкт дослідження: алгоритми та методи, що дозволяють комп'ютеру аналізувати кольорові зображення і визначати кольори на них.

Предмет дослідження: процес розробки системи розпізнавання кольорових зображень.

Методи дослідження: Основою теоретичної роботи випускного кваліфікаційного проекту є загальнонауковий аналітичний метод, а також документація офіційних розробників бібліотек та фреймворків. Було проведено систематичний аналіз статей, книг та онлайн-ресурсів, для отримати розуміння сучасних тенденцій та теоретичних основ в області машинного навчання. Також, застосоване експериментальне дослідження, проведено серію експериментів для розробки та оцінки згорткової нейронної мережі. Використано матрицю помилок для валідації та оцінки результатів роботи моделі. А також для наглядного аналізу та порівняння результатів створено функцію, яка демонструє випадкові зображення з підписом точності передбачення.

Практичне значення. Отримані результати можуть бути використані для подальшого вдосконалення нейронної мережі, а також знайти застосування у галузі пошуку або підбору одягу. Використання нейронних мереж у сфері моди дозволяє автоматизувати і прискорити різні процеси, а також значно підвищити ефективність компанії.

Структура та обсяг випускної кваліфікаційної роботи. Випускна кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаних джерел із 18 найменувань, містить 45 сторінок основного тексту, 35 рисунків.

РОЗДІЛ 1.
Аналітичне дослідження проблеми розпізнавання образів

1.1 Сучасні технології розпізнавання образів

Розпізнавання образів є важливою галуззю науки та техніки, яка дозволяє комп'ютерам автоматично розпізнавати та класифікувати зображення, відео та інші типи даних. Застосування розпізнавання образів має великий потенціал у різних галузях, включаючи медицину, безпеку, транспорт, рекламу та інші.

Однією з найбільш важливих технологій розпізнавання образів є нейронні мережі. Нейронні мережі засновані на імітації структури та функціонування людського мозку. Вони складаються з великої кількості взаємопов'язаних штучних нейронів, які співпрацюють, щоб вирішувати задачі розпізнавання образів. Нейронні мережі застосовуються для розпізнавання обличчя, класифікації зображень, аналізу відео та інших типів даних. [4]

Варто відмітити технологію глибинних нейронних мереж. Глибинні нейронні мережі складаються з багатьох шарів нейронів, що дозволяє їм ефективно розпізнавати образи з високою точністю.

Одна з ключових особливостей глибинних нейронних мереж полягає в тому, що вони можуть вчитися репрезентувати вхідні дані на різних рівнях абстракції. Наприклад, вхідні пікселі на першому шарі мережі можуть бути оброблені для визначення простих форм та контурів, а наступні шари можуть аналізувати більш складні зразки, такі як обличчя людей або тварини.

Глибинні нейронні мережі зазвичай використовуються з великими наборами даних для тренування, щоб вони могли вчитися розпізнавати зразки та здійснювати прогнози на нових даних. Тренування мережі вимагає значних обчислювальних ресурсів, таких як графічні процесори та спеціалізовані обчислювальні платформи.

Однак, одним з викликів при використанні глибинних нейронних мереж є те, що вони можуть бути досить складними для розуміння та інтерпретації. Наприклад, може бути важко розуміти, як і саме те, що робить мережа для того, щоб зробити певний прогноз або класифікацію. Використання глибинних нейронних мереж також може вимагати певних зусиль для забезпечення їх точності та надійності, а також для забезпечення безпеки та конфіденційності даних.

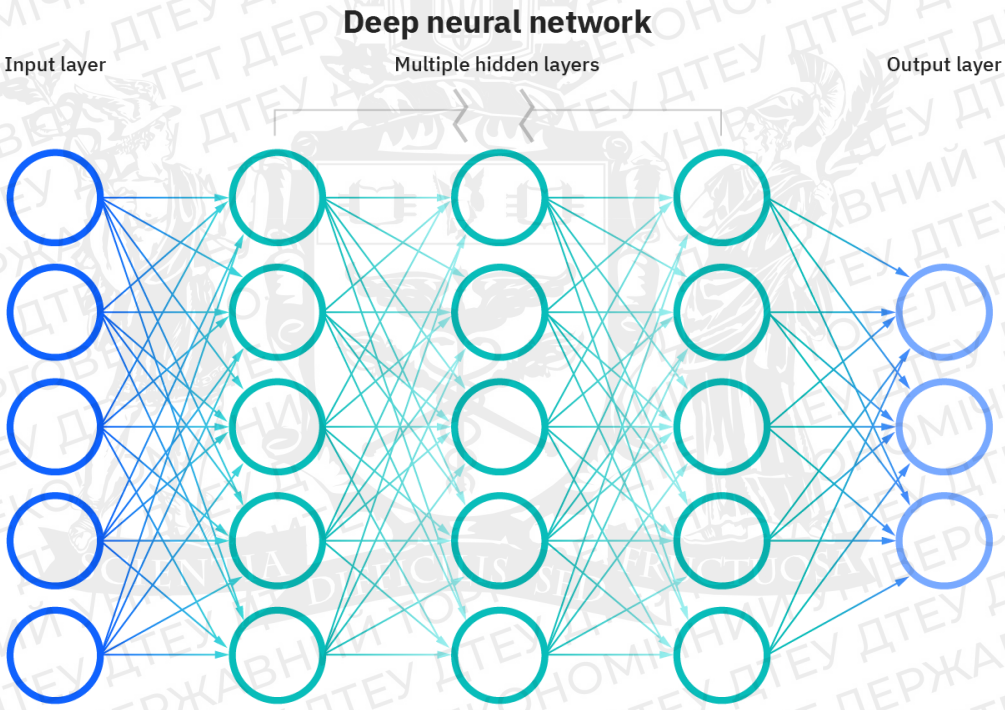

Рис. 1.1. Принцип роботи нейронної мережі

Ще однією технологією розпізнавання образів є метод опорних векторів. Цей метод використовується для класифікації даних та зображень, зокрема для розпізнавання обличчя. Метод опорних векторів заснований на виявленні оптимальної гіперплощини, яка розділяє дані на дві частини. [5] Для класифікації нових даних використовуються відстані до цієї гіперплощини.

Також у сучасних технологіях розпізнавання образів широко використовуються баєсові мережі. Баєсові мережі засновані на статистичних методах та теорії імовірностей. Вони використовуються для класифікації даних та зображень, зокрема для розпізнавання обличчя. Баєсові мережі представляють собою граф, в якому вузлами є події, а ребрами - залежності між ними. Кожен вузол містить інформацію про імовірність відбуття події, а ребра містять інформацію про умовні імовірності. Застосування баєсових мереж дозволяє здійснювати класифікацію з використанням статистичних методів та теорії імовірностей.

Іншою технологією розпізнавання образів є метод вибіркового пошуку. Цей метод використовується для пошуку об'єктів на зображеннях та відео. Він полягає в тому, що на початковому зображенні вибирається область, яка містить об'єкт, а потім виконується пошук подібних областей на інших частинах зображення. Зазвичай така модель будується на основі глибинних нейронних мереж.

Одним з ключових етапів використання методу вибіркового пошуку є формулювання цільової функції. Ця функція повинна оцінювати якість розпізнавання образів, яка залежить від значень параметрів моделі. [6]

Після того, як визначена цільова функція, метод вибіркового пошуку може бути використаний для пошуку найкращого набору параметрів моделі, який максимізує цільову функцію. Для цього на початку створюється початкова популяція індивідів, які представляють набори параметрів моделі. Ці індивіди можуть бути згенеровані випадковим чином або можуть бути вибрані на основі попереднього досвіду. Кожен індивід потім оцінюється за допомогою цільової функції, і кращі індивіди відбираються для продовження еволюції. Відбір кращих індивідів може здійснюватися за допомогою різних стратегій, наприклад, стратегія "елітного підряду", де кращі індивіди зберігаються без змін, а з інших індивідів створюються нащадки з невеликою ймовірністю зміни параметрів.

Далі виконуються етапи схрещування, мутації та заміни, які випадковим чином змінюють параметри індивідів, з метою отримання нових варіантів, що можуть мати кращі значення цільової функції. Після цього оцінюються нові індивіди за допомогою цільової функції, і цей процес повторюється до тих пір, поки не буде досягнуто певної умови зупинки, наприклад, досягнення заданої точності або досягнення заданої кількості ітерацій.

Отже, сучасні технології розпізнавання образів базуються на глибинних нейронних мережах, які використовуються для різноманітних задач, таких як класифікація зображень, детектування об'єктів та сегментація зображень. Для навчання глибинних нейронних мереж використовуються великі набори даних та різні техніки покращення ефективності навчання. Викликами у розпізнаванні образів є робота з обмеженими наборами даних та збалансування точності та швидкодії обробки, для яких використовуються різні підходи. Одним з напрямків розвитку технологій розпізнавання образів є розробка та використання методів обробки даних у режимі реального часу.

1.2 Актуальність проблеми розпізнавання образів

В сучасному світі мода є однією з найбільш динамічних та інноваційних галузей, яка постійно розвивається та змінюється. Кожного року модні тенденції змінюються, а колекції відомих брендів стають все більш складними та деталізованими. У зв'язку з цим, існує потреба в розробці ефективних систем розпізнавання образів, які можуть допомогти модним брендам та дизайнерам швидко визначати нові тенденції та дизайни.

Одним з головних елементів моди є одяг, який може бути дуже складним та деталізованим. Розпізнавання одягу може допомогти визначити стиль та дизайн одягу, а також може бути корисним у віртуальних примірках та онлайн-магазинах, де покупці можуть знайти більше інформації про продукт.

Однак, розпізнавання одягу є складною задачею, оскільки одяг може мати різні форми та кольори, а також бути в різних позах та освітленнях. Крім того, зображення одягу можуть бути частково приховані іншими об'єктами або фоном.

Важливим аспектом розпізнавання образів є точність системи. Точність є важливим фактором для багатьох систем, які використовують розпізнавання образів у своїй роботі. Вона також є критично важливим фактором у промисловості, де системи розпізнавання образів використовуються для контролю якості продукції. Також у розпізнаванні образів важливо враховувати різноманітність зображень. Зображення можуть мати різну якість та роздільну здатність, а також можуть містити шум, спотворення або бути зняті з різних ракурсів. Розробники систем розпізнавання образів повинні використовувати різноманітні алгоритми та техніки, щоб забезпечити високу точність та швидкість розпізнавання незалежно від складності зображень.

На сьогоднішній день існують деякі системи розпізнавання одягу, які використовують нейронні мережі та глибоке навчання. Однак, ці системи все ще мають деякі обмеження, такі як низьку точність розпізнавання при низькій якості зображення, або обмежену кількість категорій одягу, які можуть бути розпізнані.

Одним з прикладів застосування розпізнавання образів у сфері моди є система FashionNet, яка розроблена на базі глибоких нейронних мереж та використовується для розпізнавання категорій одягу, таких як футболки, штани, сукні та куртки. FashionNet має точність розпізнавання від 70% до 90% залежно від категорії одягу та якості зображення.

Крім того, розпізнавання одягу може бути корисним у віртуальних примірках та онлайн-магазинах, де покупці можуть використовувати функцію розпізнавання образів, щоб отримати детальніше інформацію про продукт та перевірити, як він виглядає на різних людях.

Отже, можливість розпізнавання образів у сфері моди та розпізнавання одягу є актуальною проблемою, яка має потенціал допомогти брендам та дизайнерам забезпечити ефективний аналіз та планування модних колекцій,

полегшити процес продажу та покупки одягу та підвищити якість віртуальних примірок та онлайн-магазинів.

1.3 Класифікація методів розпізнавання образів

Класифікація методів розпізнавання образів відображає різноманітні підходи до розв'язання задачі розпізнавання образів. В залежності від характеристик методів розпізнавання образів, можна виділити кілька груп методів: статистичні методи, методи машинного навчання та глибинного навчання.

Статистичні методи ґрунтуються на аналізі статистичних характеристик зображення та шаблонів. Ці методи включають методи гістограм, метод головних компонент, методи вибіркової складової та інші. Одним з найпоширеніших статистичних методів є метод Баєса. Цей метод базується на теоремі Баєса, яка використовується для обчислення ймовірності того, що зразок належить до певного класу на підставі його ознак. [7] За допомогою методу Баєса можна класифікувати зразки за допомогою великої кількості ознак, таких як кольори, текстури та форми.

Ще одним статистичним методом є метод опорних векторів (SVM). Цей метод дозволяє вирішувати задачі класифікації та регресії за допомогою створення гіперплощини, яка розділяє зразки на дві або більше груп. SVM дозволяє вирішувати задачі з високою точністю та ефективністю, що робить його популярним методом для розпізнавання образів.

Також існують методи головних компонент (PCA) та кластеризації, які використовуються для розпізнавання образів. PCA використовується для зменшення розмірності даних та виявлення головних компонент, які визначаються важливість ознак. Кластеризація використовується для групування зразків за схожими ознаками та виявлення закономірностей у даних.

Методи машинного навчання використовуються для класифікації образів шляхом навчання комп'ютерної системи на зразках зображень. Це група алгоритмів, які дозволяють комп'ютерам вчитися з даних та здійснювати прогнози на основі цих даних. Ці методи стали особливо популярними в останні роки завдяки збільшенню обчислювальної потужності та доступності великих обсягів даних. А включають такі алгоритми як метод опорних векторів, нейронні мережі, наївний баєсівський класифікатор та інші.

Методи глибинного навчання використовуються для розпізнавання образів з високою точністю та швидкістю. Ці методи використовуються для аналізу та розпізнавання складних образів, таких як зображення обличчя, рукописний текст та інші. Ці методи включають такі архітектури як глибокі нейронні мережі, конволюційні нейронні мережі та рекурентні нейронні мережі.

Крім того, методи розпізнавання образів можна розділити на кілька категорій в залежності від типу вхідних даних, таких як зображення, відео, звукові дані та інші. Зображення є одним з найпоширеніших типів вхідних даних, які використовуються для розпізнавання образів. Класифікація методів розпізнавання образів включає в себе різні методи, які можна використовувати для розв'язання задач розпізнавання образів. Кожен з цих методів має свої переваги та обмеження, які потрібно враховувати при виборі оптимального методу для конкретної задачі.

У сфері моди та розпізнавання одягу можуть бути використані різні методи розпізнавання образів. Наприклад, методи машинного навчання можуть бути використані для класифікації одягу за його типом, кольором, розміром та іншими характеристиками. Методи глибинного навчання можуть бути використані для аналізу складних шаблонів на одязі та розпізнавання моделей одягу.

Конволюційні нейронні мережі можуть бути використані для класифікації зображень одягу на підставі його зовнішнього вигляду, а рекурентні нейронні

мережі можуть бути використані для розпізнавання тексту на етикетках або описів товарів.

Основна ідея застосування конволюцій в нейронних мережах полягає у тому, що вони дозволяють враховувати локальні залежності між сусідніми пікселями на зображенні, що є важливим для багатьох задач комп'ютерного зору. Структура конволюційної нейронної мережі складається з послідовних шарів, кожен з яких здійснює певну операцію над вхідними даними. [8] Перші шари зазвичай відповідають за визначення локальних функцій ознак на основі початкових пікселів зображення. Пізніші шари використовують ці локальні функції ознак для виявлення більш складних ознак, таких як краї, текстури, об'єкти.

Рекурентні нейронні мережі призначені для обробки послідовностей даних, таких як текст, мовлення, музика. Важливою рисою є наявність зв'язків між внутрішніми станами мережі. Ці зв'язки дозволяють "пам'ятати" попередні елементи послідовності і використовувати цю інформацію при обробці наступних елементів.

Складається з одного або декількох рекурентних шарів, кожен з яких містить набір нейронів. Кожен нейрон у шарі має свій внутрішній стан, який передається в наступний часовий крок. На відміну від звичайних нейронних мереж, де всі нейрони в кожному шарі пов'язані з нейронами наступного шару, в рекурентній нейрони в одному шарі пов'язані з нейронами в тому ж шарі із попередніми часовими кроками.

Також можуть бути використані методи розпізнавання образів для автоматичного визначення модних тенденцій та аналізу поведінки споживачів. Наприклад, на основі даних про популярні моделі та кольори одягу, які були продані в певний період часу, можна зробити прогнози про те, який одяг буде популярним у майбутньому.

Таким чином, методи розпізнавання образів можуть бути використані в багатьох аспектах сфери моди та розпізнавання одягу, що підкреслює актуальність цієї проблеми в сучасному світі.

Отже, розпізнавання образів є актуальною та важливою проблемою, яка має широке застосування в різних галузях, включаючи медицину, транспорт, сферу безпеки та моду. Завдяки розпізнаванню образів, комп'ютерні системи можуть розуміти та аналізувати зображення, що дає можливість зробити точніші та швидші висновки. У зв’язку із зростанням обсягів даних та розвитком новітніх технологій, методи розпізнавання образів стають все більш точними та ефективними. Класифікація методів включає у себе статистичні методи, методи машинного навчання та нейронні мережі, які забезпечують високу точність та швидкість обробки даних. Кожен з цих методів має свої переваги та недоліки та може бути використаний в залежності від потреб застосування.

РОЗДІЛ 2.
Організація розробки системи розпізнавання образів

2.1 Обґрунтування вибору програмних засобів для розробки системи розпізнавання образів

Вибір програмних засобів для розробки системи розпізнавання образів є ключовим етапом у розробці такої системи. Для досягнення максимальної точності та швидкості роботи системи необхідно вибрати найефективніші інструменти. Для виконання даної роботи було обрано мову програмування Python, програмну бібліотеку TensorFlow та нейромережну бібліотеку Keras.

Python є однією з найбільш популярних високорівневих мов програмування для розробки систем штучного інтелекту. Python має велику кількість бібліотек, фреймворків та інструментів, що дозволяють ефективно розробляти системи розпізнавання образів на високому рівні. [9] Ця мова є однією з найпоширеніших в галузі машинного навчання та обробки даних, і її застосування в цих областях надалі зростає. Одним з основних факторів, що роблять Python таким популярним для машинного навчання, є наявність великої кількості бібліотек з відкритим кодом, таких як NumPy, Pandas, Matplotlib, Scikit-learn, TensorFlow і Keras.

Застосування Python в розробці систем розпізнавання образів має також свої переваги. Наприклад, Python є мовою програмування з високим рівнем абстракції, що дозволяє зосередитися на вирішенні завдань вищого рівня та не витрачати час на деталі реалізації. Python також має простий та зрозумілий синтаксис, що полегшує розробку та зрозуміння коду для розробників з різним рівнем досвіду.

Рис. 2.1. Python

Працюючи з Python, програмісту не потрібно приділяти великої уваги безпосередньому написанню коду: він може зосередити всю свою увагу на вирішенні більш складних проблем, пов'язаних з машинним навчанням. Крім того, мова підтримує багатопоточність, що дозволяє ефективно використовувати ресурси системи та прискорювати обчислення. Бібліотеки NumPy та Pandas надають зручний інтерфейс для роботи з багатовимірними масивами даних та роботи з даними у вигляді таблиць. Бібліотеки Matplotlib та Seaborn дозволяють візуалізувати дані у вигляді графіків та діаграм.

TensorFlow є однією з найпопулярніших бібліотек машинного навчання, розроблених компанією Google. Вона має широкі можливості у побудові нейронних мереж та оптимізації їхньої роботи. Бібліотека дозволяє працювати з різними типами даних, включаючи зображення та відео.

Одним з ключових переваг Tensorflow є його здатність до розпаралелювання обчислень за допомогою GPU та TPU, що робить цю бібліотеку особливо корисною для розробки складних та великих моделей нейронних мереж.

Крім того, вона забезпечує широкий спектр інструментів для візуалізації та налагодження моделей, що робить його досить зручним для розробки та дослідження нейронних мереж. Також, Tensorflow підтримує різні рівні абстракції, що дозволяє розробникам працювати з бібліотекою на різних рівнях складності. [10] Наприклад, можна використовувати високорівневі інтерфейси,

такі як Keras, для швидкої та простої розробки моделей, або працювати з низькорівневими операціями для більш детальної настройки та контролю.

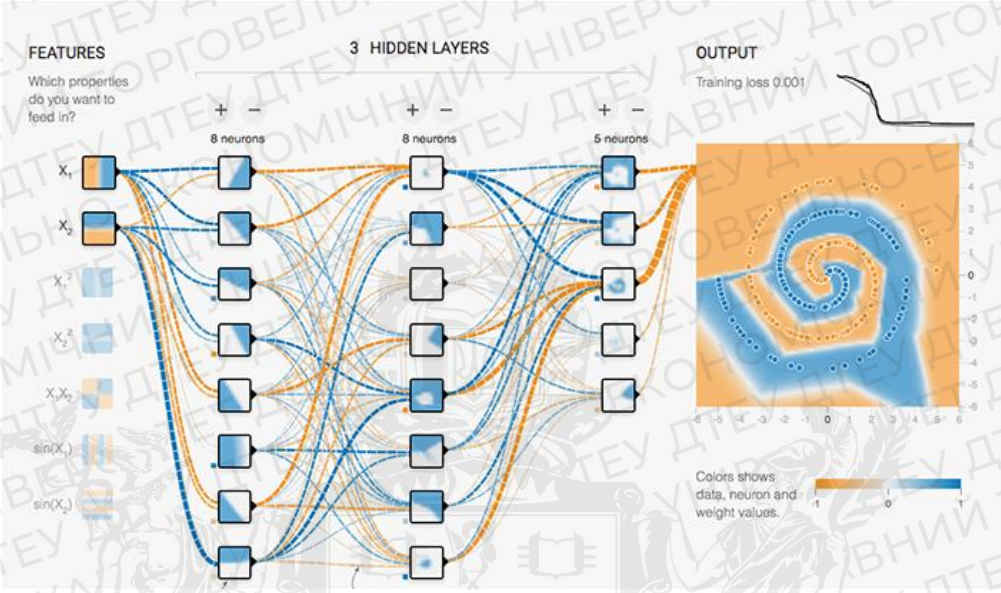

Рис. 2.2. Принцип роботи шарів

Keras є високорівневим фреймворком для розробки нейронних мереж. Його основним перевагою є простота використання та швидкість розробки. Keras забезпечує швидку розробку нейронних мереж без необхідності вивчення складних математичних формул та алгоритмів. Він був розроблений з метою забезпечити простоту використання, швидкість та гнучкість роботи.

Рис. 2.3. Keras

Однією з головних переваг Keras є його можливість працювати з TensorFlow, що дозволяє легко створювати складні нейронні мережі, такі як згорткові нейронні мережі, рекурентні нейронні мережі та комбінації різних

типів мереж. Також, слід відмітити простоту використання та інтуїтивний інтерфейс. Він надає широкий набір готових модулів та функцій для розробки нейронних мереж, що дозволяє зосередитись на проектуванні та експериментуванні з моделями. Крім того, Keras підтримує автоматичне підбирання параметрів мережі, що зменшує необхідність вручну налаштовувати параметри. Також, надає можливість використовувати готові моделі, що були натреновані на великих наборах даних, такі як VGG16, VGG19, ResNet та інші. [11] Це значно зменшує час розробки та тренування власної моделі.

Отже, основні переваги Keras, та чому був обраний саме він:

1. Простота використання: Надає зрозумілий та простий для використання інтерфейс, який дозволяє швидко створювати та настроювати нейронні мережі з мінімальними знаннями технічних деталей.
2. Дозволяє швидко та ефективно розробляти різні типи нейронних мереж, такі як згорткові, рекурентні та комбіновані мережі.
3. Keras має велику кількість вбудованих функцій та модулів для розв'язання різних проблем, пов'язаних з машинним навчанням, таких як функції активації, функції витрат, оптимізатори, архітектури мереж, тощо.
4. Підтримує використання графічних процесорів (GPU) для прискорення обчислень, що дозволяє розробляти більш складні та потужні моделі нейронних мереж.
5. Інтеграція з Tensorflow: Keras є частиною бібліотеки Tensorflow, що дозволяє використовувати всі функції Tensorflow у своїх проектах та забезпечує простоту міграції між цими двома бібліотеками.

Вибір Python, TensorFlow та Keras для розробки системи розпізнавання образів є виправданим, оскільки ці інструменти забезпечують широкі можливості у побудові нейронних мереж та оптимізації їхньої роботи. Крім того, вони є популярними у середовищі машинного навчання та мають велику кількість

документації та прикладів використання, що допоможе у виконанні роботи. Таким чином, використання цих програмних засобів дозволить швидко та ефективно розробити систему розпізнавання образів.

2.2 Розробка моделі системи розпізнавання образів

У цій дипломній роботі буде детально описано розробку моделі системи розпізнавання образів з використанням популярних інструментів: Python, Tensor Flow та Keras. Для коректного виконання поставленої задачі необхідно зрозуміти принцип роботи нейронної мережі, технічні можливості описаних вище програмних засобів.

Нейронна мережа, створена на основі Tensor Flow та Keras, працює за принципом зворотного поширення помилок, який є ключовим для навчання моделі. Навчання моделі полягає у зміні ваг (відносних значень вхідних сигналів) з кожним етапом навчання з метою зниження помилки на виході моделі. У нейронній мережі використовуються шари нейронів, які збирають та обробляють вхідні сигнали, а потім передають їх до наступного шару, також, обирається кількість епох. [12] Термін "епоха" використовується для позначення одного проходу навчання нейронної мережі через весь навчальний набір даних. Під час кожної епохи навчання, навчальні дані проходять через нейронну мережу, змінюють ваги нейронів та оновлюють параметри моделі. Кожна епоха завершується тестуванням моделі на окремому наборі даних для валідації.
Кількість епох визначається під час навчання і може бути змінена в процесі, в залежності від того, наскільки швидко модель навчається та наскільки точно вона передбачає результати. Зазвичай, збільшення кількості епох допомагає покращити точність моделі, проте може призвести до перенавчання (overfitting) - коли модель добре підлаштовується під навчальні дані, але погано працює на нових даних.

На виході останнього шару модель повертає результат розпізнавання або передбачення. Нейронні мережі використовуються для вирішення проблем з обробкою даних, таких як класифікація, передбачення, сегментація та інші.

Класифікація - це завдання, в якому модель машинного навчання намагається призначити кожному елементу вхідних даних один з певного набору попередньо визначених класів. Це може бути розпізнавання образів, класифікація тексту, розпізнавання мови та інші задачі, де потрібно визначити, до якого класу належить даний елемент.

Сегментація - це завдання, в якому модель машинного навчання намагається розділити зображення на окремі сегменти, кожен з яких відповідає певному об'єкту або регіону. Наприклад, сегментація зображень медичних знімків для визначення пухлини або сегментація зображень з вуличних камер для визначення машин.

Передбачення - це завдання, в якому модель машинного навчання намагається передбачити певний результат або значення на основі вхідних даних. Це може бути прогнозування цін на нерухомість, передбачення часового ряду фінансових даних або передбачення діагнозу на основі медичних даних.

Для того, щоб модель змогла вирішити завдання, її потрібно спочатку навчити за допомогою навчальних даних. Перш за все, необхідно визначити структуру мережі, тобто кількість шарів та кількість нейронів у кожному шарі. Обирання кількості шарів у нейронній мережі є однією з ключових задач у процесі проектування мережі. Однак, не існує єдиного "правильного" способу визначення оптимальної кількості шарів, оскільки це залежить від конкретного завдання, його складності та архітектури мережі. В загальному прийнято, що більш складні задачі вимагати більшої кількості шарів. Наприклад, задача визначення обличчя на фотографії вимагає більшої кількості шарів, ніж задача класифікації одягу. Одним з підходів є емпіричне визначення кількості шарів. Для цього використовується здоровий глузд та експертні знання про проблему,

або можна спробувати кілька різних конфігурацій мережі та порівняти їх ефективність.

Далі, навчальні дані подаються на вхід моделі і відбувається передача сигналів через шари нейронів.

Кожен нейрон у мережі містить ваги, які використовуються для зважування вхідних сигналів. На основі зважених сигналів, нейрон робить обчислення та передає свій вихід на наступний шар. Потім, на виході останнього шару, модель повертає результат.

Після отримання результату, визначається рівень помилки, точність та втрати.

Точність (accuracy) та рівень втрат (loss) є метриками для оцінки ефективності моделі під час навчання та тестування. Точність моделі визначається як відношення кількості правильних передбачень моделі до загальної кількості передбачень. Наприклад, якщо модель класифікує 89 зі 100 зображень правильно, її точність становитиме 89%.

Рівень втрат визначається як середня помилка моделі під час навчання. Це означає, що рівень втрат відображає, наскільки точно модель передбачає вихідні дані порівняно з очікуваними результатами. Рівень втрат може бути обчислений для кожного етапу навчання та допомагає визначити, наскільки добре модель навчається під час кожної ітерації.

Обидві метрики є важливими при визначенні ефективності моделі. Точність є основною метрикою для оцінки моделі в цілому, тоді як рівень втрат може допомогти виявити проблемні аспекти моделі та підказати, які аспекти потребують покращення. Наприклад, висока точність може бути досягнута шляхом простого копіювання даних, тоді як низький рівень втрат може вказувати на те, що модель погано розуміє залежності між вхідними та вихідними даними. Тому обидві метрики повинні бути враховані при оцінці ефективності моделі.

Наступним кроком є використання алгоритму зворотного поширення помилок, який оновлює ваги кожного нейрона з метою зниження помилки на

наступному проході. [13] Його основна ідея полягає у використанні ланцюгового правила диференціювання для обчислення похідних функції втрат (loss function) за вагами мережі.

Під час навчання мережі, спочатку випадковим чином ініціалізуються ваги мережі. Навчальні приклади подаються на вхід мережі, і робиться передбачення з використанням поточних ваг. Далі, рівень втрат обчислюється з використанням функції втрат, яка зазвичай є квадратичною функцією відхилення. Після обчислення втрат, похідні функції втрат за кожним ваговим коефіцієнтом обчислюються з використанням ланцюгового правила диференціювання. Далі, вагові коефіцієнти оновлюються в напрямку, протилежному до градієнту втрат. Градієнт визначає напрямок, в якому рівень втрат швидко зростає, тому оновлення ваг в напрямку, протилежному до градієнту, допомагає зменшити рівень втрат.

Алгоритм зворотного поширення помилок дозволяє оновлювати вагові коефіцієнти відповідно до кожного навчального прикладу, тому що градієнти обчислюються за кожним навчальним прикладом окремо. [14] Цей процес повторюється багато разів, поки рівень втрат не досягне мінімальної можливої величини, або поки навчання не буде зупинено з інших причин (наприклад, через перевищення кількості епох навчання).

Процес навчання відбувається за допомогою ітерацій, де ваги оновлюються на кожному кроці, з метою зниження помилки на виході моделі. Під час навчання модель обробляє навчальні дані та робить передбачення на їх основі. Потім, порівнюючи передбачені виходи з очікуваними, розраховується значення помилки. Далі, за допомогою алгоритму зворотного поширення помилок, ваги моделі змінюються з метою зниження помилки. Процес зміни ваг продовжується до тих пір, поки значення помилки не буде достатньо низьким.

Після навчання моделі, проводиться процес перевірки, в якому використовуються відкладені дані, тобто дані, які не використовувалися під час навчання. Модель оброблює відкладені дані та робить передбачення на їх основі.

Знову порівнюючи передбачені виходи з очікуваними, розраховується значення помилки. Якщо значення помилки на відкладених даних недостатньо низьке, необхідно змінити структуру моделі або збільшити кількість навчальних епох.

Для того, щоб уникнути перенавчання моделі, використовуються методи регуляризації, такі як Dropout та L1/L2 регуляризація. Dropout регуляризація - цей метод полягає у випадковому вимкненні деяких нейронів у мережі під час навчання. Він дозволяє зменшити залежність між нейронами та знизити ризик перенавчання. L1 регуляризація - додавання суми модулів ваг до функції втрат. Це приводить до зміни ваг в сторону нуля, що дозволяє зменшити розмірність моделі та підвищити її універсальність. L2 регуляризація - додавання суми квадратів ваг до функції втрат. Призводить до зменшення великих ваг, зберігаючи значимі ваги. Внаслідок цього, можна досягнути кращої здатності до узагальнення та зменшити ризик перенавчання.

Після завершення процесу навчання, модель може бути використана для передбачення або класифікації нових даних. Для цього, необхідно подати дані на вхід моделі, та отримати результат на виході.

Нейронна мережа, створена на основі Tensor Flow та Keras, працює за принципом зворотного поширення помилок, де ваги нейронів оновлюються з кожним етапом навчання з метою зниження помилки на виході моделі. Модель може бути використана для рішення завдань класифікації, передбачення та інших завдань обробки даних.

2.3 Алгоритм створення системи розпізнавання образів

Першим кроком у розробці моделі є підготовка даних. Для цього потрібно створити або обрати вже існуючий датасет, який буде містити зображення, які необхідно розпізнати, та відповідні мітки. В роботі буде використовуватися вже існуючий датасет Fashion MNIST, який широко використовується у галузі машинного навчання, а саме комп'ютерного зору й нейронних мереж. Він був розроблений з метою заміни класичного датасету MNIST (датасет рукописних цифр) та надати виклик для нових методів класифікації зображень і розвитку цієї галузі. FashionMNIST містить зображення одягу, що представляють різні категорії, і з часом став стандартним датасетом для перевірки алгоритмів машинного навчання та комп'ютерного зору, завдяки своїй зручності та великому обсязі даних. Складається датасет з 70 000 зображень розміром 28x28 пікселів. Ці зображення поділені на 60 000 зображень для тренування та 10 000 зображень для тестування.

Рис. 2.4. Датасет FashionMNIST

Датасет містить 10 класів та їх числові мітки, що представляють різні категорії одягу, а саме:

0 - Футболка
1 - Штани
2 - Світшот
3 - Сукня
4 - Пальто
5 - Сандалі
6 - Сорочка
7 - Кросівки
8 - Сумка
9 - Чоботи

Цей датасет став популярним у наукових дослідженнях, оскільки він є викликом для розробників алгоритмів машинного навчання, які використовують методи комп’ютерного зору. Він використовується для порівняння різних методів навчання, архітектур нейронних мереж та алгоритмів класифікації.

Може використовуватися з різними мовами програмування, методами та фреймворками, залежно від вимог та уподобань дослідників або розробників. Однак, як було зазначено вище, все буде працювати на основі Python, Tensor Flow та Keras, без урахування додаткових супутніх бібліотек, як, наприклад Numpy, що допоможе з обчисленнями в проекті.

Також, слід відмітити, що з метою заповнення прогалини в Keras, коли не було доступного сету зображень та полегшення роботи розробників, FashionMNIST був вбудований в Keras. Це дозволило розробникам одним рядком коду завантажувати, обробляти та навчати моделі на цьому датасеті без необхідності окремого завантаження та підготовки даних.

Загалом, FashionMNIST - це важливий датасет у галузі комп'ютерного зору та нейронних мереж. Він надає збалансований набір даних з 10 класів, що

представляють різні категорії одягу. Дозволяє працювати з різноманітними мовами програмування та фреймворками. Також, завдяки саме цьому датасету багато розробників та студентів розпочали свій шлях у сфері машинного навчання, що спонукало їх вивчати, створювати та використовувати нові методи, які дали вирішення багатьох задач.

Наступним кроком необхідно обрати оптимізатор, який буде використовуватися у проекті. Це важливий аспект, що застосовується для покращення процесу навчання моделей. Оптимізатори впливають на швидкість, ефективність та якість навчання моделей, допомагають знайти оптимальні значення параметрів.

Поняття оптимізація в машинному навчанні є процесом знаходження найкращих значень параметрів моделі, що відповідають мінімуму або максимуму певної функції втрат або цільової функції. [15] Ця функція відображає якість моделі та може бути визначена метриками, такими як точність класифікації чи середня квадратична помилка. А самі оптимізатори є алгоритмами, які використовуються для оновлення параметрів моделі з метою мінімізації функції втрат. Ці алгоритми виконують ітерації, де на кожному кроці вони аналізують градієнти функції втрат відносно параметрів і оновлюють їх значення, здійснюючи кроки в напрямку, що максимально зменшує значення функції втрат.

Слід зазначити, що таке градієнти - це вектор, який вказує напрямок та величину найшвидшого зростання функції втрат. Тобто, градієнт вимірює, наскільки функція втрат змінюється для кожного параметра моделі. Це розраховується шляхом обчислення похідної функції втрат відносно кожного параметра.

Використовуються різні типи оптимізаторів залежно від призначення та характеристик задачі, деякі оптимізатори можуть показувати кращі результати для певних типів завдань, тоді як інші можуть працювати краще з іншими задачами, наприклад:

Stochastic Gradient Descent (SGD)

Оптимізатор SGD ґрунтується на методі градієнтного спуску, де використовується стохастичне оновлення параметрів моделі на кожному кроці. У порівнянні з традиційним градієнтним спуском, де обчислюється градієнт на всьому наборі даних, SGD обчислює градієнт на випадково обраних підмножинах даних, які називаються пакети або міні-пакети. Це дозволяє працювати швидше, особливо на великих наборах даних, оскільки градієнт обчислюється на меншій підмножині замість усіх даних.

Adaptive Moment Estimation (Adam)

Adam є оптимізатором, який об'єднує ідеї з інших методів оптимізації, таких як RMSprop та Momentum. Він адаптивно налаштовує швидкість навчання для кожного параметра, використовуючи оцінки першого та другого моменту градієнта. Це дозволяє швидше збіжнути та забезпечує стабільніше навчання моделі.

Adaptive Gradient Algorithm (AdaGrad)

Оптимізатор AdaGrad змінює швидкість навчання для кожного параметра на основі історії градієнтів. Він враховує накопичені квадрати градієнтів для кожного параметра, що дозволяє більш агресивно оновлювати параметри з меншою швидкістю навчання для рідкісних параметрів.

На підставі отриманих результатів та дослідів інших розробників, які працюють у сфері комп’ютерного зору можна зробити висновок, що оптимізатор SGD є кращим вибором для навчання моделей. Він дозволяє ефективно оптимізувати параметри моделі, швидко збігається до оптимальних значень та знижує функцію втрат. Це популярний інструмент, який з успіхом використовується у різних завданнях, включно створення нейронної мережі для розпізнавання одягу.

Для коректної оцінки продуктивності моделі необхідно обрати потужний інструмент, наприклад матрицю помилок (Confusion Matrix) - це таблиця, яка

відображає результати класифікації моделі на основі відомих істинних міток і передбачуваних міток. [17]

Рис. 2.5. Принцип роботи матриці помилок

Таблиця складається з чотирьох елементів:

- True Positive (TP): Кількість прикладів, які правильно були визнані моделлю як позитивні.
- True Negative (TN): Кількість прикладів, які правильно були визнані моделлю як негативні.
- False Positive (FP): Кількість прикладів, які помилково були визнані моделлю як позитивні (помилка першого роду).
- False Negative (FN): Кількість прикладів, які помилково були визнані моделлю як негативні (помилка другого роду).

Основна перевага матриці помилок полягає в тому, що вона надає чітку та структуровану інформацію про продуктивність моделі для кожного класу, що дозволяє легко зрозуміти, як добре модель працює відповідно класу. Завдяки матриці можна легко виявити класи, до яких модель має найбільші труднощі у класифікації. Це допомагає виявити слабкі сторони моделі та покращити її роботу.

Загалом, матриця помилок є важливим інструментом для аналізу результатів нейронної мережі і добре підходить для роботи з Keras та FashionMNIST. Вона допомагає провести аналіз, наскільки добре модель класифікує дані та виявити проблемні класи, оцінити різні метрики продуктивності. Зважаючи на ці переваги, використання матриці помилок є одним з найкращих інструментальних рішень для аналізу результатів навчання нейронної мережі та її вдосконалення.

Отже, після вибору зручного датасету, аналізу можливих оптимізаторів та підбору найкращого для поточного проекту, обирається потужний інструмент для аналізу метрик моделі та точності її роботи. Завершення етапу підбору програмних засобів, розробки моделі проекту та алгоритму реалізації є вирішальним перед переходом до останнього етапу - розробки моделі.

РОЗДІЛ 3.
Реалізація розробки системи розпізнавання кольорових зображень

3.1 Програмна реалізація системи розпізнавання образів

Орієнтуючись на поставлену проблематику дипломного проекту, створюється нова модель на онлайн-платформі Google Colab, яка надає зручне хмарне середовище для аналізу даних та машинного навчання. Ця платформа дозволяє користувачам безкоштовно використовувати обчислювальні ресурси Google для створення та виконання коду на мовах програмування Python та R без необхідності встановлювати та налаштовувати середовища розробки на своєму комп'ютері.

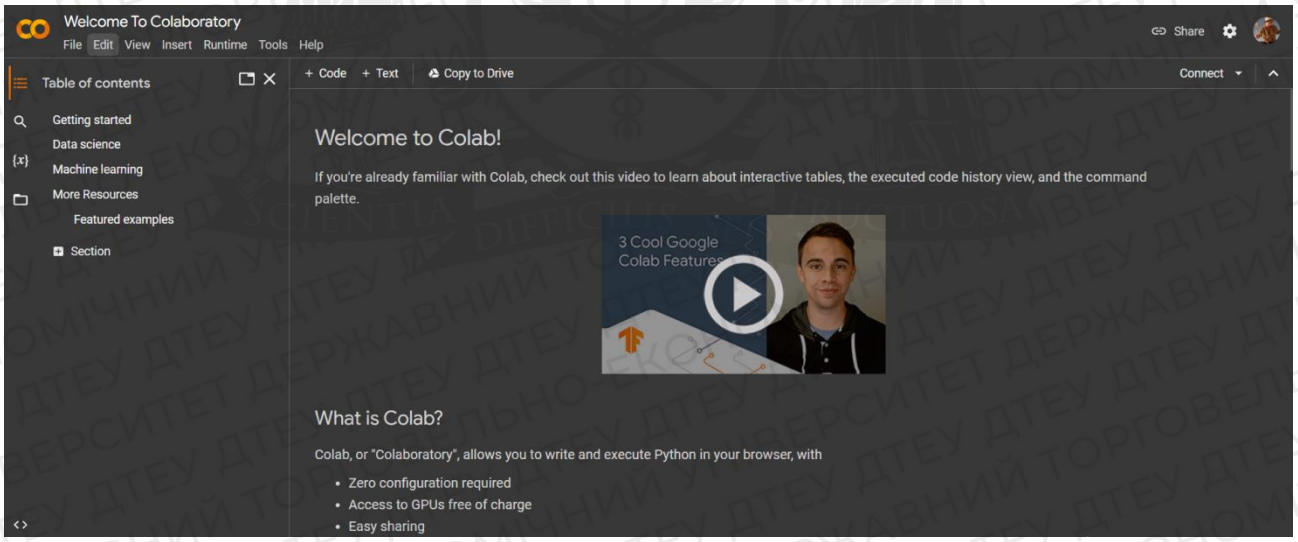

Рис. 3.1. Створення нового проекту

Для створення та коректної роботи нейронної мережі, перш за все потрібно підключити всі необхідні бібліотеки, модулі та фреймворки.

```
[ ]  # Імпортуємо модулі та бібліотеки

     import numpy as np
     import tensorflow as tf
     import matplotlib.pyplot as plt
     from tensorflow import keras
     %matplotlib inline
     from tensorflow.keras.datasets import fashion_mnist
     from tensorflow.keras.models import Sequential
     from tensorflow.keras.layers import Dense, Dropout
     from tensorflow.keras import utils
```

Рис. 3.2. Підключення необхідних бібліотек

Для роботи з масивами та візуалізації рисунків підключається бібліотека Nympy і Plotlib. Модуль FashionMNIST вже вбудований в бібліотеку Keras, тому що є одним із основних, на якому навчається багато програмістів будувати нейронні мережі. Також, потрібно імпортувати модуль Sequential, завдяки якому шари нейронної мережі йдуть один за одним. Далі підключається тип шарів Dense, який означає, що шари будуть повнозв'язковими.

Підключаються необхідні утиліти, які допоможуть перевести дані в зрозумілий для Keras формат.

Так як набір даних FashionMNIST вже імпортований до проекту, є можливість швидко загрузити їх до проекту.

```
[ ]  # Ділимо датасет на навчальну та тестову вибірку

     (x_train, y_train), (x_test, y_test) = fashion_mnist.load_data()

     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-labels-idx1-ubyte.gz
     29515/29515 [==============================] - 0s 0us/step
     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/train-images-idx3-ubyte.gz
     26421880/26421880 [==============================] - 0s 0us/step
     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-labels-idx1-ubyte.gz
     5148/5148 [==============================] - 0s 0us/step
     Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/t10k-images-idx3-ubyte.gz
     4422102/4422102 [==============================] - 0s 0us/step

[ ]  class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']
```

Рис. 3.3. Завантаження даних

Набір даних буде розділено на тестову та навчальну вибірку, відповідно test і train. Кожна з частин буде містити частину X - це зображення, і частину У - це відповідь, якому класу належить те, чи інше зображення, або, як їх часто називають - мітки.

Так як імена класів не включені в набір даних, необхідно прописати їх самостійно через class_names.

Перед створенням нейронної мережі необхідно провести попередню обробку даних. На даний момент зображення виглядають так: (Рис. 3.4)

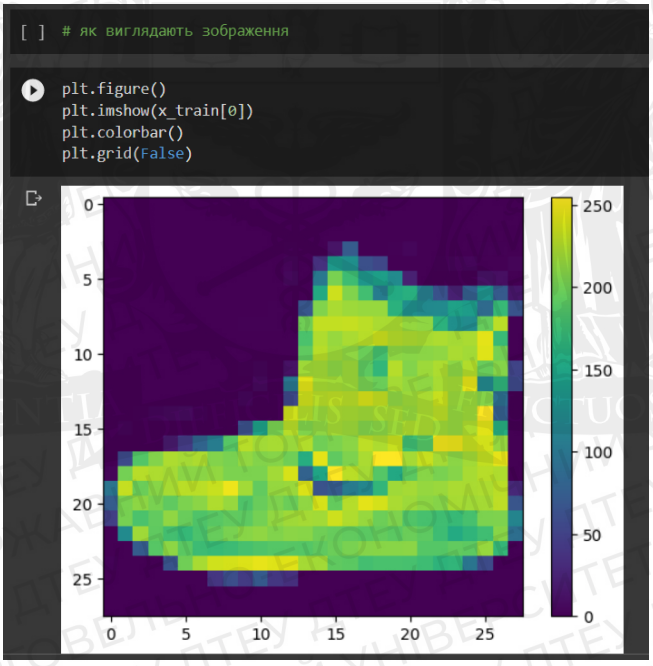

Рис. 3.4. Приклад зображення з індексом 0

За допомогою plt.imshow та задавання індексу від 0 до 59999 можна подивитися на всі зображення, які містить бібліотека. З правої частини можна бачити шкалу інтенсивності пікселів від 0 до 255, тобто самий темний піксель має значення рівне = 0, а самий світлий рівний = 255.

```
[ ]  # нормалізація даних

[ ]  x_train = x_train / 255
     x_test = x_test / 255
```

Рис. 3.5. Нормалізація даних

Для покращення алгоритмів оптимізації потрібно провести нормалізацію даних. А саме поділити інтенсивність кожного пікселя на 255, щоб всі дані на вході до нейронної мережі знаходилися в діапазоні від 0 до 1. Важливо, щоб тренувальний та тестовий датасети були оброблені однаково.

```
[ ]  # перевірка нових даних

     plt.figure()
     plt.imshow(x_train[0])
     plt.colorbar()
     plt.grid(False)
```

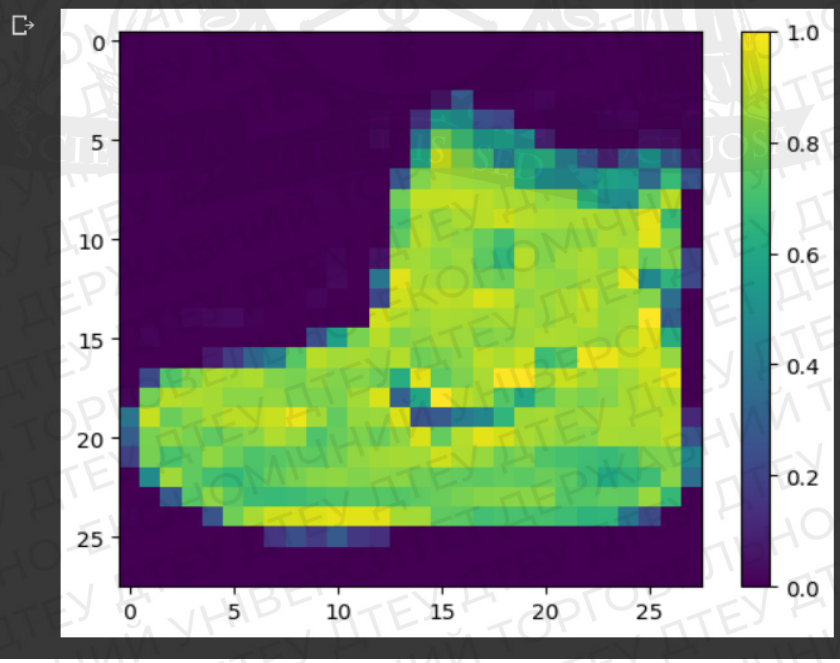

Рис. 3.6. Перевірка нових даних

Після оптимізації та повторного виведення зображення можна зробити висновок, що код спрацював коректно, тобто інтенсивність пікселів знаходиться

в діапазоні від 0 до 1, що дозволить нейронній мережі легше працювати з такими значеннями.

Для більш детального вивчення даних, з якими проводиться робота, можна вивести перші 15 зображень та відобразити під ними найменування їхніх класів.

```
plt.figure(figsize=(10,10))
for i in range (25):
  plt.subplot(5,5,i+1)
  plt.xticks([])
  plt.yticks([])
  plt.imshow(x_train[i])
  plt.xlabel(class_names[y_train[i]])
```

Рис. 3.7. Виклик зображень у новому форматі

Рис. 3.8. Ознайомлення з результатами

Зображення вивелися як потрібно, у кількості 15 одиниць та з підписами класів. Можна детальніше оцінити датасет, з яким проводиться робота.

Отже, дані підготовлені до роботи, можна переходити до створення нейронної мережі. Основним будівельним блоком є шар, і основна частина нейронного навчання складається з об'єднання простих шарів. Як зазначалося раніше, буде використовуватися модель Sequential, що дозволить послідовно пропускати шари.

```
[ ]  # створення моделі нейронної мережі

     model = keras.Sequential([
                               keras.layers.Flatten(input_shape=(28,28)),
                               keras.layers.Dense(128, activation="relu"),
                               keras.layers.Dense(10, activation="softmax")
     ])
```

Рис. 3.9. Створення нейронної мережі

Створюється послідовна модель, перший шар вказаний як Flatten перетворює формат з двомірного масиву, де кожне зображення було масивом розміром 28x28 пікселів в одномірний масив 28x28, тобто тепер зображення поступатиме в нейрон як рядок розміром 784 пікселя.

Далі йдуть два повноцінних шари, перший - вхідний повнозв'язковий, тут необхідно вирішити скільки нейронів буде в одному шарі. За даним датасетом проводилися різні дослідження, одним з найкращих за передбаченням виявився шар на 128 нейронів, тобто в кожен нейрон поступатимуть всі 784 пікселя від зображення. Наступним пишеться функція активації, була обрана функція Relu, яка довела свою ефективність у попередніх випробуваннях.

Другий шар - вихідний повнозв'язковий, в якому буде 10 нейронів за кількістю класів, в якості функції активації використовується функція Softmax, завдяки якій другий шар повертатиме масив з 10 можливих оцінок, сума яких рівна = 1, тобто кожен нейрон міститиме оцінку вірогідності, що поточне зображення належить одному з десяти класів.

```
[ ] # компіляція моделі

model.compile(optimizer=tf.keras.optimizers.SGD(), loss="sparse_categorical_crossentropy", metrics=['accuracy'])

model.summary()
```

Рис. 3.10. Налаштування, або компіляція моделі

При компіляції моделі вказуються параметри навчання. В даному випадку у якості оптимізатора використовується SGD, що розшифровується як стахастичний градієнтний спуск, для вирішення подібних задач його застосовують найчастіше.

Функція помилки вказується параметром Loss, яким використовується категорична перехрестна ентропія, така функція помилки добре праює в задачах класифікації, коли класів більше двох.

Останній параметр - точність. Вказується Accuracy, тобто доля правильних відповідей.

Після того, як модель скомпільовано, можна вивести параметри моделі командою model.sumarry().

```
model.summary()

Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 flatten (Flatten)           (None, 784)               0

 dense (Dense)               (None, 128)               100480

 dense_1 (Dense)             (None, 10)                1290

=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
_________________________________________________________________
```

Рис. 3.11. Параметри моделі

Можна зробити висновок, що готово 101,770 параметрів для тренування моделі, неактивних немає.

Отже, дані підготовлені, створено та скомпільовано модель нейронної мережі, тепер можна переходити до частини навчання. Навчання моделі відбувається за допомогою функції fit(), як і в інших задачах машинного навчання.

```
[ ] # навчання моделі

    model.fit(x_train, y_train, epochs=10)

    Epoch 1/10
    1875/1875 [==============================] - 5s 2ms/step - loss: 0.7366 - accuracy: 0.7634
    Epoch 2/10
    1875/1875 [==============================] - 4s 2ms/step - loss: 0.5134 - accuracy: 0.8250
    Epoch 3/10
    1875/1875 [==============================] - 4s 2ms/step - loss: 0.4665 - accuracy: 0.8378
    Epoch 4/10
    1875/1875 [==============================] - 4s 2ms/step - loss: 0.4410 - accuracy: 0.8467
    Epoch 5/10
    1875/1875 [==============================] - 4s 2ms/step - loss: 0.4230 - accuracy: 0.8532
    Epoch 6/10
    1875/1875 [==============================] - 4s 2ms/step - loss: 0.4087 - accuracy: 0.8585
    Epoch 7/10
    1875/1875 [==============================] - 4s 2ms/step - loss: 0.3973 - accuracy: 0.8620
    Epoch 8/10
    1875/1875 [==============================] - 4s 2ms/step - loss: 0.3878 - accuracy: 0.8659
    Epoch 9/10
    1875/1875 [==============================] - 4s 2ms/step - loss: 0.3789 - accuracy: 0.8689
    Epoch 10/10
    1875/1875 [==============================] - 4s 2ms/step - loss: 0.3725 - accuracy: 0.8701
    <keras.callbacks.History at 0x7f39b0207790>
```

Рис. 3.12. Процес навчання моделі

За допомогою параметрів x_train та y_train функції fit() передається як навчальна вибірка, так і вибірка відповідей. Також, необхідно вказати параметр epochs - кількість епох, тобто весь набір даних пройде через нейронну мережу 10 разів. Кількість епох підбирається в залежності від набору даних, але головний принцип - чим більш датасет різношерстий, тим більшу потрібно задавати кількість епох. Але необхідно пам’ятати про потужність комп’ютера або сервера,

на якому проводиться вичислення, тому що чим більший датасет, тим більше часу буде потрібно на проходження одної епохи.

Наступним кроком створюється механізм виводу самих передбачень на екран та підсвічування правильності відповіді зеленим або червоним.

Процес навчання запущено, в кінці кожного рядка результатів проходження епохи вказується функція помилки, а саме параметр loss і точність передбачення - accuracy.

Як можна помітити, у міру навчання нейронної мережі, тобто з кожною наступною епохою значення помилки знижується, а точність навпаки збільшується. Закінчення останньої, десятою епохи означає, що процес навчання нейронної мережі завершено. Можна побачити, що точність складає трохи менше 90%.

```
Epoch 9/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3789 - accuracy: 0.8689
Epoch 10/10
1875/1875 [==============================] - 4s 2ms/step - loss: 0.3725 - accuracy: 0.8701
<keras.callbacks.History at 0x7f39b0207790>
```

Рис. 3.13. Результати втрати та точності

Задля кращої візуалізації результатів необхідно створити функцію, яка буде виводити випадкові зображення із датасету і аналізувати, чи правильну відповідь дала нейронна мережа. Також, під зображенням необхідно вивести відсоток, з яким нейронна мережа зробила передбачення і правильну відповідь. В залежності від результату, текст буде підсвічувати зеленим кольором у випадку вірної відповіді, та червоним, якщо відповідь неправильна. Це допоможе швидко оцінити результати без необхідності уваги до кожного відсотка окремо, полегшить процес аналізу та сприятиме зручності.

```
[ ]  def plot_random_image(model,images, true_labels, classes):

        plt.figure(figsize=(10,15))

        for i in range(12):
            ax=plt.subplot(4,3,i+1)
            rand_index=random.choice(range(len(images)))
```

Рис. 3.14. Функція рандомізації

Створюється функція, яка буде виводити 12 випадкових зображень із вибірки. Також, необхідно імпортувати бібліотеку random для застосування усіх переваг мови python та коректної роботи коду.

```
        # Передбачування
        target_image=images[rand_index]
        pred_probs=model.predict(tf.expand_dims(target_image,axis=0))
        pred_label=classes[pred_probs.argmax()]
        true_label= classes[true_labels[rand_index]]
        plt.imshow(target_image,cmap=plt.cm.binary)

        # Зміна кольору тексту
        if pred_label==true_label:
            color="green"
        else:
            color="red"

        plt.xlabel("Pred:{} {:2.0f}% \n(True:{})".format(pred_label,
                                                100*tf.reduce_max(pred_probs),
                                                true_label),
                                                color=color, fontsize=13
                                                )
```

Рис. 3.15. Продовження коду

Також, важливим інструментом у машинному навчанні є матриця помилок (Confusion matrix), яка дозволить оцінити загальну точність передбачення цілого датасету. Матриця помилок представляється у вигляді таблиці, де кожен ряд відповідає дійсному класу об'єктів, а кожен стовпець - прогнозованому класу. Кожна комірка матриці вказує кількість об'єктів, які належать до певного дійсного класу та були класифіковані як певний прогнозований клас.

```
[25] y_probs = model.predict(x_train)

     # Перетворення ймовірності пердбачення в цілі числа
     y_preds = y_probs.argmax(axis=1)

     1875/1875 [==============================] - 4s 2ms/step

     # Матриця помилок
     cm = confusion_matrix(y_preds, y_train)
     # Плот
     disp = ConfusionMatrixDisplay(confusion_matrix=cm,display_labels=class_names)
     fig, ax = plt.subplots(figsize=(10,10))
     disp.plot(ax=ax);
```

Рис. 3.16. Реалізація матриці помилок

Загалом, для нейронної мережі, яка складається з двох основних шарів, такий результат є цілком прийнятним. Це досить високий показник, який свідчить про добре навчання моделі та її здатність правильно класифікувати вхідні дані. Точність близько 90% говорить про те, що модель правильно класифікує приблизно 9 з 10 вхідних прикладів. Однак, для деяких задач цей рівень точності може бути недостатнім, і в таких випадках можна розглядати більш складні архітектури мережі або застосовувати додаткові методи оптимізації, такі як регуляризація або збільшення кількості навчальних епох.

Двухшарова нейронна мережа з точністю близько 90% є досить ефективним рішенням для багатьох завдань та забезпечує задовільні результати в проблематиці поточної дипломної роботи.

3.2 Апробація результатів дослідження

Оскільки для навчання було використано 60000 зображень, а 10000 залишилися в тестовій вибірці, тобто нейронна мережа їх не бачила, можна переходити до тестування мережі, перевірити точність передбачення на нових

для нейронної мережі зображеннях. За допомогою функції model.evaluate() можна подивитися загальну точність передбачення.

```
[ ]  # перевірка точності моделі

[ ]  test_loss, test_acc = model.evaluate(x_test, y_test)
     print('Test accuracy:', test_acc)

     313/313 [==============================] - 1s 1ms/step - loss: 0.4043 - accuracy: 0.8570
     Test accuracy: 0.8569999933242798
```

Рис. 3.17. Загальні результати точності

Можна побачити що якість передбачення становить близько 86%, що є трохи нижче, ніж з використанням навчальної вибірки, але все рівно є досить високим результатом.

Отже, після завершення навчання, нейронну мережу можна використовувати за прямим призначенням - передбачувати, що зображено на рисунках. Для цього необхідно використовувати метод model.predict() в моделі.

```
[ ]  predictions = model.predict(x_train)

     1875/1875 [==============================] - 2s 1ms/step
```

Рис. 3.18. Метод predict

Передбачення буде проводитися на моделях, яких нейронна мережа навчалася, тому в дужках вказується x_train. Тепер, можна вивести те, що передбачить модель за зображенням, наприклад з індексом 0.

```
predictions[0]

array([8.4331786e-09, 1.0559025e-09, 2.4688855e-09, 5.4975260e-09,
       1.2471009e-09, 2.7258645e-04, 2.2361569e-08, 6.5918977e-04,
       7.1694940e-06, 9.9906105e-01], dtype=float32)
```

Рис. 3.19. Результат першого передбачення

На зображенні видно 10 значень, що відповідають кількості нейронів і кількості класів. У кожного числа в кінці стоїть значення з від'ємним показником, це означає, що значення має від'ємну степінь, після нуля є ще якась кількість нулів, тобто вірогідність близька нулю. І тільки одне число має значення в -1 степінь, саме значення має дев'ять цілих дев'яносто дев'ять сотих, це означає близько 9.99 - дуже близько до 1, а число, рівне 1 означає правильний результат на 100%.

Таким чином, модель з вірогідністю майже 100% передбачає, що це зображення відповідає останньому класу.

Щоб не затруднятися та не шукати, яке з чисел вихідних даних має максимальне значення, можна скористатися функцією argmax() із бібліотеки numpy, вона видає максимальне значення.

```
[23] np.argmax(predictions[0])
0s

     9
```

Рис. 3.20. Застосування функції максимізації

Після виконання коду функція видала максимальне значення, яке дорівнює = 9. Тепер можна перевірити точність і вивести правильну відповідь із міток.

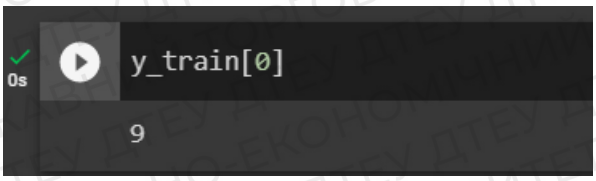

Рис. 3.21. Правильна відповідь

Правильна відповідь за поточним зображенням з індексом 0 виявилася 9, що відповідає передбаченню штучного інтелекту, це означає, що модель працює коректно. Можна протестувати з іншими зображеннями, замінивши число в індексі на інше із вибірки 60000.

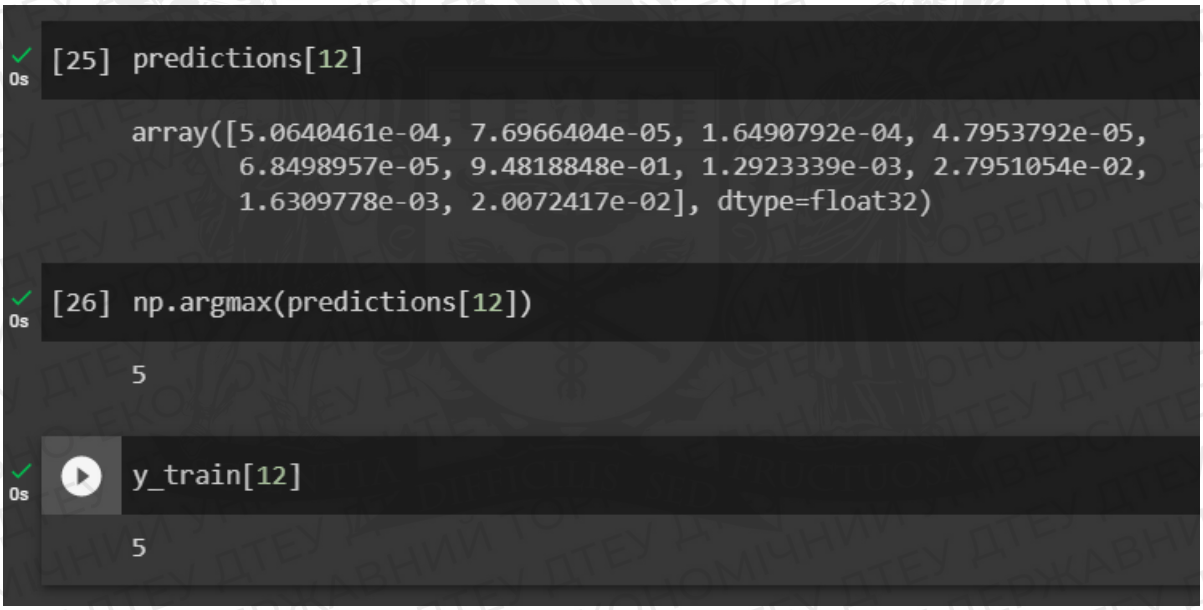

Рис. 3.22. Додаткове тестування

Протестувавши число 12 модель видає відповідь 5, що відповідає дійсності, результат є правильним.

Для наглядності результатів можна вивести та подивитися на зображення.

```
plt.figure()
plt.imshow(x_train[12])
plt.colorbar()
plt.grid(False)
```

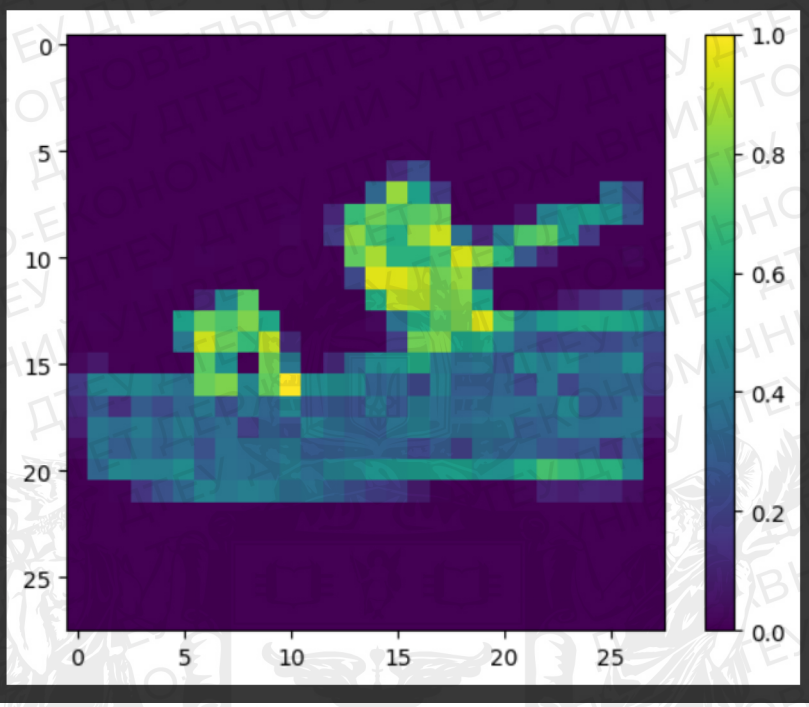

Рис. 3.23. Візуалізація результату

Також, додатково можна вивести назви класів.

```
class_names[np.argmax(predictions[12])]
```

```
'Sandal'
```

Рис. 3.24. Назва класу

Отже, нейронна мережа дала правильну відповідь, вивела коректне зображення та назву класу, які відповідають дійсності.

Тепер можна перейти до більш глибокого тестування за допомогою рандомізації зображень, створення функції якої було описано вище, та подивитися результати роботи.

```
[ ]  plot_random_image(model,x_train,y_train,class_names)
```

Рис. 3.25. Виклик функції рандомізації

Вказавши функції тренувальну вибірку, назву класу, яку передбачає нейронна мережа та реальну назву класу можна запускати роботу функції.

Рис. 3.26. Приклад роботи рандомізації

З дев’яти речей представлених на фото, нейронна мережа зробила правильне передбачення для восьми. Три з яких було передбачено зі 100% ймовірністю і ще дві з ймовірністю більше 90%, що є досить хорошим результатом.

Рис. 3.27. Додатковий приклад 1

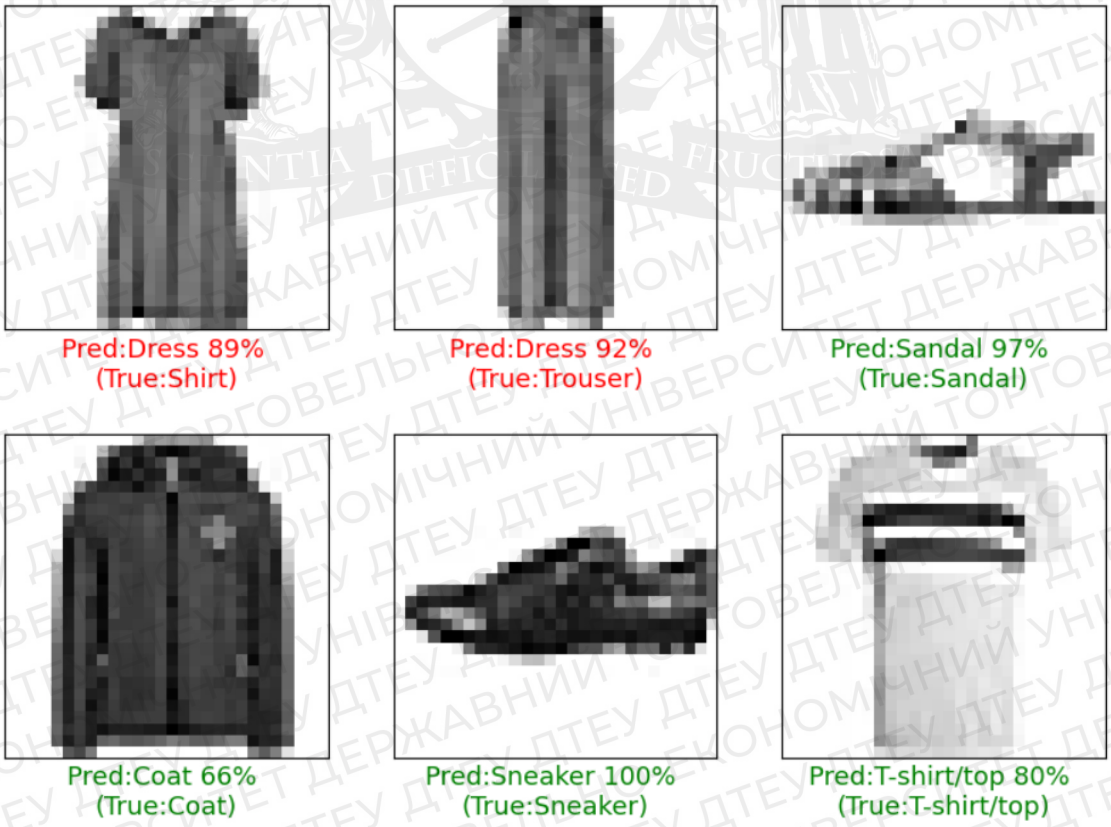

Рис. 3.28. Додатковий приклад 2

Цього разу модель вивела 12 зображень, 10 з яких були правильно передбачені. Значна частина яких передбачена з досить високим відсотком, хоча є правильні відповіді і з нижчим, наприклад 66% чи 45%.

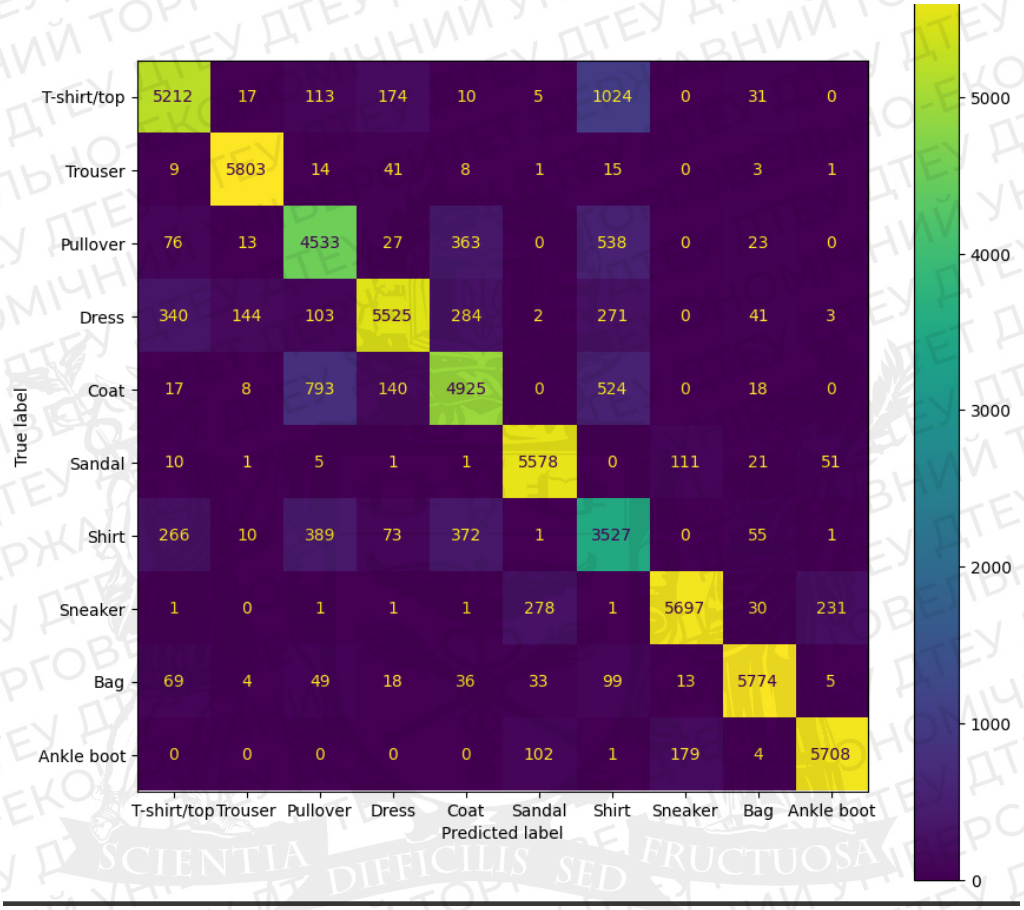

Рис. 3.29. Результат матриці помилок

Також, провівши тестування на матриці помилок можна вивести тенденцію, що найбільше помилок нейронна мережа робить між подібними класами, як сорочка/футболка, футболка/гольф або гольф/пальто.

Підсумовуючи результати, було імпортовано датасет зображень, який розділили на навчальну та тестову вибірки розміром 60000 і 10000 зображень відповідно. Далі було оптимізовано зображення розділивши інтенсивність на 255 для кращого сприйняття інформації штучним інтелектом. Наступним кроком було створення архітектури нейронної мережі, яка складалася з трьох шарів. Модель було скомпільовано, тобто вказано параметри навчання, та проведено

саме навчання за допомогою навчальної вибірки і в результаті задовільно протестовано окремо в ручну, за допомогою функції рандомізації та підведено статистику завдяки матриці помилок на тестовій вибірці.

ВИСНОВКИ

У випускній кваліфікаційній роботі представлено результати теоретичних та практичних досліджень, що полягають у створенні згорткової нейронної мережі, яка може розпізнавати предмети одягу з метою дослідження можливостей комп'ютерного зору у сфері машинного навчання і покращення особистих навичок як спеціаліста. Завдяки проведеним теоретичним дослідженням стало можливим детальне вивчення аспектів машинного навчання, створення моделі тришарової згорткової нейронної мережі і подальша практична реалізація моделі нейронної мережі та апробація результатів дослідження. В результаті отримано наступні **висновки:**

1. На момент реалізації випускної кваліфікаційної роботи створення нейронної мережі є актуальним завданням завдяки стрімкому розвитку технологій комп'ютерного зору та машинного навчання. Нейронні мережі стають потужним інструментом для аналізу великого обсягу інформації. Застосування таких нейронних мереж має великий спектр можливостей, включно зі сферою моди або підбору одягу.

2. Для розпізнавання образів можна застосовувати різні технології та методики, наприклад метод вибіркового пошуку, який базується на виборі певної ділянки зображення для пошуку об'єкта і подальшому порівнянні з іншими ділянками, або метод підбору відповідно інтенсивності пікселя, який і застосовано у дипломному проекті.

3. Завдяки сучасним мовам програмування, фреймворкам, бібліотекам та датасетам процес написання коду значно полегшується і дозволяє програмісту сфокусуватися на завданнях вищого рівня, продумуванні логіки роботи, апробації результатів та подібних задачах, що значно полегшує життя та підвищує ефективність роботи над розробкою проекту.

4. Під час створення моделі нейронної мережі використання мови Python, бібліотеки TensorFlow та фреймворка Keras довели свою ефективність, оскільки мають зручний синтаксис, значний рівень інтеграції один з одним, велику інформаційну базу, офіційну документацію та експерименти інших дослідників у

сфері комп’ютерного зору, що дозволило швидко вивчити нові аспекти машинного навчання і приступити до практичної реалізації проекту.

5. Розроблена модель нейронної мережі виявила значну ефективність у вирішені поставленої задачі, що в залежності від проведених тестувань становить близько дев’яносто відсотків. Розроблена функція демонстрації випадкових зображень дозволила наглядно оцінити ефективність передбачення моделі, а завдяки матриці помилок стало можливим виявлення основних класів, за якими нейронна мережа здійснює найбільшу кількість невірних передбачень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Творошенко І. С. Конспект лекцій з дисципліни «Цифрова обробка зображень» – Геодезія, картографія та землеустрій / І. С. Творошенко ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2017. – 75 с.
2. Методика та організація наукових досліджень : Навч. посіб. / С. Е. Важинський, Т І. Щербак. – Суми: СумДПУ імені А.С. Макаренка, 2016. – 260 с.
3. Селянкін В.В. Комп'ютерний зір. Аналіз та обробка зображень: навчальний посібник / В. В. Селянкін., 2019 - 152с.
4. Нейронні мережі : теорія та практика: навч. посіб. / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с.
5. Метод опорних векторів / (SVM) О. І. ШЕРЕМЕТ ,О. В. САДОВОЙ, 2013.
6. Форсайт Д., Понс Ж. - Комп'ютерний зір. Сучасний підхід - М.:, 2004
7. Теорія ймовірностей та математична статистика / Е. Ю. Железнякова, І. Л. Лебедєва, С. С. Лебедєв – Харків, 2018.
8. Машинне навчання з використанням мікрокомп’ютерів: навч.-метод. посіб. / за ред. О. В. Лісового та ін. – К., 2019. – 224 с.
9. Документація Python [Електронний ресурс] - https://docs.python.org/uk/3/
10. Документація TensorFlow [Електронний ресурс] - https://www.tensorflow.org/api_docs
11. Документація Keras [Електронний ресурс] - https://keras.io/api/
12. Навчання машин та штучний інтелект: конспект лекцій з дисципліни «Технлогії програмування» ч. 2 «Навчання машин та штучний інтелект» / уклад. Верескун М.В.. – Маріуполь : ДВНЗ «ПДТУ», 2019. – 84 с.
13. Інтеллектуальні методи в управлінні: навчальний посібник / Л. В. Дранишников. — Кам'янське: ДДТУ, 2018. — 416 с.
14. Нейронні мережі прямого поширення [Електронний ресурс] - https://moodle.znu.edu.ua/pluginfile.php/675802/mod_resource/content/1/Subbotin_Neural-44-63.pdf

15. Жалдак М.І., Триус Ю.В. Основи теорії і методів оптимізації: Навчальний посібник. - Черкаси: Брама-Україна, 2005. - 608 с.
16. Методи нелінійного програмування. Автоматизація та комп'ютерноінтегровані технологічні комплекси Уклад.: А.І. Жученко, Л. Р. Ладієва, О. В. Снігур. – К.: НТУУ «КПІ», 2007. - 99 с.
17. Матриця помилок в машинному навчанні [Електронний ресурс] - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html
18. Документація NumPy [Електронний ресурс] - https://numpy.org/doc/stable/reference/

ДТЕУ ДТЕУ ДТЕУ