**ДЕРЖАВНИЙ УНІВЕРСИТЕТ**
**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**
**НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ**
**ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**
**КАФЕДРА КОМП'ЮТЕРНИХ НАУК**

# КВАЛІФІКАЦІЙНА РОБОТА

на тему: «**Інтелектуальна система розпізнавання жестової мови на основі технології Python**»

на здобуття освітнього ступеня магістра
зі спеціальності 122 Комп'ютерні науки
*(код, найменування спеціальності)*
освітньо-професійної програми Комп'ютерні науки
*(назва)*

*Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело*

_____ Олексій КУЗЬМЕНКО
*(підпис)* *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав:
здобувач вищої освіти Олексій КУЗЬМЕНКО
група КНДМ-62

Керівник: к.т.н., доцент Микола ГНІДЕНКО
*науковий ступінь, вчене звання*

Рецензент:
*науковий ступінь, вчене звання*
_____
*(Ім'я, ПРІЗВИЩЕ)*

**Київ 2024**

ДЕРЖАВНИЙ УНІВЕРСИТЕТ

**ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

**Навчально-науковий інститут інформаційних технологій**

Кафедра Комп'ютерних наук

Ступінь вищої освіти Магістр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

**ЗАТВЕРДЖУЮ**
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2024 р.

# ЗАВДАННЯ
# НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кузьменку Олексію Вікторовичу
*(прізвище, ім'я, по батькові здобувача)*

1. Тема кваліфікаційної роботи: «Інтелектуальна система розпізнавання жестової мови на основі технології Python»
керівник кваліфікаційної роботи Микола ГНІДЕНКО, к.т.н., доцент.
*(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)*
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024р. №320

2. Строк подання кваліфікаційної роботи «15» грудня 2024р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література, Науково-технічна література, методи збору та аналізу даних, створення та обробка власного дата-сету для розпізнавання жестової мови, алгоритми візуалізації жестових даних.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

4.1. Дослідження методів збору даних та алгоритмів для створення дата-сету жестової мови.

4.2 Запис та обробка власного набору даних для навчання системи розпізнавання жестів.

4.3 Розробка прототипу системи для розпізнавання жестової мови на основі створеного дата-сету.

5. Перелік графічного матеріалу: *презентація*
5.1 Основні методи розпізнавання жестової мови та їх візуалізація.
5.2 Підходи до проєктування прототипу системи.
5.3 Приклади програмного коду.
5.4 Головні компоненти розробленої системи.
5.5 Демонстрація роботи системи на тестових даних.
5.6 Аналіз ефективності алгоритмів (графіки точності та втрат).
5.7 Схема архітектури системи та взаємодії модулів.
5.8 Приклади реальних сценаріїв використання системи.
5.9 Рекомендації щодо подальшого розвитку системи.

6. Дата видачі завдання «16» вересня 2024 р.

## КАЛЕНДАРНИЙ ПЛАН

| № з/п | Назва етапів кваліфікаційної роботи | Строк виконання етапів роботи | Примітка |
|---|---|---|---|
| 1 | Аналіз наявної науково-технічної літератури | 16.09-27.09.24 | Вик. |
| 2 | Аналіз методів розпізнавання жестової мови та візуалізації даних | 30.09-11.10.24 | Вик. |
| 3 | Проєктування та підбір технологій реалізації | 14.10-25.10.24 | Вик. |
| 4 | Запис та обробка власного набору даних для навчання системи | 28.10-08.11.24 | Вик. |
| 5 | Розробка алгоритмів розпізнавання жестової мови | 11.11-22.12.24 | Вик. |
| 6 | Головні компоненти створеної системи | 25.11-29.11.24 | Вик. |
| 7 | Оформлення роботи: вступ, висновки, реферат | 02.12-06.12.24 | Вик. |
| 8 | Розробка демонстраційних матеріалів | 09.12-13.12.24 | Вик. |

Здобувач вищої освіти _____ Олексій КУЗЬМЕНКО
*(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Микола ГНІДЕНКО
*(підпис)* *(Ім'я, ПРІЗВИЩЕ)*

# РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістра:72 стор., 52 рис., 10 джерел.

*Мета роботи* – підвищити ефективність розпізнавання жестів за допомогою машинного навчання та створення інтелектуальної системи на основі зібраних даних.

*Об'єкт дослідження* – процес розпізнавання жестової мови та створення навчального дата-сету.

*Предмет дослідження* – алгоритми та методи розпізнавання жестових даних із використанням технологій машинного навчання.

*Короткий зміст роботи*: У роботі проведено аналіз сучасних методів розпізнавання жестової мови, включаючи використання алгоритмів машинного навчання та нейронних мереж. Особливу увагу приділено дослідженню ефективності методів обробки зображень, зокрема згорткових нейронних мереж (CNN), які є ключовими для розпізнавання візуальних даних.

Запропоновано створення навчального дата-сету шляхом запису та обробки власних даних, що дозволяє врахувати специфіку жестових команд. Розробка цього набору даних стала важливим етапом для підвищення точності системи. Використання Python обґрунтовано як оптимального інструмента для реалізації моделей машинного навчання, завдяки широким можливостям бібліотек TensorFlow, Keras і OpenCV.

У ході роботи було розроблено інтелектуальну систему, що розпізнає жести в реальному часі за допомогою камери. Система демонструє здатність ефективно класифікувати жести та виводити результати у вигляді текстових повідомлень або команд для керування іншими пристроями.

Ключові слова: жестова мова, машинне навчання, дата-сет, Python, розпізнавання жестів, нейронні мережі.

## ABSTRACT

Text part of the master's qualification work:72 pages,52 pictures,10 sources.

*Goal of the work* is to improve the efficiency of gesture recognition using machine learning and to develop an intelligent system based on collected data.

*Object of research* – the process of sign language recognition and the creation of a training dataset.

*Subject of research* – algorithms and methods for recognizing gesture data using machine learning technologies.

*Research methods* – The study analyzes modern methods of gesture recognition, including the use of machine learning algorithms and neural networks. Particular attention is given to investigating the efficiency of image processing methods, specifically convolutional neural networks (CNN), which are critical for recognizing visual data.

A custom dataset has been proposed by recording and processing unique gesture data, allowing for the consideration of specific command gestures. Developing this dataset was a crucial step in improving system accuracy. The use of Python is justified as an optimal tool for implementing machine learning models due to the extensive capabilities of libraries such as TensorFlow, Keras, and OpenCV.

As part of the work, an intelligent system was developed to recognize gestures in real time using a camera. The system demonstrates the ability to efficiently classify gestures and output results in the form of text messages or commands for controlling other devices.



Keywords: sign language, machine learning, dataset, Python, gesture recognition, neural networks.

**ЗМІСТ**

# ВСТУП

Розпізнавання мови жестів є однією з перспективних і важливих сфер досліджень у галузі штучного інтелекту та комп'ютерного зору. Мова жестів слугує унікальним засобом комунікації, що використовує рухи рук, пальців та міміку для передачі інформації. Розвиток систем розпізнавання жестів відкриває широкі можливості для впровадження технологій у сферах доступності, віртуальної реальності, інтерфейсів та мультимодальної взаємодії з цифровими пристроями.

Сучасні підходи до автоматичного розпізнавання жестової мови базуються на застосуванні алгоритмів для навчання машин або алгоритмів, що використовуються у машинному навчанні нейронних мереж. Завдяки своїй гнучкості й потужним бібліотекам, мова програмування Python є одним із провідних інструментів у реалізації таких систем. У цьому дослідженні використовуються популярні бібліотеки TensorFlow, PyTorch та OpenCV для обробки зображень і побудови глибоких нейронних моделей.

Метою роботи є розробка інтелектуальної системи розпізнавання жестової мови з високою точністю та ефективністю. Основна увага приділяється створенню власного дата-сету, застосуванню згорткових нейронних мереж (CNN) та технік попередньої обробки даних для підвищення точності розпізнавання.

У процесі дослідження будуть вивчені сучасні методи аналізу відео та обробки зображень, а також розглянуті ключові алгоритми для інтерпретації жестів. Розроблена система забезпечить автоматичне розпізнавання жестів у реальному часі, що сприятиме покращенню взаємодії людини з комп’ютером та полегшенню комунікації для людей із порушеннями слуху.

Результати цієї роботи можуть бути використані для інтеграції систем розпізнавання жестів у технології доступності, освітні платформи та інноваційні інтерфейси в цифрових середовищах.

# 1 ДОСЛІДЖЕННЯ НЕЙРОННИХ МЕРЕЖ

Штучні нейронні мережі — це системи, які відтворюють принципи функціонування людської нервової системи з метою вирішення обчислювальних завдань. Вони здатні вирішувати складні завдання шляхом аналізу даних і поступового вдосконалення своєї роботи на основі наявних прикладів.

Нейронні мережі можуть навчатися розпізнавати об’єкти, наприклад, кішок, аналізуючи набір зображень із позначками "кіт" і "не кіт". При цьому вони самостійно формують набір ключових характеристик, необхідних для ідентифікації, і застосовують їх для розпізнавання об’єктів на нових зображеннях. Завдяки цьому нейронні мережі здатні вирішувати завдання, що вимагають високого рівня обчислювальної аналітики, схожої на процеси в мозку людини.

Основні сфери застосування штучних нейронних мереж включають класифікацію, розпізнавання образів, прогнозування та аналіз даних.Класифікація дозволяє групувати дані за певними критеріями, наприклад, оцінювати вартість нерухомості залежно від її параметрів. Прогнозування використовується для передбачення майбутніх подій, таких як зміни на ринку або економічні тренди. Розпізнавання є одним із найпоширеніших напрямів, що включає ідентифікацію облич, об’єктів на зображеннях чи інших даних.

ШНМ проявляють високу ефективність у таких галузях, як комп’ютерний зір, обробка природних мов, медична діагностика, автономна навігація та інші. Завдяки прогресу в технологіях машинного навчання та глибокого навчання вони продовжують розширювати свої можливості, досягаючи значних успіхів у таких сферах, як розпізнавання мови жестів та інноваційні напрямки сучасних досліджень.

### 1.1 Принципи роботи нейронів у інтелектуальних системах

Нейрони є базовими обчислювальними одиницями штучних нейронних мереж. Вони приймають вхідні дані, проводять базові обчислення та передають отримані результати іншим нейронам (Рисунок 1.1). У залежності від їх функцій у нейромережі, нейрони поділяються на вхідні (сині), приховані (червоні) та вихідні (зелені).

ШНМ містить велику кількістьнейронів, організованих у шари. Вхідний шар приймає початкову інформацію, приховані шари виконують обчислення, а вихідний шар видає кінцевий результат. Кожен нейрон характеризується двома основними параметрами: вхідними даними та вихідними результатами. У вхідних нейронах вихід дорівнює входу (input=output). В інших нейронах вхідна інформація обробляється за допомогою функції активації, що дозволяє моделювати складні залежності між вхідними та вихідними даними.

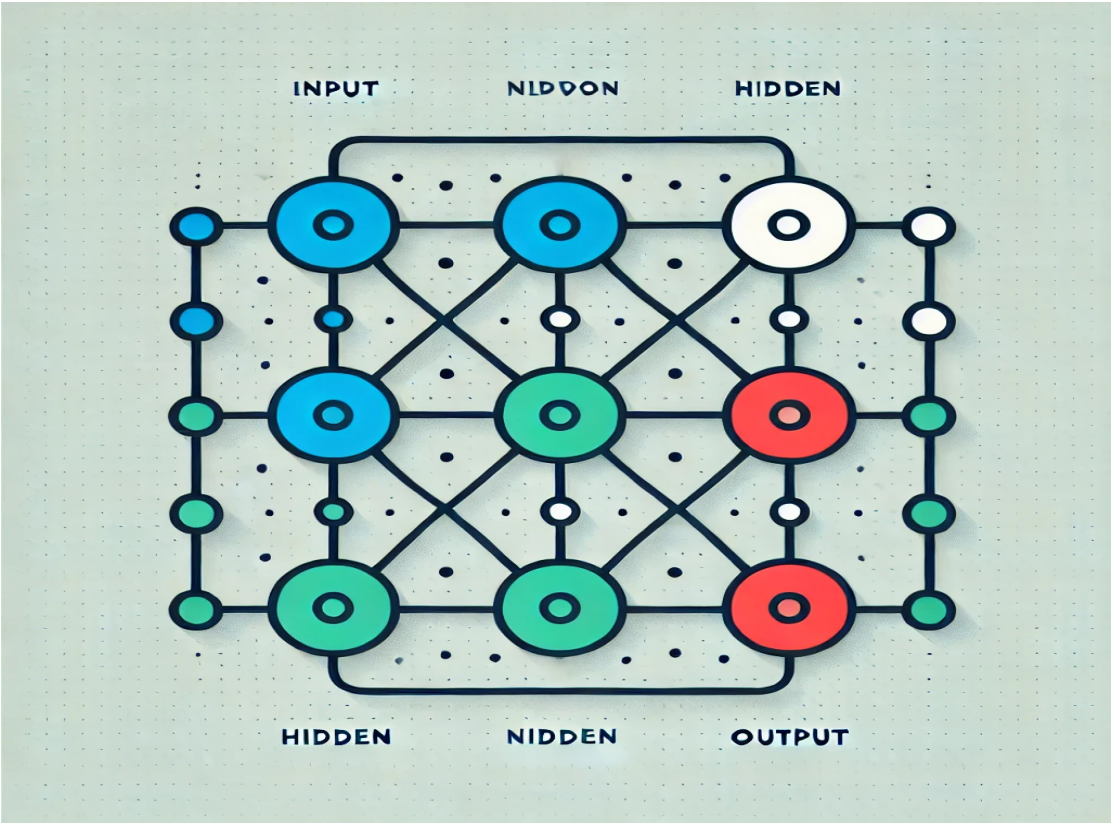



Рисунок 1.1 - Проектування нейронних мереж

Синапси забезпечують зв'язок між нейронами та характеризуються вагами, які визначають ступінь впливу одного нейрона на інший. Чим більша вага синапсу, тим сильніше його вплив на наступний нейрон.

Матриця ваг, що містить значення всіх синапсів, фактично є "мозком" нейронної мережі. Саме ці ваги дозволяють системі обробляти вхідну інформацію та отримувати необхідний результат.

Таким чином, нейрони та синапси у нейромережі працюють у тісній взаємодії, передаючи та обробляючи дані для ефективного вирішення поставлених завдань.

### 1.2 Види навчання нейронних мереж

Існує кілька основних підходів до навчання нейронних мереж, які можна класифікувати наступним чином: Supervised Learning, Unsupervised Learning, Semi-Supervised Learning та Reinforcement Learning.

Supervised Learning (навчання з учителем) використовує розмічені дані набору даних, де кожен вхідний приклад містить відповідну мітку або клас. Модель навчається прогнозувати ці мітки, аналізуючи вхідні дані, а її результати порівнюються з еталонними значеннями для оцінки точності. У процесі навчання нейромережа прагне мінімізувати помилки та наблизити свої прогнози до правильних відповідей.

Unsupervised Learning (навчання без учителя) використовується, коли вихідні дані не мають міток або класів. У цьому випадку модель самостійно шукає корисні ознаки, залежності та структури у даних. До основних завдань такого навчання належать Кластеризація, скорочення розмірності, знаходження аномалій та відновлення втрачених даних. Головною метою є створення внутрішнього представлення інформації без використання міток.

Semi-Supervised Learning (навчання з частковим використанням учителя) поєднує розмічені та нерозмічені дані. Невеликий обсяг розмічених даних використовується для первинного навчання моделі на конкретних задачах, тоді як великий набір нерозмічених даних допомагає виділити загальні ознаки та

залежності. Це дозволяє покращити продуктивність моделі, використовуючи значно менше розмічених ресурсів.

Reinforcement Learning (Навчання з підкріпленням) ґрунтується на системі заохочень і покарань для навчання моделі взаємодії з динамічним середовищем. Агент отримує відгук у формі винагород або покарань залежно від своїх дій. Основна мета полягає у тому, щоб агент навчився приймати оптимальні рішення, максимізуючи загальну винагороду. Цей підхід нагадує дресирування тварин, коли позитивне підкріплення допомагає закріпити бажану поведінку.

Таким чином, нейронні мережі можна навчати різними методами, залежно від доступності розмічених даних, необхідності виявлення ознак або потреби у взаємодії з навколишнім середовищем.

### 1.3 Глибоке навчання (Deep Learning)

Глибоке навчання є галузь машинного навчання, яка використовує глибокі нейронні мережі для вирішення складних завдань, таких як розпізнавання та класифікація. та аналіз даних. Завдяки своїм вражаючим результатам у багатьох областях, включаючи розпізнавання мови, обробку зображень, природних мов та голосу.

Особливо багатообіцяючим є використання глибокого навчання для розпізнавання мови жестів. Ця технологія дозволяє автоматично витягати ключові ознаки з великих обсягів даних, що представляють жести, та навчати моделі (Рисунок 1.2) для точного розпізнавання та класифікації різноманітних жестів.

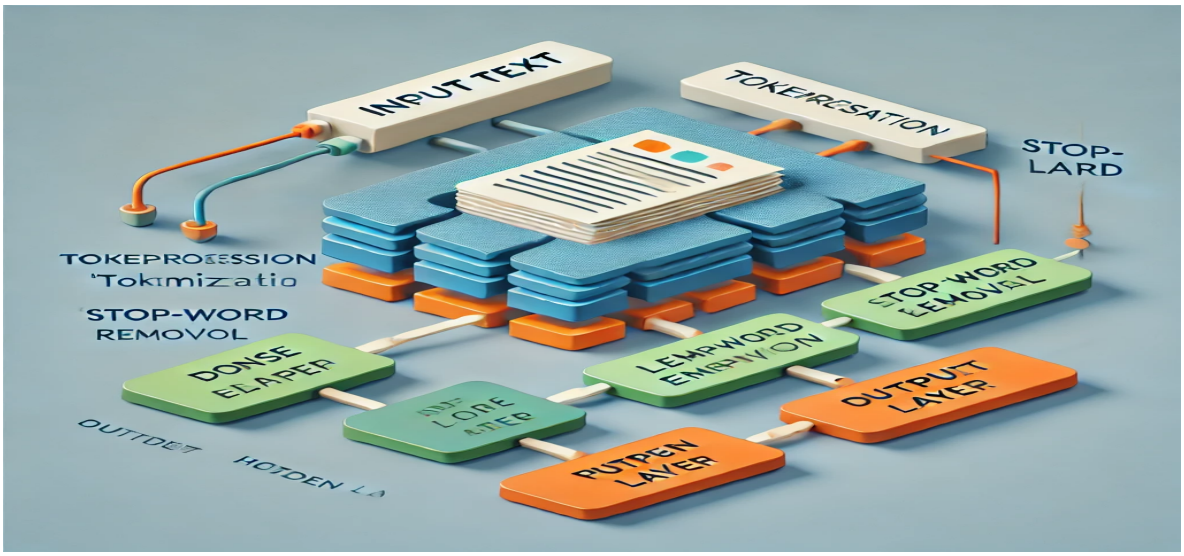



Рисунок 1.2 - Аналіз настрою на основі глибокого навчання.

Одним із популярних методів для розпізнавання жестової мови є використання згорткових нейронних мереж (CNN), які автоматично виділяють ключові особливості зображення, як-от контури та форми, передаючи їх до класифікаційного шару.

Для жестів у послідовностях застосовуються рекурентні нейронні мережі (RNN), що враховують часові залежності та контекст. Поєднання CNN для обробки зображень та RNN для аналізу послідовностей дозволяє створювати ефективніші системи.

Ефективне навчання потребує великого обсягу розмічених даних, як-от відеозаписи жестів із анотаціями. Це дає змогу тренувати моделі та досягати високої точності.

Методи глибокого навчання мають широкий потенціал у медицині, віртуальній реальності, робототехніці та ігровій індустрії. Вони допомагають створювати інтуїтивні системи, здатні розуміти жести користувача, полегшуючи взаємодію з технологіями.

# 2 ДОСЛІДЖЕННЯ ВИДІВ НЕЙРОННИХ МЕРЕЖ

## 2.1 Принципи роботи та можливості згорткової нейронної мережі

Згорткова нейронна мережа, відома також як ConvNet, є різновидом штучної нейронної мережі з глибокою прямопрямою архітектурою. Вона відрізняється здатністю до узагальнення, яка значно перевершує можливості мереж із повнозв’язними шарами. Ефективно навчається розпізнавати абстрактні ознаки об’єктів, зокрема просторових даних, і успішно використовує їх для точного ідентифікування (Рисунок 2.1).

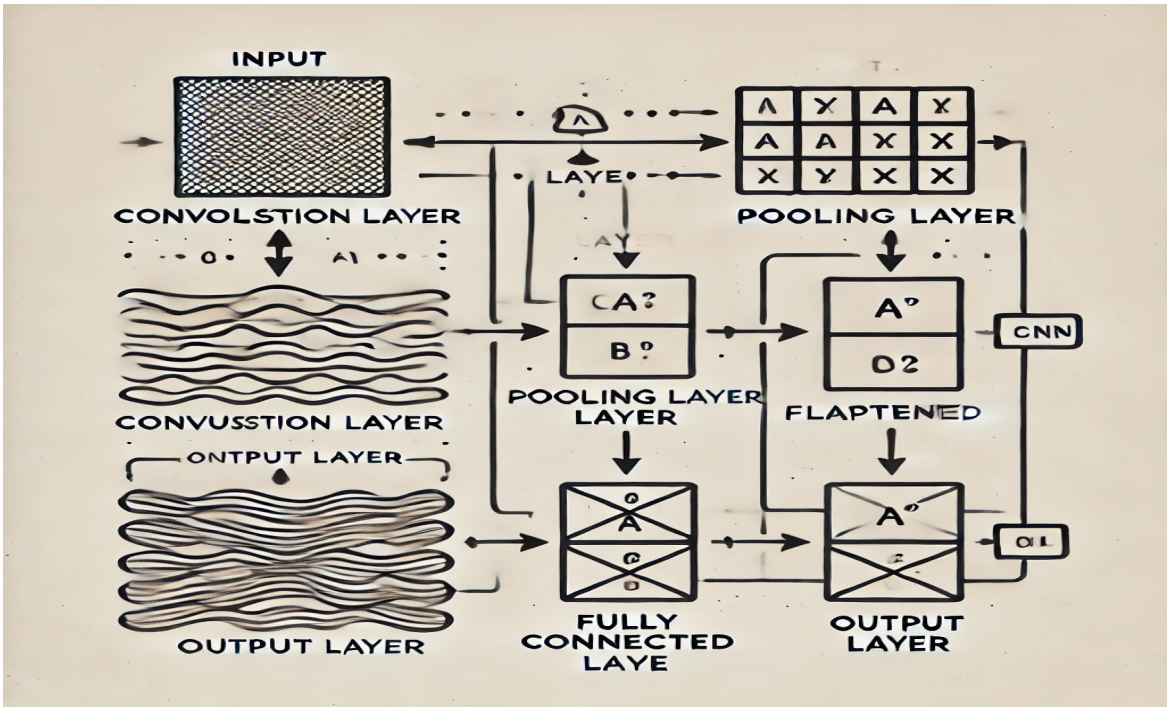



Рисунок 2.1 - Концептуальна модель згорткової нейронної мережі

Моделі глибоких згорткових нейронних мереж складаються з обмеженої кількості обробних шарів, які можуть аналізувати різні особливості вхідних даних (наприклад, зображень) на різних рівнях абстракції. Перші шари зосереджені на вилученні простих високорівневих ознак, таких як лінії, контури і текстури, тоді як глибші шари можуть вивчати більш складні низькорівневі ознаки, що відповідають об'єктам або частинам об'єктів. Базова концепція згорткової нейронної мережі показана на Рисунку 2.1, а різні шари більш детально описані в наступних розділах.

Згорткові нейронні мережі вважаються більш перспективними, ніж класичні нейронні мережі в контексті комп'ютерного зору з кількох причин. По-перше, вони можуть ефективно обробляти просторові дані: використовуючи згорткові ядра, які аналізують локальні області зображення, CNN можуть зберігати просторові зв'язки між пікселями. Це дозволяє виявляти локальні особливості, такі як контури, текстури та кольори. По-друге, зменшується кількість параметрів. Замість повністю пов'язаних шарів, які вимагають великої кількості ваг для кожного нейрона, CNN використовують згорткові фільтри, які локально сканують вхідні дані. Це зменшує обчислювальну складність і покращує продуктивність мережі. По-третє, він інваріантний до зсувів і масштабування. Завдяки шару субдискретизації згорткові нейронні мережі можуть навчитися розпізнавати об'єкти незалежно від їхнього положення або розміру на зображенні. Нарешті, ознаки мають ієрархічну структуру. Перші шари мережі виділяють базові ознаки, тоді як глибші шари комбінують їх для розпізнавання складних структур та об'єктів. Такий підхід робить CNN ефективними в таких завданнях комп'ютерного зору, як розпізнавання об'єктів, класифікація і сегментація зображень. Однією з головних переваг згорткових нейронних мереж є властивість розподілу ваги, що значно зменшує кількість параметрів, які потрібно навчати. Це запобігає перенавчанню моделі та покращує її здатність до узагальнення нових даних. У згорткових нейронних мережах шари виділення ознак і класифікації навчаються одночасно. Це забезпечує тісну кореляцію між вилученими ознаками та кінцевими результатами, роблячи вихід моделі більш регулярним і залежним від якості вилучених ознак. Реалізація великих, складних нейронних мереж менш ефективна, ніж згорткові нейронні мережі інших типів архітектури; ШНМ, за своєю природою, можуть обробляти великі обсяги даних з невеликими обчислювальними зусиллями. На сьогоднішній день згорткові нейронні мережі є основним інструментом для досягнення високих результатів у задачах комп'ютерного зору. Вони успішно використовуються в таких завданнях, як класифікація зображень, виявлення і розпізнавання об'єктів, розпізнавання облич і виразів обличчя, виявлення транспортних засобів, розпізнавання тексту і обробка мовлення. Завдяки своїй

ефективності та універсальності CNN залишаються важливим компонентом сучасних систем штучного інтелекту.

### 2.2 Область застосування згорткової нейронної мережі

Нейронні мережі показали чудові результати в розпізнаванні об'єктів, особливо в задачах розпізнавання мови жестів. Завдяки своїй здатності самостійно засвоювати ключові важливі ознаки з великої кількості зображень без необхідності ручного програмування правил, CNN перетворилися на ефективний засіб для розв'язання завдань комп'ютерного зору та обробки візуальної інформації CNN особливо ефективні в задачах, що вимагають аналізу візуальних даних, таких як зображення і відео особливо ефективні; вони широко використовуються в різних галузях, які вимагають автоматизації і високоточних результатів.1.Передбачення: Спочатку, згорткова нейронна мережа приймає вхідне зображення та проводить передбачення щодо наявності та положення різних об'єктів на зображенні.

Класифікація зображень є одним з основних застосувань згорткових нейронних мереж . Ці мережі дуже ефективні для розпізнавання об'єктів і класифікації зображень за різними категоріями.Основні етапи процесу класифікації зображень за допомогою такі:

1.Передбачення: На першому етапі отримує вхідне зображення і аналізує його, створюючи передбачення щодо наявності та розташування різних об'єктів на зображенні.

2.Попереднє навчання: Прогнози, зроблені на попередньому кроці, порівнюються з попередньо навченими прикладами та класифікаційними мітками. Цей процес базується на вагових коефіцієнтах, оптимізованих під час навчання на великих наборах зображень.

3. Класифікація: Порівнюючи передбачення з навчальними прикладами, CNN визначає категорію, до якої належить вхідне зображення. Наприклад, вони класифікуються як коти або собаки.

4. Вихідні дані: видає класифікацію, яка показує, до якого класу належить вхідне зображення, або надає ймовірність кожного з можливих класів.

Класифікація зображень широко використовується в різних галузях, таких як медична діагностика, автономна навігація, системи безпеки та розпізнавання облич; CNN можуть бути використані для автоматизації процесу класифікації та ідентифікації об'єктів на зображеннях з високою точністю.

Виявлення об'єктів: виявлення об'єктів є одним з основних застосувань згорткових нейронних мереж , які можуть ідентифікувати об'єкти на зображенні, визначаючи їхнє точне положення та окреслюючи їхні межі.

Основні процеси виявлення об'єктів за допомогою наступні:

1. Прогнозування: На першому етапі аналізує вхідне зображення і робить припущення про наявність на ньому об'єктів.

2. Локалізація об'єкта: На наступному етапі визначає місцезнаходження об'єкта на зображенні, виділяючи ділянки, де об'єкт може бути знайдений.

3. Класифікація об'єктів: Кожна виділена область надсилається до для подальшої обробки, щоб визначити клас об'єкта. Мережа визначає, чи є об'єкт, наприклад, автомобілем, людиною або собакою.

4. Межі об'єктів: На додаток до класифікації, також визначає межі об'єктів, визначаючи точне положення і форму об'єктів на зображенні.

5. Вихідні дані: видає остаточні результати виявлення об'єктів, включаючи класифікаційні мітки та межі кожного об'єкта.

Виявлення об'єктів на основі широко використовується в областях, де потрібне автоматичне розпізнавання об'єктів на зображеннях, зокрема в системах комп'ютерного зору та автономного водіння, системи безпеки, відеоспостереження та допоміжні технології для людей з обмеженими можливостями.

Сегментація зображень: сегментація зображень є одним з основних застосувань згорткових нейронних мереж. Використовуються для визначення меж об'єктів на зображенні та їх класифікації шляхом віднесення пікселів до певних класів або сегментів. Основний процес сегментації зображення складається з наступних кроків:

1. Навчання моделі: На першому етапі модель навчається на великому наборі даних, де кожне зображення має вручну проставлені мітки або межі об'єктів. У

процесі навчання мережа вчиться визначати ознаки та взаємозв'язки між пікселями, які необхідні для класифікації та сегментації об'єктів.

2. Прогнозування: Після завершення навчання модель застосовується до нового зображення, щоб визначити межі об'єкта або класифікувати пікселі за класами. Зображення обробляється мережею, яка надає прогнози або ймовірності того, що кожен піксель або область належить до певного класу.

3. Переносимість: Модель можна застосовувати до нових зображень, які раніше не аналізувалися. Мережа може ефективно виявляти та сегментувати об'єкти на нових зображеннях, використовуючи знання, набуті в процесі навчання.

Сегментація зображень на основі широко використовується в різних галузях, включаючи медичну діагностику, робототехніку, аналіз і обробку зображень. Метод забезпечує автоматичне виявлення і виділення об'єктів на зображеннях з високою точністю і швидкістю, відкриваючи широкі можливості для автоматизації та оптимізації багатьох процесів.

Аналіз відео: є важливим застосуванням згорткових нейронних мереж використовуються для розпізнавання та аналізу об'єктів, дій та подій на відеозображеннях та вилучення важливої інформації Основні етапи процесу аналізу відео з використанням CNN наступні: перший етап - розпізнавання та аналіз відеозображення, який потім використовується для ідентифікації об'єктів, дій та подій на зображенні:

1. Поділ відео на кадри: відеозображення розбивається на окремі кадри для подальшого аналізу. Кожен кадр розглядається як незалежне зображення і може бути використаний для ідентифікації об'єктів і розпізнавання дій.

2. Обробка кадрів за допомогою CNN: кожен кадр пропускається через модель CNN для виявлення та класифікації об'єктів і дій на зображенні; використовують свою здатність витягувати особливості та залежності зображення для точного розпізнавання об'єктів і подій.

3. Відстеження об'єктів: Окрім виявлення об'єктів в окремих кадрах, також можна використовувати для відстеження об'єктів у послідовності кадрів. Це

дозволяє відстежувати рух об'єктів, визначати їхні траєкторії та отримувати більш детальну інформацію про події на відео.

4. Аналіз дій та подій використовуються для розпізнавання та класифікації дій і подій на відеозображеннях. Сюди входить розпізнавання жестів, виявлення людської поведінки, класифікація сцен і виявлення аномалій. Аналіз відео з використанням надає широкі можливості для розуміння та інтерпретації вмісту відео. Він знаходить застосування у багатьох сферах, включаючи відеоспостереження, автономну навігацію, відеоаналітику, медіа-розпізнавання та багато іншого.

Обробка природної мови: Обробка природної мови є галуззю, яка займається аналізом, розумінням і створенням людської мови за допомогою комп'ютерних технологій. Хоча основна сфера застосування Convolutional Neural Networks (CNNs) — це аналіз зображень, вони також ефективно використовуються для вирішення задач NLP.

Що стосується обробки природної мови, то CNN можна використовувати для різноманітних завдань, зокрема:

1.Класифікація тексту: CNN можуть використовуватися для навчання класифікації текстових документів на основі їхнього змісту. Наприклад, вони можуть визначати, чи текст має позитивну або негативну тональність, відносити його до певної тематики або перевіряти наявність конкретної категорії.

2. Виявлення емоцій: CNN можуть застосовуватися для аналізу текстових повідомлень або контенту із соціальних мереж з метою виявлення емоцій. Вони здатні розпізнавати настрій, визначати сентимент чи емоційне забарвлення тексту.

3. Машинний переклад: CNN можуть використовуватися для вирішення завдань машинного перекладу, допомагаючи встановлювати семантичні зв'язки між словами та фразами в різних мовах.

4. Створення тексту: CNN можна використовувати для автоматичного доповнення тексту, яке генерує текст на основі контексту введення або передбачає наступне слово на основі попереднього контексту.

5. Виявлення іменованих об'єктів: CNN можна використовувати для ідентифікації та класифікації іменованих об'єктів у тексті, таких як імена людей, організацій та місць.

Хоча ці приклади показують, що ШНМ можна використовувати для різноманітних завдань в обробці природної мови, варто зазначити, що для деяких завдань, таких як послідовне моделювання, існують інші більш ефективні моделі, такі як рекурентні нейронні мережі (RNN).

Генеративні моделі: генеративні моделі, включаючи генеративні інструменти, використовуються для створення нового контенту, такого як зображення, аудіо або текст, на основі вхідних даних. Згорткові нейронні мережі (CNN) часто використовуються для обробки зображень, але також можуть застосовуватися для генеративних моделей.

Деякі методи використовують CNN для генерації зображень. Наприклад, у глибоких генеративних моделях, таких як Генеративні змагальні мережі , які називається генератором, використовує CNN для створення нових зображень. Інша частина, яка називається дискримінатором, вирішує, чи є зображення реальним, чи згенерованим.

Згорткові нейронні мережі (CNN) також можна використовувати для генерації тексту. Зазвичай це робиться шляхом поєднання CNN з рекурентною нейронною мережею (RNN), наприклад, LSTM або GRU. Рекурентний шар відповідає за моделювання послідовностей слів, тоді як CNN використовується для вилучення ознак з окремих слів і речень.

Загалом, CNN можна використовувати для генеративних моделей, але їхнє застосування зазвичай пов'язане з обробкою зображень. Для генерації тексту можна використовувати комбінацію CNN і RNN або інші моделі, придатні для моделювання послідовностей.

Слід зазначити, що згорткові нейронні мережі (CNN) успішно використовуються і за межами комп'ютерного зору, особливо в розпізнаванні та обробці мовлення. Завдяки своїй універсальності та високій ефективності, CNN широко використовуються в різних завданнях машинного навчання, зокрема

обробка природної мови (NLP) і відеоаналітика. З розвитком глибинного навчання CNN також використовуються в нових сферах, таких як автоматичне розпізнавання жестів, біометрія та обробка мультимедіа в реальному часі.

### 2.3 Основні концепції та можливості рекурентної нейронної мережі

Рекурентні нейронні мережі (RNN) є різновидом штучних нейронних мереж, спеціально розроблених для обробки послідовних даних; відмінність RNN від традиційних нейронних мереж полягає в тому, що вони мають зв'язки, які зберігають інформацію про попередні стани. Основна концепція CNN полягає в їх здатності враховувати контекст і взаємозв'язки між елементами послідовності. Кожен елемент обробляється послідовно і відповідно обчислюється певний стан, який передається для подальшої обробки. Цей стан допомагає зберегти інформацію про попередні вхідні дані і впливає на обробку наступного елемента. Тому CNN широко використовуються для задач, пов'язаних з текстом, мовленням та часовими рядами. Однак для подолання проблеми довготривалої залежності необхідно розвивати такі моделі, як довготривала короткочасна пам'ять (LSTM) і зачинені рекурентні одиниці (GRU), які є популярними завдяки своїй здатності краще та довше зберігати важливу інформацію (Рисунок 2.2)

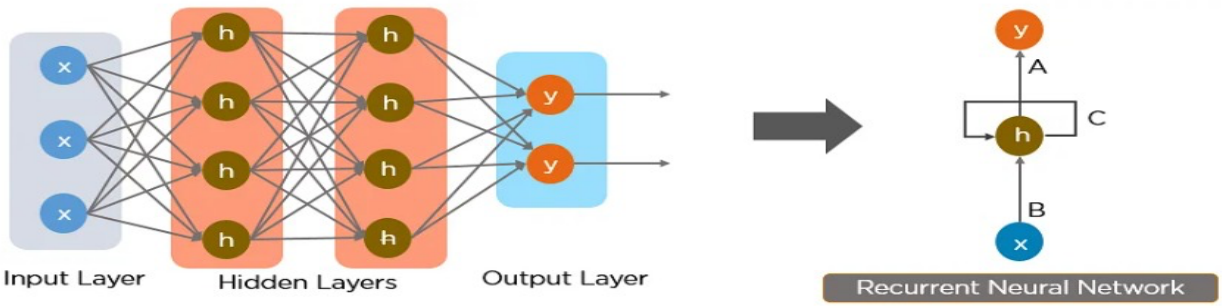



Рисунок 2.2 - Принцип роботи RNN-Recurrent Neural Networks

Однією з основних модифікацій рекурентних нейронних мереж є Long Short-Term Memory (LSTM) і Gated Recurrent Unit (GRU). Ці варіанти дозволяють RNN більш ефективно обробляти довгострокові залежності та вирішують проблему зникнення градієнтів, що може виникнути під час тренування стандартних RNN.

LSTM і GRU мають спеціальні механізми для контролю за потоком інформації, що дає змогу зберігати важливі дані на більший час, а також уникати затухання або вибуху градієнтів під час зворотного поширення помилки. Ці моделі є основою багатьох сучасних досягнень у обробці природної мови та часових рядів.

Рекурентні нейронні мережі мають широке застосування в обробці послідовних даних, зокрема в таких областях, як аналіз мови, машинний переклад, розпізнавання мовлення, створення тексту, аналіз часових рядів та інших. Вони здатні моделювати контекстуальні залежності в послідовностях і можуть обробляти та використовувати інформацію, що передається з часом. Завдяки своїй здатності враховувати попередні елементи послідовності, RNN стали основою для багатьох сучасних систем у сферах обробки природної мови (NLP), автоматичного перекладу та навіть в системах рекомендацій.

З новими досягненнями, такими як трансформери, які теж працюють з послідовними даними, застосування RNN дещо зменшилося, але вони все одно залишаються важливим інструментом в багатьох задачах, де важлива обробка даних у часі.

### 2.4 Область застосування RNN

Рекурентні нейронні мережі (RNN) широко використовуються в багатьох сферах, де важливе моделювання та аналіз послідовних даних. Ось кілька основних напрямів, у яких застосовуються RNN:

Машинний переклад: Однією з ключових областей застосування RNN є машинний переклад. Завдяки здатності RNN обробляти послідовності, вони є потужним інструментом для розв'язання задач перекладу, де важливо зберігати контекст та враховувати залежності між словами в реченні. У машинному перекладі RNN часто використовуються в архітектурах типу енкодер-декодер, де одна мережа (енкодер) кодує текст з вихідної мови, а інша (декодер) генерує переклад на цільову мову.

З розвитком глибоких нейронних мереж та трансформерних моделей, таких як BERT і GPT, застосування RNN у машинному перекладі частково було замінене, однак RNN все ще зберігають важливість у деяких специфічних завданнях.

Енкодер приймає текст на вихідній мові та створює його внутрішнє представлення, яке потім передається до декодера. Декодер поступово генерує переклад на цільову мову, використовуючи кожен згенерований токен як вхід для наступного, що дозволяє моделі враховувати контекст і взаємозв'язки між словами. Такий підхід дозволяє забезпечити більш точний переклад, адже модель може утримувати інформацію про попередні слова в реченні, покращуючи загальну якість трансляції.

З появою нових моделей, таких як трансформери (наприклад, BERT і GPT), цей процес був значно вдосконалений, оскільки трансформери можуть ефективно обробляти довгострокові залежності без необхідності послідовної обробки, що зменшує час тренування і покращує точність перекладу.

Серед популярних архітектур RNN для машинного перекладу можна виділити модель Sequence-to-Sequence (Seq2Seq), яка використовує шари короткочасна пам'ять (LSTM) чи Gated Recurrent Unit (GRU). Ці шари дозволяють моделі краще враховувати довгострокові залежності в тексті, що важливо для точності перекладу, та зменшують проблему зникнення градієнтів, яка є типовою для стандартних RNN. За допомогою LSTM і GRU, моделі можуть зберігати важливу інформацію на довгий час, що робить їх ефективними для обробки складних і довгих речень.

З новими підходами, такими як трансформери (наприклад, модель BERT), які не залежать від послідовного оброблення, ефективність Seq2Seq архітектур зменшилась, однак вони все ще залишаються популярними в деяких специфічних задачах.

Машинний переклад за допомогою RNN дозволяє автоматично перекладати текст між різними мовами, що значно покращує доступність інформації та сприяє взаємодії між людьми, які розмовляють різними мовами. Це важливий крок до глобалізації та полегшення комунікації в умовах багато мовного середовища.

Мовний аналіз: Іншою важливою сферою застосування RNN є мовний аналіз. Завдяки здатності моделювати послідовності, RNN широко використовуються для розв'язання завдань у цій області. Вони застосовуються для таких задач, як розпізнавання частин мови, аналіз синтаксису та семантики, а також для побудови систем автоматичного резюмування та класифікації текстів.

Завдяки розширенню можливостей глибокого навчання та розвитку таких моделей, як трансформери (наприклад, BERT, GPT), роль RNN в аналізі природної мови зменшується, але вони все ще залишаються корисними для задач, що потребують обробки послідовних даних.

RNN можна використовувати в мовному аналізі для виконання різних завдань, зокрема:

Мовне моделювання (Language Modeling): RNN можна застосовувати для побудови моделей, які прогнозують ймовірність наступного слова або символу в послідовності. Це має корисне застосування для автоматичної генерації тексту, автозаповнення, машинного письма та інших мовних завдань.

Сентимент-аналіз: RNN можна застосовувати для виявлення емоційного тону тексту, наприклад, позитивного, негативного або нейтрального. Модель може бути навчена на розмічених даних для класифікації текстів за їхнім сентиментом, що корисно для аналізу відгуків користувачів, публікацій у соціальних мережах тощо.

Розпізнавання іменованих сутностей: RNN можна застосовувати для виявлення та класифікації іменованих сутностей у текстах, таких як імена людей, організації, місця тощо. Модель може бути навчена на розмічених даних для ідентифікації таких сутностей і виконання різних завдань, таких як аналіз новин, обробка текстів та інші.

Машинний переклад: Як зазначалося раніше, RNN можуть застосовуватися для машинного перекладу, де вхідні та вихідні послідовності представляють тексти на різних мовах. RNN дозволяє моделі враховувати контекст і взаємозв'язки між словами, що забезпечує автоматичний переклад між мовами.

Це лише деякі приклади застосування RNN у мовному аналізі. RNN слугує ефективним засобом для моделювання послідовностей та вирішення різноманітних завдань у галузі опрацювання природної мови.

Створення тексту: Одним із застосувань рекурентних нейронних мереж (RNN) є генерація тексту. RNN можна застосовувати для створення моделей, які автоматично генерують тексти, схожі на вхідні, враховуючи ймовірність і контекст.

Для генерації тексту з використанням RNN зазвичай застосовують модель, що називається "Мовною моделлю" (Language Model). Мовна модель тренується на значному обсязі текстових даних і здатна генерувати нові тексти, враховуючи контекст і ймовірності слів.

Процес створення тексту зазвичай виглядає наступним чином:

1. Модель отримує початковий текст або символи як стартовий контекст.

2.Модель використовує свої внутрішні стани та навчені параметри для генерування нового символу або слова, враховуючи ймовірності.

3.Згенерований символ або слово додається до контексту, після чого процес повторюється для створення наступного символу або слова.

4.Цей процес триває, поки не буде досягнуто необхідної кількості символів або поки текст не буде завершено.

RNN використовує свою внутрішню пам'ять і контекст для врахування залежностей між символами та організації згенерованого тексту. Це дозволяє мережі створювати текст, який відповідає контексту та синтаксису, що робить її ефективною для генерації різноманітних видів текстів, таких як поезія, новини, діалоги та інші.

Генерація тексту з використанням RNN має застосування в різних сферах, таких як література, комп'ютерна графіка, реклама та інші.

Розпізнавання мови: Одним із застосувань Recurrent Neural Networks (RNN) є розпізнавання мови. RNN можна застосовувати для ідентифікації та класифікації мовних зразків, таких як усне мовлення, текстові фрази або інші мовні дані.

Для розпізнавання мови з використанням RNN зазвичай застосовують модель, відому як "Класифікатор мови" (Language Classifier) або "Модуль мови"

(Language Module). Ця модель навчається на наборі даних, що містить приклади текстів або мовлення різних мов, та вивчає залежності між вхідними мовними зразками і їх мовними категоріями.

Зазвичай процес розпізнавання мови виглядає наступним чином:

1.Вхідний мовний зразок, наприклад, аудіофайл з мовленням або текстова фраза, подається до моделі.

2.RNN обробляє послідовність вхідних символів або кадрів, аналізуючи їх по черзі і враховуючи контекст.

3.На виході моделі отримується класифікація, яка визначає мовну категорію, до якої відноситься вхідний зразок.

4.Залежно від конкретного застосування, можуть бути виконані додаткові операції з результатами, наприклад, переклад тексту або здійснення відповідних дій.

RNN є ефективним інструментом для розпізнавання мови завдяки своїй здатності моделювати послідовності та контекст вхідних даних. Вона може враховувати залежності між символами, словами або фразами, що дає змогу точно класифікувати мовні зразки.

Розпізнавання мови з використанням RNN знаходить застосування в різноманітних додатках, таких як системи розпізнавання мовлення, автоматичний переклад, мовні асистенти та інші мовні інтерфейси. Це дозволяє комп'ютерам сприймати й аналізувати мову, відкриваючи численні можливості для взаємодії між людьми і машинами.

Аналіз часових рядів: Одним із головних застосувань Recurrent Neural Networks (RNN) є обробка часових рядів. RNN можна використовувати для моделювання та прогнозування часових рядів, де дані представлені у вигляді послідовностей, що змінюються в часі.

RNN здатні аналізувати та враховувати залежності в часових рядах, що дає змогу моделювати та прогнозувати майбутні зміни в даних. Це особливо корисно в сферах, де важливо виявляти і передбачати тренди, цикли, сезонні коливання та інші взаємозв'язки в часових даних.

Процес аналізу часових рядів за допомогою RNN може включати такі етапи:

1.Підготовка даних: Часовий ряд подається на вхід моделі, а також можуть бути використані попередні значення ряду та інші вхідні характеристики.

2.Навчання моделі: RNN тренується на наборі часових рядів, де вона вивчає залежності та створює модель для прогнозування майбутніх значень.

3. Прогнозування: Після навчання модель може застосовуватися для передбачення майбутніх значень часового ряду, спираючись на наявні дані.

RNN є ефективним інструментом для обробки часових рядів, тому що здатна враховувати контекст та залежності в часових даних. Це дозволяє розпізнавати складні шаблони та робити точні прогнози на основі історичних даних.

Застосування RNN для обробки часових рядів широко використовується в таких сферах, як прогнозування фінансових ринків, прогнозування погоди, аналіз виробничих даних, моніторинг медичних показників та інших.

Обробка природних мов є однією з основних сфер застосування Recurrent Neural Networks (RNN). RNN можуть застосовуватися для реалізації різноманітних завдань в обробці природних мов, таких як машинний переклад, генерація тексту, виявлення частин мови, сентимент-аналіз, розпізнавання мови, побудова систем відповідей та багато інших.

Однією з головних особливостей RNN є здатність моделювати залежності в послідовностях даних. В обробці природних мов текст має послідовну структуру, де кожне слово чи символ залежать від попередніх. RNN можуть враховувати ці залежності і використовувати їх для розуміння та генерації тексту.

Застосування RNN в обробці природних мов охоплюють:

1. Машинний переклад: RNN можна використовувати для перекладу текстів між різними мовами. Вони здатні моделювати залежності між словами та використовувати їх для створення перекладу.

2.Генерація тексту: RNN можуть бути навчені створювати новий текст, враховуючи попередній контекст. Це корисно для розробки автоматичних систем генерації тексту, наприклад, для створення новинних статей або творчих текстів.

3.Сентимент-аналіз: RNN можуть застосовуватися для виявлення та класифікації емоційного настрою тексту. Вони здатні визначати позитивний, негативний чи нейтральний тон тексту, спираючись на контекст, і використовувати цю інформацію для аналізу відгуків, коментарів тощо.

4. Розпізнавання мови: RNN можуть застосовуватися для ідентифікації мови на основі акустичних сигналів. Вони здатні аналізувати звукові зразки та визначати мову, якою користувач спілкується.

5.Завдання відповідей: RNN можна використовувати для створення систем автоматичних відповідей на запитання. Вони здатні аналізувати текст запитання і співвідносити його з контекстом, щоб знайти відповідь.

Це лише деякі приклади застосування RNN, але їх потенціал може бути розширений на багато інших сфер, де необхідна обробка послідовних даних і залежностей між ними.

### 2.5 Особливості та переваги сучасних технологій DNN

Глибокі нейронні мережі (DNN) є вдосконаленими версіями традиційних ANN, що мають кілька шарів. Останнім часом DNN стають все більш популярними завдяки своїм чудовим можливостям не лише для навчання нелінійних відображень між вхідними та вихідними даними, але й для обробки складних структур векторів вхідних даних.

Основна особливість DNN полягає в тому, що вона здатна автоматично видобувати складні та абстрактні ознаки з вхідних даних без необхідності ручного визначення цих ознак. Кожен шар DNN складається з великої кількості штучних нейронів, які обробляють вхідні сигнали і передають їх до наступного шару за допомогою вагових коефіцієнтів (Рисунок 2.3). Під час навчання DNN вагові коефіцієнти налаштовуються таким чином, щоб модель максимально точно прогнозувала вихідні дані.

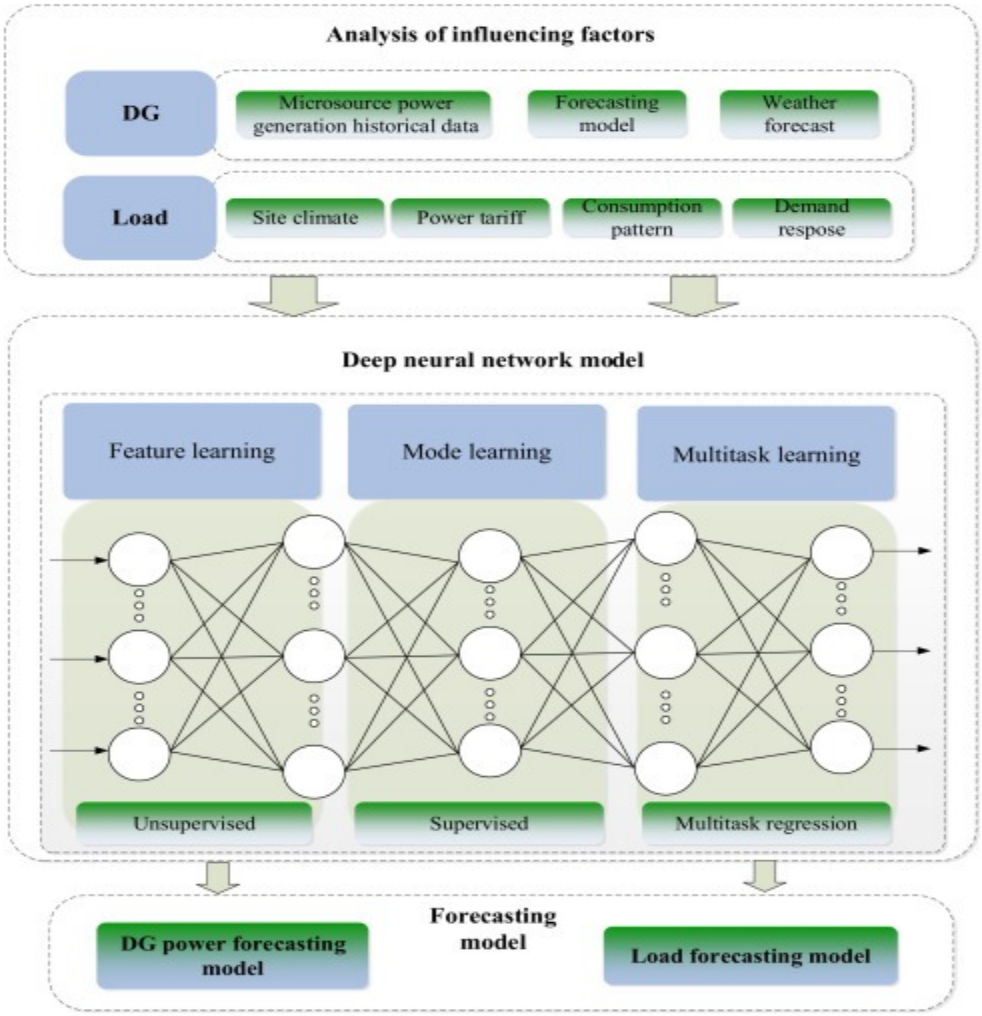



Рисунок 2.3 - Аналіз впливових факторів

Для того, щоб виконати навчання DNN для короткострокового прогнозування генерації або навантаження, необхідний навчальний набір даних, який включає вхідний вектор, що складається з таких змінних, як погода і календар, і цільовий вектор, який є виробленою потужністю (для прогнозування генерації) або навантаженням (для прогнозування навантаження).

Параметри DNN оцінюються шляхом мінімізації функції помилки суми квадратів, розрахованої на основі вихідних даних DNN. Починаючи з початкового етапу, коли параметри моделі встановлюються на початкові значення, алгоритм стохастичного градієнтного спуску виконується безперервно, щоб зменшити функцію помилки до мінімального значення. Навчання DNN складається з двох етапів: прямого і зворотного проходу на основі алгоритму зворотного поширення.

На першому етапі афінні перетворення та нелінійні активації обчислюються шар за шаром від вхідного до вихідного шару. На другому етапі похідні функції помилки за окремими вагами обчислюються у зворотному порядку від вихідного шару до вхідного.

Прогнози виробництва відновлюваної енергії враховують погодні умови та історичні дані. Ці дані, разом із звичками та моделями енергоспоживання користувачів, впливом цінової політики та моделями реагування на попит, можуть бути використані для побудови моделей прогнозування навантаження.

Згідно з теорією глибоких мереж, характеристики у верхньому шарі мережі відновлюються завдяки відкритим вірусним алгоритмам, а для налаштування мультизавданості навчання в нижньому шарі використовується регресійний шар для побудови розподілених потужностей та навантажень DNN. Моделі попереднього дня та ультракороткострокові моделі використовують неконтрольовані алгоритми багаторівневого навчання і піддаються ретельному тестуванню та аналізу помилок з використанням тестових вибірок.

### 2.6 Де використовується DNN

Глибокі нейронні мережі (DNN) використовуються в різних галузях, де потрібна обробка даних, виявлення закономірностей і вирішення складних завдань. Деякі з основних застосувань ШНМ представлені тут:

1. DNN комп'ютерного зору використовуються для обробки зображень, розпізнавання об'єктів, розпізнавання облич, сегментації зображень і розпізнавання емоцій.

Приклади:

Розпізнавання об'єктів: DNN використовуються для ідентифікації об'єктів на зображеннях і відео. Наприклад, вони можуть визначити, чи є об'єкт на зображенні автомобілем, котом або деревом. Використовується в системах безпеки, автономних транспортних засобах, системах відеоспостереження тощо.

Виявлення облич: DNN використовуються для виявлення та ідентифікації облич на зображеннях і відео; DNN можуть розпізнавати обличчя, визначати їхні

риси та виявляти емоції. Використовується в системах безпеки, системах розпізнавання облич, автоматичних фотоальбомах та інших додатках.

Сегментація зображення: DNN можна використовувати для поділу зображення на різні сегменти або області. Наприклад, об'єкти на зображенні можна ідентифікувати, призначивши клас кожному пікселю. Це важливо в медицині для сегментації органів на зображенні або в комп'ютерній графіці для створення ефектів.

Виявлення аномалій: DNN використовуються для виявлення аномалій або відхилень у зображеннях і відео. Вони можуть виявляти незвичну поведінку, несподівані об'єкти, незвичні зміни на сцені тощо. Застосовується в системах безпеки, контролі якості, медичних дослідженнях та інших сферах.

Роздільна здатність DNN використовується для відстеження об'єктів на відеозображеннях. Наприклад, можна відстежувати рух транспортних засобів, пішоходів та інших об'єктів у часі. Це використовується в автономних транспортних засобах, системах відеоспостереження, відеоіграх та багатьох інших галузях.

2. Обробка мовлення: DNN використовуються для задач, пов'язаних з обробкою мовлення, таких як розпізнавання та синтез мовлення, машинний переклад, розпізнавання мови, виявлення мовних емоцій та створення віртуальних помічників.

Приклади:

Розпізнавання мови: DNN використовуються для перетворення мови в текст; DNN можуть розпізнавати і транскрибувати записану мову в системах управління мовленням і програмах розпізнавання мови.

Машинний переклад: DNN можна навчати на великих паралельних наборах текстів, і вони можуть виконувати як лексичний, так і контекстний переклад, беручи до уваги контекст речення або всього тексту.

Емоційний аналіз: DNN застосовується для аналізу емоційного тону тексту. Позитивні, негативні або нейтральні емоції можна виявити, беручи до уваги

контекст і емоційну лексику. Це використовується, наприклад, для відгуків користувачів про товари та послуги.

Генерація тексту: DNN використовуються для автоматичної генерації тексту. Вони можуть навчатися на великих обсягах текстових даних і генерувати нові повідомлення, статті, посилання та інші типи контенту.

Класифікація тексту: DNN можна використовувати для класифікації текстових документів за різними категоріями або темами. Наприклад, він може розпізнавати спам-повідомлення, теми статей, посилання тощо. Це використовується у спам-фільтрах, пошукових системах, соціальних мережах та інших сферах.

3. Обробка природної мови: DNN використовуються для вирішення завдань опрацювання природної мови, як-от виявлення намірів, навчання моделей розуміння мови, аналіз настроїв, автоматична генерація тексту, узагальнення тексту та виявлення фейкових новин.

Приклади:

Емоційний аналіз: DNN може класифікувати текст як позитивний, негативний або нейтральний на основі контексту та емоційних характеристик. Застосовується для аналізу відгуків користувачів, соціальних мереж і громадської думки.

Генерація тексту: DNN може навчатися на великих обсягах текстових даних і генерувати нові речення, абзаци або навіть цілі тексти. Це використовується в таких сферах, як автоматичні відповіді, створення контенту і генерація текстів для рекламних та інформаційних цілей.

Виявлення лінгвістичних елементів: DNN використовуються для виявлення мовних одиниць, таких як іменники, дієслова, прикметники та фрази; DNN можуть визначати синтаксичні та семантичні особливості тексту. Це важливо для машинного перекладу, автоматизованого опрацювання мовлення та інших завдань NLP.

Виявлення іменованих об'єктів: DNN використовуються для виявлення і класифікації іменованих об'єктів у тексті, таких як люди, місця, організації, дати і географічні назви. Вони використовуються в інформаційно-пошукових системах, аналізі новинних статей, обробці документів та інших завданнях.

Машинний переклад: DNN можна навчати на паралельних текстових корпусах, вони можуть враховувати контекст і значення. Вони використовуються в системах онлайн-перекладу, локалізації програмного забезпечення та міжнародної комунікації.

4.Системи рекомендацій DNN використовуються для створення персоналізованих рекомендаційних систем у таких сферах, як електронна комерція, видавничі платформи та соціальні мережі. DNN аналізує поведінку користувачів і надає рекомендації на основі їхніх інтересів та попередніх взаємодій. Рекомендації надаються на основі інтересів користувача та його минулих взаємодій. Рекомендації надаються на основі.

Приклади:

Рекомендації в інтернет-магазинах DNN використовуються для надання персоналізованих рекомендацій щодо товарів і послуг на основі поведінки користувачів у минулому, історії покупок та інших факторів DNN аналізують великі обсяги даних про користувачів і товари, щоб передбачити індивідуальні вподобання та надати рекомендації.

Рекомендації на потокових платформах: DNN використовуються для рекомендації відео, музики та фільмів на основі історії переглядів та інших факторів. Контент може бути рекомендований користувачам на основі жанру, рейтингу, популярності та інших характеристик, щоб покращити їхній досвід перегляду відео та аудіо.

Рекомендації в соціальних мережах DNN використовуються для рекомендацій контенту, друзів, груп і сторінок на основі інтересів та взаємодії користувачів. DNN можуть надавати подальші рекомендації, аналізуючи соціальні зв'язки, минулу поведінку та інші фактори.

Рекомендації в пошукових системах: DNN використовуються для підвищення релевантності результатів пошуку, пропонуючи додатковий контент або альтернативні запити; DNN можуть аналізувати структуру тексту, контекст пошуку та інші фактори, щоб надавати рекомендації, ближчі до потреб користувача.

5. Аналіз і прогнозування даних: DNN використовуються для аналізу, прогнозування та класифікації великих обсягів даних. Вони використовуються в тих сферах, де прогнози повинні базуватися на історичних даних, таких як фінансові ринки, охорона здоров'я, промисловість і маркетинг.

Приклади:

Прогнозування у фінансах:

Прогнозування цін акцій: DNN дозволяють виявляти приховані взаємозв'язки між ринковими індикаторами, аналізувати новини, коментарі в соціальних мережах, історичні дані, щоб передбачити зміну цін акцій та волатильність ринків.

Управління портфелем: DNN оптимізують вибір інвестицій, визначаючи найбільш перспективні активи на основі фінансової історії та макроекономічних показників.

Виявлення шахрайства: Використання DNN для аналізу транзакцій у реальному часі допомагає виявляти підозрілі операції, що є ключовим у фінансовій безпеці.

Прогнозування в маркетингу:

Персоналізація рекламних кампаній: Аналізуючи дані про вподобання клієнтів, їх поведінку у мережі та історію покупок, DNN допомагають створювати таргетовані маркетингові кампанії.

Прогнозування утримання клієнтів: DNN аналізують ризик втрати клієнтів, визначаючи чинники, які можуть вплинути на лояльність, і пропонують рішення для їх утримання.

Сегментація клієнтів: Глибокі моделі допомагають групувати клієнтів за поведінковими паттернами, що покращує ефективність продажів та збільшує дохід.

Прогнозування в медицині:

Діагностика захворювань: DNN успішно застосовуються для аналізу медичних зображень (КТ, МРТ), даних ЕКГ та гістологічних знімків для виявлення патологій, таких як рак, серцеві захворювання та інші.

Персоналізована медицина: Аналізуючи генетичні дані та історію захворювань, DNN пропонують персоналізовані плани лікування для пацієнтів.

Моніторинг здоров'я: DNN обробляють дані з носимих пристроїв (розумних годинників, фітнес-браслетів) для прогнозування змін у здоров'ї, таких як аритмія чи стрибки рівня цукру в крові.

У транспортному секторі прогнозні DNN використовуються для прогнозування обсягів трафіку, транспортних потоків, часу прибуття та інших важливих аспектів транспортної системи. Аналізуючи дані про інтенсивність руху, маршрути та погодні умови, можна прогнозувати та оптимізувати транспортні потоки.

Прогнозування в метеорології: DNN використовуються для прогнозування погодних умов, температури, опадів, вітру та інших параметрів. Прогнози можна робити, аналізуючи метеорологічні дані, супутникові знімки та мінливість клімату, щоб підвищити точність прогнозів погоди.

Це лише кілька прикладів, але існує багато інших галузей, які використовують DNN, і їхня сила полягає в здатності виявляти складні залежності в даних і вчитися на великих обсягах інформації.

# 3 ТЕХНОЛОГІЇ COMPUTER VISION

Computer Vision (CV), або Комп'ютерне зорове сприйняття, є галуззю штучного інтелекту та комп'ютерних наук, яка займається автоматизацією процесу розпізнавання, аналізу та інтерпретації візуальної інформації, отриманої з зображень або відео. Задача CV полягає в тому, щоб дозволити комп'ютерам і системам отримувати, обробляти і зрозуміти зображення та відео так, як це робить людське око (Рисунок 3.1)

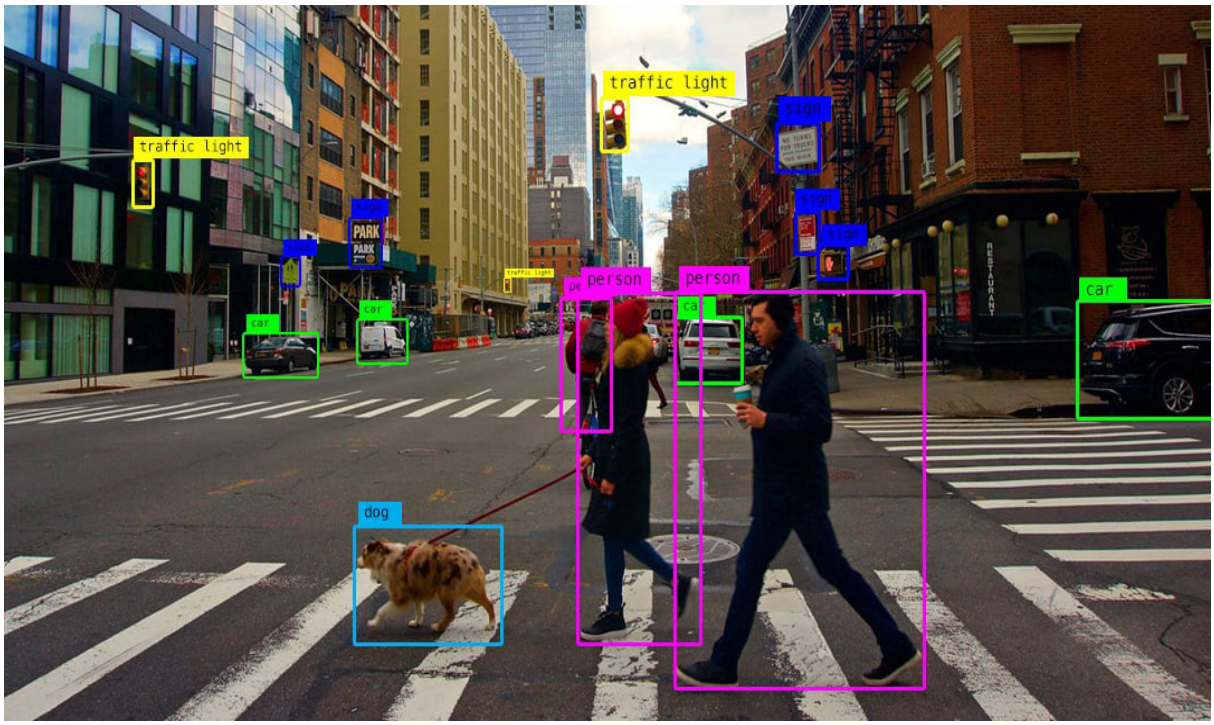



Рисунок 3.1 - Computer Vision на практиці

Основні завдання комп'ютерного зорового сприйняття включають:

## 3.1 Розпізнавання об'єктів

Це одна з ключових задач комп'ютерного зорового сприйняття, яка включає виявлення та класифікацію об'єктів у зображеннях або відео. Це дозволяє системам

і програмам ідентифікувати різні об'єкти на зображеннях, таких як люди, автомобілі, тварини, будівлі та багато інших.

Процес розпізнавання об'єктів зазвичай складається з кількох етапів:

1. Виявлення об'єктів: на цьому етапі система визначає місця на зображенні або відео, де знаходяться об'єкти. Це може бути виконано за допомогою різних алгоритмів, таких як регіональні нейронні мережі (R-CNN), YOLO (You Only Look Once) або SSD (Single Shot Multibox Detector). Виявлення об'єктів може включати пошук прямих або складних форм, таких як обличчя, автомобілі або інші об'єкти, які потрібно знайти на зображенні.

1. Класифікація об'єктів: після того, як об'єкти виявлено, система класифікує кожен з них, визначаючи, до якого класу він належить. Наприклад, після виявлення обличчя система може визначити, чи належить воно конкретній особі, чи це просто загальне людське обличчя. Для класифікації часто використовуються глибокі нейронні мережі (CNN), які тренуються на великій кількості зображень з мітками.

3. Визначення меж об'єкта: на додаток до класифікації, система може визначити точні межі об'єкта, малюючи рамку або контур навколо нього.

Розпізнавання об'єктів активно використовується в багатьох галузях:

Безпека: для виявлення підозрілих об'єктів або осіб у відеоспостереженні.

Автономні транспортні засоби: для виявлення та класифікації об'єктів на дорозі, таких як інші автомобілі, пішоходи або знаки.

Медицина: для автоматичного аналізу медичних зображень, таких як рентгенівські знімки або МРТ, де система виявляє і класифікує різні види тканин або патологій.

Робототехніка: для навігації роботів, розпізнавання об'єктів у навколишньому середовищі.

Методи для розпізнавання об'єктів постійно удосконалюються за рахунок розвитку технологій глибокого навчання та доступу до великих наборів даних для тренування моделей.

### 3.2 Обробка зображень

Це процес маніпуляції та аналізу зображень з метою покращення їх якості або корекції деяких характеристик. Це одна з основних задач комп'ютерного зорового сприйняття та широко використовується у багатьох галузях, таких як медицина, мистецтво, безпека та наукові дослідження.

Основні техніки обробки зображень включають:

1 Фільтрація зображень: застосування фільтрів для видалення шумів, розмиття або покращення певних деталей зображення. Наприклад:

Фільтри для згладжування (наприклад, гаусове розмивання) використовуються для зменшення шумів на зображенні.

Фільтри для підсилення контурів (наприклад, оператор Собеля, Кенні) використовуються для виділення важливих деталей, таких як контури об'єктів на зображенні.

2 Корекція контрасту: ця техніка дозволяє змінювати яскравість і контрастність зображення для покращення видимості деталей, особливо у випадку недостатньо освітлених або занадто темних зображень. Популярні методи:

3 Гістограмне вирівнювання — дозволяє покращити контраст зображення шляхом перерозподілу інтенсивності пікселів.

Лінійна або нелінійна корекція — зміна контрасту в певних діапазонах яскравості.

4 Масштабування та вирізання: зміна розміру зображення (масштабування) або вибір конкретної частини зображення (вирізання) для подальшої обробки або для фокусування на певних деталях.

5 Покращення різкості: процес підвищення різкості зображення для виділення дрібних деталей. Це часто використовують для зображень, які виглядають розмитими або нечіткими.

6 Видалення шуму: за допомогою різних фільтрів можна усувати або зменшувати вплив шуму (небажаних спотворень), які можуть виникнути під час

зйомки або обробки зображення. Наприклад, використання медіанних фільтрів або гаусових фільтрів для видалення "пікселів, що псують картину".

7 Трансформації зображень: застосування математичних операцій для змінення геометрії зображення, наприклад, горизонтальні або вертикальні обертання, перспективні трансформації, масштабування, або віддзеркалення.

8 Сегментація зображень: розбиття зображення на окремі частини або об'єкти для подальшого аналізу, що дозволяє виділити важливі елементи. Це особливо корисно в таких галузях, як медична діагностика або відеоспостереження.

Обробка зображень знаходить застосування в багатьох сферах:

Медицина: для покращення якості медичних зображень (рентгенів, МРТ) з метою кращого розпізнавання патологій.

Мистецтво: для покращення фотографій, реставрації старих зображень.

Відеоспостереження: для аналізу та покращення відео для виявлення об'єктів або осіб.

Наукові дослідження: для обробки зображень із телескопів, мікроскопів та інших приладів для отримання чіткішого зображення та аналізу.

Технології обробки зображень активно використовують методи машинного навчання, включаючи глибоке навчання, для покращення та автоматизації багатьох із цих процесів.

### 3.3 Визначення меж об'єкта

Це процес, під час якого система не лише класифікує об'єкти на зображеннях чи відео, а й визначає точні межі цих об'єктів, часто візуалізуючи їх за допомогою рамок або контурів. Ця задача є важливою складовою частиною багатьох алгоритмів комп'ютерного зорового сприйняття, оскільки дозволяє точно локалізувати об'єкти та виділяти їх на зображенні для подальшого аналізу.

Основні етапи та методи визначення меж об'єкта:

1 Виявлення кордонів (границь) об'єкта: Використовуються різноманітні алгоритми, які дозволяють знайти межі між об'єктами на зображенні. Це може бути зроблено через детекцію контурів або визначення області, що відповідає певному

об'єкту. Наприклад, методи такі як Canny edge detection або Sobel operator дозволяють виявляти різку зміну пікселів, що допомагає знайти межі об'єкта.

2 Визначення прямокутних рамок (bounding boxes): Визначення прямокутних рамок або "bounding boxes" є одним із найбільш популярних способів позначення меж об'єктів на зображенні. Це включає в себе встановлення координат прямокутника, що обрамляє об'єкт. У комп'ютерному зорі та глибокому навчанні використовуються різні методи для виявлення таких рамок, наприклад, YOLO (You Only Look Once) або R-CNN (Region Convolutional Neural Networks).

3 Контурна детекція: Крім прямокутних рамок, для точного визначення меж об'єктів може використовуватися техніка виявлення контурів. За допомогою таких методів можна малювати більш складні форми, які точніше відповідають контуру об'єкта. Це може включати щільну сегментацію або використання методів active contour models (наприклад, Snakes) для більш гнучкого та точного визначення меж.

4 Перспективні трансформації та інші геометричні методи: Окрім визначення меж об'єктів на звичайних зображеннях, іноді потрібно враховувати геометричні зміни, такі як перспективи, обертання, нахил або деформація об'єкта. Трансформації зображень або використання геометричних фігур може допомогти в більш точному відображенні меж об'єктів.

Застосування визначення меж об'єктів:

Автономні транспортні засоби: для визначення точних меж інших транспортних засобів, пішоходів або знаків.

Медицина: для точного виділення органів або патологічних утворень на медичних зображеннях (наприклад, в рентгені, МРТ).

Робототехніка: для розпізнавання та маніпуляції об'єктами у середовищі, наприклад, для збору предметів.

Безпека: для моніторингу за об'єктами на відеозаписах, де важливо чітко визначити межі людей або предметів.

Точне визначення меж об'єктів є важливим етапом у багатьох сферах, від медичних зображень до автономних транспортних засобів, і має значний вплив на

ефективність подальшого аналізу зображень та прийняття рішень на основі цього аналізу.

Обробка зображень знаходить застосування в багатьох сферах:

Медицина: для покращення якості медичних зображень (рентгенів, МРТ) з метою кращого розпізнавання патологій.

Мистецтво: для покращення фотографій, реставрації старих зображень.

Відеоспостереження: для аналізу та покращення відео для виявлення об'єктів або осіб.

Наукові дослідження: для обробки зображень із телескопів, мікроскопів та інших приладів для отримання чіткішого зображення та аналізу.

Прогнозування в енергетиці:

Прогнозування попиту на енергію: DNN аналізують історичні дані про споживання енергії, щоб передбачити майбутній попит.

Оптимізація використання відновлюваних джерел: DNN передбачають продуктивність вітрових і сонячних електростанцій на основі погодних умов.

Виявлення аномалій: Глибокі мережі можуть визначати витоки енергії або несправності в мережі.

Технології обробки зображень активно використовують методи машинного навчання, включаючи глибоке навчання, для покращення та автоматизації багатьох із цих процесів.

# 4 ПРОЦЕС РОЗПІЗНАВАННЯ МОВИ ЖЕСТІВ

## 4.1 Збір зображень

Цей фрагмент коду імпортує бібліотеки os для роботи з файловою системою та cv2 для обробки зображень. Він визначає шлях до директорії ./data в змінній DATA_DIR, перевіряє, чи існує ця директорія, і, якщо вона відсутня, створює її. Також оголошуються змінні для кількості класів (number_of_classes = 23) і розміру набору даних (dataset_size = 400) (Рисунок 4.1).

```
collect_imgs.py > ...
1  import os
2  import cv2
3
4
5  DATA_DIR = './data'
6  if not os.path.exists(DATA_DIR):
7      os.makedirs(DATA_DIR)
8
9  number_of_classes = 23
10 dataset_size = 400
```

Рисунок 4.1 - Створення директорію для збереження даних і налаштування параметрів для набору даних.

Цей фрагмент коду ініціалізує відеопотік з камери за допомогою cv2.VideoCapture(0) і за допомогою циклу перевіряє наявність директорії для кожного класу. Якщо відповідна директорія не існує, вона створюється. Також виводиться повідомлення, що починається збір даних для поточного класу. Цей код використовується для збирання даних для кожного класу і збереження їх у відповідні папки (Рисунок 4.2).

```
11
12  cap = cv2.VideoCapture(0)
13  for j in range(number_of_classes):
14      if not os.path.exists(os.path.join(DATA_DIR, str(j))):
15          os.makedirs(os.path.join(DATA_DIR, str(j)))
16
17      print('Collecting data for class {}'.format(j))
18
```

Рисунок 4.2 - Ініціалізація відеопотоку та створення директорій для збору даних за допомогою OpenCV.

Цей фрагмент коду ініціалізує змінну done = False, яка, ймовірно, використовується для контролю циклу. Відкривається безкінечний цикл while True, в якому зчитується кадр з відеопотоку за допомогою cap.read(), і результат зберігається в змінній frame. За допомогою cv2.putText() на кадрі додається текст "Ready? Press 'Q'! :)", який відображається на координатах (100, 50) із шрифтом cv2.FONT_HERSHEY_SIMPLEX, розміром 1.3 і кольором (0, 255, 0). Кадр показується на екрані через cv2.imshow(). В циклі перевіряється натискання клавіші. Якщо натискається клавіша "Q" (перевіряється через cv2.waitKey(25)), то цикл припиняється за допомогою break. Цей код використовує OpenCV для відображення відео та тексту на екрані, а також для зупинки відеопотоку при натисканні клавіші "Q"(Рисунок 4.3).

```
18
19  done = False
20  while True:
21      ret, frame = cap.read()
22      cv2.putText(frame, 'Ready? Press "Q" ! :)', (100, 50), cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 255,
23                  cv2.LINE_AA)
24      cv2.imshow('frame', frame)
25      if cv2.waitKey(25) == ord('q'):
26          break
27
```

Рисунок 4.3 - Фрагмент коду відображає відеопотік з текстом "Ready? Press 'Q'! :)" на екрані та зупиняє відео при натисканні клавіші "Q".

Цей фрагмент коду зчитує відеопотік та зберігає кожен кадр як зображення у форматі .jpg. Спочатку ініціалізується лічильник counter = 0, який використовується для нумерації зображень.

Далі, у циклі, поки лічильник не досягне розміру набору даних (dataset_size), програма:

Зчитує кадр з відеопотоку за допомогою cap.read().

Виводить зчитаний кадр на екран за допомогою cv2.imshow().

Чекає натискання клавіші через cv2.waitKey(25), де 25 — це затримка в мілісекундах, що дозволяє користувачеві бачити кадри.

Після цього кадр зберігається на диск у форматі .jpg, використовуючи функцію cv2.imwrite(), причому кожному зображенню дається унікальне ім'я, що складається з поточного значення лічильника (наприклад, 0.jpg, 1.jpg тощо).

Лічильник збільшується на 1 після кожного збереженого зображення. Цей код зазвичай використовується для збору зображень з відео для створення наборів даних, наприклад, для тренування моделей машинного навчання, де кожен кадр зберігається в окремому файлі (Рисунок 4.4).

```
28      counter = 0
29      while counter < dataset_size:
30          ret, frame = cap.read()
31          cv2.imshow('frame', frame)
32          cv2.waitKey(25)
33          cv2.imwrite(os.path.join(DATA_DIR, str(j), '{}.jpg'.format(counter)), frame)
34
35          counter += 1
36
```

Рисунок 4.4 - Цей код зчитує кадри з відеопотоку і зберігає їх як зображення у форматі .jpg, поки не буде досягнуто заданої кількості зображень.

Цей фрагмент коду виконує завершення роботи з відеопотоком та закриває всі вікна OpenCV:

cap.release() звільняє ресурс відеопотоку, припиняючи його захоплення та закриваючи камеру.

cv2.destroyAllWindows() закриває всі відкриті вікна, що були створені через OpenCV, звільняючи ресурси(Рисунок 4.5).

```
37    cap.release()
38    cv2.destroyAllWindows()
```

Рисунок 4.5 - Фрагмент коду звільняє ресурс відеопотоку та закриває всі відкриті вікна OpenCV

Початок створення датасету для навчання моделі та перевірка фіксації данних для подальшого використання (Рисунок 4.6).

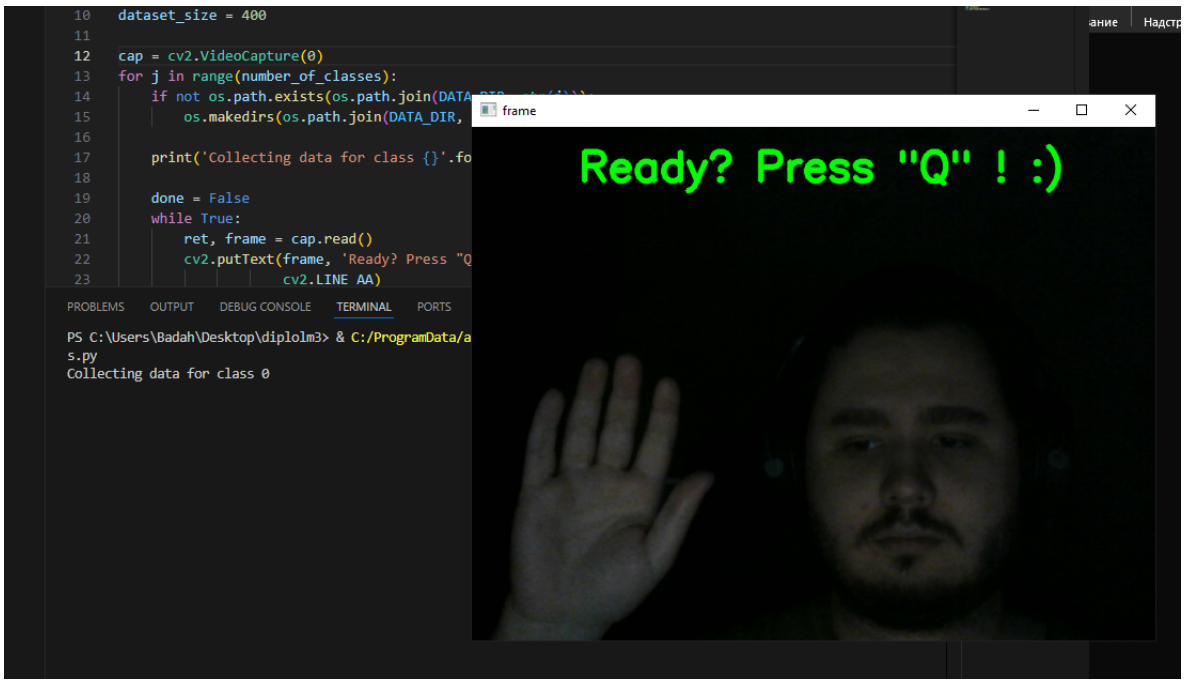



Рисунок 4.6 – Перевірка Collect images

Зафіксована дата та збережена інформація є ключовими для подальшого використання в процесі аналізу й навчання моделі. Збережені дані представляють собою структурований набір, що може слугувати вхідними даними для створення або вдосконалення моделей машинного навчання.

Цей процес включає підготовку та організацію інформації для подальшого використання, зокрема для тренування нейронних мереж або алгоритмів глибокого навчання. Збереження даних забезпечує їх доступність, цілісність та готовність до аналізу, що є критично важливим етапом у побудові ефективних моделей. У результаті дані стають основою для прогнозування, класифікації або інших завдань, пов'язаних із обробкою інформації (Рисунок 4.7).

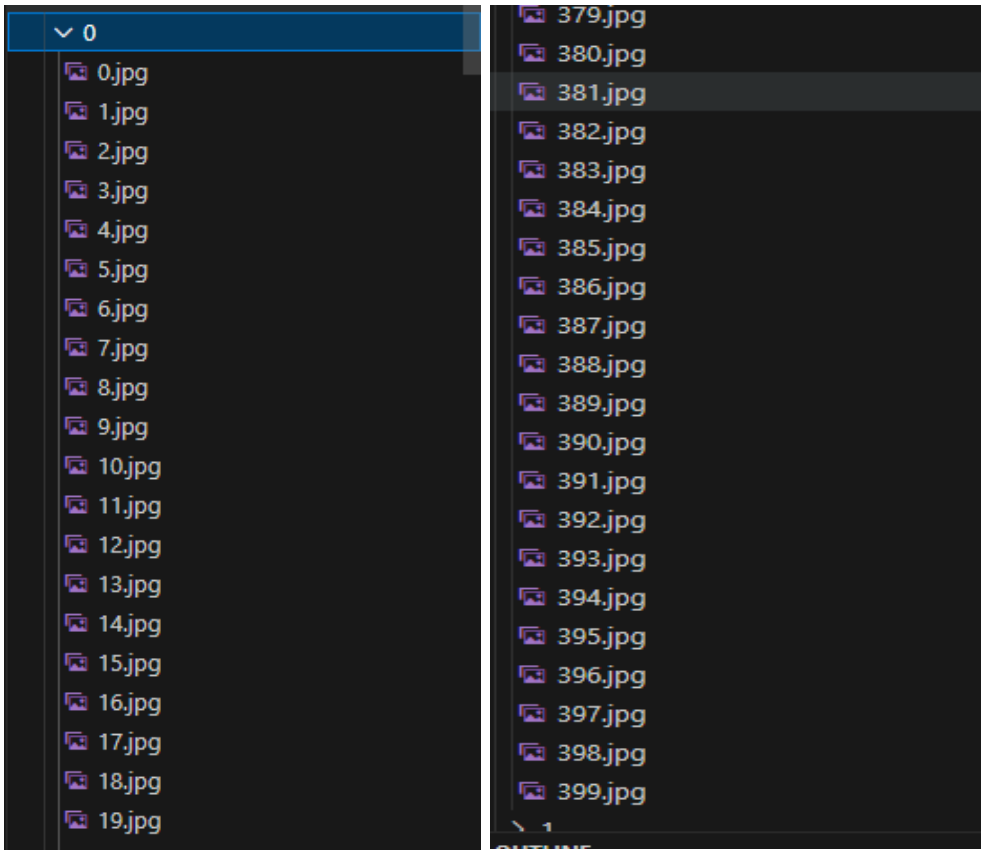



Рисунок 4.7 - Приклад збережених зображень у папці для навчання моделі.

### 4.2 Створення набору даних (dataset)

Цей фрагмент коду імпортує кілька бібліотек, необхідних для обробки даних. Бібліотека os використовується для роботи з операційною системою, наприклад, для роботи з файловою системою. pickle застосовується для серіалізації та десеріалізації даних, що дозволяє зберігати та відновлювати Python об'єкти. Бібліотека mediapipe імпортується як mp і зазвичай використовується для комп'ютерного зору та обробки зображень. cv2 (OpenCV) використовується для роботи з зображеннями та відео, а matplotlib.pyplot імпортується як plt для візуалізації графіків і діаграм. Ці бібліотеки, ймовірно, використовуються для створення набору даних, обробки зображень та візуалізації результатів (Рисунок 4.8).

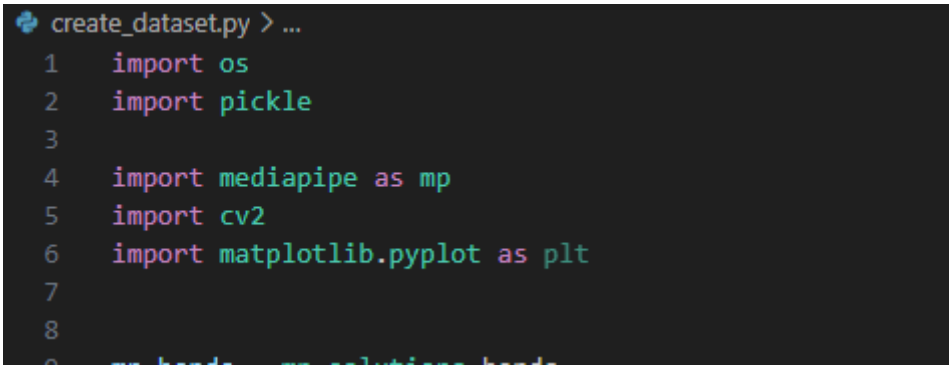



Рисунок 4.8 - Імпорт необхідних бібліотек для створення набору даних.

Цей фрагмент коду ініціалізує модулі для роботи з бібліотекою MediaPipe. mp_hands відповідає за розпізнавання та обробку рук, надаючи функціональність для виявлення та аналізу рук на зображеннях чи відео. mp_drawing містить утиліти для малювання на зображеннях, що дозволяє зображати результати, наприклад, ключові точки чи контури. mp_drawing_styles використовується для налаштування стилів малювання, наприклад, кольору або товщини ліній для візуалізації на зображеннях. Це налаштування дозволяє інтегрувати зручні інструменти для обробки та візуалізації рук в задачах комп'ютерного зору (Рисунок 4.9).

```
9   mp_hands = mp.solutions.hands
10  mp_drawing = mp.solutions.drawing_utils
11  mp_drawing_styles = mp.solutions.drawing_styles
```

Рисунок 4.9 - Ініціалізація модулів MediaPipe для роботи з розпізнаванням рук та малюванням.

Цей фрагмент коду ініціалізує об'єкт для розпізнавання рук за допомогою MediaPipe. В методі mp_hands.Hands() встановлюється два параметри: static_image_mode=True, що вказує на обробку статичних зображень, і min_detection_confidence=0.3, що визначає мінімальний поріг довіри для виявлення рук. Це дозволяє ефективно визначати руки на статичних зображеннях з певною точністю. Окрім цього, задається шлях до директорії для збереження даних — DATA_DIR = './data' (Рисунок 4.10).

```
13  hands = mp_hands.Hands(static_image_mode=True, min_detection_confidence=0.3)
14
15  DATA_DIR = './data'
```

Рисунок 4.10 - Ініціалізація об'єкта для розпізнавання рук та налаштування параметрів для обробки статичних зображень.

Цей фрагмент коду використовується для завантаження зображень з директорії та їх підготовки для подальшого використання. Спочатку створюються порожні списки data та labels для зберігання зображень та їх міток. Далі за допомогою двох вкладених циклів обробляються всі файли в підкаталогах директорії DATA_DIR. Для кожного зображення виконується його завантаження за допомогою cv2.imread(), після чого воно конвертується з формату BGR в RGB за

допомогою cv2.cvtColor(). Ці зображення потім будуть додані в список data.(Рисунок 4.11).

```
data = []
labels = []
for dir_ in os.listdir(DATA_DIR):
    for img_path in os.listdir(os.path.join(DATA_DIR, dir_)):
        data_aux = []

        x_ = []
        y_ = []

        img = cv2.imread(os.path.join(DATA_DIR, dir_, img_path))
        img_rgb = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)

```

Рисунок 4.11 - Завантаження та обробка зображень з директорії для подальшої обробки.

Цей фрагмент коду обробляє результати, отримані після розпізнавання рук за допомогою MediaPipe. Спочатку обробляється зображення для виявлення точок на руках за допомогою hands.process(img_rgb). Якщо на зображенні знайдені руки (results.multi_hand_landmarks), то для кожної руки в списку точок обчислюються координати x та y для кожної з точок. Ці координати додаються до списків x_ та y_. Далі ці координати нормалізуються (віднімається мінімальне значення координат) та зберігаються в data_aux. Потім ці дані додаються до основного списку data, а відповідна мітка для кожного зображення (вказана в dir_) додається до списку labels. Таким чином, створюється набір даних для подальшого навчання або аналізу (Рисунок 4.12).

```
28
29          results = hands.process(img_rgb)
30          if results.multi_hand_landmarks:
31              for hand_landmarks in results.multi_hand_landmarks:
32                  for i in range(len(hand_landmarks.landmark)):
33                      x = hand_landmarks.landmark[i].x
34                      y = hand_landmarks.landmark[i].y
35
36                      x_.append(x)
37                      y_.append(y)
38
39                  for i in range(len(hand_landmarks.landmark)):
40                      x = hand_landmarks.landmark[i].x
41                      y = hand_landmarks.landmark[i].y
42                      data_aux.append(x - min(x_))
43                      data_aux.append(y - min(y_))
44
45              data.append(data_aux)
46              labels.append(dir_)
```

Рисунок 4.12 - Обробка результатів розпізнавання рук і збереження координат для створення набору даних.

Цей фрагмент коду зберігає дані та мітки у файл за допомогою модуля pickle. Спочатку відкривається файл data.pickle в режимі запису ('wb'). Далі функція pickle.dump() серіалізує словник, що містить два ключі: data та labels, і записує його в файл. Після цього файл закривається за допомогою f.close(), завершуючи операцію запису. Цей процес дозволяє зберігати дані та мітки для подальшого використання (Рисунок 4.13).

```
48  f = open('data.pickle', 'wb')
49  pickle.dump({'data': data, 'labels': labels}, f)
50  f.close()
```

Рисунок 4.13 – Збереження даних і міток у файл за допомогою модуля pickle.

Це виведення з консолі, яке містить кілька попереджень і інформаційних повідомлень під час запуску скрипту.

Перше повідомлення вказує на використання custom операцій у TensorFlow Lite, що може призвести до незначних відмінностей в числових результатах через похибки в обчисленнях. Наступні попередження пов'язані з вимкненням підтримки feedback tensors через використання моделі з одним підписом інференції. Останнє повідомлення вказує на проблему з визначенням ROI (Region of Interest) у моделі, де потрібно вказати або "IMAGE_DIMENSIONS", або використати матрицю проекції "PROJECTION_MATRIX" для коректної обробки прямокутних ROI (Рисунок 4.14).

```
PS C:\Users\Badah\Desktop\diplolm3> & C:/ProgramData/anaconda3/python.exe c:/Users/Badah/Deskto
p/diplolm3/create_dataset.py
2024-12-15 23:59:50.558454: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on
. You may see slightly different numerical results due to floating-point round-off errors from
different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_
OPTS=0`.
2024-12-15 23:59:51.158135: I tensorflow/core/util/port.cc:153] oneDNN custom operations are on
. You may see slightly different numerical results due to floating-point round-off errors from
different computation orders. To turn them off, set the environment variable `TF_ENABLE_ONEDNN_
OPTS=0`.
INFO: Created TensorFlow Lite XNNPACK delegate for CPU.
WARNING: All log messages before absl::InitializeLog() is called are written to STDERR
W0000 00:00:1734299993.086973   14540 inference_feedback_manager.cc:114] Feedback manager requi
res a model with a single signature inference. Disabling support for feedback tensors.
W0000 00:00:1734299993.101604   11748 inference_feedback_manager.cc:114] Feedback manager requi
res a model with a single signature inference. Disabling support for feedback tensors.
W0000 00:00:1734299993.108125   15348 landmark_projection_calculator.cc:186] Using NORM_RECT wi
thout IMAGE_DIMENSIONS is only supported for the square ROI. Provide IMAGE_DIMENSIONS or use PR
OJECTION_MATRIX.
PS C:\Users\Badah\Desktop\diplolm3>
```

Рисунок 4.14 - Виведення консолі під час запуску скрипту з TensorFlow Lite.

### 4.3 Навчання класифікатора

Цей фрагмент коду імпортує необхідні бібліотеки для тренування класифікатора. Використовується pickle для серіалізації та десеріалізації моделей. Бібліотека RandomForestClassifier з sklearn.ensemble надає інструменти для створення класифікатора на основі випадкового лісу. train_test_split з sklearn.model_selection застосовується для розподілу даних на тренувальний та тестовий набори. Метрики для оцінки моделі, такі як accuracy_score, roc_curve, і auc імпортуються з sklearn.metrics. Для обробки міток у багатокласових задачах використовується label_binarize з sklearn.preprocessing. Для роботи з масивами даних застосовується numpy, а для візуалізації результатів — matplotlib.pyplot (Рисунок 4.15).

train_classifier.py > ...

```
1  import pickle
2  from sklearn.ensemble import RandomForestClassifier
3  from sklearn.model_selection import train_test_split
4  from sklearn.metrics import accuracy_score, roc_curve, auc
5  from sklearn.preprocessing import label_binarize
6  import numpy as np
7  import matplotlib.pyplot as plt
8
```

Рисунок 4.15 - Імпорт необхідних бібліотек для тренування класифікатора.

Цей фрагмент коду завантажує збережені дані з файлу за допомогою модуля pickle. Файл data.pickle відкривається в режимі читання бінарних даних ('rb'). Далі, за допомогою функції pickle.load(), дані з файлу завантажуються в змінну data_dict. З цієї змінної витягуються дані та мітки, які зберігаються в окремих змінних data і labels. Для цього використовується np.asarray(), щоб перетворити дані у масиви NumPy. Це дозволяє зручно обробляти і використовувати дані для подальших операцій, таких як тренування моделі. (Рисунок 4.16).

```
9  data_dict = pickle.load(open('./data.pickle', 'rb'))
10
11 data = np.asarray(data_dict['data'])
12 labels = np.asarray(data_dict['labels'])
```

Рисунок 4.16 - Завантаження збережених даних та міток з файлу за допомогою pickle.

Цей фрагмент коду виконує розподіл даних на тренувальний і тестовий набори за допомогою функції train_test_split() з бібліотеки sklearn. Дані та мітки передаються в цю функцію, де 20% даних будуть використані для тестування, а 80% — для тренування. Також передбачається, що дані будуть перемішані перед розподілом, і розподіл буде збалансований відповідно до міток.

Далі ініціалізується модель класифікатора випадкового лісу RandomForestClassifier(), яка тренується на тренувальних даних x_train та y_train. Після тренування модель використовує тестові дані x_test для прогнозування результатів, які зберігаються в змінній y_predict. Цей процес є частиною стандартного тренування та тестування моделі машинного навчання (Рисунок 4.17).

```
13
14  x_train, x_test, y_train, y_test = train_test_split(data, labels, test_size=0.2, shuffle=True, stratify=labels)
15
16  model = RandomForestClassifier()
17
18  model.fit(x_train, y_train)
19
20  y_predict = model.predict(x_test)
```

Рисунок 4.17 - Розподіл даних на тренувальний та тестовий набори та тренування моделі випадкового лісу.

Цей фрагмент коду обчислює точність класифікації моделі, порівнюючи передбачені значення (y_predict) з реальними значеннями тестової вибірки (y_test)

за допомогою функції accuracy_score. Після цього виводиться повідомлення, що показує відсоток правильно класифікованих зразків.

Замість використання простого виклику open() для відкриття файлу, у цьому коді застосовується контекстний менеджер with, що дозволяє автоматично закривати файл після завершення роботи з ним. Файл model.p відкривається для запису в бінарному форматі ('wb'), і модель зберігається в цей файл за допомогою pickle.dump().

Цей підхід забезпечує коректне і безпечне збереження моделі для подальшого використання (Рисунок 4.18)

```
26  score = accuracy_score(y_predict, y_test)
27  print('{}% of samples were classified correctly!'.format(score * 100))
28
29
30  with open('model.p', 'wb') as f:
31      pickle.dump({'model': model}, f)
```

Рисунок 4.18 - Оцінка точності моделі та збереження моделі в файл.

Цей фрагмент коду будує графік ROC-кривої для кожного класу в багатокласовій класифікації. Спочатку мітки тестового набору перетворюються на бінарний формат за допомогою функції label_binarize(), щоб кожен клас мав свою бінарну мітку. Далі модель передбачає ймовірності для кожного класу через model.predict_proba().

Для кожного класу обчислюються значення FPR (помилковий позитивний рівень) та TPR (істинний позитивний рівень) за допомогою функції roc_curve(). Потім для кожної кривої обчислюється значення AUC (площа під кривою) через auc(). Графік для кожного класу будується за допомогою plt.plot(), де відображається і сам графік, і значення AUC (Рисунок 4.19)

```
y_test_binarized = label_binarize(y_test, classes=classes)
y_proba = model.predict_proba(x_test)


plt.figure(figsize=(8, 6))
for i, class_label in enumerate(classes):
    fpr, tpr, _ = roc_curve(y_test_binarized[:, i], y_proba[:, i])
    roc_auc = auc(fpr, tpr)
    plt.plot(fpr, tpr, lw=2, label=f'Class {class_label} (area = {roc_auc:.2f})')
```

Рисунок 4.19 Побудова ROC-кривої для багатокласової класифікації та обчислення AUC.

Цей фрагмент коду додає елементи до графіку ROC-кривої. Спочатку додається лінія для випадкового вгадування, яка має координати (0, 1) для FPR і (0, 1) для TPR. Вона буде відображена сірим кольором і з пунктирним стилем лінії. Далі встановлюються межі для осей X і Y: ось X має діапазон від 0 до 1, а ось Y від 0 до 1. На графіку додаються підписи для осей та заголовок, що вказує на те, що це ROC-крива для багатокласової класифікації. Також вказано розташування легенди в нижньому правому куті графіка, і включено сітку для зручності читання графіка. (Рисунок 4.20)

```
plt.plot([0, 1], [0, 1], color='gray', lw=2, linestyle='--', label='Random guess')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic - Multiclass')
plt.legend(loc="lower right")
plt.grid()
plt.show()
```

Рисунок 4.20 - Налаштування графіку ROC-кривої та додавання елементів на діаграму.

Функція plot_results є важливим інструментом аналізу процесу навчання моделі. Вона приймає як вхідні дані історію навчання моделі (результат history.history), що містить інформацію про зміни втрат та метрик на тренувальній і валідаційній вибірках для кожної епохи. Ця функція генерує два ключові графіки, які допомагають детально оцінити ефективність моделі та ідентифікувати можливі проблеми в процесі навчання.

Графік, "Receiver Operating Characteristic", відображає значення втрат (loss) для тренувальної та валідаційної вибірки протягом усіх епох навчання. Значення втрат — це один із найважливіших показників якості роботи моделі, який демонструє, наскільки добре модель прогнозує результати на основі тренувальних даних. На цьому графіку ми спостерігаємо дві криві: одна відображає втрати на тренувальній вибірці (Train loss), а інша — на валідаційній (Validation loss). Ідеальний результат навчання моделі полягає в тому, що втрати зменшуються з кожною епохою, причому як на тренувальній, так і на валідаційній вибірках. Якщо втрати на валідаційній вибірці починають збільшуватися, це може свідчити про перенавчання моделі, коли вона стає надто пристосованою до тренувальних даних і втрачає здатність до генералізації.

Графік, створенний за допомогою plot_results, дають змогу виявити важливі деталі. Наприклад, якщо тренувальні втрати стабільно знижуються, але валідаційні втрати залишаються на одному рівні або навіть збільшуються, це може вказувати на те, що модель перенавчається. У такому випадку можна вжити заходів, таких як додавання регуляризації, збільшення кількості даних для навчання або використання інших методів запобігання перенавчанню, наприклад, ранньої зупинки (early stopping).

Крім того, графіки дозволяють оцінити ефективність навчання моделі у часовому контексті. Наприклад, якщо втрата зменшується надто повільно, це може свідчити про необхідність налаштування швидкості навчання (learning rate). З іншого боку, якщо втрати змінюються дуже швидко, це може свідчити про нестабільність процесу оптимізації, що вимагає зменшення швидкості навчання.

Функція plot_results(history.history) є невіддільною частиною аналізу ефективності навчання моделі. Вона дозволяє візуально оцінити динаміку навчання, знаходити потенційні проблеми та приймати обґрунтовані рішення щодо подальшого налаштування гіперпараметрів моделі, зміни архітектури або навіть коригування набору даних. Завдяки таким візуалізаціям можна забезпечити оптимальний процес навчання і досягти кращих результатів у прогнозуванні або класифікації (Рисунок 4.21).

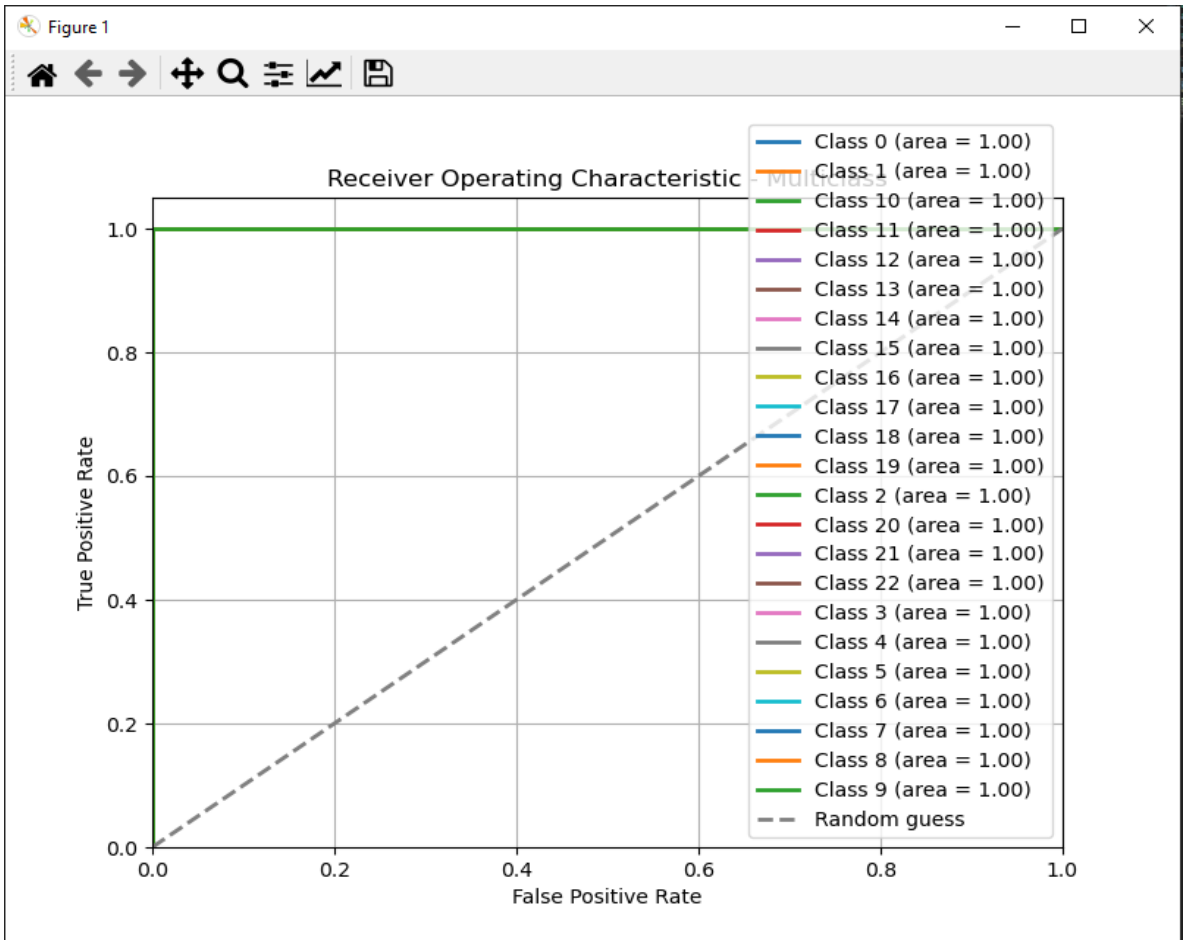



Рисунок 4.21 –ROC-криві для багатокласової класифікації з AUC для кожного класу.

Це виведення з консолі, яке показує результат виконання скрипту train_classifier.py. Виведене повідомлення вказує, що 100% зразків були

класифіковані правильно, що свідчить про високий рівень точності моделі після її тренування (Рисунок 4.22).


Рисунок 4.22 - Виведення результатів класифікації з точною оцінкою в консолі.

### 4.4 Тестування класифікатора

Цей фрагмент коду імпортує бібліотеки, необхідні для виконання інференції (застосування тренованої моделі). Бібліотека pickle використовується для завантаження серіалізованої моделі. cv2 — це бібліотека OpenCV для роботи з зображеннями та відео, mediapipe застосовується для комп'ютерного зору, а numpy використовується для роботи з масивами та матрицями даних (Рисунок 4.23).

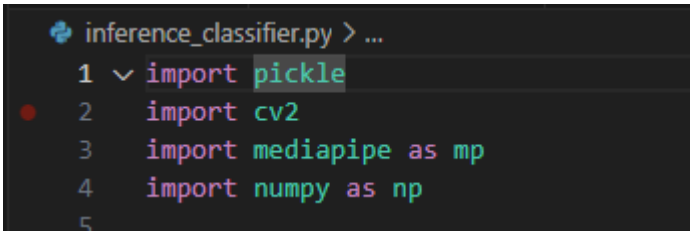



Рисунок 4.23 - Імпорт необхідних бібліотек для інференції та обробки зображень.

Цей фрагмент коду завантажує збережену модель з файлу model.p за допомогою модуля pickle. Спочатку файл відкривається в режимі читання бінарних даних ('rb'), після чого дані з файлу завантажуються в змінну model_dict. Далі з цієї змінної витягується модель, яка зберігається в ключі 'model', і присвоюється змінній model для подальшого використання (Рисунок 4.24).

```
6   model_dict = pickle.load(open('./model.p', 'rb'))
7   model = model_dict['model']
```

Рисунок 4.24 – Завантаження серіалізованої моделі з файлу за допомогою pickle.

Цей фрагмент коду ініціалізує об'єкт cap для захоплення відео з камери за допомогою OpenCV. Використовується функція cv2.VideoCapture(0), де 0 вказує на використання за замовчуванням камери комп'ютера (звичайно, це вбудована веб-камера). Після цього об'єкт cap може бути використаний для зчитування відеопотоку з цієї камери (Рисунок 4.25).

```
9   cap = cv2.VideoCapture(0)
```

Рисунок 4.25 – Ініціалізація відеопотоку з камери за допомогою OpenCV.

Цей фрагмент коду імпортує та ініціалізує різні модулі з бібліотеки mediapipe. Спочатку mp_hands налаштовується на використання рішення для розпізнавання рук з mp.solutions.hands. Далі mp_drawing імпортує утиліти для малювання (наприклад, для малювання ключових точок на зображеннях) з mp.solutions.drawing_utils. Останній рядок ініціалізує mp_drawing_styles, що використовується для додаткових стилів малювання в Mediapipe (Рисунок 4.26)

```
11  mp_hands = mp.solutions.hands
12  mp_drawing = mp.solutions.drawing_utils
13  mp_drawing_styles = mp.solutions.drawing_styles
14
```

Рисунок 4.26 – Імпорт та ініціалізація компонентів Mediapipe для розпізнавання рук і малювання.

Цей фрагмент коду визначає словник labels_dict, який містить відповідність числових міток до певних літер (наприклад, 0 -> 'A', 1 -> 'B', 2 -> 'C' і так далі). Це може бути використано для інтерпретації або перекладу числових міток, отриманих з моделі, у зрозумілі символи або класи. Далі код має нескінченний цикл while True, що, ймовірно, буде використовуватися для обробки відео або даних в реальному часі (Рисунок 4.27, 4.28).

```
labels_dict = {0: 'A', 1: 'B', 2: 'C', 3: 'D', 4: 'E', 5: 'F', 6: 'G', 7: 'H', 8: 'I', 9: 'K', 10: 'L', 11: 'M', 12: 'N',
while True:
```

```
 14: 'P', 15: 'Q', 16: 'R', 17: 'S', 18: 'T', 19: 'U', 20: 'V', 21: 'W', 22: 'X', 23: 'Y'}
```

Рисунок 4.27, 4.28 – Визначення міток для класифікації та ініціалізація циклу для обробки даних.

Цей фрагмент коду ініціалізує кілька порожніх списків: a_aux, а також інші два списки, значення яких не визначені в поточному фрагменті. Далі відбувається зчитування кадру з відеопотоку за допомогою cap.read(), де frame містить отриманий зображення, а _ (підкреслення) використовується для ігнорування додаткових значень, що повертаються разом із кадром. Потім виводяться розміри кадру за допомогою frame.shape, зберігаючи ширину в змінній w_ та висоту, яка не зберігається. Останнім кроком зображення в кольоровій схемі BGR перетворюється в RGB за допомогою cv2.cvtColor()(Рисунок 4.29).

```
a_aux = []
= []
= []

, frame = cap.read()

W, _ = frame.shape

me_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
```

Рисунок 4.29 – Визначення міток для класифікації та ініціалізація циклу для обробки даних.

Цей фрагмент коду обробляє зображення за допомогою бібліотеки mediapipe, зокрема для розпізнавання та відображення ключових точок рук. Спочатку результат обробки кадру передається функції hands.process(), де frame_rgb містить кадр у форматі RGB. Потім перевіряється, чи є в результатах кілька рук (через results.multi_hand_landmarks). Якщо кілька рук знайдено, кожну з них обробляють окремо, і для кожної руки викликається функція mp_drawing.draw_landmarks(), яка малює ключові точки на зображенні та з'єднує їх за допомогою ліній, використовуючи стиль за замовчуванням для рук та з'єднань(Рисунок 4.30).

```
ults = hands.process(frame_rgb)
results.multi_hand_landmarks:
 for hand_landmarks in results.multi_hand_landmarks:
     mp_drawing.draw_landmarks(
         frame,
         hand_landmarks,
         mp_hands.HAND_CONNECTIONS,
         mp_drawing_styles.get_default_hand_landmarks_style(),
         mp_drawing_styles.get_default_hand_connections_style())
```

Рисунок 4.30 – Обробка та малювання ключових точок рук за допомогою Mediapipe.

Цей фрагмент коду обробляє координати ключових точок рук, отриманих від Mediapipe. Для кожної руки в результатах (results.multi_hand_landmarks), координати кожної точки (x, y) зберігаються в окремі списки x_ і y_. Після цього зберігаються відносні координати (нормалізовані), віднявши мінімальні значення

по кожній осі для кращої масштабованості даних. Це дозволяє підготувати дані для подальшої обробки або навчання моделі, де координати точок будуть мати однаковий масштаб (Рисунок 4.31).

```
40  for hand_landmarks in results.multi_hand_landmarks:
41      for i in range(len(hand_landmarks.landmark)):
42          x = hand_landmarks.landmark[i].x
43          y = hand_landmarks.landmark[i].y
44
45          x_.append(x)
46          y_.append(y)
47
48      for i in range(len(hand_landmarks.landmark)):
49          x = hand_landmarks.landmark[i].x
50          y = hand_landmarks.landmark[i].y
51          data_aux.append(x - min(x_))
52          data_aux.append(y - min(y_))
```

Рисунок 4.31 – Обробка та нормалізація координат ключових точок для класифікації.

Цей фрагмент коду відповідає за обчислення координат обмежувального прямокутника для кожної руки, заснованого на нормалізованих координатах ключових точок. Значення x1, y1 і x2, y2 розраховуються як мінімальні та максимальні координати по осях X і Y, після чого вони множаться на відповідні розміри зображення (ширина W та висота H) для перетворення на піксельні координати. Після цього додаються коригування, щоб отримати точніші межі для прямокутника, зазвичай зменшуючи координати на певну кількість пікселів(Рисунок 4.32).

```
54      x1 = int(min(x_) * W) - 10
55      y1 = int(min(y_) * H) - 10
56
57      x2 = int(max(x_) * W) - 10
58      y2 = int(max(y_) * H) - 10
```

Рисунок 4.32 – Розрахунок координат обмежувального прямокутника для нормалізованих координат ключових точок.

Цей фрагмент коду відповідає за здійснення передбачення за допомогою моделі. Спочатку дані data_aux, що містять нормалізовані координати ключових точок, передаються до моделі через model.predict(), де результат зберігається в змінній prediction. Оскільки модель повертає прогноз у вигляді числового індексу, цей результат перетворюється на відповідну літеру або символ, використовуючи словник labels_dict, де числовий індекс перетворюється на символ (наприклад, A, B, C тощо).(Рисунок 4.33).

```
60    prediction = model.predict([np.asarray(data_aux)])
61
62    predicted_character = labels_dict[int(prediction[0])]
63
```

Рисунок 4.33 – Прогнозування символу за допомогою моделі та перетворення результату в літеру

Цей фрагмент коду малює прямокутник на зображенні за допомогою функції cv2.rectangle(), використовуючи координати, отримані раніше. Прямокутник визначає область, де були знайдені ключові точки руки. Після цього на зображенні відображається прогнозований символ, що міститься в змінній predicted_character, використовуючи функцію cv2.putText(). Текст розташовується в координатах, що відповідають верхній лівій частині прямокутника, і малюється чорним кольором з шрифтом cv2.FONT_HERSHEY_SIMPLEX (Рисунок 4.34).

```
64    cv2.rectangle(frame, (x1, y1), (x2, y2), (0, 0, 0), 4)
65    cv2.putText(frame, predicted_character, (x1, y1 - 10), cv2.FONT_HERSHEY_SIMPLEX, 1.3, (0, 0, 0), 3,
66                cv2.LINE_AA)
67
```

Рисунок 4.34 – Малювання обмежувального прямокутника та відображення прогнозованого символу на зображенні.

Цей фрагмент коду відображає поточний кадр на екрані за допомогою функції cv2.imshow(), де виводиться зображення з назвою вікна "frame". Функція cv2.waitKey(1) дозволяє зберігати це вікно відкритим, чекаючи натискання клавіші на 1 мілісекунду. Це забезпечує можливість перегляду результатів в реальному часі під час обробки відео або зображень (Рисунок 4.35).

```
cv2.imshow('frame', frame)
cv2.waitKey(1)
```

Рисунок 4.35 – Відображення обробленого кадру та очікування натискання клавіші.

### 4.5 Запуск і тестування системи

Цей знімок екрану показує виведення з консолі під час виконання Python скрипту. Оповіщення вказує на використання спеціальних операцій oneDNN для прискорення виконання, але з можливими незначними змінами в результатах через округлення з плаваючою комою. Також є попередження, що менеджер зворотного зв'язку потребує моделі з одною ознакою інференсу, а також попередження про використання прямокутної області заміни замість квадратної для обробки зображень. Інформація про те, що TensorFlow Lite використовує делегат XNNPACK для прискорення обчислень на CPU(Рисунок 4.36).

```
WARNING: All log messages before absl::Initial
izeLog() is called are written to STDERR
W0000 00:00:1734306335.510125    5636 inferenc
e_feedback_manager.cc:114] Feedback manager re
quires a model with a single signature inferen
ce. Disabling support for feedback tensors.
W0000 00:00:1734306335.522884    2720 inferenc
e_feedback_manager.cc:114] Feedback manager re
quires a model with a single signature inferen
ce. Disabling support for feedback tensors.
W0000 00:00:1734306337.613595   14996 landmark
_projection_calculator.cc:186] Using NORM_RECT
 without IMAGE_DIMENSIONS is only supported fo
r the square ROI. Provide IMAGE_DIMENSIONS or
use PROJECTION_MATRIX.
```

Рисунок 4.36 – Відображення обробленого кадру та очікування натискання клавіші.

Перевіряємо чи система розпізнає мою  руку (Рисунок 4.37)

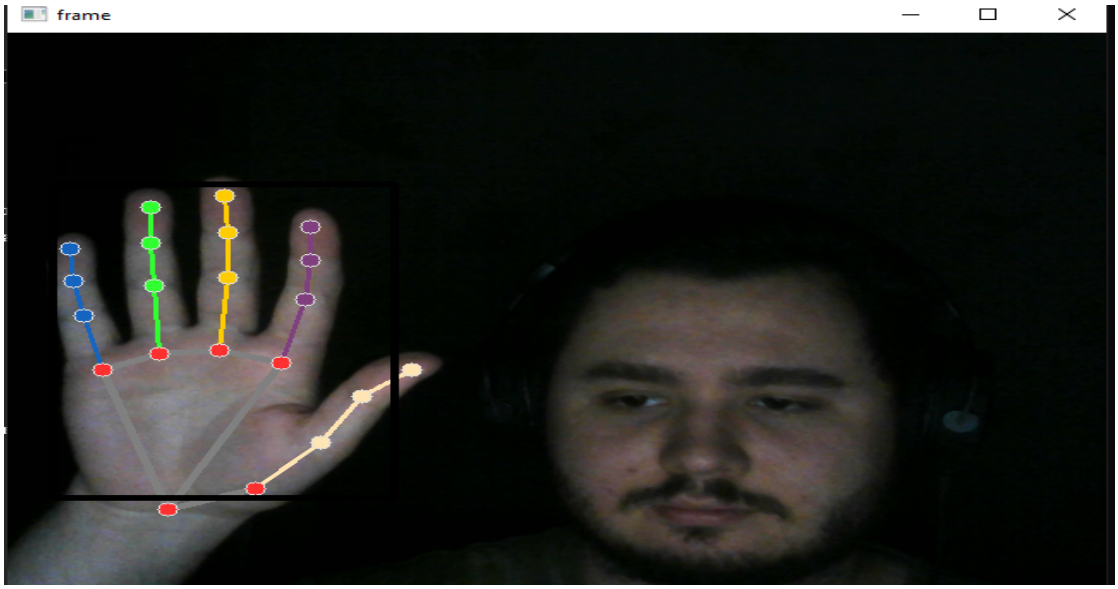



Рисунок 4.37 – Перевірка системи

Фотофіксація що система працює(спробуємо впевнитися що система розпізнає мову жестів) (Рисунок 4.38), (Рисунок 4.39), (Рисунок 4.40), (Рисунок 4.41),(Рисунок4.42),(Рисунок4.43)


Рисунок 4.38 - Англійський алфавіт жестової мови


Рисунок 4.39 - Class 0 (Буква A)

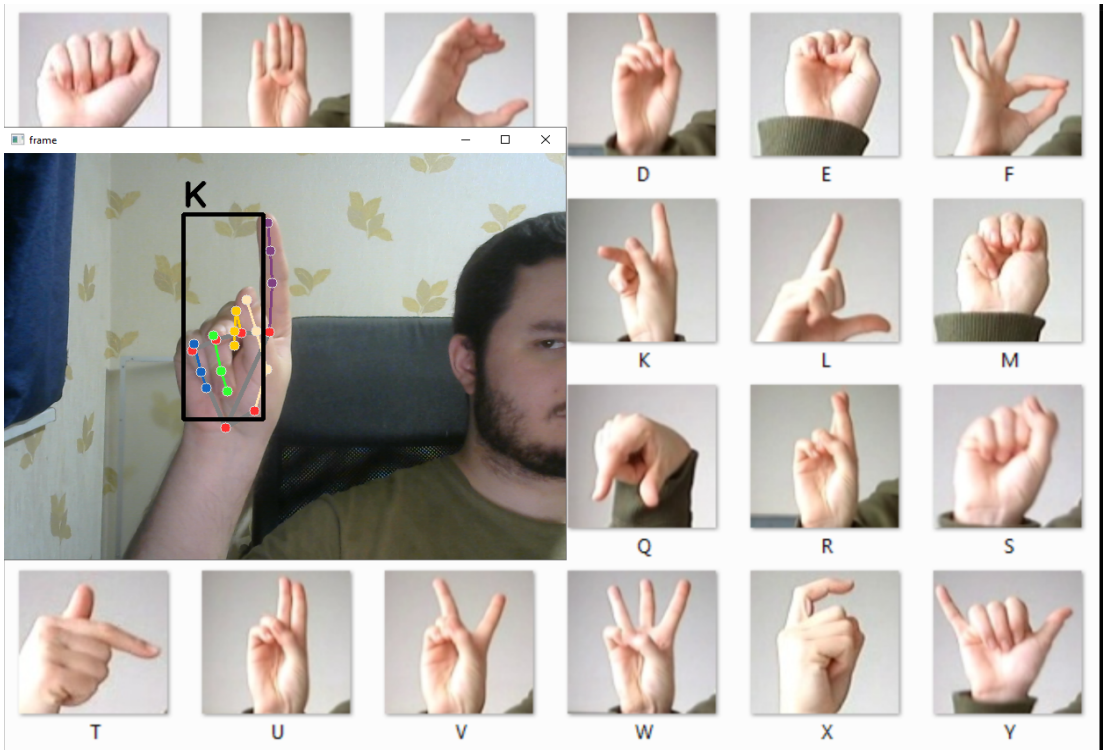



Рисунок 4.40 - Class 9 (Буква K)

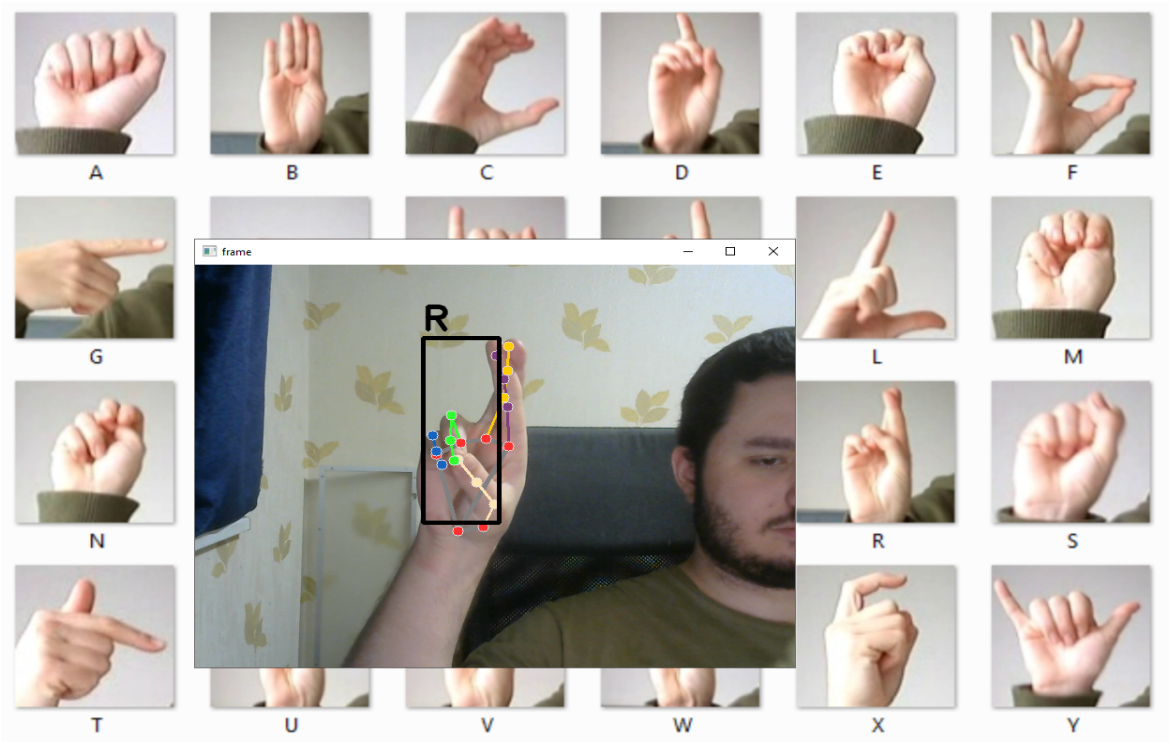



Рисунок 4.41 - Class 17 (Буква R)

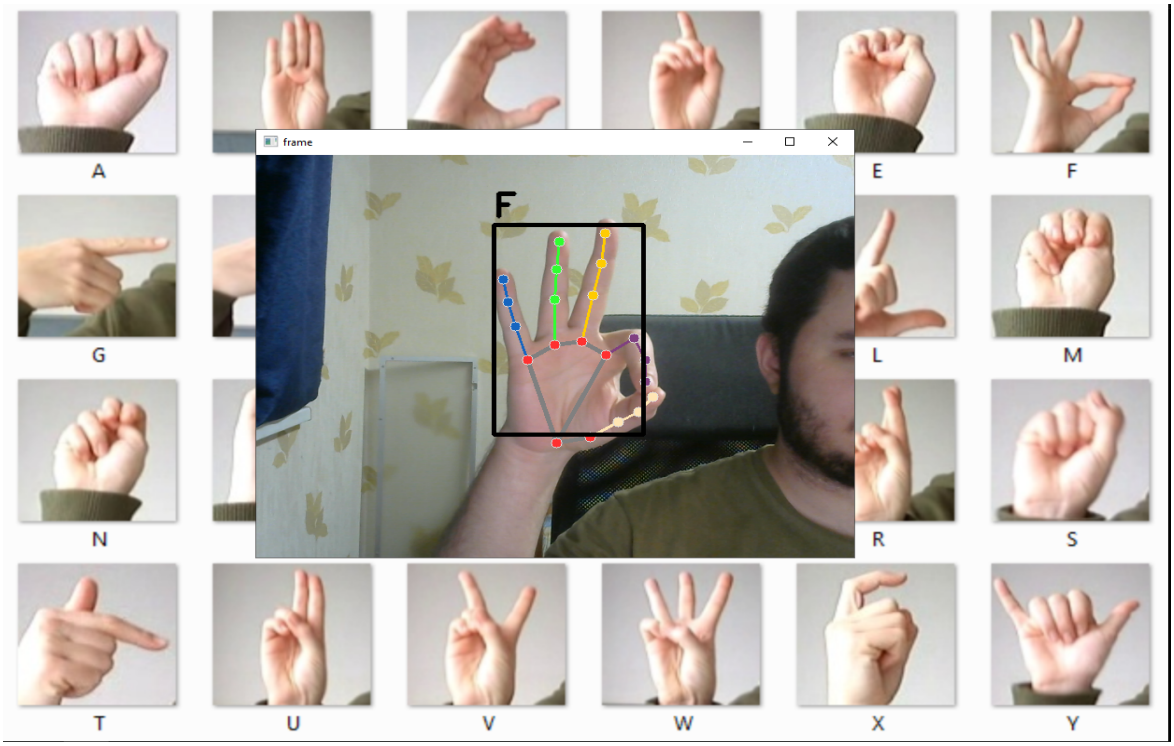



Рисунок 4.42 - Class 5 (Буква F)

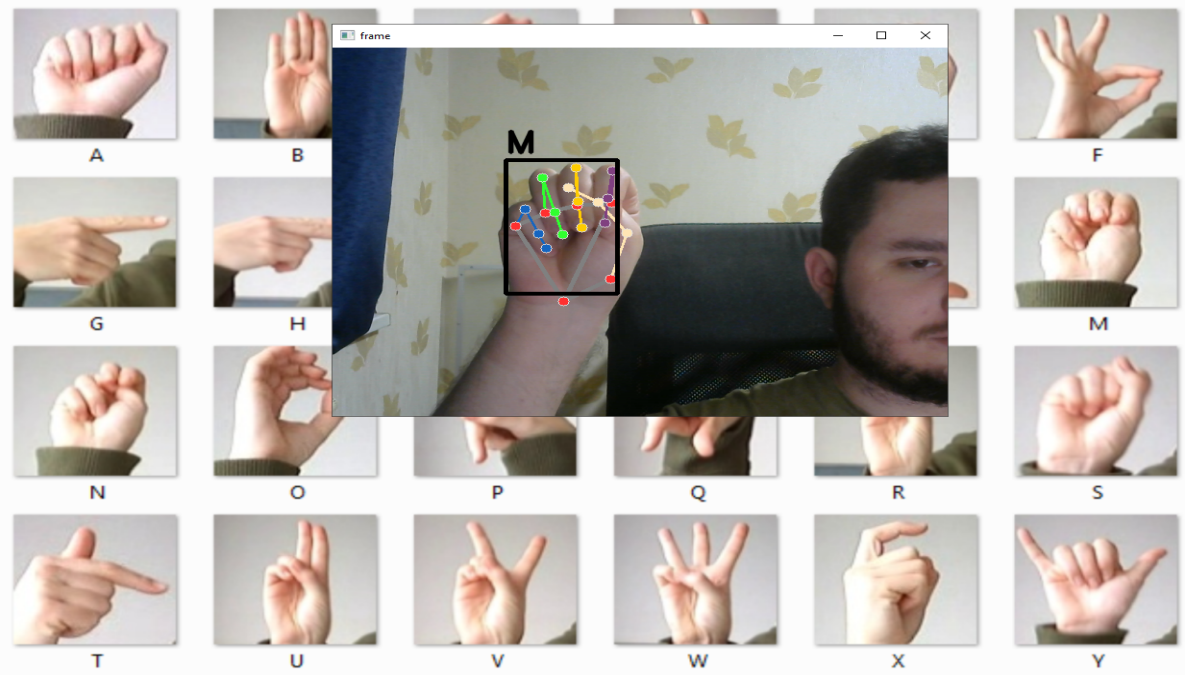



Рисунок 4.43 - Class 11 (Буква M)

# ВИСНОВОК

У даній магістерській роботі було розглянуто і проаналізовано технології, що використовуються для створення систем розпізнавання жестової мови, зокрема на основі нейронних мереж та методів машинного навчання. Основною метою було створення інтелектуальної системи, здатної розпізнавати жести людей і переводити їх в текст для подальшої взаємодії з користувачами.

Було проведено дослідження різних підходів до обробки зображень та відео, використовуючи згорткові нейронні мережі (CNN), а також розглянуті методи глибокого навчання для класифікації жестів. Зокрема, було зосереджено увагу на технологіях, таких як RNN для обробки послідовних даних та трансформери для поліпшення точності розпізнавання.

Система була розроблена із застосуванням алгоритмів машинного зору для виявлення рук та їх рухів, а також з використанням нейронних мереж для інтерпретації цих рухів. Також було створено ефективний механізм для інтеграції розпізнаних жестів в текстову форму, що дозволяє забезпечити зручний інтерфейс для людей з обмеженими можливостями.

Робота демонструє потенціал використання технологій комп'ютерного зору та машинного навчання для створення доступних систем комунікації, які можуть значно покращити якість життя людей з обмеженими можливостями, зокрема, тих, хто має труднощі з мовленням.

# ПЕРЕЛІК ПОСИЛАНЬ


1. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
2. LeCun, Y., Bottou, L., Orr, G. B., & Müller, K. R. (1998). Efficient BackProp. In Neural Networks: Tricks of the Trade (pp. 9-50). Springer.
3. Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet classification with deep convolutional neural networks. Advances in Neural Information Processing Systems, 25.
4. Simonyan, K., & Zisserman, A. (2014). Very Deep Convolutional Networks for Large-Scale Image Recognition. In Advances in Neural Information Processing Systems (Vol. 27).
5. Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
6. Russakovsky, O., et al. (2015). ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision, 115(3), 211-252.
7. Szeliski, R. (2010). Computer Vision: Algorithms and Applications. Springer.
8. He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep Residual Learning for Image Recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).
9. Hinton, G. E., & Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. Science, 313(5786), 504-507.
10. Zisserman, A., & Simonyan, K. (2015). Deep Learning for Computer Vision. In Proceedings of the European Conference on Computer Vision (ECCV).

# ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

**ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ**

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

## ІНТЕЛЕКТУАЛЬНА СИСТЕМА РОЗПІЗНАВАННЯ ЖЕСТОВОЇ МОВИ НА ОСНОВІ ТЕХНОЛОГІЇ PYTHON

Виконав студент 6 курсу
групи КНД-62
Кузьменко Олексій Вікторович
Керівник роботи
К.т.н, доц, Гніденко Микола Петрович

Київ - 2025

## МЕТА, ОБ’ЄКТ, ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи – підвищити ефективність розпізнавання жестів за допомогою машинного навчання та створення інтелектуальної системи на основі зібраних даних.

Об’єкт дослідження – процес розпізнавання жестової мови та створення навчального дата-сету.

Предмет дослідження – алгоритми та методи розпізнавання жестових даних із використанням технологій машинного навчання.

## ЗАВДАННЯ

Для реалізації було поставлено та виконано наступні завдання:

1. Проаналізовано сучасні методи розпізнавання жестової мови та існуючі алгоритми комп'ютерного зору.
2. Розроблено архітектуру системи для розпізнавання жестів з використанням бібліотек Python, таких як OpenCV та TensorFlow.
3. Розроблено алгоритми обробки зображень та класифікації жестів, включаючи попередню обробку, сегментацію та аналіз рухів.
4. Реалізовано програмне забезпечення для розпізнавання жестової мови в реальному часі з інтеграцією функцій для навчання моделей.
5. Проведено тестування системи та оцінено ефективність розроблених алгоритмів на основі реальних даних жестової мови.

## АНАЛІЗ АНАЛОГІВ

TEACHABLE MACHINE

SIGNALL


## ПОРІВНЯННЯ ХАРАКТЕРИСТИК АНАЛОГІВ ТА ПРОЕКТУ ДИПЛОМУ

| Характеристика | Google Teachable Machine | MediaPipe | SignAll | TensorFlow Model | Проєкт (моя система) |
|---|---|---|---|---|---|
| Технологія | Машинне навчання | Комп'ютерний зір | Жестова мова | Глибоке навчання | Python, OpenCV, TensorFlow |
| Область застосування | Навчальні проекти | Аналіз жестів | Переклад жестів | Класифікація | Розпізнавання жестової мови |
| Швидкість обробки | Висока | Обмежений | Середня | Висока | Висока |
| Гнучкість | Обмежена | Висока | Обмежена | Висока | Висока |
| Тип пристроїв | Веб, ПК | ПК, мобільні | Спеціалізовані | ПК, сервери | ПК, мобільні пристрої |
| Оптимізація | Середня | Висока | Середня | Висока | Висока |

## КОНЦЕПТ СИСТЕМИ

Інтелектуальна система розпізнавання жестової мови — це програмне рішення, яке використовує технології комп'ютерного зору та глибокого навчання для аналізу і розпізнавання жестів в реальному часі. Система призначена для підвищення доступності комунікації, перекладу жестової мови на текст або мову, а також інтеграції в інтерактивні програми.Основна мета системи — забезпечення точного і швидкого перекладу жестів у текстовий формат для полегшення спілкування між людьми.

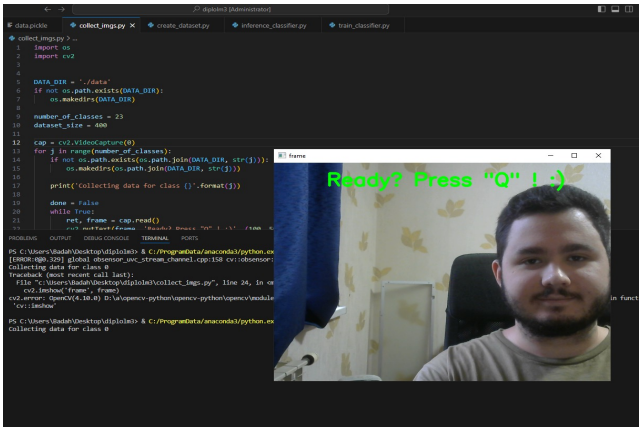
## Основні функції системи

- Розпізнавання жестів рук в реальному часі за допомогою Python-бібліотек OpenCV та MediaPipe.
- Використання нейронних мереж для класифікації жестів.
- Інтуїтивно зрозумілий інтерфейс для користувачів з обмеженими можливостями.
- Підтримка налаштувань для адаптації до різних мов жестів.Можливість інтеграції з іншими системами.

## ТЕХНІЧНЕ ЗАВДАННЯ

- Реалізувати систему захоплення жестів за допомогою бібліотек OpenCV та MediaPipe.
- Розробити алгоритм класифікації жестів на основі нейронних мереж (TensorFlow/Keras).
- Забезпечити підтримку навчання моделі для розширення словника жестів.
- Інтегрувати систему перекладу жестів у текст.
- Реалізувати інтерфейс користувача для налаштування системи та перегляду результатів.
- Провести тестування системи на різних мовах жестів.

## ПРИКЛАДИ РЕАЛІЗАЦІЇ




Створення матеріалу по якому буде створено Датасет

Фото з датасету для букви “C”

## БАЗОВІ ЄТАПИ ПІДГОТОВКИ МАТЕРІАЛУ

•**Збір даних**
•Використання камер для запису жестів або пошук готових баз даних.
•Включення різноманітних жестів, виконаних людьми різного віку,
статі та з різними фізичними характеристиками.
•**Анотація даних**
•Маркування відео чи зображень жестів відповідно до їх значення.
•Використання спеціалізованих інструментів для позначення ключових точок (рук, пальців).
•**Обробка даних**
•Видалення шумів, неправильних записів і дублюючих жестів.
•Зміна розмірів, нормалізація та приведення зображень до єдиного формату.

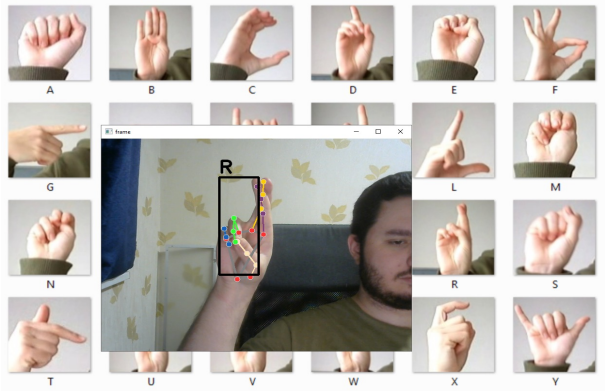
## БАЗОВІ ЄТАПИ ПІДГОТОВКИ МАТЕРІАЛУ

•**Збір даних**
•Використання камер для запису жестів або пошук готових баз даних.
•Включення різноманітних жестів, виконаних людьми різного віку,
статі та з різними фізичними характеристиками.
•**Анотація даних**
•Маркування відео чи зображень жестів відповідно до їх значення.
•Використання спеціалізованих інструментів для позначення ключових точок (рук, пальців).
•**Обробка даних**
•Видалення шумів, неправильних записів і дублюючих жестів.
•Зміна розмірів, нормалізація та приведення зображень до єдиного формату.

## РЕАЛІЗАЦІЯ СИСТЕМИ РОЗПІЗНАВАННЯ ЖЕСТОВОЇ МОВИ

Реалізація системи включає інтеграцію апаратного забезпечення та програмного забезпечення для точного розпізнавання жестів. Основні етапи:
**1.Використання камер або сенсорів**: для зйомки жестів у реальному часі (наприклад, веб-камера, Leap Motion, Kinect).
**2.Обробка зображень**: застосування алгоритмів комп'ютерного зору для визначення позиції рук, пальців і жестів.
**3.Інтерфейс користувача**: інтеграція системи з програмами для виводу тексту на основі розпізнаних жестів.

РЕЗУЛЬТАТ ТА ТЕСТУВАННЯ СИСТЕМИ


## ВИСНОВКИ

1. Розроблено систему розпізнавання жестової мови, яка використовує сучасні алгоритми комп'ютерного зору та глибокого навчання на основі бібліотек Python (OpenCV, TensorFlow, MediaPipe).
2. Інтегровано алгоритми захоплення та обробки жестів, що забезпечує високу точність і швидкість роботи в реальному часі.
3. Описано архітектуру системи, яка включає етапи попередньої обробки, формування набору даних і класифікації жестів за допомогою нейронних мереж.
4. Проведено тестування системи, яке підтвердило її ефективність і стабільність роботи.
5. Розроблена система має потенціал для подальшого вдосконалення, зокрема, додавання нових мов жестів, покращення моделі та інтеграції з іншими інтерфейсами.