Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Навчально-науковий
інститут електроенергетики
(інститут)
Факультет інформаційних технологій
(факультет)
Кафедра програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

магістра
(назва освітньо-кваліфікаційного рівня)

студента	*Басараба Данила Руслановича*
	(ПІБ)
академічної групи	*122М-22-4*
	(шифр)
спеціальності	*122 Комп'ютерні науки*
	(код і назва спеціальності)
освітньої програми	*«122 Комп'ютерні науки»*
	(назва освітньої програми)
на тему:	*Дослідження методів розпізнавання жестів людини в режимі реального часу для організації віддаленого керування відеодзвінком*

_____Д.Р. Басараб

Керівники	Прізвище, ініціали	Оцінка за шкалою		Підпис
		рейтинговою	інституційною	
розділів кваліфікаційної роботи				
спеціальний	*доц. Спірінцев В.В.*			

Рецензент				

Нормоконтролер	*проф. Лактіонов І.С.*			

Дніпро
2023

Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
Завідувач кафедри
Програмного забезпечення комп'ютерних систем
(повна назва)

_____ М.О. Алексєєв
(підпис) (прізвище, ініціали)
« » _____ 20 23 Року

ЗАВДАННЯ
на виконання кваліфікаційної роботи

спеціальності *122 Комп'ютерні науки*
(код і назва спеціальності)

студенту *122м-22-4* *Басарабу Данилу Руслановичу*
(група) (прізвище та ініціали)

Тема кваліфікаційної роботи *Дослідження методів розпізнавання жестів людини в режимі реального часу для організації віддаленого керування відеодзвінком*

1 ПІДСТАВИ ДЛЯ ПРОВЕДЕННЯ РОБОТИ

Наказ ректора НТУ «Дніпровська політехніка» від 09.10.2023 р. № 1227-с

2 МЕТА ТА ВИХІДНІ ДАНІ ДЛЯ ПРОВЕДЕННЯ РОБІТ

Об'єкт досліджень – процес розпізнавання жестів рук у відеопотоках у режимі реального часу.

Предмет досліджень – методи розпізнавання жестів рук за допомогою машинного навчання, керування жестами в застосунку для відеодзвінків в Zoom.

Мета НДР – вдосконалення методів віддаленого керування відеодзвінками в застосунку Zoom за допомогою розпізнавання жестів рук у відеопотоках в режимі реального часу з використанням нейронних мереж.

Вихідні дані для проведення роботи – теоретичні та експериментальні дослідження, основи систем розпізнавання образів, власноруч зібрані тренувальні дані.

3 ОЧІКУВАНІ НАУКОВІ РЕЗУЛЬТАТИ

Новизна запропонованих рішень полягає у створенні системи, що надає можливість використовувати розпізнавання жестів в режимі реального часу для

активації найважливіших функцій застосунку для відеодзвінків Zoom, а також вдосконаленні алгоритму розпізнавання жестів рук за рахунок аналізу не одного, а декількох передбачень перед активацією виконавчої функції.

Практична цінність результатів полягає в тому, що розроблену систему можна використовувати для покращення користувацького досвіду на відеоконференціях в Zoom, адже за допомогою визначених жестів можна увімкнути або вимкнути мікрофон чи камеру, почати демонстрацію екрану, гортати слайди презентації, починати та завершувати локальний чи хмарний запис, виводити список присутніх на відеодзвінку без використання клавіатури чи миші.

4 ВИМОГИ ДО РЕЗУЛЬТАТІВ ВИКОНАННЯ РОБОТИ

Результати досліджень мають бути подані у вигляді, що дозволяє побачити та оцінити безпосереднє використання системи розпізнавання жестів людини для організації віддаленого керування відеодзвінком. Очікується, що в результаті досліджень буде розроблена система розпізнавання і управління жестами, яка матиме високу точність і надійність.

5 ЕТАПИ ВИКОНАННЯ РОБІТ

Найменування етапів робіт	**Строки виконання робіт (початок –кінець)**
Аналіз теми та постановка задачі	12.09.2020-30.09.2023
Розробка системи віддаленого керування функціями відеодзвінка в застосунку Zoom на базі розпізнавання жестів	01.10.2020-15.11.2023
Перевірка працездатності системи та порівняння з існуючими рішеннями	16.11.2020-04.12.2023

6 РЕАЛІЗАЦІЯ РЕЗУЛЬТАТІВ ТА ЕФЕКТИВНІСТЬ

Економічний ефект від реалізації результатів роботи очікується позитивним завдяки представленню нового функціоналу для відеодзвінків, що може дати конкурентну перевагу на ринку.

Соціальний ефект від реалізації результатів роботи очікується позитивним завдяки покращенню користувацького досвіду відеодзвінками й запровадженню більш сучасного способу взаємодіяти з програмою Zoom.

7 ДОДАТКОВІ ВИМОГИ

Завдання видав _____ *Спірінцев В.В.*
(підпис) (прізвище, ініціали)

Завдання прийняв до виконання _____ *Басараб Д.Р.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *09.10.2023 р.*

Термін подання кваліфікаційної роботи до ЕК *04.12.2023*

РЕФЕРАТ

Пояснювальна записка: 84 стор., 30 рис., 2 додатка, 25 джерел.

Об’єкт дослідження: процес розпізнавання жестів рук у відеопотоках в режимі реального часу.

Предмет дослідження: методи розпізнавання жестів рук за допомогою машинного навчання, керування жестами в застосунку для відеодзвінків Zoom.

Мета роботи: вдосконалення методів віддаленого керування відеодзвінками в застосунку Zoom за допомогою жестів рук в відеопотоках в режимі реального часу з використанням нейронних мереж.

Методи дослідження: для виконання поставленої мети були використані методи машинного навчання, теоретичні основи навчання нейронних мереж, методи інтелектуальної обробки зображень у відеопотоках, а також методи скриптового програмування.

Новизна запропонованих рішень: полягає у створенні системи, що надає можливість використовувати розпізнавання жестів в режимі реального часу для активації найважливіших функцій застосунку для відеодзвінків Zoom, а також вдосконаленні алгоритму розпізнавання жестів рук за рахунок аналізу не одного, а декількох передбачень перед активацією виконавчої функції.

Практична цінність полягає в тому, розроблену систему можна використовувати для покращення користувацького досвіду на відеоконференціях в Zoom, адже за допомогою визначених жестів можна увімкнути або вимкнути мікрофон чи камеру, почати демонстрацію екрану, гортати слайди презентації, починати і завершувати локальний чи хмарний запис, виводити список присутніх на відеодзвінку без використання клавіатури чи миші.

Область застосування: Розроблена інформаційна система може застосовуватися для вирішення задач віддаленого керування відеодзвінком в застосунку Zoom в режимі реального часу.

Значення роботи та висновки: Розроблена система дозволяє поліпшити користувацький досвід управління функціями відеоконференцій в застосунку Zoom.

Прогнози щодо розвитку досліджень: Покращити систему шляхом інтеграції її безпосередньо в застосунку Zoom для створення додаткової конкурентної переваги серед аналогів на ринку.

Список ключових слів: розпізнавання жестів, відеодзвінок, віддалене керування, нейронні мережі, машинне навчання, Zoom, Python, Tensorflow, CVZone, OpenCV.

ABSTRACT

Explanatory note: 84 pages, 30 figures, 2 applications, 25 sources.

Object of research: the process of recognizing hand gestures in video streams in real time.

Subject of research: methods of recognition of hand gestures using machine learning, gesture management in the Zoom video calling program.

The purpose of the qualification work: improvement of the methods of remote control of video calls in the Zoom program using hand gestures in video streams in real time using neural networks.

Research methods: machine learning methods, theoretical foundations of learning neural networks, methods of intelligent processing of images in video streams, as well as script programming methods were used to achieve the goal.

The scientific novelty: creation of a system that enables the use of gesture recognition in realtime to activate the most important functions of the Zoom video call application, as well as the improvement of the hand gesture recognition algorithm by analyzing not one, but several predictions before activating the executive function.

The practical significance of the work is that the developed system can be used to improve the user experience at video conferences in Zoom, because with the help of defined gestures user can turn on or off the microphone or camera, start a screen demonstration, flip through the presentation slides, start and end local or cloud recording, display the list of attendees on video call without using a keyboard or mouse.

Field of application: The developed information system can be used to solve the problems of remote control of a video call in the Zoom program in real time.

Value of work and conclusions: The developed system makes it possible to improve the user experience of managing video conferencing functions in the Zoom program.

Forecasts regarding the development of research: Improve the system by integrating it directly into the Zoom program to create an additional competitive advantage among peers in the market.

Keyword list: gesture recognition, video call, remote control, neural networks, machine learning, Zoom, Python, Tensorflow, CVZone, OpenCV.

ЗМІСТ

ВСТУП

Актуальність дослідження. Відеодзвінки є найпопулярнішим способом комунікації на сьогоднішній день. Через карантинні обмеження та воєнні дії процес навчання учнів та студентів цілковито перейшов в онлайн-формат відеоконференцій, де безперечним лідером є платформа Zoom.

Одночасно з тим стрімко розвиваються й технології розпізнавання об’єктів на фото та відео. Після розпізнавання певних об’єктів можна виконувати різні команди базуючись на них, наприклад увімкнення конкретної програми після показу того чи іншого жесту рук. Жести можна використовувати для керування пристроєм, введення тексту або просто надання інформації. Розпізнавання мови жестів — це розділ комп’ютерного зору, який займається розпізнаванням людських жестів. Керування жестами має кілька переваг перед традиційними методами введення, такими як миша та клавіатура. Це може бути більш інтуїтивно зрозумілим і природним, оскільки воно базується на тому, як люди природно взаємодіють із навколишнім світом. Крім того, керування жестами може бути ефективним, оскільки дозволяє людям виконувати кілька дій одночасно.

Розпізнавання мови жестів є активною сферою досліджень, і за останні роки було досягнуто значного прогресу. Сучасні системи розпізнавання мови жестів можуть розпізнавати широкий спектр жестів, включаючи як прості статичні жести, так і складні динамічні жести.

Поєднавши ці дві перспективні галузі, а саме розпізнавання жестів та відеоконференції, можна створити систему, де управління функціями відеодзвінка можна буде здійснювати за допомогою жестів рук. Така технологія вже була анонсована самою платформою Zoom, що свідчить про її актуальність для користувачів. Проте анонсовані функції від розробників сервісу для відеодзвінків включають лише “підняття руки” та додавання реакції “палець догори”. Ознайомившись з функціоналом програми Zoom, можна зробити висновок, що крім згаданих функцій було б добре мати можливість гортання

слайдів презентацій, виставлення початку та кінця запису відео-дзвінка, демонстрації екрану та інших корисних функцій за допомогою жестів рук.

Очікується, що в результаті досліджень буде розроблена система розпізнавання і управління жестами, яка матиме високу точність і надійність.

Мета дослідження. Вдосконалення методів віддаленого керування відеодзвінками в програмі Zoom за допомогою жестів рук в відеопотоках в режимі реального часу з використанням нейронних мереж.

Завдання дослідження. Для досягнення цілі потрібно виконати наступні завдання:

- проаналізувати існуючі підходи для розпізнавання жестів та керування ними;
- провести аналіз існуючих на ринку систем розпізнавання, виявити їхні сильні та слабкі сторони;
- проаналізувати основні засоби для розробки і вибрати оптимальні з них;
- написати програмний код системи розпізнавання жестів.
- інтегрувати команди керування відеодзвінком в програмі Zoom.

Об’єктом дослідження є процес розпізнавання жестів рук в відеопотоках у режимі реального часу.

Предметом дослідження є методи розпізнавання жестів рук за допомогою машинного навчання, керування жестами в програмі для відеодзвінків Zoom.

Методи дослідження. Для виконання поставленої мети були використані методи машинного навчання, теоретичні основи навчання нейронних мереж, методи інтелектуальної обробки зображень у відеопотоках, а також методи скриптового програмування.

Особистий внесок автора. Були описані принципи роботи нейронних мереж у задачах з розпізнавання жестів рук, а також створено програму, що демонструє успішне розпізнавання жестів з подальшим викликом виконавчих функцій програми Zoom.

Новизна запропонованих рішень. Полягає у створенні системи, що надає можливість використовувати розпізнавання жестів в режимі реального часу для активації найважливіших функцій застосунку для відеодзвінків Zoom, а також вдосконаленні алгоритму розпізнавання жестів рук за рахунок аналізу не одного, а декількох передбачень перед активацією виконавчої функції.

Практичне значення. Розроблена система може використовуватися для покращення користувацького досвіду на відеоконференціях в Zoom, адже за допомогою визначених жестів можна увімкнути або вимкнути мікрофон чи камеру, почати демонстрацію екрану, гортати слайди презентації, починати і завершувати локальний чи хмарний запис, виводити список присутніх на відеодзвінку без використання клавіатури чи миші.

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1. Аналітичний огляд області дослідження

Системи розпізнавання образів (“Image recognition systems”) — це комп’ютерні програми, які можуть ідентифікувати та класифікувати об’єкти на зображеннях. Об’єкти можуть являти собою цифри, букви, обличчя, жести, дорожні знаки, або будь-які інші набори полігонів, що мають певні особливості і можуть бути вирізнені з поміж інших і віднесені до певного класу. Об'єкт класифікації прийнято називати образом [1].

Розпізнавання образів прийнято ділити на дві категорії, а саме детекція та ідентифікація. Детекція, це процес виявлення об’єкту на зображенні без віднесення його до певного із заданих класів. Ідентифікація - це процес визначення до якого із заданих класів може відноситися об’єкт на зображенні.

Системи розпізнавання образів працюють з використанням різних технологій, які умовно можна поділити на три групи: методи машинного навчання, методи обробки вхідних сигналів та методи комп’ютерної візуалізації. До методів машинного навчання відносять насамперед нейронні мережі. Машинне навчання наразі є одним з найпопулярніших підходів до розпізнавання зображень. Воно базується на ідеї навчання системи на певному еталонному наборі даних для подальшого самостійного розпізнавання образів, спираючись на отримані під час навчання, знання [2].

Машинне навчання для розпізнавання зображень має ряд переваг у порівнянні з іншими підходами. Перш за все, воно може бути дуже точним. По друге, машинне навчання може адаптуватися до нових типів образів, що не були заявлені під час навчання. По третє, машинне навчання може бути використано для розпізнавання зображень в різних умовах, наприклад для зображень поганої якості, а також з недостатньою освітленістю чи шумом. Найдоцільніше використовувати методи машинного навчання при розпізнаванні біометричних

даних (обличчя, медичні знімки, жести рук), дорожніх знаків, а також звичайних текстових символів (наприклад рукописного тексту).

Наступна технологія, що може бути використана в розпізнаванні образів називаються методами обробки сигналів у нейромережах. Вони часто використовуються для розпізнавання образів, розпізнавання мови та інших завдань, що вимагають обробки сигналів. Існує кілька різних методів обробки сигналів у нейронних мережах. Деякі з найпоширеніших методів включають фільтрування, квантування, стиснення та декодування.

Фільтрування – це процес видалення або послаблення небажаних компонентів сигналу. Фільтри можуть бути аналоговими і цифровими, їх можна використовувати для видалення шуму, корекції частотного спектру або виявлення певних частотних елементів.

Квантування - це процес перетворення аналогового сигналу в цифровий. Квантування здійснюється шляхом дискретизації сигналу в часі та дискретизації амплітуди сигналу.

Стиснення — це процес зменшення розміру цифрового сигналу без втрати значної інформації. Стиснення може бути статичним і динамічним. Статичне стиснення виконується для постійного сигналу, тоді як динамічне стиснення виконується для сигналу, який змінюється з часом.

Декодування - це процес перетворення цифрового сигналу в аналоговий. Декодування є зворотнім процесом до квантування.

Вибір методу обробки сигналу в нейронних мережах залежить від конкретних вимог та завдання. Наприклад, якщо завдання вимагає видалення шуму, можна використовувати фільтр. Якщо завдання вимагає зменшення розміру цифрового сигналу, можна використовувати стиснення.

Нижче наведено кілька прикладів використання методів обробки сигналів у нейронних мережах:

- у системах розпізнавання зображень фільтри використовуються для видалення шумів із зображень, корекції частотного спектру зображень і виявлення певних деталей у зображеннях. Квантування використовується для

перетворення зображень у цифровий формат для подальшого аналізу. Стиснення використовується для зменшення розміру цифрових зображень;

- у системах розпізнавання мови фільтри використовуються для видалення шуму з мовних сигналів, корекції частотного спектру мовних сигналів і виявлення певних звуків у мовних сигналах. Квантування використовується для перетворення мовних сигналів у цифрову форму для подальшого аналізу. Стиснення використовується для зменшення розміру цифрових мовних файлів;
- у медичних діагностичних системах фільтри використовуються для видалення шуму від медичних сигналів, таких як електроенцефалограми, електрокардіограми та ультразвукові сигнали. Квантування використовується для перетворення медичних сигналів у цифровий формат для подальшого аналізу.

Методи обробки сигналів у нейронних мережах є важливим інструментом для багатьох завдань, які потребують обробки сигналів. Вони дозволяють нейронним мережам ефективно обробляти реальну інформацію та отримувати значущі результати. Ось кілька додаткових методів обробки сигналів, які можна використовувати в нейронних мережах:

- методи сегментації, які використовуються для розбиття сигналу на менші частини. Сегментація може бути корисною для розпізнавання конкретних деталей у сигналі або для виявлення аномалій у сигналі;
- спектральні методи аналізу, також відомі як методи дослідження частотного спектру сигналу. Аналіз спектру може бути корисним для визначення конкретних частотних компонентів у сигналі або для визначення загальної форми сигналу;
- методи аналізу часу, які використовуються для вивчення динаміки сигналу в часі. Тимчасовий аналіз може бути корисним для виявлення тенденцій у сигналі або для визначення змін у сигналі з часом.

Ці методи можна використовувати для підвищення точності та ефективності нейронних мереж у різних завданнях.

Наступна група технологій, що може бути використана в роботі з розпізнаванням образів це методи обчислювального зображення. Вони використовуються для візуалізації зображень з метою підвищення точності та ефективності алгоритмів розпізнавання образів. Існує багато методів обчислювального зображення, які можна використовувати для розпізнавання образів. Деякі з найпоширеніших методів включають:

- методи морфологічної обробки, що використовуються для виявлення певних структур на зображеннях. Морфологічну обробку можна використовувати для виявлення об’єктів, контурів, країв та інших структур на зображеннях;
- методи сегментації, які використовуються для розбиття зображення на менші частини. Сегментація може бути корисною для виявлення об’єктів, контурів, меж та інших структур на зображеннях;
- методи пошуку, які використовуються для пошуку певних об’єктів на зображеннях. Пошук можна використовувати для виявлення об’єктів, контурів, країв та інших структур на зображеннях;
- методи відображення, які використовуються для відображення зображень різними способами. Відображення можна використовувати для виявлення певних деталей на зображеннях або для підвищення точності алгоритмів розпізнавання образів.

Методи обчислювального зображення можна використовувати для підвищення точності та ефективності алгоритмів розпізнавання образів у різноманітних завданнях. Наприклад, методи морфологічної обробки можна використовувати для виявлення об’єктів на зображеннях, які занадто малі або занадто слабкі, щоб їх можна було виявити за допомогою стандартних методів.

Методи сегментації можна використовувати для розбиття зображень на менші частини, що може полегшити виявлення об’єктів на зображеннях. Методи пошуку можна використовувати для пошуку конкретних об’єктів на зображеннях, що може бути корисним для таких завдань, як розпізнавання обличчя або дорожніх знаків. Методи відображення можна використовувати для

виявлення певних деталей на зображеннях, що може бути корисним для таких завдань, як розпізнавання рукописного тексту чи розпізнавання емоцій.

Ось кілька конкретних прикладів використання методів обчислювального зображення для розпізнавання образів:

- у системах розпізнавання обличчя методи морфологічної обробки можна використовувати для видалення шуму із зображень обличчя, а методи сегментації можна використовувати для розбиття зображень обличчя на менші частини, що може полегшити виявлення об’єктів на зображеннях;
- у системах розпізнавання дорожніх знаків можна використовувати методи пошуку, щоб знаходити дорожні знаки на зображеннях, а методи картографування можна використовувати для виявлення певних деталей дорожніх знаків, що може бути корисним для підвищення точності розпізнавання;
- у системах розпізнавання рукописного тексту методи морфологічної обробки можна використовувати для видалення шуму із зображень рукописного тексту, а методи сегментації можна використовувати для розбиття зображень рукописного тексту на менші частини, що може полегшити виявлення об’єктів у рукописному тексті.

Методи обчислювального зображення є важливим інструментом для багатьох завдань розпізнавання образів. Вони дозволяють алгоритмам розпізнавання образів ефективно обробляти інформацію про зображення та отримувати більш точні результати.

1.2. Нейронні мережі

Визначення нейронної мережі, яку прийнято назвати «штучною» нейронною мережею, надав винахідник одного з перших нейрокомп’ютерів доктор Роберт Хехт-Нильсен. Він визначає нейронну мережу як обчислювальну систему, що складається з ряду простих, добре взаємопов’язаних елементів, які обробляють інформацію за допомогою динамічної реакції стану на зовнішні

вхідні дані. Автор ідеї надихався справжніми біологічними нейронними зв'язками, що є в мозку людини. Основною обчислювальною одиницею мозку є нейрон. Приблизно 86 мільярдів нейронів можна знайти в нервовій системі людини, і вони з'єднані приблизно з 10^{14} — 10^{15} синапсами [3]. На рис. 1.1 зображено біологічний нейрон (ліворуч) і його математичну модель (праворуч).

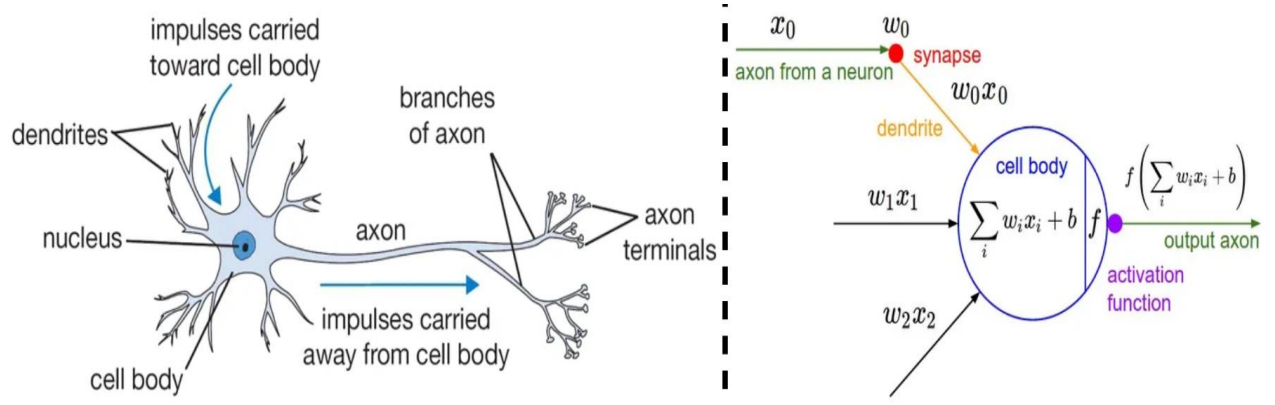

Рис. 1.1. Біологічний нейрон та його математична модель

Основною одиницею обчислення в нейронній мережі є нейрон, який часто називають вузлом або блоком. Він отримує вхідні дані від деяких інших вузлів або із зовнішнього джерела та обчислює вихідні дані. Кожен вхід має асоційовану вагу (w), яка призначається на основі його відносної важливості для інших вхідних даних. Вузол застосовує функцію до зваженої суми своїх вхідних даних. Ідея полягає в тому, що синаптичні сили можна вивчати та контролюють силу впливу та його напрямок: збудливий або гальмівний одного нейрона на інший. У базовій моделі дендрити несуть сигнал до тіла клітини, де всі вони підсумовуються. Якщо кінцева сума перевищує певний поріг, нейрон може спрацьовувати, посилаючи хвилю вздовж свого аксона. У обчислювальній моделі ми припускаємо, що точні моменти спалахів не мають значення, і лише частота пострілів передає інформацію. ми моделюємо швидкість активації нейрона за допомогою функції активації (наприклад, сигмоїдної функції), яка представляє частоту спайків вздовж аксона.

1.2.1. Архітектура нейронної мережі

З наведеного вище пояснення можна зробити висновок, що нейронна мережа складається з нейронів, що з'єднані через синапси, куди надходить інформація. Коли нейронна мережа навчається, то щоразу, коли вдається встановити новий шаблон з даних, мають формуватися нові нейронні зв'язки. Крім того нейронні мережі включають в себе наступні елементи:

- вхідні вузли, на цьому рівні обчислення не виконуються, вони просто передають інформацію наступному шару (здебільшого прихований шар). Блок вузлів також називають шаром;
- приховані вузли, у прихованих шарах виконується проміжна обробка або обчислення, вони виконують обчислення, а потім передають ваги (сигнали чи інформацію) із вхідного шару на наступний (інший прихований шар або на вихідний рівень);
- зв'язки та ваги мережа, що складається із з'єднань, кожне з'єднання передає вихід нейрона і на вхід нейрона j. У цьому сенсі і є попередником j, а j є наступником і. Кожному з'єднанню присвоюється вага Wij;
- функція активації вузла визначає вихід цього вузла з урахуванням вхідних даних або набору вхідних даних. Стандартну схему комп'ютерної мікросхеми можна розглядати як цифрову мережу функцій активації, які можуть бути «УВІМКНЕНО» (1) або «ВИМКНЕНО» (0), залежно від вхідних даних. Це схоже на поведінку лінійного перцептрона в нейронних мережах. Однак саме нелінійна функція активації дозволяє таким мережам обчислювати нетривіальні проблеми, використовуючи лише невелику кількість вузлів. У штучних нейронних мережах ця функція також називається функцією передачі;
- правило навчання, тобто алгоритм, який змінює параметри нейронної мережі, щоб даний вхідний сигнал у мережу створював бажаний результат. Цей процес навчання зазвичай зводиться до зміни вагових коефіцієнтів і порогових значень.

1.2.2. Види нейронних мереж

Існує багато класів нейронних мереж, і ці класи також мають підкласи, серед яких найуживанішими є мережі прямого зв’язку, згорткові, а також рекурентні нейронні мережі.

1.2.2.1. Нейронна мережа прямого зв'язку

Нейронна мережа прямого зв’язку – це штучна нейронна мережа, де зв’язки між елементами не утворюють циклу. У цій мережі інформація рухається тільки в одному напрямку, вперед, від вхідних вузлів, через приховані вузли (якщо такі є) і до вихідних вузлів. У мережі немає циклів і петель. Можна виділити два типи прямої нейронної мережі: одношарові та багатошарові.

Одношаровий персептрон - це найпростіша нейронна мережа прямого зв’язку, яка не містить жодного прихованого шару, що означає, що вона складається лише з одного шару вихідних вузлів. На вхідному рівні не виконуються обчислення, вхідні дані подаються безпосередньо на виходи через серію ваг [4]. Схему наведено на рис. 1.2.

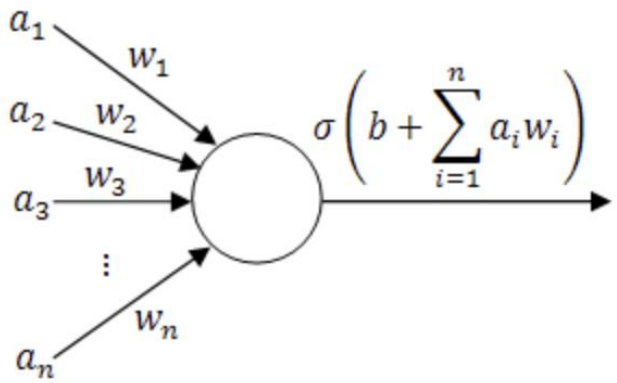

Рис. 1.2. Схема одношарового персептрону

Метод навчання одношарового дискретного персептрона має наступний вигляд. Спочатку вагам w^i, $i = 1, 2, ..., N$, та порогу w^0 присвоюються випадкові значення. Після чого пред’являються черговий екземпляр $x_{ip} = \{x_{1p}, ..., x_{Lp}\}$ з

навчальної множини, $p = 1, 2, ..., m$, де m – кількість екземплярів у навчальній вибірці. Після чого обчислюється значення на виході персептрона $y = \psi((w, x))$, де $w = \{w0, w1, w2, ..., wL\}$ – ваговий набір, що формують пам'ять нейрона.

Наступний етап це коригування ваги персептрона за формулою:

$$w^i = w^i + a\,(y^{s*} - y^s)\,x_{ip}, \tag{1.1}$$

де

a - це крок навчання,

$i = 0, 1, ..., L$;

$x^0 = 1$.

Якщо досягнуто збіжність, то процедура корекції ваг закінчується, якщо ні, то процес треба повторити. Далі по черзі подаються зображення з відомою класифікацією, вибрані з навчального набору, і ваги коригуються.

Процедура навчання триває до тих пір, поки не буде досягнута збіжність, тобто отримані ваги, що забезпечують правильну класифікацію всіх зображень з навчального набору.

Багатошаровий персептрон (MLP) - це клас мереж складається з кількох рівнів обчислювальних одиниць, як правило, пов'язаних між собою уперед. Кожен нейрон одного шару має спрямовані зв'язки з нейронами наступного шару.

У багатьох додатках пристрої цих мереж застосовують сигмоїдну функцію як функцію активації. MLP дуже корисні, і одна з вагомих причин полягає в тому, що вони здатні вивчати нелінійні представлення. Схему наведено на рис. 1.3.

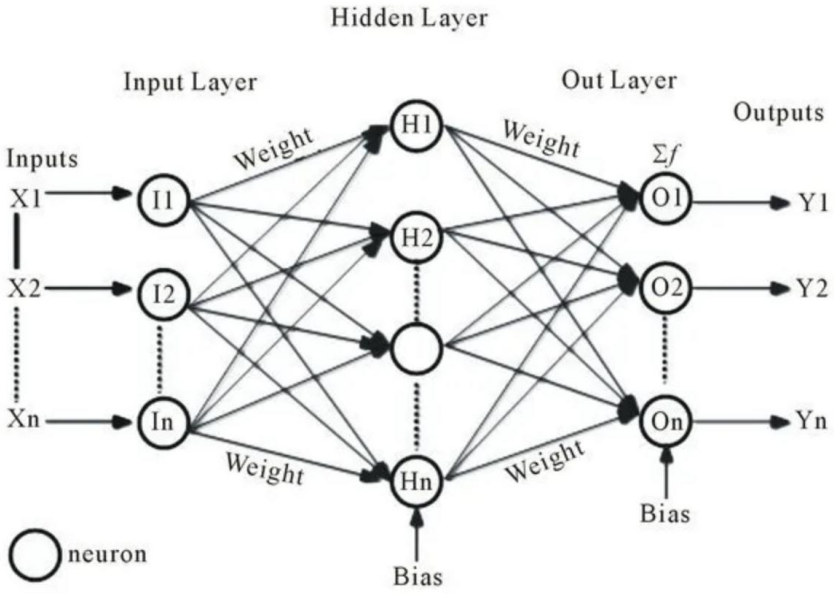

Рис. 1.3. Схема багатошарового персептрона

У кожному нейроні μ-го шару (μі-му нейроні) здійснюється перетворення вхідного вектора $x^{(\mu,i)}$ у вихідну скалярну величину $y^{(\mu,i)}$. Це перетворення складається з двох етапів: спочатку обчислюється дискримінантна функція $net(\mu,i)$, яка далі перетворюється у вихідну величину $y^{(\mu,i)}$.

Дискримінантна функція являє собою відрізок многовимірного ряду Тейлора. Коефіцієнти розкладання відрізка многовимірного ряду Тейлора утворюють вектор вагових коефіцієнтів $w^{(\mu,i)}$, або пам'ять нейрона. Дискримінантна функція нейрона має вигляд:

$$net^{(\mu,i)} = w_0^{(\mu,i)} + \sum_{j=1}^{N} w_j^{(\mu,i)} z_j^{(\mu,i)}, \qquad (1.2)$$

де

$w^{(\mu,i)} = (w_0^{(\mu,i)}, w_1^{(\mu,i)}, ..., w_N^{(\mu,i)})^T$ – вектор вагових коефіцієнтів нейрона;

$xj^{(\mu,i)}$ – j-а компонента N-вимірного вхідного вектора $x^{(\mu,i)}$.

Нелінійне перетворення $y(\mu,i) = \psi(net(\mu,i))$ задається функцією активації, що є монотонною та обмеженою. Зокрема, при позитивних або нульових виходах нейрона функцією може бути сигмоїдна функція $\psi(x) = 1/(1+e\text{-}x)$.

Вектор виходу нейронів μ-го шару позначається через $y(\mu) = (y(\mu,1), y(\mu,2), ..., y(\mu,N\mu))^T$. Процес навчання мережі, здійснюється за допомогою

мінімізації цільової функції *F(w)*, що характеризує інтегральну міру близькості виходів мережі *y(M)(k)* і вказівок учителя *y*(k)*:

$$F(w) = \frac{1}{k}\sum_{m=1}^{k} \quad Q(\varepsilon(w,m)), \tag{1.3}$$

де

k - це номер поточного циклу навчання нейронної мережі;

m - це номер попереднього циклу навчання;

w - це складений вектор-стовпець вагових коефіцієнтів мережі, який складається з векторів-стовпців $w(\mu) = (w(\mu,1)^T, w(\mu,2)^T, ..., w(N\mu)^T)$ для кожного шару $\mu = M, M - 1, ..., 1$;

Q(ε(w, k)) - це миттєвий критерій якості, який входить в інтегральний критерій якості *F(w)*;

ε(w, m) - це вектор помилки мережі, який визначається як різниця між виходами мережі *y(M)(m)* і вказівками учителя *y*(m)*.

Щоб навчити нейронну мережу, потрібно показати їй приклади об'єктів і сказати, до якого класу вони належать. Мережа поступово навчатися класифікувати нові об'єкти, використовуючи градієнтний метод. Градієнтний метод працює шляхом зміни вагових коефіцієнтів мережі в напрямку, протилежному до напрямку помилки [5].

$$w_{k+1} = w_k + \alpha_k s(w_k), \tag{1.4}$$

де

w_k - поточне наближення значень ваг і порогів до оптимального рішення;

w_{k+1} - нове наближення значень ваг і порогів до оптимального рішення;

α_k – крок збіжності;

$s(w_k)$ – напрямок пошуку в *N*-вимірному просторі ваг. Спосіб визначення $s(w_k)$ та α_k на кожній ітерації залежить від особливостей конкретного методу.

1.2.2.2. Згорткові нейронні мережі (CNN)

Згорткові нейронні мережі є найпоширенішим типом нейронної мережі для розпізнавання образів. Вони використовують операції згортки для виявлення особливостей на зображеннях. Операції кадрування працюють шляхом застосування фільтра до зображення. Фільтр — це масив чисел, який визначає, які характеристики зображення важливі для розпізнавання.

CNN зазвичай складаються з кількох рівнів. Перший шар згортає зображення, щоб знайти важливі елементи, такі як краї та кути. Наступні шари потім згортаються, щоб знайти більш складні об’єкти, наприклад фігури та об’єкти. Повна піксельна матриця не подається безпосередньо в CNN, оскільки моделі було б важко витягти характеристики та виявити шаблони з високорозмірної розрідженої матриці. Натомість повне зображення ділиться на невеликі частини, які називаються картами функцій, за допомогою фільтрів або ядер. Модель роботи згорткових нейронних мереж наведено на рис. 1.4 [6].

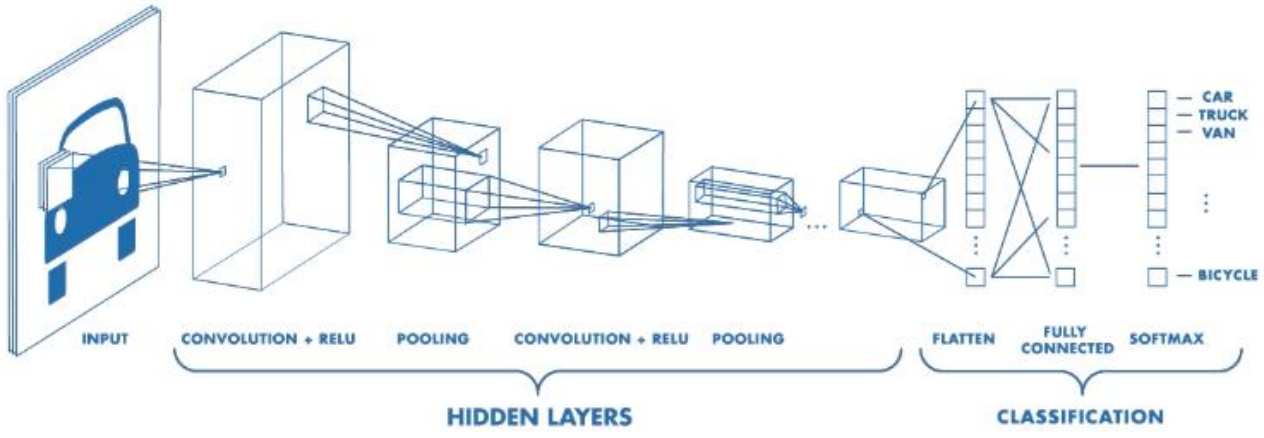

Рис. 1.4. Робота згорткових нейронних мереж у розпізнаванні зображень

Шари згортки в кожному наступному шарі можуть розпізнавати більш складні, деталізовані характеристики — візуальні представлення того, що зображено. Така «ієрархія зростаючої складності та абстракції» відома як ієрархія функцій. Відповідні менші ділянки нормалізуються, і до них застосовується функція активації. Випрямлені лінійні одиниці вважаються найкращими для завдань розпізнавання зображень. Розмір матриці зменшено,

щоб допомогти моделі машинного навчання краще виділяти функції за допомогою шарів об’єднання. Залежно від міток/класів у задачі класифікації зображення вихідний рівень передбачає, до якого класу належить вхідне зображення.

Згортковий шар застосовує до входу операцію згортки, передаючи результат до наступного шару:

$$a^l = h^{l-1} * W^l, \tag{1.5}$$

де

a^l – постсинаптичний (передактиваційний) рівень збудження нейронів l-го шару;

h^{l-1} – активаційний рівень нейронів (l–1)-го шару;

W^l – налагоджуваний параметр l-го шару.

Згортка імітує реакцію окремого нейрона на зоровий стимул. Кожен згортковий нейрон обробляє дані лише для своєї невеликої області в зображенні, яка називається рецептивним полем. Параметри шару складаються з набору фільтрів для навчання, які також називаються ядрами. Фільтри мають таку ж форму, як і рецептивне поле нейрона, але простягаються на всю глибину вхідного зображення. Під час прямого проходу кожен фільтр здійснює згортку за шириною та висотою вхідного зображення. Це означає, що фільтр ковзає по зображенню, обчислюючи скалярний добуток даних фільтру та кожного пікселя в рецептивному полі. Результат згортки називається картою збудження фільтру. Карта збудження являє собою двовимірний масив чисел, які представляють активність нейрона для кожного пікселя в рецептивному полі. Мережа навчається, які фільтри активуються, коли вона виявляє певний конкретний тип ознаки у певному просторовому положенні у вході. Наприклад, один фільтр може активуватися, коли він виявляє край на зображенні, а інший фільтр може активуватися, коли він виявляє колір. Складання карт збудження всіх фільтрів уздовж виміру глибини формує повну ємність виходу.

Агрегувальний шар згорткової нейронної мережі використовується для зменшення розміру вхідного зображення. Це робиться шляхом об'єднання виходів кластерів нейронів одного шару до одного нейрону наступного шару.

Агрегування - це різновид нелінійного зниження дискретизації. Воно полягає в тому, що точне положення ознаки в зображенні не так важливо, як її грубе положення відносно інших ознак. Агрегувальний шар поступово зменшує просторовий розмір вхідного зображення для зменшення кількості параметрів та обчислень у мережі.

Це також допомагає контролювати перенавчання. Агрегувальні шари часто ставлять між послідовними згортковими шарами.

Максимізаційне агрегування - це тип агрегування, який використовує максимальне значення з кожного з кластерів нейронів попереднього шару [7].

$$h^{l}_{x,y} = max\{h^{l-1}_{(x+i)(y+i)}\}, \tag{1.6}$$

де

$h^{l}_{x,y}$ – рівень активації нейрона з індексами (x, y) l-го шару нейронної мережі;

$i=0,...,s$, $j=0,...,s$.

1.2.2.3. Рекурентна нейронна мережа

У рекурентній нейронній мережі (RNN) зв’язки між елементами утворюють спрямований цикл, вони передають дані вперед, але також і назад, від наступних етапів обробки до попередніх. На відміну від нейронних мереж прямого зв’язку, RNN можуть використовувати свою внутрішню пам’ять для обробки довільних послідовностей вхідних даних [8].

Це робить їх застосовними для таких завдань, як несегментоване, пов’язане розпізнавання рукописного тексту, розпізнавання мовлення та інших загальних процесорів послідовності.

Вибір типу нейронної мережі для розпізнавання образів залежить від конкретних завдань, адже CNN є хорошим вибором для завдань, які потребують

виявлення особливостей у зображеннях. При цьому, RNN є хорошим варіантом для завдань, які потребують послідовної обробки даних.

1.3. Процес розпізнавання образів за допомогою машинного навчання

Розпізнавання образів за допомогою машинного навчання — це процес, за допомогою якого комп'ютер може ідентифікувати об'єкти на зображенні та виділити їх в певний заданий клас. Цей процес складається з чотирьох основних етапів [9].

- збір даних: першим кроком є збір даних, які використовуватимуться для навчання моделі. Ці дані традиційно складаються з набору зображень, у яких кожен об'єкт вже позначений як частина певного класу. Дані для розпізнавання образів зазвичай збираються за допомогою камери або сканера. Якщо зображення робляться в лабораторних умовах, можна контролювати освітлення та інші фактори, які можуть вплинути на точність розпізнавання. Якщо зображення робляться в реальному світі, необхідно враховувати ці фактори, щоб гарантувати, що модель буде працювати в різних умовах;
- обробка даних: після збору даних їх необхідно обробити, щоб зробити їх придатними для побудови моделей. Цей процес може включати такі завдання, як редагування, вирівнювання та видалення шуму. Коли зображення зібрано, їх потрібно обробити, щоб зробити їх придатними для тренувальних моделей. Цей етап є дуже важливим, адже від якості зображень напряму залежить результат навчання моделі;
- навчання моделі: на наступному етапі модель базується на наборі даних, що є еталонним, для подальшого самостійного розпізнавання. Існує багато різних алгоритмів машинного навчання, які можна використовувати для розпізнавання образів. Одним із найпоширеніших алгоритмів є згорткова нейронна мережа (CNN). CNN працює шляхом виявлення ознак на зображеннях. Вони навчаються на наборі даних, які складаються із зображення, на якому кожний об'єкт, який потрібно розпізнати, позначити. Іншим популярним алгоритмом є рекурентна нейронна мережа (RNN). RNN може відображати

послідовну швидкість, наприклад відео або аудіо. Часто використовується для розпізнавання об'єктів у відеозаписах;

- перевірка моделі: оцінка точності роботи системи. Цей процес реалізується в наборі даних, які не використовувалися для навчання моделей, але можуть бути точно віднесені до певного із заданих класів. Точність моделі розпізнавання образів вимірюється як частка правильних прогнозів, зроблених моделлю. Чим вища точність, тим більша ймовірність, що результат розпізнавання наступного об’єкту бути правильний.

1.4. Загальні принципи розпізнавання мови жестів

Жести можна використовувати для керування пристроями, спілкування з іншими або просто надання інформації. Розпізнавання мови жестів є активною сферою досліджень, і за останні роки було досягнуто значного прогресу. Загальний принцип розпізнавання жестів можна поділити на три етапи: відстеження руху, відстеження ознак, класифікація жестів.

Відстеження руху є першим етапом розпізнавання мови жестів. Метою цього етапу є визначення рухомого об’єкту на зображенні. Відстеження руху можна здійснювати різними методами, такими як відстеження ключових точок або відстеження функцій.

Відстеження жестів є другим етапом розпізнавання мови жестів. На цьому етапі система визначає певні характеристики зображень. Ці ознаки можна використовувати для визначення типу жесту.

Відстеження функцій можна виконувати за допомогою різних методів, таких як виявлення контурів, опорних точок або текстури [10]. Приклад руки з опорними точками наведено на рис. 1.5.

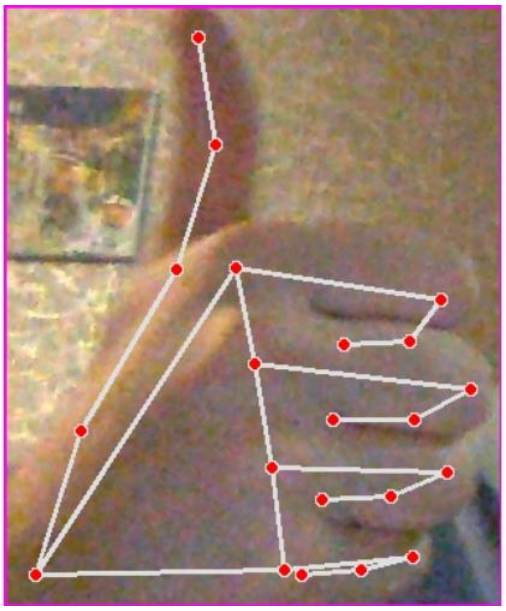

Рис. 1.5. Приклад візуалізації опорних точок на руці

Класифікація жестів є третім етапом розпізнавання мови жестів. На цьому етапі система визначає, який жест виконує людина. Класифікацію жестів можна виконати за допомогою різних методів, таких як машинне навчання або статистичний аналіз.

1.5. Методи розпізнавання жестів рук

На поточний момент можна виділити дві великі групи методів розпізнавання жестів, а саме методи на основі виділення ознак, та методи на основі створення тривимірної моделі. Перша група методів базується на використанні характерних особливостей зображень для визначення положення руки. Такі методи працюють, якщо можна знайти характерні точки або області на об'єкті, які можна використати для його ідентифікації. Недоліком даної групи методів можна назвати зменшення точності розпізнавання за умови знаходження на зображенні об’єктів, що за кольором нагадують колір шкіри людини.

Друга група методів заснована на створенні тривимірної моделі для визначення положення руки в просторі. Для реалізації цих методів необхідно в першу чергу розпізнати жест руки, порівнявши положення рук на вході зображення з двомірною проекцією моделі жесту із бази даних. Моделі об'єктів,

що використовувані в процесі розпізнавання, представлені у вигляді частин поверхні. Співставлення здійснюється шляхом порівняння відновленої поверхні з окремими поверхневими ділянками моделі або розміщення зведеного на основі дедуктивних мір. У деяких завданнях можна обійтися без використання моделей. В цьому випадку достатньо мати представлені у вигляді ділянки в поверхнях, щоб створити можливість захоплення об'єкта. Для реалізації цього методу потрібно створити велику базу даних зображень для порівняння з побудованою моделлю, а також зважати на складності при виділенні ознак з урахуванням анатомічних особливостей кожної людини [11].

1.5.1. Метод визначення країв

Два основні методи визначення країв можна описати як пошук різких змін яскравості. Перший метод використовує те, що найрізкіші зміни відбуваються в точках, де зникає двовимірний аналог другої похідної. Цей метод є застарілим і не застосовується на практиці, тому він не буде розглянутий надалі. Другий метод полягає в пошуках точок, у яких градієнт досягає екстремуму. Цей метод буде розглянуто більш детально. Детектори країв на основі градієнтів оцінюють величину градієнта, зазвичай за допомогою гауссиана як фільтра згладжування. Цю оцінку використовують для визначення положення крайових точок. Як правило, величина градієнта може бути великою вздовж широких смуг на зображенні. Однак контури об'єктів зазвичай є кривими, тому бажано отримати криву, що складена з найбільш характерних точок цієї смуги. Очевидним є підхід пошуку точок, у яких величина градієнта максимальна в напрямку, перпендикулярному до краю. При такому підході перпендикулярність напрямку до краю можна оцінити за напрямом градієнта. Більшість програм пошуку країв діють відповідно до цього алгоритму, хоча все ще тривають жваві дискусії про необхідність суворого дотримання всіх його етапів.

Величина градієнта зазвичай більша вздовж широких смуг на зображенні. Ці смуги групують у криві, які відображають крайові точки. Найпростіший

спосіб зробити це – розрізати смугу перпендикулярно до її напрямку та знайти пік. Напрямок, за яким слід розрізати смугу, визначається напрямком градієнта. На рис.1.6 зліва показана смуга з великою величиною градієнта, на рисунку в центрі – відповідне напрямок розрізу, на рисунку справа – пік у цьому напрямі.

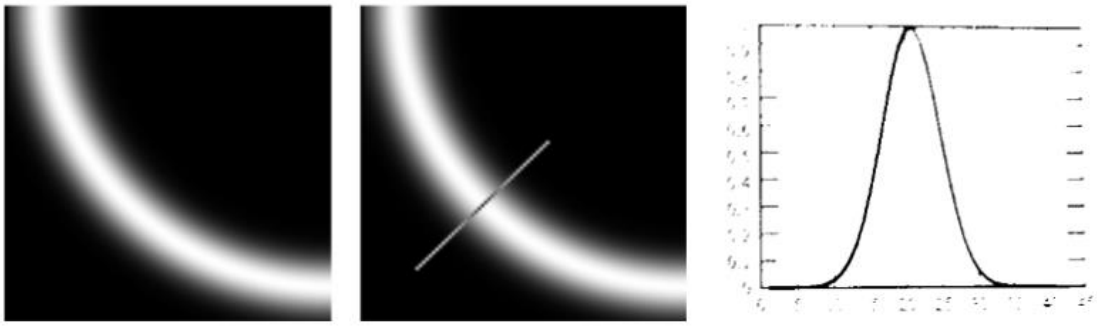

Рис. 1.6. Оцінка по напрямку градієнту

Коли відомі оцінки величини градієнта, потрібно віднайти крайні точки. Величину градієнта можна розглядати як ланцюжок невисоких пагорбів. Локальні екстремуми дають окремі точки – аналогічно вершинам пагорбів. Найкращий критерій – розрізати градієнт на маленькі ділянки вздовж напрямку градієнта, який повинен бути перпендикулярним до краю, і відзначити точки ділянки, у яких ця величина максимальна. В результаті вийде ряд точок вздовж вершин первісного ланцюжка пагорбів, цей процес називається немаксимальним пригніченням.

При немаксимальному пригніченні отримують точки, у яких величина градієнта максимальна вздовж напрямку градієнта. На рис. 1.7 в лівій частині показано, як відновлюється величина градієнта. Точками зображена сітка пікселів. Розглянемо піксель q і спробуємо визначити, чи є градієнт у цій точці максимальним. Вектор градієнта, який проходить через q, не проходить ні через один з пікселів як справа, так і зліва від нього, тому проводять інтерполяцію і знаходять величину градієнта в точках r і g. Якщо величина в точці q більша, ніж у цих двох точках, то q є крайовою точкою. Як правило, значення величини відновлюються шляхом лінійної інтерполяції, при якій для визначення значення

в цих точках використовують пікселі зліва і справа від p і r, відповідно. На рис. 1.7 з правої сторони показано, як визначаються кандидати на наступну крайову точку за умови, що q є крайовою точкою.

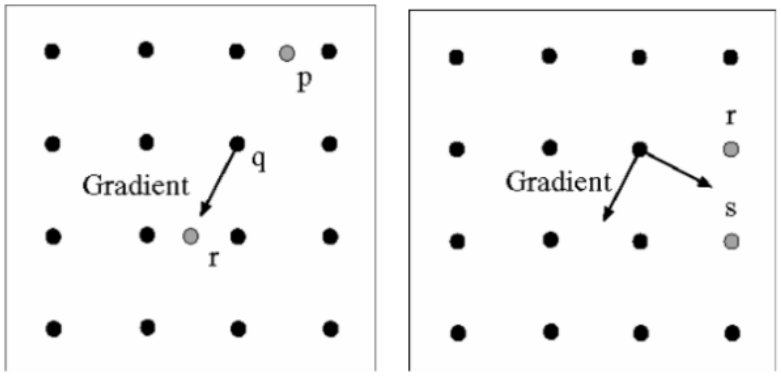

Рис. 1.7. Схема процесу немаксимального пригнічення

Відповідно напрямок пошуку – це перпендикуляр до градієнта, так що точки s і r слід розглядати як наступні крайові точки. Варто зазначити, що, не потрібно обмежуватися пікселями сітки зображення, оскільки відомо, де, ймовірно, знаходиться край між s і r. Отже, знову можна провести інтерполяцію, щоб знайти значення градієнта в точках, які не належать сітці.

Велика кількість отриманих кривих дозволяє в повній мірі описати межі об'єкта. Частково це викликано тим, що максимуми величини градієнта відзначалися незалежно від того, наскільки великим є значення цього максимуму. Частіше застосовують порогову перевірку, щоб переконатися, що максимум перевищує певну нижню межу. Це, в свою чергу, призводить до розриву кривої краю. У цьому випадку зазвичай застосовується концепція гістерезису: є два порогових значення, і на початку ланцюжка використовується більша, а потім, слідуючи по ній, менша [12].

При контурному аналізі можуть виникати складнощі, наприклад виділення непотрібних контурів, тому ускладнюється проведення складного аналізу зображення по контуру об'єкта. Контурний аналіз найчастіше

проводиться для визначення відносних розмірів об'єкта, форми, складності або проводиться пряме зіставлення з шаблоном.

1.5.2. Метод створення скелету для об’єкту на зображенні

У цій системі об'єкти представлені через розкладання їх на відрізки. Основна складність полягає в тому, що потрібно уникнути неоднозначних розкладань. У даній системі для цього знаходять дотичні до силуету кола, які мають великий радіус і охоплюють незначну частину фону. Ці кола, для визначення яких формулюється задача на знаходження екстремуму, дають стартові точки, з яких починається зростання скелету. Щоб знайти частини об'єкта, спочатку знаходять інцидентні гілки скелета, які починаються в точках старту. Потім шукають ознаки, які можуть вказувати на те, що скелет розгалужується. Для кожного компонента скелета починається нова гілка. Як тільки скелет знайдено, кожна гілка ототожнюється з частиною об'єкта. У опис частини включаються диски, пов'язані з її кінцевими точками. Для узгодження частин використовується метод, заснований на графах. Цей метод використовує міру подібності частин, хешовані таблиці та схему голосування. Це дозволяє ефективно отримувати перспективні частини моделі та категорії об'єктів. Кожна категорія об'єктів може бути представлена кількома скелетними графами. Скелетні графи, які найбільше відповідають даним, вибираються за допомогою хешування, заснованого на подібності. Потім вони голосують, щоб визначити, які з них є найбільш ймовірними. Після цього, вони узгоджуються з даними, що спостерігаються. У процесі узгодження потенційно узгоджувані пари відсікаються, якщо їх вартість перевищує певний поріг. Для надання процедурі узгодження стійкості використовується адаптивне узгодження. Це дозволяє скелетним графам моделей змінюватися в процесі узгодження під дією скелетних операторів, які враховують помилки, що виникають на етапі виділення скелета. Кожне застосування одного з цих операторів має певну ціну, а шуканими є відповідності з мінімальною ціною, що не перевищує деякого

порога. Для побудови геометричного скелета руки також може використовуватися алгоритм, який ґрунтується на застосуванні діаграм Вороного. На наступному етапі зайві дуги скелета видаляються за допомогою алгоритму відсікання [13].

1.5.3. Метод нормованої кореляції

Принцип роботи цього методу полягає в порівнянні фільтра з ділянкою зображення з центром у точці, реакція якої розглядається. При цьому оточення зображення, що відповідає ядру фільтра, перетворюється на вектор, який порівнюється з ядром. Сам по собі скалярний набір цих векторів не є достатнім для виявлення елементів, оскільки він може бути великим просто тому, що відповідає яскравій ділянці зображення. Для кращого виявлення елементів потрібно знати косинус кута між вектором фільтра та вектором оточення зображення. Це можна зробити, обчисливши квадратний корінь із суми квадратів значень відповідної області зображення та розподіливши результат на отриману величину.

Нормована кореляція – це метод пошуку моделі, який порівнює дві ділянки зображення. Якщо ділянки схожі, то величина кореляції буде великою і позитивною. Якщо ділянки різні, то величина кореляції буде маленькою та негативною. Цей метод його можна використовувати для різних цілей, як от виявлення об'єктів на зображенні або для розпізнавання жестів. Одним із потенційних застосувань нормованої кореляції є створення систем, які реагують на людські жести. Наприклад, за допомогою цього методу можна створити систему, яка включає освітлення, коли людина махає рукою, або змінює температуру в приміщенні, коли користувач показує пальцем на систему опалення та кондиціонування. У споживчих додатках часто потрібні системи розпізнавання жестів, які не вимагають багато обчислювальних ресурсів. Однак такі системи мають обмежені можливості.

При реалізації цього методу інтерфейс користувача знаходиться у певному стані очікування. Коли відбувається подія, стан інтерфейсу змінюється, і запускається керуюча команда. Цей метод є зручним способом створення інтерфейсу користувача, оскільки він дозволяє легко керувати станами системи. Комп'ютерний зір можна включити до цієї схеми лише одним способом – за допомогою виявлення подій. Це нескладно, оскільки існує лише кілька різних видів подій, і для кожного стану системи відомі події, які можуть статися. Отже, системі потрібно лише вміти визначати, чи сталася подія чи ні.

1.6. Поточний рівень досліджень в області розпізнавання жестів та керування ними

Розпізнавання жестів — це розділ комп’ютерного зору, який займається розпізнаванням жестів людини. Жести можна використовувати для керування пристроями, спілкування з іншими або просто надання інформації. Розпізнавання жестів є активною сферою досліджень, і за останні роки було досягнуто значного прогресу. Сучасні системи розпізнавання жестів можуть розпізнавати широкий спектр жестів, включаючи прості жести, такі як привітання, і складні жести, такі як комбінації жестових вказівок.

Розпізнавання жестів має широкий спектр потенційних застосувань, зокрема керування пристроями (жести можна використовувати для керування такими пристроями, як телевізори, комп’ютери та розумні будинки), спілкування та навчання (жести можна використовувати для спілкування з іншими людьми, наприклад, за допомогою мови жестів або для спілкування з людьми з інвалідністю), розваги (жести можна використовувати для взаємодії з розважальними системами, такими як відеоігри та віртуальна реальність). Саме можливість віддаленого керування є дуже перспективним напрямком розвитку комп’ютерних технологій, при чому керування може здійснюватися за допомогою різних видів вхідних сигналів.

- голосове керування, спосіб керування, який використовує голосові команди для взаємодії з комп’ютером. Голосове керування можна використовувати для виконання різноманітних завдань, таких як запуск програм, перегляд веб-сторінок і керування пристроями;
- відстеження очей, метод керування, який використовує відстеження очей для взаємодії з комп’ютером. Відстеження очей можна використовувати для переміщення курсору, вибору елементів і введення тексту;
- управління жестами, метод управління, який використовує жести для взаємодії з комп'ютером. Жести можна використовувати для виконання широкого кола завдань, таких як керування пристроями, введення тексту та спілкування;
- відстеження руху, метод керування, який використовує відстеження рухів частин тіла для взаємодії з комп’ютером. Відстеження руху можна використовувати для переміщення курсору, вибору елементів і введення тексту;
- мультисенсорне керування, спосіб керування, який використовує кілька сенсорних вводів для взаємодії з комп’ютером. Мультисенсорне керування можна використовувати для різноманітних завдань, таких як керування пристроями, введення тексту та спілкування.

Кожен метод контролю має свої переваги і недоліки. Клавіатура та миша є найточнішими методами введення, але вони можуть бути незручними для людей з обмеженими можливостями. Голосове керування просте та інтуїтивно зрозуміле, але може бути не таким точним, як інші методи введення. Відстеження очей є точним і ефективним, але вимагає спеціального обладнання. Керування жестами є природним та інтуїтивно зрозумілим, але воно може бути менш точним, ніж інші методи введення. Відстеження руху є точним і ефективним, але вимагає спеціального обладнання. Мультисенсорне керування є природним та інтуїтивно зрозумілим, але вимагає спеціального обладнання.

Сучасні системи можуть розпізнавати широкий діапазон жестів і залишатися все більш точними та надійними. Ось кілька прикладів поточних розробок у сфері розпізнавання та керування жестами:

- Microsoft розробляє систему розпізнавання жестів, за допомогою якої можна керувати пристроями Windows [14];
- Google розробляє систему розпізнавання жестів, за допомогою якої можна спілкуватися з розумними будинками [15];
- Apple розробляє систему розпізнавання жестів, яку можна використовувати для взаємодії з відеоіграми [16];
- Telegram додали функцію розпізнавання жестів на відеоповідомленні з можливістю показу анімацій відповідно до виявленого жесту [17].

Ці розробки свідчать про те, що розпізнавання та керування жестами стане все більш важливим у майбутньому.

1.7. Аналіз існуючих рішень

Одне з існуючих рішень запропоновано самою компанією Zoom [18]. За допомогою розпізнавання жестів учасники можуть давати більш помітний зворотний зв'язок. Ця функція перетворює реальний жест на існуючу реакцію Zoom, дозволяючи візуальним жестам, таким як піднята рука, автоматично відображати відповідну реакцію зустрічі та дозволяти користувачам легше реагувати під час зустрічі чи вебінару. Наразі ця функція підтримується для реакцій «Підніми руку» і «Великий палець вгору». Цього набору із двох жестів замало, адже функціонал, що в теорії може бути адаптовано під жестове розпізнавання налічує десятки корисних команд, наприклад "почати запис", "завершити запис", "увімкнути демонстрацію екрану", "вимкнути демонстрацію екрану", "переключити слайд на презентації" та інші. Саме тому було прийнято рішення створити програму, що додасть вищезгадані функції на базі розпізнавання жестів.

1.8. Постановка задачі

Метою роботи є розробка та дослідження ефективності програмного забезпечення розпізнавання жестової мови на основі машинного навчання. Для

цього необхідно розробити систему розпізнавання жестової мови, а також інтегрувати можливість виконання заданих команд на основі отриманих сигналів. Після цього необхідно провести дослідження ефективності розробленої системи.

Розробка системи буде включати в себе наступні етапи:

- збір даних для навчання нейромережі;
- написання програмного коду, що реалізує навчання моделі на зібраних даних;
- відлагодження програмного коду;
- перевірка коректності результатів роботи системи.

В якості еталонних даних будуть використані зображення із жестами руки, котрі спеціально зібрані для цього дослідження.

1.9. Висновки до першого розділу

В першому розділі кваліфікаційної роботи було проведено аналітичний огляд області дослідження, а саме методів розпізнавання жестів людини. Також було наведено теоретичні аспекти побудови різних видів нейронних мереж, а також описані існуючі методи розпізнавання та класифікації зображень із жестами. Для подальшої побудови системи буде використано метод створення скелету об’єкта на зображенні, оскільки він є найоптимальнішим серед наявних.

Крім того було проведено аналіз щодо актуальності використання жестів людини для взаємодії з інтерфейсами, наведено перелік рішень, де технології розпізнавання жестів використовуються для поліпшення користувацького досвіду. В результаті аналізу можна стверджувати, що тема дослідження є актуальною, оскільки системи розпізнавання жестів почали широко використовувати, та навіть розробники застосунку Zoom нещодавно додали підсистему розпізнавання двох різних жестів для відеоконференцій, проте функціонал може бути значно покращено й розширено.

РОЗДІЛ 2
РОЗРОБКА СИСТЕМИ ВІДДАЛЕНОГО КЕРУВАННЯ ВІДЕОДЗВІНКОМ В ZOOM НА БАЗІ РОЗПІЗНАВАННЯ ЖЕСТІВ

2.1. Платформа та середовище розробки

Програмне забезпечення, що реалізує функціонал віддаленого керування відеодзвінком буде написано мовою програмування Python з використанням бібліотек CVzone та TensorFlow.

CVzone - це бібліотека, що містить в собі функції візуалізації елементів на відео з веб-камери комп'ютера, а також потужні модулі розпізнавання рук та обличчя [19]. В своїй основі вона використовує функції бібліотек OpenCV [20] та Mediapipe.

CVzone пропонує кілька функцій для відстеження жестів руки, зокрема, модуль, який можна використовувати для відстеження рук у реальному часі з відеокамери. Він використовує модель машинного навчання MediaPipe Hands [21]. Модуль відстеження руки можна використовувати для створення різноманітних додатків, таких як системи управління жестами, системи віртуальної та доповненої реальності. Серед переваг використання цієї бібліотеки можна навести наступні:

- простота використання, бо ресурси CVzone добре задокументовані, і їх легко використовувати навіть для новачків;
- багатогранність, адже ресурси CVzone охоплюють широкий спектр тем розпізнавання об'єктів, від основ до більш складних концепцій.

TensorFlow — це бібліотека машинного навчання з відкритим кодом, розроблена командою Google Brain. Tensorflow доцільно використовувати для обчислення машинних моделей і глибокого навчання. Популярність TensorFlow пояснюється багатьма речами, але головним чином концепцією обчислювального графіка, автоматичною диференціацією та адаптивністю фреймворку Tensorflow Python API [22]. Це робить вирішення реальних проблем

за допомогою TensorFlow доступним для більшості програмістів. Серед переваг цієї бібліотеки можна навести наступні:

- Tensorflow не обмежується конкретним пристроєм. Він працює так само ефективно на стільниковому пристрої, як і на будь-якому комп'ютері. Бібліотека визначена таким чином, що її розгортання не обмежується жодним конкретним пристроєм;
- він доступний безкоштовно для всіх, хто хоче з ним працювати. Ця функція дозволяє будь-якому користувачеві використовувати цей модуль коли завгодно і де завгодно;
- Tensorflow має кращу потужність візуалізації даних, ніж будь-яка інша доступна бібліотека. Це полегшує роботу з нейронними мережами;
- Tensorflow має Tensorboard, який дозволяє легко налагоджувати вузли. Це допомагає зменшити накладні витрати на відвідування всього коду;
- для роботи TensorFlow використовує системи GPU і CPU. Користувач може вільно використовувати будь-яку архітектуру відповідно до своїх потреб. Система використовує графічний процесор, якщо це не зазначено явно. Цей процес певною мірою зменшує використання пам'яті. Завдяки цій можливості Tensorflow розглядається як бібліотека апаратного прискорення;
- він сумісний з багатьма мовами програмування, такими як Python, C++, JavaScript тощо. Це дозволяє користувачам працювати в середовищі, в якому їм комфортно;
- архітектура TensorFlow використовує TPU, що робить обчислення швидшими, ніж CPU і GPU. Моделі, побудовані на TPU, можна легко розгортати в хмарах і працювати швидше порівняно з двома іншими;
- завдяки підтримці Google Tensorflow часто оновлюється та має чудову продуктивність.

Для розробки програмного коду було обране середовище VSCode. Воно є дуже популярним серед розробників в даний момент, адже має зручний інтерфейс та потужний набір функціоналу. Редактор підтримує обрану для проекту мову програмування Python, а також інші популярні мови, зокрема C,

C++, C#, Java, JavaScript, php, Go, Rust, R, MATLAB, Swift, Kotlin, Dart, Haskell, Ruby. Серед переваг використання Visual Studio Code для розробки на Python можна навести наступні:

- автозаповнення, адже VS Code має вбудоване інтелектуальне автозаповнення, яке допоможе вам швидко та легко вводити код;
- перевірка коду завдяки вбудовані інструментам, які допоможуть вам виявити помилки та недоліки у програмному коді;
- відлагодження, VS Code має вбудовані інструменти, які допомагають відстежувати виконання коду по рядкам та виявляти помилки;
- велика кількість розширень, що додаються самими розробниками для інших розробників. Серед розширень можна знайти дуже багато корисних речей починаючи від зміни кольору інтерфейсу на більш контрастний, і закінчуючи інтелектуальними модулями перевірки правильності написання кодових конструкцій [23];
- кросплатформеність, тобто можливість використання середовища на будь якому пристрої, що підходить під мінімальні вимоги застосунку;
- VS Code є програмним забезпеченням з відкритим кодом, що означає, що ви можете використовувати та змінювати його безкоштовно.

Загалом, VS Code — це потужний і гнучкий редактор коду, який є хорошим вибором для розробки на Python. Приклад інтерфейсу користувача наведено на рис. 2.1. Він складається таких елементів як панель навігації вікнами (1), лічильник помилок та попереджень (2), вікно редагування коду (3) та область терміналу (4). З панелі навігації можна:

- відкрити файлове дерево проекту та редагувати його;
- здійснити пошук по ключовим словам та за потреби замінити їх;
- перейти в режим керування гілкою репозиторія;
- перейти в режим відлагоджувача для порядкового запуску коду та перевірки значень змінних;
- додавати та видаляти сторонні розширення;
- перейти в режим конфігурувати авто-тестів для проекту.

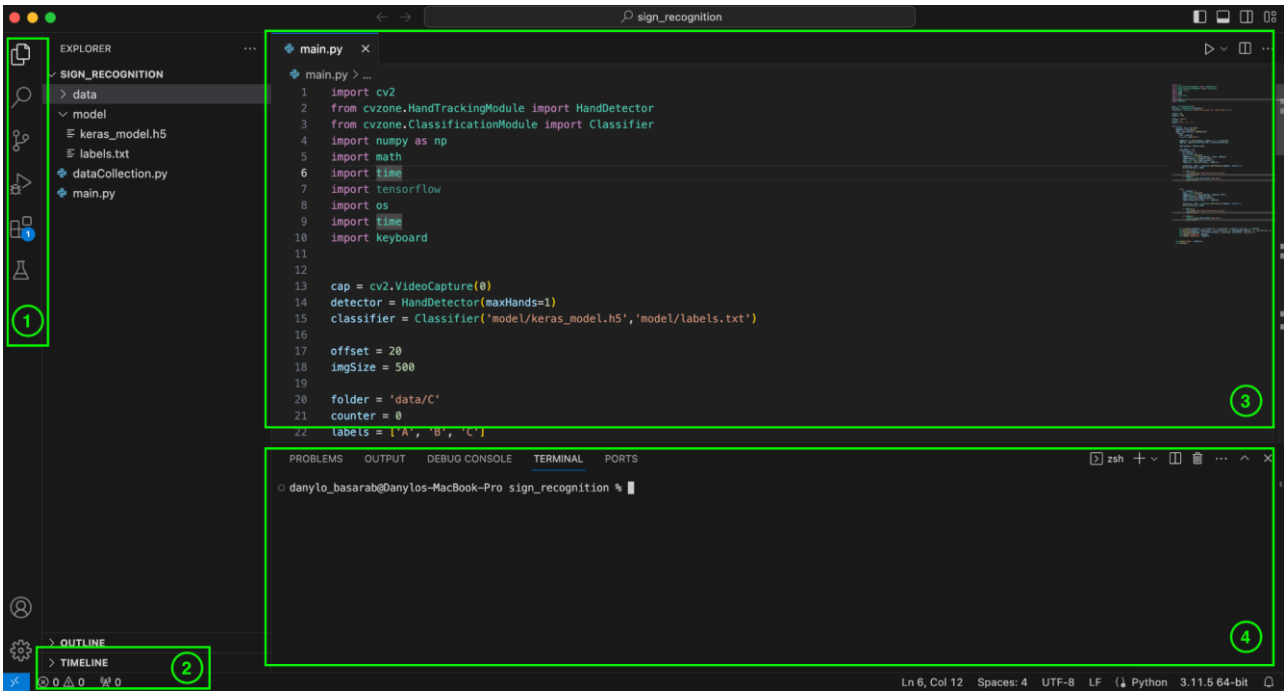

Рис. 2.1. Інтерфейс середовища розробки Visual Studio Code

2.2 Етапи виконання роботи

Проект зберігається в директорії “sign_recognition” та складається зі скрипта запису даних для навчання з вбудованої камери (dataCollection.py), директорії з даними по кожному класу (/data), директорії із натренованою моделлю (/model), а також основного файлу для перевірки працездатності програми (main.py). Директорія з натренованою моделлю, в свою чергу, містить файли “keras_model.h5” та “labels.txt”. Файлова структура зображена на рис. 2.2.

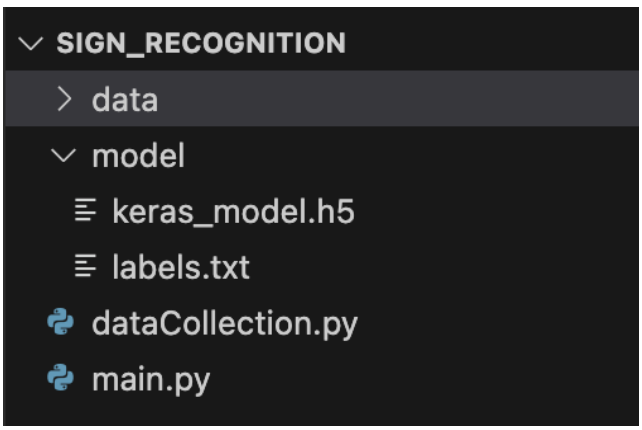

Рис. 2.2. Структура проекту

Роботу було розпочато зі збору даних, на цьому етапі необхідно написати скрипт, що буде робити велику кількість скріншотів (приблизно 300) для кожного класу еквівалентності. Контролювати початок і кінець запису скріншотів можна буде за допомогою утримання кнопки “S” на клавіатурі.

При зборі зображень необхідно пам’ятати, що дані повинні належати саме до одного класу і не мати неточності в трактуванні, адже нейронна мережа може робити помилки у разі навчаннях на неякісних даних. До неякісних також відносять занадто темні зображення, тому запис скріншотів необхідно робити в добре освітленому приміщенні. Оскільки розпізнавання жестів буде відбуватися у реальному часі, необхідно додати зображення, що будуть містити руку не під прямим кутом, а трохи зігнуту відносно камери, бажано в обидва боки. Це значно збільшить точність розпізнавання.

2.2.1. Підключення відеокамери

OpenCV - це бібліотека з відкритим кодом, яка містить алгоритми та функції для обробки зображень. Вона дозволяє виконувати широкий спектр завдань, від базового читання та відображення зображень до виявлення та відстеження об'єктів.

Бібліотека OpenCV містить функцію VideoCapture(), яка дозволяє легко захопити пряму трансляцію з камери. Для використання цієї функції потрібно вказати індекс пристрою. До вашого комп'ютера може бути підключено кілька камер, які нумеруються індексами, починаючи з 0 для вбудованої веб-камери. Функція повертає об'єкт VideoCapture, який можна використовувати для доступу до кадрів з камери (наприклад: camera = cv.VideoCapture(0)).

Після відкриття камери ми можемо отримувати з неї нові кадри за допомогою функції read(). Ця функція повертає матрицю, яка містить кадр у форматі OpenCV.

Щоб отримати доступ до наступного кадру з камери, можна використовувати функцію read(). Якщо кадр є доступним, функція повертає

значення True, інакше - False. Потім кадр можна перетворити в потрібний колірний простір за допомогою функції cvtColor() і відобразити на екрані.

```
image = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
cv.imshow('frame', image)
```

Щоб записати поточний кадр з камери у файл зображення, можна використовувати функцію imwrite(). Функція imwrite() приймає два параметри: повний шлях до файлу, в який потрібно записати кадр, та матрицю, яка містить кадр.

```
cv2.imwrite(“recording.png”, image)
```

Щоб записати пряму трансляцію з камери у відеофайл, можна використовувати функцію VideoWriter(). Функція VideoWriter() приймає чотири параметри:

- назва файлу для збереження відео;
- код відеокодека;
- частоту кадрів у секунду;
- розмір кадру.

```
cv.VideoWriter( filename, fourcc, fps, frameSize)
```

Параметр fourcc - це код, який визначає тип відеокодека, який буде використовуватися для запису відео. OpenCV підтримує різні кодеки, які мають різні характеристики, такі як якість, розмір і швидкість. Параметри частоти кадрів у секунду та розміру кадру залежать від пристрою захоплення відео. Функція VideoWriter() створює новий відеофайл і відкриває його для запису. Після цього функція починає запис кадрів з камери у відеофайл. Нарешті, функція закриває відеофайл.

2.2.2. Збір даних

В цьому підрозділі буде наведено опис програмного коду, що використовується для збору даних з метою подальшого навчання нейронної мережі. Він використовує бібліотеку OpenCV для захоплення відео з веб-камери

та бібліотеку cvzone, що включає модуль HandTrackingModule для виявлення рук у відеокадрах. Код починається з імпорту необхідних бібліотек, таких як cv2, cvzone, numpy, math, time.

Далі створюється об’єкт VideoCapture для захоплення відео з веб-камери. Після цього створюється об’єкт HandDetector для виявлення рук у відеокадрах, з параметром по максимальній кількості можливих виявлених рук:

```
cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
```

Потім код ініціалізує кілька змінних:

- offset, змінна використовується для доповнення обрізаного вручну зображення на певну кількість пікселів з усіх боків;
- imgSize, змінна визначає розмір обрізаного вручну зображення;
- folder, змінна вказує папку, де зберігатимуться обрізані вручну зображення;
- counter, змінна використовується для відстеження кількості збережених обрізаних вручну зображень.

Надалі починається основний цикл коду, усередині якого код спочатку перевіряє, чи доступний новий відеокадр. Якщо новий кадр доступний, то код визначає руки в кадрі:

```
success, img = cap.read()
hands, img = detector.findHands(img)
```

Якщо виявлено руку, код вирізає руку з кадру:

```
x,y,w,h = hand['bbox']
imgCrop = img[y-offset:y+h+offset, x-offset:x+w+offset]
```

Потім код змінює розмір обрізаного вручну зображення до фіксованого розміру. Після цього обрізане зображення руки відцентровується зі зміненим розміром на білому тлі:

```
imgResize = cv2.resize(imgCrop, (imgSize, imgSize))
imgWhite = np.ones((imgSize, imgSize, 3), np.uint8)*255
imgWhite[hGap:hCalc+hGap, :] = imgResize
```

Потім код відображає обрізане зображення руки та біле фонове зображення:

```
cv2.imshow("ImageCrop", imgCrop)
cv2.imshow("ImageWhite", imgWhite)
```

Нарешті, код перевіряє, чи натиснув користувач клавішу 's'. Якщо так, то зберігає біле фонове зображення у файлі. Інтервал між циклами становить 1 мілісекунду:

```
if key == ord('s'):
    counter += 1
    cv2.imwrite(f'{folder}/Image_{time.time()}.jpg',
imgWhite)
    print(counter)
```

Після написання коду необхідно запустити програму та зібрати приблизно по 100 зображень для кожного жесту, що буде підтримуватись системою. Надалі ці зображення буде поміщено в систему навчання нейромережі.

2.2.3. Навчання моделі

Google Teachable Machine — це веб-інструмент, який дозволяє створювати моделі машинного навчання без програмування. Він простий у використанні і доступний для всіх, незалежно від рівня підготовки. Teachable Machine підтримує картинки та аудіофайли [24]. Коли модель навчена, то можна експортувати її в різних форматах, щоб використовувати її в проектах, наприклад в Tensorflow Keras. Keras — це платформа машинного навчання, яка використовується для створення та навчання нейронних мереж. Вона має відкритий код і підтримується компанією Google. Цей високорівневий фреймворк, який дозволяє легко будувати нейронні мережі без необхідності знати деталі низького рівня. Він підтримує широкий спектр архітектур нейронних мереж, включаючи згорточні нейронні мережі, рекурентні нейронні мережі та генеративні нейронні мережі.

Keras є популярним вибором для машинного навчання, оскільки він простий у використанні та потужний. Він використовується для створення різноманітних програм, включаючи розпізнавання об’єктів, розпізнавання мови та машинний переклад [25]. Ось основні особливості Keras:

- висока продуктивність, адже цей високопродуктивний фреймворк можна використовувати для створення великих нейронних мереж;
- простота використання, бо він не потребує знання глибоких знань архітектур нейронних мереж;
- широкий діапазон архітектур, оскільки підтримується широкий спектр архітектур нейронних мереж, що дозволяє використовувати його для вирішення різних завдань.

Надалі буде наведено приклад навчання моделі із подальшим експортом навченої нейромережі у Keras. Першим етапом є створення назв для класів і додавання тренувальних даних. На рис. 2.3 зображено інтерфейс застосунку Teachable Machine, що надає змогу додати файли або з локального носія або ж з хмарного сховища.

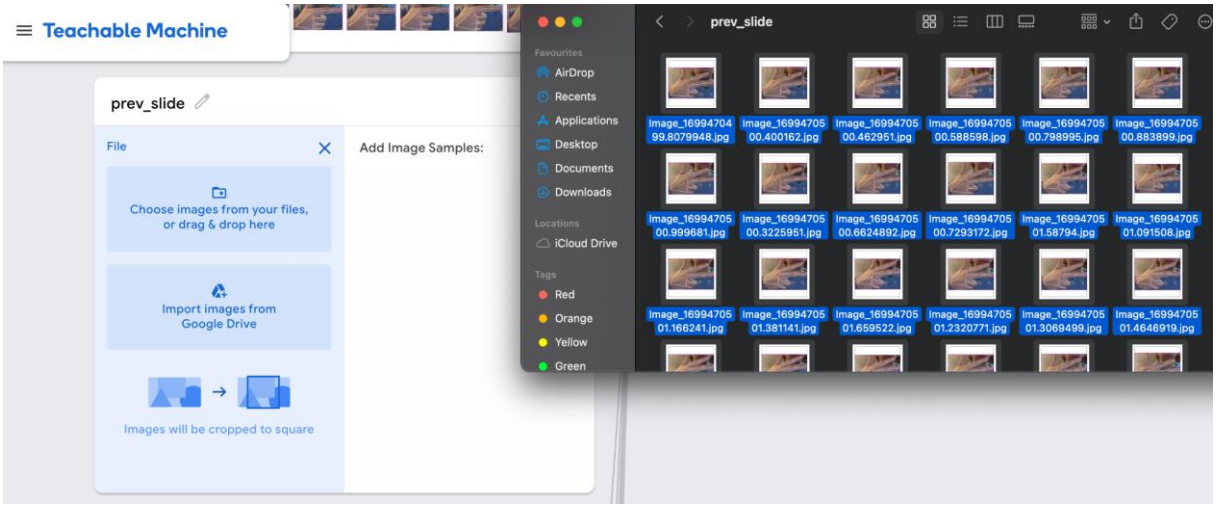

Рис. 2.3. Приклад інтерфейсу Teachable Machine для додавання тестових даних

В межах конкретно цього проекту всі тестові зображення було додані з локального сховища і є унікальними, адже вони були зроблені безпосередньо студентом, а не взяті з мережі.

Після додавання зображень до кожного зі створених класів необхідно перевірити ще раз чи точно дані відповідають заданому класу, адже помилка на цьому етапі спричинить некоректну роботу моделі в цілому. Після перевірки необхідно запустити тренування моделі, змінивши параметри навчання за потреби. На рис. 2.4 наведені параметри, що були обрані для тренування мережі, а саме 50 епох з серії в 16 зображень, темп навчання становить 0,001.

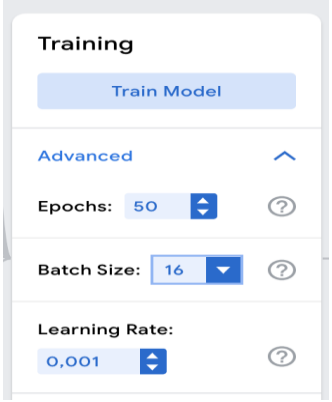

Рис. 2.4. Вікно параметрів навчання нейронної мережі в Teachable Machine

Через декілька хвилин тренування має успішно завершитись та користувач побачить, що в віджет Preview оновився і тепер показує класи даних, а також кнопка Export Model стала активною (рис. 2.5).

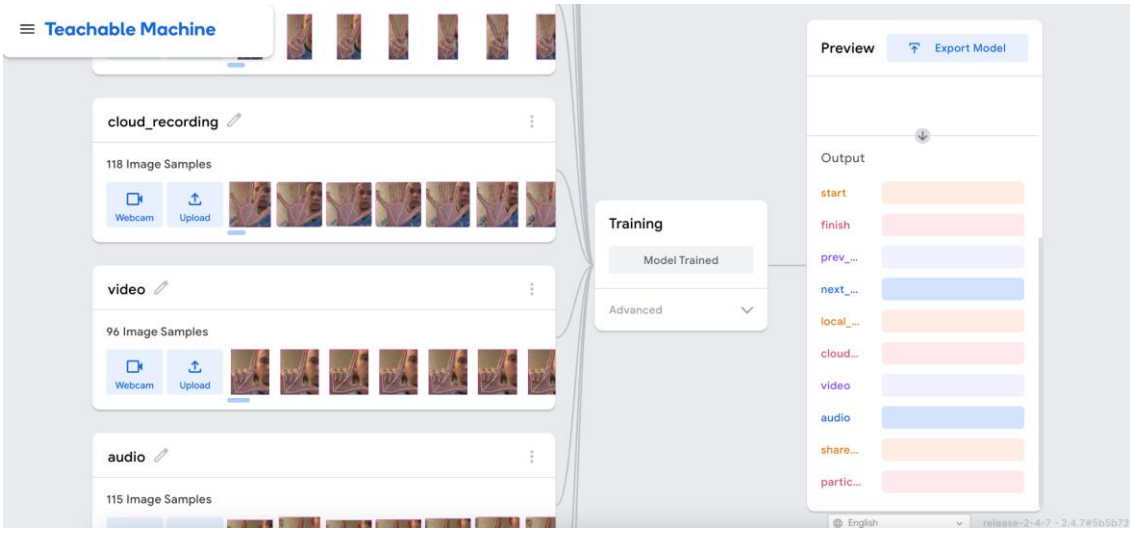

Рис. 2.5. Інтерфейс Teachable Machine після завершення навчання мережі

Інтерфейс Teachable Machine надає змогу побачити метрики точності передбачень нейронної мережі, а саме графік точості передбачень за епоху (рис. 2.6) та матрицю відповідності передбачень (рис. 2.7).

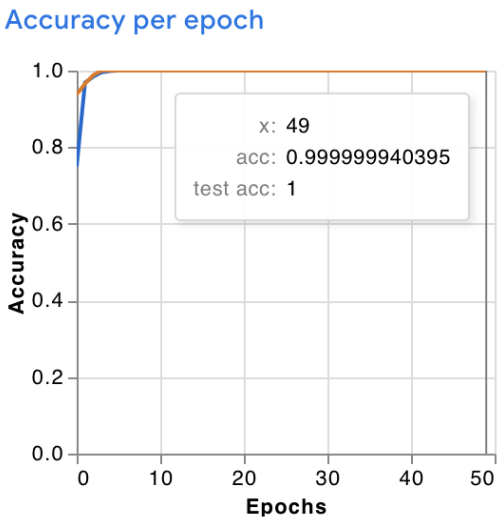

Рис. 2.6. Графік точності передбачень за певну епоху

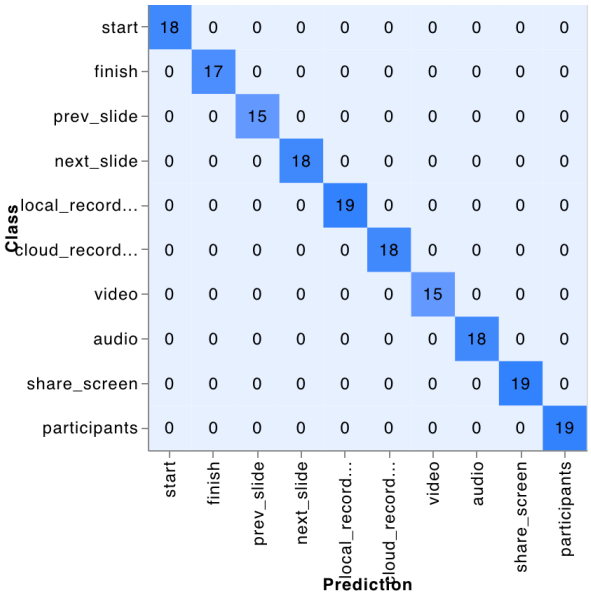

Рис. 2.7. Матриця відповідності класів та передбачень

Спираючись на наведені метрики можна стверджувати, що навчання моделі пройшло успішно і її можна експортувати для подальшої інтеграції в проект. Для того, аби завантажити файли з натренованою моделлю, які будуть використані в даному проєкті для класифікатора необхідно натиснути кнопку

Export Model, після цього обрати Tensorflow - Keras і натиснути Download my model. Інтерфейс застосунку має виглядати як на рис. 2.8. В результаті буде завантажено 2 файли: keras_model.h5 та labels.txt, які треба буде додати до файлів проекту.

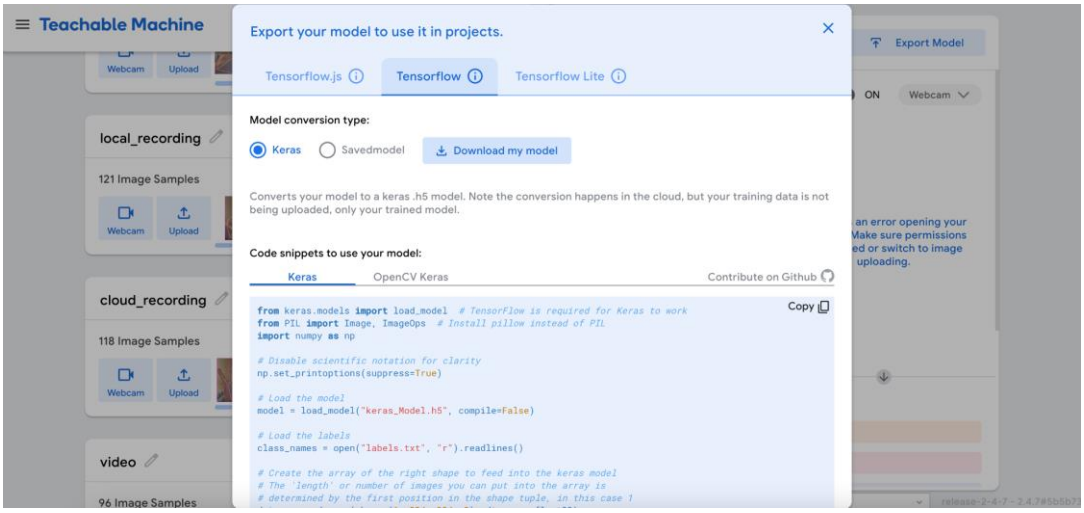

Рис. 2.8. Вікно експорту навченої моделі в Keras

2.2.4. Опис програмного коду основного файлу

Код в основному файлі працює наступним чином. Перш за все для роботи застосунку необхідно імпортувати бібліотеки cvzone, cv2, numpy, math, keyboard, statistics.

Для трансляції відео з камери необхідно додати змінну cap = cv2.VideoCapture(0), що ініціалізує об’єкт VideoCapture, який знімає відео з камери, підключеної до стандартного порту камери комп’ютера, яким зазвичай є веб-камера. Функція cv2.VideoCapture() приймає ціле число як аргумент, який представляє індекс пристрою камери. У цьому випадку 0 означає камеру за замовчуванням.

Після чого створюється екземпляр класу HandDetector під назвою detector (detector = HandDetector(maxHands=1)). Клас HandDetector використовується для виявлення рук у відеопотоці. Параметр maxHands визначає максимальну кількість рук, які детектор намагатиметься виявити в кожному кадрі. У цьому

випадку maxHands=1 означає, що детектор намагатиметься виявити лише одну руку за кадр.

Надалі створюється класифікатор, що бере дані з файлів, які були утворені внаслідок навчання моделі на даних в Google Teachable Machine (classifier = Classifier('model/keras_model.h5','model/labels.txt')). Файл .h5 містить навчені ваги моделі глибокого навчання, а файл labels.txt містить мітки для кожного класу, який може класифікувати модель.

Наступний етап це визначення розміру зображення, що транслюється, а також офсету. Також необхідно створити список з усіма мітками для розпізнавання, при цьому порядок має в точності повторювати послідовність класів з Google Teachable Machine:

```
labels = ['start control', 'end control', 'prev_slide', 'next_slide', 'local_recording', 'cloud_recording', 'video', 'audio', 'share_screen', 'participants']
```

Наступним етапом є побудова інтерфейсу. Для зручності користування застосунком було прийнято рішення відображати статичну підказку із усіма жестами, що підтримуються системою розпізнавання. Кожне таке зображення зчитується з вкладеного в проект файлу та форматується до розміру 90 на 90 пікселів:

```
size = 90
logo1 = cv2.imread('images/participants.png')
logo1 = cv2.resize(logo1, (size, size))
```

Наступним кроком є оголошення змінних, що показують чи увімкнено контроль відеодзвінком за допомогою жестів та змінних для поліпшення точності передбачення нейромережі:

```
control = False
prediction_confidence = []
confident_index = None
last_recognised_command = “”
```

Надалі розпочинається цикл, що постійно фіксує кадри з вебкамери та визначає наявність руки в кожному кадрі. Цикл while повторюється нескінченно довго доки цикл явно не буде розірвано.

```
while(True):
    success, img = cap.read()
    imgOutput = img.copy()
    hands, img= detector.findHands(img)
```

За допомогою методу cap.read() зчитується кадр із веб-камери. Метод read() повертає два значення: логічне значення, яке вказує, чи кадр було успішно прочитано, і сам кадр як масив NumPy.

Після чого створюється копію знятого кадру за допомогою методу img.copy(). Це робиться для того, щоб уникнути безпосередньої зміни вихідного кадру. За допомогою методу detector.findHands() визначається наявність рук у кадрі. Об'єкт детектор є екземпляром класу HandDetector, який відповідає за виявлення руки. Метод findHands() приймає кадр як вхідні дані та повертає два значення: список виявлених рук і змінений кадр із намальованими орієнтирами рук.

Аби статична підказка із підтримуваними жестами відображалась постійно її також треба додати у загальний цикл:

```
roi1 = imgOutput[-size-10:-10, -size-10:-10]
roi1[np.where(logo1)] = 0
roi1 += logo1
```

Перший рядок визначає область відображення для логотипу. Змінна roi1 зберігає підобласть кадру imgOutput. Нотація зрізу [-size-10:-10, -size-10:-10] визначає координати ROI. Зсув -size-10 гарантує, що ROI залишається в межах кадру. Другий рядок встановлює пікселі в ROI на нуль. Вираз np.where(logo1) створює маску, яка ідентифікує непрозорі пікселі зображення logo1. Операція roi1[np.where(logo1)] = 0 встановлює відповідні пікселі в roi1 на нуль, фактично стираючи логотип із ROI. Після чого накладається зображення logo1 на область roi1. Оператор + виконує додавання зображення, поєднуючи зображення logo1 з областю roi1.

Надалі всі операції виконуються всередині того ж циклу while(). Наступний фрагмент коду обробляє виявлену руку та готує її для подальшого аналізу:

```
if hands:
hand = hands[0]
    x, y, w, h = hand['bbox']
    imgWhite = np.ones((imgSize, imgSize, 3), np.uint8)*255
    imgCrop = img[y-offset:y+h+offset, x-offset:x+w+offset]
    imgCropShape = imgCrop.shape
    aspectRAtio = h/w
```

Перш за все перевіряється чи були виявлені руки в поточному кадрі. Змінна hands — це список виявлених рук, який повертає метод detector.findHands(). Якщо список рук порожній, код пропускає наступний блок коду. Перший рядок всередині умови передбачає, що виявлено лише одну руку. Він витягує першу роздачу зі списку роздач і зберігає її в змінній hand.

Другий рядок витягує координати обмежувальної рамки виявленої руки. Вираз hand['bbox'] повертає кортеж, що містить координати обмежувальної рамки (x, y, w, h), де x і y представляють верхній лівий кут обмежувальної рамки, w представляє її ширину, і h - висоту.

Третій рядок створює порожнє біле зображення із заданими розмірами. Функція np.ones() створює масив одиниць, а операція *255 масштабує кожен елемент до максимального значення 255, що представляє білий колір.

Четвертий рядок обрізає виявлену область руки з вихідного зображення кадру. Значення offset додають відступ навколо обмежувальної рамки, щоб забезпечити захоплення всієї області руки.

П’ятий рядок зберігає форму обрізаного зображення руки в змінній imgCropShape. Вираз imgCrop.shape повертає кортеж, що містить розміри зображення (висота, ширина, канали).

Останній рядок обчислює співвідношення сторін обрізаного зображення руки. Співвідношення сторін — це відношення висоти h зображення до його

ширини w. Це значення буде використано пізніше для зміни розміру обрізаного зображення руки.

Надалі перевіряється чи більше співвідношення сторін за одиницю із виконанням подальших дій:

```
if aspectRatio >1:
k = imgSize / h
wCalc = math.ceil(k*w)
imgResize = cv2.resize(imgCrop, (wCalc, imgSize))
imgResizeShape = imgResize.shape
wGap = math.ceil((imgSize-wCalc)/2)
imgWhite[:, wGap:wCalc+wGap] = imgResize
    else:
         k = imgSize / w
         hCalc = math.ceil(k*h)
          imgResize = cv2.resize(imgCrop, (imgSize, hCalc))
          imgResizeShape = imgResize.shape
hGap = math.ceil((imgSize-hCalc)/2)
imgWhite[hGap:hCalc+hGap, :] = imgResize
```

Цей сегмент коду керує обробкою обрізаного зображення руки, включаючи зміну розміру, центрування та передбачення. Він перевіряє, чи співвідношення сторін обрізаного зображення руки більше за одиницю, що вказує на те, що рука вища, ніж ширша. Якщо так, він обчислює коефіцієнт масштабування на основі співвідношення сторін і бажаної висоти зміненого зображення.

Потім він змінюється розмір обрізаного зображення руки до вказаної висоти, зберігаючи співвідношення сторін. Далі обчислюється проміжок ширини, необхідний для центрування зображення руки зі зміненим розміром у порожньому білому зображенні, і вставляється зображення руки зі зміненим розміром у центральну область.

Якщо ж умова aspectRatio>1 не виконується, то значення будуть обчислені аналогічним чином, але висота та ширина кадру будуть змінені місцями.

Надалі для обох випадків код працює ідентично. Система отримує прогнозовану мітку жесту та її відповідний індекс від класифікатора. Потім друкується інформацію про передбачення та додається прогнозований індекс до списку достовірності передбачення. У тому випадку, якщо список достовірності прогнозів досягає певної довжини, а саме 20 елементів, то визначається найчастіший прогноз за допомогою статичної функції mode() і очищається список:

```
prediction, index = classifier.getPrediction(imgWhite, draw=False)
print(prediction, index)
prediction_confidence.append(index)
print(prediction_confidence)
if len(prediction_confidence) > 20:
confident_index = mode(prediction_confidence)
prediction_confidence = []
```

Після чого проходить перевірка чи увімкнено режим контролю відеодзвінком. Якщо так, то програма перевіряє індекс підвищеної точності передбачень із списком можливих дій. На сегменті коду нижче наведено приклад перевірки чи зчитаний жест є жестом зупинки контролю або жестом показу попереднього слайду презентації. Якщо індекс потрапляє в одну з цих умов, то виконується задана дія:

```
if control == True:
if confident_index == 1: # end control
print('false control')
control = False
          text = 'Gesture control is OFF'
colour = (0, 0, 255)
if confident_index == 2: #prev slide
     keyboard.press_and_release('page down')
```

Після того як всі умови, що активують певні функції відеодзвінка перевірено система перевіряє чи дорівнює індекс нулю, що буде означати те, що користувач хоче активувати режим зчитування жестів з метою віддаленого

керування. Активація такого режиму передбачає не лише присвоєння іншого значення для змінної, що відповідає за контроль, а також і заміну тексту та його кольору на інтерфейсі:

```
if confident_index == 0:
    control = True
    text = 'Gesture control is ON'
    colour = (0, 255, 0)
```

Після виконання перевірок та активації дії, котрій відповідає певний жест необхідно очистити індекс покращеної точності передбачення аби уникнути дублювань дії. Також змінній, що відповідає за виведення останнього зчитаного жесту на екран присвоюється значення списку з мітками класів з індексом підвищеної точності передбачень:

```
if confident_index != None:
    last_recognised_command = labels[confident_index]
    confident_index = None
```

Завершальним етапом сегменту коду, що обмежений умовою наявності руки користувача в кадрі є виведення на екран обмежувальної рамки навколо руки та тексту згори, що показує до якого класу нейромережа віднесла показаний в режимі реального часу жест. Метод cv2.putText() приймає значення, що показують на якому зображенні буде накладено текст, сам текст, що треба вивести, координати верхнього лівого кута тексту, шрифт, розмір, колір та товщину. Метод cv2.rectangle() в свою чергу приймає значення, що покажуть на якому зображенні буде накладено прямокутник, координати лівого верхнього кута, координати правого нижнього кута, колір та товщину ліній прямокутника:

```
cv2.putText(imgOutput, labels[index], (x,y-offset-10), cv2.FONT_HERSHEY_COMPLEX, 1.7, (255,255,225), 2)
cv2.rectangle(imgOutput, (x-offset,y-offset), (x+w+offset, y+h+offset), (255,0,0), 4)
```

Завершальний сегмент коду у циклі while() додає текст що вказує чи увімкнено режим віддаленого контролю чи ні, текстову підказку із останнім розпізнаним жестом для виконання команди, а також виводить на екран саме

фінальне зображення програмного застосунку, котрий містить всі вищезгадані елементи.

```
cv2.putText(imgOutput, text, (50, 75), cv2.FONT_HERSHEY_COMPLEX, 2, colour)
cv2.putText(imgOutput, "Last recognised command is: "+last_recognised_command, (55, 120), cv2.FONT_HERSHEY_COMPLEX, 1, (255,255,255))
cv2.imshow("Image", imgOutput)
cv2.waitKey(1)
```

2.3. Висновки до другого розділу

В другому розділ кваліфікаційної роботи були детально описані всі етапи процесу створення програмного забезпечення для розпізнавання жестів людини з метою керування функціоналом програми для відеодзвінків Zoom.

В якості мови програмування було обрано Python, з використанням бібліотек OpenCV та CVzone. В якості середовища розробки програмного забезпечення було обрано Visual Studio Code.

Перш за все було написано програму, що допоможе зібрати тренувальні дані з вбудованої вебкамери. Для навчання нейронної мережі було використано рішення від компанії Google, що називається Teachable Machine. Даний застосунок отримав широку популярність через різноманітний функціонал та простоту використання. Після навчання мережі її модель було експортовано в Keras, що базується на TensorFlow, та інтегровано в проект. Завершальним етапом розробки була оптимізація алгоритму розпізнавання за допомогою запровадження індексу покращеної точності, що бере за основу математичну функцію моди із 20 послідовних передбачень нейронної мережі. За допомогою індексу покращеної точності система розуміє яку саме функцію для роботи із відеодзвінком в Zoom необхідно активувати. Крім того для уникнення зчитування небажаних жестів було введено два режими роботи, а саме активний та пасивний. За замовчуванням користувач перебує в пасивному режимі і його

жести ніяк не будуть впливати на перебіг відеоконференції, аж поки не буде показано жест “палець догори”, що переводить систему у активний режим. У цьому режимі всі вищезгадані команди можуть бути активовані. Для повернення назад в пасивний режим користувачу необхідно показати жест “палець вниз”.

РОЗДІЛ 3
ПЕРЕВІРКА ПРАЦЕЗДАТНОСТІ СИСТЕМИ ТА ПОРІВНЯННЯ З ІСНУЮЧИМИ РІШЕННЯМИ

3.1. Опис роботи програми

Інтерфейс програми являє собою зображення із вбудованої вебкамери, що транслюється в режимі реального часу, із доданими графічними елементами, а саме:

- текстовий рядок, що показує чи увімкнений режим контролю для відеодзвінку;
- текстовий рядок, що показує останній розпізнаний жест;
- рамка навколо руки користувача (якщо рука у кадрі) із текстовим рядком, що показує передбачення нейромережі;
- список із 10 зображень, на яких можна побачити які жести треба показати для виконання тої чи іншої дії.

Одразу після запуску програми користувач побачить, що контроль жестами наразі вимкнено (“Gesture control is OFF”), та відсутність відомостей про останній розпізнаний жест (“Last recognised command is:”). Приклад наведено на рис. 3.1.

Рис. 3.1. Вигляд початкового інтерфейсу програми одразу після запуску

Як зазначалось у другому розділі, програма підтримує 10 жестів для управління відеодзвінком, а саме:

- увімкнути режим віддаленого керування відеодзвінком;
- вимкнути режим віддаленого керування відеодзвінком;
- показати попередній слайд презентації;
- показати наступний слайд презентації;
- увімкнути чи вимкнути локальний запис відеоконференції;
- увімкнути чи вимкнути запис відеоконференції у хмарне сховище;
- увімкнути чи вимкнути мікрофон;
- увімкнути чи вимкнути камеру;
- увімкнути чи вимкнути транслювання власного екрану;
- показати чи сховати список присутніх на відеоконференції.

Саме такий набір функцій є достатнім для покриття базових потреб для використання програми з метою покращення користувацького досвіду у відеоконференціях. Як зазначалось раніше, для зручності користувача список усіх доступних жестів постійно знаходиться на в нижній частині екрану (рис. 3.2).

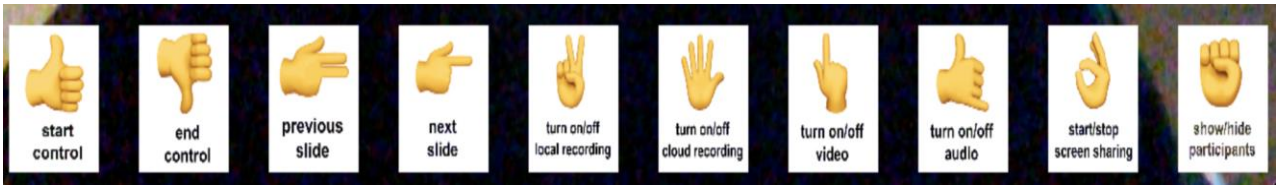

Рис. 3.2. Статична підказка із набором жестів та їх функцій

Опис роботи програми слід почати з того, що для уникнення зчитування випадкових жестів без відома користувача було прийнято рішення ввести режим блокування підсистеми активації команд. Таким чином людина може активно жестикулювати без страху активації небажаної функції. Для того, аби перейти в режим зчитування команд від підсистеми розпізнавання жестів необхідно показати й затримати на кілька секунд жест “start control”. Після показу цього жесту текст у верхній частині вікна буде змінено із “Gesture control is OFF” на

“Gesture control is ON”, а також текстове поле, що містить відомості про останній успішно зчитаний жест оновить значення на “start control”. Приклад наведено на рис. 3.3. При цьому подальші жести користувача вже будуть активовувати ту чи іншу команду відеоконференції у Zoom.

Рис. 3.3. Інтерфейс програми у разі показу користувачем жесту, що відповідає за дозвіл віддаленим керуванням

Надалі буде наведено приклади розпізнавання кожного із доступних в програмі жестів активації саме функцій програми Zoom. Одними з найважливіших жестів даної програми є ті, що дозволяють перемикати слайди на презентації, яка демонструється з екрану користувача під час відеоконференції. Користувач має змогу перемикати слайди вперед та назад без необхідності сидіти біля клавіатури. Ці та інші функції будуть особливо корисні людям, що звикли проводити відеоконференції стоячи, або ж на невеликій відстані від елементів керування. Так наприклад викладач школи чи університету може ходити кімнатою протягом відеолекції, як це було звично під час офлайн

занять. Приклади розпізнавання жестів, що відповідають за перемикання слайдів назад та вперед наведено на рис. 3.4 та 3.5 відповідно.

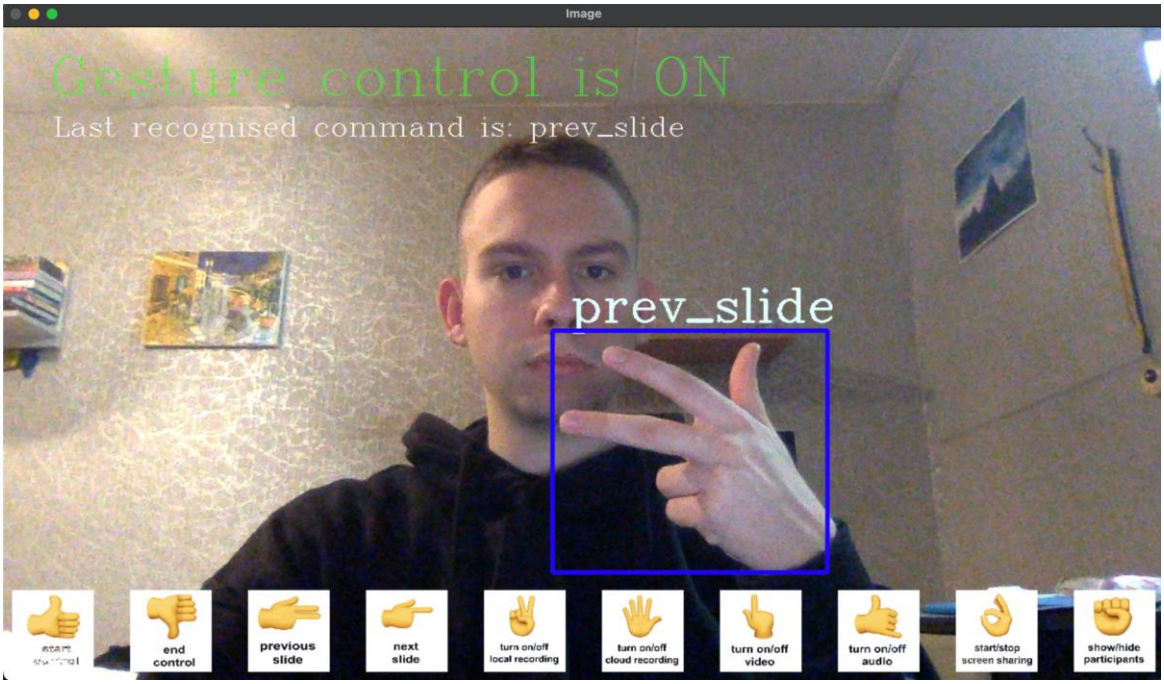

Рис. 3.4. Інтерфейс програми у разі показу користувачем жесту, що відповідає за вибір попереднього слайду презентації

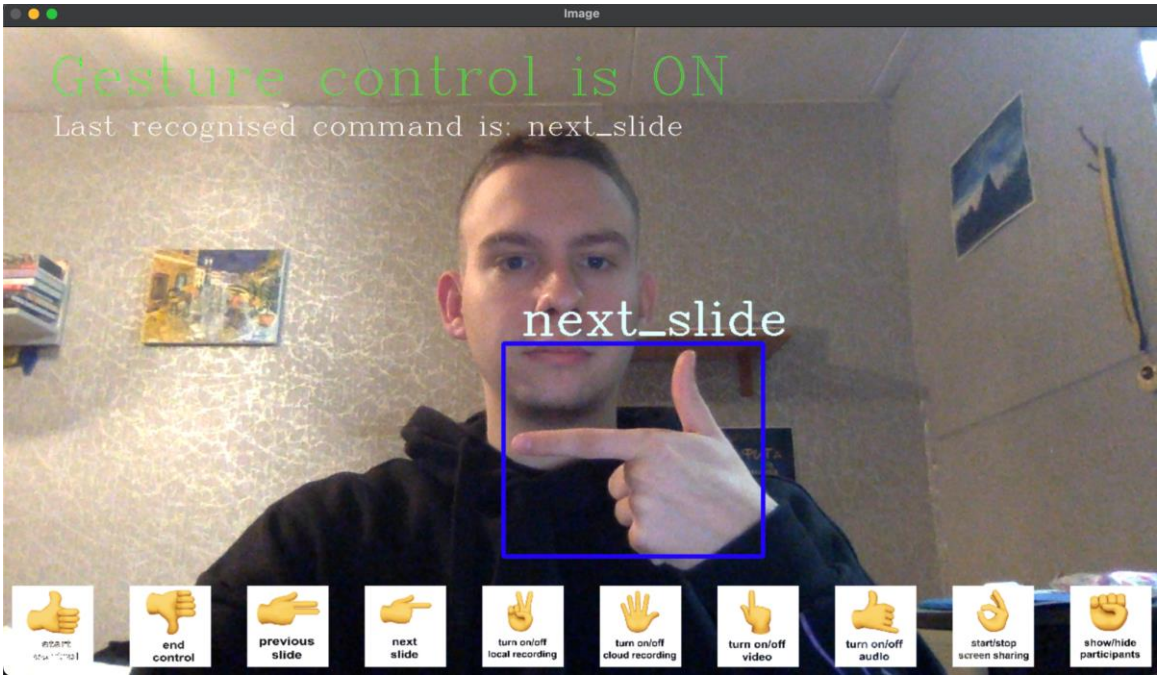

Рис. 3.5. Інтерфейс програми у разі показу користувачем жесту, що відповідає за вибір наступного слайду презентації

Наступна група жестів відповідає за записування відеоконференції для подальшого перегляду у будь-який час. Ця функція дуже корисна, особливо для навчання та роботи, адже інколи людина не може бути фізично присутня на конференції, саме у зазначений час, але інформація, що озвучувалась була б цінною. У програмі Zoom існує два способи запису конференцій: локально на комп’ютер у вигляді відео-файлу або у хмару, де доступ буде здійснюватись за посиланням. Локальний запис є хорошим варіантом, якщо користувач хоче мати повний контроль над файлом запису та якщо його турбує конфіденційність даних. Однак локальні записи можуть бути великими і не підходять для обміну з іншими. Хмарний запис зберігає файл запису в сховищі Zoom і є хорошим варіантом у разі необхідності легко поділитися записом з іншими або якщо недостатньо місця для зберігання на комп’ютері. Однак записи в хмарі можуть бути не такими безпечними, як локальні записи.

Приклад розпізнавання жесту для початку/кінця локального або хмарного запису наведено на рис. 3.6 та 3.7 відповідно.

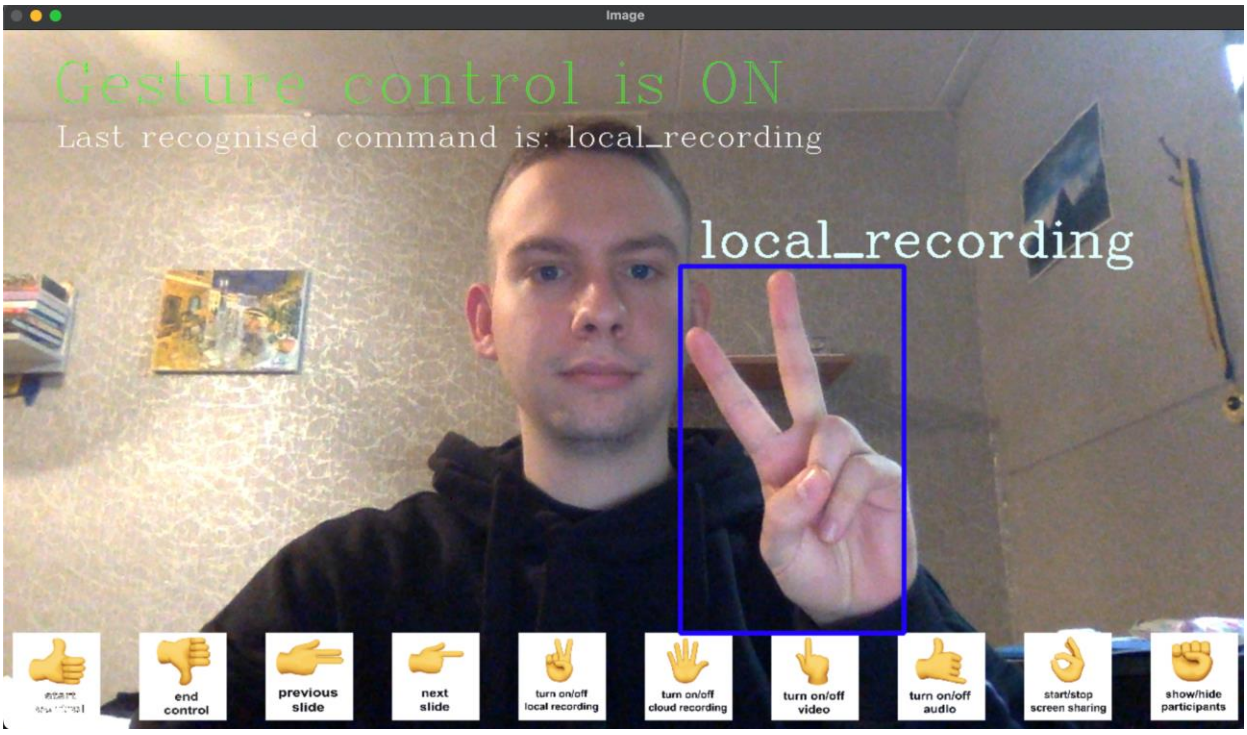

Рис. 3.6. Інтерфейс програми у разі показу користувачем жесту, що відповідає за початок/кінець локального запису відеоконференції

Рис. 3.7. Інтерфейс програми у разі показу користувачем жесту, що відповідає за початок/кінець запису відеоконференції у хмарне сховище

Також програма підтримує жести для активації та деактивації мікрофону і камери користувача. Ці функції є дуже важливими, адже жодна людина не застрахована від небажаних звуків чи обставин, що можуть потрапити на камеру. Саме тому критично важливо було додати ці функцій, щоб у разі чого людина могла спокійно управляти видимістю свого зображення та наявністю увімкненого мікрофону віддалено, без потреби підходити до клавіатури. Оскільки комбінації клавіш у Zoom однакові як для вимкнення так і увімкнення мікрофону й камери, для відміни дії необхідно буде просто показати жест ще один раз.

В якості жестів було обрано звичний жест “телефону” для взаємодії з мікрофоном, а також жест, що утворює між пальцями руки прямий кут, що може асоціюватись з рамкою зображення камери. Таким чином користувачу буде легше запам’ятати які саме жести треба показувати у разі необхідності виконання вищезгаданих операцій. Приклади розпізнавання жестів для взаємодії із камерою та мікрофоном наведені на рис. 3.8 та 3.9 відповідно.

Рис. 3.8. Інтерфейс програми у разі показу користувачем жесту, що відповідає за увімкнення/вимкнення камери

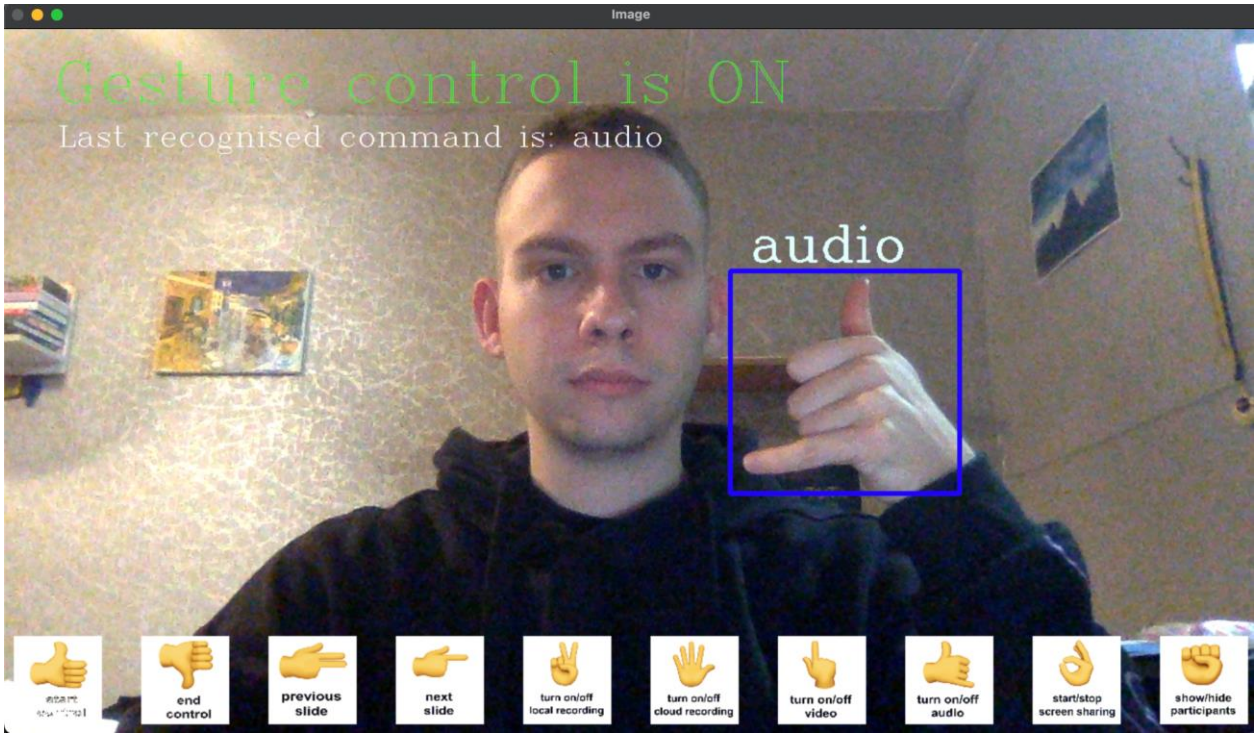

Рис. 3.9. Інтерфейс програми у разі показу користувачем жесту, що відповідає за увімкнення/вимкнення мікрофону

Наступна функція, що може бути активована або деактивована за допомогою жестів - це початок та завершення демонстрації власного екрану. Ця функція є дуже корисною адже переважна більшість відеоконференцій не обходиться без показу слайдів презентації чи сторінок книги.

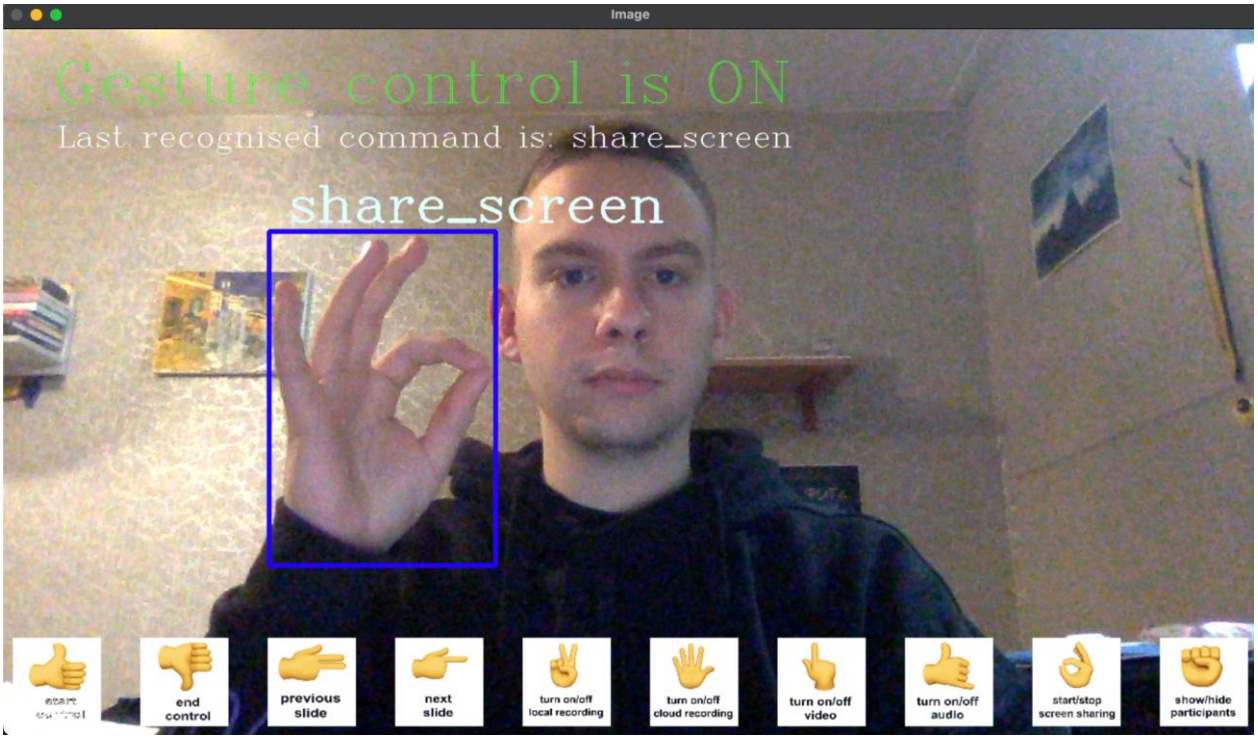

Рис. 3.10. Інтерфейс програми у разі показу користувачем жесту, що відповідає за початок/кінець демонстрації власного екрану

Остання функція, що входить в демонстраційних базовий набір для цього проекту надає можливість виведення на екран списку присутніх на відеоконференції. Повторне зчитування жесту ховає список присутніх з екрану користувача.

Така функція може бути особлива корисна викладачам в освітніх закладах, адже можна відмітити присутність учнів чи студентів у журналі, відкривши відповідний віджет за допомогою простого жесту, тобто без використання клавіатури. Приклад такого жесту наведено на рис. 3.11.

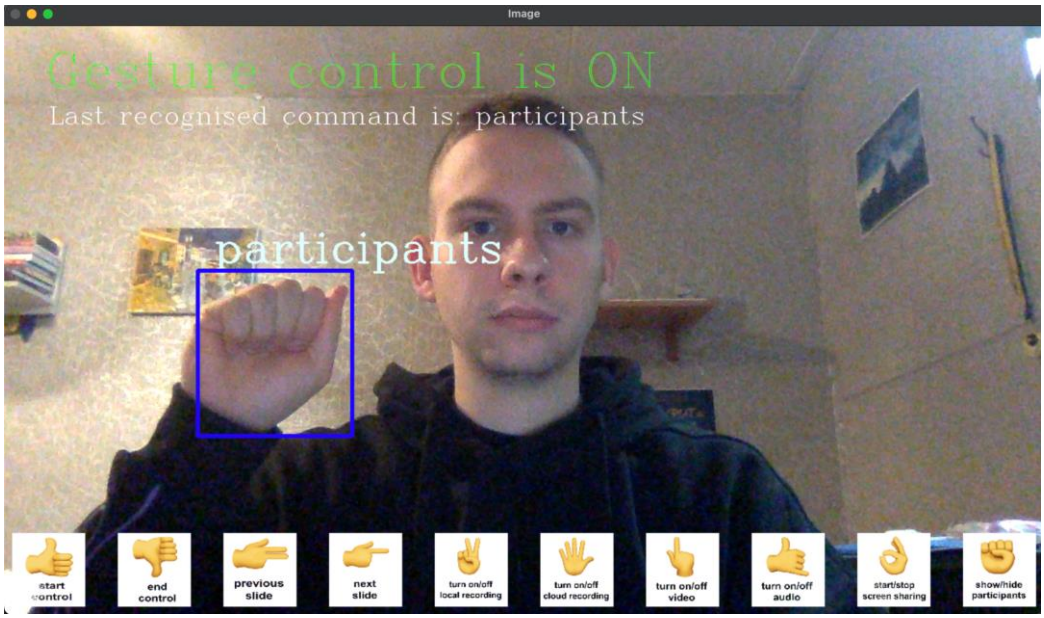

Рис. 3.11. Інтерфейс програми у разі показу користувачем жесту, що відповідає за показ/приховування списку присутніх на відеоконференції

Як зазначалось на початку розділу, програма має активний та пасивний режим активації команд для уникнення зчитування випадкових жестів. Для переходу із активного режима в пасивний необхідно показати жест “палець донизу”. Приклад розпізнавання жесту, що сповіщає про перехід у пасивний режим наведено на рис. 3.12.

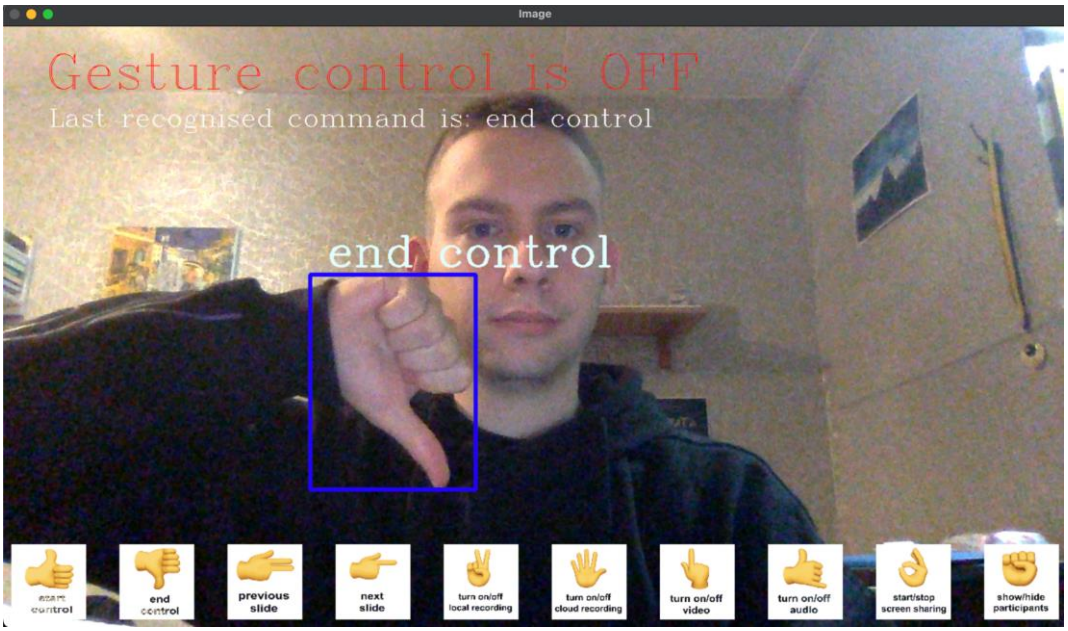

Рис. 3.12. Інтерфейс програми у разі показу користувачем жесту, що відповідає за заборону віддаленим керуванням

3.2. Порівняння із наявним функціоналом програми Zoom

Як зазначалося в першому розділі, програма Zoom нещодавно випустила функцію розпізнавання двох жестів у відкритий доступ. Нараз підтримувані жести обмежуються лише “підняттям руки” та виставленням реакції “палець догори” (рис. 3.13).

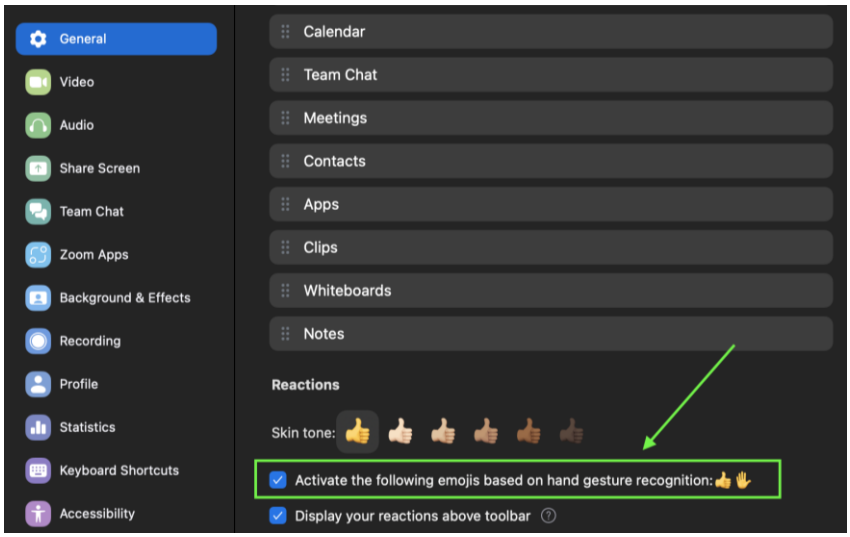

Рис. 3.13. Вікно налаштувань програми Zoom, де можна увімкнути або вимкнути розпізнавання двох жестів

Наявність такого функціоналу у загальнодоступній версії програми свідчить про актуальність даного проекту та потенційну зацікавленість самих розробників та клієнтів у можливості віддаленого керування деякими функціями програми.

Після увімкнення жестового розпізнавання у вікні налаштувань, користувач зможе протестувати коректність роботи вбудованих алгоритмів розпізнавання безпосередньо у дзвінку. Після показу жесту користувач побачить маленький значок із передбачуваним жестом, що знаходиться всередині таймера.

Таймер не дасть користувачу показати небажаний жест випадково. У разі зміни жесту або прибирання руки із кадру - таймер буде скинуто і активації дії не відбудеться. Приклад інтерфейсу вбудованого розпізнавання жестів програмою Zoom наведено на рис. 3.14 та 3.15.

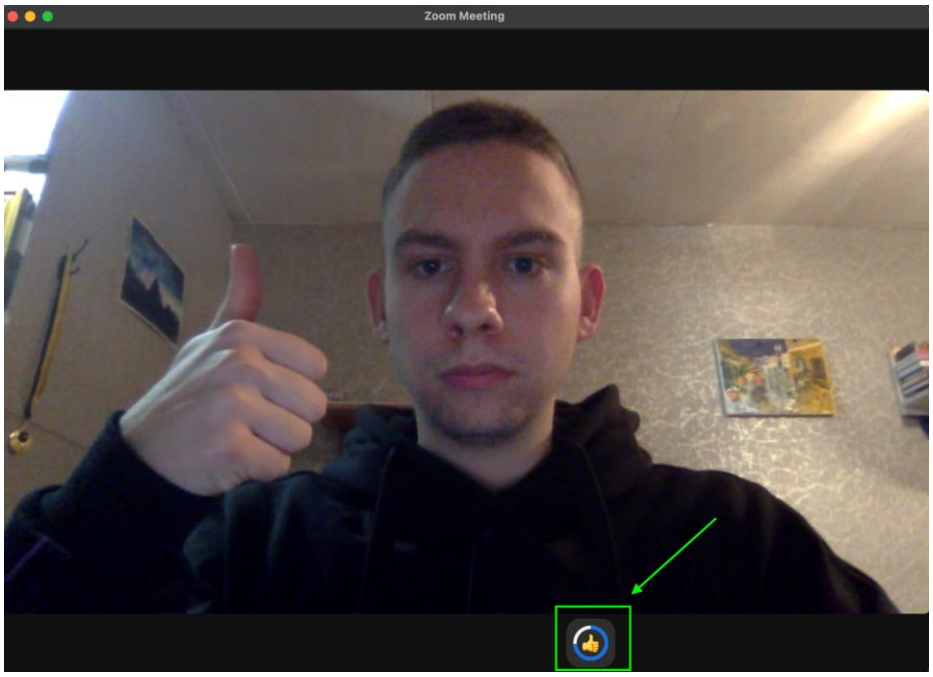

Рис. 3.14. Інтерфейс програми у разі показу користувачем жесту, що відповідає за реакцію “палець догори”

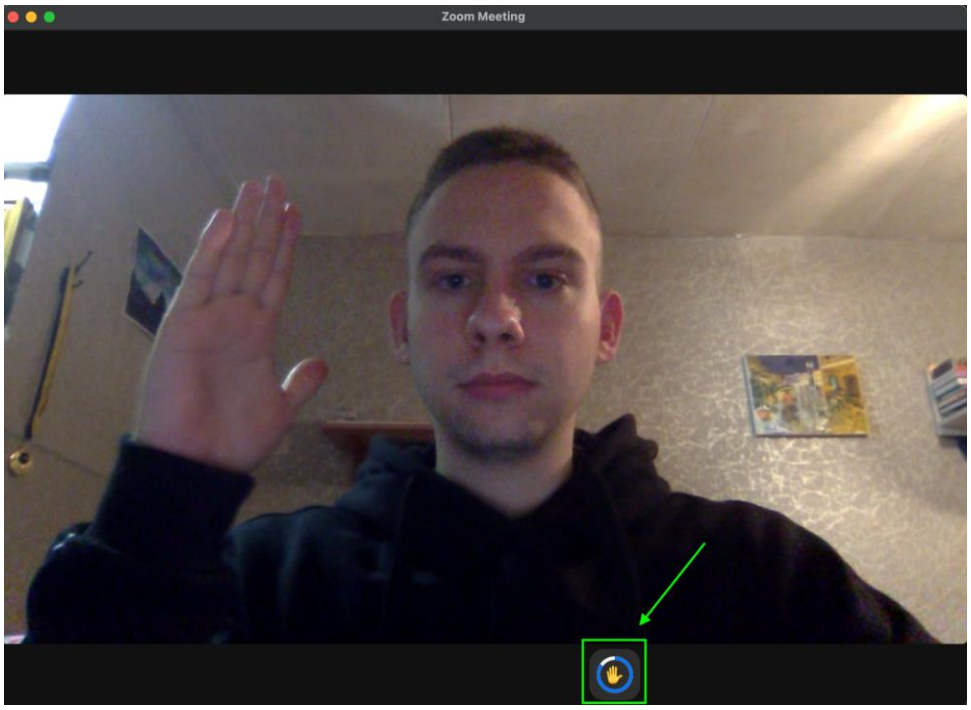

Рис. 3.15. Інтерфейс програми у разі показу користувачем жесту, що відповідає за підняття руки

Порівнявши нативне розпізнавання жестів та те, що реалізується за допомогою окремої програми можна стверджувати, що переваги нативного розпізнавання полягають в тому, що:

- це вбудована функція, а тому не треба додатково завантажувати нову програму;
- таймер із жестом, що розпізнається показано напряму у конференції, а не в окремому вікні.

В свою чергу програма, що реалізована в межах даного проекту має такі переваги:

- кількість підтримуваних жестів у 5 разів більша;
- жести корисні не тільки для слухачів, а й для спікера;
- перемикання активного та пасивного режиму розпізнавання відбувається за допомогою певного жесту, а не кнопки у налаштуваннях;
- список жестів, інформація про поточний режим управління та останній зчитаний жест завжди знаходиться на екрані додатку.

3.3. Висновки до третього розділу

В третьому розділі кваліфікаційної роботи було описано принцип роботи програмного комплексу, наявні функції для поліпшення користувацького досвіду при проведенні відеоконференцій, а також наведено порівняння із нативною підсистемою розпізнавання жестів в програмі Zoom.

Написана програма значно поліпшує користувацький досвід при проведенні відеоконференцій, порівняно із вбудованим функціоналом, адже за допомогою жестів людина може увімкнути та вимкнути мікрофон чи відеокамеру, надати доступ до свого екрану, гортати слайди презентації вперед та назад, починати і завершувати локальний запис конференції чи запис у хмарне сховище, а також виводити на екран вікно з присутніми на відеодзвінку. Цей набір команд є оптимальним для комфортного користування і покриває найуживаніші функції застосунку Zoom.

Також до переваг написаної системи перед вбудованим функціоналом для розпізнавання в Zoom можна віднести зручний інтерфейс, що містить підказки з доступними до розпізнавання жестами та функціями, що вони активують, а також показники поточного режиму активності та відомості про останній зчитаний жест.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було виконано дослідження методів розпізнавання жестів людини в режимі реального часу та створено інформаційну систему для організації віддаленого керування відеодзвінком на базі розпізнавання жестів.

Для досягнення мети роботи були виконані наступні задачі:

- проаналізувати існуючі підходи для розпізнавання жестів та керування ними;
- провести аналіз існуючих на ринку систем розпізнавання, виявити їхні сильні та слабкі сторони;
- проаналізувати основні засоби для розробки і вибрати оптимальні з них;
- написати програмний код системи розпізнавання жестів;
- інтегрувати систему розпізнавання із командами керування відеодзвінком в програмі Zoom.

Розроблена система надає змогу користувачу активувати найважливіші функції програми для відеозв’язку Zoom за допомогою жестів руки. Зокрема:

- увімкнути режим віддаленого керування відеодзвінком;
- вимкнути режим віддаленого керування відеодзвінком;
- показати попередній слайд презентації;
- показати наступний слайд презентації;
- увімкнути чи вимкнути локальний запис відеоконференції;
- увімкнути чи вимкнути запис відеоконференції у хмарне сховище;
- увімкнути чи вимкнути мікрофон;
- увімкнути чи вимкнути камеру;
- увімкнути чи вимкнути транслювання власного екрану;
- показати чи сховати список присутніх на відеоконференції.

Розроблена система дозволяє користувачу відходити від робочого місця, при цьому не втрачаючи можливість керування відеодзвінком, головною умовою

є присутність людини у кадрі, відносно недалека відстань до камери і гарне освітлення.

На відміну від вбудованого розпізнавання жестів в програмі Zoom, розроблена система підтримує в 5 разів більше жестів, зокрема дуже необхідних для різних груп користувачів, має наглядний інтерфейс з підказками та покращену точність розпізнавання.

ПЕРЕЛІК ПОСИЛАНЬ

1. What is Object Recognition and Where to Use [Електронний ресурс] - Режим доступу: https://www.plugger.ai/blog/what-is-object-recognition-and-where-to-use

2. Deep Learning For Computer Vision: Essential Models and Practical Real-World Applications [Електронний ресурс] - Режим доступу: https://opencv.org/blog/deep-learning-with-computer-vision/

3. A Gentle Introduction To Neural Networks Series [Електронний ресурс] - Режим доступу: https://towardsdatascience.com/a-gentle-introduction-to-neural-networks-series-part-1-2b90b87795bc

4. Субботін С. О. Нейронні мережі: теорія та практика: навч. посіб. Житомир: Вид. О.О. Євенок, 2020, 44 с.

5. Субботін С. О. Нейронні мережі: теорія та практика: навч. посіб. Житомир: Вид. О.О. Євенок, 2020, 46 с.

6. Модель роботи згорткових нейронних мереж [Електронний ресурс] - Режим доступу: https://www.v7labs.com/blog/image-recognition-guide

7. Субботін С. О. Нейронні мережі: теорія та практика: навч. посіб. Житомир: Вид. О.О. Євенок, 2020, 88 с.

8. What are recurrent neural networks? [Електронний ресурс] - Режим доступу: https://www.ibm.com/topics/recurrent-neural-networks

9. А. С. Довбиш, І. В. Шелехов Основи теорії розпізнавання образів. Частина 1. Суми, 2015, 5 с.

10. Myhajlo Lobur, Raed Sahawneh, Ahmad Al Khateb. Information Features of Palms Images Used for Palm’s Recognition. Proceedings of the International conference TCSET’2004. – Lviv-Slavsko, 2004. 251 с.

11. Н. Є. Кулішова, Д. А. Авдєєв. Сучасні методи вирішення проблем розпізнавання жестів людини в реальному часі // Біоніка інтелекту: наук.-техн. журнал. – 2016. – No 1(86). 103 с.

12. Н. Є. Кулішова, Д. А. Авдєєв. Сучасні методи вирішення проблем розпізнавання жестів людини в реальному часі // Біоніка інтелекту: наук.-техн. журнал. – 2016. – No 1(86). 104 с.

13. Н. Є. Кулішова, Д. А. Авдєєв. Сучасні методи вирішення проблем розпізнавання жестів людини в реальному часі // Біоніка інтелекту: наук.-техн. журнал. – 2016. – No 1(86). 106 с.

14. Project Gesture [Електронний ресурс] - Режим доступу: https://www.microsoft.com/en-us/research/project/gesture/

15. Google Meet adds gesture detection to recognize when you raise your hand on a group video call [Електронний ресурс] - Режим доступу: https://www.theverge.com/2023/11/21/23971246/google-meet-gesture-recognition-hand-raise-new-feature

16. About the Gesture Recognizer State Machine [Електронний ресурс] - Режим доступу: https://developer.apple.com/documentation/uikit/touches_presses_and_gestures/implementing_a_custom_gesture_recognizer/about_the_gesture_recognizer_state_machine

17. Реакції у відеоповідомленнях в Telegram [Електронний ресурс] - Режим доступу: https://nnews.com.ua/u-telegram-na-ios-17-z-yavylysya-reaktsiyi-u-videopovidomlennyah.html

18. Using gesture recognition in Zoom [Електронний ресурс] - Режим доступу: https://support.Zoom.us/hc/en-us/articles/4407537406093-Using-gesture-recognition

19. Офіційна документація CVZone [Електронний ресурс] - Режим доступу: - https://github.com/cvzone/cvzone

20. Офіційна документація OpenCV [Електронний ресурс] - Режим доступу: https://pypi.org/project/opencv-python/

21. Відстеження ознак через mediapipe [Електронний ресурс] - Режим доступу: https://developers.google.com/mediapipe/solutions/vision/gesture_recognizer

22. Convolutional Neural Network (CNN) in Tensorflow [Електронний ресурс] - Режим доступу: https://www.tensorflow.org/tutorials/images/cnn

23. VSCode [Електронний ресурс] - Режим доступу: - https://code.visualstudio.com/

24. Teachable Machine [Електронний ресурс] - Режим доступу: https://teachablemachine.withgoogle.com/

25. About Keras 3 [Електронний ресурс] - Режим доступу: - https://keras.io/

ДОДАТОК А

ЛІСТИНГ ПРОГРАМИ

Файл main.py:

```
import cv2
from cvzone.HandTrackingModule import HandDetector
from cvzone.ClassificationModule import Classifier
import numpy as np
import math
import time
import tensorflow
import os
import time
import keyboard
from statistics import mode

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)
classifier = Classifier('model/keras_model.h5','model/labels.txt')
offset = 20
imgSize = 1000
labels = ['start control', 'end control', 'prev_slide', 'next_slide', 'local_recording',
'cloud_recording', 'video', 'audio', 'share_screen', 'participants']
size = 90
logo1 = cv2.imread('images/participants.png')
logo1 = cv2.resize(logo1, (size, size))
logo2 = cv2.imread('images/share_screen.png')
logo2 = cv2.resize(logo2, (size, size))
logo3 = cv2.imread('images/audio.png')
logo3 = cv2.resize(logo3, (size, size))
logo4 = cv2.imread('images/video.png')
logo4 = cv2.resize(logo4, (size, size))
logo5 = cv2.imread('images/cloud_recording.png')
logo5 = cv2.resize(logo5, (size, size))
logo6 = cv2.imread('images/local_recording.png')
logo6 = cv2.resize(logo6, (size, size))
logo7 = cv2.imread('images/next_slide.png')
logo7 = cv2.resize(logo7, (size, size))
logo8 = cv2.imread('images/prev_slide.png')
logo8 = cv2.resize(logo8, (size, size))
logo9 = cv2.imread('images/end.png')
logo9 = cv2.resize(logo9, (size, size))
logo10 = cv2.imread('images/start.png')
logo10 = cv2.resize(logo10, (size, size))
control = False
prediction_confidence = []
confident_index = None
last_recognised_command = ''
text = 'Gesture control is OFF'
colour = (0, 0, 255)
while(True):
    success, img = cap.read()
    imgOutput = img.copy()
    hands, img= detector.findHands(img)

    roi1 = imgOutput[-size-10:-10, -size-10:-10]
    roi1[np.where(logo1)] = 0
    roi1 += logo1
    roi2 = imgOutput[-size-10:-10, -size-140:-140]
    roi2[np.where(logo2)] = 0
    roi2 += logo2
    roi3 = imgOutput[-size-10:-10, -size-270:-270]
    roi3[np.where(logo3)] = 0
    roi3 += logo3
```

```
roi4 = imgOutput[-size-10:-10, -size-400:-400]
roi4[np.where(logo4)] = 0
roi4 += logo4
roi5 = imgOutput[-size-10:-10, -size-530:-530]
roi5[np.where(logo5)] = 0
roi5 += logo5
roi6 = imgOutput[-size-10:-10, -size-660:-660]
roi6[np.where(logo6)] = 0
roi6 += logo6
roi7 = imgOutput[-size-10:-10, -size-790:-790]
roi7[np.where(logo7)] = 0
roi7 += logo7
roi8 = imgOutput[-size-10:-10, -size-920:-920]
roi8[np.where(logo8)] = 0
roi8 += logo8

roi9 = imgOutput[-size-10:-10, -size-1050:-1050]
roi9[np.where(logo9)] = 0
roi9 += logo9
roi10 = imgOutput[-size-10:-10, -size-1180:-1180]
roi10[np.where(logo10)] = 0
roi10 += logo10
if hands:
    hand = hands[0]
    x,y,w,h = hand['bbox']
    imgWhite = np.ones((imgSize, imgSize, 3), np.uint8)*255
    imgCrop = img[y-offset:y+h+offset, x-offset:x+w+offset]

    imgCropShape = imgCrop.shape
    aspectRAtio = h/w
    if aspectRAtio >1:
        k = imgSize / h
        wCalc = math.ceil(k*w)
        imgResize = cv2.resize(imgCrop, (wCalc, imgSize))
        imgResizeShape = imgResize.shape
        wGap = math.ceil((imgSize-wCalc)/2)
        imgWhite[:, wGap:wCalc+wGap] = imgResize
        prediction, index = classifier.getPrediction(imgWhite, draw=False)
        print(prediction, index)
        prediction_confidence.append(index)
        print(prediction_confidence)
        if len(prediction_confidence) > 20:
            confident_index = mode(prediction_confidence)
            prediction_confidence = []
        if control == True:
            if confident_index == 1: # end control
                print('false control')
                control = False
                text = 'Gesture control is OFF'
                colour = (0, 0, 255)

            if confident_index == 2: #prev slide
                keyboard.press_and_release('page down')
            if confident_index == 3: #next slide
                keyboard.press_and_release('page up')

            if confident_index == 4: #local recording
                keyboard.press_and_release('shift+command+R')

            if confident_index == 5: #cloud recording
                keyboard.press_and_release('shift+command+c')

            if confident_index == 6: #video
                keyboard.press_and_release('shift+command+v')
            if confident_index == 7: #audio
                keyboard.press_and_release('shift+command+a')
            if confident_index == 8: #screen sharing
                keyboard.press_and_release('shift+command+s')
                keyboard.press_and_release('enter')
```

```
                if confident_index == 9: #participants
                    keyboard.press_and_release('command+u')
            if confident_index == 0:
                control = True
                text = 'Gesture control is ON'
                colour = (0, 255, 0)

        else:
            k = imgSize / w
            hCalc = math.ceil(k*h)
            imgResize = cv2.resize(imgCrop, (imgSize, hCalc))
            imgResizeShape = imgResize.shape
            hGap = math.ceil((imgSize-hCalc)/2)
            imgWhite[hGap:hCalc+hGap, :] = imgResize

            prediction, index = classifier.getPrediction(imgWhite, draw=False)
            print(prediction, index)
            prediction_confidence.append(index)
            print(prediction_confidence)
            if len(prediction_confidence) > 20:
                confident_index = mode(prediction_confidence)
                prediction_confidence = []
            if control == True:
                if confident_index == 1: # end control
                    print('false control')
                    control = False
                    text = 'Gesture control is OFF'
                    colour = (0, 0, 255)
                if confident_index == 2: #prev slide
                    keyboard.press_and_release('page down')
                if confident_index == 3: #next slide
                    keyboard.press_and_release('page up')
                if confident_index == 4: #local recording
                    keyboard.press_and_release('shift+command+R')
                if confident_index == 5: #cloud recording
                    keyboard.press_and_release('shift+command+c')
                if confident_index == 6: #video
                    keyboard.press_and_release('shift+command+v')
                if confident_index == 7: #audio
                    keyboard.press_and_release('shift+command+a')
                if confident_index == 8: #screen sharing
                    keyboard.press_and_release('shift+command+s')
                    keyboard.press_and_release('enter')

                if confident_index == 9: #participants
                    keyboard.press_and_release('command+u')
            if confident_index == 0:
                control = True
                text = 'Gesture control is ON'
                colour = (0, 255, 0)

        if confident_index != None:
            last_recognised_command = labels[confident_index]
        confident_index = None

        cv2.putText(imgOutput, labels[index], (x,y-offset-10), cv2.FONT_HERSHEY_COMPLEX, 1.7,
(255,255,225), 2)
        cv2.rectangle(imgOutput, (x-offset,y-offset), (x+w+offset, y+h+offset), (255,0,0), 4)
        #cv2.imshow("ImageCrop", imgCrop)
        #cv2.imshow("ImageWhite", imgWhite)

    cv2.putText(imgOutput, text, (50, 75), cv2.FONT_HERSHEY_COMPLEX, 2, colour)
    cv2.putText(imgOutput, "Last recognised command is: "+last_recognised_command, (55, 120),
cv2.FONT_HERSHEY_COMPLEX, 1, (255,255,255))
    cv2.imshow("Image", imgOutput)
    cv2.waitKey(1)
```

Файл dataCollection.py:

```
import cv2
from cvzone.HandTrackingModule import HandDetector
import numpy as np
import math
import time

cap = cv2.VideoCapture(0)
detector = HandDetector(maxHands=1)

offset = 20
imgSize = 500

folder = 'data/participants'
counter = 0

while(True):
   success, img = cap.read()
   hands, img = detector.findHands(img)
   if hands:
       hand = hands[0]
       x,y,w,h = hand['bbox']

       imgWhite = np.ones((imgSize, imgSize, 3), np.uint8)*255
       imgCrop = img[y-offset:y+h+offset, x-offset:x+w+offset]

       imgCropShape = imgCrop.shape

       aspectRAtio = h/w
       if aspectRAtio >1:
           k = imgSize / h
           wCalc = math.ceil(k*w)
           imgResize = cv2.resize(imgCrop, (wCalc, imgSize))
           imgResizeShape = imgResize.shape
           wGap = math.ceil((imgSize-wCalc)/2)
           imgWhite[:, wGap:wCalc+wGap] = imgResize

       else:
           k = imgSize / w
           hCalc = math.ceil(k*h)
           imgResize = cv2.resize(imgCrop, (imgSize, hCalc))
           imgResizeShape = imgResize.shape
           hGap = math.ceil((imgSize-hCalc)/2)
           imgWhite[hGap:hCalc+hGap, :] = imgResize

       # cv2.imshow("ImageCrop", imgCrop)
       # cv2.imshow("ImageWhite", imgWhite)

   cv2.imshow("Image", img)

   key = cv2.waitKey(1)
   if key == ord('s'):
       counter += 1
       cv2.imwrite(f'{folder}/Image_{time.time()}.jpg', imgWhite)
       print(counter)
```

ВІДГУК КЕРІВНИКА

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ «ДНІПРОВСЬКА ПОЛІТЕХНІКА»

Факультет інформаційних технологій

Кафедра програмного забезпечення комп'ютерних систем

ВІДГУК

Наукового керівника Спірінцева В'ячеслава Васильовича, доц. каф. ПЗКС

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

на магістерську роботу

студента Басараба Данила Руслановича

(прізвище, ім'я, по батькові)

курсу II групи 122м-22-4

спеціальності 122 Комп'ютерні науки

освітньої програми Комп'ютерні науки

на тему Дослідження методів розпізнавання жестів людини в режимі реального часу для організації віддаленого керування відеодзвінком

Актуальність теми Представлена магістерська кваліфікаційна робота присвячена дослідженню методів розпізнавання жестів й інтеграції даної технології з програмою для відеодзвінків Zoom. На поточний момент програма Zoom дуже активно використовуються користувачами з усього світу, тому покращення користувацького досвіду за рахунок зчитування жестів та активації певних функцій керування є актуальним напрямком дослідження.

Мета досліджень Полягає у вдосконаленні методів віддаленого керування відеодзвінками в програмі Zoom за допомогою зчитування жестів рук у відеопотоках в режимі реального часу з використанням нейронних мереж.

Коротка характеристика розділів роботи Перший розділ роботи присячений огляду предметної області та можливих шляхів вирішення поставлених завдань.

Другий розділ містить відомості про обрані технології для реалізації проекту, а також докладний опис всіх етапів розробки програми, що активує задані функції програми у разі зчитування відповідного жеста користувача. Третій розділ присвячений опису функцій розробленого додатку, його тестуванню, а також порівнянню із аналогом, що вбудовано в програму Zoom

Практичне значення роботи Полягає в тому, що розроблену систему можна використовувати для покращення користувацького досвіду на відеоконференціях в Zoom, адже за допомогою визначених жестів можна увімкнути або вимкнути мікрофон чи камеру, почати демонстрацію екрану, гортати слайди презентації, починати та завершувати локальний чи хмарний запис, виводити список присутніх на відеодзвінку без використання клавіатури чи миші.

Зауваження та недоліки Розроблена системи має власний користувацький інтерфейс, а отже не інтегрована напряму в застосунок Zoom, проте цей аспект не заважає системі повноцінно функціонувати.

Висновки та оцінка Магістром було проведено аналіз та порівняння можливих методів розв'язання поставленої задачі та обрано оптимальний варіант. Під час виконання магістерської кваліфікаційної роботи студент Басараб Д.Р. проявив себе грамотним, кваліфікованим спеціалістом здатним приймати самостійно складні технічні рішення. Вважаю, що магістерська кваліфікаційна робота заслуговує оцінку 85 «добре», а Басараб Д.Р. – присвоєння кваліфікації «магістра» з комп'ютерних наук.

Науковий керівник Спірінцев В.В., доц. каф. ПЗКС

(прізвище, ім'я, по батькові, посада, місце роботи)

«___»_____20__ р. _____ (підпис)

РЕЦЕНЗІЯ
на кваліфікаційну роботу

студента Басараба Данила Руслановича

(прізвище, ім'я, по батькові)

курсу II групи 122м-22-4

кафедри програмного забезпечення комп'ютерних систем

спеціальності 122 Комп'ютерні науки

Тема роботи Дослідження методів розпізнавання жестів людини в режимі реального часу для організації віддаленого керування відеодзвінком

Стисла характеристика розділів роботи Перший розділ роботи присячений огляду предметної області та можливих шляхів вирішення поставлених завдань. Другий розділ містить відомості про обрані технології для реалізації проекту, а також докладний опис всіх етапів розробки програми, що активує задані функції програми у разі зчитування відповідного жеста користувача. Третій розділ присвячений опису функцій розробленого додатку, його тестуванню, а також порівнянню із аналогом, що вбудовано в програму Zoom

Пропозиції, внесені студентом, рівень їх наукового обґрунтування В даній кваліфікаційній роботі студентом надано декілька пропозицій стосовно вирішення поставлених завдань. Кожна з пропозицій була обґрунтована та підкріплена науковими даними.

Практичне значення роботи Полягає в тому, що розроблену систему можна використовувати для покращення користувацького досвіду на відеоконференціях в Zoom, адже за допомогою визначених жестів можна увімкнути або вимкнути мікрофон чи камеру, почати демонстрацію екрану, гортати слайди презентації, починати та завершувати локальний чи хмарний запис, виводити список присутніх на відеодзвінку без використання клавіатури чи миші.

Якість оформлення роботи Магістерська кваліфікаційна робота, яку подано на рецензію, виконана у повному обсязі у встановлений термін. Робота є добре

структурованою та достатньо проілюстрованою. Викладена основна суть проблеми, що вирішується в ході виконання роботи, і шляхів її вирішення.

Недоліки в роботі Розроблена система не дозволяє використовувати всі доступні функції застосунка Zoom, а лише найбільш важливі з них. Проте обраний набір функціоналу задовольняє потреби переважної більшості користувачів.

Загальний висновок Магістерська кваліфікаційна робота виконана у відповідності з завданням із дотриманням всіх вимог.

Оцінка магістерської роботи Робота заслуговує оцінку «добре», а студент Басараб Д.Р. – присвоєння кваліфікації «магістра» з комп'ютерних наук.

Рецензент _____

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання, посада, місце роботи)

«___»_____20__ р. _____ (підпис)

ДОДАТОК Б

ПЕРЕЛІК ДОКУМЕНТІВ НА ОПТИЧНОМУ НОСІЇ

Ім’я файла	**Опис**
Пояснювальні документи	
Диплом_Басараб.docx	Пояснювальна записка роботи. Документ Word.
Диплом_Басараб.pdf	Пояснювальна записка роботи в форматі PDF
Програма	
Program.rar	Архів. Містить коди програми і откомпільовану програму
Презентація	
Презентація_Басараб.pptx	Презентація роботи