Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет __Комп'ютерної інженерії та управління_____
(повна назва)

Кафедра Автоматизації проектування обчислювальної техніки
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти____другий (магістерський)_____
(рівень вищої освіти)

Система відео розпізнавання жестів мови глухонімих
з використанням нейронних мереж у реальному часі
(тема)

Виконав: студент 2 курсу, групи СКСм-22-2
Тихомиров В. І.
(прізвище, ініціали)
Спеціальність *123 _Комп'ютерна інженерія*
Тип програми ____освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)
Освітня програма_ *Спеціалізовані*
_комп'ютерні _системи
(повна назва освітньої прграми)
Керівник роботи _доцент Хаханова Г. В._
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ____(підпис)____ _Чумаченко С.В
(прізвище, ініціали)

2023 р.

Харківський національний університет радіоелектроніки

Факультет_____Комп'ютерної інженерії та управління _____

Кафедра_____Автоматизації проектування обчислювальної техніки_____

Рівень вищої освіти____другий (магістерський)_____

Спеціальність_____123_ Комп'ютерна інженерія _____
(шифр і назва)

Тип програми _____Освітньо-професійна_____
(освітньо-професійна або освітньо-наукова)

Освітня програма _____Спеціалізовані комп'ютерні системи_____
(повна назва

ЗАТВЕРДЖУЮ:

Зав.кафедри_____
(підпис)

"____" _____ 20 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові ____Тихомирову Валентину Ігоровичу _____
(прізвище, ім'я, по батькові)

1. Тема роботи (проекту) Система відео розпізнавання жестів мови глухонімих з використанням нейронних мереж у реальному часі

затверджена наказом по університету від "03"____11_____2023 р. № _1282 Ст_____

2. Термін подання студентом роботи до екзаменаційної комісії ____15.12.2023_____
3. Вихідні дані до роботи (проекту)_____

Мова жестів ASL

Машинне навчання

Розпізнавання кінцівок з використанням Computer Vision

Мова програмування Python

Середовище розробки VSCode

4. Перелік питань, що потрібно опрацювати у роботі _____

Особливості навчання нейронних мереж

Порівняльний аналіз методів розпізнавання жестів рук у реальному часі

Адаптація методів розпізнавання жестів рук для розпізнавання ASL

Вибір платформи та інструменті для реалізації середи для аналізу

Проектування середи для аналізу ефективності перетворень звуку

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 24 слайди

6. Консультанти розділів роботи (проекту)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

7. Дата видачі завдання 02.09.2023

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи (проекту)	Термін виконання етапів проекту (роботи)	Примітка
1	Видача теми проекту, узгодження і затвердження	02.09.2023 - 03.09.2023	
2	Аналіз існуючих методів розпізнавання жестів рук	02.09.2023 - 10.09.2023	
3	Дослідження алгоритмів розпізнавання жестів рук	02.09.2023 - 17.09.2023	
4	Адаптація та розробка методів розпізнавання жестів руки з використанням нейронних мереж	02.09.2023 - 25.09.2023	
5	Програмна реалізація системи нейронної мережі для розпізнавання жестів рук	02.09.2023 - 05.10.2023	
6	Тестування системи	02.09.2023 - 10.10.2023	
7	Оформлення пояснювальної записки	02.09.2023 - 20.10.2023	
8	Перевірка виконаного проекту керівником	02.09.2023 - 20.10.2023	
9	Захист проекту	02.09.2023 - 12.01.2024	

Студент _____
(підпис)

Керівник роботи _____ доц. Хаханова Г.В.

РЕФЕРАТ

Пояснювальна записка містить 74 сторінки, 38 рисунків, 35 джерел за переліком посилань.

НЕЙРОННІ МЕРЕЖІ, КОМП’ЮТЕРНИЙ ЗІР, МАШИНЕ НАВЧАННЯ, ASL, PYTHON, NUMPY, OPENCV, ГІСТОГРАМИ НАПРЯМКУ

У даній кваліфікаційній роботі було проведено порівняльний аналіз різних методів розпізнавання жестів глухонімих. Було досліджено різні типи нейронних мереж, а також підходи до їх навчання. Спроектовано і розроблено додаток з використанням нейронної мережі та комп’ютерного зору, що реалізує розпізнавання жестів глухонімих алфавіту. Для розпізнавання жестів використовується згорткова штучна нейронна мережа. Код програми написано мовою програмування Python.

ABSTRACT

Thesis contains 74 pages, 38 figures, 35 sources according to the list of links.

NEURAL NETWORKS, COMPUTER VISION, MACHINE LEARNING, ASL, PYTHON, NUMPY, OPENCV, DIRECTION HISTOGRAMS

In this thesis, a comparative analysis of various methods of recognizing of hand sign language was carried out. Different types of neural networks were investigated, as well as approaches to their training. An application was designed and developed using a neural network and computer vision, which realizes the recognition of gestures of the deaf and mute alphabet. A convolutional artificial neural network is used for gesture recognition. The program code is written in the Python programming language.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

ASL – American Sign Language;

ШНМ – штучна нейронна мережа;

ЗНМ – згорткова нейронна мережа;

GUI – Graphical User Interface (англ.) – графічний користувальницький інтерфейс. Різновид чоловік-компьютерного інтерфейсу, в якому елементи інтерфейсу, представлені користувачеві, виконані в вигляді графічних зображень;

ML – машинне навчання (англ. Machine Learning);

CV – комп’ютерний зір (англ. Computer Vision).

ВСТУП

Наше життя пов’язане з інформаційними технологіями. Швидкий розвиток способів спілкування між людьми призвів до того, що неможливо залишити осторонь проблему спілкування з зовнішнім світом. Людина може розпізнавати жести.

Без додаткових пристроїв спілкуватися та взаємодіяти з пристроями та машинами природно. Розпізнавання жестів можна використовувати для керування різними інженерними пристроями, які можуть використовуватися в медицині або сфері розваг.

Удосконалення обчислювальних потужностей і поява великих баз зображень дозволили навчати глибокі нейронні мережі. Ці мережі можуть обробляти та аналізувати велику кількість даних і моделювати когнітивні процеси в різних областях завдяки своїй багатошаровій архітектурі. Згорткові нейронні мережі, також відомі як CNN, досягли значного успіху в розпізнаванні образів. Нейронні мережі дозволяють досягти кращих результатів у більшості складних завдань, які традиційні методи не можуть вирішити. Нейромережі можуть використовувати один інструмент для вирішення багатьох завдань, оскільки вони адаптивні.

Більшість людей здорові та самодостатні, тому вони не звертають уваги на проблеми людей з обмеженими можливостями, оскільки вони вважають їх незначними, віддаленими та просто неважливими в повсякденному житті. Крім того, слід зазначити, що саме група людей з особливими можливостями стикається з такими проблемами, як відсутність підтримки суспільства, нестача послуг, проблеми з наданням цих послуг, недостатнє фінансування для усунення бар’єрів, відсутність консультування та включеності до суспільного життя, тощо.

Важливість питання про підвищення доступності комп’ютерної взаємодії для користувачів з особливими потребами підтверджує важливість

даної роботи. У процесі розробки програмного продукту планується покращити універсальність використання для всіх груп користувачів за допомогою додатку. Зображення звичайної веб-камери дозволяють системі визначати та ідентифікувати рухи та дії людини. Потім система використовує ці зображення для спілкування з іншими людьми чи керування програмними продуктами. Віртуальна реальність, навчальні програми, керування пристроями та спілкування в соціальних мережах є потенційним використанням.

Мета роботи полягає в тому, щоб дослідити та розробити способи розпізнавання як статичних, так і динамічних жестів руки, щоб людина могла безконтактно взаємодіяти з комп’ютером. Метою є вирішення таких основних завдань: вивчення та розробка існуючих методів відстеження та розпізнавання жестів руки, а також безконтактної людино-машинної взаємодії; вивчення та розробка методів опису, вилучення та розпізнавання конфігурацій руки; і розробка нових методів розпізнавання жестів руки на основі нейронних мереж для управління комп’ютерними системами.

У результаті проведених досліджень було розроблено власний програмно-апаратний комплекс, який міг розпізнавати жести рук і зберігати результати в базі даних для обробки та інтерпретації. Було розроблено аудіо плеєр, який можна керувати за допомогою жестів.

1 АНАЛІЗ ІСНУЮЧИХ МЕТОДІВ РОЗПІЗНАВАННЯ ЖЕСТІВ РУК

1.1 Проблема розпізнавання жестів

Автоматизація спілкування між глухими та чуючими людьми залежить від розуміння жестової мови. Наразі більшість таких комунікацій залежить від людей, які перекладають, що не завжди зручно.

Автоматичне розпізнавання мови жестів прагне самостійно зрозуміти значення знаків. Відповідно до вимог кінцевих користувачів, жести можна перетворити на звук або текст. Для поліпшення суспільного добробуту та надання кожній людині рівних можливостей важливо розуміти мову жестів. Однак розпізнавання жестів залишається складною проблемою, як і раніше, незважаючи на численні дослідження, проведені останніми десятиліттями [1]. Перекладач повинен розуміти багатомодальну інформацію, таку як поза та рух руки, вираз обличчя та поза людського тіла. Крім того, жестова мова складається з тисяч слів, деякі з яких дуже схожі. Тим не менш, одні й ті ж знаки можуть значно відрізнятися за зовнішнім виглядом, якщо їх показують різні люди або з різних ракурсів.

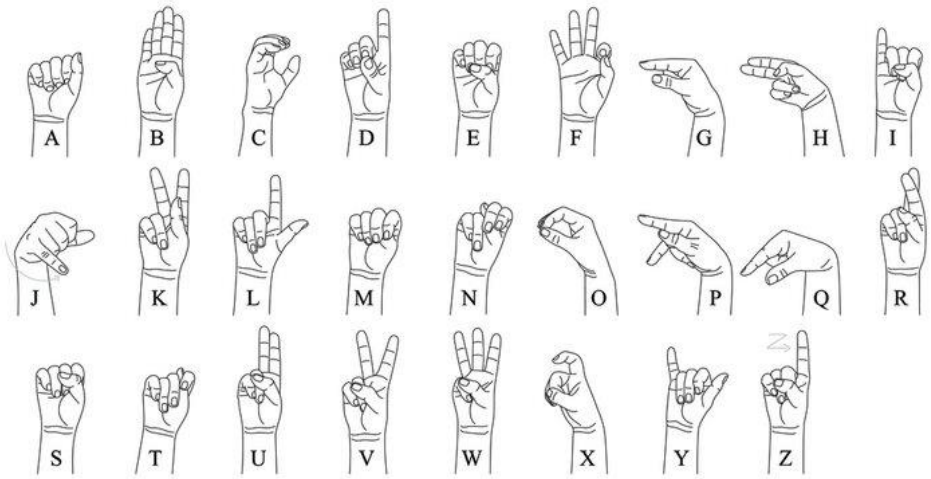

Рисунок 1.1 – ASL(Американська жестова мова)

Розглянемо поняття – людино-комп'ютерна взаємодія. Людино-комп’ютерна взаємодія (HCI) – це багатодисциплінарний науковий напрям, який розвивається з метою дослідження різних аспектів використання інтерактивних комп’ютерних систем, призначених для людей, а також методів їх оцінки та впровадження [5].

Взаємодія, заснована на жестах людини, є одним із підходів до міжособистісної взаємодії (HCI), який останнім часом стає все більш поширеним [6, 7, 8].

Жести показують інтенсивність переживань, якість і спрямованість відносин, а також приналежність до культури та групи. Жести, як і міміка, можуть нести інформацію про людину без мови. Жести говорять про ставлення людини до якоїсь особи, події чи предмету, коли вони супроводжують мову або замінюють її.

У сфері взаємодії людини та комп’ютера (HCI) жести використовуються для передачі інформації в комп’ютер, яка в подальшому може використовуватися для ідентифікації людини, управління комп’ютерами, літаками, ігровими аватарами та інших цілей.

В задачах розпізнавання образів і обробки цифрових зображень часто застосовується функція інтенсивності виду f(x, y), що залежить від цілочисельних координат x і y належать так званої растрової решітці W2 [11]. Точка зображення знаходиться в системі координат OXY за допомогою пари (x, y), яка починається з лівого верхнього кута решітки. Зазвичай функція f(x, y) задає в точці (x, y) яскравість, освітленість, насиченість, ступінь поглинання та інші спектральні характеристики.

Нехай є просторова точка M, яка має координати (x, y, z) і ортогональна проекція цієї точки, m, яка має координати (x, y) на площину спостереження Z. Коли ми говоримо про цифрове зображення d(x, y), то в кожній точці (x, y) d(x, y) приймає невід’ємні цілі значення, рівні величині відстані від точки M до точки m. Це називається дальностним зображенням.

Завжди можна перетворити дальностне зображення в безліч точок {(x, y, d(x, y))} у тривимірному просторі, що є моделлю об’єкта, що спостерігається.

Під конфігурацією руки ми розуміємо положення кисті та пальців. Розпізнавання жестів руки розглядають по-різному в літературі, включаючи вимірювання положення долоні, плечей і кінчиків пальців руки, визначення конфігурації та траєкторії руху руки та інше. У цій роботі ми розглянемо три основні завдання розпізнавання жестів руки: розпізнавання позицій долоні та кінчиків пальців руки в тривимірному просторі; розпізнавання статичного жесту за допомогою еталонних конфігурацій; і динамічний жест за допомогою траєкторії руху долоні руки.

Система розпізнавання жестів руки складається з комп’ютерних програм і математичних алгоритмів, які можуть розпізнавати певну групу жестів руки.

Розпізнавання жестів можна використовувати в таких сферах людської діяльності, як наприклад: керування комп’ютером і побутовими приладами, де певна команда жестів відображається в комп’ютерному додатку для кожної конфігурації руки. Коли людина демонструє жест, система визначає форму руки та відправляє комп’ютеру відповідну команду. Рухи руки призводять до рухів курсору, а позиція долоні відповідає позиції курсору миші на екрані. Команди натискання кнопок мишки зображують різні конфігурації рук. Розпізнані позиції кінчиків пальців 18-ї руки можуть бути використані одночасно кількома пальцями руки для управління комп’ютером або побутовими приладами; створення природних людино-машинних інтерфейсів для глухонімих; застосування системи розпізнавання жестів для введення тексту в комп’ютер за допомогою жестів руки, що для глухонімих людей простіше і природніше, ніж введення тексту за допомогою клавіатури.

На одному кінці мережі можна розпізнати розпізнані жести руки, а на іншому кінці мережі можна показати анімацію цих же жестів, передаючи по мережі лише характеристики показаних жестів; - маніпулювання тривимірними моделями об'єктів; Для роботи з тривимірними моделями на

сьогоднішній день зазвичай використовують комп’ютерну мишу, яка не дуже зручна для цієї роботи. Можна створити систему взаємодії людини з комп’ютером (HCI), яка дозволить керувати моделями в усіх напрямках тривимірного простору, використовуючи тривимірні координати пальців руки та кінчиків пальців; - додатки віртуальної реальності. Можна створити додатки віртуальної реальності, які дозволяють користувачам «доторкатися» до віртуальних об’єктів, доповнюючи систему розпізнавання жестів пристроями, такими як стереоскопічні окуляри.

Системи, пов’язані з реабілітацією пацієнтів [12], ігрові програми [13] і т.д. можуть доповнити цей список. Багато з наведених додатків вже працюють у різних пристроях і програмах [14]. Інші додатки ще знаходяться в процесі дослідження.

Існуючі методи розпізнавання жестів руки поділяються на дві основні групи. Перша група включає методи, які базуються на дослідженні зовнішніх ознак жесту, а друга група ґрунтується на дослідженні тривимірної моделі руки.

Далі розглянемо дослідження з розпізнавання жестів руки людини.

1.2. Дослідження існуючих рішень розпізнавання жестів

Аналіз лише зовнішнього вигляду цільового об'єкта (форми, позиції тощо) є особливістю методів, заснованих на аналізі зовнішніх ознак жесту. Для розпізнавання нічого не зберігається про фізичні властивості об’єкта. Розглянемо відомі рішення та підходи до визначення жестів руки людини на основі зовнішніх ознак.

1.2.1. LeapMotion

Цей гаджет є комп'ютерним контролером нового покоління, здатний замінити традиційні (і всім набридлі) способи керування комп'ютером. До ПК

або ноутбука Leap Motion підключається за допомогою USB. Потім система створює робочий простір певного розміру, приблизно 0,03 метрів кубічних. Всередині цього простору користувач має працювати з системою. Жестовий інтерфейс гаджета дозволяє виявляти рухи в межах сотої частки міліметра.

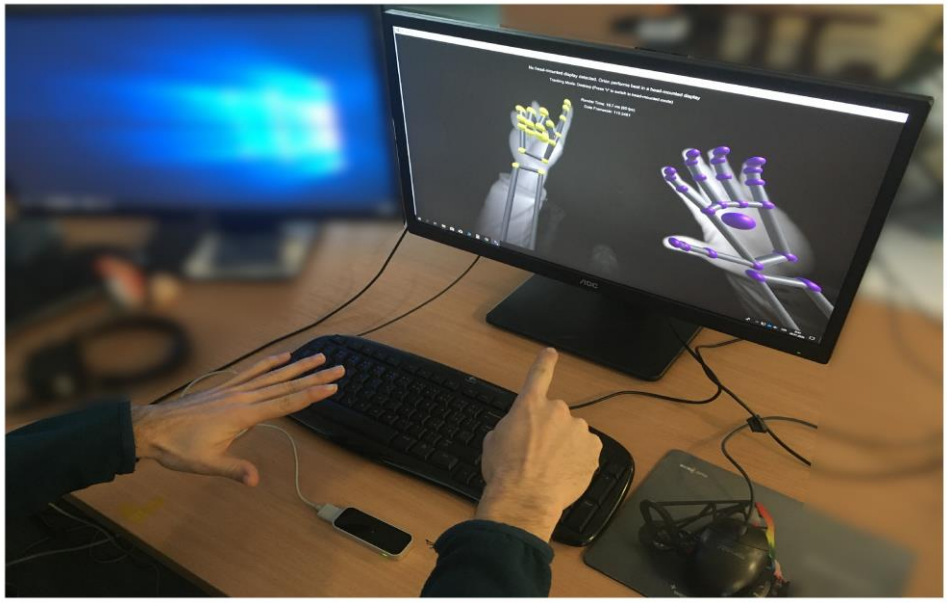

Рисунок 1.3 – Сенсор LeapMotion

1.2.2. Браслет Myo

Myo – розробка компанії Thalmic Labs. У пристрої застосований масив електродів, які реєструють активність у м`язах, коли користувач жестикулює. Дані передаються по бездротовому зв`язку Bluetooth і обробляються спеціальним програмним забезпеченням, що перетворює їх в керуючі команди. Цікаво, що електроди не потребують прямого контакту з шкірою. Система вже зараз розпізнає приблизно 20 жестів, ігноруючи випадкові рухи. Контролер здатний реагувати на індивідуальні рухи пальців, обертання руки і долоні. Гаджет можна використовувати на різну ширину передпліччя завдяки еластичним кріпленням між секціями.

Рисунок 1.4 – Браслет Myo

1.2.3 Сенсор Kinect

Kinect – безконтактний сенсорний ігровий контролер, спочатку випущений для консолі Xbox 360 і значно пізніше доступний для персональних комп’ютерів з операційною системою Windows. розроблений Microsoft. Це периферійний пристрій, який додається до грального консолі Xbox 360 – дозволяє користувачеві взаємодіяти з нею без використання контактного ігрового контролера, використовуючи усні команди, пози тіла, об’єкти або зображення. Збільшення кількості користувачів Xbox 360 є метою проекту.

Рисунок 1.5 – Сенсор Kinect

1.3. Методи розпізнавання мови жестів глухонімих

У ЛКВ є багато способів розпізнавання мови жестів. Основна їхня мета полягає в тому, щоб полегшити взаємодію, ідентифікуючи точні знаки та жести користувача. Усі ці методи складаються з наступних етапів: зчитування та попередня обробка зображення; подання жестів у вигляді масивів координатних точок; і, нарешті, категоризація, або визначення жесту.

Існує три основні способи розпізнавання знаків, відповідно до технології розпізнавання жестів. Одне походить від використання камер глибини та контролерів, а друге – від техніки зору, в якій відслідковуються рухи рук і інтерпретуються відповідні знаки [2]:

– стерео камери. Цей метод, заснований на тому, що комп'ютер фіксує зображення, перетворює жести рук у відповідне текстове повідомлення. Веб-камера записує рухи рук, а кадри розділяються на різні частини відповідно до кількості пальців і кута між ними. Спочатку область кисті виділяється з фону, а потім сегментуються пальці та долоня, щоб виявити та розпізнати пальці. Класифікатор, який базується на зібраних зображеннях із подібними параметрами розташування рук, обличчя та тіла, використовується для розпізнавання жестів рук. Такі системи, засновані на комп'ютерному баченні, можуть запропонувати рішення, які є більш простими, непомітними та безконтактними. Більшість програмних рішень, які використовують цей метод, можуть підтримувати розпізнавання жестів у реальному часі. Сила залежність від факторів навколишнього середовища, таких як місце камери, фон і чутливість до рівня освітлення;

– контролери для жестів. Цей метод використовує невеликий USB-пристрій, який за допомогою інфрачервоних світлодіодів і монохроматичних камер нічного бачення спостерігає за півсферичною зоною на відстані приблизно одного метра від користувача. Інфрачервоне світло зчитують світлодіоди, а камери фіксують майже 200 кадрів за секунду. Контролер зчитує такі дані, як вказування на точку, плавний рух або стискання, з цієї системи.

Використання синхронізованих кольорових зображень і карт глибини для пристроїв захоплення, які використовують переважно сенсор Microsoft Kinect, призвело до значного прогресу в цій технології розпізнавання жестів. Датчик Kinect [3], який використовує виявлення жестів, є камерою глибини. Рисунок 1.6 демонструє, як він працює. Використовуючи дистанційну адаптивну схему, ознаки були виділені. Алгоритм динамічної трансформації тимчасової шкали даних і алгоритм випадкових значень використовуються для перетворення жесту руки в текст. Недоліком цієї системи є те, що дані не є точними та достовірними;

Рисунок 1.6 – Приклад знімка, зробленого на камеру глибини

– рукавички, оснащені гіроскопами та акселерометрами. Цей метод має багато переваг. Вбудовані датчики руху не впливають на навколишнє середовище, тому вони більш точні, ніж розпізнавання на основі зору в умовах поганого бачення.

Крім того, гіроскопи та акселерометри безпосередньо приєднані до рук користувача, що дозволяє отримати додаткову інформацію про розташування рук. Деякі моделі розумних рукавичок дозволяють використовувати їх бездротово.

На основі даних мікроелектричних і акселерометричних датчиків відносні кути між долонею та пальцями аналізуються для визначення певного

руху. Рукавички за допомогою інтенсивності мікроелектричних датчиків можуть автоматично визначати початкові та кінцеві точки двох важливих сегментів жесту.

– натільні датчики. Цей метод є досить точним та ефективним с точки зору точності та швидкості роботи. Датчики на тілі зчитують м'язову активність, електромагнітні імпульси та надсилають отримані дані на певний контролер. З точки зору зручності – це не найкраще рішення, оскільки воно може обмежувати рухи людини.

Підхід з використанням однієї камери для зчитування зображення та визначення жестів був використаний під час розробки цього проекту. Для досягнення цієї цілі було використано камеру LogitechC-170, яка є веб-камерою.

2 ДОСЛІДЖЕНЯ АЛГОРИТМІВ РОЗПІЗНАВАННЯ ЖЕСТІВ РУК

2.1. Розпізнавання жестів на основі аналізу різниць зображень (MEI)

На практиці оцінка умов однорідного фону не завжди можлива. В таких випадках зазвичай використовується метод, заснований на аналізі MEI (центру маси різниць зображень) руки в кадрах відеоряду.

Різниці кадрів відеоряду (MEI) дозволяють аналізувати рух об’єкта в відеоряді в реальному часі, отримуючи стабільне, але не завжди однорідне зображення. На практиці ця технологія та її варіанти (наприклад, рухова історія зображення - MHI) використовуються в таких додатках, як інтерактивний віртуальний тренер з аеробіки та інтерактивна кімната для розповіді історій.

2.2. Розпізнавання жестів рук на основі аналізу зовнішніх ознак жесту

Аналіз лише зовнішнього вигляду цільового об'єкта (форми, позиції тощо) є особливістю методів, заснованих на аналізі зовнішніх ознак жесту. Для розпізнавання не зберігається жодна інформація про фізичні характеристики даного об'єкту.

Розглянемо відомі дослідження та підходи до визначення жестів руки людини на основі зовнішніх ознак.

2.1.1 Розпізнавання позиції і орієнтації за допомогою моментів зображення

Моменти нульового та першого порядку зображення $M_{0,0}$ $M_{0,1}$, $M_{1,0}$ і $M_{2,0}$ $M_{1,1}$ $M_{0,2}$ можна знайти за допомогою функції інтенсивності f(x, y) згідно з формулою 2.1:

$$M_{0,0} = \sum_x \sum_y f(x,y),$$
$$M_{0,1} = \sum_x \sum_y y \times f(x,y),\ M_{1,0} = \sum_x \sum_y x \times f(x,y), \quad (2.1)$$

Розпізнавання простих жестів рук і додатки HCI на основі зображень можуть бути використані за певних умов.

Наприклад, у роботі [15] розглядається додаток, який використовує рухи руки для керування іграшковим роботом. У цьому додатку напрямки руху робота визначаються орієнтацією руки (рисунок 2.1).

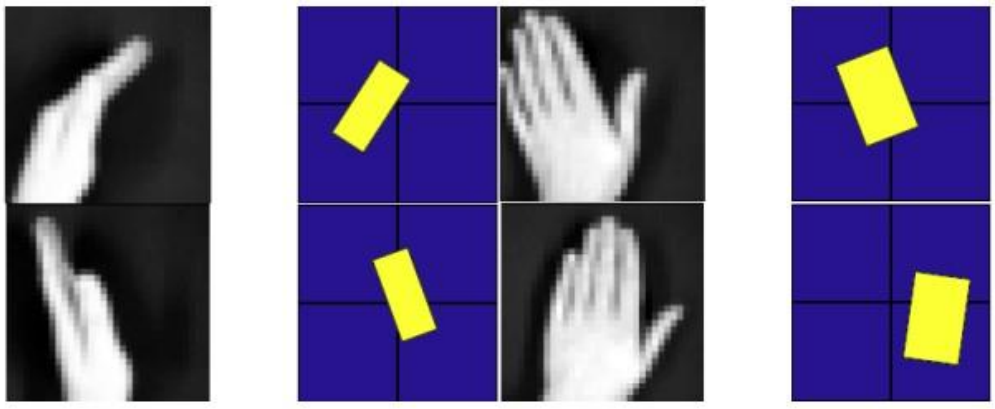

Рисунок 2.1 — Управління жестами руки. Прямокутник показує розпізнаний напрямок, позицію і розмір зображення руки

За умови, що фон зображення однорідний і рука є домінуючим об’єктом зображення, обчислення моментів зображення використовуються для розпізнавання позиції (xc,yc) і орієнтації руки в бінарному зображенні.

Умова arctan(i, j)є(-π,π] задовольняється функцією arctan. Значення l_1 і l_2 використовуються для визначення розміру зображення руки.

$$l_1 = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}}, l_2 = \sqrt{\frac{(a+c) + \sqrt{b^2 + (a-c)^2}}{2}} \qquad (2.2)$$

На рисунку 2.1 для кожного зображення приведені прямокутники з центром (x_c y_c)та розміром l_1 x l_2.

2.2.2 Розпізнавання жестів на основі аналізу гістограм напрямків

Більшість програм для комп'ютерного зору вимагають додаткової конфігурації людини, крім позиції та орієнтації руки. Аналіз так званих гістограм напрямків (histograms of orientation) і карт напрямків (orientation maps) на зображеннях, які менш чутливі до змін освітлення навколишнього середовища, допомагає вирішити проблеми, описані в роботах [18, 19].

Нехай інтенсивність дво- або напівтонового зображення руки є f(x, y). Наступним чином визначається локальний напрямок (x, y) в точці (x, y):

$$\theta(x,y) = \arctan\big(f(x,y)\big) - f(x-1,y), f(x,y) - f(x,y-1) \qquad (2.3)$$

Евклідова відстань між векторами двох зображень є відстанню між двома пальцями руки. На рисунку 2.2 показано зображення жестів руки, а також відповідні гістограми напрямків при N = 36.

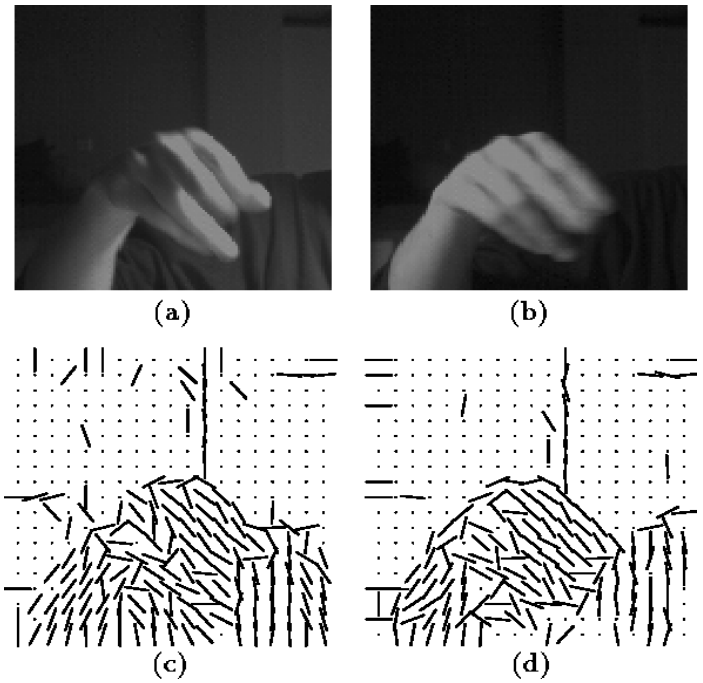

Рисунок 2.2 – Гістограма напрямків для жесту руки

Додаток для управління анімаційним краном використовує наведені жести та технологію розпізнавання конфігурацій руки, засновану на гістограмах напрямків в роботі [18]. У роботі відзначається, що додаток відчуває зміни напрямку та розміру руки в реальному часі. Кожен оператор потребує навчання в додатку.

Використовуючи цю технологію, можна в режимі реального часу визначити конфігурацію руки, якщо виконуються наступні умови:

– фон зображення однорідний;
– руки є домінуючим об’єктом у зображенні;
– жести обрані таким чином, щоб їхні гістограми напрямків значно відрізнялися один від одного.

2.2.3 Розпізнавання конфігурації і позиції із застосуванням кольорових рукавичок

Кольорові рукавички часто використовуються для ідентифікації жестів руки. Розпізнавання конфігурації руки та відстеження рухів долоні в

тривимірному просторі можна зробити за допомогою однієї відеокамери (рисунок 2.3).

Використовувана рукавичка складається з двадцяти сегментів, кожен з яких містить десять різних кольорів. Використання невеликої кількості кольорів дозволяє визначити колір вибраної точки зображення рукавички при різних освітленнях, тоді як спеціальне розташування колірних сегментів не дозволяє отримати однакове зображення при різних конфігураціях рук.

Дана система знайшла застосування в задачах управління анімаційними персонажами і розпізнавання жестів ручної азбуки глухонімих ASL.

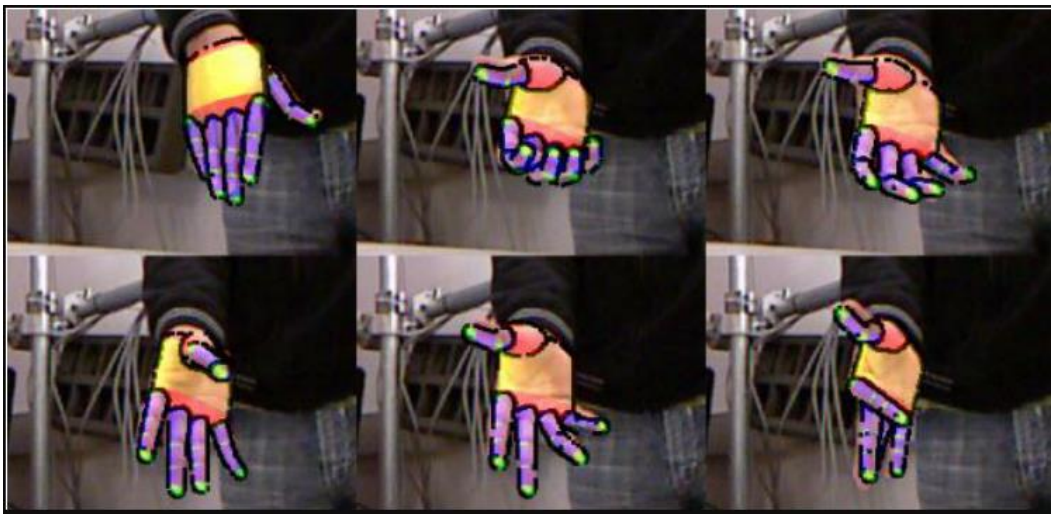

Рисунок 2.3 – Розпізнавання жестів руки за допомогою кольорових рукавичок

Алгоритм двостороннього розмиття (фільтр двостороннього розмиття) згладжує початкове зображення, видаляє фонові точки, які класифікуються відповідно до моделі Гаусових сумішей, і перетворює отримане зображення в растрове зображення меншого розміру для розпізнавання конфігурації руки. Для порівняння отримане растрове зображення порівнюється з 18000 еталонними зображеннями, які відповідають різним конфігураціям руки з

відстанню $d(r^i, r^j)$ між растровими зображеннями r^i та r^j оцінюється за допомогою формули 2.4.

$$\tilde{d}(r^j, r^i), \qquad \tilde{d}(r^j, r^i) \sqrt{\frac{1}{|C_i|} \sum \quad \min(u - x)^2 + (v - y)^2}. \qquad (2.4)$$

Дану систему можна використовувати для розпізнавання жестів ручної азбуки глухонімих і управління анімаційними персонажами.

2.2.4 Розпізнавання конфігурації і позиції на основі аналізу контуру зображення руки

Контур зображення руки часто вибирають у якості дескриптора конфігурації руки. Наприклад, дослідження [33] розглядає метод розпізнавання жестів ручної азбуки глухонімих ASL, який базується на аналізі контуру зображення руки. Для полегшення завдання розглядаються два динамічних жести, а деякі замінюються новими. Всі точки, які не відповідають кольору шкіри людини, видаляються з кольорового зображення руки. За допомогою фільтра Гауса отримане зображення згладжується до бінарного зображення. Оператор Собеля використовується для визначення контурів руки. (В обробці зображень границі можна виділити за допомогою оператора Собеля. Це дискретний диференціальний оператор, який може обчислити наближене значення градієнта або норму градієнта для яскравості зображення. Оператор Собеля базується на використанні невеликих сепарабельних цілочисельних фільтрів, які згортають зображення у напрямку.

Описаний прямокутник контуру руки, центр якого відповідає початку координат, обчислюється.

В статті [34] розглядається питання ідентифікації положення кінчиків пальців руки за допомогою перегляду контуру руки на кольоровому зображенні. Як і в [33], перші кроки алгоритму включають видалення фонових

точок, згладжування та виділення зображення руки. Використовуючи контур руки, пальці можна розрізнити за допомогою аналізу локальних вигинів. Даний алгоритм використовувався для контролю веб-камери комп'ютера.

На рисунку 2.4 показано приклад пошуку контурів руки людини.

Проблема з цим алгоритмом полягає в тому, що він розпізнає лише прості жести, і якщо є фон, жест може бути не розпізнаним. Освіта та складність фону впливають на роботу алгоритма. Цей метод зазвичай використовується в процесі попередньої обробки зображення, а не самостійно.

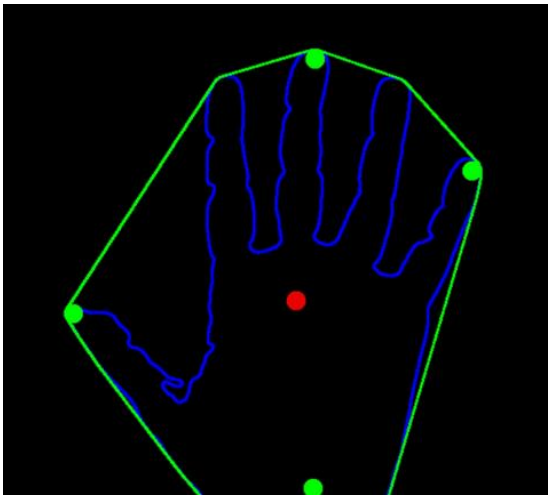

Рисунок 2.4 – Пошук контуру руки

2.2.5 Розпізнавання позиції і конфігурації руки методом випадкових лісів

Алгоритм випадкових лісів - це алгоритм машинного навчання, який використовується для задач класифікації, регресії та кластеризації. Цей алгоритм був використаний для розпізнавання жестів для класифікації конфігурації руки та визначення позицій основних фізичних характеристик людини [26]. Ця технологія була основою системи розпізнавання жестів людини, розробленої Microsoft у 2010 році.

Метод, який пропонується [25], дозволяє визначити позиції двадцяти частин тіла (рисунок 2.5.а), включаючи позиції долонь обох рук,

використовуючи на вході дальнє зображення людини. Випадкові ліси рішень, навчені на вибірці зображень відповідних 100000 різних поз тіла людини, використовуються для позиціонування цікавих точок тіла. Алгоритм розпізнавання складається з наступних етапів:

– розділення зображення людини в дальностному зображенні;
– використання алгоритму випадкових лісів для поділу зображення людини на 31 кластер;
– на основі цих кластерів, визначення позицій, які цікавлять частини тіла. Рисунок 2.5 показує візуальне представлення.

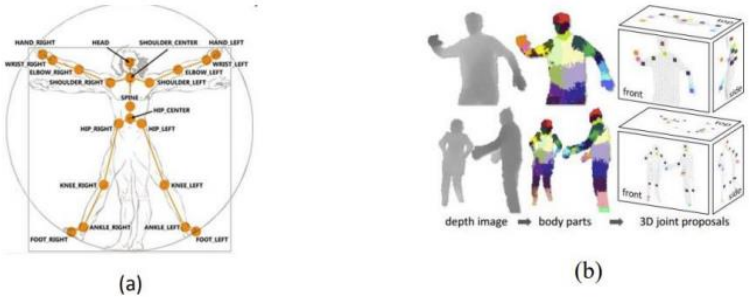

Рисунок 2.5 - Розпізнавані точки тіла людини та кроки розпізнавання

Згідно з інформацією, наведеною в [25], цей алгоритм може ідентифікувати жести людини зі швидкістю 200 кадрів в секунду під час роботи на Xbox 360. На один день потрібно навчити три дерева з глибиною двадцяти на безлічі з мільйона зображень і на суперкомп’ютері з процесором з 1000 ядер.

В роботі [26] випадкові ліси використовуються для визначення конфігурації руки для великої кількості статичних жестів, які використовуються в ручній азбуці ASL. Розроблена система має архітектуру, представлену на рисунку 2.6.

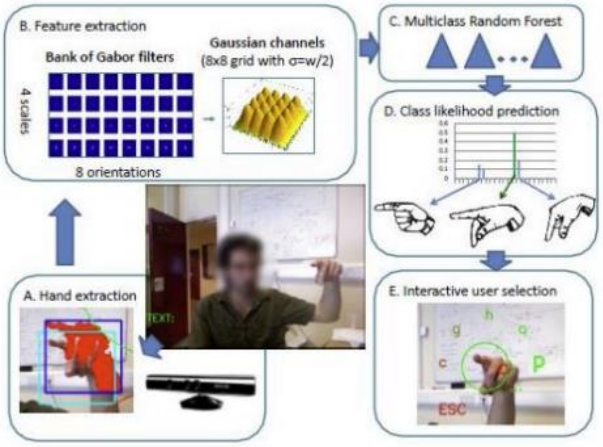

Рисунок 2.6 – Архітектура моделі розпізнавання жестів руки з кількома статичними жестами

За допомогою сенсора Kinect [27], який працює як пристрій введення, система може бачити людину в дальній і кольоровій перспективі. Програмні платформи OpenNI і Nite [12] шукають руку людини, а потім сегментують її як зв’язаний об’єкт у дальностному зображенні. Після того, як зображення руки були згладжені, їм додається фільтр Габора. Зображення перетворюються на матрицю розміром 8 на 8, яка служить описом демонстрованого жесту. Алгоритм випадкових лісів використовувався для навчання системи. Розроблена система може в реальному часі ідентифікувати 24 жести ручної азбуки ASL. Середня якість розпізнавання запропонованого алгоритму становить 75% під час тестування системи на жестах п’яти різних людей. Точність розпізнавання при використанні лише кольорового зображення становить 73%, а точність розпізнавання при використанні лише дальностного зображення становить 69%.

2.2.6 Розпізнавання жестів із застосуванням штучних нейронних мереж

Модель штучних нейронних мереж (ШНМ) базується на принципах біологічних нейронних мереж. ІНС зазвичай використовують як інструмент машинного навчання для завдань розпізнавання жестів. У цих завданнях на

вхід ІНС передаються характеристики жесту, а на виході ІНС виходить розпізнаний жест. ІНС відрізняються структурою та методами навчання.

Межі прямого поширення, рекурентні нейронні мережі, самоорганізовані карти Кохонена та згорткові нейронні мережі є найбільш поширеними типами ІНС, які використовуються в задачах розпізнавання. На рисунку 2.7 показано рекурентну нейронну мережу, яка використовується для розпізнавання статичних жестів руки користувача азбуки глухонімих Японії. Вхідний пристрій роботи складається з рукавички з вбудованими сенсорами, які можуть відображати характеристики жесту (10 точок, 28 точок і 3 кута). Характеристики жесту передаються на вхід нейронної мережі після нормалізації. На виході мережа видає масив із сорока двох елементів, які відповідають сорока двом жестам руки. Розпізнавання показало 71.4% зареєстрованих користувачів і 47.8% незареєстрованих.

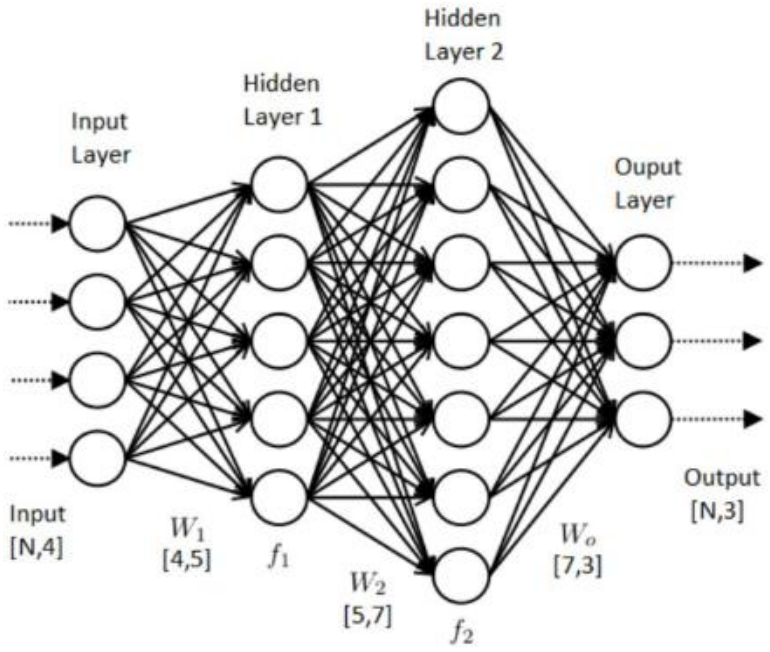

Рисунок 2.7 – Приклад моделі нейронної мережі для розпізнавання жестів рук

Алгоритм «нейронний газ» використовується для розпізнавання конфігурацій руки. Конкурентне навчання Хебба використовується для навчання. Якість розпізнавання 90.45% і швидкість 1,5 секунди заявлені. Відеокамера є вхідним пристроєм. Запропонована система використовує рукавичку, марковану різними кольорами, щоб полегшити розпізнавання. Якість розпізнавання при використанні повністю рекурентної нейронної мережі становила 95,11%, а при використанні ІНС Елмана – 89,67%.

2.2.7 Розпізнавання жестів із застосуванням прихованих моделей Маркова

Прихована Марковська модель (ПММ) – це статистична модель, яка широко використовується для розпізнавання жестів і розпізнавання мови та листи. ПММ складається з безлічі станів, змінних і залежностей, з імовірним розподілом для кожного стану серед усіх можливих вихідних значень.

У задачах розпізнавання жестів зазвичай створюється ПММ, яка навчається розпізнавати конкретний жест руки. Кожна модель передається на вхід на етапі розпізнавання. Обчислюється функція відповідності переданих характеристик і кожної моделі. ідентифікує жест, для якого ця функція має найбільше значення. ПММ можна використовувати для розпізнавання безперервних і ізольованих жестів.

ПММ може визначити десять динамічних жестів руки. Якість розпізнавання ізольованих жестів складає 98,94%, а безперервних – 95,7%.

2.3 Метод розпізнавання жестів руки на основі аналізу тривимірної моделі руки

Наявність одного або кількох двомірних зображень жесту на вході дозволяє комп’ютерному зору використовувати цю технологію розпізнавання жестів для отримання детальної тривимірної конфігурації руки. Надається

тривимірне зображення положення та орієнтації долоні та ключових точок пальців руки.

Порівняння проекцій тривимірної моделі руки з вхідними зображеннями дозволить розпізнати жести. У проектуванні тривимірної моделі руки на площину виходить двовимірне зображення, яке порівнюється з вхідним зображенням. Якщо відстань між двома зображеннями менше встановленого порогу, то жест ідентифікується. Якщо відстань більше, конфігурація моделі руки змінюється і порівнюється з вхідним зображенням. Конфігурація руки в кожному кадрі відеоряду зазвичай збігається або незначно відрізняється від конфігурації руки в попередньому кадрі, хоча гіпотези, які використовуються в даних систем, різноманітні.

Основним недоліком цих методів є те, що вони вимагають значних ресурсів. Іноді розпізнавання жесту в одному кадрі вимагає навіть п’ятнадцять секунд, або шістдесят шість мілісекунд при використанні графічного процесору. Таким чином, у багатьох роботах є обмеження на кількість точок артикуляції.

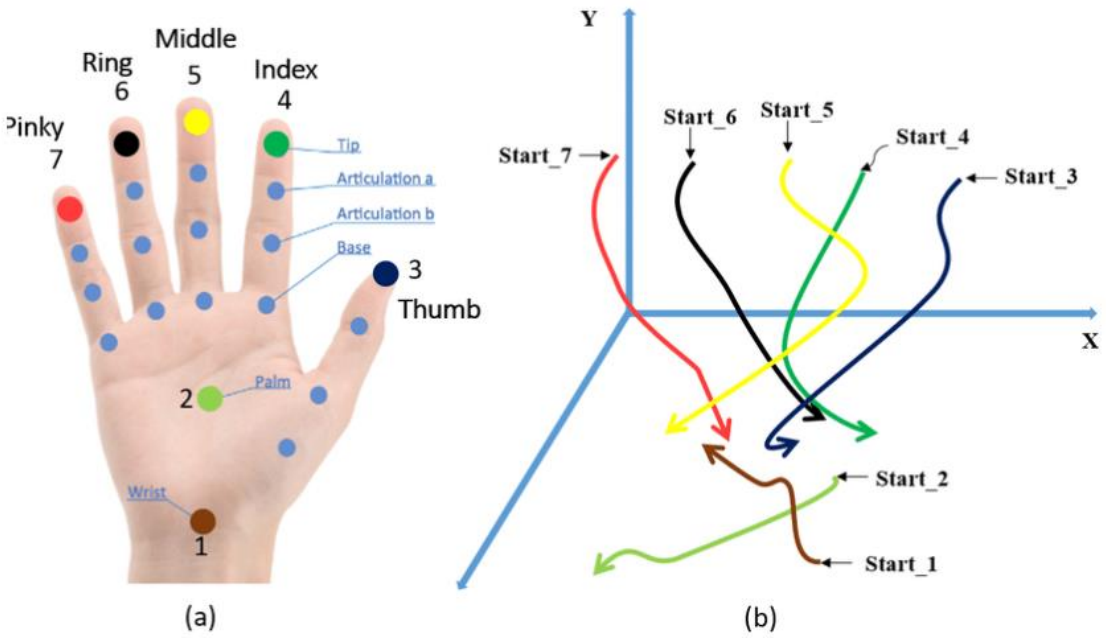

Рисунок 2.8 - Структура руки людини для тривимірного аналізу

2.4 Розпізнавання жестів за допомогою ключових координатних точок

Переведіть зображення рук, обличчя та тіла в набір тривимірних координат у просторі, перш ніж перейти до наступних команд. Ви повинні побачити, що ці координати зберігають розташування вашої руки та окремих пальців на ній.

Для пошуку точного просторового розташування векторів для розпізнавання жестів слід використовувати метод координат точок.

Щоб точно визначити довжину вектора на зображеному жесті, необхідно провести достатньо обчислень за наступною формулою:

$$AB = \sqrt{(x_b - x_a)^2 + (y_b - y_a)^2 + (z_b - z_a)^2}, \quad (2.5)$$

де (x_a, y_a, z_a, x_b, y_b, z_b) - це відповідні координати початку й кінця вектора AB.

У цьому випадку можна визначити, який з пальців зігнутий, порівнявши довжину векторів, що йдуть від верхньої фаланги до основи пальця або долоні. Пропорційне з’єднання краще. Якщо відношення довжин векторів вказівного та середнього пальців більше 1,5, то середній палець вважається зігнутим.

3 АДАПТАЦІЯ ТА РОЗРОБКА МЕТОДІВ РОЗПІЗНАВАННЯ ЖЕСТІВ РУКИ З ВИКОРИСТАННЯМ НЕЙРОНИХ МЕРЕЖ

3.1 Проектування згорткової нейронної мережі

Згорткова нейронна мережа (CNN) складається з одного або більше об’єднаних або поєднаних згортальних шарів. Використовуючи варіацію багатошарового перцептрона, розглянутого вище, CNN використовує цей самий перцептрон.

Для вхідних даних згорткові шари використовують операцію згортки, а результат передається в наступний шар (рисунок 1.3). У результаті цієї операції мережа може бути глибше з меншою кількістю параметрів.

3.1.1 Архітектура згорткової нейронної мережі

Дані ЗНМ будуть мати архітектуру, яка складатиметься з двох повнозв’язних, трьох згорткових шарів і трьох агрегувальних (пулінгових) шарів. Пулінгові та згорткові шари чергуються між собою.

Можна задати кількість вхідних і вихідних каналів для кожного згорткового шару. Це потрібно для розрахунку кількості входів і виходів кожного шару.

У першому згортковому шарі буде один вхідний канал (залежно від кількості каналів зображення) і вісім вихідних каналів. Другий згортковий шар має 8 вхідних каналів і 16 вихідних; третій шар має 16 вхідних і 32 вихідних. Кількість вхідних каналів залежить від кількості вихідних попереднього шару. Крім того, шар має ядро згортки розміром 5×5. Після кожного згорткового шару слід пулінговий шар із фільтром розміром 2×2.

Формула, яка показує кількість входів і виходів згорткового шару, є такою:

$$I = W_i \times H_i \times C_i, \quad (3.1)$$

де W_i – ширина вхідного зображення;

H_i – висота вхідного зображення;

C_i – кількість вхідних каналів.

$$O = W_o \times H_o \times C_o, \quad (3.2)$$

де W_o – ширина вихідного зображення;

H_o – висота вихідного зображення;

C_o – кількість вихідних каналів.

Відповідно до формули 3.1, кількість входів першого шару дорівнює 60 × 60 × 1 = 3600.

Формула використовується для визначення розміру зображення після проходження згорткового шару [25].

$$W_o = \frac{W_i - K + 2P}{S} + 1, \quad (3.3)$$

де K – розмір ядра згортки;

P – вирівнювання (за замовчуванням рівне 0);

S – крок згортки (за замовчуванням рівне 1).

$$H_o = \frac{H_i - K + 2P}{S} + 1. \quad (3.4)$$

Після проходження першого згорткового шару розмір зображення визначається за формулами (3.3), (3.4) і дорівнює 56 на 56.

У першому шарі було 25088 виходів (3.2).

Зображення переходить на пулінговий шар після проходження згорткового шару. Там воно ущільнюється. Для визначення розміру зображення після проходження агрегувального шару використовуються формули:

$$W_o = \frac{W_i}{F}, \tag{3.5}$$

де F – розмір фільтру агрегувального шару.

$$H_o = \frac{H_i}{F}, \tag{3.6}$$

Після видалення пулінгового шару зображення матиме виміри відповідно до формул (3.5) та (3.6), тобто 28 на 28.

Формула (3.1) показує, що на другому згортковому шарі 6272 входи. За формулами (3.3) і (3.4) після проходження зображення має розмір 24 на 24 сантиметри.

Формула (3.2) показує, що другий шар має 9216 виходів.

Відповідно до формул (3.5) та (3.6), зображення було зменшено до 12 на 12 пікселів за допомогою агрегованого шару.

У третьому згортковому шарі 2304 входи (3.1). Після проходження зображення має розмір 8 на 8 (3.3).

2048 вихідів з третього рівня (3.2).

Пулінговий шар ущільнив розмір зображення до 4 на 4 (3,5) і (3,6).

Після проходження всіх пулінгових і згорткових шарів зменшене зображення передається на повнозв’язний шар. Кількість входів цього повнозв’язного шару дорівнює 512, що визначається за формулою (3.1). 256 виходів.

На вихідному шарі нейронної мережі є 256 входів і 30 виходів, що дорівнює кількості класів, присутніх у наборі даних.

3.1.2 Активація нейронів

Функція активації ReLU (Rectified Linear Unit) - лінійний випрямлювач застосовується до кожного згорткового та повнозв’язного шару ЗНМ, за винятком вихідного шару (рис. 3.1). Якщо функція приймає від’ємний аргумент, вона повертає 0, але якщо приймає додатний аргумент, вона повертає саме число. Визначається таким чином:

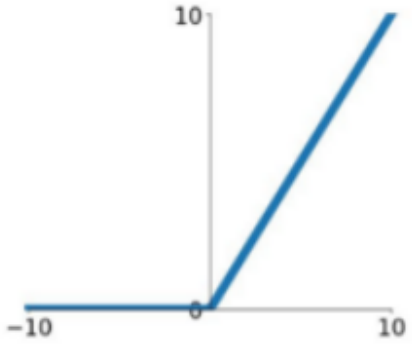

Рисунок 3.1 – Графік функції ReLU

3.1.3 Модуль оптимізації

Алгоритм оптимізації Adam(Adaptive Moment Estimation, адаптивна оцінка моменту), який має швидкість навчання 0.001, використовується для обробки даної нейронної мережі. Adam, також відомий як адаптивна оцінка моменту, є алгоритмом стохастичної оптимізації, який забезпечує адаптивну швидкість навчання.

Adam вибирає індивідуальну швидкість навчання для кожного параметра. У Адамі параметри, які зазвичай отримують менше або менш часті оновлення, отримують більше (вірно і зворотне). Коли відповідні швидкості навчання відрізняються за параметрами, це прискорює навчання.

3.1.4 Функція втрат при навчанні нейронної мережі

Функція втрат категорійної перехресної ентропії, також відома як логарифмічна функція втрат, використовується для оптимізації параметрів нейронної мережі. Для задач класифікації з кількома класами вона використовується для визначення різниці між реальним і очікуваним виходом алгоритму.

Таким чином визначається категорія перехресної ентропії:

$$\mathrm{CCE} = -\log\left(\frac{e^{s_p}}{\sum_j^C e^{s_j}}\right), \tag{3.7}$$

де s – вектор вихідних значень;

C – кількість класів.

Поєднує функції активації нейрона Softmax і втрати крос-ентропії:

$$f(s)_i = \frac{e^{s_p}}{\sum_j^C e^{s_j}}, \tag{3.8}$$

де f(s) – вектор, до якого застосована функція активації.

$$\mathrm{CCE} = -\sum_i^C t_i \log(f(s)_i), \tag{3.9}$$

де t_i – вектор передбачуваних ймовірностей;

s_i – вектор фактичних ймовірностей.

Завдяки функції Softmax кожен елемент вихідного вектору отримує експоненціальну функцію. Потім вектор нормалізується шляхом ділення елементів на їх суму. Таким чином, елементи знаходяться в діапазоні від 0 до 1, а їх загальна сума становить 1.

Розрахунок оцінки, яка узагальнює середню різницю між фактичними та передбачуваними значеннями, здійснюється за допомогою Cross-Entropy Loss алгоритму. Щоб підвищити точність моделі, цю оцінку слід зменшити до 0 або ідеального значення, а показник перехресної ентропії має бути від 0 до 1.

3.2 Проектування процесу розпізнавання жестів

Розпізнавання жестів було реалізовано за допомогою модуля комп'ютерного зору OpenCV. Відкривати, записувати та відображати зображення, відео та трансляції з веб-камери, захоплювати кадри та обробляти їх можна з його допомогою.

На рисунку 3.2 показано три етапи процесу розпізнавання жестів веб-камери:

Рисунок 3.2 – Процес розпізнавання жестів рук через веб камеру

Перший етап – знаходження зображення. Користувач відкриває програму та використовує жести. Кожну секунду веб-камера захоплює тридцять кольорових зображень і передає кожне з них на обробку.

Другий етап – попередня перевірка. Зібране кольорове зображення перетворюється на монохромне сіре та змінює розмір. Стандартизується значення пікселів до межі від 0 до 1.

Третій етап – розпізнавання. Нейронна мережа отримує оброблене зображення. Вона зрозуміла жест і повернула переклад.

3.3 Вибір алгоритмів і підходу машинного навчання

Згорткові нейронні мережі повинні бути використані для розпізнавання жестів машинного навчання. Вони спеціалізуються на розпізнаванні та класифікації об’єктів, отриманих через завантажене зображення або прямо отримані за допомогою комп’ютерного бачення завдяки своїй внутрішній побудові. Рисунок 3.3 показує структуру таких мереж.

Нейронні мережі такого типу відрізняються від інших тим, що вони можуть обробляти більш складні зображення з меншою залежністю від факторів, таких як якість і центрування. Хоча традиційні нейронні мережі добре розпізнають числа, вони погано класифікують речі. Використання згорткових шарів для фільтрації є основною причиною цієї невідповідності [9].

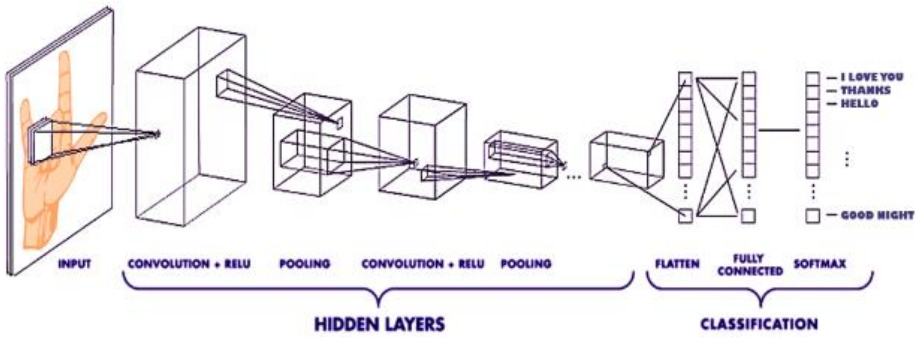

Рисунок 3.3 – Схема шарів згорткової нейронної мережі

На початку потрібно розглянути метод, за допомогою якого згортковий шар обчислює рівень, не використовуючи порівняння між мозком і нейроном. Параметри цього шару складаються з багатьох фільтрів, які можна

налаштувати. Хоча кожен з цих фільтрів має дуже невелику ширину та висоту, він охоплює всю глибину загальної площі.

Наприклад, стандартний фільтр на першому шарі CNN може бути представлений у форматі 5x5x3. Це означає, що фільтр має розміри 5 пікселів у висоту та 5 пікселів у ширину, а останнє число передає кількість кольорових каналів у цьому конкретному випадку. Під час прямого проходу фільтри стискаються на ширину та висоту вхідного зображення, а під час прямого проходу обчислюються локальні результати між записами фільтра та вхідними значеннями пікселів у будь-якому місці.

Після переміщення фільтра по ширині та висоті вхідного обсягу результатом буде двовимірна карта активації, яка містить результати проходження фільтра в кожній просторовій позиції. На першому рівні нейронна мережа розглядатиме фільтри, які спрацьовують під час аналізу певної форми візуальної характеристики (наприклад, колірної плями або межі елемента). На наступних рівнях нейронна мережа розглядатиме повні шаблони.

На початку потрібно розглянути метод, за допомогою якого згортковий шар обчислює рівень, не використовуючи порівняння між мозком і нейроном. Параметри цього шару складаються з багатьох фільтрів, які можна налаштувати. Хоча кожен з цих фільтрів має дуже невелику ширину та висоту, він охоплює всю глибину загальної площі.

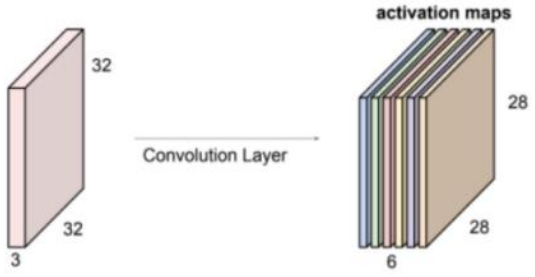

Рисунок 3.4 – Схема, яка зображує вихід після проходження через згортковий шар нейронної мережі

Наприклад, стандартний фільтр на першому шарі CNN може бути представлений у форматі 5x5x3. Це означає, що фільтр має розміри 5 пікселів у висоту та 5 пікселів у ширину, а останнє число передає кількість кольорових каналів у цьому конкретному випадку. Під час прямого проходу фільтри стискаються на ширину та висоту вхідного зображення, а під час прямого проходу обчислюються локальні результати між записами фільтра та вхідними значеннями пікселів у будь-якому місці.

Після переміщення фільтра по ширині та висоті вхідного обсягу результатом буде двовимірна карта активації, яка містить результати проходження фільтра в кожній просторовій позиції. На першому рівні нейронна мережа розглядатиме фільтри, які спрацьовують під час аналізу певної форми візуальної характеристики (наприклад, колірної плями або межі елемента). На наступних рівнях нейронна мережа розглядатиме повні шаблони.

Після проходження шару згортки в результаті буде отримана карта активації, яку необхідно нормалізувати за допомогою функції активації.

$$\mathrm{ReLU}(x) = \max(x, 0) \tag{3.10}$$

Мета цієї функції полягає в тому, щоб виключити всі негативні значення кожного пікселя, тобто просто прирівняти їх до нуля.

Архітектури основних згорткових нейронних мереж наведено нижче [10, 11]: модель R-CNN є першою та найшвидшою моделлю для виконання операцій класифікації зображень на основі окремих фрагментів.

Незважаючи на те, що Fast R-CNN і Faster R-CNN є вдосконаленими версіями попередньої моделі, вони не підходять для реального часу обробки.

Одним із найшвидших типів є дизайн YOLO, який зосереджений на знаходженні положення об’єкта за один прохід; архітектурний підхід SSD використовує ідею VGG16, але відрізняється від попереднього [12]. Він побудований на новій інформації та найкращих моментах YOLO. Будучи в

курсі переваг і недоліків різних методів реалізації згорткової мережі, було вирішено створити систему розпізнавання на SSD.

Коли ми розглядаємо цю архітектуру більш детально, ми бачимо, що конструкція SSD гірше працює на маленьких об’єктах, ніж Fast R-CNN.

Їх не можна знайти в кінцевому підсумку, лише на перших рівнях, де роздільна здатність вхідних даних ще більша. Але складність полягає в тому, що на даний момент можна класифікувати лише ознаки низького рівня, такі як кути, краї та кольорові плями, які надають мало інформації для ідентифікації; точність зростає прямо пропорційно кількості межових фрагментів, хоча і повільніше; конструкція фільтра впливає на підвищення точності; архітектура SSD має менше помилок локалізації, ніж R-CNN, але при однаковому введенні вона може мати SSD має краще покриття з точки зору розташування, розміру та співвідношення сторін і надає більше прогнозів. Завдяки попереднім удосконаленням SSD може знизити роздільну здатність вхідного зображення до 256 на 256 пікселів без втрати точності.

Після видалення призначеної області та використання зображень з нижчою роздільною здатністю модель може працювати в системах реального часу та виконувати завдання класифікації та розпізнавання значно швидше, ніж найсучасніший Fast R-CNN.

Для тих, хто використовує твердотільні накопичувачі, найкраща альтернатива - використання нової платформи MediaPipe від Google, яка була випущена в 2020 році.

У цій структурі є набір інструментів, які допомагають вирішити різноманітні проблеми категоризації та ідентифікації об’єктів [13]: обличчя та його контури; зіниці очей, рук, позиції тіла, волосся та об’єкти.

Метою кваліфікаційної роботи є використання інструментів платформи MediaPipe Hands, яка відповідає за ідентифікацію рук на вхідних зображеннях.

Модуль MediaPipe Hands використовує конвеєр ML, який складається з кількох моделей, які працюють разом. Одна з цих моделей - модель виявлення долоні, яка обробляє повне зображення та повертає орієнтовану обмежувальну

рамку руки. Модель використовує детектор долоні для орієнтації руки в обрізаній області зображення та повертає високоякісні тривимірні ключові точки рук. Інший варіант MediaPipe Face Mesh використовує детектор обличчя та модель орієнтації обличчя.

Надання моделі орієнтиру руки точно обрізаного зображення руки зменшує потребу в додаткових даних, таких як масштабування, трансляція та обертання. Натомість це дозволяє мережі використовувати більшу кількість своїх можливостей для точного прогнозування координат. Крім того, культури можуть бути створені на основі орієнтирів рук, визначених у попередньому кадрі. Коли модель орієнтира не може визначити присутність руки, запускається виявлення долоні для переміщення руки.

Конвеєр реалізується у вигляді графіка MediaPipe, який використовує підграф відстеження орієнтирів вручну з модуля ручного орієнтира та відтворює за допомогою спеціального підграфа ручної візуалізації. Підграф відстеження орієнтирів руки використовує орієнтир руки з того самого модуля, а підграф визначення долоні використовує орієнтир долоні з того самого модуля.

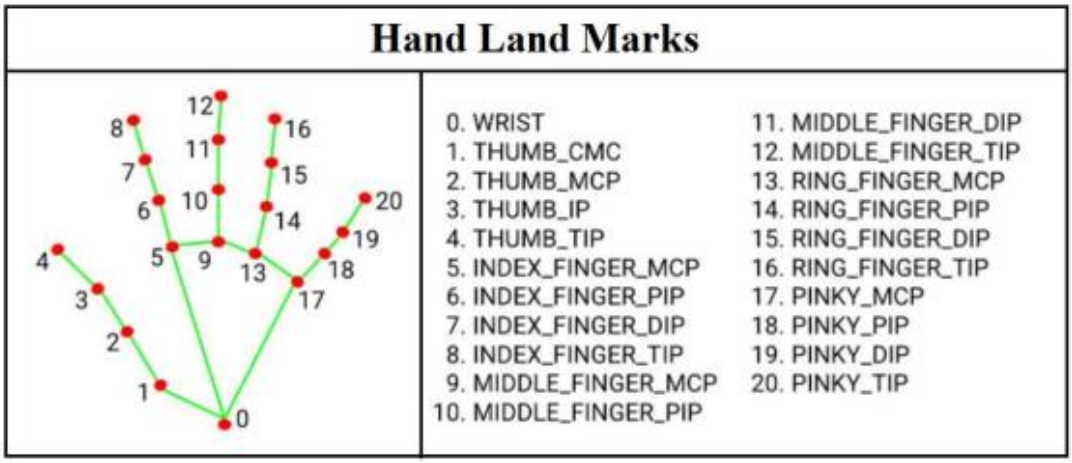

Рисунок 3.5 – Ключові точки для розпізнавання жестів руки для нейронної мережі

За допомогою регресії, тобто прямого прогнозування координат, модель орієнтира руки знаходить 21 ключову точку тривимірної координати кисті всередині визначених областей кисті після виявлення долоні на всьому зображенні. Модель вивчає внутрішнє подання пози та стійки рук, навіть коли руки, які частково видимі, перекриті іншою частиною тіла.

Крім того, високоякісна синтетична 3D-модель руки відтворюється на різних фонах, щоб краще охопити можливі пози руки та забезпечити більший контроль за природою геометрії руки. Як показано на рисунку 2.5, після завершення другого етапу нейронна мережа надає 21 важливу координату в тривимірному просторі.

3.4 Обрані технології та бібліотеки програмної мови

Python було обрано як базову мову програмування для цієї роботи. Python - це мова програмування загального призначення, яка поширена в будь-яких сценаріях. Його часто називають «об’єктно-орієнтованою мовою сценаріїв», що означає, що він поєднує підтримку операційної логіки (ООП) із загальною орієнтацією на роль сценарію. Фактично, коли люди описують файл коду Python, вони часто використовують слово «скрипт», а не «програма». Він часто вибирають для написання комплексних програм аналізу даних і машинного навчання, оскільки має багато ключових характеристик:

– програмне забезпечення високого рівня. Python орієнтований на читабельність, узгодженість і загальну якість, що відрізняє його від інших інструментів у сфері сценаріїв. Однорідність коду Python робить його простішим для розуміння, навіть якщо ви не написали його. Крім того, Python добре підтримує більш складні механізми повторного використання програмного забезпечення, як-от об’єктно-орієнтоване програмування (ООП);

– успіх розробника Окрім скомпільованих або статичних типизованих мов, таких як C, C++ та Java, Python значно підвищує продуктивність

розробників. Код Python зазвичай становить приблизно одну третину одна п’ята розміру коду C++ або Java. Це означає менше набору, налагодження та підтримки після факту. Крім того, програми Python запускаються миттєво, без тривалих процесів компіляції та посилань, необхідних для деяких інших інструментів, що ще більше прискорює роботу програміста;

– перенесення програми Більшість програм Python працюють на більшості основних комп’ютерних платформ без проблем;

– наприклад, код Python на Linux і Windows зазвичай зводиться лише до копіювання сценарійного коду між машинами. Крім того, Python надає широкий спектр можливостей кодування для переносних графічних користувальницьких інтерфейсів, програм доступу до баз даних, веб-систем і так далі. У Python навіть інтерфейси операційної системи, такі як програма запуску та обробка каталогів, настільки універсальні, наскільки це можливо;

– допомога бібліотек. Як стандартна бібліотека, Python має велику кількість попередньо вбудованих і портативних функцій. Від відповідності шаблону тексту до мережевих сценаріїв, ця бібліотека підтримує широкий спектр завдань програмування на рівні програми. Python можна розширювати як власними бібліотеками, так і великою колекцією програмне забезпечення, щоб підтримувати сторонні програми. Побудова веб-сайтів, числове програмування, доступ до послідовних портів, розробка ігор та багато іншого можна зробити за допомогою стороннього домену Python. Наприклад, бібліотеки OpenCV, NumPy та Keras будуть детально описані в наступних підрозділах. Використовуючи широкий спектр механізмів інтеграції, скрипти Python можуть легко взаємодіяти з іншими компонентами програми. Python тепер може використовуватися як інструмент для налаштування та розширення товарів завдяки цим інтеграціям. Сьогодні код Python має здатність викликати C і бібліотеки C ++, які можна викликати з програм C і C ++; ці бібліотеки можуть бути інтегровані з Java та іншими програмами.Компоненти NET мають можливості спілкування через такі фреймворки, як COM, послідовні порти для зв’язку з пристроями та мережеві

інтерфейси, такі як SOAP, XML-RPC та CORBA. Це не автономний інструмент.

3.4.1 Бібліотека OpenCV

Бібліотека машинного навчання та комп'ютерного зору з відкритим вихідним кодом. У ньому міститься понад 2500 алгоритмів, включаючи класичні та сучасні алгоритми для комп'ютерного зору та машинного навчання. Python, який використовується в цій роботі, Java, C++ і Matlab є кількома мовами, які входять до інтерфейсів цієї бібліотеки.

OpenCV має модульну структуру, тому він містить кілька спільних або статичних бібліотек у пакеті. Наступні модулі доступні. Основна функціональність (core) - це компактний модуль, який визначає основні структури даних. Він містить основні функції, які використовуються всіма іншими модулями, а також щільний багатовимірний масив Mat.

Модуль обробки зображень, відомий як imgproc, забезпечує лінійну та нелінійну фільтрацію зображень, геометричні перетворення зображень (зміна розміру, перекошування перспективи, загальне переназначення на основі таблиці), перетворення кольорового простору, гістограми та інші функції.

Відеоаналіз - це модуль, який включає оцінку руху, віднімання фону та алгоритми відстеження об'єктів.

Калібрування камери та 3D-реконструкція (calib3d) - це основні алгоритми геометрії декількох видів, калібрування однокамерної та стереокамери, алгоритми оцінки пози об'єкта, алгоритми стереовідповідності та елементи 3D-реконструкції.

Модель 2D Характеристики (features2d) містить основні детектори, дескриптори та збіги дескрипторів.

Виявлення об'єктів (objdetect) - це процес пошуку об'єктів і екземплярів заздалегідь визначених класів, таких як обличчя, очі, кружки, люди, машини тощо.

Графічний інтерфейс високого рівня (highgui) - це простий інтерфейс, який має прості можливості.

Інтерфейс Video I/O, також відомий як videoio, є простим у використанні для зйомки відео та відеокодеків;

OpenCV-Python - це бібліотека прив'язок Python, призначена для вирішення проблем із комп'ютерним зором;

Numpy, високооптимізована бібліотека для числових операцій із синтаксисом, схожим на MATLAB, використовується в OpenCV-Python. Усі структури масивів OpenCV перетворюються на масиви Numpy, і з них починається їх використання. Крім того, це полегшує інтеграцію з іншими Numpy-бібліотеками, такими як SciPy та Matplotlib.

Бібліотека може виконувати різноманітні перетворення як статичних зображень, так і послідовностей кадрів, таких як відео, і використовувати отримані результати для машинного навчання за допомогою модуля objdetect.

Бібліотека OpenCV була зібрана за допомогою CMake.

CMake - це відкритий, крос-платформовий генератор сценаріїв складання.

CMake не займається безпосередньо складанням, а лише створює файли управління складанням з файлів CMakeLists.txt[24]. Ці функції включають складання Makefile в системах Unix, використання make для збірки файлів projects/workspaces (.dsp/.dsw) у Windows і проект XCode для Mac OS X.

У CMake багато переваг, таких як проста мова сценаріїв, можливість розширення функціональності через модулі, мінімальне число залежностей (немає зв'язку з M4, Perl або Python), підтримка кешування, наявність засобів для крос-компіляції, підтримка генерації файлів складання для різних систем складання та компіляторів, наявність утиліт ctest і cpack для визначення сценаріїв тестування та складання пакунків, а також утиліт.

CMake використовується в таких проектах, як KDE, LLVM/Clang, MySQL, MariaDB, ReactOS і Blender, і служить альтернативою Autotools. Серцевий код CMake доступний під ліцензією BSD і створений мовою C++.

3.4.2 API Keras

Незважаючи на те, що глибокі нейронні мережі дуже популярні, розробникам важко використовувати їх через складність основних фреймворків. Було кілька пропозицій щодо вдосконалених і спрощених API високого рівня для побудови моделей нейронних мереж, які, як правило, здаються схожими на відстань; однак більш детальне вивчення показує, що вони відрізняються.

Keras є одним із найкращих API для нейронних мереж високого рівня. Кілька внутрішніх механізмів обчислень нейронних мереж підтримуються в коді, написаному на Python. Керас працює з Python, є модульним і простим у використанні. «Розроблений для людей, а не для машин» і «дотримувався найкращих практик зменшення когнітивного навантаження», API був створений.

Для створення нових моделей ви можете об’єднати різні модулі, такі як нейронні рівні, функції витрат, оптимізатори, схеми ініціалізації, функції активації та схеми регуляризації. Просто додайте нові модулі, включаючи нові класи та функції. Використовуючи код Python, а не окремі файли конфігурації моделі, моделі створюються. Рисунок 3.6 показує структуру такої конфігурації.

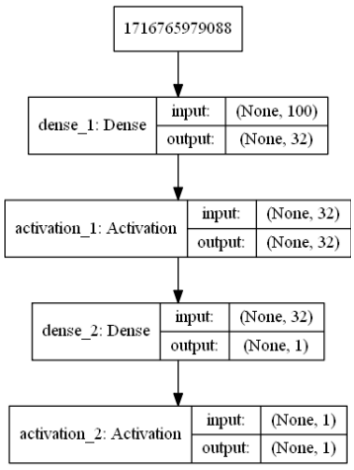

Рисунок 3.6 – Модель Keras

3.4.3 Бібліотека Numpy

Numerical Python або NumPy - це бібліотека Python з відкритим кодом, яка використовується майже в кожній галузі науки та техніки. Це основа наукових екосистем Python та PyData і є універсальним стандартом роботи з числовими даними в Python. NumPy використовують усі, від початківців до досвідчених дослідників, які проводять надзвичайно сучасні промислові та наукові дослідження та розробки.

Бібліотека NumPy містить матричні структури даних і масиви даних, які мають кілька вимірів. Додаткову інформацію про це ви знайдете в наступних розділах. Він надає ndarray, однорідний n-вимірний об’єкт масиву, а також методи ефективної роботи з ним. NumPy може виконувати різні математичні операції над масивами. Він додає до Python потужні структури даних, які гарантують ефективні обчислення за допомогою масивів і матриць. Крім того, він надає велику бібліотеку математичних функцій високого рівня, які працюють з цими масивами та матрицями.

Хоча список Python може включати в себе кілька різних типів даних, всі елементи масиву NumPy повинні бути однорідними. Якби масиви не були однорідними, математичні операції над ними були б неможливими. Массиви NumPy компактніші та швидші за списки Python. Масив набагато зручніший у використанні та споживає менше пам'яті. NumPy надає механізм визначення типів даних, що дозволяє ще більше оптимізувати код, і він використовує значно меншу кількість пам’яті для зберігання даних.

Масив є основною структурою даних бібліотеки NumPy. Це сітка значень, яка містить інформацію про вихідні дані, способи пошуку елемента та методи його інтерпретації. Він містить сітку елементів, які можна індексувати кількома способами. Масив dtype (тип даних) означає всі елементи одного типу. Можна проіндексувати масив набором цілих неотрицальних чисел, булевих значень, іншого масиву або цілих чисел. Ранг масиву є мірою кількості. Набір цілих чисел, які визначають розмір масиву вздовж кожного виміру, називається формою масиву. Вкладені списки Python

можна використовувати для ініціалізації масивів NumPy: a = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]).

Квадратні дуги дозволяють нам отримати доступ до елементів масиву. Пам’ятайте, що коли ви отримуєте доступ до елементів, індексація NumPy починається з 0. Це означає, що ви можете отримати доступ до елемента „0”, якщо ви хочете отримати доступ до першого елемента у своєму масиві.

Як правило, масив є контейнером фіксованого розміру, у якому всі елементи мають однаковий тип і розмір. Форму масиву визначає кількість його розмірів і елементів. Сукупність невід’ємних цілих чисел, які визначають розміри кожного виміру, називається формою масиву. У NumPy ось називається розміром. Це означає, що якщо у вас є двовимірний масив, який виглядає так: [[0., 0., 0.], [1., 1., 1.]].

3.4.4 Згорткова нейронна мережа (Convolutional Neural Network)

Convolutional Neural Network, також відома як Згорткова Нейронна Мережа, є логічним розвитком концепцій когнітронних і неокогнітронних архітектур, і вона дала найкращі результати в області розпізнавання людей. Успіх залежить від можливостей обліку топології зображення в двох вимірах, на відміну від багатошарового персептрона.

Згорткові нейронні мережі здатні протистояти змінам масштабу, зсувам, поворотам і змінам ракурсу, серед інших видів спотворень. Три архітектурні ідеї використовуються в згорткових нейронах для забезпечення інваріантності до зміни масштабу, повороту зрушення та просторового спотворення:

– локальні рецепторні поля, які забезпечують двовимірну зв’язність нейронів;

– загальні вагові коефіцієнти синапсів, які забезпечують виявлення деяких рис у будь-якому місці зображення і зменшують загальне число вагових коефіцієнтів;

– ієрархічна організація з просторовими підвибірками.

ЗНМ складається з трьох різних шарів: персептрона, який є «звичайною» нейронною мережею, субдискретизуючих шарів (які можуть бути отримані шляхом підвибірки, підбирки або сублімації) і згорткових (конволівних) шарів.

Згорткові мережі є хорошим компромісом між біологічно правдоподібними мережами та традиційним багатошаровим персептроном. Наразі їх використовують для покращення розпізнавання зображень. В середньому точність розпізнавання таких мереж перевищує точність звичайних ІНС на 11–17 відсотків. ЗНО є основною технологією deep learning. Концепція загальних ваг стала основою успіху СНС.

В порівнянні з їх предком, неокогнітронними мережами, ці мережі мають небагато параметрів, які можна налаштувати, незважаючи на їх величину. В таких варіантах ЗНМ, які схожі на неокогнітрони, відбувається часткова відмова від пов’язаних ваг. Однак алгоритм навчання, який базується на зворотному поширенні помилки, залишається незмінним. Чисте розпаралелювання процесу згортки по кожній карті та зворотна згортка при поширенні помилки по мережі дозволяє СНС працювати на послідовній машині та навчатися швидко.

3.5 Обґрунтування вибору програмної реалізації

У процесі проектування системи було вивчено та оцінено сферу діяльності та вимоги клієнта. Після детального аналізу було вирішено створити програмний продукт, який використовує бібліотеку OpenCV. Для цього інструмент CMake, мови програмування Python, бібліотеки pytorch і бібліотеки vlc використовувалися для обробки аудіо та медіа інформації.

Python - це мова програмування загального призначення, створена, щоб допомогти програмістам бути більш продуктивними. Python дозволяє писати практично все, що завгодно, включаючи веб-/настольні додатки, ігри, автоматизовані скрипти, комплексні системи розрахунку, системи управління

життєзабезпеченням і багато іншого, без будь-яких проблем. Подальший супровід програм, написаних на Python, стає легшим і приємнішим за рахунок простоти коду, а з точки зору бізнесу це призведе до зниження витрат і підвищення продуктивності працівників[27].

Бібліотека pyTorch була обрана для створення моделі нейронної мережі, оскільки це науковий обчислювальний пакет на основі Python, який використовує потужність графічних процесорів. Це також одна з кращих дослідницьких платформ глибокого навчання, розроблена для швидкості та гнучкості. Він відомий тим, що виконує дві з найбільш високорівневих функцій: побудова глибоких нейронних мереж і тензорні обчислення з сильною підтримкою прискорення GPU[28].

Можливість створювати власні моделі нейронних мереж є основною причиною успіху PyTorch, оскільки він повністю використовує мову Python.

Одним із основних елементів PyTorch є простий інтерфейс: він пропонує простий у використанні API, що робить його дуже простим у управлінні та працює як Python. Обчислювальні графіки. Крім того, PyTorch надає чудову платформу, яка надає динамічні обчислювальні графіки, що дозволяє їх змінювати під час виконання.

Бібліотека OpenCV була обрана для препроцесингу, оскільки вона написана на C, добре оптимізована та може використовувати переваги багатоядерних процесорів. OpenCV використовує CPU, коли він встановлюється стандартними пакетами, що впливає на продуктивність. В результаті було вирішено зібрати цю бібліотеку для GPU власноруч за допомогою інструменту cMake і вихідного коду.

Для забезпечення розпізнавання людських жестів була розроблена власна модель, а дані були отримані з датасету Kaggle. Цей датасет був обраний, оскільки він містить більше 6000 зображень для п’ятнадцяти різних жестів, зроблених камерою LeapMotion.

Бібліотека vlc була обрана для роботи з аудіо інформацією, оскільки вона містить проекти з відкритим вихідним кодом і має значну підтримку з

боку розробників. Велика швидкість роботи, надійність і численні вбудовані можливості також є важливими характеристиками.

Поєднання даних технологій дозволяє створювати високоякісний та надійний продукт, який захищений від патентних позовів розробників, оскільки всі ці технології захищені ліцензіями, які надають доступ до вихідних кодів даних проектів.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ НЕЙРОННОЇ МЕРЕЖІ ДЛЯ РОЗПІЗНАВАННЯ ЖЕСТІВ РУК

Систему розпізнавання жестів можна умовно розділити на два рівні. Перший рівень – призначений для користувача системи управління і є людиномашинним інтерфейсом, за допомогою якого система керується. Другий рівень – інтерфейс взаємодії, який використовується для навчання нейронної мережі.

4.1 Структура програми

Розроблений програмний продукт використовує ШНМ і КЗ для розпізнавання жестів. Переклад відбувається з веб-камери, записаного відео або теки з зображеннями. Коли ви вибираєте переклад з веб-камери повинні транслювати жести в режимі реального часу. Переклад з відео або теки з зображеннями можна зробити, вибравши відповідне відео або теку.

Процес навчання системи розпізнавання жестів є першим етапом взаємодії з комп’ютером. Програма відкриє вікно з відеопотоком і виведе на екран назву жесту та номер кадру, якому вона зараз навчається. У цьому моменті потрібно зафіксувати жест перед камерою; для кращого результату після кожного повідомлення про успішну фіксацію кадру потрібно змінювати руки.

У кваліфікаційній роботі використовується метод навчання нейронної мережі, відомий як «навчання з вчителем». Таким чином, щоб навчити програму розпізнавати жести, їй потрібно надати набір жестів. Після цього система почне навчатися на основі цих даних.

Процес навчання має бути постійним. Система не має процесу донавчання. Отже, щоб розпізнати новий жест, потрібно заново запускати процес навчання. Цей недолік, однак, можна виправити в майбутньому.

Налаштування на тридцять кадрів на один жест займає в середньому десять хвилин. Отже, щоб бути готовим до завершення навчання, потрібно брати до уваги цей показник, оскільки він не може бути зупинений.

Дев’ять файлів початкового коду з розширенням містять код додатку:

– dataset.py – тут створюються два файли, які розширюються для тестування та тренування.csv, що містить посилання на зображення та класи цих зображень;

– gesture_dataset.py – включає клас, який може керувати створеними файлами;

– network.py – у цьому файлі міститься клас ЗНМ з ініціалізацією шарів і атрибутами класу, а також метод опису переміщення даних по нейронній мережі;

– model.py – файл, який містить клас нейромережевої моделі розпізнавання. Побудова графіків втрат і точності, а також функції збереження та навчання моделі;

– torch_converter.py – використовує збережену модель PyTorch як модель ONNX для роботи з модулем комп’ютерного зору;

– image_detector.py – файл містить список класів зображень, можливості підготовки зображення до передачі на натреновану нейронну мережу, можливості передачі зображення на нейронну мережу та можливості виводу перекладу;

– webcam_recognition.py – цей файл використовує зчитування зображень з веб-камери та показ перекладу на екран;

– webcam_ui.py – містять опис графічного інтерфейсу користувача;

– option_ui.py – опис інтерфейсу користувача для файлу option.py.

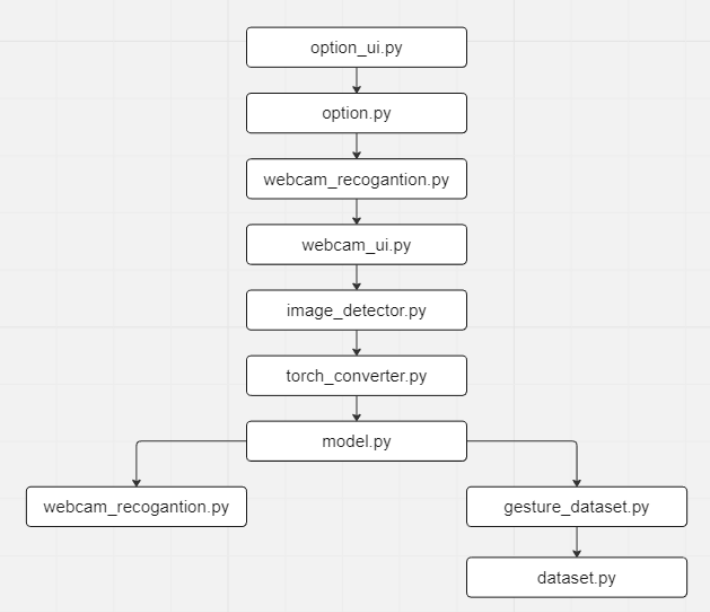

Рисунок 4.1 – Блок схема взаємодії компонентів програми

4.2 Набір даних для навчання нейронної мережі

У папках train і test знаходяться фотографії навчання та тестування відповідно. У назві файлу міститься номер класу, виділений символом «_». Цей номер можна виокремити та зберегти за допомогою регулярних виразів.

Назва файлу та номер класу будуть представлені в двох колонках створеного набору даних (рисунок 4.2).

	A	B
1	filename	class
2	image_10_0.jpg	10
3	image_10_1.jpg	10
4	image_10_10.jpg	10
5	image_10_11.jpg	10
6	image_10_13.jpg	10
7	image_10_14.jpg	10
8	image_10_15.jpg	10
9	image_10_16.jpg	10
10	image_10_17.jpg	10
11	image_10_18.jpg	10

Рисунок 4.2 – Таблиця з назвами файлів для навчання

Щоб натренувати нейронну мережу, також були використнні датасети з API Keras, які знаходяться у відкритому доступі.

```
def datasetCreation():
    directory = './images/'

    imagesFolders = os.listdir(directory)

    for fldr in imagesFolders:
        csv_data = {'filename': [], 'class': []}
        fileNames = [os.path.basename(filename) for filename in os.listdir(directory + fldr)]
        csv_data['filename'].extend(fileNames)
        for flnm in fileNames:
            className = int(re.findall(r'_(.*?)_', flnm)[0]) - 1
            csv_data['class'].append(className)

        df = pd.DataFrame(data=csv_data)
        df.to_csv(directory + fldr + '_labels.csv', index=False)
```

Рисунок 4.3 – Код функції, яка використовується для створення набору даних

За допомогою цієї функції будуть створені два файли: train_labels.csv і test_labels.csv. У подальшому створюється клас набору даних, який містить три магічні методи __init__, __len__ і __getitem__, як показано на рис. 4.3. Це підклас абстрактного класу Dataset, який представляє набір даних у фреймворці PyTorch.

Annotations, directories і transforms створюються методом __init__, який є конструктором класу. Змінна annotations містить вміст файлу csv, а директорія вказує шлях до зображень із зазначеними іменами у файлі. transforms - це трансформації, які застосовуються до зображень, про які йдеться пізніше. Довжина змінної annotations повертається за допомогою методу __len__. Метод __getitem__ використовує індекс для пошуку та повернення зображення та його класу. Крім того, до зображення застосовуються відповідні трансформації, якщо значення змінної трансформації не дорівнює False.

```
class GestureRecognitionDataset(Dataset):
    def __init__(self, csv_file, directory, transform=None):
        self.annotations = pd.read_csv(csv_file)
        self.directory = directory
        self.transform = transform

    def __len__(self):
        return len(self.annotations)

    def __getitem__(self, index):
        imagePath = os.path.join(self.directory, self.annotations.iloc[index, 0])
        image = io.imread(imagePath)
        y_label = torch.tensor(int(self.annotations.iloc[index, 1]))

        if self.transform:
            image = self.transform(image)

        return image, y_label
```

Рисунок 4.4 – Код класу зі створення датасетів

4.3 Згорткова нейронна мережа

Створюється клас, який наслідує клас nn для реалізації ЗНМ. Module є базовим класом для всіх нейромереж, доступних у бібліотеці PyTorch. У класі є два методи: __init__ і перед.

В конструкторі створюються чотири шари ЗНМ: три пулінгові, три згорткові та два повноз’єднані. Параметри, які можна змінити, включають кількість вхідних і вихідних каналів, а також розмір вікна згортки.

Метод передавання описує переміщення даних через нейронну мережу. Цей метод створює одновимірний масив з n необроблених і ненормалізованих елементів. Кожен елемент масиву є лінійною комбінацією виходів попереднього шару, помножених на ваги та доданих зсуву, де n - кількість класів. Вірогідності кожного класу можна отримати, застосовуючи функцію Softmax до масиву.

```
class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(1, 8, (5, 5))
        self.conv2 = nn.Conv2d(8, 16, (5, 5))
        self.conv3 = nn.Conv2d(16, 32, (5, 5))

        self.pool = nn.MaxPool2d(2, 2)

        self.fc1 = nn.Linear(32 * 4 * 4, 256)
        self.fc2 = nn.Linear(256, NUM_CLASSES)

    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = self.pool(F.relu(self.conv3(x)))
        x = x.view(-1, 32 * 4 * 4)
        x = F.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Рисунок 4.5 – Реалізація нейронної мережі

Для навчання та створення нейронної мережі для розпізнавання жестів створюється клас з методами __init__, train, test, saveModel, buildPlot.

В конструкторі ініціалізуються зміні directory, batch_size, num_epochs, img_size. directory – папка з файлами з наборами даних, batch_size – розмір партії зображень, що буде поступати на нейронної мережі, num_epochs – кількість епох, яку буде навчатися модель, img_size – розмір зображення для навчання.

Набори даних визначаються як для навчання, так і для тестування. Відповідно до рисунку 4.5, для навчального набору кольорові зображення перетворюються в сірі, а розмір змінюється відповідно до визначення конструктора. Щоб урізноманітнити набір даних, зображення випадковим чином віддзеркалюються по вертикалі та повертаються на випадковий градус у діапазоні від тридцяти до тридцяти градусів, а також вони нормалізуються для тестового набору даних.

```
self.train_transform = transforms.Compose([transforms.ToPILImage(),
                                           transforms.Grayscale(),
                                           transforms.Resize((self.img_size, self.img_size)),
                                           transforms.RandomHorizontalFlip(),
                                           transforms.RandomRotation(degrees=(-30, 30)),
                                           transforms.ToTensor(),
                                           transforms.Normalize([0.5], [0.5])])

self.test_transform = transforms.Compose([transforms.ToPILImage(),
                                          transforms.Grayscale(),
                                          transforms.Resize((self.img_size, self.img_size)),
                                          transforms.ToTensor(),
                                          transforms.Normalize([0.5], [0.5])])
```

Рисунок 4.6 – Обробка зображення перед навчанням та тестуванням

Також у конструкторі визначаються функція втрат та модуль оптимізації.

Метод тренування передбачає навчання нейромережевої моделі розпізнавання. На нейромережу надсилається пакет зображень. Для розрахунку втрати вхід функції втрат отримує вихід ЗНМ, передбачуваний клас і цільовий клас. Для використання методу loss.backward, який накопичує градієнт для кожного параметру, але спочатку потрібно обнулити всі градієнти за допомогою функції optimizer.zero_grad. Після цього метод оптимізатора.крок змінює параметри відповідно до поточного градієнта та правил оновлення. Крім того, на кожному кроці тренування зберігаються значення втрати та точності; графік буде побудовано за цими параметрами пізніше (рисунок 4.6).

У методі тестування нейромережева модель розпізнавання тестується. Навчена модель отримує зображення, які вона раніше не бачила.

Передбачення моделі порівнюються з правильними результатами; змінна n_correct містить кількість правильних передбачень, а змінна n_samples містить загальну кількість тестів. Як показано на рисунку 3.8, загальна кількість тестів ділиться на кількість правильних передбачень для отримання тестувальної точності.

```
def train(self):
    for epoch in range(self.num_epochs):
        self.running_loss = 0.0
        for i, (images, labels) in enumerate(self.train_loader):
            outputs = self.net(images)
            loss = self.loss_function(outputs, labels)
            _, predictions = torch.max(outputs, 1)

            self.optimizer.zero_grad()
            loss.backward()
            self.optimizer.step()

            self.running_loss += loss.item() * images.size(0)

            self.n_samples += labels.size(0)
            self.n_correct += (predictions == labels).sum().item()

            if (i + 1) % 10 == 0:
                print(f'epoch {epoch + 1}/{self.num_epochs}, step {i + 1}/{self.n_total_steps}, loss = {loss.item():.4f}')
        acc = 100.0 * self.n_correct / self.n_samples
        self.loss_values.append(self.running_loss / len(self.train_dataset))
        self.acc_values.append(acc)
```

Рисунок 4.7 – Фрагмет коду методу навчання моделі

Навчена нейронна мережа може бути збережена за допомогою методу saveModel. Фінкція torch.save(self.net.state_dict()) зберігає модель нейромережі та її навчені параметри.

Графіки точності та втрат, які були збережені під час навчання нейронної мережі, можна знайти за допомогою методу buidPlot (рисунок 4.8).

```
def buidPlot(self):
    ax1 = plt.subplot2grid((2, 1), (0, 0))
    ax2 = plt.subplot2grid((2, 1), (1, 0), sharex=ax1)

    ax1.plot(self.acc_values, label="accuracies")
    ax1.legend(loc=2)
    ax2.plot(self.loss_values, label="losses")
    ax2.legend(loc=2)
    plt.savefig(f'plot_{self.num_epochs}_epochs.png', dpi=300)
    plt.show()
```

Рисунок 4.8 – Метод побудови графіків при тестуванні

Точність можна визначити як відсоток кількості правильних передбачень, зроблених за певну епоху. Втрата – це розрахована за допомогою відповідної функції втрат помилки, яка описує різницю між реальними та отриманими значеннями. Застосовується для підвищення ефективності

параметрів нейронних мереж. Для побудови графіків використовуються інструменти бібліотеки Matplotlib, як показано на рисунку 4.9.

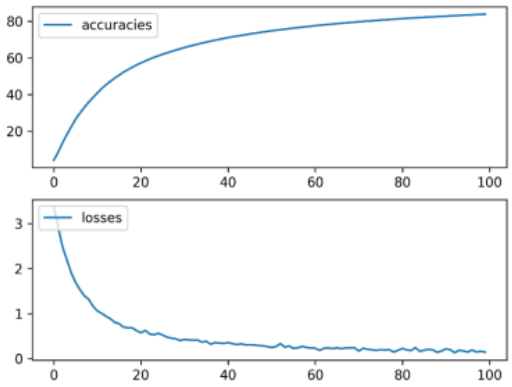

Рисунок 4.9 – Графік точності та втрат при навчанні

4.4 Розпізнавання жестів з веб-камери

В файлі webcam_recognition містяться класи WebcamRecognition (клас інтерфейсу) і Worker1 (клас потоку), які відповідають за розпізнавання жестів веб-камери та керують запуском і зупинкою трансляції.

Запуск веб-камери виконується функцією cv2 в методі запуску класу Worker1.VideoCapture(0), де номер веб-камери становить 0. У результаті виконання процедур, розпізнана літера виводиться в текстове поле інтерфейсу (рисунок 4.10).

```
def run(self):
    self.ThreadActive = True
    capture = cv2.VideoCapture(0)
    letter_list = []
    while self.ThreadActive:
        ret, img = capture.read()
        if ret:
            img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
            img = cv2.flip(img, 1)

            convertToQtFormat = QImage(img.data, img.shape[1], img.shape[0], QImage.Format_RGB888)
            pic = convertToQtFormat.scaled(350, 350, Qt.KeepAspectRatio)
            self.image_update.emit(pic)

            img = imagePreparation(img, IMG_SIZE, False)
            predicted_letter, probability = imageClassification(img, IMG_SIZE)
```

Рисунок 4.10 - Функція run

За вибір теки для розпізнавання зображення відповідає за метод selectFolder. Усі зображення передаються на нейронну мережу після вибору теки, а переклад відразу виводиться в текстове поле. Перше зображення виводиться на екран.

```
def selectFolder(self):
    self.textBrowser.clear()
    self.directory = str(QFileDialog.getExistingDirectory(self, "Select Directory"))
    self.image_list = glob.glob(f'{self.directory}/*.jpg')
    if self.image_list:
        self.image_updateSlot(self.image_list[0])
        self.current_image = self.image_list[0]
        self.updateLabel()
        self.textBrowserUpdate()
    return self.directory
```

Рисунок 4.11 – Функція вибору папки

Метод image_updateSlot змінює зображення в інтерфейсі програми. Метод textBrowserUpdate виводить літери в текстове поле.

5 ТЕСТУВАННЯ СИСТЕМИ

5.1 Розпізнавання жестів рук

Нейронна мережа навчалася на фотографіях жестів рук однієї людини, тому вона може помилково розпізнавати жести рук інших людей.

У розділі наведено необхідні апаратні та системні компоненти для встановлення системи розпізнавання жестів, яка використовує машинне навчання. Перерахував можливі сценарії хибних значень і розповів про результати своєї роботи.

Рисунок 5.1 – Тестове слово «Hello»

Таким чином, зображення рук двох інших людей будуть використані для тестування програми. Для перевірки правильної роботи, було складено слово «Hello». Тестовий набір жестів був перекладений за допомогою веб камери.

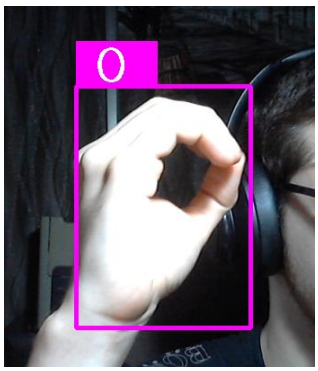

Рисунок 5.2 – Приклад розпізнавання літери «O» з ASL

Іноді, модель розпізнавання жестів мови глухонімих може робити помилки, оскільки на розпізнавання впливає дуже велика кількість стороніх факторів: освітлення, якість зйомки веб камери, ракурс з якого знімає камера, кут нахилу руки з жестом відносно об’єктиву та інші.

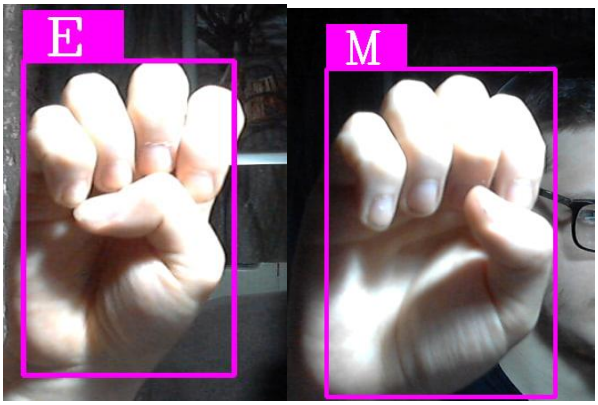

Рисунок 5.3 – Похибка при розпізнаванні літери «E» в ASL

Така похибка при користуванні обумовлена тим, що жести для літери «E» та «M» у алфавіті ASL дуже схожі – єдина відмінність полягає у підігнутому великому пальці руки. Саме він дає таку похибку при розпізнаванні.

Рисунок 5.4 – Відмінність між літерами у алфавіті ASL

Під час тестування десять з дев’яти жестів було розпізнано, тестування можна вважати вдалим, і цю помилку можна виправити, збільшуючи кількість

навчальних зображень. Серед недоліків можна визначити, довгий час розпізнавання з деякими похибками при неідеальних умовах.

5.2 Керівництво користувача

Дана система розпізнавання жестів рук має два режими роботи:

– режим розробника – у цьому режимі на екрані доступно більше інформації про процес роботи програми, яка може бути корисна саме для розробника;
– режим користувача – у цьому режимі на екрані показуються тільки ті елементи інтерфейсу, з якими буде взаємодіяти саме кінцевий користувач.

Для запуску програми у режимі розробника необхідно перейти у папку проекта та виконати команди на рисунку 5.5.

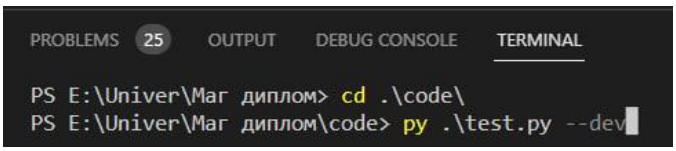

Рисунок 5.5 – Запуск програми у режимі розробника

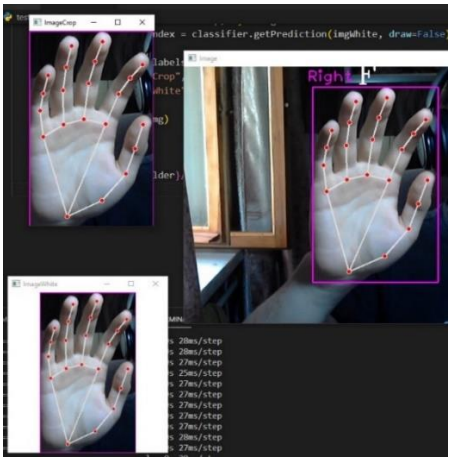

Рисунок 5.6 – Інтерфейс програми у режимі розробника

Під час розробки, дуже необхідно бачити усі нюанси програми. У режимі розробника доступно 3 вікна:

– головне вікно – вікно, на яке виводиться зображення з камери, а також інтрефейс розпізнаваня жестів руки(сітка з ключових точок руки, напис з тим, яка рука була розпізнана і який жест зараз показує людина);
– вікно imageCrop – це вікно, у якому виводиться лише розпізнана рука. Це було зроблено для відслідковування правильного розпізнавання руки навченою нейронною мережею;
– вікно imageWhite – вікно, у якому виводиться розпізнана рука, але х потрібними розмірами, для навчання моделі.

Для роботи програми у режимі кінцевого користувача необхідно запустити наступні команди.

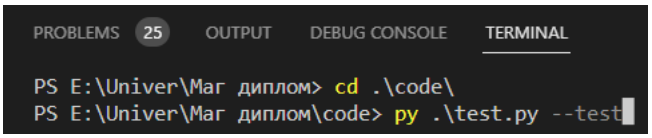

Рисунок 5.7 – Запуск програми у режимі користувача

Після запуску програми у режимі розробника, користувач може побачити наступний інтерфейс:

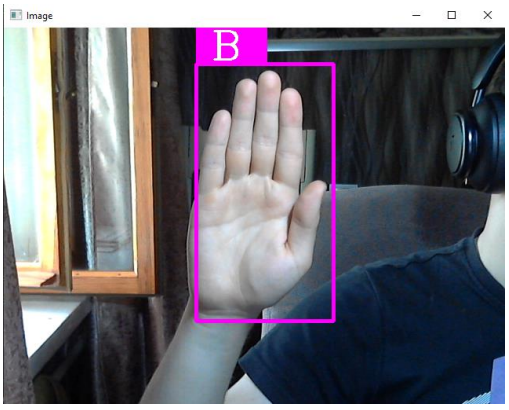

Рисунок 5.8 - Інтерфейс програми у режимі користувача

Як можемо бачити, у режимі користувача лише один екран – головний. На ньому відображається відео з веб камери, а також інтерфейс розпізнавання жесту руки. Коли нейрона мережа розпізнає руку, вона обводить її по контуру. Якщо ж нейроній мережі вдалося розпізнати будь-який жест руки, вона додатково показує відповідну літеру з алфавіту ASL.

Користувач може на веб-камеру у реальному часі змінювати жести руки, а нейрона мережа буде їх розпізнавати та виводити цю інформацію на екран користувача.

ВИСНОВКИ

В кваліфікаційній роботі вирішується проблема розпізнавання жестів мови глухонімих у реальному часі, які можна використовувати для взаємодії людей з обмеженими можливостями у цифровому просторі. Завдання є актуальним через його практичність, велику кількість проведених досліджень у цій галузі, і тим, що якість існуючих алгоритмів розпізнавання жестів рук і пальців, як динамічних, так і статичних, для використання кольорових відеокамер і тривимірних сенсорів, все ще недостатня для побудови практичних систем людино-машинного взаємодії.

Отримані основні результати: виконана необхідна формалізація, розроблено та досліджено різноманітні методи та алгоритми для попередньої та спеціальної обробки дальностних зображень. До них належать: алгоритм, який використовує один прохід для вилучення зображення руки з дальностного зображення за відомою позицією долоні зі складністю O(n), де n – кількість точок вихідного зображення; метод розпізнавання позицій долоні та кінчиків пальців руки в тривимірному просторі

Необхідні теореми та твердження були розроблені та підтверджені. Було вивчено тему, досліджено та проаналізовано роботи вітчизняних і іноземних авторів, було зібрано власний набір даних, спроектовано нейронну мережу та схематично розглянуто процес розпізнавання.

Алгоритм перекладу жесту в літери алфавіту також був розроблений. Після розробки програма була протестована.

Серед недоліків системи можна виділити наступні речі:

– недостатня швидкодія;
– короткий процес навчання моделі, що може призводити до неточностей при розпізнаванні мови глухонімих;
– інтерфейс користувача не має допоміжного функціоналу.

Серед переваг розробленої системи можна виділити такі аспекти:

- не високі технічні вимоги;
- навіть при короткому періоді навчання моделі нейронної мережі, система має досить високий рівень успішного розпізнавання усіх жестів алфавіту ASL;
- легко може бути розширена додатковим функціоналом.

Оскільки подібних програм наразі немає у публічному доступі, додаток є актуальним. Крім того, це система з відкритим вихідним кодом та набір даних, тому користувачі можуть змінювати або доповнювати датасет і навчати мережу за потреби.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. T. Starner, J. Weaver, A. Pentland. Real-time American sign language recognition using desk and wearable computerbased video. Pattern Analysis and Machine Intelligence [Electronic resource]. – 1998. – Access mode: https://ieeexplore.ieee.org/document/735811/ (lastaccess: 04.06.2021). Title from the screen.

2. Y. LeCun et al. Deep learning [Electronic resource]. – 2015. – Access mode: https://www.nature.com/articles/nature14539 (lastaccess: 04.06.2021). Title from the screen.

3. M.Z. Islam, M.S. Hossain, R. Islam and K. Andersson. Static Hand Gesture Recognition using Convolutional Neural Network with Data [Electronic resource]. – 2019. – Access mode: http://www.diva-portal.org/smash/get/diva2:1299000/ FULLTEXT01.pdf/ (lastaccess: 01.06.2021). Title from the screen.

4. V. Nair, G.E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines [Electronic resource]. – 2019 – Access mode: https://www.cs.toronto.edu/ ~fritz/ absps/reluICML.pdf/ (lastaccess: 01.06.2021). Title from the screen.

5. Гафаров Ф.М. Искусственные нейронные сети и приложения: учеб. пособие / Ф.М. Гафаров, А.Ф. Галимянов. – Казань: Изд-во Казан. ун-та, 2018. – 121 с.

6. K. Simonyan, A. Zisserman. Very deep convolutional neural networks for large-scale image recognition Network [Electronic resource]. – 2015. – Access mode: https://arxiv.org/abs/1409.1556/ (дата звернення 31.05.2021). – Title from the screen.

7. M.D. Zeiler, R. Fergus. Visualizing and Understanding Convolutional Neural Networks [Electronic resource]. – 2014. – Access mode: https://arxiv.org/ abs/1409.1556/ (lastaccess: 31.05.2021). – Title from the screen.

8. M. Lin, Q. Chen, and S. Yan. Network In Network [Electronic resource]. – 2013. – Access mode: https://arxiv.org/pdf/1312.4400/ (lastaccess: 31.05.2021). – Title from the screen

9. Python BeginnersGuide Overview: URL: https://wiki.python.org/moin/BeginnersGuide/Overview (Дата звернення 18.04.2022)

10. Matplotlib – Pyplot tutorial: URL: https://matplotlib.org/stable/tutorials/introductory/pyplot.html#sphx-glr-tutorials introductory-pyplot-py (Дата звернення 23.04.2022)

11. What is NumPy?: URL: https://numpy.org/devdocs/user/whatisnumpy.html (Дата звернення 14.04.2022)

12. What is TensorFlow? The machine learning library explained: URL: https://www.infoworld.com/article/3278008/what-is-tensorflow-the-machine-learning library-explained.html (Дата звернення 19.04.2022)

13. Simulataneously detecting face, hand motion, and pose in real-time on mobile devices: URL: https://heartbeat.fritz.ai/simultaneously-detecting-face-hand motion-and-pose-inreal-time-on-mobile-devices-27849560fc4e (Дата звернення: 24.04.2022)

14. ArcSoftGesture Recognition Technology: URL: http://www.arcsoft.com/ technology/ gesture.html (Дата звернення: 22.04.2022)

15. Сіряк Р.В. Модель обробки потокових даних для розпізнавання окремих одиниць жестової мови / Р.В. Сіряк, І.С. Скарга-Бандурова // Вісник Національного технічного університету «ХПІ». Серія: Інформатика та моделювання. – 2018. – № 42. – С. 73-81

16. CS231n: Convolutional Neural Networks for Visual Recognition [Electronic resource] – Access mode: https://cs231n.github.io/convolutional-networks/ – Title from the screen

17. S. Liwicki, M. Everingham. Automatic recognition of fingerspelled words in British Sign Language [Electronic resource]. – 2009. – Access mode:

https://ieeexplore.ieee.org/document/5204291/ (lastaccess: 31.05.2021). – Title from the screen.

18. Smelyakov K., Datsenko A., Skrypka V., Akhundov A. Efficiency of Image Reduction Algorithms with Small-Sized and Linear Details // 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), 8-11 Oct. 2019, Kyiv, Ukraine. – P. 745-750.

19. Freeman W.T., Roth M. Orientation Histograms for Hand Gesture Recognition // In International Workshop on Automatic Face and Gesture Recognition, 1994. — 296 – 301 с.

20. Pansare J.R., Gawande S.H., Ingle M. Real-Time Static Hand Gesture Recognition for American Sign Language (ASL) in Complex Background // Journal of Signal and Information Processing, Volume 3, Number 3, 2012. — 364-367 с.

21. Adam Zewe. A simpler path to better computer vision [Text] // MIT News Office. – 2022. – P. 1 – 3.

22. James Martin. Model Architecture, Output, and Evaluation [Text] // Machine Learning Applications in Nonproliferation: Assessing Algorithmic Tools for Strengthening Strategic Trade Controls. – 2020. – P. 17 – 33.

23. David Quinto-Pozos. Sign Language Contact and Interference: ASL and LSM [Text] // Language in Society. – 2008. – P. 161 – 189.

24. J. Rocco Blais, Adam M. Jungdahl. Artificial Intelligence in a Human Intelligence World [Text] // American Intelligence Journal. – 2019. – P. 108 – 113.

25. Garrison W. Cottrell. New Life for Neural Networks [Text] // American Association for the Advancement of Science Journal. – 2006. – P. 454 – 455.

26. B. D. Ripley. Neural Networks and Related Methods for Classification [Text] // Journal of the Royal Statistical Society. – 1994. – P. 409 – 456.

27. Хайкін С. Нейронні мережі: повний курс / С. Хайкін // М .: Діалектика, 2019. - 1104 с.

28. Каллан, Р. Нейронні мережі: Короткий довідник / Р. Каллан. - М .: Вільямс І.Д., 2017. - 288 с.

29. Israel Tabarez Paz, Neil Hernández Gress, Miguel González Mendoza. Pattern Recognition with Spiking Neural Networks [Text] // Lecture Notes in Computer Science. 2013. – P. 279 – 288.

30. Devashsih Sethia, Pallavi Singh. Gesture Recognition for American Sign Language Using Pytorch and Convolutional Neural Network [Text] // Lecture Notes in Electrical Engineering. 2022. – P. 307 – 317.

31. L'Heureux A., Grolinger K., Hany F. Elyamany, Miriam A.M. Capretz. Machine Learning With Big Data: Challenges and Approaches // IEEE Access. – 2016. – Vol. 5. – P. 7776 – 7797.

32. Warren E. Agin. A Simple Guide to Machine Learning [Text] // Business Law Today. 2017. – P. 1 – 5.

33. Juniy Chai, Hao Zeng, Anming Li, Eric W.T. Ngai. Deep learning in computer vision: A critical review of emerging techniques and application scenarios [Text] // Machine Learning with Applications. 2021. – P. 3 – 7.

34. Ivan Gridin. Automated Deep Learning Using Neural Network Intelligence: Develop and Design Pytorch and TensorFlow Models Using Python [Text]. 2019. – P. 277 – 333.

35. Garrido, José M. Introduction to computational models with Python [Text] // Boca Raton: CRC Press, Taylor Francis Group. 2016. – P. 300 – 320.

Відомості кваліфікаційної роботи

Система відео розпізнавання жестів мови глухонімих з використанням нейронних мереж у реальному часі

	Прізвище та ініціали відповідальної особи	Підпис	Дата
Роботу виконав студент групи СКСм-22-2. Структура кваліфікаційної роботи: – пояснювальна записка 74с.; – графічний матеріал 24 арк.	Тихомиров В.І.		01.12.2023
Керівник роботи	Хаханова Г.В		02.12.2023
Перевірка на плагіат здійснена. Оригінальність авторського тексту складає 85%	Литвинова Є.І.		06.12.2023
Нормоконтроль проведено :	Хаханова Г.В		05.12.2023